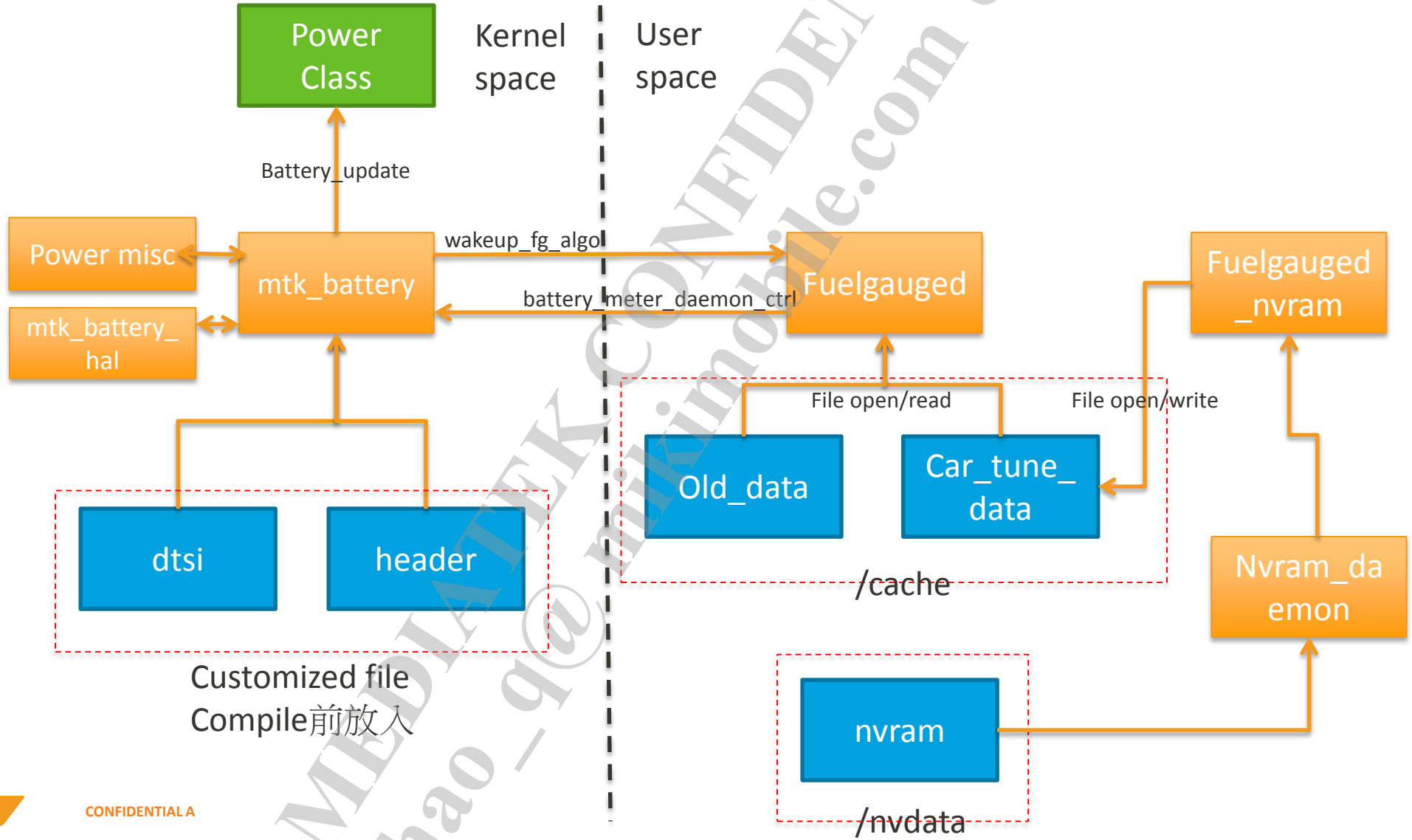
CONFIDENTIAL A

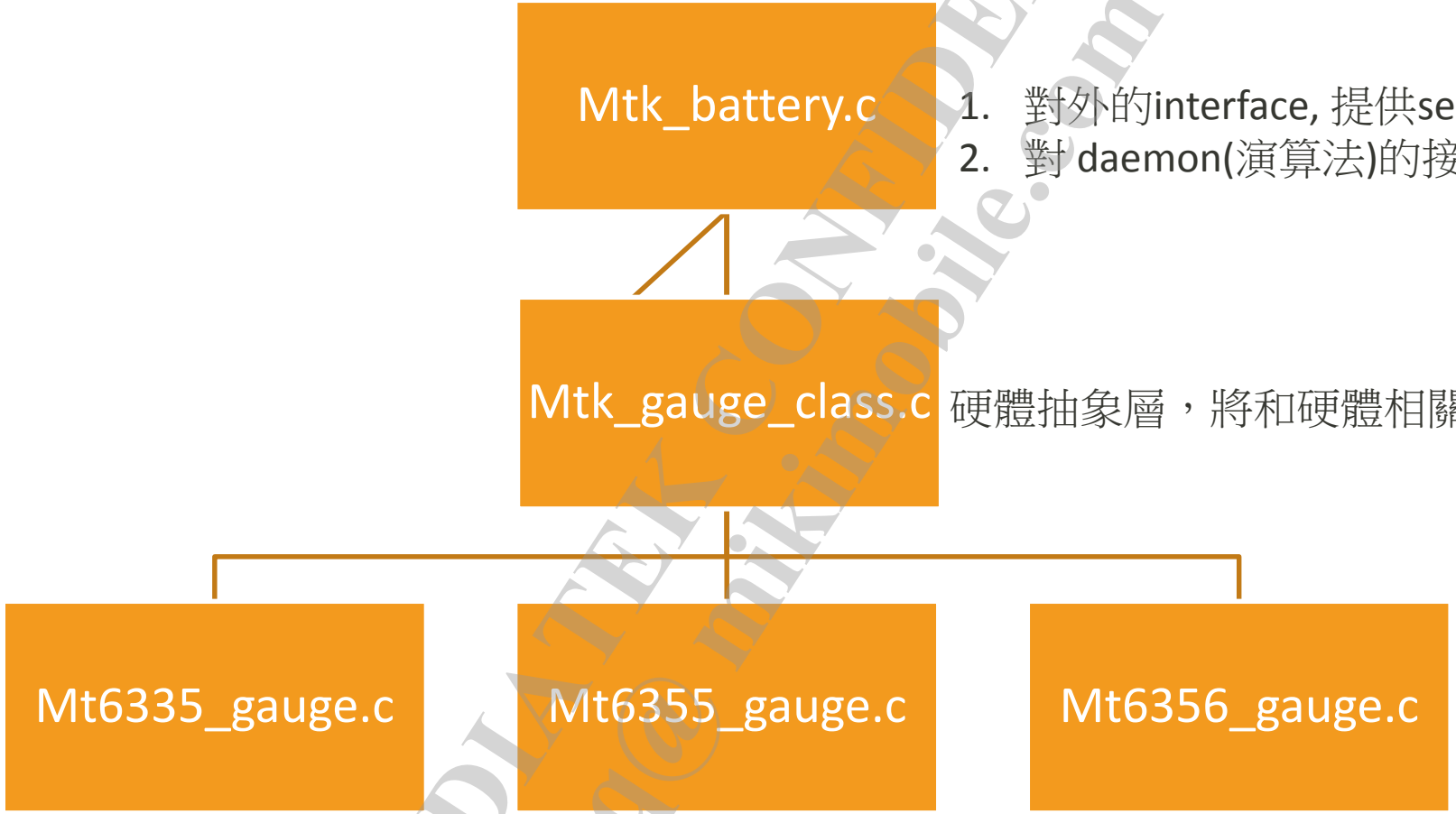
GM3.0 SW架構 Init flow 關機條件

Timo Liao

GM30 軟體架構



GM3軟體架構(kernel)

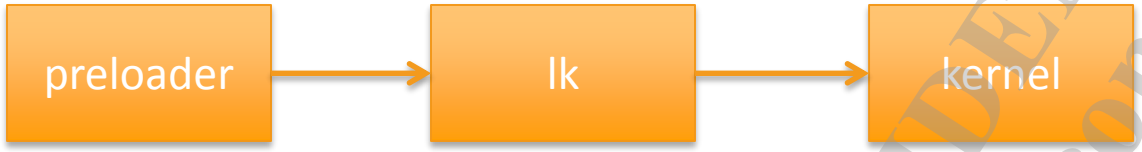


1. 對外的interface, 提供service及API
2. 對 daemon(演算法)的接口

硬體抽象層，將和硬體相關的行為於此層抽象化

針對不同PMIC進行不同operation

Initial flow



PL
(1) Bat exist檢查
(2) fgauge initial
(3) Read boot voltage, shutdown time and pass to kernel

LK
(1) 停充,read ptim V,I and pass to kernel
(2) 若有charger 充電到3.45v

kernel
(1) Battery driver probe
(2) 吃Header / DTSI 參數
(3) Listen interrupt
(4) Wake up daemon
(5) 讀寫硬體

daemon

daemon
(1) 計算c_soc/v_soc/ui_soc
(2) 設定interrupt

preloader

vendor/mediatek/proprietary/bootable/bootloader/preloader/platform/mt67xx/src/drivers/pmic.c
在pmic_init()時帶起 pl_battery_init()

```
pmic_init
1255        ret_val = pmic_config_interface(0x8412, 0x0008, 0xFFFF, 0); //[3]=0, WDTRSTB_STATUS_CLR
1256
1257
1258        /* ask peter peng programming guide TBD */
1259        ret_val = pmic_config_interface(PMIC_RG_SMPS_TESTMODE_B_ADDR, 0x0001,
1260                PMIC_RG_SMPS_TESTMODE_B_MASK, PMIC_RG_SMPS_TESTMODE_B_SHIFT); //RG_SMPS_TESTMODE
1261
1262        pl_battery_init(false);
1263
1264        if (hw_check_battery() == 1) {
1265 #if !CFG_EVB_PLATFORM
1266                pl_check_bat_protect_status();
1267 #endif
1268        }
```

preloader

vendor/mediatek/proprietary/bootable/bootloader/preloader/platform/mt67xx/src/drivers/battery.c

```
void pl_battery_init(bool force_init)
{
#if !CFG_EVB_PLATFORM
        bool is_battery_exist;

        is_battery_exist = hw_check_battery();
        print("[pl_battery_init] is_fg_init:%d , force_init:%d bat:%d\n",
                is_fg_init, force_init, is_battery_exist);

        if (is_fg_init == true) {
                print("[pl_battery_init] is_fg_init: %d , skip init\n", is_fg_init);
                return;
        }

        if (force_init == false && is_battery_exist == false) {
                print("[pl_battery_init] is_fg_init:%d , force_init:%d bat:%d , skip init\n",
                        is_fg_init, force_init, is_battery_exist);
                return;
        }

#if CFG_BATTERY_DETECT
        if (force_init == true) {
                while (hw_check_battery() == 0) {
                        mdelay(300);
                        platform_wdt_all_kick();
                }
        }
#endif

        fuel_gauge_init();

#endif
}
```

非EVB board會做 hw_battery check

Check battery exist

vendor/mediatek/proprietary/bootable/bootloader/preloader/platform/mt67xx/src/drivers/charging_bat.c

```
45 int hw_check_battery(void)
46 {
47         /* ask shin-shyu programming guide */
48
49     #ifdef MTK_DISABLE_POWER_ON_OFF_VOLTAGE_LIMITATION
50         print("ignore bat check !\n");
51         return 1;
52     #else
53         #if CFG_EVB_PLATFORM
54             print("ignore bat check\n");
55             return 1;
56         #else
57         U32 val=0;
58         U32 ret_val;
59
60         ret_val=pmic_config_interface( (U32)(MT6335_LDO_VBIF28_CON0),
61                                                 (U32)(1),
62                                                 (U32)(PMIC_RG_VBIF28_SW_EN_MASK),
63                                                 (U32)(PMIC_RG_VBIF28_SW_EN_SHIFT)
64                                                 );
65
66
67         pmic_upmu_set_rg_baton_en(1);
68         /* pmic_upmu_set_baton_tdet_en(1); */
69         //mdelay(100);
70         val = pmic_upmu_get_rgs_baton_undet();
71
72
73         if(val == 0)
74         {
75             print("bat is exist.\n");
76             return 1;
77         }
78         else
79         {
80             print("bat NOT exist.\n");
```

Preloader – gauge init

```
void fuel_gauge_init(void)
{
        int fg_reset_status;
        signed int efuse_cal;
        int fg_curr_time;
        int shutdown_pmic_time;
        int do_init_fgadc_reset;
        int ret;
        int hw_id, sw_id;
        U32 reset_sel;

        reset_sel = upmu_get_reg_value(PMIC_RG_FGADC_RST_SRC_SEL_ADDR);
        //only for GM 3.0 : set FGADC reset source selection = 1 ,GM3.0 reset
        ret = pmic_config_interface((U32)(PMIC_RG_FGADC_RST_SRC_SEL_ADDR), (U32)(1),
                (U32)(PMIC_RG_FGADC_RST_SRC_SEL_MASK),
                (U32)(PMIC_RG_FGADC_RST_SRC_SEL_SHIFT));

        /*fg_reset_status = pmic_get_register_value(PMIC_FG_RSTB_STATUS);*/
        ret = pmic_read_interface((U32)(PMIC_FG_RSTB_STATUS_ADDR), (&fg_reset_status),
                (U32)(PMIC_FG_RSTB_STATUS_MASK),
                (U32)(PMIC_FG_RSTB_STATUS_SHIFT));
```

GM3務必設定 PMIC_RG_FGADC_RST_SRC_SEL_ADDR =1
(FGADC reset source selection) ,GM2 千萬不可設定此register

Preloader conclusion

- 判斷電池是否存在
- 對Gauge hw進行init
- 判斷gauge是否被reset過(判斷是否曾拔過電池)
- 讀取開機電壓 boot_vbat
- 讀取關機時間 shutdowntime
- 判斷有無發生2sec reboot
- Preloader init順序有dependency,請勿更動init順序

LK –file list

- vendor/mediatek/proprietary/bootable/bootloader/lk/
  - platform/mt6799/mt_battery.c
  - platform/mt6799/platform.c
  - platform/mt6799/include/platform/boot_mode.h
  - app/mt_boot/mt_boot.c
- 主要卡低電時的開機條件, 如果有charger , 充電到3.45v才可開進kernel

LK -- init

```
#ifndef MTK_DISABLE_POWER_ON_OFF_VOLTAGE_LIMITATION
#ifndef MTK_BATLOWV_NO_PANEL_ON_EARLY
        if (bat_vol < BATTERY_LOWVOL_THRESOLD)
#else
        if (is_low_battery(bat_vol))
#endif
        {
                if (g_boot_mode == KERNEL_POWER_OFF_CHARGING_BOOT && upmu_is_chr_det() == KAL_TRUE) {
                        dprintf(CRITICAL, "[%s] Kernel Low Battery Power Off Charging Mode\n", __func__);
                        g_boot_mode = LOW_POWER_OFF_CHARGING_BOOT;

                        check_bat_protect_status();

                } else {
                        if (bat_vol < BAT_LV_NO_CHR) {
                                dprintf(CRITICAL, "[BATTERY] battery voltage(%dmV) <= CLV ! Can not Boot I
#ifndef NO_POWER_OFF
                                mt6575_power_off();
#endif
                                while (1) {
                                        dprintf(CRITICAL, "If you see the log, please check with RTC power
                                }
                        }
                }
        }
#endif
```

電壓低於3.45V , 就卡在LK , 直到電壓高於3.45V才開進kernel

LK – init con.

```
#if defined(DLPT_FEATURE_SUPPORT)

        if (g_boot_mode != META_BOOT && g_boot_mode != FACTORY_BOOT && g_boot_mode != ATE_FACTORY_BOOT) {
                /* pmic_set_register_value(PMIC_BATON_TDET_EN, 1); */
                pmic_set_register_value(PMIC_RG_BATON_EN, 1);
                if (pmic_get_register_value(PMIC_RGS_BATON_UNDET) == 1) {
                        dprintf(CRITICAL, "[BATTERY] No battry plug-in. Power Off.");
                        mt6575_power_off();
                }
        }
        pchr_turn_on_charging(KAL_FALSE);
        /* disable SW charger power path */
        switch_charger_power_path_enable(KAL_FALSE);
        mdelay(50);
        get_dlpt_imix_r();
        /* after get imix, re-enable SW charger power path */
        switch_charger_power_path_enable(KAL_TRUE);
        mdelay(50);
        check_bat_protect_status();
        if (is_charging == 1) {
                pchr_turn_on_charging(KAL_TRUE);
                dprintf(CRITICAL, "turn on charging \n\r");
        }
#endif //#if defined(DLPT_FEATURE_SUPPORT)
```

電池電壓已高於3.45V ,
就關 charger, 關 power path, 算出電池的R, 後續準備傳入 kernel, for DLPT使用

Kernel -- init

```
4773 static int __init battery_init(void)
4774 {
4775         struct netlink_kernel_cfg cfg = {
4776                 .input = nl_data_handler,
4777         };
4778
4779         int ret;
4780
4781         daemo_nl_sk = netlink_kernel_create(&init_net, NETLINK_FGD, &cfg);
4782         bm_err("netlink_kernel_create protol= %d\n", NETLINK_FGD);
4783
4784         if (daemo_nl_sk == NULL) {
4785                 bm_err("netlink_kernel_create error\n");
4786                 return -1;
4787         }
4788         bm_err("netlink_kernel_create ok\n");
4789
4790 #ifdef CONFIG_OF
4791         /* register battery_device by DTS */
4792 #else
4793         ret = platform_device_register(&battery_device);
4794 #endif
4795
4796         ret = platform_driver_register(&battery_driver_probe);
4797         ret = platform_driver_register(&battery_dts_driver_probe);
4798
4799         is_init_done = true;
4800         bm_err("[battery_init] Initialization : DONE\n");
4801
4802         return 0;
```

kernel-4.4/drivers/power/mediatek/battery/mtk_battery.c

1. 掛 netlink handler (nl_data_handler)
2. 建 netlink

沒有DTS的flow : (CONFIG_OF 關掉時)

1. 先用 platform_device_register() 註冊device node
2. 再用 platform_driver_register() 註冊driver

有DTS的flow :
platform_driver_register(&battery_dts_driver_probe);

Kernel – find dtsi table

```
#ifdef CONFIG_OF
        /* register battery_device by DTS */
#else
        ret = platform_device_register(&battery_device);
#endif

        ret = platform_driver_register(&battery_driver_probe);
        ret = platform_driver_register(&battery_dts_driver_probe);
```

```
static struct platform_driver battery_dts_driver_probe = {
        .probe = battery_dts_probe,
        .remove = NULL,
        .shutdown = NULL,
        .suspend = NULL,
        .resume = NULL,
        .driver = {
                .name = "battery-dts",
#ifdef CONFIG_OF
                .of_match_table = mtk_bat_of_match,
#endif
        },
};
```

```
#ifdef CONFIG_OF
static const struct of_device_id mtk_bat_of_match[] = {
        {.compatible = "mediatek,bat_gm30",},
        {},
};
MODULE_DEVICE_TABLE(of, mtk_bat_of_match);
#endif
```

找尋名字為 mediatek,bat_gm30 的 dtsi table

Kernel – dtsi table

- 跟DTSI相關的檔案如下:
- kernel-4.4/arch/arm64/boot/dts/mediatek/mt6799.dtsi
- kernel-4.4/arch/arm64/boot/dts/mediatek/bat_setting/
  - mt6799_battery_prop.dtsi
  - mt6799_battery_prop_ext.dtsi
  - mt6799_battery_table.dtsi
  - mt6799_battery_table_ext.dtsi

Kernel – dtsi table

```
bat_gm30: bat_gm30{
        compatible = "mediatek,bat_gm30";
        DIFFERENCE_FULLOCV_ITH = <(150)>;
        SHUTDOWN_1_TIME = <(60)>;        /*
        KEEP_100_PERCENT = <(3)>;        /*
        R_FG_VALUE = <(10)>;     /*      R_sense
        EMBEDDED_SEL = <(0)>;    /*      Configu
        PMIC_SHUTDOWN_CURRENT = <(20)>; /*
        FG_METER_RESISTANCE = <(50)>;    /*
        CAR_TUNE_VALUE = <(100)>;        /*
        TEMPERATURE_T0 = <(50)>;         /*
        TEMPERATURE_T1 = <(25)>;         /*
        TEMPERATURE_T2 = <(10)>;         /*
        TEMPERATURE_T3 = <(0)>; /*       Battery
        TEMPERATURE_T4 = <(-10)>;        /*
        g_FG_PSEUDO100_T0 = <(98)>;      /*
        g_FG_PSEUDO100_T1 = <(98)>;      /*
        g_FG_PSEUDO100_T2 = <(95)>;      /*
        g_FG_PSEUDO100_T3 = <(90)>;      /*
        g_FG_PSEUDO100_T4 = <(80)>;      /*
        Q_MAX_SYS_VOLTAGE_BAT0 = <(3200)>;
        Q_MAX_SYS_VOLTAGE_BAT1 = <(3200)>;
        Q_MAX_SYS_VOLTAGE_BAT2 = <(3200)>;
        Q_MAX_SYS_VOLTAGE_BAT3 = <(3200)>;
#if (CONFIG_MTK_ADDITIONAL_BATTERY_TABLE == 1)
#include "mt6799_battery_table_ext.dtsi"
#else
#include "mt6799_battery_table.dtsi"
#endif
}
```

kernel-4.4/arch/arm64/boot/dts/mediatek/bat_setting/mt67xx_battery_prop_ext.dtsi
此dtsi檔案 ,客戶可由 GMAT tool 產生出來

Kernel – dtsi init

```
static struct platform_driver battery_dts_driver_probe = {
        .probe = battery_dts_probe,
        .remove = NULL,
        .shutdown = NULL,
        .suspend = NULL,
        .resume = NULL,
        .driver = {
                .name = "battery-dts",
#ifdef CONFIG_OF
                .of_match_table = mtk_bat_of_match,
#endif
        },
};
```

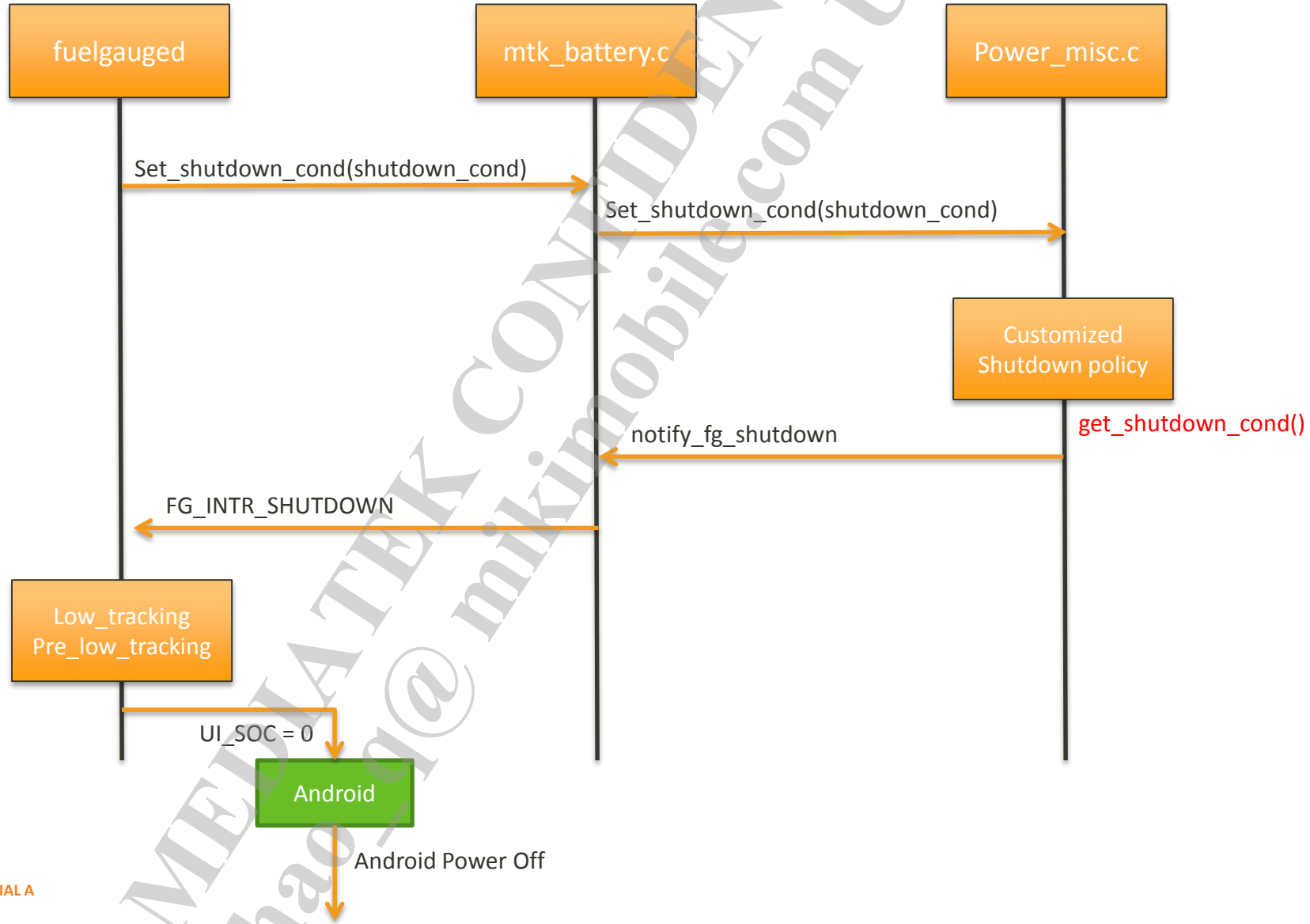
```
#ifdef CONFIG_OF
static int battery_dts_probe(struct platform_device *dev)
{
        int ret = 0;

        bm_err("******** battery_dts_probe!! ********\n");
        battery_device.dev.of_node = dev->dev.of_node;
        ret = platform_device_register(&battery_device);
        if (ret) {
                bm_err("****[battery_dts_probe] Unable to register device (%d)\n", ret);
                return ret;
        }

        fg_custom_init_from_dts(dev);

        return 0;
}
#endif
```

Shutdown flow

- 將是否要shutdown的動作, 獨立到power_misc.c當中作
- 客戶可以修改power_misc.c中的關機條件, gauge只提供事件提醒, 以及若要關機時, 設定ui_soc low tracking的callback
- Fuelgauged通知”soc 0%” 與 “ui_soc 已持續60 min”兩個事件
- 以這些event通知fuelgauge進行ui_soc處理

FG_INTR_SHUTDOWN = 16384,	Power misc通知關機	Daemon啟動low_tracking機制將ui_soc降到0
FG_INTR_VBAT2_L = 0x40000,	3.4v shutdown	Daemon啟動pre_low_tracking機制將ui_soc降到1
FG_INTR_VBAT2_H = 0x80000,		若重接charger, 大於3.5v之後, 就會取消 pre_low_tracking
FG_INTR_DLPT_SD = 0x200000,	DLPT shutdown	daemon將ui_soc直接設為0

Shutdown
SOC_ZERO_PERCENT, UISOC_ONE_PERCENT, LOW_BAT_VOLT



Shutdown_cond

1. 目前預設關機條件有
    1. Soc低於0%
    2. Ui_soc顯示1%且持續了60 min以上
    3. VBAT < 3.4v (支持buck boost的平台不需這個條件)
    4. DLPT < 3.1v

2. Shutdown流程一般是
    a) Fg daemon發現滿足某些shutdown_condition
    b) 透過Set_shutdown_cond()通知kernel這個event
    c) power_misc.c收到event後, 客戶可依需求定義policy 決定要如何處理關機event
    a) 若決定關機, 則通知daemon作low tracking
    b) Low tracking到ui_soc = 0之後, android就會進行關機流程

Shutdown_cond可調整參數的處理方式

客戶可自行修改
Kernel_4.4/drivers/power/mediatek/power_misc.c

```
int get_shutdown_cond(void)
{
    int ret = 0;

    if (sdc.shutdown_status.is_soc_zero_percent)
        ret |= 1;
    if (sdc.shutdown_status.is_uisoc_one_percent)
        ret |= 1;
    if (sdc.lowbatteryshutdown)
        ret |= 1;
    return ret;
}
```

Fg_daemon會在每次interrupt發生時,
調用此function來收集是否要關機

故客戶可以改寫這邊的邏輯
目前我們是讓三種關機條件任一成立, 就return 1
return 1之後, fg_daemon就會開始降ui_soc

#請客戶不要在get_shutdown_cond()
裡面用迴圈等待或sleep, 這會卡住daemon thread
這個function會一直被call到, 不需要在裡頭wait

GM3演算法參數初始化

在kernel中當battery driver被probe時，就會讀取dtsi中的值,放到fg_cust_data
battery_dts_probe ->fg_custom_init_from_dts

```
#ifdef CONFIG_OF
static int battery_dts_probe(struct platform_device *dev)
{
        int ret = 0;

        bm_err("******** battery_dts_probe!! ********\n");
        battery_device.dev.of_node = dev->dev.of_node;
        ret = platform_device_register(&battery_device);
        if (ret) {
                bm_err("****[battery_dts_probe] Unable to register device (%d)\n", ret);
                return ret;
        }

        fg_custom_init_from_dts(dev);

        return 0;
}
#endif
```

everyday genius