Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Програмно-апаратне забезпечення системи контролю li-ion акумуляторних батарей

Виконав(ла): студент(ка) *VI* курсу, групи *СІм-61*

спеціальності *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

_____ Волоський В.П.
(підпис) (прізвище та ініціали)

Керівник _____ Луцик Н.С.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Тиш Є.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Осухівська Г.М.
(підпис) (прізвище та ініціали)

Рецензент _____ Дячук С.Ф.
(підпис) (прізвище та ініціали)

Тернопіль
2021

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет Факультет комп'ютерно-інформаційних систем та програмної інженерії
(повна назва факультету)

Кафедра Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри

(підпис) (прізвище та ініціали)

« » 2021 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістр
(назва освітнього ступеня)

за спеціальністю 123 «Комп'ютерна інженерія»
(шифр і назва спеціальності)

студенту Волоський Володимир Петрович
(прізвище, ім'я, по батькові)

1. Тема роботи Програмно-апаратне забезпечення системи контролю Li-ion акумуляторних батарей

Керівник роботи Луцик Надія Степанівна, доцент кафедри, кандидат технічних наук
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*28*» *10* 2021 року № *4/7-916*

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи технічні характеристики акумуляторних батарей та безпека їх есплатації відносно міжнародного стандарту IEC 61951-1:2017 "Secondary cells and batteries containing alkaline or other non-acid electrolytes - Portable sealed rechargeable single cells"

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ.1 аналіз предметної області та огляд існуючих систем контролю акумуляторних батарей
2 Вибір засобів та методів для пристрою bms
3 Розробка та тестування апаратно – програмної частини
4 Охорона праці та безпека в надзвичайних ситуаціях. Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1.Тема та мета роботи. 2. Актуальність роботи. 3. Електрична структурна схема пристрою. 4. Алгоритми роботи пристрою. 5. Схема структурна. 6. Графіки напруг при включеені. 6. Графік напруг при включеені. 7. Графіки напруг елемметів батареї при зардці/розрядці. 8. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці та	Осухівська Г.М., зав. Каф. КС		
безпека в надзвичайних	Стадник І.Я., проф. каф. ОХ		
ситуаціях			

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	28.10.2021	Виконано
2	Аналіз завдання	29.10.2021	Виконано
3	Виконання розділу 1	05.11.2021	Виконано
4	Виконання розділу 2	11.11.2021	Виконано
5	Виконання розділу 3	21.11.2021	Виконано
6	Виконання розділу 4	26.11.2021	Виконано
7	Оформлення пояснювальної записки	01.12.2021	Виконано
8	Оформлення графічного та презентаційного матеріалу	08.12.2021	Виконано
9	Перевірка роботи на антиплагіат	13.12.2021	Виконано
10	Попередній захист	16.12.2021	Виконано
11	Захист	21.12.2021	

Студент _____ Волоський В.П.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Луцик Н. Б.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Програмно-апаратне забезпечення системи контролю Li-ion акумуляторних батарей // Волоський Володимир Петрович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерних систем та мереж, група СІм – 61 // Тернопіль, 2021. С. – 66 , рис. – 31 , табл. – 1 , додатки – 2 , бібліографія – 24.

Ключові слова: АКУМУЛЯТОР, БАЛАНСУВАННЯ, ЛІТІЙ-ІОН, БМС, СТРУМ, НАПРУГА, ДАТЧИК, SOC, SOH, OCV.

Метою роботи є дослідження та розробка апаратно-програмного комплексу системи контролю літій-іонних акумуляторних батарей

В розділі 1 було розглянуто використання літій-іонних акумуляторів у техніці, їхні технічні характеристики та можливість їхнього вторинного використання.

В розділі 2 проведено аналіз існуючих алгоритмів пасивного балансування послідовно включених акумуляторів типу 18650. Вибрано мікросхеми вимірювання температури, напруги, струму та головний мікроконтролер.

В розділі 3 проведено розробку та тестування алгоритмів визначення внутрішнього опору, визначення температури, створення таблиць OCV та алгоритму балансування.

В розділі 4 описано основні вимоги експлуатації, відповідно до норм міжнародного стандарту ДСТУ EN IEC 62040-1:2020 "Системи безперебійного живлення. Частина 1. Вимоги щодо безпеки (EN IEC 62040-1:2019, IDT; IEC 62040-1:2017, IDT)", та норми охорони праці при використанні системи контролю акумуляторних батарей.

SUMMARY

Software and hardware for the Li-ion batteries control system // Voloskyi Volodymyr Petrovych // Ternopil National Ivan Pulyuy Technical University, Faculty of Computer Systems and Networks, SIM-61Group // Ternopil, 2021 // p. - 66, fig. - 31, table. - 1, , Add. - 2, Ref. - 24.

Keywords: BATTERY, BALANCE, LITHIUM ION, BMS, CURRENT, VOLTAGE, SENSOR, SOC, SOH, OCV.

The aim of the work is to study and develop the hardware and software complex of the control system of lithium-ion batteries

Section 1 considered the use of lithium-ion batteries in technology, their technical characteristics and the possibility of their secondary use.

Section 2 analyzes the existing algorithms of passive balancing of series-connected batteries of type 18650. Chips for measuring temperature, voltage, current and the main microcontroller are selected.

Section 3, the development and testing of algorithms for determining the internal resistance, determining the temperature, creating OCV tables and balancing algorithms.

Section 4 describes the basic requirements for operation in accordance with the norms of the international standard DSTU EN IEC 62040-1: 2020 “Uninterruptible power supply systems. Part 1. Safety requirements (EN IEC 62040-1: 2019, IDT; IEC 62040-1: 2017, IDT)”, and labor protection standards when using the battery control system.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

АКБ – акумуляторна батарея

АЦП – аналогово-цифровий перетворювач

МНД-транзистор – Транзистор метал-діелектрик-напівпровідник

BMS – battery management system (система контролю акумуляторних батарей)

CAN – Controller Area Network (елемент постійної фази)

CPU – central processing unit (центральний процесор)

CRC – Cyclic redundancy check (циклічний надлишковий код)

CS – Chip Select

ESS – Energy Exchange Solutions

EV – electric vehicles (електромобіль)

GND – ground

HEV – hybrid electric vehicle (гібридних автомобілів)

isoSPI – Isolated Serial Peripheral Interface

Li-Ion – lithium-ion battery (літій-іон)

LVCMOS - Low voltage complementary metal oxide semiconductor

MCU – Microcontroller Unit

MISO – Master In Slave Out

MOSI – Master Out Slave In

NERL – National Renewable Energy Laboratory

OCV – Open-circuit voltage (Напруга розімкнутого кола)

PGA - Pin grid array

PHEV – plug-in hybrid electric vehicle (плагін-гібридів)

PTC – Positive temperature coefficient

PWM – pulse-width modulation (широтно-імпульсна модуляція)

SCLK – Serial Clock

SoC – State of charge (рівень заряду)

SOH – State of health (рівень залишкової ємності від початкової)

SOS – State of Safety (рівень безпеки)

SPI – Serial Peripheral Interface (послідовний периферійний інтерфейс)

TSSOP – Thin Shrink Small Outline Package

UART – universal asynchronous receiver/transmitter (універсальний асинхронний приймач/передавач)

ЗМІСТ

ВСТУП

Актуальність теми роботи: Акумуляторні батареї є важливою складовою сучасних технологій, у зв’язку з цим виникає проблема правильного використання даного ресурсу. Для забезпечення надійності АКБ, використовуються системи управління акумуляторними батареями. Оскільки у літій-іонних акумуляторних батареях елементи ніколи не можуть бути абсолютно однаковими через умови навколишнього середовища, процедуру виготовлення. Дані системи мають на меті збільшення терміну служби, надійності та оптимізації використання енергії акумуляторів.

Система управління акумулятором має завдання збільшення терміну служби акумулятор при оптимізації доступності та надійності. Щоб досягти високих напруг, багато окремих елементів з'єднані послідовно. Через те що окремі елементи в батареї відрізняються виготовленням і різноманітністю, старіння відбувається по-різному, загальна продуктивність системи обмежена найслабшим або найбільш зарядженим елементом. Крім того, кожен елемент має індивідуальну напругу і температурну характеристику. У випадку коли контролюється тільки напруга стека, існує ризик, що елемент зарядиться до високої напруги. Це може призвести до безповоротних пошкоджень АКБ, пожежі або вибуху. Для запобігання згаданим розбіжностям елементів по напрузі та ємності та забезпечення безпечних умов використання акумуляторних батарей використовуються системи контролю та балансування. Отже, без відповідної системи вирівнювання, напруга на окремих елементах буде змінюватися з часом, а ємність батареї швидко зменшуватиметься

Тому актуальною є розробка апаратно-програмного комплексу керування процесом заряду, розряду та балансування літій-іонних акумуляторних батарей.

Метою роботи: є збільшення терміну роботи літій-іонних акумуляторних батарей методом пасивного балансування. Розробка програмно апаратної системи управління літій-іонними акумуляторами з можливістю балансування кожного елемента.

Для досягнення вказаної мети, в роботі були поставлені такі задачі:

– аналіз алгоритмів балансування та систем управління АКБ;

– формування вимог до системи керування з максимальним часом безвідмовної роботи системи;

– розробка моделі системи контролю акумуляторних батарей;

– перевірка на практиці, в реальних умовах теоретично побудованих методів та засобів.

Об’єкт дослідження: процес забезпечення безперебійного живлення об’єкту.

Предметом дослідження: система керування літій-іонними акумуляторними батареями з використання балансування.

Методи дослідження. Теоретичною основою дослідження є класифікація й аналіз існуючих моделей алгоритмів роботи BMS та алгоритмів балансування акумуляторних батарей. Після чого необхідно провести математичний аналіз та оптимізацію алгоритму з абстрагуванням від неважливих елементів системи.

- Провести огляд існуючих публікацій у сфері систем управління батареями.
- Розробити модель АКБ з системою управління.
- Провести тестування існуючих алгоритмів.
- Визначення переваг та недоліків методів балансування.
- Сформувати вимоги до нової системи з врахування переваг та недоліків попередньо досліджених систем.
- Розробка алгоритму відповідно до вимог.
- Тестування алгоритму існуючої системи, визначення переваг та недоліків.

Наукова новизна одержаних результатів:

- Проаналізовано існуючі алгоритми пасивного балансування та системи контролю літій-іонних акумуляторних батарей та на основі отриманих результатів було розроблено власний алгоритм балансування.
- Досліджено вплив розробленого алгоритму пасивного балансування на тривалість роботи системи.

Практичне значення одержаних результатів. Впровадження алгоритму пасивного балансування літій-іонних акумуляторних батарей, який покращує ефективність балансування послідовно включених елементів, та розроблено систему контролю акумуляторних батарей.

Публікації. Дані досліджень були публіковані у наступних збірниках: X Міжнародної науково-практичної конференції молодих учених та студентів «Актуальні задачі сучасних технологій» (24-25 листопада 2021 року) Тернопільського національного технічного університету імені Івана Пулюя та IX науково-технічна конференція «Інформаційні моделі, системи та технології» (8 – 9 грудня 2021 року) Тернопільського національного технічного університету імені Івана Пулюя.

Структура роботи. Робота складається з пояснювальної записки та графічної частини. Пояснювальна записка складається із вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. Обсяг роботи: пояснювальна записка – 66 арк. формату А4, графічна частина – 8 аркушів формату А1.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОГЛЯД ІСНУЮЧИХ СИСТЕМ КОНТРОЛЮ АКУМУЛЯТОРНИХ БАТАРЕЙ

1.1 Призначення системи контролю Li-ion акумуляторів

Система управління акумулятором має завдання збільшення його терміну служби та надійність для повсякденного використання. За структурою літій-іонна акумуляторна батарея автомобілів складається з багатьох елементів, які мають послідовно-паралельне підключення. Паралельне підключення використовується для збільшення ємності АКБ а послідовне для досягання високих напруг. Контроль стану батареї відбувається з допомого стеження за станом кожного послідовно включеного елемента. Через те що окремі елементи в батареї відрізняються виготовленням, старіння відбувається по різному, загальна продуктивність система обмежена найслабшим або найбільш зарядженим елементом. Крім того, кожен елемент має індивідуальну напругову і температурну характеристику. Тому, якщо тільки напруга стека контролюється, існує ризик, що елемент досягне високої напруги і приведе до безповоротніх пошкоджень, пожежі або вибуху. Щоб запобігти згаданій ситуації, окремо напруги повинні бути виміряні та відрегульовані для кожної банки. Сьогодні, щоб звести до мінімуму проблему різних елементів, як правило, використовуються методи пасивного керування процесом зарядки, вирівнювання та збалансування напруг елементів або активного керування, які передають енергію між елементами. Тому в практиці розглядаються плюси і мінуси методів балансування, а також відповідні стратегії управління. Впровадження моделей для вирішення, коли це буде корисно і доречно застосовувати пасивну або активну систему.

Серед усіх технологій акумуляторів, літій-іонні батареї розглядаються більшістю виробників автомобілів. Тим не менше, безпекові аспекти все ще відіграють важливу роль, особливо для літій-іонних акумуляторів. Межі експлуатації, такі як максимальна/мінімальна напруга елемента, температура або

струм, не можна перевищувати. Електромобіль містить в собі не одну батарею. Кожна з них містить безліч елементів, які з'єднані послідовно або паралельно, або обидва варіанти. Ці елементи не рівні, навіть якщо вони приходять від того самого постачальника і від тієї самої виробничої лінії, оскільки виробництво акумуляторів є доволі складним і однорідність якості все ще залишається великим викликом для виробників акумуляторів. Крім того, через різну швидкість саморозряду або старіння, загальний стан продуктивності системи обмежений найслабшим або найбільш зарядженим елементом. Для запобігання згаданих розбіжностей елементів у напрузі та ємності, використовуються системи балансування елементів. Ці системи поділяються на дві групи, дисипативні (пасивні) та недисипативні (активні) системи балансування. Пасивні системи розсіюють надлишок енергії за допомогою резисторів паралельно елементам. Схема підключення акумуляторів для пасивного балансування представлена на рисунку 1.1 де в якості балансуючої мікросхеми використовується LTC6810.

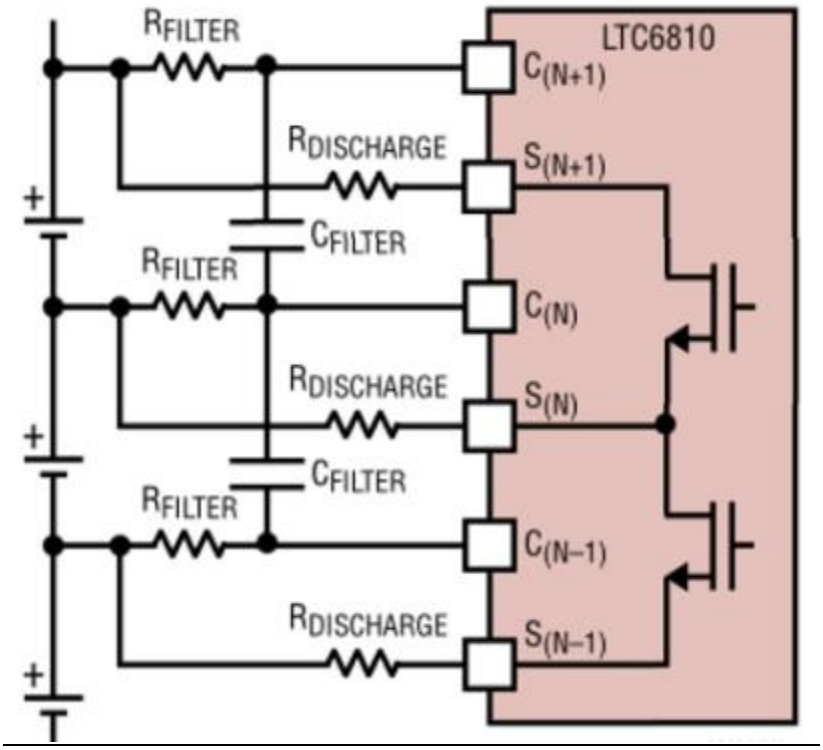

Рис. 1.1 Схема підключення акумуляторів до мікросхеми пасивного балансування

В активних системах, енергія сильніших елементів передається слабшим осередкам, що використовують конденсатор, індуктор або топології на основі перетворювача [1]. На рисунку 1.2 зображено мікросхему активного балансування LTC3300 – це автономний двонаправлений зворотний контролер для літієвих і LiFePO4 акумуляторів, який забезпечує балансувальний струм до 10 А; оскільки він є двонаправленим, заряд з будь-якої вибраної комірки може передаватися з високою ефективністю до сусідніх елементів. Один LTC3300 може збалансувати до шести елементів.[2, 3]

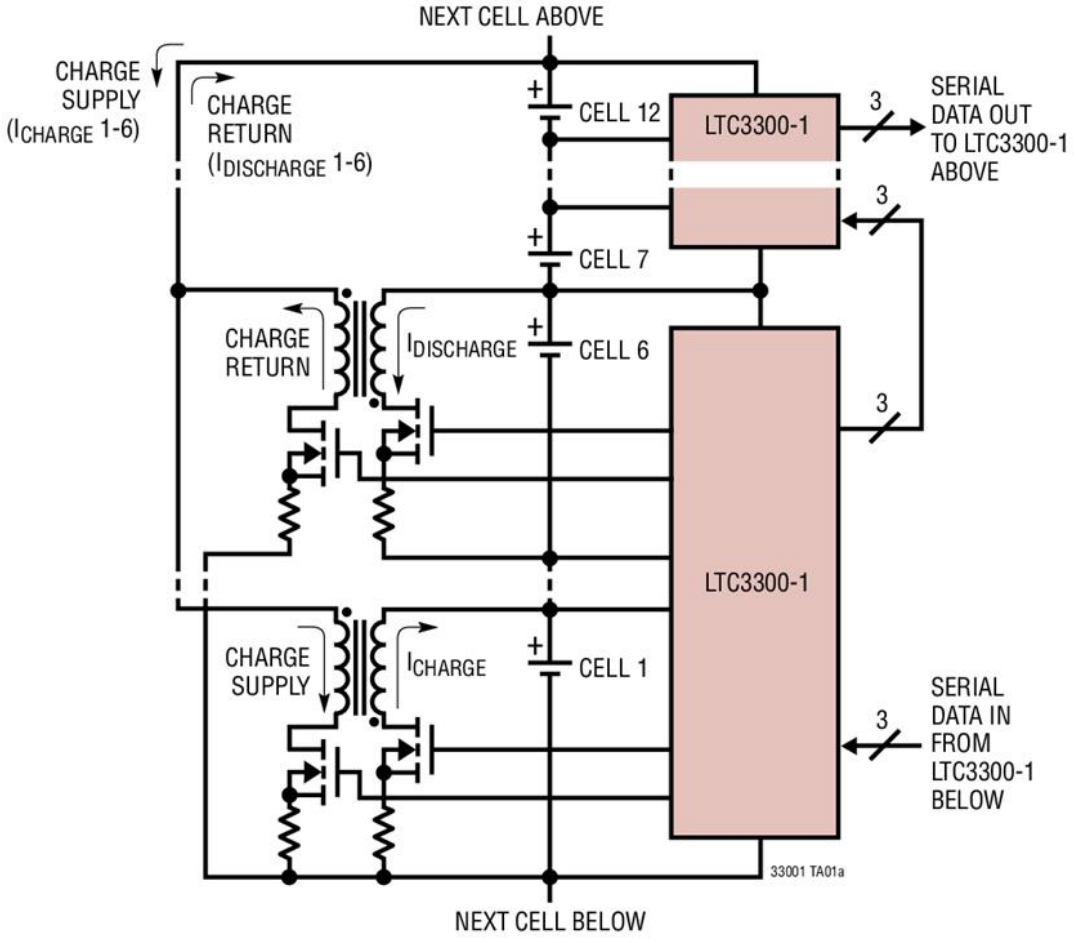

Рис. 1.2. Схема підключення акумуляторів до мікросхеми активного балансування

Розробкою та вдосконаленням даних систем займаються провідні компанії автомобілебудування такі як: Nisan, Tesla, Skoda, та інші [2].

1.2 Повторне використання Li-ion акумуляторів

Повторне використання акумуляторних батарей є важливим пунктом у сучасному суспільстві оскільки кількість електромобілів зростає, а разом з цим збільшується кількість АКБ. Виходячи з описів виробників та наявної літератури, як тільки акумулятори EV досягли 70–80% своєї номінальної ємності, вважається, що їх роль як акумуляторів першого терміну служби EV досягла кінця, тому що такі батареї призведуть до меншого пробігу та швидкості. Відсоток залишкової ємності являє собою SOH акумулятора. За оцінками, ця деградація відбувається через 5–8 років використання або еквівалентно 100 000 миль (160 000 км) подорожі. Тим не менш, виведені з експлуатації батареї електромобілів, навіть із меншим SOH, все ще можуть бути використані для інших застосувань, таких як житлові домогосподарства або розбіжність потужності в електромережевих фотоелектричних установках. Завдяки вторинному використанні загальна тривалість життя акумулятора збільшується, а вартість акумулятора може бути зменшена як для основних, так і для вторинних користувачів. Як зазначено в дослідженнях NERL очікується що акумулятори які використовуватися в автомобільній індустрії збережуть приблизно 70% від їхньої початкової ємності та потенційно зможуть працювати ще 10 років при їх правильному повторному використанні [3].

Повторне використання акумуляторних батарей для електромобілів для використання вторинного терміну служби може сприяти скороченню викидів CO2 на 56% порівняно з природним паливним газом, особливо під час пікового попиту [4]. Крім екологічної вигоди від перепрофілювання батарей для електромобілів, це могло б стати джерелом доходу від продуктів, які все одно було б утилізовано.

Використання цих батарей у різних програмах потребуватиме певної інтеграції та перебудови. Стандартна напруга для типових промислових систем становить близько 800–1000V [5]. Порівняння вимог батарей електромобіля та повторного промислового використання представлені в табл. 1.1.

Таблиця 1.1

Вимоги до акумуляторних батарей

	Автомобільні батареї	Батареї вторинного використання в промисловості
Напруга,V	400	800-1000
Час роботи при навантаженні 10A, h	~16800	~87800
Температура навколишнього середовища,°C	Від -40 до 60	Від 10до 35
Тривале навантаження	2-3 C	<0,5 C
Моментальне навантаження	>5 C	>2 C
Регулювання температури	Активне	Пасивне/Активне
SOH, %	Від 100 до 70	Від 80 до 70 (для вторинного використання)

З таблиці 1.1 видно що акумуляторні батареї є придатними для вторинного використання, як стаціонарні накопичувальні станції, які зможуть забезпечити електроенергією будинок або зарядити автомобіль при відсутності зовнішнього постачання. Щоб вирішити цю проблему, Tesla розробила систему під назвою Powerwall, яка являє собою акумуляторну батарею, що живиться від сонячних панелей, яка зберігає енергію для використання під час відключень електроенергії. Сховище можна використовувати як для домашнього використання, так і для зарядки електромобілів.

У 2019 році Tesla впровадила гібридні нагнітачі, які використовують сонячну енергію та систему накопичення енергії батареї. Ці гібридні нагнітачі мають схожу концепцію з Powerwall, де вони використовуються для швидкої зарядки електромобілів під час відключення електроенергії. В якості альтернативи для нових акумуляторів, які використовуються в Tesla Powerwall та інших подібних ESS, можна використовувати систему з вторинних акумуляторів для накопичення енергії та використання її під час відключень або великого навантаження на загальну мережу.

У нещодавній співпраці між Nissan та 4R Energy Corp використані батареї, з електромобілів Nissan Leaf, надсилаються на завод 4R Energy Corp для оцінки. Потім батареї поділяють на три класи, а саме батареї класу А - ці батареї все ще знаходяться в чудовому стані та придатні для повторного використання в акумуляторному блоку електромобіля; батареї класу В - які будуть використовуватися для промислового обладнання, такого як вилкові навантажувачі та великі ESS; і батареї класу С - ці батареї будуть використовуватися в резервних блоках живлення [6].

Однак ці акумулятори повинні пройти кілька процесів, перш ніж почати процес використання їх для вторинного використання. Життєвий цикл батареї буде проходити кілька етапів, таких як повернення батареї, оцінка, інтеграція та переробка.

Якщо розглядати систему яка використовується вторинні акумулятори як резервний блок живлення - тоді необхідно передбачати специфіку BMS для даної установки. Оскільки акумулятори бувають з різним рівнем зношуваності тому необхідно враховувати це при проектуванні системи.

1.3 Методи зарядки Li-ion акумуляторів

Є декілька методів для зарядки Li-Ion елементів. Один із методів полягає в застосуванні постійної напруги з обмеженим струмом до акумулятора протягом трьох годин, а потім зупинки. Використовуючи цей метод, батарея буде заряджена на 100% через 3 години за умови, що струм заряду встановлений в межах від С1 і С/2 [7].

Другий, подібний метод полягає в застосуванні постійної напруги, обмеженої струмом, до акумулятора під час контролю струму заряду. Під час першої частини циклу зарядки зарядний пристрій перебуває в режимі постійного струму, при цьому напруга акумулятора повільно зростає. Коли напруга буде наближається до запрограмованої постійної (плаваючої) напруги, струм заряду починає

експоненціально падати. Коли струм заряду падає до низького значення приблизно 0.1C , зарядний пристрій припиняє заряджатися як це показано на рисунку 1.3.

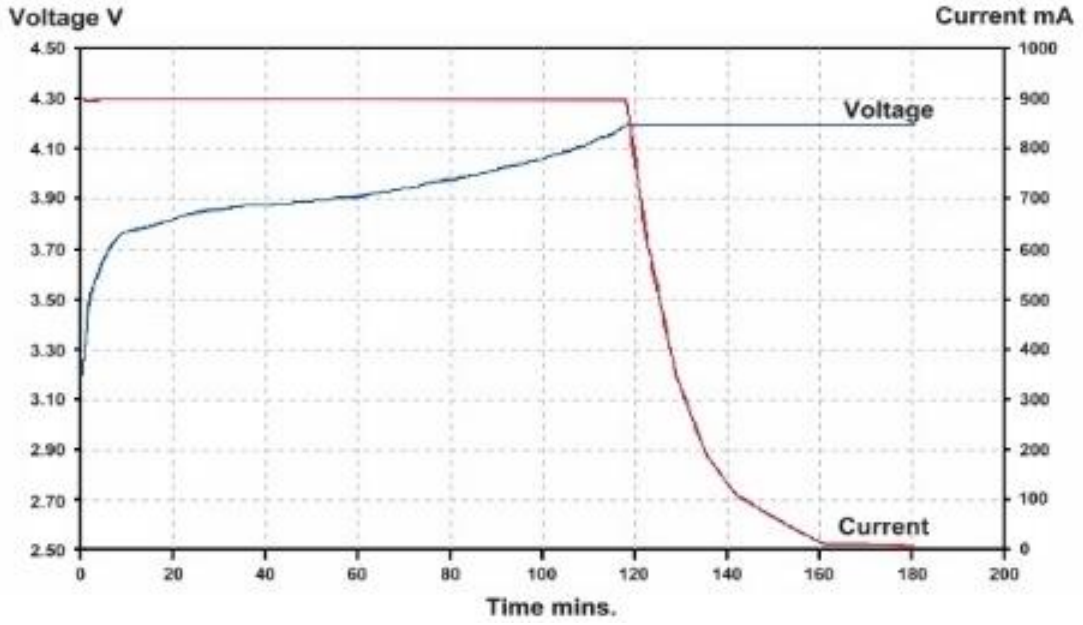

Рис. 1.3. Графік зарядки одного Li-ion акумулятора

Залежно від вибраного мінімального струму зарядки, акумулятор заряджений на 95% до 100%. Оскільки літій-іонні акумулятори не здатні поглинати перезаряд, весь зарядний струм повинен припинитися, коли акумулятор повністю зарядиться [8].

1.4 Методи захисту акумуляторних батарей реалізовані на BMS

Для захисту Li-ion акумуляторних батарей використовуються як апаратні так і програмні методи. Це означає що система керування акумулятором необхідна також для забезпечення безпеки експлуатації. BMS зазвичай оснащений електронним перемикачем, реле або транзистором, який від'єднує акумулятор від зарядного пристрою або навантаження в критичних умовах, які можуть призвести до небезпечних реакцій. До таких умов можна віднести наступні:

– перезаряд: це коли акумулятор заряджається до вищої напруги ніж це зазначено в документації для Li-ion акумулятора (це значення є 4.2V);

– висока та низька температура: це коли внутрішня температура елементів акумулятора перевищує їхні безпечні робочі діапазони температур здебльшо дані температури коливаються в межах від -20°C до 50°C.

– перерозряд: коли батарея розряджена нижче дозволеної мінімальної напруги для Li-ion акумулятора це значення є 3.0V.

– перевантаження по струму: це коли батарея піддається короткому замиканню або великому споживанню струму для Li-ion акумуляторів, може становити 30A.

– зворотна полярність: це коли клеми акумулятора неправильно підключені до пристрою.

Для забезпечення безпеки за вказаними вище параметрами необхідне як програмне так і апаратне забезпечення для прикладу На рисунку 1.4 показана схема відключення акумулятора при зниженні напруги. Схема контролює напругу літій-іонної батареї та відключає навантаження, щоб захистити акумулятор від глибокого розряду, коли його напруга падає нижче допустимої.

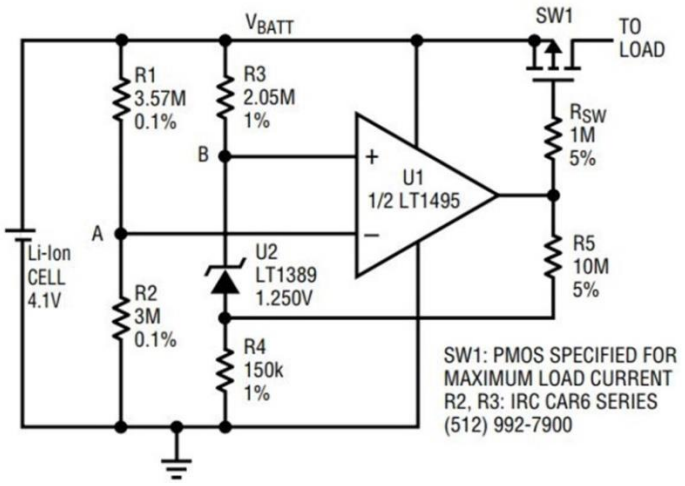

Рис. 1.4. Схема блокування при зниженні напруги

Режим роботи з низьким зарядом батареї вказується, наприклад, коли мобільний телефон автоматично вимикається після того, як індикатор низького заряду батареї блимає деякий час. Якщо телефон залишити в такому стані на кілька місяців, схема захисту, показана на рисунку 1.4, не призведе до надмірного розрядження та

пошкодження акумулятора, оскільки захисна схема споживає менше 4,5 мкА струму. Для інших захисних схем, які зазвичай вимагають більшого струму, швидкість розряду є більш високою, що дозволяє напрузі акумулятора впасти нижче безпечної межі за менший час [9].

1.5 Огляд існуючих систем безперебійного живлення на базі Li-ion акумуляторів

Системи безперебійного живлення складаються з двох основних частин це інвертор та акумлятортор. В процесі дослідження існуючих систем, які використовують літій-іонні акумулятори в якості енергосховища, було розглянуто ряд існуючих рішень [16].

Однією з існуючих систем контролю літі-іонними акумуляторними є Orion BMS яка спеціально розроблена для забезпечення виконання жорстких вимог захисту та управління акумуляторними батареями для електромобілів (EV), плагін-гібридів (PHEV) та гібридних автомобілів (HEV) з автомобільною маркою якості.

Orion BMS реалізує великий список функцій, призначених для захисту акумуляторної батареї (рис 1.5). Ці особливості включають [15]:

- Розрахунок проценту заряду.
- Захист акумулятора від перезаряду та перерозряду.
- Інтелектуальне балансування акумулятора (пасивне).
- Управління зарядним пристроєм.
- Моніторинг температури.
- Стежить за справністю акумулятора.

Рис. 1.5. Orion BMS

Елементи захищені від перепаду напруги, перевантаження по струму, перегріву та переохолодження на основі запрограмованих мінімальних та максимальних значень у профілі акумулятора.

Забезпечується інтелектуальне, ефективне балансування між елементами, щоб максимально збільшити корисний діапазон батареї. BMS також відслідковує стан окремого елементу та загальної батареї та повідомлятиме про помилку, якщо група або окремі елементи мають стан, непридатний для подальшої роботи.

Система Orion BMS розроблена для роботи в середовищах з сильним електричним шумом, наприклад, в електромобілях. Централізована конструкція дозволяє Orion BMS витримувати дуже значні EMI.

Система Orion BMS розроблена в основі, щоб її можна було легко програмувати. Він має гнучкість для використання як в автомобільному, так і в стаціонарному застосуванні [15].

Надане програмне забезпечення підключається до BMS Orion через can шину і може бути використано для моніторингу акумуляторної батареї, доступу та видалення кодів помилок, оновлення системних налаштувань та зміни інформації про профіль акумулятора [15].

Налаштування профілів акумулятора включають:

– мінімальна і максимальна напруги;

- граничні значення максимального струму;
- таблиці компенсації температури та напруги;
- налаштування управління зарядним пристроєм;
- налаштування зв'язку;
- багато інших параметрів.

Попередньо розраховані межі струму дуже важливі для застосування в транспортних засобах, де комп’ютер керування приводом повинен швидко приймати рішення на основі потужності, доступної від електричної системи. Відсутність надійних меж може призвести до того, що головний комп'ютер керування буде споживати забагато струму з акумулятора. Щоб дотримуватися обмеження, головний комп’ютер буде змушений зменшити струм [15].

Розрахунки лімітів враховують справність акумуляторної батареї, внутрішній опір, температуру акумулятора, а також забезпечують встановлення максимальних меж у програмованому профілі акумулятора для розряду струму при різних температурах. Значення можуть бути виражені в амперах або кіловатах.

Система Orion BMS розраховує стан заряду акумулятора, головним чином, відстежуючи струм, що надходить і виходить із батареї, методику, яку зазвичай називають кулоновим підрахунком [15].

Хоча цей метод є точним, жодна система ніколи не є досконалою. Через це Orion BMS використовує налаштування профілю акумулятора для пошуку розбіжностей між розрахунковим станом заряду (на основі напруги) та виміряним станом заряду (на основі підрахунку кулонів). Коли будуть виявлені відмінності, BMS призведе до того, що розрахований стан заряду буде «дрейфувати» до правильного стану заряду.

Деякі інші системи BMS миттєво виправляють стан заряду, а не дрейфують. Хоча це може здатися не проблемою, часто зовнішні системи контролюють стан заряду акумулятора і приймають рішення на основі цієї кількості. Дрейф SoC дозволяє з часом регулювати стан заряду, попереджає інші системи та запобігає коливанням.

Однією з важливих особливостей системи управління акумуляторами є надання користувачеві загального стану акумуляторної батареї. Система Orion BMS контролює внутрішній опір кожного елементу та відстежує ємність найслабшої клітини. Він використовує цю інформацію для розрахунку відсотка здоров'я елементу у відсотках від 0 до 100%. Потім BMS порівнює обчислену інформацію про стан елементу із попередньо запрограмованими пороговими значеннями, і якщо будь-які осередки (або весь пакет) досягають порогового значення, встановлюється код несправності [18].

Система Orion BMS використовує інтелектуальний підхід до балансування, який прагне підтримувати та покращувати баланс від циклу до циклу.

Orion BMS використовує пасивне балансування для розрядки найбільш заряджених елементів для підтримки рівноваги в акумуляторній батареї. Пасивні шунтуючі резистори розсіюють до 200 мА на елемент. Хоча ця кількість може здатися невеликою, цього струму більш ніж достатньо для підтримки рівноваги у дуже великих акумуляторах. Різниця в швидкостях саморозряду елементу часто вимірюється від десятків до сотень мкА (при цьому мкА становить 1/1000 мА.) Навіть при дуже високій різниці в швидкості саморозряду 1мА, балансуючий струм 200мА перевищує швидкість розряду в 200 разів [15].

1.6 Висновки до розділу 1

У даному розділі розглянуто літій-іонні акумулятори та їхнє використання в: енергетиці, автомобілебудуванні, електроінструментах, медичному обладнанні та ін. Незважаючи на їхню поширеність, у них також є ряд недоліків таких, як:

– необхідність схеми захисту від перезаряду, перерозряду та обмеження по струму;

– необхідність захисту від механічних пошкоджень;

– швидке старіння акумуляторів;

– для створення великих напруг необхідне з’єднання багатьох акумуляторів послідовно;

– складність у зберіганні.

Для зменшення впливу даних недоліків Li-ion акумуляторів використовують системи контролю акумуляторних батарей, її завданнями є:

– моніторинг напруги, струму, температури;

– захист від перевантаження по струму;

– захист акумуляторів по напрузі;

– виявлення замикання на землю або короткого замикання;

– балансування;

– комунікація з контролерами верхнього рівня;

– розрахунок стану заряду(SoC), старіння(SOH), безпеки(SOS), енергії, внутрішнього опору.

РОЗДІЛ 2
ВИБІР ЗАСОБІВ ТА МЕТОДІВ ДЛЯ ПРИСТРОЮ BMS

2.1 Вибір алгоритму балансування

Для коректної роботи акумуляторної батареї необхідно забезпечити можливість балансування окремих її елементів. Складність вибору алгоритму полягає в тому що, найчастіше використовувані алгоритми пасивного балансування є простими і мають низьку точність, або складними і дають високу точність. На сьогоднішній день виділяють так алгоритми балансування які базується на:

- поточній напрузі кожної комірки батареї;
- напрузі при розімкненому колі Open-circuit voltage (OCV);
- проценту заряду комірки State of charge (SoC) [10].

Метод балансування батарей який базується на поточній напрузі кожної комірки батареї є простим в реалізації, але може бути не ефективними. Він заснований на припущення, що елементи мають однаковий SoC, якщо їх виміряна напруга на клем теж рівна.

На рисунку 2.1 представлено алгоритм балансування в залежності від моментальної напруги. Першим етапом визначається напруга кожного елементу акумулятора після чого поводиться перевірка чи різниця мінімальної V_{bat_min} та максимальної V_{bat_max} напруги відрізняються більше ніж допустиму різницю V_{delta}. Якщо різниця напруг є більшою включається балансування усіх банок з напругою вищою за мінімальну на різницю. Балансування буде продовжуватися до тих пір поки усі банки не стануть в допустимих межах, тобто різниця між мінімальною та максимальною напругою буде менша ніж V_{delta}.

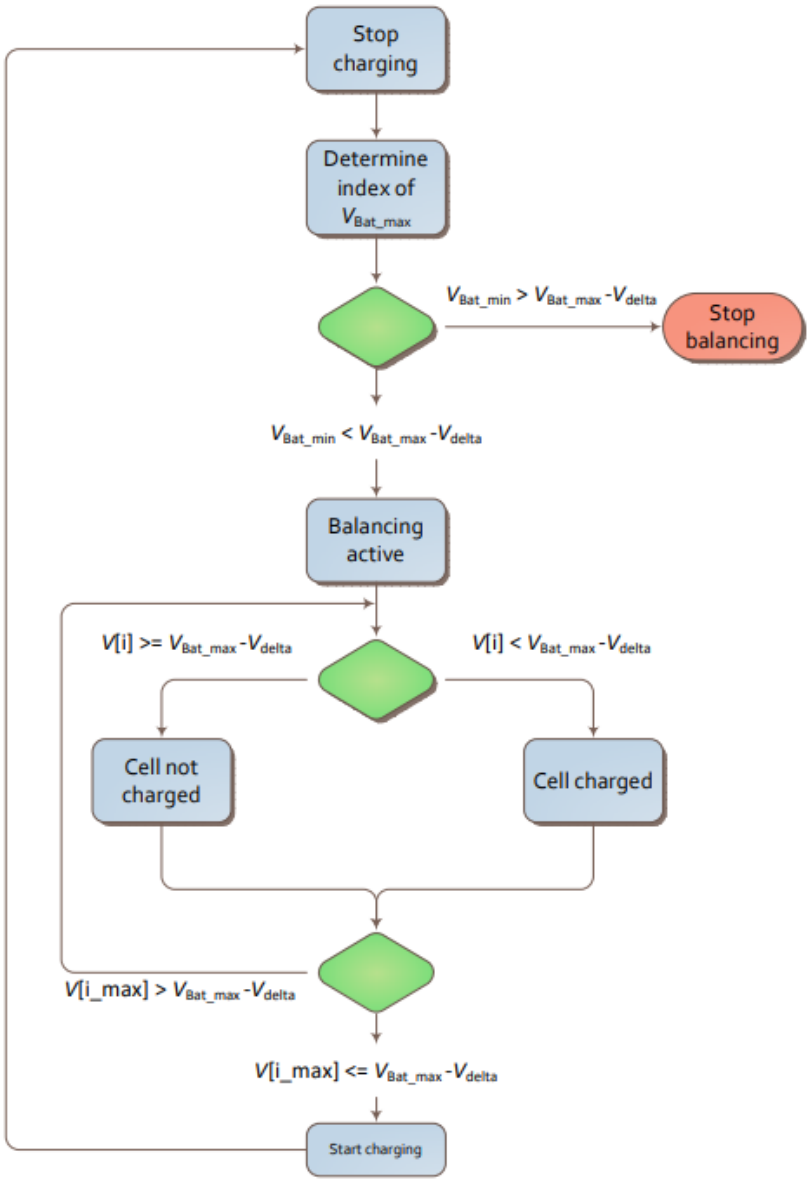

Рис. 2.1. Контрольоване балансування на основі напруги

Однак ця специфікація рідко виконується і справедлива лише для напруги розімкнутого ланцюга. Наприклад, елемент який має більший степінь старіння буде класифікований як і сусідній, хоча їхні внутрішні опори будуть відрізнятися. Через більший опір на постарілому елементні збільшиться падіння напруги. Процес балансування розпочнеться під час зарядки системи, для знаходження усіх елементів які балансуються необхідно визначити напругу на їхніх клемах, щоб знайти найбільш та найменш заряджений елемент.

Даний алгоритм є простим у реалізації оскільки він не вимагає попереднього створення таблиці OCV та не є прив’язаним до SoC. Алгоритм доцільно

використовувати коли необхідно отримати акумулятори однакової напруги але при цьому ємність кожної банки буде різна. Під час зарядки акумулятора з більшим опором більша кількість енергії буде розсіяна в тепло. І такий розбаланс може призводити до втрати більше 13% ємності АКБ [1].

Алгоритм, який базується на оцінці значення OCV передбачає, що заздалегідь, на випробувальному стенді отримано криві OCV та SoC для діапазону температур від -20 °C до +60 °C, які зберігаються в пам’яті мікроконтролера для визначення поточної енергії акумулятора за його напругою.

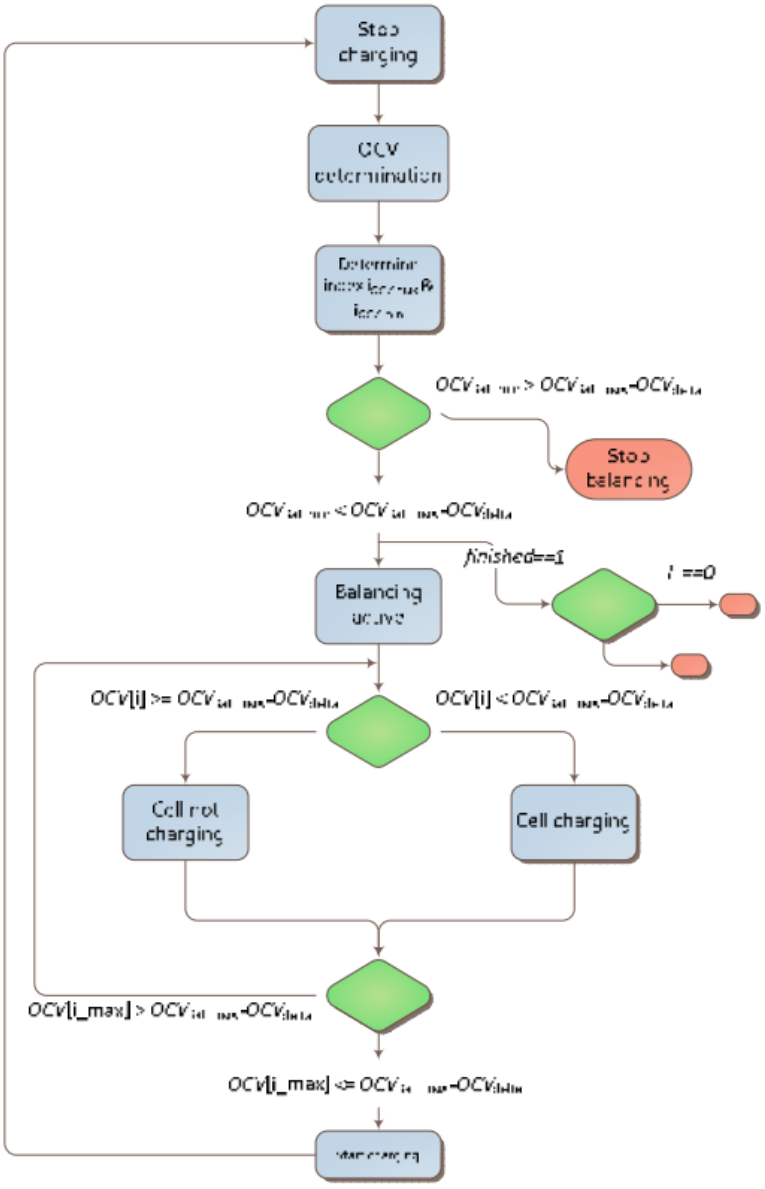

Рис. 2.2. Контрольоване балансування на основі OCV алгоритм

Для оцінки значення OCV протягом експлуатації прогнозується внутрішній опір і його падіння напруги буде за формулою 2.1.

$$OCV_{bat} = V_{bat} - I_{bat}R_{bat} \quad (2.1)$$

Таким чином напругу, кожного окремого елемента можна збалансувати. Даний алгоритм є недосконалим, оскільки у зв'язку із старінням АКБ ємність кожної комірки змінюється по різному, що призведе до втрат енергії та прискорення старіння батарей. Розбаланс для такого алгоритму на початку експлуатації може бути не значним, але з часом він досягає 8% [1].

Алгоритм, який базується на оцінці значення SoC — це технічно найскладніший алгоритм балансування. Він базується на інформації про історію SoC кожної комірки та обчисленні часу, який необхідний для балансування кожної комірки на основі методики [1]. Даний алгоритм представлений на рисунку 2.3.

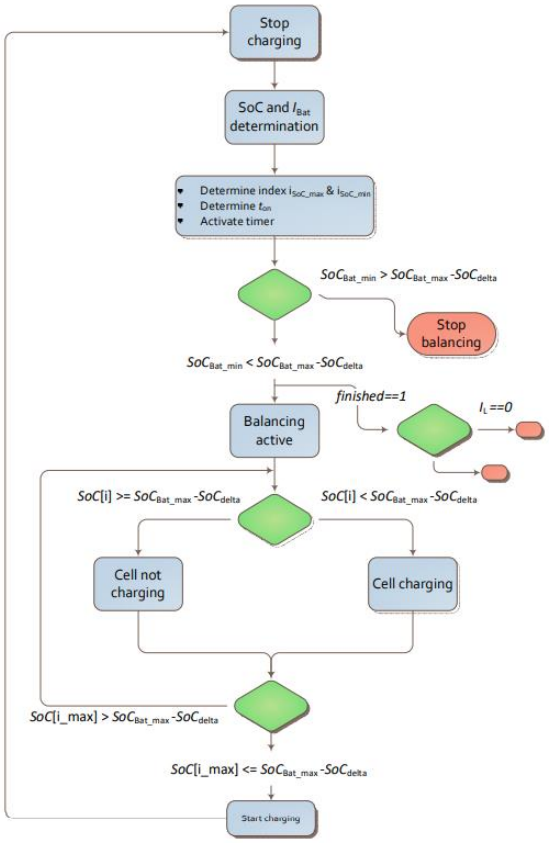

Рис. 2.3. Контрольоване балансування на основі SoC алгоритм

Визначення надлишку енергії комірки виконується за формулою 2.2:

$$SoC_{dit,k} = \frac{t_{on,k} + \frac{V_{Bat,t}}{R_{i,k}} * 100\%}{C_k * 3600\,{}^{S}/_{h}} \quad (2.2)$$

Тому час, протягом якого має тривати балансування, можна описати формулою 2.3 як:

$$SoC_{dit,k} = \frac{\left(SoC_{t_0,k} - SoC_{min,k}\right) * 36\,{}^{S}/_{h} * C}{\frac{V_{Bat,t}}{C_k * R_{i,k}} + \frac{I_{line}}{C_{k_{min,k}}} - \frac{I_{line}}{C_k}} \quad (2.3)$$

Оскільки вимірювання струму і напруги не є ідеальними це може призвести до не точності розрахунку, тому необхідно періодично коригувати час балансування кожного елементу та перезаписувати максимальну ємність акумулятора в кінці заряду . В результаті застосування цього алгоритму розбаланс комірок АКБ можна звести до 1.6%, однак він є складним в реалізації [11].

Оптимального балансування АКБ, що можна досягнути об’єднавши методи балансування за напругою та OCV. Цей модифікований алгоритм при старті системи визначає внутрішній опір кожної комірки. Якщо різниця внутрішніх опорів при однаковій напрузі більша допустимого значення, робота такої системи є неможливою бо комірка є надто деградованою і АКБ може не балансуватись, в іншому випадку система визначає внутрішню енергію комірок за допомогою таблиць OCV та розпочинає роботу. Старт балансування комірок відбувається, із початком зарядки АКБ. Балансуватимуться невеликими струмами усі комірки, які мають різницю напруг більшу за задане значення ΔV. Балансування АКБ буде складати 75-100% часу зарядки.

2.2 Алгоритм визначення внутрішнього опору Li-ion акумулятора

Для вимірювання або оцінки внутрішнього опору (Ri) Li-on елемента були запропоновані різні методи. У цій роботі Ri береться як еквівалентний резистор, розміщений у класичній електричній моделі напруга-джерело-резистор (рис. 2.4), який вироблятиме падіння напруги для заданого струму.

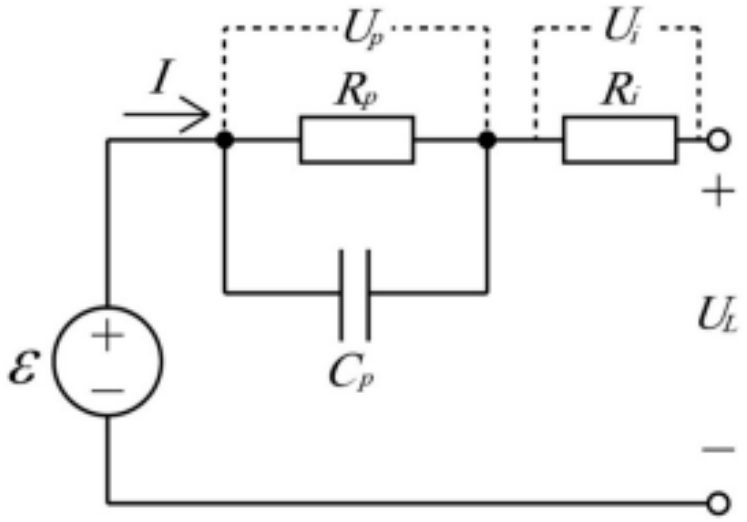

Рис. 2.4. модель еквівалентної схеми батареї

Цей Ri також можна розділити на два окремих внески: омічний опір (який є постійним на кожній частоті (позначається RΩ) і опір поляризації, який змінюється зі зміною потоку струму (позначається Zp). Цей опір поляризації створюється за допомогою послідовність RC-ланцюгів, позначених Rp і Cp.

Кожен RC-ланцюг матиме свою постійну часу. Деякі з них матимуть швидку постійну часу відгуку з $\tau \approx 1$ мс, а деякі матимуть повільну постійну часу відгуку з $\tau \approx 1$ с.

При реалізації алгоритму вимірювання опору на контролері реального часу Rin оцінюється кожні пару секунд через велику кількість осередків. Таким чином, частина Cp швидкої реакції завжди буде заряджена, що робить пов'язаний з ним Rp ефективним для падіння напруги. RΩ буде представленням (RΩ + Rpol) для частин

швидкої реакції Zp, а Rp буде представленням лише частинами із повільною відповіддю.

Для отримання Ri за кроткий час згідно з одним із методів використовуються формула 2.4.

$$R_{in} \approx \frac{\Delta V}{\Delta I} \tag{2.4}$$

де ΔV — падіння напруги,

ΔI — падіння струму,

Ri — опір батареї.

Те саме рівняння застосовується для позитивного імпульсу (тобто в режимі заряду). Цю оцінку можна виконувати багато разів протягом імпульсу струму, що дозволяє відокремити RΩ від Rp, як показано на рисунку 2.5.

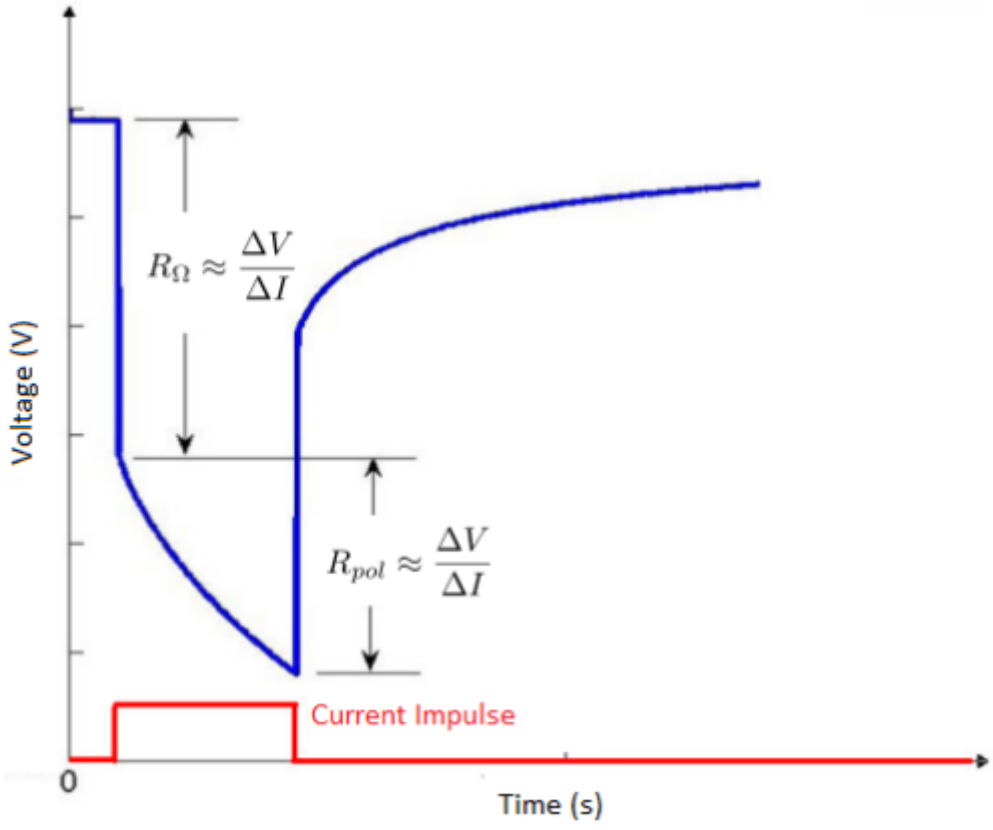

Рис. 2.5. Оцінка Ri, RΩ та Rp

Відомо, що на Ri впливає на температура елемента, величина струму, SoC та старіння. Програмне забезпечення BMS має реалізувати метод диференціації варіації Ri на основі кожного з цих факторів.

Чим більше оцінок Ri може отримати BMS, тим краще вона може встановлювати співвідношення зі струмом, SoC та температурою. Хоча BMS не контролює поточні вимоги, вона повинна постійно оцінювати зміни. Для BMS, використовується поріг ±200 мА (як заряд, так і розряд), що показало хороші результати. Якщо BMS змогла отримати оновленні дані по Ri для кожної комірки з частотою 100 мГц це вважається хорошою вибіркою для Ri вона є важлива для застосування електромобілях, оскільки вона визначає доступну потужність (P), виділення тепла (втрати L) та ефективність елемента (Eff), як описано в. Якщо Ri не правильно оцінено, це може призвести від некоректного обчислення до перегрів та загоряння батареї [12]. Даний алгоритм відповідає стандарту визначення внутрішнього опору літій-іонних IEC 61951-1:2013 “Secondary cells and batteries containing alkaline or other non-acid electrolytes - Portable sealed rechargeable single cells - Part 1: Nickel-cadmium” [26].

2.3 Опис алгоритм створення таблиці OCV

Точне моделювання OCV має велике значення для управління літій-іонними акумуляторами. Щоб отримати значення OCV відповідного SoC між двома сусідніми точками вимірювання, OCV описується як функція(2.5) SoC. Найпоширенішою функцією, яка використовується для опису цього зв’язку, є поліном степеневої функції.

$$U_{ocv} = k_n SOC^n + k_{n-1} SOC^{n-1} + \cdots + k_1 SOC + k_0 \quad (2.5)$$

де ($k0$, $k1$, … kn) — параметри, які необхідно оцінити.

Для досягнення більш точної та надійної оцінки SoC батареї пропонуються різні методи, найбільш використовуваним є метод на основі підрахунку Кулонів.

Алгоритми на основі підрахунку Кулонів часто використовуються як основна технологія для оцінки SoC батареї в BMS. Вони виражають SoC як відношення наявної потужності до номінальної. Ємності в батареї можна розрахувати шляхом вимірювання кількості енергії втраченої під час розрядження та інтегрування її за час Δr.

Загальне рівняння для розрахунку SoC:

$$soc = SOC_0 + \frac{\int_{t_0}^{t_0+\tau} I_{bat}\Delta\tau}{Q_{rated}} * 100 \tag{2.6}$$

де SOC_0 — представляє початковий SoC;

I_{bat} — представляє струм який протікає через акумулятор;

Q_{rated} — номінальна ємність батареї.[13]

Регулярно необхідно точно оцінювати SoC0 для цього зазвичай використовується модель OCV-SoC. З цієї причини модель OCV-SoC широко використовується у багатьох роботах для визначення характеристик та моніторингу батареї. Функція відображення OCV-SoC може бути реалізована як таблиці пошуку, або як аналітичний вираз [13].

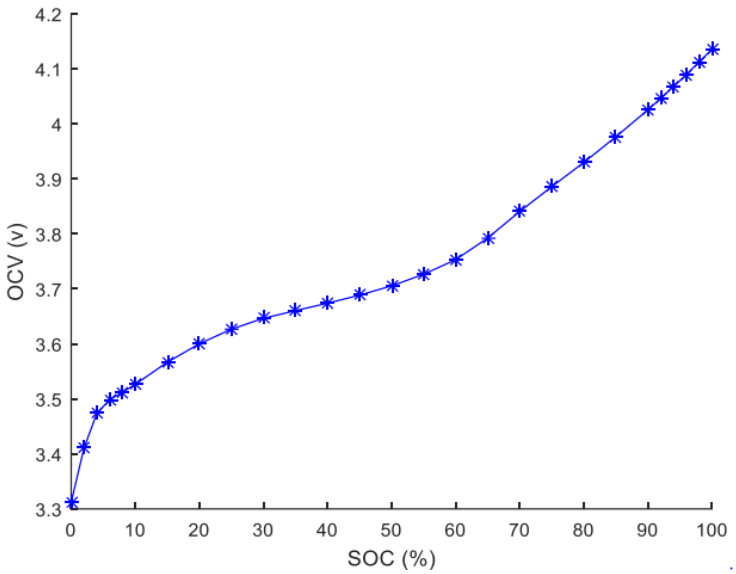

Рис. 2.6. Крива OCV-SoC для літій-іонних акумуляторів.

Останнє рішення забезпечує кращу обчислювальну ефективність і може бути узагальнено для всіх типів літій-іонних батарей, на відміну від таблиць пошуку, які є специфічними для кожної батареї і все ще занадто важкі для реалізації. Аналітичний вираз кривої OCV-SoC в літературі визначено по-різному. На рисунку 2.6 показано типова крива літій-іонного OCV-SoC. модель.

У різних роботах намагалися дати відповідне рівняння цій кривій [21, 22]. Насправді, повідомлялося, що співвідношення OCV-SoC можна апроксимувати до лінійного відрізка, який не корелює з реальністю, а дає прийнятну точність і полегшує реалізацію. Було запропоновано багато інших форм підгонки OCV-SoC, які враховують нелінійну поведінку батареї та дають високу точність. Однак вони вважаються непридатний для апаратної реалізації. Щоб поєднати як точність, так і низьку складність, OCV-SoC крива була виражена як кусково-лінійна система, щоб мати надійну початкову оцінку SoC [19].

2.4 Вибір мікросхеми датчика струму

Для коректної роботи системи важливо забезпечити має точність вимірювання струму. Для вимірювання струму використовується мікросхема ADS131M04. ADS131M04 - це чотириканальний 24-бітовий дельта-сигма (ΔΣ), аналого-цифровий перетворювач (АЦП) з одночасною дискретизацією, цей пристрій відмінно підходить для вимірювання енергії. Входи АЦП можуть бути безпосередньо підключені до мережі резисторів або силового трансформатора для вимірювання напруги або до трансформатора струму, шунта або котушки Роговського для вимірювання струму. Окремі канали АЦП можна налаштувати незалежно від входу датчика. Підсилювач з програмованим підсиленням (PGA) та низьким рівнем шуму забезпечує коефіцієнт підсилення від 1 до 128 для сигналів низького рівня. Крім того, цей пристрій інтегрує калібрування фази канал-канал і регістри калібрування зміщення та посилення, щоб допомогти усунути помилки сигнального ланцюга. Опції циклічної перевірки надлишковості (CRC) можна окремо ввімкнути на введенні, виведенні даних, щоб забезпечити цілісність зв’язку. Повний аналоговий

інтерфейс (AFE) пропонується в 20-контактному корпусі TSSOP та специфікується в промисловому діапазоні температур від –40°C до +125°C [14].

Для ADS131M04 потрібні як аналогові, так і цифрові джерела живлення. Аналоговий блок живлення (AVDD – AGND) може працювати від 2,7V до 3,6V. Цифровий блок живлення (DVDD – DGND) підтримує як 1,8V, так і 3,3V. АЦП отримує опорну напругу від вбудованого опорного 1,2V.

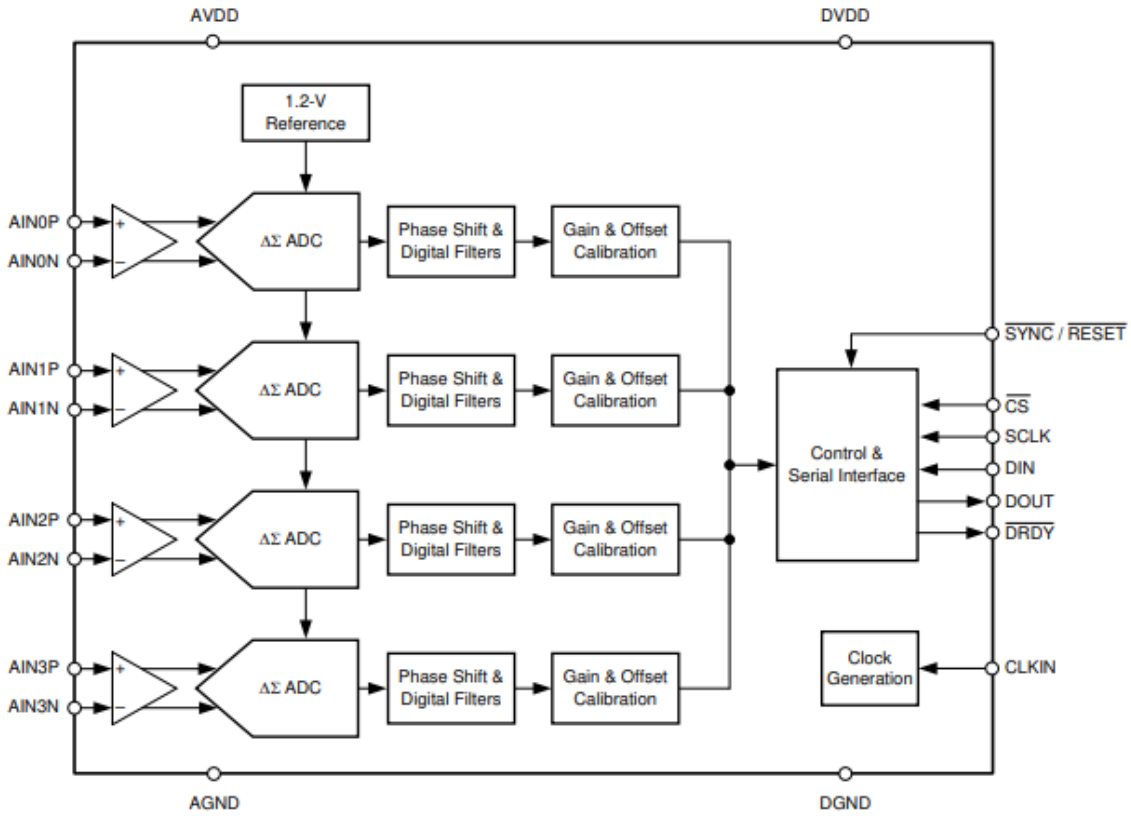

Рис. 2.7. Функціональна блок-схема ADS131M04

Як можна бачити з рисунку 2.7 кожен канал на ADS131M04 містить цифровий фільтр децимації, який демодулює вихід модуляторів ΔΣ. Фільтр забезпечує швидкість передачі даних до 32kSPS на канал у режимі високої роздільної здатності. Відносну фазу вибірок можна налаштувати між каналами, таким чином забезпечуючи точну компенсацію фазової характеристики датчика. Регістри калібрування зміщення та посилення можна запрограмувати для автоматичного налаштування вихідних вибірок на виміряні помилки зміщення та посилення. Дану мікросхему можна під’єднати до мікроконтролера за допомогою SPI шини.

На рисунку 2.8 показано, як можна підключити мікросхему ADS131B04-Q1 до мікроконтролера з використанням мінімальної кількості контактів інтерфейсу. Ця конфігурація корисна під час використання ізоляції даних для мінімізації кількості необхідних каналів ізоляції або коли контакти мікроконтролера (MCU) обмежені.

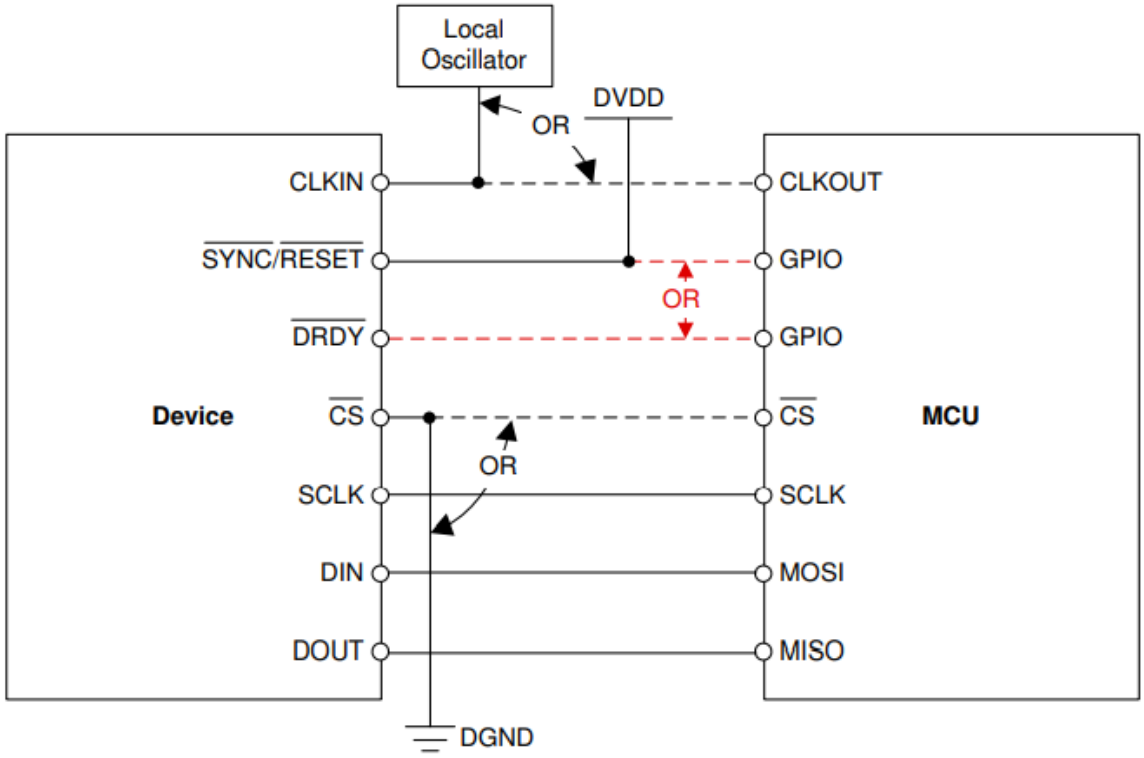

Рис. 2.8. Схема підключення ADS131M04-l до мікроконтролера

Для контакту CLKIN потрібен тактовий сигнал LVCMOS, який може бути згенерований MCU або створений за допомогою локального вихідного генератора LVCMOS, коли пристрій налаштовано для використання із зовнішнім годинником. В іншому випадку необхідно контакт CLKIN підключити до DGND, якщо використовується внутрішній генератор. Підключення SYNC/RESET до DVDD відбувається якщо він не використовується. Контакт DRDY можна залишити непідключеному стані, якщо не використовується. Підключення SYNC/RESET або DRDY до MCU відбувається що MCU залишався синхронізованим з перетвореннями АЦП. Якщо MCU забезпечує CLKIN, періоди CLKIN можна підрахувати для визначення періоду вибірки. Синхронізацію неможливо відновити, якщо на годиннику виникає помилка біта, і вибірки можуть бути пропущені, якщо

контакти SYNC/RESET або DRDY не використовуються. CS може мати низький рівень, якщо ADS131B04-Q1 є єдиним пристроєм на шині SPI. CRC введення та виведення використовується для захисту від помилкового читання та запису в регістрі, якщо CS знаходиться і постійно низькому рівні.

На рисунку 2.9 зображено типову схему використання мікросхеми ADS131M04 в системі керування акумуляторними батареями.

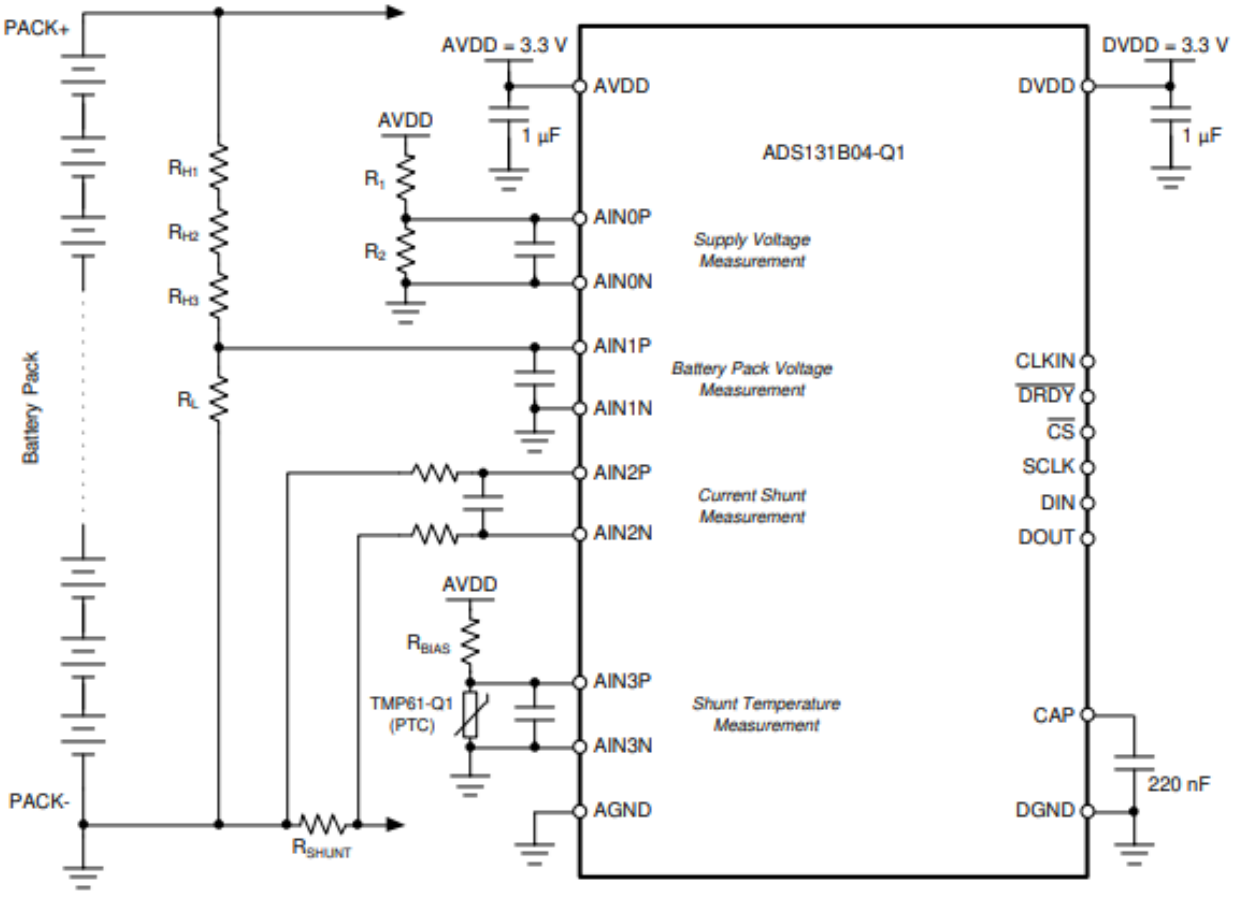

Рис. 2.9. ADS131B04-Q1 у типовій системі керування акумулятором

З рисунку 2.9 видно що мікросхема ADS131M04 виконує наступні основні функції:

– вимірювання струм акумулятора з високою роздільною здатністю та точністю за допомогою шунтуючого резистора малого опору;

– вимірювання пікових струмів і виявлення умов перевантаження струму або короткого замикання;

– вимірювання напругу акумуляторної батареї, використовуючи високовольтний резисторний дільник;

– вимірювання температуру шунта за допомогою терморезистора з лінійним позитивним температурним коефіцієнтом (PTC), TMP61-Q1.

У типових BMS струм протікає через шунтуючий резистор необхідно вимірювати в обох напрямках для зарядки та розрядки акумуляторної батареї. Ця можливість біполярного вимірювання напруги важлива, оскільки одна сторона шунта підключена до того ж потенціалу GND, що й контакт AGND ADS131B04-Q1, що означає що абсолютна напруга, яку повинен вимірювати пристрій, до 140 мВ. Щоб забезпечити швидке виявлення перевантаження струму протягом 1 мс, забезпечуючи високу точність і роздільну здатність, ADS131B04-Q1 працює зі швидкістю 4kSPS (OSR = 1024, режим високої роздільної здатності) з використанням режиму глобального переривання, в якому проводиться вимірювання з мінімальною похибкою зміщення за температурою та часом. Час перетворення з використанням цих налаштувань становить 0,754 мс. Роздільна здатність може бути додатково покращена шляхом усереднення результатів перетворення за більш тривалий період часу в мікроконтролері, який взаємодіє з ADS131B04-Q1.

2.5 Вибір мікросхеми BMB

Для повного контролю та передбачення роботи АКБ необхідно використовувати спеціалізовані мікросхеми для отримання даних з кожного послідовно включеного елементу акумулятора. Так як більшість систем які використовують в будинках є низьковольтними та їхня напруга складає 48V, для отримання такої напруги достатньо 12 послідовно з’єднаних акумуляторів типу 18650. Для контролю цих акумуляторів було вибрано мікросхему ltc6811 яка може вимірювати напругу та балансувати 12 послідовно включених акумуляторів із загальною похибкою вимірювання менше 1,2 mV. Діапазон вимірювання елемента

від 0V до 5V робить LTC6811. Усі 12 акумуляторів можна виміряти за 290 мкс, а для високого зниження шуму можна вибрати нижчу швидкість збору даних.

Декілька пристроїв LTC6811 можна під’єднати послідовно, що дозволяє одночасно контролювати елементи велику кількість акумуляторних батарей. Кожен LTC6811 має інтерфейс isoSPI для високошвидкісного, стійкого до радіочастотного зв’язку.

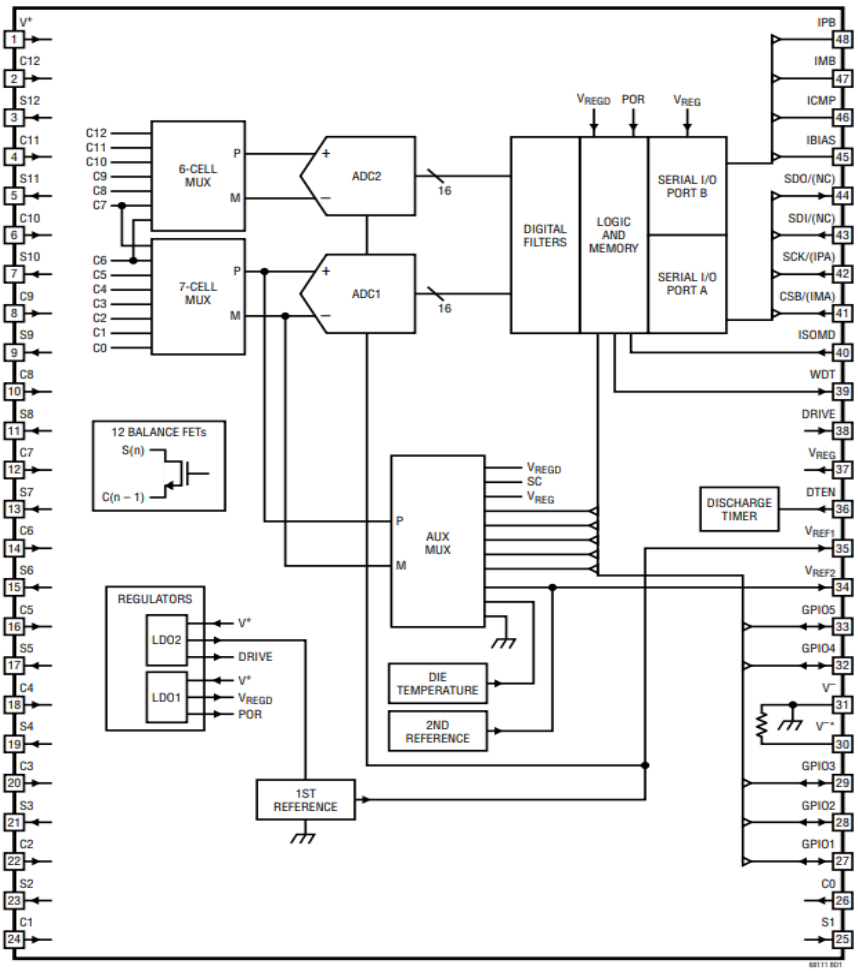

Рис. 2.10. Функціональна блок-схема LTC-6811-1

LTC6811 може живитися безпосередньо від акумуляторної батареї або від ізольованого джерела живлення. LTC6811 включає пасивне балансування для кожної елементу з індивідуальним керуванням робочим циклом ШІМ. Інші функції включають вбудований регулятор на 5V, п'ять ліній введення-виведення загального призначення і режим сну, в якому споживання струму зменшується до 4 мкА.

На рисунку 2.10 представлена функціональна схема LTC-6811-1 на якій:

– C0-C12: входи вимірювання напруги акумуляторів;

– S1–S12: балансні входи/виходи 12 внутрішніх N-MOSFET підключені між S(n) і C(n – 1) для розрядження елементів;

– V+: контакт живлення;

– V–:контакти землі;

– разом, зовнішні по відношенню до IC.

– VREF2: буферизована друга опорна напруга;

– VREF1: опорна напруга АЦП;

– GPIO[1:5]: введення/виведення загального призначення.

Також однією із важливих функції даної мікросхеми є балансування при його включенні на певну комірку вихід S може повільно розрядити цю комірку, підключивши її до резистора. Кожен вихід S підключений до внутрішнього N-канального МДН-транзистором з максимальним опором 25 Ом. Зовнішній резистор повинен бути підключений послідовно з цими МДН-транзистораом, щоб більша частина тепла могла розсіюватися за межами корпусу LTC6811.

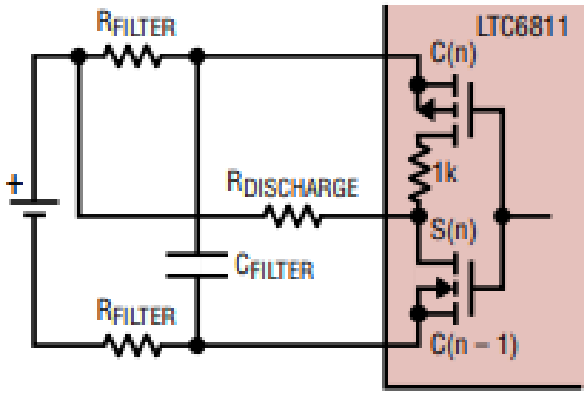

Рис. 2.11. Схема підключення пасивного балансування

Внутрішні перемикачі розряду МДН-транзистора можна використовувати для пасивного балансування елементів, як показано на рисунку 2.11, з балансуючим струмом 60 мА або менше. Струм балансування більше 60 мА не рекомендується

для внутрішніх перемикачів через надмірне нагрівання матриці. При розрядженні елементів із внутрішніми перемикачами слід стежити за температурою кристала.

2.6 Висновки до розділу 2

В даному розділі проведено аналіз існуючих алгоритмів та компонентів необхідних для розробки системи керування акумуляторними батареями. В результаті аналізу було виділено три основних алгоритми балансування, які базуються на: поточній напрузі комірки батареї, напрузі комірки при розімкненому колі Open-circuit voltage (OCV), і рівні заряду комірки State of charge (SoC). У зв’язку із недоліками кожного з існуючих алгоритмів було прийнято рішення про розробку власного алгоритму який поєднуватиме в собі функції перших двох існуючих, що в результаті повинно покращити ефективність балансування, в порівнянні з алгоритмами взятими за основу, та зменшити навантаження на мікроконтролер, в порівняні з алгоритмом SoC. Для забезпечення коректного балансування також було вибрано алгоритм визначення внутрішнього опору, який впливає на точність обчислення внутрішньої енергії батареї, та алгоритм створення таблиці OCV що впливає на визначення рівня старіння батареї.

Вказані вище алгоритми формують основні функції які має виконувати BMS та дали можливість вибрати компоненти для пристрою так як: LTC6811 (мікросхема контролю та балансування елементів акумуляторної батареї), ADS131B04-Q1 (мікросхема АЦП яка використовуватиметься для отримання даних струму, загальної напруг на вході/виході та температури плати).

Усі вище сказані технології забезпечують хорошу масштабованість та дозволяють використовувати дану систему для нових та вживаних батарей, в домашніх або промислових цілях.

РОЗДІЛ 3
РОЗРОБКА ТА ТЕСТУВАННЯ АПАРАТНО - ПРОГРАМНОЇ ЧАСТИНИ

3.1 Розробка плати BMS

На основі проведеного аналізу було розроблено плата контролю акумуляторних батарей повинна містити наступні компоненти: датчик температури, мікросхему контролю та балансування батареї, датчик струму, для зберігання конфігураційних параметрів енергонезалежну пам’ять, також важливим компонентом є драйвер попередньої зарядки BQ76200PW. В якості датчика температури використовується термістор NTCG163JF103FTDS, для балансування акумуляторної батареї вибрано мікросхему LTC6811. Для отримання даних про струм використовується мікросхема ADS131A04, яка є підключена до мікрорконтролера STM32F103C8 (рис 3.1), який опитує мікросхему та повертає опрацьовані дані про струм на головний контролер плати BMS за допомогою UART шини.

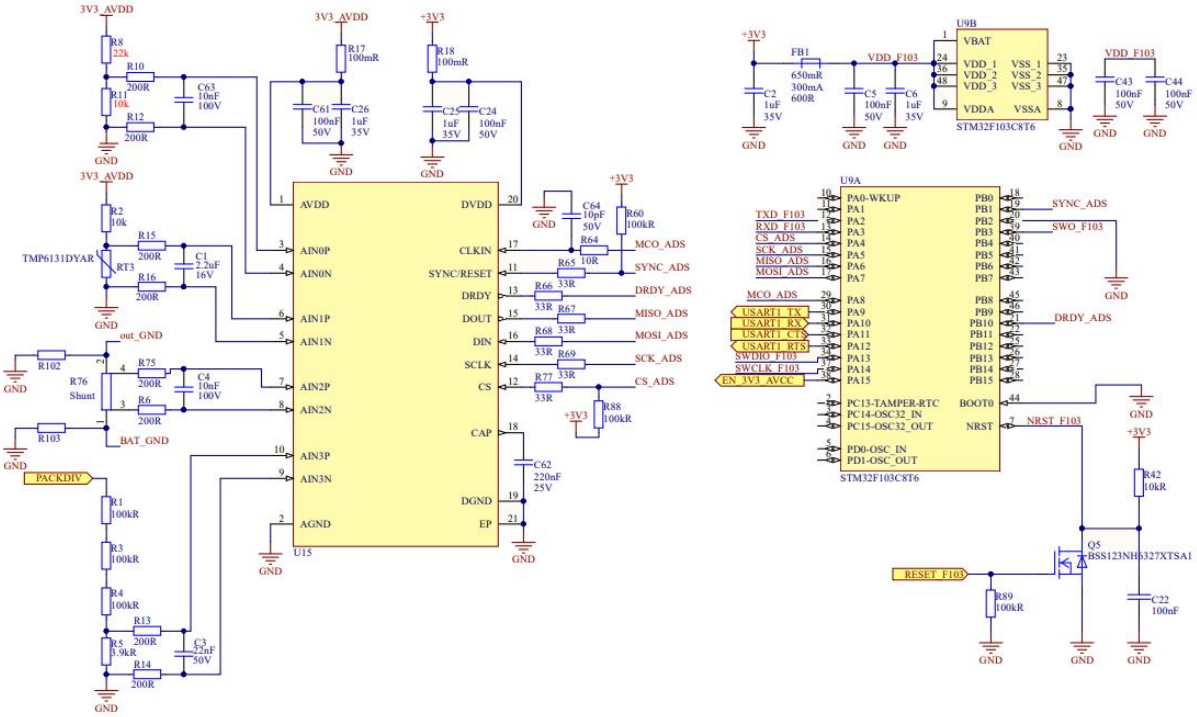

Рис. 3.1. Електрично-принципова схема датчика струму

Для зберігання конфігураційних використовується енергонезалежна пам'ять W25Q32. Головним контролером плати BMS виступає STM32F407VET6.

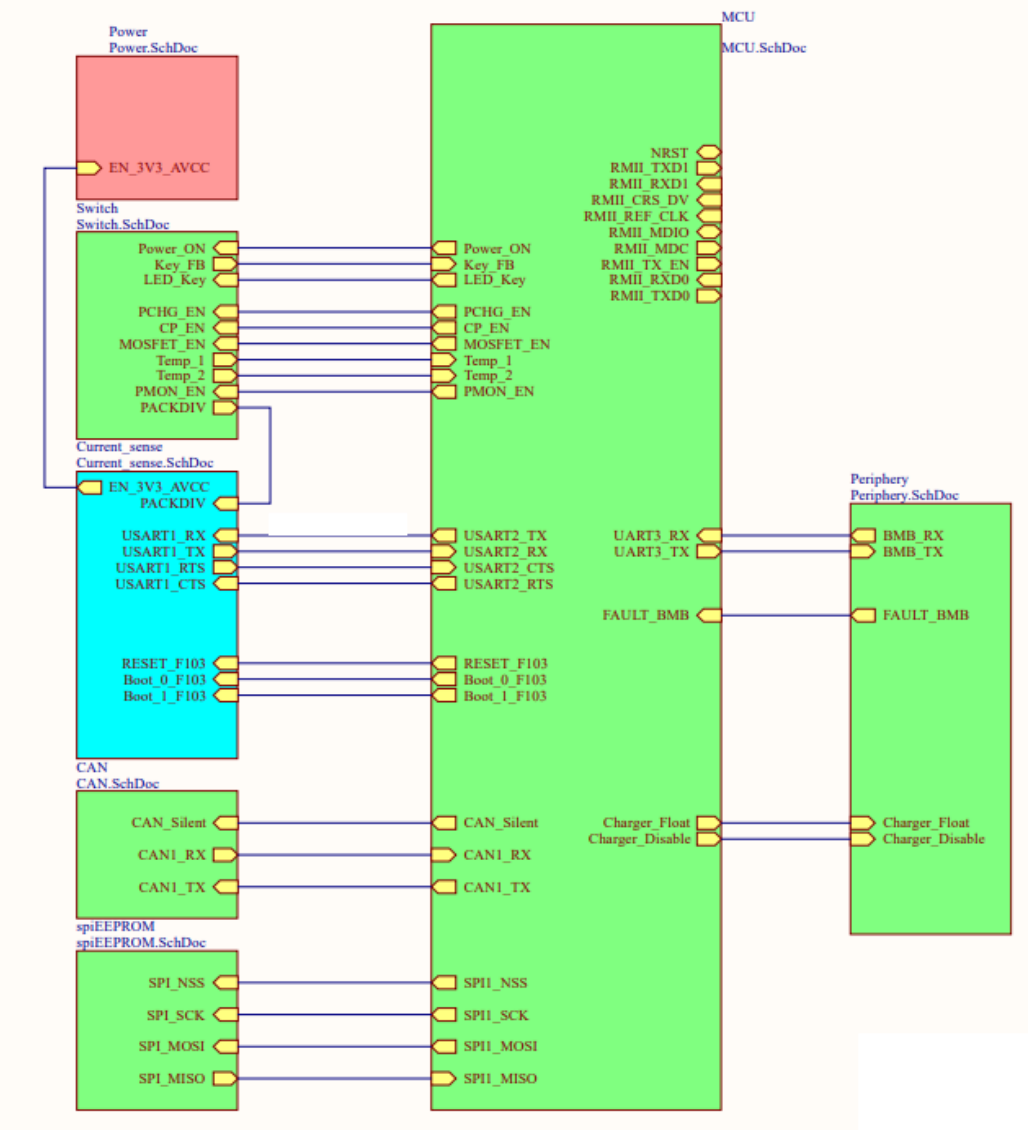

Рис 3.2. Структурна схема плати BMS

На рисунку 3.2 представлено схему підключень модулів BMS. Для підключення W25Q32 використовується SPI1, для підключення зовнішньої периферії використовується CAN шина, для керування мікросхемою попередньої зарядки використовуються піни PCHG_EN, CP_EN, MOSFET_EN, PMON_EN, для підключення датчика струму та BMB використовуються UART2 та UART3 відповідно.

3.2 Розробка алгоритму балансування

Зарядна станція розпочинає зарядку акумулятора, в свою чергу модуль балансування який вмонтований в батарею задає режим зарядки (звичайний режим або режим балансування). Модуль балансування - це сучасний BMS пристрій який має наступні функції: моніторингу за станом, захист від перезаряду та перерозряду елементу. Проаналізувавши попередні алгоритми було розроблено власний алгоритм який базується на алгоритмах основних на: поточній напрузі комірки батареї та напрузі комірки при розімкненому колі Open-circuit voltage (OCV), але на відміну від запропонованих алгоритмів, балансування буде відбуватися тільки у випадку якщо батареї заряджаються - так як пасивне балансування під час розрядки є недоцільним. Для точності вимірювань необхідне попереднє створення таблиць залежності ємності акумулятора від напруги (OCV) та внутрішніх опорів кожного елементу батареї.

Результуючий алгоритм включення системи контролю можна розбити на наступні етапи :

– старт системи а та перевірка підключеності усіх необхідних компонентів;

– визначення напруги на кожній банці та визначення температури батареї;

– визначення енергії акумулятора за таблицею OCV;

– визначення мінімальної та максимальної напруги на банках, визначення загальної напруги та температури батареї;

– відкриття та перевірка коректності роботи реле;

– визначення стану батареї: розрядка чи зарядка. В залежності від стану включення або виключення балансування;

– перевірка критичних станів або сигналів про виключення батарей.

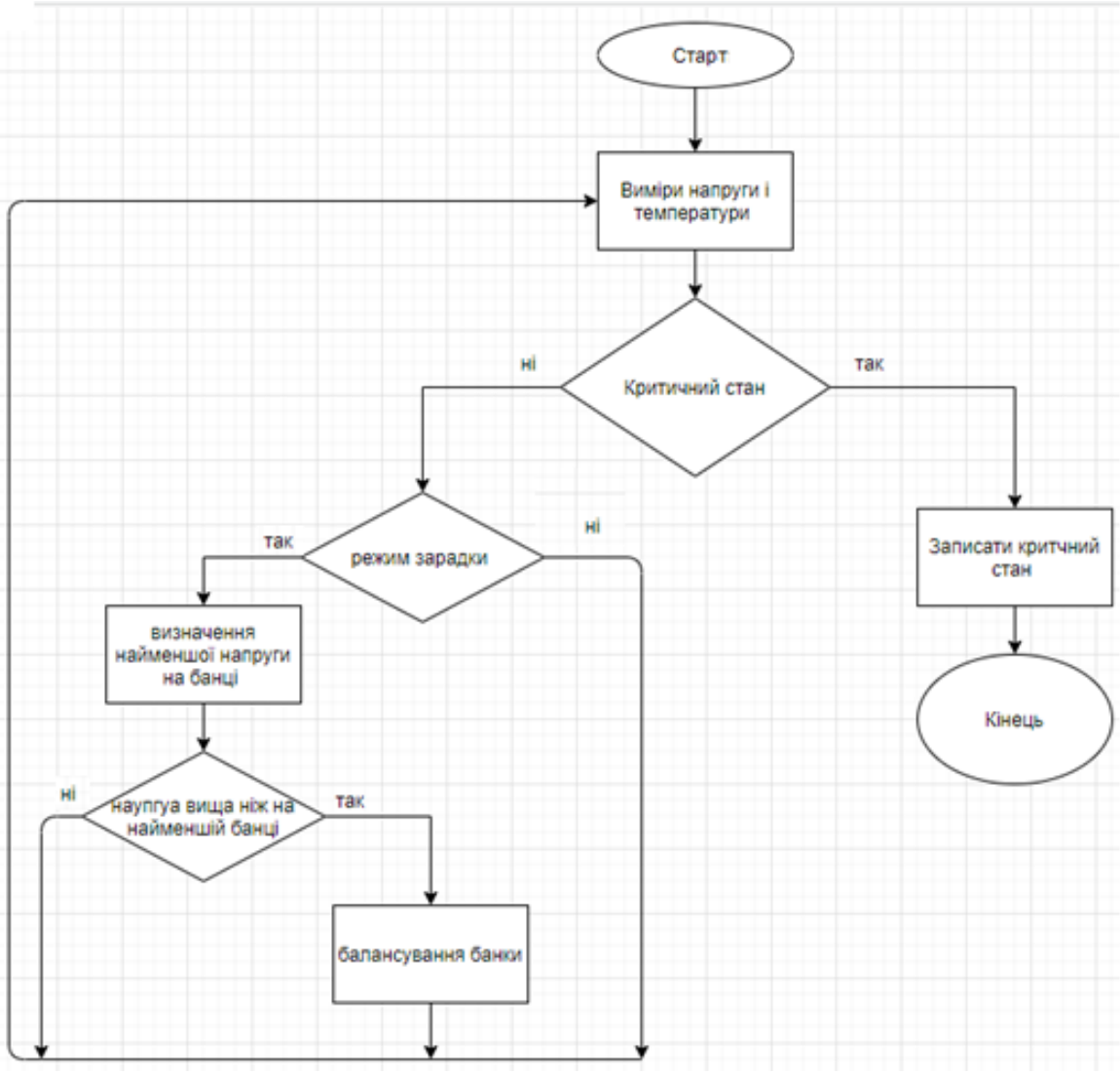

Рис. 3.3. Алгоритм роботи системи балансування

На рисунку 3.3 представлений алгоритм роботи системи балансування. Для коректності включення системи при її старті необхідно визначати внутрішній опір комірок батареї. Різниця внутрішніх опорів за приблизно однакової напруги має немає сильно відрізнятися якщо дана умова не справджується - робота системи з таким акумулятором неможливою тому що АКБ є надто деградованим та продовження роботи може призвести до завдання шкоди навколишньому середовищу. Якщо внутрішній опір є приблизно рівним система обчислює внутрішню енергію батареї за допомогою таблиць OCV після чого розпочинає роботу. Тільки у стані зарядки включається балансування акумуляторної батареї. Балансуються комірки, які мають напруг більшу ніж напруга на мінімальній комірці

на ΔV. Час балансування системи складає 75% часу зарядки та відбувається струмом 100mA [16].

```
379 void BMSModule::balanceCells(float VoltMin){
380     uint8_t payload[4];
381     uint8_t buff[30];
382     uint8_t balance = 0;
383     payload[0] = moduleAddress << 1;
384     payload[1] = REG_BAL_CTRL;
385     payload[2] = 0;
386     bmsUtil->sendData(payload, 3, true,buff, 30);
387     VoltMin = getMinVolt();
388     memset(balanceState,0,6);
389     for (int i = 0; i < 6; i++){
390         if ( (balanceState[i] == 0) && (getCellVoltage(i) > (VoltMin+0.02)) ) {
391             balanceState[i] = 1;}
392         else if(balanceState[i] == 1){
393             memset(balanceState,0,6);
394             balance = 0;
395             break;}
396         if (balanceState[i] == 1) balance |= (1<<i);
397     }
398     if (balance != 0){
399         payload[0] = moduleAddress << 1;
400         payload[1] = REG_BAL_TIME;
401         payload[2] = 0x0A;
402         bmsUtil->sendData(payload, 3, true,buff, 30);
403         payload[0] = moduleAddress << 1;
404         payload[1] = REG_BAL_CTRL;
405         payload[2] = balance;
406         bmsUtil->sendData(payload, 3, true,buff, 30);
407      }
408 }
```

Рис. 3.4. Функція балансування

На рисунку 3.4 представлена функція вибору елементів які необхідно балансувати та відправка команд на UART до мікросхеми для їхнього балансування. З даного коду помітно, що балансуватимуться усі елементи акумуляторної батареї які мають напругу більшу ніж мінімальна напруга +0,02V, які використовуються для згладжування похибок вимірювання та унеможливлення утворення коливального процесу балансування який призведе до саморозрядки акумуляторної батареї.

3.2 Тестування алгоритму визначення внутрішнього опору батареї

Розроблений алгоритм визначення внутрішнього опору записує дані про опір кожної батареї в eeprom. Для точного визначення внутрішнього опору батареї в будь-який момент часу створюється таблиця відповідності опору до напруги яка є на батареї.

Першим етапом алгоритму визначення внутрішнього опору акумуляторної батареї є створення усіх необхідних змінних таких як: масивів напруг з та без навантаженням, струм який протікатиме в системі, масив опорів, лічильник зчитувань напруги з елементів та об’єкт таймера. Приклад коду показаний на рисунку 3.5.

```
float volt_1[countBoard][countBattery] = {0};
float volt_2[countBoard][countBattery] = {0};
float current = 0;
float Resistionces[countBoard][countBattery] = {0};
float voltCurrent[2] ={0,0};
uint8_t countRead;
float time1;
Timer timeRead;
```

Рис. 3.5. Створення змінних для визначення опору

Наступним етапом є отримання поточних значень напруги кожного елементу акумуляторної батареї та запис отриманих даних у відповідні комірки першого масиву (рис. 3.6).

```
while(timeRead.read()<1){
  if(timeRead.read()>time1){
    time1 +=0.2;
    bms.getAllVoltTemp();
    countRead++;
    for(int i = 1; i<= countBoard;i++){
      for(int h =0;h<countBattery;h++){
          volt_1[i-1][h] += bms.getVoltBattery(i,h);
      }
    }
  }
}
```

Рис. 3.6. Отримання напруги без навантаження

Після чого необхідно відкрити силові транзистори на 10 секунд та подати навантаження 0.1С від ємності акумулятора. Після 9 секунд розрядки відбувається друге визначення напруги кожного елементу акумуляторної батареї та струму який протікає в системі (рис 3.7).

```
253   while(timeRead.read()<9){}
254   time1 = timeRead.read();
255   while(timeRead.read()<1){
256     if(timeRead.read()>time1){
257       time1 +=0.2;
258       countRead++;
259       CurrentS.readData(0x316);
260       current += CurrentS.getCurrent();
261       bms.getAllVoltTemp();
262       for(int i = 1; i<= countBoard;i++){
263           for(int h =0;h<countBattery;h++){
264               volt_2[i-1][h] += bms.getVoltBattery(i,h);
265           }
266       }
267     }
268   }
```

Рис. 3.7. Отримання напруги з навантаження

Після завершення навантаження необхідно закрити транзистори та усереднити отримані дані. За цими даними можна визначити внутрішній опір кожного елементу батареї та записати його на енергонезалежну пам'ять. Результат визначення внутрішнього опору для автомобільних акумуляторних батарей представлено на рисунку 3.8.

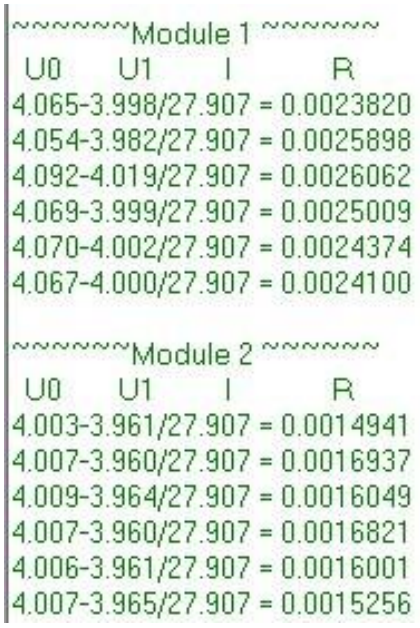

Рис. 3.8. Внутрішній опір елементів акумуляторної батареї

Як видно з рисунку 3.8 внутрішні опори в межах одного блоку є приблизно однаковими з чого можна зробити висновок, що даний блок є рівномірно деградованим, але опори в межах системи є кардинально різними що призводить до недоцільності використання даних блоків в одній системі.

Даний алгоритм повторюється декілька раз для одних і тих самих блоків, але з різною напругою.

3.3 Тестування алгоритму створення таблиць OCV

Алгоритм створення таблиці OCV використовується для визначення максимальної ємності одного елементу акумуляторної батареї, залишаючись в робочих межах. У зв’язку з тим що кожна батарея є обмежена ємністю найслабшого елемента - тому є недоцільно визначати ємність кожного елемента, а достатньо лише найслабшого в системі. Для цього було розроблено спеціальний алгоритм який в процесі розрядки визначає скільки енергії віддав кожен елемент та їхню напругу. Елемент з найменшою напругою записується в таблицю. Для того щоб дана таблиця не мала безкінечну кількість значень - вибірки відбуваються з кроком 0,01V під час падіння напруги на елементі (рис 3.9).

```
139     while(!bms.getIsCritVolt()){
140         if(times.read()>counterTime){
141             counterTime +=0.2;
142             CurrentS.readData(0x316);
143             current = CurrentS.getCurrent();
144             bms.getAllVoltTemp();
145             if(current < 0 ){
146                 if(current>-0.5){
147                     EBatInConfig +=((bms.getMinVoltCell()*current)/18000)
148                                 -(current * current * (bms.getResistanceOnMinVoltCell()))/18000;
149                 }
150                 else{
151                 ErrorData++;
152                 pc.printf("%f->%f->%f\n",current,bms.getMinVoltCell(),0.15);
153                 current = -0.25;
154                 EBatInConfig +=((bms.getMinVoltCell()*current)/18000)
155                                 -(current * current *  (bms.getResistanceOnMinVoltCell()))/18000;
156                 }
157             }
158             float UminIdeal = bms.getMinVoltIdeally(current);
159             if(bms.getIsCritVolt()){
160                 bq76200.power_off();
161             }
162             if(UminIdeal <= (UMin0 - deltVolt) || bms.getIsCritVolt()){
163                 UMin0 -= deltVolt;
164                 voltConfig.resize(voltConfig.size()+1);
165                 watConfig.resize(watConfig.size()+1);
166                 voltConfig[counter] =UminIdeal;
167                 watConfig[counter] = EBatInConfig*(-1);
168                 counter++;
169             }
170         }
171     }
```

Рис. 3.9. Основний цикл програми

OCV батареї має монотонне відношення до SoC. Це відношення, яке зазвичай моделюється як функція $V_0(s) = f(s)$ де $V_0(s)$ позначає OCV, та є в межах від 0 до 1. Для створення таблиць OCV необхідно провести розрядку акумуляторної батареї постійним струмом 1C від ємності в результаті отримані даній представляють співвідношення між кількістю енергій та напругою акумулятора. Дану таблицю можна використовувати для визначення SoC для акумуляторів які знаходяться в стані спокою, тобто на них не відбувається вплив зовні і вони не відновлюється після тривалого навантаження.

У рисунку 3.9 наведений графік таблиці OCV-SoC, представлений 115 значеннями напруг та відповідними їм енергіям, отриманих з кроком 0,01 вольта під час розрядки SoC. Для створення таблиць використовувалися акумуляторні батареї типу 18650 з внутрішнім опором 0,15Om.

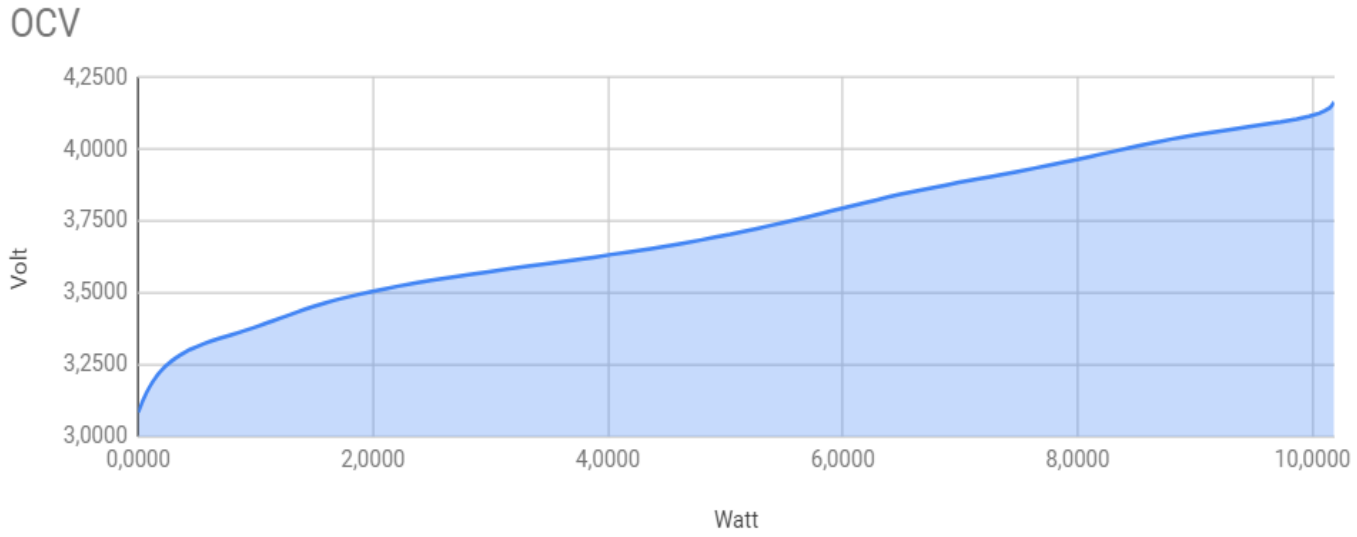

Рис. 3.10. Графік таблиці OCV

З рисунку 3.10 видно що ємність найслабшої банки системи приблизно дорівнює 10W що дозволяє припустити що енергія яку можна отримати з системи є кількістю банок помножена на енергію найслабшого елементу. Тому можна зробити висновок що система що складається з двох автомобільних акумуляторів має енергію приблизно рівну 8640W (8,6kW). Так як одна акумуляторна батарея складається з 432 акумуляторів типу 18650.

3.4 Тестування алгоритму визначення температури

Термістор є резистивним пристроєм, і тому відповідно до закону Ома, якщо пропустити через нього струм, на ньому виникне падіння напруги. Оскільки терморезистор є пасивним типом датчика, тобто для його роботи потрібен сигнал збудження, будь-які зміни його опору в результаті зміни температури можуть бути перетворені в зміну напруги [15].

Найпростіший спосіб зробити це – використовувати термістор як частину схеми подільника потенціалу, як показано н рисунку 3.11. Постійна напруга живлення прикладається до послідовного ланцюга резистора та термістора, а вихідна напруга вимірюється через термістор.

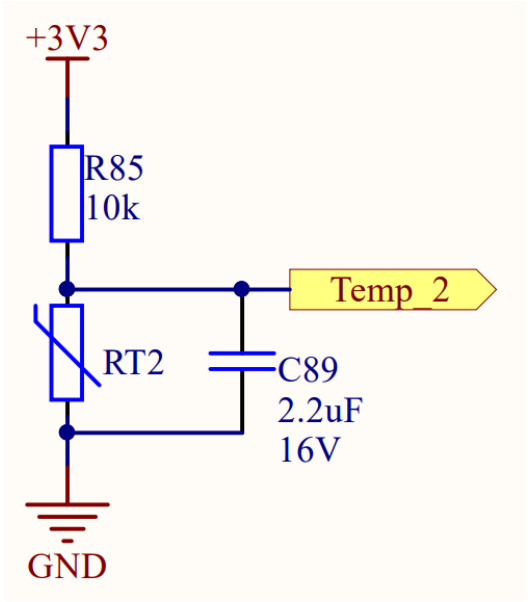

Рис. 3.11. Схема підключення термістора

Тут використовуємо термістор 10 кОм з послідовним резистором 10 кОм, то вихідна напруга при базовій температурі 25 С буде становити половину напруги живлення як 10 Ом/(10 Ом+10 Ом) = 0,5. Для визначення температури відповідно до напруги для термісторів NTC необхідно використовувати Рівняння Стейнхарта і Харта:

$$T = \frac{1}{A + B * \ln(R) + C(\ln(R))^3} \quad (3.1)$$

де A, B, C – коефіцієнти;

R – внутрішній опір термістора при температурі;

T – температура в Кельвінах (K).

Коефіцієнтом C нехтуємо оскільки точність 1/10 є достатньою для коректної роботи BMS. Коефіцієнтом B є вказаним в документації та термістор в даному випадку він складатиме 1/3435 для температурного діапазону від 25°C до 85°C. Коефіцієнтом A для даної схеми становить 0,003352329. Для отримання внутрішнього опору його необхідно виразити з формули подільника напруги. Формула для отримання R матиме наступний вигляд:

$$R_t = \frac{vTemp}{vRef - vTemp} * R_r \quad (3.1)$$

де R_t– опір термістора, Ohm;

$vRef$ – опорна напруга, V;

$vTemp$ – вихідна напруга, V;

R_r – опір резистора.

Для обчислення температури було написано код який переставлено на рисунку 3.12.

```
4  float Termo::getTempertureC(){
5      float A = 0.003352329;
6      float Rr = 10000;
7      float vRef =  _vRef->read();
8      float vTemp =  _vTemp->read();
9      float tempK_1 = (1.0/(A + (1.0/3435)*log((vTemp/(vRef-vTemp)*Rr))))
10     float tempC_1 = tempK_1 - 273.15;
11     return tempC_1;
12 }
```

Рис. 3.12. Функція обчислення температури

З рисунку 3.12 видно що функція getTempertureC повертає значення температури в °C.

3.5 Тестування алгоритму балансування

Для тестування алгоритму балансування було розроблено прототип системи в яка складається з:

– однієї плати BMS з датчиком струму;

– двох плат BMB;

– 12 акумуляторів типу 18650;

– резситорного навантаження;

– лабораторного навантаження.

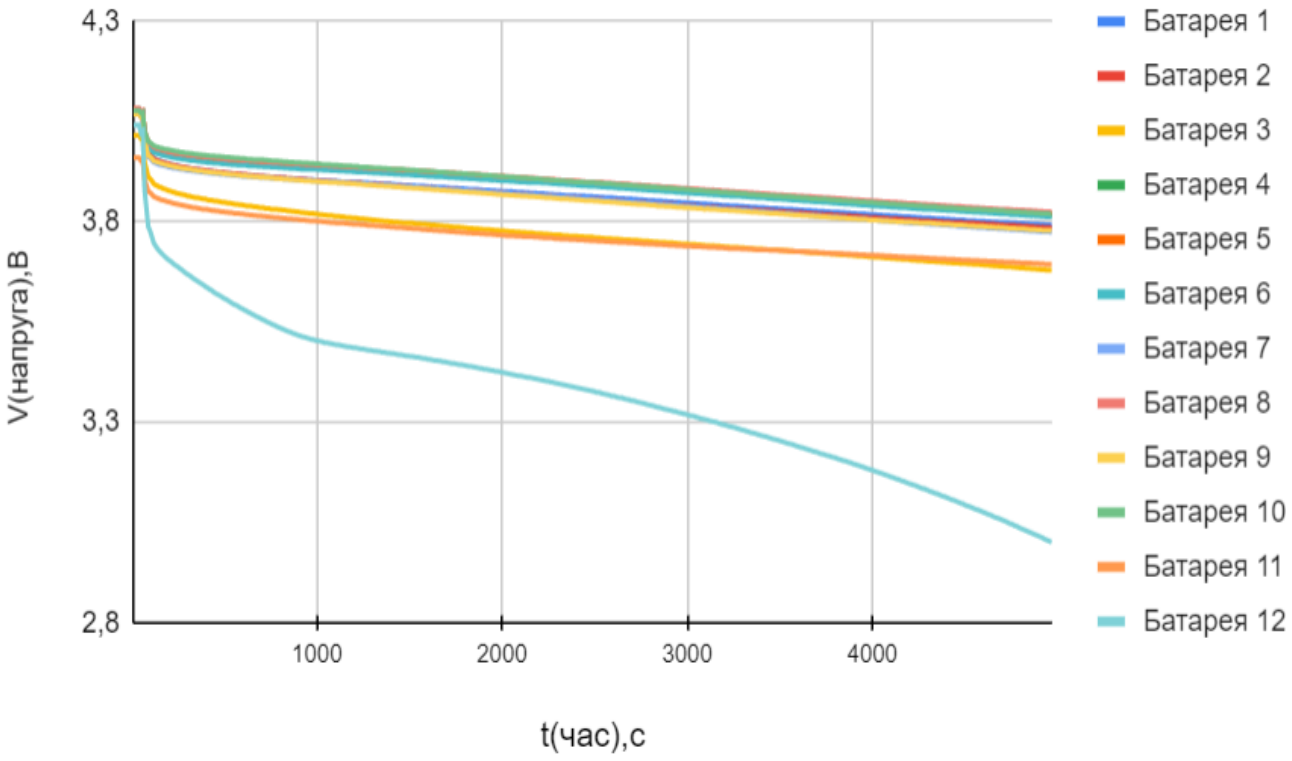

Рис. 3.13. Графік напруг акумуляторних батарей з різним внутрішнім опором під час розрядки

Для першого тестування було відібрано 12 вживаних акумуляторних батарей одного виробника з однаковою внутрішньою ємність відповідно до документації, але з різним внутрішнім опором. У зв’язку з чим акумуляторна батарея з найбільшим внутрішнім опором розрядилась найшвидше та нагрілась до найвищої температури в порівнянні з іншими акумуляторами на рисунку 3.12 дана батарея зображена під номером 12. У наслідок того що один із елементів досягнув критичного стану робота системи була призупинена та кінцевий споживач спожив з системи приблизно 1/4 усієї енергії що складає 30W. Використання даної системи в подальшому є недоцільно, тому що це призведе до швидкого старіння усіх інших елементів система або пожежі через перегрів одного з елементів.

Для наступного тестування було відібрано елементи з приблизно рівними внутрішніми опорами в результаті чого система пропрацювала під навантаженням значно довше до настання критичного стану, приблизно 22500с. При цьому з

системи було отримано 95% від загальної кількості всіє енергії, що приблизно рівне 114W(рис. 3.14).

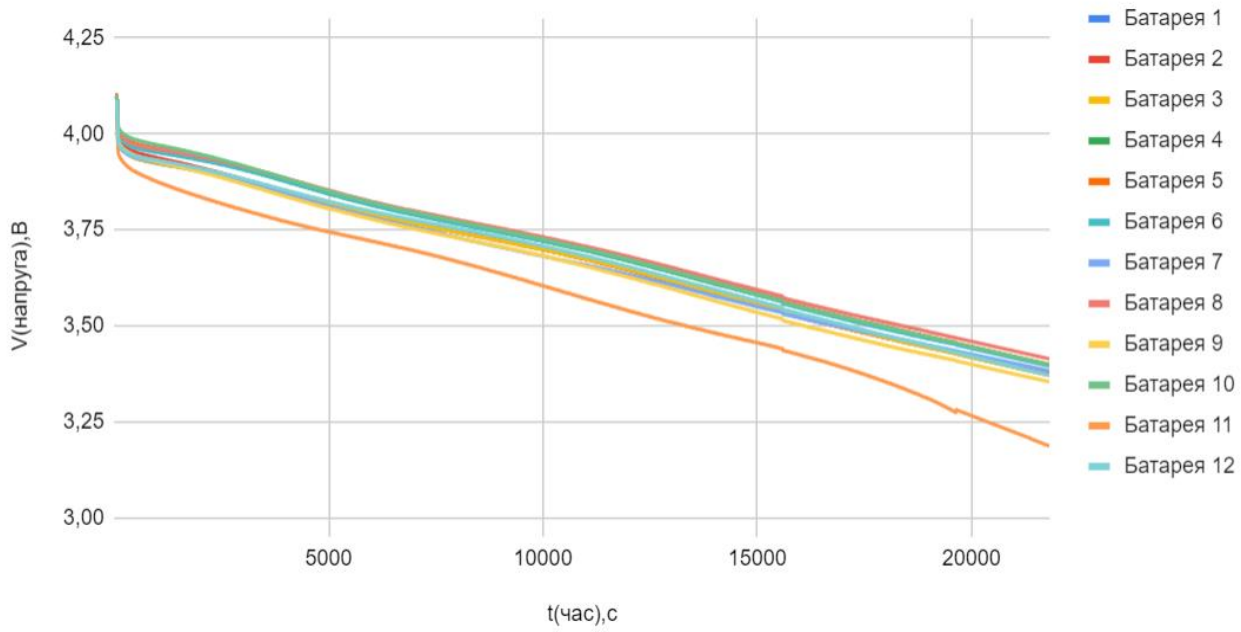

Рис. 3.14. Графік напруг акумуляторних батарей з приблизно рівним внутрішнім опором під час розрядки

Використання даної системи є доцільним в подальшому, тому систему з даних акумуляторами було заряджено.

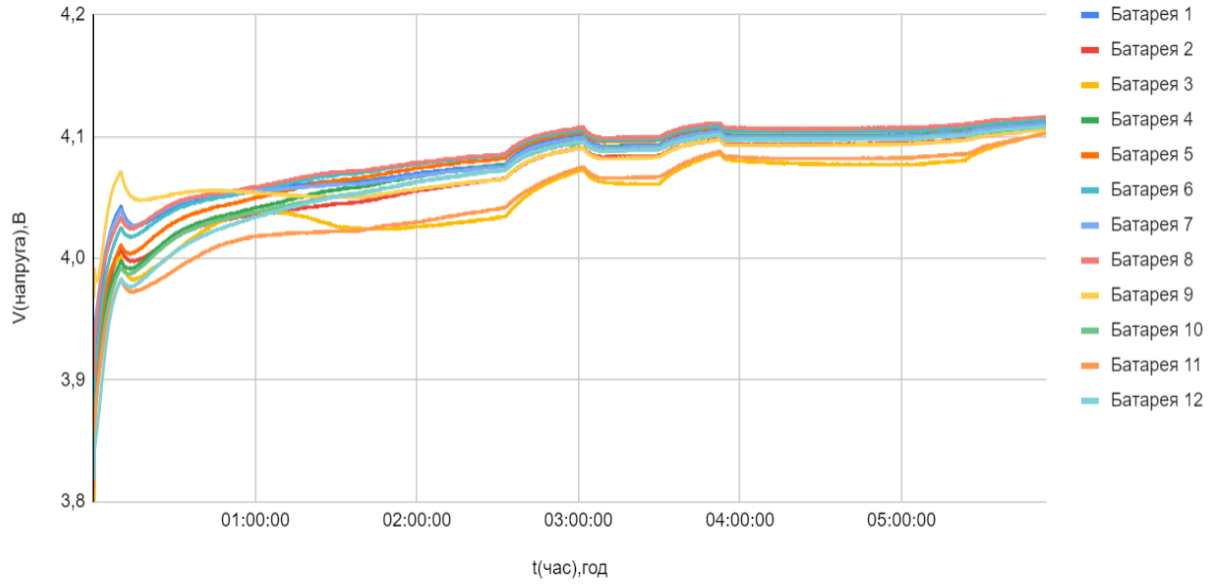

Рис. 3.15. Графік напруг акумуляторних батарей з приблизно рівним внутрішнім опором під час зарядки

На рисунку 3.15 представлено графік зарядки акумуляторної батареї з балансуванням з даного графіку можна побачити як відбувалось сповільнення зарядки елементів з більшою напругою.

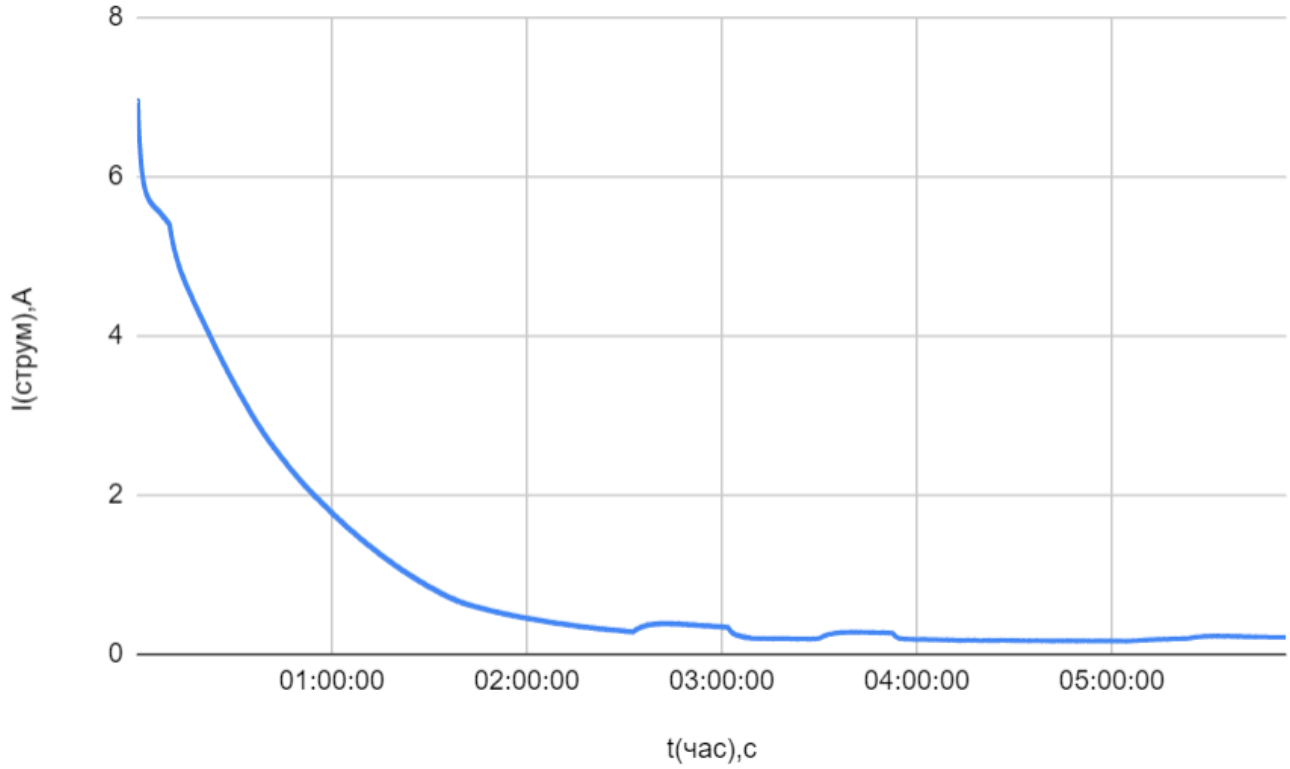

Рис. 3.16. Графік струму зарядки акумуляторних батарей з приблизно рівним внутрішнім опором

На рисунку 3.16 зображено зміну струму зарядки відносно часу, оскільки процес зарядки передбачає зменшення струму при зростанні напруги. В результаті даної зарядки акумуляторні батареї було збалансовано до напруги 4.13V та енергія яка знаходилась в системі була рівна 118W, що складає 96,6% від повної ємності АКБ відповідно до OCV. Запропонований алгоритм дає кращі результати проценту зарядки внутрішньої енергії АКБ відносно номінальної ємності в порівнянні з алгоритмом балансування на основі напруги комірки при розімкненому колі Open-circuit voltage (OCV) на 5,2%, та на 9,8% відносно алгоритм який базується на поточній напрузі комірки батареї.

3.6 Висновки до розділу 3

В даному розділі проведено розробку та тестування системи контролю та балансування літій іонних акумуляторних батарей.

Розробка передбачала:

- розробку алгоритму балансування;
- розробку плати прототипу та алгоритму її роботи;
- розробку датчика струму та алгоритму його роботи;
- розробку алгоритму визначення внутрішнього опору;
- розробку алгоритму створення таблиць OCV.

Розроблений алгоритм балансування дозволяє накопичення більшої кількості енергії при зарядці за допомогою вирівнювання напруг на послідовно включених елементах. У зв’язку з чим ефективність використання батареї як накопичувача енергії складатиме 96,6%.

Тестування передбачало перевірку:

- алгоритму визначення внутрішнього опору;
- алгоритму створення таблиць OCV;
- алгоритму балансування;
- комунікації з BMB;
- коректності включення виключення системи;
- роботи з енергонезалежною пам’яттю

РОЗДІЛ 4
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Охорона праці

. У роботі було розроблено програмно-апаратний комплекс системи контролю акумуляторних батарей. Оскільки основним компонентом даної системи є li-ion акумуляторні батареї, які при використанні вимагають дотримання вимог і норм охорони праці, а також правил техніки безпеки. Обов'язковою складовою при експлуатації є система захисту акумуляторних батарей.

На сьогодні основним нормативним документом, який визначає і регламентує норми та правила експлуатації систем безперебійного живлення є ДСТУ EN IEC 62040-1:2020 "Системи безперебійного живлення. Частина 1. Вимоги щодо безпеки (EN IEC 62040-1:2019, IDT; IEC 62040-1:2017, IDT)" [4]. У якому встановлені правила експлуатації таких систем.

Оскільки однією з основних складових даної системи є li-ion акумуляторні батареї - потрібно приділити особливу увагу дотриманню вимог відповідних стандартів для даних елементів живлення під час проєктування системи. Стандарти кінцевих пристроїв вимагають, щоб акумуляторні батареї відповідали вимогам стандарту UL 2054 "Household and Commercial Batteries"[23], який регламентує норми використання акумуляторних батарей в домашніх та комерційних умовах. Вимоги даного нормативного документа висуваються з метою зменшення ризику спалаху, вибуху, саморуйнування при використанні елементів всередині споживчих виробів.

Правила експлуатації та терміни служби акумуляторних батарей регламентовано експлуатаційними нормами середнього ресурсу акумуляторних свинцевих стартерних батарей колісних транспортних засобів і спеціальних машин, виконаних на колісних шасі [24]. Термін служби акумуляторних батарей передусім залежить від кількості циклів зарядки, розрядки акумуляторних батарей.

На основі стандартів EN–60095-1/93 “LEAD-ACID STARTER BATTERIES - PART 1: GENERAL REQUIREMENTS AND METHODS OF TEST”, ДСТУ EN IEC 62040-1:2020 та UL 2054 були виділені наступні правила, використання акумуляторів в системі контролю та балансування[22-25]:

- Не можна використовувати зарядні пристрої, які не призначені для цього типу АКБ.

- Процес зарядки повинен відбуватися за плюсових температур. Заряджати при від'ємній температурі можна лише літій-титанові акумулятори.

- Забороняється завдавати акумуляторам механічних пошкоджень таких як: проколювання, свердління, розбирання, розгерметизація, здавлювання, деформація та інших дій, здатних пошкодити захисну оболонку.

- Не можна допускати короткого замикання елементів, занурювати їх у рідкі середовища чи піддавати впливу вогню.

- Зберігати їх потрібно в сухому, прохолодному та недоступному для дітей місці.

- Рівень заряду елементів, що відправляються на зберігання, повинен становити 40-80%.

- Важливо не допускати перегріву та охолодження акумуляторів. Допустимий температурний діапазон для Li-ion елементів – від –20 до +60° С, а для моделей з типом хімії LiFePO4 – від –30 до +55 °С.

- Не можна залишати акумулятори під дією прямого сонячного проміння, біля опалювальних приладів та джерел відкритого вогню.

- Не можна залишати Li-ion акумулятори в приладах, що піддаються нагріванню.

- Потрібно виключити ймовірність перезаряду (для класичних літій-іонних АКБ – вище 4,2V на елемент) і не допускати глибокого розряду (нижче 3V на елемент).

- Акумулятори без захисної плати можна використовувати у ліхтарях та інших приладах, які мають власну систему захисту, або для створення АКБ із

загальною платою BMS. В інших випадках наявність захисної плати на акумуляторах є обов'язковою.

З метою безпеки краще не допускати небезпечних станів літієвих акумуляторних батарей. Якщо з певних причини склалася небезпечна ситуація, необхідно:

- Перед контактом із підозрілими акумуляторами – захистити руки рукавичками, тіло – комбінезоном, обличчя та органи дихання – маскою. Будь-які дії з такими елементами живлення можна робити тільки осторонь людей і легкозаймистих предметів, щоб мінімізувати можливі ризики.

- При пошкодженні герметувальної оболонки АКБ та витіканні електроліту – помістити батарею у відро чи іншу ємність та додати сухий пісок.

- При попаданні рідини, що виділяється з батарей, на шкіру або слизові - промити їх водою протягом 15 хвилин. У випадку якщо рідина потрапила у вічі – звернутися до офтальмолога.

- При загорянні літієвих акумуляторів - не використовувати воду і засоби гасіння, що містять її. При контакті літію з водою відбувається активне виділення водню, процес посилюється. Оптимальний варіант – користуватися сухим піском. По можливості локалізувати осередок спалаху.

- При активному газоутворенні та задимленні приміщення - залишити його.

- При попаданні диму в дихальні шляхи, що зазвичай супроводжується кашлем або іншими симптомами роздратування, звернутися до лікаря.

При розробці системи контролю акумуляторних батарей було проаналізовано та враховано вимоги охорони праці та правила техніки безпеки, що дозволило забезпечити безпечні умови використання системи користувачами.

4.2 Інженерний захист персоналу об’єкту та населення

Цивільний захист – це функція держави, спрямована на захист населення, територій, навколишнього природного середовища та майна від надзвичайних

ситуацій шляхом запобігання таким ситуаціям, ліквідації їх наслідків і надання допомоги постраждалим у мирний час та в особливий період.

Правовою основою цивільного захисту є Конституція України, цей Кодекс, інші закони України, а також акти Президента України та Кабінету Міністрів України.

Закон України «Про об'єкти підвищеної небезпеки» визначає правові, економічні, соціальні та організаційні основи діяльності, пов'язаної з об'єктами підвищеної небезпеки, і спрямований на захист життя і здоров'я людей та довкілля від шкідливого впливу аварій на цих об'єктах через запобігання їх виникненню, обмеження (локалізацію) розвитку і ліквідацію наслідків.

Під ідентифікацією об'єктів підвищеної небезпеки розуміють обов'язковий облік таких об'єктів, де використовують, виготовляють, переробляють чи 2 транспортують небезпечні речовини у кількості, що становить суттєву загрозу мешканцям прилеглих територій і навколишньому середовищу.

Декларація безпеки об'єкта підвищеної небезпеки – це документ, у якому викладено стратегію запобігання великим аваріям на такому об'єкті.

Під небезпечним виробничим об'єктом (НВО) розуміють об'єкт, на якому здійснюється технологічний процес, функціонально пов'язаний з використанням машин, механізмів, обладнання, що характеризуються підвищеним ступенем ризику завдання шкоди життю та здоров'ю людей. Термін небезпечний виробничий об'єкт, що застосовується у Законі України «Про промислову безпеку», має ширше значення, аніж термін об'єкт підвищеної небезпеки, визначення якого наведено в Законі України «Про об'єкти підвищеної небезпеки».

Закон України «Про пожежну безпеку» визначає, що забезпечення пожежної безпеки є складовою частиною виробничої та іншої діяльності посадових осіб, працівників підприємств, установ, організацій та підприємців. Це повинно бути відображено у трудових договорах (контрактах) і статутах підприємств, установ та організацій. Забезпечення пожежної безпеки підприємств, установ та організацій покладається на їх керівників, якщо інше не передбачено відповідним договором.

Оповіщення про загрозу або виникнення надзвичайних ситуацій. Оповіщення про загрозу або виникнення надзвичайних ситуацій полягає у своєчасному доведенні такої інформації до органів управління цивільного захисту, сил цивільного захисту, суб'єктів господарювання та населення.

Оповіщення про загрозу або виникнення надзвичайних ситуацій забезпечується шляхом:

– функціонування загальнодержавної, територіальних, місцевих автоматизованих систем централізованого оповіщення про загрозу або виникнення надзвичайних ситуацій, спеціальних, локальних та об'єктових систем оповіщення;

– централізованого використання телекомунікаційних мереж загального користування, у тому числі мобільного (рухомого) зв'язку, відомчих телекомунікаційних мереж і телекомунікаційних мереж суб'єктів господарювання в порядку, встановленому Кабінетом Міністрів України, а також мереж загальнонаціонального, регіонального та місцевого радіомовлення і телебачення та інших технічних засобів передавання (відображення) інформації;

– автоматизації процесу передачі сигналів і повідомлень про загрозу або виникнення надзвичайних ситуацій;

– функціонування на об'єктах підвищеної небезпеки автоматизованих систем раннього виявлення надзвичайних ситуацій та оповіщення;

– організаційно-технічної інтеграції різних систем централізованого оповіщення про загрозу або виникнення надзвичайних ситуацій та автоматизованих систем раннього виявлення надзвичайних ситуацій та оповіщення;

– функціонування в населених пунктах, а також місцях масового перебування людей сигнально-гучномовних пристроїв та електронних інформаційних табло для передачі інформації з питань цивільного захисту.

4.3 Висновки до розділу 4

В даному розділі описані вимоги до безпечного використання системи контролю та балансування літій-іонних акумуляторних батарей які базуються на

існуючих українських та міжнародних стандартах та вимогах АКБ. Основна увага приділена вимогам безпеки використання Li-ion акумуляторних батарей. Також в даному розділі описані основні вимоги до захисту персоналу та населення.

ВИСНОВКИ

В процесі дослідження методів і засобів реалізації системи контролю та балансування акумуляторних батарей було проаналізовано наявні системи, їхні алгоритми та розроблено власну систему.

В розділі 1 було розглянуто використання літій-іонних акумуляторів у техніці, їхні технічні характеристики та можливість повторного використання. Для вторинного використання застосовуються акумулятори електромобілів, які вже є непридатними для роботи EV, оскільки втратили до 30% своєї внутрішньої енергії. Також в даному розділі було розглянуто наявну систему для контролю балансування акумуляторних батарей вторинного використання. В процесі тестування було виявлено основний недолік даної системи — орієнтованість роботи на використання акумуляторів автомобіля Tesla Model S.

В розділі 2 проведено аналіз сучасних алгоритмів пасивного балансування послідовно включених акумуляторів типу 18650. Недоліками яких є низький ККД або складність в реалізації на мікроконтролері, відсутність перевірок для забезпечення безпеки акумулятора. Також проаналізовано та вибрано алгоритми визначення внутрішнього опору, складання таблиць OCV та визначення температури. Вибрано мікросхеми вимірювання температури, напруги, струму та мікроконтролер на базі якого буде працювати система.

В розділі 3 спроєктовано плату BMS та на основі проаналізованих алгоритмів балансування розроблено алгоритм пасивного балансування літій-іонних акумуляторних батарей з врахуванням їхніх недоліків. Також проведено тестування алгоритмів визначення внутрішнього опору, температури, створення таблиць OCV та балансування. В результаті тестувань було визначено ККД розробленого алгоритму, який складає 96,6%, що на 5,2% краще відносно алгоритму балансування на основі напруги комірки при розімкненому колі Open-circuit voltage (OCV), та на 9,8% краще відносно алгоритму, який базується на поточній напрузі комірки батареї.

В розділі 4 описано правила використання системи контролю акумуляторних батарей відповідно до норм міжнародного стандарту ДСТУ EN IEC 62040-1:2020 “Системи безперебійного живлення. Частина 1. Вимоги щодо безпеки (EN IEC 62040-1:2019, IDT; IEC 62040-1:2017, IDT)”.

В результаті спроєктовано та розроблено, відповідно до міжнародного стандарту IEC 61951-1:2017 “Secondary cells and batteries containing alkaline or other non-acid electrolytes - Portable sealed rechargeable single cells”, програмно-апаратний комплекс системи контролю та балансування літій-іонних акумуляторних батарей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Scott K., Nork S. Active battery cell balancing | analog devices. Mixed-signal and digital signal processing ICs | Analog Devices. URL:

2. Linear Technology. Farnell Global. URL: https://www.farnell.com/datasheets/1802780.pdf (date of access: 30.10.2021).

3. Identifying and Overcoming Critical Barriers to Widespread Second Use of PEV Batteries / J. Neubauer et al. Office of Scientific and Technical Information (OSTI), 2015. URL: https://doi.org/10.2172/1171780 (date of access: 30.10.2021).

4. Про затвердження Правил експлуатування акумуляторних свинцевих стартерних батарей колісних транспортних засобів і спеціальних машин, виконаних на колісних шасі : Наказ М-ва трансп. та зв'язку України від 02.07.2008 р. № 795 : станом на 3 лип. 2018 р. URL: https://zakon.rada.gov.ua/laws/show/z0689-08#Text (дата звернення: 2.11.2021)

5. Palamar A., Karpinskyy M. Control of an Uninterruptible Power Supply in a DC Microgrid System. 10th International Symposium «Topical Problems in the Field of Electrical and Power Engineering» and «Doctoral School of Energy and Geotechnology II», Pärnu, Estonia. 2011. P. 80–84.

6. Haram M. H. S. M., Lee J. W., Ramasamy, G. Ngu, E. E., Thiagarajah S. P., Lee, Y. H. Feasibility of utilising second life EV batteries: Applications, lifespan, economics, environmental impact, assessment, and challenges. Alexandria Engineering Journal, 60(5), 2021, 4517-4536.

7. Hoffart F. Lithium Ion Battery Charger Allows Choice of Termination Method and Includes 100mA Adjustable Low Dropout Regulator. SIGNAL. 100(80), 60.

8. LEE A. Li-Ion Battery Protection Circuit Draws Only 4.5 µA. ELECTRONIC DESIGN. 1999. P. 2.

9. Fleischer C., Ostendorp B., Sauer D. U. Simulative comparison of balancing algorithms for active and passive cell balancing systems for lithium-ion batteries. In Proc. Adv. Automotive Battery Conf.. 2013, February. P 11.

10. Elbows of Internal Resistance Rise Curves in Li-Ion Cells / C. Strange et al. Energies. 2021. Vol. 14, no. 4. P. 1206. URL: https://doi.org/10.3390/en14041206 (date of access: 5.11.2021).

11. Tessier A., Dubois M., Trovão J. Real-Time Estimator Li-ion Cells Internal Resistance for Electric Vehicle Application. World Electric Vehicle Journal. 2016. Vol. 8, no. 2. P. 410–421. URL: https://doi.org/10.3390/wevj8020410 (date of access: 10.12.2021).

12. Elmahdi F., Ismail L., Noureddine M. Fitting the OCV-SOC relationship of a battery lithium-ion using genetic algorithm method. E3S Web of Conferences. 2021. Vol. 234. P. 00097. URL: https://doi.org/10.1051/e3sconf/202123400097 (date of access: 15.12.2021).

13. ADS131M04 4-Channel, Simultaneously-Sampling, 24-Bit, Delta-Sigma ADC. Texas Instruments. URL: https://www.ti.com/lit/ds/symlink/ads131m04.pdf?ts=1636616967814&ref_url=https%3A%2F%2Fwww.ti.com%2Fproduct%2FADS131M04 (date of access: 17.11.2021).

14. Orion BMS Operation Manual Rev 2.1. Orion Li-Ion Battery Management System | Affordable & Reliable EV Li-Ion BMS. URL: https://www.orionbms.com/manuals/pdf/operational_manual.pdf (date of access:10.11.2021).

15. Palamar A. Intelligent control and monitoring module for uninterruptible power supply system. II International Scientific and Practical Conference «Theoretical and Applied Aspects of Device Development on Microcontrollers and FPGAs» (MC&FPGA-2020), Kharkiv, Ukraine. 2020. P. 12-13.

16. Волоський В., Лешчишин Ю., Романишин Н. Комп’ютерна система контролю та балансування літійіонних акумуляторних батарей. Актуальні задачі сучасних технологій : X Міжнар. науково-практ. конф. молодих уч. та студентів, м. Тернопіль, 25–26 листоп. 2021 р. Тернопіль, 2021. С. 87–88.

17. Steinhart J. S., Hart S. R. Calibration curves for thermistors. Deep Sea Research and Oceanographic Abstracts. 1968. Vol. 15, no. 4. P. 497–503. URL: https://doi.org/10.1016/0011-7471(68)90057-0 (date of access: 19.12.2021).

18. Chang W.-Y. The State of Charge Estimating Methods for Battery: A Review. ISRN Applied Mathematics. 2013. Vol. 2013. P. 1–7. URL: https://doi.org/10.1155/2013/953792 (date of access: 20.11.2021).

19. Estimation of lithium-ion battery state-of-charge using an extended kalman filter / M. Lagraoui et al. Bulletin of Electrical Engineering and Informatics. 2021. Vol. 10, no. 4. P. 1759–1768. URL: https://doi.org/10.11591/eei.v10i4.3082 (date of access: 21.11.2021).

20. Baccouche I., Jemmali S., Mlayah A. Manai, B. Amara, N. E. B. Implementation of an improved Coulomb-counting algorithm based on a piecewise SOC-OCV relationship for SOC estimation of li-IonBattery. 2018. P. 178-187

21. Волоський В., Лешчишин Ю., РОманишин Н. АЛГОРИТМ БАЛАНСУВАННЯ LI-ION АКУМУЛЯТОРНИХ БАТАРЕЙ НА ОСНОВІ ПОТОЧНОЇ НАПРУГИ ТА НАПРУГИ ПРИ РОЗІМКНЕНОМУ КОЛІ. ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ : IX НАУКОВО-ТЕХН. КОНФ., м. Тернопіль, 8–9 груд. 2021 р. Тернопіль, 2021. С. 106.

22. ДСТУ EN IEC 62040-1:2020. Системи безперебійного живлення. Чинний від 2021-06-01. Вид. офіц. Київ : ДП «УкрНДНЦ», 2020. 150 с.

23. UL 2054. Household and Commercial Batteries. Official edition. Chicago, 1997. 36 p.

24. IEC 61951-1:2017. Secondary cells and batteries containing alkaline or other non-acid electrolytes - Secondary sealed cells and batteries for portable applications - Part 1: Nickel. Effective from 2017-03-07. Official edition. 2017. 81 p

25. Паламар А. М., Паламар М. О. Метод підвищення надійності компонентів модульної комп’ютеризованої системи безперебійного живлення. Матеріали міжнародної наукової конференції «Іван Пулюй: життя в ім’я науки та України» (до 175-ліття від дня народження), Тернопіль. 2020. С. 91-92.

ДОДАТОК А

Тези конференцій

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний технічний університет імені Івана Пулюя (Україна)
Університет імені П'єра і Марії Кюрі (Франція)
Маріборський університет (Словенія)
Технічний університет у Кошице (Словаччина)
Вільнюський технічний університет ім. Гедимінаса (Литва)
Білоруський національний технічний університет (Республіка Білорусь)
Міжнародний університет цивільної авіації (Марокко)
Наукове товариство ім. Т.Шевченка

АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ

Збірник

тез доповідей

Том І

X Міжнародної науково-практичної конференції молодих учених та студентів

24-25 листопада 2021 року

УКРАЇНА
ТЕРНОПІЛЬ – 2021

СЕКЦІЯ: КОМПЮТЕРНО-ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА СИСТЕМИ ЗВ'ЯЗКУ

Матеріали X Міжнародної науково-практичної конференції молодих учених та студентів «АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ» – Тернопіль 24-25 листопада 2021 року

УДК 004.031.6:621.317.7

В.П. Волоський, к.т.н. Ю.З. Лещишин, Н.Р. Романишин
Тернопільський національний технічний університет імені Івана Пулюя, Україна

КОМП'ЮТЕРНА СИСТЕМА КОНТРОЛЮ ТА БАЛАНСУВАННЯ ЛІТІЙ-ІОННИХ АКУМУЛЯТОРНИХ БАТАРЕЙ

V.P. Voloskyi, Ph.D. Yu. Z. Leshchyshyn, N.R.Romanyshyn

COMPUTER CONTROL SYSTEM AND BALANCING OF LITHIUM-ION BATTERIES

Сучасні тенденції активного використання альтернативних джерел електричної енергії пов'язані із потребою зберігання енергії, оскільки генеруючі потужності працюють лише у певні моменти часу (наявність сонця, вітру, тощо). Одним із способів зберігання електричної енергії є використання літій-іонних акумуляторів. Такий тип акумуляторних батарей (АКБ) містить багато комірок, які для ефективного та тривалого використання, потребують контролю режимів заряду та розряду. Ці задачі покладені на систему контрою батареї (СКБ) (battery management system — BMS).

СКБ відслідковує режими заряду і розряду батареї, захищає її від перезаряду і пере розряду, що є критичними для літій-іонних акумуляторів. Також важливою задачею СКБ є балансування комірок батареї, що суттєво підвищує ефективність і тривалість використання батареї. Зокрема дану технологію найбільш активно використовують при розробці різноманітних рухомих засобів від безпілотників до електроавтомобілів (Nisan, Tesla та інші), що збільшує запас ходу до 20% [1].

Тому важливою задачею є побудова комп'ютерної системи контрою та балансування літій-іонних акумуляторних батарей, яка б не лише виконувала функції СКБ але й мала можливість обміну даними з контролером вищого рівня та гнучкого нарощення архітектури при збільшенні ємкості енергосховища.

Для побудови такої СКБ необхідно вирішити такі завдання: 1) забезпечити режими роботи батареї та її захист від перезаряду і перерозряду (вирішується існуючими мікросхеми чи алгоритмами); 2) забезпечити балансування комірок батареї, що є не простою задачею, оскільки існує багато алгоритмів балансування і вони потребують оптимального налаштування до параметрів літій-іонного акумулятора та його старіння в процесі експлуатації; 3) забезпечити гнучкість зміни архітектури при збільшенні ємкості АКБ шляхом застосування ієрархічної структури з обміном по протоколу CAN, SPI та ін. Найскладнішим є саме друге завдання.

Складність балансування полягає в тому що, найчастіше використовувані алгоритми пасивного балансування є простими і мають низьку точність, або складними і дають високу точність балансування. Існуючі алгоритми поділяють на такі, які базуються на: поточній напрузі комірки батареї, напрузі комірки при розімкненому колі Open-circuit voltage (OCV), і рівні заряду комірки State of charge (SoC).

Алгоритм, який базується на напрузі комірки батареї є легким в реалізації, але може бути не точним, оскільки в даному алгоритмі припускається, що всі послідовно включені комірки мають однаковий SoC, якщо напруга на їхніх клемах є однаковою. Однак дане твердження буде хибним, якщо одна із комірок має більший внутрішній опір, а отже дасть менше енергії, на відміну від сусідньої комірки з меншим опором. І такий розбаланс може призводити до втрати більше 13% ємності АКБ [1].

Алгоритм, який базується на оцінці значення OCV передбачає, що заздалегідь, на випробувальному стенді отримано криві OCV та SoC для діапазону температур від -20 °C до +60 °C, які зберігаються в пам'яті мікроконтролера для визначення поточної енергії акумулятора за його напругою. Даний алгоритм є недосконалим, оскільки у зв'язку із старінням АКБ ємність кожної комірки змінюється по різному, що призведе до втрат енергії та прискорення старіння батарей. Розбаланс для такого алгоритму на початку експлуатації може бути не значним, але з часом він досягає 8% [1].

Алгоритм, який базується на оцінці значення SoC — це технічно найскладніший алгоритм балансування. Він базується на інформації про історію SoC кожної комірки та обчисленні часу, який необхідний для балансування кожної комірки на основі методики [1]. В результаті застосування цього алгоритму розбаланс комірок АКБ можна звести до 1.6% [1], однак він є складним в реалізації, оскільки необхідно з високою точністю рахувати накопичену енергію кожною коміркою та час балансування, зберігаючи величину енергії в процесі як заряду так і її розряду.

Оптимального балансування АКБ, що можна досягнути об'єднавши методи балансування за напругою та OCV. Цей модифікований алгоритм при старті системи визначає внутрішній опір кожної комірки. Якщо різниця внутрішніх опорів при однаковій напрузі більша допустимого значення, робота такої системи є неможливою бо комірка є надто деградованою і АКБ може не балансуватись, в іншому випадку система визначає внутрішню енергію комірок за допомогою таблиць OCV та розпочинає роботу. Старт балансування комірок відбувається, із початком зарядки АКБ. Балансуватимуться невеликими струмами усі комірки, які мають різницю напруг більшу за задане значення ΔV. Балансування АКБ буде складати 75-100% часу зарядки.

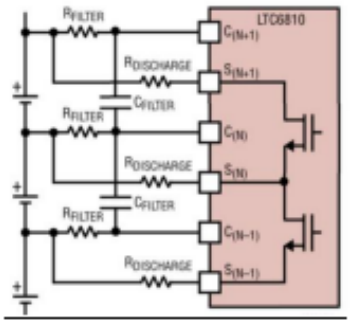

Рисунок 1 – Схема підключення акумуляторів до мікросхеми балансування

Такий, модифікований алгоритм доцільно використовувати при використанні сучасних спеціалізованих мікросхем типу LTC6810, особливо, якщо кола вимірювання напруг та балансування будуть незалежними, як це показано на рисунку 1.

Отже, СКБ, яка використовує модифікований алгоритм, що базується на об'єднанні двох існуючих алгоритмів. Балансування відбувається невеликими струмами протягом всього часу заряджання з досить високою точністю. Це зменшує витрати енергій на балансування, а також зменшує час балансування АКБ. Алгоритм розроблено спеціально для сучасних мікросхем балансування літій-іонних акумуляторів і є простим в реалізації. А його застосування для побудови систем зберігання електричної енергії та рухомих об'єктів з електричним приводом підвищує їх ефективність на 15-20% за рахунок повнішого накопичення і віддачі енергії АКБ. Сама ж АКБ матиме рівномірне старіння комірок, що підвищить її строк експлуатації. В подальшому необхідно визначити ефективність балансування запропонованого алгоритму у порівнянні з іншими подібними алгоритмами, що потребує спеціалізованого вимірювального обладнання з високою точністю.

Література:

1. Fleischer C. Simulative comparison of balancing algorithms for active and passive cell balancing systems for lithium-ion batteries / C. Fleischer, B. Ostendorp, D. Uwe Sauer. – 802. – P. 9.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

IX НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

8–9 грудня 2021 року

ТЕРНОПІЛЬ
2021

УДК 004.031.6:621.317.7

В.П. Волоський, к.т.н. Ю.З. Лещишин, Н.Р. Романишин
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

АЛГОРИТМ БАЛАНСУВАННЯ LI-ION АКУМУЛЯТОРНИХ БАТАРЕЙ НА ОСНОВІ ПОТОЧНОЇ НАПРУГИ ТА НАПРУГИ ПРИ РОЗІМКНЕНОМУ КОЛІ

V.P. Voloskyi, Ph. D. Yu. Z. Leshchyshyn, N.R. Romanishin

LI-ION BATTERY BALANCING ALGORITHM BASED ON CURRENT VOLTAGE AND OPEN CIRCUIT VOLTAGE

Сучасні транспортні засоби і не тільки, використовують Li-ion акумуляторні батареї (АКБ). Однак напруга однієї Li-ion комірки є низькою, тому електромобілі та накопичувачі енергії великої потужності використовують напругу у сотні вольт для оптимізації продуктивності силових систем. Тобто батареї таких систем будуть мати велику кількість комірок, з'єднаних послідовно і паралельно.

Хоча комірки батареї, як правило виготовлені за однакових умов, однак найменша різниця у внутрішньому опорі комірок, чи номінальній ємності, чи рівні старіння або температурі навколишнього середовища призведе до дисбалансу заряду цих елементів батареї. Такий дисбаланс комірок батареї призводить до зменшення її ємності та пошкодження комірок внаслідок перерозряду або перезаряду. Уникають дисбалансу шляхом балансування комірок за алгоритмами, які базуються на: поточній напрузі комірки батареї, напрузі комірки при розімкненому колі Open-circuit voltage (OCV), і рівні заряду комірки State of charge (SoC).

Запропонований алгоритм об'єднує в собі алгоритми балансування які базуються на поточній напрузі комірки батареї та на напрузі комірки при розімкненому колі (OCV). Цей алгоритм при старті системи визначає внутрішній опір кожної комірки. Якщо різниця внутрішніх опорів при однаковій напрузі більша допустимого значення, робота такої системи є неможливою бо комірка є надто деградованою і АКБ може не балансуватись, в іншому випадку система визначає внутрішню енергію комірок за допомогою таблиць OCV та розпочинає роботу. Старт балансування комірок відбувається, із початком зарядки АКБ. Балансуватимуться усі комірки, які мають різницю напруг більшу за задане значення ΔV. Час балансування таких комірки буде складати 75% часу зарядки АКБ. Такий, модифікований алгоритм доцільно використовувати при використанні сучасних спеціалізованих мікросхему LTC6810, особливо, якщо кола вимірювання напруг та балансування будуть незалежними [1].

Застосування розробленого алгоритму для побудови систем зберігання електричної енергії та електроавтомобілів підвищує їх ефективність на 15-20% за рахунок повнішого накопичення і віддачі енергії АКБ. Сама ж АКБ матиме рівномірне старіння комірок та захист від перерозряду або перезаряду, що підвищить її термін експлуатації. В подальшому необхідно визначити криві заряду та ефективність балансування запропонованого алгоритму у порівнянні з іншими подібними алгоритмами, що потребує спеціалізованого вимірювального обладнання з високою точністю.

Література.

1. Волоський В.П. Комп'ютерна система контролю та балансування літій-іонних акумуляторних батарей // В.П. Волоський, Ю.З. Лещишин, Н.Р. Романишин // Зб. тез доповідей X-ї Міжнародної науково-технічної конференції молодих учених та студентів «Актуальні задачі сучасних технологій» – Тернопіль: ТНТУ, 2021.

ДОДАТОК В

Лістинг програми

```
#include <Arduino.h>
#include <stdint.h>
#include <SPI.h>
#include "Linduino.h"
#include "LT_SPI.h"
#include "UserInterface.h"
#include "LTC681x.h"
#include "LTC6813.h"
#include "balanse_cell.h"
#define ENABLED 1
#define DISABLED 0
#define DATALOG_ENABLED 1
#define DATALOG_DISABLED 0
#define PWM 1
#define SCTL 2
void measurement_loop(uint8_t datalog_en);
void print_menu(void);
void print_wrconfig(void);
void print_wrconfigb(void);
void print_rxconfig(void);
void print_rxconfigb(void);
void print_cells(uint8_t datalog_en);
void print_aux(uint8_t datalog_en);
void print_stat(void);
void print_aux1(uint8_t datalog_en);
void print_sumofcells(void);
void check_mux_fail(void);
void print_selftest_errors(uint8_t adc_reg ,int8_t error);
void print_overlap_results(int8_t error);
void print_digital_redundancy_errors(uint8_t adc_reg ,int8_t error);
void print_open_wires(void);
void print_pec_error_count(void);
int8_t select_s_pin(void);
void print_wrpwm(void);
void print_rxpwm(void);
void print_wrsctrl(void);
void print_rxsctrl(void);
void print_wrpsb(uint8_t type);
void print_rxpsb(uint8_t type);
void print_wrcomm(void);
void print_rxcomm(void);
void check_mute_bit(void);
void print_conv_time(uint32_t conv_time);
void check_error(int error);
void serial_print_text(char data[]);
void serial_print_hex(uint8_t data);
char read_hex(void);
```

```
char get_char(void);
const uint8_t ADC_OPT = ADC_OPT_DISABLED;
const uint8_t ADC_CONVERSION_MODE =MD_422HZ_1KHZ;
const uint8_t ADC_DCP = DCP_DISABLED;
const uint8_t CELL_CH_TO_CONVERT =CELL_CH_ALL;
const uint8_t AUX_CH_TO_CONVERT = AUX_CH_ALL;
const uint8_t STAT_CH_TO_CONVERT = STAT_CH_ALL;
const uint8_t SEL_ALL_REG = REG_ALL;
const uint8_t SEL_REG_A = REG_1;
const uint8_t SEL_REG_B = REG_2;
const uint16_t MEASUREMENT_LOOP_TIME = 500;
const uint16_t OV_THRESHOLD = 41000;
const uint16_t UV_THRESHOLD = 30000;
const uint8_t WRITE_CONFIG = DISABLED;
const uint8_t READ_CONFIG = DISABLED;
const uint8_t MEASURE_CELL = ENABLED;
const uint8_t MEASURE_AUX = DISABLED;
const uint8_t MEASURE_STAT = DISABLED;
const uint8_t PRINT_PEC = DISABLED;
cell_asic BMS_IC[TOTAL_IC];
bool REFON = true;
bool ADCOPT = false;
bool GPIOBITS_A[5] = {false,false,true,true,true};
bool GPIOBITS_B[4] = {false,false,false,false};
uint16_t UV=UV_THRESHOLD;
uint16_t OV=OV_THRESHOLD;
bool DCCBITS_A[12] =
{false,false,false,false,false,false,false,false,false,false,false,fal
se};
bool DCCBITS_B[7]= {false,false,false,false,false,false,false};
bool DCTOBITS[4] = {true,false,true,false};
bool FDRF = false;
bool DTMEN = true;
bool PSBITS[2]= {false,false};
void setup()
{
 Serial.begin(115200);
 quikeval_SPI_connect();
 spi_enable(SPI_CLOCK_DIV16);
 LTC6813_init_cfg(TOTAL_IC, BMS_IC);
 LTC6813_init_cfgb(TOTAL_IC,BMS_IC);
 for (uint8_t current_ic = 0; current_ic<TOTAL_IC;current_ic++)
 {

LTC6813_set_cfgr(current_ic,BMS_IC,REFON,ADCOPT,GPIOBITS_A,DCCBITS_A,
DCTOBITS, UV, OV);

LTC6813_set_cfgrb(current_ic,BMS_IC,FDRF,DTMEN,PSBITS,GPIOBITS_B,DCCBI
TS_B);
 }
 LTC6813_reset_crc_count(TOTAL_IC,BMS_IC);
 LTC6813_init_reg_limits(TOTAL_IC,BMS_IC);
 print_menu();
```

```
}
 \brief Main loop
  @return void
*****************************************************************************
*
void loop()
{
  if (Serial.available())
 {
  uint32_t user_command;
  user_command = read_int();
  if(user_command=='m')
  {
   print_menu();
  }
  else
  {
   Serial.println(user_command);
   run_command(user_command);
  }
 }
}
 \brief executes the user command
  @return void
*****************************************
void run_command(uint32_t cmd)
{

 uint8_t streg=0;
 int8_t error = 0;
 uint32_t conv_time = 0;
 int8_t s_pin_read=0;

 switch (cmd)
 {
  case 1:
   wakeup_sleep(TOTAL_IC);
   LTC6813_wrcfg(TOTAL_IC,BMS_IC);
   LTC6813_wrcfgb(TOTAL_IC,BMS_IC);
   print_wrconfig();
   print_wrconfigb();

   wakeup_idle(TOTAL_IC);
   error = LTC6813_rdcfg(TOTAL_IC,BMS_IC);
   check_error(error);
   error = LTC6813_rdcfgb(TOTAL_IC,BMS_IC);
   check_error(error);
   print_rxconfig();
   print_rxconfigb();
   break;
  case 2:
   wakeup_sleep(TOTAL_IC);
   error = LTC6813_rdcfg(TOTAL_IC,BMS_IC);
```

```
 check_error(error);
 error = LTC6813_rdcfgb(TOTAL_IC,BMS_IC);
 check_error(error);
 print_rxconfig();
 print_rxconfigb();
 break;

case 3:
 wakeup_sleep(TOTAL_IC);
 LTC6813_adcv(ADC_CONVERSION_MODE,ADC_DCP,CELL_CH_TO_CONVERT);
 conv_time = LTC6813_pollAdc();
 print_conv_time(conv_time);
 break;

case 4:
 wakeup_sleep(TOTAL_IC);
 error = LTC6813_rdcv(SEL_ALL_REG,TOTAL_IC,BMS_IC);
 check_error(error);
 print_cells(DATALOG_DISABLED);
 break;

case 5:
 wakeup_sleep(TOTAL_IC);
 LTC6813_adax(ADC_CONVERSION_MODE, AUX_CH_TO_CONVERT);
 conv_time = LTC6813_pollAdc();
 print_conv_time(conv_time);
 break;

case 6:
 wakeup_sleep(TOTAL_IC);
 error = LTC6813_rdaux(SEL_ALL_REG,TOTAL_IC,BMS_IC);
 check_error(error);
 print_aux(DATALOG_DISABLED);
 break;

case 7:
 wakeup_sleep(TOTAL_IC);
 LTC6813_adstat(ADC_CONVERSION_MODE, STAT_CH_TO_CONVERT);
 conv_time = LTC6813_pollAdc();
 print_conv_time(conv_time);
 break;

case 8:
 wakeup_sleep(TOTAL_IC);
 error = LTC6813_rdstat(SEL_ALL_REG,TOTAL_IC,BMS_IC);
 check_error(error);
 print_stat();
 break;

case 9:
 wakeup_sleep(TOTAL_IC);
 LTC6813_adcvax(ADC_CONVERSION_MODE,ADC_DCP);
 conv_time = LTC6813_pollAdc();
```

```
 print_conv_time(conv_time);
 wakeup_idle(TOTAL_IC);
 error = LTC6813_rdcv(SEL_ALL_REG, TOTAL_IC,BMS_IC);
 check_error(error);
 print_cells(DATALOG_DISABLED);
 wakeup_idle(TOTAL_IC);
 error = LTC6813_rdaux(SEL_REG_A,TOTAL_IC,BMS_IC);
 check_error(error);
 print_aux1(DATALOG_DISABLED);
 break;

case 10:
 wakeup_sleep(TOTAL_IC);
 LTC6813_adcvsc(ADC_CONVERSION_MODE,ADC_DCP);
 conv_time = LTC6813_pollAdc();
 print_conv_time(conv_time);
 wakeup_idle(TOTAL_IC);
 error = LTC6813_rdcv(SEL_ALL_REG, TOTAL_IC,BMS_IC);
 check_error(error);
 print_cells(DATALOG_DISABLED);
 wakeup_idle(TOTAL_IC);
 error = LTC6813_rdstat(SEL_REG_A,TOTAL_IC,BMS_IC);
 check_error(error);
 print_sumofcells();
 break;

case 11:
 wakeup_sleep(TOTAL_IC);
 LTC6813_wrcfg(TOTAL_IC,BMS_IC);
 LTC6813_wrcfgb(TOTAL_IC,BMS_IC);
 measurement_loop(DATALOG_DISABLED);
 print_menu();
 break;

case 12:
 wakeup_sleep(TOTAL_IC);
 LTC6813_wrcfg(TOTAL_IC,BMS_IC);
 LTC6813_wrcfgb(TOTAL_IC,BMS_IC);
 measurement_loop(DATALOG_ENABLED);
 print_menu();
 break;
case 13:
 wakeup_sleep(TOTAL_IC);
 LTC6813_clrcell();
 LTC6813_clraux();
 LTC6813_clrstat();
 wakeup_idle(TOTAL_IC);
 LTC6813_rdcv(SEL_ALL_REG, TOTAL_IC,BMS_IC);
 print_cells(DATALOG_DISABLED);
 LTC6813_rdaux(SEL_ALL_REG,TOTAL_IC,BMS_IC);
 print_aux(DATALOG_DISABLED);
 LTC6813_rdstat(SEL_ALL_REG,TOTAL_IC,BMS_IC);
 print_stat();
```

```
   break;

  case 14:
   wakeup_sleep(TOTAL_IC);
   LTC6813_diagn();
   LTC6813_pollAdc();
   error = LTC6813_rdstat(SEL_REG_B,TOTAL_IC,BMS_IC);
   check_error(error);
   check_mux_fail();
   break;

  case 15:
   error = 0;
   wakeup_sleep(TOTAL_IC);
   error =
LTC6813_run_cell_adc_st(CELL,TOTAL_IC,BMS_IC,ADC_CONVERSION_MODE,ADCOP
T);
   print_selftest_errors(CELL, error);
   error = 0;
   wakeup_sleep(TOTAL_IC);
   error = LTC6813_run_cell_adc_st(AUX,TOTAL_IC,
BMS_IC,ADC_CONVERSION_MODE,ADCOPT);
   print_selftest_errors(AUX, error);

   error = 0;
   wakeup_sleep(TOTAL_IC);
   error = LTC6813_run_cell_adc_st(STAT,TOTAL_IC,
BMS_IC,ADC_CONVERSION_MODE,ADCOPT);
   print_selftest_errors(STAT, error);
   break;
  case 16:
   wakeup_sleep(TOTAL_IC);
   error = (int8_t)LTC6813_run_adc_overlap(TOTAL_IC,BMS_IC);
   print_overlap_results(error);
   break;

  case 17:
   wakeup_sleep(TOTAL_IC);
   error =
LTC6813_run_adc_redundancy_st(ADC_CONVERSION_MODE,AUX,TOTAL_IC,
BMS_IC);
   print_digital_redundancy_errors(AUX, error);

   wakeup_sleep(TOTAL_IC);
   error =
LTC6813_run_adc_redundancy_st(ADC_CONVERSION_MODE,STAT,TOTAL_IC,
BMS_IC);
   print_digital_redundancy_errors(STAT, error);
   break;

  case 18:
   wakeup_sleep(TOTAL_IC);
   LTC6813_run_openwire_single(TOTAL_IC, BMS_IC);
```

```
 print_open_wires();
 break;
case 19:
 wakeup_sleep(TOTAL_IC);
 LTC6813_run_openwire_multi(TOTAL_IC, BMS_IC);
 break;
case 20:
 wakeup_sleep(TOTAL_IC);
 LTC6813_run_gpio_openwire(TOTAL_IC, BMS_IC);
 print_open_wires();
 break;

case 21:
 print_pec_error_count();
 break;

case 22:
 LTC6813_reset_crc_count(TOTAL_IC,BMS_IC);
 print_pec_error_count();
 break;

case 23:
 s_pin_read = select_s_pin();
 wakeup_sleep(TOTAL_IC);
 LTC6813_set_discharge(s_pin_read,TOTAL_IC,BMS_IC);
 LTC6813_wrcfg(TOTAL_IC,BMS_IC);
 LTC6813_wrcfgb(TOTAL_IC,BMS_IC);
 print_wrconfig();
 print_wrconfigb();
 wakeup_idle(TOTAL_IC);
 error = LTC6813_rdcfg(TOTAL_IC,BMS_IC);
 check_error(error);
 error = LTC6813_rdcfgb(TOTAL_IC,BMS_IC);
 check_error(error);
 print_rxconfig();
 print_rxconfigb();
 break;

case 24:
 wakeup_sleep(TOTAL_IC);
 LTC6813_clear_discharge(TOTAL_IC,BMS_IC);
 LTC6813_wrcfg(TOTAL_IC,BMS_IC);
 LTC6813_wrcfgb(TOTAL_IC,BMS_IC);
 print_wrconfig();
 print_wrconfigb();
 wakeup_idle(TOTAL_IC);
 error = LTC6813_rdcfg(TOTAL_IC,BMS_IC);
 check_error(error);
 error = LTC6813_rdcfgb(TOTAL_IC,BMS_IC);
 check_error(error);
 print_rxconfig();
 print_rxconfigb();
 break;
```

```
  case 25:

     PWM configuration data.
     1)Set the corresponding DCC bit to one for pwm operation.
     2)Set the DCTO bits to the required discharge time.
     3)Choose the value to be configured depending on the
     required duty cycle.
     Refer to the data sheet.
    ******************************************************
    wakeup_sleep(TOTAL_IC);
    for (uint8_t current_ic = 0; current_ic<TOTAL_IC;current_ic++)
    {
     BMS_IC[current_ic].pwm.tx_data[0]= 0x88;
     BMS_IC[current_ic].pwm.tx_data[1]= 0x88;
     BMS_IC[current_ic].pwm.tx_data[2]= 0x88;
     BMS_IC[current_ic].pwm.tx_data[3]= 0x88;
     BMS_IC[current_ic].pwm.tx_data[4]= 0x88;
     BMS_IC[current_ic].pwm.tx_data[5]= 0x88;
    }

    LTC6813_wrpwm(TOTAL_IC,0,BMS_IC);
    for (uint8_t current_ic = 0; current_ic<TOTAL_IC;current_ic++)
    {
     BMS_IC[current_ic].pwmb.tx_data[0]= 0x88;
     BMS_IC[current_ic].pwmb.tx_data[1]= 0x88;
     BMS_IC[current_ic].pwmb.tx_data[2]= 0x88;
    }

    LTC6813_wrpsb(TOTAL_IC,BMS_IC);
    print_wrpwm();
    print_wrpsb(PWM);
    wakeup_idle(TOTAL_IC);
    error=LTC6813_rdpwm(TOTAL_IC,0,BMS_IC);
    check_error(error);

    error=LTC6813_rdpsb(TOTAL_IC,BMS_IC);
    check_error(error);
    print_rxpwm();
    print_rxpsb(PWM);
    break;
  case 26:

     S pin control.
     1)Ensure that the pwm is set according to the requirement using
the previous case.
     2)Choose the value depending on the required number of pulses on
S pin.
     Refer to the data sheet.

**************************************************************************
****************
    wakeup_sleep(TOTAL_IC);
```

```
 for (uint8_t current_ic = 0; current_ic<TOTAL_IC;current_ic++)
 {
  BMS_IC[current_ic].sctrl.tx_data[0]= 0xFF;
  BMS_IC[current_ic].sctrl.tx_data[1]= 0xFF;
  BMS_IC[current_ic].sctrl.tx_data[2]= 0xFF;
  BMS_IC[current_ic].sctrl.tx_data[3]= 0xFF;
  BMS_IC[current_ic].sctrl.tx_data[4]= 0xFF;
  BMS_IC[current_ic].sctrl.tx_data[5]= 0xFF;
 }

 LTC6813_wrsctrl(TOTAL_IC,streg,BMS_IC);
 for (uint8_t current_ic = 0; current_ic<TOTAL_IC;current_ic++)
 {
  BMS_IC[current_ic].sctrlb.tx_data[3]= 0xFF;
  BMS_IC[current_ic].sctrlb.tx_data[4]= 0xFF;
  BMS_IC[current_ic].sctrlb.tx_data[5]= 0xFF;
 }
 LTC6813_wrpsb(TOTAL_IC,BMS_IC);
 print_wrsctrl();
 print_wrpsb(SCTL);
 wakeup_idle(TOTAL_IC);
 LTC6813_stsctrl();

 wakeup_idle(TOTAL_IC);
 error=LTC6813_rdsctrl(TOTAL_IC,streg,BMS_IC);
 check_error(error);

 error=LTC6813_rdpsb(TOTAL_IC,BMS_IC);
 check_error(error);
 print_rxsctrl();
 print_rxpsb(SCTL);
 break;

case 27:
 wakeup_sleep(TOTAL_IC);
 LTC6813_clrsctrl();
 wakeup_idle(TOTAL_IC);
 error=LTC6813_rdsctrl(TOTAL_IC,streg,BMS_IC);
 check_error(error);
 LTC6813_rdpsb(TOTAL_IC,BMS_IC);
 print_rxsctrl();
 print_rxpsb(SCTL);
 break;

case 28:

   Ensure to set the GPIO bits to 1 in the CFG register group.
 ***********************************************************
 for (uint8_t current_ic = 0; current_ic<TOTAL_IC;current_ic++)
 {

  BMS_IC[current_ic].com.tx_data[0]= 0x81;
```

```
   BMS_IC[current_ic].com.tx_data[1]= 0x10;
   BMS_IC[current_ic].com.tx_data[2]= 0xA2;
   BMS_IC[current_ic].com.tx_data[3]= 0x50;
   BMS_IC[current_ic].com.tx_data[4]= 0xA1;
   BMS_IC[current_ic].com.tx_data[5]= 0x79;
  }
  wakeup_sleep(TOTAL_IC);
  LTC6813_wrcomm(TOTAL_IC,BMS_IC);
  print_wrcomm();
  wakeup_idle(TOTAL_IC);
  LTC6813_stcomm(3);
  wakeup_idle(TOTAL_IC);
  error = LTC6813_rdcomm(TOTAL_IC,BMS_IC);
  check_error(error);
  print_rxcomm();
  break;
case 29:

    Ensure to set the GPIO bits to 1 in the CFG register group.
  ***********************************************************
  for (uint8_t current_ic = 0; current_ic<TOTAL_IC;current_ic++)
  {

   BMS_IC[current_ic].com.tx_data[0]= 0x6A;
   BMS_IC[current_ic].com.tx_data[1]= 0x08;
   BMS_IC[current_ic].com.tx_data[2]= 0x00;
   BMS_IC[current_ic].com.tx_data[3]= 0x08;
   BMS_IC[current_ic].com.tx_data[4]= 0x01;
   BMS_IC[current_ic].com.tx_data[5]= 0x39;
  }
  wakeup_sleep(TOTAL_IC);
  LTC6813_wrcomm(TOTAL_IC,BMS_IC);
  print_wrcomm();
  wakeup_idle(TOTAL_IC);
  LTC6813_stcomm(3);
  wakeup_idle(TOTAL_IC);
  error = LTC6813_rdcomm(TOTAL_IC,BMS_IC);
  check_error(error);
  print_rxcomm();
  break;
 case 30:

   Ensure to set the GPIO bits to 1 in the CFG register group.
  ***********************************************************
  for (uint8_t current_ic = 0; current_ic<TOTAL_IC;current_ic++)
  {

   BMS_IC[current_ic].com.tx_data[0]= 0x6A;
   BMS_IC[current_ic].com.tx_data[1]= 0x08;
   BMS_IC[current_ic].com.tx_data[2]= 0x00;
   BMS_IC[current_ic].com.tx_data[3]= 0x08;
   BMS_IC[current_ic].com.tx_data[4]= 0x6A;
   BMS_IC[current_ic].com.tx_data[5]= 0x18;
```

```
   }
   wakeup_sleep(TOTAL_IC);
   LTC6813_wrcomm(TOTAL_IC,BMS_IC);
   wakeup_idle(TOTAL_IC);
   LTC6813_stcomm(3);
   for (uint8_t current_ic = 0; current_ic<TOTAL_IC;current_ic++)
   {

    BMS_IC[current_ic].com.tx_data[0]= 0x0F;
    BMS_IC[current_ic].com.tx_data[1]= 0xF9;
    BMS_IC[current_ic].com.tx_data[2]= 0x7F;
    BMS_IC[current_ic].com.tx_data[3]= 0xF9;
    BMS_IC[current_ic].com.tx_data[4]= 0x7F;
    BMS_IC[current_ic].com.tx_data[5]= 0xF9;
   }
   wakeup_idle(TOTAL_IC);
   LTC6813_wrcomm(TOTAL_IC,BMS_IC);
   wakeup_idle(TOTAL_IC);
   LTC6813_stcomm(1);
   wakeup_idle(TOTAL_IC);
   error = LTC6813_rdcomm(TOTAL_IC,BMS_IC);
   check_error(error);
   print_rxcomm();
   break;
  case 31:
   wakeup_sleep(TOTAL_IC);
   LTC6813_mute();
   wakeup_idle(TOTAL_IC);
   error = LTC6813_rdcfgb(TOTAL_IC,BMS_IC);
   check_error(error);
   check_mute_bit();
   break;

  case 32:
   wakeup_sleep(TOTAL_IC);
   LTC6813_unmute();
   wakeup_idle(TOTAL_IC);
   error = LTC6813_rdcfgb(TOTAL_IC,BMS_IC);
   check_error(error);
   check_mute_bit();
   break;
  case 33:

    Please ensure you have set the GPIO bits according to your
requirement
    in the configuration register.( check the global variable
GPIOBITS_A )

**********************************************************************
**
   wakeup_sleep(TOTAL_IC);
   for (uint8_t current_ic = 0; current_ic<TOTAL_IC;current_ic++)
   {
```

```
LTC6813_set_cfgr(current_ic,BMS_IC,REFON,ADCOPT,GPIOBITS_A,DCCBITS_A,
DCTOBITS, UV, OV);

LTC6813_set_cfgrb(current_ic,BMS_IC,FDRF,DTMEN,PSBITS,GPIOBITS_B,DCCBI
TS_B);
   }
   LTC6813_wrcfg(TOTAL_IC,BMS_IC);
   LTC6813_wrcfgb(TOTAL_IC,BMS_IC);
   print_wrconfig();
   print_wrconfigb();
   break;

  case 34:
  for (int i = 0; i<100; i++)
  {
   conv_time = balanse_cell(TOTAL_IC,ADC_CONVERSION_MODE,ADC_DCP,
CELL_CH_TO_CONVERT);
   print_conv_time(conv_time);
   wakeup_sleep(TOTAL_IC);
   error = LTC6813_rdcv(SEL_ALL_REG,TOTAL_IC,BMS_IC);
   check_error(error);
   print_cells(DATALOG_DISABLED);
  }

    *
    *
   s_pin_read = select_s_pin();
   wakeup_sleep(TOTAL_IC);
   LTC6813_set_discharge(s_pin_read,TOTAL_IC,BMS_IC);
   LTC6813_wrcfg(TOTAL_IC,BMS_IC);
   LTC6813_wrcfgb(TOTAL_IC,BMS_IC);
   print_wrconfig();
   print_wrconfigb();
   wakeup_idle(TOTAL_IC);
   error = LTC6813_rdcfg(TOTAL_IC,BMS_IC);
   check_error(error);
   error = LTC6813_rdcfgb(TOTAL_IC,BMS_IC);
   check_error(error);
   print_rxconfig();
   print_rxconfigb();
   break;*
   break;
  case 'm':
   print_menu();
   break;
  default:
   char str_error[]="Incorrect Option \n";
   serial_print_text(str_error);
   break;
 }
}
 \brief For writing
```

```
 @return void
*****************************************************************************
*****************************************************************************
*****
void measurement_loop(uint8_t datalog_en)
{
 int8_t error = 0;
 char input = 0;

 Serial.println(F("Transmit 'm' to quit"));

 while (input != 'm')
 {
   if (Serial.available() > 0)
   {
    input = read_char();
   }

   if (WRITE_CONFIG == ENABLED)
   {
    wakeup_idle(TOTAL_IC);
    LTC6813_wrcfg(TOTAL_IC,BMS_IC);
    LTC6813_wrcfgb(TOTAL_IC,BMS_IC);
    print_wrconfig();
    print_wrconfigb();
   }

   if (READ_CONFIG == ENABLED)
   {
    wakeup_idle(TOTAL_IC);
    error = LTC6813_rdcfg(TOTAL_IC,BMS_IC);
    check_error(error);
    error = LTC6813_rdcfgb(TOTAL_IC,BMS_IC);
    check_error(error);
    print_rxconfig();
    print_rxconfigb();
   }

   if (MEASURE_CELL == ENABLED)
   {
    wakeup_idle(TOTAL_IC);
    LTC6813_adcv(ADC_CONVERSION_MODE,ADC_DCP,CELL_CH_TO_CONVERT);
    LTC6813_pollAdc();
    wakeup_idle(TOTAL_IC);
    error = LTC6813_rdcv(0, TOTAL_IC,BMS_IC);
    check_error(error);
    print_cells(datalog_en);
   }

   if (MEASURE_AUX == ENABLED)
   {
    wakeup_idle(TOTAL_IC);
    LTC6813_adax(ADC_CONVERSION_MODE , AUX_CH_ALL);
```

```
    LTC6813_pollAdc();
    wakeup_idle(TOTAL_IC);
    error = LTC6813_rdaux(0,TOTAL_IC,BMS_IC);
    check_error(error);
    print_aux(datalog_en);
   }

   if (MEASURE_STAT == ENABLED)
   {
    wakeup_idle(TOTAL_IC);
    LTC6813_adstat(ADC_CONVERSION_MODE, STAT_CH_ALL);
    LTC6813_pollAdc();
    wakeup_idle(TOTAL_IC);
    error = LTC6813_rdstat(0,TOTAL_IC,BMS_IC);
    check_error(error);
    print_stat();
   }

   if(PRINT_PEC == ENABLED)
   {
    print_pec_error_count();
   }

    delay(MEASUREMENT_LOOP_TIME);
 }
}
 \brief Prints the main menu
 @return void
*********************************
void print_menu(void)
{
 Serial.println(F("List of LTC6813 Command:"));
 Serial.println(F("Write and Read Configuration: 1              |Loop
measurements with data-log output : 12              |Set Discharge: 23
"));
 Serial.println(F("Read Configuration: 2                    |Clear
Registers: 13                          |Clear Discharge: 24  "));
 Serial.println(F("Start Cell Voltage Conversion: 3              |Run
Mux Self Test: 14                         |Write and Read of PWM :
25"));
 Serial.println(F("Read Cell Voltages: 4                    |Run ADC
Self Test: 15                          |Write and Read of S control :
26"));
 Serial.println(F("Start Aux Voltage Conversion: 5              |ADC
overlap Test : 16                         |Clear S control register :
27"));
 Serial.println(F("Read Aux Voltages: 6                     |Run
Digital Redundancy Test: 17                    |SPI Communication :
28"));
 Serial.println(F("Start Stat Voltage Conversion: 7              |Open
Wire Test for single cell detection: 18             |I2C
Communication Write to Slave :29"));
```

```
 Serial.println(F("Read Stat Voltages: 8                    |Open Wire
Test for multiple cell or two consecutive cells detection:19 |I2C
Communication Read from Slave :30"));
 Serial.println(F("Start Combined Cell Voltage and GPIO1, GPIO2
Conversion: 9 |Open wire for Auxiliary Measurement: 20
|Enable MUTE : 31"));
 Serial.println(F("Start Cell Voltage and Sum of cells : 10
|Print PEC Counter: 21                          |Disable MUTE : 32"));
 Serial.println(F("Loop Measurements: 11                    |Reset PEC
Counter: 22                          |Set or reset the gpio pins: 33 \n
"));
 Serial.println(F("\n Set PWM Discharge: 34"));
 Serial.println(F("Print 'm' for menu"));
 Serial.println(F("Please enter command: \n "));
}
 \brief Prints the configuration data that is going to be written to
the LTC6813
 to the serial port.
 @return void
 ************************************************************************
***********
void print_wrconfig(void)
{
  int cfg_pec;
  Serial.println(F("Written Configuration A Register: "));
  for (int current_ic = 0; current_ic<TOTAL_IC; current_ic++)
  {
   Serial.print(F("CFGA IC "));
   Serial.print(current_ic+1,DEC);
   for(int i = 0;i<6;i++)
   {
    Serial.print(F(", 0x"));
    serial_print_hex(BMS_IC[current_ic].config.tx_data[i]);
   }
   Serial.print(F(", Calculated PEC: 0x"));
   cfg_pec = pec15_calc(6,&BMS_IC[current_ic].config.tx_data[0]);
   serial_print_hex((uint8_t)(cfg_pec>>8));
   Serial.print(F(", 0x"));
   serial_print_hex((uint8_t)(cfg_pec));
   Serial.println("\n");
  }
}
 \brief Prints the Configuration Register B data that is going to be
written to
 the LTC6813 to the serial port.
 @return void
 ************************************************************************
***********
void print_wrconfigb(void)
{
  int cfg_pec;
  Serial.println(F("Written Configuration B Register: "));
  for (int current_ic = 0; current_ic<TOTAL_IC; current_ic++)
```

```
  {
   Serial.print(F("CFGB IC "));
   Serial.print(current_ic+1,DEC);
   for(int i = 0;i<6;i++)
   {
    Serial.print(F(", 0x"));
    serial_print_hex(BMS_IC[current_ic].configb.tx_data[i]);
   }
   Serial.print(F(", Calculated PEC: 0x"));
   cfg_pec = pec15_calc(6,&BMS_IC[current_ic].configb.tx_data[0]);
   serial_print_hex((uint8_t)(cfg_pec>>8));
   Serial.print(F(", 0x"));
   serial_print_hex((uint8_t)(cfg_pec));
   Serial.println("\n");
  }
}
 \brief Prints the configuration data that was read back from the
 LTC6813 to the serial port.
 @return void
 *****************************************************************
void print_rxconfig(void)
{
 Serial.println(F("Received Configuration A Register: "));
 for (int current_ic=0; current_ic<TOTAL_IC; current_ic++)
 {
  Serial.print(F("CFGA IC "));
  Serial.print(current_ic+1,DEC);
  for(int i = 0; i < 6; i++)
  {
   Serial.print(F(", 0x"));
   serial_print_hex(BMS_IC[current_ic].config.rx_data[i]);
  }
  Serial.print(F(", Received PEC: 0x"));
  serial_print_hex(BMS_IC[current_ic].config.rx_data[6]);
  Serial.print(F(", 0x"));
  serial_print_hex(BMS_IC[current_ic].config.rx_data[7]);
  Serial.println("\n");
 }
}
 \brief Prints the Configuration Register B that was read back from
 the LTC6813 to the serial port.
 @return void
 *****************************************************************
void print_rxconfigb(void)
{
 Serial.println(F("Received Configuration B Register: "));
 for (int current_ic=0; current_ic<TOTAL_IC; current_ic++)
 {
  Serial.print(F("CFGB IC "));
  Serial.print(current_ic+1,DEC);
  for(int i = 0; i < 6; i++)
  {
   Serial.print(F(", 0x"));
```

```
    serial_print_hex(BMS_IC[current_ic].configb.rx_data[i]);
   }
   Serial.print(F(", Received PEC: 0x"));
   serial_print_hex(BMS_IC[current_ic].configb.rx_data[6]);
   Serial.print(F(", 0x"));
   serial_print_hex(BMS_IC[current_ic].configb.rx_data[7]);
   Serial.println("\n");
  }
}
 \brief Prints cell voltage codes to the serial port
 @return void
 *****************************************************************
void print_cells(uint8_t datalog_en)
{
 for (int current_ic = 0 ; current_ic < TOTAL_IC; current_ic++)
 {
  if (datalog_en == 0)
  {
   Serial.print(" IC ");
   Serial.print(current_ic+1,DEC);
   Serial.print(", ");
   for (int i=0; i<18; i++)
   {
    Serial.print(" C");
    Serial.print(i+1,DEC);
    Serial.print(":");
    Serial.print(BMS_IC[current_ic].cells.c_codes[i]*0.0001,4);
    Serial.print(",");
   }

  }
  else
  {
   Serial.print(" Cells, ");
   for (int i=0; i<BMS_IC[0].ic_reg.cell_channels; i++)
   {
    Serial.print(BMS_IC[current_ic].cells.c_codes[i]*0.0001,4);
    Serial.print(",");
   }
  }
 }
 Serial.println();
}
 \brief Prints GPIO voltage codes and Vref2 voltage code onto the
serial port
 @return void
 *****************************************************************
********
void print_aux(uint8_t datalog_en)
{
 float sTemp[6] = {0};
 for (int current_ic =0 ; current_ic < TOTAL_IC; current_ic++)
 {
```

```
if (datalog_en == 0)
{
 Serial.print(" IC ");
 Serial.print(current_ic+1,DEC);
 for (int i=0; i < 5; i++)
 {
  Serial.print(F(" GPIO-"));
  Serial.print(i+1,DEC);
  Serial.print(":");
  sTemp[i] = BMS_IC[current_ic].aux.a_codes[i]*0.0001;
  Serial.print(sTemp[i],4);

  Serial.print(",");
 }

 for (int i=6; i < 10; i++)
 {
  Serial.print(F(" GPIO-"));
  Serial.print(i,DEC);
  Serial.print(":");

  Serial.print(BMS_IC[current_ic].aux.a_codes[i]*0.0001,4);
 }
 float vRef = BMS_IC[current_ic].aux.a_codes[5]*0.0001;
 Serial.print(F(" Vref2"));
 Serial.print(":");
 Serial.print(vRef,4);
 Serial.println();

 Serial.print(" OV
 Serial.print((uint8_t)BMS_IC[current_ic].aux.a_codes[11],HEX);
 Serial.println();
 sTemp[2]=1.5;
 float A = 1.0f
 float B = 1.0f
 float VR_T = vRef-sTemp[1];
 float T_VR_T = sTemp[1]
 float T_VR_T_1000 = 10000*T_VR_T;
 float T_VR_T_1000_log = log(T_VR_T_1000
 float A_B_T_VR_T_1000_log = A+B*T_VR_T_1000_log;
 Serial.println(A,5);
 Serial.println(B,5);
 Serial.println(VR_T,5);
 Serial.println(T_VR_T,5);
 Serial.println(T_VR_T_1000,5);
 Serial.println(T_VR_T_1000_log,6);
 Serial.println(A_B_T_VR_T_1000_log,5);

 float tempC2 = (1.0
 Serial.print(" tempC : ");
 Serial.println(tempC2);
}
else
```

```
  {
   Serial.print(" AUX, ");
   for (int i=0; i < 12; i++)
   {
    Serial.print((uint8_t)BMS_IC[current_ic].aux.a_codes[i]*0.0001,4);
    Serial.print(",");
   }
  }
 }
 Serial.println("\n");
}
 \brief Prints Status voltage codes and Vref2 voltage code onto the
serial port
 @return void
 ************************************************************************
********
void print_stat(void)
{
 for (int current_ic =0 ; current_ic < TOTAL_IC; current_ic++)
 {
  double itmp;

  Serial.print(F(" IC "));
  Serial.print(current_ic+1,DEC);
  Serial.print(F(" SOC:"));
  Serial.print(BMS_IC[current_ic].stat.stat_codes[0]*0.0001*30,4);
  Serial.print(F(","));
  Serial.print(F(" Itemp:"));
  itmp = (double)((BMS_IC[current_ic].stat.stat_codes[1] * (0.0001
  Serial.print(itmp,4);
  Serial.print(F(","));
  Serial.print(F(" VregA:"));
  Serial.print(BMS_IC[current_ic].stat.stat_codes[2]*0.0001,4);
  Serial.print(F(","));
  Serial.print(F(" VregD:"));
  Serial.print(BMS_IC[current_ic].stat.stat_codes[3]*0.0001,4);
  Serial.println();
  Serial.print(F(" OV
  Serial.print(F(", 0x"));
  serial_print_hex(BMS_IC[current_ic].stat.flags[0]);
  Serial.print(F(", 0x"));
  serial_print_hex(BMS_IC[current_ic].stat.flags[1]);
  Serial.print(F(", 0x"));
  serial_print_hex(BMS_IC[current_ic].stat.flags[2]);
   Serial.print(F("\tMux fail flag:"));
  Serial.print(F(", 0x"));
  serial_print_hex(BMS_IC[current_ic].stat.mux_fail[0]);
   Serial.print(F("\tTHSD:"));
  Serial.print(F(", 0x"));
  serial_print_hex(BMS_IC[current_ic].stat.thsd[0]);
  Serial.println();
 }
 Serial.println("\n");
```

```
}
 \brief Prints GPIO voltage codes (GPIO 1 & 2)
 @return void
 ***********************************************************************
********
void print_aux1(uint8_t datalog_en)
{
 for (int current_ic =0 ; current_ic < TOTAL_IC; current_ic++)
 {
  if (datalog_en == 0)
  {
   Serial.print(" IC ");
   Serial.print(current_ic+1,DEC);
   for (int i=0; i < 2; i++)
   {
    Serial.print(F(" GPIO-"));
    Serial.print(i+1,DEC);
    Serial.print(":");
    Serial.print(BMS_IC[current_ic].aux.a_codes[i]*0.0001,4);
    Serial.print(",");
   }
  }
  else
  {
   Serial.print("AUX, ");
   for (int i=0; i < 12; i++)
   {
    Serial.print(BMS_IC[current_ic].aux.a_codes[i]*0.0001,4);
    Serial.print(",");
   }
  }
 }
 Serial.println("\n");
}
 \brief Prints Status voltage codes for SOC onto the serial port
 ***********************************************************************
********
void print_sumofcells(void)
{
 for (int current_ic =0 ; current_ic < TOTAL_IC; current_ic++)
 {
  Serial.print(F(" IC "));
  Serial.print(current_ic+1,DEC);
  Serial.print(F(" SOC:"));
  Serial.print(BMS_IC[current_ic].stat.stat_codes[0]*0.0001*30,4);
  Serial.print(F(","));
 }
 Serial.println("\n");
}
 \brief Function to check the MUX fail bit in the Status Register
  @return void
*************************************************************************
void check_mux_fail(void)
```

```
{
 int8_t error = 0;
 for (int ic = 0; ic<TOTAL_IC; ic++)
  {
   Serial.print(" IC ");
   Serial.println(ic+1,DEC);
   if (BMS_IC[ic].stat.mux_fail[0] != 0) error++;

   if (error==0) Serial.println(F("Mux Test: PASS \n"));
   else Serial.println(F("Mux Test: FAIL \n"));
  }
}
 \brief Prints Errors Detected during self test
  @return void
*****************************************************************
void print_selftest_errors(uint8_t adc_reg ,int8_t error)
{
 if(adc_reg==1)
 {
  Serial.println("Cell ");
  }
 else if(adc_reg==2)
 {
  Serial.println("Aux ");
  }
 else if(adc_reg==3)
 {
  Serial.println("Stat ");
  }
 Serial.print(error, DEC);
 Serial.println(F(" : errors detected in Digital Filter and Memory
\n"));
}
 \brief Prints the output of the ADC overlap test
  @return void
*****************************************************************
void print_overlap_results(int8_t error)
{
 if (error==0) Serial.println(F("Overlap Test: PASS \n"));
 else Serial.println(F("Overlap Test: FAIL \n"));
}
 \brief Prints Errors Detected during Digital Redundancy test
  @return void
*****************************************************************
void print_digital_redundancy_errors(uint8_t adc_reg ,int8_t error)
{
 if(adc_reg==2)
 {
  Serial.println("Aux ");
  }
 else if(adc_reg==3)
 {
  Serial.println("Stat ");
```

```
  }
 Serial.print(error, DEC);
 Serial.println(F(" : errors detected in Measurement \n"));
}
 \brief Prints Open wire test results to the serial port
 ************************************************************************
********
void print_open_wires(void)
{
 for (int current_ic =0 ; current_ic < TOTAL_IC; current_ic++)
 {
  if (BMS_IC[current_ic].system_open_wire == 65535)
  {
   Serial.print("No Opens Detected on IC ");
   Serial.print(current_ic+1, DEC);
   Serial.println();
  }
  else
  {
   Serial.print(F("There is an open wire on IC "));
   Serial.print(current_ic + 1,DEC);
   Serial.print(F(" Channel: "));
   Serial.println(BMS_IC[current_ic].system_open_wire);
  }
 }
 Serial.println("\n");
}
  \brief Function to print the number of PEC Errors
  @return void
 ************************************************************************
********
void print_pec_error_count(void)
{
 for (int current_ic=0; current_ic<TOTAL_IC; current_ic++)
 {
   Serial.println("");
   Serial.print(BMS_IC[current_ic].crc_count.pec_count,DEC);
   Serial.print(F(" : PEC Errors Detected on IC"));
   Serial.println(current_ic+1,DEC);
 }
 Serial.println("\n");
}
 \brief Function to select the S pin for discharge
 @return void
 ******************************************************
int8_t select_s_pin(void)
{
 int8_t read_s_pin=0;

 Serial.print(F("Please enter the Spin number: "));
 read_s_pin = (int8_t)read_int();
 Serial.println(read_s_pin);
 return(read_s_pin);
```

```
}
 \brief prints data which is written on PWM register onto the serial
port
 @return void
 ************************************************************************
********
void print_wrpwm(void)
{
 int pwm_pec;
 Serial.println(F("Written PWM Configuration: "));
 for (uint8_t current_ic = 0; current_ic<TOTAL_IC; current_ic++)
 {
  Serial.print(F("IC "));
  Serial.print(current_ic+1,DEC);
  for(int i = 0; i < 6; i++)
  {
   Serial.print(F(", 0x"));
   serial_print_hex(BMS_IC[current_ic].pwm.tx_data[i]);
  }
  Serial.print(F(", Calculated PEC: 0x"));
  pwm_pec = pec15_calc(6,&BMS_IC[current_ic].pwm.tx_data[0]);
  serial_print_hex((uint8_t)(pwm_pec>>8));
  Serial.print(F(", 0x"));
  serial_print_hex((uint8_t)(pwm_pec));
  Serial.println("\n");
 }
}
 \brief Prints received data from PWM register onto the serial port
 @return void
 ************************************************************************
********
void print_rxpwm(void)
{
 Serial.println(F("Received pwm Configuration:"));
 for (uint8_t current_ic=0; current_ic<TOTAL_IC; current_ic++)
 {
  Serial.print(F("IC "));
  Serial.print(current_ic+1,DEC);
  for(int i = 0; i < 6; i++)
  {
   Serial.print(F(", 0x"));
   serial_print_hex(BMS_IC[current_ic].pwm.rx_data[i]);
  }
  Serial.print(F(", Received PEC: 0x"));
  serial_print_hex(BMS_IC[current_ic].pwm.rx_data[6]);
  Serial.print(F(", 0x"));
  serial_print_hex(BMS_IC[current_ic].pwm.rx_data[7]);
  Serial.println("\n");
 }
}
 \brief prints data which is written on S Control register
 @return void
```

```
 ***********************************************************************
********
void print_wrsctrl(void)
{
  int sctrl_pec;
 Serial.println(F("Written Data in Sctrl register: "));
 for (int current_ic = 0; current_ic<TOTAL_IC; current_ic++)
 {
  Serial.print(F(" IC: "));
  Serial.print(current_ic+1,DEC);
  Serial.print(F(" Sctrl register group:"));
  for(int i = 0; i < 6; i++)
  {
   Serial.print(F(", 0x"));
   serial_print_hex(BMS_IC[current_ic].sctrl.tx_data[i]);
  }

  Serial.print(F(", Calculated PEC: 0x"));
  sctrl_pec = pec15_calc(6,&BMS_IC[current_ic].sctrl.tx_data[0]);
  serial_print_hex((uint8_t)(sctrl_pec>>8));
  Serial.print(F(", 0x"));
  serial_print_hex((uint8_t)(sctrl_pec));
  Serial.println("\n");
 }
}
 \brief prints data which is read back from S Control register
 @return void
 ***********************************************************************
********
void print_rxsctrl(void)
{
  Serial.println(F("Received Data:"));
 for (int current_ic=0; current_ic<TOTAL_IC; current_ic++)
 {
  Serial.print(F(" IC "));
  Serial.print(current_ic+1,DEC);

  for(int i = 0; i < 6; i++)
  {
  Serial.print(F(", 0x"));
  serial_print_hex(BMS_IC[current_ic].sctrl.rx_data[i]);
  }

  Serial.print(F(", Received PEC: 0x"));
  serial_print_hex(BMS_IC[current_ic].sctrl.rx_data[6]);
  Serial.print(F(", 0x"));
  serial_print_hex(BMS_IC[current_ic].sctrl.rx_data[7]);
  Serial.println("\n");
 }
}
 \brief Prints data which is written on PWM
 the serial port
  @return void
```

```
 ***************************************************************************
********
void print_wrpsb(uint8_t type)
{
 int psb_pec=0;
 Serial.println(F(" PWM
 for (int current_ic = 0; current_ic<TOTAL_IC; current_ic++)
 {
   if(type == 1)
   {
    Serial.print(F(" IC: "));
    Serial.println(current_ic+1,DEC);
    Serial.print(F(" 0x"));
    serial_print_hex(BMS_IC[current_ic].pwmb.tx_data[0]);
    Serial.print(F(", 0x"));
    serial_print_hex(BMS_IC[current_ic].pwmb.tx_data[1]);
    Serial.print(F(", 0x"));
    serial_print_hex(BMS_IC[current_ic].pwmb.tx_data[2]);

    Serial.print(F(", Calculated PEC: 0x"));
    psb_pec = pec15_calc(6,&BMS_IC[current_ic].pwmb.tx_data[0]);
    serial_print_hex((uint8_t)(psb_pec>>8));
    Serial.print(F(", 0x"));
    serial_print_hex((uint8_t)(psb_pec));
    Serial.println("\n");
   }
   else if(type == 2)
   {
    Serial.print(F(" IC: "));
    Serial.println(current_ic+1,DEC);
    Serial.print(F(" 0x"));
    serial_print_hex(BMS_IC[current_ic].sctrlb.tx_data[3]);
    Serial.print(F(", 0x"));
    serial_print_hex(BMS_IC[current_ic].sctrlb.tx_data[4]);
    Serial.print(F(", 0x"));
    serial_print_hex(BMS_IC[current_ic].sctrlb.tx_data[5]);

    Serial.print(F(", Calculated PEC: 0x"));
    psb_pec = pec15_calc(6,&BMS_IC[current_ic].sctrlb.tx_data[0]);
    serial_print_hex((uint8_t)(psb_pec>>8));
    Serial.print(F(", 0x"));
    serial_print_hex((uint8_t)(psb_pec));
    Serial.println("\n");
   }
 }
}
 \brief Prints received data from PWM
  onto the serial port
  @return void
 ***************************************************************************
********
void print_rxpsb(uint8_t type)
{
```

```
 Serial.println(F(" PWM
 if(type == 1)
 {
   for (int current_ic=0; current_ic<TOTAL_IC; current_ic++)
   {
    Serial.print(F(" IC: "));
    Serial.println(current_ic+1,DEC);
    Serial.print(F(" 0x"));
    serial_print_hex(BMS_IC[current_ic].pwmb.rx_data[0]);
    Serial.print(F(", 0x"));
    serial_print_hex(BMS_IC[current_ic].pwmb.rx_data[1]);
    Serial.print(F(", 0x"));
    serial_print_hex(BMS_IC[current_ic].pwmb.rx_data[2]);

    Serial.print(F(", Received PEC: 0x"));
    serial_print_hex(BMS_IC[current_ic].pwmb.rx_data[6]);
    Serial.print(F(", 0x"));
    serial_print_hex(BMS_IC[current_ic].pwmb.rx_data[7]);
    Serial.println("\n");
   }
 }
  else if(type == 2)
 {
   for (int current_ic = 0; current_ic<TOTAL_IC; current_ic++)
   {
    Serial.print(F(" IC: "));
    Serial.println(current_ic+1,DEC);
    Serial.print(F(" 0x"));
    serial_print_hex(BMS_IC[current_ic].sctrlb.rx_data[3]);
    Serial.print(F(", 0x"));
    serial_print_hex(BMS_IC[current_ic].sctrlb.rx_data[4]);
    Serial.print(F(", 0x"));
    serial_print_hex(BMS_IC[current_ic].sctrlb.rx_data[5]);

    Serial.print(F(", Received PEC: 0x"));
    serial_print_hex(BMS_IC[current_ic].sctrlb.rx_data[6]);
    Serial.print(F(", 0x"));
    serial_print_hex(BMS_IC[current_ic].sctrlb.rx_data[7]);
    Serial.println("\n");
   }
 }
}
 \brief prints data which is written on COMM register onto the serial
port
 @return void
 ***********************************************************************
********
void print_wrcomm(void)
{
 int comm_pec;
 Serial.println(F("Written Data in COMM Register: "));
 for (int current_ic = 0; current_ic<TOTAL_IC; current_ic++)
 {
```

```
  Serial.print(F(" IC- "));
  Serial.print(current_ic+1,DEC);

  for(int i = 0; i < 6; i++)
  {
   Serial.print(F(", 0x"));
   serial_print_hex(BMS_IC[current_ic].com.tx_data[i]);
  }
  Serial.print(F(", Calculated PEC: 0x"));
  comm_pec = pec15_calc(6,&BMS_IC[current_ic].com.tx_data[0]);
  serial_print_hex((uint8_t)(comm_pec>>8));
  Serial.print(F(", 0x"));
  serial_print_hex((uint8_t)(comm_pec));
  Serial.println("\n");
 }
}
 \brief Prints received data from COMM register onto the serial port
 @return void
 ************************************************************************
********
void print_rxcomm(void)
{
  Serial.println(F("Received Data in COMM register:"));
 for (int current_ic=0; current_ic<TOTAL_IC; current_ic++)
 {
  Serial.print(F(" IC- "));
  Serial.print(current_ic+1,DEC);

  for(int i = 0; i < 6; i++)
  {
   Serial.print(F(", 0x"));
   serial_print_hex(BMS_IC[current_ic].com.rx_data[i]);
  }
  Serial.print(F(", Received PEC: 0x"));
  serial_print_hex(BMS_IC[current_ic].com.rx_data[6]);
  Serial.print(F(", 0x"));
  serial_print_hex(BMS_IC[current_ic].com.rx_data[7]);
  Serial.println("\n");
 }
}
 \brief Function to check the Mute bit in the Configuration Register
  @return void
************************************************************************
void check_mute_bit(void)
{
 for (int current_ic = 0 ; current_ic < TOTAL_IC; current_ic++)
 {
  Serial.print(F(" Mute bit in Configuration Register B: 0x"));
  serial_print_hex((BMS_IC[current_ic].configb.rx_data[1])&(0x80));
  Serial.println("\n");
 }
}
 \brief Function to print the Conversion Time
```

```
 @return void
 ************************************************************************
********
void print_conv_time(uint32_t conv_time)
{
 uint16_t m_factor=1000;
 Serial.print(F("Conversion completed in:"));
 Serial.print(((float)conv_time
 Serial.println(F("ms \n"));
}
 \brief Function to check error flag and print PEC error message
 @return void
 ************************************************************************
********
char hex_digits[16]=
{
 '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D',
'E', 'F'
};
 \brief Global Variables
char hex_to_byte_buffer[5]=
{
 '0', 'x', '0', '0', '\0'
};
 \brief Buffer for ASCII hex to byte conversion
char byte_to_hex_buffer[3]=
{
 '\0','\0','\0'
};
 \brief Read 2 hex characters from the serial buffer and convert them
to a byte
 @return char data Read Data
char read_hex(void)
{
 byte data;
 hex_to_byte_buffer[2]=get_char();
 hex_to_byte_buffer[3]=get_char();
 get_char();
 get_char();
 data = strtol(hex_to_byte_buffer, NULL, 0);
 return(data);
}
 \brief Read a command from the serial port
 @return char
char get_char(void)
{

 while (Serial.available() <= 0);
 return(Serial.read());
}
```