Using the nPM1300 and nPM1304 Fuel Gauge

Application Note

4490_439 / 2025-12-03

Contents

Revision history

Date	Description
December 2025	Added Fuel gauge configuration on page 11
August 2025	• Added content for nPM1304 • Moved content in Guidelines for battery profiling on page 16 to nPM PowerUP app Editorial changes
January 2025	Moved Generating a battery model on page 13 to Profiling a battery with nPM PowerUP in the nPM PowerUP app documentation
April 2024	Updated: • Recommended reading on page 18 Editorial changes
February 2024	Updated: • Fuel gauge overview on page 6 Added: • Fuel gauge power consumption on page 7 • Fuel gauge example for a Bluetooth gaming mouse on page 8 Editorial changes
December 2023	Updated Fuel gauge overview on page 6
October 2023	Updated: • Introduction on page 4 • Minimum requirements on page 5 • Fuel gauge overview on page 6 • Generating a battery model on page 13 • Added Evaluating a battery model on page 13 • Predicting a battery state of charge on page 14 • Guidelines for battery profiling on page 16 Editorial changes
July 2023	First release

1 Introduction

This application note describes the seamless integration of the fuel gauge solutions for the nPM1300 and nPM1304 *Power Management Integrated Circuit (PMIC)*s with a *System on Chip (SoC)* to ensure accurate and reliable monitoring of battery performance in low-power device applications.

State-of-charge estimation techniques based on open-circuit voltage methods often yield inconsistent results under fluctuating load and temperature conditions. The fuel gauge solution considers all these variations to provide a stable and accurate state-of-charge prediction.

The PMIC has a comprehensive fuel gauge that uses integrated battery current, voltage, and temperature measurements. The advanced fuel gauge algorithm combines battery measurements with a battery model to provide stable and precise state-of-charge predictions, with a typical error of less than ±3%, under rated operating conditions of the battery. With temperature compensation, it ensures exceptional accuracy across the battery's operating temperature range.

The fuel gauge algorithm can be run on most Nordic host SoCs for a reliable estimation of the battery state of charge. The battery model is generated by a one-time battery profiling process in the nPM PowerUP computer app.

For nPM1300, use the nPM1300 EK together with the nPM Fuel Gauge Board extension.

The nPM1304 EK includes battery profiling functionality to generate the battery model for the fuel gauge. The nPM Fuel Gauge Board is not needed.

The fuel gauge is also compatible with other non-Nordic SoCs. Contact your local Nordic sales representative for more information.

2 Minimum requirements

Before you start, check that you have the correct hardware and software.

Hardware requirements for nPM1300

- nPM1300 EK
- nPM Fuel Gauge Board
- Nordic *SoC* or *System in Package (SiP)*

The EK and the nPM Fuel Gauge Board are used to profile and generate the model. Once the battery model is extracted, you only need nPM1300 and the SoC (or SiP) in your application to do fuel gauging.

Hardware requirements for nPM1304

- nPM1304 EK
- Nordic *SoC* or *SiP*

The EK is used to profile and generate the model. Once the battery model is extracted, you only need nPM1304 and the SoC (or SiP) in your application to do fuel gauging.

Software requirements

- nPM PowerUP app available in nRF Connect for Desktop

3 Fuel gauge overview

The nPM1300 and nPM1304 fuel gauges provide ultra-low power battery fuel gauging without compromising on accuracy.

The fully integrated design eliminates the need for additional external components, simplifying your design process and reducing board space and cost. The fuel gauge can be turned off to reduce quiescent current (Iq) during sleep and power-down of the device.

To predict the battery state of charge, the nPM1300 and nPM1304 fuel gauges require a battery model to give a mathematical representation of a battery's behavior. Battery modeling is made possible through the nPM PowerUP computer app available in nRF Connect for Desktop. The calculated state of charge is relative to the battery's present maximum capacity and always shows 100% when the battery is fully charged, even if the actual capacity has reduced due to aging.

The nPM PowerUP battery profiling solution allows you to independently develop a battery model tailored to your specific needs. A one-time profiling of the battery in the nPM PowerUP app generates the battery model. The battery model can be further evaluated in the nPM PowerUP app before integrating it in your application. For more information, see the nRF Connect SDK samples at nPM13xx Fuel gauge sample.

The fuel gauge algorithm can run on Nordic host *SoC*s (nRF52, nRF53, nRF54, and nRF91 series) to ensure an accurate estimation of the battery state of charge by controlling and retrieving measurement information from the PMIC through *Two-wire Interface (TWI)* communication. Other non-Nordic SoCs can also be used. Contact your local Nordic sales representative for more information.

To configure fuel gauge intervals for an optimal synchronization with different activity levels of a device, consider both the average and peak battery current consumption for a given activity level relative to the capacity of the selected battery. This ensures a balance between accurate state-of-charge estimation and low-power fuel gauging.

For a typical *Bluetooth*® low energy application, it is recommended to run the fuel gauge algorithm once every second in the active state and every five seconds in the idle state. The fuel gauge is not required to operate during host sleep intervals. For more information, see Fuel gauge example for a Bluetooth gaming mouse on page 8.

The following diagram shows an overview of the nPM1300 and nPM1304 fuel gauges.

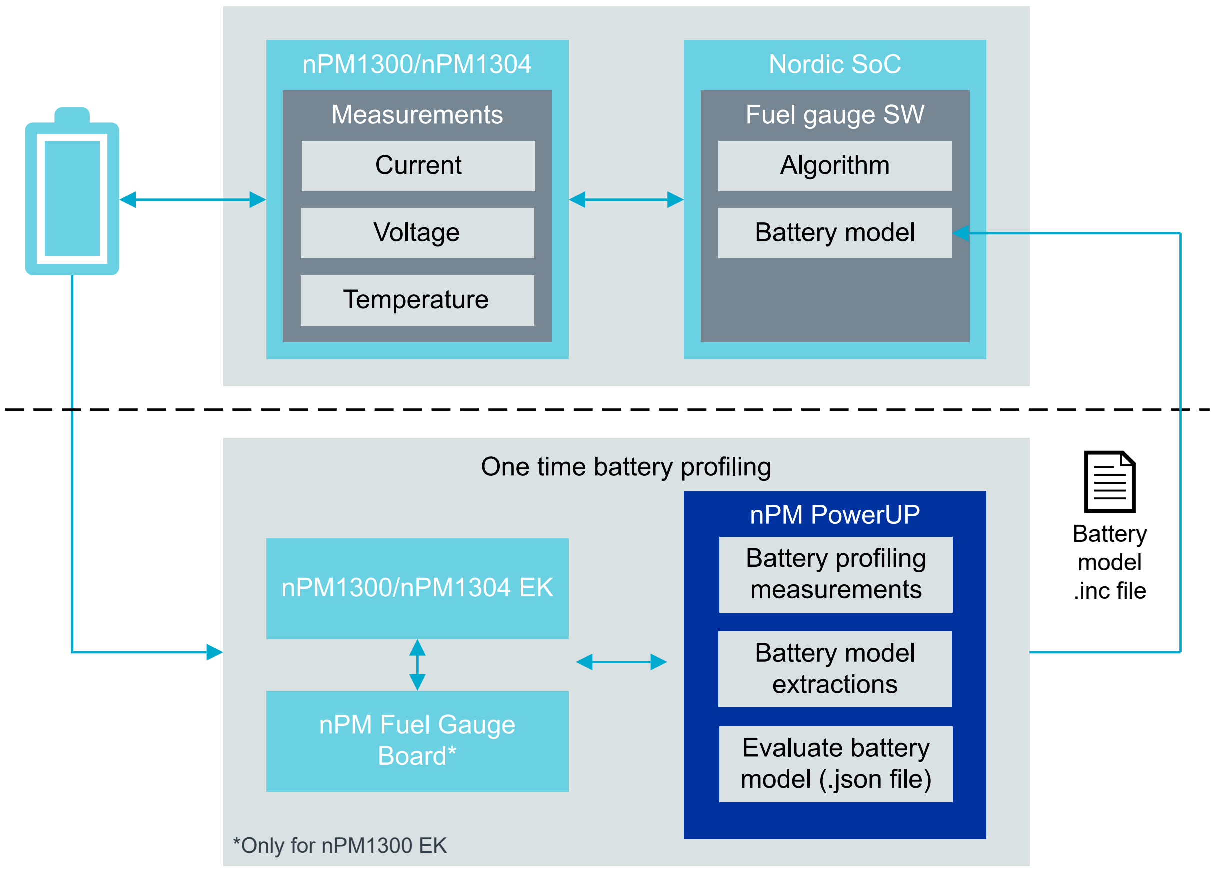

Figure 1: nPM1300 and nPM1304 fuel gauge with battery profiling

3.1 Fuel gauge power consumption

The total current consumption of a battery fuel gauge is affected by several factors such as operating mode, measurement frequency, *TWI* communication, and CPU usage. nPM1300 and nPM1304 deliver ultra-low power fuel gauging through an algorithm that is designed to operate exclusively during active periods of a device. This eliminates the need for system wake up during sleep intervals. Typical fuel gauge consumption is 8 µA to 10 µA in the active state when using a one-second fuel gauge iteration interval.

The following tables present examples for fuel gauge current consumption and resource usage when using the nRF5340 *SoC* as the host. All measurements are specified for a 3.7 V battery voltage at room temperature.

	CPU usage	**TWI communication**	**Measurements**
Time consumed (ms)	0.1	2.2	7.1
Average current (mA)	3.6	1.0	1.1
Charge per iteration (µC)	0.4	2.2	7.7
Total charge for each fuel gauge iteration (µC)	10.3		

Table 1: Fuel gauge example for a battery with NTC thermistor

	CPU usage	TWI communication	Measurements
Time consumed (ms)	0.1	1.5	5.6
Average current (mA)	3.6	1.0	1.1
Charge per iteration (µC)	0.4	1.5	6.0
Total charge for each fuel gauge iteration (µC)	7.9		

Table 2: Fuel gauge example for a battery without NTC thermistor

The average current values used in the tables are from laboratory measured values and the nRF5340 Product Specification:

- nRF5340 application CPU current, running CoreMark® from flash at 64 MHz clock
- nRF5340 current consumption for TWIM, transferring data at 400 kbps

3.1.1 Fuel gauge example for a Bluetooth gaming mouse

This example presents a typical load profile for a Bluetooth gaming mouse, including corresponding power consumption data for fuel gauging. The mouse operates in three distinct states: active, idle, and sleep.

The three states of the gaming mouse are shown in Figure 2: Typical load profile for a gaming mouse on page 9. In the active state, the gaming mouse is in a heightened operational mode, which responds to frequent and dynamic movements as the user interacts with it during gaming or similar activities.

In the idle state, the mouse enters a standby mode while remaining powered on. The mouse does not actively engage in user interaction but is ready for prompt responsiveness. The idle state has lower activity levels than the active state.

The sleep state is a low-power mode the mouse enters after being idle for a set period. This low-power mode significantly reduces energy consumption during extended periods of inactivity.

By customizing the iteration interval of the fuel gauge to match the different activity levels of the gaming mouse, the *PMIC* achieves a balance between accurate state-of-charge estimation and low-power fuel gauging.

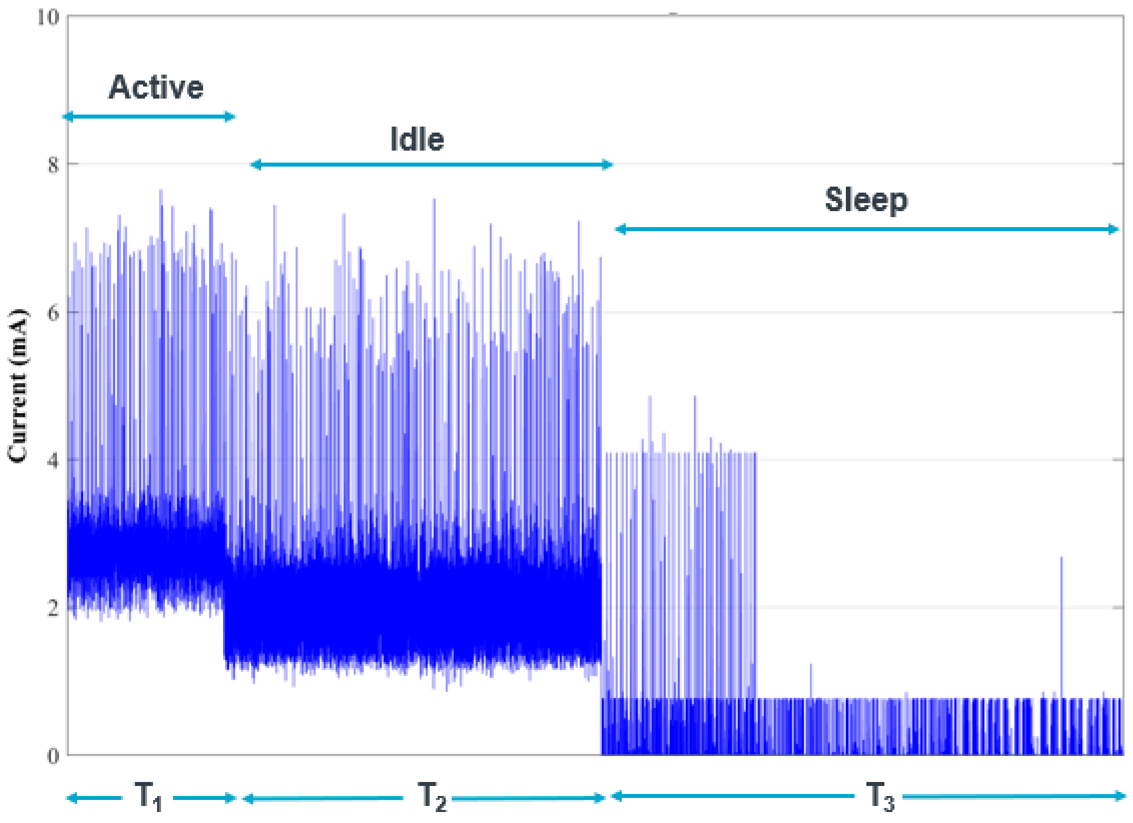

Figure 2: Typical load profile for a gaming mouse

The following table shows an example of fuel gauging for a mouse that has medium activity level and is used for moderate periods. The mouse battery does not have an NTC thermistor.

Device state	Time (h)	Average current (mA)	Fuel gauge iteration interval (s)	Device total charge (C/day)	Device average load current (mA/day)	Fuel gauging total charge (C/day)	Fuel gauging average current (µA/day)
Active	3	2.80	1	77.6	0.9	0.12	1.0
Idle	6	2.07	5				
Sleep	15	0.05	0				

Table 3: Fuel gauging for a gaming mouse with medium activity, used for moderate periods

The following table shows an example of fuel gauging for a mouse that has high activity level and is used for longer periods. The mouse battery does not have an NTC thermistor.

Device state	Time (h)	Average current (mA)	Fuel gauge iteration interval (s)	Device total charge (C/ day)	Device average load current (mA/day)	Fuel gauging total charge (C/ day)	Fuel gauging average current (µA/day)
Active	6	2.80	1	136.4	1.58	0.23	3.0
Idle	10	2.07	5				
Sleep	8	0.05	0				

Table 4: Fuel gauging for a gaming mouse with high activity, used for longer periods

4 Fuel gauge configuration

Configure the fuel gauge to manage initialization, memory, and sleep modes for accurate state-of-charge measurements and reliable operation.

4.1 Fuel gauge initialization

Initialize or reset the fuel gauge only when the battery is at rest and system activity is low.

The `nrf_fuel_gauge_init()` function uses the first measurement of battery voltage, current, and temperature. To ensure an accurate baseline, take this measurement when the battery is chemically stable, at rest, not charging, and not under a significant load.

The ideal moment to initialize or reset the fuel gauge is immediately after a reboot before you enable downstream peripherals or the charger. This ensures a minimal load on the battery.

Some initialization error might still occur, but the fuel-gauge algorithm corrects these errors gradually over time as the battery discharges.

The fuel-gauge library safely reinitializes every time a full-charge event has been completed. This ensures that the battery reports a state of charge close to 100% when it is fully charged and prevents the accumulation of estimation errors.

4.2 Fuel gauge RAM retention

The fuel gauge library uses a few hundred bytes of memory for run-time state management. Memory size depends on the fuel gauge version. For the best fuel gauge user experience, use RAM retention to ensure consistent and stable charge estimates. Without RAM retention, the fuel gauge must be reinitialized often. This causes initialization errors that reduce the prediction accuracy, but the algorithm corrects these errors over time.

In applications where the host *SoC* powers down in ultra-low power mode, RAM contents, including the fuel gauge state memory, are lost. To preserve fuel gauge continuity, use nonvolatile memory, such as RRAM or flash, to preserve the fuel gauge states before entering low-power modes. Upon wakeup, the fuel gauge can resume operations seamlessly and avoid the need for reinitialization.

To retrieve the fuel gauge state memory, use the `nrf_fuel_gauge_state_get()` function. The application must ensure that this state information is stored properly. To resume operation from a stored state, use the `state` parameter in `nrf_fuel_gauge_init_parameters` when initializing the fuel gauge. For more information, see nrfxlib API: nRF Fuel Gauge library.

4.3 Sleep current configuration

To save power, the fuel gauge is not required to operate during host *SoC* sleep intervals.

If the host stays powered and in sleep mode with RAM contents retained, use the `nrf_fuel_gauge_idle_set()` function to set the fuel gauge library to an idle state just before the host enters sleep mode. Notifying the library of idle periods increases the accuracy of predictions during extended periods of low activity with few battery measurements. If this function is not called, discharge currents during low-power states might be overestimated.

To provide the expected average current to the fuel gauge, use `nrf_fuel_gauge_idle_set()`. This input helps the library estimate battery current more accurately during the idle phase. When normal activity resumes, use `nrf_fuel_gauge_process()` to return the fuel gauge to active tracking. This significantly accelerates fuel gauge convergence, even after long inactivity. For more information, see nrfxlib API: nRF Fuel Gauge library.

5 Profiling a battery with nPM PowerUP

Use the nPM PowerUP app together with the nPM1300 *Evaluation Kit (EK)* or the nPM1304 EK to profile your battery and generate a battery model.

5.1 Generating a battery model

Follow the steps in Profiling a battery with nPM PowerUP to profile a battery and use the generated battery model to initialize and run the fuel gauge.

5.1.1 nPM Fuel Gauge Board for nPM1300 EK

The nPM Fuel Gauge Board is a cost-effective, plug-and-play extension board for profiling *Li-ion*, *Lithium-polymer (Li-Poly)*, and Lithium iron phosphate ($LiFePO_4$) batteries together with the nPM1300 *EK* and the nPM PowerUP app. The nPM1304 EK includes battery profiling functionality to generate the battery model, so the nPM Fuel Gauge Board is not needed.

The board is specifically designed for the battery discharge and data collection needed for battery modeling, eliminating the need for expensive measurement equipment or specialized expertise. Plug the nPM Fuel Gauge Board into the nPM1300 EK and configure it using the nPM PowerUP app to allow easy and convenient battery profiling without complicated setup procedures.

The nPM Fuel Gauge Board supports batteries with capacities up to 3000 mAh across the battery's operating voltage and temperature range. Larger batteries can be used, but profiling takes more time.

6 Predicting a battery state of charge

The estimated state of charge (%) from the fuel gauge has been compared against measured data using an accurate coulomb counter-based state-of-charge test for a *Li-Poly* battery at room temperature.

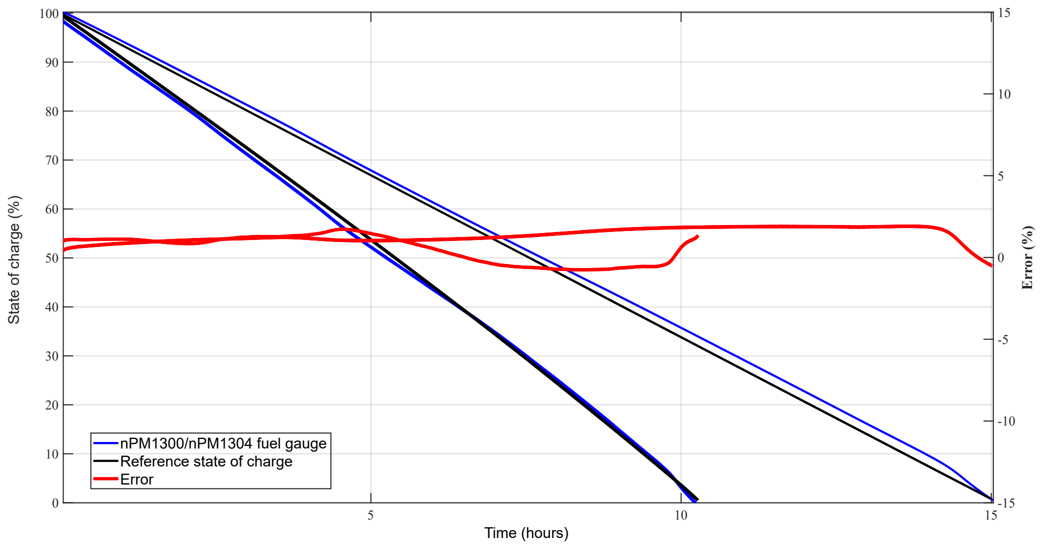

Figure 3: Estimated and measured state of charge (%) at different loads

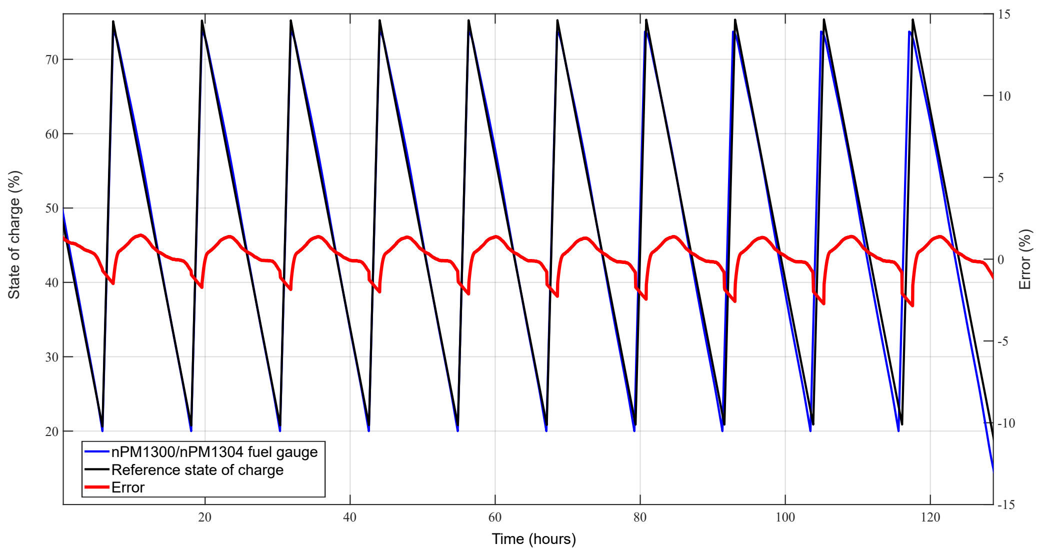

Figure 4: Estimated and measured state of charge (%) over several charge and discharge cycles

The following figure compares the fuel gauge with a reference state of charge to an Open-Circuit Voltage (OCV) based state-of-charge estimation method.

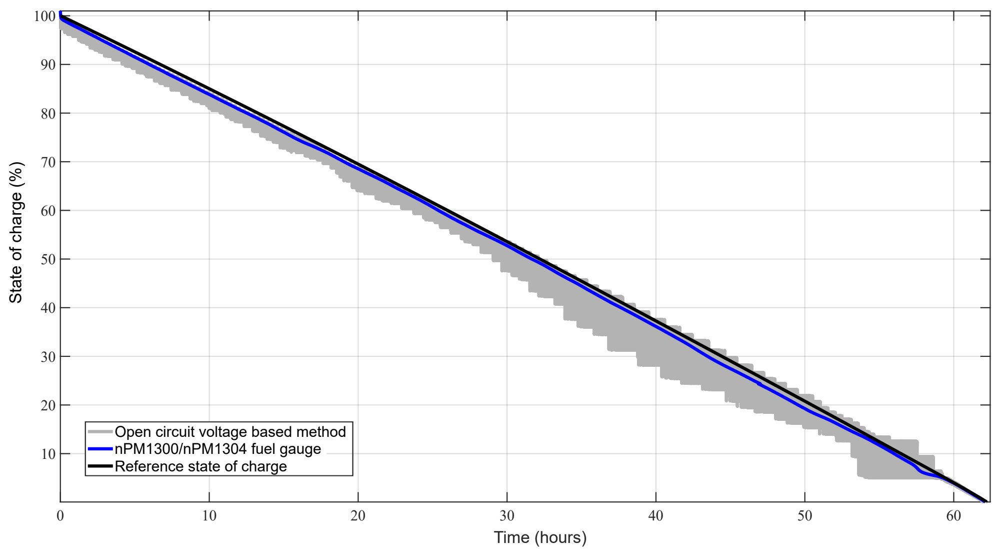

Figure 5: Comparing fuel gauge estimates against OCV based look-up table method

7 Guidelines for battery profiling

Follow the Guidelines for battery profiling in the nPM PowerUP app when profiling your battery.

Glossary

Evaluation Kit (EK)
A platform used to evaluate different development platforms.

Li-ion
Lithium-ion

Lithium-polymer (Li-Poly)
A rechargeable battery of lithium-ion technology using a polymer electrolyte instead of a liquid electrolyte.

Negative Temperature Coefficient (NTC)
A negative temperature coefficient refers to materials where there is a decrease in electrical resistance when their temperature is raised.

Power Management Integrated Circuit (PMIC)
A chip used for various functions related to power management.

System in Package (SiP)
Several integrated circuits, often from different technologies, enclosed in a single module that performs as a system or subsystem.

System on Chip (SoC)
A microchip that integrates all the necessary electronic circuits and components of a computer or other electronic systems on a single integrated circuit.

Two-wire Interface (TWI)
An I^2C compatible serial communication protocol that enables devices to exchange data by using a two-wire bus system, allowing multiple devices to be connected and controlled by a master device.

Recommended reading

In addition to the information in this document, you may need to consult other documents.

Nordic documentation

- nPM1300 Product Specification
- nPM1300 EK Hardware
- nPM1300 EK product page
- nPM Fuel Gauge Board Hardware
- nPM Fuel Gauge Board product page
- nPM1304 Datasheet
- nPM1304 EK Hardware
- nPM1304 EK product page
- nPM PowerUP app

Legal notices