Application Report
SLUA976–July 2019

Battery Discharge Implementation in Warm or Hot Temperatures

Gautham Ramachandran, Alexander Makeever

ABSTRACT

Lithium-ion (Li-ion) batteries tend to become dangerous when they are overcharged at high temperatures. Safely charging these batteries has become one of the most important design specifications in battery-powered portable equipment. Progress has been made in establishing industry standards such as the Japan Electronics and Information Technology Industries Association (JEITA) guidelines for improving battery-charging safety. However, this only applies if the battery temperature is exceeded before or around when the charging process began.

There are many instances where the battery could have been fully charged and then the battery temperature could have risen as it was taken into a hot car or the battery is exposed to heat coming from other peripherals. This concept can be extended for long-term storage of battery-enabled products, and certain requirements for shipping within a certain state of charge (SOC) comes into account. Similar principles discussed in this application report can be applied to design around supercapacitors as well.

This application example discusses how a designer can take this requirement into consideration and design a reliable and safe product. While this application note talks about warm and hot cases, it can be extended to other temperature requirements or actions that can be performed. This example uses the BQ25150 battery charger device, along with internal ADC comparators and a current sink, to slowly discharge the battery. Once a safe battery voltage is reached, the current sink is thereafter cut-off. An automatic fault checking routine is set to achieve very low power consumption and ensure that the bias currents for checking the battery pack thermistor is only turned ON when taking a measurement. This saves power and is less workload for the microcontroller since it does not always have to sample external channels.

Contents

List of Figures

List of Tables

Trademarks

All trademarks are the property of their respective owners.

1 Introduction

Temperature monitoring is important to safely charge a battery, as extremely high or low temperatures can reduce the longevity of a battery if not handled properly. The JEITA standard indicates that when a battery reaches a "warm" temperature of about 45°C, it must only be charged to 100–200 mV less than the full cell voltage. When actively charging the battery, this requirement can be met by changing the battery regulation voltage and stopping the charge cycle when the battery reaches this reduced voltage. However, a battery that has already been charged to the full cell-voltage target cannot be regulated by a charge cycle if it later becomes warm. In this case, to avoid damaging the battery, some capacity must be drained by the battery management system until the battery reaches a safe voltage. While this method reduces the usable battery capacity for that particular cycle, the battery lifetime is impacted less by the high temperature.

This report outlines an algorithm used to discharge a warm or hot battery using the BQ2515x devices. In order to make use of this functionality, a microcontroller host device must be used to keep track of the temperatures and battery voltages reported by the IC and control the operation of the device based on these parameters. In testing this algorithm, an MSP430 microcontroller was used.

2 Schematic

In order to implement the warm or hot battery discharge algorithm, place an 0402 200-Ω surface mount resistor (such as the Vishay CRCW-HP e3) between the PMID and /PG pins, as shown in Figure 1. This discharges a 4.2-V battery at around 20 mA and the maximum current allowed through the open-drain FET connected to the /PG pin in the BQ25155 device.

NOTE: Always check the ratings on the FET and the external resistor when using an open-drain pin to perform similar discharge mechanisms.

The /PG pin, configured as a general purpose open-drain output, acts as a switch to begin or stop the 20-mA battery discharge. In order for this resistor to only sink current from the battery instead of the adapter (if present), PMID must be configured internally through I^2C to be powered exclusively from the battery upon the beginning of the discharge.

DEVICE	ALLOWABLE SINK CURRENT THROUGH /PG PIN
BQ25150	5 mA
BQ25155	20 mA

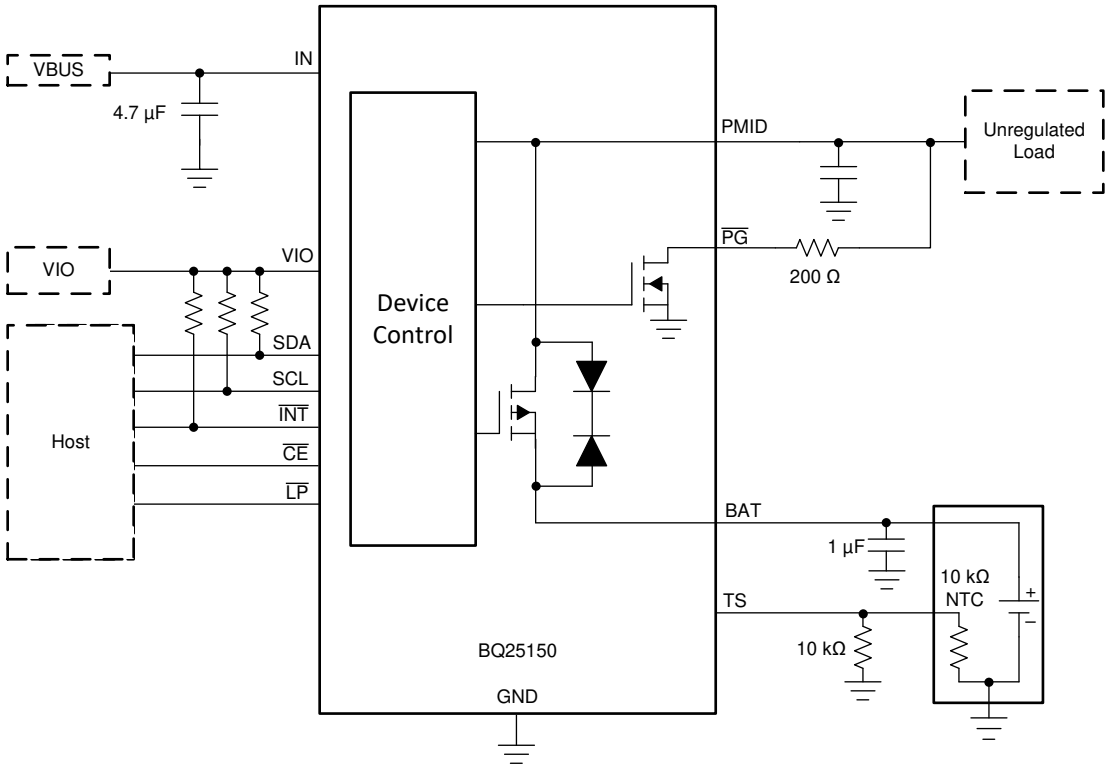

Figure 1. Discharge Schematic

3 Algorithm Implementation

3.1 Setup and Main Code

This algorithm uses an interrupt-driven approach to discharge a warm battery over a 200-Ω resistor. Three comparators within the BQ25150 device are used to trigger the interrupts based on temperature and battery voltage. As seen in Figure 2, these comparators must first be initialized based on their values and polarities. Comparator 1 must be initialized to trigger when the temperature goes from normal to warm, and Comparator 2 must be initialized to trigger when the battery voltage goes higher than the safe warm battery voltage. Comparator 3 must also be initialized to trigger when the temperature goes from warm to normal, but the interrupt must be masked until the battery is discharging.

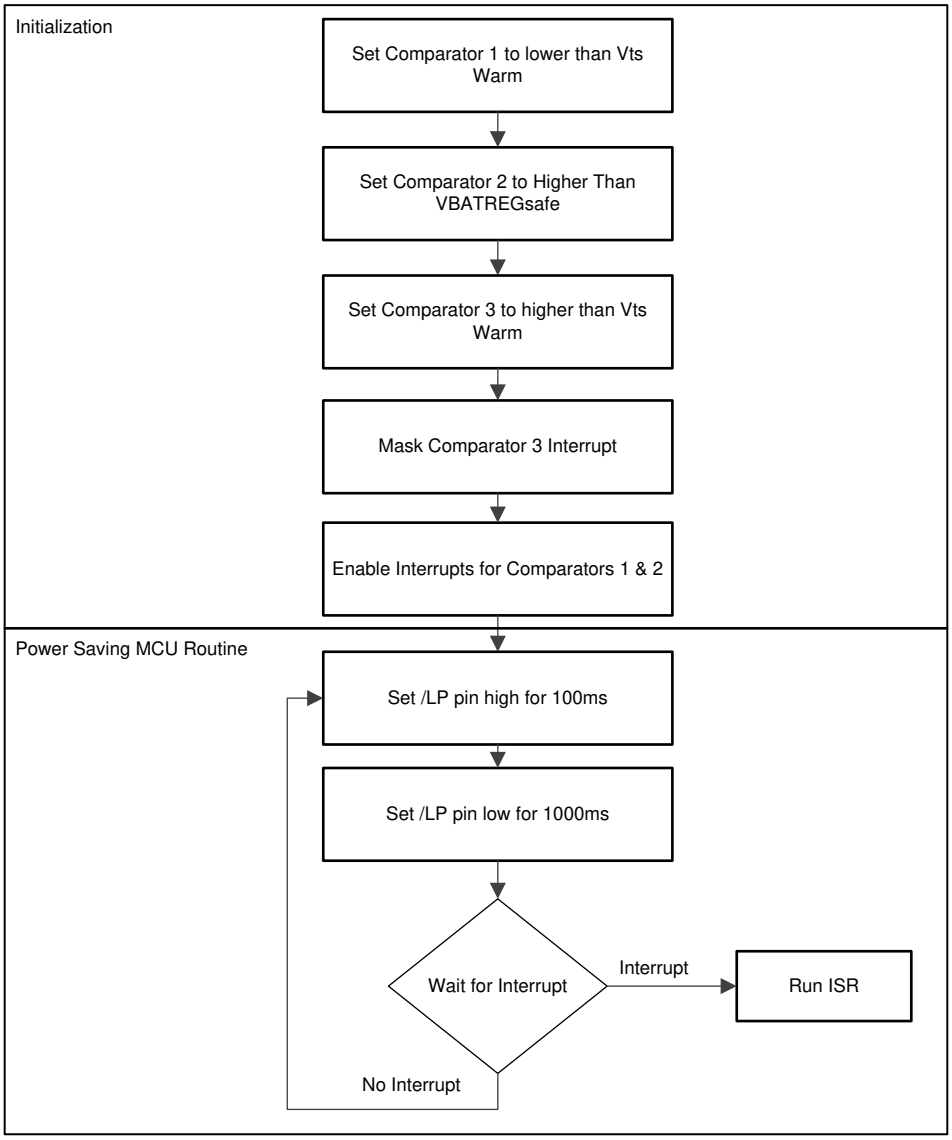

Figure 2. Main Code Flow Chart

To conserve power, the microcontroller must enter an ongoing loop in which it continually toggles the /LP pin on the battery charger, as shown in Figure 3. This triggers the charger to make the temperature and battery voltage measurements it needs through the ADC without burning excess power while the adapter is not present. In this example, the /LP pin is pulled high for about 100 ms, and pulled low for about 1000 ms, but this can be changed as the user sees fit so long as the /LP pin is pulled high for long enough to make the ADC measurement. As /LP is toggled, the MCU is waiting for the interrupt. As soon as an interrupt is triggered, the Interrupt Service Routine begins. This interrupt has two different instances in this algorithm: one instance to begin discharging the battery, and one instance to stop discharging the battery. There are a few ways to determine which instance must run. All of these options are based on reading a register that is set during battery discharge and cleared otherwise, or vice-versa. In this example, the third comparator mask bit is read. If the bit is set to mask the comparator interrupt, then the routine must run the battery discharge instance, and if the bit is cleared to unmask the comparator interrupt, then the routine must run the battery discharge halt instance.

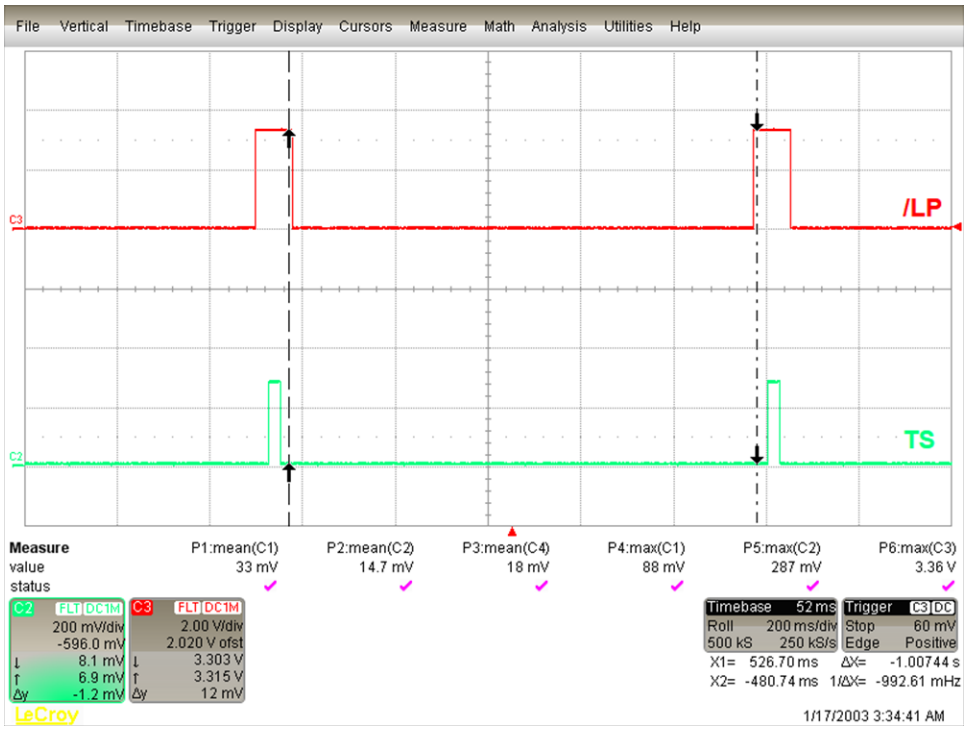

Figure 3. LP Toggling (VIN Not Present)

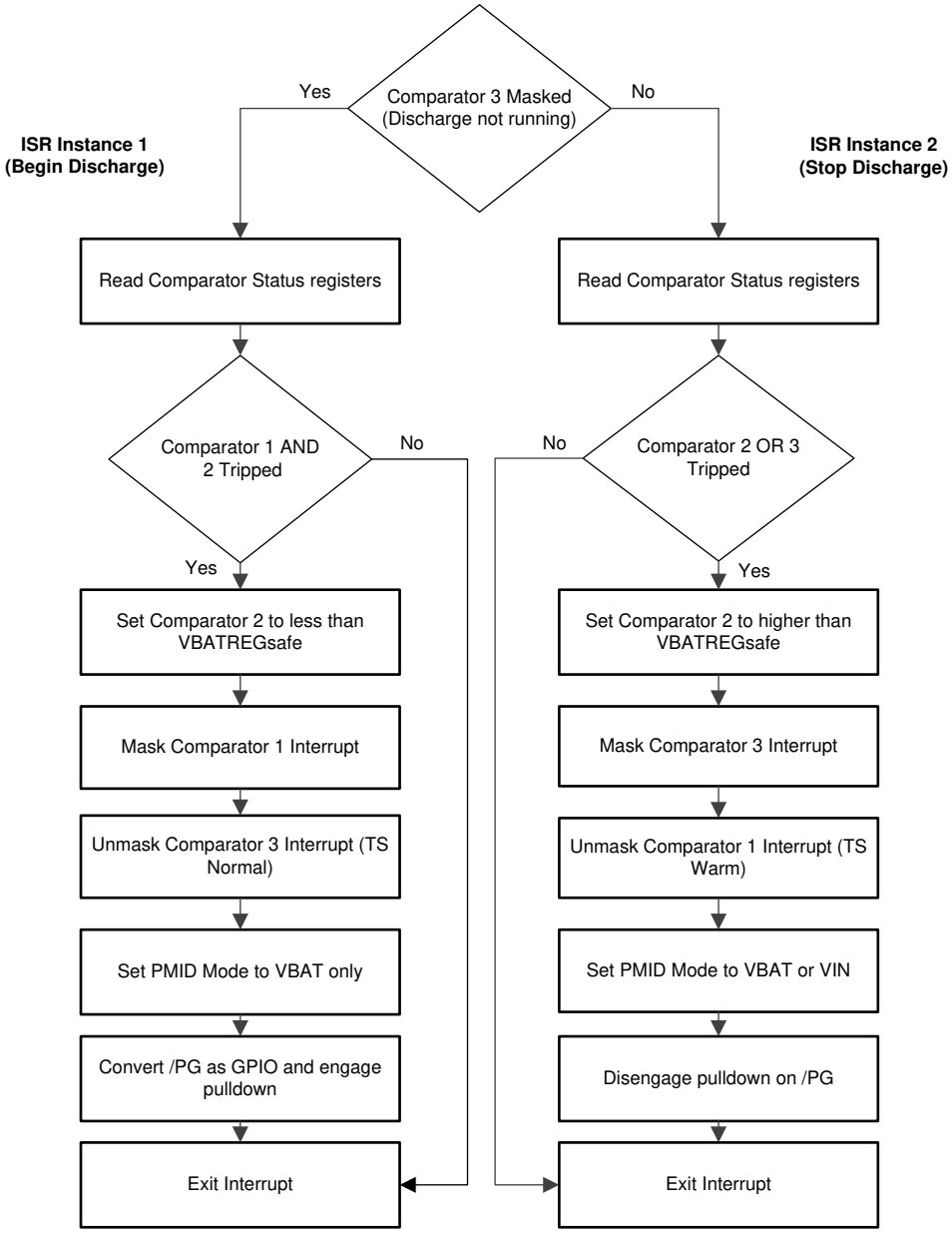

Figure 4. ISR Flow Chart

3.2 Beginning Battery Discharge

When the first instance of the interrupt runs, the microcontroller must check to make sure that both of the comparators have been tripped, indicating that the battery is both warm and above the safe warm battery voltage threshold. This can be done by performing an I^2C read on the comparator status registers. If one or neither of the comparators has been tripped, then the interrupt must exit immediately without making any changes to the device. However, if both comparator statuses show that their condition has been met, then the discharge can begin.

1. Flip the polarity of Comparator 2 so it triggers when the battery voltage is below the safe warm battery voltage threshold.
2. Mask Comparator 1 and unmask Comparator 3 to effectively flip the polarity of when the temperature-based interrupt occurs. Making these two comparator changes triggers an interrupt as soon as the battery is in a safe state, either at a normal temperature or at a safe warm battery voltage.

3. For the actual discharge, the MCU sends an I²C write command to set PMID to be powered by VBAT only, and to convert /PG to a general purpose open drain output. Assuming the 200-Ω resistor is connected between PMID and /PG, this allows exclusively battery current to be sunk through the resistor.
4. Internally pull the /PG pin to ground.
5. Exit the interrupt.

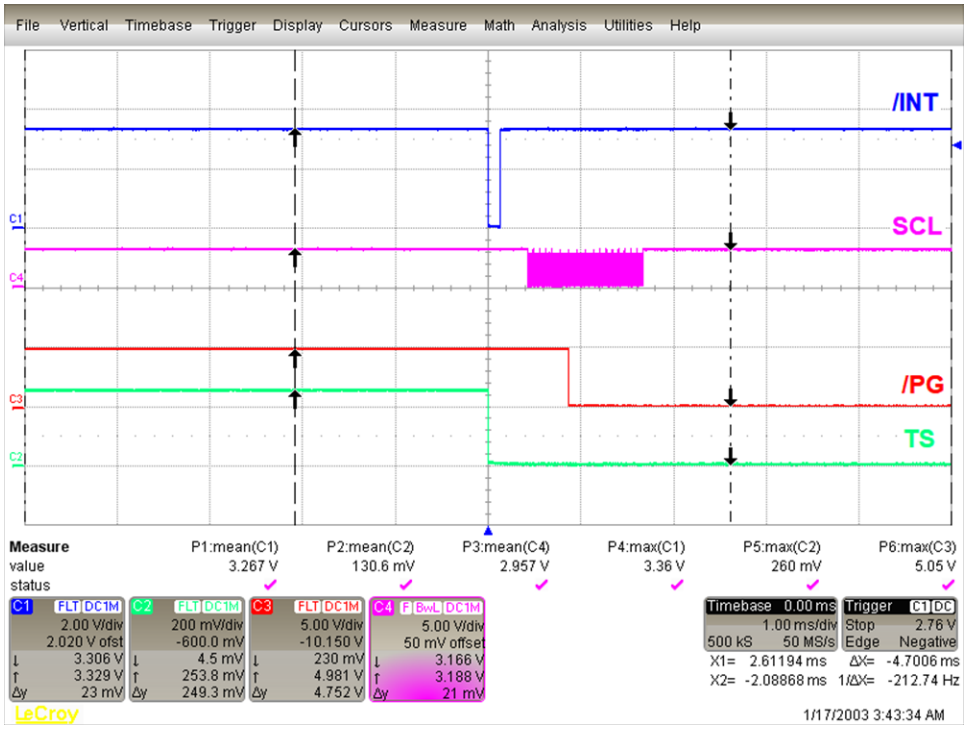

Figure 5. Begin Discharge

As seen in Figure 6, the discharge effectively reduces the battery voltage to a safe threshold. With the 45 mAh battery used in testing, the battery was discharged by 100 mV in roughly five minutes. This rate of discharge decreases as the battery voltage is decreased since the resistor sinking the current is fixed, but the discharge is only necessary until the battery voltage reaches a safe threshold. To accelerate this process, power can be burned by turning ON peripherals on the module which cannot be seen. Since the comparators are sampling the battery voltage, having multiple discharge paths is still okay.

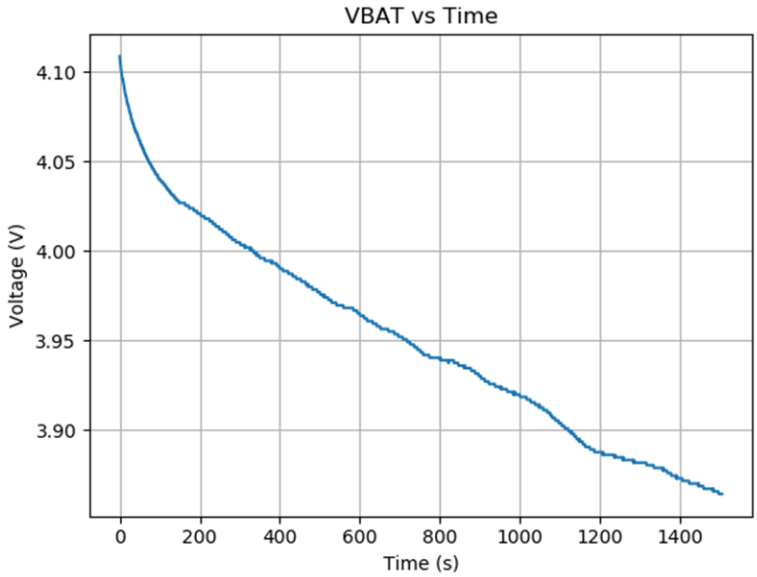

Figure 6. Discharge Curve

3.3 Stopping Battery Discharge

Similarly, for the second instance of the interrupt, the comparator status bits must be read over I^2C.

1. In this case, stop the discharge if either, or both, of the comparator conditions are met. This stops the discharge as soon as any safe condition is encountered.
2. Change the comparators back to their exact configurations from before the first interrupt instance. That is, set Comparator 1 as unmasked and Comparator 3 as masked. Set the polarity of Comparator 2 back to greater than the safe warm battery voltage. These changes cause interrupts to be triggered when the battery enters an unsafe temperature and voltage combination, as they did before the initial discharge.
3. Set PMID back to being powered by either the battery or the input voltage.
4. Set /PG to high impedance to stop discharging the battery, which can be accomplished through I^2C commands.

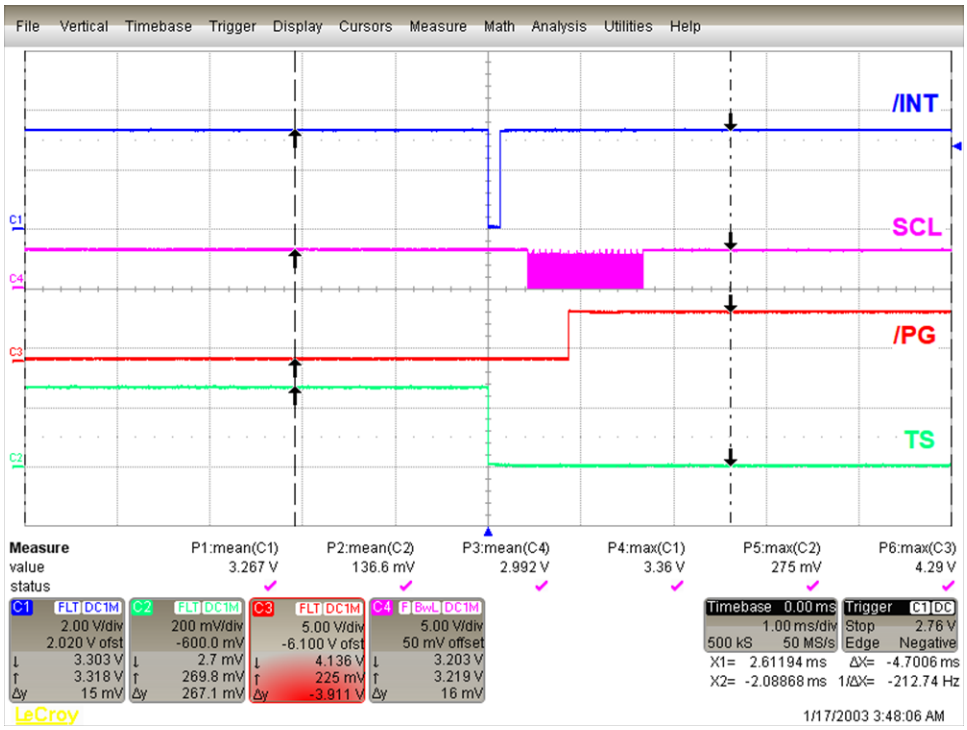

Figure 7. Stop Discharge

3.4 Software Implementation Example Using the MSP430F5529 LaunchPad

```
    /*--COPYRIGHT--, BSD
* Copyright (c) 2019, 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 "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
void initTimer(void);
void setTimer_A_Parameters(void);
#define clkspeed 3          // 24Mhz Clock Select
//#define clkspeed 1      // 8Mhz Clock Select
int i;
uint16_t Err;
uint8_t ADCCheck = 0;
uint8_t RegValues = 0;
/*
 * ======== main ========
 */
void main(void)
{
    WDT_A_hold(WDT_A_BASE); // Stop watchdog timer
    // Minumum Vcore setting requiblue 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
    GPIO_setAsOutputPin(LED_PORTR, LED_PINR);
    GPIO_toggleOutputOnPin(LED_PORTR, LED_PINR);
    initI2C();
    GPIO_toggleOutputOnPin(LED_PORTR, LED_PINR);
    GPIO_setAsInputPin(BQ_INT);
    GPIO_setAsOutputPin(BQ_LP);
    GPIO_setOutputHighOnPin(BQ_LP);
    waitms(2);
    GPIO_toggleOutputOnPin(LED_PORTR, LED_PINR);
    __enable_interrupt();  // Enable interrupts globally
    GPIO_setOutputHighOnPin(BQ_LP);
    waitms(10);
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_ICCTRL0, 0x02, &Err); //HW Reset
    waitms(300);
    GPIO_setOutputLowOnPin(BQ_LP);
    waitms(1200);
    GPIO_setOutputHighOnPin(BQ_LP);

    //Disable Watchdog and Enable TS
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_CHARGERCTRL0, 0x90, &Err);

    // Disable interrupts for all the rest except ADC comparator
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_MASK0, 0xFF, &Err); //Mask all
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_MASK1, 0xBF, &Err); //Mask all
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_MASK2, 0x97, &Err); //Enable Comp1 and Comp2
interrupts
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_MASK3, 0xFF, &Err); //Mask all
```

```
    // Enable ADC channels for VBAT and TS
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_ADC_READ_EN, 0x0C, &Err);

    // Enable ADC
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_ADCCTRL0, 0x62, &Err); //enable ADC COMP1 as TS
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_ADCCTRL1, 0x68, &Err); //enable ADC COMP2 as VBAT
(=0x60; =0x80 for Comp3 as TS also)

    // Set Comparator 1 polarity and threshold -> TS < 0.265V
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_ADCALARM_COMP1_M, 0x38, &Err);
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_ADCALARM_COMP1_L, 0x80, &Err);

    // Set Comparator 2 polarity and threshold -> TVBAT < 4V
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_ADCALARM_COMP2_M, 0xAA, &Err);
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_ADCALARM_COMP2_L, 0xA8, &Err);

    //Set Comparator 3 polarity and threshold -> TS > 0.265V
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_ADCALARM_COMP3_M, 0x38, &Err);
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_ADCALARM_COMP3_L, 0x88, &Err);

    //Set PG pin as GPIO and set to HI-Z
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_ICCTRL1, 0x08, &Err);
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_ICCTRL2, 0x10, &Err);

    //Change target battery voltage to 4V during warm.
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_TS_FASTCHGCTRL, 0x44, &Err);

    //GPIO_setAsInputPinWithPullUpResistor(BQ_INT2);
    GPIO_setAsInputPin(BQ_INT2);
    GPIO_enableInterrupt(BQ_INT2);
    GPIO_selectInterruptEdge(BQ_INT2, GPIO_HIGH_TO_LOW_TRANSITION);
    GPIO_clearInterrupt(BQ_INT2);

    while(1)
    {
        GPIO_setOutputHighOnPin(BQ_LP); //Toggling LP to consume minimal current
        waitms(80);
        GPIO_setOutputLowOnPin(BQ_LP);
        waitms(1000);

    }
}

void turnOnDischarge(){

    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_ICCTRL1, 0x09, &Err);    //Setting PMID to be
powered only by VBAT
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_ICCTRL2, 0x00, &Err);    //Pulling down PG,
beginning discharge
                StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_ADCALARM_COMP2_M, 0xAA, &Err);
//Changing battery comparator polarity
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_ADCALARM_COMP2_L, 0xA0, &Err);    //VBAT < 4V
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_MASK2, 0xC1, &Err); //Masking comparator 1 and
unmasking comparator 3
}

void turnOffDischarge(){
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_ICCTRL1, 0x09, &Err); //Setting PMID to VBAT or VIN
power
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_ICCTRL2, 0x10, &Err); //Pull PG to HI-
Z to stop discharge
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_ADCALARM_COMP2_M, 0xAA, &Err); //Changing comparator
polarity
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_ADCALARM_COMP2_L, 0xD8, &Err); //VBAT > 4V
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_MASK2, 0x93, &Err); //Masking comparator 3 and
```

```
unmasking comparator 1
}

/*
 * ======== 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);

}

#if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
#pragma vector=PORT2_VECTOR
__interrupt
#elif defined(__GNUC__)
__attribute__((interrupt(PORT2_VECTOR)))
#endif
void Port_2(void)
{
    switch(__even_in_range(P2IV,0x04))
    {
    case P2IV_P2IFG2:

        // code to run when condition happens
        GPIO_setOutputHighOnPin(BQ_LP);
        __delay_cycles(10000);
        StdI2C_P_RX_Single(BQ25150_ADDR, BQ25150_STAT2, &RegValues, &Err);
        StdI2C_P_RX_Single(BQ25150_ADDR, BQ25150_MASK2, &ADCCheck, &Err);
                            if (((ADCCheck & 0x40) == 0x40) && (((RegValues & 0x30) == 0x30)
|| ((RegValues & 0x30) == 0x20) || ((RegValues & 0x30) == 0x10))) //If either Comparator 1 OR
Comparator 3 are tripped (signifying battery voltage/temp is safe), turn off battery discharge.
        {
            turnOnDischarge();
        }
else if(((ADCCheck & 0x40) == 0x00) && ((RegValues & 0x30) == 0x30) || ((RegValues & 0x30) ==
0x20) || ((RegValues & 0x30) == 0x10))) // If either Comparator 1 OR Comparator 3 are tripped
(signifying battery voltage/temp is safe), turn off battery discharge.
        {
            turnOffDischarge();
        }
        //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_setOutputLowOnPin(LED_WARN);
        GPIO_clearInterrupt(BQ_INT2);
        break;
    default:
        break;

    }
}
```

In summary, this demonstration shows an example of how to handle batteries at adverse temperatures. For any questions on this document or other charger devices, visit http://e2e.ti.com. See the *BQ25155 I²C Controlled 1-Cell 500-mA Linear Battery Charger with 10-nA Ship Mode, PowerPath with Regulated System (PMID) Voltage, ADC, and LDO Data Sheet*.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265