EN-2160
Electronic Design Realization

Expandable Battery Management System
Design Details

Name	Index Number
Dilshan N.L.	*210129P*
Dodangoda D.K.S.J.	*210150V*
Gunawardana W.N.M.	*210199D*
Wathudura T.R.	*210682D*

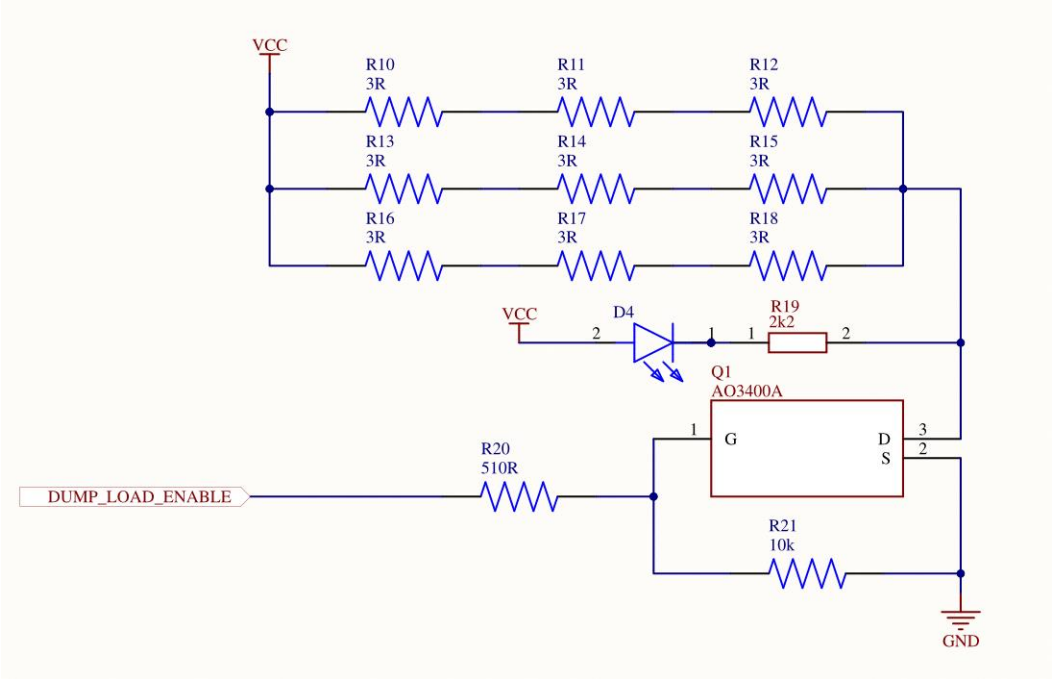
Table of Contents

Circuit Design Details

Cell Module

This module is an integral component designed for the real-time monitoring of individual battery cells within a pack, each with a nominal voltage of 3.7V. This module is specifically developed to ensure the optimal performance, safety, and longevity of battery packs used in various applications, ranging from portable electronic devices to electric vehicles. The detailed functionalities, including voltage and temperature monitoring, communication via UART, and protective measures, are outlined below:

1. Over-voltage monitoring and voltage balancing circuit:



The battery monitoring system incorporates a sophisticated control strategy to prevent overcharging, which is critical for maintaining battery health and safety. The system utilizes a microcontroller unit (MCU) that continuously monitors the voltage levels of each battery cell within the pack. When the voltage of a cell exceeds the recommended threshold of 3.7V, indicative of potential overcharging, the MCU takes proactive measures to regulate the charge and maintain voltage stability.

❖ **Activation of the Dump Load Enabler:**

- Upon detection of an over-voltage condition, the MCU activates the Dump Load Enabler, which in turn controls a MOSFET. The MOSFET acts as a switch that reroutes the excess current away from the battery cell.

❖ **Current Diversion Through Resistors:**

- The redirected current is passed through a predefined set of resistors. These resistors are specifically chosen for their ability to handle the excess current without overheating or damage. By flowing through these resistors, the overcurrent is safely dissipated as heat, and the voltage across the battery cell is effectively lowered and stabilized at the

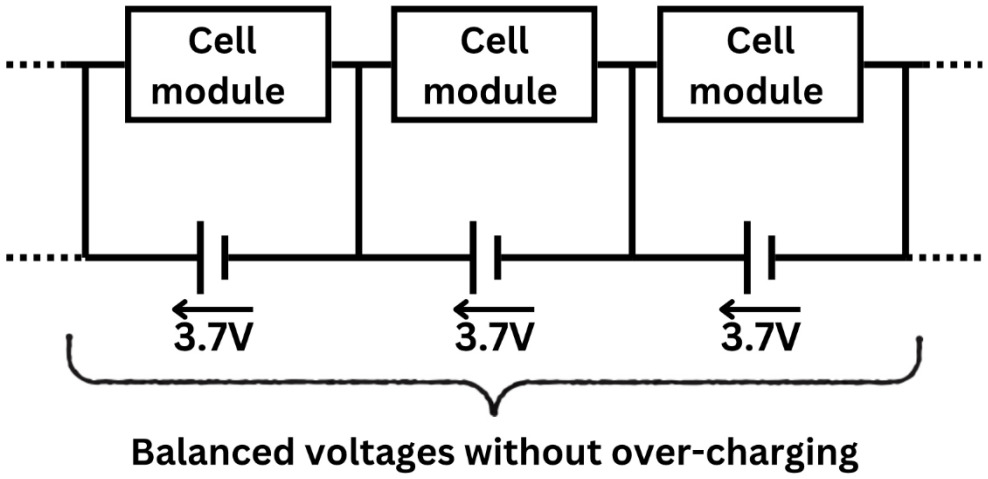
safe level of 3.7V.

❖ **Balancing Charge Across Series-Connected Battery Packs:**

- This controlled method of handling overcharge conditions not only protects individual cells but also contributes to the overall balance of charge across series-connected battery packs. By ensuring that no single cell overcharges, the system promotes uniformity in the charge state across the entire battery pack, which is essential for optimal performance and longevity of the batteries.

❖ **System Benefits:**

- This approach not only prevents the detrimental effects of overcharging, such as potential thermal runaway and reduced battery life but also enhances the reliability and efficiency of the battery management system. By maintaining each cell within its optimal voltage range, the system ensures that the battery pack delivers consistent performance and exhibits extended service life.



2. Temperature Sensing Circuit:

To ensure optimal performance and safety, the temperature of both the battery and the circuit within the battery management system (BMS) is meticulously monitored. This dual temperature monitoring is achieved using the CMFB103F3950FANT temperature sensor, a highly accurate and reliable component selected for its precision in thermal measurement.

❖ **Sensor Specifications and Functionality:**

- The CMFB103F3950FANT sensor is utilized for its exceptional responsiveness and stability across a broad range of temperatures. It is designed to provide precise temperature readings, which are critical for preventing thermal runaway and enhancing the longevity of the battery system. The sensor's ability to operate under varying environmental conditions without degradation makes it an ideal choice for the demanding requirements of battery management.

❖ **Separate Monitoring of Circuit and Battery Temperatures**:

In our system, separate sensors are deployed to measure.

1. The Internal Circuit Temperature: Monitoring the temperature of the electronic circuitry is essential to prevent overheating and potential damage to the BMS components. Elevated temperatures within the circuit can indicate inefficiencies or malfunctions, which may require immediate intervention.

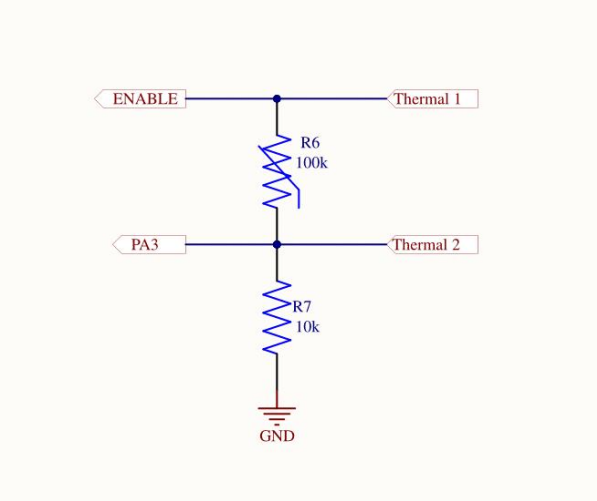
2. The Battery Temperature: Direct measurement of the battery's temperature provides data necessary to effectively manage charging and discharging processes. It helps in maintaining the battery within safe operational limits, thus safeguarding against hazards and prolonging battery life.

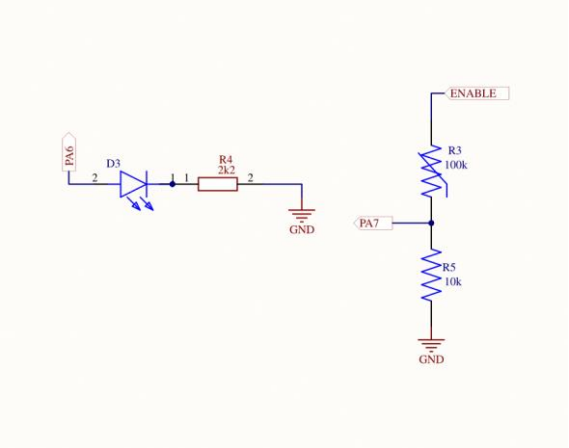
Both sets of data are critical for the BMS to execute real-time decisions regarding cooling measures, charge rate adjustments, and load balancing, which are dependent on temperature fluctuations.

❖ **Integration into BMS**:

- Each CMFB103F3950FANT sensor is integrated into the system architecture via dedicated interfaces that relay temperature data to the central processing unit of the BMS. This integration allows for continuous monitoring and instant feedback, enabling dynamic adjustments to operational parameters in response to temperature changes detected by the sensors.

- This strategic approach to temperature monitoring underscores our commitment to system safety and efficiency, ensuring that all components operate within their thermal thresholds, thereby optimizing the performance and durability of the battery system.





3. Micro-controller Unit:

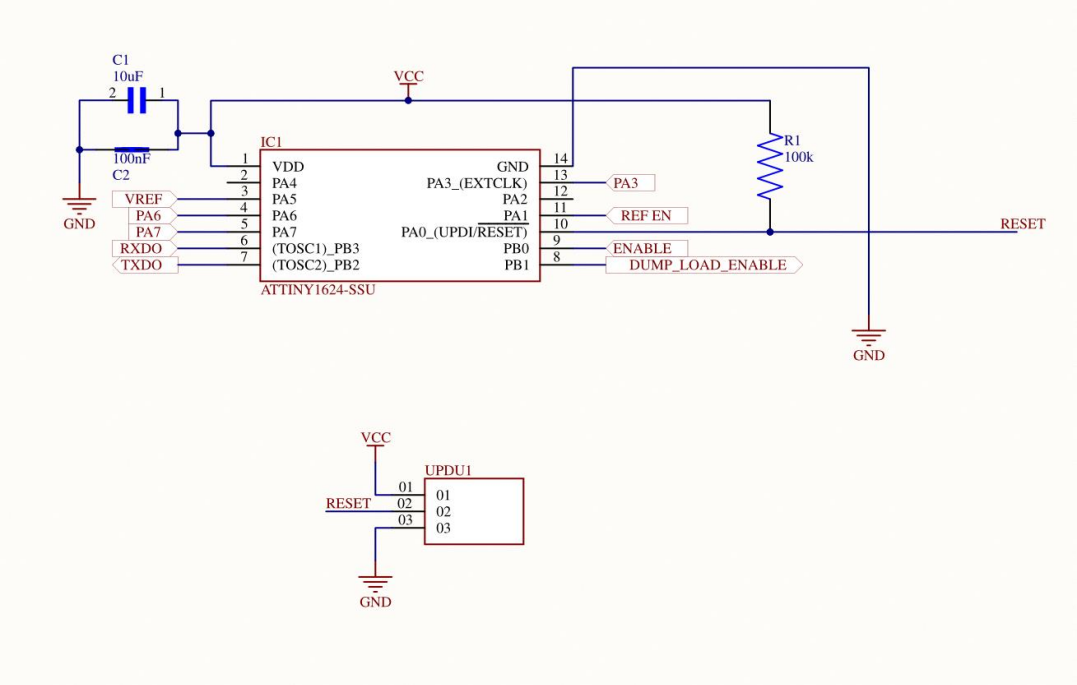
The battery management system employs a microcontroller to perform critical functions, including the measurement of battery cell voltages and the monitoring of temperatures using connected sensors. The microcontroller used in this system is the ATTINY1624-SSU, selected for its reliability, compact size, and capability to handle multiple inputs efficiently.

❖ Voltage and Temperature Measurement:

- The primary role of the microcontroller within the system is to continuously measure the voltage of each battery cell. This data is vital for assessing the state of charge and overall health of the battery. In parallel, the microcontroller reads inputs from temperature sensors strategically placed to monitor both the internal and ambient temperatures associated with the battery cells. These temperature readings help prevent conditions that could lead to overheating, thereby ensuring safe operation under various load conditions.

❖ Programming the Microcontroller:

- To facilitate precise control over these functions, the ATTINY1624-SSU microcontroller is programmed using a UPDI (Unified Program and Debug Interface) programmer. The UPDI programmer enables efficient uploading of necessary firmware into the microcontroller, which governs its operation. This programming method is advantageous for its simplicity and effectiveness, allowing for easy updates and debugging of the system software.
- The utilization of the UPDI programmer with the ATTINY1624-SSU microcontroller ensures that the system can be finely tuned to respond appropriately to the complex dynamics of battery management. This setup not only guarantees precision in monitoring but also provides flexibility in system configuration to accommodate future enhancements or adjustments in operational protocols.



4. UART transmitter and receiver:

In the battery management system (BMS), communication and data interchange between individual battery cell modules and the main controller module are facilitated using the Universal Asynchronous Receiver/Transmitter (UART) protocol. This choice of communication standard is instrumental due to its reliability, simplicity, and effectiveness in

systems requiring serial data exchange.

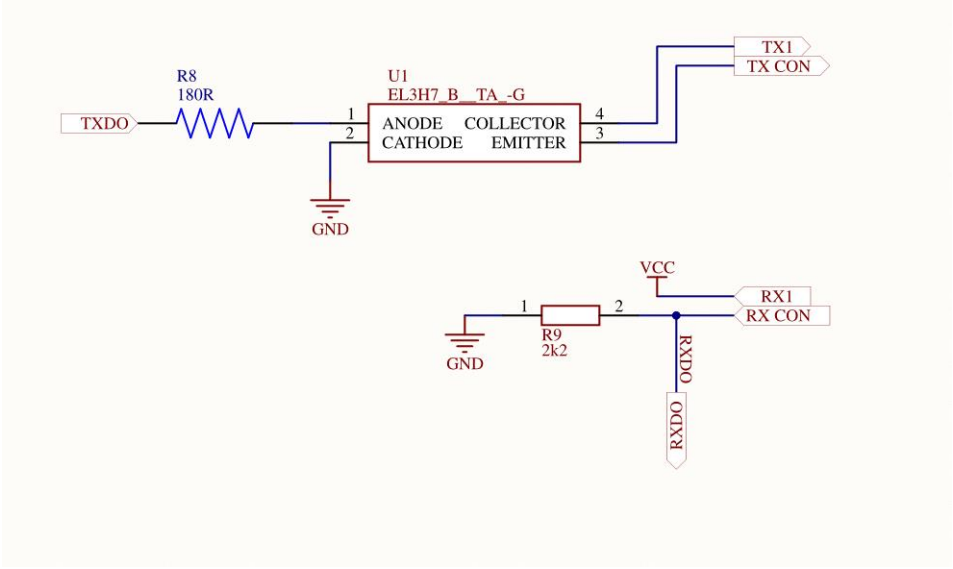
❖ Implementation of UART Communication

- UART protocol is utilized to establish a robust communication link within the BMS. Each battery cell module is equipped with UART capabilities, allowing it to transmit critical data such as voltage levels and temperature readings to the main controller module. This data is essential for real-time monitoring and management of the battery cells, enabling the main controller to make informed decisions regarding charge regulation, health assessment, and fault diagnostics.

❖ Importance of Series Connectivity

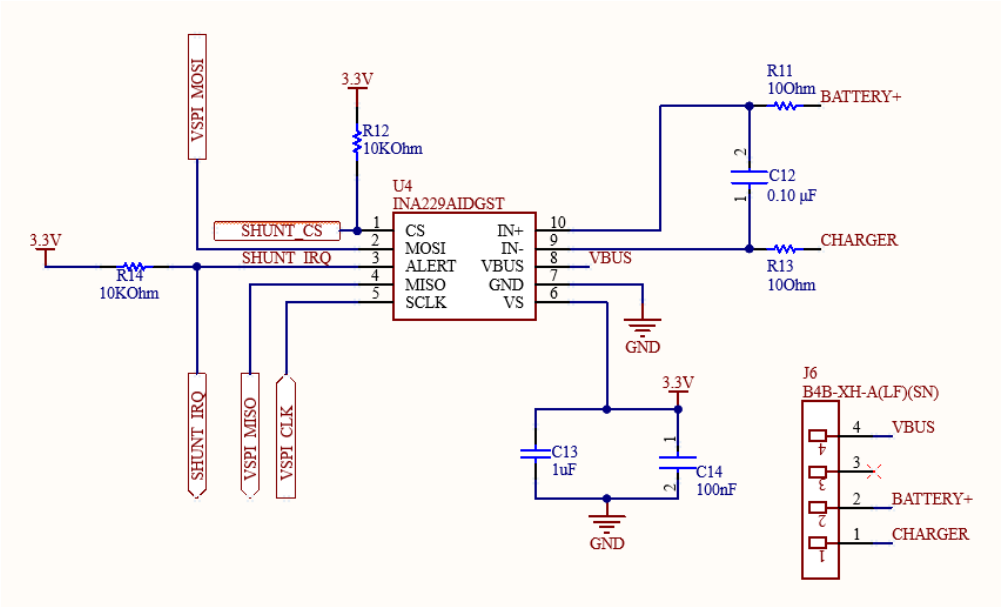
- The battery system is designed to be series-expandable, allowing for scalability in energy capacity and voltage levels. In such configurations, maintaining coherent and synchronized operation among multiple serially connected battery cell modules becomes crucial. The interconnection of each module in series via UART ensures seamless data transmission across the system, preserving the integrity and timing of the communicated signals. This setup not only supports the current system architecture but also accommodates future expansions with minimal adjustments.

The strategic implementation of UART interconnectivity thereby enhances the overall efficiency and reliability of the BMS, ensuring that each component within the series-connected array can communicate effectively with the main controller. This capability is vital for maintaining balanced charge cycles, monitoring state-of-health across the battery array and executing coordinated responses to operational anomalies.



Controller Module

1. Onboard current shunt monitor



The onboard current shunt monitor uses the INA219, a high-side current sensor from Texas Instruments.

❖ Circuit Components:

1. INA219 (U4): This is a current shunt and power monitor with an I2C interface.
2. Shunt Resistor (R11): Placed between the load (battery) and the ground to measure the voltage drop caused by the load current.
3. Pull-up Resistors (R12, R13): Connected to the I2C lines to pull them up to 3.3V.
4. Bypass Capacitors (C11, C12, C13, C14): Used to filter noise and stabilize the power supply.
5. Connector (J6): For interfacing with the battery and charger.

❖ Circuit Description:

1. Current Measurement:

- The INA219 monitors the current flowing through the shunt resistor (R11).
- The voltage drop across R11 is proportional to the current passing through it.
- INA219 measures this voltage drop and, using the known resistance value of R11, calculates the current using Ohm's Law (I = V/R).

2. Power and Bus Voltage Measurement:

- INA219 also measures the bus voltage (VBUS), which is the voltage at the load side of the shunt resistor.
- By measuring both the current and the bus voltage, the INA219 can calculate the power consumed by the load.

3. I2C Communication:

- The INA219 communicates with a microcontroller (not shown in the diagram) through the I2C interface.

- The SCL (clock) and SDA (data) lines are connected to the microcontroller's I2C bus.
- Pull-up resistors (R12, R13) are used to pull these lines up to 3.3V, ensuring proper logic levels for I2C communication.

4. Power Supply and Grounding:

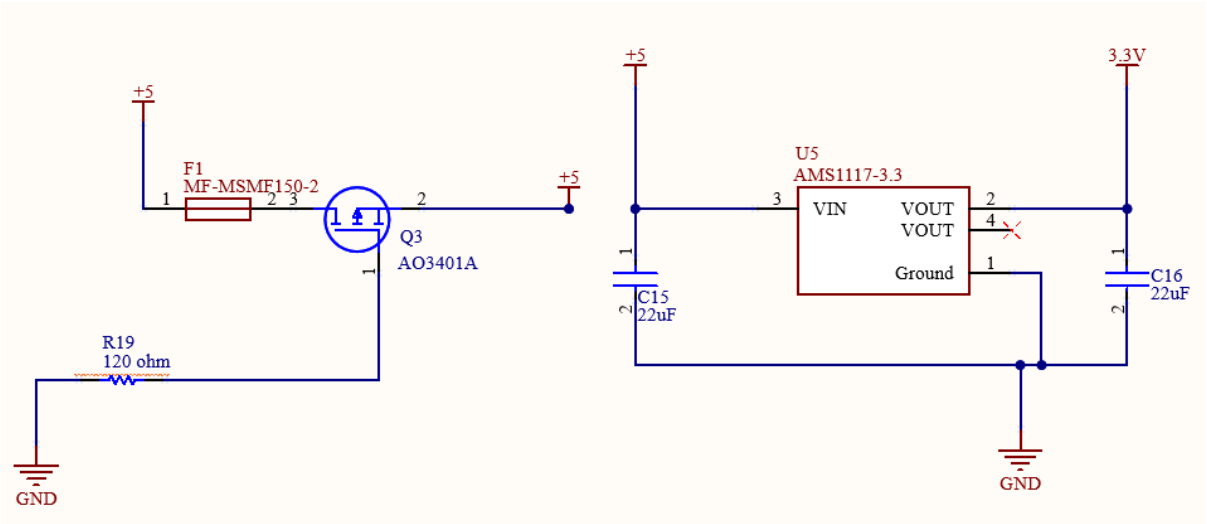
- The circuit is powered by a 3.3V supply.
- Bypass capacitors (C11, C12, C13, C14) are placed near the power pins to filter out any high-frequency noise and stabilize the power supply to the INA219 and the I2C bus.

❖ Operation:

- When the system is powered on, the INA219 continuously monitors the voltage drop across the shunt resistor and the bus voltage.
- These measurements are sent to the microcontroller via the I2C interface.
- The microcontroller can then use these readings to monitor the current, voltage, and power consumption of the connected load, which in this case is a battery and charger system.

This circuit is designed to provide real-time monitoring of current and voltage, which is crucial for applications such as battery management systems. By integrating the INA219, the circuit offers precise measurement and easy interfacing with a microcontroller via I2C, making it suitable for the project.

2. Reverse polarity protection



This circuit is designed to provide reverse polarity protection and voltage regulation for the battery management system

❖ Reverse Polarity Protection:

1. MOSFET (Q3 - AO3401A):

- The P-channel MOSFET (Q3) is used for reverse polarity protection.

- When the correct polarity is applied (+5V at the source, ground at the gate through R19):
- The MOSFET turns on, allowing current to flow from the source to the drain, thus providing +5V to the rest of the circuit.
- If the polarity is reversed (ground at the source, +5V at the gate through R19):
- The MOSFET remains off, preventing current from flowing and protecting the circuit from damage.

2. Resettable Fuse (F1 - MF-MSMF150-2):

- F1 is a resettable fuse that provides overcurrent protection.
- If the current exceeds the fuse's rating (1.5A), the fuse will trip and open the circuit, preventing damage to the components.
- Once the fault condition is removed, the fuse will reset automatically.

❖ Voltage Regulation:

1. Voltage Regulator (U5 - AMS1117-3.3):

- U5 is a linear voltage regulator that converts the +5V input to a stable +3.3V output.
- The VIN pin (pin 3) receives the +5V input.
- The VOUT pin (pin 2) provides the regulated +3.3V output.
- The GND pin (pin 1) is connected to ground.

2. Capacitors (C15 and C16 - 22uF):

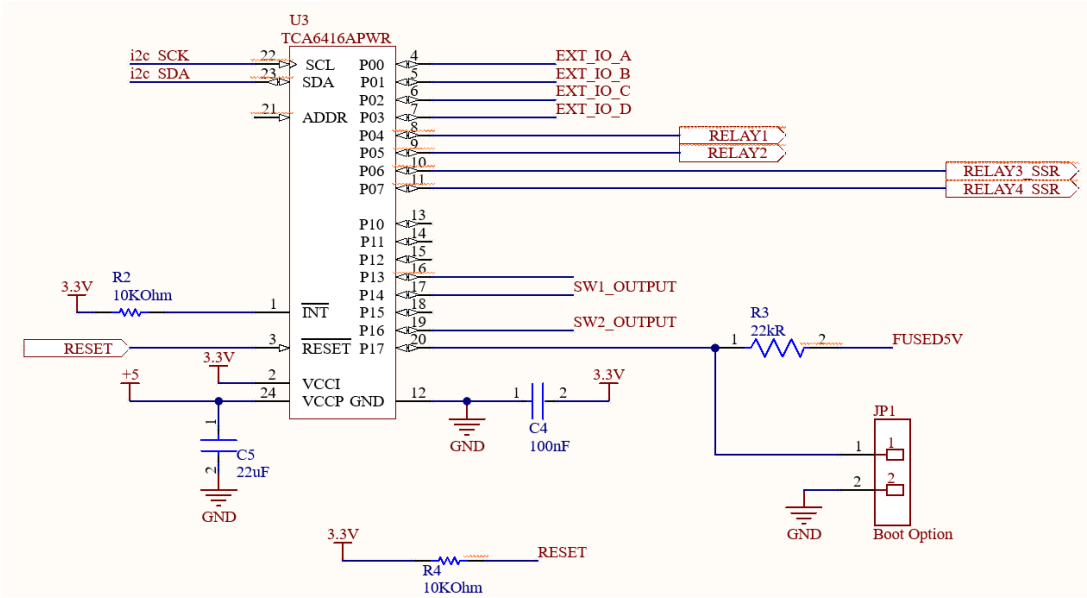
- C15 and C16 are decoupling capacitors that stabilize the input and output voltages by filtering out noise and providing a smooth voltage supply.
- C15 is connected between the VIN pin and ground.
- C16 is connected between the VOUT pin and ground.

Overall Functionality:

- The circuit ensures that power is only delivered to the battery management system if the input voltage is correctly polarized.
- It protects the system from reverse polarity and overcurrent conditions.
- It provides a stable +3.3V output from a +5V input, suitable for powering components that require a lower voltage.

By incorporating these components, the circuit enhances the reliability and safety of the battery management system.

3. IO Expander



This schematic illustrates the integration of TCA6416A I/O expander within the battery management system (BMS).

❖ Functional Overview:

1. I2C Communication:

- Pins 22 (SCL) and 23 (SDA): These pins facilitate communication with the microcontroller via the I2C bus, allowing the microcontroller to control and monitor the states of the GPIO pins on the I/O expander.

2. General Purpose Input/Output (GPIO) Expansion:

- Pins P00 – P17 The TCA6416A provides 16 GPIO pins, which can be configured individually as either inputs or outputs. These pins are connected to various external components:
- EXT_IO_A to EXT_IO_D (Pins 4-7): General-purpose external I/O lines.
- RELAY1 and RELAY2 (Pins 8 and 9): Control signals for mechanical relays, used to switch high power loads or other circuits.
- RELAY3_SSR and RELAY4_SSR (Pins 10 and 11): Control signals for solid-state relays (SSR), offering faster switching and longer life compared to mechanical relays.
- SW1_OUTPUT and SW2_OUTPUT (Pins 18 and 19): Output signals to control or indicate the status of switches, for user inputs or control signals for other devices.

3. Power and Ground Connections:

- VCCI (Pin 24): 3.3V supply voltage for the device's logic.
- VCCP (Pin 2): 5V supply voltage, likely used for driving the external components connected to the GPIOs.
- GND (Pin 1 and others): Common ground reference for the circuit.
- Capacitors:
- C4 (100nF): Provides noise filtering and stabilization for the power supply.
- C5 (22uF): Larger decoupling capacitor to smooth out voltage variations and provide

stable power.

4. Control and Configuration:

- RESET (Pin 3): This pin is used to reset the I/O expander. The reset circuit includes a 10kΩ pull-up resistor (R2) and a RESET switch, allowing manual reset of the device.
- INT (Pin 15): The interrupt pin can be used to alert the microcontroller of any changes in the state of the input pins, enabling efficient event-driven programming.

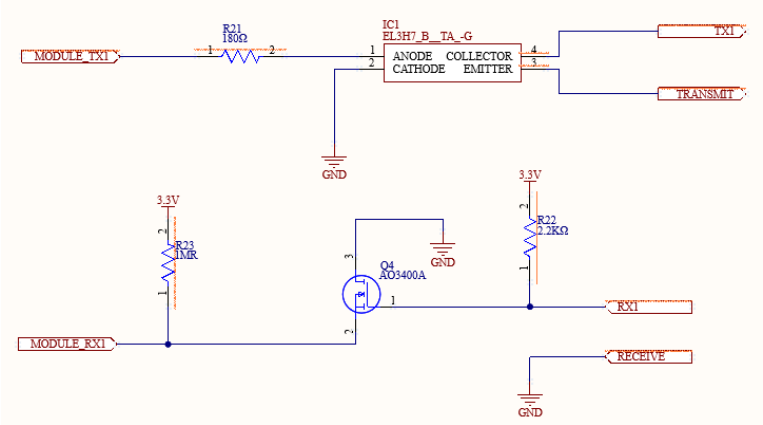
5. Boot Option (JP1):

- JP1 Connector: Provides a jumper configuration for boot options, which might be used to select different operational modes or configurations for the system.
- R3 (22kΩ resistor): Pull-up resistor associated with the boot option selection, ensuring a defined logic level when the jumper is not connected.

❖ Overall Functionality:

The TCA6416A I/O expander significantly increases the I/O capacity of the BMS, enabling it to interface with a variety of external components such as relays, switches, and other digital inputs/outputs. This extension of I/O capability allows the BMS to monitor and control more aspects of the battery system, enhancing its functionality and flexibility. Through the I2C interface, the microcontroller can easily manage these additional I/O points, making the system more scalable and adaptable to complex requirements.

4. Standard module communication circuit



This schematic illustrates the standard module communication circuit in the BMS. The circuit facilitates the communication between the BMS and external modules, enabling data transmission and reception.

❖ Key Components and Their Functionality:

1. Optocoupler (IC1 - EL3H7):

- Pins 1 (Anode) and 2 (Cathode): The input side of the optocoupler, where the signal from the BMS is applied. The optocoupler isolates the transmitting circuit electrically from the receiving circuit.

- Pins 3 (Collector) and 4 (Emitter): The output side of the optocoupler, where the isolated signal is transmitted to the external module.
- Function: The optocoupler provides galvanic isolation, protecting the BMS from voltage spikes and noise that might originate from the external modules. It ensures that the signal integrity is maintained across the isolation barrier.

2. Resistors:

- R21 (180Ω): Limits the current through the optocoupler when transmitting data from the BMS (TXI).
- R22 (22kΩ): Pull-down resistor ensuring that the RXI line remains at a known logic level when the transistor is not conducting.
- R23 (1MΩ): Pull-up resistor for the gate of the MOSFET, ensuring it remains off when not driven by the optocoupler.
- Function: These resistors are critical for current limiting, pulling voltages to defined levels, and ensuring stable operation of the circuit components.

3. MOSFET (Q4 - AO3400A):

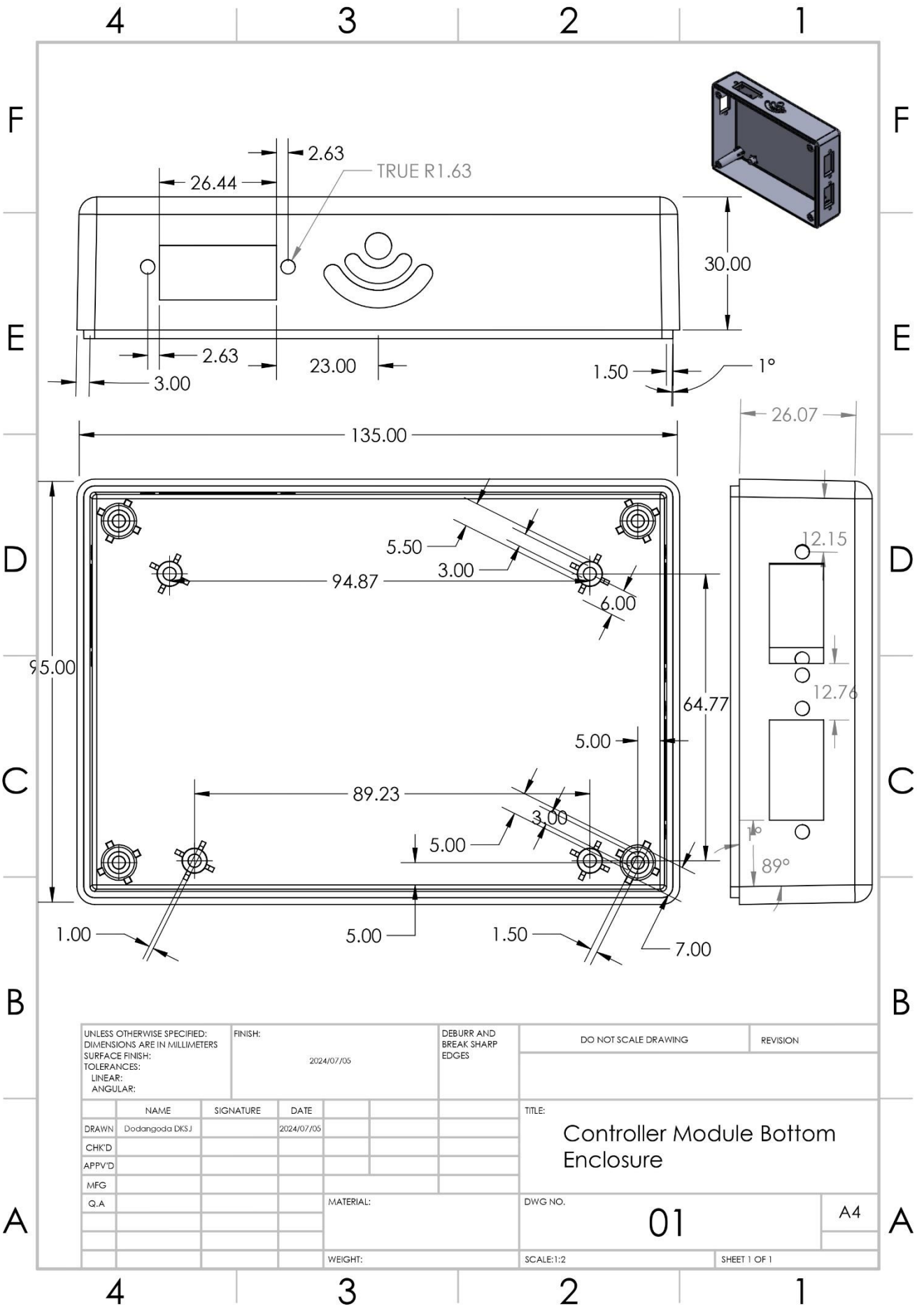
Acts as a switch that is controlled by the optocoupler. When the optocoupler is activated by the BMS transmit signal, the MOSFET allows current to flow, transmitting the signal to the receiving module.

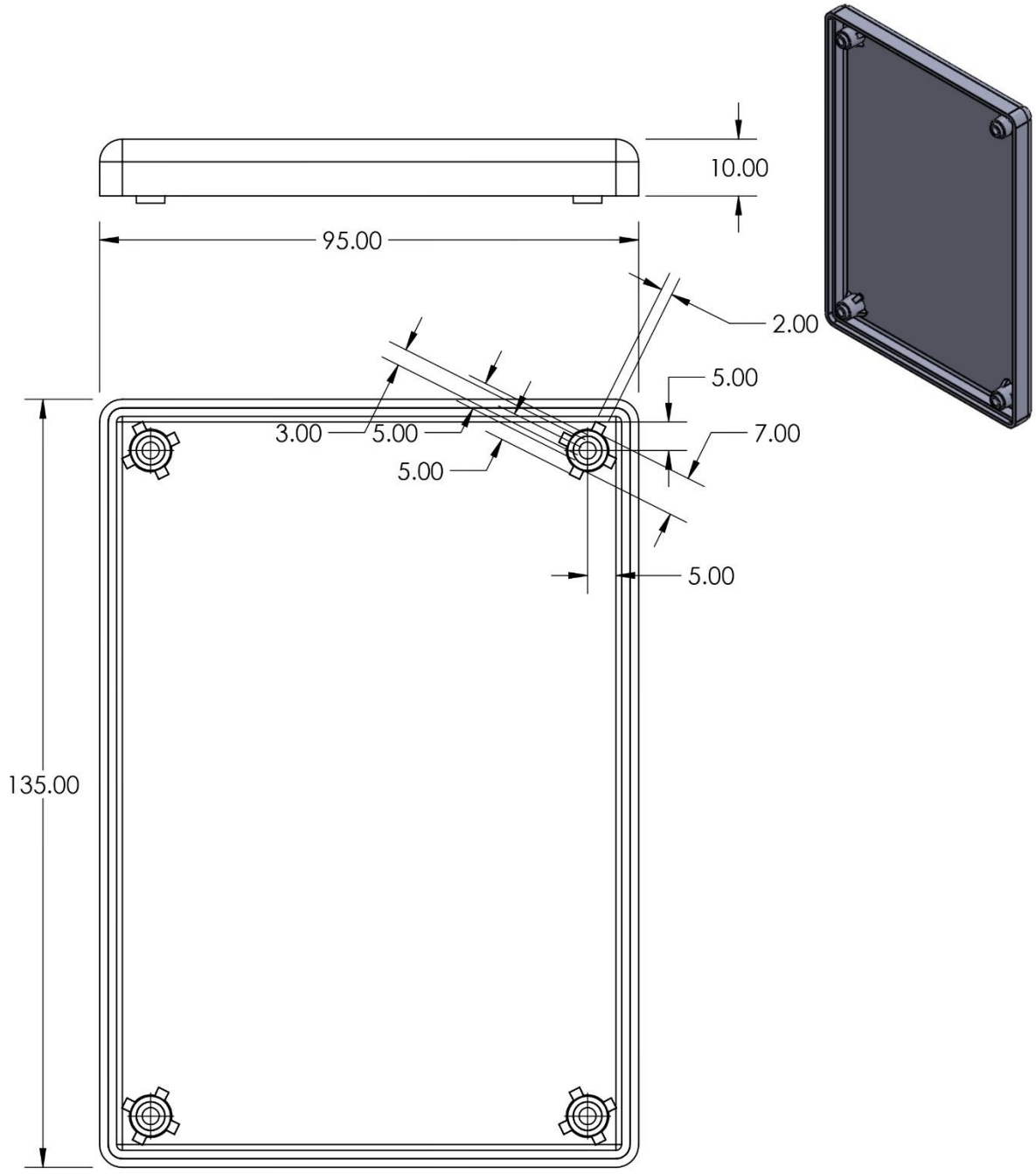
4. Communication Lines:

- MODULE TXI: Input from the BMS transmitting module.
- TXI: Transmit line to the external module after passing through the optocoupler.
- MODULE RXI: Input from the receiving module to the BMS.
- RXI: Receive line from the external module, controlled by the MOSFET.
- TRANSMIT and RECEIVE: External connectors for communication with external modules.

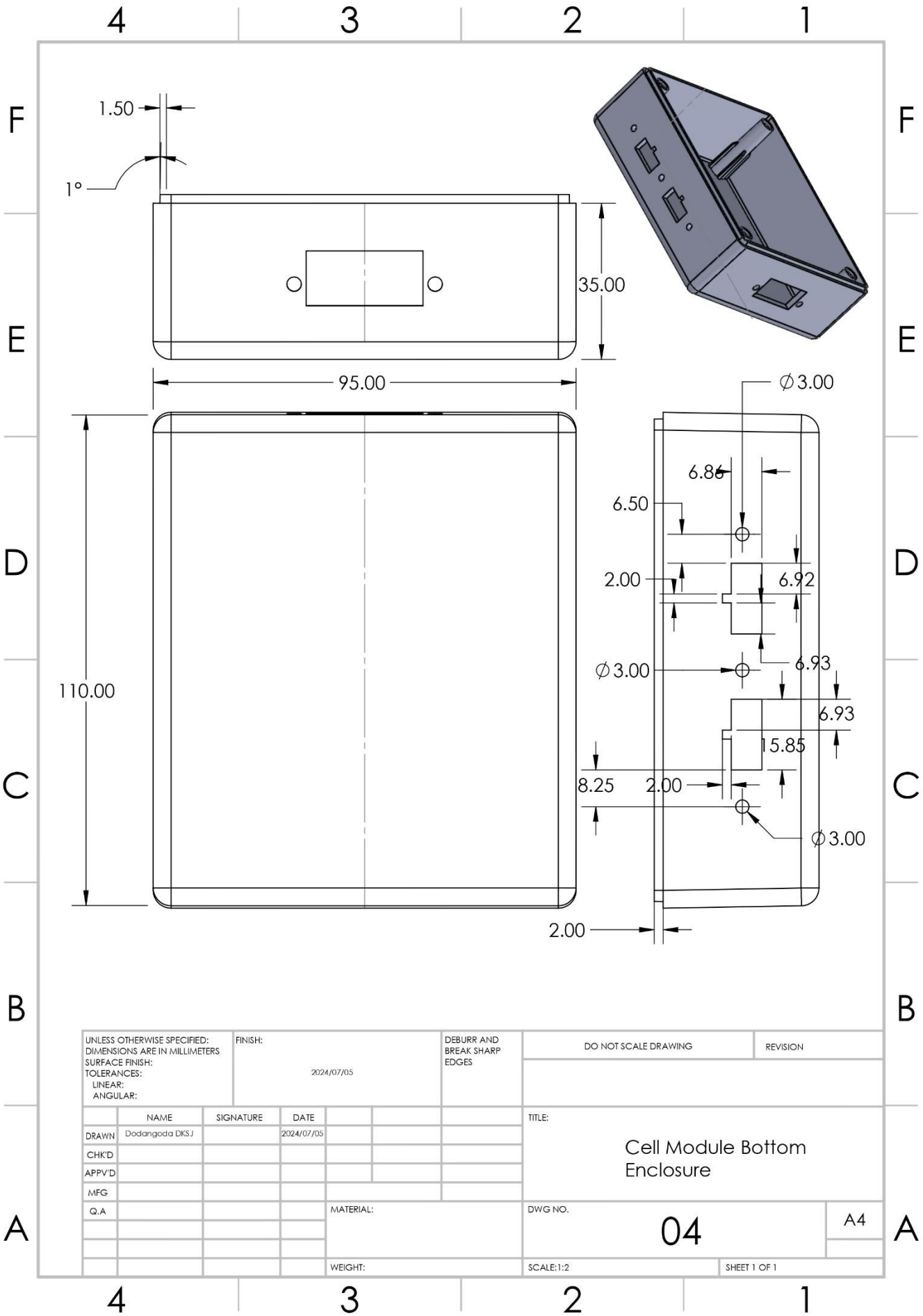
This circuit provides a robust and isolated communication interface for the battery management system, ensuring safe and reliable data exchange between the BMS and external modules. The use of an optocoupler ensures electrical isolation, protecting the BMS from potential electrical disturbances, while the MOSFET facilitates controlled signal reception. The resistors ensure proper current flow and voltage levels for stable operation.

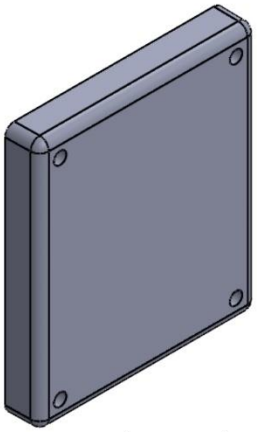
Enclosure Design Details

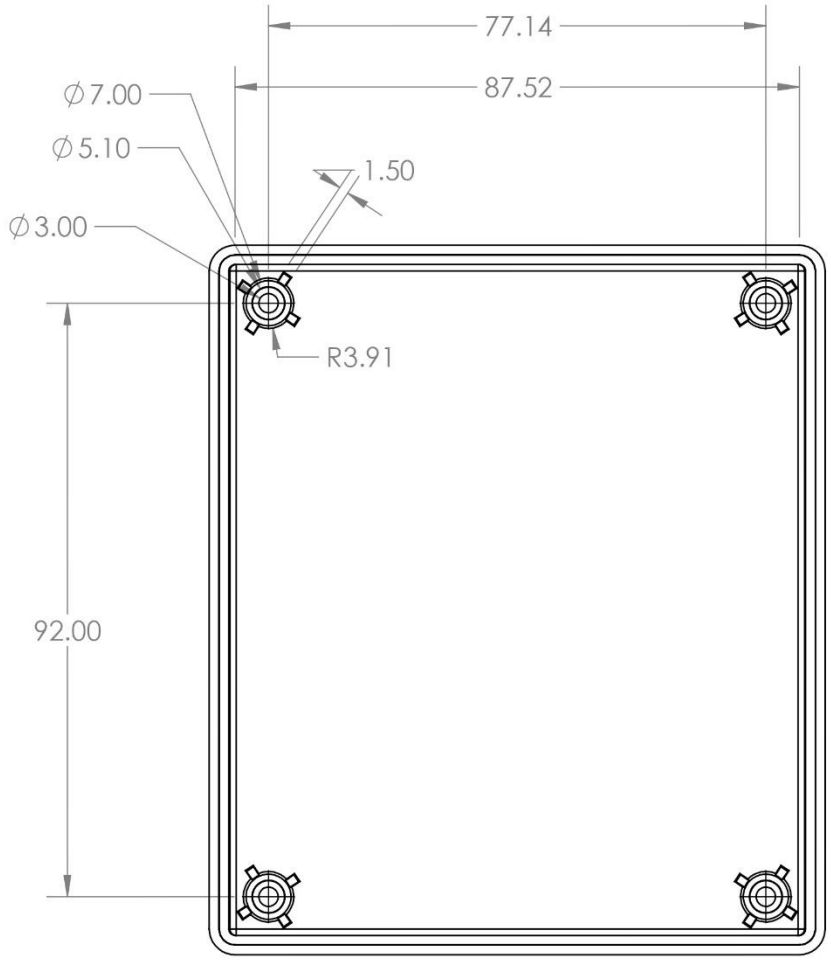


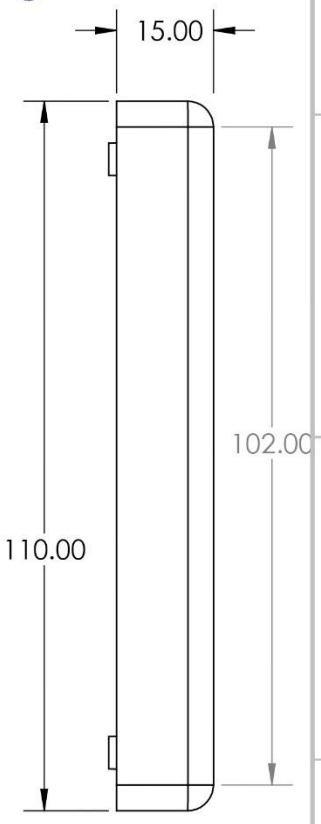


UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:		FINISH: 2024/07/05			DEBURR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING	REVISION
	NAME	SIGNATURE	DATE			TITLE:	
DRAWN	Dodangoda DKSJ		2024/07/05			Controller Module Top Enclosure	
CHK'D							
APPV'D							
MFG							
Q.A				MATERIAL:		DWG NO. 02	A4
				WEIGHT:		SCALE:1:1	SHEET 1 OF 1


4
3
2
1
F
E
D
C
B
A
1.50
1°
35.00
95.00
110.00
∅3.00
6.86
6.50
2.00
6.92
∅3.00
6.93
6.93
15.85
8.25
2.00
∅3.00
2.00
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:
FINISH:
2024/07/05
DEBURR AND BREAK SHARP EDGES
DO NOT SCALE DRAWING
REVISION
NAME
SIGNATURE
DATE
DRAWN
Dodangoda DKSJ
2024/07/05
CHK'D
APPV'D
MFG
Q.A
MATERIAL:
WEIGHT:
TITLE:
Cell Module Bottom Enclosure
DWG NO.
04
A4
SCALE:1:2
SHEET 1 OF 1







UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:		FINISH: 2024/07/05			DEBURR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING	REVISION
	NAME	SIGNATURE	DATE			TITLE:	
DRAWN	Dodangoda DKSJ		2024/07/05			Cell Module Top Enclosure	
CHK'D							
APPV'D							
MFG							
Q.A				MATERIAL:		DWG NO. 03	A4
				WEIGHT:		SCALE:1:2	SHEET 1 OF 1

Programming

- In this section we will focus on programming of the **ATTINY1624-SSU** Microcontroller unit manufactured by Microchip.
- **AVR Port Manipulation method** was used while **Microchip Studio** was used as the programming software.
- The following sections from the datasheet were used for the correct implementation of the code.

17.3 Functional Description

17.3.1 Initialization

After Reset, all outputs are tri-stated, and digital input buffers enabled even if there is no clock running.

The following steps are all optional when initializing PORT operation:

- Enable or disable the output driver for pin Pxn by respectively writing '1' to bit n in the PORTx.DIRSET or PORTx.DIRCLR register
- Set the output driver for pin Pxn to high or low level respectively by writing '1' to bit n in the PORTx.OUTSET or PORTx.OUTCLR register
- Read the input of pin Pxn by reading bit n in the PORTx.IN register
- Configure the individual pin configurations and interrupt control for pin Pxn in PORTx.PINnCTRL

17.3.2.1 Basic Functions

Each pin group x has its own set of PORT registers. I/O pin Pxn can be controlled by the registers in PORTx.

To use pin number n as an output, write bit n of the PORTx.DIR register to '1'. This can be done by writing bit n in the PORTx.DIRSET register to '1', which will avoid disturbing the configuration of other pins in that group. The n^{th} bit in the PORTx.OUT register must be written to the desired output value.

Similarly, writing a PORTx.OUTSET bit to '1' will set the corresponding bit in the PORTx.OUT register to '1'. Writing a bit in PORTx.OUTCLR to '1' will clear that bit in PORTx.OUT to '0'. Writing a bit in PORTx.OUTTGL or PORTx.IN to '1' will toggle that bit in PORTx.OUT.

To use pin n as an input, bit n in the PORTx.DIR register must be written to '0' to disable the output driver. This can be done by writing bit n in the PORTx.DIRCLR register to '1', which will avoid disturbing the configuration of other pins in that group. The input value can be read from bit n in the PORTx.IN register as long as the ISC bit is not set to INPUT_DISABLE.

Writing a bit to '1' in PORTx.DIRTGL will toggle that bit in PORTx.DIR and toggle the direction of the corresponding pin.

17.5.1 Data Direction

Name: DIR
Offset: 0x00
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	DIR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DIR[7:0] Data Direction
This bit field controls the output driver for each PORTx pin.
This bit field does not control the digital input buffer. The digital input buffer for pin n (Pxn) can be configured in the Input/Sense Configuration (ISC) bit field in the Pin n Control (PORTx.PINnCTRL) register.
The available configuration for each bit n in this bit field is shown in the table below.

Value	Description
0	Pxn is configured as an input-only pin, and the output driver is disabled
1	Pxn is configured as an output pin, and the output driver is enabled

17.5.2 Data Direction Set

Name: DIRSET
Offset: 0x01
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	DIRSET[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DIRSET[7:0] Data Direction Set
This bit field controls the output driver for each PORTx pin, without using a read-modify-write operation.
Writing a '0' to bit n in this bit field has no effect.
Writing a '1' to bit n in this bit field will set the corresponding bit in PORTx.DIR, which will configure pin n (Pxn) as an output pin and enable the output driver.
Reading this bit field will return the value of PORTx.DIR.

17.5.5 Output Value

Name: OUT
Offset: 0x04
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	OUT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – OUT[7:0] Output Value
This bit field controls the output driver level for each PORTx pin.
This configuration only has an effect when the output driver (PORTx.DIR) is enabled for the corresponding pin.
The available configuration for each bit n in this bit field is shown in the table below.

Value	Description
0	The pin n (Pxn) output is driven low
1	The Pxn output is driven high

17.5.3 Data Direction Clear

Name: DIRCLR
Offset: 0x02
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	DIRCLR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DIRCLR[7:0] Data Direction Clear
This bit field controls the output driver for each PORTx pin, without using a read-modify-write operation.
Writing a '0' to bit n in this bit field has no effect.
Writing a '1' to bit n in this bit field will clear the corresponding bit in PORTx.DIR, which will configure pin n (Pxn) as an input-only pin and disable the output driver.
Reading this bit field will return the value of PORTx.DIR.

- For more details:

https://www.alldatasheet.com/datasheet-pdf/pdf/1425243/MICROCHIP/ATTINY1624-SSU.html

- The **“diybms_tinyAVR2.cpp”** file provides a comprehensive description and implementation of various hardware-related functionalities critical for the Battery Management System (BMS) on the tinyAVR2 microcontroller platform.

1. **Pin Configurations**: Detailed setup of the microcontroller's pins, specifying their roles (input/output) and functionalities. This ensures that each pin is correctly assigned for its intended purpose.

2. **ADC Configurations**: Configuration of the Analog-to-Digital Converter (ADC) settings to facilitate accurate analog signal readings. This involves setting up the ADC resolution, reference voltage, and input channels to read various analog parameters.

3. **ADC Readings**: Implementation of routines to perform ADC conversions and retrieve the analog values. These readings are crucial for monitoring the battery's voltage and temperature.

4. **LED Functionalities**: Management of LED indicators to provide visual feedback on the system's status. This includes configuring the pins connected to the LEDs and controlling their behavior (e.g., blinking patterns) to indicate different operating conditions, such as normal operation, warnings, or errors.

```
/*


CELL MODULE FOR ATTINY1624

(c)2019 to 2021 Stuart Pittaway

MODIFIED BY
2024 Nidula Gunawardana

COMPILE THIS CODE USING MICROCHIP STUDIO

LICENSE
Attribution-NonCommercial-ShareAlike 2.0 UK: England & Wales (CC BY-NC-SA 2.0
UK)
https://creativecommons.org/licenses/by-nc-sa/2.0/uk/

* Non-Commercial — You may not use the material for commercial purposes.
* Attribution — You must give appropriate credit, provide a link to the license,
  and indicate if changes were made.
  You may do so in any reasonable manner, but not in any way that suggests the
    licensor endorses you or your use.
* ShareAlike — If you remix, transform, or build upon the material, you must
  distribute your
  contributions under the same license as the original.
* No additional restrictions — You may not apply legal terms or technological
  measures
  that legally restrict others from doing anything the license permits.
*/
/*



/*

HARDWARE ABSTRACTION CODE FOR tinyAVR2

*/

#include <avr/io.h>
#include <util/delay.h>

// ATtiny1624 specific register manipulation code

void FlashNotificationLed(size_t times, uint32_t milliseconds) {
    for (size_t i = 0; i < times; i++) {
        PORTA.OUTSET = 0x40; // PA6 = PIN6_bm (0x40) NotificationLedOn
        _delay_ms(milliseconds);
        PORTA.OUTCLR = 0x40; // PA6 = PIN6_bm (0x40) NotificationLedOff
        _delay_ms(milliseconds);
    }
}

void PowerOn_Notification_led() {
    FlashNotificationLed(4, 150);
}
```

```
void double_tap_Notification_led() {
    FlashNotificationLed(2, 50);
}

void ConfigurePorts() {
    // pin out
    // AVR PIN / ARDUINO PIN MAPPING
    // PB0 /7= ENABLE
    // PB1 /6= DUMP LOAD ENABLE
    // PB2 /5= TXD
    // PB3 /4= RXD
    // PA0 /11= RESET
    // PA1 /8= REF_ENABLE
    // PA2 /9= NOT CONNECTED
    // PA3 /10= EXTERNAL TEMP SENSOR (ADC) (ADC0=AIN3)
    // PA4 /0= VOLTAGE INPUT (ADC) (ADC0=AIN4)
    // PA5 /1= VREFERENCE (ADC) (VREFA/ ADC0=AIN5)
    // PA6 /2= NOTIFICATION LED
    // PA7 /3= INTERNAL TEMP SENSOR (ADC)(ADC0=AIN7)

    // Set Port A digital outputs
    PORTA.DIRSET = 0x42; // PIN1_bm | PIN6_bm | PIN2_bm = 0x42

    // Set Port B digital outputs
    PORTB.DIRSET = 0x07; // PIN0_bm | PIN1_bm | PIN2_bm = 0x07
    // Set RX as input
    PORTB.DIRCLR = 0x08; // PIN3_bm = 0x08

    // Set Port A analogue inputs
    PORTA.DIRCLR = 0x88; // PIN3_bm | PIN7_bm = 0x88

    // Disable digital input buffer for unused pins and analog inputs
    PORTA.PIN0CTRL = 0x03; // PORT_ISC_INPUT_DISABLE_gc = 0x03
    PORTA.PIN1CTRL = 0x03;
    PORTA.PIN2CTRL = 0x03;
    PORTA.PIN3CTRL = 0x03;
    PORTA.PIN4CTRL = 0x03;
    PORTA.PIN5CTRL = 0x03;
    PORTA.PIN6CTRL = 0x03;
    PORTA.PIN7CTRL = 0x03;

    PORTB.PIN0CTRL = 0x03;
    PORTB.PIN1CTRL = 0x03;

    // Step 1: Enable ADC
    ADC0.CTRLA = 0x01; // ADC_ENABLE_bm = 0x01
    // PRESC[3:0], DIV16 = 5Mhz/2 = 2500000hz
    ADC0.CTRLB = 0x01; // ADC_PRESC_DIV2_gc = 0x01
    // SAMPDUR[7:0]
    ADC0.CTRLE = 0x80; // 128
    // WINSRC / WINCM[2:0]
    ADC0.CTRLD = 0x00;
    ADC0.PGACTRL = 0x00;

    // Set pins to initial state
```

```
    PORTB.OUTCLR = 0x02; // DumpLoadOff, PIN1_bm = 0x02
    PORTA.OUTCLR = 0x02; // ReferenceVoltageOff, PIN1_bm = 0x02
    PORTA.OUTCLR = 0x80; // TemperatureVoltageOff, PIN7_bm = 0x80
    PORTA.OUTCLR = 0x40; // NotificationLedOff, PIN6_bm = 0x40
}

uint16_t BeginADCReading(uint8_t mode) {
    uint16_t value = 0;

    // Enable ADC
    ADC0.CTRLA = 0x01; // ADC_ENABLE_bm = 0x01

    // TIMEBASE[4:0] / REFSEL[2:0]
    ADC0.CTRLC = 0x04; // TIMEBASE_1US = 0x04 | ADC_REFSEL_VDD_gc = 0x00

    // Take multiple samples (over sample)
    ADC0.COMMAND = 0x88; // ADC_MODE_BURST_SCALING_gc = 0x80 |
      ADC_START_IMMEDIATE_gc = 0x08
    while (!(ADC0.INTFLAGS & 0x01)); // ADC_RESRDY_bm = 0x01
    value = (uint16_t)ADC0.RESULT;

    // Switch off ADC
    ADC0.CTRLA &= ~0x01; // ADC_ENABLE_bm = 0x01

    return value;
}
```

- The main.cpp file in this project serves as the core component responsible for executing critical monitoring and control functionalities. Its primary tasks include:
  - **Battery Voltage Monitoring**: Continuously measures the battery voltage to ensure it remains within safe and optimal operating ranges. This is crucial for maintaining the health and longevity of the battery, as well as ensuring the reliability of the device it powers.
  - **Temperature Monitoring**: Tracks the temperature of the battery and internal circuits. This is essential for preventing overheating, which could lead to component damage or failure, and for optimizing performance by maintaining an ideal temperature range.
  - **PID Controller Implementation**: Utilizes a Proportional-Integral-Derivative (PID) controller to regulate both the battery voltage and the temperatures of the battery and internal circuits.

- The PID controller adjusts the system's operational parameters in real-time to maintain these values within their desired setpoints. It does this by:
  - Proportional Control: Responding to the current error between the desired setpoint and the actual value.
  - Integral Control: Addressing the accumulation of past errors to eliminate steady-state offset.
  - Derivative Control: Predicting future errors based on the current rate of change, helping to dampen oscillations and improve stability.

- By integrating these functionalities, the main.cpp file ensures robust and efficient management of the system's power and thermal conditions, thereby enhancing the reliability and performance of the overall device.

```
/*


CELL MODULE FOR ATTINY1624

(c)2019 to 2021 Stuart Pittaway

MODIFIED BY
2024 Nidula Gunawardana

COMPILE THIS CODE USING MICROCHIP STUDIO

LICENSE
Attribution-NonCommercial-ShareAlike 2.0 UK: England & Wales (CC BY-NC-SA 2.0
UK)
https://creativecommons.org/licenses/by-nc-sa/2.0/uk/

* Non-Commercial — You may not use the material for commercial purposes.
* Attribution — You must give appropriate credit, provide a link to the license,
  and indicate if changes were made.
  You may do so in any reasonable manner, but not in any way that suggests the
    licensor endorses you or your use.
* ShareAlike — If you remix, transform, or build upon the material, you must
  distribute your
  contributions under the same license as the original.
* No additional restrictions — You may not apply legal terms or technological
  measures
  that legally restrict others from doing anything the license permits.
*/
/*

*/

#define F_CPU 2000000UL // 2 MHz clock speed

#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/wdt.h>
#include <util/delay.h>
#include <avr/eeprom.h>
#include <diybms_tinyAVR2.h>
#include <FastPID.h>
#include <SerialEncoder.h>
#include <packet_processor.h>

#define RX_BUFFER_SIZE 64

uint8_t SerialPacketReceiveBuffer[8 + sizeof(PacketStruct)];

SerialEncoder PacketSerial;

CellModuleConfig Config;

PacketProcessor Processor(&Config);

volatile bool WatchdogTriggered = false;
```

```
volatile uint8_t InterruptCounter = 0;
volatile uint16_t PulsePeriod = 0;
volatile uint16_t OnPulseCount = 0;

void DefaultConfig()
{
  Config.Calibration = 1.0;
  Config.BypassTemperatureSetPoint = 65;
  Config.BypassThresholdmV = 4100;
}

void watchdog()
{
  WatchdogTriggered = true;
  Processor.IncrementWatchdogCounter();
}

void onPacketReceived()
{
  diyBMSHAL::EnableSerial0TX();

  if (Processor.onPacketReceived((PacketStruct *)SerialPacketReceiveBuffer))
  {
    diyBMSHAL::NotificationLedOn();
  }

  PacketSerial.sendBuffer(SerialPacketReceiveBuffer);

  diyBMSHAL::FlushSerial0();
  diyBMSHAL::NotificationLedOff();
}

FastPID PID(5.0, 1.0, 0.1, 3, 8, false);

void ValidateConfiguration()
{
  if (Config.Calibration < 0.8 || Config.Calibration > 10.0)
  {
    Config.Calibration = 1.0;
  }

  if (Config.BypassTemperatureSetPoint > DIYBMS_MODULE_SafetyTemperatureCutoff)
  {
    Config.BypassTemperatureSetPoint = DIYBMS_MODULE_SafetyTemperatureCutoff - 10;
  }
}

void StopBalance()
{
  Processor.WeAreInBypass = false;
  Processor.bypassCountDown = 0;
  Processor.bypassHasJustFinished = 0;
  Processor.PWMSetPoint = 0;
  Processor.SettingsHaveChanged = false;
```

```
  OnPulseCount = 0;
  PulsePeriod = 0;

  diyBMSHAL::StopTimer1();
  diyBMSHAL::DumpLoadOff();
}

void setup()
{
  wdt_disable();
  wdt_reset();

  bool JustPoweredUp = true;

  if ((GPIOR0 & 0x08) == 0x08)
  {
    watchdog();
    JustPoweredUp = false;
  }

  diyBMSHAL::SetPrescaler();
  diyBMSHAL::SetWatchdog8sec();
  diyBMSHAL::ConfigurePorts();

  if (JustPoweredUp)
  {
    diyBMSHAL::PowerOn_Notification_led();
  }
  if (!Settings::ReadConfigFromEEPROM((uint8_t *)&Config, sizeof(Config),
    EEPROM_CONFIG_ADDRESS))
  {
    DefaultConfig();
  }

  ValidateConfiguration();

  PID.setOutputRange(0, 255);

  StopBalance();

  // Set up UART
  UBRR0 = F_CPU / (8 * DIYBMSBAUD) - 1;
  UCSR0A |= (1 << U2X0);
  UCSR0B |= (1 << RXEN0) | (1 << TXEN0);
  UCSR0C |= (1 << UCSZ01) | (1 << UCSZ00);

  PacketSerial.begin(&onPacketReceived, sizeof(PacketStruct),
    SerialPacketReceiveBuffer, sizeof(SerialPacketReceiveBuffer));
}

void BalanceTimer()
{
  InterruptCounter++;
  PulsePeriod++;
  if (InterruptCounter == 255)
  {
```

```
    InterruptCounter = 0;
  }
  if (InterruptCounter <= Processor.PWMSetPoint)
  {
    diyBMSHAL::DumpLoadOn();
    OnPulseCount++;
  }
  else
  {
    diyBMSHAL::DumpLoadOff();
  }

  if (PulsePeriod == 1000 && OnPulseCount != 0)
  {
    float CurrentmA = ((float)Processor.CellVoltage() / (float)LOAD_RESISTANCE);

    float milliAmpHours = (CurrentmA * ((float)OnPulseCount / (float)1000.0)) *
      (1.0 / 3600.0);

    Processor.MilliAmpHourBalanceCounter += milliAmpHours;

    OnPulseCount = 0;
    PulsePeriod = 0;
  }
}

ISR(TCA0_OVF_vect)
{
  BalanceTimer();
  TCA0.SINGLE.INTFLAGS = 0x01;
}

inline void identifyModule()
{
  if (Processor.identifyModule > 0)
  {
    diyBMSHAL::NotificationLedOn();
    Processor.identifyModule--;

    if (Processor.identifyModule == 0)
    {
      diyBMSHAL::NotificationLedOff();
    }
  }
}

void loop()
{
  wdt_reset();
  identifyModule();

  if (Processor.SettingsHaveChanged)
  {
    StopBalance();
  }
```

```
  if (WatchdogTriggered)
  {
    diyBMSHAL::double_tap_Notification_led();
    StopBalance();
  }

  diyBMSHAL::TemperatureVoltageOn();

  Processor.TakeAnAnalogueReading(ADC_INTERNAL_TEMP);

  if (Processor.bypassCountDown == 0)
  {
    Processor.TakeAnAnalogueReading(ADC_EXTERNAL_TEMP);

    diyBMSHAL::ReferenceVoltageOn();

#if (SAMPLEAVERAGING == 1)
    Processor.TakeAnAnalogueReading(ADC_CELL_VOLTAGE);
#else
    for (size_t i = 0; i < 5; i++)
    {
      Processor.TakeAnAnalogueReading(ADC_CELL_VOLTAGE);
    }
#endif
  }

  diyBMSHAL::ReferenceVoltageOff();
  diyBMSHAL::TemperatureVoltageOff();

  if (WatchdogTriggered == true && !(UCSR0A & (1 << RXC0)))
  {
  }
  else
  {
    for (size_t i = 0; i < 200; i++)
    {
      PacketSerial.checkInputStream();
      _delay_ms(1);
    }
  }

  int16_t InternalTemp = Processor.InternalTemperature();

  if (InternalTemp > DIYBMS_MODULE_SafetyTemperatureCutoff || InternalTemp >
    (Config.BypassTemperatureSetPoint + 10))
  {
    PID.clear();
    StopBalance();
  }

  if (Processor.BypassCheck() && InternalTemp <
    DIYBMS_MODULE_SafetyTemperatureCutoff)
  {
    if (!Processor.WeAreInBypass)
    {
      Processor.WeAreInBypass = true;
```

```
      Processor.bypassCountDown = 50;
      Processor.bypassHasJustFinished = 0;

      diyBMSHAL::StartTimer1();
      Processor.PWMSetPoint = 0;
    }
  }

  if (Processor.bypassCountDown > 0)
  {
    if (InternalTemp < (Config.BypassTemperatureSetPoint - 6))
    {
      Processor.PWMSetPoint = 0xFF;
    }
    else
    {
      Processor.PWMSetPoint = PID.step(Config.BypassTemperatureSetPoint,
        InternalTemp);

      if (PID.err())
      {
        PID.clear();
        StopBalance();
      }
    }

    Processor.bypassCountDown--;

    if (Processor.bypassCountDown == 0)
    {
      StopBalance();
      Processor.bypassHasJustFinished = 150;
    }
  }

  if (Processor.bypassHasJustFinished > 0)
  {
    Processor.bypassHasJustFinished--;
  }

  WatchdogTriggered = false;

  if (!Processor.WeAreInBypass && Processor.bypassHasJustFinished == 0 && !(UCSR0A
    & (1 << RXC0)))
  {
    PID.clear();
    diyBMSHAL::Sleep();
  }
}

int main(void)
{
  setup();
  while (1)
  {
```

```
    loop();
  }
}
```

- We hereby declare that **Arduino was not used for programming the microcontroller from the beginning.**

Daily Log Entries

No	Date	Log Entry
1	01/02/2024	❖ Evaluated problems in the industry & selected the project. ❖ Created a preliminary project plan outlining resources, phases, tasks, and timeline. ❖ Discussed potential design approaches and technical challenges.
2	07/02/2024	❖ Read articles, papers related to BMS design. ❖ Made sure everyone had a good understanding about the project. ❖ Planned next steps.
3	18/02/2024	❖ Created the stake holder map. ❖ Identified the users of a BMS. ❖ Identified the needs of the users. ❖ Planned the strategies to fulfill these needs. ❖ Planned next steps.
4	22/02/2024	❖ Did a thorough analysis of how we are going to fulfill these needs. ❖ Identified different sub systems required to fulfill the above needs.
5	6/03/2024	❖ Conceptual designs. ❖ Created the block diagram view of the project. ❖ Created possible conceptual designs as sketches.
6	11/03/2024	❖ Started Circuit Design & Enclosure design. • Began designing the circuit layout. • Sketched initial circuit layouts on paper, outlining potential power and signal flow pathways. • Discussed the use of protection circuits. • Considered possible safety mechanisms. • Sketched a detailed view of the enclosure.
7	13/03/2024	❖ Continued Circuit Design & Enclosure design. • Continued refining the circuit design. • Incorporated microcontroller and communication interfaces. • Drew rough schematics of the circuit layout. • Continued sketching the enclosure.

8	14/03/2024	❖ Schematic Design & Enclosure design • Started creating digital schematics. • Focused on organizing the schematics logically. • Used hierarchical schematic design procedure. • Started designing the enclosure on SolidWorks.
9	15/03/2024	❖ Schematic Design & Enclosure design. • Continued building and refining the schematic design, making sure all key components and connections are represented. • Conducted a review with the team to check for consistency and completeness. • Made necessary adjustments. ❖ Continued designing the enclosure on SolidWorks.
10	20/03/2024	❖ PCB Layout & Enclosure design • Transitioned from schematic design to PCB layout, starting with component placement on the board. • Focused on optimal placement for signal integrity and power management. • Considered size constraints and the placement of key interfaces. ❖ Continued designing the enclosure on SolidWorks.
11	21/03/2024	❖ PCB Layout & Enclosure design • Continued routing connections between components. • Ensured proper trace widths and clearance for power handling and signal lines. ❖ Continued designing the enclosure on SolidWorks.
12	24/03/2024	❖ PCB Layout & Enclosure design • Conducted a detailed design review, checking the PCB layout against the schematic and requirements. • Made final adjustments based on review feedback.
13	25/03/2024	❖ Enclosure Design • Completed the design of the enclosure. • Created the mold for the enclosure.
14	10/04/2024	❖ Ordered necessary components.

15	15/04/2024	❖ PCB Production. • Submitted finalized PCB design files to a manufacturer for production.
16	17/04/2024	❖ Programming the Microcontroller. • Identified the main functions which should be performed by the microcontroller. • Selected suitable software for programming.
17	19/04/2024	❖ Started coding and it was continued parallelly with the following tasks.
18	23/04/2024	❖ PCB arrived.
19	25/04/2024	❖ Components arrived. ❖ Continued programming.
20	29/04/2024	❖ PCB Assembly & Soldering - Part 1 • Began assembling components on the PCB according to the design. • Carefully placed and soldered each component, starting with smaller ones and progressing to larger ones.
21	30/04/2024	❖ PCB Assembly & Soldering - Part 2 • Continued assembly and soldering.
22	31/04/2024	❖ PCB Assembly & Soldering - Part 3 • Completed assembly and soldering of the PCB.
23	06/05/2024	❖ Initial Power-Up & Testing • Conducted the initial power-up tests for the BMS. • Checked for any short circuits, incorrect connections, and component functionality.
24	07/05/2024	❖ Testing & Troubleshooting - Part 1 • Performed tests, including voltage, current, and temperature measurements.