Getting started with the AEK-POW-BMS63EN battery management system (BMS) evaluation board based on the L9963E

Introduction

In a multicell battery pack, placing cells in series augments the possibilities of cell imbalance, which equates to a slower but persistent degradation of the battery.

There are always slight differences in the state of charge (SOC), self-discharge rate, capacity, impedance, and temperature characteristics, even for cells of the same model from the same manufacturer and even from the same batch of production.

These differences can lead to a divergence in a battery cell voltage over time. Cells with lower capacity or higher internal impedance tend to have higher voltage than the rest of the series cells at full charge. These cells are weakened further by continuous overcharge cycles. The higher voltage of weak cells at charge completion causes accelerated capacity degradation.

On the other hand, in discharge, the weak cells tend to have lower voltage than the other cells, due to either higher internal resistance or the faster rate of discharge that results from their smaller capacity. This means that if any of the weak cells hits the cell undervoltage-protection limit while the pack voltage is still sufficient to power the system, the full capacity of the battery is not used.

One of the emerging technologies for enhancing battery safety and extending battery life is advanced cell balancing. Since new cell balancing technologies track the amount of balancing needed by individual cells, the usable life of battery packs is increased, and overall battery safety is enhanced.

A battery management system (BMS) can manage battery cells, enhancing their safety and duration.

A BMS combines hardware and software to monitor and send acquired data to a dedicated device to protect the battery from overload or over discharge, lengthening its life cycle.

In the earlier BMSs, the SOC estimation was based only on the voltage reading.

Nowadays, instead, this estimation is based on Kalman filters, which provide more accurate data on the SOC by reading the voltage, the current, and the temperature of battery cells.

The modern BMS includes functions such as cell monitoring, balancing, safety and protection of the batteries, the state of charge estimation and thermal management:

- Cell monitoring: this function is based on the acquisition of the current, the voltage, and the temperature of each cell of the battery pack.
- Cell balancing: consists of distributing the energy and maintaining the SOC of the cells at a similar level. There are two different types of balancing: active and passive. Active balancing consists of transferring the exceeding energy of cells with a higher SOC to cells with a lower SOC to reach the same level. Passive balancing dissipates the exceeding energy of cells with a higher SOC, generating heat. Cell balancing is a key function to maintain the battery capacity and lengthen its duration.
- Battery safety and protection: a BMS ensures that the battery works in safe conditions for the users and the battery itself.
- State of charge (SOC) and state of health (SOH): an accurate SOC estimation is necessary to lengthen the battery life cycle, prevent battery damage, and ensure efficiency and accurate calculations of the SOH and cell balancing.

Our AEK-POW-BMS63EN is a battery management system (BMS) evaluation board that can handle from 1 to 31 Li-ion battery nodes. Each battery node manages from 4 to 14 battery cells, for a voltage range between 48 V and 800 V.

The board is based on the L9963E, which is designed for operation in both hybrid (HE) and full electric (BE) vehicles using lithium battery packs, but its use can be extended to other Transportation and Industrial applications.

The L9963E is an ASIL-D, AEC-Q1000 qualified device, compliant with ISO26262.

The main activity of the L9963E is monitoring cells and battery node status through stack voltage measurement, cell voltage measurement, temperature measurement, and coulomb counting. Measurement and diagnostic tasks can be executed either on demand or periodically, with a programmable cycle interval. Measurement data are available for an external microcontroller to perform charge balancing and to compute the state of charge (SOC) and the state of health (SOH).

The AEK-POW-BMS63EN can measure 5 values of temperature through 5 GPIOs (which can be configured as analog inputs to connect NTCs).

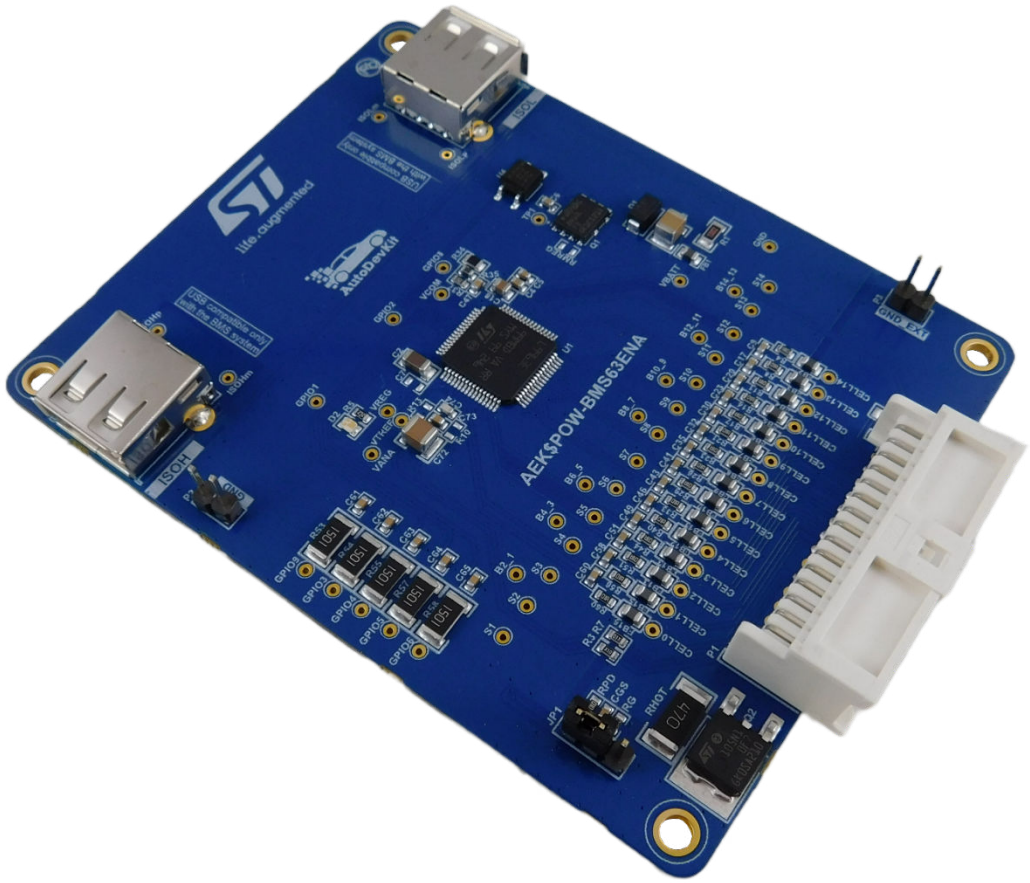
The AEK-POW-BMS63EN provides an elaborate monitoring network to sense the voltage, current, and temperature of each cell. This sensing allows elaborating the SOC of each battery cell and, consequently, the state of charge of all battery packs. The SOC allows assessing the remaining battery capacity, which equates to the remaining driving range. It also allows estimating the SOH of each cell, through an algorithm that calculates the difference in time of the real SOC from the nominal SOC during the charging/discharging phases. Using the SOC of each cell, it is possible to activate the cell passive balancing.

The AEK-POW-BMS63EN can work in two different daisy chain topologies: centralized and dual access ring.

In these topologies, the board communicates through an MCU board with one (for the centralized configuration) or two (for the dual access ring configuration) AEK-COM-ISOSPI1, which allows converting SPI signals in isolated SPI signals (and vice versa), thereby reducing the number of necessary wires from 4 to 2 and implementing differential communication for higher noise immunity. The communication is performed at two different speeds: high speed of 2.66 Mbps and low speed of 333 kbps.

Note: The AEK-POW-BMS63EN evaluation board is designed for R&D laboratory use only; it is not intended for field use in vehicles.

Figure 1. AEK-POW-BMS63EN evaluation board

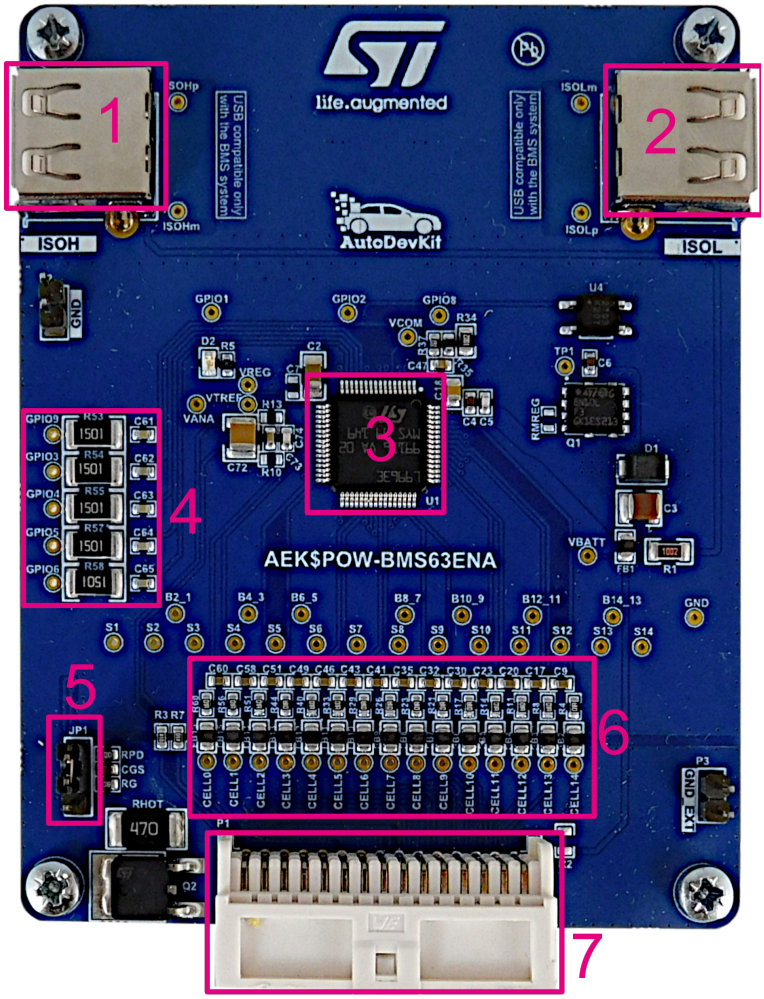


1 Hardware overview

1.1 Main components

1. ISOH port to connect the board to the AEK-COM-ISOSPI1
2. ISOL port to connect the board to another AEK-POW-BMS63EN in a daisy chain
3. L9963E
4. GPIOs for external NTC connection
5. Hot plug protection
6. Balancing resistors
7. Connector for the battery pack

Figure 2. AEK-POW-BMS63EN main components



1.2 L9963E

The L9963E is intended for operation in both hybrid electric (HE) and full electric (FE) vehicles using lithium battery packs. The IC embeds all the features needed to perform battery management. A single device can monitor from 4 up to 14 cells.

The device can be supplied with the same battery it monitors.

The L9963E main activity consists of monitoring cells and battery pack status through stack voltage measurement, cell voltage measurement, temperature measurement, and coulomb counting. Measurement and diagnostic tasks can be executed either on demand or periodically, with a programmable cycle interval.

Measurement data is available for an external microcontroller to perform charge balancing and to compute the state of health (SOH) and state of charge (SOC).

The IC works in normal mode performing measurement conversions, diagnostics, and communication. The device can also be put into a cyclic wakeup state in order to reduce the current consumption from the battery.

Passive cell balancing can be performed either via internal discharge path or via external MOSFETs. The controller can either manually control the balancing drivers or start a balancing task with a fixed duration. In the second case, the balancing may be programmed to continue also when the IC enters a low power mode called silent balancing, to avoid unnecessary current absorption from the battery pack.

Thanks to the GPIOs, the device also offers the possibility to operate a distributed cell temperature sensing via external NTCs resistances.

The external microcontroller can communicate with L9963E via SPI protocol. The physical layer can either be a classic 4-wire based SPI or 2-wire transformer/capacitive based isolated interface through a dedicated isolated transceiver device.

The L9963E performs automatic validation of any failure involving the cells or the whole battery pack. The device can detect the loss of the connection to a cell or GPIO terminal. Moreover, it features a hardware self-check (HWSC) that verifies the correct functionality of the internal analog comparators and the ADCs. All these checks are automatically performed in case a failure involving both cells or when the battery pack is detected. The current sensing interface used for coulomb counting is also capable of detecting failures such as open wires and overcurrent in sleep mode. The cell balancing terminals can detect any short/open fault and the internal power MOS are protected against overcurrent.

1.3 Voltage operating range

The AEK-POW-BMS63EN maximum voltage range for each cell is 4.2 V.

The power supply range is from 9.6 V to a maximum of 64 V.

1.3.1 Linear regulators

The AEK-POW-BMS63EN features several linear voltage regulators, which are switched on according to a specific sequence at power-up (see Figure 11. Finite state machine of the voltage conversion routine).

VREG

This linear regulator exploits an external MOS to decrease the power dissipation inside the L9963E.

It acts as a pre-regulator, supplying all other internal regulators (VANA, VCOM, VTREF, and VDIG). It is switched off in low power modes (sleep, silent balancing, off phase of the cyclic wakeup).

VANA

This low drop regulator supplies all the ADC, comparators, monitors, main bandgap, current generator, and other analogic blocks.

VCOM

The isolated communication receiver/transmitter and the GPIO output buffers are supplied by this low drop regulator.

VTREF

This low drop regulator is used to supply external components such as NTCs for temperature sensing.

The recommended application circuit in NTC analog front end guarantees that each NTC channel sinks no more than 500 µA.

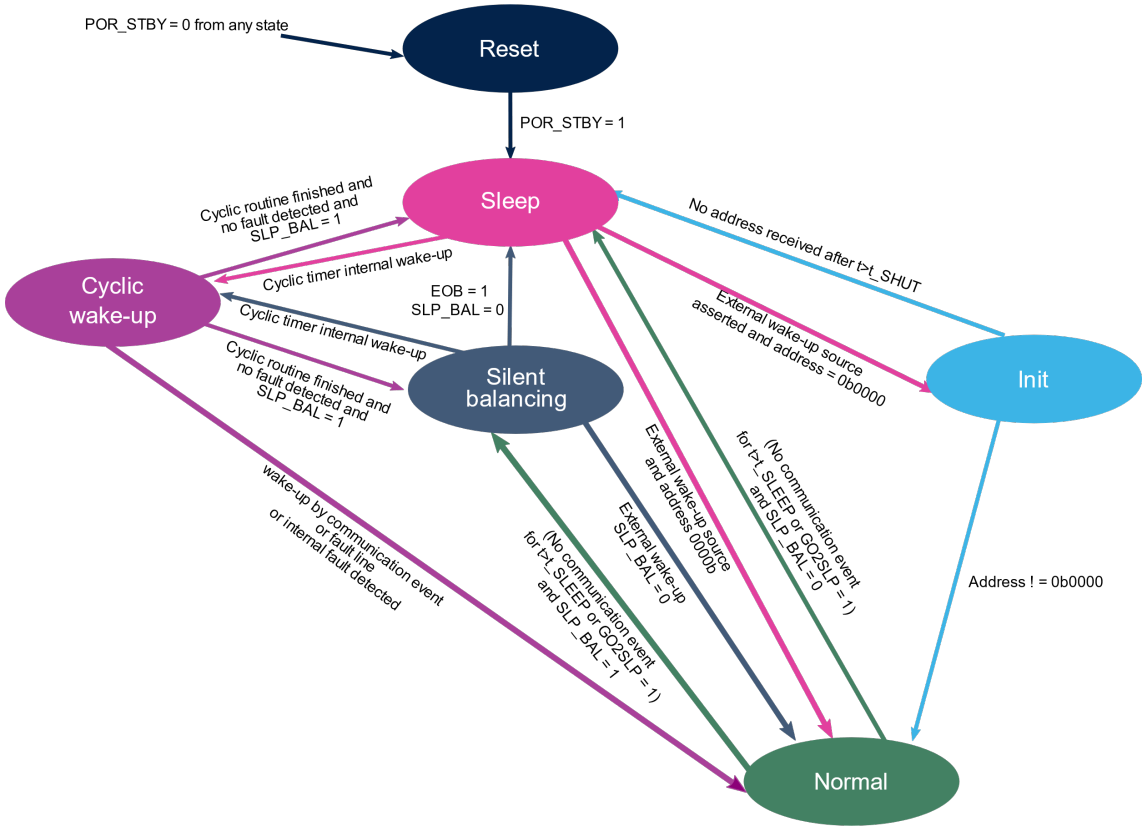
VTREF regulator is disabled by default. Its operation can be controlled via SPI.

In absolute measurements, there is no reference value, while the ratiometric measurement is based on reference value defined by the VTREF regulator. If the VTREF goes low in case of an error, the VTREF varies to compensate this error.

All of the above regulators have dedicated UV/OV diagnostics.

1.3.2 Simplified state machine

Figure 3. State machine



1.3.2.1 *Reset and sleep states*

When the standby logic is reset, all registers on the device are in the reset state. The battery voltage is still under threshold. No operation is possible during this state.

The sleep state is reached:

- From the reset state, when POR_STBY rises.
- From other states, if a GO2SLP command is sent or no communication is received for t > t_SLEEP.
- From the init state in case the device address is still 0b0000 after t > t_SHUT.
- From the cyclic wakeup state when the silent balancing is not resumed.

In the sleep state, the device is sensitive to external sources (such as ISO lines, fault line, SPI_CS (SPI_CLK) pins, and a GPIO pin for master units) to wake up the main logic. A slow oscillator works in this state to allow the device to wake up every t = t_{CYCLIC_SLEEP} + t_{CYCLIC_WUP} and move to the cyclic wakeup state.

During the sleep state, the current consumption is significantly reduced to the I_{SLEEP} current value: only the communication wakeup source monitoring, low-speed oscillator for cyclic wakeup timer, and the corresponding reference and power supply are activated.

1.3.2.2 *Init state*

In the init state, after having been woken up, the device waits for the microcontroller to send the address assignment command (refer to Section 1.3.2.2.1 Addressing procedure). If the address command is received before the initialization timer expires (t_SHUT), the device address is stored into a standby logic register (chip_ID) and the device enters the normal state.

The chip_ID field is then locked and no longer editable. Two actions can correctly re-initialize the device (including the chip_ID):

- Hardware reset: (POR_STBY)
- Software reset: set SW_RST and GO2SLP in the same frame

Note: the software reset leaves the communication timeout (CommTimeout) unmodified, and clears the chip_ID.

If only SW_RST is sent, the device waits for CommTimeout and then moves to the sleep state.

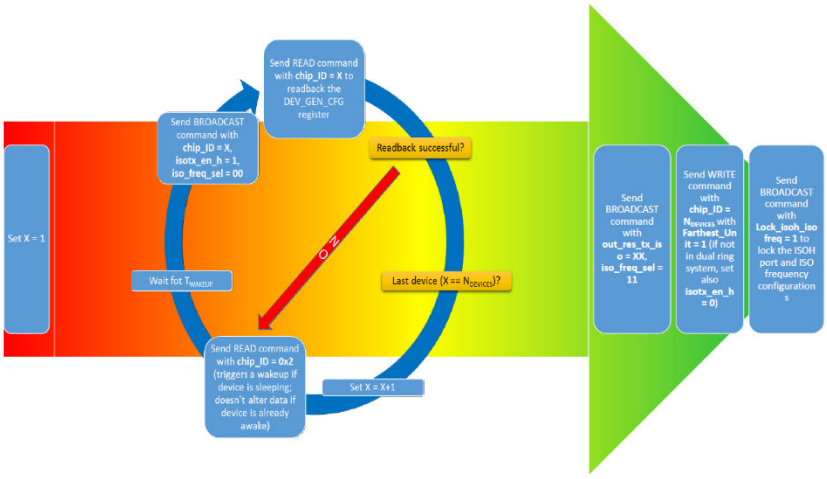
If the initialization timer (t_SHUT) expires before the command is received, the device goes back to the sleep state.

Any failure is masked until the device receives an address.

1.3.2.2.1 Addressing procedure

The following figure shows the daisy-chain addressing procedure for a stack of $N_{DEVICES}$.

Figure 4. Daisy chain addressing procedure

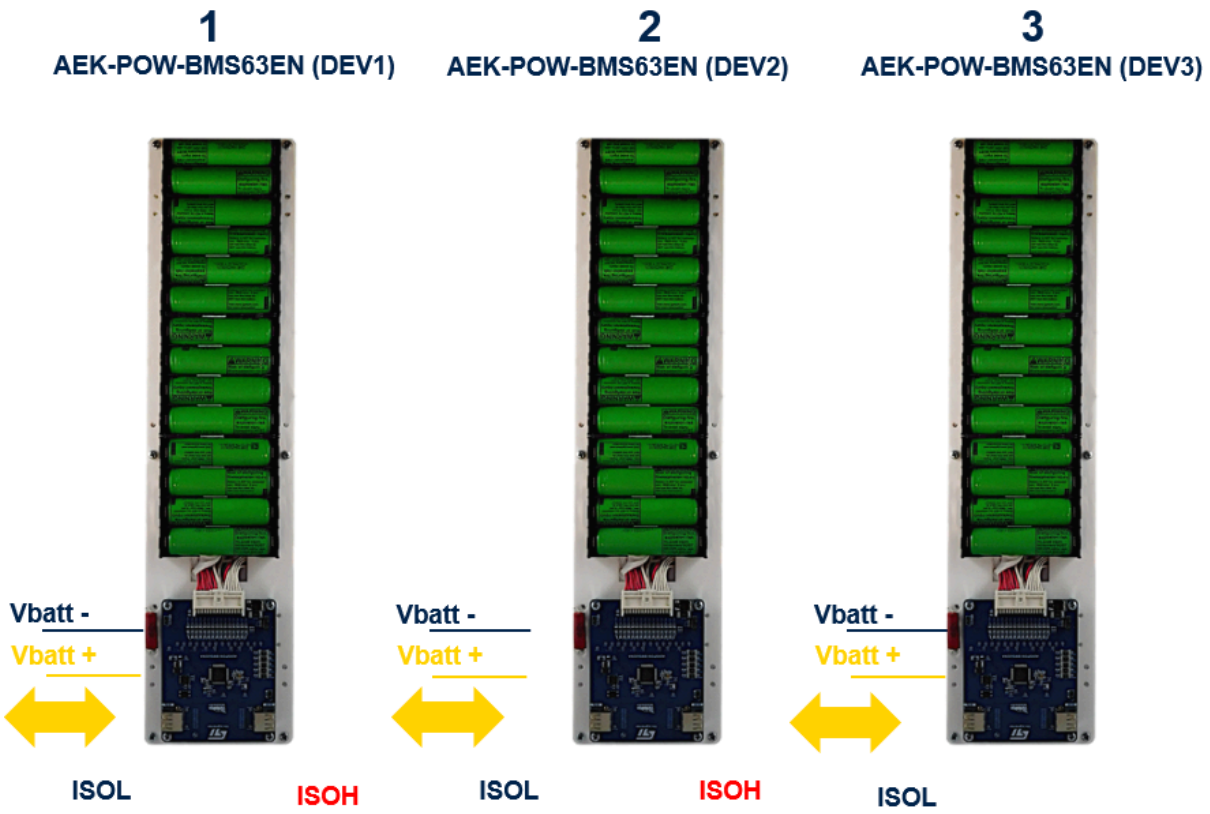


1.3.2.2.2 AEK-POW-BMS63EN addressing procedure

This procedure is performed through the “StartUpWakeUp” function, which configures the AEK-COM-ISOSPI1 and assigns an ID to each AEK-POW-BMS63EN of the chain.

IDs are sequentially assigned to all the BMS included in the system: ID no. 1 is assigned to the first BMS in the chain, ID no. 2 is assigned to the second, and so on.

Figure 5. **Addressing procedure example**



To determine if the addressing procedure has been successful, check the D2 LED on the BMS. If it remains always on, it means that the procedure has been correctly executed and each BMS has its own ID. Only if you unplug the connector, the BMS loses their ID.

1.3.2.3 *Normal state*

In this state, all references are powered. ADCs and interfaces are ready for measurement and data transmission, respectively.

The commands sent by the microcontroller can be read from both ISO lines and SPI pins.

When receiving a valid command, the L9963E executes the corresponding operations, such as voltage, current, and overtemperature measurements.

Some safety operations (OV, UV, OT, and VBAT monitoring) are automatically checked in background.

If the communication with the MCU is missing for t > t_SLEEP (programmable via CommTimeout, maskable via comm_timeout_dis) or a GO2SLP command is received, the device moves either to the sleep state or to the silent balancing state, depending on the slp_bal_conf bit and balancing state.

1.3.2.4 *Silent balancing state*

In this state, one or more cells are balanced with a reduced current consumption with respect to the normal state.

Active resources are the same as the ones of the sleep state plus the balance drivers and the necessary bias circuitry.

To enter the silent balancing state from the normal state, check the following conditions:

1. Cell balancing must be on.
2. The slp_bal_conf flag has to be set to 1.
3. Verify the “go to sleep” condition (an explicit GO2SLP command or communication timeout expiration).

If cell balancing is previously requested in the normal state and the slp_bal_conf flag is set to 1, when a condition to go to sleep (low consumption) occurs, the device enters the silent balancing state.

To exit from the silent balancing state, there are three possible events:

- A wakeup signal on communication or fault line can force the chip to stop balancing and then go back to the normal state.
- An external fault must bring the device to the normal state and stop the balancing.
- As soon as the required balancing target stops, the end of balancing (EOB) bit is set to 1 and the chip enters the sleep state.

If the Cyclic signal is raised, the device goes to the cyclic wakeup state, runs the diagnosis, and then resumes silent balancing (if slp_bal_conf flag = 1).

1.3.2.5 *Cyclic wakeup state*

From both sleep and silent balancing states, the device moves periodically (once every t_{CYCLIC_SLEEP}) to the cyclic wakeup state to monitor eventual faults.

The ADC must be on to check possible critical battery conditions. Any detected fault moves the device to the normal state.

An on-demand operation is possible only once the device has moved to the normal state in case of any detected fault.

Conditions to leave this state are:

- Any fault detected moves the device to the normal state.
- A wakeup from the fault line or communication line moves the device to the normal state if the defined monitoring tasks are finished; the device can move to the sleep or silent balancing states automatically based on the state before cyclic conversions (slp_bal_conf flag).

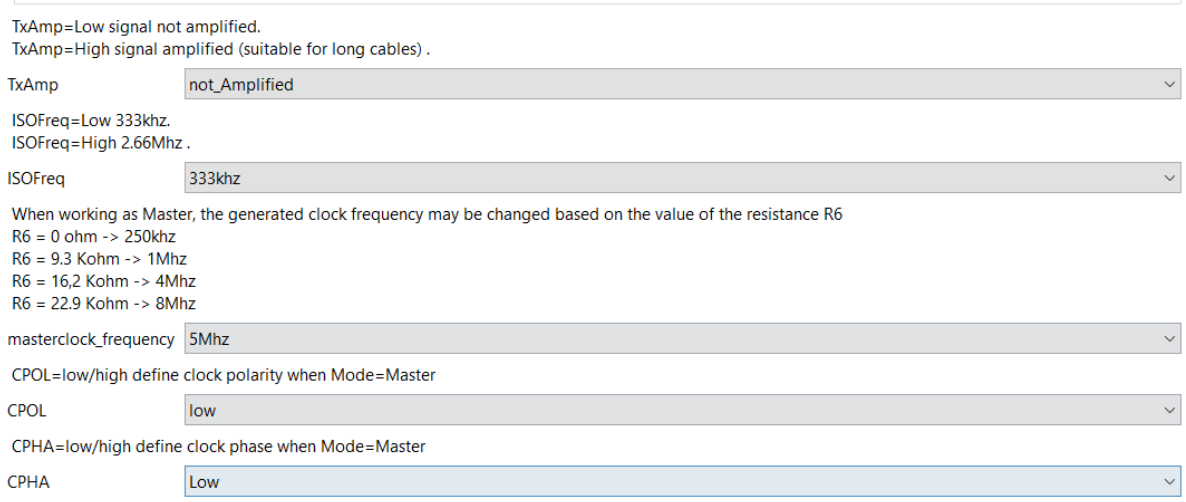
2 BMS topologies

The AEK-POW-BMS63EN can work in two different daisy chain topologies: centralized and dual access ring.

To build these two configurations, we used the AEK-COM-ISOSPI, which allows converting SPI signals in isolated SPI signals, thereby reducing the number of necessary wires from 4 to 2 and implementing differential communication for higher noise immunity.

To allow the AEK-COM-ISOSPI to communicate with the AEK-POW-BMS63EN, configure the AEK-COM-ISOSPI as per scenario no. 2 described in the AEK-COM-ISOSPI user manual, and shown in the figure below.

Figure 6. AEK-COM-ISOSPI configuration



2.1 Centralized configuration

In a centralized daisy chain configuration, a series of BMS is connected to an MCU board (e.g., AEK-MCU-C4MLIT1) through a single transceiver connected to the AEK-POW-BMS63EN isolated ISOL port. The BMS are connected to each other through the isolated ISOH port.

The MCU communicates with the AEK-COM-ISOSPI1 hosted L9963T transceiver through the SPI protocol. The transceiver converts these signals into ISO SPI signals to communicate with the BMS.

Figure 7. **Centralized BMS diagram**

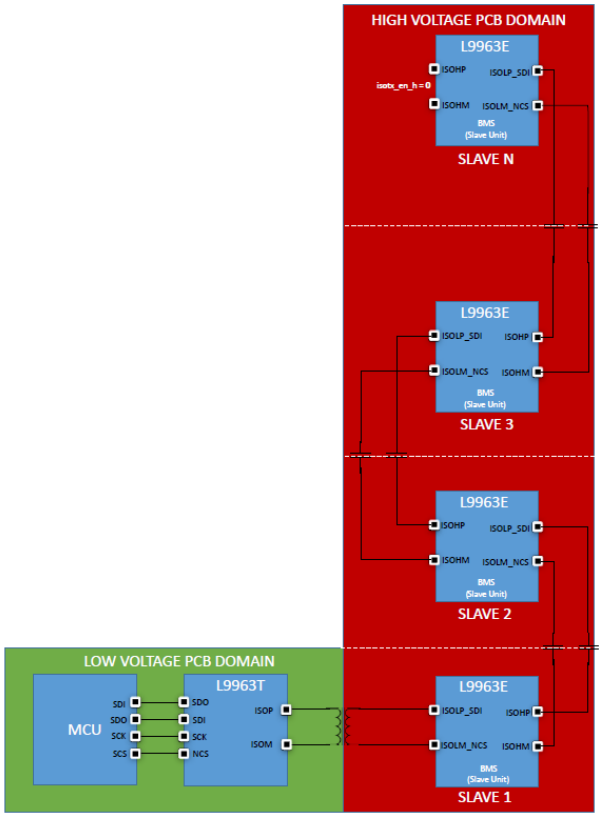
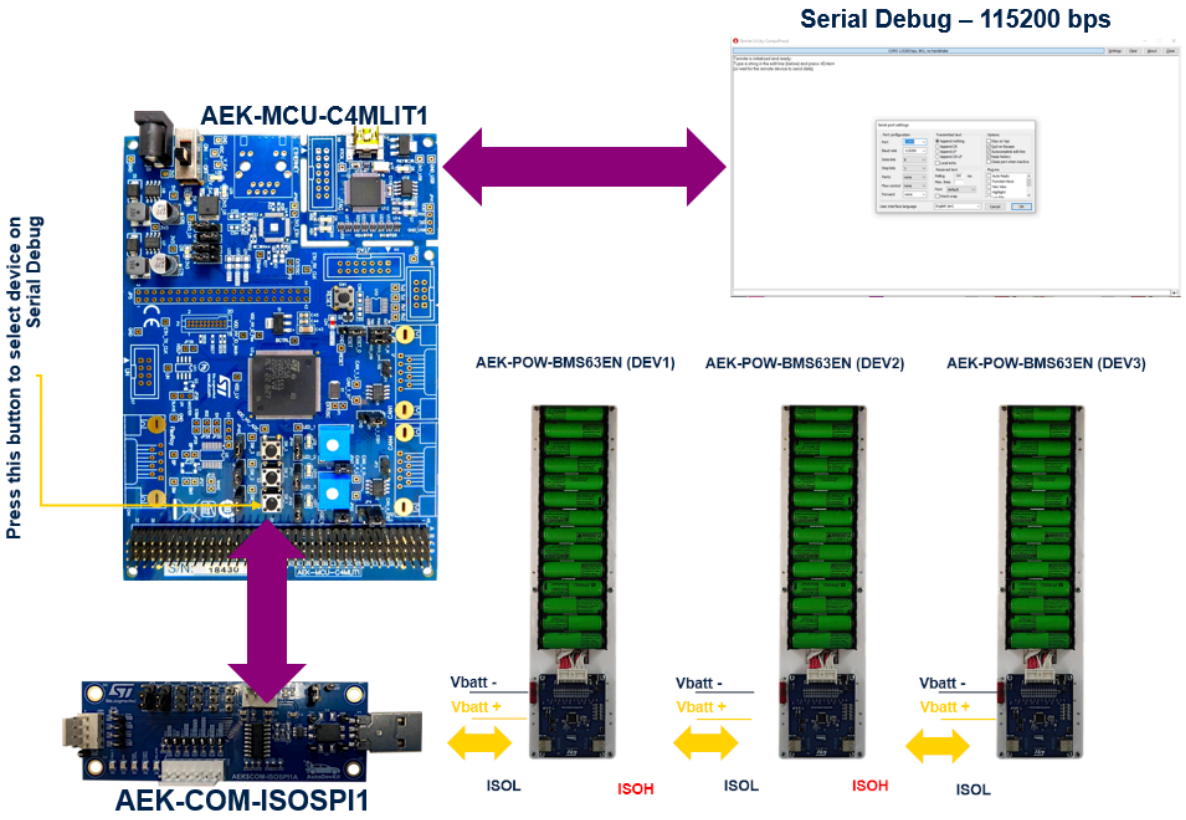


Figure 8. **Example of centralized BMS with AEK-COM-ISOSPI1 and AEK-MCU-C4MLIT1**



2.2 Dual access ring configuration

A dual access ring configuration is realized by adding another transceiver that makes the communication bidirectional. The secondary ring is used as a backup in case the primary ring fails. Data moves in opposite directions around the rings, and each ring remains independent of the other unless the primary ring fails. The two rings are connected to continue the flow of data traffic.

Figure 9. Dual access ring BMS diagram

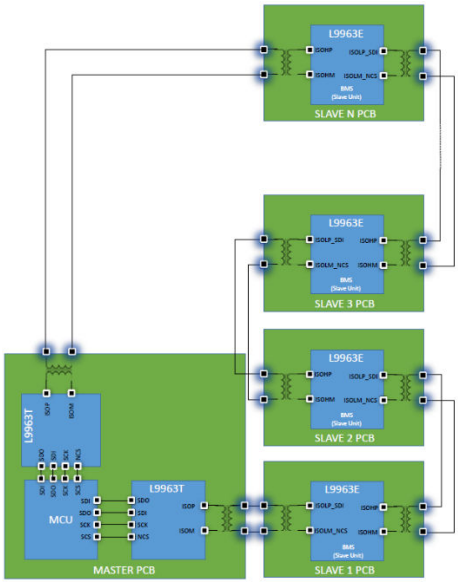
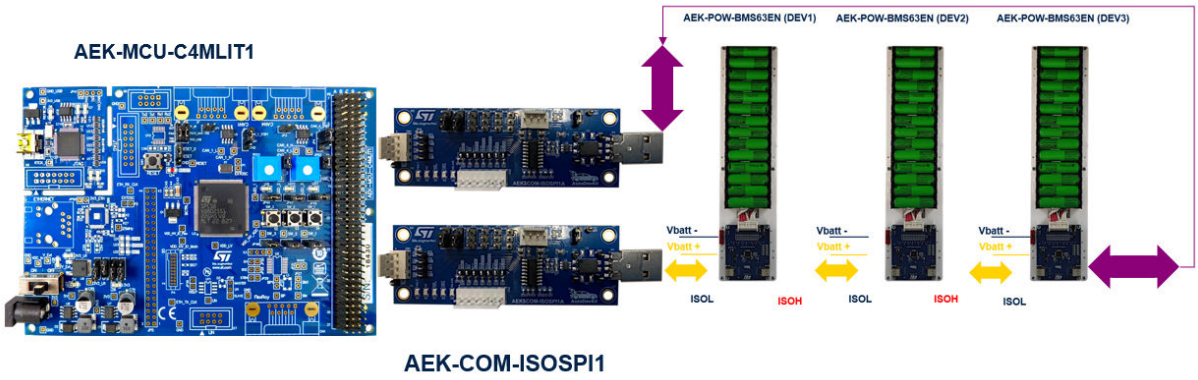


Figure 10. Dual ring configuration example



3 Voltage conversion routine

L9963E implements a flexible voltage conversion routine, whose main goals are:

- Providing on-demand information about the cell voltage, the stack voltage, and the cell temperature.
- Providing on-demand diagnostic information about device functionality.
- Periodically monitoring the cells and the stack status, along with the device functionality.
- Limit power consumption by activating only the necessary resources.
- Automatically validate any eventual failure detected during the routine execution.

The voltage conversion routine can be executed in three different ways according to the microcontroller commands. The different modes are mutually exclusive: only one routine at a given time is allowed and multiple threads are not supported.

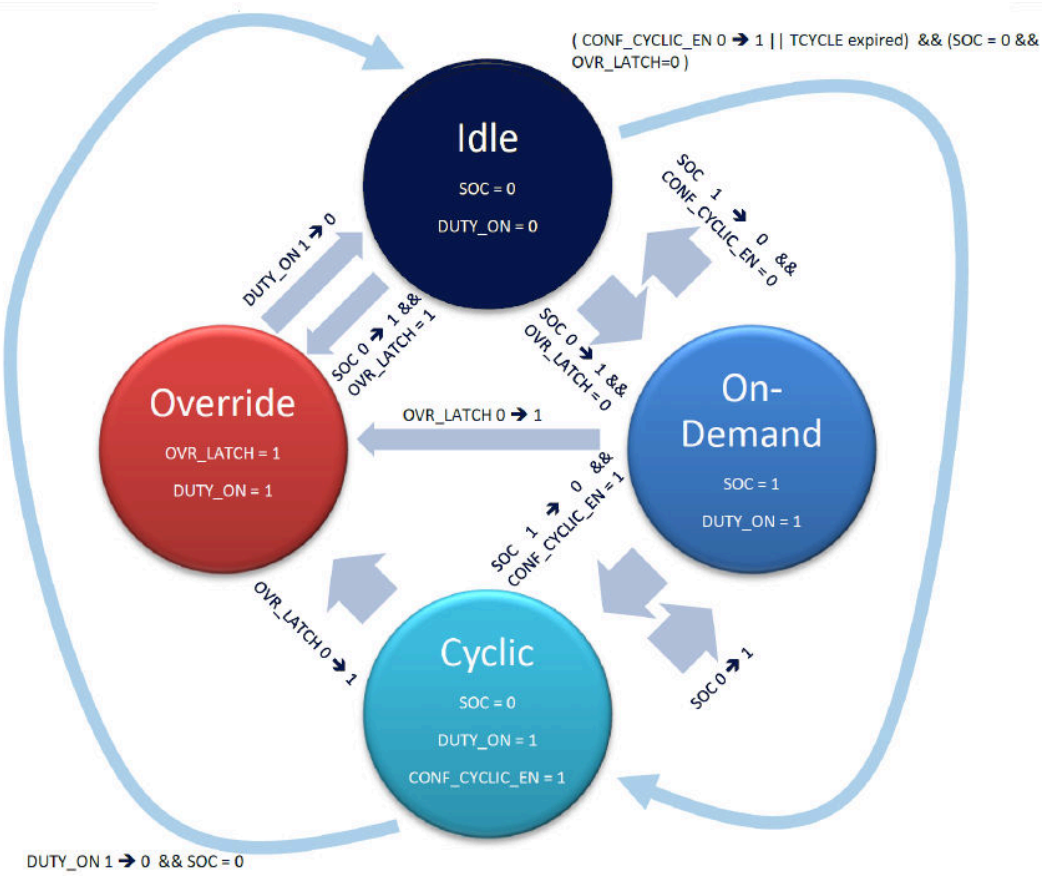
The execution modes have different priorities:

- Configuration override is high priority since its purpose is to perform diagnostics upon failure detection to validate the failure type. It can interrupt any ongoing activity and, once done, the voltage conversion routine is moved to the idle state, waiting for the microcontroller to interpret the diagnostic data.
- On-demand conversions are low priority. They are meant to allow the microcontroller to perform measurements or diagnostic at chosen time instants. They cannot co-exist with cyclic conversions: to run an on-demand conversion, cyclic conversions must be disabled and the MCU must wait for their termination (monitor the DUTY_ON flag). On the other hand, on-demand conversions cannot interrupt themselves or a configuration override.
- Cyclic conversions are low priority. Their main purpose is to monitor the battery pack and the L9963E status. However, they can also be used to retrieve periodically measurement data. They can be interrupted by the configuration override. They cannot co-exist with on-demand conversions: before enabling cyclic conversions, the MCU must wait for any ongoing on-demand conversion to end first (monitor the DUTY_ON flag).

The following FSM describes the functionality and the transitions among the different operating modes of the voltage conversion routine.

Figure 11. Finite state machine of the voltage conversion routine

SOC = state of conversion



To start on-demand conversions, the user must set SOC = 1 in the ADCV_CONV register: in case the coulomb counting routine is enabled, every time an on-demand voltage conversion is requested by setting SOC = 1, the actual conversion start is delayed until the first useful current conversion takes place. This allows a perfect synchronization between voltage and current samples but might result in a maximum delay of $T_{CYCLEADC\_CUR}$, which must be taken into account by user SW and added to the recommended T_{DATA_READY} as per the device datasheet.

To start cyclic conversions, the user must set CONF_CYCLIC_EN = 1 in the ADCV_CONV register. The ADC_FILTER_CYCLE determines the duration of the routine steps.

Cyclic conversions can be used for diagnostic and measurement purposes:

- In case the routine is only intended for diagnostic purposes, the user can set CYCLIC_UPDATE = 0. This setting causes any conversion result to be used only for internal comparisons. Data are subsequently discarded and registers containing measurement results are not updated.
- In case measurement results are used for diagnostics and SOC estimation, the user must set CYCLIC_UPDATE = 1, thus causing measurement register update upon each step completion, as for on-demand conversions. Note: results of a previous on-demand conversion might be overwritten by the ones of cyclic executions.

Two counters are implemented for driving the cyclic execution:

- TCYCLE, which is an SPI programmable timer accounting for cycle period.
- NCYCLE, which is an internal counter, incremented by 1 every time TCYCLE expires: it counts the number of cycles executed.

These counters are started/stopped upon FSM transitions.

4 Coulomb counting routine

The coulomb counting routine is performed to evaluate the charge injected/subtracted during vehicle operation.

To enable it, the CoulombCounter_en bit must be set to 1.

Disabling the coulomb counter by setting CoulombCounter_en to 0 does not reset the accumulator (CoulombCounter_msb, CoulombCounter_lsb) and sample counter (CoulombCntTime) registers. The MCU is responsible for resetting the coulomb counter, clearing any data previously stored.

5 Safety and diagnostic features

The L9963E provides an extended set of safety mechanisms to reach the required ASIL standard. It monitors potentially damaging conditions for the battery pack.

5.1 Cell UV/OV diagnostic

It is possible to select the value for the overvoltage threshold as well as for the undervoltage threshold of the cells.

This diagnostic feature is completed by analyzing, inside the logic block, the digital information provided by the voltage measurement ADCs. Measurements are performed just on enabled cells.

In case of cell UV/OV (VCELL_OV/UV):

- The corresponding fault flag is set and latched in the VCELL_OV / VCELL_UV register.
- Fault is propagated through the FAULT line.
- Balancing is stopped in case of UV event:
  - A cell UV causes balancing to be stopped on the whole cell stack.
  - A cell balance UV causes balancing activity to be stopped only on the affected cell.
  - The conversion routine goes into configuration override.

5.2 Total battery VBAT diagnostic

The total stack voltage diagnostic is implemented through three different safety mechanisms:

- Arithmetic sum of the digital information of cell ADC (within the cell conversion step of the voltage conversion routine). Such a value is then compared to the digital thresholds (programmable via registers).
- Direct conversion of the voltage at VBAT pin through internal resistive divider. The result is compared to fixed thresholds. This diagnostic is mainly intended to protect the IC against absolute maximum rating (AMR) violation on the VBAT pin. It can also be used as a redundant coherency check with the arithmetic sum of cells.
- Continuous sense of the VBAT pin voltage with a VBAT_UV/OV comparator, featuring fixed thresholds. It is used as an overvoltage warning or an undervoltage warning. This diagnostic is intended to provide a fast reaction against transient overvoltage and undervoltage events.

This UV/OV comparator is always enabled to guarantee a continuous safety check on VBAT voltage.

5.3 VBAT overvoltage

The aim of this diagnostic is to detect a dangerous increase of battery voltage to protect the circuitry connected to VBAT.

If VBAT > VBAT_OV_WARNING (COMP) (for a time longer than TVBAT_FILT) or VBATT_SUM > VBAT_OV (SUM) or VBATT_MONITOR > VBAT_CRITICAL_OV_TH, the overvoltage fault is directly reported in registers and notified to the microcontroller with 3 dedicated flags, according to the Fault communication procedure.

5.4 VBAT undervoltage

The aim of this diagnostic is to detect a decrease of battery voltage to notify this fault that may cause system malfunctions.

5.5 Cell open wire diagnostic

The open cell detection can be performed through the voltage conversion routine. The diagnostic strategy depends on the ADC_CROSS_CHECK bit.

5.5.1 Cell open with ADC_CROSS_CHECK = 0

If the cell terminal diagnostics step of the voltage conversion routine is executed having programmed ADC_CROSS_CHECK = 0, then the diagnostic addresses the following failures:

- RLPF (low pass filter resistor) degradation: diagnostic has been implemented to guarantee that low pass filter resistor in series to the Cx pin is below the critical limit RLPF_OPEN:
  - On odd cells, RLPF degradation will cause the assertion of the corresponding CELLx_OPEN flag.
  - On even cells, flag assertion depends on the RLPF degradation:
    - Small degradation (RLPF < 24 kΩ typ. with 10 nF CLPF) only causes the assertion of the corresponding CELLx_OPEN flag.
    - Large degradation (RLPF > 24 kΩ typ. with 10 nF CLPF) causes the assertion of both the corresponding CELLx_OPEN flag and the lower odd cell CELLx-1_OPEN flag.
- L9963E C1-C14 pin open.
- L9963E C0 pin open or PCB connector open.

Diagnostic is present on enabled cells only (VCELLx_EN = 1).

5.5.2 Cell open with ADC_CROSS_CHECK = 1

ADC_CROSS_CHECK = 1, then the diagnostic addresses the following issues:

- Failure in the filtering capacitor CLPF (low pass filter capacitor) causing an excessive leakage from cell.
- ADC error due to bandgap shift or failure on the conversion path.

For each pair of consecutive cells, the two corresponding ADCs, each of whom is referenced to a different bandgap, are measuring the voltage drop on the external RLPF.

Since no pull-down current is applied while measurement is ongoing, the voltage drop on RLPF is expected to be null, and the two measurement results should match.

If one of the two ADCs is experiencing an issue, or an excessive leakage from the CLPF is causing a voltage drop on the RLPF, a mismatch in the results occurs. If such a mismatch is greater than VADC_CROSS_FAIL, failure is detected.

5.6 PCB open diagnostic

To detect loss of cell wire at PCB connector, follow the procedure described in the L9963E datasheet.

Note: when performing PCB open diagnostic, other diagnostics such as Cell UV/OV diagnostic and balancing open load diagnostic might also be triggered. They must be then discarded by user software.

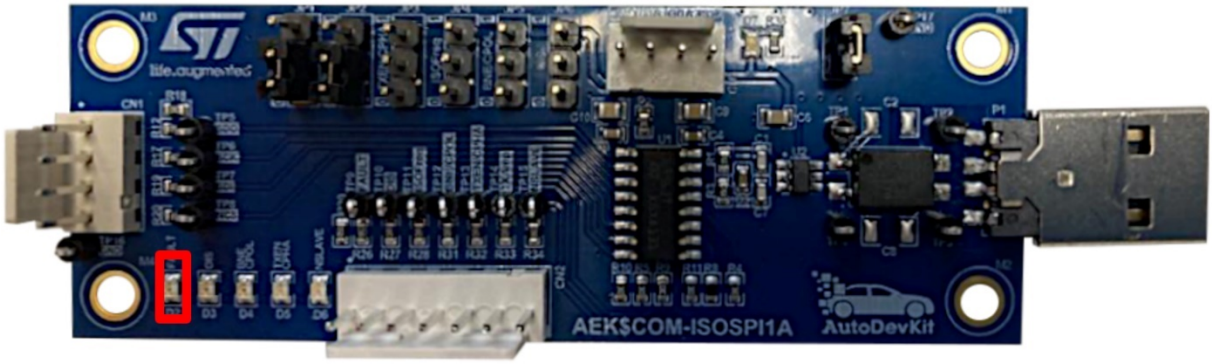
5.7 Die temperature diagnostic and overtemperature

An internal temperature sensor continuously monitors the temperature of the chip.

The chip prevents overheating through an overtemperature threshold TSD (which includes a hysteresis TSD_HY). Once the die temperature reaches TSD, a thermal shutdown circuit will force the chip to reduce the consumption by stopping balancing. A fault is reported to the microcontroller with a dedicated bit OTchip and propagated through the FAULT Line. When the temperature of the die returns to a normal level, L9963E can resume the normal operation. Balancing is released after the microcontroller reads OTchip latch.

6 Fault condition

Figure 12. **Fault LED on the AEK-COM-ISOSPI1**

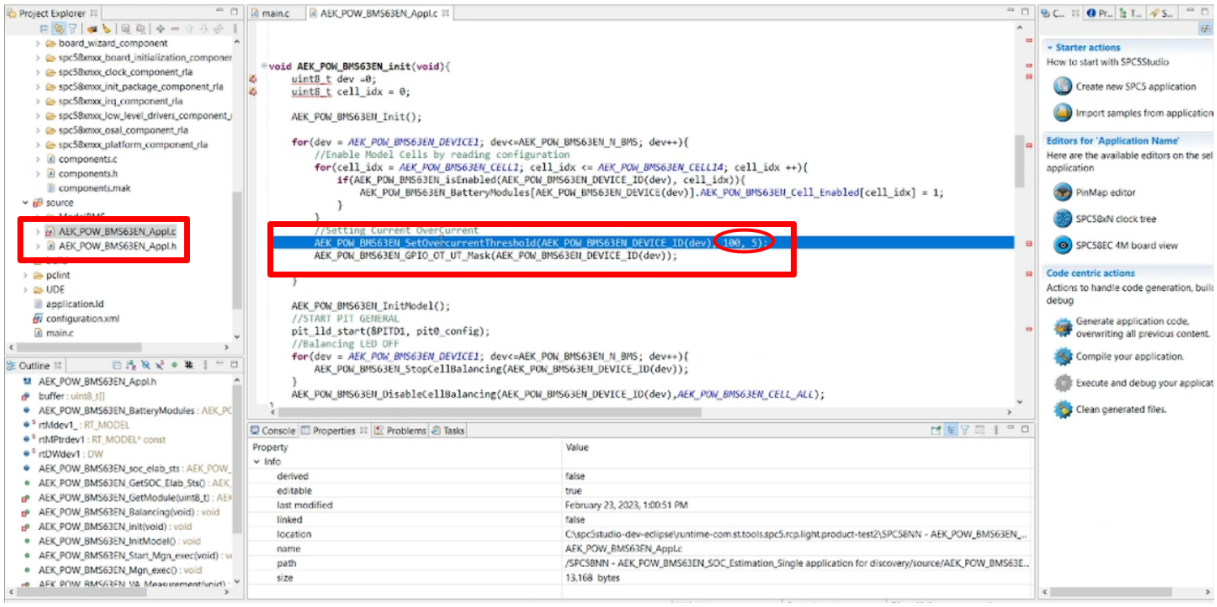


The fault LED on the AEK-COM-ISOSPI1 is related to the state of all the BMS nodes in the daisy chain. If an undervoltage, overvoltage, overcurrent, or overtemperature occurs on any cell of a BMS, a fault condition is detected. To solve this condition, diagnosis via software code must be activated.

The overcurrent detection is linked to a threshold defined in the application, not in the software driver. The threshold must be modified according to the load.

The image below shows an example of the function used to set the overcurrent threshold; e.g., 100 mΩ resistor and 5 A load.

Figure 13. **Function setup example for overcurrent threshold, resistor, and load values**



The overcurrent threshold can be set through the setOvercurrentThreeshold function inside the AEK_POW_BMS63EN_Init() define.

Figure 14. **AEK_POW_BMS63EN_Init() definition**

```
void AEK_POW_BMS63EN_init(void){
    uint8_t dev =0;
    uint8_t cell_idx = 0;

    AEK_POW_BMS63EN_Init();

    for(dev = AEK_POW_BMS63EN_DEVICE1; dev<=AEK_POW_BMS63EN_N_BMS; dev++){
        //Enable Model Cells by reading configuration
        for(cell_idx = AEK_POW_BMS63EN_CELL1; cell_idx <= AEK_POW_BMS63EN_CELL14; cell_idx ++){
            if(AEK_POW_BMS63EN_isEnabled(AEK_POW_BMS63EN_DEVICE_ID(dev), cell_idx)){
                AEK_POW_BMS63EN_BatteryModules[AEK_POW_BMS63EN_DEVICE(dev)].AEK_POW_BMS63EN_Cell_Enabled[cell_idx] = 1;
            }
        }
        //Setting Current OverCurrent
        AEK_POW_BMS63EN_SetOvercurrentThreshold(AEK_POW_BMS63EN_DEVICE_ID(dev), 100, p);
        AEK_POW_BMS63EN_GPIO_OT_UT_Mask(AEK_POW_BMS63EN_DEVICE_ID(dev));

    }
```

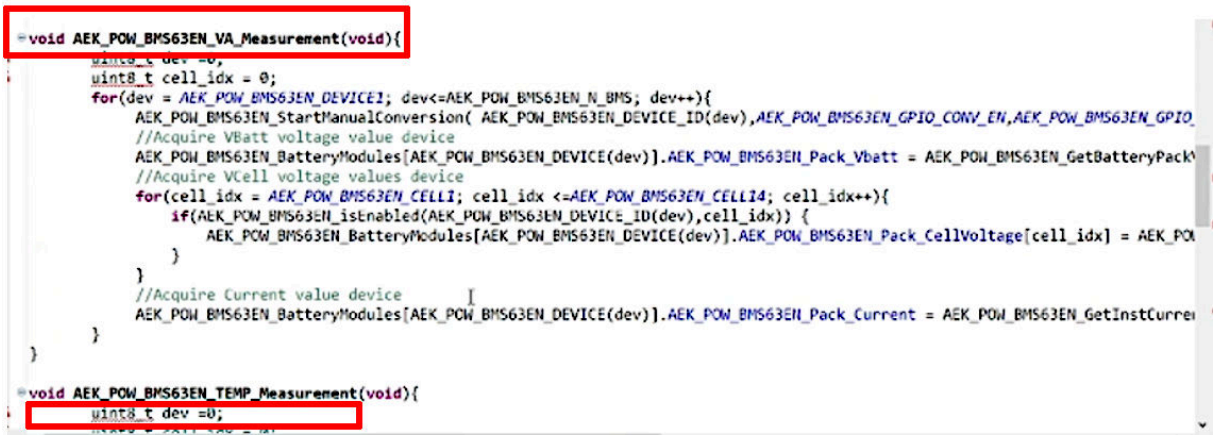
AEK_POW_BMS3EN_Init() is located in the driver lib of the file component (AEK_POW_BMS3EN_id.c).

Figure 15. **AEK_POW_BMS63EN_Init() location**

```
main.c | AEK_POW_BMS63EN_Appl.c | AEK_POW_BMS63EN_id.c

/* Module exported functions.                                        */
/*====================================================================*/

/* * * * * * * * * * * * * * * * * * * * */
/*   G E N E R A L   F U N C T I O N S   */
/* * * * * * * * * * * * * * * * * * * * */
/*
 *  AEK_POW_BMS63EN Driver initialization.
 */
void AEK_POW_BMS63EN_Init(void) {
  int i = 0;
  AEK_POW_BMS63EN_cfg_objects_init();
  AEK_POW_BMS63EN_Start(&AEK_POW_BMS63EN_Drv, &AEK_POW_BMS63EN_config);
    /* Enable Tranceiver and Trasmission*/
    for(i = 0; i<AEK_POW_BMS63EN_N_TRANCEIVER;i++){
        AEK_COM_ISOSPI_EnableTransceiver(i);
        AEK_COM_ISOSPI_EnableTransceiverComm(i);
        AEK_COM_ISOSPI_ISOTX_Attenuate(i);
        AEK_COM_ISOSPI_SetAsSlave(i);
    }

    /*StartUpWKUP Driver with all AEK_POW_BMS63ENE devices*/;
    AEK_POW_BMS63EN_StartUpWKUP();

    for(i = 0; i<AEK_POW_BMS63EN_N_TRANCEIVER;i++){
        AEK_COM_ISOSPI_ConfigISOFreq(i,AEK_POW_BMS63EN_Drv.param_conf[0].AEK_POW_BMS63EN_iso_freq);
    }

    AEK_POW_BMS63EN_ParameterConfiguration();
}
```

In the application AEK_POW_BMS3EN_Appl.c file, there is also a function that reads the current and temperature.

Figure 16. **Functions for current and temperature reading**



```
void AEK_POW_BMS63EN_VA_Measurement(void){
        uint8_t cell_idx = 0;
        for(dev = AEK_POW_BMS63EN_DEVICE1; dev<=AEK_POW_BMS63EN_N_BMS; dev++){
            AEK_POW_BMS63EN_StartManualConversion( AEK_POW_BMS63EN_DEVICE_ID(dev),AEK_POW_BMS63EN_GPIO_CONV_EN,AEK_POW_BMS63EN_GPIO_
            //Acquire VBatt voltage value device
            AEK_POW_BMS63EN_BatteryModules[AEK_POW_BMS63EN_DEVICE(dev)].AEK_POW_BMS63EN_Pack_Vbatt = AEK_POW_BMS63EN_GetBatteryPack
            //Acquire VCell voltage values device
            for(cell_idx = AEK_POW_BMS63EN_CELL1; cell_idx <=AEK_POW_BMS63EN_CELL14; cell_idx++){
                if(AEK_POW_BMS63EN_isEnabled(AEK_POW_BMS63EN_DEVICE_ID(dev),cell_idx)) {
                    AEK_POW_BMS63EN_BatteryModules[AEK_POW_BMS63EN_DEVICE(dev)].AEK_POW_BMS63EN_Pack_CellVoltage[cell_idx] = AEK_PO
                }
            }
            //Acquire Current value device
            AEK_POW_BMS63EN_BatteryModules[AEK_POW_BMS63EN_DEVICE(dev)].AEK_POW_BMS63EN_Pack_Current = AEK_POW_BMS63EN_GetInstCurre
        }
}

void AEK_POW_BMS63EN_TEMP_Measurement(void){
        uint8_t dev =0;
```

Figure 17. **Resistor value to modify**

```
AEK_POW_BMS63EN_GetInstCurrentMeasurement(AEK_POW_BMS63EN_DEVICE_ID(dev), 100);
```

7 Cell balancing

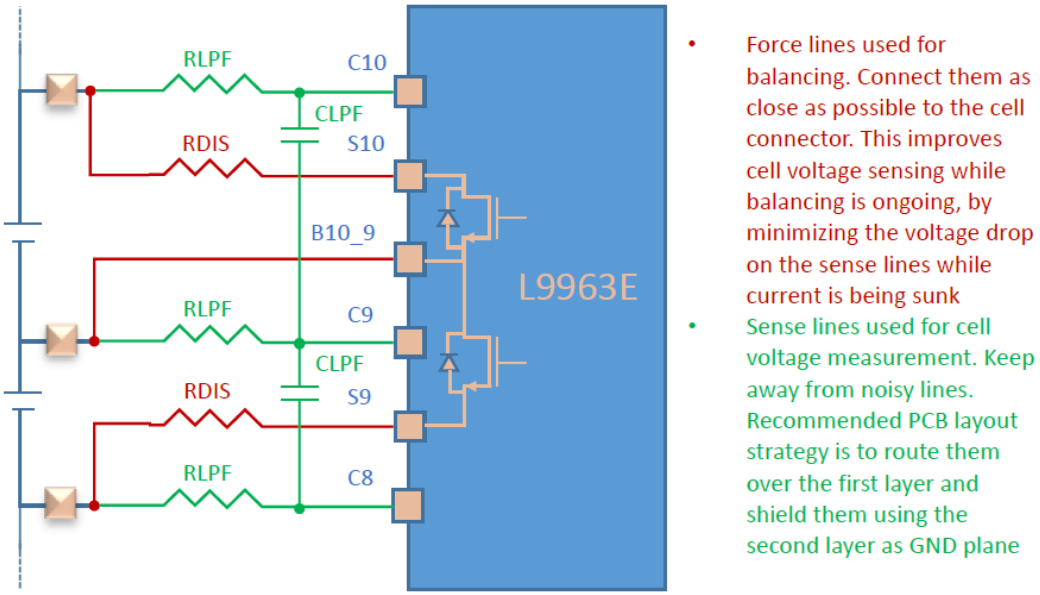
In the L9963E, the Sx and Bx_x-1 pins are used to balance the charge of the cells by discharging the ones with a higher SOC. Balancing can be performed either with external resistors or internal MOSFETs.

Cell balance drivers are powered by VBAT stack voltage. Hence, balancing is theoretically possible even at low cell voltages, except for cell 14. In case $V_{CELL14} < V_{CELL14\_BAL\_MIN}$, the corresponding balancing circuitry does not operate properly, and false overcurrent detection may occur.

7.1 Passive cell balancing with internal MOSFETs

The board is designed using internal MOSFETs.

Figure 18. Cell monitoring with internal balancing



The on-chip MOSFETs are switched on to sink a current from the cell, thus dissipating charge on RDIS. The affordable balancing current is restricted by the thermal relief on the current source circuits.

The maximum balance current on each cell is 200 mA. All cells can be balanced simultaneously, if the junction temperature does not exceed the maximum operating defined in the datasheet. To prevent thermal overstress, the die temperature diagnostic and overtemperature protections are implemented.

8 Methods for SOC estimation

The state of charge (SOC) of a cell indicates the energy stored by a cell, defined as the ratio of the charge quantity of a cell in a given moment and its total capacity.

The traditional calculation method is the coulomb counting.

According to this method, a charging/discharging current I(t) is integrated in time to calculate the residual energy quantity of the cell itself.

SOC(t) in a specific instant is calculated as the sum of the SOC related to the previous instant SOC(t-1) and the quantity of energy removed/added to that specific instant, as shown in the following formula:

$$SOC_{(t)} = SOC_{(t-1)} \cdot \int_{t-1}^{t} \frac{I_t \cdot \eta}{Q_n} \cdot dt$$

where:

- SOC(t) is the state of charge of the cell in the instant (t)
- SOC(t-1) is the state of charge of the cell in the previous instant (t-1)
- Q is the total capacity of the cell
- η is the cell efficiency

This method does not consider the speed, and hence the dynamics of the cell charge/discharge, temperature, hysteresis of charge/discharge and especially the cell age, which directly influences efficiency.

Moreover, this method is not accurate due to the error of integration and to the initial SOC estimation, and to the eventual inaccuracy of the cell current measurement.

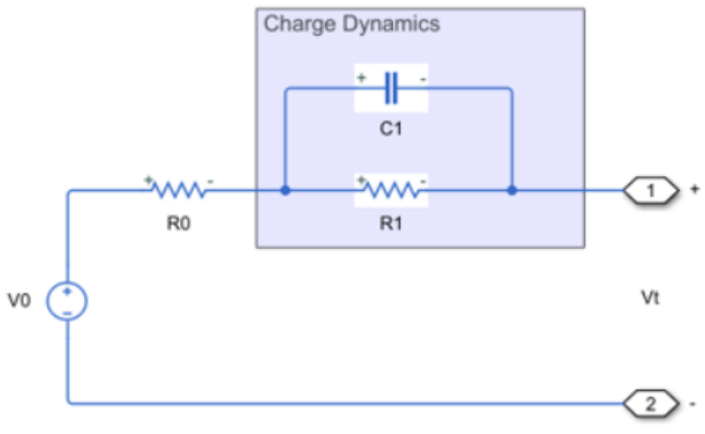
Other issues are related to the difficult estimation in case the cell undergoes an open circuit voltage variation (OCV with no load voltage) that is not significant in relation to the actual SOC (for example, in case LFP batteries are used).

The Kalman filter represents a promising alternative approach, which allows overcoming these obstacles through a higher effort in terms of computational resources.

This filter is based on a non-linear model of a cell through an equivalent circuit.

In our approach, we have used a first-grade equivalent circuit, as shown in the image below.

Figure 19. Equivalent circuit



In this circuit:

- Vo is the no-load voltage
- Vt is the voltage at the battery poles
- R0 represents the ohmic behavior of the cell
- R1C1 takes into account the dynamic behavior during charge and discharge

The equations that define the model are:

$$\frac{dSOC}{dt} = -\frac{i}{3600AH(T)}$$

$$\frac{dV_1}{dt} = \frac{i}{C_1(SOC,T)} - \frac{V_1}{R_1(SOC,T)\cdot C_1(SOC,T)}$$
$$V_t = V_0(SOC,T) - iR_0 - V_1$$

Where:

- SOC is the state of charge
- i is the current
- V_0 is the no-load voltage
- V_t is the terminal voltage
- AH is the ampere-hour rating
- R_1 is the first polarization resistance
- C_1 is the parallel RC capacitance
- T is the temperature

As opposed to the coulomb counting approach, this model considers the voltage at the battery poles and the cell capacity, as well as the temperatures and charge/discharge dynamics.

The equivalent circuit parameters used in the equations above must be estimated during the characterization phase.

Building the Kalman filter requires the definition of the state to estimate and the functions of process and observation, as shown below.

$$x = [SOC V_1]^T$$
$$f(x,i) = \begin{bmatrix} -\frac{i}{3600AH(t)} \\ -\frac{i}{C_1(SOC,T)} - \frac{V_1}{R_1(SOC,T)\cdot C_1(SOC,T)} \end{bmatrix}$$
$$h(x,i) = V_0(SOC,T) - iR_0 - V_1$$

The Kalman filter has been configured in Simulink according to the above functions.

The parameters (Vo, R0, C1, and R1) have been configured through lookup tables, after characterizing an INR 18650 MJ1 battery by LG.

The x status matrix (see above) estimated by the Kalman filter can define the SOC over time during the charge/discharge operations and at no load.

From the Kalman model in Simulink, we generated the C code through the embedded coder tool. Then, the code has been integrated into the L9963E drivers to estimate the SOC in real-time. For further details, see the related Mathworks page.

Another method to estimate the SOC is based on neural networks. This method requires a network training phase in all the operative conditions of the cell that belongs to the battery pack (during the charge/discharge operations and at no load).

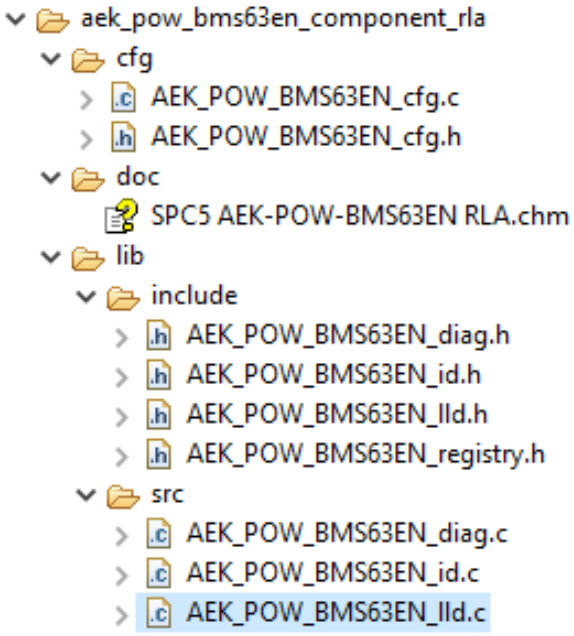
9 AutoDevKit ecosystem

The application development employing the AEK-POW-BMS63EN takes full advantage of the AutoDevKit ecosystem, whose basic components are:

- AutoDevKit Studio IDE (STSW-AUTODEVKIT)
- PLS UDE and OpenOCD programmers and debuggers

9.1 Component folder structure

Figure 20. AEK-POW-BMS63EN component folder structure



The cfg folder contains all the configuration files.

The doc folder contains the doxygen documentation.

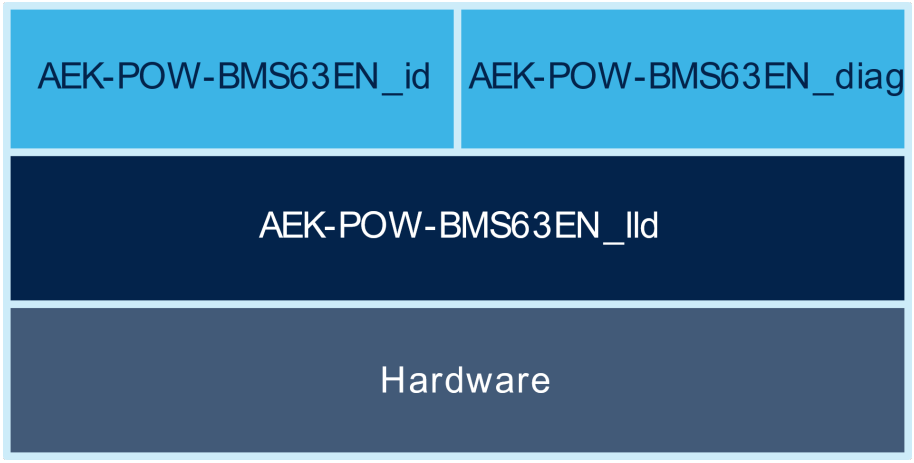
The lib folder contains the component header and source files.

9.2 Software component architecture (AEK-POW-BMS63EN component RLA)

The following image shows the architecture of the software components created for the AEK-POW-BMS63EN evaluation board, which consists of the following layers:

- AEK-POW-BMS63EN_lld
- AEK-POW-BMS63EN_id
- AEK-POW_BMS63EN_diag

Figure 21. **Software architecture**



AEK-POW-BMS63EN_lld contains all the APIs that read and write on the L9963E through the AEK-COM-ISOSPI1 in single access mode and burst access mode (refer to the datasheet for further information).

AEK-POW-BMS63EN_id contains the APIs for the addressing procedure, in centralized and dual access ring configuration, and for the AEK-COM-ISOSPI1 in terms of signal amplitude and frequency. It also contains the APIs to start the voltage conversion routine, the coulomb counting routine, and the passive balancing. The code generated for the SOC estimation communicates with this layer, which provides the voltage, current, and temperature measurements to the generated C code, after the routines are configured on demand. The model for the SOC estimation is not part of the component but it is considered as a layer linked to the application defined by the user. In fact, it is possible to replace our SOC model with others customized by the user.

AEK-POW_BMS63EN_diag contains the APIs for diagnostic functions.

9.3 AEK-POW-BMS63EN in AutoDevKit

9.3.1 Centralized configuration example

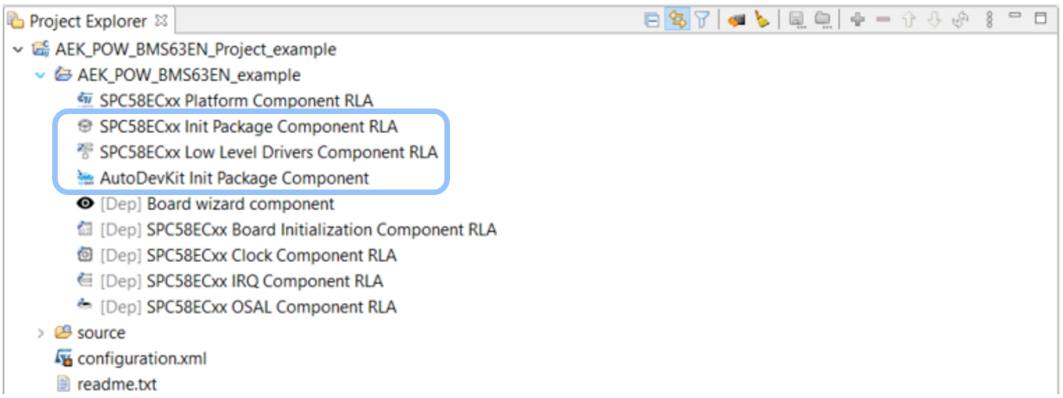
This example is an application for the AEK-POW-BMS63EN in a centralized daisy chain topology using the AEK-MCU-C4MLIT1 as the microcontroller board. To recreate this scenario, follow the procedure below.

Step 1. Create a new SPC5-STUDIO application for the SPC58EC series microcontroller and add the following components:

- SPC58ECxx Init Package Component RLA
- SPC58ECxx Low Level Drivers Component RLA
- AutoDevKit Init Package Component

These components must be added immediately, or the other components will not be visible.

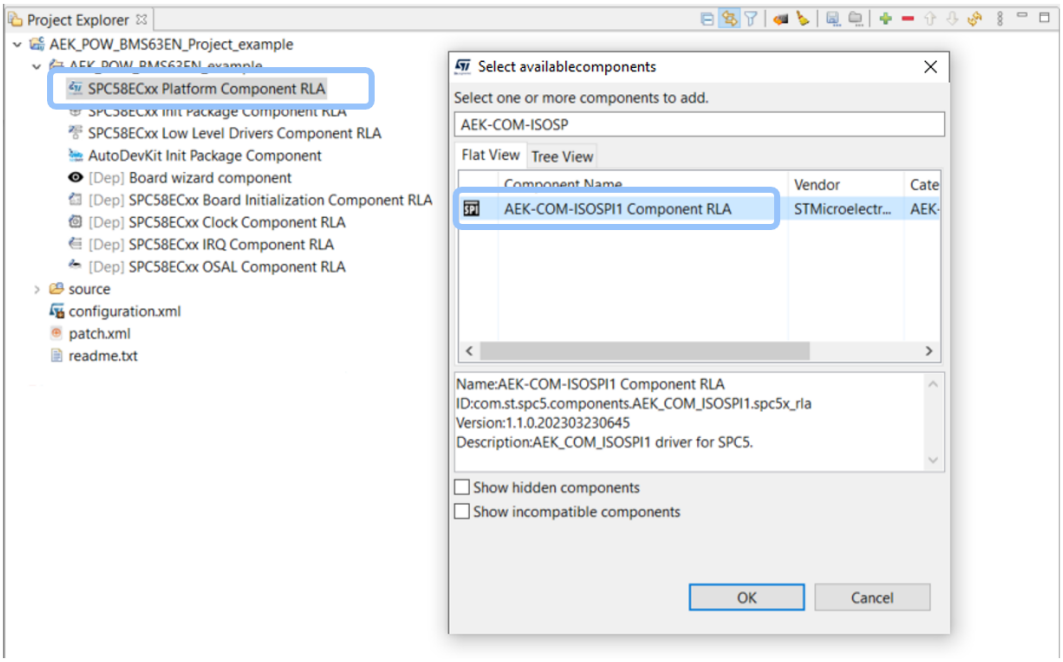
Figure 22. **Adding components**



Step 2. Add the following additional components:

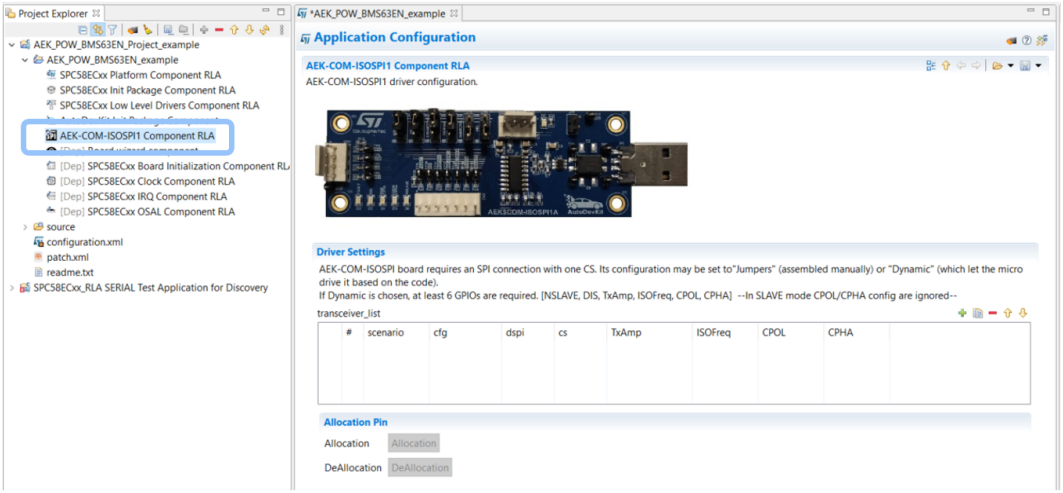
– AEK-COM-ISOSPI1 Component RLA

Figure 23. Adding AEK-COM-ISOSPI1 Component RLA



Step 3. Select [**AEK-COM-ISOSPI1 Component RLA**] to open the [**Application Configuration**] window.

Figure 24. Selecting AEK-COM-ISOSPI1 Component RLA

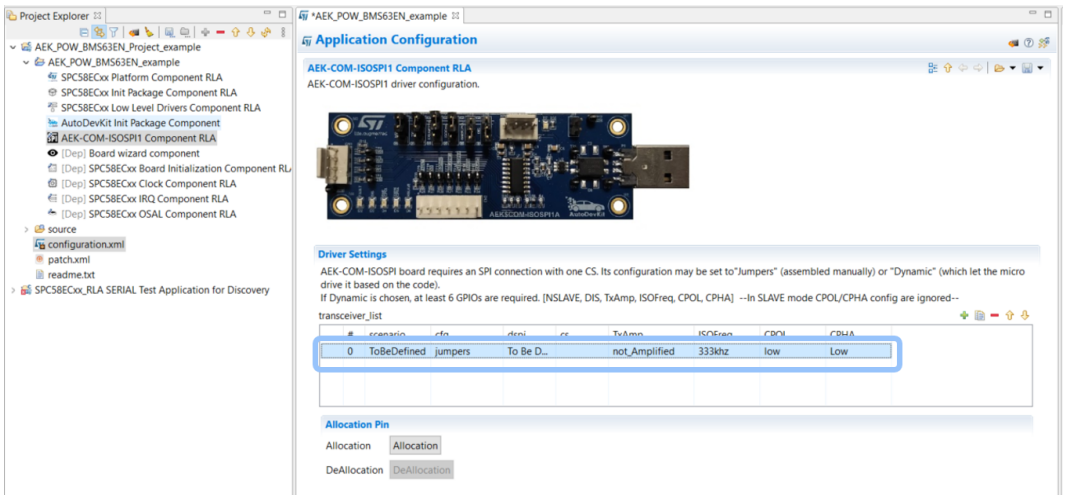


Step 4. Click on [+] to add a new element to the board list.

Figure 25. Adding a new element

Step 5. Double-click on the newly added element to configure the board.

Figure 26. AEK-COM-ISOPI1 configuration



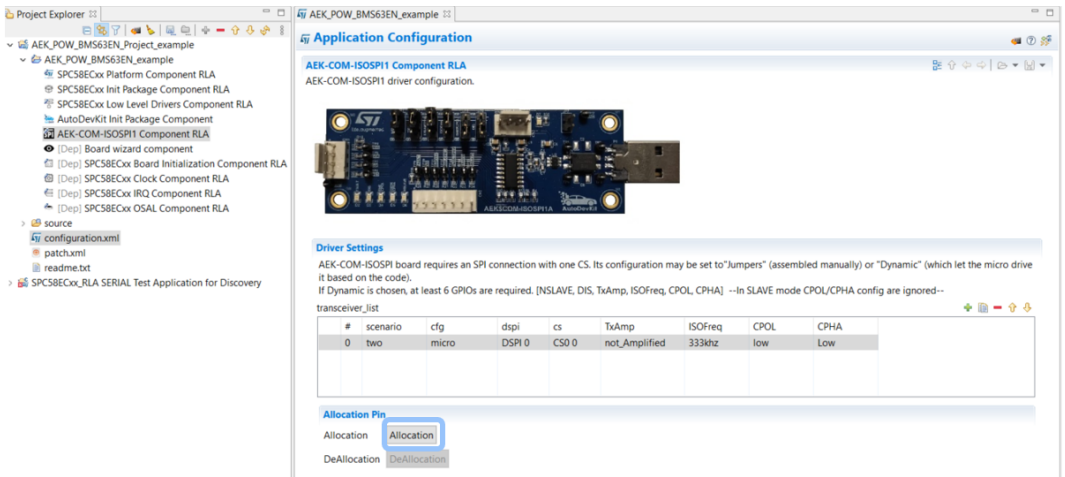
Step 6. Configure parameters

- Select scenario [**two**]
- Select [**DSPI**] protocol number and [**CS**] chip select
- Select the [**cfg**] configuration as [**micro**]
- Select [**TxAmp**] as [**not amplified**]
- Select [**ISOFreq**] as [**333 KHz**]
- [**Select Master Clock Frequency**] as [**5 MHz**]
- Select [**CPOL**] and [**CPHA**] as [**low**]

Figure 27. AEK-COM-ISOPI1 scenario two configurations

Step 7. Click on the [**Allocation**] button to allocate AEK-POW-ISOSPI1 component.

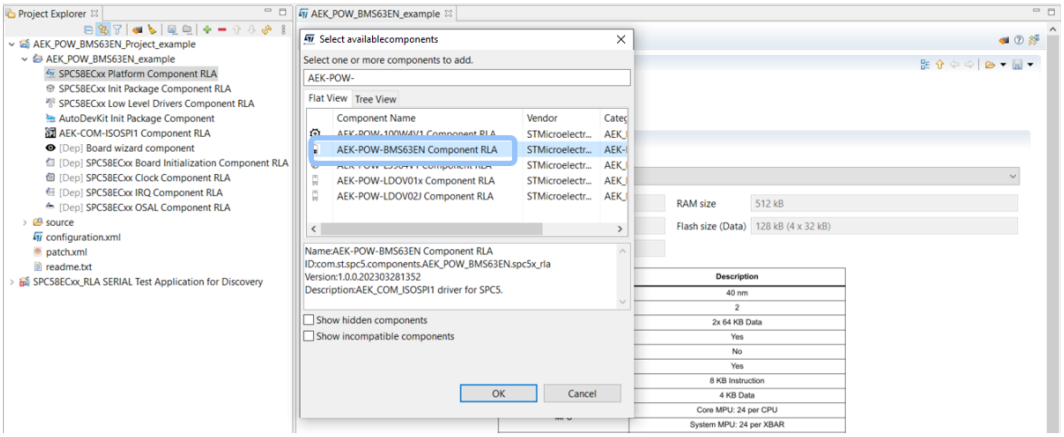
Figure 28. Component allocation



Step 8. Add the following additional components:

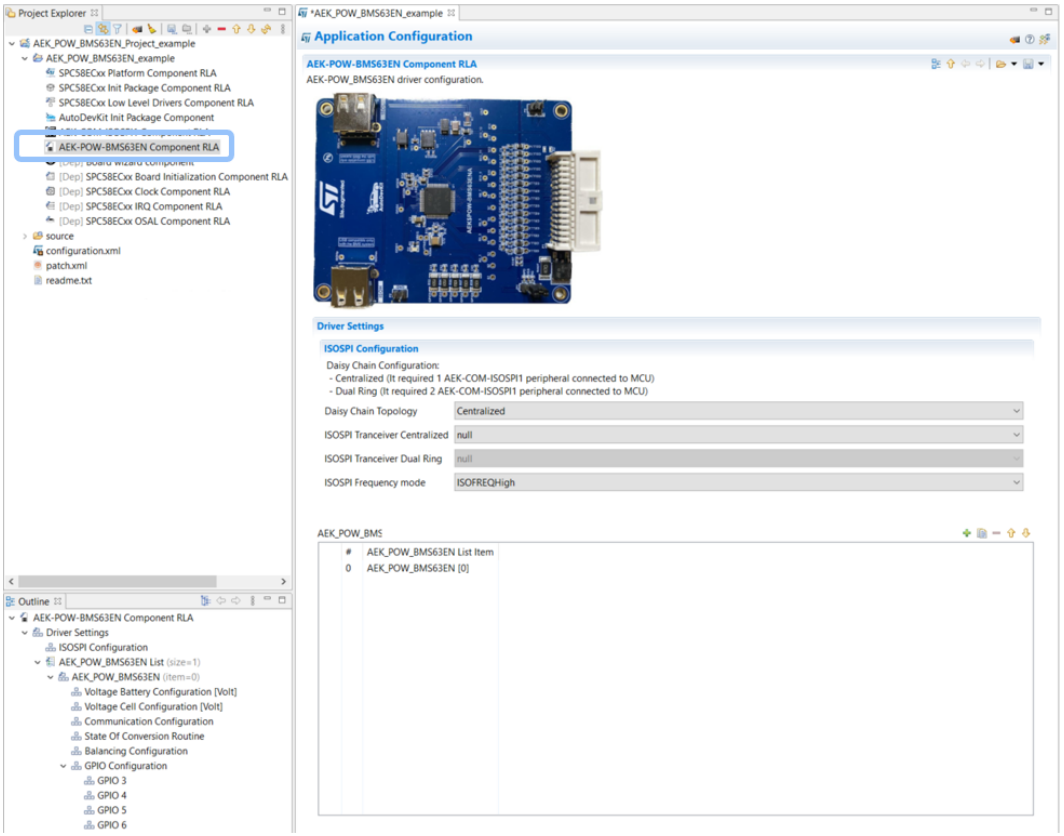
– AEK-POW-BMS63EN Component RLA

Figure 29. Adding AEK-POW-BMS63EN Component RLA



Step 9. Select [**AEK-POW-BMS63EN Component RLA**] to open the [**Application Configuration**] window.

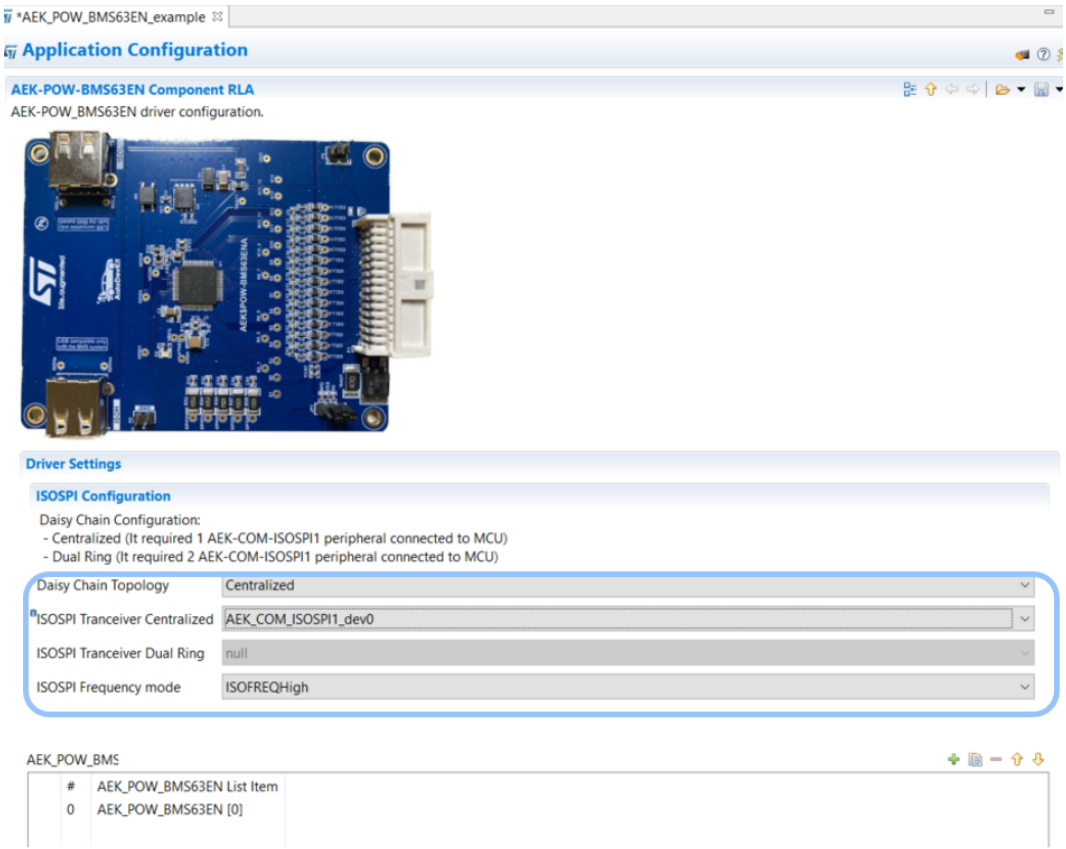
Figure 30. Selecting AEK-POW-BMS63EN Component RLA



Step 10. Select ISOSPI configuration settings:

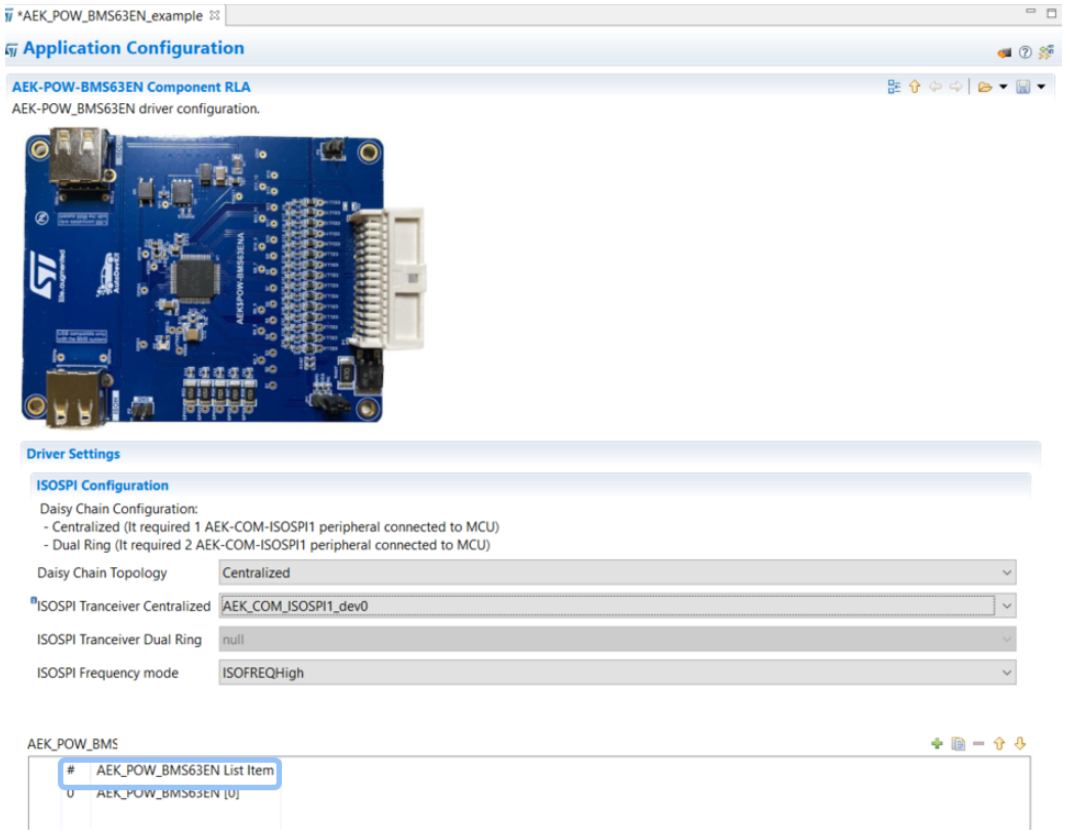
- Select [**Daisy Chain Topology**] as [**Centralized**].
- Select the already allocated in the [**ISOSPI Transceiver Centralized**] field.
- Select [**ISOSPI Frequency Mode**] as [**ISOFREQHigh**].

Figure 31. ISOSPI configuration



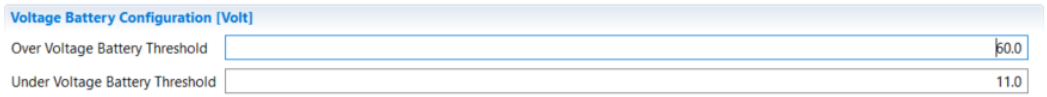
Step 11. Click on AEK_POW_BMS63EN list item to configure each component allocated.

Figure 32. Component configuration



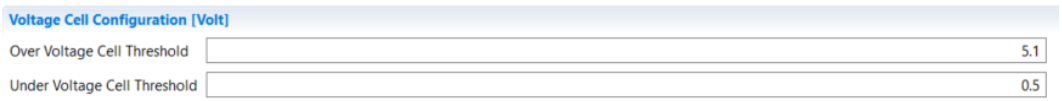
Step 12. Configure Over/Under Voltage Battery Thresholds.

Figure 33. Over/undervoltage battery threshold configuration



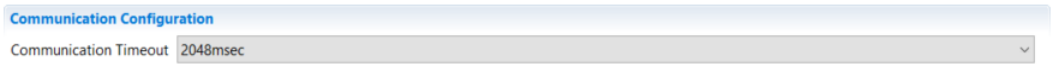
Step 13. Configure Over/Under Voltage Cell Thresholds.

Figure 34. Over/undervoltage cell threshold configuration



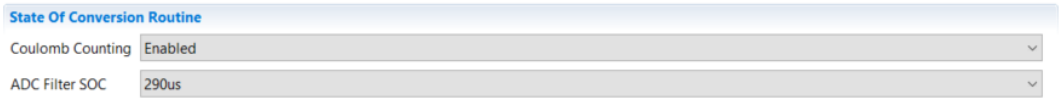
Step 14. Configure Communication timeout value with AEK-POW-ISOSPI1.

Figure 35. Communication configuration



Step 15. Configure Coulomb counting to Enable Current measurement and ADC Filter SOC timing.

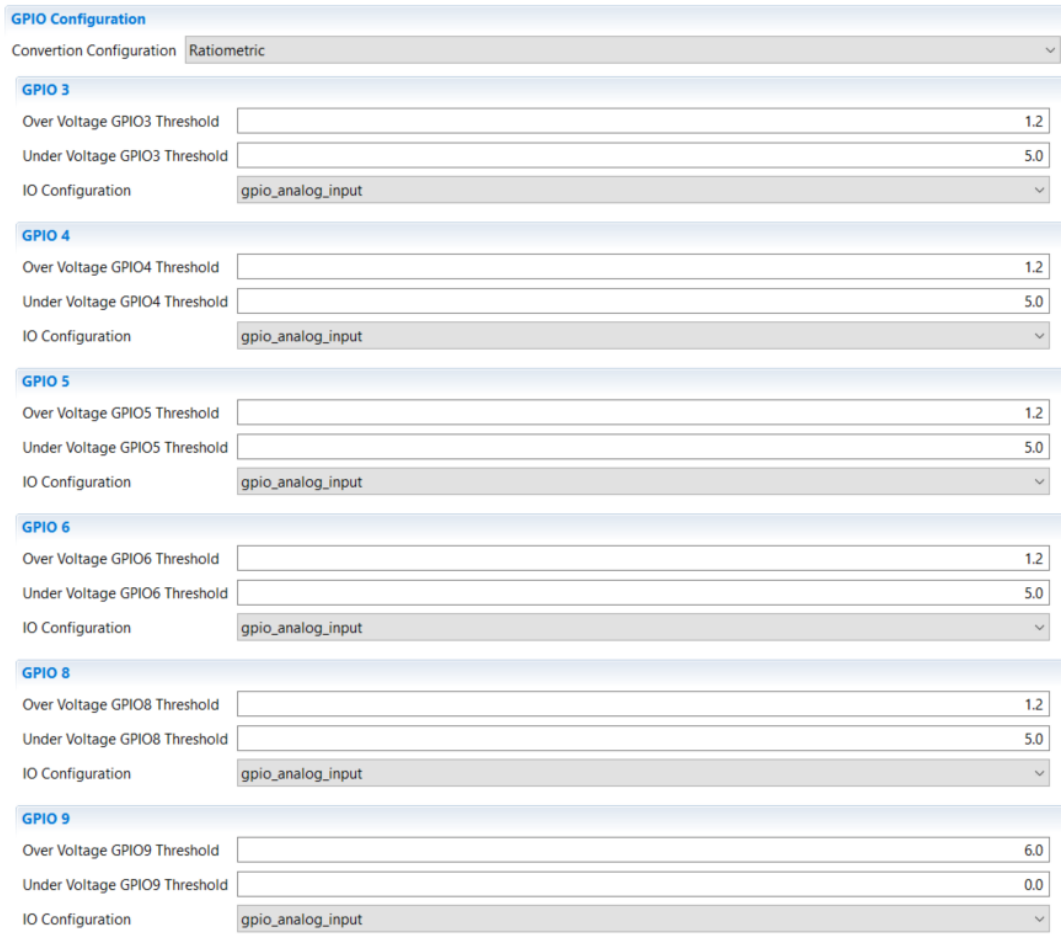
Figure 36. Coulomb counting and ADC filter SOC configuration



Step 16. Configure the GPIOs:

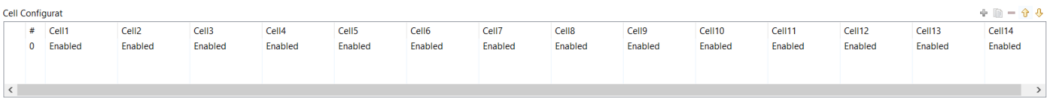
– Configure Conversion configuration for the analog input GPIO.
– Configure Over/Under voltage thresholds.
– Configure IO mode for each GPIO.

Figure 37. GPIO configuration



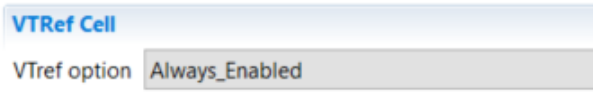
Step 17. Enable/Disable Cells AEK-POW-BMS63EN.

Figure 38. Enabling/disabling cells



Step 18. Check that the VTref option is always enabled.

Figure 39. VTref configuration



Step 19. Include AEK_POW_BMS63EN_id.h in your main application and write your application code.

Step 20. Click on [**Board View**] to display the hardware connection between the AEK-MCU-C4MLIT1 board and the AEK-COM-ISOSP1.

Figure 40. Available editors

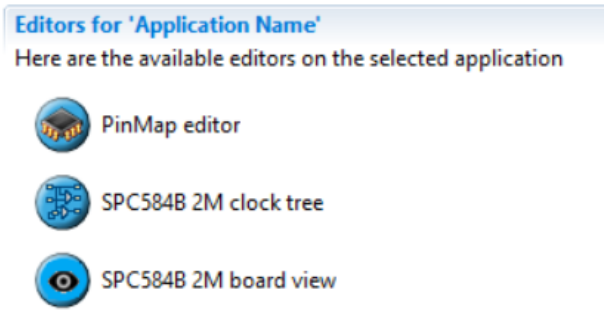
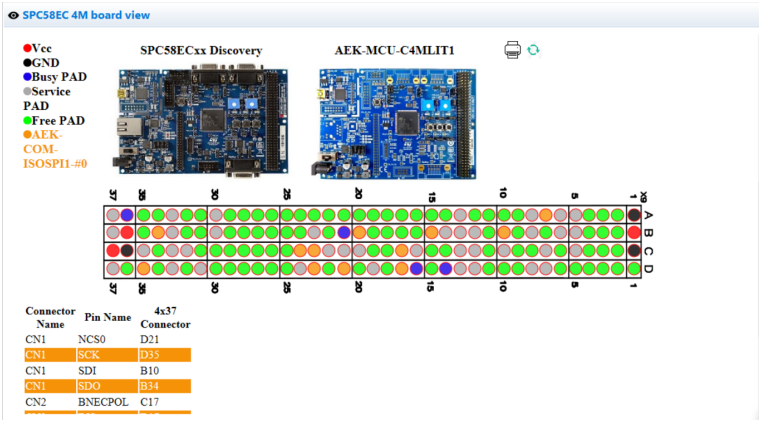
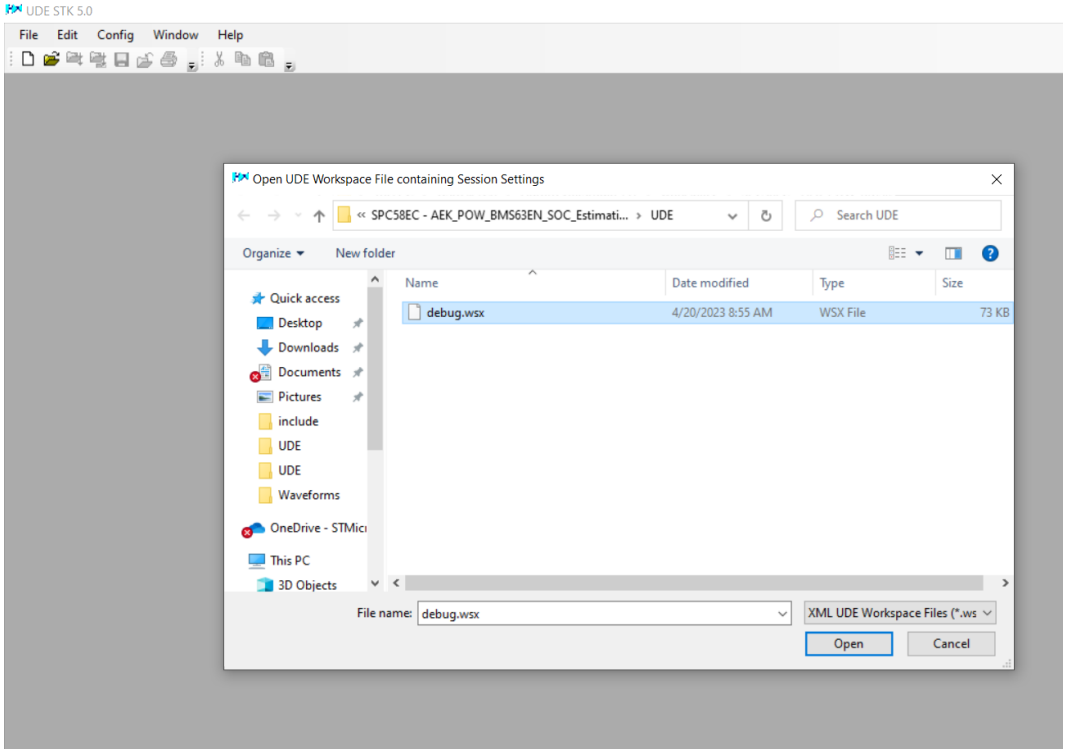


Figure 41. Board view



Step 21. Open [**UDE Starterkit**] and import the ". wsx" file from your application workspace to flash your application.

Figure 42. wsx file import



Step 22. Switch off the AEK-MCU-C4MLIT1 and connect the battery pack to the board.

Step 23. Switch on the AEK-MCU-C4MLIT1 to start your BMS application.

Note: If you want to define a centralized daisy chain configuration with more than one AEK-POW-BMS63EN, it is necessary to allocate more than one component in step 11.

9.3.2 Dual access ring configuration example

This example creates an application for the AEK-POW-BMS63EN in dual access ring topology, using AEK-MCU-C4MLIT1 as the microcontroller board.

To recreate this scenario, follow the procedure described in Centralized configuration example, except for step 10, where you must select [**Dual Ring**] as the daisy chain topology.

9.4 Available demos for AEK-POW-BMS63EN

In the AutoDevKit starting from release 2.1.0, there are several available demos for the AEK-POW-BMS63EN:

- SPC58EC – AEK_POW_BMS63EN_SOC_Estimation_Single application for discovery: a demo application for the AEK-MCU-C4MLIT1 for state of charge estimation with one AEK-POW-BMS63EN in centralized daisy chain topology.
- SPC58EC – AEK_POW_BMS63EN_SOC_Estimation_Centralized application for discovery: a demo application for the AEK-MCU-C4MLIT1 for state of charge estimation with three AEK-POW-BMS63EN in a centralized daisy chain topology.
- SPC58EC – AEK_POW_BMS63EN_SOC_Estimation_DualRing application for discovery: a demo application for the AEK-MCU-C4MLIT1 for state of charge estimation with two AEK-POW-BMS63EN in a dual ring daisy chain topology.
- SPC584B – AEK_POW_BMS63EN_SOC_Estimation_Single application for discovery: a demo application for the SPC584B discovery board for state of charge estimation with one AEK-POW-BMS63EN in a centralized daisy chain topology.

- SPC582B – AEK_POW_BMS63EN_SOC_Estimation_Single application for discovery: a demo application for the AEK-MCU-C1MLIT1 for state of charge estimation with one AEK-POW-BMS63EN in a centralized daisy chain topology.
- SPC58NN – AEK_POW_BMS63EN_SOC_Estimation_Single application for discovery: a demo application for the SPC58NN (Bernina) discovery board for state of charge estimation with one AEK-POW-BMS63EN in a centralized daisy chain topology.

9.4.1 Centralized application (SPC58EC – AEK_POW_BMS63EN_SOC_Estimation_Centralized application for discovery demo)

This application is based on the SPC58EC microcontroller and can estimate the SOC of 3x14 cells (in our case, the battery used are INR 18650 MJ1 by LG) connected with 3 AEK-POW-BMS63EN evaluation boards.

The results of SOC estimation, cell voltage, NTC temperatures, and current are printed via a serial port with a speed rate of 115200 bps.

Each AEK-POW-BMS63EN is driven by one AEK-COM-ISOSPI1 in a centralized daisy chain topology.

Below is the main code for the SPC58EC – AEK_POW_BMS63EN_SOC_Estimation_Centralized application for discovery demo:

```
/*****************************************************************************
 *
 * Copyright Â© 2022 STMicroelectronics - All Rights Reserved
 *
 * License terms: STMicroelectronics Proprietary in accordance with licensing
 * terms SLA0089 at www.st.com.
 *
 * THIS SOFTWARE IS DISTRIBUTED "AS IS," AND ALL WARRANTIES ARE DISCLAIMED,
 * INCLUDING MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
 *
 * EVALUATION ONLY â€“ NOT FOR USE IN PRODUCTION
 *****************************************************************************/
/* Inclusion of the main header files of all the imported components in the
 order specified in the application wizard. The file is generated
 automatically.*/
#include "components.h"
#include "AEK_POW_BMS63EN_Appl.h"
#include "wkpu_lld_cfg.h"
#include "serial_lld_cfg.h"
#include <stdio.h>
#include <string.h>
volatile uint8_t i_device_disp=0;
void main_core0(void) {
  char message[11];
  AEK_POW_BMS63EN_module_t AEK_POW_BMS63EN_BatteryModule;
  uint8_t cell_idx = 0;
  /* Enable Interrupts */
  irqIsrEnable();
  sd_lld_start(&SD5,&serial_config_BMS_serial);
  /* Application main loop.*/
  for ( ; ; ) {
  if((osalThreadGetMilliseconds()%100)==0){
    AEK_POW_BMS63EN_BatteryModule = AEK_POW_BMS63EN_GetModule(i_device_disp);
//SOC Elaboration Done
sprintf(message, "DEV:%d     \n", (int)(i_device_disp+1));
sd_lld_write(&SD5, (uint8_t *)message, (uint16_t)(sizeof(message)/sizeof(message[0])));
//Printing SOC
for(cell_idx = AEK_POW_BMS63EN_CELL1; cell_idx <= AEK_POW_BMS63EN_CELL14; cell_idx ++){
sprintf(message, "S%.2d:%.3d    ", cell_idx+1, (int)((AEK_POW_BMS63EN_BatteryModule.AEK_POW_B
MS63EN_Pack_SOC[cell_idx]) * 100));
sd_lld_write(&SD5, (uint8_t *)message, (uint16_t)(sizeof(message)/sizeof(message[0])));
}
sprintf(message,"          \n");
sd_lld_write(&SD5, (uint8_t *)message, (uint16_t)(sizeof(message)/sizeof(message[0])));
//Printing Bal
for(cell_idx = AEK_POW_BMS63EN_CELL1; cell_idx <= AEK_POW_BMS63EN_CELL14; cell_idx ++){
sprintf(message, "B%.2d:%.3d    ", cell_idx+1, AEK_POW_BMS63EN_BatteryModule.AEK_POW_BMS63EN_
Pack_Bal_cmd[cell_idx]);
sd_lld_write(&SD5, (uint8_t *)message, (uint16_t)(sizeof(message)/sizeof(message[0])));
}
```

```
sprintf(message,"          \n");
sd_lld_write(&SD5, (uint8_t *)message, (uint16_t)(sizeof(message)/sizeof(message[0])));
//Printing Voltage
for(cell_idx = AEK_POW_BMS63EN_CELL1; cell_idx <= AEK_POW_BMS63EN_CELL14; cell_idx ++){
sprintf(message, "V%.2d:%.3f  ", cell_idx+1, AEK_POW_BMS63EN_BatteryModule.AEK_POW_BMS63EN_Pa
ck_CellVoltage[cell_idx]);
sd_lld_write(&SD5, (uint8_t *)message, (uint16_t)(sizeof(message)/sizeof(message[0])));
}
sprintf(message,"          \n");
sd_lld_write(&SD5, (uint8_t *)message, (uint16_t)(sizeof(message)/sizeof(message[0])));
//Printing Current
sprintf(message, "C:%.4f    \n", AEK_POW_BMS63EN_BatteryModule.AEK_POW_BMS63EN_Pack_Current);
sd_lld_write(&SD5, (uint8_t *)message, (uint16_t)(sizeof(message)/sizeof(message[0])));
}
  }
}
void *sbrk(size_t incr)
{
extern uint8_t __heap_base__;
extern uint8_t __heap_end__;
static uint8_t *p=&__heap_base__;
static uint8_t *newp;
newp = p+ incr;
if(newp> &__heap_end__)
{
return (void*)-1;
}
return p =newp;
}
void int28_irq_cb(WKPUDriver *wkpup)
{
(void)wkpup;
i_device_disp = i_device_disp+1;
if (i_device_disp==AEK_POW_BMS63EN_N_BMS) i_device_disp=0;
}
int main(void) {
componentsInit();
irqIsrEnable();
wkpu_lld_start(&WKPUD1, &wkpu_config_wkpu_cfg);
AEK_POW_BMS63EN_init();
runCore0();
AEK_POW_BMS63EN_Start_Mgn_exec();
while (1) {
}
}
```

This application enables two cores:

- Core2 (main core)
  - Used to start the manager process (AEK_POW_BMS63EN_Mng_exec()) by using a PIT (i.e., a 5 Hz programmable timer):

```
void AEK_POW_BMS63EN_Mgn_exec(){
pal_lld_togglepad(PORT_F,LED3);
AEK_POW_BMS63EN_soc_elab_sts = AEK_POW_BMS63EN_SOC_ELABORATION_BUSY;
AEK_POW_BMS63EN_VA_Measurement();
AEK_POW_BMS63EN_TEMP_Measurement();
if(AEK_COM_ISOSPI_GetFault(ISOSPI_DEV0)){
AEK_POW_BMS63EN_DIAG_Measurement();
}
AEK_POW_BMS63EN_Model_exec();
AEK_POW_BMS63EN_Balancing();
AEK_POW_BMS63EN_soc_elab_sts = AEK_POW_BMS63EN_SOC_ELABORATION_DONE;
pal_lld_togglepad(PORT_F,LED3);
}
```

This process starts the manual conversion routine to measure cell voltage, cell current, and GPIO NTC temperature. It also performs the state of charge estimation by using the C-code generated through a Simulink model and enables the manual balancing during the Normal state of the FSM.

```
void int28_irq_cb(WKPUDriver *wkpup)
{
(void)wkpup;
i_device_disp = i_device_disp+1;
if (i_device_disp==AEK_POW_BMS63EN_N_BMS) i_device_disp=0;
}
```

- Core0 (auxiliary core)
  - Used to start a process to print via serial the following data (with a speed rate of 115200 bps):
    - State of charge percentage for each cell
    - Cell voltage values
    - Current value
    - Balancing status
  - This process is called every 100 ms.

9.4.2 Dual ring application (SPC58EC – AEK_POW_BMS63EN_SOC_Estimation_DualRing application for discovery demo)

This application is based on the SPC58EC microcontroller and can estimate the SOC of 1x14 cells (in our case, the battery used are INR 18650 MJ1 by LG) connected with 2 AEK-POW-BMS63EN evaluation boards.

The results of SOC estimation, cell voltage, NTC temperatures, and current are printed via serial protocol with a speed rate of 115200 bps.

Each AEK-POW-BMS63EN is driven by one AEK-COM-ISOSPI1 in a dual ring daisy chain topology.

Find below the main code for the SPC58EC – AEK_POW_BMS63EN_SOC_Estimation_DualRing application for discovery demo:

```
/*****************************************************************************
 *
 * Copyright Â© 2022 STMicroelectronics - All Rights Reserved
 *
 * License terms: STMicroelectronics Proprietary in accordance with licensing
 * terms SLA0089 at www.st.com.
 *
 * THIS SOFTWARE IS DISTRIBUTED "AS IS," AND ALL WARRANTIES ARE DISCLAIMED,
 * INCLUDING MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
 *
 * EVALUATION ONLY â€" NOT FOR USE IN PRODUCTION
 *****************************************************************************/
/* Inclusion of the main header files of all the imported components in the
 order specified in the application wizard. The file is generated
 automatically.*/
#include "components.h"
#include "AEK_POW_BMS63EN_Appl.h"
#include "wkpu_lld_cfg.h"
#include "serial_lld_cfg.h"
#include <stdio.h>
#include <string.h>
volatile uint8_t i_device_disp=0;
void main_core0(void) {
  char message[11];
  AEK_POW_BMS63EN_module_t AEK_POW_BMS63EN_BatteryModule;
  uint8_t cell_idx = 0;
  /* Enable Interrupts */
  irqIsrEnable();
  sd_lld_start(&SD5,&serial_config_BMS_serial);
  /* Application main loop.*/
  for ( ; ; ) {
  if((osalThreadGetMilliseconds()%100)==0){
    AEK_POW_BMS63EN_BatteryModule = AEK_POW_BMS63EN_GetModule(i_device_disp);
//SOC Elaboration Done
sprintf(message, "DEV:%d     \n", (int)(i_device_disp+1));
sd_lld_write(&SD5, (uint8_t *)message, (uint16_t)(sizeof(message)/sizeof(message[0])));
//Printing SOC
for(cell_idx = AEK_POW_BMS63EN_CELL1; cell_idx <= AEK_POW_BMS63EN_CELL14; cell_idx ++){
sprintf(message, "S%.2d:%.3d    ", cell_idx+1, (int)((AEK_POW_BMS63EN_BatteryModule.AEK_POW_B
MS63EN_Pack_SOC[cell_idx]) * 100));
```

```
sd_lld_write(&SD5, (uint8_t *)message, (uint16_t)(sizeof(message)/sizeof(message[0])));
}
sprintf(message,"          \n");
sd_lld_write(&SD5, (uint8_t *)message, (uint16_t)(sizeof(message)/sizeof(message[0])));
//Printing Bal
for(cell_idx = AEK_POW_BMS63EN_CELL1; cell_idx <= AEK_POW_BMS63EN_CELL14; cell_idx ++){
sprintf(message, "B%.2d:%.3d    ", cell_idx+1, AEK_POW_BMS63EN_BatteryModule.AEK_POW_BMS63EN_
Pack_Bal_cmd[cell_idx]);
sd_lld_write(&SD5, (uint8_t *)message, (uint16_t)(sizeof(message)/sizeof(message[0])));
}
sprintf(message,"          \n");
sd_lld_write(&SD5, (uint8_t *)message, (uint16_t)(sizeof(message)/sizeof(message[0])));
//Printing Voltage
for(cell_idx = AEK_POW_BMS63EN_CELL1; cell_idx <= AEK_POW_BMS63EN_CELL14; cell_idx ++){
sprintf(message, "V%.2d:%.3f  ", cell_idx+1, AEK_POW_BMS63EN_BatteryModule.AEK_POW_BMS63EN_Pa
ck_CellVoltage[cell_idx]);
sd_lld_write(&SD5, (uint8_t *)message, (uint16_t)(sizeof(message)/sizeof(message[0])));
}
sprintf(message,"          \n");
sd_lld_write(&SD5, (uint8_t *)message, (uint16_t)(sizeof(message)/sizeof(message[0])));
//Printing Current
sprintf(message, "C:%.4f    \n", AEK_POW_BMS63EN_BatteryModule.AEK_POW_BMS63EN_Pack_Current);
sd_lld_write(&SD5, (uint8_t *)message, (uint16_t)(sizeof(message)/sizeof(message[0])));
}
  }
}
void *sbrk(size_t incr)
{
extern uint8_t __heap_base__;
extern uint8_t __heap_end__;
static uint8_t *p=&__heap_base__;
static uint8_t *newp;
newp = p+ incr;
if(newp> &__heap_end__)
{
return (void*)-1;
}
return p =newp;
}
void int28_irq_cb(WKPUDriver *wkpup)
{
(void)wkpup;
i_device_disp = i_device_disp+1;
if (i_device_disp==AEK_POW_BMS63EN_N_BMS) i_device_disp=0;
}
int main(void) {
componentsInit();
irqIsrEnable();
wkpu_lld_start(&WKPUD1, &wkpu_config_wkpu_cfg);
AEK_POW_BMS63EN_init();
runCore0();
AEK_POW_BMS63EN_Start_Mgn_exec();
while (1) {
if((osalThreadGetMicroseconds()%10000)==0)
AEK_POW_BMS63EN_DualRingTX_reverse();
}
}
```

This application is the same as the previous one (centralized topology), except that every 10000 microseconds, the ISOSPI communication is reversed in the opposite direction.

9.4.3 Single centralized application (SPC58EC – AEK_POW_BMS63EN_SOC_Estimation_Single application for discovery demo)

This application is based on the SPC58EC microcontroller and can estimate the SOC of 14 cells in a single BMS node (in our case, the battery used are INR 18650 MJ1 by LG) connected with an AEK-POW-BMS63EN evaluation board.

The results of SOC estimation, cell voltage, NTC temperatures, and current are printed via serial port with a speed rate of 115200 bps.

Find below the main code for the SPC58EC – AEK_POW_BMS63EN_SOC_Estimation_Single application for discovery demo:

```
*
 * Copyright Â© 2022 STMicroelectronics - All Rights Reserved
 *
 * License terms: STMicroelectronics Proprietary in accordance with licensing
 * terms SLA0089 at www.st.com.
 *
 * THIS SOFTWARE IS DISTRIBUTED "AS IS," AND ALL WARRANTIES ARE DISCLAIMED,
 * INCLUDING MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
 *
 * EVALUATION ONLY â€" NOT FOR USE IN PRODUCTION
 *****************************************************************************/
/* Inclusion of the main header files of all the imported components in the
 order specified in the application wizard. The file is generated
 automatically.*/
#include "components.h"
#include "AEK_POW_BMS63EN_Appl.h"
#include "wkpu_lld_cfg.h"
#include "serial_lld_cfg.h"
#include <stdio.h>
#include <string.h>
volatile uint8_t i_device_disp=0;
void main_core0(void) {
  char message[11];
  AEK_POW_BMS63EN_module_t AEK_POW_BMS63EN_BatteryModule;
  uint8_t cell_idx = 0;
  /* Enable Interrupts */
  irqIsrEnable();
  sd_lld_start(&SD5,&serial_config_BMS_serial);
  /* Application main loop.*/
  for ( ; ; ) {
  if((osalThreadGetMilliseconds()%100)==0){
    AEK_POW_BMS63EN_BatteryModule = AEK_POW_BMS63EN_GetModule(i_device_disp);
//SOC Elaboration Done
sprintf(message, "DEV:%d     \n", (int)(i_device_disp+1));
sd_lld_write(&SD5, (uint8_t *)message, (uint16_t)(sizeof(message)/sizeof(message[0])));
//Printing SOC
for(cell_idx = AEK_POW_BMS63EN_CELL1; cell_idx <= AEK_POW_BMS63EN_CELL14; cell_idx ++){
sprintf(message, "S%.2d:%.3d    ", cell_idx+1, (int)((AEK_POW_BMS63EN_BatteryModule.AEK_POW_B
MS63EN_Pack_SOC[cell_idx]) * 100));
sd_lld_write(&SD5, (uint8_t *)message, (uint16_t)(sizeof(message)/sizeof(message[0])));
}
sprintf(message,"          \n");
sd_lld_write(&SD5, (uint8_t *)message, (uint16_t)(sizeof(message)/sizeof(message[0])));
//Printing Bal
for(cell_idx = AEK_POW_BMS63EN_CELL1; cell_idx <= AEK_POW_BMS63EN_CELL14; cell_idx ++){
sprintf(message, "B%.2d:%.3d    ", cell_idx+1, AEK_POW_BMS63EN_BatteryModule.AEK_POW_BMS63EN_
Pack_Bal_cmd[cell_idx]);
sd_lld_write(&SD5, (uint8_t *)message, (uint16_t)(sizeof(message)/sizeof(message[0])));
}
sprintf(message,"          \n");
sd_lld_write(&SD5, (uint8_t *)message, (uint16_t)(sizeof(message)/sizeof(message[0])));
//Printing Voltage
for(cell_idx = AEK_POW_BMS63EN_CELL1; cell_idx <= AEK_POW_BMS63EN_CELL14; cell_idx ++){
sprintf(message, "V%.2d:%.3f  ", cell_idx+1, AEK_POW_BMS63EN_BatteryModule.AEK_POW_BMS63EN_Pa
ck_CellVoltage[cell_idx]);
sd_lld_write(&SD5, (uint8_t *)message, (uint16_t)(sizeof(message)/sizeof(message[0])));
}
sprintf(message,"          \n");
sd_lld_write(&SD5, (uint8_t *)message, (uint16_t)(sizeof(message)/sizeof(message[0])));
//Printing Current
sprintf(message, "C:%.4f    \n", AEK_POW_BMS63EN_BatteryModule.AEK_POW_BMS63EN_Pack_Current);
sd_lld_write(&SD5, (uint8_t *)message, (uint16_t)(sizeof(message)/sizeof(message[0])));
}
  }
}
void *sbrk(size_t incr)
{
```

```
extern uint8_t __heap_base__;
extern uint8_t __heap_end__;
static uint8_t *p=&__heap_base__;
static uint8_t *newp;
newp = p+ incr;
if(newp> &__heap_end__)
{
return (void*)-1;
}
return p =newp;
}
int main(void) {
componentsInit();
irqIsrEnable();
wkpu_lld_start(&WKPUD1, &wkpu_config_wkpu_cfg);
AEK_POW_BMS63EN_init();
runCore0();
AEK_POW_BMS63EN_Start_Mgn_exec();
while (1) {
}
}
```

This application enables two cores:

- Core2 (main core)
  - Used to start the manager process (AEK_POW_BMS63EN_Mng_exec()) through a PIT (i.e., a 5 Hz programmable timer:)

```
void AEK_POW_BMS63EN_Mgn_exec(){
pal_lld_togglepad(PORT_F,LED3);
AEK_POW_BMS63EN_soc_elab_sts = AEK_POW_BMS63EN_SOC_ELABORATION_BUSY;
AEK_POW_BMS63EN_VA_Measurement();
AEK_POW_BMS63EN_TEMP_Measurement();
if(AEK_COM_ISOSPI_GetFault(ISOSPI_DEV0)){
AEK_POW_BMS63EN_DIAG_Measurement();
}
AEK_POW_BMS63EN_Model_exec();
AEK_POW_BMS63EN_Balancing();
AEK_POW_BMS63EN_soc_elab_sts = AEK_POW_BMS63EN_SOC_ELABORATION_DONE;
pal_lld_togglepad(PORT_F,LED3);
}
```

    This process starts the manual conversion routine to measure the cell voltage, current, and GPIO NTC temperature.
    Moreover, this process performs the SOC estimation by using the C-code generated through a Simulink model and enables the manual balancing during the Normal state of the FSM.
- Core0 (auxiliary core)
  - Used to start a process that can print via serial the following data (with a speed rate of 115200 bps):
    - State of charge percentage for each cell
    - Cell voltage values
    - Current value
    - Balancing status
  - This process is called every 100 ms.

10 Available APIs

The APIs listed in the following tables are declared in the AEK_POW_BMS63EN_id.h file.

Table 1. General purpose APIs for the AEK-POW-BMS63EN

Key API name	Description
`AEK_POW_BMS63EN_Init` (void)	Driver instance(s) initialization.
`AEK_POW_BMS63EN_ParameterConfiguration` (void)	Driver instance(s) parameter configuration.
`AEK_POW_BMS63EN_ConfigISOFreq (AEK_POW_BMS63EN_iso_freq_t AEK_POW_BMS63EN_iso_freq)`	Configures the isolated communication frequency.
`AEK_POW_BMS63EN_DisableCommTimeout (unsigned AEK_POW_BMS63EN_devnum)`	Disables the communication timeout.
`AEK_POW_BMS63EN_EnableCommTimeout (unsigned AEK_POW_BMS63EN_devnum)`	Enables the communication timeout.
`AEK_POW_BMS63EN_EnterSleepMode (unsigned AEK_POW_BMS63EN_devnum)`	Sends the device(s) to the 'SLEEP' or 'SILENT BALANCING' state.
`AEK_POW_BMS63EN_SWReset (unsigned AEK_POW_BMS63EN_devnum)`	Performs a software reset and sends the device(s) to the 'SLEEP' state.
`AEK_POW_BMS63EN_StartUpWKUP (void)`	Wakes up the overall stack of devices.
`AEK_POW_BMS63EN_GetState (unsigned AEK_POW_BMS63EN_devnum)`	Gets the current state.

Table 2. Application-oriented APIs for the AEK-POW-BMS63EN

Key API name	Description
`AEK_POW_BMS63EN_CoulombCounting (unsigned AEK_POW_BMS63EN_devnum, float qtc_R_shunt, float qtk)`	Performs the Coulomb counting.
`AEK_POW_BMS63EN_DisableCoulombCounting (unsigned AEK_POW_BMS63EN_devnum)`	Disables the Coulomb counting.
`AEK_POW_BMS63EN_EnableCoulombCounting .(unsigned AEK_POW_BMS63EN_devnum)`	Enables the Coulomb counting.

Table 3. Cell-oriented APIs for the AEK-POW-BMS63EN

Key API name	Description
`AEK_POW_BMS63EN_DisableCell (unsigned AEK_POW_BMS63EN_devnum, AEK_POW_BMS63EN_cell_id_t cell_id)`	Disables a selected cell.
`AEK_POW_BMS63EN_DisableCellBalancing (unsigned AEK_POW_BMS63EN_devnum, AEK_POW_BMS63EN_cell_id_t cell_id)`	Disables the balancing of a selected cell.
`AEK_POW_BMS63EN_Get_Conf_OVL (unsigned AEK_POW_BMS63EN_devnum)`	Gets Configuration Override latched status.
`AEK_POW_BMS63EN_EnableCellBalancing (unsigned AEK_POW_BMS63EN_devnum, AEK_POW_BMS63EN_cell_id_t cell_id )`	Enables the balancing of a selected cell.
`AEK_POW_BMS63EN_DisableBalCurrentEvenCells (unsigned AEK_POW_BMS63EN_devnum)`	Disables the balancing current on the even cells.

Key API name	Description
AEK_POW_BMS63EN_DisableBalCurrentOddCells (unsigned AEK_POW_BMS63EN_devnum)	Disables the balancing current on the odd cells.
AEK_POW_BMS63EN_EnableBalCurrentEvenCells (unsigned AEK_POW_BMS63EN_devnum)	Enables the balancing current on the even cells.
AEK_POW_BMS63EN_EnableBalCurrentOddCells (unsigned AEK_POW_BMS63EN_devnum)	Enables the balancing current on the odd cells.
AEK_POW_BMS63EN_EnableCell (unsigned AEK_POW_BMS63EN_devnum, AEK_POW_BMS63EN_cell_id_t cell_id)	Enables a selected cell.
AEK_POW_BMS63EN_GetBalancingStatus (unsigned AEK_POW_BMS63EN_devnum)	Gets the current balancing status of a given instance.
AEK_POW_BMS63EN_GetCellVoltMeasurement (unsigned AEK_POW_BMS63EN_devnum, AEK_POW_BMS63EN_cell_id_tcell_id)	Reads the ADC conversion data of a selected cell.
AEK_POW_BMS63EN_SetBalancingMode (unsigned AEK_POW_BMS63EN_devnum, AEK_POW_BMS63EN_cell_bal_mode_tcell_bal_mode)	Sets the balancing mode of a given cell.
AEK_POW_BMS63EN_SetCellTimedBalThreshold (unsigned AEK_POW_BMS63EN_devnum, AEK_POW_BMS63EN_cell_id_tcell_id, uint16_t cell_thr)	Sets the threshold for the timed balancing mode of a given cell.
AEK_POW_BMS63EN_StartCellBalancing (unsigned AEK_POW_BMS63EN_devnum)	Starts the balancing on the enabled cells of a device.
AEK_POW_BMS63EN_StopCellBalancing (unsigned AEK_POW_BMS63EN_devnum)	Stops the balancing on the enabled cells of a device.

Table 4. GPIO 9, 3 configuration APIs for the AEK-POW-BMS63EN

Key API name	Description
AEK_POW_BMS63EN_ConfigGPIO (unsigned AEK_POW_BMS63EN_devnum, AEK_POW_BMS63EN_gpio_t AEK_POW_BMS63EN_gpio, AEK_POW_BMS63EN_gpio_conf_t AEK_POW_BMS63EN_gpio_conf)	Configures the GPIO 3 - 9.
AEK_POW_BMS63EN_SetGPIOModeAbsolute (unsigned AEK_POW_BMS63EN_devnum, AEK_POW_BMS63EN_gpio_t AEK_POW_BMS63EN_gpio)	Sets the GPIO absolute mode (GPIO between 3 and 9).
AEK_POW_BMS63EN_SetGPIOModeRatiometric (unsigned AEK_POW_BMS63EN_devnum, AEK_POW_BMS63EN_gpio_t AEK_POW_BMS63EN_gpio)	Sets the GPIO ratiometric mode (GPIO between 3 and 9).
AEK_POW_BMS63EN_SetGPIOOVThreshold (unsigned AEK_POW_BMS63EN_devnum, AEK_POW_BMS63EN_gpio_t AEK_POW_BMS63EN_gpio, uint16_t gpio_ov_thr)	Sets the GPIO 3 -> 9 overvoltage threshold.
AEK_POW_BMS63EN_SetVTREFEnabled (unsigned AEK_POW_BMS63EN_devnum)	Sets the GPIO 3 -> 9 undervoltage threshold.
AEK_POW_BMS63EN_SetVTREFDisabled (unsigned AEK_POW_BMS63EN_devnum)	Disables VTref.
AEK_POW_BMS63EN_GetCalibCurrentMeasurement (unsigned AEK_POW_BMS63EN_devnum, uint16_t Rshunt)	Gets Calib Current Measurement.
AEK_POW_BMS63EN_GetInstCurrentMeasurement (unsigned AEK_POW_BMS63EN_devnum, uint16_t Rshunt)	Gets Inst Current Measurement.

Key API name	Description
AEK_POW_BMS63EN_GetBatteryPackVoltMeasurement (unsigned AEK_POW_BMS63EN_devnum)	Gets GPIO Voltage Measurement.
AEK_POW_BMS63EN_GetVTREFVoltMeasurement (unsigned AEK_POW_BMS63EN_devnum)	Gets VTREF Voltage Measurement.
AEK_POW_BMS63EN_GetVoltageGPIOMeasurement (unsigned AEK_POW_BMS63EN_devnum, AEK_POW_BMS63EN_gpio_t AEK_POW_BMS63EN_gpio, float Vref, AEK_POW_BMS63EN_ratio_abs_mode_t mode)	Gets GPIO Voltage Measurement with absolute or ratiometric mode.
AEK_POW_BMS63EN_GPIO_OT_UT_Mask (unsigned AEK_POW_BMS63EN_devnum)	Masks Over/Under Temperature fault detection.
AEK_POW_BMS63EN_SetGPIOConv (unsigned AEK_POW_BMS63EN_devnum)	Enables GPIO Measurement.
AEK_POW_BMS63EN_GPIO_CSA_Mask (unsigned AEK_POW_BMS63EN_devnum)	Masks CSA GPIO over/under current detection.
AEK_POW_BMS63EN_CoulombCounterReset (unsigned AEK_POW_BMS63EN_devnum)	Resets Coulomb Counter.
AEK_POW_BMS63EN_isEnabled (unsigned AEK_POW_BMS63EN_devnum, AEK_POW_BMS63EN_cell_id_t cell_idx	Checks Enabled Cell.
AEK_POW_BMS63EN_SetOvercurrentThreshold (unsigned AEK_POW_BMS63EN_devnum, uint16_t Rshunt, float current_th_A)	Sets OverCurrent Threshold.

Table 5. Other APIs for the AEK-POW-BMS63EN

Key API name	Description
AEK_POW_BMS63EN_ObjectInit (AEK_POW_BMS63EN_Driver *AEK_POW_BMS63ENp)	Initializes the standard part of a AEK_POW_BMS63EN_Driver structure.
AEK_POW_BMS63EN_Start (AEK_POW_BMS63EN_Driver *AEK_POW_BMS63ENp, AEK_POW_BMS63EN_Config *config)	Configures and activates the AEK-POW-BMS63EN.
AEK_POW_BMS63EN_Stop (AEK_POW_BMS63EN_Driver *AEK_POW_BMS63ENp)	Deactivates the AEK-POW-BMS63EN.
AEK_POW_BMS63EN_SPIBroadcastAccess (AEK_POW_BMS63EN_Driver *AEK_POW_BMS63ENp, unsigned regnum, AEK_POW_BMS63EN_register_t value)	Sends a 'write' command to all the device units.
AEK_POW_BMS63EN_SPIBurstAccess (AEK_POW_BMS63EN_Driver *AEK_POW_BMS63ENp, unsigned devnum, unsigned command, AEK_POW_BMS63EN_register_t *reg_values)	Reads registers in a burst from a given device.
AEK_POW_BMS63EN_SPIQueuedAccess (AEK_POW_BMS63EN_Driver *AEK_POW_BMS63ENp, uint8_t n_bytes, const uint8_t *txqueue, uint8_t *rxqueue)	Writes a queue.
AEK_POW_BMS63EN_SPIWriteRegister (AEK_POW_BMS63EN_Driver *AEK_POW_BMS63ENp, unsigned devnum, unsigned regnum, AEK_POW_BMS63EN_register_t value)	Writes a register.
AEK_POW_BMS63EN_SPIReadRegister (AEK_POW_BMS63EN_Driver *AEK_POW_BMS63ENp, unsigned devnum, unsigned regnum)	Reads a register.

Key API name	Description
`AEK_POW_BMS63EN_DualRingTX_reverse (void)`	Reverse TX in dual ring mode.

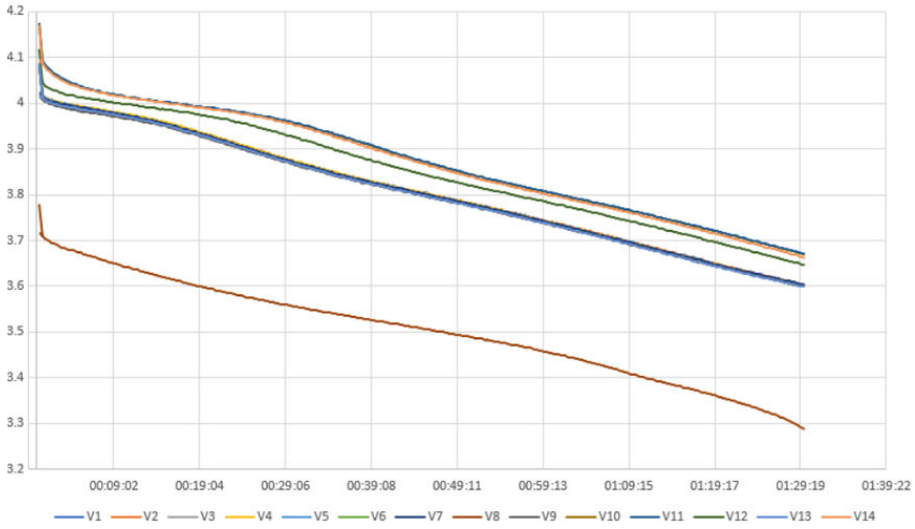
For further details on the APIs, refer to the doxygen documentation under the doc folder (see Component folder structure).

11 Waveforms

The following waveform shows the time trend of the voltages of the 14 cells connected to an AEK-POW-BMS63EN in a single centralized configuration, with an active load of 1 A and balancing activated by the cell 8 voltage value.

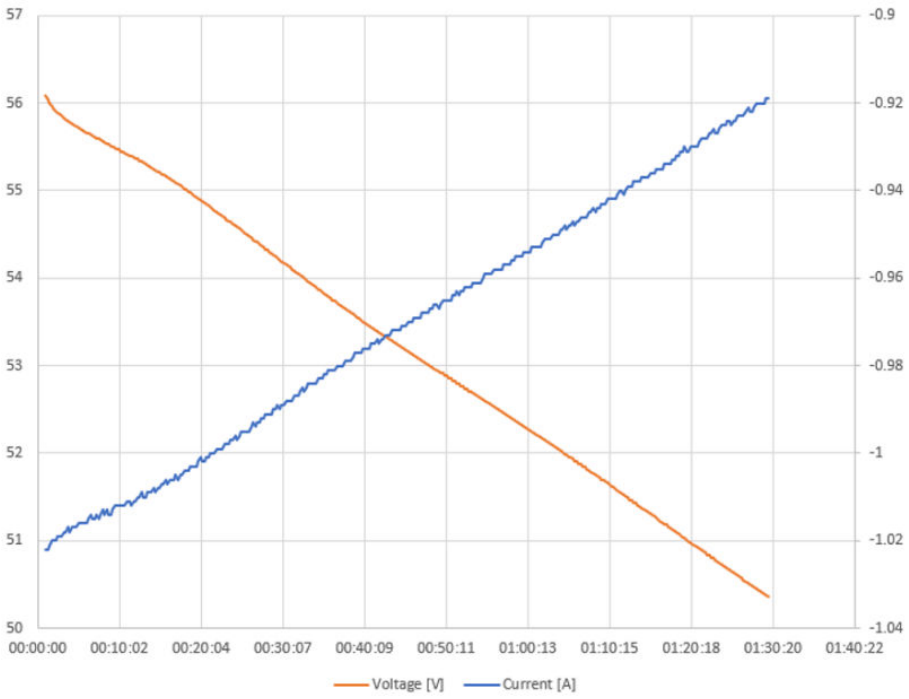
The x axis shows the time (in hours). The y axis shows the 14 cell voltages in Volts.

Figure 43. 14-cell voltage discharge with an active load of 1 A



The following waveform shows the time trend of the battery pack voltage (defined by 14 cells connected to an AEK-POW-BMS63EN in a single centralized configuration) with an active load of 1 A. The x axis shows the time (in hours). The y axis shows the battery pack voltage in Volts and the load current.

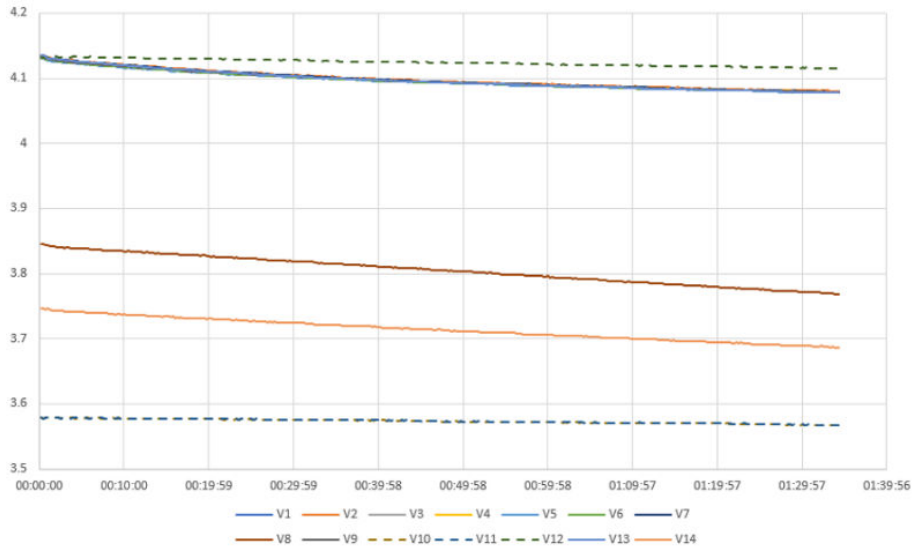
Figure 44. Battery pack voltage discharge with an active load of 1 A



The following waveform shows the time trend of the voltages of the 14 cells connected to an AEK-POW-BMS63EN in a single centralized configuration, when balancing is active.

The x axis shows the time (in hours). The y axis shows the voltages of the 14 cells in Volts.

Figure 45. 14 cell voltages during balancing



11.1 Process timing

The following table defines the time length of the tasks defined in the “SPC584B – AEK_POW_BMS63EN_SOC_Estimation_Single application for discovery” demo.

Table 6. Task process timing

Task	Timing
Voltage and current estimation	58.18 ms
Temperature NTC estimation	36.14 ms
SOC estimation	15.87 ms
Fault estimation	7.052 ms

The figures below show the waveforms of the estimations of the table above.

Figure 46. **Voltage estimation waveform**

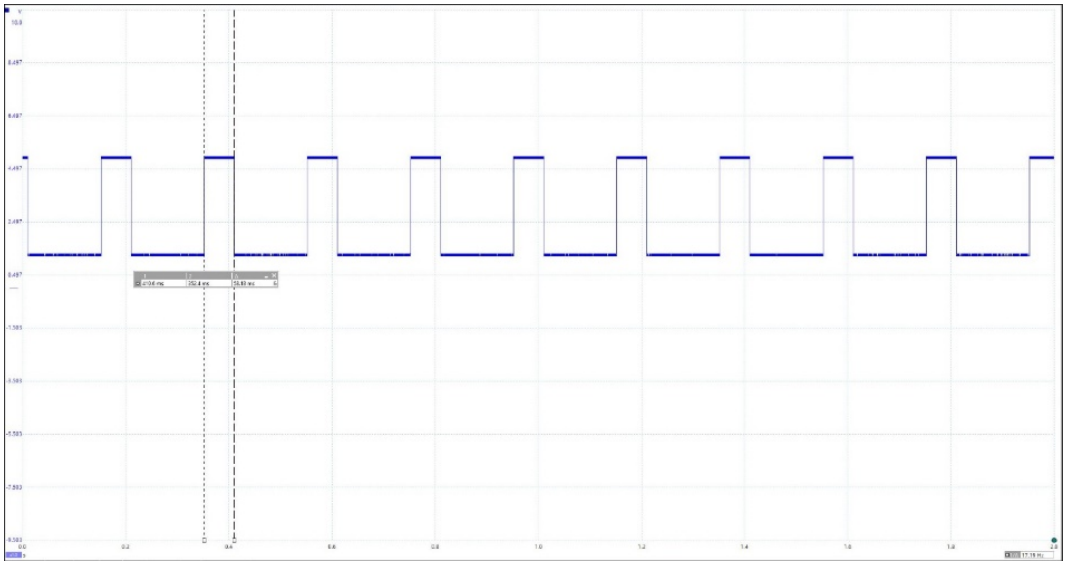


Figure 47. **NTC temperature estimation waveform**

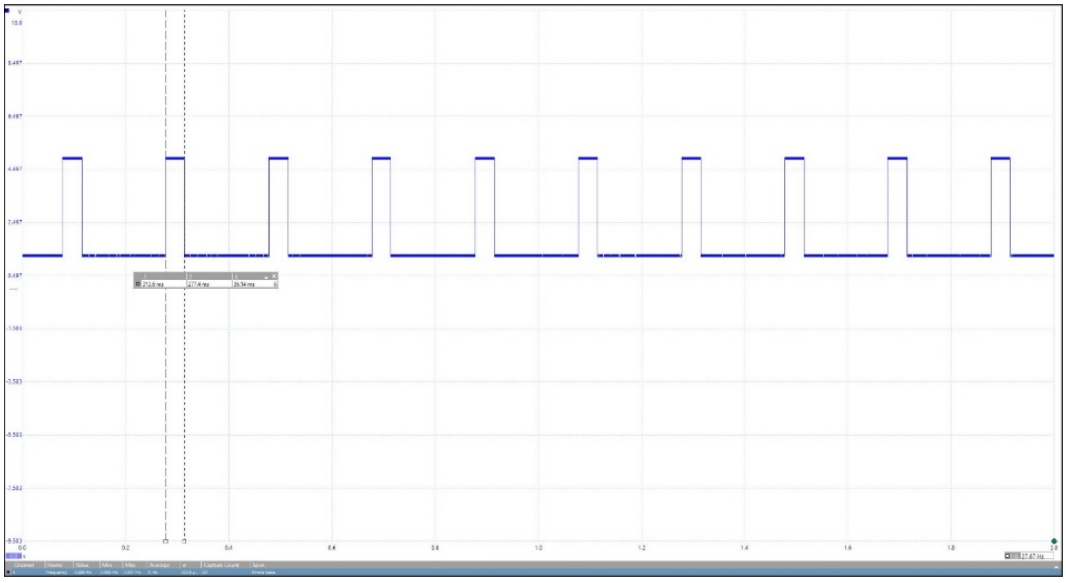


Figure 48. **SOC estimation waveform**

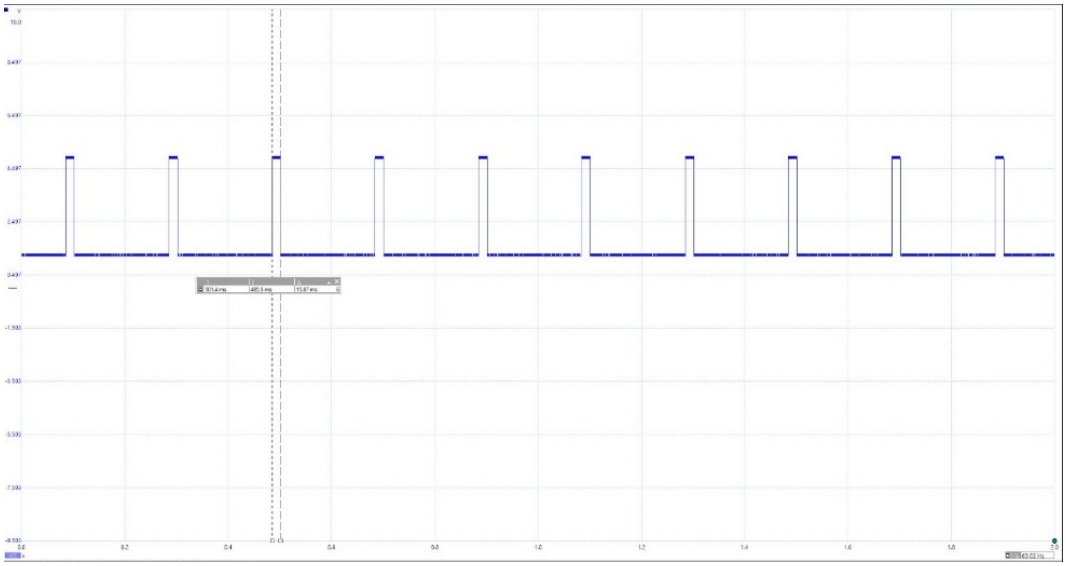
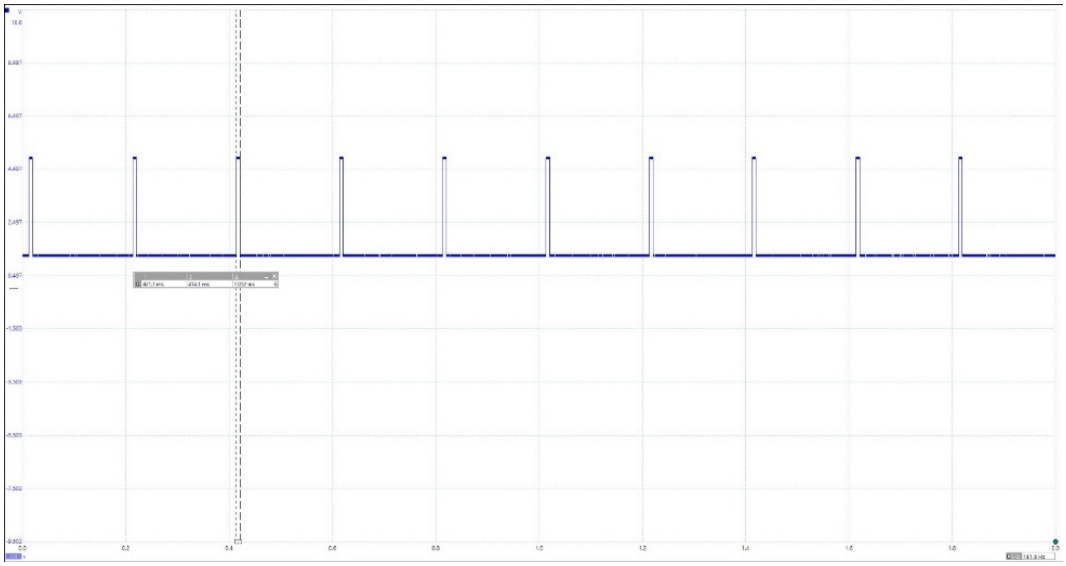
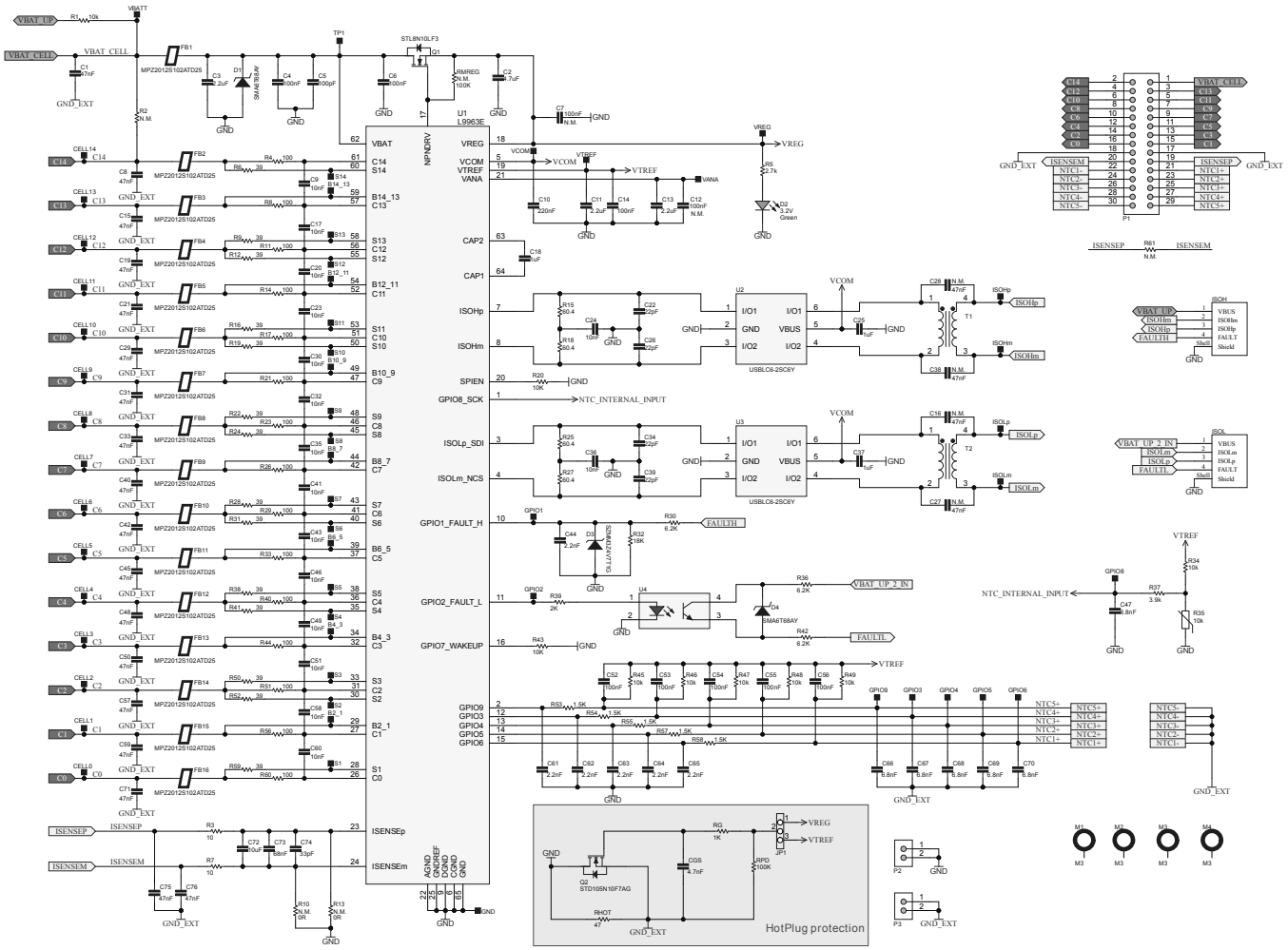


Figure 49. **Fault estimation waveform**



12 Schematic diagrams

Figure 50. **AEK-POW-BMS63EN schematic diagram**



13 Bill of materials

Table 7. **BOM**

Item	Q.ty	Ref.	Part / Value	Description	Manufacturer	Order code
1	18	C1, C8, C15, C19, C21, C29, C31, C33, C40, C42, C45, C48, C50, C57, C59, C71, C75, C76	47nF	0603 - 50V - X7R Class II	WE	885012206093
2	1	C2	4.7uF	1206 - 50V - X7R Class II	WE	885012208094
3	1	C3	2.2uF	1210 - 100V - X7R Class II	WE	885012209071
4	3	C4, C6, C14	100nF	0603 - 100V - X7R Class II	WE	885012206120
5	1	C5	100pF	0603 - 100V - X7R Class II	WE	885012206102
6	2	C7, C12	N.M.	0603	N.M.	N.M.
7	16	C9, C17, C20, C23, C24, C30, C32, C35, C36, C41, C43, C46, C49, C51, C58, C60	10nF	0603 - 50V - X7R Class II	WE	885012206089
8	1	C10	220nF	0603 - 50V - X7R Class II	WE	885012206125
9	2	C11, C13	2.2uF	0805 - 25V - X7R Class II	WE	885012207079
10	4	C16, C27, C28, C38	N.M.	1206	N.M.	N.M.
11	3	C18, C25, C37	1uF	0805 - 50V - X7R Class II	WE	885012207103
12	4	C22, C26, C34, C39	22pF	0603 - 50V - NP0 Class I	WE	885012006053
13	6	C44, C61, C62, C63, C64, C65	2.2nF	0603 - 50V - X7R Class II	WE	885012206085
14	6	C47, C66, C67, C68, C69, C70	6.8nF	0603 - 50V - X7R Class II	WE	885012206088
15	5	C52, C53, C54, C55, C56	100nF	0603 - 50V - X7R Class II	WE	885012206095
16	1	C72	10uF	1210 - 50V - X7R Class II	WE	885012209073
17	1	C73	68nF	0603 - 50V - X7R Class II	WE	885012206094
18	1	C74	33pF	0603 - 50V - NP0 Class I	WE	885012006054
19	1	CGS	4.7nF	0603 - 50V - X7R Class II	WE	885012206087
20	2	D1, D4	SMA6T68AY	Automotive 600 W, 68V TVS in SMA	STMicroelectronics	SMA6T68AY

Item	Q.ty	Ref.	Part / Value	Description	Manufacturer	Order code
21	1	D2	Green	0805 - Led Green - 3.2V	WE	150080GS75000
22	1	D3	SZMM3Z4V7T1G	4.7V Zener Voltage Regulators, 300mW	Onsemi	SZMM3Z4V7T1G
23	16	FB1, FB2, FB3, FB4, FB5, FB6, FB7, FB8, FB9, FB10, FB11, FB12, FB13, FB14, FB15, FB16	1K@100MHz	Ferrite Beads Multi-Layer Power 1KOhm 25% 100MHz 1.5A 0.15Ohm DCR 0805	TDK	MPZ2012S102ATD25
24	2	ISOH, ISOL	61400416021	USB 2.0 Type A, Receptacle, Horizontal, THT	WE	61400416021
25	1	JP1		THT Vertical 3 pins Header, Pitch 2.54 mm, Single Row	WE	61300311121
26	1	P1		2.00mm - WR-WTB - Male Dual Row Horizontal Shrouded Header w. positive locking	WE	62403021722
27	2	P2, P3	61300211121	2.54mm - WR-PHD Pin Header, THT, pitch 2.54mm, Single Row, Vertical, 2p	WE	61300211121
28	1	Q1	STL8N10LF3, PowerFLAT 5x6 WF	Automotive-grade N-channel 100 V, 25 mΩ typ., 7.8 A STripFET™ F3 Power MOSFET in a PowerFLAT™ 5x6 package	STMicroelectronics	STL8N10LF3
29	1	Q2	STD105N10F7AG, DPAK	Automotive-grade N-channel 100 V, 6.8 mΩ typ., 80 A, STripFET™ F7 Power MOSFET in a DPAK package	STMicroelectronics	STD105N10F7AG
30	1	R1	10k	1206 - ±1% - 0.66W	Panasonic	ERJUP8F1002V
31	1	R2	N.M.	0805	N.M.	N.M.
32	2	R3, R7	10	0603 - ±1% - 0.25W	Panasonic	ERJPA3F10R0V
33	15	R4, R8, R11, R14, R17, R21, R23, R26, R29, R33, R40, R44, R51, R56, R60	100	0603 - ±1% - 0.25W	Panasonic	ERJPA3F1000V
34	1	R5	2.7k	0603 - ±1% - 0.125W	Vishay	MCT06030C2701FP500
35	14	R6, R9, R12, R16, R19, R22, R24, R28, R31, R38, R41, R50, R52, R59	39	2010 - ±1% - 1.25W	TE Connectivity	CRGP2010F39R
36	3	R10, R13, RMREG	N.M.	0603	N.M.	N.M.

Item	Q.ty	Ref.	Part / Value	Description	Manufacturer	Order code
37	4	R15, R18, R25, R27	60.4	0603 - ±1% - 0.1W	Panasonic	ERJ3EKF60R4V
38	2	R20, R43	10K	0603 - ±1% - 0.2W	Panasonic	ERJP03F1002V
39	3	R30, R36, R42	6.2K	0805 - ±1% - 0.5W	Panasonic	ERJP06F6201V
40	1	R32	18K	0603 - ±1% - 0.2W	Panasonic	ERJP03F1802V
41	6	R34, R45, R46, R47, R48, R49	10k	0805 - ±1% - 0.5W	Panasonic	ERJP6WF1002V
42	1	R35	10k	0603 - ±1% - 0.1W	TDK	NTCG163JH103HTDS
43	1	R37	3.9k	0603 - ±1% - 0.1W	Panasonic	ERJ3EKF3901V
44	1	R39	2K	0603 - ±1% - 0.2W	Panasonic	ERJP03F2001V
45	5	R53, R54, R55, R57, R58	1.5K	2010 - ±1% - 2W	TE Connectivity	35021K5FT
46	1	R61	N.M.	N.M.	N.M.	N.M.
47	1	RG	1K	0603 - ±1% - 0.25W	Panasonic	ERJPA3F1001V
48	1	RHOT	47	2512 - ±5% - 1W	TE Connectivity	352047RJT
49	1	RPD	100K	0603 - ±1% - 0.25W	Panasonic	ERJP03F1003V
50	2	T1, T2	125uH	Transformer for BMS	WE	74941000
51	1	U1	L9963E, TQFP 64 10x10x1.0	Automotive chip for battery management applications with daisy chain up to 31 devices	STMicroelectronics	L9963E
52	2	U2, U3	USBLC6-2SC6Y, SOT23-6L	Automotive ESD protection for high speed interfaces.	STMicroelectronics	USBLC6-2SC6Y
53	1	U4	PS2703-1-A	4-Pin Phototransistor Optocoupler	Renesas	PS2703-1-A
54	1	for blister	60900213421	WR-PHD 2.54 mm Multi-Jumper Jumper with Test Point	WE	60900213421
55	4	for blister	970080365	WA-SPAII Plastic Spacer Stud, metric, internal/ internal	WE	970080365
56	4	for blister	97790403111	WA-SCRW Pan Head Screw w. cross slot M3	WE	97790403111
57	1	for blister	624030213322	WR-WTB 2.00 mm Female Dual Row Terminal Housing w. positive locking	WE	624030213322
58	30	for blister	62400113722	WR-WTB 2.00 mm Female Dual Row Crimp Contact	WE	62400113722

14 Board versions

Table 8. **AEK-POW-BMS63EN versions**

Finished good	Schematic diagrams	Bill of materials
AEK$POW-BMS63ENA(1)	AEK-POW-BMS63EN schematic diagrams	AEK-POW-BMS63EN bill of materials

1. *This code identifies the AEK-POW-BMS63EN evaluation board first version. It is printed on the board PCB.*

15 Regulatory compliance information

Notice for US Federal Communication Commission (FCC)

For evaluation only; not FCC approved for resale

FCC NOTICE - This kit is designed to allow:

(1) Product developers to evaluate electronic components, circuitry, or software associated with the kit to determine

whether to incorporate such items in a finished product and

(2) Software developers to write software applications for use with the end product.

This kit is not a finished product and when assembled may not be resold or otherwise marketed unless all required FCC equipment authorizations are first obtained. Operation is subject to the condition that this product not cause harmful interference to licensed radio stations and that this product accept harmful interference. Unless the assembled kit is designed to operate under part 15, part 18 or part 95 of this chapter, the operator of the kit must operate under the authority of an FCC license holder or must secure an experimental authorization under part 5 of this chapter 3.1.2.

Notice for Innovation, Science and Economic Development Canada (ISED)

For evaluation purposes only. This kit generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to Industry Canada (IC) rules.

À des fins d'évaluation uniquement. Ce kit génère, utilise et peut émettre de l'énergie radiofréquence et n'a pas été testé pour sa conformité aux limites des appareils informatiques conformément aux règles d'Industrie Canada (IC).

Notice for the European Union

This device is in conformity with the essential requirements of the Directive 2014/30/EU (EMC) and of the Directive 2015/863/EU (RoHS).

Notice for the United Kingdom

This device is in compliance with the UK Electromagnetic Compatibility Regulations 2016 (UK S.I. 2016 No. 1091) and with the Restriction of the Use of Certain Hazardous Substances in Electrical and Electronic Equipment Regulations 2012 (UK S.I. 2012 No. 3032).

Revision history

Table 9. **Document revision history**

Date	Version	Changes
16-May-2023	1	Initial release.

Contents

List of figures

List of tables