Application Note

State of Charge Estimation Using Smart Battery Charger

Nick Brylski

ABSTRACT

The state of charge (SOC) of a battery is a measurement that represents the remaining capacity of the battery as a percentage of its usable capacity. A fuel gauge is typically used to calculate the SOC by measuring battery voltage, current, and temperature as inputs to a gauging algorithm. Using a fuel gauge often results in an accurate SOC prediction. However, there are drawbacks to using a fuel gauge such as increased system cost, larger solution size, and additional power consumption. In applications where a high SOC accuracy is not required, a simple, voltage-based gauging method can be sufficient. This application note discusses a method of implementing a voltage-based SOC using the built in ADC of a battery charger IC, such as the BQ25155. This application note is also applicable to other chargers in the same family, including BQ25150 and BQ25157, or any charger which allows for battery voltage measurements.

Table of Contents

List of Figures

List of Tables

Trademarks

1 Introduction

A simple way to provide battery SOC in your product is through a static lookup table that correlates battery voltage to SOC. This lookup table can be generated by discharging the battery from full while measuring voltage and current.

2 Battery Characterization

The following steps must be taken to generate the voltage versus SOC table. This setup requires a voltmeter, ammeter, electronic load, and appropriate data logging equipment. Figure 2-1 shows a diagram of the test setup.

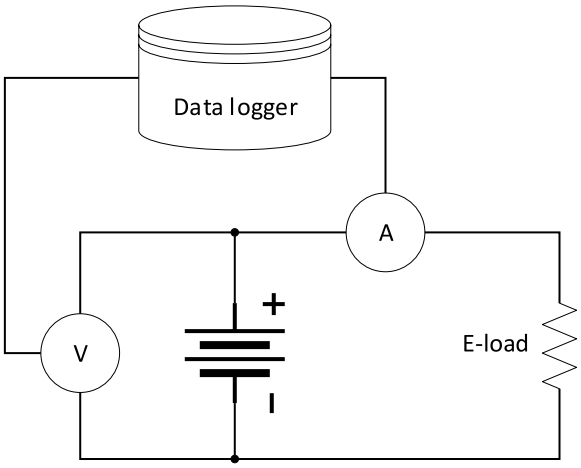

Figure 2-1. Test Setup

1. Ensure the battery is charged to full per its manufacturer's specification.
2. Connect the voltmeter to positive battery terminal.
3. Connect the electronic load to positive battery terminal with ammeter in series.
4. Ensure logging software is turned on and the voltage and current readings are being taken at regular intervals (every 5 to 10 seconds is okay).
5. Set load to your applications typical discharge rate and discharge battery to its end-of-discharge voltage.

A C/10 value or less is ideal to minimize the effects of relaxation. Note this process takes a while (> 10 hours). There are more comments on this assumption in the *Best Use Cases* section.

3 Generating the Lookup Table

A common method of calculating SOC takes the batteries remaining capacity (RemCap) and divides by the batteries maximum capacity, Q_{max}, where both of these parameters are measured in milliamp-hour (mAh).

First, calculate the total amount of passed charge (battery maximum capacity) over the course of the test.

$$Q_{max} = \sum_{k=1}^{m} i[k] \times \Delta t \quad (1)$$

i[k] is current at reading k, Δt is the time difference between readings, and m is the total number of readings.

The remaining capacity can be computed at each reading n.

$$\mathrm{RemCap[n]} = Q_{max} - \sum_{k=1}^{n} i[k] \times \Delta t \quad (2)$$

SOC can then be calculated.

$$\mathrm{SOC[n]} = \frac{RemCap[n]}{Q_{max}} \times 100 \quad (3)$$

Graphing the battery voltage against SOC generates the typical SOC curve for one li-ion cell.

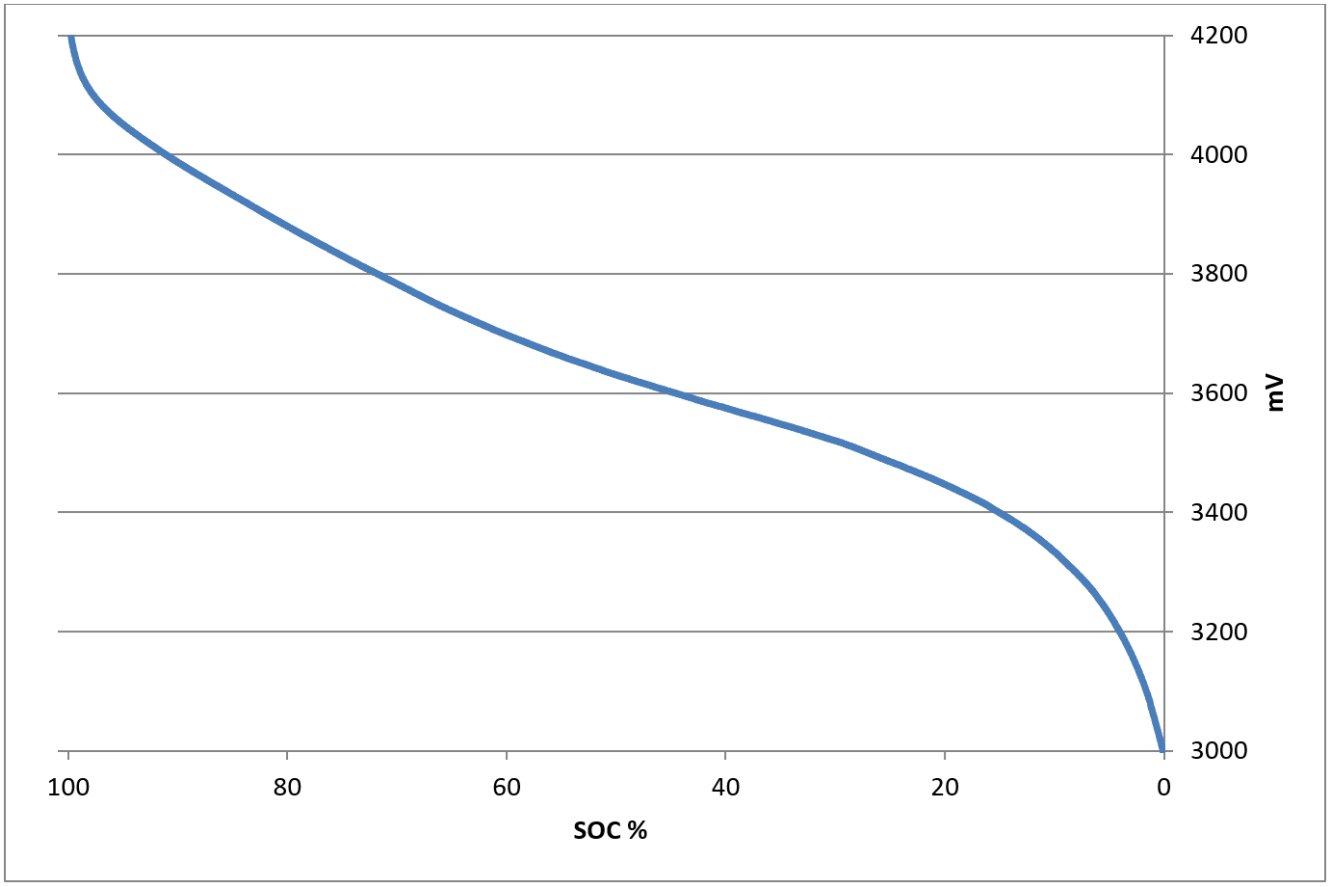

Figure 3-1. Vbat vs SOC

This method has been used to determine the accuracy of TI fuel gauges as seen here. The curve above represents the exact SOC for the given discharge.

Example code has been included in the *Python Lookup Table Generator* section that generates a polynomial regression based on the data generated from the SOC characterization. This data is then mapped to a 101-pt hexadecimal lookup table for easy import into an MCU application. Using 16-bit resolution, this table would only take up 202 bytes of memory.

4 BQ25155 Register Configuration

The BQ25155 is a highly integrated battery charge management IC that integrates the most common functions for wearable and portable devices, namely a charger, a regulated output voltage rail for system power, 16 bit ADC for battery and system monitoring, a LDO, and push-button controller. The BQ25155 IC integrates a linear charger with PowerPath that enables quick and accurate charging for small batteries while providing a regulated voltage to the system. The regulated system voltage (PMID) output can be configured through I 2C based on the recommended operating condition of downstream ICs and system loads for optimal system operation.

Tto limit the number of ADC conversions, and hence power consumption, the ADC conversions when in active battery mode can be limited to a period determined by the ADC_READ_RATE bits. In the case where the ADC_READ_RATE is set to manual mode, the host has to set the ADC_CONV_START bit to initiate the ADC conversion. After the ADC conversion is completed and the data is ready, the ADC_READY flag is set and an interrupt is sent to the host. In low power mode, the ADC remains OFF for minimal IC power consumption. The host must switch to active battery mode (set LP high) before performing an ADC measurement.

The BQ25155 allows for the creation of advanced battery monitoring and gauging firmware through its registers. Relevant registers have been listed below in Table 4-1.

Also included in the *MSP430 Code Snippet* section is an MSP430 code snippet that shows interfacing with the BQ25155.

Table 4-1. BQ25155 Registers

	Address	R or R/W	Note
CHRG_CV_STAT	0x0	R	Constant voltage charging mode (taper mode) status.
CHARGE_DONE_STAT	0x0	R	Charge done status. Can be used to force SOC to 100% in FW.
CHRG_CV_FLAG	0x3	R	Constant voltage charging mode (taper mode) flag. Can be configured as an interrupt to alert host MCU.
CHARGE_DONE_FLAG	0x3	R	Charge done flag. Can be configured as an interrupt to alert host MCU.
BAT_UVLO_FAULT_FLA G	0x4	R	Battery under voltage flag. Can be used to force SOC to 0% in FW, threshold configurable from 2.4–3 V. Can be configured as an interrupt to alert host MCU.
ADC_READY_FLAG	0x5	R	ADC ready flag. Can be configured as an interrupt to alert host MCU.
ADC_READ_RATE_1:0	0x40	R/W	Read rate for ADC measurements in BAT Only operation. Can be configured for manual or automatic conversions.
ADC_CONV_START	0x40	R/W	ADC conversion start trigger. Bit goes back to 0 when conversion is complete. Used for manual reads.
ADC_COMP1_2:0	0x40	R/W	ADC channel for comparator 1. Can be configured for VBAT channel.
1_ADCALARM_15:4	0x53	R/W	ADC alarm 1 threshold, can be configured to trigger interrupt when VBAT goes or below specified threshold. If more than one threshold is needed, two more alarms are available (all can be set to watch VBAT).
EN_VBAT_READ	0x58	R/W	Enable measurement for battery voltage (VBAT) channel.

5 Best Use Cases

There are some limitations to the method discussed that must be noted. SOC is known to be a function of the cells open-circuit voltage (OCV). This method assumes that SOC is a function of the cells terminal voltage. This approximation is most accurate under the following conditions:

1. When the systems load profile resembles a constant current discharge (like what was used to characterize the battery). Having varying currents introduces some error that can be tolerable based on application and magnitude of current variance.
2. Low cycle-count batteries. High cycle-count batteries undergo changes in their resistance causing a different loaded voltage profile.
3. Low currents (< C/10). Batteries that are discharged at low C rates experience less voltage relaxation. If a low C rate discharge of a battery is halted (system going to sleep mode), the amount of SOC error introduced due to the upward relaxation is less than a battery discharging at a high rate.
4. Room temperature. At the extremes, hot and cold temperatures can shift a cell's OCV considerably. Expect increased error as the cell temperatures deviates from the temperature it was characterized at.

A voltage-based fuel gauge to consider is the BQ27621-G1. This gauge achieves higher accuracy by estimating the current based on the terminal voltage. This gauge is appropriate for low current applications where a true coulomb-counting gauge can have issues with current measurement accuracy. One such coulomb counting gauge to consider is the BQ27421-G1.

6 Python Lookup Table Generator

```
#Command Line Arguments: [TI format gauge output csv file], [Polynomial order for regression]
#Output: Plots VBAT vs SOC reported by TI gauge and creates a polynomial regression of the
specified order.
#        This regression is plotted on the same graph as the data and is mapped to a 101-pt
hexadecimal lookup table given in the "lookup_table.txt" file.


import matplotlib.pyplot as plt
import numpy as np
import csv
import sys



vbat_arr = []
num_reads = 0
read_arr = []
soc_arr = []

poly4x = []
poly4y = []
poly3y = []
poly3x = []

bin_vals = []

#reads in the TI Gauge csv file and puts the data into the corresponding list.
#Adjust this section if not using TI gauge
with open (sys.argv[1]) as csv_file1:
    csv_reader = csv.reader(csv_file1, dialect='excel',delimiter=',')
    line_count = 0
    for row in csv_reader:
        if (line_count > 8): #To get rid of the labels and other unnecessary stuff
            vbat_arr.append(float(row[6]))
            soc_arr.append(int(row[16]))
            num_reads += 1
        line_count += 1


#create a polynomial regression of the order specified in cmd line
polyfunc = np.polyfit(soc_arr, vbat_arr, int(sys.argv[len(sys.argv)-1]))
poly4 = np.poly1d(polyfunc)



#This for loop creates an x and y list from the regression such that it can be plotted later.
#It also calculates the hex values for the battery voltages needed to create the lookup table.
for i in range (0, 101):
    poly4y.append(poly4(i))
    poly4x.append(i)
    vbat_16 = int(round(((poly4(i)/1000)*(2**16))/6)) #Vbat formula found in datasheet
    bin_vals.append(hex(vbat_16))

#This for loop outputs the lookup table to the file called "lookup_table.txt"
with open ("lookup_table.txt","w+") as outfile:
    for i in range(0, 101):
        outfile.write(str(bin_vals[i])[0] + str(bin_vals[i])[1] + str(bin_vals[i])[2].upper() +
str(bin_vals[i])[3].upper() + str(bin_vals[i])[4].upper() + str(bin_vals[i])[5].upper() + ",\n")
#Ensures that hex letters are all uppercase
outfile.close()

#The rest of this is for plotting the data collected and the calculated regression
plt.plot(soc_arr, vbat_arr, 'r', label='Battery Data')
plt.yticks([3000, 3200, 3400, 3600, 3800, 4000, 4200, 4400])
plt.plot(poly4x, poly4y,'b',label='Regression')

plt.xlabel('SOC (%)')
plt.ylabel('Battery Voltage (mV)')
plt.gca().invert_xaxis() #Reverses x axis so that 100% is shown as the leftmost value
plt.title('Battery Voltage vs SOC')
plt.legend()
plt.grid(b=True, which='major', axis='both')

plt.show()
```

7 MSP430 Code Snippet

```
{...

    //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, 0xFE, &Err); //Mask all but VIN_PGOOD
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_MASK1, 0xBF, &Err); //Mask all
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_MASK2, 0xF7, &Err); //Mask all
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_MASK3, 0xFF, &Err); //Mask all

    // Enable ADC channels for VBAT only
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_ADC_READ_EN, 0x08, &Err);

    // Enable ADC
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_ADCCTRL0, 0x80, &Err); //Set ADC to perform conversion
every 1s at 24ms conversion speed
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_ADCCTRL1, 0x00, &Err); //Disables comparator channels

    //Set PG pin as GPIO for discharge
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_ICCTRL1, 0x08, &Err);
    StdI2C_P_TX_Single(BQ25150_ADDR, BQ25150_ICCTRL2, 0x10, &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);

    StdI2C_P_RX_Single(BQ25150_ADDR, BQ25150_ADCDATA_VBAT_M, &VBAT_Meas_M, &Err); //Finding current
battery voltage
    StdI2C_P_RX_Single(BQ25150_ADDR, BQ25150_ADCDATA_VBAT_M, &VBAT_Meas_L, &Err);
    VBAT_Meas = (uint16_t)(VBAT_Meas_M << 8) | VBAT_Meas_L; //Converting to 16-bit integer
    cur_SOC = find_initial_SOC(VBAT_Meas); //Finding initial SOC
    sprintf(SOC_string, "SOC = %d %%        ", cur_SOC);

    while(1) {

        waitms(1000); //Period between SOC updates
        StdI2C_P_RX_Single(BQ25150_ADDR, BQ25150_ADCDATA_VBAT_M, &VBAT_Meas_M, &Err);
        StdI2C_P_RX_Single(BQ25150_ADDR, BQ25150_ADCDATA_VBAT_L, &VBAT_Meas_L, &Err);
        VBAT_Meas = (uint16_t)(VBAT_Meas_M << 8) | VBAT_Meas_L;
        cur_SOC = update_SOC_discharge(cur_SOC, VBAT_Meas); //Update SOC
        sprintf(SOC_string, "SOC = %d %%        ", cur_SOC);


    }
}

uint8_t find_initial_SOC(uint16_t VBAT_Meas){
    int i;

    for (i = 99; i >= 0; i--){
        if (VBAT_Meas >= SOC_lookup_table[i]){
            return (uint8_t)(i + 1);
        }
    }
    return 0;
}

uint8_t update_SOC_discharge(uint8_t cur_SOC, uint16_t VBAT_Meas){
    int i;

    for (i = cur_SOC - 1; i >= 0; i--){ //Begins at current SOC so it doesn't have to go through
the whole array each time.
        if (VBAT_Meas >= SOC_lookup_table[i]){
            return (uint8_t)(i + 1); //Must add 1 to the SOC found because array is zero-indexed.
        }
    }

    return 0; //Default is SOC = 0.
}
```

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), 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, regulatory 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 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.

TI objects to and rejects any additional or different terms you may have proposed.

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