CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Development of Standalone Control Unit for Automotive Battery Management System

Master's thesis in Computer science and engineering

MAITHRI LINGAIAH JAYARAMU

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2020

Master's thesis 2020

Development of Standalone Control Unit for Automotive Battery Management System

MAITHRI LINGAIAH JAYARAMU

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2020

Development of Standalone Control Unit for Automotive Battery Management System
MAITHRI LINGAIAH JAYARAMU

Supervisor: Dr. Per Stenstörm, Department of Computer Science and Engineering
Advisor: Dr. Guenther Mohr, Infineon Technologies AG
Examiner: Dr. Per Larsson-Edefors, Department of Computer Science and Engineering

Master's Thesis 2020
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2020

Development of Standalone Control Unit for Automotive Battery Management System
MAITHRI LINGAIAH JAYARAMU
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Conventional vehicles have made a great impact on climatic changes which in turn has made way for the electric vehicles in the market as they have zero emissions. Electric vehicles run on large number of batteries. Batteries are very sensitive to temperature variations and incorrect charging. The performance of the battery pack is dependent on various battery operating parameters that involve battery temperature, depth of discharge, charge and discharge rates. Monitoring the battery operating parameters is essential for safe and reliable operation of the cells, energy management system and maintenance of the battery.

This thesis is about firmware development for battery monitoring, battery parameters measurement and battery state estimation using a Battery monitoring IC which is part of battery management system(BMS) in hybrid electric vehicles(HEV), plug-in hybrid electric vehicles (PHEV) and battery electric vehicles(BEV). The customized hardware has a small ARM processor integrated into an FPGA which will establish connection with the Battery Monitoring IC and does the required configurations. Certain battery operating parameters are measured and monitored periodically by the Battery Monitoring IC and shunt based measurement system. Based on the calculated state of charge (SoC) further safety decisions are taken. This system is used in an integrated system, thus reduces data processing load for the main controller and also reduces the data traffic. As the system is modular, it is reusable in many other projects. The results of this thesis show that the firmware is capable of doing the battery monitoring, measure the battery operating parameters and perform battery state estimation. The tests implemented on the test bench were to verify the implementation, and to evaluate the battery state estimation algorithm.

Keywords: Battery management system, battery monitoring IC, measurement, battery operating parameters, SoC, safety, battery state estimation.

Acknowledgements

I would like to thank my supervisor at Infineon Technologies AG, Dr.Guenther Mohr for his constant guidance and support during the execution of the master thesis. I would also like to thank Dr. Klaus Hoermaier for his support and sharing his expertise on Battery Monitoring IC. I thank my managers Paul Arts, Christian Lenzhofer and Geoffrey Acres for providing me the opportunity and necessary resources for the execution of master thesis.

Lastly I would like to thank my supervisor at Chalmers, Prof. Dr. Per Stenström and examiner Prof. Dr. Per-Larsson Edefors for their valuable feeback and suggested improvements during the process of thesis.

Maithri Lingaiah Jayaramu, Gothenburg, November 2020

Contents

List of Figures

List of Tables

List of Abbreviations

ADC Analog-to-Digital Converter
AHB Advanced High-performance Bus
AXI Advanced Extensible Interface
BEV Battery Electric Vehicle
BMS Battery Management System
CMSIS Cortex Microcontroller Software Interface Standard
CRC Cyclic Redundancy Check
DTCM Data Tightly Coupled Memory
EMS Energy Management System
EV Electric Vehicle
FPGA Field Programmable Gate Array
GPIO General Purpose Input Output
HEV Hybrid Electric Vehicle
IC Integrated Circuit
IDE Integrated Development Environment
IP Intellectual Property
ITCM Instruction Tightly Coupled Memory
MOB Master On Bottom
MOT Master On Top
NTC Negative Temperature Coefficient
NVIC Nested Vector Interrupt Controller
PHEV Plug-in Hybrid Electric Vehicle
Rx Receiver
SOC State Of Charge
SOH State Of Health
SPI Serial Peripheral Interface
Tx Transmitter
UART Universal Asynchronous Receiver Transmitter

1
Introduction

The demand for electric vehicles(EVs) is increasing day by day because of their zero emission, low noise, continuously increasing oil prices and difficulty in extraction of the fossil fuels. The energy management systems(EMS) in EV are undergoing lot of advancements due to the emerging technologies and constant improvement to succeed in dealing with limited driving range. EMS require innovative technologies like battery management system(BMS) for proper management of energy to yield better performance and safety. Also with the high growth oriented market for EV[1], the high performance of the battery packs need to be monitored and managed for high reliability and safe operation. The BMS will control many energy storage functions, monitor and manage the battery of the EMS in EV. A battery for EV consists of in series connected cells, most popularly lithium-ion cells. A high voltage battery pack usually contains a huge number of cell elements and this introduces some challenges. If one of the cells gets damaged in a large number of cells which are serially connected, then it will damage all the cells. As EV uses high currents and voltage, it has to ensure safety of the passengers or any person as there is no compromise in safety perspective. So we need to monitor all the cells regularly for proper functioning of the battery.

The performance of the battery pack is dependent on various parameters that involve battery temperature, depth of discharge, and charge and discharge rates. Monitoring these battery parameters is very essential for the battery life and also for safe and reliable operation of battery cells as they can overheat and cause explosion by overcharging or could be destroyed by a deep undercharging. Individual electronic components can be used to measure the battery parameters but handling these components on a board and placing them in a compact space over the top of each cell becomes tedious as there are large number of cells connected in series and/or parallel combination. Also the automotive environment is subject to extreme variations in temperature, dust, wind and water, vibrations which can have harsh impact on the electronic components and damage the system.

For safety and reliability perspective, the alternating method would be to use an IC as it will be small, compact and efficient. This will also avoid the complexity with the wires. Infineon Techonologies AG is a global leader in semiconductor devices with their products in the field of automotive, industrial power control, power and sensor systems, connected secure systems and many more. They have developed a new product for the BMS which is a battery monitoring IC specially designed for hybrid electric vehicles(HEV), plug-in hybrid electric vehicles (PHEV) and battery

electric vehicles(BEV) but can also be used for other applications.

Along with monitoring, BMS also has many other functions to perform like controlling the charge and discharge rates, battery modelling, state of charge(SoC) and state of health(SoH) estimation and operation safety[2].

1.1 Aim

The goal of the thesis is to develop firmware for the customized microcontroller unit (MCU) that continuously controls and communicates to the battery monitoring IC (BMS IC) of the battery management system (BMS). This BMS is part of the automotive system. Along with monitoring and measuring the battery parameters which are voltage, temperature and current, the firmware also does processing of the monitored battery parameters to evaluate SoC of the battery based on coulomb counting method[3]. If the evaluated values are found to be out of bounds then the main microcontroller is alerted to take further measures.

1.2 Scope

The control unit contains the main microcontroller along with sub-microcontroller. The sub-microcontroller is the system which is of interest for this thesis as shown in Figure 1.1.

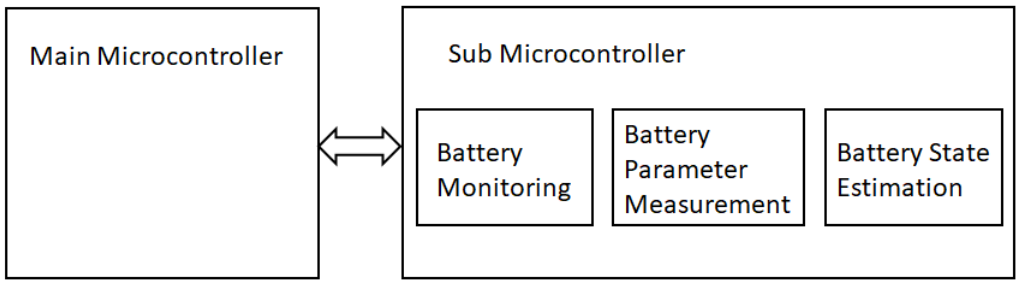

Figure 1.1: System Structure

The proposed design is for the sub-microcontroller that does the monitoring, measurement and state estimation of the battery. The design is done in such a way that with certain small adaptations in the configuration parameters and data storage buffers the firmware can be used to measure and monitor large number of cells. As the control unit hardware is developed in the company, there was no other choice of using a different hardware for this thesis. It is assumed that the control unit hardware remains the same for the future versions. This assumption is made as next version of hardware is not yet available. The system's features can be characterised into five main categories:

- Communication to BMS IC - Different communication protocols are used which are, UART communication to serial terminal, company specific com-

munication protocol over a high speed UART interface, and serial peripheral interface (SPI) to communicate to analog to digital converter (ADC).
- Data acquisition from BMS IC and ADC - Measured voltage and temperature data from the BMS IC registers are read through the high speed UART interface and data from ADC is read through SPI interface.
- Post processing of acquired data - Measured data is in the raw form and it has to be converted to voltage, temperature in Celsius and current in Amperes.
- SoC estimation - Battery state estimation algorithm based on coulomb count method is implemented[3].
- Alerting when there is out of bounds SoC estimation result - This is a safety measure to alert other systems if the SoC is out of bounds.

Similarly, the testing is done on one BMS IC test hardware measuring one cell, and later it can be extended to support the maximum number of BMS ICs to measure and monitor the batteries used in the battery management system of the project.

1.3 Contributions

Battery operating parameters such as voltage, temperature and current have a significant impact on the SoC estimation and operation of the cell. Hence these battery parameters have to be measured from each cell accurately. Earlier methods used to measure these parameters include usage of sensors for measuring each battery parameter[4]. In an automotive system, space is an important factor as there are large number of cells connected in series or parallel connections. Using individual electrical components or sensors to measure these parameters might consume more space, size, increase cost, high power consumption, fault generation and increased wiring connections that add to more delay and noise. Latest methods use battery monitoring IC to reduce these problems. Battery monitoring IC is designed to achieve high battery monitoring accuracy, faster data acquisition and error detection. With these capabilities, the battery monitoring IC enables the microcontroller to efficiently manage the energy storage in the system. The contribution of this thesis work is to provide a methodology to control and communicate with the Infineon's battery monitoring IC and current measurement system, then utilize the measured battery operating parameters to estimate the state of charge of the battery by implementing the SoC algorithm. In addition, the design is implemented in a hardware proof-of-concept prototype to verify the functionality and various tests have been conducted to check the measured parameters and the state of charge estimation.

1.4 Thesis Structure

This thesis report comprises of seven chapters. Chapter 2 describes the BMS and also presents the SoC estimation and the topology of BMS. Chapter 3 gives a brief introduction to the ARM architecture emphasising on Cortex-M1 and its peripherals. Chapter 4 describes the project hardware. Chapter 5 describes the implementation. There is also explanation of the state machine implemented. Chapter 6 presents the results obtained from establishing communication between FPGA board and

single BMS IC evaluation board, voltage, temperature and current measurement, post processing of acquired data and SoC Estimation. Chapter 7 describes the conclusions and scope for future work.

2

Battery Management Systems

In this chapter the main components of the battery management system(BMS) like the different types of batteries, introduction to BMS, battery monitoring, state of charge, state of health, BMS topology are presented.

2.1 Cell and Battery Introduction

The basic unit of a battery is the cell [5]. A battery pack in vehicles consists of large number of cells connected in series and/or parallel combination to provide sufficient voltage and optimal energy. The well known batteries used in the EVs are the Lithium Ion(Li-Ion) batteries. Several other batteries are also used in the EVs like Lead Acid battery, nickel-metal hydride (NiMH) battery and nickel-metal cadmium (NiCd) battery. Among different types of batteries, Lithium Ion and NiMH batteries are the most popular because they provide high energy and are environment friendly.

2.1.1 Lithium Ion(Li-Ion) Cells

Li-Ion cells are currently the most powerful, efficient and high energy cells available in automotive market today[6]. Li-ion cells nominal voltage is 3.2 - 3.7V[2]. Li-Ion cells also have long cycle life capability and extended shelf-life with low self discharge when not in use. Li-Ion battery does not give out any toxic fumes as it is made up of materials that are environment friendly[7]. Li-Ion Cell are made up of cathode, anode, electrolyte and separator[8]. During its operation when the Li-Ion cell is charged, Li-Ions are released from the cathode and then they move through the electrolyte towards the anode. Energy is stored in the cell during this process. During discharging, the Li-Ions move back from anode towards cathode and energy is released that powers the cell. Both temperature and operating voltage of the Li-Ion cell must be kept within the range. If gone beyond bounds then the cell might get permanently damaged or its performance degrades.

2.1.2 Nickel-Metal Hydride (NiMH) Cells

Nickel based batteries have been long used in EVs for their long operating life, high power and good deep discharge. NiMH cells are similar to nickel cadmium (NiCd) cells but NiMH cells are eco-friendly compared to NiCd cells which use highly toxic substance cadmium as electrode. Thus NiMH cells have almost replaced NiCd cells. NiMH cells are made up of a positive electrode, negative electrode and a separator.

Positive electrode is made up of Nickel Oxyhydroxide and negative electrode is made up of hydrogen absorbing alloy. NiMH cells have a nominal voltage of 1.2V. When the cells are not in continuous use, self discharge is high and require regular charging.

2.2 Introduction to BMS

An EV has huge number of cells connected in the battery pack and monitoring each cell voltage in a battery pack is very necessary to provide cell protection. Measuring cell temperature is also important as EV works with high current and overheating of the cells can set the battery on fire. For safe and efficient operation of these cells a BMS is required[7].

The BMS plays a critical role in monitoring the battery, protecting the battery, controlling the performance of the battery and reducing maintenance costs. The BMS in automotive systems is being part of the energy management system performs all the necessary functions which it does in other applications. Along with this it quickly reacts to energy requirements and provides necessary information for other on board systems such as engine management system, vehicle communication systems, climate control system, security and safety systems. Reliability of BMS is important as there are many interferences in an automotive system.

The battery must be charged and discharged within the safe operating region mentioned in the specifications of the battery. BMS should make sure during charging and discharging that the cells are not overcharged or undercharged and send an alarm or turn off the current at the right time to avoid any damage and extend the battery life[5]. Based on the type of cell used in the batteries the BMS adapts to its operating conditions and specifications. If the BMS includes cell balancing then it makes the charge level on all cells to be the same and thus extending the shelf-life of the cell. Another function of BMS is storing the information of the cell. This helps in knowing the age of the battery which can help in maintenance of the battery. BMS uses standard protocols like SPI, CAN, UART etc to communicate with other systems.

2.2.1 Battery Monitoring

Monitoring the battery operating parameters such as voltage, temperature and current from every cell is an important part of BMS. A battery pack is made up of large number of cells connected in series and/or parallel combination. If the cells are connected in series then the current across all cells is the same. Thus when cells are connected in series we have to measure voltage across each cell and current across all the cells and vice-versa for the parallel connection of cells. Measuring the voltage across each cell can be done using a analog to digital converter(ADC) but as there are large number of cells, changing the voltage measuring point across each cell becomes difficult to manage. Using a battery monitoring IC will simplify this process. There are different types of battery monitoring ICs available based on the requirement of the system. Multi-cell battery monitoring IC can measure individual

voltage across multiple cells (12 to 15 cells or even more) connected in series where the range of number of cells depends on the allowed voltage of the used technology. Single cell battery monitoring IC can measure individual voltage across each single cell where cells are connected in series. Temperature measurement can be done using negative temperature coefficient(NTC) thermistor or can also be done using battery monitoring IC with highly accurate on-chip temperature measurement and monitoring.

For current measurement, a shunt based measurement system can be used that has a shunt resistor connected at the end of the battery cells connected in series. The voltage drop across the shunt resistor is input to a differential op-amp and then to an ADC which gives the digital output to the microcontroller unit. Then the firmware running on the microcontroller will read the result from the ADC and provide the current measurement results. By measuring the battery operating parameters we can estimate SoC.

2.2.2 State of Charge (SoC)

SoC is the estimation of the remaining battery capacity expressed in percentage. This will tell us how much longer battery can be used before it needs recharging. Once the capacity of battery is known then we can calculate SoC using the SoC algorithms. The algorithm that will be used is the coulomb counting technique to calculate the SoC using the formula:

$$SoC = SoC(t_0) + \frac{1}{C_r}\int_{t_0}^{t_0+\tau} I_{batt}\,\mathrm{d}t \tag{2.1}$$

where $SoC(t_0)$ is the initial SoC, C_r is the battery capacity given by the manufacturer, I_batt is the battery current[3]. Coulomb counting method is also known as current integration for integrating the charging and discharging current readings over the battery operating periods. During charging and discharging there are losses and also there are self discharging loss for all the batteries. These losses accumulate over a period of time and hence these losses should be considered for better SoC estimation. To know how deeply the battery is discharged, sometimes the depth of discharge (DOD) is used.

$$DOD = 100 - SoC\% \tag{2.2}$$

2.2.3 State of Health (SoH)

In the battery's lifespan, its health, performance and long cycle life capability starts fading gradually from the day it is manufactured due to the aging effect, battery's usage and storage, its internal chemical changes, environment in which it is used and its handling over the time until most of the battery capacity fades and battery becomes unusable. SoH tells how long will the battery last with a capability to deliver the specified performance compared to its initial capacity[10]. SoH can be stated as follows:

$$SoH = \frac{C_m}{C_r}.100\% \tag{2.3}$$

where C_m is the maximal releasable capacity and C_r is the battery capacity[3]. If the battery capacity reduces to 80% of its initial capacity then the battery becomes unusable[11].

2.2.4 Topology

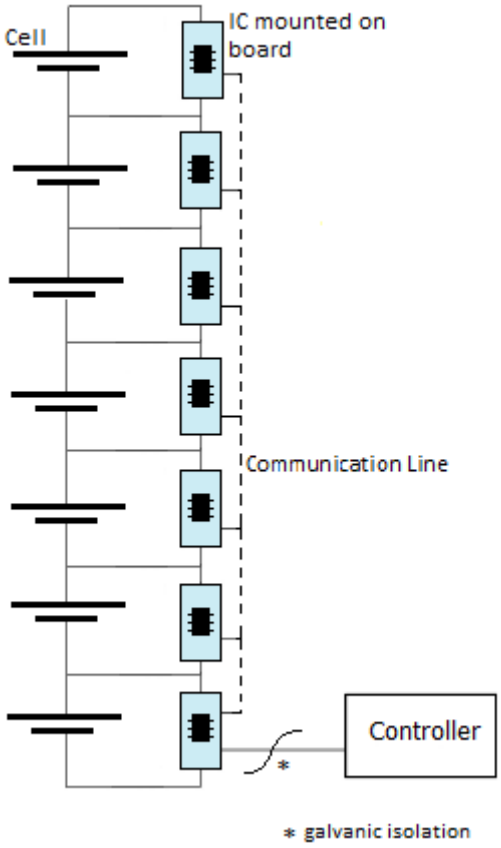

Figure 2.1: BMS Distributed topology

In a system there are large number of batteries and they have to be connected in a particular manner to meet the requirements of high current, energy storage, voltage output and compact space. There are different kinds of BMS topologies that can be categorized into centralized, modular, ring and distributed. Since a battery monitoring IC is used for monitoring each cell, thus a topology similar to distributed topology is used. In distributed topology, as shown in Figure 2.1, each cell is connected with an IC board[5]. In this type of connection where the IC board is placed close to the cell, this reduces the wires and wiring errors and it offers more reliability as there is less monitoring electronics across each cell. Temperature measurement of each cell can be done by the electronics on board or by the IC itself.

3

ARM Architecture

The ARM Cortex M Processor Series is a RISC processor designed for cost and power-efficient microcontrollers. These processors are scalable and easy-to use. The Cortex-M1 processor is intended for deeply embedded applications that need a small processor to be integrated into an FPGA[12]. The instruction set architecture supported is ARMv6-M architecture. There is no floating point hardware in it and thus making Cortex M1 processor to be cost efficient. ARMv6-M is a 32-bit architecture which also supports ARMv6-M 16-bit and 32-bit halfword aligned thumb and thumb-2 instruction set[13]. Applications developed on ARMv6-M architecture can be executed without any modification on ARMv7-M architecture. Figure 3.1 shows the ARM Cortex-M1 processor block diagram without debug module. Main blocks are the core, nested vectored interrupt controller(NVIC), memory interface and AHB Private Peripheral Bus(AHB-PPB) interface is used here to access the NVIC. These blocks of the processor are explained in detail in following sections.

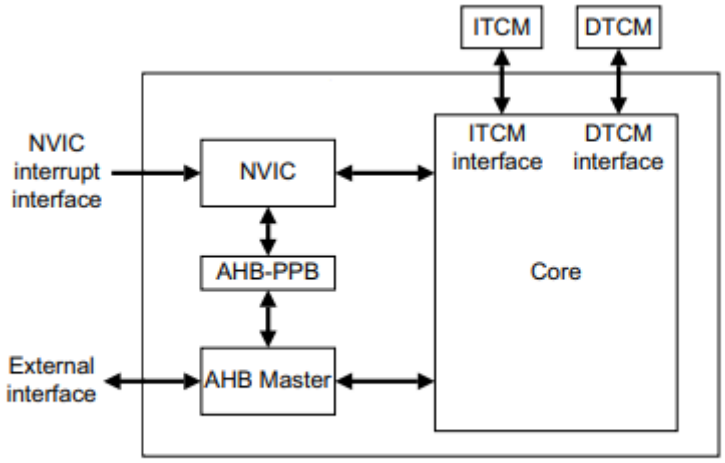

Figure 3.1: Cortex M1 Processor block diagram [13]

3.1 Programmer's Model

ARM supports thread mode and handler mode in ARMv6-M architecture. Thread mode can be privileged or unprivileged. Handler mode is privileged mode. ARMv6-M only supports privileged operation unless it has an extension for supporting un-

privileged/privileged operation. Thread mode is the basic mode in ARMv6-M for application execution[14].

3.1.1 Processor Data types

The data types supported by ARM processors are :

- Byte is 8 bits.
- Halfword is 16 bits.
- Word is 32 bits.
- Doubleword is 64 bits.

3.1.2 Core Register Set

There are 13 general purpose registers which are 32-bit in size. They are R0-R12 registers. R13-R15 are special function registers. R13 is used as Stack Pointer(SP), R14 is Link Register(LR), R15 is Program Counter (PC). Program Status Regsiter, xPSR comprises of three subregisters- Application Program Status(APSR) register reports the application-level programs status. It contains flags that are updated by different instructions. Interrupt Program Status Register(IPSR) holds the exception number of the exception executed by the processor. Execution Program Status Register, EPSR holds the thumb state bit.

3.2 Nested Vectored Interrupt Controller (NVIC)

The NVIC works in close association with the processor in order to have low latency processing time for interrupts. Figure 3.2 shows the NVIC register map. All the registers are 32 bit wide and each bit corresponds to one of the 32 interrupts.

- Interrupt Set-Enable Register - As the name of the register specifies, setting a bit enables the corresponding interrupt. For clearing the enable state, Interrupt Clear-Enable Register should be used.
- Interrupt Clear-Enable Register - Setting a bit disables the corresponding interrupt. Already active interrupts will not have any effect. It only applies for new interrupts.
- Interrupt Set-Pending Register - Setting a bit pends the corresponding interrupt. Clearing the bit by writing 0 to the pending interrupt has no effect.
- Interrupt Clear-Pending Register - Setting a bit clears the pending state of the corresponding interrupt. If the corresponding interrupt was not in pending state then writing a 1 does not have any effect.
- Interrupt Priority Registers - There are 8 interrupt priority registers. These registers allocate the priority for the interrupts. Bits [7:6] in each byte of 4 bytes of this register holds the priority.

Name of register	Type	Address	Reset value
Interrupt Set Enable Register	R/W	0XE000E100	0x00000000
Interrupt Clear Enable Register	R/W	0XE000E180	0x00000000
Interrupt Set Pending Register	R/W	0XE000E200	0x00000000
Interrupt Clear Pending Register	R/W	0XE000E280	0x00000000
Priority 0 Register	R/W	0XE000E400	0x00000000
Priority 1 Register	R/W	0XE000E404	0x00000000
Priority 2 Register	R/W	0XE000E408	0x00000000
Priority 3 Register	R/W	0XE000E40C	0x00000000
Priority 4 Register	R/W	0XE000E410	0x00000000
Priority 5 Register	R/W	0XE000E414	0x00000000
Priority 6 Register	R/W	0XE000E418	0x00000000
Priority 7 Register	R/W	0XE000E41C	0x00000000

Figure 3.2: NVIC Register Map [13]

3.3 Exceptions

As processor and NVIC work closely together, they prioritize and manage all exceptions for efficient processing of interrupts. The exceptions are handled in Handler mode as the processor switches to this mode on occurrence of an exception. Exceptions can occur at any time resulting from an error condition or from other events or peripheral devices. As shown in Table 3.1, NVIC supports configurable priority levels from 0 to 3 for SVCall, PendSV, SysTick and external interrupts[13]. For external interrupts, priority levels can be assigned to IP_N field in the Interrupt Priority Register.

Table 3.1: List of Exceptions

Exception number	Exception	Priority
1	Reset	-3
2	Non-maskable Interrupt	-2
3	Hard Fault	-1
11	SVC	Configurable
14	PendSV	Configurable
15	Systick	Configurable
16-47	External Interrupts 0-31	Configurable

3.4 Memory

Tightly coupled memories are present on the processor and are fast memories. Cortex-M1 processor supports configurable tightly coupled memories (TCMs) for code and data separately known as instruction tightly coupled memory (ITCM) and data tightly coupled memory (DTCM) respectively. ITCM and DTCM both have a configurable size from 0KBytes - 64 KBytes each. There are dedicated interfaces for accessing these memories by the core.

3.5 CMSIS Package

Cortex Micro-controller Software Interface Standard (CMSIS) is hardware abstraction layer developed by ARM for Cortex devices. This helps in porting software easily and re-use the software components. CMSIS provides interfaces to set and clear interrupts in NVIC interrupt registers in-order to configure the interrupts. All the CMSIS C function definitions for NVIC Interrupt registers are provided in header file, core_cm1.h. This package also provides system tick timer configuration and register access functions.

4
Hardware

The project is implemented using different hardware like Zedboard FPGA[15] with integrated Cortex-M1 processor, battery monitoring IC integrated on a evaluation board consisting of the transceiver and current measurement evaluation board. Segger J-Link is used to download the executable software onto the Cortex-M1, adaptor board is used for USB to UART module. Battery monitoring IC will be referred as BMS IC here onwards.

4.1 Hardware implementation on FPGA

In the customized hardware implementation on the FPGA, Xilinx Cortex-M1 IP is used. This IP along with Cortex-M1 processor has an integrated AHB Master bus to AXI bridge[16] which allows AHB Master on Cortex-M1 to communicate with AXI slaves or interconnects. Other modules are AXI interconnect, AXI GPIO, AXI UART, AXI HSUART0, HSUART1, AXI SPI0, SPI1. The hardware design implementation was done by the company, specific to the company requirements.

4.1.1 Memory Map

Figure 4.1 shows the processor memory map. ITCM is configured as 32 KBytes and DTCM is configured as 32 KBytes. Entire software image is downloaded into ITCM for better performance. Code loaded into the ITCM RAM executes from the address 0x00000000 from boot-up. The vector table is also initialized and resides in the ITCM. Peripheral address space is mapped as per the customized hardware. HSUART is the high speed UART. There are two peripherals HSUART 0 and HSUART 1. Both can be used for communication with the BMS IC. Currently only HSUART 0 is being used. UART module is used for printing the output on the serial console. GPIO module is added for debugging purpose. SPI 0 is for current measurement. In private peripheral bus address space, debug access port(DAP), breakPoint unit(BPU), data watchpoint(DWT) are used for debugging.

4.1.2 Interrupt Map

Table 4.1 shows the interrupts mapped in the system. The system tick timer interrupt occurs every 1ms. UART interrupt is generated whenever we receive characters on the serial terminal. High speed UART0 receive interrupt is generated when BMS IC responds to the read or write request from the microcontroller.

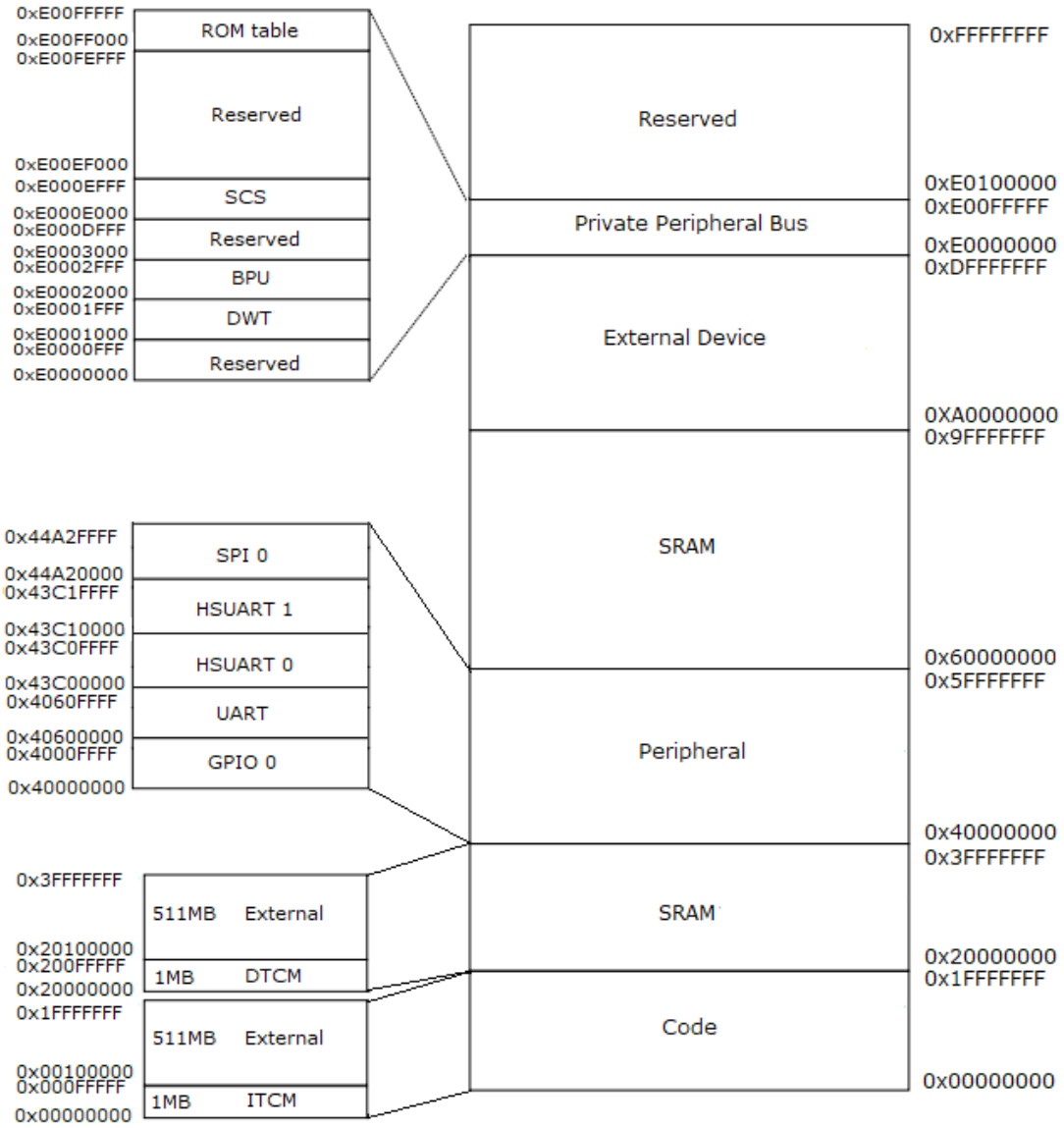

Figure 4.1: Processor memory map [13]

Table 4.1: Interrupts mapped in the system

Number	Interrupt Name	Description
-1	SysTick_IRQn	System Tick timer interrupt
0	UART_SerTerm_IRQn	UART interrupt
1	HSUART0_Rx_IRQn	High Speed UART0 Receive interrupt

4.2 Transceiver IC

The transceiver IC is a general purpose IC used in BMS which enables communication between the microcontroller and BMS IC. It has both transmitter and receiver in a single package and it offers 2 Mbps data rate for fast response. The IC has

two UART ports for serial communication to host microcontroller and two company proprietary communication protocol ports for daisy chain[17] communication inside battery pack.

4.3 BMS IC

BMS IC is a single cell supervisor that communicates with FPGA board and other BMS ICs using company proprietary communication protocol to communicate in a chain. The communication speed is upto 2 Mbps and is a half-duplex communication which means one interface (either low side or high side) can send or receive at a time. The cell measurement range is 0V - 5V. The IC needs a wake-up signal and it wakes up when it receives 4-8 pulses with frequency 50KHz - 650KHz either on its two input pins. As soon as it has woken up, it requires numbering for addressing. Temperature measurement starts automatically and its measured value is stored in its internal register. To start the voltage measurement certain configurations must be updated in the configuration registers.

4.4 Current Measurement Circuit

The evaluation board from Analog Devices[18] used for the current measurement consists of 16-bit ADC AD7983 from Analog Devices[19], 5V reference used to supply the ADC, op-amps and power supply. Op-amps are set with unity gain buffers. The op-amp positive rail is connected to +7.5V power supply and the negative rail is connected to -2.5V.

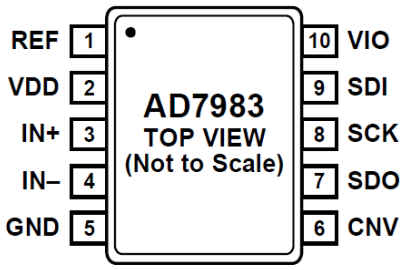

Figure 4.2: ADC AD7983 pin configuration [19]

The ADC has a 10-pin configuration containing SPI-compatible serial interface port as shown in Figure 4.2. The CNV pin which is the Convert Input pin has multiple functions with two modes which are Chain mode and $\overline{CS}$ mode. We use the $\overline{CS}$ mode for our project. In $\overline{CS}$ mode, when CNV is high, it triggers the ADC for conversion and when CNV is low, it enables the output pin SDO which contains the ADC data.

5

Implementation

The block diagram of the control unit in Figure 5.1 consists of FPGA implementation module, BMS ICs connected in a chain, cells connected in series in a battery pack, shunt resistance at the end of the chain of cells, transceiver IC and ADC. The microcontroller includes an integrated ARM Cortex-M1 which is 32-bit processor and two high speed UART modules is implemented on the FPGA. One high speed UART module is enough to implement the BMS topology explained in Section 5.1. One BMS IC is connected across one cell. First BMS IC in the chain is connected on one side to the master using the transceiver IC and on the other side to the next BMS IC. Microcontroller is the master and BMS ICs are the slaves. BMS IC and the microcontroller communicate using the company proprietary protocol through galvanic isolated high speed UART interfaces. Differential op-amp is connected across the shunt resistance and it provides the ADC with the voltage drop measured across the shunt resistance. Microcontroller communicates with the ADC using the SPI protocol.

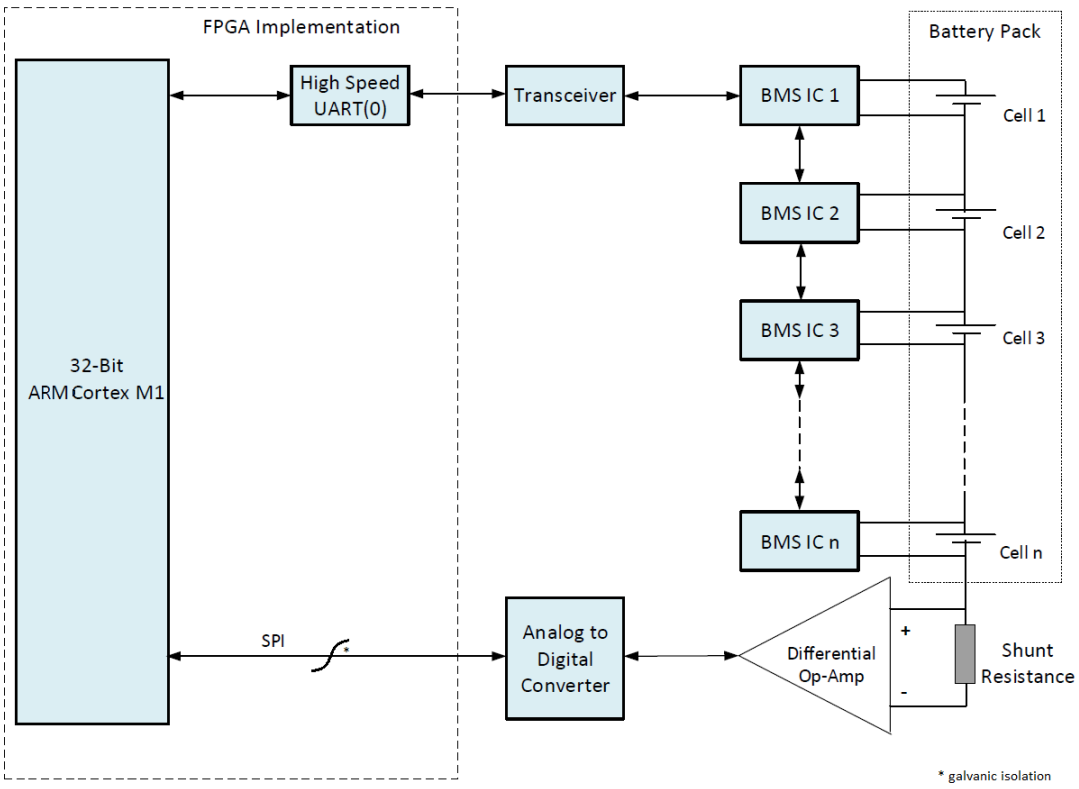

Figure 5.1: Block diagram of Control Unit

5.1 BMS Topology

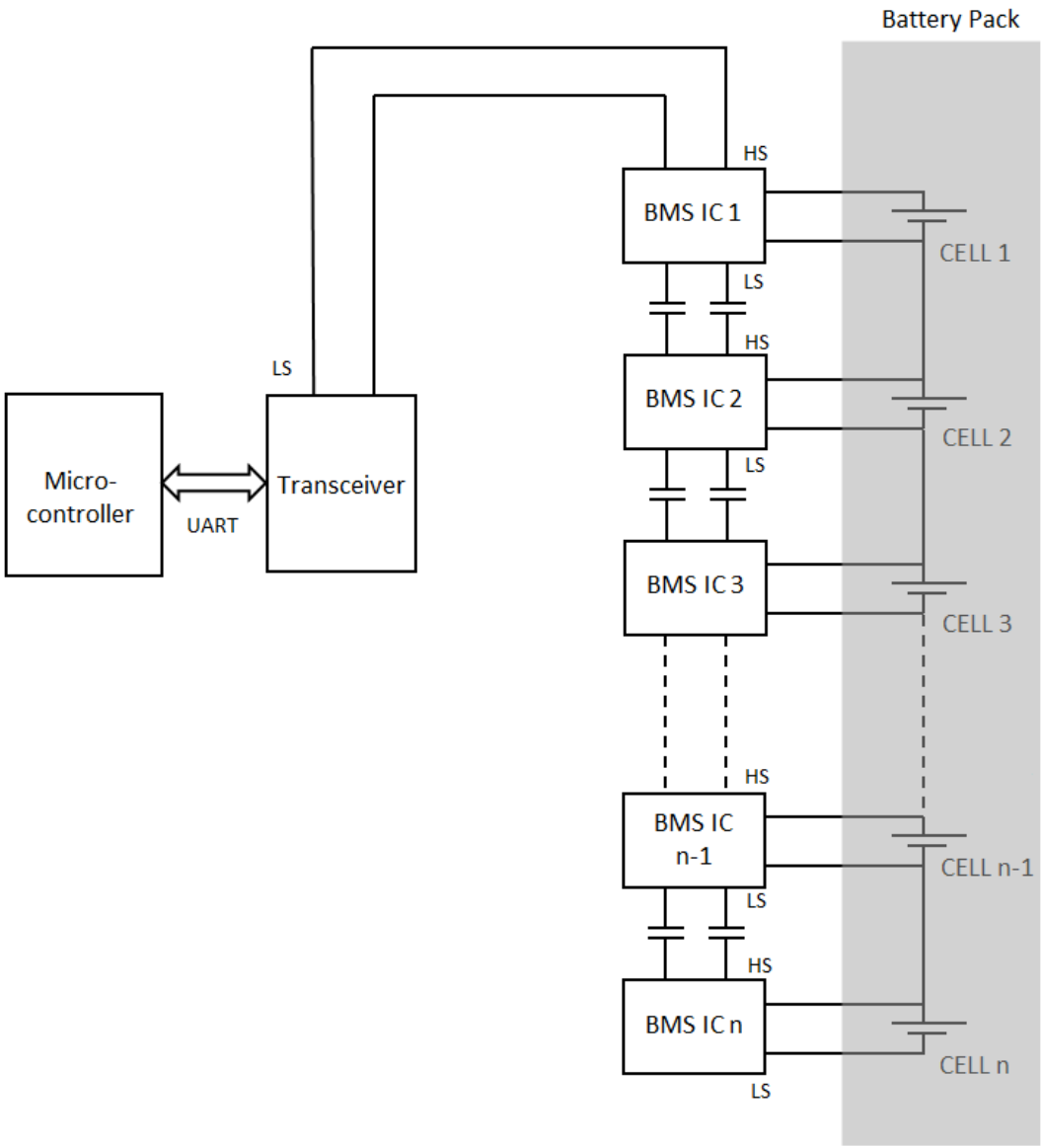

Figure 5.2: Master On Top(MOT) Topology

Master On Top or Master On Bottom topologies are standard daisy chain configurations. In these topologies, the microcontroller which is the master is either on the top or bottom of the BMS IC chain. As shown in Figure 5.2 in MOT topology, the requests from the master will be forwarded from low side(LS) interface and the first BMS IC will receive it on the high side(HS) in the chain. As shown in Figure 5.3 in MOB topology, the requests from the master will be forwarded from the high side(HS) interface and the last BMS IC will receive it on the low side(LS) interface in the chain. The MOT or MOB topology can be selected depending on which interface is used for receiving the wake-up signal. The IC has the capability to switch

between Tx and Rx for message propagation. The MOB or MOT topology is mainly with respect to the hardware connection of the BMS IC to the microcontroller and the interfaces through which the communication takes places. There is no change in the software implementation with respect to the BMS topology.

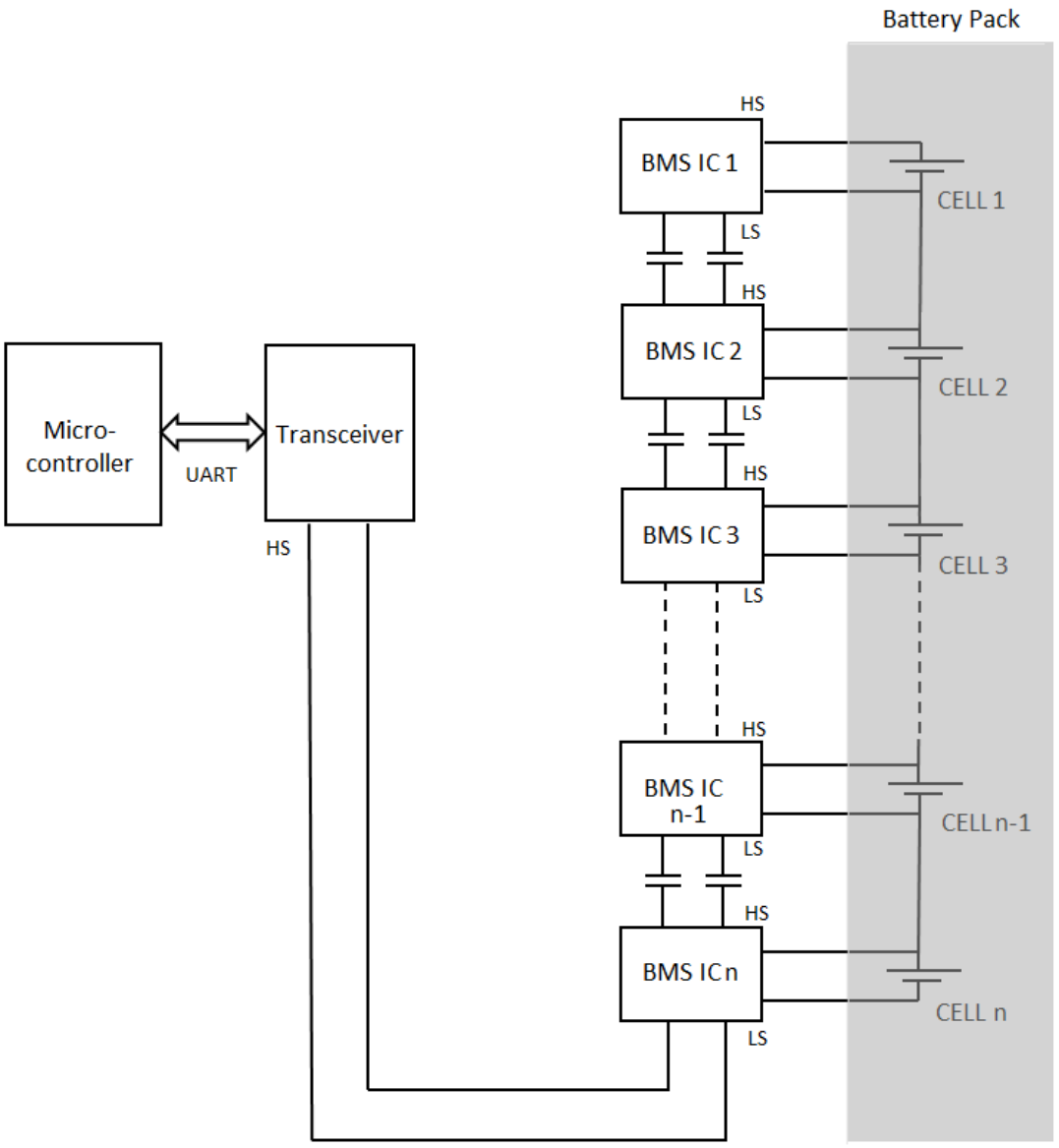

Figure 5.3: Master On Bottom(MOB) Topology

5.2 Communication Protocol

Different communication protocols have been used in the project namely the company proprietary protocol for communication of microcontroller to BMS IC and other BMS ICs in a chain, UART interface for sending messages to serial terminal and serial peripheral interface (SPI) for communication between ADC on evaluation

board for current measurement and microcontroller.

5.2.1 Communication to BMS IC

Communication from microcontroller to BMS IC and other BMS ICs in a chain is done using company proprietary communication protocol. In this protocol microcontroller is the master and BMS ICs are designed as slaves. As the microcontroller is the master, high flexibility is achieved on host side. The protocol is a half-duplex interface and uses read and write commands for communication. The commands are sent over a 2 MBaud UART interface. For each read and write commands, the addressed BMS IC replies with the requested data and it is transmitted over the chain to the microcontroller in the direction of the topology used.

5.2.2 Serial Peripheral Interface (SPI)

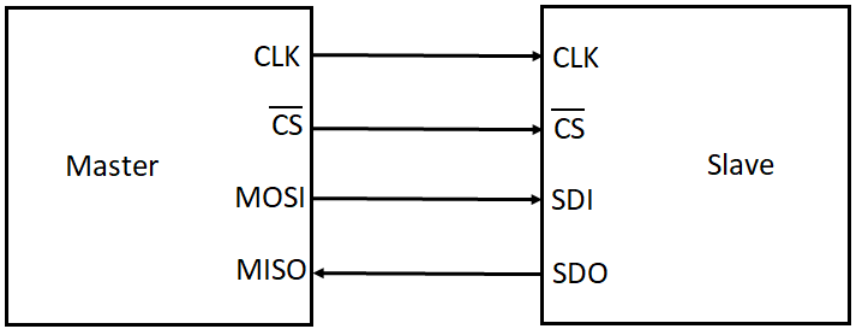

Figure 5.4: SPI Master Slave Communication Signals

SPI is synchronous serial communication interface which is also full duplex. It is a master slave communication between a single master and single or multiple slaves. Figure 5.4 shows SPI master slave connection. CLK is the clock generated by the master. $\overline{CS}$ is the chip select signal. It is active low signal and has to be pulled high to release a slave. MOSI is master out slave in signal line on which data is transmitted from master to slave. MISO is master in slave out signal line on which data is transmitted from slave to master. In this thesis implementation, MOSI is never used.

5.3 Software Integrated Development Environment (IDE)

Firmware is implemented using C language with IDEs such as Keil[20] and MATLAB. The complete development software is done in Keil and MATLAB is used to post process the raw measured voltage, temperature and current values and plot their actual values over time.

5.4 Software Implementation

Initial step is to initialize the peripherals and clear all pending interrupts. SysTick timer is configured to generate interrupt every 1ms and used for doing current, temperature and voltage measurement every 10ms. SysTick_Config function is used from CMSIS package to configure the SysTick interrupt. NVIC interrupts are enabled using NVIC_EnableIRQ function from CMSIS package. Microcontroller performs CRC calculation as part of the communication protocol to read or write to BMS IC. CRC-8-SAE J1850 standard is implemented to compute the CRC[21].

Figure 5.5 shows the software flow diagram explaining the initialization steps and program execution and switch to state machine.

- **WakeUp** - BMS IC requires 4-8 wake up pulses to come out of the sleep mode. The wake up sequence is sent to the first BMS IC using the communication protocol on the high speed UART0 interface. The wakeup pulses are sent to the lowside or highside interface based on the MOT/MOB topology. After the first BMS IC is woken up, it regenerates the wake up pulse for the next BMS IC. The regenerated signal is high frequency signal. This way all the BMS ICs are made to wake up. There is a small wait added here for all the BMS ICs to wake up. Also the baud rate of high speed UART is now switched to 500KHz for further communication. There is also a counter which is updated every time we are in the Numbering ICs state to number ICs within the limit defined. This is added because sometimes the ICs might not wake up or there might be failure during Numbering ICs.

- **Numbering ICs** -After the wake up of the ICs they have to be numbered before they are addressed. The numbering information is sent to each of the BMS IC. It is started with numbering the first BMS IC as 1 and wait for the response. If there is proper response without erros, we continue until we have numbered the last one in each block which is 62. There can be 2-3 blocks of 62 ICs in each block depending on the configuration done in the system. If there was a fault in numbering IC then we increase the counter and switch to WakeUp state.

- **Configure** - After numbering the BMS ICs, they know their position in each block and can be addressed using the communication protocol. The register configurations are done for the voltage measurement.

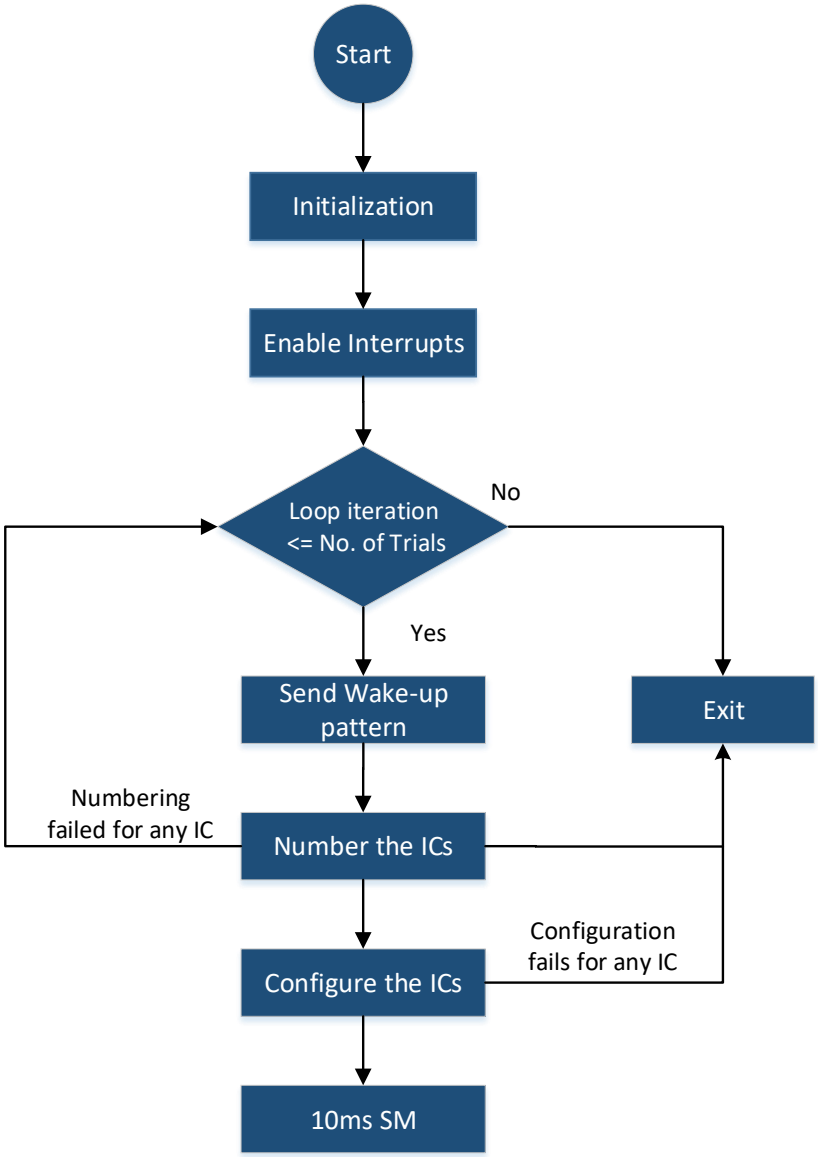

Figure 5.5: Software Flow Diagram

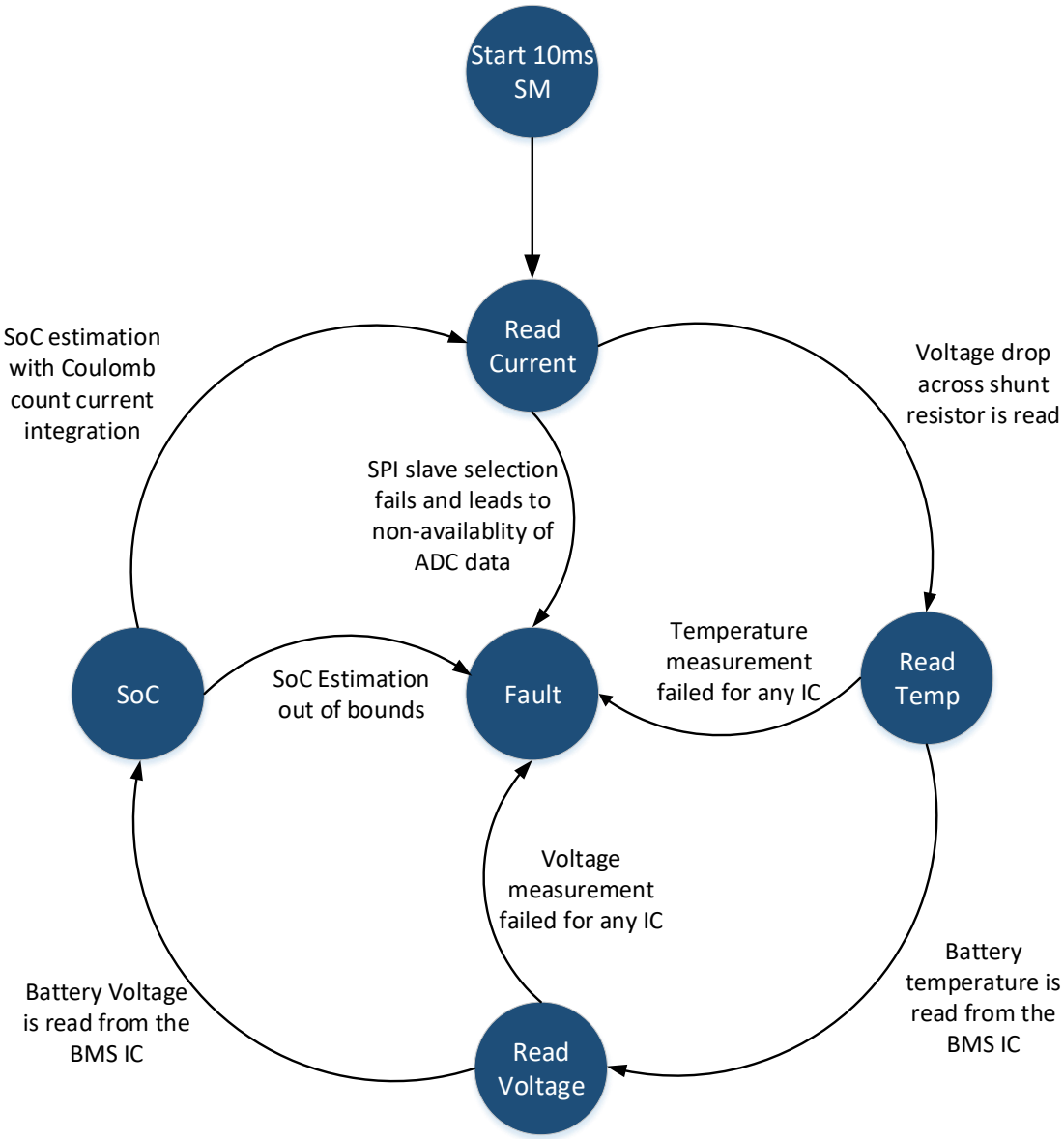

Figure 5.6: State Machine

Figure 5.6 shows the state machine implementation stating the transition between the states during the program execution. Each state is explained below:

- **Read Current** - After doing the required configurations for BMS ICs, we enter into the state machine. The first state in the state machine is for current measurement. In this state, we begin the SPI communication by selecting the slave which is ADC in our case. Start the transfer by making required settings. Polled until the receive FIFO is filled. Now the ADC data is available and stored in the buffer and also sent to the serial terminal to print into the

file. SPI slave is deselected to start the ADC conversion. If there is a failure during the SPI communication then the execution goes into the Fault state with the fault flag set for the error from current measurement so when the fault message is printed on the serial terminal, the user gets to know that the fault is from the Read Current state. This state is reentered every 10ms to get a new current measurement.

- **Read Temp** - This state is entered from Read Current state and does the temperature measurement for all ICs in this state. If a fault occurs while requesting the BMS IC for temperature data or while receiving the temperature data, relevant fault flag is set. As explained in the previous state, this flag will help the user to know the state where the fault occured.
- **Read Voltage** - We go to this state after doing the temperature measurement. The microcontroller has to start the voltage measurement by writing 1 to the start measurement bit in the measurement control register of BMS IC. This is done by sending a command to the BMS IC each time we need the voltage measurement because voltage measurement bit is cleared after the voltage measurement by the measuring unit in the IC. Relevant fault flag is set when fault occurs.
- **SoC** - This state is entered from state Read Voltage. Coulomb count current integration algorithm used to calculate SoC using the equation 2.1 is implemented here. SoC is calculated and checked if it is within the safe operating region and the voltage is within the limits of the particular cell type. If SoC goes outside safe operating region or the voltage is beyond the limits then fault flag is set. During current measurement, the voltage drop is measured across the shunt resistor. This voltage is in the raw form and has to be converted to real values for implementing the SoC algorithm. This is done using the equation:

$$Voltage\_Drop = \frac{5 * (MeasuredValue)}{(2^{16} - 1)} \tag{5.1}$$

  In equation 5.1, MeasuredValue is the voltage drop across shunt resistor, 2^{16} is the ADC data length and 5 is the reference voltage. By Ohm's law we have the equation:

$$Current = \frac{Voltage\_Drop}{Shunt\_Resistance} \tag{5.2}$$

  In equation 5.2, Current is the current, Voltage_Drop is the voltage calculated using equation 5.1 and Shunt_Resistance is the resistance of the shunt resistor. We use this equation to find the current.
- **Fault** - We enter into this state from many other states namely Read Current, Read Temp, Read Voltage and SoC whenever we get a fault. In this state, the particular fault is printed out on the serial terminal and the program execution ends.

6
Results and Discussions

In this chapter the tests performed and their results and discussions are presented.

6.1 Test Setup

The two boards, FPGA board with ARM cortex M1 processor and BMS IC evaluation board are connected using high speed UART interface. Standard jumper wires are used to do the connections. The cell whose parameters are to be measured has 2.6 V without the load connected across it. A DC Motor is connected as load across the cell whose parameters are to be measured. Shunt based current measurement board with ADC is connected to the FPGA board through SPI interface and the SPI signals are also measured on the oscilloscope. Tera-term console is used to get the output printed into a file.

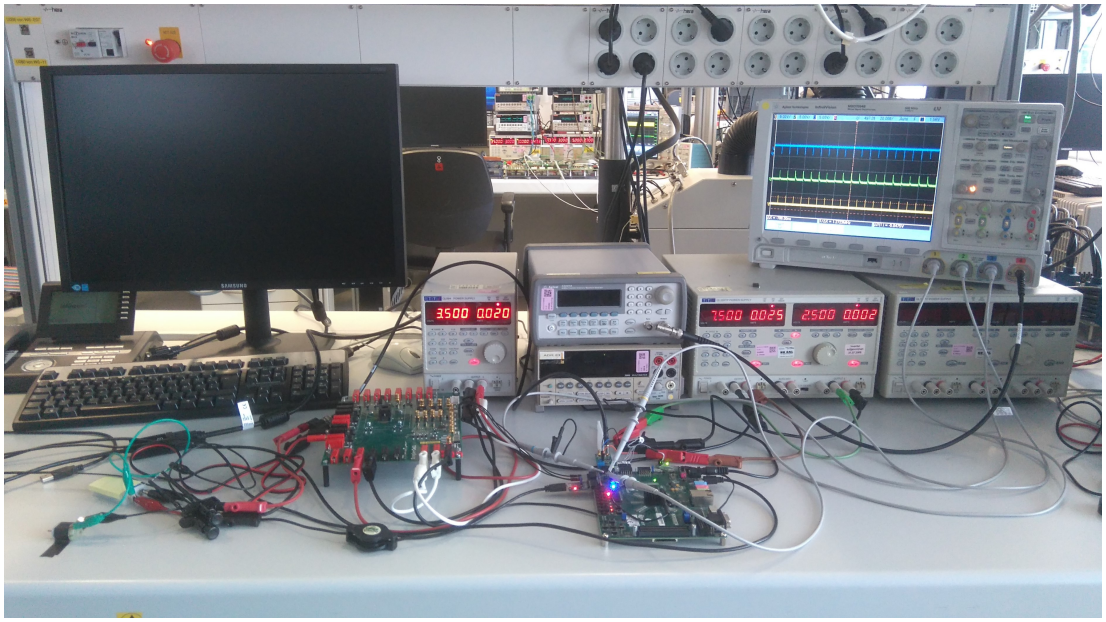

Figure 6.1: Test Setup

6.2 Test Procedure

First step is to establish communication between the microcontroller and the BMS IC evaluation board by sending the wake-up signal. In figure 6.2 the BMC IC is

made to wake up from the microcontroller, BMS IC wakes up and then it regenerates the wake up signal for the next BMS IC.

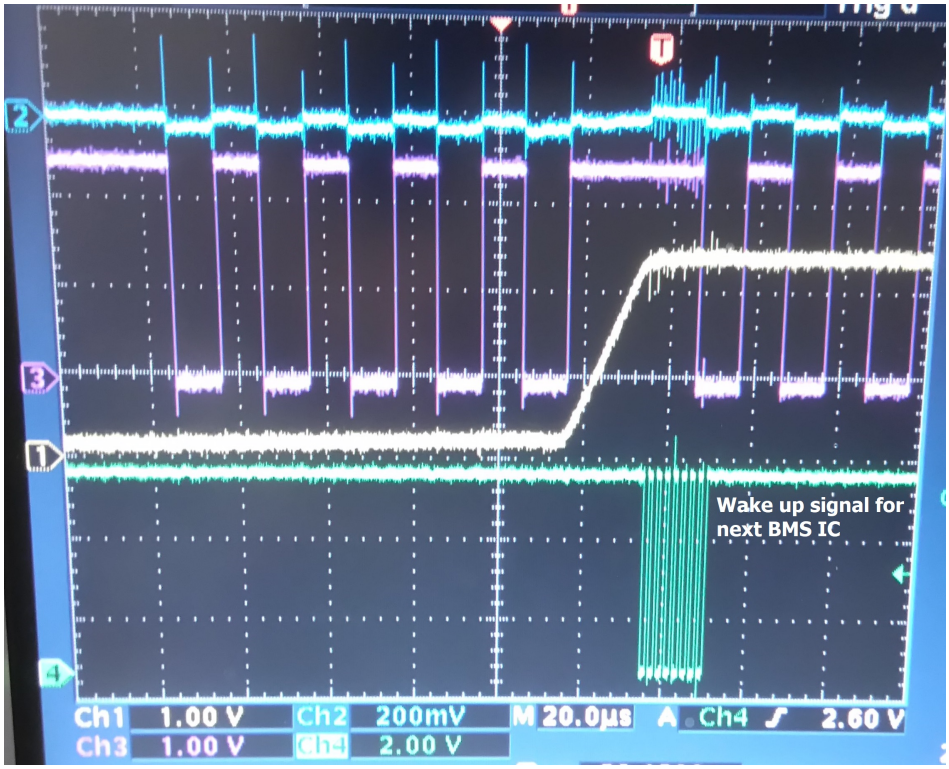

Figure 6.2: Wakeup signal generation

In figure 6.2 Channel 3 is the wake-up signal at a lower frequency of 50 KHz generated by the microcontroller to wake up the first BMS IC. Channel 4 is the wake up signal for the next BMS IC which is at the higher frequency of 500 KHz.

Next step is to number the BMS ICs so that it becomes easier for communication. In this test setup as there is only one BMS IC, it is numbered as 1. For voltage measurement, required configurations are updated into voltage configuration registers of BMS IC. No other configuration is required for temperature measurement.

The voltage of the cell was measured to know the initial voltage across the load and it was around 2.45V .Test measurement started with running the DC motor continuously and hindering its speed in some intervals. Measured raw values of voltage and temperature of the cell are captured into the file. The captured voltage raw values are post-processed in a MATLAB script using the formula shown below:

$$Voltage = \frac{5 * (RawValue)}{(2^{14} - 1)} \tag{6.1}$$

In equation 6.1, 2^{14} is the ADC data length and 5 is the maximum voltage that can be input to the BMS IC. Figure 6.3 shows the plot for Cell voltage measurement captured over a duration of 55 seconds. Whenever the DC motor speed was hindered,

we can see that the voltage consumed by the motor drops and accordingly it is measured by the BMS IC.

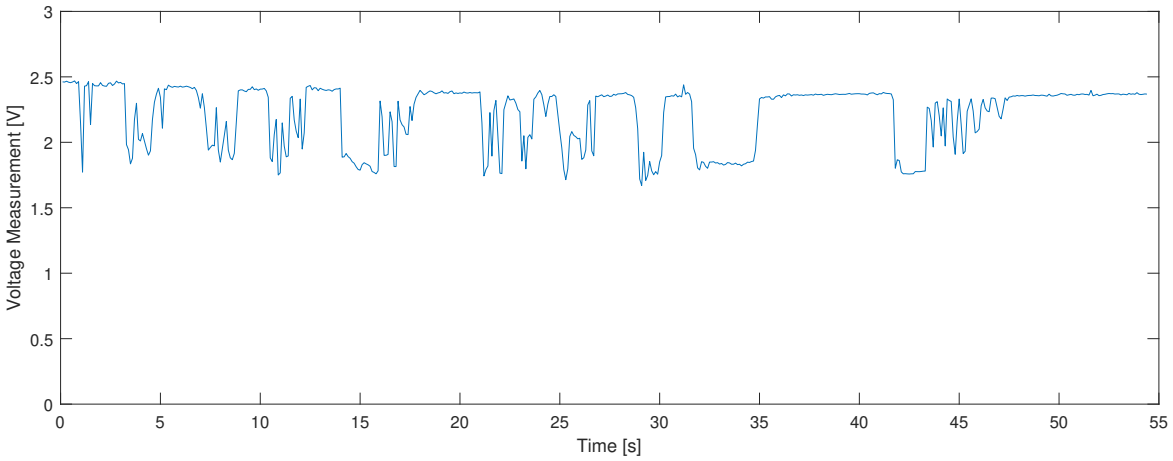

Figure 6.3: Cell voltage measurement

The measured temperature raw value is post processed in a MATLAB script using the formula show below:

$$Temperature = \frac{Measured\_Temperature}{9.12} - 273.15 \tag{6.2}$$

In equation 6.2, constant 9.12 depends on the bit-width of the ADC chosen, subtracting 273.15 is converting from Kelvin to Celsius. In the test, the measured temperature raw value is 2768 and the final post processed result calculated using equation 6.2 is 30.3588 deg C.

For the current measurement, the voltage measured by the ADC across the shunt resistance is sent over the SPI interface to the microcontroller. Also the voltage across the shunt resistance was verified using the multimeter. The SPI logic levels are as shown in Figure 6.4. SDO signal contains the ADC output which is read by the microcontroller. ADC output was further processed using equations 5.1 and 5.2 and the current value was calculated.

6.2.1 Test for SoC Estimation

Another test was conducted for SoC Estimation with discharge cycle of NiMH Cells. In this test, two fully charged NiMH cells were connected in series to act as single cell and DC motor was connected as the load across the cells. For current measurement, shunt resistance used is 4.7 Ohms which was suitable for measuring low currents across the load. Before starting the test, the voltage of the cells were measured to know the initial voltage across the load. After carrying out the initialization steps mentioned in the previous test setup, the battery parameters measurement started and test executed until the cell was discharged to 10% SoC which is the defined cutoff in the firmware for the safety of the cells. The measurement of battery parameters and SoC estimation performed in the firmware was also sent to the serial

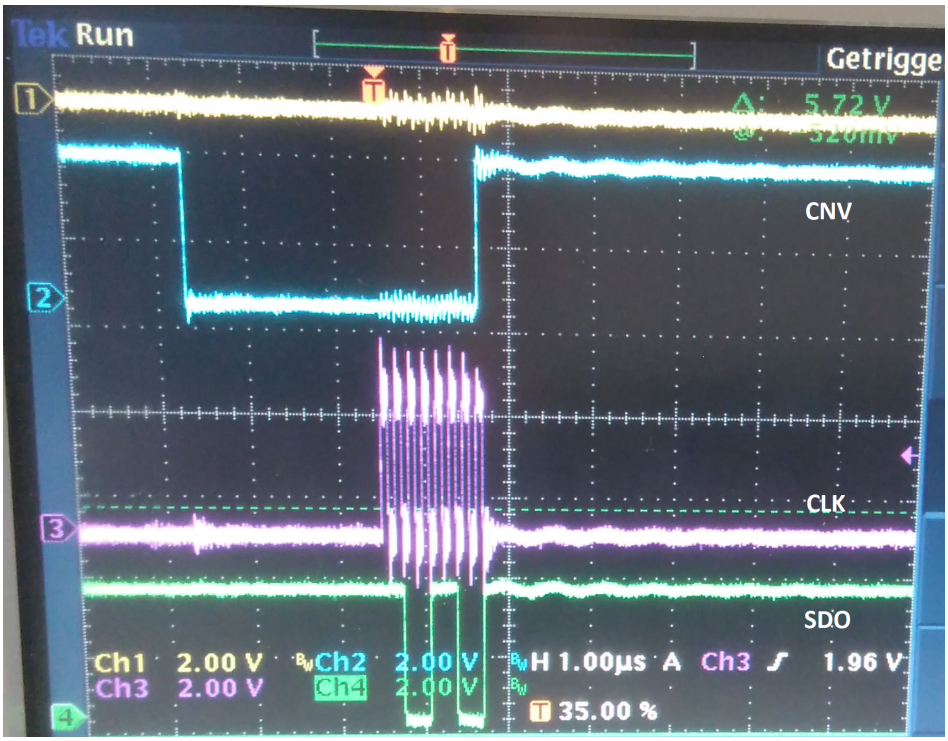

Figure 6.4: SPI logic signals

terminal and logged into a file. The data in the file is then plotted using a MATLAB script to analyse the measured parameters and the SoC estimation.

Figure 6.5 shows the plot for the measured values of voltage, current, temperature and SoC Estimation. From the voltage plot, we can see that the voltage measurement starts at 2.7V when the cells are fully charged and which is same as that measured before the start of the test. As the testing goes on the voltage drops gradually across the load and reached 2.3V at the end of the test. From the current plot, we can see that the current measurement starts with 300mA and varies along the testing duration based on the speed of the DC motor. From the temperature plot, it is seen that temperature measurement starts with 24.5 deg C. When the test runs continuously for several hours there is an increase in temperature of the cells. The temperature plot shows the increasing temperature over the time. From the SoC plot we can see that the SoC starts with 100% charge state and as the cells discharge by running the motor over a period of time, SoC gradually decreases to 10% and exits the program execution with the message to the user for charging the cells as the charging circuit is not present in this system of test.

6.3 Discussions

The ambient temperature of the test room was 25 to 31 deg C during various tests. The measured temperature values from the tests was same as that of ambient temperature for short duration tests. When the test was kept running for long durations, it was observed that the temperature of the cell increased with time which

is as shown in Figure 6.5. This is due to the continuous operation of the cell during the test. Also, different temperature test measurements were taken at varying room temperature and everytime the measured value was as expected. From all these tests, it was concluded that the temperature measured by BMS IC was almost accurate.

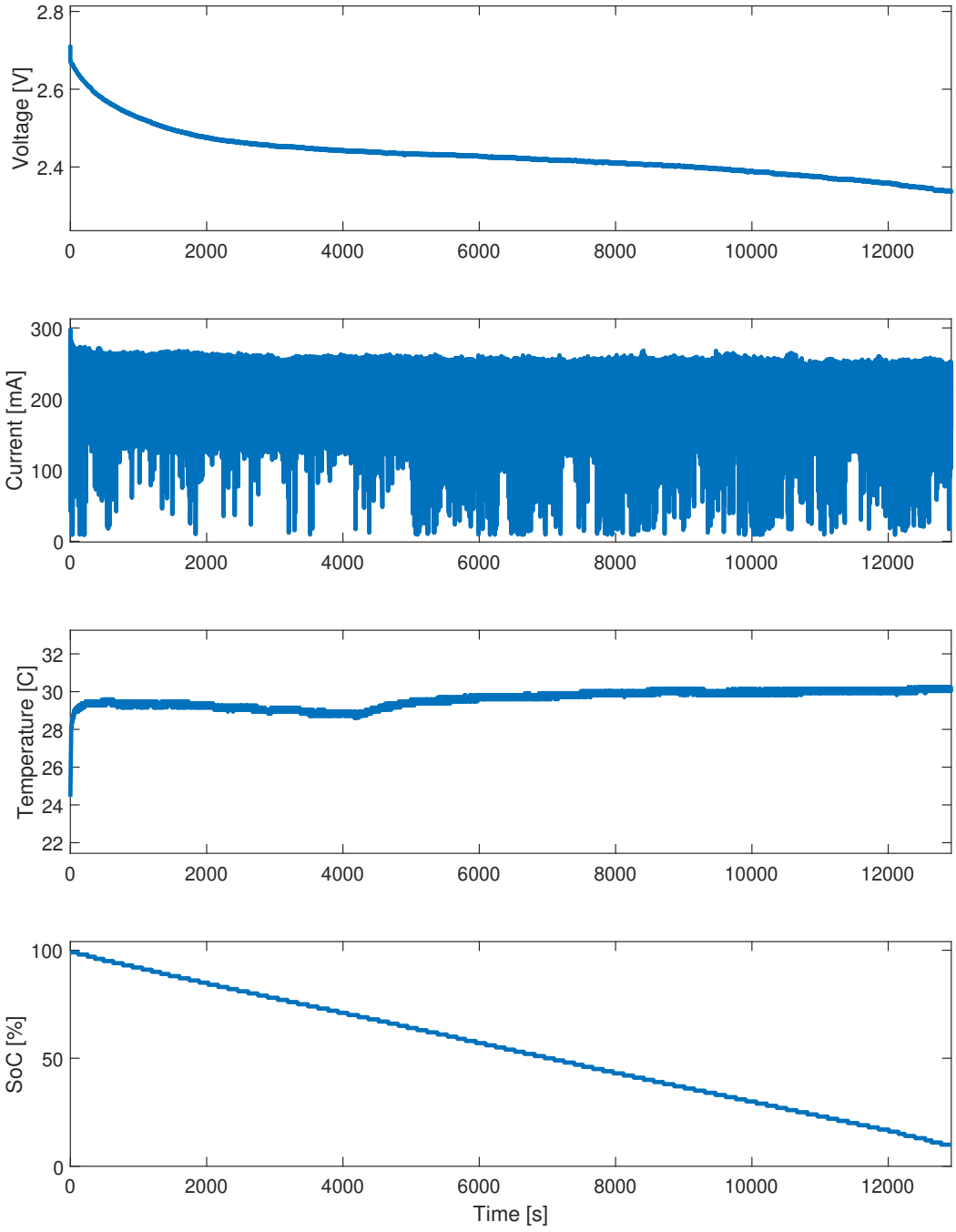

Figure 6.5: Voltage plot, Current plot, Temperature plot and SoC Plot

7
Conclusion

This thesis presents the implementation for the control unit that is capable of establishing communication with the BMS IC and also along with the current measurement system to obtain the measured battery operating parameters and use them for battery state estimation. The control unit design was also implemented in a hardware proof-of-concept prototype to verify the functionality. The test measurement results obtained for voltage and temperature from BMS IC, which is discussed in Chapter 6 shows that results are accurate even during the varying load. The voltage, temperature and current measurements for the complete discharge cycle of the battery matched with the expected results. Coulomb Count method used for the implementation of battery state estimation is done considering only the discharge cycles of the battery. Due to the lack of the charging unit in the hardware setup, it was not possible to accommodate charge cycle of the battery. Since the main microcontroller design was not known the information on out of bounds SoC estimation result is printed on the serial terminal. If in the future there is a requirement to use a different ARM Cortex processor then the firmware can easily be ported to other ARM Cortex processors. This control system is modular and reusable in many other projects.

7.1 Future Work

This thesis shows one of the methods of the control unit implementation for the Battery Management Systems. The implementation shows that ARM Cortex-M1 processor can function as a control unit to perform the necessary tasks required in the BMS and is thus proven in the results. As a next step in further development of thesis involves battery state estimation including charge cycle along with self discharge rates of the battery to be implemented. Various filter techniques could be implemented to remove unwanted noise in the current measurement. When there is out of bounds SoC estimation result, information must be sent to the main microcontroller to take further actions. This feature can be accommodated in the ARM processor itself so that it can be made a standalone unit. Testing with more number of BMS ICs should be carried out and also with high load on the battery. To ensure good battery life, SoH is another topic of interest for the future work on this thesis.

Bibliography

[1] Scrosati, Bruno Garche, Jürgen Tillmetz, Werner. *Advances in Battery Technologies for Electric Vehicles*. Elsevier; 2010.

[2] Jiang J, Zhang C. *Fundamentals and Applications of Lithium-Ion Batteries in Electric Drive Vehicles*. Singapore: John Wiley and Sons, Incorporated; 2015.

[3] Analog Devices. https://www.analog.com/media/en/technical-documentation/technical-articles/A-Closer-Look-at-State-Of-Charge-and-State-Health-Estimation-Techniques-....pdf Accessed on 16.01.2020.

[4] Rahman, Ataur and Rahman, Md and Rashid, Mahbub. *Wireless Battery Management System of Electric Transport*. IOP Conference Series: Materials Science and Engineering. 260. 012029. 10.1088/1757-899X/260/1/012029; 2017

[5] Andrea D. *Battery Management Systems for Large Lithium Ion Battery Packs*. Norwood: Artech House; 2010.

[6] Santhanagopalan, Shriram Smith, Kandler Neubauer, Jeremy Kim, Gi-Heon Keyser, Matthew Pesaran, Ahmad. *Design and Analysis of Large Lithium-Ion Battery Systems*. Artech House; 2015.

[7] Liu, K., Li, K., Peng, Q. et al. *A brief review on key technologies in the battery management system of electric vehicles*. Front. Mech. Eng. 14, 47–64 (2019) doi:10.1007/s11465-018-0516-8

[8] Pistoia G, editor. *Lithium-Ion Batteries : Advances and Applications*. Oxford: Elsevier; 2014.

[9] Circuit Digest. https://circuitdigest.com/article/battery-management-system-bms-for-electric-vehicles. Accessed on 15.01.2020.

[10] Kularatna, Nihal. *Energy Storage Devices for Electronic Systems - Rechargeable Batteries and Supercapacitors*. Elsevier; 2015.

[11] Berg, Helena. *Batteries for Electric Vehicles - Materials and Electrochemistry*. Cambridge University Press; 2015.

[12] ARM Limited. Arm® Cortex®-M1 DesignStart™ FPGA-Xilinx edition User Guide. http://infocenter.arm.com. Accessed on 21.01.2020

[13] ARM Limited. Cortex - M1 Technical Reference Manual. http://infocenter.arm.com. Accessed on 21.01.2020.

[14] ARM Limited. ARM®v6-M Architecture Reference Manual. http://infocenter.arm.com. Accessed on 21.01.2020.

[15] Avnet Engineering Services. http://zedboard.org/product/zedboard. Accessed on 05.10.2019.

[16] ARM Limited. PrimeCell Infrastructure AMBA 2 AHB to AMBA 3 AXI Bridges (BP136). http://infocenter.arm.com. Accessed on 21.01.2020.

[17] J. R. Sanchez, F. Rusek, M. Sarajlic, O. Edfors, and L. Liu, “Fully Decentralized Massive MIMO Detection Based on Recursive Methods,” in 2018 IEEE International Workshop on Signal Processing Systems (SiPS), Oct 2018.
[18] Analog Devices, Evaluation Board User Guide. https://www.analog.com/media/en/technical-documentation/user-guides/UG-340.pdf. Accessed on 16.04.2020.
[19] Analog Devices, AD7983 Data Sheet. https://www.analog.com/media/en/technical-documentation/data-sheets/AD7983.pdf. Accessed on 16.04.2020.
[20] Keil. http://www2.keil.com/mdk5/editions/lite. Accessed on 25.08.2019
[21] Cypress Semiconductor Corporation, “Cyclic Redundancy Check(CRC),” 2013.