## 摘 要

本文档旨在指导帮助用户基于芯海科技 CSU3AF10 MCU 设计的移动电源如何开发、设计、调试 C-C 和 C-L 带载休眠唤醒功能，加入此功能后可以大大降低移动电源在充饱手机等设备后的工作电流，从而延长移动电源的使用时间和寿命。

本文提到的术语说明如下：

USB TypeC：USB-IF 协会推出的 USB TypeC 标准。

C-C：指安卓手机使用的 USB TypeC to USB TypeC 线缆。

C-L：指苹果手机专用的 USB TypeC to Lighting 线缆。

C-C 带载：指连接着手机的 C-C 线缆连接到移动电源的 USB TypeC 接口。

C-L 带载：指 C-L 线缆连接到移动电源的 USB TypeC 接口。

C-C 退载：指 C-C 线缆从移动电源的 USB TypeC 接口上拔出或手机等设备从 C-C 线缆的 USB TypeC 端口上拔出。

C-L 退载：指 C-L 线缆从移动电源的 USB TypeC 接口上拔出，需要注意手机等设备从 C-L 线缆的 Lighting 端口拔出不算作 C-L 退载。

## 适用范围

| 类型 | 适用产品型号或系列 | 说明 |
|---|---|---|
| MCU | CSU3AF10 系列 MCU | |

## 版 本

| 历史版本 | 修改内容 | 日期 |
|---|---|---|
| V1.0 | 初版生成 | 2023-05-05 |

# 目 录

**概述.....4**
**1 C-C & C-L 带载休眠唤醒功能的实现方案说明.....5**
1.1 轻载检测功能模块.....5
1.2 休眠模式功能模块.....5
1.2.1 休眠模式 1（Sleep_Mode_1）的基本配置要求.....5
1.2.2 休眠模式 2（Sleep_Mode_2）的基本配置要求.....5
1.3 带载检测功能模块.....6
1.3.1 C-C 带载休眠的配置要求.....6
1.3.2 C-L 带载休眠的配置需求.....6
1.4 退载检测功能模块.....6
1.4.1 C-C 退载检测功能模块.....7
1.4.2 C-L 退载检测功能模块.....7
1.5 手机插入识别唤醒功能模块.....7
1.5.1 C-C 线缆手机插入识测唤醒功能模块.....7
1.5.2 C-L 线缆手机插入识别唤醒功能模块.....7
**2 C-C & C-L 带载休眠唤醒功能的硬件设计说明.....8**
2.1 CSU3AF10 移动电源参考方案电路图设计说明.....8
2.2 C-L 线手机插入识别唤醒功能的电路设计说明.....9
**3 C-C & C-L 带载休眠唤醒功能的程序流程图和程序设计说明.....10**
3.1 C-C & C-L 带载休眠唤醒功能的程序流程图设计说明.....10
3.1.1 C-C & C-L 带载休眠唤醒功能的程序流程图.....10
3.1.2 C-C & C-L 带载休眠唤醒功能的程序流程图的流程步骤.....12
3.2 C-C & C-L 带载休眠唤醒功能的程序设计说明.....13
3.2.1 轻载检测功能模块的程序设计说明.....13
3.2.2 休眠前功能配置的程序设计说明.....14
3.2.3 带载休眠功能配置的程序设计说明.....16
3.2.4 退载后休眠功能配置的程序设计说明.....17
3.2.5 检测 CC 引脚电平状态的功能程序设计说明.....18
3.2.6 带载检测功能程序设计说明.....19
3.2.7 退载检测功能程序设计说明.....20
3.2.8 手机插入识别唤醒功能模块.....21
**4 C-C & C-L 带载休眠功能的开发设计注意事项.....23**
4.1 休眠前打开 CC 引脚下拉电阻方法.....23
4.1.1 打开 TypecA 模块的 CC 引脚下拉电阻.....23
4.1.2 打开 TypecB 模块的 CC 引脚下拉电阻.....24
4.1.3 打开 TypecA & TypecB 模块的 CC 引脚下拉电阻.....25
4.2 唤醒定时器配置要求.....26
4.3 唤醒后延时 ADC.....26
4.4 检测区别 C-C 带载 or C-L 带载.....27

**5 C-C & C-L 带载休眠唤醒功能方案的测试结果.....28**
5.1 C-C 带载休眠功能测试.....29
5.2 C-L 带载休眠功能测试.....30
5.3 C-C 手机插入识别唤醒.....31
5.4 C-L 手机插入识别唤醒.....32
5.5 C-C & C-L 带载休眠系统功耗测试结果.....32

# 概述

本说明文档是基于芯海科技 CSU3AF10 MCU 设计的移动电源方案如何开发、设计、调试 C-C 和 C-L 带载休眠唤醒功能的设计指南，此文档将会详细介绍说明 C-C 和 C-L 带载休眠唤醒功能的实现方案、硬件设计方法、程序设计方法、测试结果演示和设计过程中需要注意的事项。

# 1 C-C & C-L 带载休眠唤醒功能的实现方案说明

为了实现 CSU3AF10 移动电源方案支持 C-C & C-L 带载休眠唤醒功能，需要在芯海科技 CSU3AF10 移动电源参考方案中增加开发设计以下功能模块：轻载检测功能模块、休眠模式功能模块、带载检测功能模块、退载检测功能模块、手机插入识别唤醒功能模块。

## 1.1 轻载检测功能模块

为了实现 C-C & C-L 带载休眠唤醒功能，最基本的前提必须设计轻载检测功能模块，为了不影响正常带载的充放电功能，必须合理设计轻载检测功能模块。

轻载检测功能的设计方法建议如下：用户可以通过检测判断 VBUS 的总电流小于用户设定的轻载电流值并且持续一段时间后进行判定为轻载模式。例如检测判断 VBUS 电流小于 200mA 连续 10S 钟后进入轻载模式。

## 1.2 休眠模式功能模块

根据芯海科技 CSU3AF10 MCU 的特性和 C-C & C-L 带载休眠的功能要求，如果需要实现 C-C & C-L 带载休眠唤醒功能，需求设计两种休眠模式，其用途和基本配置要求分别如下：

休眠模式 1（Sleep_Mode_1）：用于 C-C 和 C-L 退载后进入的休眠模式；

休眠模式 2（Sleep_Mode_2）：用于 C-C 和 C-L 带载时进入的休眠模式；

两种休眠模式的基本配置要求请参考此文档的 1.2.1 和 1.2.2 章节描述。

### 1.2.1 休眠模式 1（Sleep_Mode_1）的基本配置要求

在此休眠模式下，需求使能 USB TypeC 自动扫描功能（DRP），用于识别检测 USB TypeC 接口的设备插入。当检测到设备插入后，产生中断唤醒系统退出休眠模式，进入正常工作模式。

在此休眠模式下，需求使能按键中断功能，用于检测按键。当检测到按键按下后，产生中断唤醒系统退出休眠模式，进入正常工作模式。

在此休眠模式下，需求使能 DC-DC IC 中断功能，用于检测 DC-DC IC 的中断事件。当检测到 DC-DC IC 的中断事件后，唤醒系统退出休眠模式，进入正常工作模式。

### 1.2.2 休眠模式 2（Sleep_Mode_2）的基本配置要求

在此休眠模式下，需求使能 CC 引脚的下拉电阻（5.1K）功能，用于保证 C-C 或 C-L 正常带载，同时保证不会额外增加休眠功耗。

在此休眠模式下，需求使能定时器中断功能，用于间隔唤醒系统进行检测 C-C 或 C-L 是否退载，间隔唤醒时间建议设定为 500ms。

### 1.3 带载检测功能模块

因为 C-C 和 C-L 带载休眠的特性不太一样，为了保证 C-C 和 C -L 带载休眠功能正常，需求正确检测区别出是 C-C 带载还是 C-L 带载。

检测判断的方法如下：休眠前使能 CC 引脚的下拉电阻（5.1K）功能和使能定时器中断功能，定时器中断唤醒系统后检测 CC 电平状态和 VBUS 电平状态，如果检测到 CC 电平状态大于用户设定的电压值（例如 0.2V）或 VBUS 是有电压的状态，则判定为 C-C 带载，否则，判定为 C-L 带载。

由于不同型号的手机的特性不太一样，为了更加准确区别出是 C-C 带载还是 C-L 带载，建议多做几次（例如 3 次）CC 电平状态和 VBUS 电平状态的检测后再作最终决定。

当检测区别出是 C-C 带载还是 C-L 带载后，必须根据 C-C 带载休眠和 C-L 带载休眠的特性和要求进行配置系统的各个模块的功能，系统配置要求请参考此文档的 1.3.1 和 1.3.2 章节描述。

#### 1.3.1 C-C 带载休眠的配置要求

如果检测到是 C-C 带载休眠，只需要按照此文档的 1.2.2 章节的说明进行配置系统的各个模块的功能即可。

#### 1.3.2 C-L 带载休眠的配置需求

如果检测到是 C-L 带载休眠，除了按照此文档的 1.2.2 章节的说明进行配置系统的各个模块的功能外，还需要对系统的其他功能模块进行如下配置：

需求将 DM 引脚配置为 GPIO 功能，为了正确打开 DM 引脚对应的外部中断功能；

需求使能 DM 引脚的外部中断功能，用于 C-L 线缆的手机的插入识别检测；

MCU 休眠时，使能所有 CC 引脚的下拉电阻（5.1K）功能，从而降低休眠功耗；

MCU 唤醒后，使能所有 CC 引脚的上拉电流源（如 80uA）功能，用于检测 C-L 是否退载。

### 1.4 退载检测功能模块

因为 C-C 和 C-L 带载休眠的特性不太一样，所以 C-C 和 C -L 退载检测的方法也不相同，具体检测方法请参考 1.4.1 和 1.4.2 章节描述。

### 1.4.1 C-C 退载检测功能模块

检测判断 C-C 退载的方法如下：定时器中断唤醒 MCU 后，使能所有 CC 引脚的下拉电阻（5.1K）功能，使能 ADC 进行检测 CC 电平状态，如果检测到所有 CC pin（CC1-A、CC2-A、CC1-B、CC2-B）的电平状态均小于用户设定的电压值（例如 0.2V）时，则判定为 C-C 退载。

当检测到 C-C 退载后，退出休眠模式 2，根据休眠模式 1 的配置要求配置后进入休眠模式 1。

### 1.4.2 C-L 退载检测功能模块

检测判断 C-L 退载的方法如下：定时器中断唤醒 MCU 后，使能所有 CC 引脚的上拉电流源（例如 80uA）功能，使能 ADC 进行检测 CC 电平状态，如果检测到所有 CC pin（CC1-A、CC2-A、CC1-B、CC2-B）的电压状态均大于用户设定的电压值（例如 2.4V）时，则判定为 C-L 退载。

当检测到 C-L 退载后，退出休眠模式 2，根据休眠模式 1 的配置要求配置后进入休眠模式 1。

## 1.5 手机插入识别唤醒功能模块

因为 C-C 和 C-L 带载休眠的特性不太一样，所以 C-C 和 C -L 带载休眠模式下手机插入识别唤醒的方法不相同，具体检测方法请参考 1.5.1 和 1.5.2 章节描述。

### 1.5.1 C-C 线缆手机插入识测唤醒功能模块

在 C-C 退载后，系统进入休眠模式 1，为了能够正确实现手机插入识别唤醒功能，需要使能 USB TypeC 自动扫描功能（DRP）和打开自动扫描（DRP）中断功能，当检测识别到 C-C 线缆的手机等设备插入后，产生 USB TypeC 的 DRP 中断，此中断便可以唤醒系统退出休眠模式进入正常工作模式。

### 1.5.2 C-L 线缆手机插入识别唤醒功能模块

为了能够正确实现 C-L 线缆的手机插入识别唤醒功能，需要将 DM 引脚配置为 GPIO 功能，并使能 DM 引脚的外部中断功能，当检测识别到 C-L 线缆的手机等设备插入后，产生外部中断，此中断便可以唤醒系统退出休眠模式进入正常工作模式。

# 2 C-C & C-L 带载休眠唤醒功能的硬件设计说明

## 2.1 CSU3AF10 移动电源参考方案电路图设计说明

以下为基于芯海科技 CSU3AF10 MCU 设计的移动电源方案参考电路图：CSU3AF10_EVB V1.1.pdf。

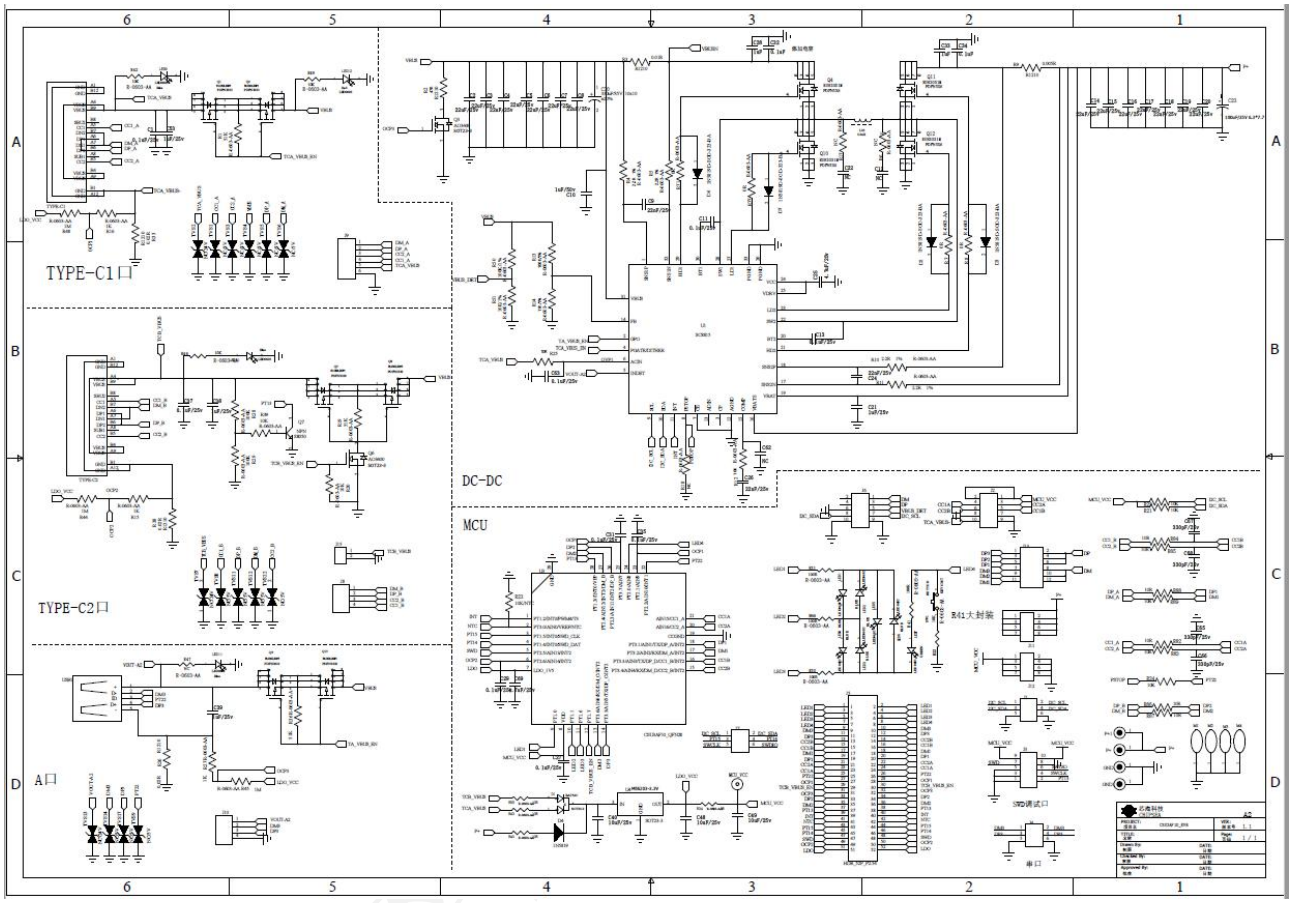



图 1 CSU3AF10 EVB V1.1 电路图

## 2.2 C-L 线手机插入识别唤醒功能的电路设计说明

在 C-L 带载休眠时，为了能够正确实现手机插入识别唤醒功能，需求在“CSU3AF10 移动电源方案参考电路图”基础上进行如下修改，将 DM 引脚通过 1M 电阻上拉至某只 GPIO，参考设计电路如下：

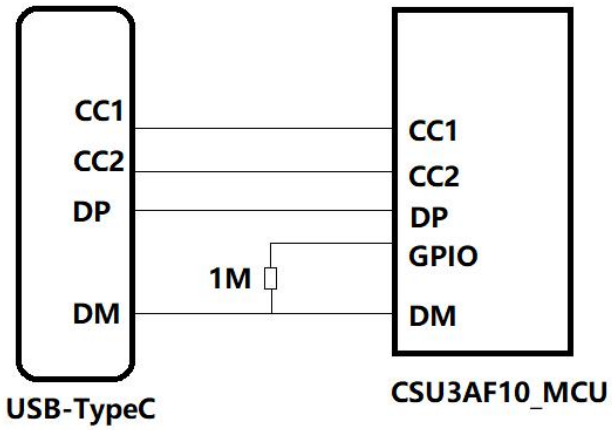



**图 2 C-L 线手机插入识别唤醒参考设计电路图**

C-L 线缆手机插入识别唤醒功能原理如下：

将 GPIO 引脚设置输出高电平状态；

将 DM 引脚设置为 GPIO 功能，且将 DM 引脚设置为 floating 状态；

使能 DM 引脚对应的外部中断功能，并设置下降沿唤醒功能；

在 C-L 的 Lighting 端没有手机插入时，DM 引脚将会被 GPIO 拉至高电平状态；

当 C-L 的 Lighting 端有手机插入时，DM 引脚将会产生下降沿信号，此信号会中断唤醒系统，从而可以实现 C-L 线缆手机插入识别唤醒功能；

在检测到手机后，将 GPIO 引脚设置为 floating 状态，将 DM 引脚设置为 DM 功能，进行多协议握手沟通。

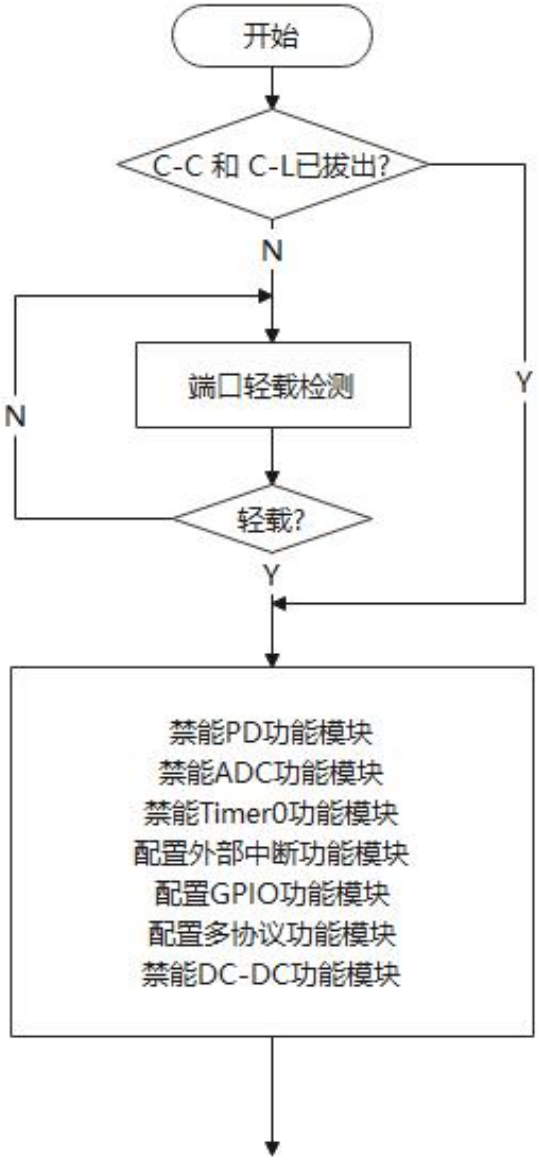
# 3 C-C & C-L 带载休眠唤醒功能的程序流程图和程序设计说明

如果用户想在基于芯海科技 CSU3AF10 MCU 设计的移动电源方案增加 C-C & C-L 带载休眠唤醒功能，用户可以在芯海科技提供的 CSU3AF10 移动电源方案基础上参考以下程序流程图说明和程序设计说明进行开发设计此功能。

## 3.1 C-C & C-L 带载休眠唤醒功能的程序流程图设计说明

C-C & C-L 带载休眠唤醒功能的程序流程图主要说明了 C-C 和 C-L 带载如何进入休眠和 C-C & C-L 退载如何转换休眠模式，以及手机等设备插入识别唤醒等功能处理。

### 3.1.1 C-C & C-L 带载休眠唤醒功能的程序流程图




此箭头连接至下页流程图开始处箭头

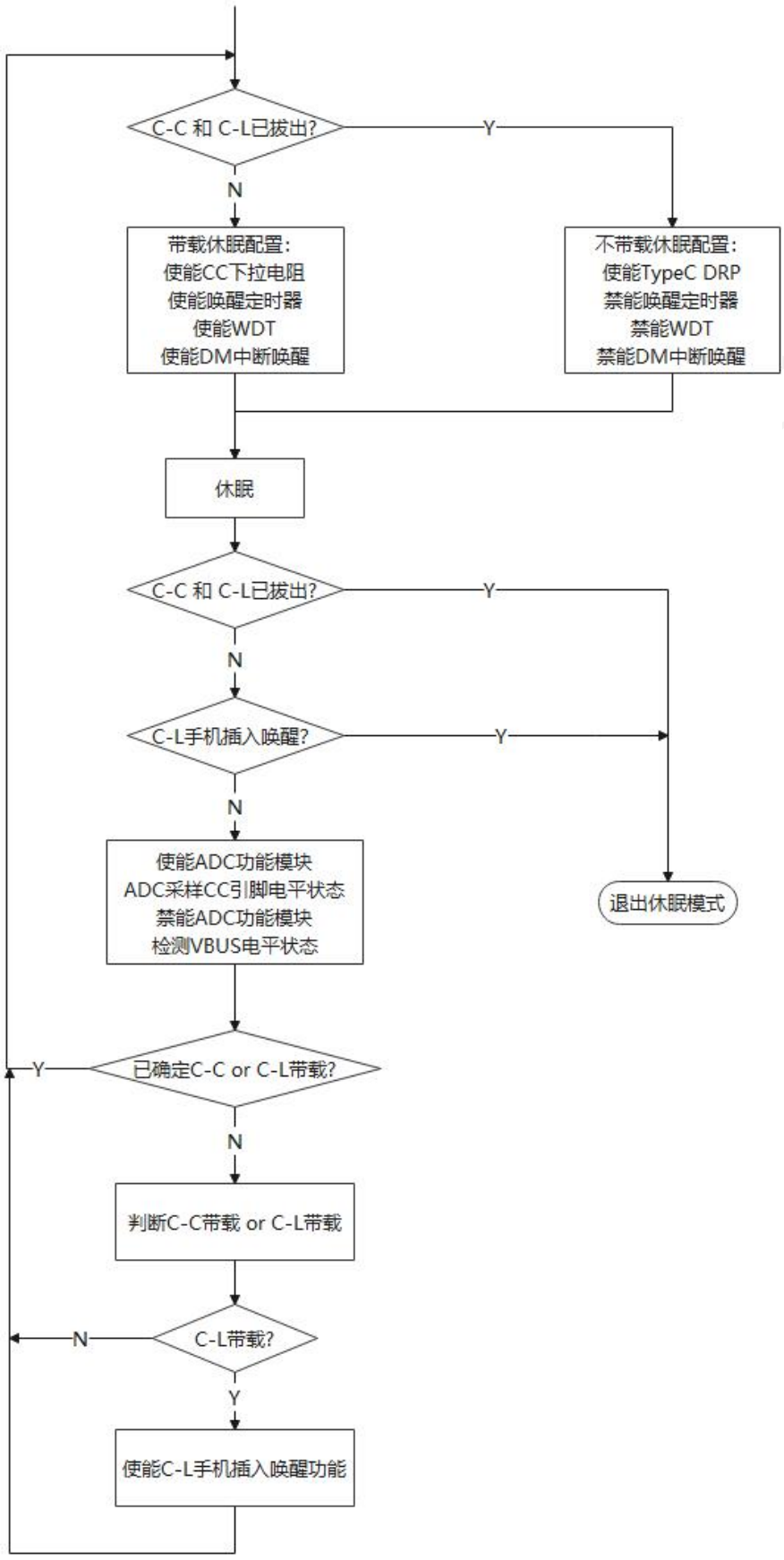



**图 3 C-C & C-L 带载休眠唤醒功能的程序流程图**

### 3.1.2 C-C & C-L 带载休眠唤醒功能的程序流程图的流程步骤

以下详细介绍说明上述 C-C & C-L 带载休眠唤醒功能的程序流程图的流程步骤和功能。

步骤一：根据 C-C 和 C-L 线缆是否拔出的状态进行端口轻载检测：

如果 C-C 和 C-L 线缆已拔出，跳转到步骤二：

否则，将会执行端口轻载检测，直到检测判断到轻载后跳转到步骤二：

步骤二：在休眠前，根据系统要求对以下各个功能模块进行配置后跳转到步骤三：

禁能 PD 功能模块

禁能 ADC 功能模块

禁能 Timer0 功能模块

配置外部中断功能模块

按键唤醒

DC-DC 中断唤醒

配置 GPIO 功能模块

配置多协议功能模块

禁能 DC-DC 功能模块

步骤三：在休眠前，根据 C-C 和 C-L 线缆是否拔出进行如下配置后跳转到步骤四：

如果 C-C 和 C-L 线缆已拔出，进行如下配置：

使能 TypeC DRP

禁能唤醒定时器

禁能 WDT 功能模块

禁能 DM 中断唤醒

如果 C-C 和 C-L 线缆未拔出，进行如下配置：

禁能 TypeC DRP，固定使能 TypeC CC 下拉电阻

使能唤醒定时器

使能 WDT 功能模块

使能 DM 中断唤醒

步骤四：进入休眠模式，唤醒后跳转到步骤五；

步骤五：系统唤醒后，根据 C-C 和 C-L 线缆是否拔出的状态进行如下处理：

如果 C-C 和 C-L 线缆已拔出，跳转到步骤九；

否则，跳转到步骤六：

步骤六：系统唤醒后，根据是否为 C-L 线缆手机插入唤醒进行如下处理：

如果检测到是 C-L 线缆手机插入唤醒，跳转到步骤九；

否则，跳转到步骤七；

步骤七：使能 ADC 功能模块对 CC 引脚和 VBUS 电平状态进行检测，完成后跳转到步骤八；

步骤八：检测是 C-C 带载还是 C-L 带载：

如果未确定是 C-C 还是 C-L 带载，进行检测判断是 C-C 还是 C-L 带载，跳转到步骤三：

如果检测到是 C-L 带载，使能 C-L 线缆手机插入唤醒功能后跳转到步骤三；

如果检测到是 C-C 带载，不使能 C-L 线缆手机插入唤醒功能直接跳转到步骤三；

如果已经确定是 C-C 还是 C-L 带载，跳转到步骤三；

步骤九：退出休眠模式，进入正常工作模式。

## 3.2 C-C & C-L 带载休眠唤醒功能的程序设计说明

### 3.2.1 轻载检测功能模块的程序设计说明

轻载检测功能模块的程序设计方法建议如下：用户可以通过检测判断 VBUS 的总电流小于用户设定的轻载电流值并且持续一段时间后进行判定为轻载模式。例如检测判断 VBUS 电流小于 200mA 连续 10S 钟后确定为轻载模式，程序设计举例如下：

```
353 static void sleep_and_wakeup(void)
354 {
355     if((plug_state_context.chg_dischg_state == STATE_DISCHG)//放电中
356         && (h_typeca.output.state == ATTACHED_SRC || h_typecb.output.state == ATTACHED_SRC))//Ca口连接着设备
357     {
358 //      Printf(printf_ibus,dcdc_data.telemetry.ibus_fb>>8);
359 //      Printf(printf_ibus,dcdc_data.telemetry.ibus_fb>>0);
360         if(dcdc_data.telemetry.ibus_fb>200)     //200mA, 轻载小电流判断
361         {
362             if(plug_state_context.sleep_flag.n_sleep_cnt)
363                 plug_state_context.sleep_flag.n_sleep_cnt--;
364         }
365         else
366         {
367             if(plug_state_context.sleep_flag.n_sleep_cnt<SLEEP_TIME)
368                 plug_state_context.sleep_flag.n_sleep_cnt++;
369         }
370     }
```

SLEEP_TIME 定义如下：

app_main.c | mod_adc_convert.c | cstartup.asm | app_interrupt.c | app_init.h | app_debug.h

```
194 /** sleep time**/
195 #define SLEEP_TIME                  (100)           /*!< 100*100mS*/
196
```

### 3.2.2 休眠前功能配置的程序设计说明

为了实现 C-C 和 C-L 带载正常休眠和唤醒的功能，以及最大程度降低移动电源的休眠功耗，在系统进入休眠前，根据系统要求必须合理对 MCU 的各个功能模块按照如下要求进行配置：

禁能 WDT 功能模块：

通过调用函数 wdt_disable()进行禁能 WDT 功能模块。

禁能 PD 功能模块：

通过调用函数 peripheral_pd_init()和 pd_enable(false)进行禁能 PD 功能模块。

禁能 ADC 功能模块：

通过调用函数 mod_adc_convert_disable()进行禁能 ADC 功能模块。

禁能 Timer0 功能模块：

通过调用函数 timer0_disable()进行禁能定时器 Timer0 功能模块。

配置外部中断功能模块：

通过调用函数 exi_init()进行配置外部中断功能模块。

根据系统要求进行使能按键中断，使能 DC-DC IC 中断引脚中断，以便中断唤醒系统。

配置 GPIO 功能模块：

通过调用函数 io_set_init()进行配置 GPIO 功能模块。

禁能 DC-DC 功能模块：

通过调用函数 mod_inoutput_disable()进行禁能 DC-DC 功能模块。

配置多协议功能模块：

通过调用函数 multi_protocol_sleep()进行配置多协议功能模块。

程序设计举例如下：

```
385     //延时5s进入休眠
386     if(plug_state_context.sleep_flag.n_sleep_cnt >= SLEEP_TIME)
387     {
388         plug_state_context.sleep_flag.n_sleep_cnt = 0;
389         plug_state_context.sleep_flag.b_sleep_event = true;
390 #if (!CFG_SYS_WDT_DISABLE)
391         //disable看门狗
392          wdt_disable();
393 #endif
394         //disable PD模块
395         peripheral_pd_init();
396 
397         //disable PD模块
398         pd_enable(false);
399 
400         //disable ADC
401         mod_adc_convert_disable();
402 
403         //disable timer0
404         timer0_disable();
405 
406         //enable 外部中断
407         exi_init();          //打开外部中断2功能
408 
409         /** io功能选择，输入输出**/
410         io_set_init();
411         CFGR1 &= (~(1<<2));          //关闭睡眠打开温度检测功能，降低休眠功耗
412 
413         //disable PSTOP
414         mod_inoutput_disable();
415 
416         //multi sleep
417         multi_protocol_sleep();
```

### 3.2.3 带载休眠功能配置的程序设计说明

如果 C-C 或 C-L 线缆未拔出进入的休眠模式，需求进行如下配置进入休眠：

使能 CC 下拉电阻：

使能唤醒定时器：

使能 WDT。

```
        //休眠前Type-c模块设置
        if(h_typeca.output.state == ATTACHED_SRC || h_typecb.output.state == ATTACHED_SRC)
        {
            TM0CON = 0x00;          // 停止定时器0
            CCDRP1 |= 0x04;         // 开typec_en
            SCNPRD1 = 0x00;
            CCDRP1 |= 0x03;         // 开drp
            CCACTL1 = 0x00;         // 开下拉电阻
            CCBCTL1 = 0x00;         // 开下拉电阻
            AUTODET1 &= (~0xC0);    // 关auto det
            CCIE1 = 0x00;
            CCIF1 |= 0x0F;

            MCK |= (1<<5);          // 开10K lirc
            CLKSEL0 |= 0x03;        // timer0 选10K 计时,100us计数一/////
            CLKDIV0 = 0x02;         // 4分频
            TM0IN = 568 / 4;        // 56.8ms
            TM0CNT = 0;
            TM0IF = 0;
            T0RSTB = 0;
            TM0IE = 0;
            GIE = 0;
            T0EN = 1;
//          PT30 = 0;
            while (!TM0IF);
//          PT30 = 1;
            TM0IF = 0;
            CCDRP1 = 0x03;
            TM0CON = 0x00;          // 停止定时器0
            GIE = 1;

            interrupt_flag.byte = 0;
            wakeup_phase_debounce = 0;
            wakeup_phase = 0;
            timer2_init();          // 设置Timer2 500ms定时唤醒
            #if (!CFG_SYS_WDT_DISABLE)
                /** 初始化看门狗**/
                wdt_init();
            #endif
            sleep_mode = 2;         // 休眠模式2
        }
        else
        {
            sleep_mode = 1;         // 休眠模式1
        }
```

### 3.2.4 退载后休眠功能配置的程序设计说明

如果在休眠过程中，C-C 和 C-L 线缆拔出，需求进行如下配置进入休眠：

使能 TypeC DRP：

禁能唤醒定时器：

禁能 WDT：

禁能 DM 中断唤醒。

```
465         //disable DCDC模块 and close pgate
466         mod_dcdc_disable();
467
468         do{
469             if(sleep_mode==1)
470             {
471                 fwlib_typec_cc_pull_high_down_config(TYPECA,TYPEC_CCX_CUR_330UA,DISABLE);    //打开上拉电流源，关闭下拉
472                 fwlib_typec_cc_pull_high_down_config(TYPECB,TYPEC_CCX_CUR_330UA,DISABLE);    //打开上拉电流源，关闭下拉
473                 CCDRP1 = 0x04;      //开typec_en
474                 typec_sleep(TYPECA);
475                 typec_sleep(TYPECB);
476                 SCNPRD5 = 0;
477                 timer2_disable();
478                 EXIIE &= ~0x06;
479                 interrupt_flag.byte = 0;
480                 plug_state_context.state.byte = 0;
481                 plug_state_context.chg_dischg_state = STATE_NULL;
482                 mod_display_led_clear();
483                 #if (!CFG_SYS_WDT_DISABLE)
484                     //disable看门狗
485                      wdt_disable();
486                 #endif
487             }
488             else
489             {
490                 if(interrupt_flag.bit.b_int_dc_dc==1)
491                 {
492                     interrupt_flag.bit.b_int_dc_dc = 0;
493                     mod_dcdc_disable();
494                 }
495             }
496             adc_disable();          //禁能ADC模块
```

### 3.2.5 检测 CC 引脚电平状态的功能程序设计说明

定时器唤醒系统后，使能 ADC 功能模块，对 CC1A、CC2A、CC1B、CC2B 引脚进行电平采样，检测 VBUS 电平状态，完成采样检测后，禁能 ADC 功能模块。

如果是 C-C 带载休眠，需要关闭 CC 上拉电流源，打开下拉电阻，进行 CC 引脚电平状态采样检测；

如果是 C-L 带载休眠，需要打开 CC 上拉电流源，关闭下拉电阻，进行 CC 引脚电平状态采样检测。

app_main.c

```
570  if(interrupt_flag.bit.b_int_timer2==1)
571  {
572      interrupt_flag.bit.b_int_timer2 = 0;
573      if(wakeup_phase==2 || wakeup_phase==3)
574      {
575          fwlib_typec_cc_pull_high_down_config(TYPECA,TYPEC_CCX_CUR_080UA,DISABLE);   //打开上拉电流源，关闭下拉
576          fwlib_typec_cc_det_set(TYPECA, ENABLE);                                     //打开自动检测
577          fwlib_typec_cc_pull_high_down_config(TYPECB,TYPEC_CCX_CUR_080UA,DISABLE);   //打开上拉电流源，关闭下拉
578          fwlib_typec_cc_det_set(TYPECB, ENABLE);                                     //打开自动检测
579          CCDRP1 = 0x04;                                                              //打开typec_en
580      }
581      adc_enable();                       //使能ADC模块
582      for(uint16_t i=0;i<100;i++);        //延时后再做ADC检测
583      uint8_t adc_convert_loop = 0, adc_convert_continue = 1;
584      do{
585          adc_convert.adc_branch = 4;
586          adc_convert.cc1a.adc_code_sl = 0;
587          adc_convert.cc2a.adc_code_sl = 0;
588          adc_convert.cc1b.adc_code_sl = 0;
589          adc_convert.cc2b.adc_code_sl = 0;
590          while(!mod_adc_convert());      //唤醒后检测CC电平;
591      }while(adc_convert_continue==1 && ++adc_convert_loop<1);
592      if(wakeup_phase==2 || wakeup_phase==3)
593      {
594          fwlib_typec_cc_det_set(TYPECA, DISABLE);                                    //关闭自动检测
595          fwlib_typec_cc_pull_high_down_config(TYPECA,TYPEC_CCX_CUR_OFF,ENABLE);      //关闭上拉电流源，打开下拉
596          fwlib_typec_cc_det_set(TYPECB, DISABLE);                                    //关闭自动检测
597          fwlib_typec_cc_pull_high_down_config(TYPECB,TYPEC_CCX_CUR_OFF,ENABLE);      //关闭上拉电流源，打开下拉
598      }
599      iic_receive_bytes(SC_STATUS, (BANK_DCDC uint8_t *)&dcdc_data.reg.status, 1);   //检测TypecA VBUS 是否有电压
```

### 3.2.6 带载检测功能程序设计说明

因为 C-C 和 C-L 带载休眠的特性不太一样，为了保证 C-C 和 C -L 带载休眠功能正常，需求正确检测区别出是 C-C 带载还是 C-L 带载。

检测判断的方法如下：休眠前使能 CC 引脚的下拉电阻（5.1K）功能和使能定时器中断功能，定时器中断唤醒系统后检测 CC 电平状态和 VBUS 电平状态，如果检测到 CC 电平状态大于用户设定的电压值（例如 0.2V）或 VBUS 是有电压的状态，则判定为 C-C 带载，否则，判定为 C-L 带载。

由于不同型号的手机的特性不太一样，为了更加准确区别出是 C-C 带载还是 C-L 带载，建议多做几次（例如 3 次）CC 电平状态和 VBUS 电平状态的检测后再作最终决定。

如果检测到是 C-L 带载，为了实现 C-L 线缆手机插入检测识别唤醒功能，需要配置 DM 引脚为 GPIO 功能，并且使能其外部中断功能。

```
601     if(wakeup_phase_debounce<10)
602         wakeup_phase_debounce++;
603 
604     switch(wakeup_phase)
605     {
606         case 0: //判断CC线 or CL线带载休眠
607             if( adc_convert.cc1a.adc_code_sl>0x0F0 \
608             ||  adc_convert.cc2a.adc_code_sl>0x0F0 \
609             ||  adc_convert.cc1b.adc_code_sl>0x0F0 \
610             ||  adc_convert.cc2b.adc_code_sl>0x0F0 \
611             ||  M_DCDC_REG_GET(dcdc_data.reg.status, AC_OK, 1)==1   \
612             ||  PT13==0 \
613             )
614             {
615                 wakeup_phase_debounce = 0;
616                 wakeup_phase = 1;       //CC线带载休眠，wakeup_phase置 1
617             }
618             else
619             {
620                 if(wakeup_phase_debounce>=3)
621                 {
622                     CFGR6 &= ~0x30;     // 设置PT32（DM1）为GPIO数字口，以便作为外部中断唤醒
623                     CFGR5 &= ~0x03;     // 设置PT24（DM2）为GPIO数字口，以便作为外部中断唤醒
624                     EXICON &= ~0x3C;
625                     EXIIF &= ~0x06;
626                     INTCFG1 |= 0x00;
627                     INTCFG2 |= 0x10;
628                     INTCFG3 |= 0x04;
629                     interrupt_flag.bit.b_int_DM1=0;
630                     interrupt_flag.bit.b_int_DM2=0;
631                     EXIIE |= 0x06;
632                     wakeup_phase_debounce = 0;
633                     wakeup_phase = 2;   //CL线带载休眠，wakeup_phase置 2
634                 }
635             }
636             break;
```

### 3.2.7 退载检测功能程序设计说明

因为 C-C 和 C-L 带载休眠的特性不太一样，所以 C-C 和 C -L 退载检测的方法也不相同，具体检测功能程序设计方法如下：

C-C 退载检测功能模块

检测判断 C-C 退载的方法如下：定时器中断唤醒 MCU 后，使能所有 CC 引脚的下拉电阻（5.1K）功能，使能 ADC 进行检测 CC 电平状态，如果检测到所有 CC pin（CC1-A、CC2-A、CC1-B、CC2-B）的电平状态均小于用户设定的电压值（例如 0.2V）时，则初步判定为 C-C 退载。

```
637             case 1: //CC线带载休眠(开机状态)
638                 if(adc_convert.cc1a.adc_code_sl<0x0F0 \
639                 && adc_convert.cc2a.adc_code_sl<0x0F0 \
640                 && adc_convert.cc1b.adc_code_sl<0x0F0 \
641                 && adc_convert.cc2b.adc_code_sl<0x0F0 \
642                 )
643                 {
644                     //sleep_mode = 1;           //转休眠模式1
645                     wakeup_phase = 3;           //转 phase 3 状态做进一步判断设备是否拔出
646                 }
647                 break;
```

进一步检测确认 C-C 退载的方法如下：定时器中断唤醒 MCU 后，使能所有 CC 引脚的上拉电流源（例如 80uA）功能，使能 ADC 进行检测 CC 电平状态，如果检测到所有 CC pin（CC1-A、CC2-A、CC1-B、CC2-B）的电压状态均大于用户设定的电压值（例如 2.4V）时，则判定为 C-C 退载。

```
660             case 3: //CC线带载休眠（关机状态）
661                 if((adc_convert.cc1a.adc_code_sl>0xF00  || adc_convert.cc1a.adc_code_sl<0x0F0)\
662                 && (adc_convert.cc2a.adc_code_sl>0xF00  || adc_convert.cc2a.adc_code_sl<0x0F0) \
663                 && (adc_convert.cc1b.adc_code_sl>0xF00  || adc_convert.cc1b.adc_code_sl<0x0F0) \
664                 && (adc_convert.cc2b.adc_code_sl>0xF00  || adc_convert.cc2b.adc_code_sl<0x0F0) \
665                 )
666                 {
667                     //CC电压>2.4V, 设备拔出
668                     sleep_mode = 1;         //转休眠模式1
669                 }
670                 break;
```

◆ C-L 退载检测功能模块

◆ 检测判断 C-L 退载的方法如下：定时器中断唤醒 MCU 后，使能所有 CC 引脚的上拉电流源（例如 80uA）功能，使能 ADC 进行检测 CC 电平状态，如果检测到所有 CC pin（CC1-A、CC2-A、CC1-B、CC2-B）的电压状态均大于用户设定的电压值（例如 2.4V）时，则判定为 C-L 退载。

```
        case 2: //CL线带载休眠
            if(adc_convert.cc1a.adc_code_sl>0xF00 \
            && adc_convert.cc2a.adc_code_sl>0xF00 \
            && adc_convert.cc1b.adc_code_sl>0xF00 \
            && adc_convert.cc2b.adc_code_sl>0xF00 \
            )
            {
                //CC电压>2.4V, 设备拔出
                sleep_mode = 1;          //转休眠模式1
            }
            break;

        default:
            break;
    }
```

### 3.2.8 手机插入识别唤醒功能模块

因为 C-C 和 C-L 带载休眠的特性不太一样，所以 C-C 和 C -L 带载休眠模式下手机插入识别唤醒的方法不相同，具体检测方法请参考 2.5.1 和 2.5.2 章节描述。

C-C 线缆手机插入识测唤醒功能模块

在 C-C 退载后，系统进入休眠模式 1，为了能够正确实现手机插入识别唤醒功能，需要使能 USB TypeC 自动扫描功能（DRP）和打开自动扫描（DRP）中断功能，当检测识别到 C-C 线缆的手机等设备插入后，产生 USB TypeC 的 DRP 中断，此中断便可以唤醒系统退出休眠模式进入正常工作模式。

```
    do{
        if(sleep_mode==1)
        {
            fwlib_typec_cc_pull_high_down_config(TYPECA,TYPEC_CCX_CUR_330UA,DISABLE);   //打开上拉电流源，关闭下拉
            fwlib_typec_cc_pull_high_down_config(TYPECB,TYPEC_CCX_CUR_330UA,DISABLE);   //打开上拉电流源，关闭下拉
            CCDRP1 = 0x04;      //开typec_en
            typec_sleep(TYPECA);
            typec_sleep(TYPECB);
            SCNPRD5 = 0;
```

```
        if(sleep_mode==1)
        {
            break;              //休眠模式1唤醒后，退出休眠模式loop
        }
        else
```

◆ C-L 线缆手机插入识别唤醒功能模块

◆ 为了能够正确实现 C-L 线缆的手机插入识别唤醒功能，需要将 DM 引脚配置为 GPIO 功能，并使能 DM 引脚的外部中断功能，当检测识别到 C-L 线缆的手机等设备插入后，产生外部中断，此中断便可以唤醒系统退出休眠模式进入正常工作模式。

```
622             if(wakeup_phase_debounce>=3)
623             {
624                 CFGR6 &= ~0x30;      // 设置PT32（DM1）为GPIO数字口，以便作为外部中断唤醒
625                 CFGR5 &= ~0x03;      // 设置PT24（DM2）为GPIO数字口，以便作为外部中断唤醒
626                 EXICON &= ~0x3C;
627                 EXIIF &= ~0x06;
628                 INTCFG1 |= 0x00;
629                 INTCFG2 |= 0x10;
630                 INTCFG3 |= 0x04;
631                 interrupt_flag.bit.b_int_DM1=0;
632                 interrupt_flag.bit.b_int_DM2=0;
633                 EXIIE |= 0x06;
634                 wakeup_phase_debounce = 0;
635                 wakeup_phase = 2;    //CL线带载休眠，wakeup_phase置 2
636                 PT2EN5= 1;  PT2PU5= 0;  PT25= 1;
637
638             }
```

C-L 线缆检测到手机插入后唤醒退出休眠模式：

```
562         if(wakeup_phase==2)
563         {
564             if(interrupt_flag.bit.b_int_DM1==1 \
565             || interrupt_flag.bit.b_int_DM2==1 )
566             {
567                 PT2EN5= 0;  PT2PU5= 0;  PT25= 0;
568                 break;      //CL线休眠时插入手机唤醒后，退出休眠模式loop
569             }
570         }
```

# 4 C-C & C-L 带载休眠功能的开发设计注意事项

## 4.1 休眠前打开 CC 引脚下拉电阻方法

无论是 C-C 带载还是 C-L 带载进入休眠，在第一次进入休眠前，必须打开 CC 引脚的下拉电阻。根据项目使用的端口不同，打开 CC 引脚的下拉电阻的方法不相同，具体方法如下所述。

### 4.1.1 打开 TypecA 模块的 CC 引脚下拉电阻

如果项目只使用到 USB Typec_A 端口，必须按照如下所示程序进行配置打开 CC 引脚的下拉电阻：

```
419     //休眠前Type-c模块设置
420     if(h_typeca.output.state == ATTACHED_SRC || h_typecb.output.state == ATTACHED_SRC)
421     {
422         TM0CON = 0x00;          // 停止定时器0
423         CCDRP1 |= 0x04;         // 开typec_en
424         SCNPRD1 = 0x00;
425         CCDRP1 |= 0x02;         // 开drp
426         CCACTL1 = 0x00;         // 开下拉电阻
427         CCBCTL1 = 0x00;         // 开下拉电阻
428         AUTODET1 &= (~0xC0);    // 关auto det
429         CCIE1 = 0x00;
430         CCIF1 |= 0x0F;
431
432         MCK |= (1<<5);          // 开10K lirc
433         CLKSEL0 |= 0x03;        // timer0 选10K 计时,100us计数一/////
434         CLKDIV0 = 0x02;         // 4分频
435         TM0IN = 568 / 4;        // 56.8ms
436         TM0CNT = 0;
437         TM0IF = 0;
438         T0RSTB = 0;
439         TM0IE = 0;
440         GIE = 0;
441         T0EN = 1;
442         PT30 = 0;
443         while (!TM0IF);
444         PT30 = 1;
445         TM0IF = 0;
446         CCDRP1 = 0x02;
447         TM0CON = 0x00;          // 停止定时器0
448         GIE = 1;
```

### 4.1.2 打开 TypecB 模块的 CC 引脚下拉电阻

如果项目只使用到 USB Typec_B 端口，必须按照如下所示程序进行配置打开 CC 引脚的下拉电阻：

```
419     //休眠前Type-c模块设置
420     if(h_typeca.output.state == ATTACHED_SRC || h_typecb.output.state == ATTACHED_SRC)
421     {
422         TM0CON = 0x00;          // 停止定时器0
423         CCDRP1 |= 0x04;         // 开typec_en
424         SCNPRD1 = 0x00;
425         CCDRP1 |= 0x01;         // 开drp
426         CCACTL1 = 0x00;         // 开下拉电阻
427         CCBCTL1 = 0x00;         // 开下拉电阻
428         AUTODET1 &= (~0xC0);    // 关auto det
429         CCIE1 = 0x00;
430         CCIF1 |= 0x0F;
431
432         MCK |= (1<<5);          // 开10K lirc
433         CLKSEL0 |= 0x03;        // timer0 选10K 计时,100us计数一/////
434         CLKDIV0 = 0x02;         // 4分频
435         TM0IN = 568 / 4;        // 56.8ms
436         TM0CNT = 0;
437         TM0IF = 0;
438         T0RSTB = 0;
439         TM0IE = 0;
440         GIE = 0;
441         T0EN = 1;
442         PT30 = 0;
443         while (!TM0IF);
444         PT30 = 1;
445         TM0IF = 0;
446         CCDRP1 = 0x01;
447         TM0CON = 0x00;          // 停止定时器0
448         GIE = 1;
```

### 4.1.3 打开 TypecA & TypecB 模块的 CC 引脚下拉电阻

如果项目同时使用了 USB Typec_A & TypecB 端口，必须按照如下所示程序进行配置打开 CC 引脚的下拉电阻：

```
419         //休眠前Type-c模块设置
420         if(h_typeca.output.state == ATTACHED_SRC || h_typecb.output.state == ATTACHED_SRC)
421         {
422             TM0CON = 0x00;          // 停止定时器0
423             CCDRP1 |= 0x04;         // 开typec_en
424             SCNPRD1 = 0x00;
425             CCDRP1 |= 0x03;         // 开drp
426             CCACTL1 = 0x00;         // 开下拉电阻
427             CCBCTL1 = 0x00;         // 开下拉电阻
428             AUTODET1 &= (~0xC0);    // 关auto det
429             CCIE1 = 0x00;
430             CCIF1 |= 0x0F;
431
432             MCK |= (1<<5);          // 开10K lirc
433             CLKSEL0 |= 0x03;        // timer0 选10K 计时,100us计数一/////
434             CLKDIV0 = 0x02;         // 4分频
435             TM0IN = 568 / 4;        // 56.8ms
436             TM0CNT = 0;
437             TM0IF = 0;
438             T0RSTB = 0;
439             TM0IE = 0;
440             GIE = 0;
441             T0EN = 1;
442 //          PT30 = 0;
443             while (!TM0IF);
444 //          PT30 = 1;
445             TM0IF = 0;
446             CCDRP1 = 0x03;
447             TM0CON = 0x00;          // 停止定时器0
448             GIE = 1;
```

### 4.2 唤醒定时器配置要求

无论是 C-C 还是 C-L 带载休眠，需要配置定时器间隔唤醒系统进行检测 C-C 和 C-L 是否退载，所以必须使用具有中断唤醒功能的定时器（如 Timer2）进行定时唤醒，唤醒定时器配置注意事项如下：

定时器时钟源必须设置为内部 10K 振荡器时钟；

定时器必须打开中断功能，定时时间到了才能唤醒系统；

为了降低休眠功耗，间隔唤醒时间建议设定为 500ms。

### 4.3 唤醒后延时 ADC

系统唤醒后，为了保证 ADC 数据准确，打开 ADC 模块后，必须延时一段时间后再进行 ADC 采样。

```
581         adc_enable();                           //使能ADC模块
582         for(uint16_t i=0;i<100;i++);            //延时后再做ADC检测
583         uint8_t adc_convert_loop = 0, adc_convert_continue = 1;
584         do{
585             adc_convert.adc_branch = 4;
586             adc_convert.cc1a.adc_code_sl = 0;
587             adc_convert.cc2a.adc_code_sl = 0;
588             adc_convert.cc1b.adc_code_sl = 0;
589             adc_convert.cc2b.adc_code_sl = 0;
590             while(!mod_adc_convert());          //唤醒后检测CC电平;
591         }while(adc_convert_continue==1 && ++adc_convert_loop<1);
```

### 4.4 检测区别 C-C 带载 or C-L 带载

由于 C-C 带载休眠和 C-L 带载休眠特性不一样，所以系统需要检测区别出是 C-C 带载休眠还是 C-L 带载休眠；

由于不同型号的手机会有不同的特性，为了准确判断 C-C 带载还是 C-L 带载，建议多做几次检测判断处理后再决定是 C-C 带载还是 C-L 带载。

```
606         switch(wakeup_phase)
607         {
608             case 0: //判断CC线 or CL线带载休眠
609                 if( adc_convert.cc1a.adc_code_sl>0x0F0 \
610                 ||  adc_convert.cc2a.adc_code_sl>0x0F0 \
611                 ||  adc_convert.cc1b.adc_code_sl>0x0F0 \
612                 ||  adc_convert.cc2b.adc_code_sl>0x0F0 \
613                 ||  M_DCDC_REG_GET(dcdc_data.reg.status, AC_OK, 1)==1    \
614                 ||  PT13==0 \
615                 )
616                 {
617                     wakeup_phase_debounce = 0;
618                     wakeup_phase = 1;       //CC线带载休眠，wakeup_phase置 1
619                 }
620                 else
621                 {
622                     if(wakeup_phase_debounce>=3)
623                     {
624                         CFGR6 &= ~0x30;     // 设置PT32（DM1）为GPIO数字口，以便作为外部中断唤醒
625                         CFGR5 &= ~0x03;     // 设置PT24（DM2）为GPIO数字口，以便作为外部中断唤醒
626                         EXICON &= ~0x3C;
627                         EXIIF &= ~0x06;
628                         INTCFG1 |= 0x00;
629                         INTCFG2 |= 0x10;
630                         INTCFG3 |= 0x04;
631                         interrupt_flag.bit.b_int_DM1=0;
632                         interrupt_flag.bit.b_int_DM2=0;
633                         EXIIE |= 0x06;
634                         wakeup_phase_debounce = 0;
635                         wakeup_phase = 2;   //CL线带载休眠，wakeup_phase置 2
636                         PT2EN5= 1;  PT2PU5= 0;  PT25= 1;
637
638                     }
639                 }
640                 break;
641             case 1: //CC线带载休眠(开机状态)
```

# 5 C-C & C-L 带载休眠唤醒功能方案的测试结果

以下为使用增加了 C-C 和 C-L 带载休眠唤醒功能的芯海科技 CSU3AF10 Demo Code（版本号: csu3afx10_sdk_1.11_V0.0.10）在芯海科技 CSU3AF10 demo board（CSU3AF10 EVB V1.1）上测试带着手机休眠和手机插入识别唤醒的测试波形和结果，测试使用的手机为红米 K40，电池供电电压为 8V。

测试通道信号说明：通道 1 为 VBUS 电压，通道 2 为 CC1A 波形，通道 3 为 DM1 波形，通道 4 为 PT13 波形，PT13 为低电平表示 MCU 进入休眠状态，PT13 为高电平表示 MCU 处理唤醒状态。

测试环境说明：下图为测试环境 CSU3AF10 demo board（CSU3AF10 EVB V1.1）：

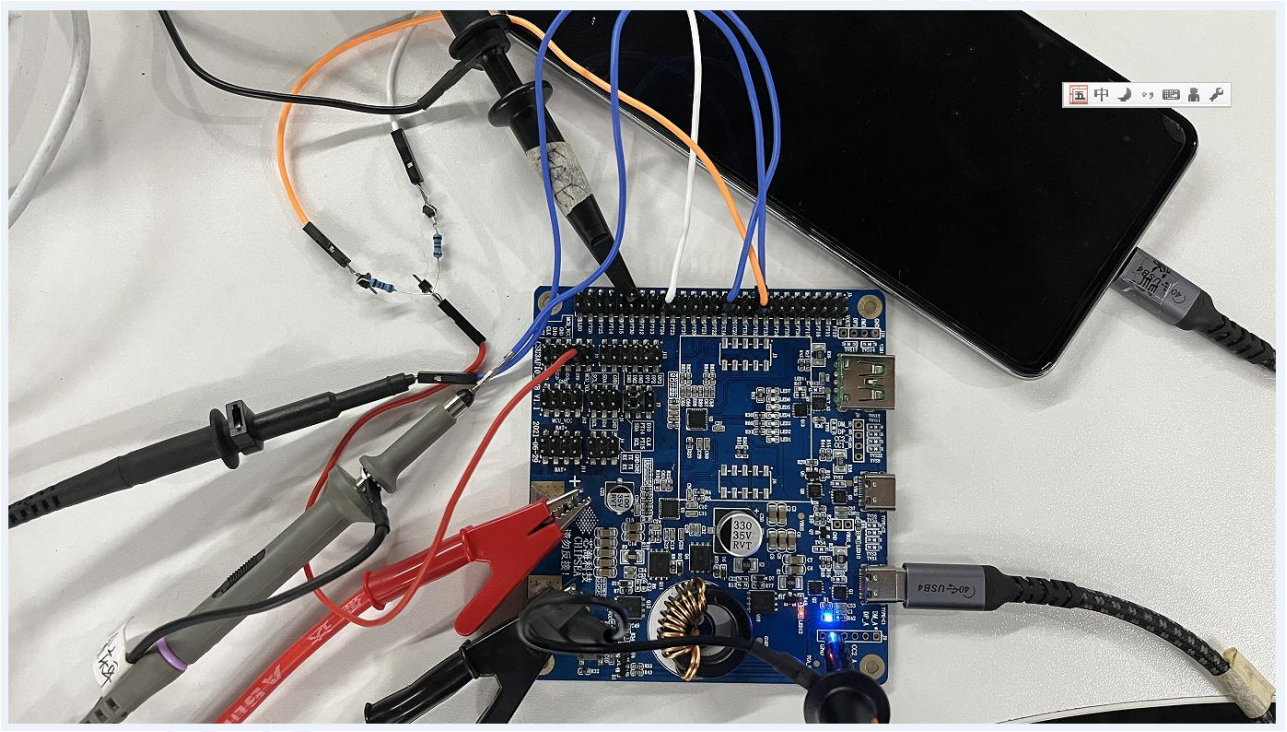


图 4 C-C 和 C-L 带载休眠唤醒功能测试环境

## 5.1 C-C 带载休眠功能测试

下图为使用红米 K40 手机通过 C-C 线缆插入 USB Typec_A 进行充电，充饱后进入休眠的波形，进入休眠后，间隔 500ms 唤醒系统进行检测手机是否拔出。

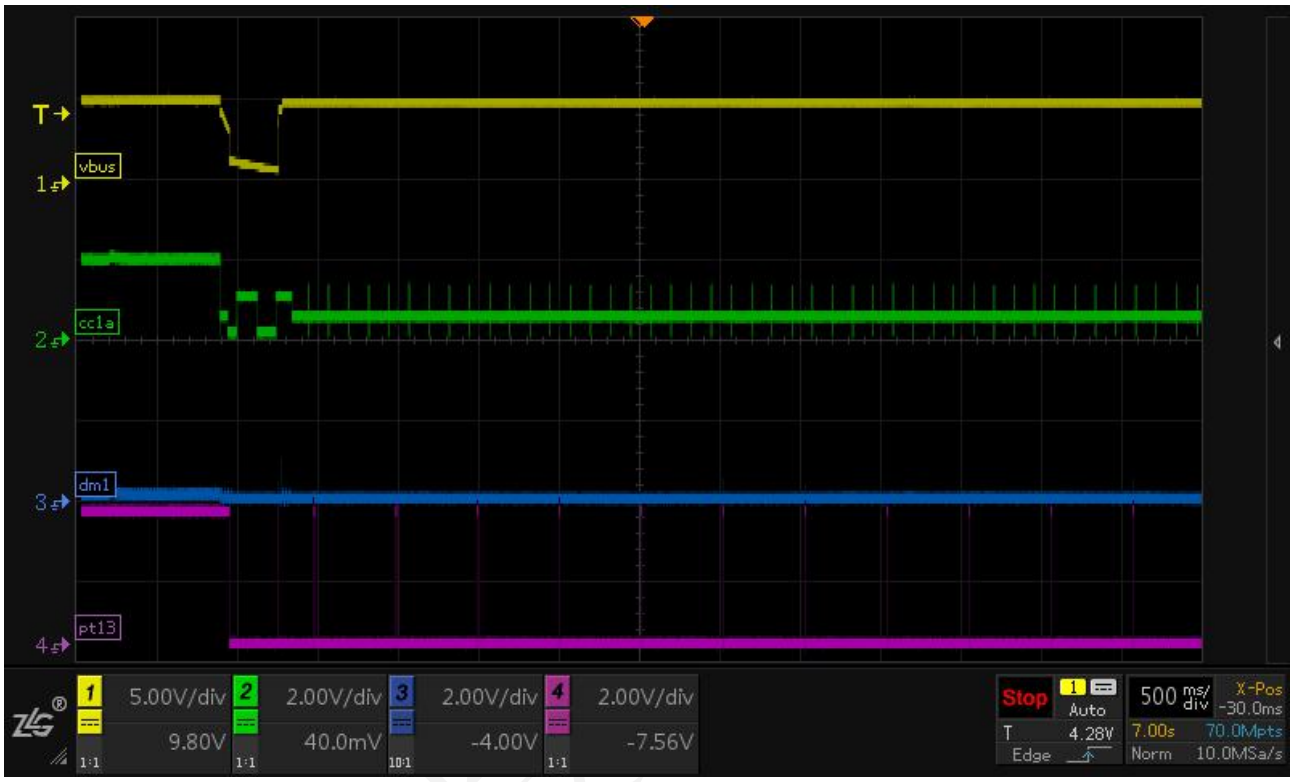



图 5 C-C 带载休眠功能测试波形

下图为放大刚开始进入休眠时刻的波形：

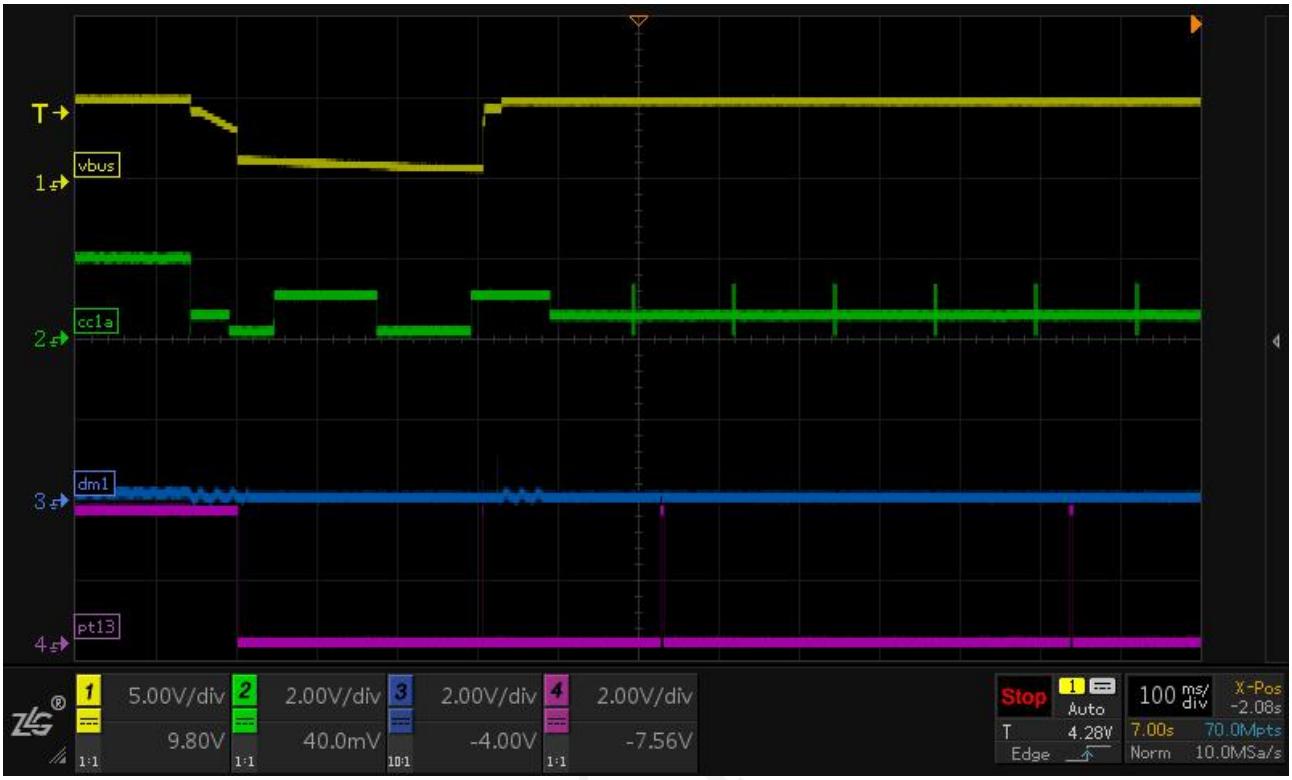



图 6 C-C 带载休眠功能测试波形(放大)

## 5.2 C-L 带载休眠功能测试

下图为使用红米 K40 手机通过 C-L 线缆插入 USB Typec_A 进行充电，充饱后进入休眠的波形，进入休眠后，间隔 500ms 唤醒系统进行检测 C-L 线缆是否拔出。

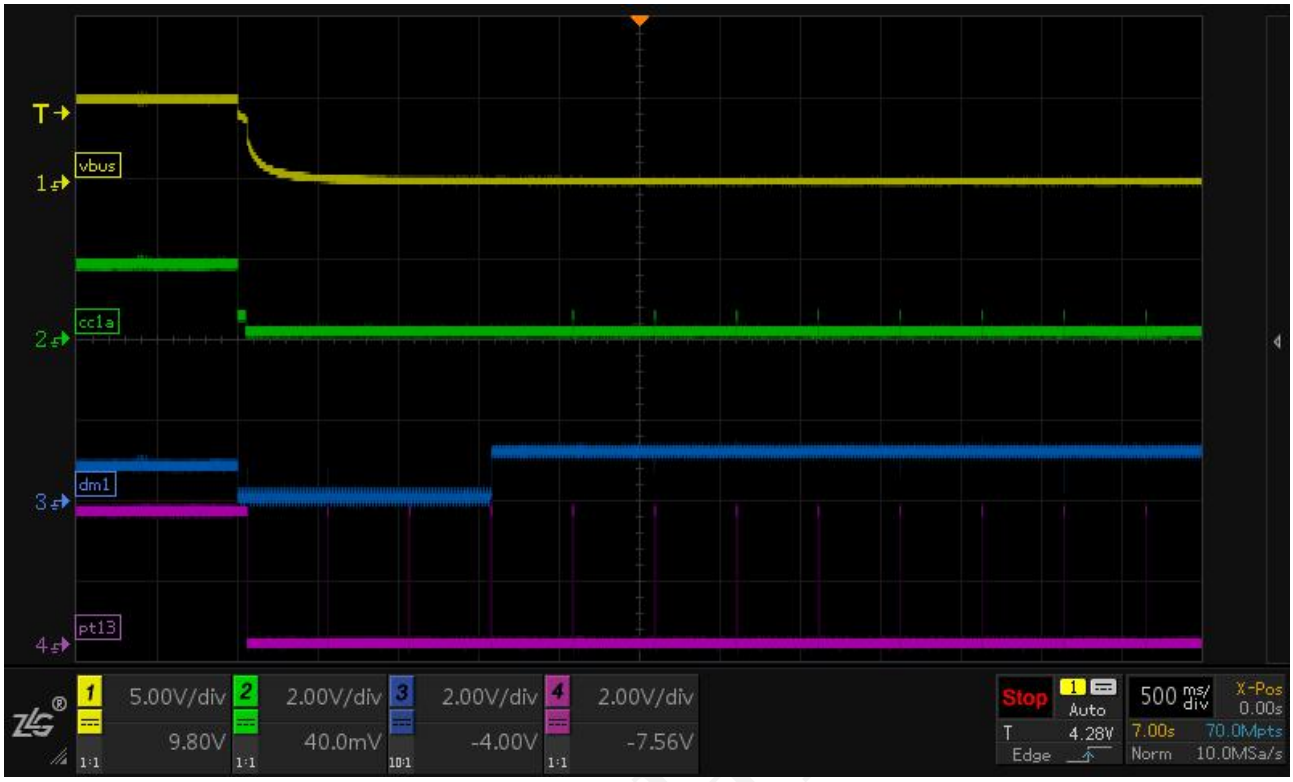



图 7 C-L 带载休眠功能测试波形

## 5.3 C-C 手机插入识别唤醒

下图为拔出手机后，通过 C-C 线缆插入红米 K40 手机到 USB Typec_A 唤醒系统的波形。

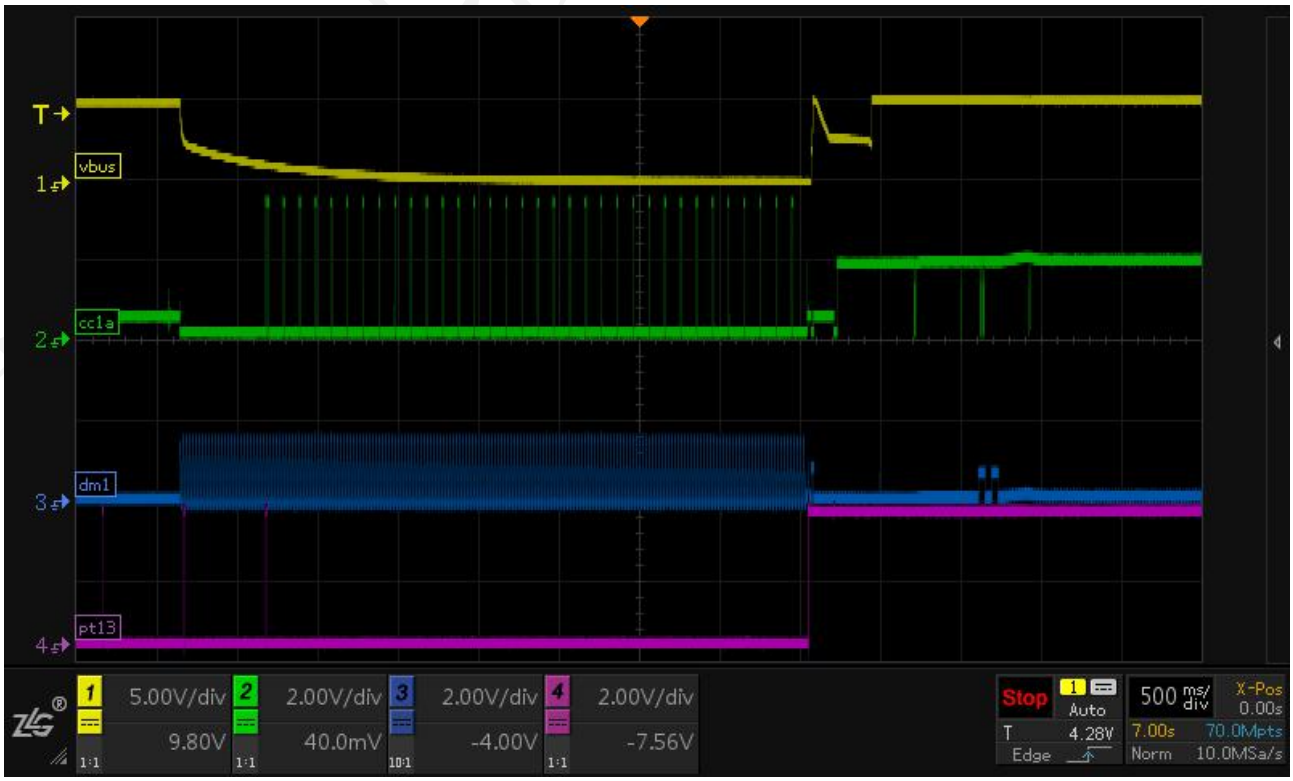


图 8 C-C 手机插入唤醒功能测试波形

## 5.4 C-L 手机插入识别唤醒

下图为拔出手机后，通过 C-L 线缆插入红米 K40 手机到 USB Typec_A 唤醒系统的波形。

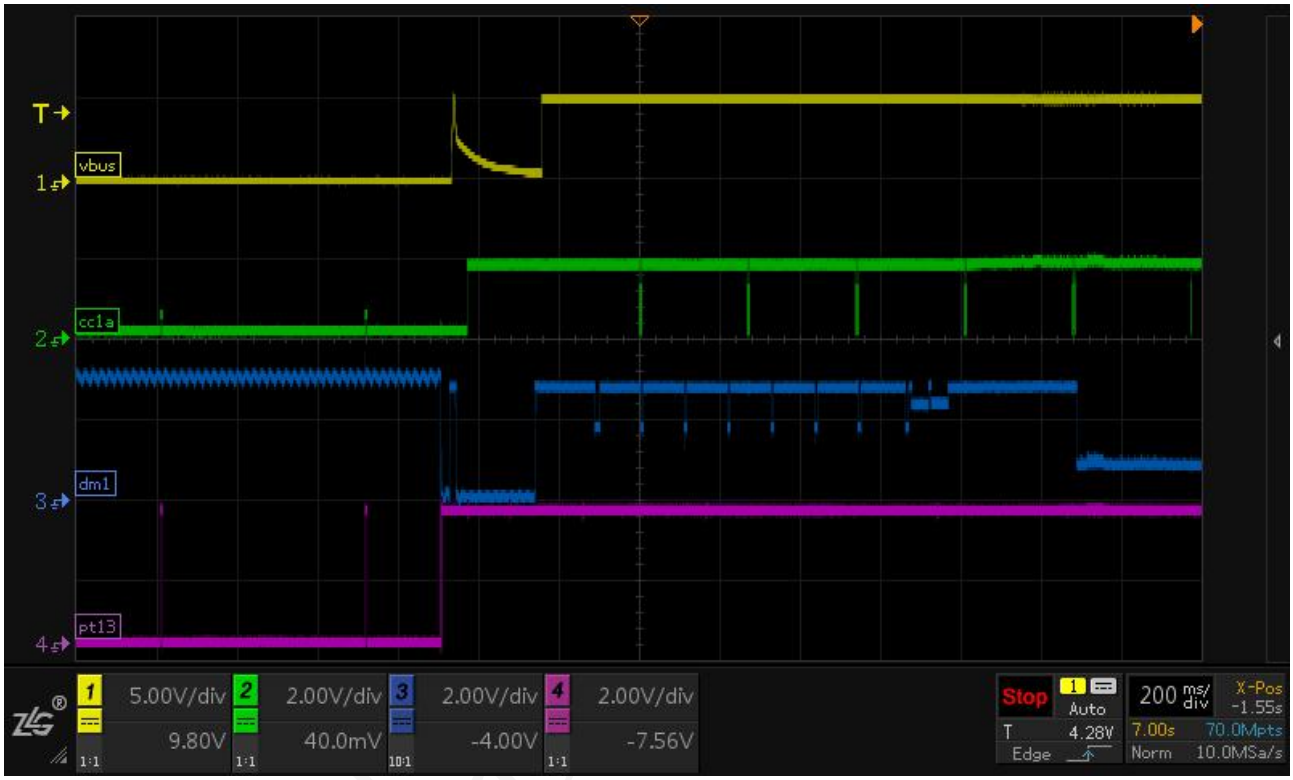



图 9 C-L 手机插入唤醒功能测试波形

## 5.5 C-C & C-L 带载休眠系统功耗测试结果

以下为基于芯海科技 CSU3AF10 MCU 设计的移动电源参考方案增加 C-C 和 C-L 带载休眠唤醒功能后，在不同的场景下测试的电池端的系统功耗：

场景 1：USB Typec_A & USB TypecB 不带载进入休眠时，测试得系统电池端的功耗约为 160uA；

场景 2：USB Typec_A or USB Typec_B 连接着 C-L 线缆,测试得系统电池端的功耗约为 160uA~310uA；

场景 3：USB Typec_A or USB TypecB 连接着 C-C 线缆（连接着手机）,测试得系统电池端的功耗约为 600uA~740uA；

表 1 C-C & C-L 带载休眠系统功耗测试参考结果

| 端口 | 测试场景 | 带载测试 | 测试休眠功耗（IBat） | 测试条件 |
|---|---|---|---|---|
| TypecA | 未连接任何设备 | C-C & C-L 不带载休眠 | 160uA | VBAT:8V，<br>Timer2:没有设置定时唤醒。 |
| TypecB | 未连接任何设备 | | | |
| TypecA | 连接 C-L 线（连接着手机） | TypecA C-L 带载休眠 | 160uA~310uA | VBAT:8V，<br>Timer2:设定 500ms 定时唤醒，<br>打开 WDT 功能，<br>打开了 DM 唤醒功能。 |
| TypecB | 未连接任何设备 | | | |
| TypecA | 未连接任何设备 | TypecB C-L 带载休眠 | 160uA~310uA | |
| TypecB | 连接 C-L 线（连接着手机） | | | |
| TypecA | 连接 C-C 线（连接着手机） | TypecA C-C 带载休眠 | 600uA~740uA | VBAT:8V，<br>Timer2:设定 500ms 定时唤醒，<br>关闭 WDT 功能，<br>关闭了 DM 唤醒功能。 |
| TypecB | 未连接任何设备 | | | |
| TypecA | 未连接任何设备 | TypecB C-C 带载休眠 | 600uA~740uA | |
| TypecB | 连接 C-C 线（连接着手机） | | | |

## 免责声明和版权公告

本文档中的信息，包括供参考的 URL 地址，如有变更，恕不另行通知。

本文档可能引用了第三方的信息，所有引用的信息均为“按现状”提供，芯海科技不对信息的准确性、真实性做任何保证。

芯海科技不对本文档的内容做任何保证，包括内容的适销性、是否适用于特定用途，也不提供任何其他芯海科技提案、规格书或样品在他处提到的任何保证。

芯海科技不对本文档是否侵犯第三方权利做任何保证，也不对使用本文档内信息导致的任何侵犯知识产权的行为负责。本文档在此未以禁止反言或其他方式授予任何知识产权许可，不管是明示许可还是暗示许可。

Wi-Fi 联盟成员标志归 Wi-Fi 联盟所有。蓝牙标志是 Bluetooth SIG 的注册商标。

文档中提到的所有商标名称、商标和注册商标均属其各自所有者的财产，特此声明。



芯海科技
CHIPSEA
股票代码:688595