RAA489206 R-BMS B Sample Code Specification

Introduction

This manual describes the sample code for a battery management system intended for use with the RAA489206 Industrial Battery Management IC, and it contains a specification of the system's general principle of operation, states and modes, measured data, and configuration parameters.

Contents

1. Software Package Overview

The sample code for a Battery Management System (BMS) contains applications that provide battery protection and safety, cell balancing, capacity control and communication. They connect to the Application Programming Interface (API) available for the RAA489206 Battery Management IC (BMIC). This sample code is designed to assist users in designing their own battery applications and to be used as a starting point for evaluation and design of the products. It should NOT be directly used in an end product. The basic functions included in the sample code are:

- Current, voltage, and temperature measurement
- Undercurrent, overcurrent, undervoltage, overvoltage, under-temperature, and over-temperature protections
- System self-diagnostic and memory test
- State machine
- System state, battery status, faults, failures, and measured data reporting (application)
- Simple Cell Balancing (application)
- Simple Remaining Capacity Control (application)

1.1 Assumptions and Advisory Notes

The following is assumed for the user:

- A basic understanding of microcontrollers, embedded systems hardware, battery management systems, and Li-based battery cells.
- Prior experience working with Integrated Development Environments (IDEs) such as e^{2}studio and Flexible Software Package (FSP).
- Also, a familiarity with the following:
  - Renesas RA family ARM Cortex-M microcontrollers.
  - Renesas Industrial BMIC - RAA489206.
  - *Industrial Battery Front End API Software Manual*.

2. System Outline

The sample code is designed for a BMS that consists of separate BMIC and MCU. Figure 1 shows a general block diagram of an R-BMS B system. The MCU runs all the control algorithms and is in full control of the BMIC. However, the BMIC has built-in automatic management and safety functions such as automatically turning off the charge and discharge FETs (CFET and DFET) after triggering either undervoltage or overvoltage, undercurrent or overcurrent, or under-temperature or over-temperature protections.

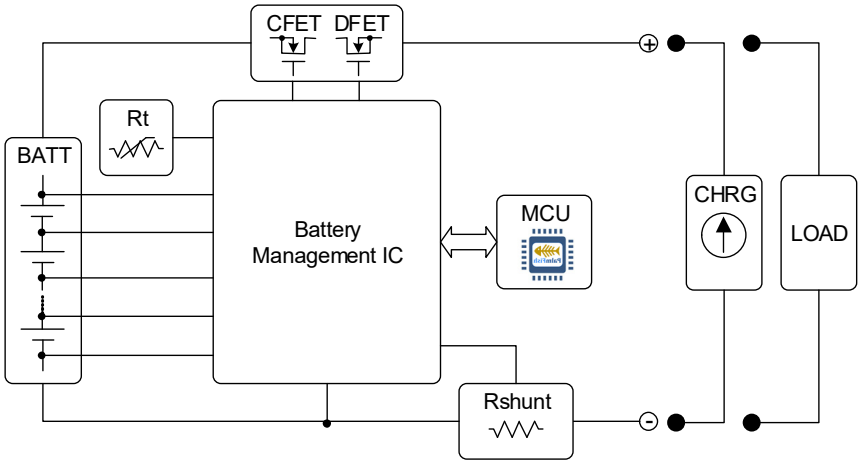

Figure 1. General Block Diagram of R-BMS B for RAA489206

Figure 2 shows the general structure of the R-BMS B Software. It consists of three layers in terms of software architecture:

- Layer 3 – Application Layer: Applications with a high level of abstraction that control the system and are described in the current document.
- Layer 2 – BMIC Abstraction Layer (Middleware)[1]: The Application Programming Interface (API) for BMICs and its implementation for RAA489206.
- Layer 1 – Hardware Abstraction Layer (HAL): The Renesas Flexible Software Package (FSP)[2] available for the Renesas Advanced MCU in use.

Every two adjacent layers of the software architecture are connected using an API that provides modularity and flexibility: Layers 1 and 2 use the FSP API, and Layers 2 and 3 use the BMIC API (for more information, refer to the *Industrial Battery API Software Manual*). As the BMIC Abstraction Layer handles the device specifics, most of the BMS sample code is applicable to other BMICs in the product family. However, a change of the BMIC and API implementation would require changes in the application layer.

1. The API for BMICs and its implementation for RAA489206 are available on the respective product pages.
2. The Renesas FSP is available for the Renesas RA MCUs and is a part of e^{2}studio IDE.

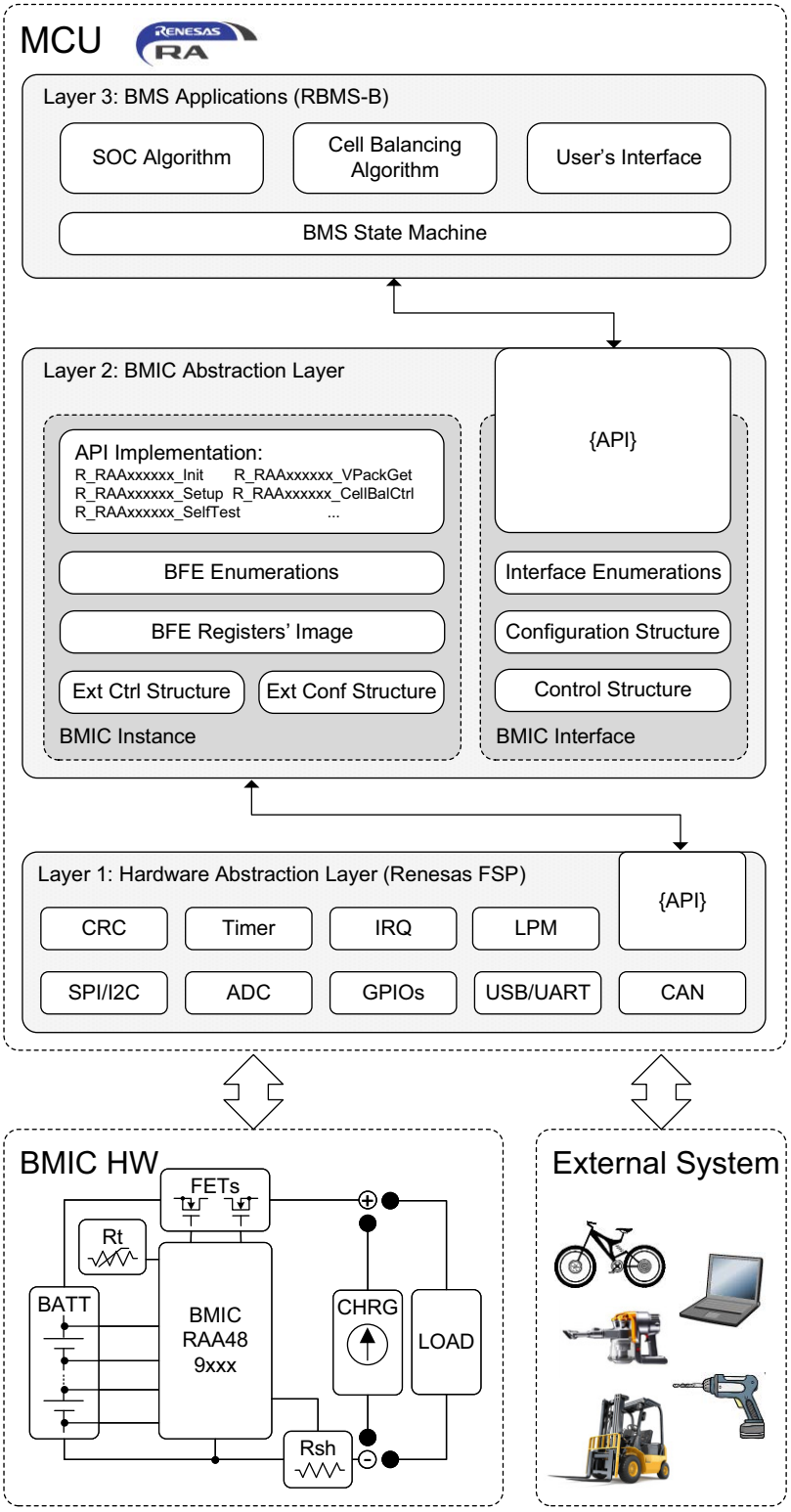

Figure 2. General Structure of the R-BMS B Software for RAA489206

2.1 Supported Devices

The sample code supports various combinations of microcontrollers from the ARM Cortex-M Renesas RA Family and the Renesas Industrial BMICs with available API implementations. However, the current version of the code is developed and tested with the following devices:

2.1.1 Battery Management IC

The supported BMIC is the RAA489206 (RTKA489206DK0000BU Evaluation Board), and it features the following: 4-16S, pack voltage, pack current, cell voltage, external temperature and internal temperature, high/low side FET driver, self-test functions, low-current consumption, and load detection.

2.1.2 MCU

The supported MCU is RA4E1 (ISO_DONGLE_EV1Z REV D Isolated Communication Dongle), and it features the following: 48-pin, Arm Cortex-M33, 128KB SRAM, and 256KB Code flash.

2.2 Safety Functions and Protections

The R-BMS B RAA489206 sample code provides safety functions on different hardware and software levels with a redundancy of the protections. The RAA489206 BMIC has internal registers containing voltage, current, and temperature thresholds. These registers are set during device initialization in the BMIC Abstraction Layer (Layer 2). They are compared to the measured values by the hardware and can set fault flags and override the FET control to bring the system into a safe state. When the BMIC detects a fault condition, the information travels through the HAL drivers (Layer 1) and the BMIC Abstraction Layer (Layer 2) to reach the BMS Application Layer (Layer 3) where the system decides how to handle the event. The same physical value (voltage, current, temperature) is also compared by the MCU in the BMS Application Layer (Layer 3) so that the system has two separate mechanisms that can detect the same fault condition and provide additional safety. Some tests occur only in the BMIC hardware (such as the short-circuit detection) or in the software (such as disabling the pack because of too low/high cell voltage or too high cell temperature). To verify the proper operation of these safety mechanisms, the system periodically runs a self-test of the BMIC, an open-wire test, and a memory test that validates the content of the configuration registers. All these tests are provided by the BMIC Abstraction Layer (Layer 2) and partially run in the software or directly in the BMIC hardware.

The control of the CFET and DFET is distributed between the BMIC and the MCU. The ON/OFF states of the FETs are controlled by the system state machine (such as in Sleep Mode, both FET are OFF). However, when the BMIC first detects a fault condition, it can directly turn off any or both of the FETs before any reaction of the software. The same action occurs if the MCU sticks and does not communicate with the BMIC for some time. To ensure that the MCU reads the actual CFET and DFET states, the MCU reads the BMIC GPIO pins which are configured for FET control[3]. Using the same approach, the MCU can also override the FET control and force turn off the CFET and DFET in critical situations such as Permanent Failure or Power-Down Mode.

3. For more information, refer to the *RAA489206 Datasheet*.

3. Operation States and Modes

There are three different groups of states and modes related to the system:

- States and modes of the R-BMS B Battery Management System – They outline conditions of the BMS software and are defined there. A state executes its function and moves to the next state or mode, while a mode loops inside until a valid condition for transition appears.
- States of the Renesas Advanced Family MCU – They are hardware states of the MCU that are described in *RA4E1 Group, User's Manual: Hardware*. There is a state of the MCU corresponding to a particular state or mode of the BMS (Table 1).
- States and modes of the Battery Management IC – They are specific to the used BMIC and are managed by the BMIC instance. The states and modes of the BMIC do not correspond directly to those of the BMS.

Figure 3 shows the general flowchart of R-BMS B for the RAA489206 and depicts the states, modes, and the possible transitions between them.

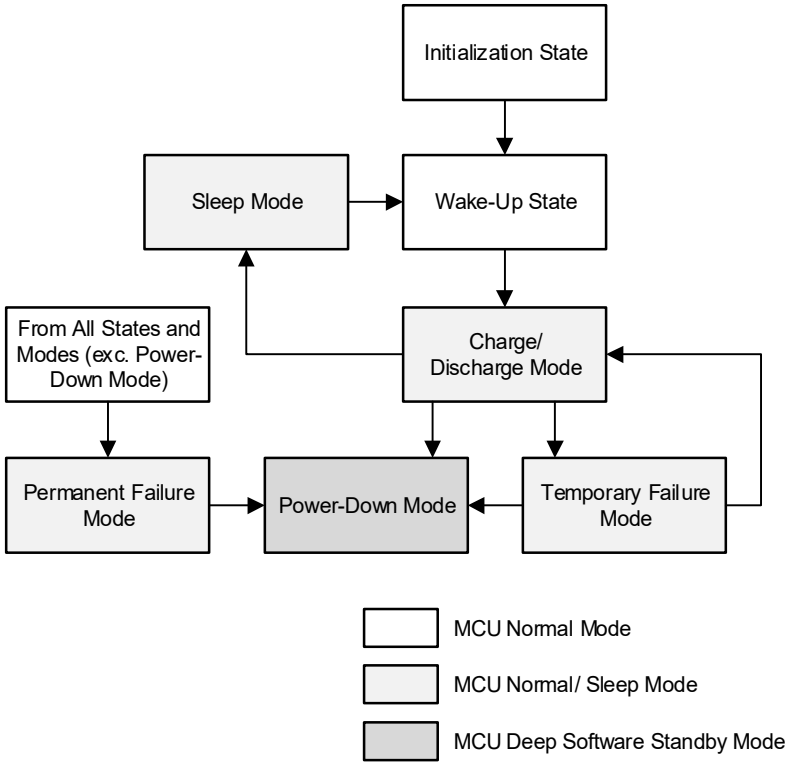

Figure 3. General Flowchart of R-BMS B for RAA489206

The BMS flowchart incorporates three combinations of MCU modes:

- MCU Normal mode – This is the highest performance MCU mode where the CPU and all initialized peripheral functions are active resulting also in the highest power consumption. It is used only in the BMS states where the system does not loop.
- MCU Normal/Sleep mode – The system is triggered every 125ms by a periodic interrupt timer to perform measurements, calculations, communications, and others in MCU Normal mode; next, the MCU enters Sleep mode when the CPU stops but the content of its internal registers is retained, and the peripheral functions continue to operate. In this case, the MCU provides a balance between power-saving and performance.
- MCU Deep Software Standby mode – This is the lowest power consumption mode where the CPU and all the peripheral functions and oscillators stop. This mode prevents excessive discharge of the battery. It is possible to cancel Deep Software Standby mode only by triggering a hardware reset. In this state, the MCU retains the levels of its outputs.

Table 1 shows the relations between the BMS, MCU, BMIC and charge and discharge FET states (and modes). In most cases, the CFET and DFET states are fixed; however, in Wake-Up state, the FETs retain their condition from the previous Initialization state, Sleep mode, or Temporary Failure mode. In Temporary Failure Mode, the CFET and DFET are turned on or off depending on the particular fault flag that is set. In some modes, the FET states also depend on the battery status (Table 12).

Table 1. MCU, BMIC and FET States in the Different Modes and States

BMS Mode/State	CFET	DFET	MCU Mode	BMIC Mode
Initialization State	OFF	OFF	MCU Normal mode	BMIC Idle mode
Wake-Up State	Inherit from previous state or mode.	Inherit from previous state or mode.	MCU Normal mode	BMIC Idle mode
Charge/Discharge Mode	OFF (*FULLY_CHARGED* = *true* & *DISCHARGING* = *false*) ON (In all other cases)	ON	MCU Normal mode/ Sleep mode	BMIC Idle/ Scan mode
Sleep Mode	OFF	OFF	MCU Normal mode/ Sleep mode	BMIC Idle/ Scan / Low Power mode
Temporary Failure Mode	Faults union (Table 14): Any bit from 0 to 11 is set and *DISCHARGING* = *false* or any bit from 24 to 31 is set: OFF All bits from 0 to 11 are clear or *DISCHARGING* = *true*[1] and all bits from 24 to 31 are clear: ON	Faults union (Table 14): Any bit from 12 to 23 is set and *CHARGING* = *false* or any bit from 24 to 31 is set: OFF All bits from 12 to 23 are clear or *CHARGING* = *true*[2] and all bits from 24 to 31 are clear: ON	MCU Normal mode/ Sleep mode	BMIC Idle/ Scan mode
Permanent Failure Mode	OFF	OFF	MCU Normal mode/ Sleep mode	BMIC Idle/ Scan / Low Power mode
Power-Down Mode	OFF	OFF	MCU Deep Software Standby mode	BMIC Ship mode

1. The system turns back on the CFET, if it detects the discharging current even when a charge related fault remains present (such as the cell overvoltage fault).
2. The system turns back on the DFET, if it detects the charging current even when a discharge related fault remains present (such as cell the undervoltage fault).

3.1 Initialization State

The system enters the Initialization state when the system powers on or the MCU resets. In this state, the MCU initializes its resources and peripherals, checks for incorrect system parameters (zero or out of range), and opens the BMIC interface (initializes the BMIC). Figure 4 shows the state flowchart. Table 2 gives the conditions for the transition from this state. The Initialization state is a transitional state, and the system does not loop or stick into it. Both FETs are OFF and the MCU is running in Normal mode. In the Initialization state, the system runs an initial memory test provided by the API to validate the default (expected) values of the BMIC registers after reset. Next, the system configures the BMIC, and it enables a communication timeout to disable the battery pack if the MCU stops to communicating.

The status of the LEDs on the MCU board is as follows: GREEN (D3) - OFF, RED (D4) - OFF.

The Initialization state uses the following API functions, and the actual implementation is provided by the BMIC instance:

- *p_initialize* – Initializes the BMIC interface by enabling and configuring the necessary peripheral modules of the MCU, resets, and identifies the BMIC and other device-specific actions.
- *p_memoryCheck* – Runs memory tests inside the BMIC to verify the default values of the registers (*BFE_CHECK_DEF_VALS*).
- *p_setup* – Configures the BMIC device by writing into all the configuration registers.
- *p_wdControl* – Turns on the communication timeout.

Table 2. Conditions for Transition from Initialization State

From	To	Condition
Initialization State	Wake-Up State	No errors were returned by the FSP and BSP functions, and no incorrect system parameter is detected.
	Permanent Failure Mode	Any unrecoverable error is returned by the FSP and BSP functions, or an incorrect system parameter is detected.

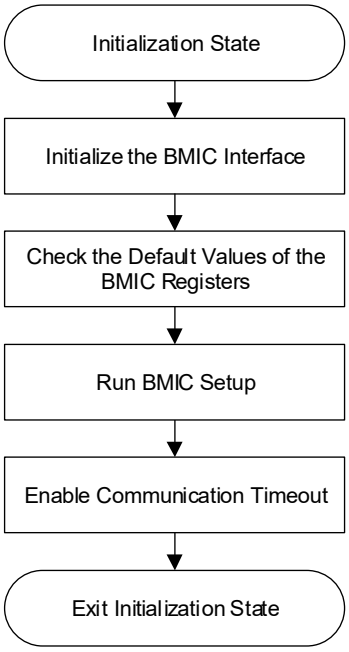

Figure 4. Initialization State Flowchart

3.2 Wake-Up State

The system enters Wake-Up State after initialization or exiting Sleep Mode. In Wake-Up State, the system puts the BMIC in Idle mode. Next, it verifies the content of the configuration registers and runs a complete self diagnostic of the BMIC. After that the system determines which mode to enter (Figure 5). Table 3 gives the conditions for transition from this state. The condition of both FETs remains unchanged and the MCU is running in MCU Normal mode.

The status of the LEDs on the MCU board is as follows: GREEN (D3) - ON, RED (D4) - ON.

The Wake-Up State uses the following API functions (and the actual implementation of these API functions are provided by the BMIC instance):

- *p_modeSet* – Forces the BMIC to enter Idle mode (*BFE_MODE_IDLE*).
- *p_memoryCheck* – Runs memory tests to verify the configuration registers (*BFE_CHECK_CONF_REGS*).
- *p_selfDiagnostic* – Runs a full self-diagnostic test for the BMIC (*BFE_FULL_TEST*).
- *p_faultsAllRead* – Reads the faults when the self-diagnostic test fails.

Table 3. Conditions for Transition from Wake-Up State

From	To	Condition
Wake-Up State	Permanent Failure Mode	Any of the following conditions are true: ▪ Any unrecoverable error is returned by the FSP and BSP functions. ▪ The *failures* member of the BMS Failures union (Table 15) is non-zero.
	Charge/Discharge Mode	None of the Permanent Failure mode conditions are true.

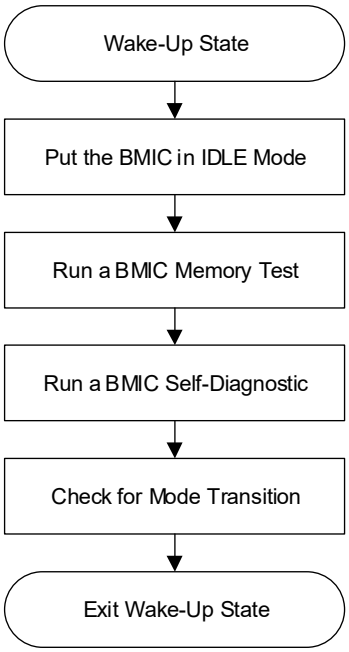

Figure 5. Wake-Up State Flowchart

3.3 Charge/Discharge Mode

After exiting the Wake-Up mode or Temporary Failure mode, the BMIC enters the Charge/Discharge mode. Table 4 shows the conditions for exiting to another mode. In the Charge/Discharge mode the DFET is always ON, and the CFET is controlled depending on the system status and detection of a discharge current (Table 1). The MCU is in MCU Normal mode during the steps in the flowchart; it goes to the MCU Sleep mode to save power until the next periodic timer interrupt appears (every 125ms).

- Figure 7 shows a flowchart depicting the main loop in this mode.
- Figure 6 shows the timeline of the main loop.

After registering a periodic timer interrupt, the MCU exits Sleep mode and re-enables the Continuous Scan mode if it has been stopped in a previous mode or by a fault detection of the BMIC.

Next, the MCU reads the pack current timer value. If it has incremented after the last read and the new measurement data is available, the MCU reads the pack current, pack voltage, cell voltages and all temperatures. The acquired values are tested for software fault or failure conditions. The system checks the BMIC for any detected hardware fault or failure conditions. Most of these tests use the ADC, and the detection usually takes 125ms or more (in case of active cell balancing). The system detects most of the faults and failures when they appear repeatedly. *Note:* Critical events like short-circuits are managed directly in the BMIC without any time delay added by the system.

After testing for faults, the system runs the cell balancing application and recalculates the remaining capacity. Next, it reads the actual state of the CFET and DFET and sets their required state based on the measurements and system state (Table 1). For every normal_mem_test_int cycle, the system runs a memory test for the configuration registers of the BMIC. The system checks if there are valid conditions to change the current mode, and if there are no valid conditions, it enters the MCU Sleep mode until the next timer interrupt. However, before entering the MCU Sleep mode, the system updates the user interface.

When the system detects any fault or failure event, the MCU increments a dedicated counter (and when necessary, clears the fault bit in the BMIC). If the event is triggered again and in a row for *voltage_event_detection_time*, *current_event_detection_time*, or *temp_event_detection_time*, the system registers a fault or failure. In the other case, the counters are reset. This mechanism provides a fault response

delay in case of noise or disturbances in the measured values. ***Important:*** In case of an open-wire, short-circuit, or charge pump failure, the system immediately reacts to the event.

Note: In the Charge/Discharge mode, the BMIC runs the Continuous Scan mode, triggers measurements independently from the MCU, and makes decisions to turn OFF the CFET and DFET.

The status of the LEDs on the MCU board is as follows:

- Status Charging: GREEN (D3) - blinking slow; RED (D4) - OFF.
- Status Discharging: GREEN (D3) - blinking fast; RED (D4) - OFF.
- Status Fully Charged: GREEN (D3) - ON; RED (D4) - OFF.

The Charge/Discharge mode uses the following API functions (and the actual implementation of these API functions are provided by the BMIC instance):

- p_allGet – Reads the pack current and voltage, the cell voltages, temperatures, internal voltages, and basic diagnostic.
- p_iPackGet – Reads the pack current timer value.
- p_faultsAllRead – Checks the BMIC for faults and reads all fault registers, if any are present.
- p_faultsAllClear – Clears specific faults that have a delay to check, if they appear again before changing the mode.
- p_faultsCheck – Checks if the BMIC has detected any fault.
- p_fetControl – Turns ON/OFF the DFET and CFET to control the end of charging (Table 1).
- p_memoryCheck – Runs memory test to check the configuration registers (*BFE_CHECK_CONF_REGS*).
- Application specific API functions.

Table 4. Conditions for Transitions from Charge/Discharge Mode

From	To	Condition
Charge/Discharge Mode	Power-Down Mode	*v_cell_min* ≤ *power_down_voltage* for *power_down_detection_time*
	Permanent Failure Mode	Any of the following conditions is true: Any unrecoverable error is returned by the FSP and BSP functions. The *failures* member of the BMS Failures union (Table 15) is non-zero.
	Temporary Failure Mode	The *faults* member of the Faults union (Table 14) is non-zero.
	Sleep Mode	All the following conditions are true: *DISCHARGING* = *false* for *sleep_no_current_timeout* CHARGING = false for *sleep_no_current_timeout* The *cell_balancing* member of the Cell Balancing Data Structure (Table 21) is zero.

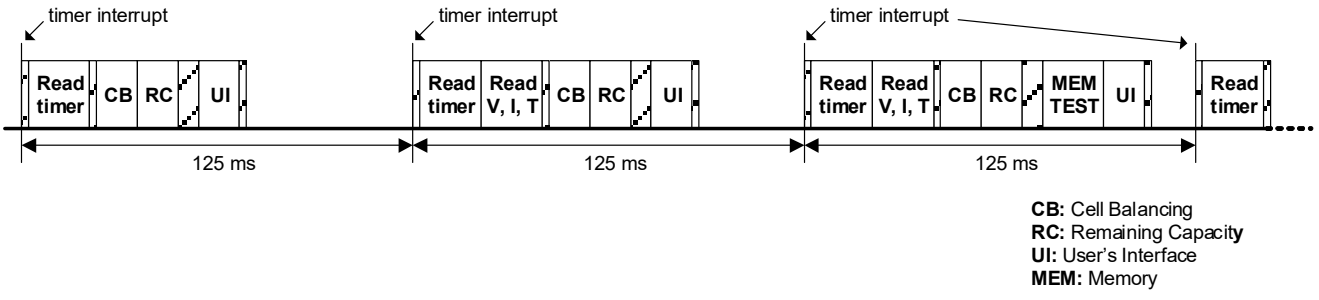

Figure 6. Timeline of the Main Loop in Charge/Discharge and Temporary Failure Mode

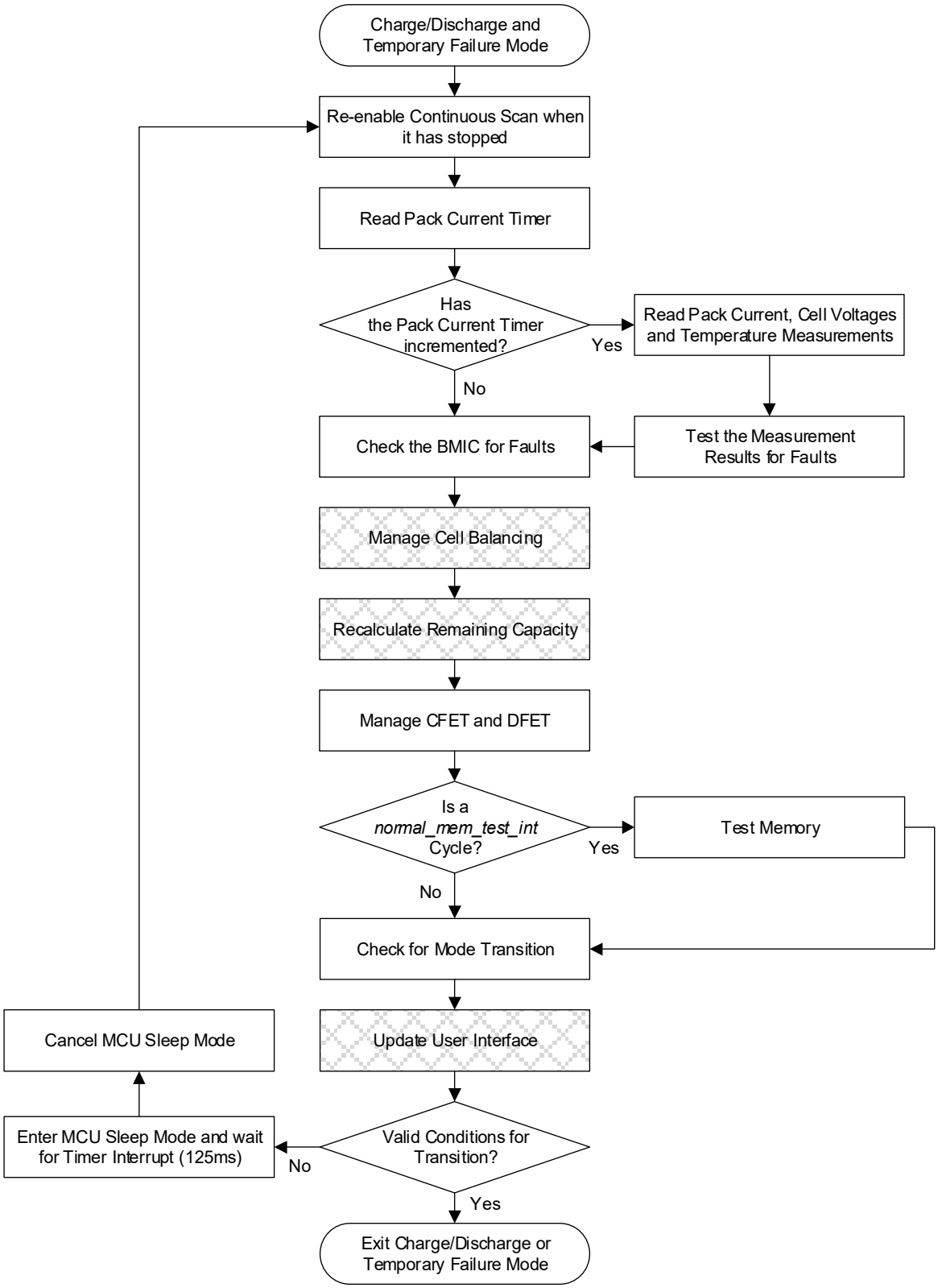

Figure 7. Charge/Discharge and Temporary Failure Mode Flowchart

3.4 Temporary Failure Mode

The system enters Temporary Failure mode from the Charge/Discharge mode when any temporary fault appears (Table 14). Depending on the particular fault, one of the FETs could remain ON (such as the CFET to allow pack charging when there is an undervoltage fault), or both FETs could be OFF (such as with charge and discharge over-temperature fault where the pack must cool down before any further use). The system runs through the same loop as in the Charge/Discharge modes (see Figure 7), and the MCU is in MCU Normal mode or MCU Sleep mode to save power until the next periodic timer interrupt comes (every 125ms). Table 5 shows the conditions to exit the Temporary Failure mode.

Note: In the Temporary Failure mode, the BMIC runs the Continuous Scan mode, triggers measurements independently from the MCU, and makes decisions to turn OFF the CFET and DFET.

The status of the LEDs on the MCU board is as follows: GREEN (D3) - OFF, RED (D4) - blinking slow.

The Temporary Failure mode uses the same API functions as described in Charge/Discharge Mode.

Table 5. Conditions for Transition from Temporary Failure Mode

From	To	Condition
Temporary Failure Mode	Power-Down Mode	*v_cell_min* ≤ *power_down_voltage* for *power_down_detection_time*
	Permanent Failure Mode	Any of the following conditions are true: ▪ Any unrecoverable error is returned by the FSP and BSP functions. ▪ The *failures* member of the BMS Failures union (Table 15) is non-zero.
	Charge/Discharge Mode	All fault conditions are cleared.

3.5 Sleep Mode

The system enters Sleep mode from the Charge/Discharge mode when the current is below the detection threshold and cell balancing is not active. In this mode, most of the time the BMIC is in Low Power mode, and the MCU is in Sleep mode. Therefore, both CFET and DFET are OFF to ensure no currents flow undetected to or from the battery pack. The MCU additionally overrides the FET control using the GPIOs of the BMIC and validates the OFF states. The MCU wakes up every 125ms to check if the *loop_interval_sleep* time has passed and for presence of a load or charger. The MCU goes back to Sleep mode to save power until the next periodic timer interrupt comes.

- Figure 8 shows the flowchart depicting the main loop in Sleep mode.
- Figure 9 shows the timeline of the main loop.

Every *loop_interval_sleep* second, the MCU wakes up the BMIC to measure the pack voltages and temperatures and to check for presence of a load or charger. All measurements are triggered and read in a sequence within the same loop. The acquired values are tested for fault or failure conditions. Next, the system calculates the remaining battery capacity to check if there are valid conditions to exit Sleep Mode, and if there are no valid conditions, it enters the MCU Sleep mode until the next interrupt. Table 6 shows the conditions to exit to another mode. In Sleep mode, the system runs a self-diagnostic and memory test every *sleep_diag_int* cycle to verify the correct operation of the BMIC.

The status of the LEDs on the MCU board is as follows: GREEN (D3) - ON when the MCU is awake and OFF when it is in LPM, RED (D4) - OFF.

The Sleep mode uses the following API functions (and the actual implementation of these API functions are provided by the BMIC instance):

- p_modeSet – Forces the BMIC to enter Low Power or Idle mode (*BFE_MODE_LOW_POWER*, *BFE_MODE_IDLE*).
- p_allGet – Measures the pack current and voltage, the cell voltages, temperatures, internal voltages, and basic diagnostic.
- p_faultsAllRead – Checks the BMIC for faults and reads any fault registers that are present.
- p_faultsCheck – Checks if the BMIC has detected any fault.

- p_fetControl – Turns off the DFET and CFET.
- p_memoryCheck – Runs a memory test to check the configuration registers (*BFE_CHECK_CONF_REGS*).
- p_selfDiagnostic – Runs a self-diagnostic test for the BMIC (*BFE_TEST_OW*).

Table 6. Conditions for Transition from Sleep Mode

From	To	Condition
Sleep Mode	Permanent Failure Mode	Any unrecoverable error is returned by the FSP and BSP functions.
	Wake-Up State	Any of the following conditions is true: ▪ DISCHARGING = true ▪ CHARGING = true ▪ A charger is detected. ▪ A load is detected. ▪ There is a Wake-up button interrupt. ▪ Any faults or failures indication appears.[1]

1. The system should exit Sleep mode after detecting any precondition for fault or failure even before there is an actual valid condition, which is after the first report of overvoltage by the BMIC and not waiting to appear again.

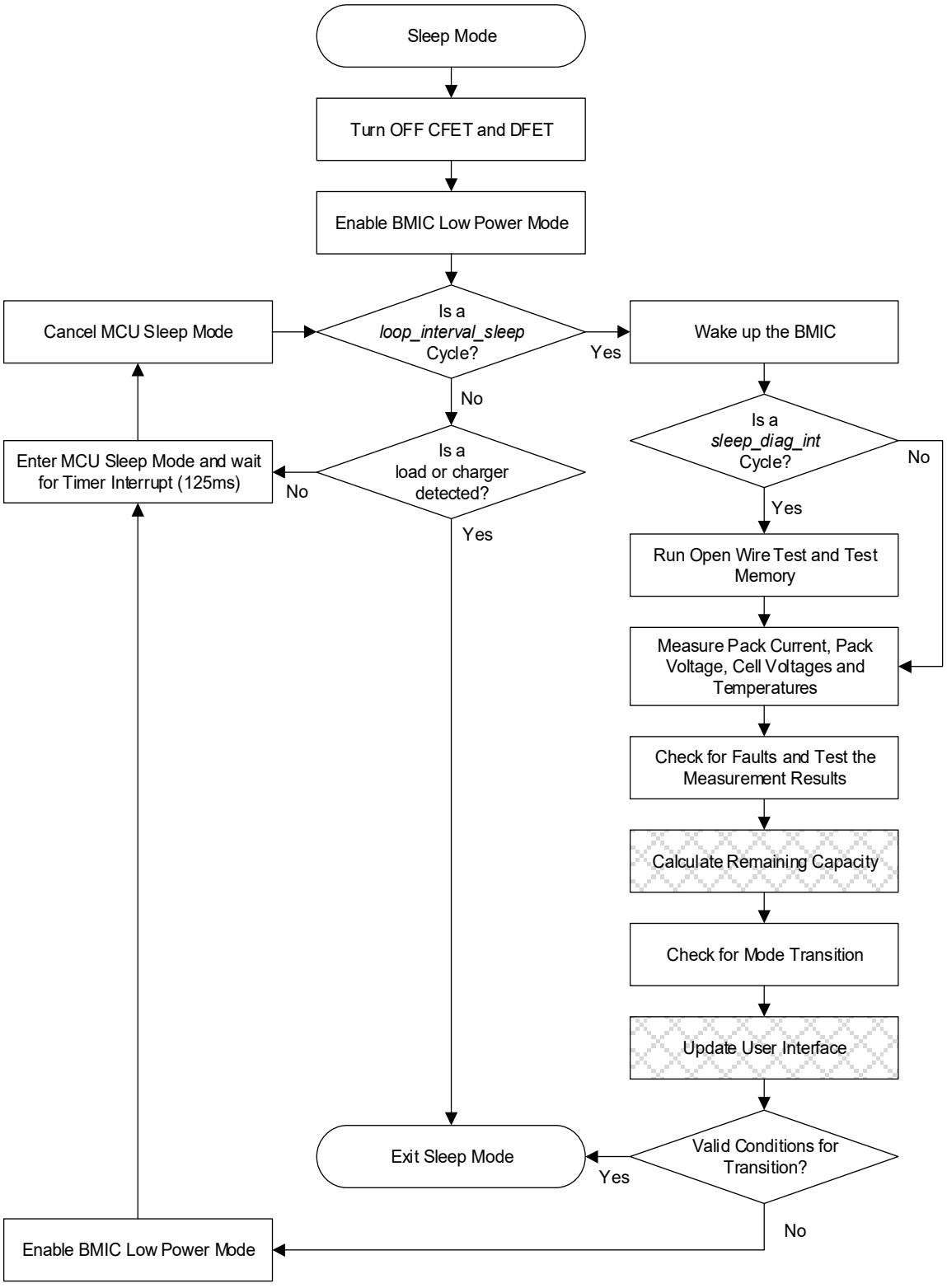

Figure 8. Sleep Mode Flowchart

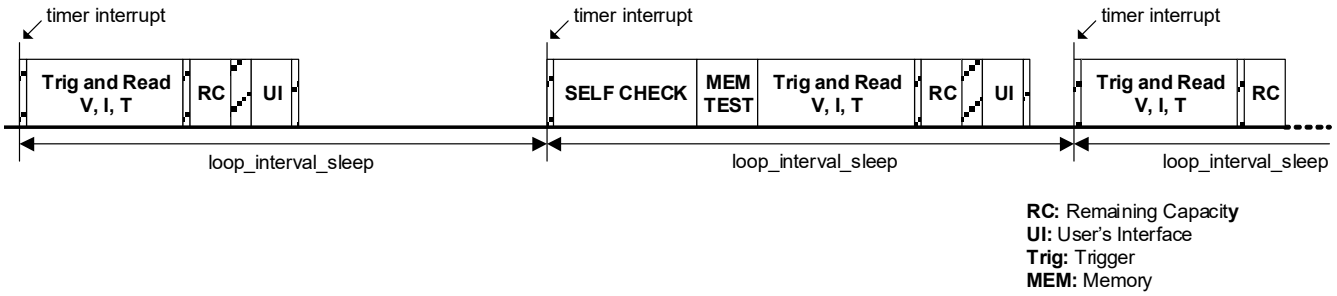

Figure 9. Timeline of the Main Loop in Sleep Mode

3.6 Permanent Failure Mode

The system enters Permanent Failure mode from any other mode (except Power-Down mode) when either an API function returns a critical error or a failure is detected (Table 15). The system cannot restore operation after it enters Permanent Failure mode. The two possibilities for a transition are to enter Power-Down mode when the battery pack is depleted or an external source triggers a hardware reset (Table 7). In this mode, most of the time the BMIC is in Low Power mode, and the MCU is in Sleep mode. Both CFET and DFET are OFF to ensure no currents flow to or from the battery pack. The MCU additionally overrides the FET control using the GPIOs of the BMIC and validates the OFF states. The MCU wakes up every 125ms to check if the *loop_interval_perm_failure* time has passed. The MCU returns to Sleep mode to save power until the next periodic timer interrupt. Figure 10 shows the flowchart depicting the main loop in the Permanent Failure mode.

For every second of *loop_interval_perm_failure*, the MCU wakes up the BMIC and measures the pack current, pack voltage, the cell voltages, and temperatures. All measurements are triggered and read in a sequence within the same loop. Then the system calculates the remaining battery capacity and updates the user interface. If the cell voltages are below the power-down voltage threshold, the system goes to Power-Down mode. Otherwise, the MCU sends a command to the BMIC to enter Low Power Mode, and it goes to Sleep mode until the next timer interrupt.

The status of the LEDs on the MCU board is as follows: GREEN (D3) - OFF, RED (D4) - ON.

The Permanent Failure mode uses the following API functions (and the actual implementation of these API functions are provided by the BMIC instance):

- p_modeSet – Forces the BMIC to enter a mode or state (BFE_MODE_LOW_POWER, BFE_MODE_IDLE).
- p_allGet – Measures the pack current, pack voltage, the cell voltages, temperatures, internal voltages, and basic diagnostics.
- p_fetControl – Turns OFF both FETs to prevent charging or discharging.

Table 7. Conditions for Transition from Permanent Failure Mode

From	To	Condition
Permanent Failure Mode	Initialization State	MCU hard reset
	Power-Down Mode	*v_cell_min* ≤ *power_down_voltage* for *power_down_detection_time*

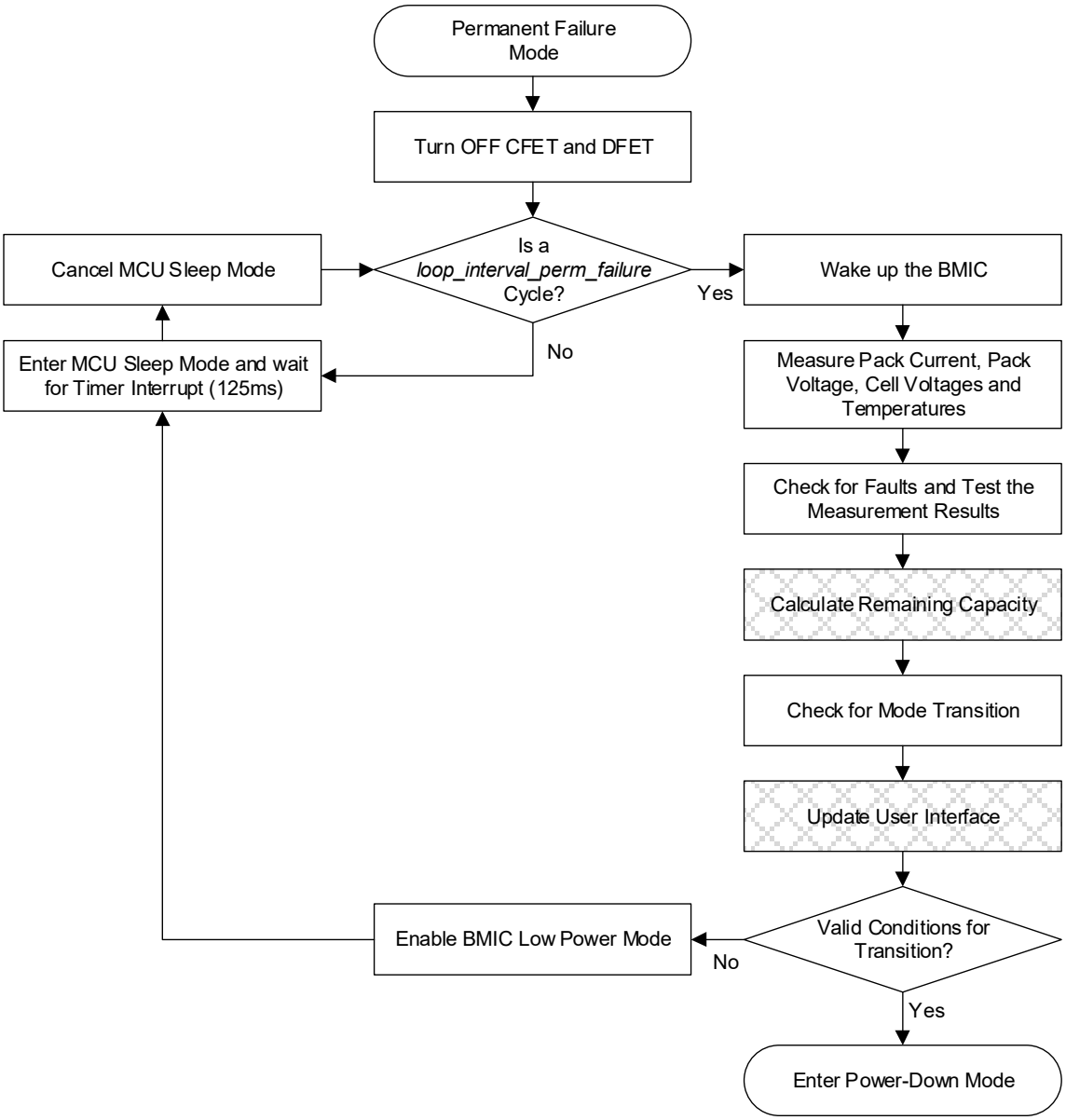

Figure 10. Permanent Failure Flowchart

3.7 Power-Down Mode

The system enters Power-Down mode from Charge/Discharge mode, Temporary Failure mode, or Permanent Failure mode when the battery pack is completely depleted (that is, any cell voltage is below the power-down voltage threshold). This mode has the lowest power consumption to prevent battery cell damage. The only possibility for exit is to externally reset the MCU (Table 8). Figure 11 shows the flowchart depicting the Power-Down mode. There are not any software loops but only transition steps. Both CFET and DFET are OFF to ensure no currents flow to or from the battery pack. The MCU additionally overrides the FET control using the GPIOs of the BMIC and validates the OFF states. Next, it sends a command to the BMIC to enter Shipping mode. The system updates the user interface, and the MCU enters the MCU Deep Software Standby mode where the CPU and all peripherals are disabled. In Power-Down mode, the system does not perform any measurements, respond to communications, nor continue to update the user interface.

The status of the LEDs on the MCU board is as follows: GREEN (D3) - OFF, RED (D4) - OFF.

The Power-Down mode uses the following API functions (and the actual implementation of these API functions are provided by the BMIC instance):

- p_fetRead – Read and check the FETs states to make sure both are OFF.
- p_fetControl – Turn OFF both FETs to prevent charging or discharging.
- p_modeSet – Forces the BMIC to enter a mode or state (BFE_MODE_SHIP).

Table 8. Conditions for Transition from Power-Down Mode

From	To	Condition
Power-Down Mode	Initialization State	MCU hard reset

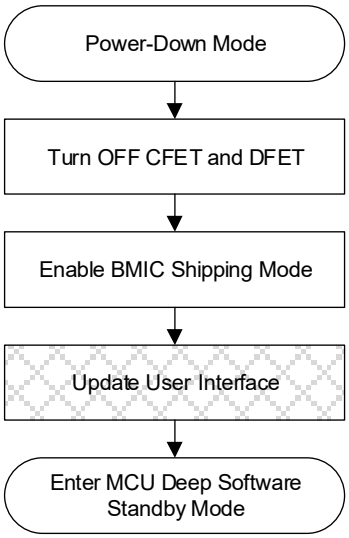

Figure 11. Power-Down Mode Flowchart

4. System Parameters

The system configuration is highly flexible and contains many constants that control the different events and actions. Table 9 shows the structure containing the general system configuration parameters. They are grouped into sections by functions. All values within this structure are hard coded in the Code Flash of the MCU. However, some of the system parameters such as hardware overvoltage threshold, discharge overcurrent, or any application specification are not part of this general configuration structure. The system parameters can be found in the settings of the particular API Implementation or in the applications (Cell Balancing, Remaining Capacity Control).

Table 9. Members of the BMS General Configuration Structure

typedef struct st_bms_general_cfg_t			
Member	**Type**	**Unit**	**Description**
Device Information			
device_name	const char	-	An array containing the system name or identifier. The size of the array is: 32
Basic Settings			
end_of_charge_voltage	const uint16_t	[mV]	The system detects full charge and opens the CFET when *v_cell_max* ≥ *end_of_charge_voltage* and *i_pack_charge* ≤ *end_of_charge_current* for *end_of_charge_timeout*
end_of_charge_current	const int16_t	[mA]	
end_of_charge_timeout	const uint16_t	[125ms]	
full_charge_release_voltage	const uint16_t	[mV]	The system releases the full charge status when the voltage drops below this limit.
power_down_voltage	const uint16_t	[mV]	If any cell voltage is lower than the *power_down_voltage* for longer than *power_down_detection_time*, the system powers down.
power_down_detection_time	const uint16_t	[125ms]	
charge_detection_current	const int32_t	[mA]	Current threshold for the system to detect charging.
discharge_detection_current	const int32_t	[mA]	Current threshold for the system to detect discharging.
sleep_no_current_timeout	const uint16_t	[125ms]	Enter Sleep Mode after this timeout when charge and discharge currents are below the detection threshold.
loop_interval_sleep	const uint8_t	[s]	The interval when the system is in Sleep mode. (Default: 10)
loop_interval_perm_failure	const uint8_t	[s]	The interval when the system is in Permanent Failure mode. (Default: 120)
normal_temp_meas_int	const uint8_t	[125ms]	Every time interval in Charge/Discharge, Full Charge or Temporary Failure mode when the system also measures the temperatures.
normal_diag_int	const uint16_t	[125ms]	Every time interval in Charge/Discharge, Full Charge or Temporary Failure mode when the system runs self-diagnostic.
normal_mem_test_int	const uint16_t	[125ms]	Every time interval in Charge/Discharge, Full Charge or Temporary Failure mode when the system runs a memory test.
sleep_diag_int	const uint16_t	[125ms]	Every time interval in Sleep mode when the system runs a self-diagnostic.
System Information			
bfe_api_in_use	const e_bms_bfe_api_t	-	The BMIC API Interface that is in use.
Control Flags			
ext_cell_temperatures	const uint8_t	-	An array indicating which external temperature inputs are monitoring the cell temperature (true – cell, false – no cell) The size of the array is: 2

Table 9. Members of the BMS General Configuration Structure (Cont.)

typedef struct st_bms_general_cfg_t			
Member	**Type**	**Unit**	**Description**
ext_other_temperatures	const uint8_t	-	An array indicating which external temperature inputs are monitoring other than cell temperature (true – other, false – nothing) The size of the array is: 2
Safety Settings (Temperature)			
cell_discharge_overtemperature	const int16_t	[0.1deg.C]	If any cell temperature during discharging is higher than this threshold for longer than *temp_event_detection_time*, the system enters Temporary Failure mode.
cell_discharge_undertemperature	const int16_t	[0.1deg.C]	If any cell temperature during discharging is lower than this threshold for longer than *temp_event_detection_time*, the system enters Temporary Failure mode.
cell_discharge_temp_hysteresis	const uint8_t	[0.1deg.C]	The discharge over/under-temperature event release hysteresis.
cell_charge_overtemperature	const int16_t	[0.1deg.C]	If any cell temperature during charging is higher than this threshold for longer than *temp_event_detection_time*, the system enters Temporary Failure mode.
cell_charge_undertemperature	const int16_t	[0.1deg.C]	If any cell temperature during charging is lower than this threshold for longer than *temp_event_detection_time*, the system enters Temporary Failure mode.
cell_charge_temp_hysteresis	const uint8_t	[0.1deg.C]	The charge over/under-temperature event release hysteresis.
other_overtemperature	const int16_t	[0.1deg.C]	If any other temperature (not used for cell monitoring) is higher than this threshold for longer than *temp_event_detection_time*, the system enters Temporary Failure mode.
other_undertemperature	const int16_t	[0.1deg.C]	If any other temperature (not used for cell monitoring) is lower than this threshold for longer than *temp_event_detection_time*, the system enters Temporary Failure mode.
other_temp_hysteresis	const uint8_t	[0.1deg.C]	The other over/under-temperature event release hysteresis.
perm_failure_overtemp	const int16_t	[0.1deg.C]	If any cell temperature is higher than this threshold for longer than *temp_event_detection_time*, the system enters Permanent Failure mode.
temp_event_detection_time	const uint16_t	[125ms]	The temperature event is detected, if the condition is still present after this time.
Important! The temperature-related HW thresholds are available in the Layer 2 configuration data.			
Safety Settings (Current)			
charge_overcurrent	const int32_t	[mA]	If the charge current is higher than this threshold for longer than *charge_overcurrent_detection_time* the system enters Temporary Failure mode.
charge_overcurrent_detection_time	const uint16_t	[125ms]	
charge_overcurrent_clear_timeout	const uint16_t	[125ms]	After this timeout, the system clears the charge overcurrent fault and attempts to re-enable the CFET.
discharge_overcurrent	const int32_t	[mA]	If the charge current is higher than this threshold for longer than *discharge_overcurrent_detection_time* the system enters Temporary Failure mode.
discharge_overcurrent_detection_time	const uint16_t	[125ms]	
discharge_overcurrent_clear_timeout	const uint16_t	[125ms]	After this timeout, the system clears the discharge overcurrent fault and attempts to re-enable the DFET.
Important! The current-related HW thresholds are available in the Layer 2 configuration data.			

Table 9. Members of the BMS General Configuration Structure (Cont.)

typedef struct st_bms_general_cfg_t			
Member	**Type**	**Unit**	**Description**
Safety Settings (Voltage)			
cell_overvoltage	const uint16_t	[mV]	If any cell voltage is higher than this threshold for longer than *voltage_event_detection_time*, the system enters Temporary Failure mode.
cell_undervoltage	const uint16_t	[mV]	If any cell voltage is lower than this threshold for longer than *voltage_event_detection_time*, the system enters Temporary Failure mode.
cell_voltage_hysteresis	const uint16_t	[mV]	The cell voltage hysteresis level to clear the fault condition caused by cell over/undervoltage.
delta_cell_overvoltage	const uint16_t	[mV]	If the delta cell voltage is higher than this threshold for longer than *voltage_event_detection_time*, the system enters Permanent Failure mode.
pack_overvoltage	const uint32_t	[mV]	If the pack voltage is higher than this threshold for longer than *voltage_event_detection_time*, the system enters Temporary Failure mode.
pack_undervoltage	const uint32_t	[mV]	If the pack voltage is lower than this threshold for longer than *voltage_event_detection_time*, the system enters Temporary Failure mode.
pack_voltage_hysteresis	const uint16_t	[mV]	The pack voltage hysteresis level to clear the fault condition caused by pack over/undervoltage.
perm_failure_overvoltage	const uint16_t	[mV]	If any cell voltage is higher than this threshold for longer than *voltage_event_detection_time*, the system enters Permanent Failure mode.
voltage_event_detection_time	const uint16_t	[125ms]	The overvoltage event is detected if the condition is still present after this time.
Look Up Tables			
temperature_tbl	const int16_t	[0.1deg.C]	Temperatures from the thermistor table. The size of the array is: 8
resistance_at_temperature_tbl	const uint32_t	[Ω]	Resistance corresponding to the temperatures from the thermistor table. The size of the array is: 8
thermistor_pull_up_resistance	const uint32_t	[Ω]	Pull-up resistance of the thermistor.
Applications			
The configuration settings for the applications can be found in the relevant sections.			

Table 10 contains an enumeration with the available options about BMIC API interface selection for the system.

Table 10. BMS BMIC API Enumeration

typedef enum e_bms_bfe_api_t	
Constant	**Description**
BMS_RAA489206	The RAA489206 BMIC API Implementation is in use.

5. System Status and Data

The system status and data are stored in dedicated structures. The applications can access this data directly (such as the Remaining Capacity Control uses the cell voltages and pack current to calculate the remaining capacity). Table 11 shows the content of the BMS Status Structure. It contains the present and previous mode of the state machine (Table 13); the state of the CFET and DFET; a pointer to the Battery Status Structure (Table 12); and a pointer to the faults structure and failures structure.

Table 11. Members of the BMS Status Structure

typedef struct st_bms_status_t		
Member	**Type**	**Description**
* p_battery_status	u_batt_status_t	The status of the battery pack based on the current direction and battery cell condition
bms_mode	e_bms_mode_t	The current mode of the BMS
bms_mode_prev	e_bms_mode_t	The previous mode of the BMS
* p_faults	u_bms_faults_t	Pointer to a union, containing all fault flags
* p_failures	u_bms_failures_t	Pointer to a union, containing all failure flags
cfet_state	e_bfe_fet_state_t	Charing FET state
dfet_state	e_bfe_fet_state_t	Discharging FET state

Table 12. Battery Status Union

typedef union u_batt_status_t		
Member	**Type**	**Description**
batt_status	uint8_t	A variable containing all battery status flags
flags_b	-	Structure with flag bits
CHARGING	uint8_t (1 bit)	Battery pack charging flag (Bit[0]): ▪ Set when *i_pack_charge* ≥ *charge_detection_current* ▪ Cleared when i_pack_charge < charge_detection_current
DISCHARGING	uint8_t (1 bit)	Battery pack discharging flag (Bit[1]): ▪ Set when *i_pack_discharge* ≥ *discharge_detection_current* ▪ Cleared when i_pack_discharge < discharge_detection_current
FULLY_CHARGED	uint8_t (1 bit)	Battery pack fully charged flag (Bit[2]): ▪ Set when *v_cell_max* ≥ *end_of_charge_voltage* & *i_pack_charge* ≤ *end_of_charge_current* for *end_of_charge_timeout* ▪ Cleared when *v_cell_max* < *full_charge_release_voltage*
FULLY_DISCHARGED	uint8_t (1 bit)	Battery pack fully discharged flag (Bit[3]): ▪ Set when *UVPF = 1 or UVPHF = 1* (Table 14) ▪ Cleared when *UVPF = 0 and UVPHF = 0* (Table 14)
DISABLED	uint8_t (1 bit)	Battery pack disabled flag (Bit[4]): Set when the system is in Permanent Failure Mode or Power-Down Mode
RSV_5_7	uint8_t (3 bit)	Reserved (Bits[5:7])

Table 13. BMS Modes and States Enumeration

typedef enum e_bms_mode_t	
Constant	**Description**
BMS_INIT_STATE	Initialization State
BMS_WAKE_UP_STATE	Wake-Up State
BMS_ CHARGE_DISCHARGE_MODE	Charge/Discharge Mode
BMS_SLEEP_MODE	Sleep Mode
BMS_TEMP_FAILURE_MODE	Temporary Failure Mode
BMS_PERM_FAILURE_MODE	Permanent Failure Mode
BMS_POWER_DOWN_MODE	Power-Down Mode

The data about the system level faults is stored in a union (Table 14) so that it can be accessed in two different ways. The union contains a 32-bit variable: faults whose bits correspond to the specific system faults and a structure giving access to each of those bits individually. For example, in Temporary Failure mode, the state of the CFET and DFET is controlled based on which bits are set in the faults variable.

Table 14. Members of the BMS Faults Union

typedef union u_bms_faults_t		
Member	**Type**	**Description**
faults	uint32_t	A variable containing all fault flags
flags_b	-	Structure with flag bits
COCF	uint32_t (1 bit)	Charge overcurrent fault flag (Bit[0]): ▪ Set when *i_pack_charge* ≥ *charge_overcurrent* for longer than *charge_overcurrent_detection_time* ▪ Cleared after *discharge_overcurrent_clear_timeout* or *DISCHARGING* = *true*
COCHF	uint32_t (1 bit)	Charge overcurrent hardware fault flag (Bit[1]): ▪ Set when a charge overcurrent fault is being reported by the BMIC API implementation for longer than *current_event_detection_time_hw* ▪ Cleared after *discharge_overcurrent_clear_timeout* or *DISCHARGING* = *true*
RSV_2	uint32_t (1 bit)	Reserved (Bit[2])
OVCF	uint32_t (1 bit)	Cell overvoltage fault flag (Bit[3]): ▪ Set when *v_cell_max* ≥ *cell_overvoltage* for longer than *voltage_event_detection_time* ▪ Cleared when v_cell_max < cell_overvoltage - cell_voltage_hysteresis
OVCHF	uint32_t (1 bit)	Cell overvoltage hardware fault flag (Bit[4]): ▪ Set when a cell overvoltage fault is being reported by the BMIC API implementation for longer than *voltage_event_detection_time_hw* ▪ Cleared when the cell overvoltage fault is no longer being reported by the BMIC API implementation for longer than the detection time (check BMIC settings)
OVPF	uint32_t (1 bit)	Pack overvoltage fault flag (Bit[5]): ▪ Set when *v_pack* ≥ *pack_overvoltage* for longer than *voltage_event_detection_time* ▪ Cleared when v_pack < pack_overvoltage - pack_voltage_hysteresis
OVPHF	uint32_t (1 bit)	Pack overvoltage hardware fault flag (Bit[6]): ▪ Set when a pack overvoltage fault is being reported by the BMIC API implementation for longer than *voltage_event_detection_time_hw* ▪ Cleared when the pack overvoltage fault is no longer being reported by the BMIC API implementation for longer than the detection time (check BMIC settings)
OTCCF	uint32_t (1 bit)	Cell charge over-temperature fault flag (Bit[7]): ▪ Set when *CHARGING* = true and *t_cell_max* ≥ *cell_charge_overtemperature* for longer than *temp_event_detection_time* ▪ Cleared when *t_cell_max* < *cell_charge_overtemperature* - cell_charge_temp_hysteresis

Table 14. Members of the BMS Faults Union (Cont.)

typedef union u_bms_faults_t		
Member	**Type**	**Description**
OTCCHF	uint32_t (1 bit)	Cell charge over-temperature hardware fault flag (Bit[8]): ▪ Set when a cell discharge over-temperature fault is being reported by the BMIC API implementation for longer than *temp_event_detection_time_hw* ▪ Cleared when the cell over-temperature fault is no longer being reported by the BMIC API implementation for longer than the detection time (check BMIC settings)
UTCCF	uint32_t (1 bit)	Cell charge under-temperature fault flag (Bit[9]): ▪ Set when *CHARGING* = true and *t_cell_min* ≤ *cell_charge_undertemperature* for longer than *temp_event_detection_time* ▪ Cleared when *t_cell_ min* > *cell_charge_undertemperature* + cell_charge_temp_hysteresis
UTCCHF	uint32_t (1 bit)	Cell charge under-temperature hardware fault flag (Bit[10]): ▪ Set when a cell charge under-temperature fault is being reported by the BMIC API implementation for longer than *temp_event_detection_time_hw* ▪ Cleared when the cell under-temperature fault is no longer being reported by the BMIC API implementation for longer than the detection time (check BMIC settings)
RSV_11	uint32_t (1 bit)	Reserved (Bit[11])
DOCF	uint32_t (1 bit)	Discharge overcurrent fault flag (Bit[12]): ▪ Set when *i_pack_discharge* ≥ *abs(discharge_overcurrent)* for longer than dis*charge_overcurrent_detection_time* ▪ Cleared after *discharge_overcurrent_clear_timeout* or *CHARGING = true*
DOCHF	uint32_t (1 bit)	Discharge overcurrent hardware fault flag (Bit[13]): ▪ Set when a discharge overcurrent fault is being reported by the BMIC API implementation for longer than *current_event_detection_time_hw* ▪ Cleared after *discharge_overcurrent_clear_timeout* or *CHARGING = true*
UVCF	uint32_t (1 bit)	Cell undervoltage fault flag (Bit[14]): ▪ Set when *v_cell_min* ≤ *cell_undervoltage* for longer than *voltage_event_detection_time* ▪ Cleared when v_cell_min > cell_undervoltage + cell_voltage_hysteresis
UVCHF	uint32_t (1 bit)	Cell undervoltage hardware fault flag (Bit[15]): ▪ Set when a cell undervoltage fault is being reported by the BMIC API implementation for longer than *voltage_event_detection_time_hw* ▪ Cleared when the cell undervoltage fault is no longer being reported by the BMIC API implementation for longer than the detection time (check BMIC settings)
UVPF	uint32_t (1 bit)	Pack undervoltage fault flag (Bit[16]): ▪ Set when *v_pack* ≤ *pack_undervoltage* for longer than *voltage_event_detection_time* ▪ Cleared when v_pack > pack_undervoltage + pack_voltage_hysteresis
UVPHF	uint32_t (1 bit)	Pack undervoltage hardware fault flag (Bit[17]): ▪ Set when a pack undervoltage fault is being reported by the BMIC API implementation for longer than *voltage_event_detection_time_hw* ▪ Cleared when the pack undervoltage fault is no longer being reported by the BMIC API implementation for longer than the detection time (check BMIC settings)
OTDCF	uint32_t (1 bit)	Cell discharge over-temperature fault flag (Bit[18]): ▪ Set when *DISCHARGING* = true and *t_cell_max* ≥ *cell_discharge_overtemperature* for longer than *temp_event_detection_time* ▪ Cleared when *t_cell_max* < *cell_discharge_overtemperature* - cell_discharge_temp_hysteresis
OTDCHF	uint32_t (1 bit)	Cell discharge over-temperature hardware fault flag (Bit[19]): ▪ Set when a cell discharge over-temperature fault is being reported by the BMIC API implementation for longer than *temp_event_detection_time_hw* ▪ Cleared when the cell over-temperature fault is no longer being reported by the BMIC API implementation for longer than the detection time (check BMIC settings)
UTDCF	uint32_t (1 bit)	Cell discharge under-temperature fault flag (Bit[20]): ▪ Set when *DISCHARGING* = true and *t_cell_ min* ≤ *cell_discharge_undertemperature* for longer than *temp_event_detection_time* ▪ Cleared when *t_cell_ min* > *cell_discharge_undertemperature* + cell_discharge_temp_hysteresis

Table 14. Members of the BMS Faults Union (Cont.)

typedef union u_bms_faults_t		
Member	**Type**	**Description**
UTDCHF	uint32_t (1 bit)	Cell discharge under-temperature hardware fault flag (Bit[21]): ▪ Set when a cell discharge under-temperature fault is being reported by the BMIC API implementation for longer than *temp_event_detection_time_hw* ▪ Cleared when the cell under-temperature fault is no longer being reported by the BMIC API implementation for longer than the detection time (check BMIC settings)
RSV_22_23	uint32_t (2 bits)	Reserved (Bits[22:23])
OTPOF	uint32_t (1 bit)	Other over-temperature fault flag (Bit[24]): ▪ Set when *t_other* (max) ≥ *other_overtemperature* for longer than *temp_event_detection_time* ▪ Cleared *t_other* (max) < *other_overtemperature* - *other_temp_hysteresis*
UTPOF	uint32_t (1 bit)	Other under-temperature fault flag (Bit[25]): ▪ Set when *t_other* (min) ≤ *other_undertemperature* for longer than *temp_event_detection_time* ▪ Cleared when *t_other* (min) > *other_undertemperature* + *other_temp_hysteresis*
OTPOHF	uint32_t (1 bit)	Other over-temperature hardware fault flag (Bit[26]): ▪ Set when an other over-temperature fault is being reported by the BMIC API implementation for longer than *temp_event_detection_time_hw* ▪ Cleared when the other over-temperature fault is no longer being reported by the BMIC API implementation for longer than the detection time (check BMIC settings)
UTPOHF	uint32_t (1 bit)	Other under-temperature hardware fault flag (Bit[27]): ▪ Set when an other under-temperature fault is being reported by the BMIC API implementation for longer than *temp_event_detection_time_hw* ▪ Cleared when the other under-temperature fault is no longer being reported by the BMIC API implementation for longer than the detection time (check BMIC settings)
IOTPHF	uint32_t (1 bit)	Internal over-temperature hardware fault flag (Bit[28]): ▪ Set when an internal over-temperature fault is reported by the BMIC API implementation. ▪ Cleared when the internal over-temperature fault is no longer reported by the BMIC API implementation for longer than the detection time (check BMIC settings)
SCHF	uint32_t (1 bit)	Short-circuit hardware fault flag (Bit[29]): ▪ Set when a short-circuit fault is reported by the BMIC API implementation ▪ Cleared when the BMIC API implementation reports that the load is removed
RSV_30_31	uint32_t (2 bits)	Reserved (Bits[30:31])

The system-level failures data is stored in another union (Table 15) so that it can be accessed in two different ways. The union contains a 32-bit variable: failures whose bits correspond to the specific system failures and a structure giving access to each of those bits individually. The battery pack is disabled when any bit in the failures variable is set and the system enters Temporary Failure mode.

Table 15. Members of the BMS Failures Union

typedef union u_bms_failures_t		
Member	**Type**	**Description**
failures	uint32_t	A variable containing all failure flags
flags_b	-	Structure with flag bits
STFL	uint32_t (1 bit)	Self-test failure flag (Bit[0]) – Set when a self-test fault is reported by the BMIC API implementation.
REGFL	uint32_t (1 bit)	Voltage regulator failure-flag (Bit[1]) – Set when a failure in the voltage regulator is reported by the BMIC API implementation.
OWFL	uint32_t (1 bit)	Open-wire failure flag (Bit[2]) – Set when an open-wire fault is reported by the BMIC API implementation.
COVFL	uint32_t (1 bit)	Critical overvoltage failure flag (Bit[3]) – Set when *v_cell_max* ≥ *perm_failure_overvoltage* for longer than *voltage_event_detection_time*
COTFL	uint32_t (1 bit)	Critical over-temperature failure flag (Bit[4]) – Set when *t_cell_max* ≥ *perm_failure_overtemp* for longer than *temp_event_detection_time*
IMBFL	uint32_t (1 bit)	Cell imbalance failure flag (Bit[5]) – Set when a cell imbalance fault is reported by the BMIC API implementation.
COMMFL	uint32_t (1 bit)	BMIC communication failure flag (Bit[6]) – Set when a communication fault is reported by the BMIC API implementation.
RSV_7	uint32_t (1 bit)	Reserved (Bit[7])
MEMFL	uint32_t (1 bit)	Memory check failure flag (Bit[8]) – Sets when a memory check accomplished by the BMIC API implementation returns an error
FETFL	uint32_t (1 bit)	FET failure flag (Bit[9]) – Sets when the BMIC API implementation returns an error in the FETs driver
RSV_10_14	uint32_t (5 bits)	Reserved (Bits[10:14])
FSP_ERR	uint32_t (1 bit)	FSP API function has returned an error (Bit[15])
BFE_ERR	uint32_t (8 bits)	BMIC API functions error return (Bits[16:23])
RSV_24_31	uint32_t (8 bits)	Reserved (Bits[24:31])

Note: The BMIC implementation layer is detecting faults and returning information about them. They are mapped to either the BMS Faults Union or to the BMS Failures Union depending on the effect over the system. For more details, refer to the descriptions in Table 14 and Table 15.

The BMS Data Structure combines all measured values on a system level (Table 16). It contains the following: the battery pack voltage, a pointer to a structure containing the pack currents (Table 17), a pointer to a structure containing all temperatures in the system (Table 18), and a pointer to a structure containing all the cell voltages (Table 19).

Table 16. Members of the BMS Data Structure

typedef struct st_bms_data_t			
Member	**Type**	**Unit**	**Description**
v_pack	uint32_t	[mV]	Battery pack voltage
* p_currents	const st_bms_i_pack_t	-	Pointer to a structure, containing the battery pack current
* p_temperatures	const st_bms_temps_t	-	Pointer to a structure, containing all temperatures
* p_cells	const st_bms_v_cell_t	-	Pointer to a structure, containing all cell voltages

Table 17. Members of the BMS Pack Currents Structure

typedef struct st_bms_i_pack_t			
Member	**Type**	**Unit**	**Description**

Table 17. Members of the BMS Pack Currents Structure

i_pack	int32_t	[mA]	Battery pack current
i_pack_charge	int32_t	[mA]	Battery pack charge current
i_pack_discharge	int32_t	[mA]	Battery pack discharge current
i_pack_abs	int32_t	[mA]	Absolute battery pack current
i_pack_avg	int32_t	[mA]	Average battery pack current.[1] The average current value is calculated by moving an averaging window of 1 min.

1. The battery pack current averaging resets when the current changes its direction.

Table 18. Members of the BMS Temperatures Structure

typedef struct st_bms_temps_t			
Member	**Type**	**Unit**	**Description**
t_bfe	int16_t	[0.1deg.C]	BMIC internal temperature
t_cell[]	int16_t	[0.1deg.C]	An array containing the data for the external temperature inputs of the BMIC. The size of the array is the maximum number of external temperature inputs for the selected BMIC.
t_other[]	int16_t	[0.1deg.C]	An array containing the data for the external temperature inputs of the BMIC. The size of the array is the maximum number of external temperature inputs for the selected BMIC.
t_cell_min	int16_t	[0.1deg.C]	The minimum cell temperature
t_cell_max	int16_t	[0.1deg.C]	The maximum cell temperature
t_other_min	int16_t	[0.1deg.C]	The minimum other temperature
t_other_max	int16_t	[0.1deg.C]	The maximum other temperature

Table 19. Members of the BMS Cell Voltages Structure

typedef struct st_bms_v_cell_t			
Member	**Type**	**Unit**	**Description**
v_cell[]	uint16_t	[mV]	An array containing the data for all cell voltages. The size of the array is the maximum number of cells for the selected BMIC.
v_cell_max	uint16_t	[mV]	Maximum cell voltage
v_cell_min	uint16_t	[mV]	Minimum cell voltage
v_cell_delta	uint16_t	[mV]	The voltage difference between the minimum and maximum cell

6. Simple Cell Balancing (Application)

The sample code has an application for cell balancing that uses the Auto Cell Balancing function, which is integrated in the RAA489206. The BMIC determines which cells must be balanced, and it controls the cell balancing FETs as a part of the scan loop. However, the system controls the process and checks for the necessary conditions. Table 20 shows a structure that contains the cell balancing configuration parameters. Cell balancing can be enabled or disabled in general using the *cb_app_enable* constant and runs only during battery charging or idle. The system is not balancing any cells during discharge. Cell balancing is allowed only when the temperatures of the cells are within the range defined by *cb_temp_min_enable* and *cb_temp_max_enable*. The total time of running cell balancing is limited by *cb_max_time*. The counter is reset when the battery pack is balanced or cell balancing is not allowed anymore. *Note:* The parameters controlling the Auto Cell Balancing Sequence of the BMIC can be found in the configuration of the BMIC API implementation.

Table 20. Members of the Cell Balancing Parameters Structure

typedef struct st_bms_cb_cfg_t			
Member	**Type**	**Unit**	**Description**
cb_app_enable	const uint8_t	-	Turn on and off the module
cb_temp_min_enable	const int16_t	[0.1deg.C]	The minimum cell temperature to allow cell balancing (cb_temp_min_enable ≥ t_cell_min)[1][2]
cb_temp_max_enable	const int16_t	[0.1deg.C]	The maximum cell voltage per cell to allow cell balancing (cb_temp_max_enable ≤ t_cell_max)[1][2]
cb_max_time	const uint32_t	[s]	The total time cell balancing can run. The timer is reset when the pack is balanced or the direction of the current changes.

1. When this condition is not met, the cell balancing is inhibited for all cells.
2. The temperature comparison has a 1deg. C hysteresis to avoid oscillation over the threshold.

Table 21 shows the content of the cell balancing data structure. When cell balancing is allowed and active, cell_balancing is true. When there are no cells that require balancing, the pack_balanced is true.

Table 21. Members of the Cell Balancing Data Structure

typedef struct st_bms_cb_data_t			
Member	**Type**	**Unit**	**Description**
pack_balanced	uint8_t	-	Battery pack is (1: Balanced; 0: Not balanced)
cell_balancing	uint8_t	-	(1: Running; 0: Not running) cell balancing

Figure 12 shows the flowchart of the cell balancing algorithm. The system goes trough the flow every time in Charge/Discharge mode after reading the cell voltages as a part of the main loop (Figure 5). Cell balancing is not running in any other modes. The system checks if cell balancing is enabled and if the battery pack is balanced. Next, it checks if the temperatures are within the allowed range, the system is charging, and there are no detected faults or failures. The system checks if the maximum time for cell balancing has expired. This counter resets every time the pack is balanced or there are no valid conditions for it. The MCU next sends a command to the BMIC to start the auto cell balancing sequence or skips this command when it has already been running.

The Simple Cell Balancing Application uses the following API functions (and the actual implementation of these API functions are provided by the BMIC instance):

- p_cellBalanceControl - Initiates a single auto cell balancing sequence (BFE_BALANCE_MODE_AUTO).
- p_isCellBalancing - Returns the status of cell balancing in the BMIC.

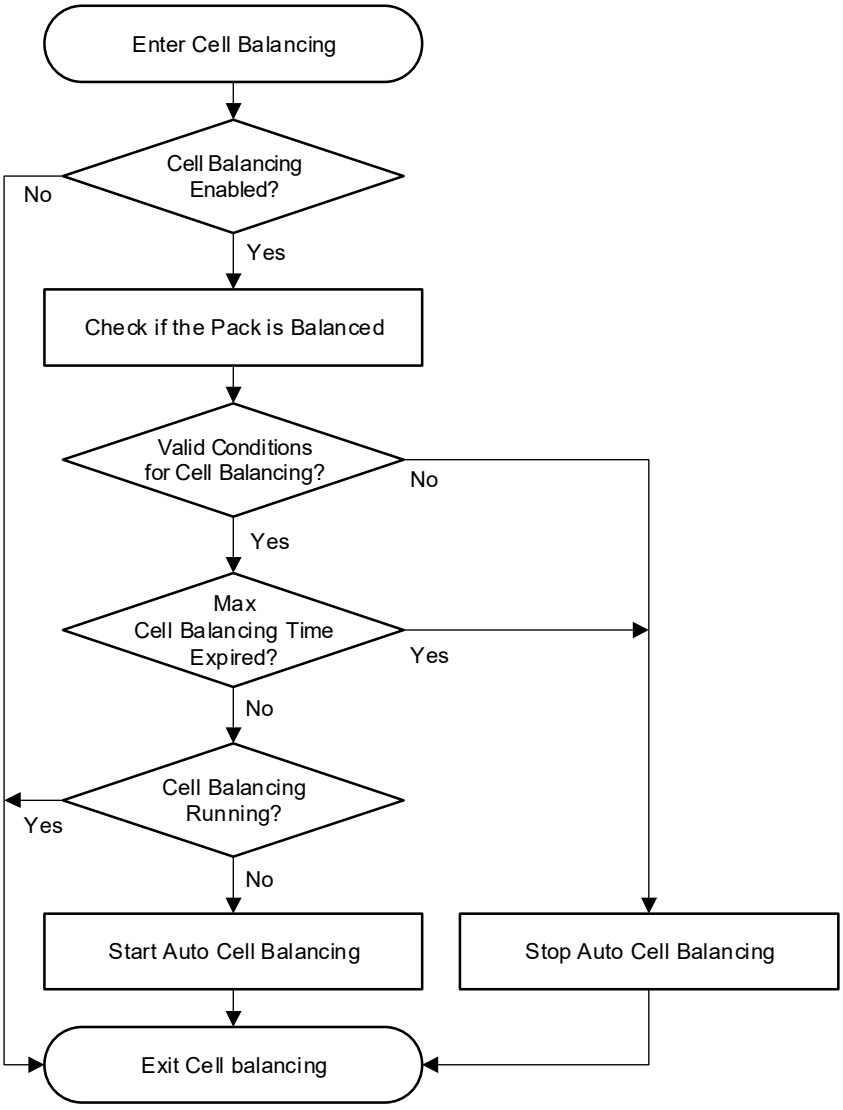

Figure 12. Cell Balancing Flowchart

7. Simple Remaining Capacity Control (Application)

The sample code has a basic application for capacity control that calculates the Relative State of Charge (RSOC) and relevant parameters as Impedance, Number of Cycles, and Remaining Capacity (RC). This application uses a Relative State of Charge (RSOC) calculation method based on the cell voltage and a Look-up-Table (LUT) but no integration of the charge or discharge currents or coulomb counting. This method provides a moderate accuracy for Li-Ion battery cells. Table 22 shows a structure that contains the configuration parameters for capacity control which are as follows: a table with the capacity versus open circuit voltage (two arrays with the same size – *capacity_tbl* and *voltage_at_capcity_tbl* forming the RC LUT), the initial cell impedance, the initial Full Charge Capacity (FCC) of the battery, and the capacity change after which a charging cycle can be detected. All interim values in the LUT are calculated based on linear interpolation. The resultant value is also limited by the minimum and maximum elements of the table and can not go outside this range. The increase in impedance because of cell deterioration over cycles is not considered.

Table 22. Members of the State of Charge Parameters Structure

typedef struct st_bms_soc_cfg_t			
Member	**Type**	**Unit**	**Description**
soc_app_enable	const uint8_t	-	Turn on and off the module.
capacity_tbl[]	const uint32_t	[mAh]	Capacity values from the battery capacity table.
voltage_at_capacity_tbl[]	const uint16_t	[mV]	Open circuit voltages corresponding to the capacities from the battery capacity table.
cell_impedance_init	const uint32_t	[uΩ]	The initial cell impedance.
capacity_full_charge_init	const uint32_t	[mAh]	Initial full-charge capacity
capacity_cycle_detection	const uint32_t	[mAh]	Change of the capacity to detect a cycle.
battery_cell_name	const char[20]	-	An array containing a name or identifier of the battery cells.

Table 23 shows the content of the data structure: the Relative State of Charge (RSOC), the Remaining Capacity (RC), the Full Charge Capacity (FCC), and the number of charging cycles.

Table 23. Members of the State of Charge Data Structure

typedef struct st_bms_soc_data_t			
Member	**Type**	**Unit**	**Description**
state_of_charge_relative	uint8_t	[%]	Relative State of Charge
capacity_remaining	const uint32_t	[mAh]	Remaining capacity
capacity_full_charge	const uint32_t	[mAh]	Full-charge capacity
cycle_count	uint16_t	-	Number of charging cycles

The number of cycles (*cycle_count*) increments, when all the following conditions are true:

- During charging the capacity has increased with more than *capacity_cycle_detection*.
- The charging has stopped.
- The pack is being discharged.

The Remaining Capacity (*capacity_remaining*) is recalculated every second using the LUT and the open-circuit cell voltage (*v_cell_min_ocv*) based on Equation 1:

(EQ. 1) $$v\_cell\_min\_ocv = v\_cell\_min + cell\_impedance\_init \times (i\_pack\_discharge - i\_pack\_charge)$$

The equation is valid for both parallel and series FETs which is the case when charging and discharging can occur either simultaneously or consecutively. Then the Relative State of Charge (*state_of_charge_relative*) is calculated based on Equation 2:

(EQ. 2) $$state\_of\_charge\_relative[\%] = \frac{capacity\_remaining}{capacity\_full\_charge} \times 100\%$$

8. Status Display

The status display provides a User Interface (UI) that displays information about the measured values and the status of the BMS (Figure 13). It is based on a command line interface where the RA MCU is sending data through a USB Serial Port (PCDC USB driver that creates a virtual COM port) to a workstation, running a terminal emulator program.

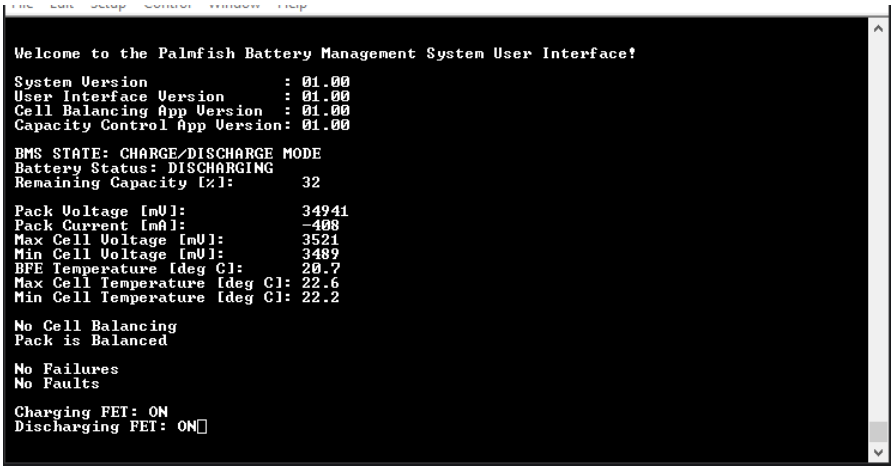

Figure 13. User Interface Window

The UI displays and updates the following information periodically or immediately when the mode changes or a fault/failure appears:

- The system and app versions
- Current BMS state (from BMS Status Structure in Table 11)
- The battery status (from BMS Status Structure in Table 11)
- The active BMS faults (when available)
- The active BMS failures (when available)
- The pack voltage and temperatures
- The pack current
- The minimum and maximum cell voltage
- The relative state of charge
- The cell balancing status
- The status of the CFET and DFET

9. Revision History

Revision	Date	Description
1.00	May 22, 2025	Initial release.

IMPORTANT NOTICE AND DISCLAIMER

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

These resources are intended for developers who are designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only to develop an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third-party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising from your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.