Application Note

使用 2 引脚接口为 TWS 高效充电

Andrew Wallace, Nick Brylski, Gautham Ramachandran, and Albert Lo

摘要

本报告将演示通过 2 引脚接口实现的高效充电方案。本报告将概述允许 TWS 系统在充电模式和通信模式之间切换所需的硬件。通过随耳塞电池电压动态调整充电盒输出电压，可实现高效率。本报告还将介绍充电盒和耳塞内 MCU 的软件流程，并提供使用的所有代码和原理图。

内容

1 引言....2
2 系统概述....2
2.1 充电盒....2
2.2 耳塞....3
3 充电盒算法实现方案....3
3.1 初始化和主代码....3
3.2 UART 中断和输出电压调节....4
4 耳塞算法实现方案....6
4.1 初始化和主代码....6
4.2 中断和传输....7
5 测试步骤....8
6 测试结果....8
6.1 动态电压调节....9
6.2 具有 4.6V 输出的 BQ25619....10
6.3 具有 5V 输出的标准升压....12
7 总结....14
8 原理图....14
9 PCB 布局....17
10 软件....18
10.1 充电盒 main.c....18
10.2 耳塞 main.c....22
11 修订历史记录....26

插图清单

图 2-1. 系统方框图....2
图 3-1. 充电盒算法....4
图 3-2. PWM 算法....5
图 4-1. 耳塞算法....6
图 4-2. 耳塞中断例程....7
图 6-1. 动态电压调节设计的充电周期....9
图 6-2. 动态电压调节设计的热性能....10
图 6-3. BQ25619 4.6V 输出设计的充电周期....11
图 6-4. BQ25619 4.6V 输出设计的热性能....12
图 6-5. 5V 输出设计的充电周期....13
图 6-6. 5V 输出设计的热性能....14
图 8-1. BQ25619 原理图....14
图 8-2. BQ25155 原理图....15
图 8-3. TLV62568P 原理图....15

图 8-4. 充电盒开关原理图 16
图 8-5. 耳塞开关原理图 17
图 9-1. PCB 顶视图 17

表格清单

表 4-1. 耳塞 BQ25155 寄存器 6
表 4-2. 充电盒 BQ25619 寄存器 6
表 6-1. 结果比较 8

商标

1 引言

真无线立体声 (TWS) 客户致力于增加每个满电量充电盒电池可提供的耳塞充电次数，并在充电过程中尽量减少发热，从而提供良好的用户体验。使用线性电池充电器为耳塞电池充电的情况非常常见。因此，为了更大限度地提高充电效率，必须根据当前耳塞电池电压动态更新充电盒输出电压。这样一来，充电盒和电池之间就需要通信接口。除了耳塞电池电压外，该接口还可以将充电状态、器件温度和其他故障等有用信息传回充电盒。

耳塞到充电盒接口所需的引脚数是 TWS 设计中的一个关键机械注意事项。很多耳塞到充电盒通信的实现都需要在每个耳塞上使用三到五个引脚。从机械角度而言，理想状态是使用最少数量引脚，即两个引脚。接下来面临的挑战则是如何在允许充电的同时实现 2 引脚通信接口。

2 系统概述

图 2-1 显示了 2 引脚充电系统的方框图。下文介绍该系统内部各器件的功能和特性。

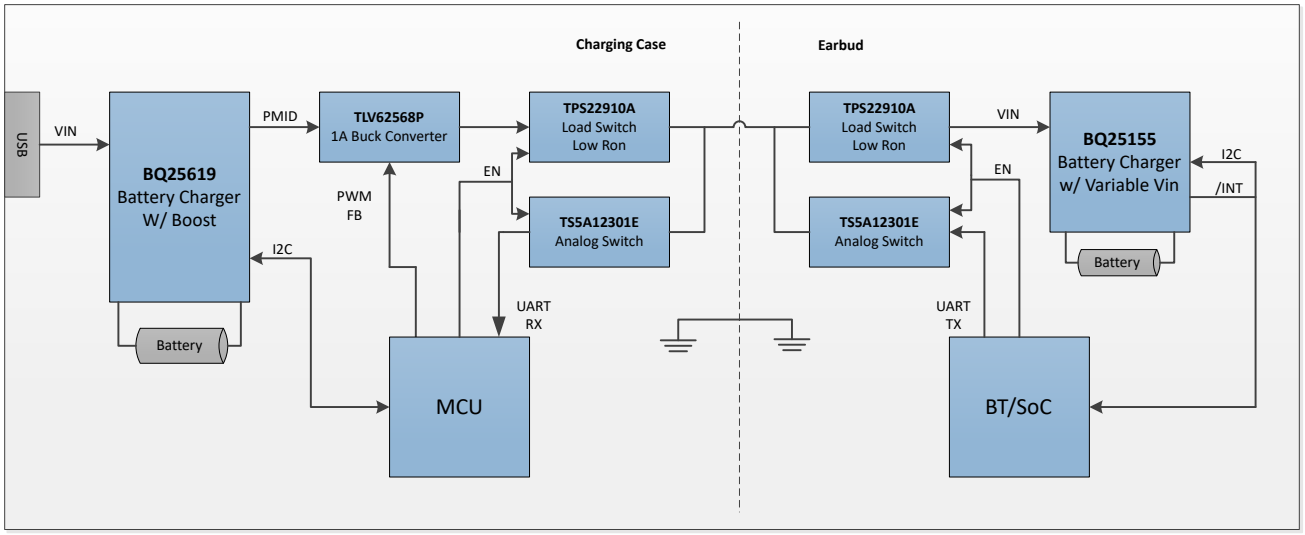

图 2-1. 系统方框图

2.1 充电盒

2.1.1 BQ25619

此器件是一款高效 1.5MHz 同步开关模式降压充电器。它具有集成电源路径特性，可在有 USB 适配器时同时为充电盒电池和耳塞电池充电。当与 USB 适配器断开连接时，BQ25619 可输出 4.6V 至 5.15V 的可选升压。这样就无需在系统中使用单独的升压转换器。在该系统中，升压模式设置为 4.6V，以便以更高的效率进行充电。

2.1.2 TLV62568P

这是一个 1A 降压转换器，用于调节充电盒的输出电压。通过向反馈引脚施加 RC 滤波 PWM 信号来调节此输出电压。此处链接的白皮书概述了为动态电压控制设计反馈系统的过程。该系统使用动态电压控制来更大限度地减小耳塞中线性电池充电器器件的压降。余量的减小可更大限度地降低充电期间耳塞中线性电池充电器的功率耗散和升温。

2.1.3 TPS22910A

这是一个用于控制充电盒输出电源轨的负载开关。当系统处于通信模式时，它还将降压转换器的输出电容与信号线路隔离开来。这是一个低电平有效负载开关，无快速输出放电 (QOD)。该器件允许电源路径为系统默认设置，因此充电盒或耳塞中的电池电量耗尽不会使系统砖化。

2.1.4 TS5A12301E

此器件是用于在耳塞和充电盒之间传递 UART 通信的模拟开关。此器件可将逻辑信号切换至 1.2V，并具有断电保护特性。断电保护特性使得器件能够在器件处于充电模式时保护 MCU 上的 UART 引脚。不具有断电保护特性的模拟开关器件不适用于此应用，因为当耳塞输入电压介于 3.3V 和 4.6V 之间时，会超过很多开关的 I/O 额定电压 Vcc + 0.5V。若不具有此特性，超过此额定值可能会导致 MCU GPIO 引脚上发生过压事件。

2.1.5 MCU

在本文所述的系统中，使用 MSP430FR5529 MCU 控制充电盒处理功能。此应用要求 MCU 具有一个 UART 通道和一个 GPIO 以启用通信方案。为了控制 BQ25619，需要一个 I^2C 通道，并且还可以将五个 GPIO 引脚连接到 BQ25619 以实现器件的输入和控制。为了控制 TLV62568P，需要一个启用 PWM 的 GPIO 引脚来动态控制器件的输出，还可以使用一个额外的 GPIO 来读取器件的 PMID_GOOD 引脚。

节 10.1 中显示了充电盒的软件。

2.2 耳塞

2.2.1 BQ25155

此器件是一款内置 ADC 的线性锂离子电池充电器。它允许在充电时提供可变输入电压。此特性可支持实现本报告中概述的高效充电方案。该器件具有内置 ADC，可监控多个引脚，包括电池电压 (VBAT)、电池温度 (TS)、外部 ADC 通道 (ADCIN)。还有三个比较器可实现多种不同的功能，包括此处的应用手册中介绍的过热电池放电功能。该器件还具有一个中断引脚，可在许多不同标志上触发 MCU 中断。这个基于标志的中断系统用于实现本文概述的中断驱动型通信。为了更大限度延长耳塞的货架期，BQ25155 运输模式具有 10nA 的超低电流消耗。空间受限型应用可从 BQ25155 的 12mm^2 解决方案尺寸中受益。对于此应用，BQ25155 的 VINDPM 功能被禁用，以允许在低 VIN 下充电。

2.2.2 TPS22910A

这是一个用于控制耳塞输入电源轨的负载开关。当系统处于通信模式时，它还会将 BQ25155 的输入电容与信号线路隔离开来。这是一个低电平有效负载开关，无快速输出放电 (QOD)。具有 QOD 的器件不能用于此应用，因为它会通过放电电阻将通信线路短接至接地。该器件的低电平有效特性允许电源路径为系统默认值，因此充电盒或耳塞中的电池电量耗尽不会使系统砖化。

2.2.3 TS5A12301E

此器件是用于在耳塞和充电盒之间传递 UART 通信的模拟开关。此器件可将逻辑信号切换至 1.2V，并具有断电保护特性。断电保护特性使得器件能够在器件处于充电模式时保护 MCU 上的 UART 引脚。不具有断电保护特性的模拟开关器件不适用于此应用，因为当耳塞输入电压介于 3.3V 和 4.6V 之间时，会超过很多开关的 I/O 额定电压 Vcc + 0.5V。若不具有此特性，超过此额定值可能会导致 MCU GPIO 引脚上发生过压事件。

2.2.4 BT/SOC

在本文所述的系统中，使用 MSP430FR5529 MCU 控制耳塞处理功能。在 TWS 应用中，该插座通常是具有音频流功能的蓝牙 SOC。此应用要求器件具有一个 UART 通道、一个 I^2C 通道、一个启用中断的 GPIO 和一个用于开关控制的 GPIO，以启用通信方案。为了控制 BQ25155，可使用三个额外的 GPIO 引脚。

节 10.2 中显示了耳塞的软件。

3 充电盒算法实现方案

3.1 初始化和主代码

对于此通信方案，充电盒充当主器件，耳塞充当从器件。图 3-1 显示了充电盒算法的流程图。

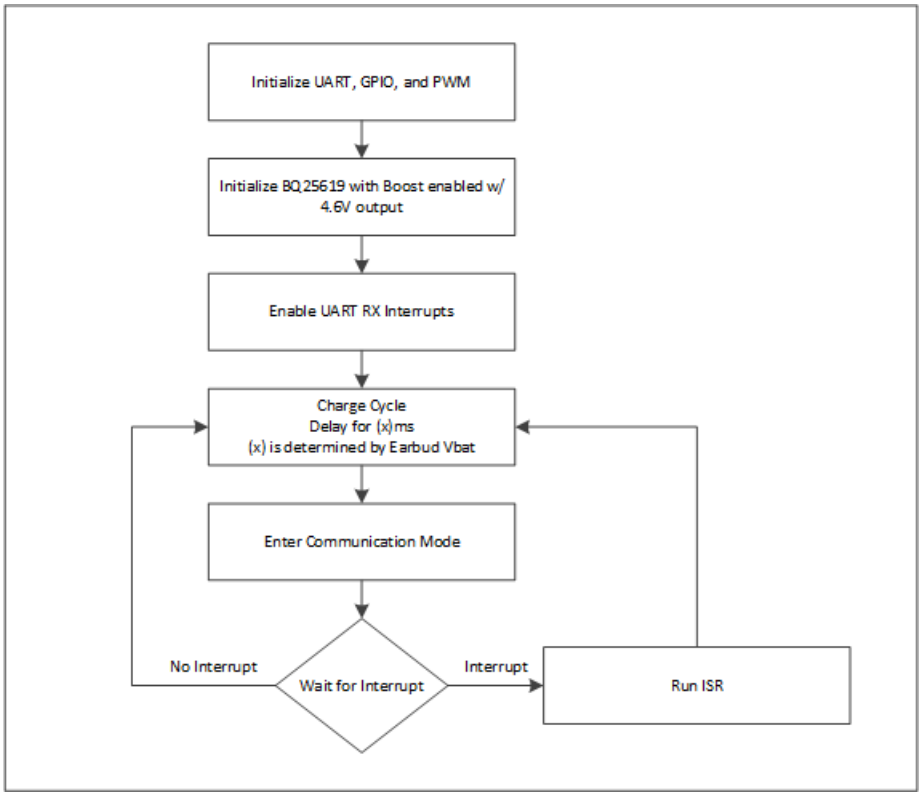

图 3-1. 充电盒算法

首先初始化充电盒以满足用户的系统要求。这里的设置包括 BQ25619 的输出电压和电流、PWM 频率、启动 PWM 占空比、UART 初始化等。初始化后，系统的中断被启用。此时，系统准备好在插入耳塞后开始充电周期。

检测到耳塞时，便会开始充电周期。在用户设置的时间内，充电盒将输出 4.6V 电压。这是为了确保耳塞电池不会耗尽，并且器件可以在通信周期内做出响应。在用户设置的时间过后，充电盒将通过禁用 TPS22910A 负载开关和启用 TS5A12301E UART 开关来启动通信周期。然后，充电盒将等待来自耳塞的 UART 传输触发中断。

这些通信周期以用户定义的间隔发生，此间隔应基于电池的充电曲线。充电周期开始时，预计已放电的电池将在 3.0V 至 3.7V 的范围内。这将取决于电池放电的深度以及客户为电池切断电压选择的设置。此时，耳塞电池的电压将快速变化，因此通信间隔之间的时间需要很短（约为 5 秒）。随着耳塞电池电压升高，电压将以较慢的速度变化，从而增大间隔，更大限度地缩短系统需要处于通信模式的时间。

3.2 UART 中断和输出电压调节

当 UART RX 中断被触发时，该中断将检查是否已接收到充电完成字节。如果充电已完成，MCU 会将充电盒设置为低功耗模式并等待系统状态更改。如果未接收到充电完成字节，MCU 将存储接收到的耳塞电池电压并继续中断。

然后，将使用新接收到的耳塞电池电压计算 PWM 占空比。此计算基于连接到 TLV62568P 反馈引脚的电阻分压器和滤波器。可点击此处获取 Excel 计算器*使用 DAC 进行输出电压调节的设计工具*，用于计算这些值。

$$PWMDuty = \left(7.607 - \left(1.434 \times \left(Vbat + 0.33\right)\right)\right) \times \frac{40}{3.3} \quad (1)$$

需要对上述计算进行调整，以便产生比当前耳塞电池电压高出约 200mV 的输出电压，满足耳塞电池充电器所需的余量。完成此计算后，会限定 PWM 占空比值以防止施加低于 3V 或高于 4.5V 的电压。如果发现该值正常，MCU 将调整 PWM 占空比，输出电压将随之调整。

关闭模拟开关并且重新激活负载开关即可再次进入充电模式。然后，中断将终止。

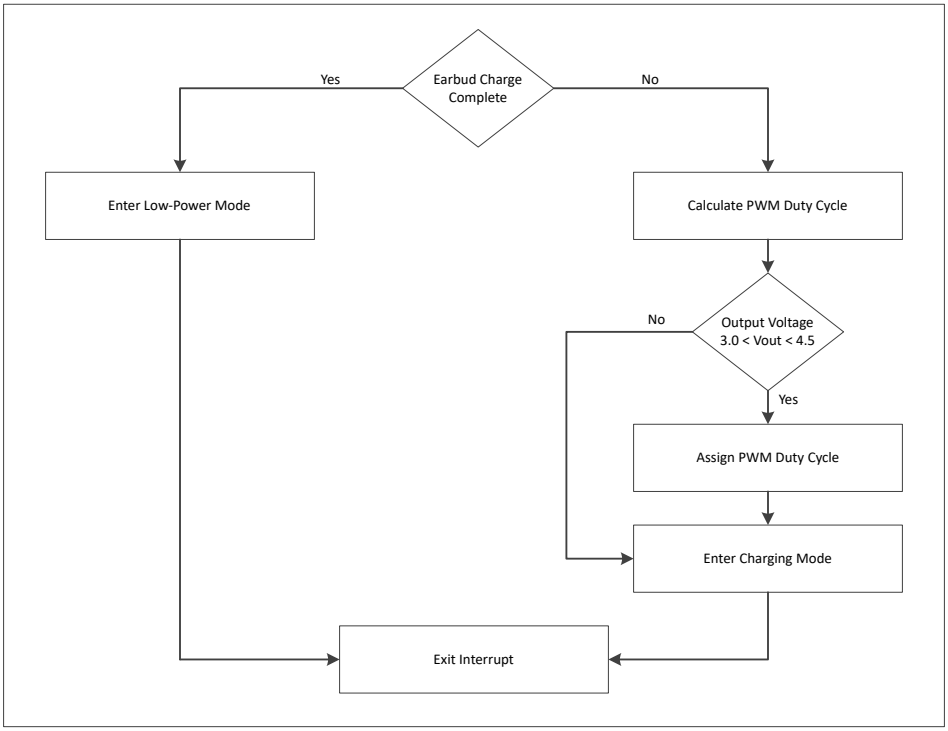

图 3-2. PWM 算法

4 耳塞算法实现方案

4.1 初始化和主代码

对于此系统实现的通信，耳塞充当从器件。这可防止耳塞在充电盒处于电源模式时进入通信模式。仅当耳塞检测到 VIN_PGOOD_FLAG（寄存器地址 0x3）已置位且 BQ25155 的内部 ADC 读数为 0V 时，才会进入通信模式并发送消息。这表示充电盒已进入通信模式。图 4-1 显示了耳塞算法的流程图。

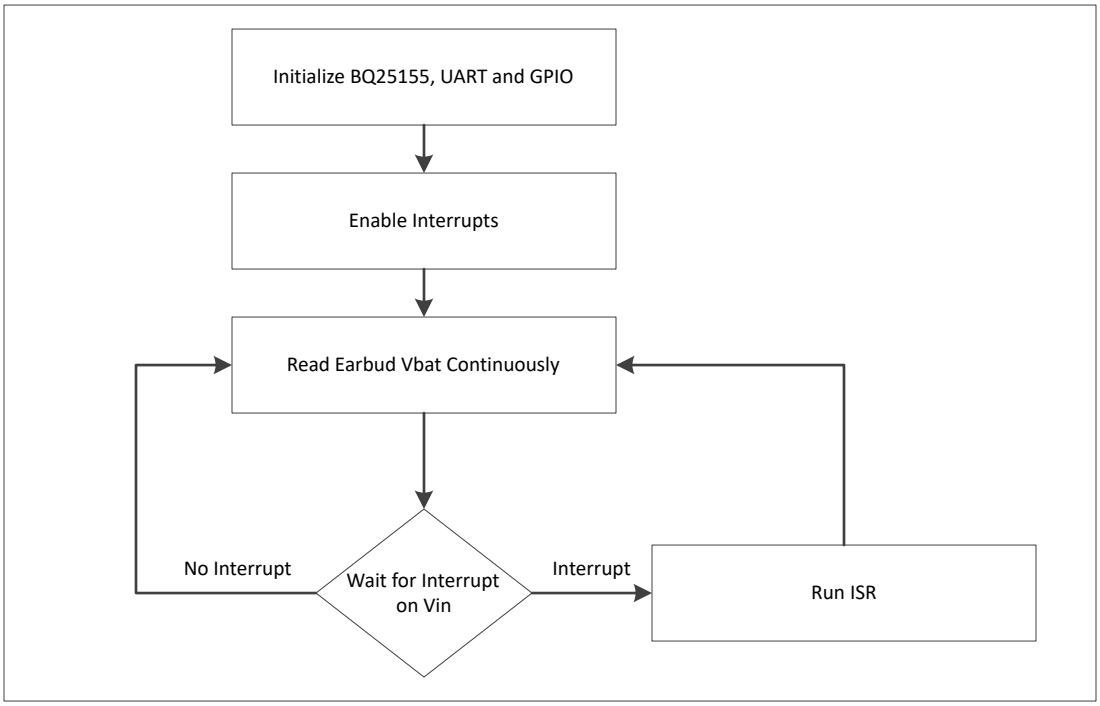

图 4-1. 耳塞算法

首先初始化耳塞以满足用户的系统要求。这里的设置包括 BQ25155 的充电电流、ADC 转换速率等。初始化后，系统中断被启用。此时，系统已准备好在连接到充电盒后启动充电周期。

以下两个表显示了已修改的寄存器。

表 4-1. 耳塞 BQ25155 寄存器

名称	容值	用途
ICHG_CTRL	0x50	将 ICHG 设置为 100mA
CHARGERCTRL0	0x92	禁用看门狗计时器
ADCCTRL0	0x58	将 ADC 设置为每次转换 3ms 时的连续读取
ADCCTRL1	0x00	禁用比较器
ADC_READ_EN	0xFE	启用 ADC 读取通道

表 4-2. 充电盒 BQ25619 寄存器

名称	容值	用途
REG01（充电器控制 0）	0x3A	启用升压模式
REG05（充电器控制 1）	0x8E	禁用看门狗计时器
REG06（充电器控制 2）	0xC6	将升压电压设置为 4.6V

系统初始化并连接到充电盒后，它将开始充电并等待响应充电盒触发的通信周期。在等待通信周期时，耳塞每隔 0.5 秒存储一次电池电压。这样做的原因在于，当 Vin 变为 0 从而触发通信周期时，耳塞电池电压将轻微下降，在此期间读取读数将导致传输的电压低于所需的充电电压。

当充电盒进入通信模式时，输入电压将降至 0，这将导致 BQ25155 设置一个标志，以指示 Vin 已降至可接受的电压以下并在其 INT 引脚上触发中断。这将触发 ISR。

4.2 中断和传输

当 MCU 进入 ISR 时，它将首先检查中断是否由 VIN_PGOOD_FLAG 引起。之所以进行此检查，是因为 BQ25155 具有很多它可以设置的其他可中断标志，对于此应用，我们仅使用 Vin 标志。最终用户可以选择为 BQ25155 设置的其他标志采取不同的操作。

如果 Vin 标志已置位，BQ25155 的内部 ADC 将用于限定中断。这是通过每 3ms 读取一次 250ms 的 Vin 并比较最近的三个值来完成的。如果在 250ms 内未找到三个连续值来确认中断，则中断将超时

如果中断被限定，则对充电完成寄存器进行检查。如果充电已完成，将发送充电完成位。如果充电未完成，则在主循环中读取的耳塞电池电压将通过 UART 进行传输。禁用耳塞负载开关并启用模拟开关即可完成传输。然后，数据将被推入 UART 发送缓冲区并发送到充电盒。然后，通过切换开关，耳塞将立即重新进入充电模式。这可防止在充电盒重新进入充电模式以响应通信时意外为耳塞逻辑引脚供电。执行此步骤后，耳塞将返回主循环。

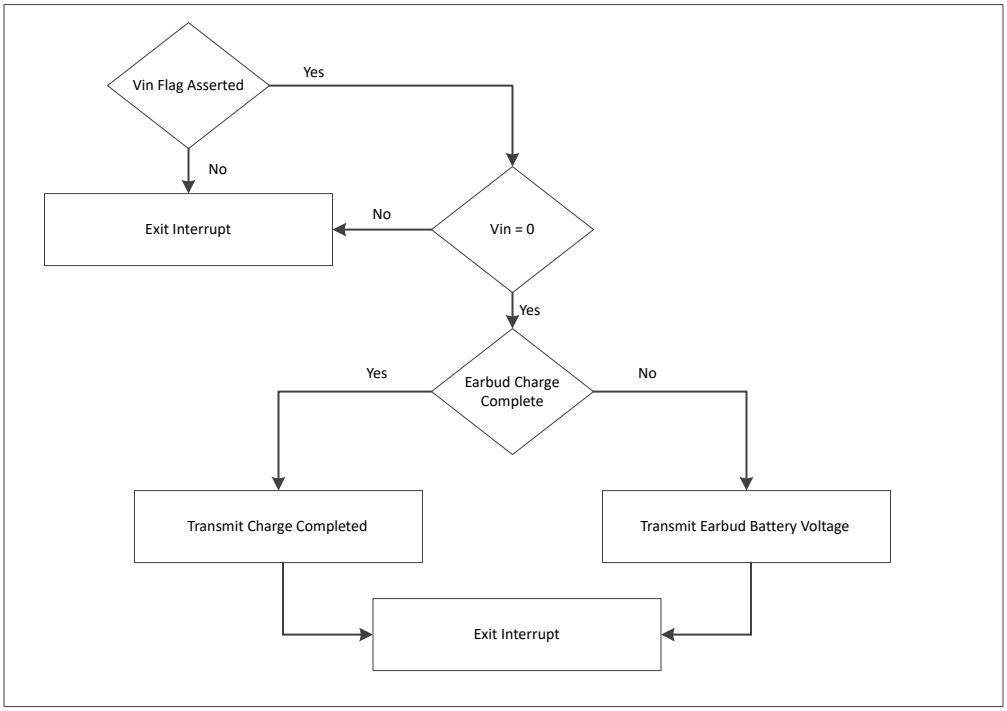

图 4-2. 耳塞中断例程

5 测试步骤

这些结果是通过使用 200mAh 快速充电电流对 195mAh 耳塞电池执行完整充电周期获得的。电池盒由充满电的 400mAh 电池供电。效率数据是通过测量瞬时输入电压和电流以及输出电压和电流来计算的。这些值分别在电池盒电池和耳塞电池的端子处测量。获得瞬时电压和电流后，将它们相乘即可得出系统的瞬时输入和输出功率。然后对整个充电周期内的瞬时功率进行积分，以得出系统的总输入和输出能量。

该数据是使用示波器收集的，示波器在大约 83 分钟的整个充电周期内的采样率为 500S/s。电压数据直接在每个电池的端子上测得的。电流数据是使用 INA240 器件在与电池和系统输入/输出端子串联的 10mΩ 检测电阻上测得的。

热数据是使用 FLIR 热像仪获取的。环境室温为 25°C，没有为系统提供直接通风。

6 测试结果

本节通过展示三种不同 TWS 充电系统的充电效率和散热来呈现各充电系统的性能。下面的表 6-1 提供了结果的简单比较。

表 6-1. 结果比较

	动态电压调节	BQ25619 4.6V 输出	标准升压 5V 输出
输出电压	3V 至 5V（PWM 可调节）	4.6V	5V
充电效率	84.7%	83.2% (-1.5%)	76.3% (-8.4%)
耳塞发热	27.7°C	31.9°C (+15.2%)	33.3°C (+20.2%)
充电盒电池持续时间	最好	较好	基线

6.1 动态电压调节

上述第一个系统即为本文重点讨论的系统。此设计使用支持 2 引脚通信的高效充电方案，提供了出色的性能。通过减小充电盒输出电压与耳塞电池电压之间的差异，线性电池充电器耗散的功率会降低。这样可实现出色的热性能并延长充电盒电池寿命。

下面是使用建议的解决方案对充电周期进行的示波器捕获。充电盒输入端的最大电流为 265mA，同时为耳塞电池提供 200mA 的电流。可以看到的每个脉冲都是一个通信周期。

图 6-1. 动态电压调节设计的充电周期

图 6-2 显示了 FLIR 热像仪拍摄的图像。该图像显示了充电过程中系统处于最高温度的情况。27.7°C 的温度仅比环境室温 25°C 高 2.7°C。

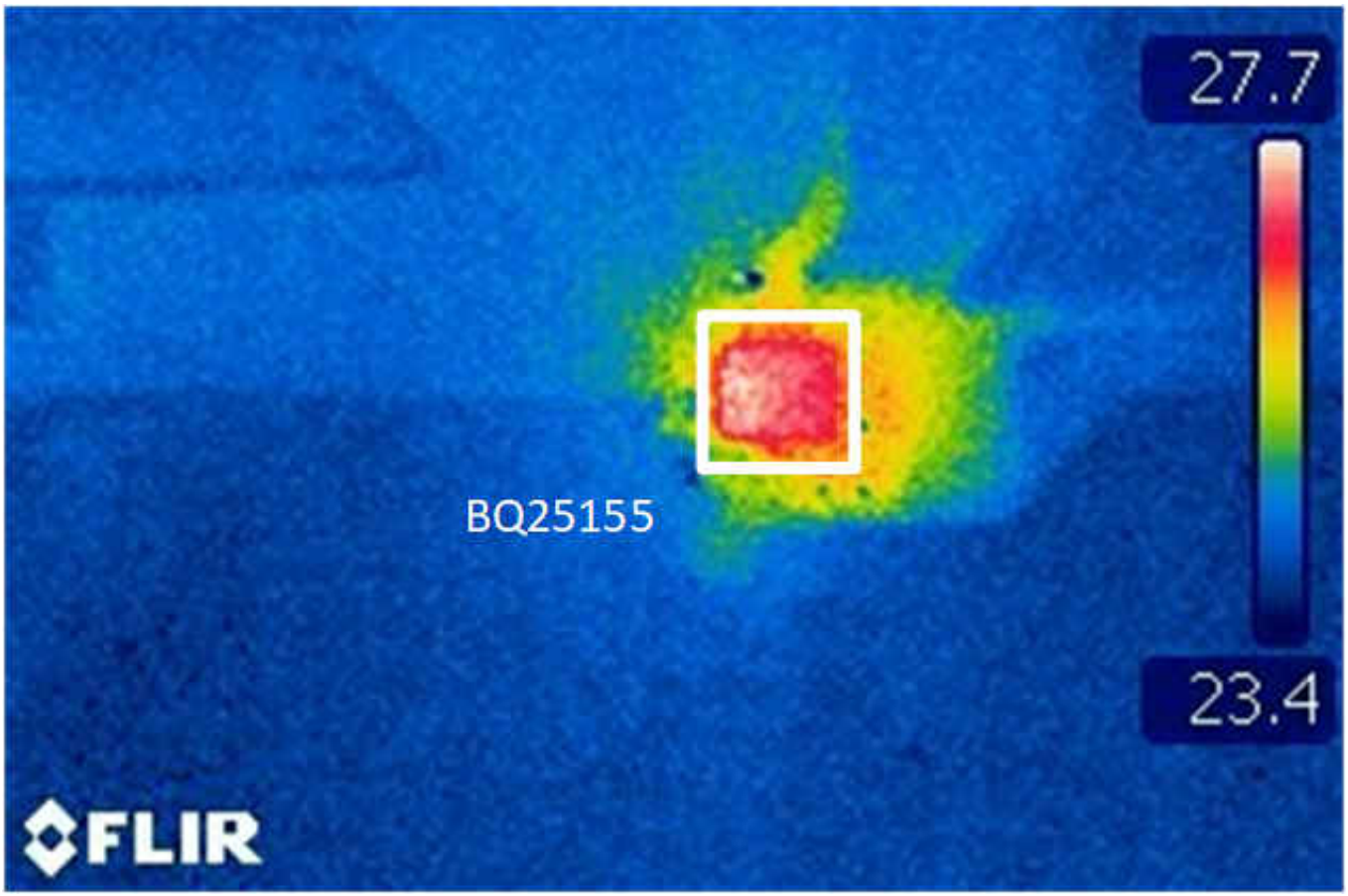

图 6-2. 动态电压调节设计的热性能

6.2 具有 4.6V 输出的 BQ25619

本文采用的第二个系统使用 BQ25619 的升压转换器，输出为 4.6V。该解决方案的效率高于具有 5V 中间电压的系统，但热性能并不理想。

图 6-3. BQ25619 4.6V 输出设计的充电周期

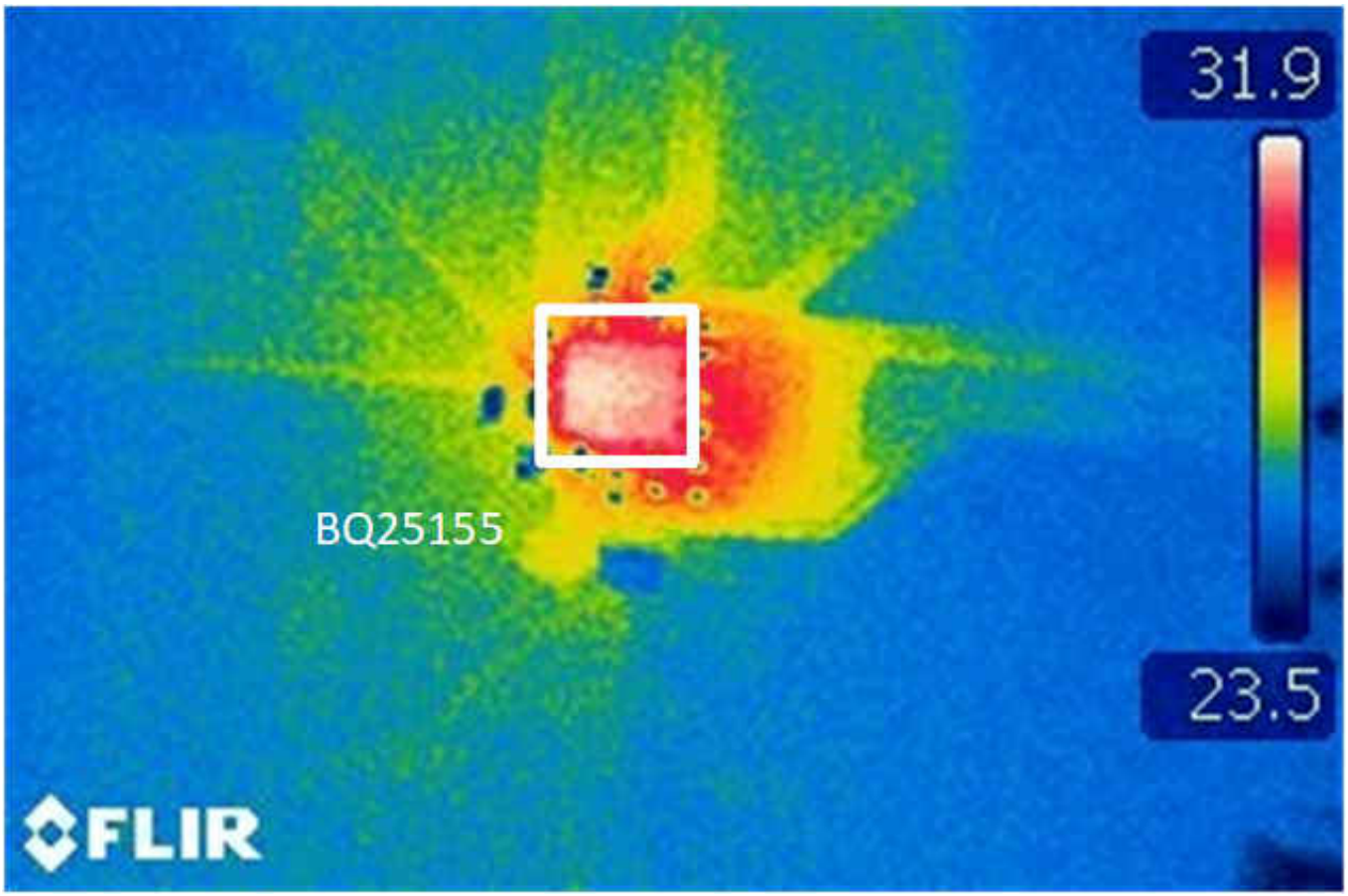

图 6-4. BQ25619 4.6V 输出设计的热性能

6.3 具有 5V 输出的标准升压

第三个系统是基线充电盒，它在充电盒输出端施加标准 5V 电压。该系统在效率和散热方面的性能欠佳。

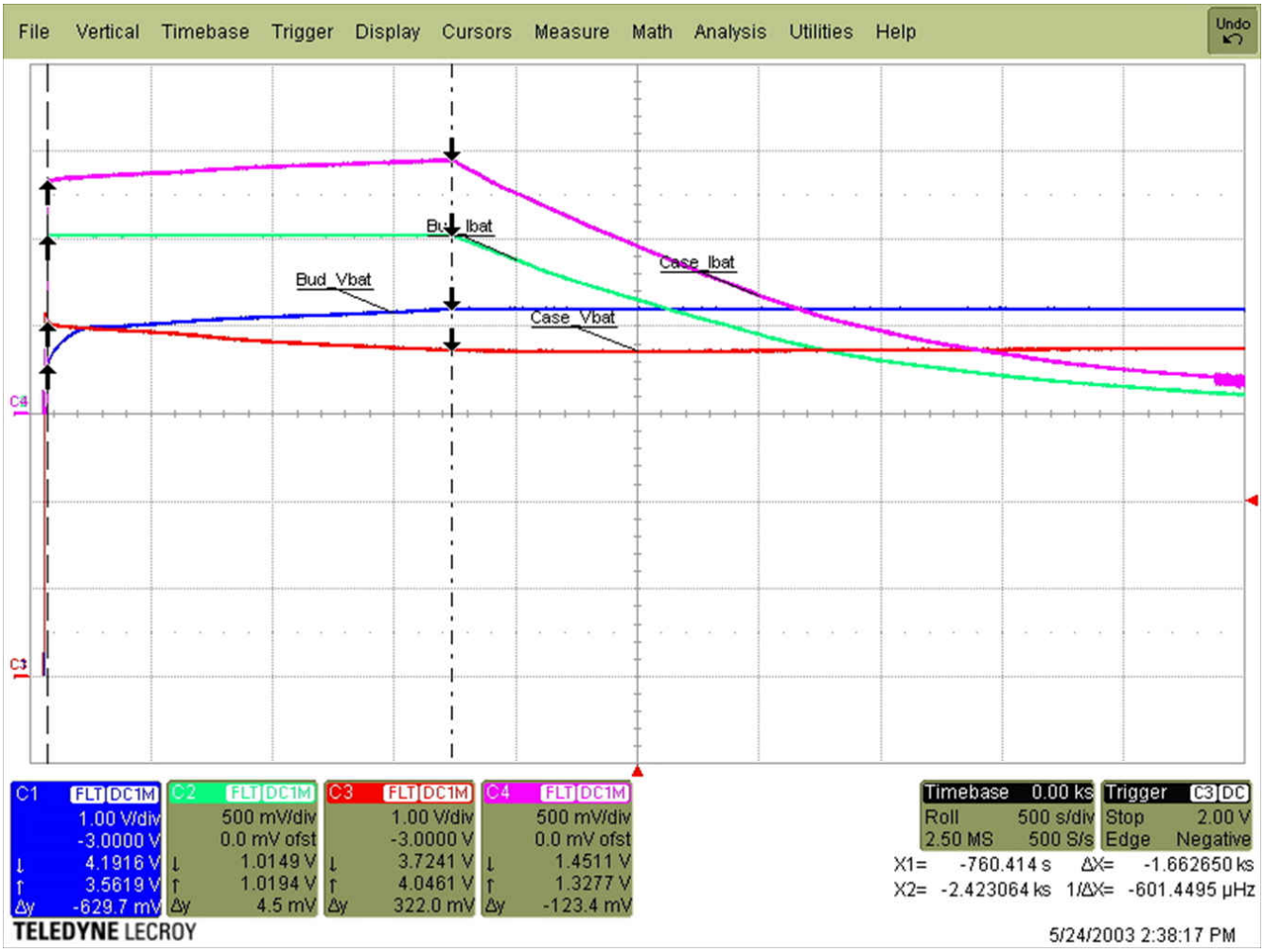

图 6-5. 5V 输出设计的充电周期

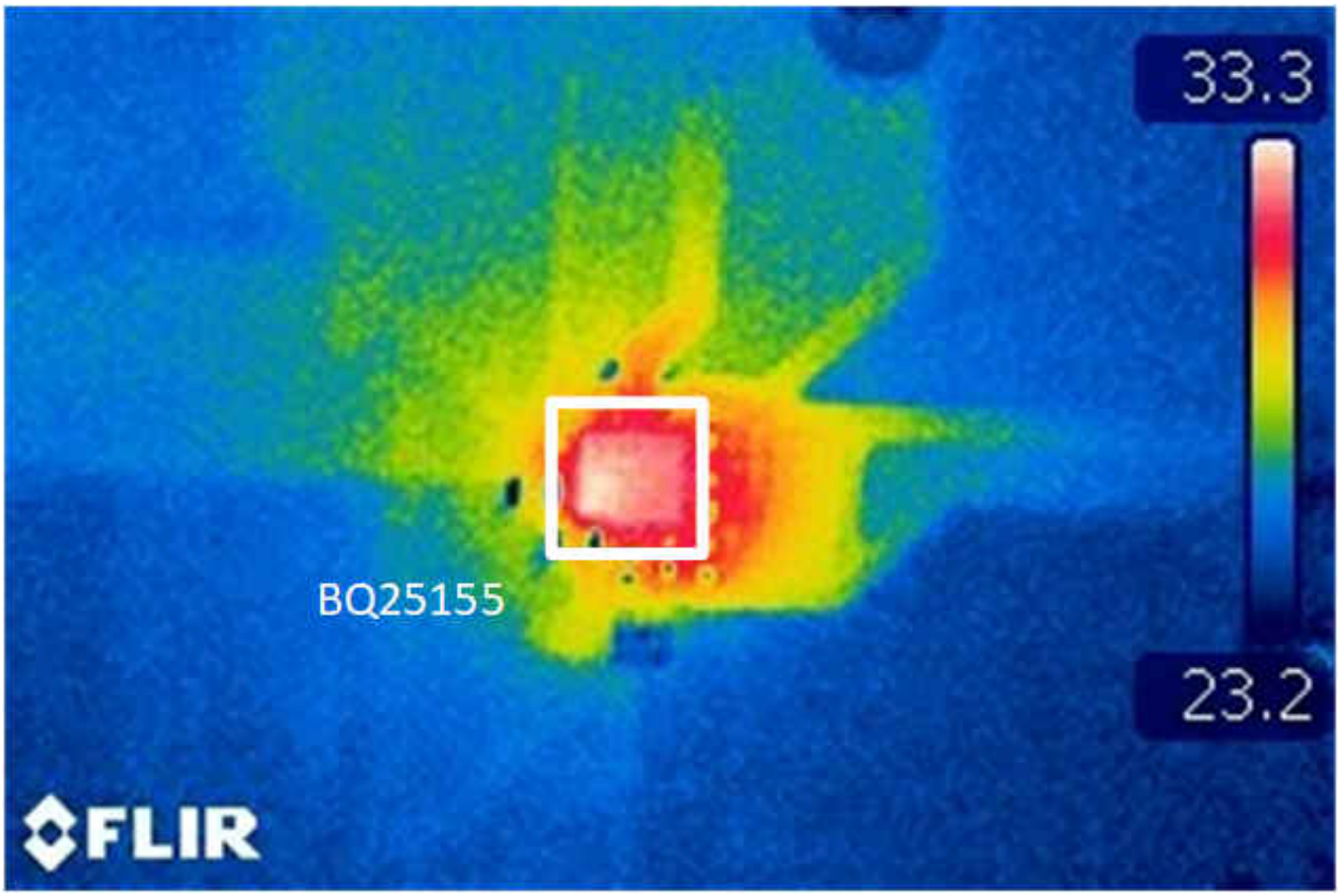

图 6-6. 5V 输出设计的热性能

7 总结

测试结果表明，TI 设计具有显著的散热优势。这是通过动态调整充电盒内 BQ25619 的调节电压来实现的。在本报告中，我们展示了如何通过节省空间的 2 引脚接口实现这一点。有关 TWS 系统的更多 TI 文档，请点击此处。

8 原理图

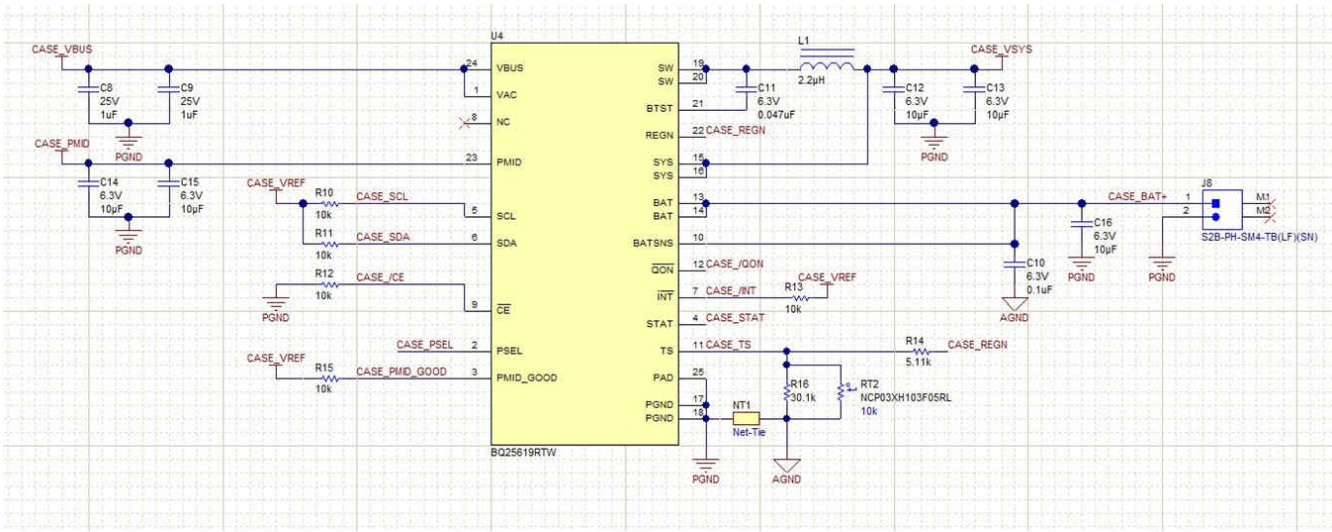

图 8-1. BQ25619 原理图

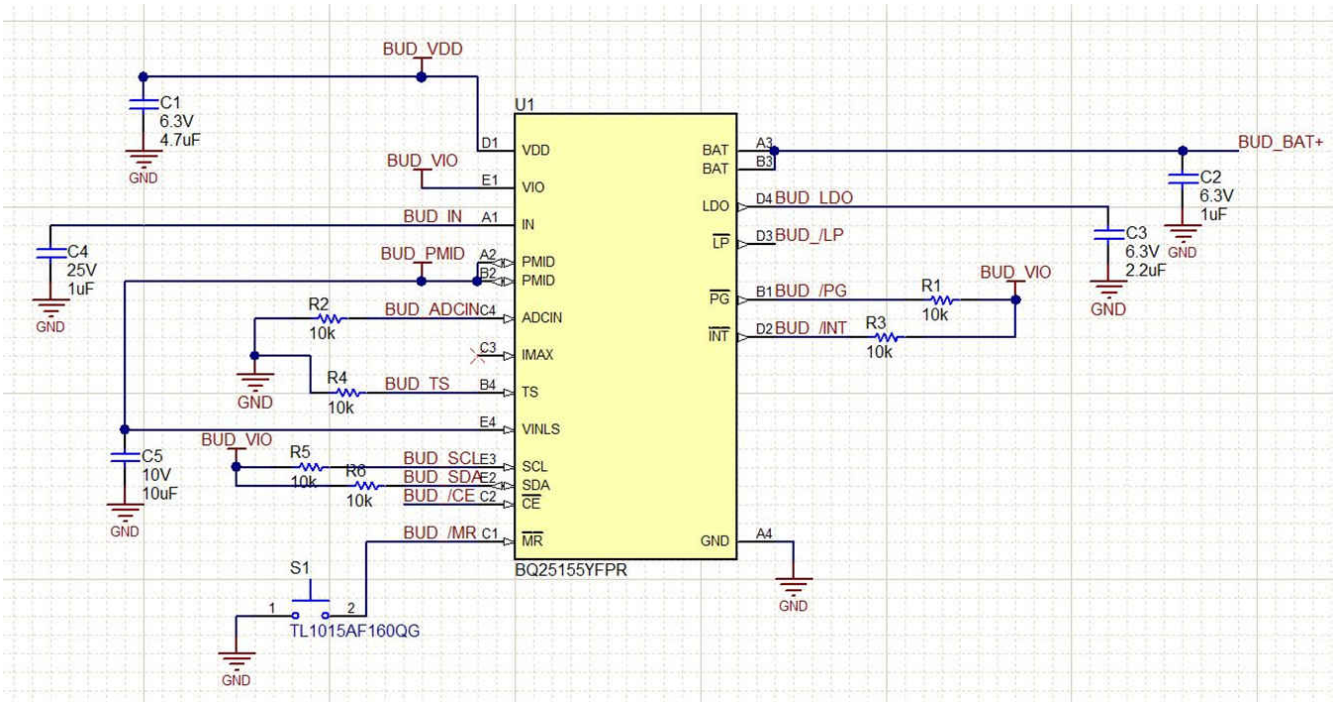

图 8-2. BQ25155 原理图

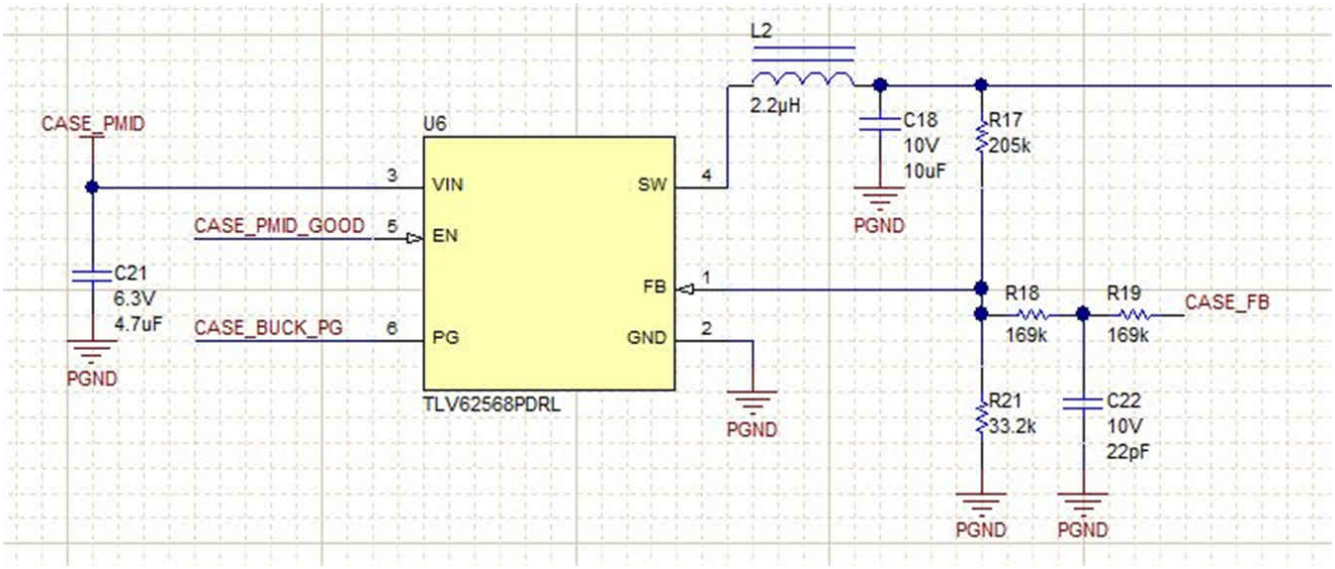

图 8-3. TLV62568P 原理图

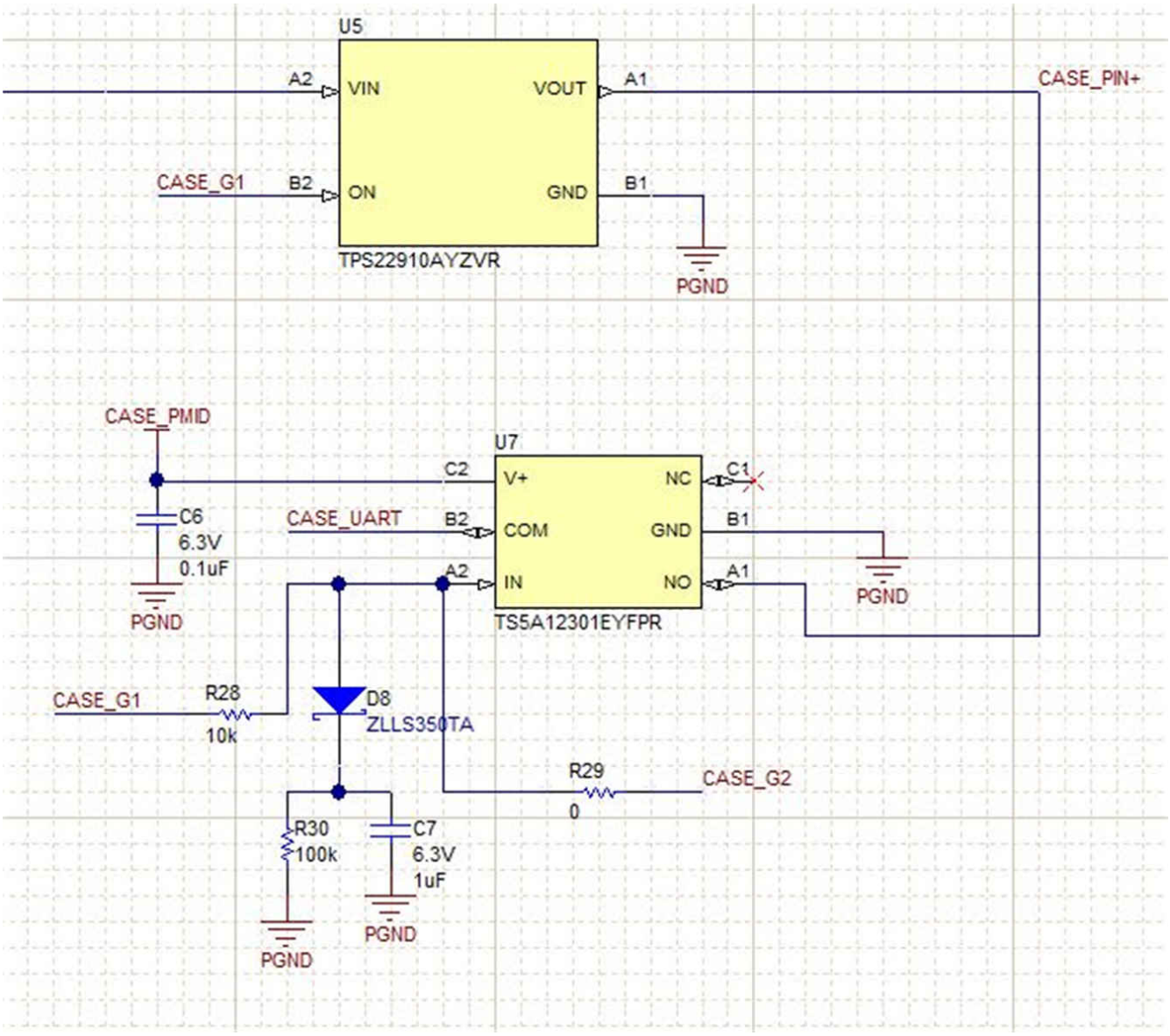

图 8-4. 充电盒开关原理图

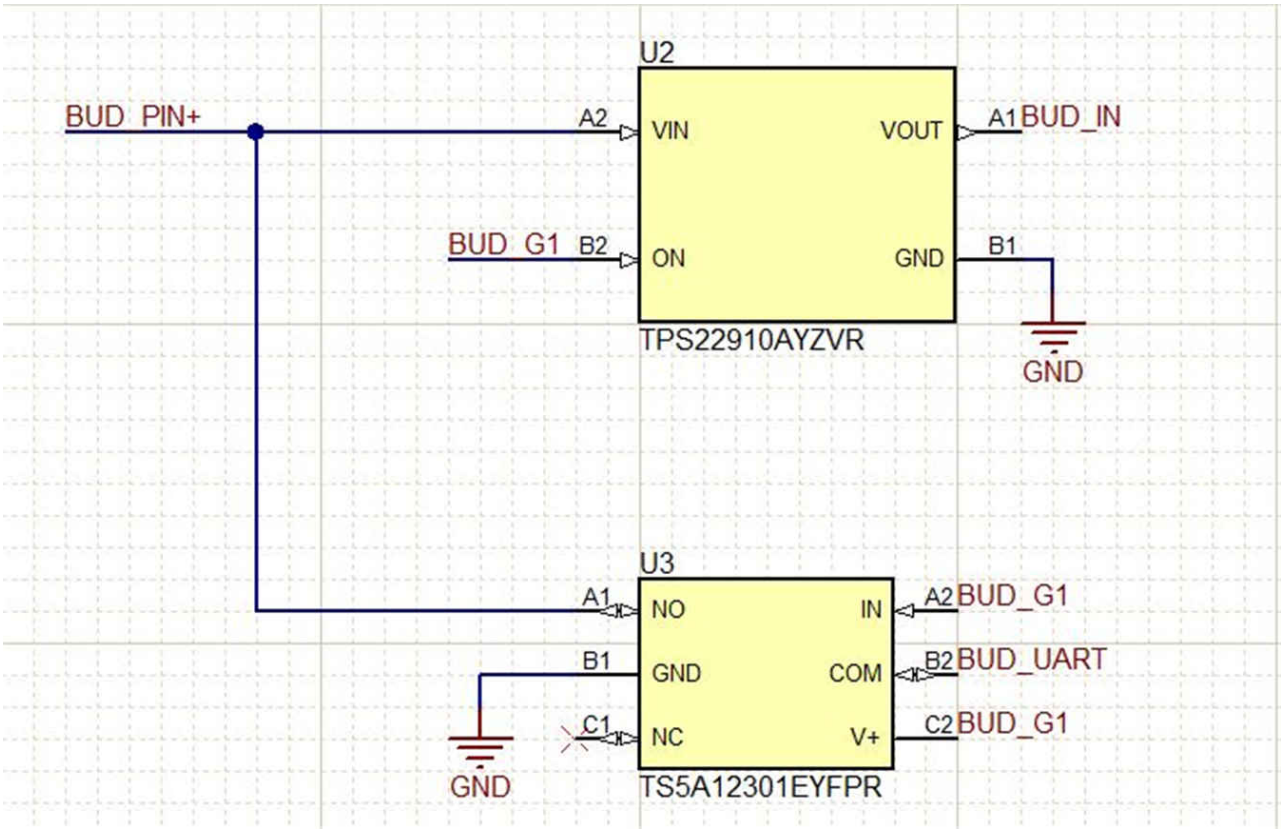

图 8-5. 耳塞开关原理图

9 PCB 布局

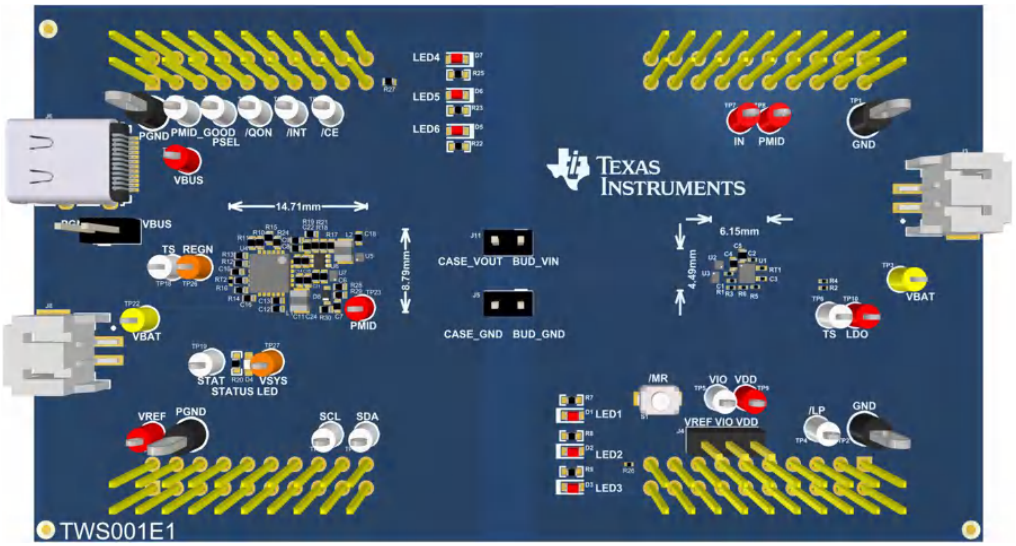

图 9-1. PCB 顶视图

10 软件

10.1 充电盒 main.c

```
/* --COPYRIGHT--,BSD
 * Copyright (c) 2016, Texas Instruments Incorporated
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * *  Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * *  Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * *  Neither the name of Texas Instruments Incorporated nor the names of
 *    its contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
 * OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
 * OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 * --/COPYRIGHT--*/
/*
 * ======== main.c ========
*/
#include <string.h>
#include <stdint.h>
#include <inttypes.h>
#include <string.h>
#include <stdio.h>
#include "board_functions.h"
#include <stdlib.h>
#include "driverlib.h"
#include "StdI2C.h"
#include "BQ25150.h"
//#include "board_timer.h"
#include "USB_config/descriptors.h"
#include "USB_API/USB_Common/device.h"
#include "USB_API/USB_Common/usb.h"                 // USB-specific functions
#include "USB_API/USB_CDC_API/UsbCdc.h"
#include "USB_app/usbConstructs.h"
#include "OLED/Oled_SSD1306.h"
#include "StdPollI2C.h"
//#include <Wire.h>
/*
 * your own board.
 */
#include "hal.h"
//#include "Energia.h"
//#include "pins_energia.h"
// Function declarations
uint8_t retInString(char* string);
void initTimer(void);
void setTimer_A_Parameters(void);
// Global flags set by events
volatile uint8_t bCDCDataReceived_event = FALSE; // Indicates data has been rx'ed
// without an open rx operation
char str[50];
#define NUM_SAMPLES 8
#define clkspeed 3          // 24Mhz Clock Select
//#define clkspeed 1        // 8Mhz Clock Select
short samples[NUM_SAMPLES] ;
int sample_index = 0 ;
char str[50];
char cmdstring[5];
```

```
char cs2[3];
uint8_t response = 0;
//unsigned char buffer[10]={1,2,3,4,5,6,7,8,9,10};
volatile char wholeString[MAX_STR_LENGTH] = "";
volatile uint8_t modecounter = 0;
volatile uint8_t pluscounter = 0;
///Timer_A_initUpModeParam Timer_A_params = {0};
int i;
unsigned char* PRxData;                     // Pointer to RX data
unsigned char RXByteCtr;
volatile unsigned char RxBuffer[128];       // Allocate 128 byte of RAM
unsigned char* PTxData;                     // Pointer to TX data
unsigned char TXByteCtr;
const unsigned char TxData[] =              // Table of data to transmit
{
 0x11,
 0x22,
 0x33,
 0x44,
 0x55
};
volatile uint8_t Button_Released = 0;
unsigned int result;
const char* hexstring = "0xabcdef0";
int raddr;
int rdata;
char buf[5];
uint8_t uraddr;
uint8_t urdata;
uint16_t Err;
// Holds the outgoing string
char outString[MAX_STR_LENGTH] = "";
uint8_t connectedflag = 0;
uint8_t echoflag = 1;
int ubtncounter = 0;
uint8_t RegValuesL = 0;
uint8_t RegValuesM = 0;
uint8_t RegValueMSB = 0;
uint8_t ADCCheck = 0;
uint32_t stobusf;
char vBat;
char vBatString;
uint32_t stobuf;
uint32_t stobuf1;
double stobuf2;
double PwmDuty;
uint8_t RegValues = 0;
double batteryVoltageCheck = 0;
//__delay_cycles(1000); 1000 = 100us
uint8_t rthex;
// Set/declare toggle delays
//uint16_t SlowToggle_Period = 20000 - 1;
//uint16_t FastToggle_Period = 1000 - 1;
//  ======== main ========
void main(void)
{
    WDT_A_hold(WDT_A_BASE); // Stop watchdog timer
    // Minumum Vcore setting required for the USB API is PMM_CORE_LEVEL_2 .
    PMM_setVCore(PMM_CORE_LEVEL_2);
    USBHAL_initPorts();           // Config GPIOS for low-power (output low)
    USBHAL_initClocks(8000000 * clkspeed);   // Config clocks.MCLK=SMCLK=FLL=8MHz; ACLK=REFO=32kHz
    initTimer();            // Prepare timer for LED toggling
    initI2C();
//  ======== UART Setup ========
    GPIO_setAsInputPinWithPullDownResistor(GPIO_PORT_P3,GPIO_PIN4); // P3.4 = Input With pulldown
    P3SEL = BIT3+BIT4;                          // P3.4,5 = USCI_A0 TXD/RXD
    UCA0CTL1 |= UCSWRST;                        // **Put state machine in reset**
    UCA0CTL1 |= UCSSEL_2;                       // SMCLK
    UCA0BR0 = 6;                                // 1MHz 9600 (see User's Guide)
    UCA0BR1 = 0;                                // 1MHz 9600
    UCA0MCTL = UCBRS_0 + UCBRF_13 + UCOS16;     // Modln UCBRSx=0, UCBRFx=0,
                                                // over sampling
    UCA0CTL1 &= ~UCSWRST;                       // **Initialize USCI state machine**
//  ======== GPIO Setup ========
    GPIO_setAsOutputPin(LED_4);
    GPIO_setOutputLowOnPin(LED_4);
    GPIO_setAsInputPin(CASE_PMID_GOOD);
    GPIO_setAsInputPinWithPullUpResistor(CASE_INT);
    GPIO_setAsInputPin(CASE_BUCK_PG);
```

```
    GPIO_setAsInputPinWithPullUpResistor(CASE_START);
    GPIO_setAsOutputPin(CASE_FB);
    GPIO_setOutputLowOnPin(CASE_FB);
    GPIO_setAsOutputPin(CASE_PSEL);
    GPIO_setOutputLowOnPin(CASE_PSEL);
    GPIO_setAsOutputPin(CASE_QON);
    GPIO_setOutputHighOnPin(CASE_QON);
    GPIO_setAsOutputPin(CASE_CE);
    GPIO_setOutputLowOnPin(CASE_CE);
    GPIO_setAsOutputPin(CASE_G1);
    GPIO_setOutputLowOnPin(CASE_G1);
    GPIO_setAsOutputPin(LED_5);
    GPIO_setOutputLowOnPin(LED_5);
//    GPIO_setAsOutputPin(LED_6);
//    GPIO_setOutputLowOnPin(LED_6);
//  ======== PWM Setup ========
    P2DIR |= BIT5; //Set pin 2.5 to the output direction.
    P2SEL |= BIT5; //Select pin 2.5 as our PWM output.
    TA2CCTL2 = OUTMOD_7;
    TA2CCR0 = 40; //Set the period in the Timer A0 Capture/Compare 0 register to 40 us.
    TA2CCR2 = 14; //The period in microseconds that the power is ON, default 14 = 35%, ~= 4.5v
OUTPUT
    TA2CTL = (TASSEL_2 | MC_1); //TASSEL_2 selects SMCLK as the clock source, and MC_1 tells it to
count up to the value in TA0CCR0.
 //  ======== BQ25619 Register Setup ========
    StdI2C_P_TX_Single(BQ25619_ADDR, BQ25619_REG01, 0x3A, &Err); // BST_CONFIG = 1, Boost mode
Enable, REG01 = 01h, value = 0x3A
    StdI2C_P_RX_Single(BQ25619_ADDR, BQ25619_REG01, &RegValues, &Err);
    StdI2C_P_TX_Single(BQ25619_ADDR, BQ25619_REG05, 0x8E, &Err); // Disable Watchdog
    StdI2C_P_RX_Single(BQ25619_ADDR, BQ25619_REG05, &RegValues, &Err);
    StdI2C_P_TX_Single(BQ25619_ADDR, BQ25619_REG06, 0xC6, &Err); // Set Boost voltage to 0xE6 =
5V, 0xC6 for 4.6V
    StdI2C_P_RX_Single(BQ25619_ADDR, BQ25619_REG06, &RegValues, &Err);
    GPIO_setOutputHighOnPin(LED_4);
    waitms(500);
    GPIO_setOutputLowOnPin(LED_4);
    waitms(500);
//  ======== Ready while loop ========
    //This while loop holds the program with the interrupts disabled.This allows synchronization
with the ear bud
    //short BQ_START pin 4.3 to ground to exit loop
    while(GPIO_getInputPinValue(CASE_START) == 1)
    {
        GPIO_toggleOutputOnPin(LED_5);
        __delay_cycles(3000000);
        GPIO_toggleOutputOnPin(LED_5);
        __delay_cycles(3000000);
        GPIO_toggleOutputOnPin(LED_5);
        __delay_cycles(3000000);
        GPIO_toggleOutputOnPin(LED_5);
        __delay_cycles(15000000);
    }
    GPIO_setOutputLowOnPin(LED_5);
//  ======== Interrupt Enables ========
    __enable_interrupt();  // Enable interrupts globally
    UCA0IE |= UCRXIE;
//  ======== Main while loop ========
    //Allows adjustment of the communication cycle time
    while(1)
    {
        GPIO_toggleOutputOnPin(LED_4);
//        __delay_cycles(24000000); //delay 1s
          __delay_cycles(120000000); //delay 5s
//        __delay_cycles(240000000); //delay 10s
//        __delay_cycles(1440000000); //delay 60s
        if(batteryVoltageCheck >= 3.8){
            __delay_cycles(120000000); //delay 5s
        }
        if(batteryVoltageCheck >= 3.85){
            __delay_cycles(240000000); //delay 10s
        }
        if(batteryVoltageCheck >= 3.95){
            __delay_cycles(240000000); //delay 10s
        //    __delay_cycles(240000000); //delay 10s
        //    __delay_cycles(240000000); //delay 10s
        }
        if(batteryVoltageCheck >= 4.05){
            __delay_cycles(1440000000); //delay 60s
        //    __delay_cycles(1440000000); //delay 60s
```

```
        }
        if(batteryVoltageCheck >= 4.19){
            __delay_cycles(1440000000); //delay 60s
            __delay_cycles(1440000000); //delay 60s
            __delay_cycles(1440000000); //delay 60s
            __delay_cycles(1440000000); //delay 60s
//            __delay_cycles(1440000000); //delay 60s
//            __delay_cycles(1440000000); //delay 60s
//            __delay_cycles(1440000000); //delay 60s
        }
        GPIO_setOutputHighOnPin(CASE_G1);      //enter communication mode
    }
}
/*
 * ======== TIMER1_A0_ISR ========
 */
#if defined(__TI_COMPILER_VERSION__) || (__IAR_SYSTEMS_ICC__)
#pragma vector=TIMER0_A0_VECTOR
__interrupt void TIMER0_A0_ISR (void)
#elif defined(__GNUC__) && (__MSP430__)
void __attribute__ ((interrupt(TIMER0_A0_VECTOR))) TIMER0_A0_ISR (void)
#else
#error Compiler not found!
#endif
{
    // wake on CCR0 count match
    TA0CCTL0 = 0;
    __bic_SR_register_on_exit(LPM0_bits|GIE);
}
//  ======== UART RX Interrupt ========
#if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
#pragma vector=USCI_A0_VECTOR
__interrupt void USCI_A0_ISR(void)
#elif defined(__GNUC__)
void __attribute__ ((interrupt(USCI_A0_VECTOR))) USCI_A0_ISR (void)
#else
#error Compiler not supported!
#endif
{
  switch(__even_in_range(UCA0IV,4))
  {
  case 0:break;                             // Vector 0 - no interrupt
  case 2:                                   // Vector 2 - RXIFG
      GPIO_toggleOutputOnPin(LED_5);
      RegValueMSB = UCA0RXBUF;              //Assigns reveived byte to RegValueMSB
      if (RegValueMSB == 0xD5){ //checks for charge complete byte
          StdI2C_P_TX_Single(BQ25619_ADDR,0x01 , 0x1A, &Err);//disable BQ25619 boost mode on charge
complete
      }
      else{
      stobuf1 = RegValueMSB * 6;
      stobuf2 = (double)stobuf1 / 256;       //calculate Vbat
      batteryVoltageCheck = stobuf2;
      if(batteryVoltageCheck <= 4.05){
          //PwmDuty = 40-((stobuf2-2.6)/.095);
          PwmDuty =((7.606829268 - (1.434146341*(stobuf2 + .33)))*(40/3.3)); //~= .3v headroom for
longer intervals
      }
      // stobuf2  is holder for Vbat
      //PwmDuty can range from 0 = 0% = 5.3V to 40 = 100% = 3V
      if (PwmDuty >= 14 && PwmDuty <= 40){ //qualify PwmDuty, never raise voltage past 4.5
          TA2CCR2 = PwmDuty; //assign PwmDuty cycle
      }
      else{
      TA2CCR2 = 14;
      }
      __delay_cycles(40000);
      GPIO_setOutputLowOnPin(CASE_G1);      //enter power mode
      }
    break;
  case 4:break;                             // Vector 4 - TXIFG
  default: break;
  }
}
//Released_Version_5_10_00_17
```

10.2 耳塞 main.c

```
/* --COPYRIGHT--,BSD
 * Copyright (c) 2016, Texas Instruments Incorporated
 * All rights reserved.
 *
 * 只要符合下列条件，便可再次分发和使用源代码和
 * 二进制形式的以下内容，不管有无进行修改：
 *
 * *   在再次分发源代码时必须保留以上版权声明、
 *     本条件清单以及下述免责声明。
 *
 * *   在以二进制形式进行再次分发时，必须复制
 *     分发时提供的文件和/或其他材料中的以上
 *     版权声明、本条件清单以及下述免责声明。
 *
 * *   除非事先得到特别的书面许可，否则请勿
 *     使用德州仪器 (TI) 公司和贡献者的名称
 *     来支持和推广衍生自该软件的产品。
 *
 * 本软件由版权持有者和贡献者“按原样”
 * 提供，且不承认任何明示或暗示的保证，
 * 包括但不限于对适销性或在某特定用途
 * 方面的适用性的隐含保证。在任何情况下，
 * 对于任何因使用本软件而造成的直接、间接、附带、特殊，
 * 惩罚性或结果性损害（包括但不限于：
 * 替代性商品或服务的采购；使用、数据或利润的损失；
 * 或者业务中断），无论这些损害是如何引起的，
 * 在何种责任理论上，无论是出于合同、严格责任还是
 * 侵权行为（包括疏忽或其他），版权所有者或贡献者
 * 概不负责，即使已被提前告知出现这种损害的可能性。
 * --/COPYRIGHT--*/
// ======== main.c ========
#include <string.h>
#include <stdint.h>
#include <inttypes.h>
#include <string.h>
#include <stdio.h>
#include "board_functions.h"
#include <stdlib.h>
#include <msp430.h>
#include "driverlib.h"
#include "StdI2C.h"
#include "BQ25150.h"
//#include "board_timer.h"
#include "USB_config/descriptors.h"
#include "USB_API/USB_Common/device.h"
#include "USB_API/USB_Common/usb.h"                 // USB-specific functions
#include "USB_API/USB_CDC_API/UsbCdc.h"
#include "USB_app/usbConstructs.h"
#include "OLED/Oled_SSD1306.h"
#include "StdPollI2C.h"
#include "hal.h"
// Function declarations
uint8_t retInString(char* string);
void initTimer(void);
void setTimer_A_Parameters(void);
// Global flags set by events
volatile uint8_t bCDCDataReceived_event = FALSE; // Indicates data has been rx'ed
// without an open rx operation
char str[50];
#define NUM_SAMPLES 8
#define clkspeed 3          // 24Mhz Clock Select
//#define clkspeed 1        // 8Mhz Clock Select
short samples[NUM_SAMPLES] ;
int sample_index = 0 ;
char str[50];
char cmdstring[5];
char cs2[3];
uint8_t response = 0;
//unsigned char buffer[10]={1,2,3,4,5,6,7,8,9,10};
volatile char wholeString[MAX_STR_LENGTH] = "";
volatile uint8_t modecounter = 0;
volatile uint8_t pluscounter = 0;
///Timer_A_initUpModeParam Timer_A_params = {0};
int i;
unsigned char* PRxData;                     // Pointer to RX data
unsigned char RXByteCtr;
```

```
volatile unsigned char RxBuffer[128];       // Allocate 128 byte of RAM
unsigned char* PTxData;                     // Pointer to TX data
unsigned char TXByteCtr;
const unsigned char TxData[] =              // Table of data to transmit
{
 0x11,
 0x22,
 0x33,
 0x44,
 0x55
};
volatile uint8_t Button_Released = 0;
unsigned int result;
const char* hexstring = "0xabcdef0";
int raddr;
int rdata;
char buf[5];
uint8_t uraddr;
uint8_t urdata;
uint16_t Err;
// Holds the outgoing string
char outString[MAX_STR_LENGTH] = "";
uint8_t connectedflag = 0;
uint8_t echoflag = 1;
int ubtncounter = 0;
uint8_t RegValueLSB = 0;
uint8_t RegValueMSB = 0;
uint8_t VbatMSB = 0;
uint8_t ADCCheck = 0;
uint32_t stobusf;
uint8_t ADCcount = 0;
uint8_t VinReadA = 0;
uint8_t VinReadB = 0;
uint8_t VinReadC = 0;
uint8_t VinLow = 0;
double vIn = 0;
char vBat[];
//uint8_t RegValueM = 0;
//uint8_t RegValueL = 0;
uint32_t stobuf;
uint32_t stobuf1;
double stobuf2;
uint8_t RegValues = 0;
//__delay_cycles(1000); 1000 = 100us
uint8_t rthex;
// Set/declare toggle delays
//uint16_t SlowToggle_Period = 20000 - 1;
//uint16_t FastToggle_Period = 1000 - 1;
/*
 * ======== main ========
 */
void main(void)
{
    WDT_A_hold(WDT_A_BASE); // Stop watchdog timer
    // Minumum Vcore setting required for the USB API is PMM_CORE_LEVEL_2 .
    PMM_setVCore(PMM_CORE_LEVEL_2);
    USBHAL_initPorts();           // Config GPIOS for low-power (output low)
    USBHAL_initClocks(8000000 * clkspeed);   // Config clocks.MCLK=SMCLK=FLL=8MHz; ACLK=REFO=32kHz
    //USBHAL_initClocks(24000000);
    initTimer();            // Prepare timer for LED toggling
    //USB_setup(TRUE, TRUE); // Init USB & events; if a host is present, connect
    initI2C();
//  ======== UART Setup ========
    P3SEL = BIT3+BIT4;                          // P3.4,5 = USCI_A0 TXD/RXD
    __delay_cycles(20000);
    UCA0CTL1 |= UCSWRST;                        // **Put state machine in reset**
    UCA0CTL1 |= UCSSEL_2;                       // SMCLK
    UCA0BR0 = 6;                                // 1MHz 9600 (see User's Guide)
    UCA0BR1 = 0;                                // 1MHz 9600
    UCA0MCTL = UCBRS_0 + UCBRF_13 + UCOS16;     // Modln UCBRSx=0, UCBRFx=0,
                                                // over sampling
    UCA0CTL1 &= ~UCSWRST;                       // **Initialize USCI state machine**
//  ======== GPIO Setup ========
    //LED Setup
    GPIO_setAsOutputPin(LED_1);
    GPIO_setOutputLowOnPin(LED_1);
    GPIO_setAsOutputPin(LED_2);
    GPIO_setOutputLowOnPin(LED_2);
    GPIO_setAsOutputPin(LED_3);
```

```
    GPIO_setOutputLowOnPin(LED_3);
    //GPIO Setup
    GPIO_setAsInputPinWithPullUpResistor(BQ_INT);
    GPIO_setAsInputPin(BQ_PG);
    GPIO_setAsInputPinWithPullUpResistor(BQ_START);
    GPIO_setAsOutputPin(BQ_CE);
    GPIO_setOutputLowOnPin(BQ_CE);
    GPIO_setAsOutputPin(BQ_LP);
    GPIO_setOutputHighOnPin(BQ_LP);
    GPIO_setAsOutputPin(BQ_G1);
    GPIO_setOutputLowOnPin(BQ_G1);
    GPIO_toggleOutputOnPin(LED_1);
    waitms(500);
    GPIO_toggleOutputOnPin(LED_1);
    waitms(500);
    GPIO_toggleOutputOnPin(LED_1);
    waitms(500);
    GPIO_toggleOutputOnPin(LED_1);
//  ======== BQ25155 Register Setup ========
    StdI2C_TX_Single(BQ25150_ADDR, BQ25150_MASK0, 0x5E, &Err);
    StdI2C_TX_Single(BQ25150_ADDR, BQ25150_MASK1, 0xBF, &Err);
    StdI2C_TX_Single(BQ25150_ADDR, BQ25150_MASK2, 0xF1, &Err);
    StdI2C_TX_Single(BQ25150_ADDR, BQ25150_MASK3, 0x77, &Err);
    StdI2C_TX_Single(BQ25150_ADDR, BQ25150_VBAT_CTRL, 0x3C, &Err);
    StdI2C_TX_Single(BQ25150_ADDR, BQ25150_ICHG_CTRL, 0x50, &Err);//sets Ichg to 200mA, 0xA0 for
200mA, 0x50 for 100mA, 0x20 for 40mA
    StdI2C_TX_Single(BQ25150_ADDR, BQ25150_PCHRGCTRL, 0x02, &Err);
    StdI2C_TX_Single(BQ25150_ADDR, BQ25150_TERMCTRL, 0x14, &Err);
    StdI2C_TX_Single(BQ25150_ADDR, BQ25150_BUVLO, 0x00, &Err);
    StdI2C_TX_Single(BQ25150_ADDR, BQ25150_CHARGERCTRL0, 0x92, &Err);
    StdI2C_TX_Single(BQ25150_ADDR, BQ25150_CHARGERCTRL1, 0xC2, &Err);
    StdI2C_TX_Single(BQ25150_ADDR, BQ25150_ILIMCTRL, 0x06, &Err);
    StdI2C_TX_Single(BQ25150_ADDR, BQ25150_MRCTRL, 0x2A, &Err);
    StdI2C_TX_Single(BQ25150_ADDR, BQ25150_ICCTRL0, 0x10, &Err);
    StdI2C_TX_Single(BQ25150_ADDR, BQ25150_ICCTRL1, 0x00, &Err);
    StdI2C_TX_Single(BQ25150_ADDR, BQ25150_ICCTRL2, 0x40, &Err);
    StdI2C_TX_Single(BQ25150_ADDR, BQ25150_ADCCTRL0, 0x40, &Err); //0x58 Sets ADC to continuous
with 3ms conv., 0x40 sets to continuous with 24ms conv.
    StdI2C_TX_Single(BQ25150_ADDR, BQ25150_ADCCTRL1, 0x00, &Err);
    StdI2C_TX_Single(BQ25150_ADDR, BQ25150_ADCALARM_COMP1_M, 0x23, &Err);
    StdI2C_TX_Single(BQ25150_ADDR, BQ25150_ADCALARM_COMP1_L, 0x20, &Err);
    StdI2C_TX_Single(BQ25150_ADDR, BQ25150_ADCALARM_COMP2_M, 0x38, &Err);
    StdI2C_TX_Single(BQ25150_ADDR, BQ25150_ADCALARM_COMP2_L, 0x90, &Err);
    StdI2C_TX_Single(BQ25150_ADDR, BQ25150_ADCALARM_COMP3_M, 0x00, &Err);
    StdI2C_TX_Single(BQ25150_ADDR, BQ25150_ADCALARM_COMP3_L, 0x00, &Err);
    StdI2C_TX_Single(BQ25150_ADDR, BQ25150_ADC_READ_EN, 0xFE, &Err);
    StdI2C_TX_Single(BQ25150_ADDR, BQ25150_TS_FASTCHGCTRL, 0x34, &Err);
    StdI2C_TX_Single(BQ25150_ADDR, BQ25150_TS_COLD, 0x7C, &Err);
    StdI2C_TX_Single(BQ25150_ADDR, BQ25150_TS_COOL, 0x6D, &Err);
    StdI2C_TX_Single(BQ25150_ADDR, BQ25150_TS_WARM, 0x38, &Err);
    StdI2C_TX_Single(BQ25150_ADDR, BQ25150_TS_HOT, 0x28, &Err);
//  ======== Ready while loop ========
    //This while loop holds the program with the interrupts disabled.This allows synchronization
with the case
    //short BQ_START pin 3.7 to ground to exit loop
    while(GPIO_getInputPinValue(BQ_START) == 1) //Wait on start condition before enabling interrupts
    {
        GPIO_toggleOutputOnPin(LED_2);
        __delay_cycles(3000000);
        GPIO_toggleOutputOnPin(LED_2);
        __delay_cycles(3000000);
        GPIO_toggleOutputOnPin(LED_2);
        __delay_cycles(3000000);
        GPIO_toggleOutputOnPin(LED_2);
        __delay_cycles(24000000);
    }
    GPIO_setOutputLowOnPin(LED_2);
//  ======== Interrupt Enables ========
    __enable_interrupt();  // Enable interrupts globally
    GPIO_enableInterrupt(BQ_INT);
    GPIO_selectInterruptEdge(BQ_INT, GPIO_HIGH_TO_LOW_TRANSITION);
    GPIO_clearInterrupt(BQ_INT);
//  ======== Main while loop ========
    //reads Vbat, waits for case to drive Vin low
    while(1)
    {
         StdI2C_P_RX_Single(BQ25150_ADDR, BQ25150_ADCDATA_VBAT_M, &VbatMSB, &Err); //Read Vbat
         StdI2C_P_RX_Single(BQ25150_ADDR, BQ25150_ADCDATA_VIN_M, &RegValueMSB, &Err); //Read Vin
         GPIO_toggleOutputOnPin(LED_1);
```

```
        __delay_cycles(12000000);
    }
}
/*
 * ======== TIMER1_A0_ISR ========
 */
#if defined(__TI_COMPILER_VERSION__) || (__IAR_SYSTEMS_ICC__)
#pragma vector=TIMER0_A0_VECTOR
__interrupt void TIMER0_A0_ISR (void)
#elif defined(__GNUC__) && (__MSP430__)
void __attribute__ ((interrupt(TIMER0_A0_VECTOR))) TIMER0_A0_ISR (void)
#else
#error Compiler not found!
#endif
{
    // wake on CCR0 count match
    TA0CCTL0 = 0;
    __bic_SR_register_on_exit(LPM0_bits|GIE);
}
//  ======== BQ25155 Interrupt ========
#if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
#pragma vector=PORT1_VECTOR
__interrupt
#elif defined(__GNUC__)
__attribute__((interrupt(PORT1_VECTOR)))
#endif
void Port_1(void)
{
    switch(__even_in_range(P1IV,0x03))
    {
    case P1IV_P1IFG3:
        StdI2C_P_RX_Single(BQ25150_ADDR, BQ25150_STAT0, &RegValueMSB, &Err); //Read STAT0 REG
        RegValueMSB &= 0x01; // Check if VIN_PGOOD_STAT is asserted
        if(RegValueMSB == 0x00){ //if VIN_PGOOD_STAT is not asserted check to see if Vin has been
driven low
//                __delay_cycles(6000000);
//
//                StdI2C_P_RX_Single(BQ25150_ADDR, BQ25150_ADCDATA_VIN_M, &RegValueMSB, &Err); //
Read Vin
//                stobuf = RegValueMSB;
//                stobuf1 = stobuf * 6;
//                vIn = (double)stobuf1 / 256;
          ADCcount = 0;
          VinLow = 0;
          VinReadA = 5;
          VinReadB = 5;
          VinReadC = 5;
          while(ADCcount < 85 && VinLow == 0){
            __delay_cycles(120000);
            StdI2C_P_RX_Single(BQ25150_ADDR, BQ25150_ADCDATA_VIN_M, &RegValueMSB, &Err); //Read Vin
            stobuf = RegValueMSB;
            stobuf1 = stobuf * 6;
            vIn = (double)stobuf1 / 256;
            VinReadC = VinReadB;
            VinReadB = VinReadA;
            VinReadA = vIn;
            ADCcount++;
            if(VinReadA < 3 && VinReadB < 3 && VinReadC < 3){
                VinLow = 1;
                VinReadA = 0;
                VinReadB = 0;
                VinReadC = 0;
                vIn = 0;
            }
          }
                if(VinLow == 1){//if Vin is <3V begin communication process
                    StdI2C_P_RX_Single(BQ25150_ADDR, BQ25150_STAT0, &RegValueMSB, &Err); //Read
STAT0 REG
                    RegValueMSB &= 0x20; // Check if CHARGE_DONE_STAT is asserted
//                    RegValueMSB = 0x20; //uncomment to assert charge complete bit for testing
                    if(RegValueMSB){                        // if CHARGE_DONE_STAT is asserted
transmit 0xD5, 0xD5 = Vbat = 5v, this is chosen as the arbitrary charge complete bit for simplicity
                        GPIO_setOutputHighOnPin(BQ_G1);     // Enter Comms Mode
                        __delay_cycles(4000);
                        while (!(UCA0IFG&UCTXIFG));         // USCI_A0 TX buffer ready?
                        UCA0TXBUF = 0xD5;                   // TX -> 0xD5 = 5V, out of charge range
                        __delay_cycles(4000);
                        GPIO_toggleOutputOnPin(LED_3);
                        GPIO_setOutputLowOnPin(BQ_G1);      //Enter Power Mode
```

```
                        GPIO_toggleOutputOnPin(LED_2);
                        __delay_cycles(4000000);
                        GPIO_toggleOutputOnPin(LED_2);
                        __delay_cycles(4000000);
                        GPIO_setOutputHighOnPin(LED_2);
                    }
                    else{
                    stobuf = VbatMSB;
                    stobuf1 = stobuf * 6;
                    stobuf2 = (double)stobuf1 / 256; //stobuf2 = double of Vbat
                    GPIO_setOutputHighOnPin(BQ_G1);    // Enter Comms Mode
                    __delay_cycles(4000);
                    while (!(UCA0IFG&UCTXIFG));        // USCI_A0 TX buffer ready?
                    UCA0TXBUF = VbatMSB;               // TX -> vBat MSB
                    __delay_cycles(4000);
                    GPIO_toggleOutputOnPin(LED_3);
                    GPIO_setOutputLowOnPin(BQ_G1);     //Enter Power Mode
                    }
                }
        }
        //Clear all interrupt flags
        StdI2C_P_RX_Single(BQ25150_ADDR, BQ25150_FLAG0, &RegValues, &Err);
        StdI2C_P_RX_Single(BQ25150_ADDR, BQ25150_FLAG1, &RegValues, &Err);
        StdI2C_P_RX_Single(BQ25150_ADDR, BQ25150_FLAG2, &RegValues, &Err);
        StdI2C_P_RX_Single(BQ25150_ADDR, BQ25150_FLAG3, &RegValues, &Err);
        GPIO_clearInterrupt(BQ_INT);
        break;
    default:
        break;
    }
}
//Released_Version_5_10_00_17
```

11 修订历史记录

注：以前版本的页码可能与当前版本的页码不同

Changes from Revision * (June 2022) to Revision A (May 2023)	Page
• 更新了整个文档中的表格、图和交叉参考的编号格式	1

重要声明和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他功能安全、信息安全、监管或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265