CONFIDENTIAL B

Charger study

Alex Ju

Agenda

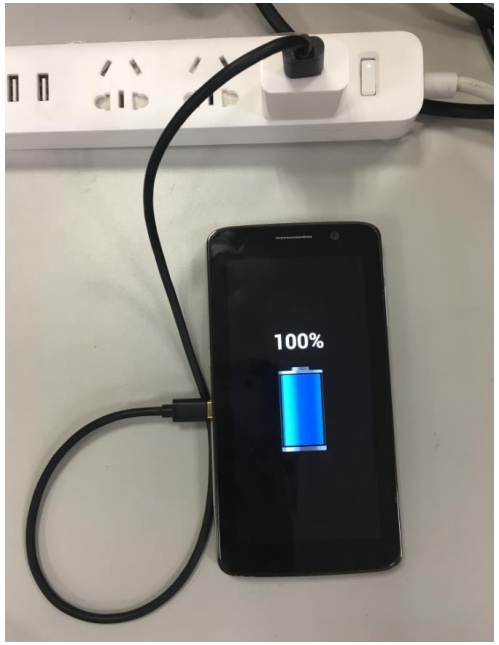
- 充电基本概念
- MTK Charger框架
- MTK Charger log 解读
- MTK Charger常见问题
- MTK Charger code
  - Pl、Lk、kernel

Agenda

- 充电基本概念
- MTK Charger框架
- MTK Charger log 解读
- MTK Charger常见问题
- MTK Charger code
  - Pl、Lk、kernel

基本概念

- 使用手机充电器，对手机内的锂电池进行充电



民用电 220v
充电器输出 5v ~ 12v
电池电压范围3.4v ~ 4.4v

普通充电器输出为5v
快充充电器输出电压高于5v

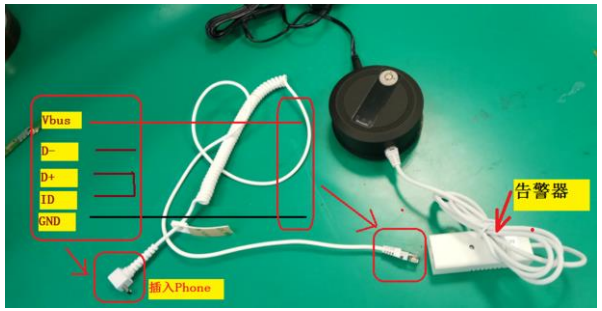
注意：
手机充电接口**接触不良**，是无法给手机充电的！！！

充电器类型

标准充电器（DCP）
5V 1A
5V 2A

PC USB（SDP）
5V 0.5A

PC CDP
5V 1.5A



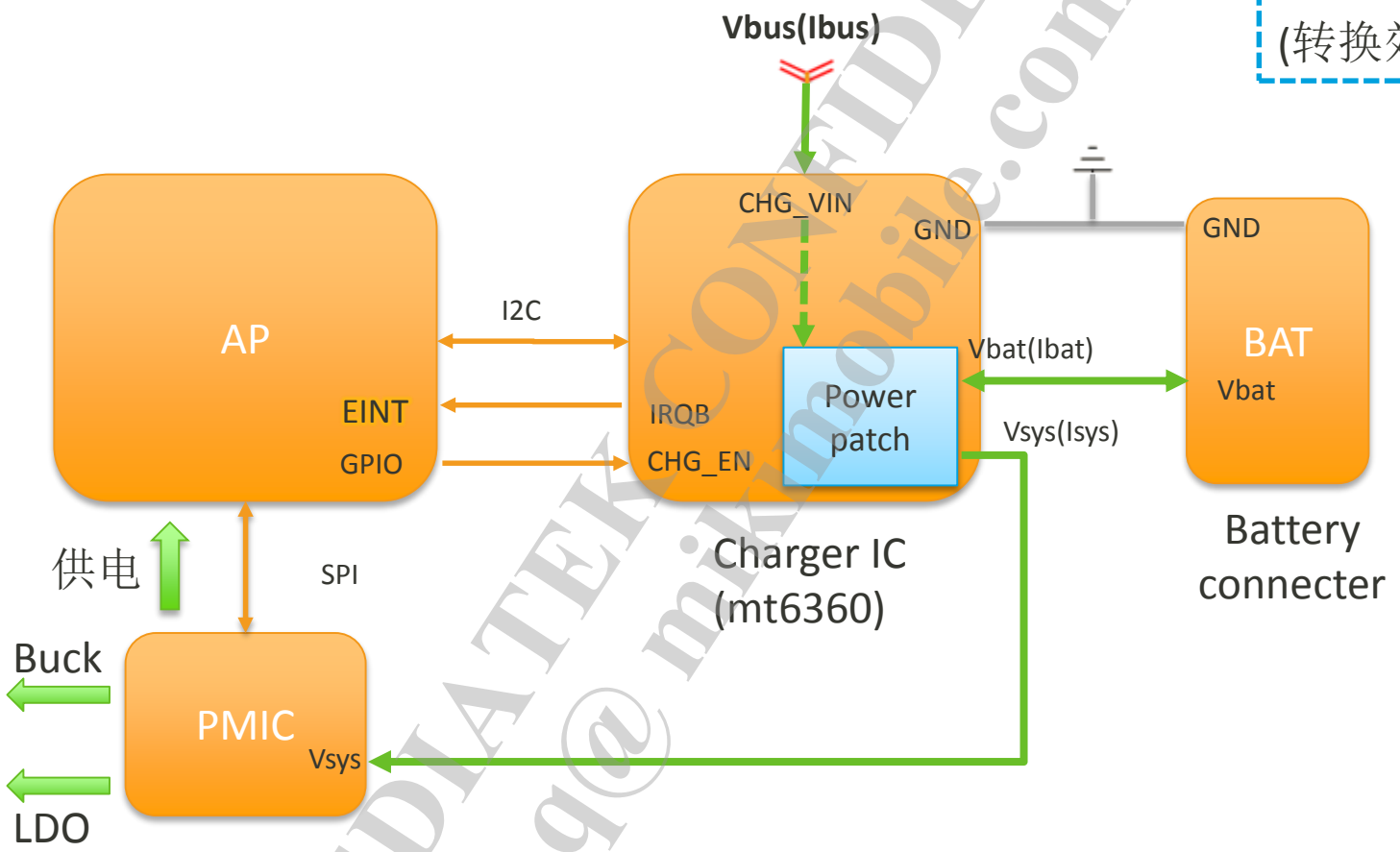
非标准充电器
（手机店报警器）
5V 0.5A

DCP： Dedicated Charging Port
SDP： Standard Downstream Port
CDP： Charging Downstream Port

SDP和CDP都是电脑上的端口
CDP接口由电池或闪电图标标记

充电示意图

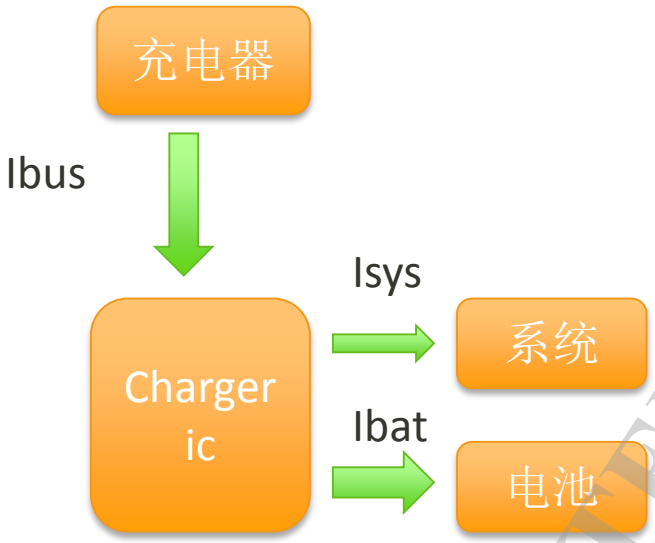
Ibus * Vbus * 转换效率 =
Ibat * Vbat + Isys *Vsys
(转换效率一般在0.9左右)



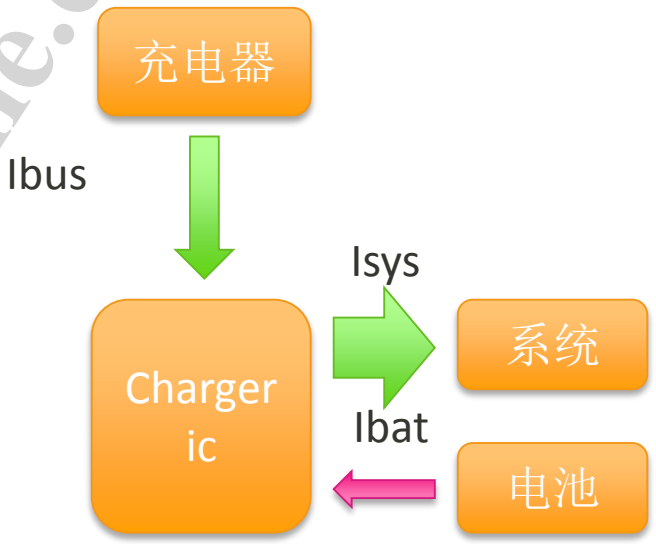
CHG_EN：Charger enable input, active-low.
IRQB：Interrupt output, active-low open-drain
Vsys：power supply for system

充电电流示意图

充电电流 > 系统耗流



充电电流 < 系统耗流



Power path：

- Charger ic从充电器抽电流，电流先提供给系统使用（Vsys），剩下的电流流进电池
- 如果充电器抽的电流不够系统使用（或者无充电器存在），从电池抽电供系统使用

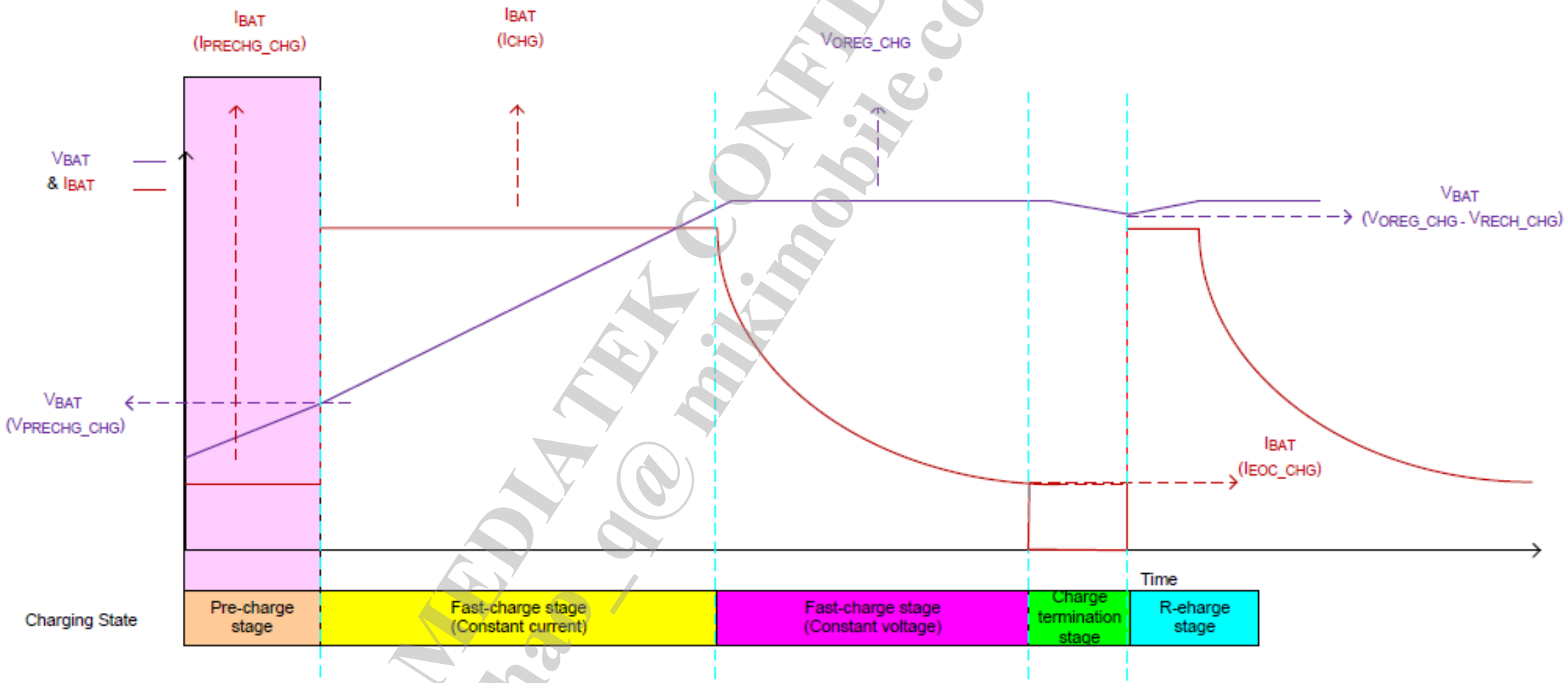
充电阶段简要说明

以mt6360为例

Trickle Charge	Pre_charge	Constant current Fast charge	Constant voltage Fast charge	Charge done	Recharger
• VBAT < 2V • 100mA充电 充电电压和电流不可更改 can not be programmable	• **VPREC > VBAT > 2V** • 150ma充电 VPREC = 3V 电压、电流均可修改 Reg 0x18	• VBAT > VPREC • VPREC = 3V 0.5~3A充电 电压、电流均可修改	• VBAT >= VOREG VOREG 默认4.2v 充电电流会逐渐下降（sw不可控） 电压门限可更改	1. VBAT >= VOREG 2. Ibat < IECO deglitch time 2ms IEOC默认250ma	• Vbat < VOREG – VREC VREC 默认100mv

充电曲线

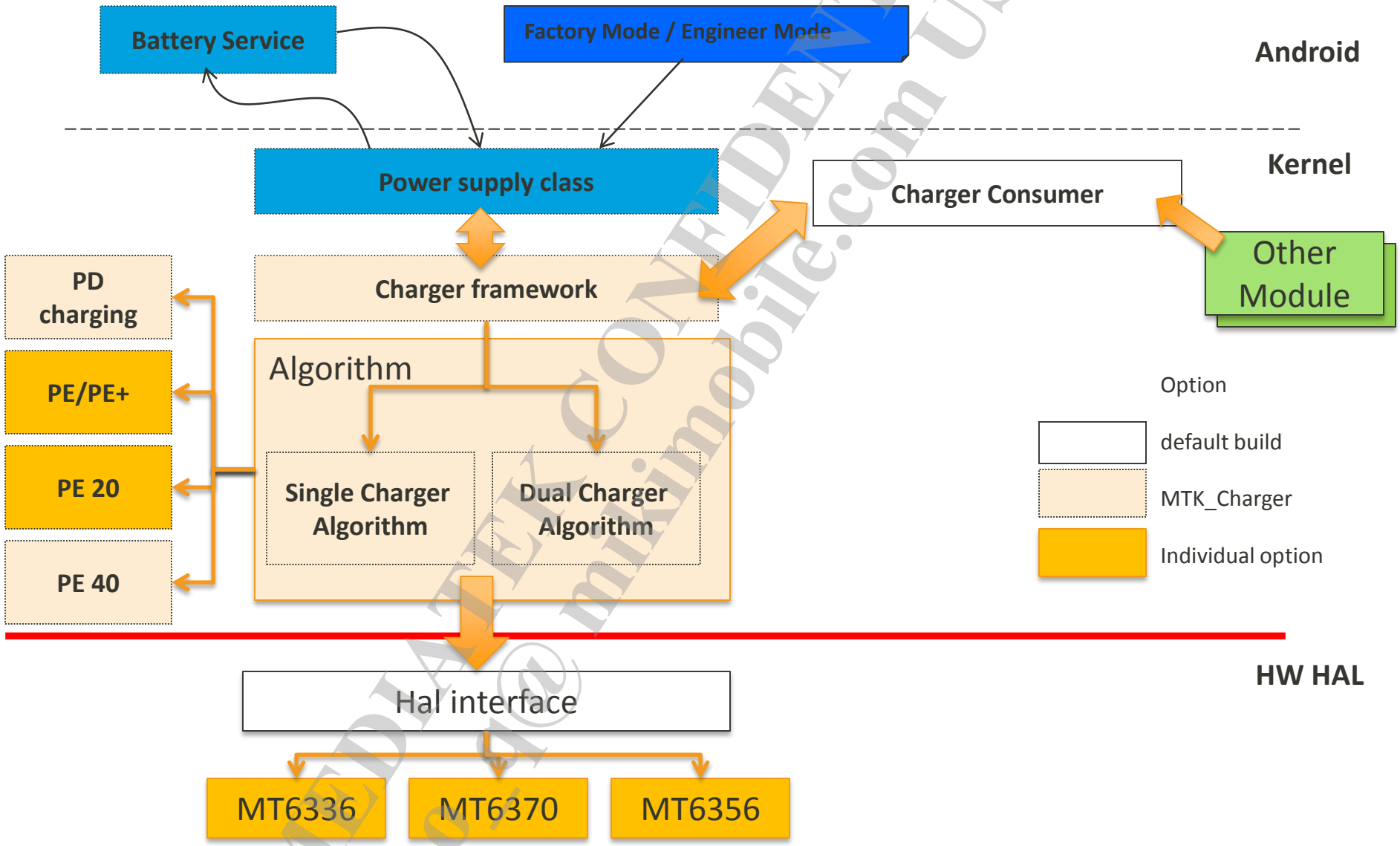
摘自6360 datasheet 图3.3



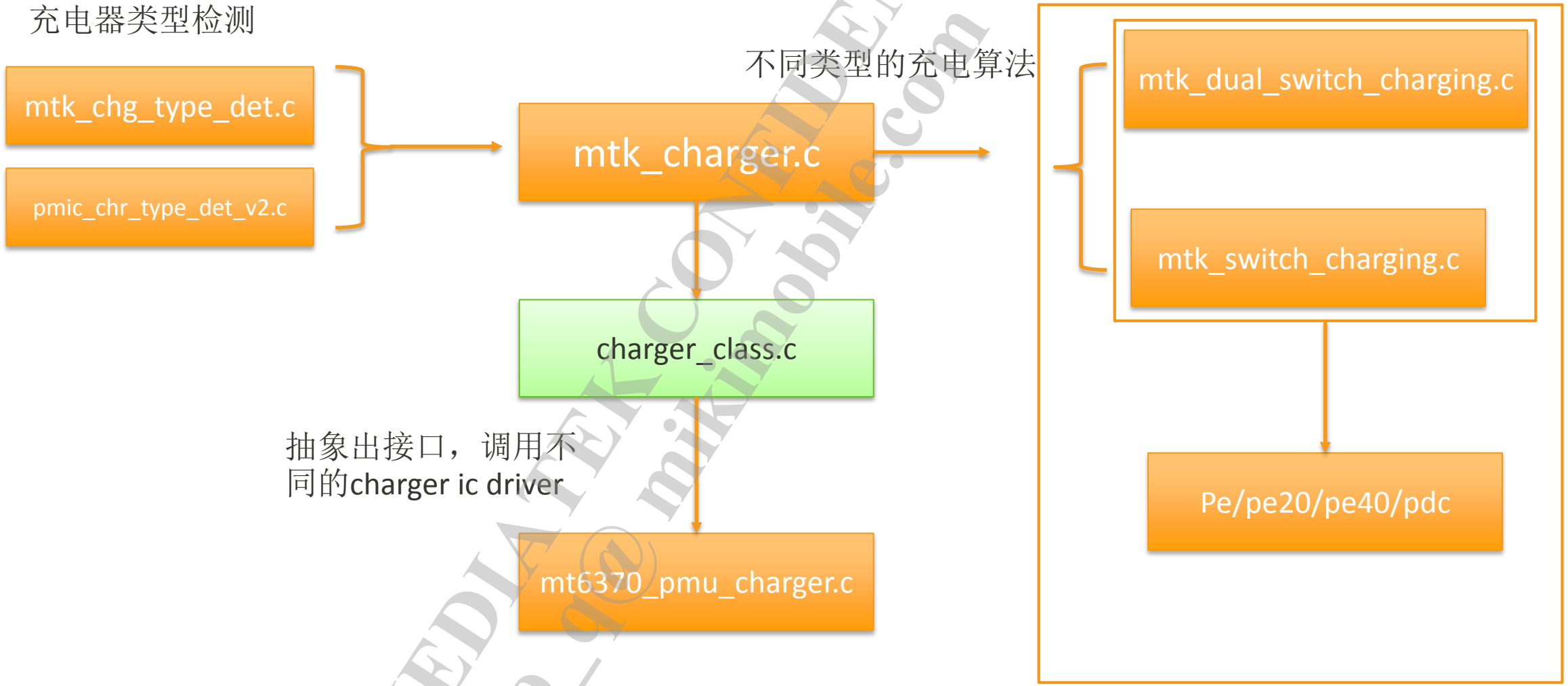
Agenda

- 充电基本概念
- MTK Charger框架
- MTK Charger log 解读
- MTK Charger常见问题
- MTK Charger code
  - Pl、Lk、kernel

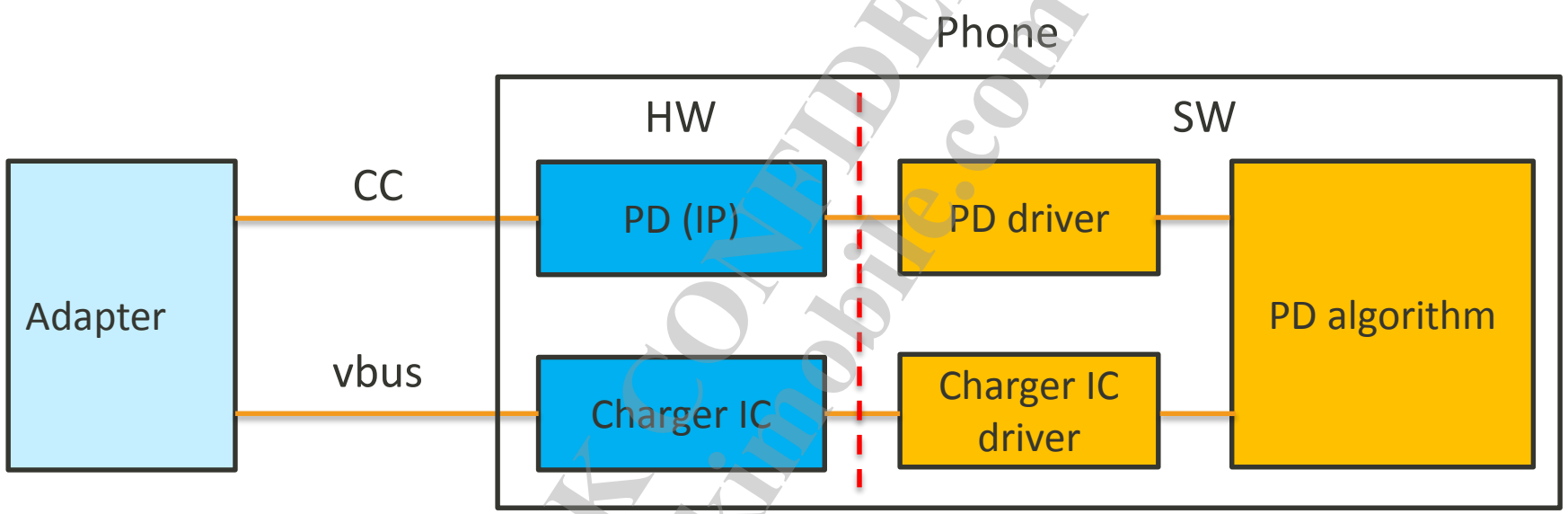
MTK Charger Architecture



Charger in kernel



PD Architecture

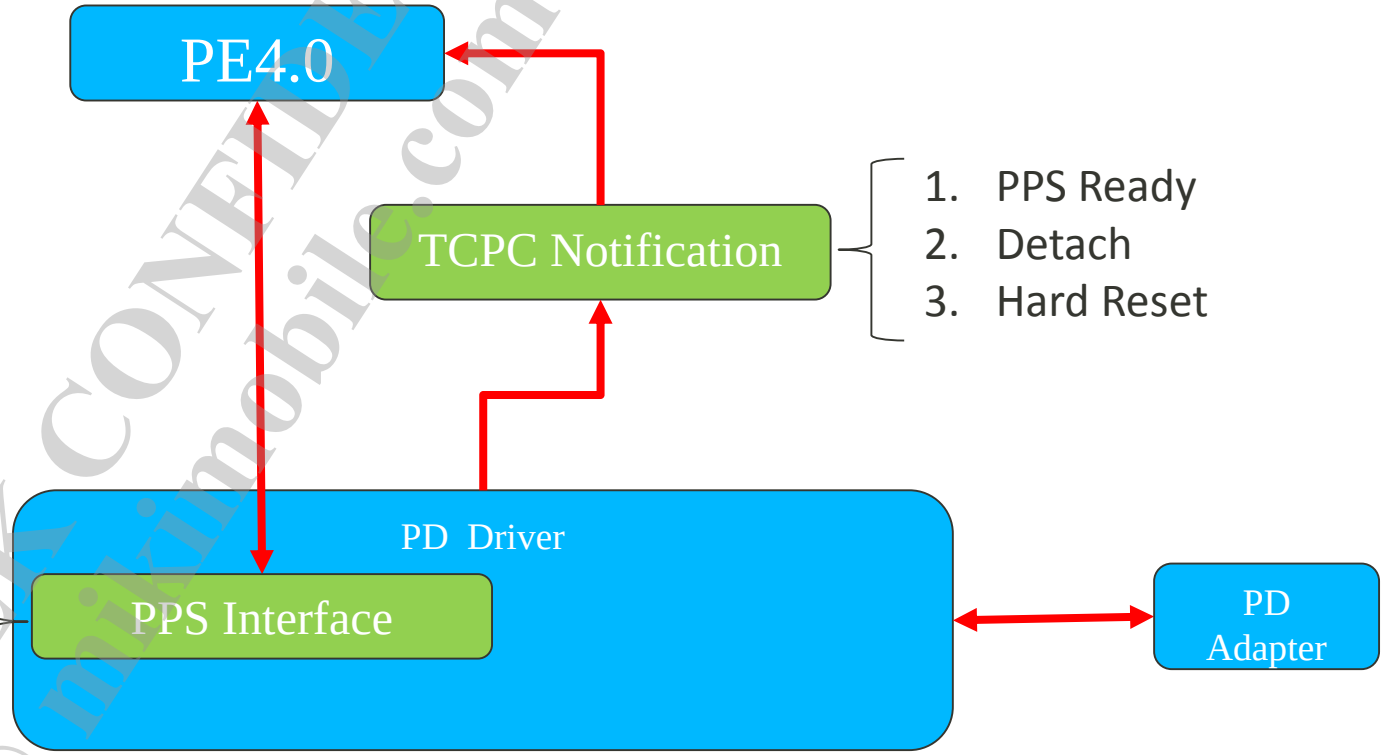


- Adapter CC pin跟PD (IP)連接, PD algorithm透過PD driver去query Adapter能力和決定電壓電流

PD Driver

- PD driver
  - mtk_pdc_intf.c
  - tcpm.c
  - pd_dpm_pdo_select.c
- Charger IC driver
  - mt6370_pmu_charger.c
- PMIC driver
- PD charging algorithm
- Integrate API to set current limit
  - mtk_pdc_check_charger() 確認是否支援PD
  - mtk_pdc_get_setting() 查詢TA可提供電壓電流檔位
  - mtk_pdc_setup() 設定TA電壓電流檔位

PE40 PPS part Dummy Flow Chart




```
extern int tcpm_inquire_pd_source_apdo(struct tcpc_device *tcpc, uint8_t apdo_type, uint8_t *cap_i, struct
tcpm_power_cap_val *cap);

extern int tcpm_set_apdo_charging_policy(struct tcpc_device *tcpc, uint8_t policy, int mv, int ma, const
struct tcp_dpm_event_cb_data *data);

extern int tcpm_reset_pd_charging_policy(struct tcpc_device *tcpc, const struct tcp_dpm_event_cb_data
*data);

extern int tcpm_dpm_pd_get_pps_status(struct tcpc_device *tcpc, const struct tcp_dpm_event_cb_data
*data, struct pd_pps_status *pps_status);

extern int tcpm_dpm_pd_get_status(struct tcpc_device *tcpc, const struct tcp_dpm_event_cb_data *data,
struct pd_status *status);

extern int tcpm_dpm_pd_request(struct tcpc_device *tcpc, int mv, int ma, const struct
tcp_dpm_event_cb_data *data);

extern int tcpm_set_pd_charging_policy(struct tcpc_device *tcpc, uint8_t policy, const struct
tcp_dpm_event_cb_data *data);

extern int tcpm_dpm_pd_alert(struct tcpc_device *tcpc, uint32_t ado,
                             const struct tcp_dpm_event_cb_data *data);
```

2、 Best Performance Idea

Vbus * Ibus * X = Phone power consumption = Vsys * Isys + battery voltage * ibat

- Vbus: adapter output voltage
- Ibus: adapter output current
- X: charger IC 轉換效率,
- ICL: Input current limit, charger IC 限制Ibus 上限
- Vsys: system voltage
- Isys: system current
- ibat: battery current

Current Limit Setting

	AICR	ICHG
PE4	Single: 3A Dual: 2A/2A	Single: 3A Dual: 2A/2A
TypeC	Rp = 10K: 3A Rp = 22K: 1.5A Rp = 56K: 500mA	Rp = 10K: 3A Rp = 22K: 2A Rp = 56K: 500mA
PD	3A	3A
Standard host (SDP)	500mA	500mA
Nonstandard charger	500mA	500mA
Standard charger (DCP)	3A	2A
PE/PE2	Single: 3A Dual: 2A/2A	Single: 3A Dual: 2A/2A
Charging host (CDP)	1.5A	1.5A

Agenda

- 充电基本概念
- MTK Charger框架
- MTK Charger log 解读
- MTK Charger常见问题
- MTK Charger code
  - Pl、Lk、kernel

Pl jump to lk protect

电池电压　　Pl jump to lk threshold

- [pl_check_bat_protect_status]: check VBAT=3885mV with 0mV, VCHR=4730mV ,VCHR_HV=6500mv, start charging
- [pl_check_bat_protect_status]: check VBAT=3885mV with 0mV, stop charging

```
#if SWCHR_POWER_PATH
#define BATTERY_LOWVOL_THRESOLD 0
#else
#define BATTERY_LOWVOL_THRESOLD 3300
#endif
```

/vendor/mediatek/proprietary/bootable/bootloader/preloader/platform/mt6775/src/drivers/charging_bat.c

lk jump to kernel protect

[1334] [check_bat_protect_status]: check VBAT=3823 mV with 3450 mV

如果要修改lk跳kernel电压门限，修改BATTERY_LOWVOL_THRESOLD即可

如果电池电压小于3.45v则进行充电
当电池大于3.45v，才会开进kernel

```
#define BATTERY_LOWVOL_THRESOLD          3450
```

/vendor/mediatek/proprietary/bootable/bootloader/lk/platform/common/power/mtk_battery.h

Charger in Kernel

```
typedef enum {
    CHARGER_UNKNOWN = 0,
    STANDARD_HOST,
    CHARGING_HOST,
    NONSTANDARD_CHARGER,
    STANDARD_CHARGER,
    APPLE_2_1A_CHARGER,
    APPLE_1_0A_CHARGER,
    APPLE_0_5A_CHARGER,
} CHARGER_TYPE;
```

插入充电器log：[285:charger_thread]: mtk_is_charger_on plug in, tyupe:3
拔出充电器log：[285:charger_thread]: mtk_charger_plug_out

插入充电器后10s打印一次下面的log
[285:charger_thread]Vbat=3817,Ibat=8895,I=0,VChr=4721,T=33,Soc=10: 9,CT:3:3 hv:1 pd:0:0

- Vbat：电池电压，单位mv
- Ibat：电池电流，正值为充电，负值为放电，单位0.1ma
- I=0：未用到，一直是0
- VChr：充电器（vbus）电压，单位mv
- T：电池NTC采集温度，摄氏度
- Soc：第一个为底层soc，第二个为uisoc
- CT：charger type简写，表示充电器类型

Pmic 充电器类型检测g_chr_type = hw_charging_get_charger_type();
/kernel-4.9/drivers/misc/mediatek/pmic/mt6357/v1/pmic_chr_type_det_v2.c

charger_type 数字解释

插入充电器log：[285:charger_thread]: mtk_is_charger_on plug in, tyupe:3
拔出充电器log：[285:charger_thread]: mtk_charger_plug_out

Pmic 充电器类型检测g_chr_type = hw_charging_get_charger_type();
/kernel-4.9/drivers/misc/mediatek/pmic/mt6357/v1/pmic_chr_type_det_v2.c

```
enum charger_type {
        CHARGER_UNKNOWN = 0,
        STANDARD_HOST,              /* USB : 450mA */
        CHARGING_HOST,
        NONSTANDARD_CHARGER,        /* AC : 450mA~1A */
        STANDARD_CHARGER,           /* AC : ~1A */
        APPLE_2_1A_CHARGER, /* 2.1A apple charger */
        APPLE_1_0A_CHARGER, /* 1A apple charger */
        APPLE_0_5A_CHARGER, /* 0.5A apple charger */
        WIRELESS_CHARGER,
};
```

Mt6370 dump log

mt6370_dump_register: ICHG = 1000mA, AICR = 1000mA, MIVR = 4400mV, IEOC = 200mA, CV = 4400mV
mt6370_dump_register: VSYS = 3840mV, VBAT = 3810mV, IBAT = 750mA, IBUS = 900mA, VBUS = 4725mV
mt6370_dump_register: CHG_EN = 1, CHG_STATUS = progress, CHG_STAT = 0xA0
mt6370_dump_register: CHG_CTRL1 = 0x10, CHG_CTRL2 = 0x1B

ICHG：充电电流设置，后端电流，charger ic 到电池的电流
AICR：充电电流设置，前端电流，充电器到charger ic的电流
MIVR：触发AICL机制的门限电压
IEOC：设置的截至充电电流
CV：设置的CV点
IBAT：charger ic检测的流进电池理的电流
IBUS：charger ic检测的usb线上的电流
若有不理解之处，请参考《MT6370_Design_notice_Vxx》《date sheet》

“force:” log解读

- 未限流
- force:0 thermal:-1,-1 pe4:-1,-1,0 setting:1000 1000 type:1 USB充电

  -1表示不限流　AICR设置1A　ICHG设置1A
- Ibat限流500ma
- force:0 thermal:-1,500 pe4:-1,-1,0 setting:1000 500 type:1 USB充电

  AICR不限流　ICHG限500ma　AICR设置1A　ICHG设置500mA

Agenda

- 充电基本概念
- MTK Charger框架
- MTK Charger log 解读
- MTK Charger常见问题
- MTK Charger code
  - Pl、Lk、kernel

充电图标API

- 显示充电图标

```
charger_manager_notifier(info,CHARGER_NOTIFY_START_CHARGING);
```

- 取消充电图标

```
charger_manager_notifier(info,CHARGER_NOTIFY_STOP_CHARGING);
```

修改充电电流

- mt67xx.dts

```
lk_charger: lk_charger {
        compatible = "mediatek,lk_charger";
        enable_pe_plus;
        power_path_support;
        max_charger_voltage = <6500000>;
        fast_charge_voltage = <3000000>;

        /* charging current */
        usb_charger_current = <500000>;
        ac_charger_current = <2050000>;
        ac_charger_input_current = <3200000>;
        non_std_ac_charger_current = <500000>;
        charging_host_charger_current = <500000>;
        ta_ac_charger_current = <3000000>;
        pd_charger_current = <500000>;

        /* battery temperature protection */
        temp_t4_threshold = <50>;
        temp_t3_threshold = <45>;
        temp_t1_threshold = <0>;
};
```

```
charger: charger {
        compatible = "mediatek,charger";
        algorithm_name = "SwitchCharging";
        /* enable_sw_jeita; */
        enable_pe_plus;
        enable_pe_2;
        enable_pe_3;
        enable_pe_4;
        enable_type_c;
        power_path_support;

        /* common */
        battery_cv = <4350000>;
        max_charger_voltage = <6500000>;
        min_charger_voltage = <4600000>;

        /* charging current */
        usb_charger_current_suspend = <0>;
        usb_charger_current_unconfigured = <70000>;
        usb_charger_current_configured = <500000>;
        usb_charger_current = <500000>;
        ac_charger_current = <2050000>;
        ac_charger_input_current = <3200000>;
        non_std_ac_charger_current = <500000>;
        charging_host_charger_current = <500000>;
        apple_1_0a_charger_current = <650000>;
        apple_2_1a_charger_current = <800000>;
        ta_ac_charger_current = <3000000>;
```

charger_check_status

- 温度大于50度、低于0度禁止充电
  - #define MIN_CHARGE_TEMP 0
  - #define MAX_CHARGE_TEMP 50
- 默认最长充电12h，超过后会停止充电
  - #define MAX_CHARGING_TIME (12 * 60 * 60) /* 12 hours */
  - /kernel-4.9/drivers/power/supply/mediatek/charger/mtk_charger_init.h
  - safety_timer = <12>; /* hour */（CHG_CTRL12\WT_FC）
  - Mt6370.dtsi

GM30 充电时能否睡下去

- 标准充电器
  - AC可以，但是不会长时间睡眠，睡一会就起来看看
- USB
  - 插usb无法睡眠，usb会持锁

回充代码

- 默认是根据充电ic的状态决定复充的
- 6360/6370默认vbat低于cv值100mv进行re charger

```
/kernel-4.9/drivers/power/supply/mediatek/charger/mtk_switch_charging.c
int mtk_switch_chr_full(struct charger_manager *info)
{
……
swchg_select_cv(info);
info->polling_interval = CHARGING_FULL_INTERVAL;
charger_dev_is_charging_done(info->chg1_dev, &chg_done);
if (!chg_done) {
swchgalg->state = CHR_CC;
charger_dev_do_event(info->chg1_dev, EVENT_RECHARGE, 0);
```

外挂charger注意事项

- 如果客户使用自己的 charger，需要实作以下功能
- CHARGE_IN中断需要通知gauge
  - Charger in时需要呼叫 fg_charger_in_handler()
- 实作 battery callback:
  - 当charger状态改变时, 透过callback告知 gauge: EOC/ start_charging/ stop_charing/notify_charge_err 等状态
- 实作battery_get_charger_zcv:
  - 如果外挂charger 硬体支援上电时取得zcv的话, 需要实作 battery_get_charger_zcv(), 回报正确资料

MT6370无法充电

- 背景：
  - 使能充电，却看不到充电电流
  - mt6370_pmu_charger: mt6370_dump_register: CHG_EN = 1, CHG_STATUS = ready, CHG_STAT = 0x80
- 原因：
  - 与MT6370 chg_en PIN 相连的GPIO状态配置错了
- 修改方法：
  - 配置该GPIO为低电平

充电需要两个条件，上面的log表明，gpio状态配错了

1：使能充电（ CHG_EN = 1 ）

2：mt6370侧chg_en引脚是低电平

参考GPIO配置如下，摘自o1.mp6

```
<gpio154>
  <eint_mode>false</eint_mode>
  <def_mode>0</def_mode>
  <inpull_en>false</inpull_en>
  <inpull_selhigh>false</inpull_selhigh>
  <def_dir>OUT</def_dir>
  <out_high>false</out_high>
  <smt>false</smt>
  <ies>true</ies>
</gpio154>
```

充电指示灯（iSink）

- PCHG_LED是一路特殊的ISINK，有HW /SW两种控制方法：
  - 默认是hw控制，插入充电器，当pmic检测到有charger的时候，会自动打开这一路ISINK
  - 把PMIC的CHRIND_EN_CTRL这个regs的CHRIND_EN_SEL这个bit置为1的话，会变成SW控制，此时，这路ISINK就是一路普通的ISINK，有sw的regs可以控制电流，占空比参数（具体可以pmic datasheet）。
- 切换为sw控制（mt6357为例）
  - pmic_set_register_value(PMIC_CHRIND_EN_SEL, 1);//1F12的bit3 先设为1，切换到SW控制
  - pmic_set_register_value(PMIC_CHRIND_EN, 0);//再将1F12的bit4设为0，就可以关闭了

双充改单充

- 现象：插入充电器，前1分钟充电电流不稳定

1.修改defconfig文件

/kernel-4.9/arch/arm64/configs/{project_name}_bsp_defconfig
/kernel-4.9/arch/arm64/configs/{project_name}_dubug_defconfig
CONFIG_CHARGER_RT9465=y //要注释掉，可能是其他的slave charger ic
CONFIG_MTK_DUAL_CHARGER_SUPPORT=y //要注释掉

2.修改ProjectConfig.mk

/device/mediateksample/{project_name}_bsp/ProjectConfig.mk
MTK_DUAL_CHARGER_SUPPORT = yes //改为no

3.修改平台dts，把mt67XX.dts中的algorithm_name修改为SwitchCharging

/kernel-4.9/arch/arm/boot/dts/
/kernel-4.9/arch/arm64/boot/dts/mediatek/

```
charger: charger {
        compatible = "mediatek,charger";
        algorithm_name = "SwitchCharging";
```

关闭充电后，usb线上仍有电流

- 6370/6371具有路径管理(power path)功能
  - Ibus = Isys +Ibat
  - usb线上电流 = 系统耗电 + 进电池电流
- 关闭充电后，进电池的电流（Ibat）为0，usb线上的电流是供给系统使用的（Isys）
- 要想线上无电流，关闭充电的同时，需要关闭power path
- API

//打开或关闭power path前，请先检查power path当前状态

- charger_dev_is_powerpath_enabled(chg_dev, &is_en);
- charger_dev_enable_powerpath(chg_info->chg1_dev, true);

Meta mode无法充电

- 背景：
  - Meta mode是连接usb模式下进去的，Meta模式預期是會多台phone接同一台pc, 因此是为了防止pc usb port被抽垮
- 限制：
  - 有另外去限制input_current为200mA并停止充电, 目的同样是减少抽电诉求：
- 诉求：
  - 若想在meta下充电, 则只要把charging_enable=false的条件拿掉，再设定充电电流即可，但就會增加usb port被抽跨的几率
  - 所以请先确认使用方式真的需要在meta下充電再做修改

/kernel-4.9/drivers/power/supply/mediatek/switch_charging.c

```
} else if ((g_platform_boot_mode == META_BOOT) || (g_platform_boot_mode == ADVMETA_BOOT)) {
        battery_log(BAT_LOG_CRTI,
                "[BATTERY] In meta or advanced meta mode, disable charging.\n");
        charging_enable = false;
```

meta模式充电修改

- mtk_switch_charging.c

```
static void swchg_turn_on_charging(struct charger_manager *info)
{
……————————————
} else if ((get_boot_mode() == META_BOOT) ||
                        ((get_boot_mode() == ADVMETA_BOOT))) {
            charging_enable = false;  //拿掉这句就可以充电了
            info->chg1_data.input_current_limit = 200000; /* 200mA */   //修改ibus上的电流的，以确保ibus上200ma
            charger_dev_set_input_current(info->chg1_dev,info->chg1_data.input_current_limit);
+++         charger_dev_set_charging_current(info->chg1_dev,500000);//需要设置下后端电流，可以自行修改
……————————————
```

meta模式充电修改

- mtk_dual_switch_charging.c

```
static void dual_swchg_turn_on_charging(struct charger_manager *info)
{
……————————————
            } else if ((get_boot_mode() == META_BOOT) ||
                        (get_boot_mode() == ADVMETA_BOOT)) {
                    chg1_enable = false;        //修改为true， 只打开master
                    chg2_enable = false;
+++         charger_dev_set_input_current(info->chg1_dev,500000); //设置充电电流
+++         charger_dev_set_charging_current(info->chg1_dev,500000);

                    pr_notice("In meta mode, enable charging\n"); //修改log提示
            } else {
……————————————————
```

meta模式,充电线上电流怎么设为为0

- 默认是停止充电的（进电池的电流为0），但是input current为200ma，由于搭配的charger ic 6370是支持power path的，所以会看到充电线上还有200ma电流

```
mtk_switch_charging.c /mtk_dual_switch_charging.c
static void swchg_turn_on_charging(struct charger_manager *info)
{
……................
        } else if ((get_boot_mode() == META_BOOT) ||
                        ((get_boot_mode() == ADVMETA_BOOT))) {
            charging_enable = false;  //保留这句
            info->chg1_data.input_current_limit = 200000; /* 200mA */   //修改为0
            charger_dev_set_input_current(info->chg1_dev,info->chg1_data.input_current_limit);
        +++charger_dev_enable_powerpath(info→chg1_dev,false);//关闭power path功能
        +++ chr_err("In meta mode, disable charging and power path, set input current limit to 0mA\n");
```

Charger Type Detection in Preloader

- /vendor/mediatek/proprietary/bootable/bootloader/preloader/platform/mt67XX/src/drivers/charging_bat.c

```
CHARGER_TYPE mt_charger_type_detection(void)
{

    #ifdef MTK_EXT_CHGDET_SUPPORT
            mtk_ext_chgdet(&g_ret);      //Charger ic do bc12
    #else
            g_ret = hw_charger_type_detection();  //PMIC do bc12
    #endif
```

chr type: 1

```
typedef enum {
    CHARGER_UNKNOWN = 0,
    STANDARD_HOST,              // USB : 450mA
    CHARGING_HOST,
    NONSTANDARD_CHARGER,        // AC : 450mA~1A
    STANDARD_CHARGER,           // AC : ~1A
    APPLE_2_1A_CHARGER,         // 2.1A apple charger
    APPLE_1_0A_CHARGER,         // 1A apple charger
    APPLE_0_5A_CHARGER,         // 0.5A apple charger
} CHARGER_TYPE;
```

Get Type from atag in LK

lk/platform/mt67xx/platform.c

```
int dram_init(void)
{
    :
        for (tags = (void *)BOOT_ARGUMENT_LOCATION; tags->hdr.size; tags = boot_tag_next(tags)) {
            switch (tags->hdr.tag) {
                :
                case BOOT_TAG_CHR_INFO:
                    g_boot_arg->charger_type = tags->u.chr_info.charger_type;
                    break;
```

Lk阶段不具有charger detect功能，需要由pl阶段传charger type到lk

Agenda

- 充电基本概念
- MTK Charger框架
- MTK Charger log 解读
- MTK Charger常见问题
- MTK Charger code
  - Pl、Lk、kernel

Charger Type Detection in Preloader

- /vendor/mediatek/proprietary/bootable/bootloader/preloader/platform/mt67XX/src/drivers/charging_bat.c

```
CHARGER_TYPE mt_charger_type_detection(void)
{

    #ifdef MTK_EXT_CHGDET_SUPPORT
            mtk_ext_chgdet(&g_ret);
    #else
            g_ret = hw_charger_type_detection();
    #endif
```

External charger type detection

PMIC charger type detection

chr type: 1

```
typedef enum {
    CHARGER_UNKNOWN = 0,
    STANDARD_HOST,              // USB : 450mA
    CHARGING_HOST,
    NONSTANDARD_CHARGER,        // AC : 450mA~1A
    STANDARD_CHARGER,           // AC : ~1A
    APPLE_2_1A_CHARGER,         // 2.1A apple charger
    APPLE_1_0A_CHARGER,         // 1A apple charger
    APPLE_0_5A_CHARGER,         // 0.5A apple charger
} CHARGER_TYPE;
```

Preloader Charger Loop

- 进入while充电loop条件
  - 电池电压小于pl开进lk threshold（ bat_val < BATTERY_LOWVOL_THRESOLD ）
  - 有插充电器

```
void pl_check_bat_protect_status(void)
{

        while (bat_val < BATTERY_LOWVOL_THRESOLD)
        {
                if(upmu_is_chr_det() == KAL_FALSE)
                {
                        pal_log_info( "[PL][BATTERY] No Charger, Power OFF !\n");
                        break;
                }
```


/vendor/mediatek/proprietary/bootable/bootloader/preloader/platform/mt6785/src/drivers/charging_bat.c

Preloader Charger Error

- PI充电异常（1）
  1. 充电器过压 （chr_volt>V_CHARGER_MAX）

```
chr_volt= get_charger_volt(1);
if(chr_volt>V_CHARGER_MAX)
{
  pal_log_info( "[PL][BATTERY] charger voltage is too high :%d
  break;
}
```

/vendor/mediatek/proprietary/bootable/bootloader/preloader/platform/mt6785/src/drivers/charging_bat.c

Preloader Charger Error

- PI充电异常（2）
  2. 充电器电压过低　　（current<100 && chr_volt<4400）
  3. PI充电电流过低　　（current<100 && chr_volt<4400）

```
current=get_charging_current(1);
chr_volt=get_charger_volt(1);
if(current<100 && chr_volt<4400)
{
        cnt++;
        pal_log_info( "[PL][BATTERY] charging current=%d charger volt=%d\n\r
}
if(cnt>=8)
{

    pal_log_info( "[PL][BATTERY] charging current and charger volt too low !!

    pchr_turn_on_charging(KAL_FALSE);
            break;
}
```

/vendor/mediatek/proprietary/bootable/bootloader/preloader/platform/mt6785/src/drivers/charging_bat.c

Pl jump to lk protect

电池电压

Pl jump to lk threshold

- [pl_check_bat_protect_status]: check VBAT=3885mV with 0mV, VCHR=4730mV ,VCHR_HV=6500mv, start charging
- [pl_check_bat_protect_status]: check VBAT=3885mV with 0mV, stop charging

```
#if SWCHR_POWER_PATH
#define BATTERY_LOWVOL_THRESOLD 0
#else
#define BATTERY_LOWVOL_THRESOLD 3300
#endif
```

/vendor/mediatek/proprietary/bootable/bootloader/preloader/platform/mt6775/src/drivers/charging_bat.c

Agenda

- 充电基本概念
- MTK Charger框架
- MTK Charger log 解读
- MTK Charger常见问题
- MTK Charger code
  - Pl、Lk、kernel

lk

- VBAT >= 3.45v ➔ boot to kernel
- VBAT < 3.45v ➔ charging
  - VBAT <= 3v
    - Pre-charge: 150mA
  - VBAT > 3v
    - DCP ➔ 2A
    - Others ➔ 500mA
  - VBUS > 6.5v ➔ power off
  - TBAT > 50度 ➔ exit charging
  - TBAT > 45度 or < 0度 ➔ 500mA
- Check charging statue every 20s
- 充電動畫(press power key)
- 低電充電提示(VBAT < 3.45v and no charger)
- 過壓提示(VBUS > 6.5v)

DTS： Lk Charger Setting

- Lk charger 设定的dtsi放在kernel目录下面

参考路径：/kernel-4.XX/arch/arm64/boot/dts/mediatek/mt67xx.dts

```
lk_charger: lk_charger {
        compatible = "mediatek,lk_charger";
        enable_anime;
        /* enable_pe_plus; */
        enable_pd20_reset;
        power_path_support;
        max_charger_voltage = <6500000>;
        fast_charge_voltage = <3000000>;

        /* charging current */
        usb_charger_current = <500000>;
        ac_charger_current = <2050000>;
        ac_charger_input_current = <3200000>;
        non_std_ac_charger_current = <500000>;
        charging_host_charger_current = <1500000>;
        ta_ac_charger_current = <3000000>;
        pd_charger_current = <500000>;
```

LK Log – Charger Setting

[init_cust_data_from_dt]:chroff:0,pe:1,powpath:1,vchrmax:6500,vfast:3000,usb:500,ac:2050 3200,nac:500,cdp:500,ta:3000,pd:500,t:50 45 0

- chroff: disable/enable charger
- pe: enable pe
- powpath: power path support
- vchrmax: max charger voltage
- vfast: voltage threshold to charge large current
- usb: usb charging current limit
- ac: ac charging current limit, ac input current limit
- nac: nonstandard charger current limit
- cdp: charging host current limit
- ta: pe current limit
- pd: pd current limit
- t: stop temp. max temp. min temp.

this log show charger setting in lk

LK Log – Charging or Not

is_low_battery, FALSE

- VBAT > 3450 → Jump to kernel

is_low_battery, TRUE

[check_bat_protect_status]: check VBAT=3049 mV with 3450 mV

charger_type: 4

mt_charger_set_aicr: aicr = 3200

mt_charger_set_ichg: ichg = 2050

[check_bat_protect_status]: check VBAT=3265 mV with 3450 mV, start charging...

:

- 3049 < 3450 → Charging
- charger_type: 4 → standard charger
- check VBAT=3265 mV with 3450 mV, start charging... → Check VBAT change

Set type to atag in Preloader

- Lk does't support charger type detection
- It uses the detected result in preloader by atag

Preloader/platform/mt6771/src/drivers/platform.c

```
#if CFG_BOOT_ARGUMENT_BY_ATAG
void platform_set_boot_args_by_atag(unsigned *ptr)
{
    :
    tags = ptr;
    tags->hdr.size = boot_tag_size(boot_tag_chr_info);
    tags->hdr.tag  = BOOT_TAG_CHR_INFO;
#if !CFG_FPGA_PLATFORM
    tags->u.chr_info.charger_type = mt_charger_type_detection();
#else
    tags->u.chr_info.charger_type = STANDARD_HOST;
#endif
    ptr += tags->hdr.size;
```

Get Type from atag in LK

lk/platform/mt67XX/platform.c

```
int dram_init(void)
{
    :
        for (tags = (void *)BOOT_ARGUMENT_LOCATION; tags->hdr.size; tags = boot_tag_next(tags)) {
            switch (tags->hdr.tag) {
                :
                case BOOT_TAG_CHR_INFO:
                    g_boot_arg->charger_type = tags->u.chr_info.charger_type;
                    break;
```

lk/platform/common/power/mtk_charger.c

```
static void select_charging_current_limit(void)
{
    int input_current_limit;
    int charging_current_limit;
    CHARGER_TYPE chr_type = g_boot_arg->charger_type;

    dprintf(INFO, "charger_type: %d\n", chr_type);
```

Lk阶段不具有charger detect功能，需要由pl阶段传charger type到lk

mtk_charger_init

```
int mtk_charger_init(void)
{
        init_charger_custom_data();
        ret = init_cust_data_from_dt();

        if (ret) {
                charger_driver_init();
                return ret;
        }


        if(!is_disable_charger()) {
                for (i = 0; i < size; i++) {
                        charger_init = mtk_charger_init_list[i];
                        ret = (*charger_init)();
                        if (ret < 0)
                                dprintf(CRITICAL,
                                        "%s: fail to init charger(%d), ret = %d\n",
                                        __func__, i, ret);
                }
        }
```

Get charger data from dts

Init charger ic driver

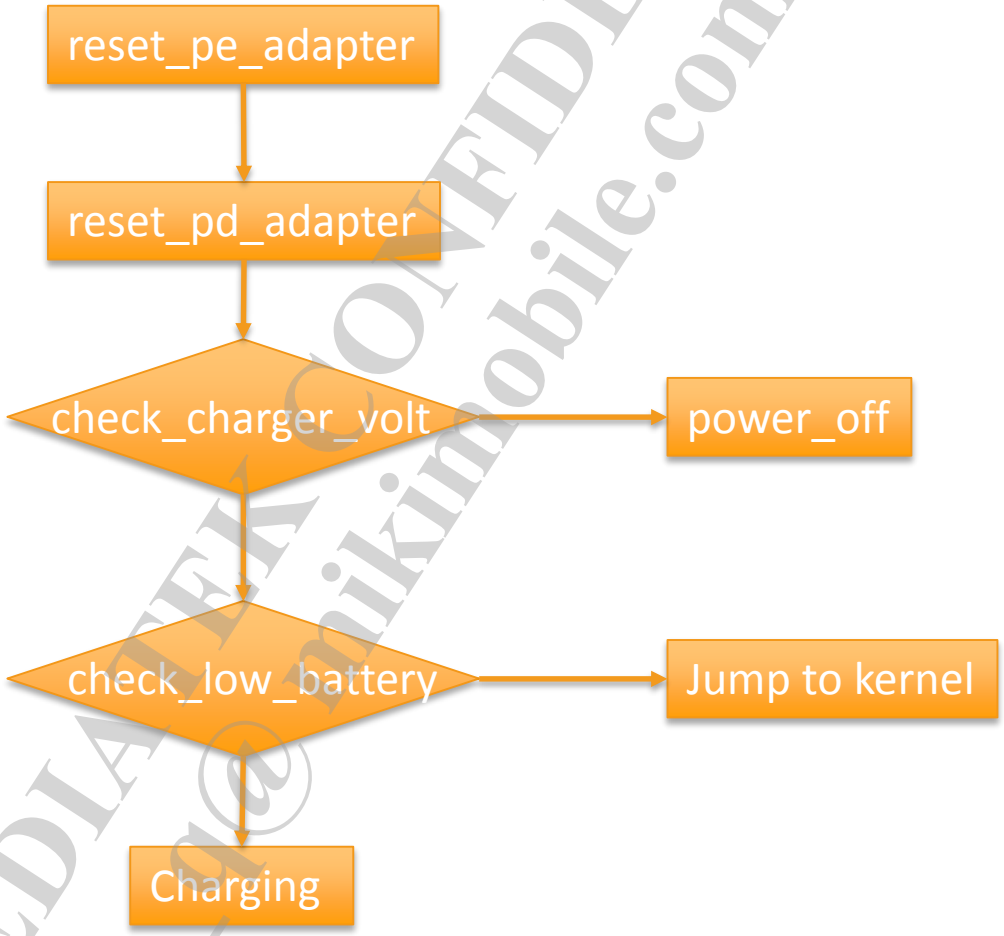
/vendor/mediatek/proprietary/bootable/bootloader/lk/platform/common/power/mtk_charger_intf.c

mtk_charger_start

重置快充设置，防止快充情况下关机，进入kpoc 高压充电

OV protection

Check if need charging



/vendor/mediatek/proprietary/bootable/bootloader/lk/platform/common/power/mtk_charger.c

check_charger_volt

- 触发charger 喊关机的条件
  - Case 1: Vbus <4V
  - Case 2: Support power path && vbat > 6.5v

1

```
chr_volt = get_chr_volt();
if (chr_volt < chr_cust_data.min_charger_voltage) {
        dprintf(INFO, "vbus is less than %dmv, power off\n",
                chr_cust_data.min_charger_voltage);
        chr_power_off();
}
```

2

```
if (chr_cust_data.power_path_support == true) {
        chr_volt = get_chr_volt();
        if (chr_volt > chr_cust_data.max_charger_voltage) {
                dprintf(CRITICAL, "Charger Over Voltage:%d\n, power off...", chr_volt);
                show_plug_out_notify();
                chr_power_off();
        }
}
```

显示图片提示

/vendor/mediatek/proprietary/bootable/bootloader/lk/platform/common/power/mtk_charger.c

check_low_battery

```
void check_low_battery(void)
{
  int bat_vol;
  bat_vol = get_bat_volt(1);

  if (is_low_battery(bat_vol)) {
          if (g_boot_mode == KERNEL_POWER_OFF_CHARGING_BOOT && upmu_is_chr_det() == 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 {
                  dprintf(CRITICAL, "[BATTERY] battery voltage(%dmV) <= CLV ! Can not Boot Linux Kernel !! \n\
                  show_low_battery_notify();
                  chr_power_off();
                  while (1) {
                          dprintf(CRITICAL, "If you see the log, please check with RTC power off API\n\r");
                  }
          }
  }
}
```

Low battery && kpoc mode && vbus

LK charger loop

Low battery && !kpoc mode && !vbus

/lk/platform/common/power/mtk_charger.c

is_low_battery

```
bool is_low_battery(int val)
{

  if (val < BATTERY_LOWVOL_THRESOLD) {
          dprintf(INFO, "%s, TRUE\n", __func__);
          g_bat_low = true;
  }
  else {
          dprintf(INFO, "%s, FALSE\n", __func__);
          g_bat_low = false;
  }

  return g_bat_low;
}
```

Vbat < 3.45v 返回true

/lk/platform/common/power/mtk_charger.c

```
#define BATTERY_LOWVOL_THRESOLD CUST_BATTERY_LOWVOL_THRESOLD
#else
#define BATTERY_LOWVOL_THRESOLD                3450
#endif
```

check_bat_protect_status

```
void check_bat_protect_status(void)
{

    ret = mtk_charger_enable_charging(primary_mchr, false);

    bat_val = get_bat_volt(5);
    dprintf(CRITICAL, "[%s]: check VBAT=%d mV with %d mV\n", __func__,
            bat_val, BATTERY_LOWVOL_THRESOLD);

    while (bat_val < BATTERY_LOWVOL_THRESOLD) {

        check_charger_battery_on();
        check_charger_volt();
```

Disable charger and get VBAT

Check if VBAT < 3.45v

Check if battery exists
Check charger voltage

/lk/platform/common/power/mtk_charger.c

check_bat_protect_status(cont.)

```
while (bat_val < BATTERY_LOWVOL_THRESOLD) {

        ...........

        temperature = force_get_tbat(true);
        dprintf(INFO, "%s: T=%d\n", __func__, temperature);
        if(temperature > chr_cust_data.temp_t4_threshold) {
                dprintf(CRITICAL,"[BATTERY] Battery over Temperature or NTC fail
                        chr_cust_data.temp_t4_threshold);
                break;
        }
```

If temp > 50, end charging and jump kernel

```
        if (bat_val < chr_cust_data.fast_charge_voltage ||
                temperature > chr_cust_data.temp_t3_threshold ||
                temperature < chr_cust_data.temp_t1_threshold)
                reset_default_charging_current_limit();
        else
                select_charging_current_limit();

        ret = mtk_charger_enable_charging(primary_mchr, true);
```

If VBAT < 3v or temp > 45 or temp < 0,
set AICR to 500mA
else
set AICR according to charger type

Enable charging

check_bat_protect_status(cont.)

```
while (bat_val < BATTERY_LOWVOL_THRESOLD) {

        ............

    if (chr_cust_data.enable_anime && bat_val > chr_cust_data.fast_charge_voltage) {
            if (is_first) {
                    show_charging_anime();
                    is_first = 0;
            }
            for (i = 0; i < MAX_SLEEP_LOOP; i++) {
                    mtk_wdt_restart();
                    check_charger_battery_on();
                    /* set polling period */
                    thread_sleep(1000);
                    if(get_powerkey_pressed_status()) {
                            clear_powerkey_pressed_status();
                            show_charging_anime();
                    }
            }
    } else {
```

Only when Vbat > 3v show anime

对应图片

wait 20s for charing
(MAX_SLEEP_LOOP = 20)

When press power key, show anime

check_bat_protect_status(cont.)

```
while (bat_val < BATTERY_LOWVOL_THRESOLD) {

        ............



        } else {
                for (i = 0; i < MAX_SLEEP_LOOP; i++) {
                        mtk_wdt_restart();
                        check_charger_battery_on();
                        /* set polling period */
                        thread_sleep(1000);
                }
        }
```

when Vbat < 3v , no ainimation

wait 20s for charing
(MAX_SLEEP_LOOP = 20)

check_bat_protect_status(cont.)

```
while (bat_val < BATTERY_LOWVOL_THRESOLD) {

        ............

        if (is_battery_on()) {
                gauge_get_current(&curr_sign, &bat_current);
                bat_current = bat_current / 10;
                dprintf(INFO, "%s:IBAT=%d\n", __func__, curr_sign ? bat_current : -1 * bat_current);
        }

        if (g_boot_arg->charger_type == STANDARD_CHARGER && bat_val > chr_cust_data.fast_charge_voltage) {
                ret = mtk_charger_enable_charging(primary_mchr, false);
                if (ret < 0)
                        dprintf(INFO, "%s: disable charging failed, ret
        }
        bat_val = get_bat_volt(5);
        dprintf(INFO, "[%s]: check VBAT=%d mV with %d mV, start charging...\n", __func__,
                bat_val, BATTERY_LOWVOL_THRESOLD);
}
dprintf(INFO, "[PROFILE] ------- Charging takes %d ms -------- \n", (int)get_timer(time_charging));
mtk_wdt_restart();
reset_default_charging_current_limit();
```

measure Ibat

Disable charging to measure VBAT when using large charging current

End while

Set AICR to 500mA when charging completed

Kpoc animation in lk

- 3V以下不会亮背光

Kpoc animation in lk

```
void platform_init(void)
{
    #ifdef MTK_KERNEL_POWER_OFF_CHARGING
            if (kernel_charging_boot() == 1) {
                    PROFILING_START("show logo");
    #ifdef MTK_BATLOWV_NO_PANEL_ON_EARLY
                    CHARGER_TYPE CHR_Type_num = CHARGER_UNKNOWN;
                    CHR_Type_num = hw_charging_get_charger_type();
                    if ((g_boot_mode != LOW_POWER_OFF_CHARGING_BOOT)
                            ((CHR_Type_num != STANDARD_HOST) && (CHR
    #endif // MTK_BATLOWV_NO_PANEL_ON_EARLY
                            mt_disp_power(TRUE);
    #ifndef MACH_FPGA_NO_DISPLAY
                            mt_disp_show_low_battery();
    #endif
                            mt65xx_leds_brightness_set(6, 110);
    #ifdef MTK_BATLOWV_NO_PANEL_ON_EARLY
                    }
    #endif
```

/vendor/mediatek/proprietary/bootable/bootloader/lk/platform/mt67XX/platform.c

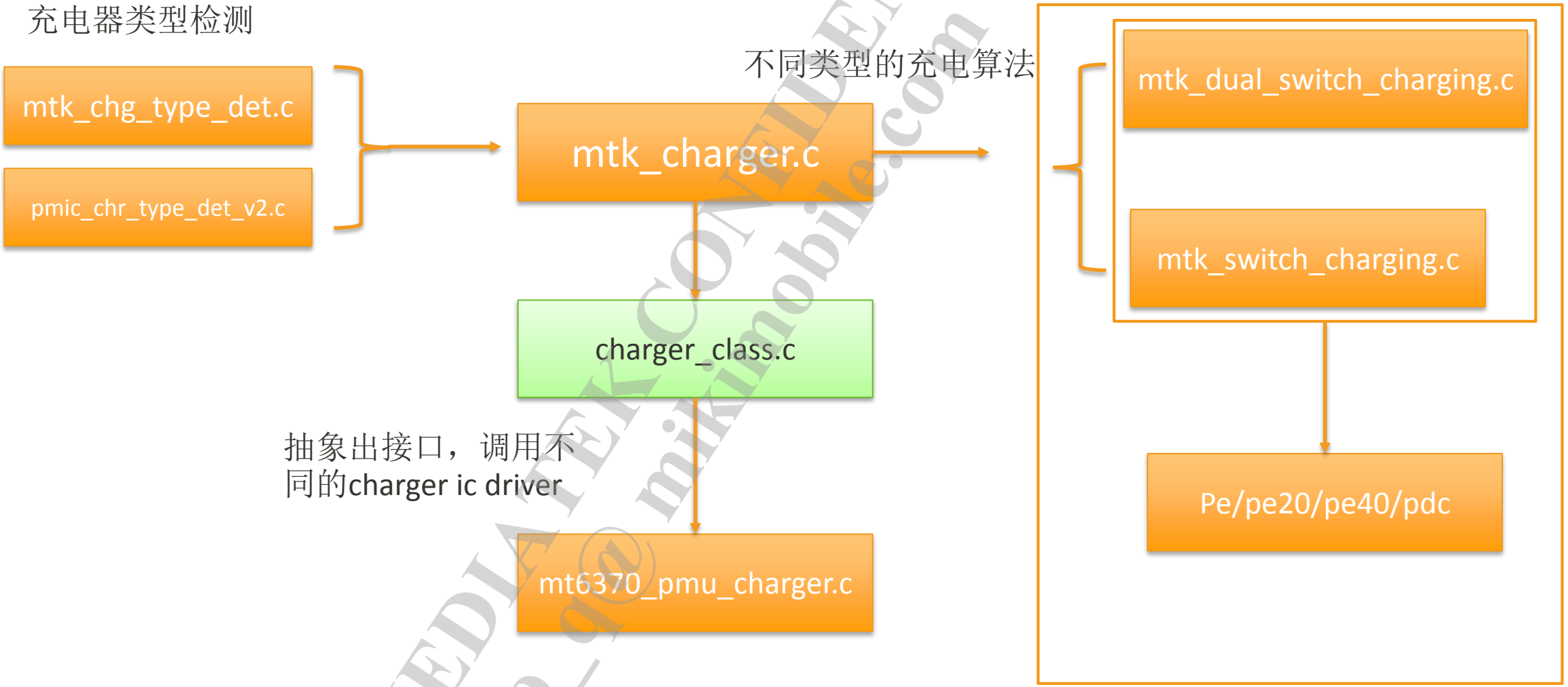
Kpoc animation in lk

```
//time_sleeping = get_timer(0);                    //vbat > 3V，才会有充电动画
if (chr_cust_data.enable_anime && bat_val > chr_cust_data.fast_charge_voltage) {
        if (is_first) {
                show_charging_anime();
                is_first = 0;
        }
        for (i = 0; i < MAX_SLEEP_LOOP; i++) {
                mtk_wdt_restart();
                check_charger_battery_on();
                /* set polling period */
                thread_sleep(1000);
                if(get_powerkey_pressed_status()) {
                        clear_powerkey_pressed_status();
                        show_charging_anime();
                }
        }
} else {
```

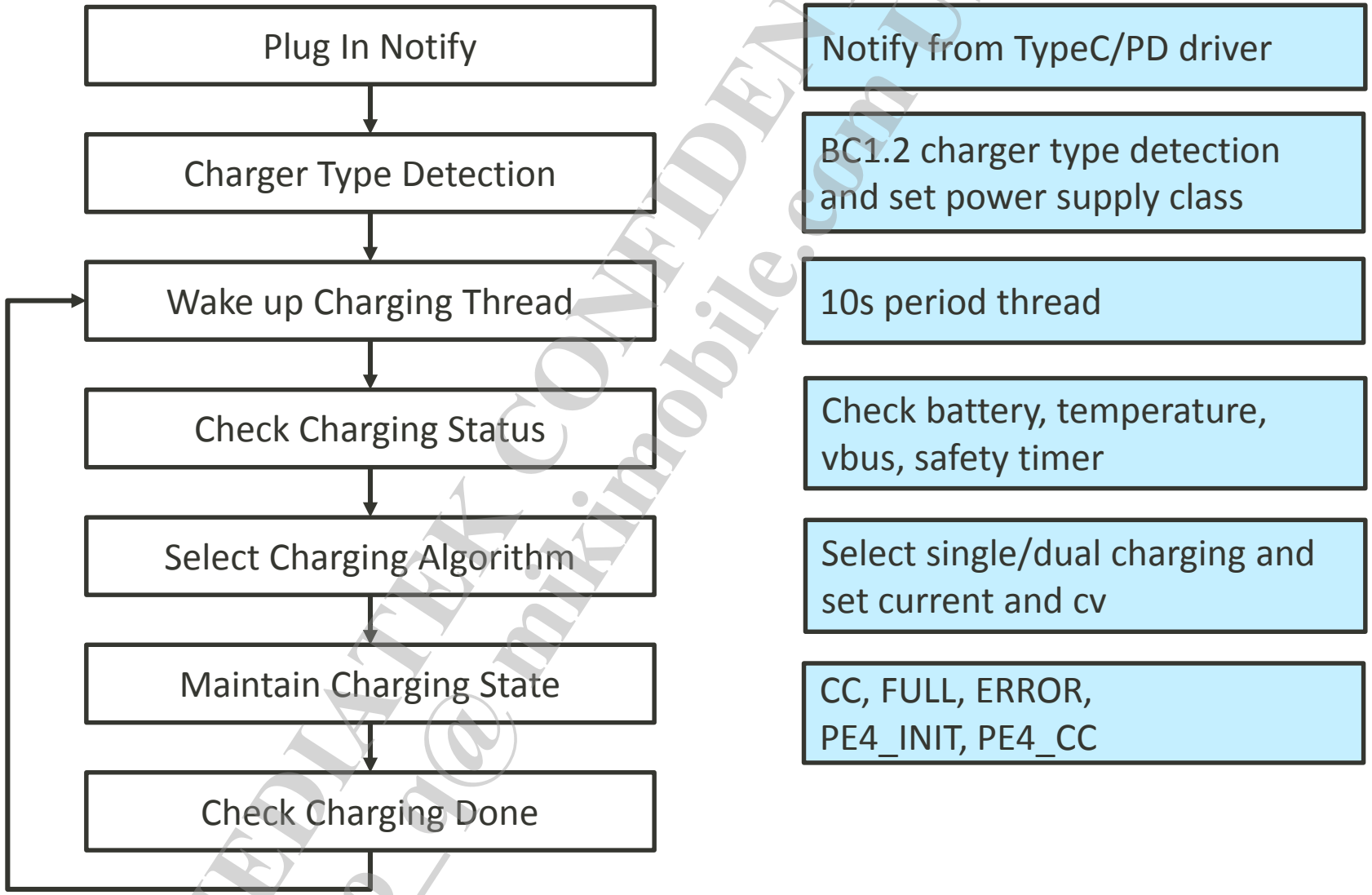
Agenda

- 充电基本概念
- MTK Charger框架
- MTK Charger log 解读
- MTK Charger常见问题
- MTK Charger code
  - Pl、Lk、kernel

Charger in kernel



Kernel - Plug in Flow Chart



Type C： Kernel - Plug in Flow

pd_tcp_notifier_call

↓

plug_in_out_handler

↓

chgdet_task_threadfn

↓

Charger Type Detection

Charger ic判port
搭配6358、6359的平台都是charger ic 判port

```
#ifdef CONFIG_MTK_EXTERNAL_CHARGER_TYPE_DETECT
        if (cti->chg_consumer)
                charger_manager_enable_chg_type_det(cti->chg_consumer,
                                                    attach);
#else
        mtk_pmic_enable_chr_type_det(attach);
#endif
```

pmic判port

/kernel-4.14/drivers/power/supply/mediatek/charger/mtk_chg_type_det.c

External Charger Type Detection Flow

```
int charger_manager_enable_chg_type_det(struct charger_consumer *consumer,
  bool en)
{

    if (info != NULL) {
            switch (info->data.bc12_charger) {
            case MAIN_CHARGER:
                    chg_dev = info->chg1_dev;
                    break;
            case SLAVE_CHARGER:
                    chg_dev = info->chg2_dev;
                    break;
            default:
                    chg_dev = info->chg1_dev;
                    chr_err("%s: invalid number, use main charger as default\n",
                            __func__);
                    break;
            }

            chr_err("%s: chg%d is doing bc12\n", __func__,
                    info->data.bc12_charger + 1);
            ret = charger_dev_enable_chg_type_det(chg_dev, en);
```

maincharger ic 判port

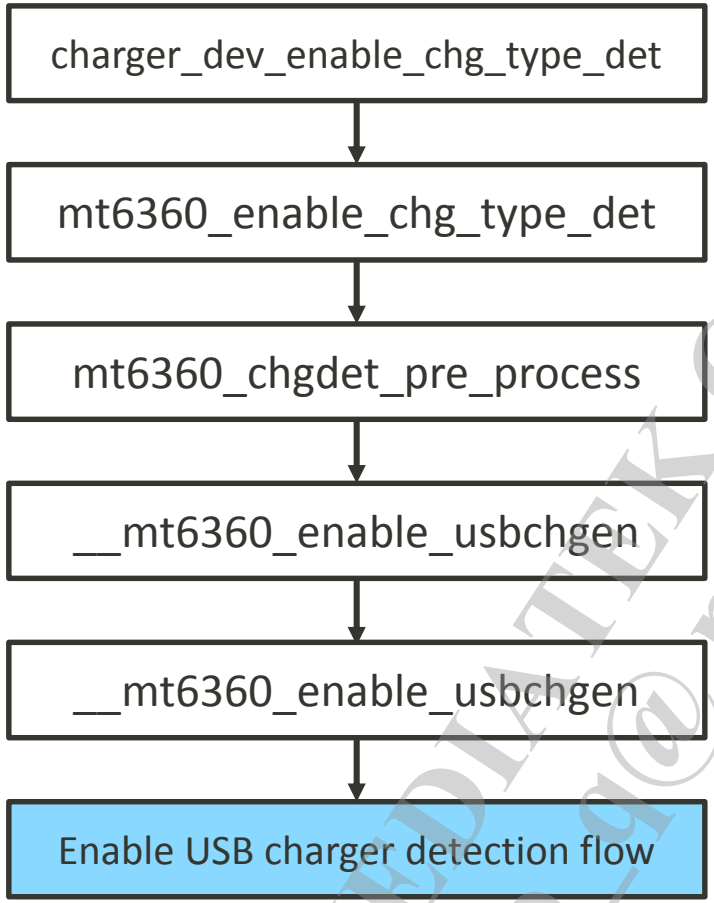
Slave charger ic 判port

呼叫charger ic driver 判port 函数

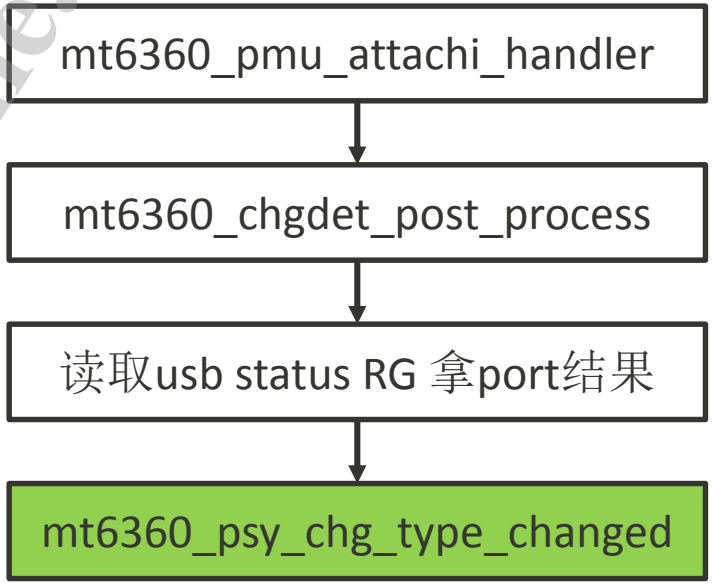
/kernel-4.14/drivers/power/supply/mediatek/charger/mtk_charger.c

mt6360_enable_chg_type_det

mt6360_pmu_chg.c

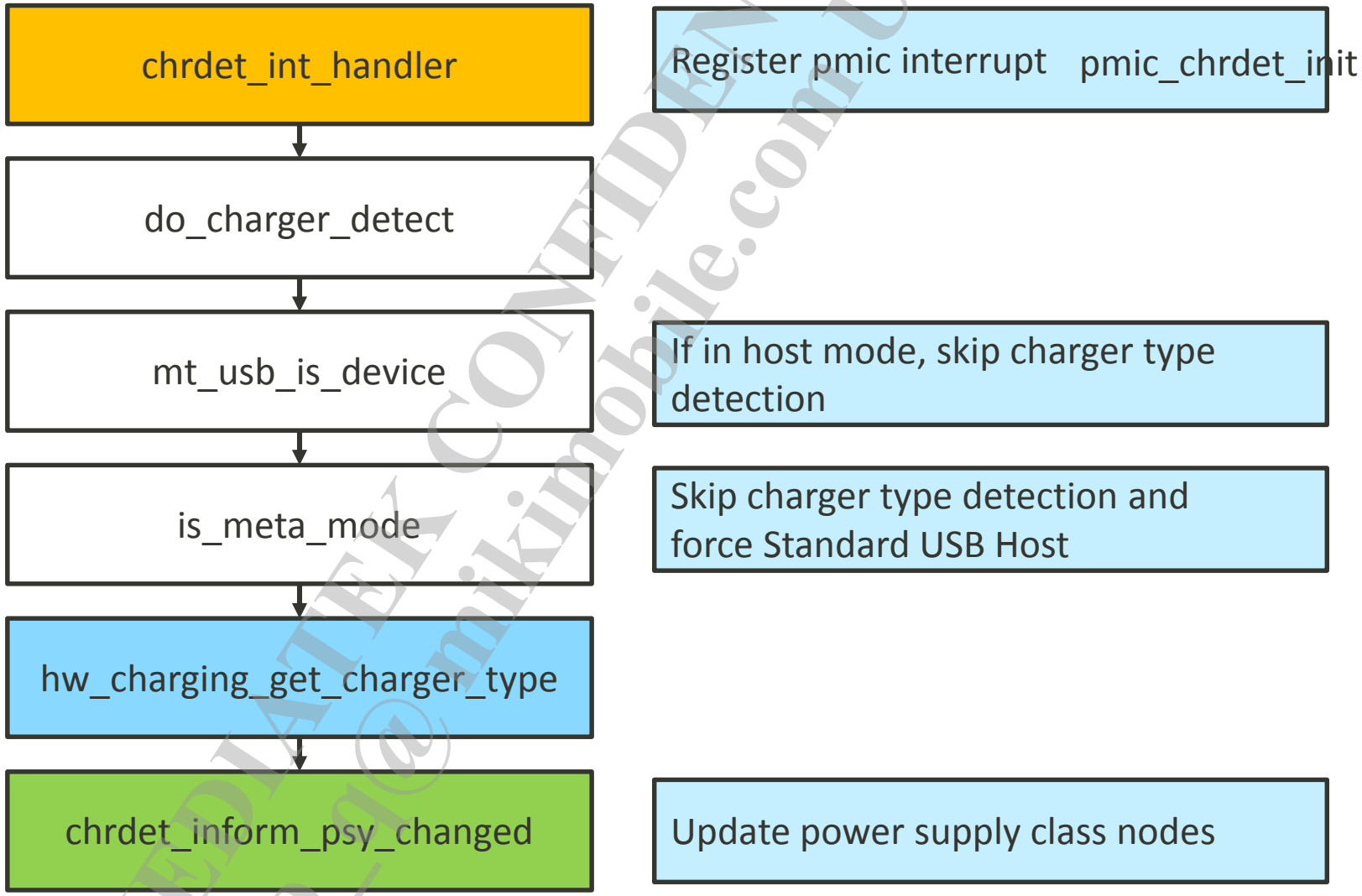


判完port后的中断



通知charger 判port 结果

PMIC Charger Type Detection Flow



/kernel-4.9/drivers/misc/mediatek/pmic/mt6357/v1/pmic_chr_type_det_v2.c

PMIC Charger Type Detection

pmic_chr_type_det_v2.c

```
int hw_charging_get_charger_type(void)
{
    enum charger_type CHR_Type_num = CHARGER_UNKNOWN;
    :
    hw_bc11_init();
    if (hw_bc11_DCD()) {
        if (hw_bc11_stepA1())
            CHR_Type_num = APPLE_2_1A_CHARGER;
        else
            CHR_Type_num = NONSTANDARD_CHARGER;
    } else {
        if (hw_bc11_stepA2()) {
            if (hw_bc11_stepB2())
                CHR_Type_num = STANDARD_CHARGER;
            else
                CHR_Type_num = CHARGING_HOST;
        } else
            CHR_Type_num = STANDARD_HOST;
    }
    if (CHR_Type_num != STANDARD_CHARGER)
        hw_bc11_done();
    else
        pr_info("charger type: skip bc11 release for BC12 DCP SPEC\n");
    dump_charger_name(CHR_Type_num);
#ifdef __FORCE_USB_TYPE__
    CHR_Type_num = STANDARD_HOST;
    pr_info("charger type: Froce to STANDARD_HOST\n");
#endif
    return CHR_Type_num;
}
```

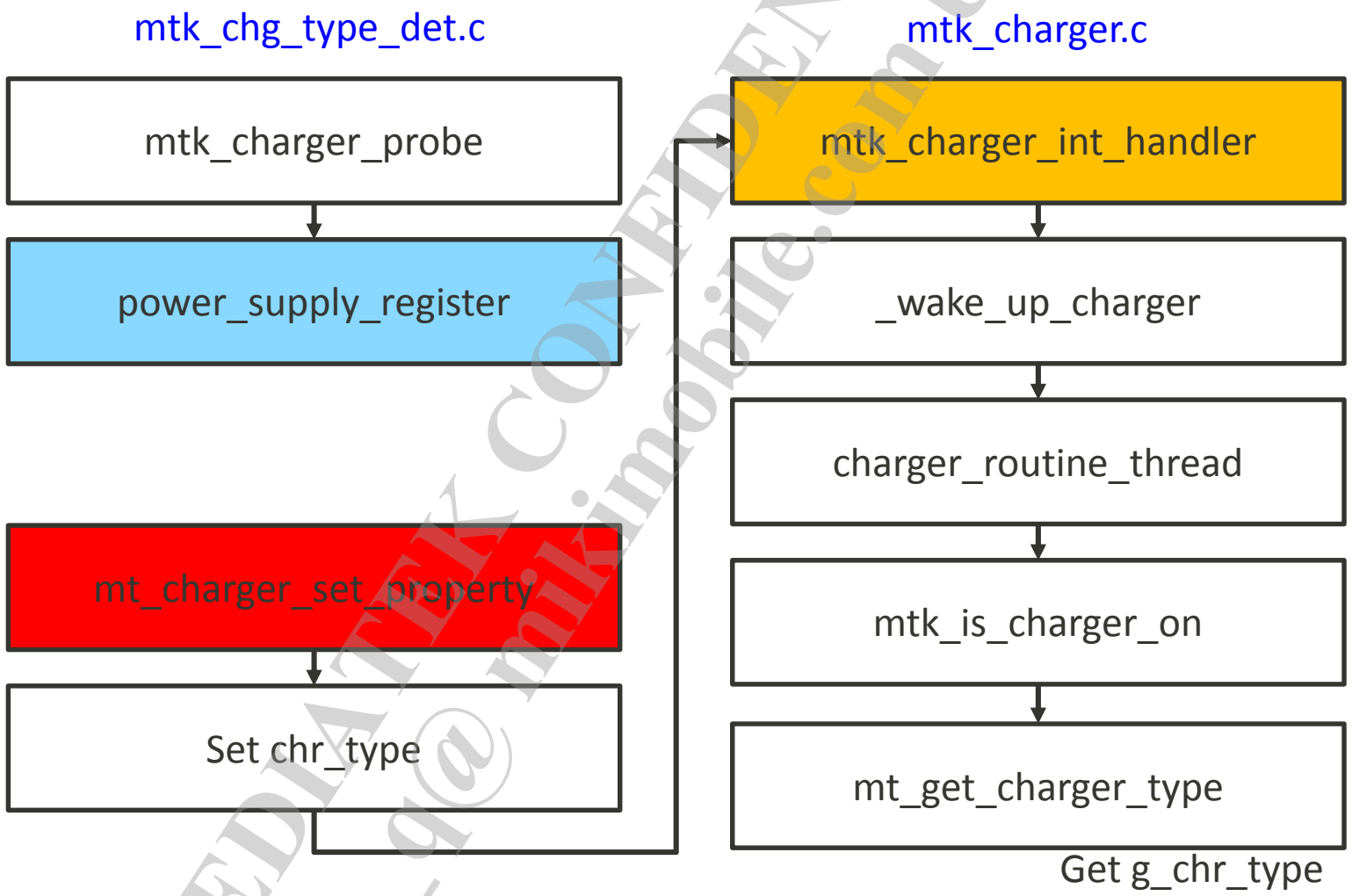
DCD

Primary detection ➔ SDP or DCP/CDP

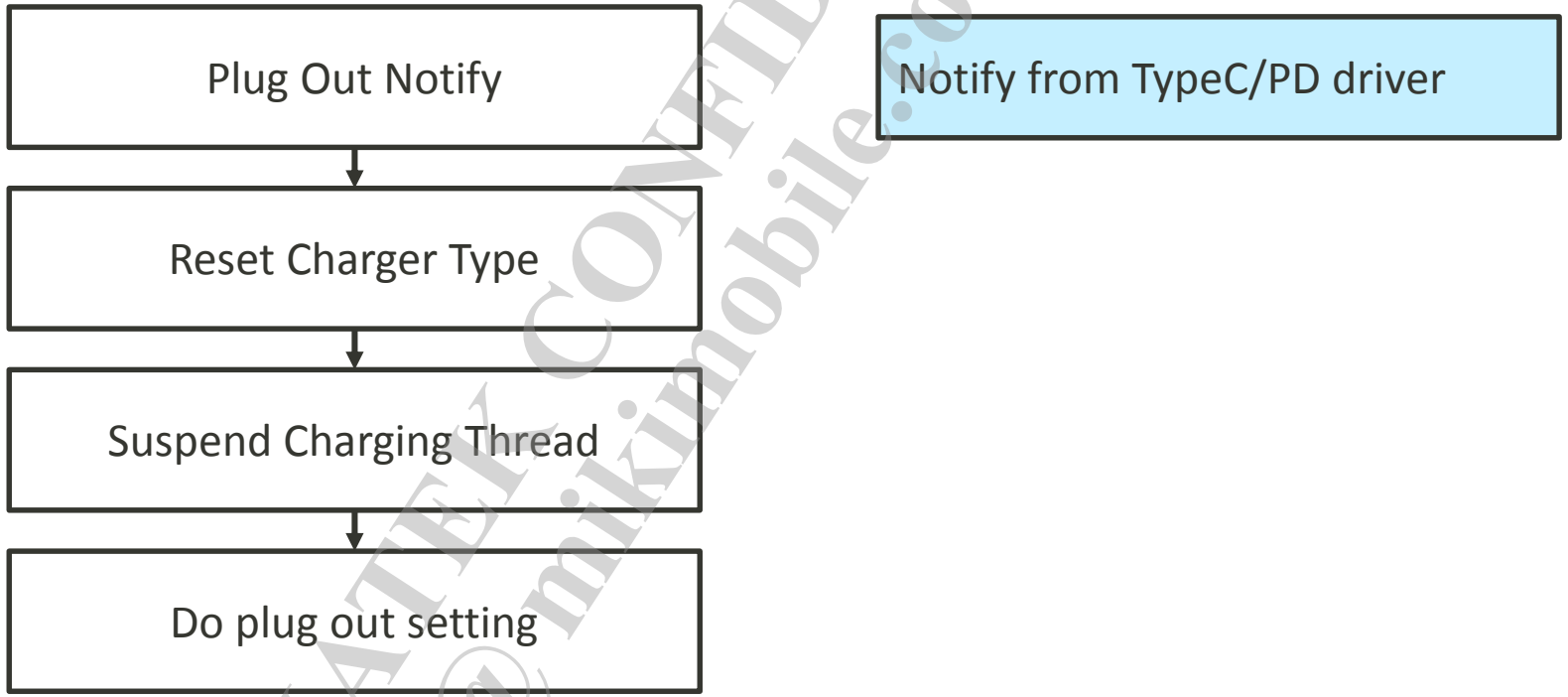
Secondary detection ➔ DCP or CDP

Force USB

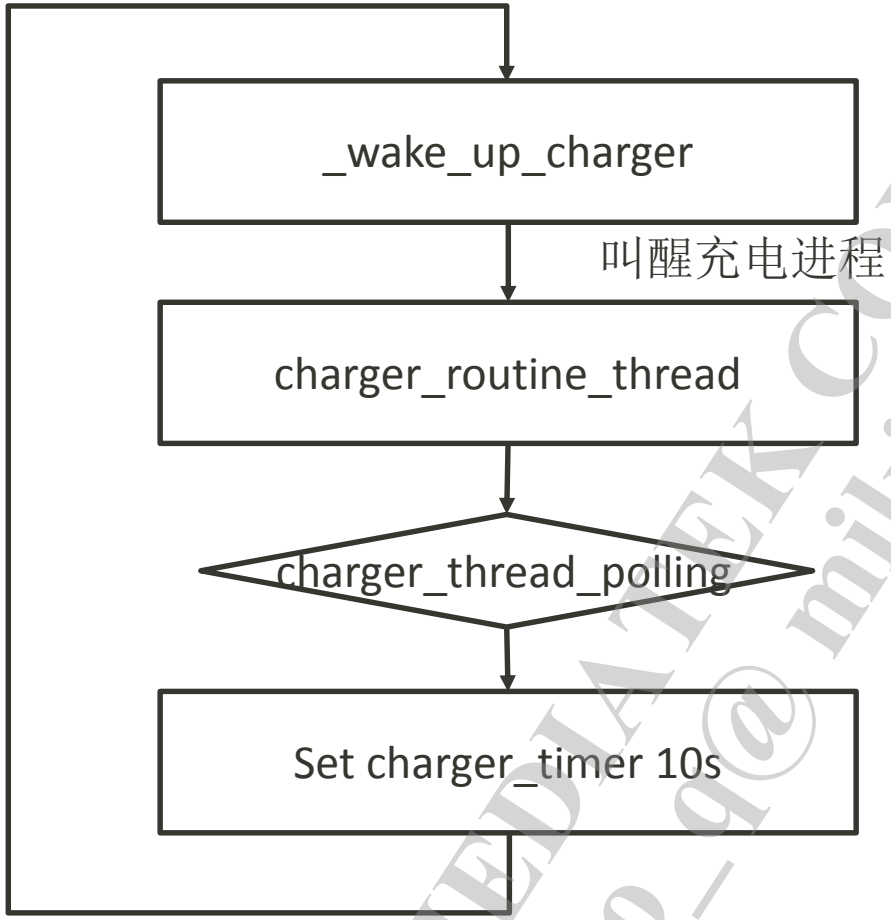
How to Get Charger Type in Charger?



Kernel - Plug out Flow Chart



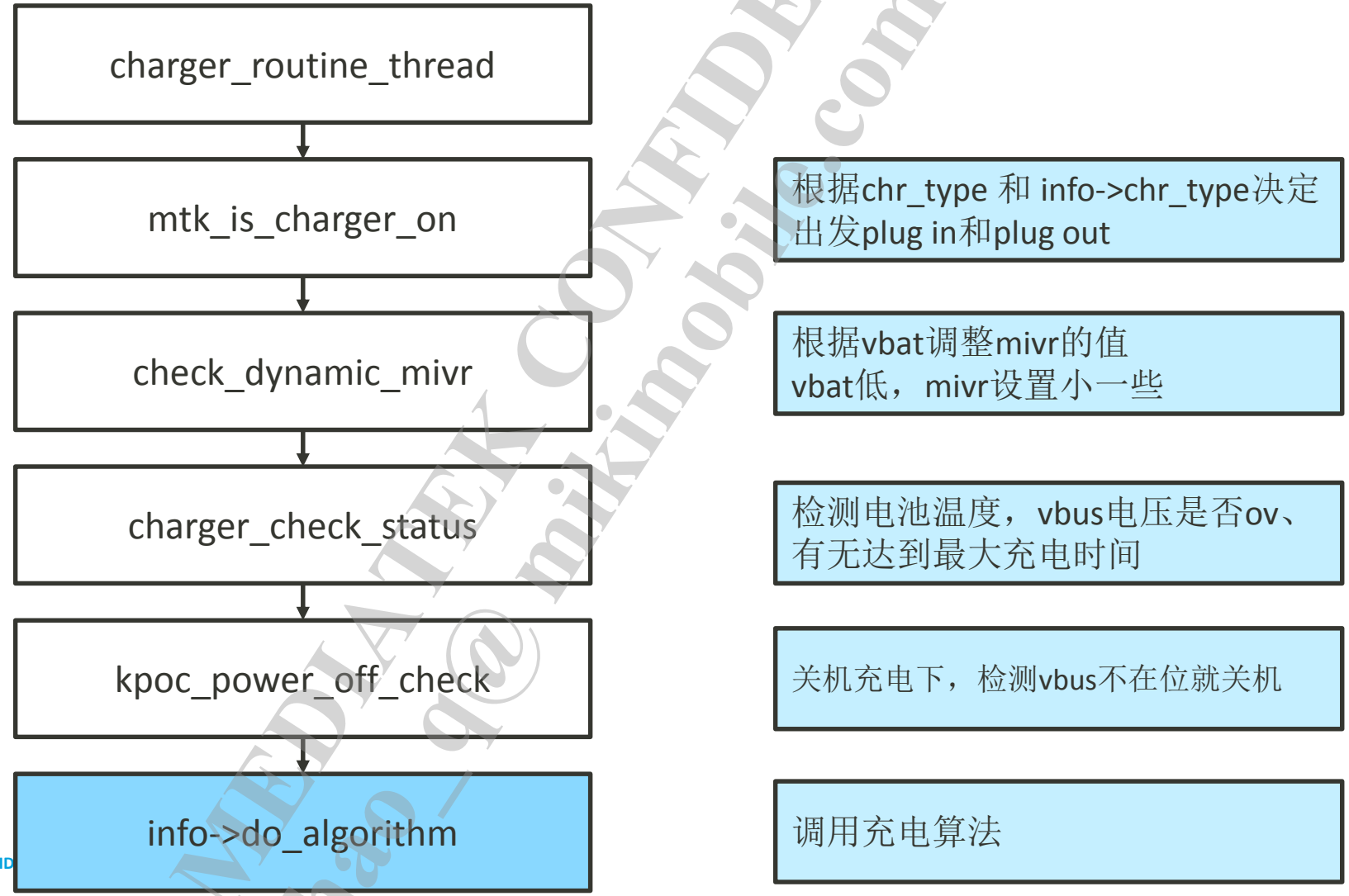
charger_routine_thread



插入充电器charger_thread_polling会置位1，充电进程会10s polling

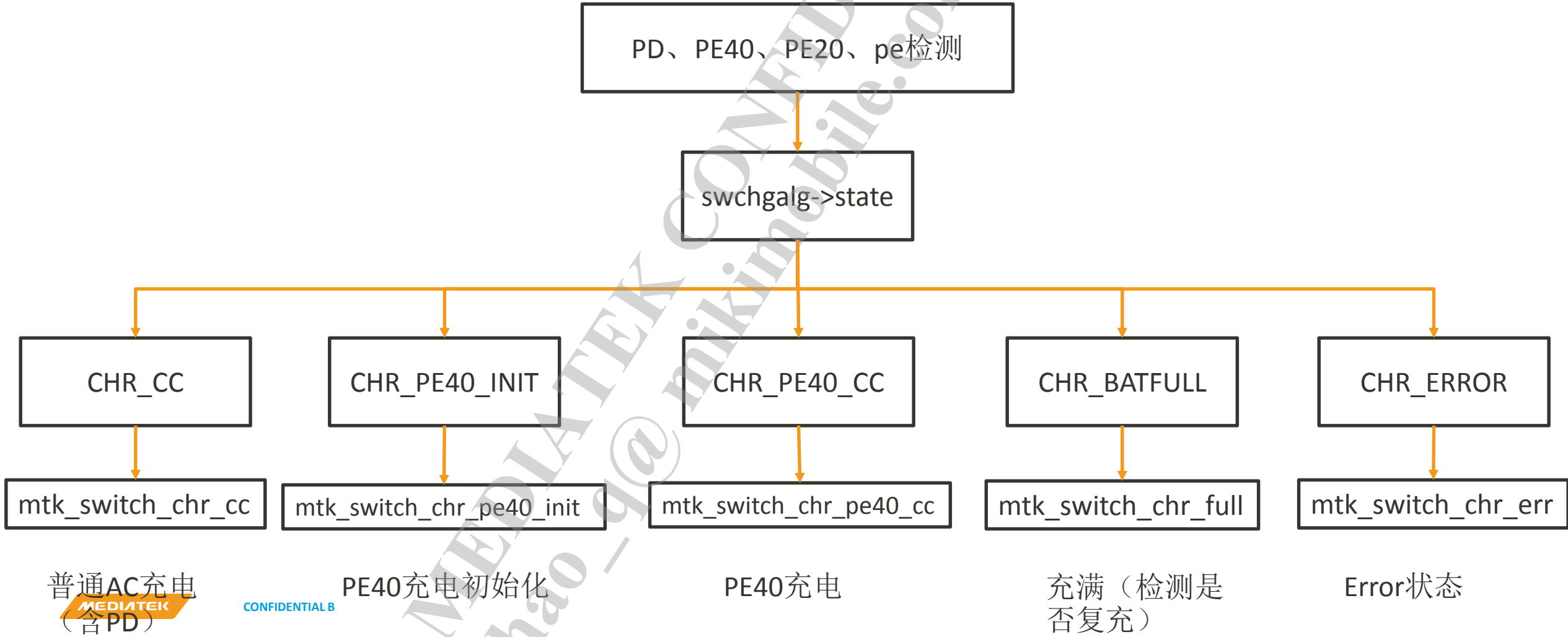
Alrm timer， 系统suspend的情况下，此timer 每10s打醒系统

charger_routine_thread

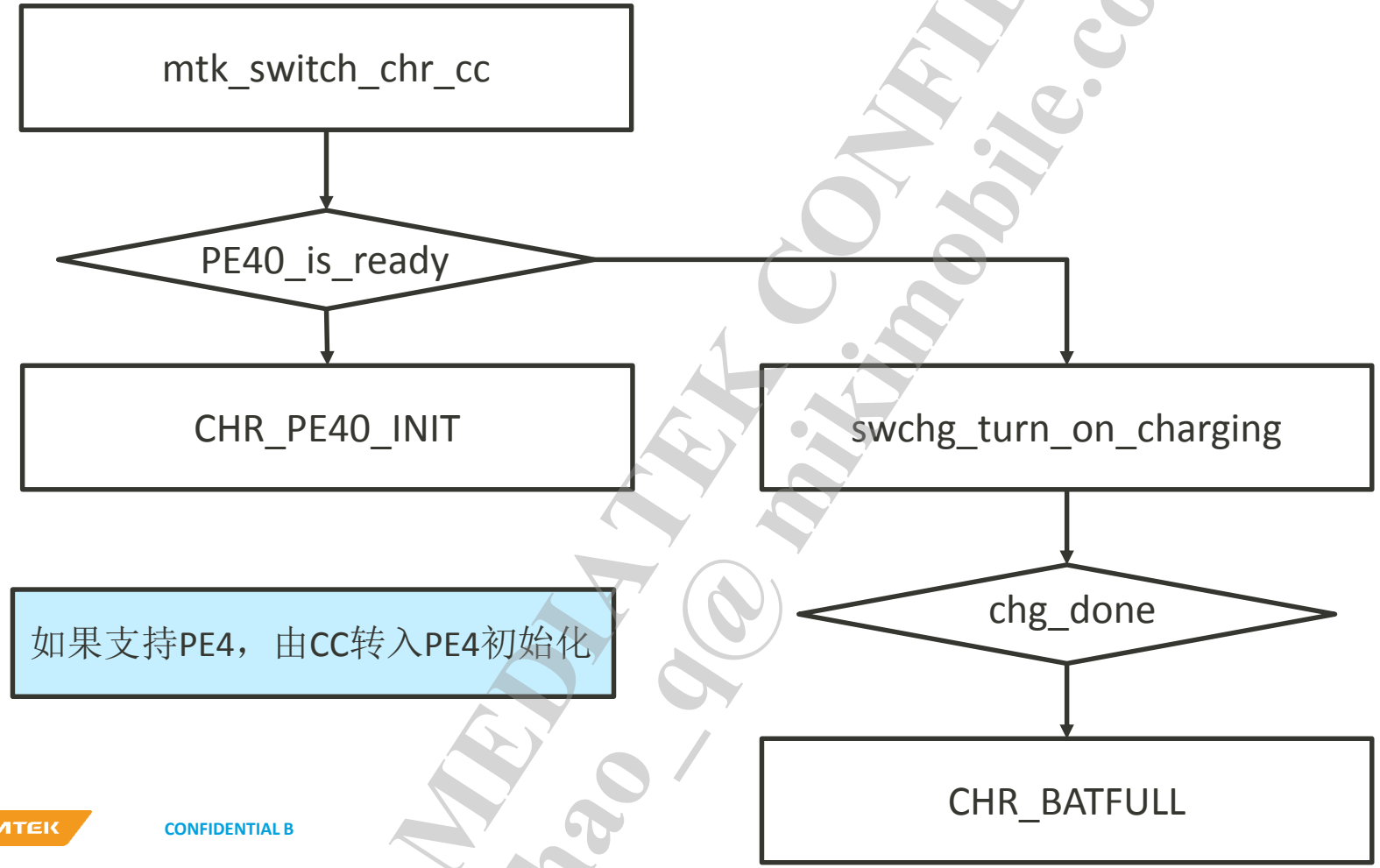


mtk_switch_charging_run

```
info->do_algorithm = mtk_switch_charging_run;
```



mtk_switch_chr_cc

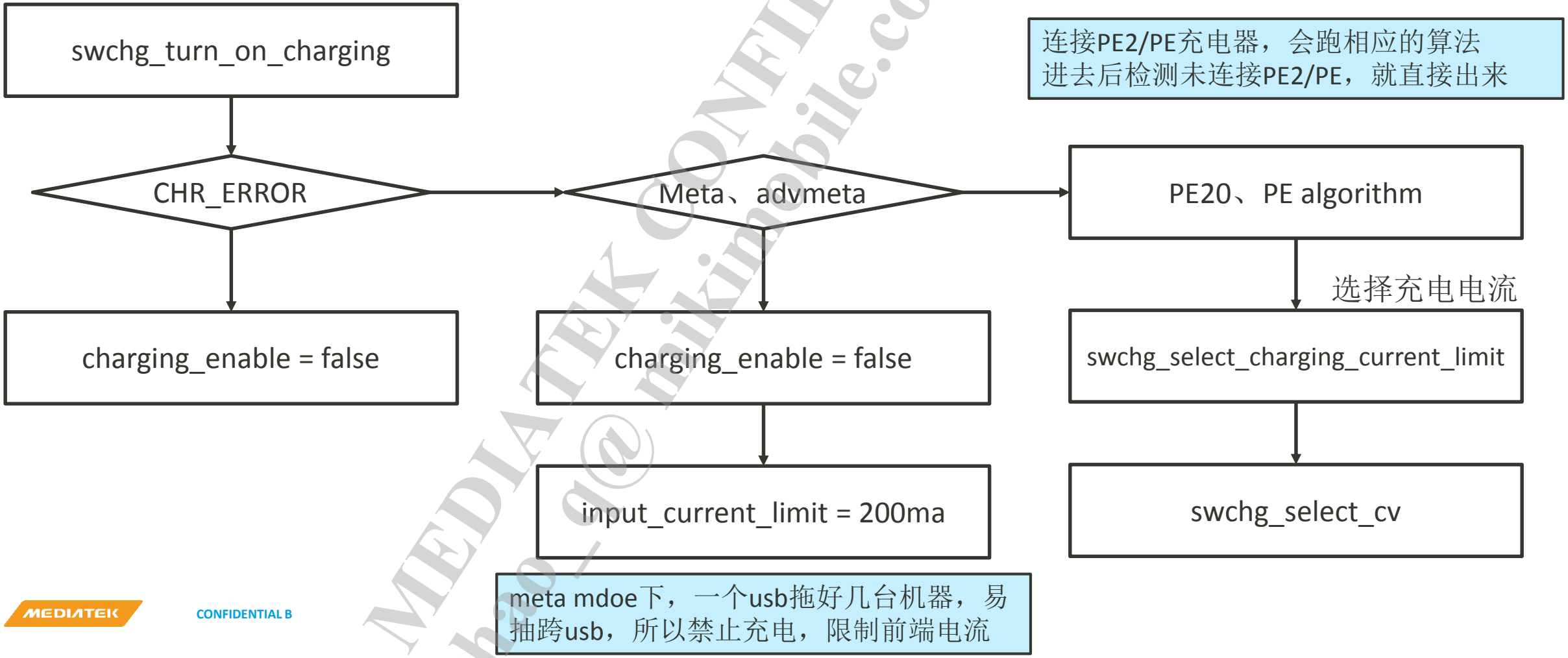


如果支持PE4，由CC转入PE4初始化

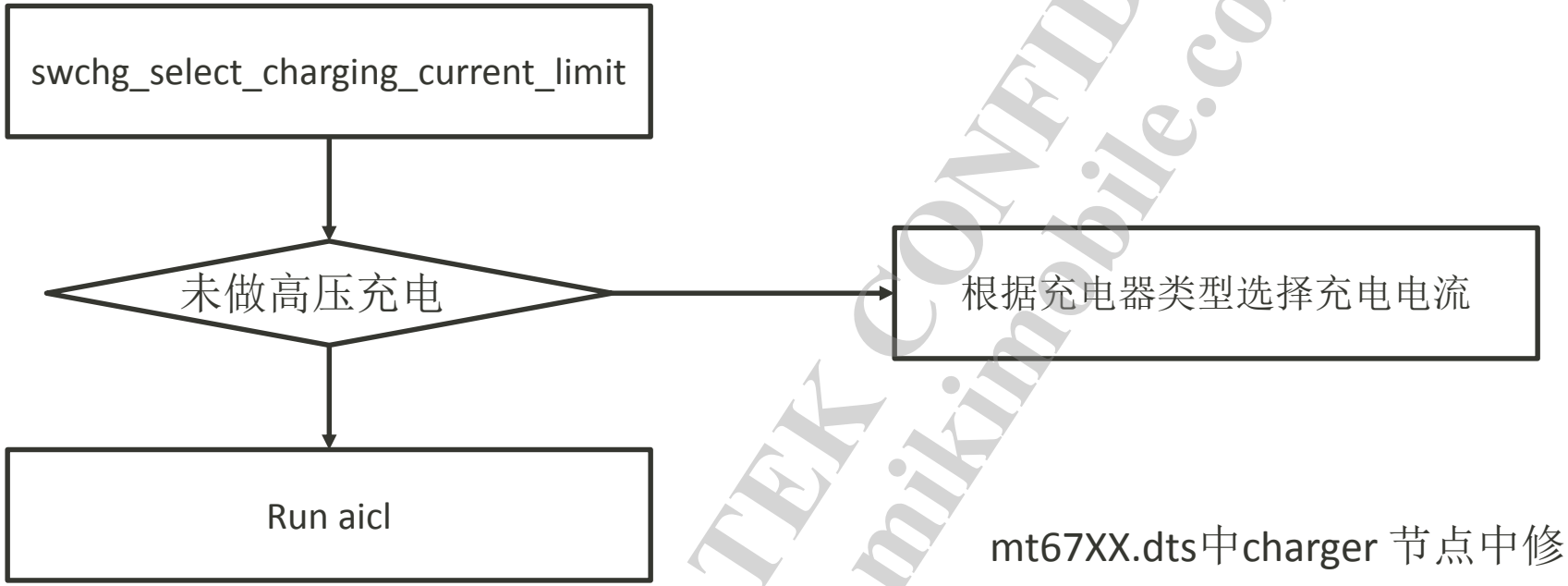
根据供电器类型，
选择充电电流

检测charger ic 状态，如果是
chg_done，表示已经充满

swchg_turn_on_charging



swchg_select_charging_current_limit



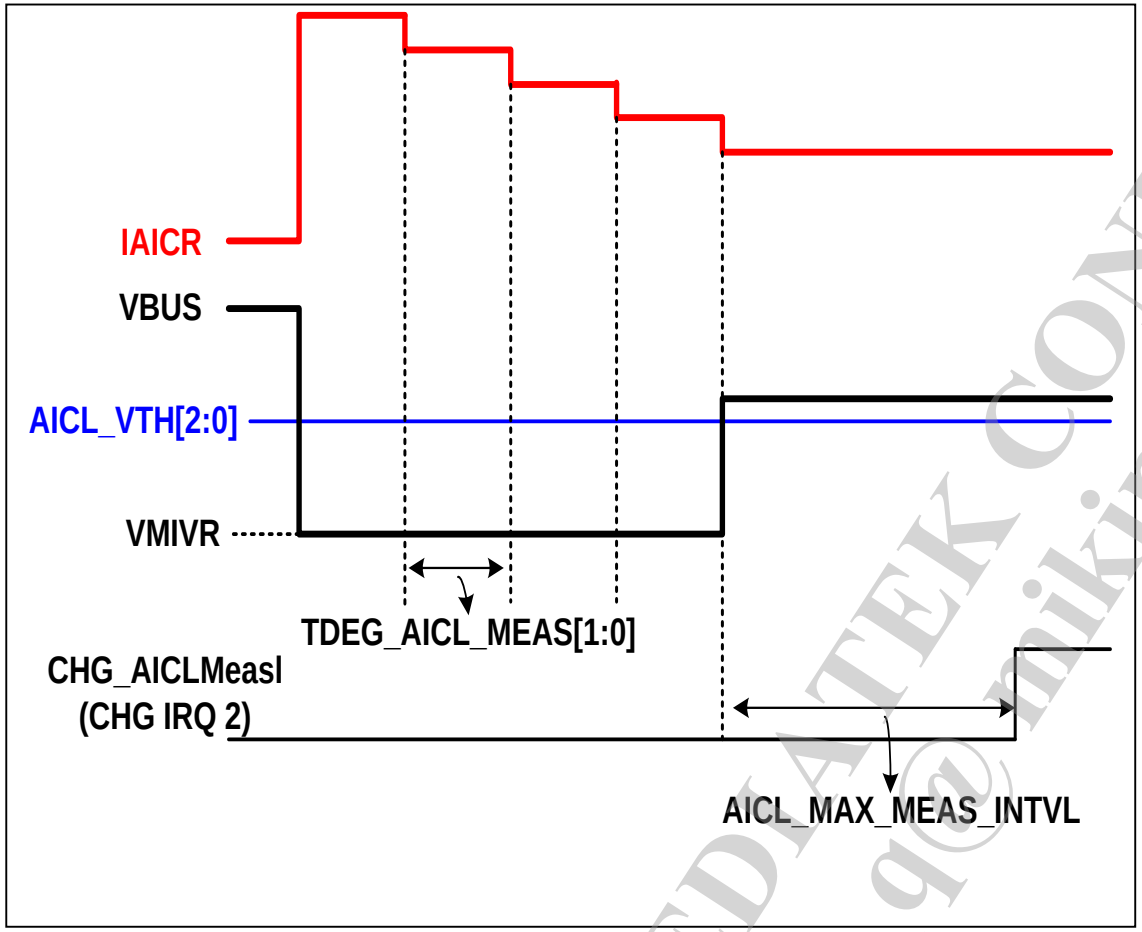
mt67XX.dts中charger 节点中修改充电电流

P = U↓ * I↑

防止电流抽太大，vbus
跌落发生plug out

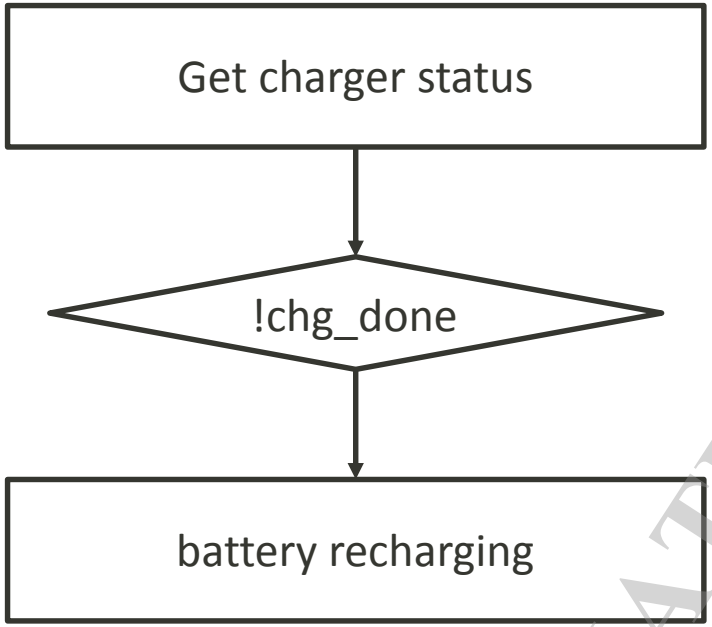
AICL(Average Input Current Level)

6360改名为AICC



- P = U↓ * I↑
- 充电器功率一定，增加充电电流，Vbus会减小。当Vbus小于MIVR电压时，导致cable out，停止充电。
- AICL的功能是防止充电电流过高时，Vbus电压drop的现象。
- AICL通过的一步一步的降低充电电流，进而提升Vbus的电压，直到Vbus电压高于AICL threshold电压为止（即当下可以输出最大电流的Vbus）
  - threshold一般为MIVR+0.2V，
  - 当Vbat过低时，平台抽压过大，此时threshold即为MIVR

CHR_BATFULL

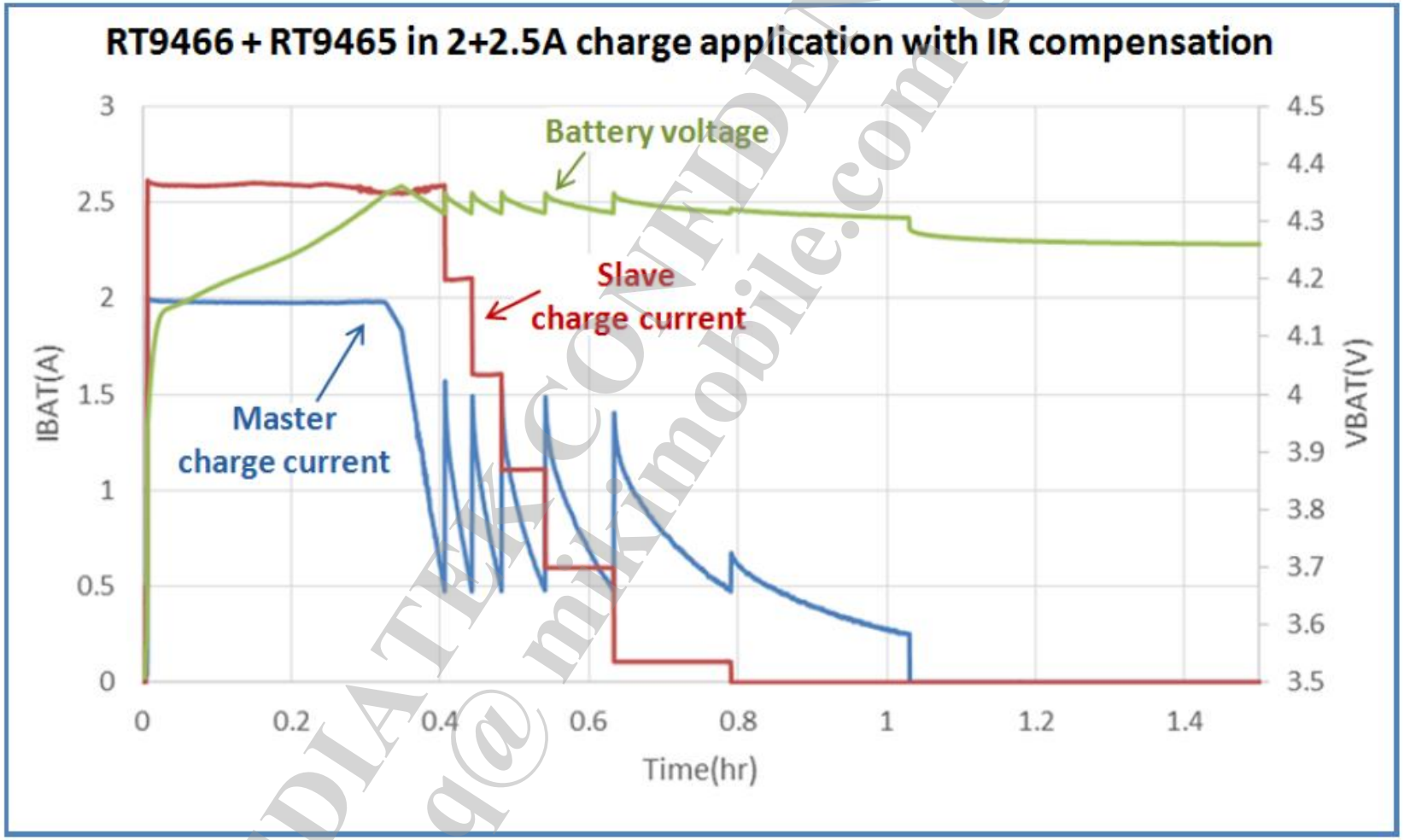


Once if VBAT is lower than VOREG – VREC (default = 100mV), the MT6360 will start to charging battery

Dual Charger: Master-Slave Operation

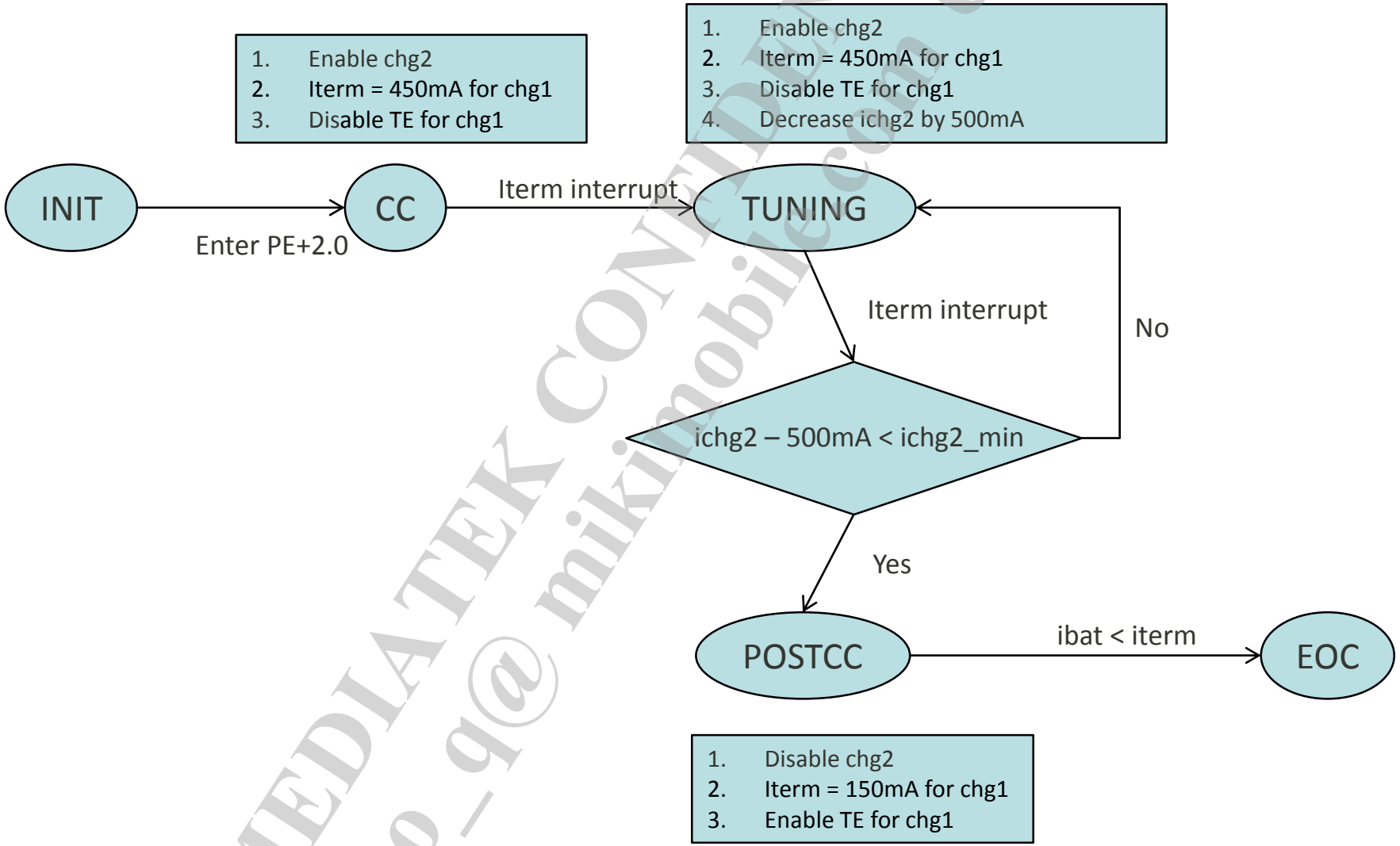
- Master Charger
  - detect pre-charge, CC mode, CV mode and end of charge termination
- Master in CC mode
  - set the slave in constant current mode
- Master in CV mode
  - Decrease the current of slave gradually
- Switch off slave when slave current is zero

Master-Slave Charging



CC TUNING POSTCC

Dual Charger State Transition Flow



Slave Control

- charger_dev_is_chip_enabled
- charger_dev_is_enabled
- charger_dev_enable_chip
- charger_dev_enable
- charger_dev_get_charging_current
- charger_dev_get_min_charging_current
- charger_dev_set_charging_current
- charger_dev_set_input_current
- charger_dev_set_constant_voltage