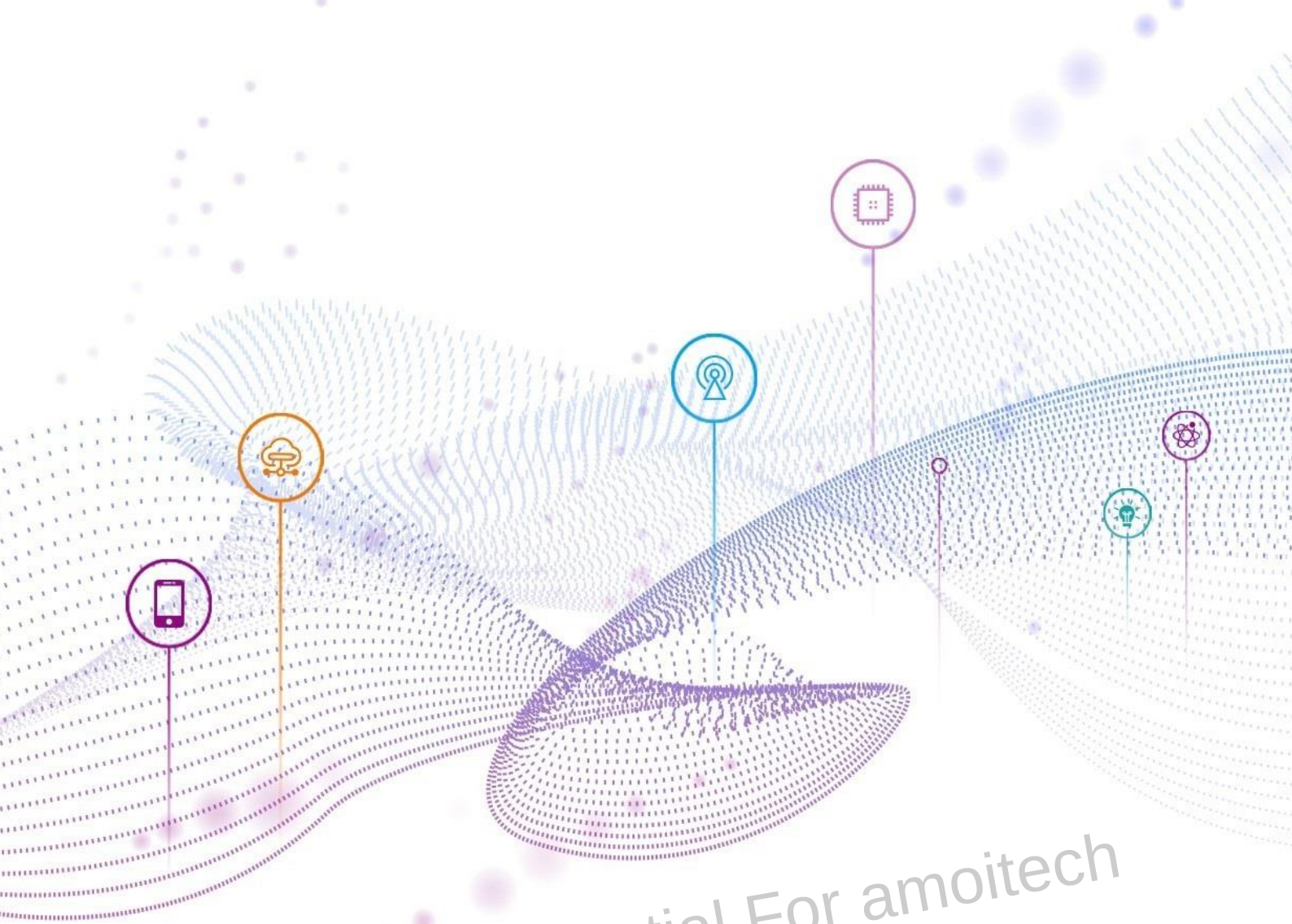


UIS8310 Charge 介绍

文档版本	V1.0
发布日期	2020-11-20

紫光展锐（上海）科技有限公司

紫光展锐（上海）科技有限公司

前 言

概述

本文档主要阐述了充电流程及用户配置相关的介绍，方便相关人员尽快熟悉该模块及使用相关功能。

读者对象

本文档主要适用于需要在 UIS8310 项目上了解 Charge 功能及其软件实现的人员。

缩略语

缩略语	英文全名	中文解释
ADC	Analog-to-Digital Converter	模拟数字转换器
HAL	Hardware Abstraction Layer	硬件抽象层
NV	Non-Volatile	非易失性

符号约定

在本文中可能出现下列标志，它所代表的含义如下。

符号	说明
说明	用于突出重要/关键信息、补充信息和小窍门等。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害。

变更信息

文档版本	发布日期	修改说明
V1.0	2020-11-20	第一次正式发布。

关键字

Charge、充电模块、充电管理、放电管理。

目 录

图目录

1 Charge 驱动

1.1 简介

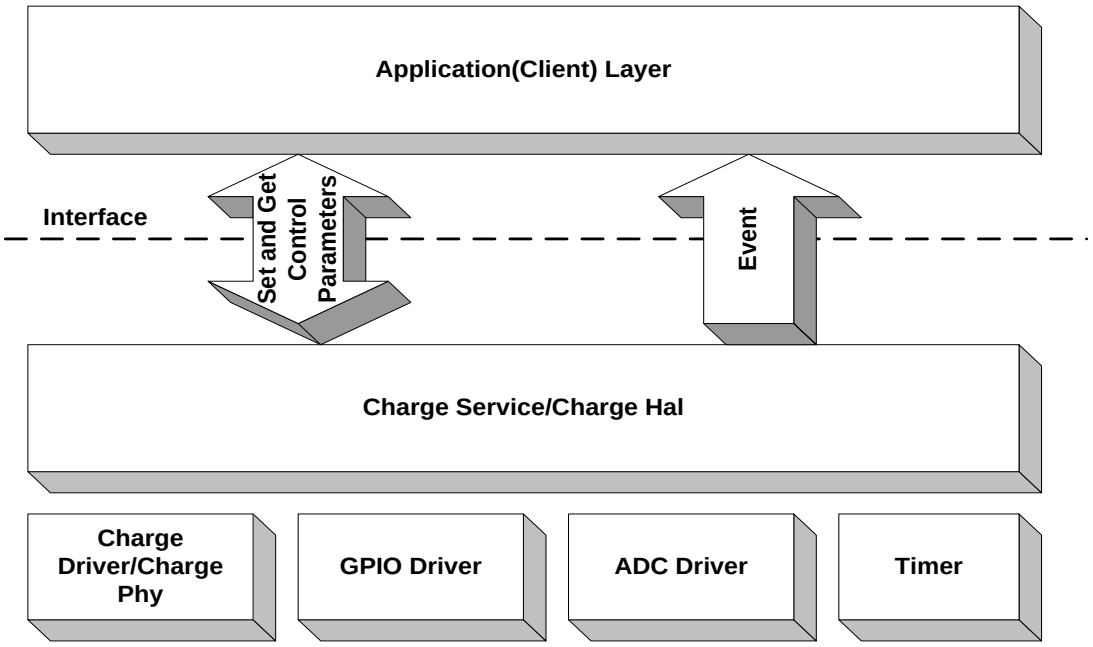
Charge 软件模块的任务便是对芯片充电管理模块的控制，实现充电过程的服务化，为上层提供控制和监视的接口，同时提供电量统计和获取功能，以便实现人机界面中的电量显示。

主要功能包括：

- 充电模块的可配置化。
- 为上层提供查询机制。
- 放电管理。
- 充电管理。
- 充电器插拔与类型检测。
- 充电器与电池的过压保护。
- 充电器与电池的过温保护。

1.2 Charge 模块与系统的关系

图1-1 系统结构框图



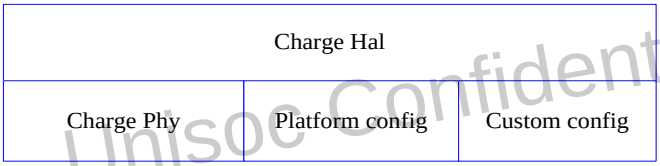
Charge Service 在系统中的位置如图 1-1 所示，它为上层应用程序提供了一种透明的服务机制，用户不需要关心 Charge 的过程，而只需要在适当的时候启动 Service，并接收它们所注册的事件（Event），同时它还支持函数查询功能。

Charge 依赖的模块主要有：

- Timer：利用周期性的 Timer 监控 Charge 各子模块的状态，有 3 个 Timer，分别监视电池电压、充电过程和充电器电压。
- GPIO：用于检测充电器是否插入及充电器类型。
- ADC（Analog-to-Digital Converter）：通过读取 Charger 相关 ADC 通道的值，知道各部分的状态，一共有 4 路 ADC。
  - Vbat_ADC：反映电池电压状态。
  - Vchg_ADC：反映充电器电压状态。
  - Battery Temp ADC：反映电池温度的状态。
  - Charger Temp ADC：反映充电器温度的状态。
- NV（Non-Volatile）：通过 NV 里的配置项来灵活控制 Charge 的各 Feature，做到可配置。

1.3 Charge 模块的层次结构

图1-2 Charge 模块层次结构



Charge 模块的层次结构如图 1-2，主要由 4 部分组成：

- Charge Hal（Hardware Abstraction Layer）层：Charge 模块的核心部分，实现 Charge 模块的管理以及为上层提供服务，是平台公用代码，与硬件无关，对应的文件为 charge.c。
- Charge Phy 层：平台相关代码，实现具体的芯片控制动作，对应的文件名为 chg_phy_v*.c。
- Platform config：平台配置层，如需要的 ADC 通道等。
- Custom config：用户配置层，根据客户的需求实现可配置化，相关配置在 NVItem 里。

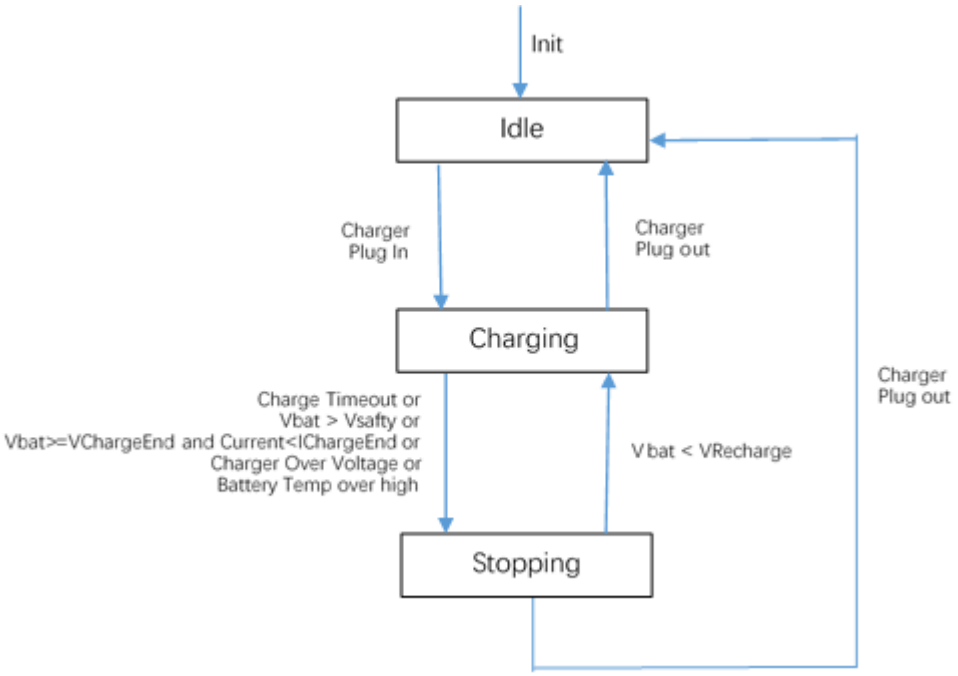
1.4 Charge Hal 层简介

Charge 模块的 Hal 层代码，主要有以下几个部分组成：

- 一个状态机。
- 一个 Task。
- 3 个 Timer。
- 一个用于统计电池电压的 Buffer。

1.4.1 Charge 模块的状态机

图1-3 Charge 模块状态机



Charge 模块的状态机如图 1-3 所示，Charge 模块初始化之后在任意时刻都会有个状态与之对应，Charge 模块的动作都是依据不同的状态去执行的，一些外部事件会促使状态的迁移。

Idle

Idle State：CHGMNG_IDLE，系统上电后，通过调用 CHGMNG_Init()，可使整个充电模块进入 Idle 状态，在此状态下，充电模块将等待接收充电器的插入中断，并使系统有机会进入充电(Charging)状态。

Charging

Charging State：CHGMNG_CHARGING，当系统接收到充电器插入的中断后，便会进入 Charging State。在此状态下，充电管理模块会启动芯片的充电模块，自动进行恒流和恒压充电，并通过一个周期性定时器不断检测和调整芯片的充电状态。在充电器连接的情况下，任何原因导致的充电停止，系统都会进入充电停止状态（Charge Stopping State）

说明

当充电器拔除时，此状态会切换至 Idle 状态。

Stopping

Charge Stopping State：CHGMNG_STOPPING，当充电管理模块发现充电结束的条件满足时（Charging State 中描述的三种条件），便会将状态切换至 Charging Stopping 状态。在该状态下，芯片充电模块将被关闭，但充电管理器仍然会周期性检测电池的电压状态。当电池电压低于用户设定的重新充电电压点或其他条件满足重新开始充电的条件时，充电状态便会重新跃迁至 Charging 状态，并开始新一轮的充电过程。

说明

当充电器拔除时，此状态会切换至 Idle 状态。

1.4.2 Charge 模块的 Task

_CHGMNG_SrvThread 是 Charge 模块的 Task，Charge 模块的绝大部分工作都有这个 Task 完成，它主要接收 4 种消息：

- CHGMNG_CHARGER_PLUG_IN_MSG：充电器插入时由 GPIO 的中断处理函数向 Task 发送此消息，当 Task 收到这个消息时会进行状态迁移，打开充电等操作。
- CHGMNG_CHARGER_PLUG_OUT_MSG：充电器拔出时由 GPIO 的中断处理函数向 Task 发送此消息，当 Task 收到这个消息时会进行状态迁移，关闭充电等操作。
- CHGMNG_CHARGER_MONITOR_MSG：当充电器处于连接状态（Charging State 或 Charge Stopping State）时，充电监控 Timer 会周期性（1s）的向 Task 发送此消息，Task 利用这个周期消息实现充电监控。
- CHGMNG_VBAT_MONITOR_MSG：电池监控 Timer（2.5±0.5s）会周期性的向 Task 发送此消息，Task 利用这个周期消息实现对电池的监控。

1.4.3 Charge 模块中的 Timer

为了实现充电与电池的管理 Charge 模块需要用到 3 个 Timer：

- 电池监控 Timer：_CHGMNG_VBatTimerHandler，实际上本身并不是 Timer 而是注册到 Doidle Timer 上的 Callback，周期由 Doidle Timer 决定，大概是 2.5s，系统初始化的时候注册，Timer 到期时就会向_CHGMNG_SrvThread 发送 CHGMNG_VBAT_MONITOR_MSG 消息，用于电池电压与电量的统计。
- 充电监控 Timer：g_charging_timer，timeout 函数为_CHGMNG_ChargeTimerHandler，周期为 1s 的 Timer，在系统初始化时注册，当充电插入时激活，充电器拔出时关闭，Timer 到期时会向 _CHGMNG_SrvThread 发送 CHGMNG_CHARGER_MONITOR_MSG 消息，用于充电控制。
- 充电器监控 Timer：g_ovp_timer，timeout 函数为_CHGMNG_OvpTimerHandler，周期为 100ms 的 Timer，在系统初始化时注册，当充电器插入时激活，充电器拔出时关闭，由于这个 Timer 过于频繁，所以它不会向_CHGMNG_SrvThread 发送消息，_CHGMNG_OvpTimerHandler 函数负责处理所有的事情，用于实现充电器的过压保护。

1.4.4 统计电池电压的 Buffer

vbat_queue 用于统计电池的电压，长度为 128，用这个 Buffer 的目的是统计一段时间电压的平均值，对电池电压进行平滑处理，防止电池电压的抖动对电池状态判断的干扰，系统对电池电压的判断用到的是这个平均值（module_state.bat_statistic_vol）。

vbat_queue 主要有两种应用场景：

- 放电：放电时 128 个成员全部用到，更新频率取决于_CHGMNG_VBatTimerHandler 周期，_CHGMNG_VBatTimerHandler 的周期大概是 3s，那么 vbat_queue 完全更新一遍的时间大概是 128*3s。由这里可以看出，如果用电源模拟电池的话，当调整电源电压时，程序需要大概 6.4min 才能反映出电压的变化（经常因为这个特性，测试组会误报一些 Bug）。
- 充电：充电时要求反映速度会比放电时快一些（判断是否充满以及是否过充等），所以 128 个成员不会全部用到，而是只用到了前 16 个，更新的频率由_CHGMNG_ChargeTimerHandler（1s）决定，那么完全更新一次 vbat_queue 之需要 16s。

1.5 Charge Hal 层模块功能介绍

1.5.1 Charge 模块的初始化

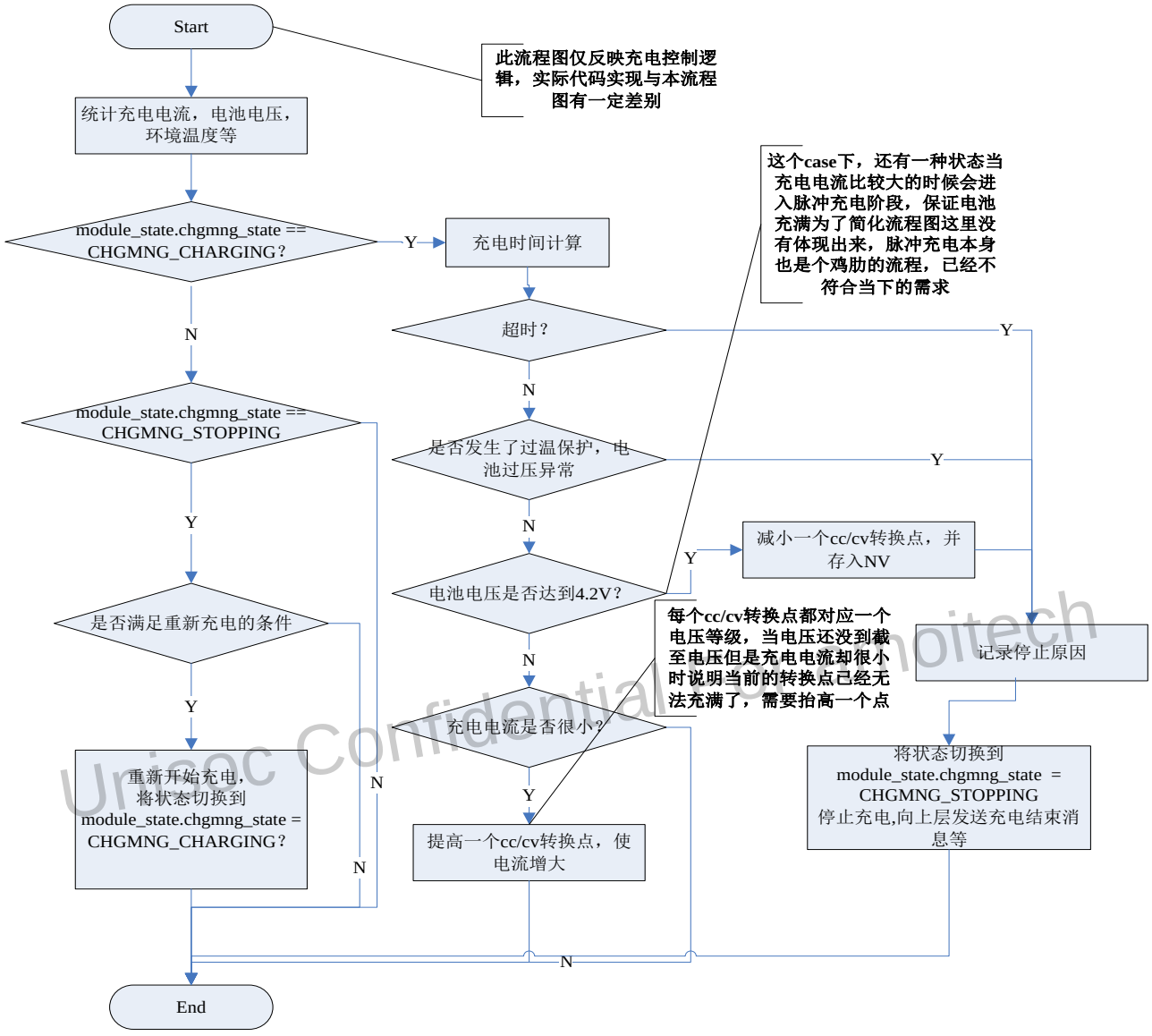
在手机启动阶段，首先会获取 NV 里面 Charge 相关的配置参数，并利用 Charge 的 API 设置到 Charge 模块中，然后调用 Charge 的初始化函数（CHGMNG_Init）对 Charge 模块进行初始化。CHGMNG_Init 函数主要完成 CHR_SERVICE 的注册，各 Timer 与 Task 的注册，vbat_queue 的初始化，Charge 硬件的相关初始化等。CHGMNG_Init 函数末尾，在充电器没有连接的情况下，会判断电池的电压是否小于关机电压，如果小于的话，会执行关机动作。

1.5.2 充电管理

充电器插入会触发 GPIO 中断，中断处理函数会向_CHGMNG_SrvThread 发送 CHGMNG_CHARGER_PLUG_IN_MSG 消息，_CHGMNG_SrvThread 会激活充电监控 Timer 与充电器监控 Timer，做一些充电状态的初始化，然后开始充电并向上层发送开始充电消息。_CHGMNG_SrvThread 会根据 CHGMNG_CHARGER_MONITOR_MSG 消息的触发周期调用_CHGMNG_ChargeMonitorRoutine() 函数实现充电控制。

_CHGMNG_ChargeMonitorRoutine()

图1-4 _CHGMNG_ChargeMonitorRoutine()流程图



_CHGMNG_ChargeMonitorRoutine()函数的流程图如图 1-4 所示，图 1-4 中所说的充电超时时间、充电截至电压等参数在 NV 里面可以配置。充电停止后，如果充电器未拔出，电池电压降到“复充电压值”以下时，会重新开始充电，而复充电压值也是可以在 NV 里面配置。图 1-4 中提到的脉冲充电和 CC/CV 转换点分别如图 1-5 和图 1-6 所示。

图1-5 脉冲充电

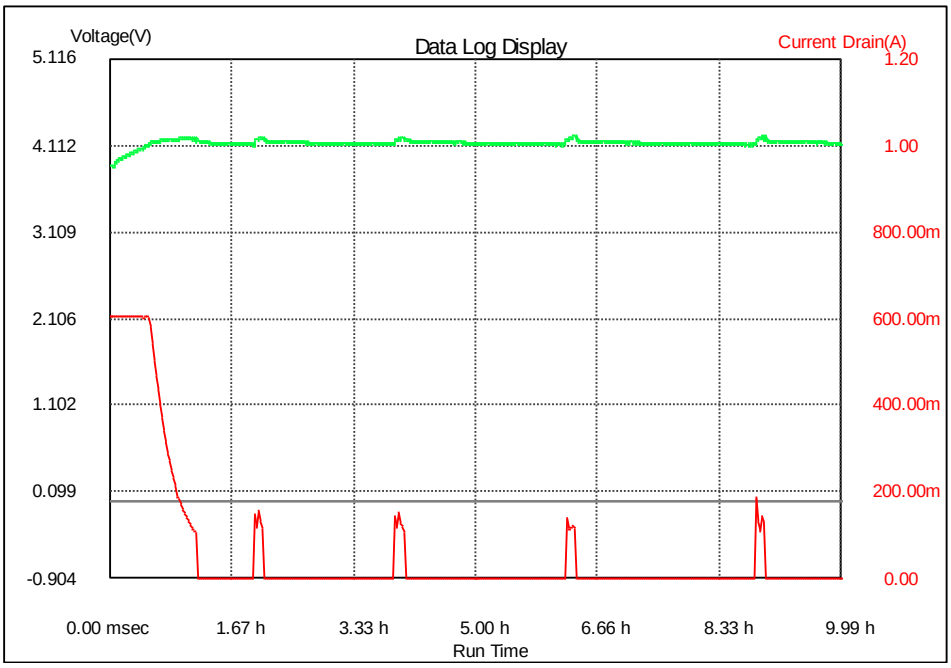
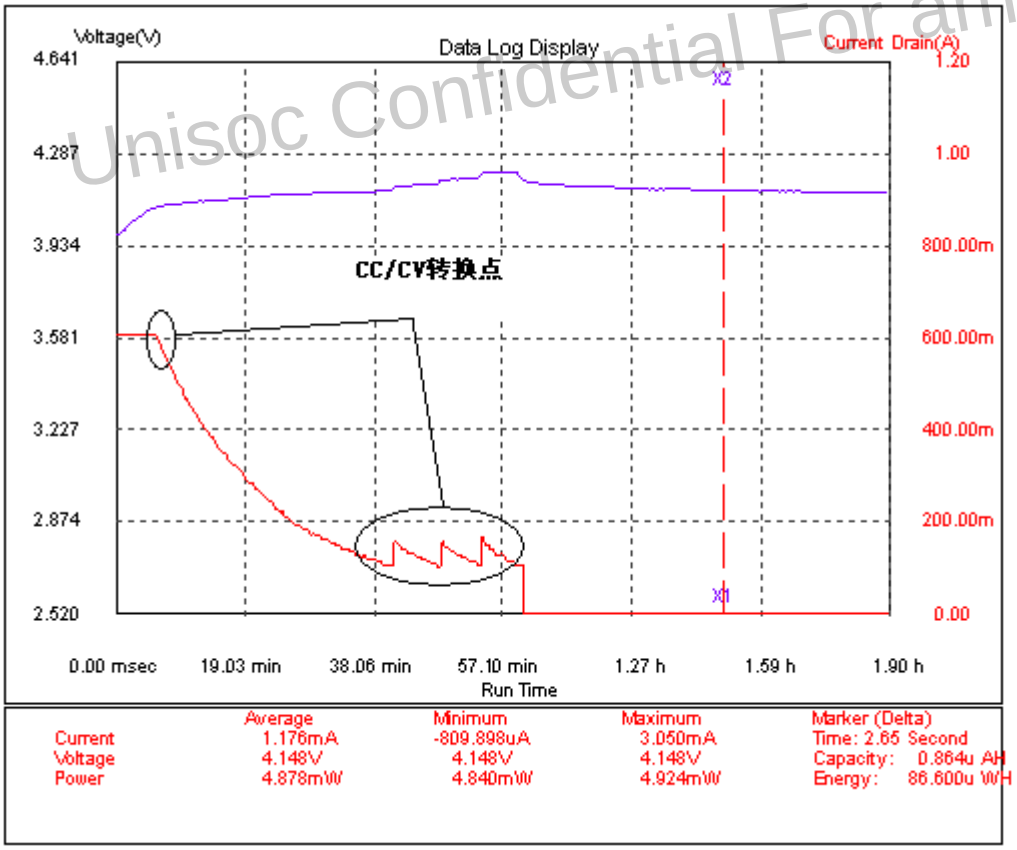


图1-6 CC/CV 转换点



充电器监控 Timer

这个 Timer 的处理对时间要求比较严格，所以没有放到_CHGMNG_ChargeMonitorRoutine()函数里，而是单独有个处理函数_CHGMNG_OvpTimerHandler()。

_CHGMNG_OvpTimerHandler()函数的功能为：

- 检测到充电器过压的次数超过十次且用时在 2s 之内，表示充电器的电压已经触发了过压保护（OVP，软件可配置，NV 里），伴随的动作为停止充电，向上层发送停止充电消息，将模块状态切换到 module_state.chgmng_state = CHGMNG_STOPPING，并将停止原因记录为充电器过压保护。
- 当过压保护产生以后，_CHGMNG_OvpTimerHandler()函数一旦重新检测到充电器电压已经回落到合理的范围内（resume 电压值软件可配置），就会重新开始充电，并且伴随状态与标志的切换。

充电器类型识别

当充电器插入的时候，GPIO 中断先将充电电流档位设置到一个比较低的档位上，因为这时还不知道是什么类型的充电器，USB 模块与 Charge 模块共用一个 GPIO 中断。充电器类型识别会用到 SC2702 CHGR_STATUS 寄存器[8:5]判断是 SDP、CDP 还是 DCP。依据判断出充电器类型，并重新设置充电电流的档位。

过温保护

Charge 模块内部实现了充电过温保护功能，因为这个功能不是所有的客户都会用到，所以是可选的，如果需要过温保护功能需要做到如下几点：

- #define CHGMNG_TEMP_MONITOR 宏要打开
- 在 NV 里面设置 OTP 的类型以及温度参数，所谓的类型就是温度取得的方式,如下：

```
typedef struct
{
    uint16 over_low;
    uint16 over_high;
    uint16 resume_low;
    uint16 resume_high;
}CHGMNG_OTP_PARAM_T;
```


关于脉冲充电

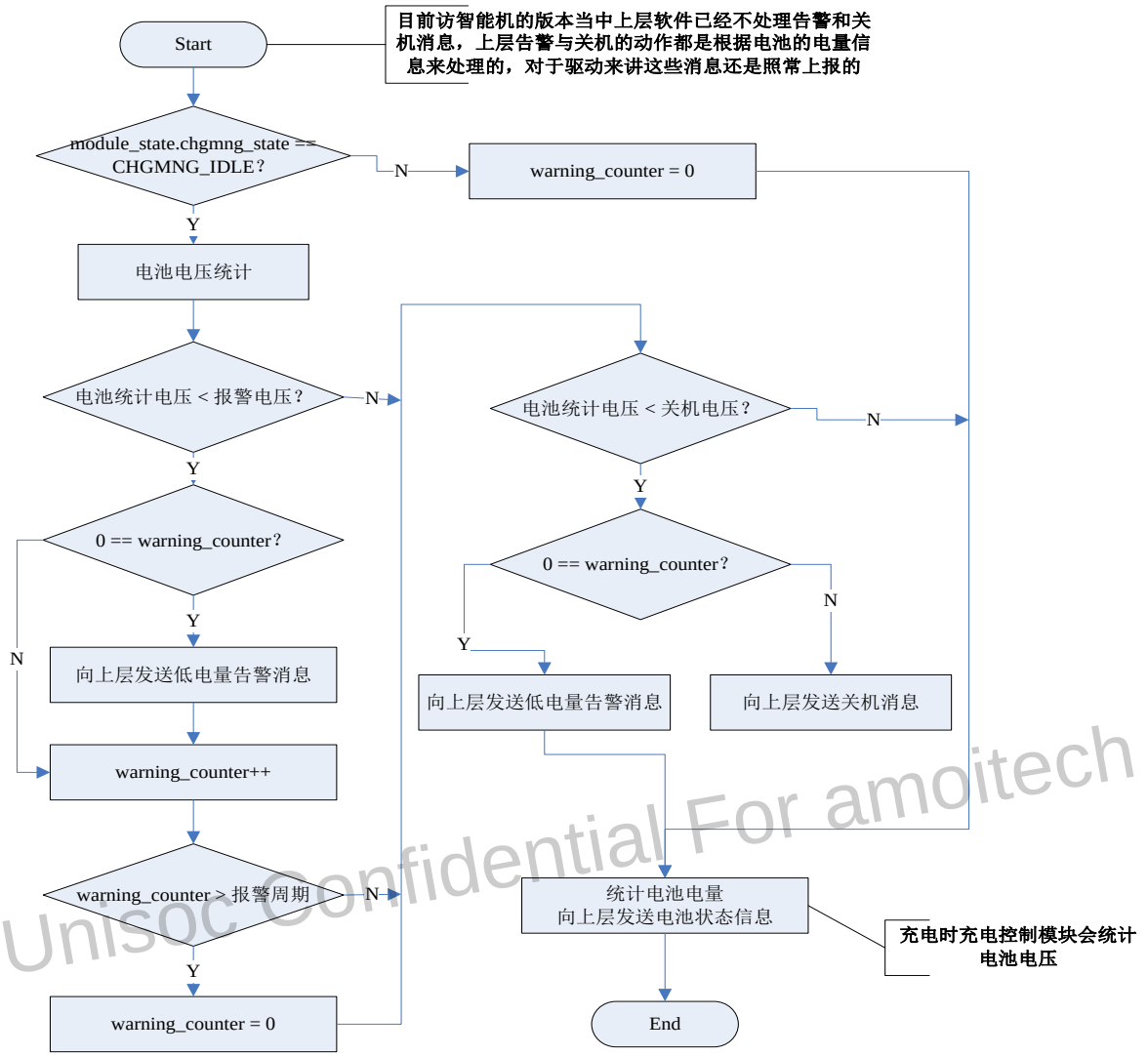
Charge 模块在充电过程中除了 CHGMNG_CHARGING 状态外，还有一个脉冲充电的状态 CHGMNG_PULSECHARGING，之所以没有反映到上文的状态机与流程图里，是因为在当下已经不大符合要求，是历史的产物，但是也有一定作用，大概的策略就是当充电电流比较大的时候，在充电结束阶段，为了使电池充满而进行的一段脉冲充电，充 3s，停 1s。

1.5.3 放电管理

_CHGMNG_SrvThread 会根据 CHGMNG_VBAT_MONITOR_MSG（由电池监控 Timer 发送）消息的触发，周期性调用_CHGMNG_VbatMonitorRoutine()函数实现放电管理。

_CHGMNG_VbatMonitorRoutine()函数在充电时仅进行统计电池电量，函数流程图如图 1-7 所示，图 1-7 中的关机电压与报警电压等参数，在 NV 里可以配置。

图1-7 _CHGMNG_VbatMonitorRoutine()函数流程图



1.5.4 电池电量统计

电池电量统计是根据电压换算出来的，能够给用户一定的提示，但是用电压是无法准确得到电池电量的，电压转换成电量是依据两张电量与电压的换算表完成的：

```
LOCAL uint16 dischg_bat_capacity_table[BAT_CAPACITY_STEP][2] =
{
    {4120,  100},
    {4060,  90},
    {3979,  80},
    {3900,  70},
    {3840,  60},
    {3800,  50},
    {3760,  40},
```



```
    {3730,  30},
    {3700,  20},
    {3650,  15},
    {3600,  5},
    {3501,  0},
};

LOCAL uint16 chg_bat_capacity_table[BAT_CAPACITY_STEP][2]=
{
    {4200,  100},
    {4180,  90},
    {4119,  80},
    {4080,  70},
    {4020,  60},
    {3970,  50},
    {3920,  40},
    {3880,  30},
    {3860,  20},
    {3830,  15},
    {3730,  5},
    {3251,  0},
};
```


这两张表在 NV 里面可以根据用户的需求定制，放电的时候用 dischg_bat_capacity_table 统计电池电量，充电的时候用 chg_bat_capacity_table 统计电池电量，表里面有 11 个区间，每个区间电量与电压都是线形的换算关系，这样就可以保证任何一个电池的电压通过这个表就可以得到一个百分比。

例如：放电时电池的电压是 x=3820mV（根据 dischg_bat_capacity_table 表可以查出在 50%~60%之间），那么可以推导出电池电量 y 为：

```
Y = （x - 3800）/（3840 - 3800）*（60 - 50）+ 50 = 55
```

说明

有时无法整除，所以在用电池电量作为判断条件时，千万不要写成 if（电池电量 == m）的语句，很可能 m 是得不到的。

电池电量统计函数为_CHGMNG_VoltagetoPercentum()，这个函数用来实现上面的算法，为了避免电池电量的抖动，有个限制条件：放电的时候电量只降不增，充电时电量只增不降。。

之所以用两个表统计而不用一个表，是因为电池有内阻。因为电池的内阻使我们很难通过 ADC 量取到电池电芯的电压，假设量取的电压为 V_1，电芯的电压为 V，电池的内阻为 R，电流为 I，就会有如下关系：

- 充电时：$V_1 = V + I*R$
- 放电时：$V_1 = V - I*R$

基于上面因素，chg_bat_capacity_table 表的各档位数值相应的会比 dischg_bat_capacity_table 表的数值高。

1.5.5 Charge 模块相关 ADC 校准

- Fuel-gage 的单点校准值存储在 Efuse Data 里面。
- 测量 VBAT 电池电压的 ADC 输出数字码，可以从寄存器 VOLT_VALUE[12:0]读出，理论上 ADC 的 1LSB 代表 6/2^13=1.467mV。实际 1LSB 的电压值是需要通过 ATE 校准来计算得到的，校准电压的计算公式为：

```
VBAT=VOLT_VALUE[12:0]/( Efuse_Block3[8:0]-256+2867)*4.2
```

说明

- 单位是 V。
- 公式的前提是 VOLT_VALUE 的最高位 VOLT_VALUE[12]是 0。
- 使用的理想值是 2867 而不是 6963，两者差 4096。

- 测量 VBAT 电池输出电流的 ADC 输出数字码可以从寄存器 CURT_VALUE[13:0]读出，理论上 ADC 的 1LSB 代表 3/2^13=3.66e-4A，校准电压的计算公式为：

```
I_SENSE= (CURT_VALUE[13:0]-8192)*2867/( Efuse_Block3[8:0]-256+2867)*3/2^13
```

说明

- 单位是 A。
- I_SENSE 的值可能为正值，也可能为负值。如果是正值，代表电池充电状态。如果是负值，代表电池放电状态。

1.5.6 为上层提供的服务

Charge 模块通过两种途径为上层提供服务：

- Server-Client 消息机制，每次电池监控 Timer 到期时，Charge 模块都会向 Client 发送 CHR_CAP_IND 消息，为上层提供查询的节拍，当有突发事件发生时 Charge 模块也会向上层发送相应的消息，如充电器插拔等，具体消息如下：

```
typedef enum
{
    // Charge message.
    CHR_CAP_IND = 0x1,          // Notify the battery's capacity

    CHR_CHARGE_START_IND,     // start the charge process.
    CHR_CHARGE_END_IND,       // the charge ended.

    CHR_WARNING_IND,            // the capacity is low, should charge.
    CHR_SHUTDOWN_IND,           // the capacity is very low and must shutdown.
    CHR_CHARGE_FINISH,          // the charge has been completed.
    CHR_CHARGE_DISCONNECT,    // the charge be disconnect
    CHR_CHARGE_FAULT,           // the charge fault, maybe the voltage of charge is too low.
    CHR_MSG_MAX_NUM
} CHR_SVR_MSG_SERVICE_E;
```

- Charge 模块提供一个为上层查询用的 API：CHGMNG_GetModuleState()，基本上所有需要的信息都可以通过这个 API 得到，仅通过一个 API 可以简化查询的复杂度，这个 API 会返回一个结构体的指针，信息如下：

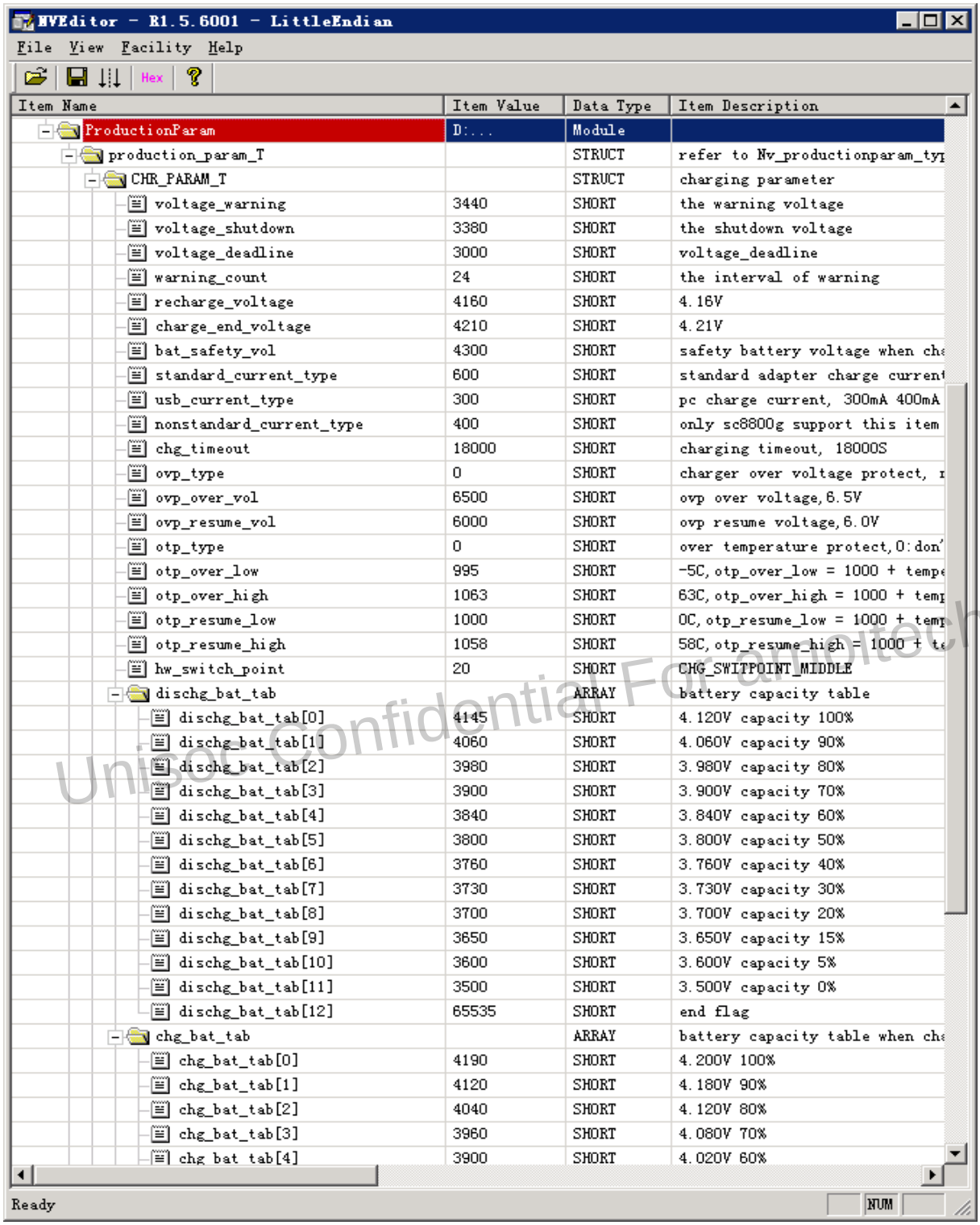

```
typedef struct
{
    CHGMNG_STATE_E      chgmng_state;           //charge module state
    uint32              bat_statistic_vol;      //statistic voltage,
    uint32              bat_cur_vol;            //current voltage, twinkling value
    uint32              bat_remain_cap;         //remain battery capacity
    uint32              charging_current;       //charging current value,reserved
    uint32              charging_temperature;   ///statistic vbat temperature.
    CHGMNG_ADAPTER_TYPE_E   adp_type;   //adapter type when chargeing,reserved
    CHGMNG_STOPREASON_E     charging_stop_reason;
    uint32              chg_vol;
}CHGMNG_STATE_INFO_T;
```

- chgmng_state：Charge 模块的状态
- bat_statistic_vol：电池电压的统计电压
- bat_cur_vol：最近一次采样的瞬时电压
- bat_remain_cap：电池的剩余电量
- charging_current：当充电时，记录充电电流
- charging_temperature：当充电时，如果过温保护功能打开，记录温度
- adp_type：充电器类型
- charging_stop_reason：如果充电停止了，记录停止原因
- chg_vol：充电器电压

1.5.7 用户可配置化

如图 1-8 所示，Charge 模块中用到的参数几乎都可以在 NV 里面配置，尽管默认值已经能够使 Charge 模块很好的工作，但是为了支持不同客户对 Charge 特性的需求，NVItem 为客户定制提供了手段，手机启动阶段在 Charge 模块初始化之前，已经将 NV 里面的设置项取出，并传给 Charge 模块，从这时起 Charge 模块的行为就会按照客户配置而定，下面简单介绍一下 Charge 相关的 NVItem。

图1-8 Charge 模块 NV 参数



放电相关配置参数

- voltage_warning：报警电压，当电池电压小于这个值时，Charge 模块就会周期性的向 Client 发送低电量告警消息 CHR_WARNING_IND。

- voltage_shutdown：关机电压，当电池电压小于这个值时 Charge 模块会向 Client 发送关机消息 CHR_SHUTDOWN_IND，另外一个用处是，作为开机门限，开机阶段如果充电器未连接，电池电压小于这个值的话，不允许开机。
- voltage_deadline：现在这个配置基本上已经废弃，它的目的是防止电池电压瞬间被拉低，触发硬件关机，软件先于硬件探测出来，请不要将这个值设置到 3V 以上。
- warning_count：控制低电压告警的周期，每一次触发报警电压，软件计数器就会加 1，当加得的值等于 warning_count 时就会发送 CHR_WARNING_IND 消息，并且计数器清 0，默认配置的报警间隔为 2.5*24s。

充电相关配置参数

- recharge_voltage：当电池充满以后，如果电池电压回落到这个值的时，会重新开始充电。
- charge_end_voltage：充电截至电压，充电满的必要条件就是电池的电压必须要高于这个值。
- bat_safety_vol：电池充电保护电压，充电时无论什么情况导致电池的电压大于这个值时，程序会强制关闭充电。
- standard_current_type：标准充电器的充电电流档位，请留意具体的芯片支持的电流档位，只有芯片支持的档位才会生效。
- usb_current_type：USB 充电电流档位为 300mA/400mA/500mA。
- nonstandard_current_type：非标准充电器的充电电流档位，不是所有的芯片都支持非标准充电器的识别，需要视芯片的情况而定。
- chg_timeout：充电超时时间。
- ovp_type：充电器过压保护的类型，目前诺基亚类型的基本上不用，一般情况下默认值即可。
- ovp_over_vol：充电器电压高于这个值时触发过压保护机制，停止充电。
- ovp_resume_vol：当充电器过压保护产生时，如果电压回落到这个值，会重新开始充电。
- otp_over_low：当温度低于这个值时，触发过温保护，停止充电。
- otp_over_high：当温度高于这个值时，触发过温保护，停止充电。
- otp_resume_low：当温度因为过低产生过温保护时，如果温度涨到这个值，重新开始充电。
- otp_resume_high：当温度因为过高产生过温保护时，如果温度回落到这个值，重新开始充电。
- hw_switch_point：初始的 CC/CV 转换点，这个成员比较特殊，当手机充完电以后，会将这个自适应得到的转换点，重写回这个成员处，以便被下次开机时优化充电曲线。

说明

温度相关的参数设置有个限制条件：因为 NV 里面无法存负数，所以记录的方式是 1000+要设置的温度，如-5℃记录为 995℃。

电池电量相关参数

- dischg_bat_tab：放电时电池电压与电量的对应表，用户可以根据这个表按照自己的要求定制，请配合 chg_bat_tab 使用。
- chg_bat_tab：充电时电池电压与电量的对应表，用户可以根据这个表按照自己的要求定制，请配合 dischg_bat_tab。

2 数据结构

本章主要介绍 Charge 相关的枚举类型及结构体。

2.1 宏定义

宏名称	含义
CHARGER_CURRENT_300MA	充电电流 300mA
CHARGER_CURRENT_350MA	充电电流 350mA
CHARGER_CURRENT_400MA	充电电流 400mA
CHARGER_CURRENT_450MA	充电电流 450mA
CHARGER_CURRENT_500MA	充电电流 500mA
CHARGER_CURRENT_550MA	充电电流 550mA
CHARGER_CURRENT_600MA	充电电流 600mA
CHARGER_CURRENT_650MA	充电电流 650mA
CHARGER_CURRENT_700MA	充电电流 700mA
CHARGER_CURRENT_750MA	充电电流 750mA
CHARGER_CURRENT_800MA	充电电流 800mA
CHARGER_CURRENT_900MA	充电电流 900mA
CHARGER_CURRENT_1000MA	充电电流 1000mA

2.2 枚举类型

2.2.1 CHGMNG_STATE_E

【功能】

充电管理模块在某一时刻所处的状态。

【定义】

```
typedef enum CHGMNG_StateCode_enum
{
    CHGMNG_IDLE = 0,
    CHGMNG_STARTING,
    CHGMNG_CHARGING,
    CHGMNG_PULSECHARGING,
    CHGMNG_STOPPING
} CHGMNG_STATE_E;
```

【参数说明】

参数名称	含义
CHGMNG_IDLE	充电器未连接
CHGMNG_STARTING	开始充电流程
CHGMNG_CHARGING	正在充电
CHGMNG_PULSECHARGING	为了使电池充的足够满，在充电快结束时的一段脉冲充电
CHGMNG_STOPPING	充电已停止但充电器未拔出

2.2.2 CHGMNG_STOPREASON_E

【功能】

充电停止的原因。

【定义】

```
typedef enum CHGMNG_StopReason_enum
{
    CHGMNG_INVALIDREASON = 0,
    CHGMNG_CHARGERUNPLUG = 1,
    CHGMNG_TIMEOUT,
    CHGMNG_VBATEND,
    CHGMNG_CHARGEDONE,
    CHGMNG_OVERVOLTAGE,
    CHGMNG_OVERTEMP
} CHGMNG_STOPREASON_E;
```


【参数说明】

参数名称	含义
CHGMNG_INVALIDREASON	初始化状态
CHGMNG_CHARGERUNPLUG	充电器拔出
CHGMNG_TIMEOUT	充电超时
CHGMNG_VBATEND	电池电压高于安全值
CHGMNG_CHARGEDONE	充电完成
CHGMNG_OVERVOLTAGE	充电器电压过高
CHGMNG_OVERTEMP	电池温度过高

2.2.3 CHGMNG_MSG_E

【功能】

Charge 任务可以接受并处理的消息。

【定义】

```
typedef enum {
    CHGMNG_VBAT_MONITOR_MSG = 1 ,
    CHGMNG_CHARGER_MONITOR_MSG,
    CHGMNG_CHARGER_PLUG_IN_MSG,
    CHGMNG_CHARGER_PLUG_OUT_MSG,
    CHGMNG_MODULE_RESET_MSG,
    CHGMNG_MAX_MSG
} CHGMNG_MSG_E;
```

【参数说明】

参数名称	含义
CHGMNG_VBAT_MONITOR_MSG	电池电压监控消息
CHGMNG_CHARGER_MONITOR_MSG	充电状态监控消息
CHGMNG_CHARGER_PLUG_IN_MSG	充电器插入消息
CHGMNG_CHARGER_PLUG_OUT_MSG	充电器拔出消息
CHGMNG_MODULE_RESET_MSG	充电软件模块复位消息
CHGMNG_MAX_MSG	Charge 任务最大消息数

2.2.4 CHR_SVR_MSG_SERVICE_E

【功能】

Charge 模块向上层应用发送的消息。

【定义】

```
typedef enum
{
    CHR_CAP_IND = 0x1,
    CHR_CHARGE_START_IND,
    CHR_CHARGE_END_IND,
    CHR_WARNING_IND,
    CHR_SHUTDOWN_IND,
    CHR_CHARGE_FINISH,
    CHR_CHARGE_DISCONNECT,
    CHR_CHARGE_FAULT,
    CHR_MSG_MAX_NUM
} CHR_SVR_MSG_SERVICE_E;
```

【参数说明】

参数名称	含义
CHR_CAP_IND	将电池容量通知给上层软件
CHR_CHARGE_START_IND	开始充电流程
CHR_WARNING_IND	低电告警消息
CHR_SHUTDOWN_IND	低电关机消息
CHR_CHARGE_FINISH	充电完成消息
CHR_CHARGE_DISCONNECT	充电器已断开
CHR_CHARGE_FAULT	充电出错，如充电器电压过高等
CHR_MSG_MAX_NUM	最大消息数量

2.2.5 CHGMNG_ADAPTER_TYPE_E

【功能】

充电适配器类型。

【定义】

```
typedef enum
{
    CHGMNG_ADP_UNKNOW,
    CHGMNG_ADP_STANDARD,
    CHGMNG_ADP_NONSTANDARD,
    CHGMNG_ADP_USB,
} CHGMNG_ADAPTER_TYPE_E;
```

【参数说明】

参数名称	含义
CHGMNG_ADP_UNKNOW	未知类型充电器
CHGMNG_ADP_STANDARD	标准充电器
CHGMNG_ADP_NONSTANDARD	非标充电器
CHGMNG_ADP_USB	USB 充电

2.3 结构体

2.3.1 CHGMNGSVR_SIG_T

【功能】

Charge 任务的消息结构。

【定义】

```
typedef struct CHGMNGSVR_SIG_tag
{
    SIGNAL_VARS
    uint32   sig_param;
}CHGMNGSVR_SIG_T;
```

【参数说明】

参数名称	含义
SIGNAL_VARS	Signal Header
sig_param	Signal code

2.3.2 CHGMNG_DISCHARGE_PARAM_T

【功能】

放电过程相关参数，可以在 NV 配置。

【定义】

```
typedef struct
{
    uint16   warning_vol;
    uint16  shutdown_vol;
    uint16   deadline_vol;
    uint16   warning_count;
}CHGMNG_DISCHARGE_PARAM_T;
```

【参数说明】

参数名称	含义
warning_vol	当电池电压低于这个值会发送警告信息
shutdown_vol	当电池电压低于这个值会发送关机提示信息
deadline_vol	当电池电压低于这个值时会关机，此值要低于 shutdown_vol
warning_count	低电量告警次数

2.3.3 CHGMNG_CHARGE_PARAM_T

【功能】

充电管理相关参数，可以在 NV 配置。

【定义】

```
typedef struct
{
    uint16 rechg_vol;
    uint16 chg_end_vol;
    uint16 bat_safety_vol;
    uint16 standard_chg_current;
    uint16 usb_chg_current;
    uint16 nonstandard_chg_current;
    uint16 chg_timeout;
}CHGMNG_CHARGE_PARAM_T;
```


【参数说明】

参数名称	含义
rechg_vol	充电结束后充电器未拔出时，重新启动充电的电压值
chg_end_vol	充电截止电压
bat_safety_vol	电池安全电压值
standard_chg_current	标准充电器的充电电流
usb_chg_current	USB 充电电流
nonstandard_chg_current	非标充电器的充电电流
chg_timeout	充电超时时间

2.3.4 CHGMNG_OVP_PARAM_T

【功能】

过压保护参数，可以在 NV 配置。

【定义】

```
typedef struct
{
    uint16 ovp_type;
    uint16 ovp_over_vol;
    uint16 ovp_resume_vol;
}CHGMNG_OVP_PARAM_T;
```


【参数说明】

参数名称	含义
ovp_type	过压保护类型
ovp_over_vol	过压保护电压
ovp_resume_vol	恢复电压

2.3.5 CHGMNG_OTP_PARAM_T

【功能】

过温保护参数，可以在 NV 配置。

【定义】

```
typedef struct
{
    uint16 otp_type;
    uint16 over_low;
    uint16 over_high;
    uint16 resume_low;
    uint16 resume_high;
}CHGMNG_OTP_PARAM_T;
```

【参数说明】

参数名称	含义
otp_type	• 0：不支持 • 1：环境温度 • 2：电池温度
over_low	过低温度
over_high	过高温度
resume_low	从低温保护恢复的温度值
resume_high	从高温保护恢复的温度值

2.3.6 CHGMNG_STATE_INFO_T

【功能】

记录充电模块消息。

【定义】

```
typedef struct
{
    CHGMNG_STATE_E              chgmng_state;
    uint32                      bat_statistic_vol;
    uint32                      bat_cur_vol;
    uint32                      bat_remain_cap;
    uint32                      charging_current;
     uint32                      charging_temperature;
    CHGMNG_ADAPTER_TYPE_E   adp_type;
    CHGMNG_STOPREASON_E       charging_stop_reason;
}CHGMNG_STATE_INFO_T;
```

【参数说明】

参数名称	含义
chgmng_state	充电流程中当前所处的状态
bat_statistic_vol	进行统计平均后的电池电压值
bat_cur_vol	当前电池电压值，瞬时值
bat_remain_cap	剩余电池容量
charging_current	充电电流值
charging_temperature	电池温度
adp_type	充电适配器类型
charging_stop_reason	充电停止的原因

3 接口函数

本章主要介绍常用的一些接口函数及其功能

3.1 CHGMNG_GetModuleState

【函数功能】

获取充电模块的信息。

【函数原型】

```
PUBLIC CHGMNG_STATE_INFO_T* CHGMNG_GetModuleState(void);
```

【参数说明】

无。

【返回值】

指向充电模块信息的指针。

3.2 CHGMNG_ChargerPlugInHandler

【函数功能】

充电器插入或拔出时会调用此函数。

【函数原型】

```
PUBLIC void CHGMNG_ChargerPlugInHandler (uint32 gpio_id, uint32 gpio_state);
```

【参数说明】

参数名称	含义
gpio_id	GPIO ID
gpio_state	GPIO 状态

【返回值】

无。

3.3 CHGMNG_GetVBATADCValue

【函数功能】

该函数用于获取电池电压的 ADC 值。

【函数原型】

```
PUBLIC uint32 CHGMNG_GetVBATADCValue (void);
```

【参数说明】

无。

【返回值】

返回电池电压的 ADC 值。

3.4 CHGMNG_VoltageToAdcvalue

【函数功能】

该函数将电压转换为 ADC 值。

【函数原型】

```
PUBLIC uint16 CHGMNG_VoltageToAdcvalue (uint16 votage);
```

【参数说明】

参数名称	含义
votage	电压

【返回值】

返回 ADC 值。

3.5 CHGMNG_AdcvalueToVoltage

【函数功能】

该函数将 ADC 值转换为电压值。

【函数原型】

```
PUBLIC uint16 CHGMNG_AdcvalueToVoltage (uint32 adcvalue);
```

【参数说明】

参数名称	含义
adcvalue	ADC 值

【返回值】

返回电压值。

3.6 CHGMNG_IsChargeConnect

【函数功能】

检查充电器是否连接。

【函数原型】

```
BOOLEAN CHGMNG_IsChargeConnect (void);
```

【参数说明】

无。

【返回值】

- TRUE：已连接。
- FALSE：未连接。

