Design and Implementation Process for Controls Integration using CAN bus on a Full Function Electric Vehicle Conversion

by

Hugo Provencher

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

Master of Applied Science in **Automotive Engineering**

Faculty of Engineering and Applied Science
University of Ontario Institute of Technology
March 2014

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

The literature reviews used, mainly "Controller Area Networks For Vehicles", were written by the author.

The EcoCAR reports used were written by many authors. As a co-author, I declare that what is used in the thesis was my own work unless otherwise referenced.

Abstract

From the electrical engineering perspective, this thesis addresses the design and implementation of the conversion process from a hybrid electric to a full function electric vehicle (FFEV). The architecture selection process and main components of an electric vehicle (EV) are described, and an exhaustive literature review on the controller area network (CAN) is presented. The electrical and control system integration strategy is explained, along with the model-based algorithm programmed into the vehicle integration module (VIM). Emulating electronic control units (ECUs) from the original powertrain and controlling additional ones for the electrical drivetrain through CAN bus, along with keeping the same functionalities of a typical production vehicle makes this vehicle conversion similar to a factory built model. Finally, the tests and results originating from this conversion to a full electric powertrain are discussed. The vehicle features a 83.5 kWh Li-ion battery built in-house, resulting in an estimated range of 482 km.

Keywords: CAN, EcoCAR, electric vehicle, transportation, energy, lithium battery

Dedication

I dedicate this thesis to Amar El Tarazi and Loïc Moquin-Léger. I remember meeting you and joining my first solar car team the second week of my undergrad at Polytechnique. Over those 2 years of preparation and through the World Solar Challenge 2007 in Australia, you both transferred me your passion for solar cars and electrical engineering, and taught me the fundamental technical basis I'll ever need in my career.

I have been using these skills on a daily basis while working on my second solar car project and on the EcoCAR challenge. Amar, first introduced as controls and high voltage lead of Esteban 4, and Loïc, as team lead, I now consider both of you as good friends of mine.

Acknowledgements

I owe my deepest gratitude to the main core of the UOIT EcoCAR team Joseph Brennan, Gavin Clark, Pierre Hinse, Mike Maduro, Lesley McLelland, Dr. Greg Rohrauer, Hugues Marceau, Helen Qin and Shawn Sandham. It was a great pleasure and an honour getting to know you and working with you Eco-Fellows.

A huge thanks to Dr. Greg Rohrauer, my thesis adviser, for trusting me and believing in my electrical engineering skills, and also for giving me the opportunity of building an electric car and high responsibilities inaccessible to young engineers outside academia.

I am indebted to Gavin Clark who played an important role during my graduate studies as a friend, and also as a dedicated colleague who spent hours proof-reading my EcoCAR reports, literature reviews and this thesis.

It is an honour for me to have worked with and learned from Pierre Hinse which I consider a knowledgeable mentor. He has made available his support in a number of ways in the design and implementation of the UOIT EcoCAR.

I would like to show my gratitude to Amanda Kalhous for being an awesome mentor, first as a GM mentor for the EcoCAR project, and after helping me joining her at GM as a life and work mentor.

Thanks to my family to whom my work might be a mystery, but still supported me despite the distance between us.

I would like to thank Dr. Richard Marceau for his advice on the writing process of this thesis.

Finally, thanks to all the people who worked with me closely or sporadically on the EcoCAR Challenge.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

A

A	Ampere
A/C	Air Conditioning
AC	Alternative Current
ACC	Accessory
AC/AC	AC to AC Converter
AC/DC	AC to DC Converter
ACK	Acknowledgement
ACCM	Air Conditioning Compressor Module
Ah	Amp hour
ALDL	Assembly Line Diagnostic Link
ANL	Argonne National Laboratory
AOS	Automatic Occupant Sensor
APM	Auxiliary Power Module
Avg.	Average

B

b	binary
BCM	Body Control Module
BEV	Battery Electric Vehicle
BMS	Battery Management System
BPCM	Battery Pack Control Module
Byte	1 byte = 8 bits

C

CAN	Controller Area Network
CANH	CAN High
CANL	CAN Low
CARB	California Air Resources Board
CD	Charge depleting
CGM	Communication Gateway Module
CIDI	Compression-ignition direct-injection
CIS	Chassis Inertial Sensor
CRC	Cyclic Redundancy Check

D

d	decimal
DC	Direct Current
DC/AC	DC to AC Converter
DC/DC	DC to DC Converter
DE	Discrete Event
DLC	Data Length Code (CAN bus)
DLC	Data Link Connector (OBD II)
DOD	Depth of Discharge

DOE	Department of Energy
DSML	Domain-Specific Modeling Language
DTC	Diagnostic Trouble Code

E

EBCM	Electronic Brake Control Module
ECC	Electronic Climate Control
ECM	Engine Control Module
EcoCAR	EcoCAR The NeXt Challenge (inter-university competition)
ECU	Electronic Control Unit
EMI	Electromagnetic Interference
EOF	End Of Frame
EPA	Environmental Protection Agency
EPS	Electronic Power Steering
EREV	Extended Range Electric Vehicle
ESD	Electrostatic Discharge (capacitor)
ESS	Energy Storage System
EV	Electric Vehicle
FCV	Fuel Cell Vehicle

F

FE	Fuel Economy
FFEV	Full Function Electric Vehicle
FSCM	Fuel System Control Module
FSM	Finite State Machine
FTP	Federal Test Procedure (drive cycle)

G

GCC	GNU Compiler Collection
GFI	Ground Fault Interrupt
GHG	Greenhouse Gas
GM	General Motors
GMLAN	General Motors Local Area Network
GUI	Graphical User Interface

H

h	hexadecimal
HCP	Hybrid Control Processor
HEV	Hybrid Electric Vehicle
HIL	Hardware-In-the-Loop
HS	High-Speed
HV	High Voltage
HVCM	High Voltage Control Module
HW	Highway (drive cycle)
HWFET	Highway Fuel Economy Test (drive cycle)

I

ICS	Instrument Cluster Screen
ICE	Internal Combustion Engine
ID	Identifier
IDE	Identifier Extension
IGBT	Insulated gate bipolar transistor
IPC	Instrument Panel Cluster
IRC	Integrated Radio Controller
ISO	Independent System Operator
I/O	Input / Output

J

K

km	kilometer
kW	kilowatt
kWh	kilowatt hour

L

LCD	Liquid-Crystal Display
LED	Light Emitting Diode
Li-ion	Lithium-ion battery
Li-Battery	Lithium Battery
LiFePO	Lithium Iron Phosphate
LIN	Local Interconnect Network
LS	Low-Speed
lsb	least significant bit
LSB	Least Significant Byte

M

MCPA	Motor Control Processor A
MCPB	Motor Control Processor B
mi	mile
MIL	Malfunction Indication Light (check engine light)
MIS	Manual Interrupt Switch
MOSFET	Metal–Oxide–Semiconductor Field-Effect Transistor
MOV	Metal Oxide Varistor
mpgge	miles per gallon gasoline energy equivalent
mpg	miles per gallon
MPG	Milford Proving Ground
msb	Most significant bit
MSB	Most Significant Byte
Msg	Message

N

NCM	Nickel-Cobalt-Manganese
NiCd	Nickel-Cadmium
NiMH	Nickel-Metal Hydride
NRZ	Non-Return to Zero
NTC	Negative Temperature Coefficient

O

OBCM	On-Board Charger Module
OBD-II	On-Board Diagnostic II
ODE	Ordinary Differential Equation
OEM	Original Equipment Manufacturer
OnStar	OnStar System made by General Motors
OSI	Open Systems Interconnection

P

PbA	Lead Acid
PDE	Partial Differential Equation
PHEV	Plug-in Hybrid Electric Vehicle
PRND	Park, Reverse, Neutral, Drive
PSAT	Powertrain System Analysis Toolkit
PSD	Power Split Device
PTE	Powertrain Expansion
PTW	Pump-to-Wheel
PWM	Pulse Width Modulation

Q

R

RAM	Random Access Memory
RFA	Remote Function Actuator
RMS	Root Means Square
ROS	Roll Over Sensor
R_s	Slope Control
RTI	Real Time Workshop
RTR	Remote Transmission Request
Rx	Receive
RXD	Receive Digital

S

SAE	Society of Automotive Engineers
SCM	Supervisory Control Module
SDARS	Satellite Digital Audio Receiver System
SDF	Synchronous Data Flow
SDM	Sensing and Diagnostic Module
SI	Spark-ignition

SOC	State of Charge
SOF	Start Of Frame
SOH	State of Health
SRR	Substitute Remote Request
SUV	Sport Utility Vehicle
SW	Single-Wire

T

TCM	Transmission Control Module
Temp.	Temperature
TIM	Traction Inverter Module
TPIM	Traction Power Inverter Module
TTL	Transistor-Transistor Logic
Tx	Transmit
TXD	Transmit Digital

U

UART	Universal Asynchronous Receiver-Transceiver
UDDS	Urban Dynamometer Driving Schedule (drive cycle)
UF	Utility Factor
UOIT	University of Ontario Institute of Technology
U.S.	United-States
US DOE	United States Department of Energy

V

V	Volt
V_{ac}	AC voltage
V_{dc}	DC voltage
V_{DD}	Supply voltage
V_{ref}	Reference voltage
V_{SS}	Ground
VIM	Vehicle Integration Module
VTD	Vehicle Theft Deterrent
VTS	Vehicle Technical Specifications

W

W	Watt
Wh	Watt hour
WTP	Well-to-Pump
WTW	Well-to-Wheel

X

Y

Z

ZEV	Zero Emission Vehicle

Symbols

Ω	Ohm

Chapter 1

Introduction

1.1 EcoCAR: The NeXt Challenge

EcoCAR: The NeXt Challenge, also called EcoCAR Challenge, debuted in the Fall of 2008 and ended 3 academic years later in the Summer of 2011 [1]. The objective of this inter-university competition was to re-engineer the powertrain of a donated vehicle into a more eco-friendly prototype, while maintaining consumer appeal in the areas of performance, utility and safety. Powertrain architectures that reduce fuel consumption; well-to-wheel greenhouse gas emissions and energy consumption; and tail pipe emission are considered eco-friendly.

In 2008, the University of Ontario Institute of Technology (UOIT) was amongst the 16 universities across North America awarded participation in the EcoCAR Challenge. Selection was based on a competitive RFP to which over 50 top engineering institutions responded across North America. During the first year of the challenge (2008-2009), collegiate students had to design and simulate their greener powertrain. The following year (2009-2010), competing universities were given their vehicle, a 2009 GM 2-mode hybrid Saturn VUE. They had the year to implement their design before competing in dynamic events at the second year competion taking place at GM's Proving Ground in Yuma Arizona. The third year of the challenge (2010-2011) was intended for testing, calibrating and optimizing the vehicle's new and greener architecture, before the final competition taking place at the Milford Proving Grounds in June 2011.

The main sponsors and organizers of this competition were the US Department of Energy (DOE), General Motors (GM) and Argone National Laboratory, a division of the DOE. From the sponsor standpoint, the primary objective was to prepare the next generation of engineers for careers in the green vehicle industry [1]. The UOIT EcoCAR prototype is shown on Figure 1.

Figure 1. UOIT EcoCAR Prototype in the Climatic Wind Tunnel at the Automotive Center of Excellence.

1.2 Previous Research

To satisfy the constantly growing demand for green vehicles and comply with new government policies, OEMs spend considerable amounts of money to develop environmentally friendly alternative propulsion technologies. It is projected that hybrid electric vehicles (HEV) and battery electric vehicles (BEV) will share 30% of the passenger vehicle market by 2020 [2]. Below is a list of historic and current production electric vehicles from the late 90s to 2014:

- GM EV1 and S10 EV
- Toyota RAV4 EV
- Ford Ranger EV
- Chrysler TEVan and EPIC Mini-van
- Tesla Roadster and Model S
- Nissan Leaf
- GM Chevrolet Volt and Spark EV
- GM Cadillac ELR
- Ford Focus BEV
- Mitsubishi iMEV
- BMW i3 city
- Fiat 500e
- Honda Fit EV
- Smart Fortwo EV

The method of building electric vehicles can be divided in 3 categories: a ground up design, a factory conversion, and a hobbyist conversion. Having been engineered from the drawing board explicitly as an electric vehicle, the famous EV1 and now Telsa Model S and BMW i3 are from the first group. The Toyota RAV4 EV, the Nissan Leaf and GM Volt are based of existing internal combustion engine (ICE) vehicle platforms, respectively from a RAV4, Nissan Versa and Chevrolet Cruze. These can be classified as factory converted EVs as are most other products on the market. Hobbyist conversions are normally gasoline or diesel powered vehicles, to which an electrical powertrain is added, most often using lead acid batteries and DC motors, though this is now changing. Hobbyist conversions need to remain affordable and may lead to many consumer acceptability concessions. However, hobbyists converting cars are often ready to accept several trades offs and compromises to see their vehicles powered by a greener energy source.

The Tesla's structure is designed around the battery, the weight is minimized and it is practically the only long range capable EV on the market as of today. Having the range of a Tesla, and the look of a production cross-over vehicle, without being designed from scratch but rather along the lines of a factory conversion, is what makes the vehicle described in this thesis so special, and that it existed 1.5 years before Tesla's Model S came to market. It describes the best electric vehicle the UOIT EcoCAR team could come up with, not starting from a ground up design. The vehicle visually and functionally looks like it came out of the factory, as shown in Figure 1 and Figure 41 respectively. It can be driven without any speed, grade or cabin climate restrictions for more than 400 km. In other words, the average consumer would never be challenged for autonomous range requirements in any suburban setting.

1.3 Goal of Thesis

The goal of this thesis is to explain, with an electrical engineering focus, the design and implementation of the process employed to convert an existing HEV into an EV having over 400 km of range with similar look and functionality to a production vehicle, i.e. a full function electric vehicle (FFEV). An important part of such a project is to design and implement the high voltage powertrain by removing and emulating obsolete ECUs and replacing them with relevant ones in the new powertrain. The center piece of these added controllers is the vehicle integration module (VIM) used to communicate with each controller, emulate CAN messages and control the high voltage distribution hardware. Finally, this thesis is one of the very few detailed non-confidential documents on the topic.

1.4 Research Contributions

The UOIT EcoCAR project was a highly complex engineering project that would not have been possible without the author. The author, who joined the EcoCAR after the first year of the competition, acted as the controls and high voltage lead and was solely responsible for all the hardware and software interfaces. A literature review of hybrid powertrains is presented in Chapter 2. Followed by a review of the architecture and component selection process, made by his teammates, and finally a breakdown of the components selected for the project.

Chapter 3 is one of the few non-confidential documents on the controller area network protocol (CAN) fundamental theory applied for vehicles. It summarizes in a single chapter a clear, easy and exhaustive description of the CAN bus communication protocol for those interested in its fundamental theory, along with relevant and useful hands-on knowledge beyond the low level theory acquired by the author and directed to a broad range of CAN users.

The author put significant effort to first understand the surrogate vehicles electrical/software interface, along with the new systems that had to be implemented to

convert it to a full function electric vehicle (FFEV). A methodology was then developed to provide a systematic approach to replace and emulate ECUs. This methodology is outlined in chapter 4 and 5. Although team members helped with physical wiring and building of components, the author generated all software interfacing with hardware. The entire controls development and implementation of the UOIT EcoCAR was accomplished by the author, along with a significant portion of the wiring layout, tracing and debugging.

Chapter 6 summarizes project results that could not have been achieved without the author's significant contribution. Section 6.4 highlights the author's accomplishments in more details, such as the integration of the high voltage and controls of the vehicle integration module (VIM), battery management system (BMS), charger and motor controller. Additionally, the original vehicle CAN bus was thoroughly studied leading to the accurate static and dynamic emulations of removed ECUs which resulted in a diagnostic trouble code (DTC) free vehicle. CAN communication was also used to integrate the full electric powertrain.

Chapter 7 outlines a list of original contributions, such as the bench test created to test the added powertrain electrical subsystems, and the design and implementation of an in-house instrument cluster screen with CAN capabilities. The knowledge gained by the author through years of developing electric vehicle prototypes is captured by a list of hands-on personal best practices.

1.5 Summary of Thesis Sections

Chapter 1 introduces objectives of this research.

Chapter 2 explains the UOIT EcoCAR powertrain architecture selection process and provides an overview on the EV powertrain components chosen.

Chapter 3 covers the fundamentals of the CAN communication protocol, along with hands-on theory and applications useful to many CAN users in the automotive industry.

Chapter 4 describes in detail the electrical and control system integration strategy.

Chapter 5 examines the vehicle integration controller, more specifically the vehicle integration module (VIM), making this strategy possible, as well as its model-based program.

Chapter 6 presents with an electrical focus the tests and results of the UOIT EcoCAR project.

Finally, chapter 7 concludes by stating the author's contributions, highlighting directions for future research and development, and giving hands-on best practices.

Chapter 2

Architecture Selection Process

2.1 Introduction

In the previous chapter, the goals of the research and development efforts described in this document were introduced. This chapter addresses why the UOIT EcoCAR team selected a full electric powertrain while all the other universities taking part in the competition chose an architecture with 2 energy sources, i.e. mostly plug-in hybrids. The different levels of hybridization are first detailed. Then, typical fuels are listed and their well-to-wheel (WTW) influence is analyzed. After having established the powertrains and fuels available, component sizing and vehicle performance simulations were run. To conclude this chapter, the components selected to implement the chosen architecture are described.

2.2 Powertrain Selection and Configurations

2.2.1 Degree of Hybridization

The degree of hybridization of a vehicle can be defined by its powertrain electrification level, in other words the capacity of the energy storage system (ESS) and the electric motor powertrain. In Figure 2, the different degrees of hybridization are categorized in ascending level of electrification.

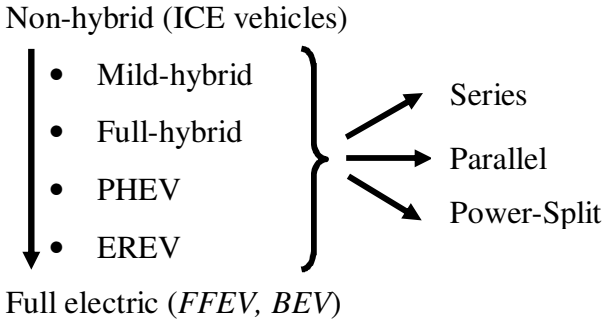


Figure 2. Degree of Hybridization.

In Figure 2, the expression “non-hybrid” refers to internal combustion engine (ICE) vehicles. Vehicles having a single electric propulsion source are deemed full function electric vehicles (FFEV) also called battery electric vehicles (BEV). In between ICEs and EVs are various levels of hybrids. Plug-in hybrid vehicles (PHEV) can charge from the grid (grid dependant), whereas mild/full hybrid electric vehicles (HEV) do not (grid independent). Extended range electric vehicles (EREV) use a full electric propulsion for a limited range and revert to hybrid operation for the extended range. Hydrogen fuel cell hybrids also fall into this category. It is to be noted that PHEVs can be considered as a specific category of HEVs having a stronger electric drive and the ability to charge from the grid. Similarly, EREVs are PHEVs with a motor powerful enough to sustain the electric mode regardless of the vehicle speed as long as the battery holds charge [3].

2.2.2 Technologies

To increase fuel efficiency, hybrids commonly use the following 3 advanced technologies [2]:

- **Automatic engine idle/stop**

  The engine is stopped at low speeds and started back up during faster acceleration. This feature is possible due to a powerful electric motor driven by the high voltage battery.

- **Regenerative braking**

  The vehicle kinetic energy is recaptured and transformed to electrical energy by the electric machine and power inverter, and stored in the battery during deceleration.

- **Efficient ICE operation (electric motor drive/assist)**

  In parallel or power-split HEVs, the ICE can be decoupled from the axle while the electric motor still powers the vehicle in an attempt to keep the ICE operating only in efficient regions.

2.2.3 Powertrain Configurations

As shown on Figure 2, the 3 powertrain configurations, also known as drivelines, may be: series, parallel and power-split (parallel/series).

2.2.3.1 Series

Series hybrids have their axle mechanically coupled to an electric motor providing a full electric propulsion. In other words, only the electric motor is connected to the drivetrain. They also have an ICE coupled to a generator to charge the battery which acts as an energy buffer between the ICE and the electric motor. Figure 3 illustrates this linear driveline configuration from which the name derives.

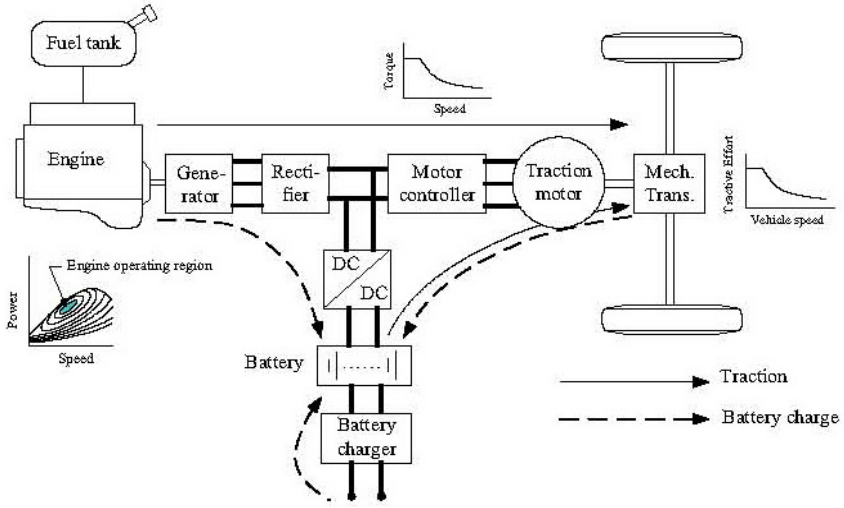


Figure 3. Series Configuration [5].

2.2.3.2 Parallel

In parallel hybrids, the electric motor and the ICE mechanical outputs are both coupled to the drivetrain, hence parallel configuration. In this case the engine is not directly coupled to the battery, only the electric motor is. This principle is shown in Figure 4.

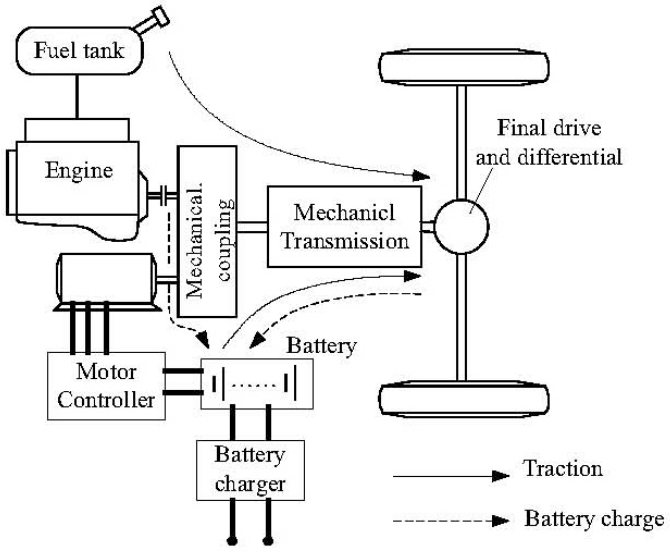


Figure 4. Parallel Configuration [5].

2.2.3.3 Power-Split

Also known as series-parallel hybrids, power-split HEVs feature the combination of series and parallel HEV characteristics. In this configuration, a power splitting device (PSD) couples the ICE, generator and electric motor. Therefore, the ICE can charge the battery through the generator, as in series hybrids, and also mechanically power the drivetrain, as in parallel hybrids. This concept is displayed in Figure 5.

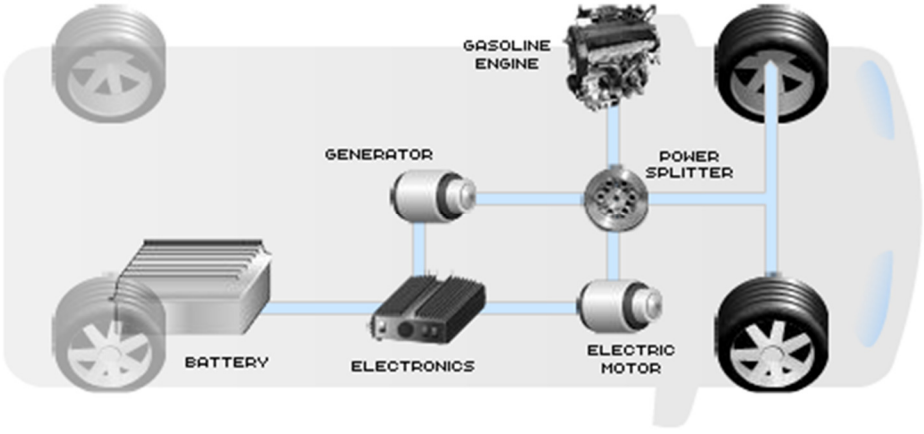


Figure 5. Power-Split Configuration [6].

2.3 Fuel Selection

In vehicles, the onboard energy is usually referred to as fuel. This advanced vehicle engineering competition allowed participating teams the following choice of fuels:

- **E10**: a reformulated gasoline with 10% ethanol
- **E85**: a reformulated gasoline with 85% ethanol
- **BD20**: a reformulated diesel with 20% biodiesel
- **H_2**: hydrogen required for fuel cell to operate
- **Electricity**: electricity in a rechargeable battery

The onboard energy consumed by a vehicle is not the only energy having an environmental impact. This energy is referred to as pump-to-wheel (PTW), whereas the energy required by the generation or transformation process to render it usable by vehicles is called well-to-pump (WTP). When analyzing the overall energy and greenhouse gas (GHG) production of vehicles, it is important to use the well-to-wheel (WTW) analysis which includes both the well-to-pump and pump-to-wheel energy. For example, electric cars have no pump-to-wheel GHG emissions, but are still accountable to those released by the generating plant producing the electricity used to charge them, i.e. well-to-pump emissions.

2.4 Well-to-Wheel Influence and Architectures Considered

Including universities from Canada and the United-States, the EcoCAR competition specifies a North American energy mix for electricity comprised of the assumed 13% generated in Canada and 87% produced in the U.S [1]. Knowing that almost all vehicles are deployed locally, the UOIT EcoCAR team also thought it interesting to determine which architecture was the most eco-friendly for Ontario. That's why, the EcoCAR energy mix was also compared to one for the North Eastern U.S. and for the province of Ontario. Table 1 summarizes the energy use and the GHG production of 11 different possible architectures for 3 energy mixes for electricity: non-hybrids using either E10 or E85, HEVs and PHEVs using either E10, E85 or BD20, a compression-ignition direct-

injection (CIDI) vehicle using BD20, a FCV and a BEV [7]. In this table, the green to red colour scale means best to worst choice respectively.

Table 1. Well-to-Wheel energy and GHG production [7].

Vehicle \ Energy Mix	Corrected Energy (kWh/km)			Corrected GHG (g/km)		
	EcoCAR Energy Mix	NE US Mix	Ontario Energy Mix	EcoCAR Energy Mix	NE US Mix	Ontario Energy Mix
E10 (baseline)	0.69	0.69	0.68	232	231	229
EtOH FFV: E85	1.18	1.17	1.16	185	182	176
Grid-Independent SI HEV: E10	0.47	0.47	0.46	157	157	155
Grid-Independent SI HEV: E85	0.79	0.82	0.81	126	121	116
Grid-Connected SI HEV: E10	0.46	0.45	0.42	160	152	135
Grid-Connected SI HEV: E85	0.68	0.69	0.65	139	128	109
CIDI Vehicle: BD20	1.43	1.43	1.43	163	162	159
Grid-Independent CIDI HEV: BD20	0.77	0.77	0.76	112	111	109
Grid-Connected CIDI HEV: BD20	0.66	0.66	0.62	130	122	104
Electric Vehicle	0.38	0.37	0.27	142	123	81
FCV: G.H2	0.39	0.39	0.38	123	120	115

Using the EcoCAR energy mix for electricity, electric vehicles ranked 1st for energy consumption and grid independent CIDI HEVs using biodiesel were best in emissions. However, fuel cell vehicles seem to be the best choice overall in terms of energy use and GHG production for the EcoCAR energy mix for electricity. Grid-dependent and grid-independent SI HEVs using E10 were respectively 3rd and 4th in energy use for the EcoCAR and Ontario energy mixes, thus they were considered as potential architectures by UOIT.

When only considering GHG emissions with the EcoCAR mix, electric vehicles ranked 6th showing that they are not the logical architecture to score high at this competition in regards with emissions. However, they score best in energy use, and considering the points weighting were second only to FCVs. The hydrogen-based architecture was not an option due to infrastructure requirements, so the full electric vehicle architecture placed first on UOIT's list. Additionally, electric vehicles are clearly more eco-friendly than any other propulsion technologies when using electricity generated in Ontario. Competition rules aside and considering that the electricity generated in Ontario is greener than the energy mix defined by the EcoCAR challenge,

the most environmentally friendly prototype the UOIT EcoCAR team could build would be a purely electric one.

Nevertheless, the decision making was not only influenced by the competition rules, but also by the university's field of expertise and the infrastructure available. Having no hydrogen refuelling infrastructure required of fuel cell vehicle teams, these choices were eliminated from the potential architectures. E85 and bio-diesel were also excluded, since the university does not have sophisticated emissions testing equipment. Having a large electric vehicle fleet, UOIT's expertise resided in the domain of electric vehicles which heavily influenced the decision matrix. This fleet comprises a total of 3 Chevrolet S10-EVs, 2 Ford Ranger EVs, a RAV4-EV, Chrysler TEV mini-van, 2 restored electric buses, a 2007 Toyota Prius PHEV and a 2003 Honda Insight HEV.

At this point in the powertrain selection process, only 3 architectures remained: HEVs using reformulated gasoline with 10% ethanol (E10), PHEVs using E10 and the full electric implementation. Among these, the full electric scored best especially from the energy consumption assessment. The next section further analyzes UOIT's top 3 architectures, which were: EV, EREV-E10, PHEV-E10.

2.5 EV, EREV-E10 and PHEV-E10 Performance Simulations

The 3 following architectures were studied in detail:

- EV
- EREV-E10
- PHEV-E10

Analysis was done through modeling and simulation using a software designed by Argonne National Laboratory known as PSAT (Powertrain System Analysis Toolkit, recently renamed Autonomie) [4]. The powertrain specifications - i.e. motor and generator power, battery energy capacity and vehicle weight - and the simulation results of acceleration, energy consumption for different drive cycles and vehicle range are

detailed in Appendix A. Table 2 presents a summary of the simulation results using the EcoCAR North American energy mix.

Table 2. Performance Simulation Summary [7].

	PHEV-30	EREV-60	Full Electric
Avg. Eco. (mpgge)	60.6	75.4	93.5
Electricity (Wh/km)	246	241	222
GHG (g/km)	178	162	155
Utility factor	0.512	0.739	0.971
Acc. 0-97 km/h (s)	8.9	8.7	9.5
Weight (kg)	2027	2043	2139
Electric Range (km)	47	95	444
Total Range (km)	~600	~600	444

It is to be noted that all 3 architectures meet the competition performance requirements which are specified in Table 3.

Table 3. EcoCAR Competition Performance Requirements [5].

Specifications	Competition Requirements
GHG Emission	≤ 217 g/km
Accel 0-97 km/h	≤ 14 s
Weight	≤ 2268 kg
Range	≥ 320 km

The battery capacity was necessarily chosen to achieve a range greater than 320 km, since the test procedures for range evaluation at competition were not clearly outlined by the organizers in the rules. These 3 simulations indicated the full electric architecture still has the lowest GHG emissions (155 g/km). The estimated fuel economy was 2.51 L/100 km (2.67 L/100 km in the vehicle technical specifications to be conservative) compared to 8.3 L/100 km for the production vehicle.

2.6 Selected Architecture

After considering the competition requirements and scoring criteria, the list of 11 possible architectures was reduced. Realistically, fuel cells could not be supported at UOIT because of the lack of an hydrogen refuelling infrastructure. Similarly, requiring advanced emissions testing equipment, bio-diesel and E85 fuel conversions were also rejected.

With the fields of expertise at the university and the provincial energy mix, the number of viable architectures was narrowed down to 3. In order to nominate the winner, these 3 architectures were modeled and simulated. Meeting all the competition requirements and having the lowest GHG emissions, a full electric vehicle prototype was selected as the UOIT EcoCAR team's design. The targets for this electric prototype were set by the team based on simulations and competition requirements, and are summarized in Table 4.

Table 4. Targeted Vehicle Technical Specifications [7].

Specification	Competition		UOIT Team
EcoCAR	**Production VUE**	**Competition Requirement**	**VTS Target**
Accel 0-60 mph	10.6 s	≤14 s	13.5 s
Accel 50-70 mph	7 s	≤10 s	7.5 s
UF Weighted FE*	8.3 L/100 km (28.3 mpg)	7.4 L/100 km (32 mpgge****)	2.67 L/100km (88 mpgge)
GHG emission***	250 g/km	224 g/km	165 g/km
Towing Capacity	680 kg (1500 lb)	≥680 kg @ 3.5% 20 min @ 72 kph (45 mph)	≥680 kg @ 3.5% 20 min @ 72 kph (45 mph)
Cargo Capacity	0.83 m^3	H: 457 mm (18") D: 686 mm (27") W: 762 mm (30")	H: 457 mm (18") D: 686 mm (27") W: 762 mm (30")
Passenger Capacity	5	≥4	5
Braking 60-0 mph	38 m- 43 m (123 -140 ft)	< 51.8 m (170 ft)	51.8 m (170 ft)
Mass	1758 kg (3875 lb)	≤ 2268 kg** (5000 lb)	2130 kg
Starting Time	≤ 2 s	≤ 15 s	≤ 2 s
Ground Clearance	198 mm (7.8")	≥178 mm (7")	210 mm (8.25")
Range (UF=.971)	> 580 km (360 mi)	≥ 320 km (200 mi)	402 km (250 mi)

*PSAT values were scaled back ~6% to arrive at VTS, charging efficiency 90%, upstream energy included, utility factor (UF) weighted fuel economy (FE).

**2318 kg, (5111 lbs) with optional rear callipers.

***GHG (green house gas emissions) are CO2 warming equivalents on a well to wheel basis.

**** mpgge (miles per gallon gasoline energy equivalent)

2.7 Selected Components

The examination of the state-of-the-art architectures for green vehicles, achieved so far in this chapter, led to choosing a battery electric vehicle design, along with establishing the performance goals for the prototype. After defining the architecture into which the GM donated vehicle was to be converted, the component selection process began. The purpose of this section is to acquaint the reader with the advanced components required by EVs and explain their selection process. Unfortunately for the UOIT EcoCAR team, not all the ideal components or first choices could be acquired for various reasons such as competition restrictions, availability or cost.

A typical EV is composed of a motor, an energy storage system (ESS), a battery charger, a DC/DC converter, a high voltage control module (HVCM) and a vehicle integration module (VIM). This collection of devices requires a sophisticated controller in order to function properly. Motors used in EVs are not simple DC motors. They are usually powered by a 3-phased high voltage AC source. However, the energy storage system provides DC high voltage, thus a motor controller, also named traction inverter module (TIM), is required to effectively convert DC voltage to 3-phased AC. The ESS is composed of a battery and a management system (BMS) that monitors the current, voltage, state of charge, temperature, etc. A battery charger is required to convert the AC from a power outlet to DC current at the battery. A high voltage control module (HVCM) is in charge of the high voltage safety and the power flow paths by controlling contactors on the high voltage bus located in the high voltage distribution box. A vehicle integration module (VIM) is useful for doing general controls and for bridging devices with one another. An EV does not have an alternator to power the accessories in the car, thus a DC/DC converter must be added to the vehicle. For high voltage safety, a ground fault interrupt (GFI) monitors current leakage between the accessories grounded to the chassis and the floating high voltage systems. Extra gauges and displays are normally added to visually output information specific to EVs. It is also common to find a datalogger in EV prototypes for research purposes.

All the controllers mentioned previously interact with the car through sensors (input) and actuators (output). Examples of sensors would be potentiometers for the accelerator, the brake pedal and the ignition key; voltage and current sensors in every high voltage device; speed sensors on the wheels, the shifter lever or the interlock between high voltage components. On the other hand, actuators would be contactors on the high voltage bus; firing signals for MOSFETs and IGBTs in power electronic devices or a signal to control the small DC motor powering the speedometer needle in the instrument cluster.

This subsection described the following additional key components added to the production VUE to implement the electric architecture: battery cells, battery management system (BMS), charger, motor, motor controller, vehicle integration module (VIM); along with the secondary devices such as: a DC/DC, datalogger, ground fault interruptor (GFI) and dashboard graphical user interface. These components are positioned in the UOIT EcoCAR EV prototype as shown in Figure 6.

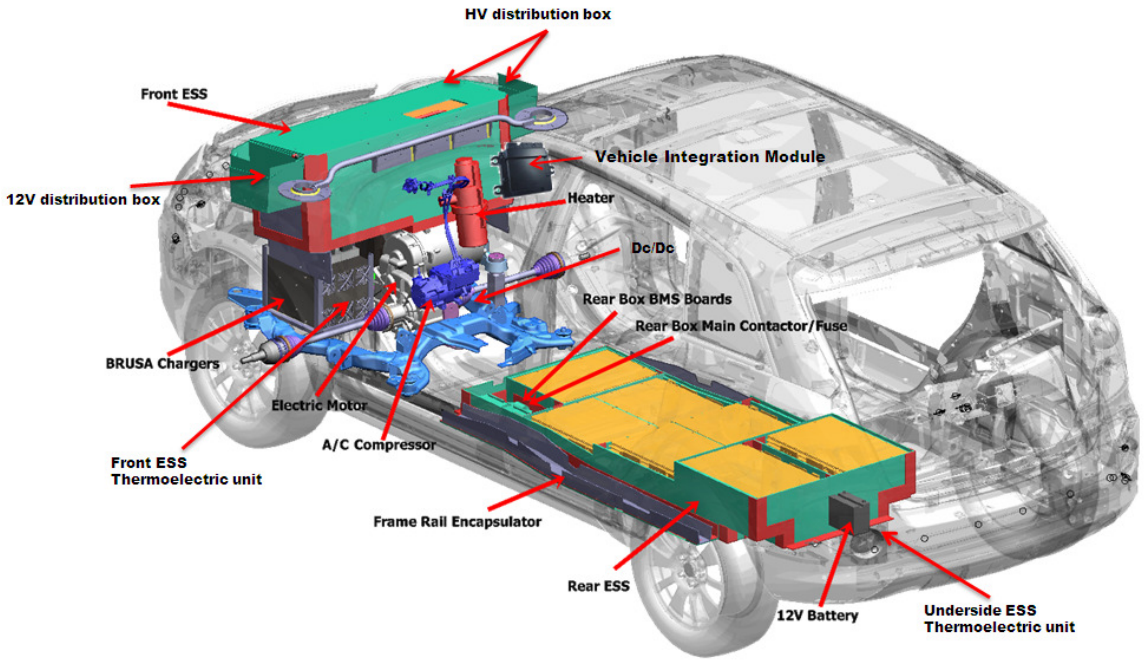


Figure 6. EV ECUs in the UOIT EcoCAR EV Prototype [9].

2.7.1 Battery

Batteries are known as the main detractor causing a late arrival of electric cars on the market. At first, heavy lead acid batteries were tried, then it was nickel cadmium (NiCd) followed by nickel-metal hydride (NiMH) with a higher energy density, but still too heavy for automotive applications. Today, lithium batteries are the main prospect for EVs, PHEVs and HEVs. However, they are fragile, expensive and flammable. Therefore, they require a protection and management system known as a battery management system (BMS). Since they're also a source of energy for EVs and PHEVs, it's essential for a BMS to monitor them and send information to the driver. This subsection presents the battery cells and battery management system (BMS) selected for the EcoCAR project at UOIT.

2.7.1.1 Battery Chemistry: Lithium Polymer

The main criteria for the battery were highest volumetric and gravimetric energy density. Although 18650 format cells as used in laptops, and also used in Tesla products, have the highest energy density by both gravimetric and volumetric measure, they are extremely difficult to integrate due to the quantity required. The approach of using the largest single cells of highest energy density for packing simplification, component count reduction and volume minimization of the whole ESS was pursued. The Kokam 240 Ah cell, using NCM-lithium ion chemistry met these objectives in the best regard among all the cells surveyed and available at the time. Figure 7 compares the different cells analysed and Table 5 reviews characteristics of the Kokam cell chosen.

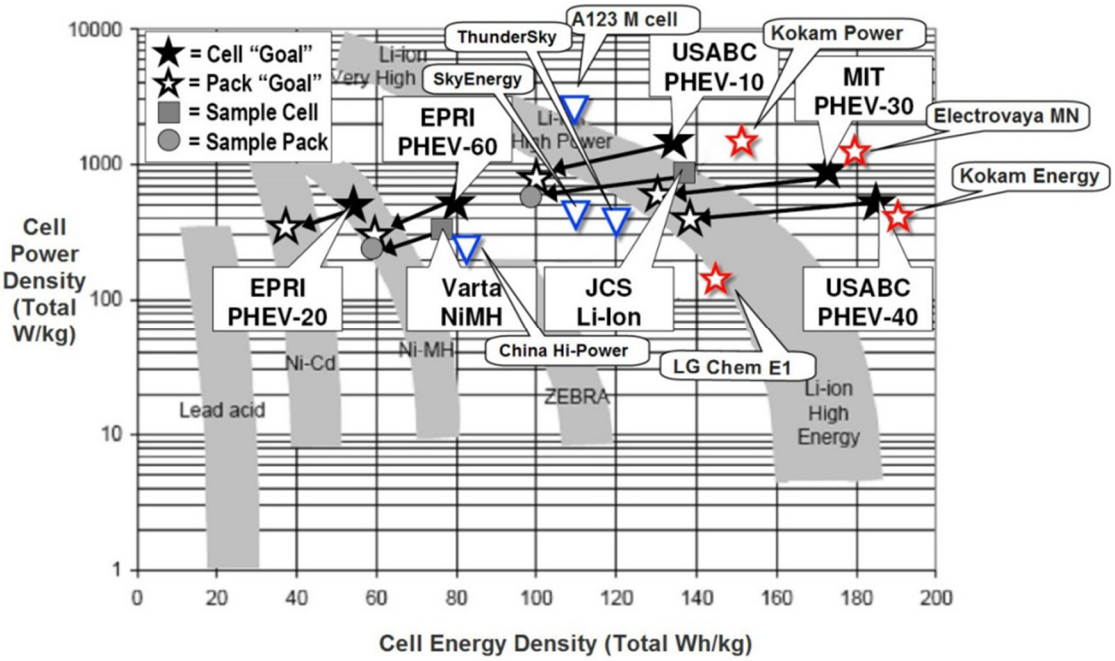


Figure 7. Cell Comparison [10].

Table 5. Kokam SLPB 160460330 [11].

	1 cell	Final Battery Design
# of cell	1	94
V_{min}	2.7 V	282.0 V (3.0 V/cell)
V_{nom}	3.7 V	347.8 V (3.7 V/cell)
V_{max}	4.2 V	394.8 V (4.2 V/cell) (competition limit = 400 V)
Size (mm)	325 x 455 x 16	
Operating temperature (degC)	Charge : 0-40 Discharge : -20-60	Charge : 0-40 Discharge : -20-60
Max. charge current	1 C	47.4 A continuous (charger limit)
Max. discharge current	2C	240 A continuous (inverter limit)
Peak discharge current	480 A	400 A (inverter limit)
Cycle life @ 80 DOD	> 800 cycles	≈ 1500 cycles
Weight	4.9 kg	460.1 kg
Capacity	240 Ah	
Energy	0.888 kWh	83.5 kWh

The battery design was based on using a single string of cells, individually monitored for voltage and temperature. A modular cell frame design was evolved that incorporated liquid cooling in response to organizer's demands for active temperature control. The frames were stacked into different height modules among 2 battery boxes, one under the vehicle, the other replacing the IC engine space underhood. The designs had to be validated for crush space and puncture resistance, an involved and lengthy exercise for the mechanical engineering structural team. The cells were also chosen for their cyclic life, safety with respect to thermal runaway, acquisition price and delivery.

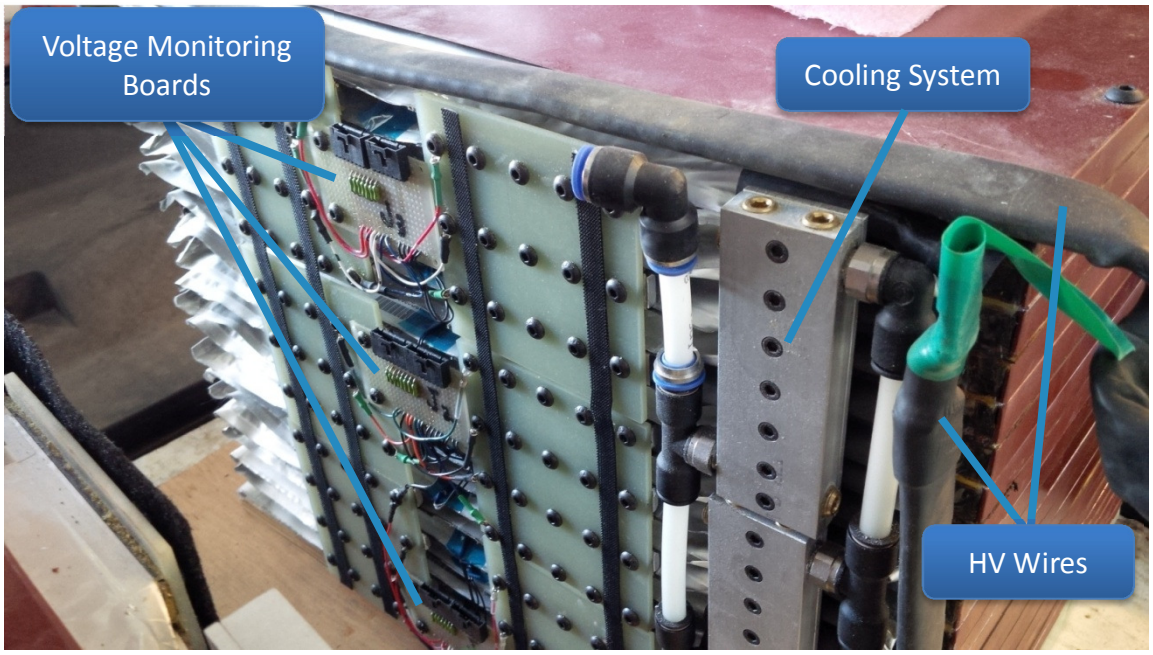
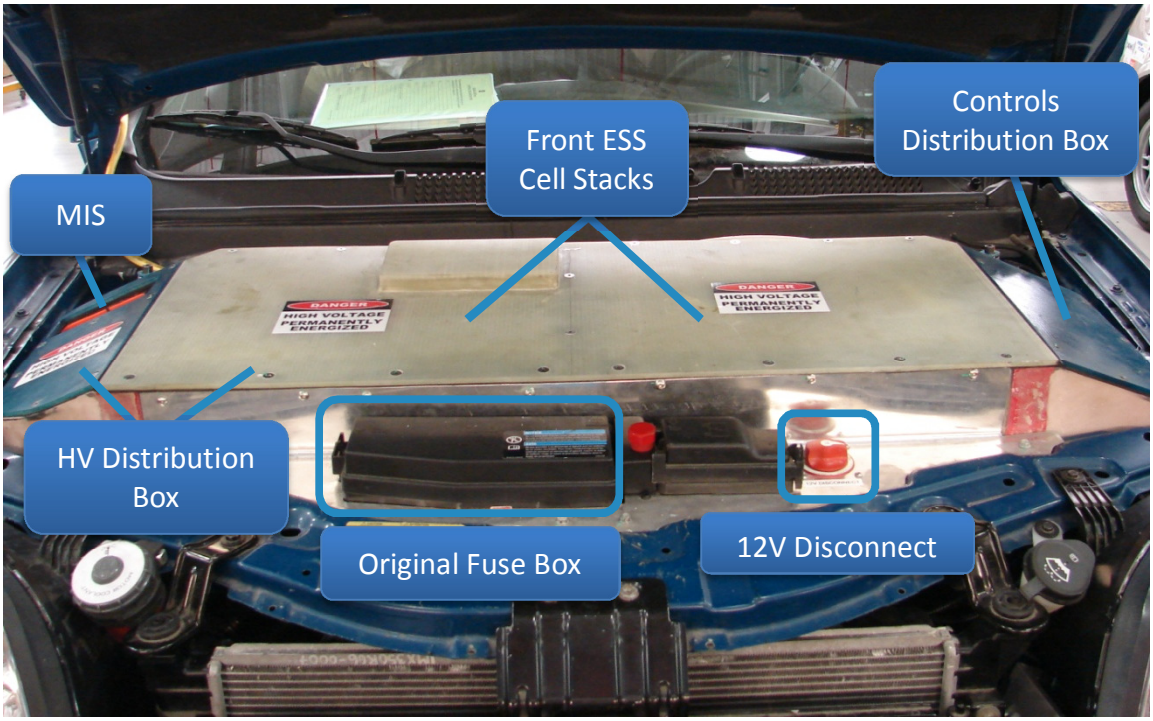
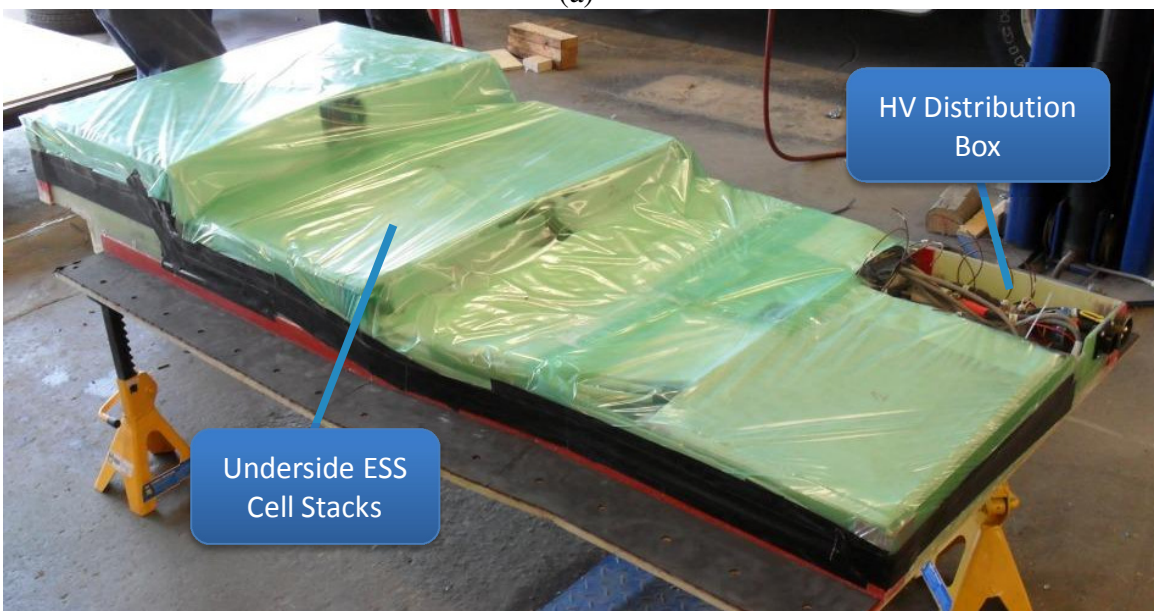


Figure 8. Cell Stack, including cooling frames and voltage monitoring boards.



(a)



(b)

Figure 9. (a) Front ESS (b) Underside ESS.

2.7.1.2 Battery Management System: REAP

Lithium batteries need to be charged and discharged with caution in order to avoid hazardous situations and ensure the battery life does not degrade. During the charge or discharge processes, a battery management system accomplishes the supervisory and

control tasks required to maintain the battery in optimal condition. Some examples of these tasks are to:

- Monitor individual cell conditions (max. and min. cell voltage, max. and min. cell temperature and battery current)
- Calculate the State-of-Charge (SOC) and State-of-Health (SOH)
- Protect the cells from over discharge
- Prolong battery life by terminating the charge early
- Balance the cell voltages

Although there are many BMS systems available today, in 2009 the choices were limited and one of the most advanced designs employing a flying capacitor cell balancing strategy, CAN bus communication and programmability was available thru a British company named REAP Systems. The selected BMS model employed was the REAP BMS14C and its specifications are listed in Table 6 [12]. It should also be mentioned that one of the 7 BMS modules was reconfigured by the manufacturer to be compatible with 10 cells instead of 14, for a battery totalizing 94 cells. A picture of the BMS board is displayed in Figure 10 and an in-vehicle layout in Figure 11.

Table 6. BMS14C Specifications [12].

	1 module	Final BMS Design
# of modules	1	7
# of cell monitored	14	94
# of thermistor	7	49
Weight (g)	68	476
Size (mm)	132 x 72 x 19	7(132 x 72 x 19)

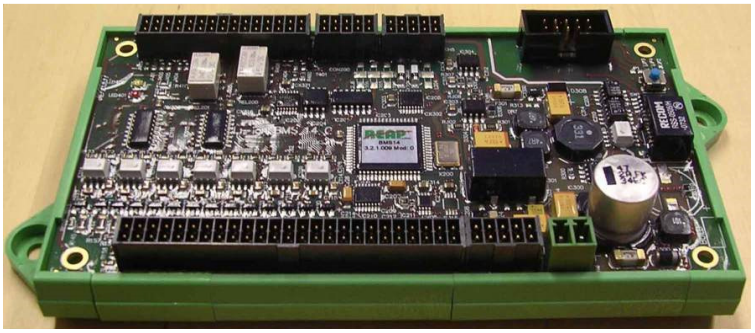
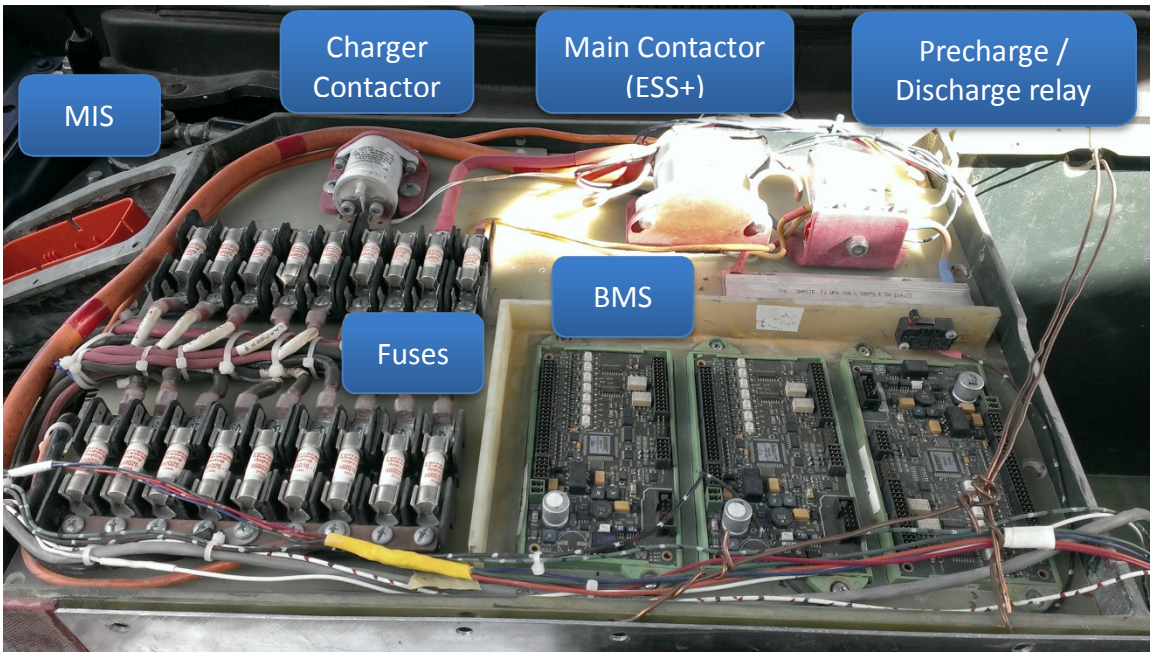
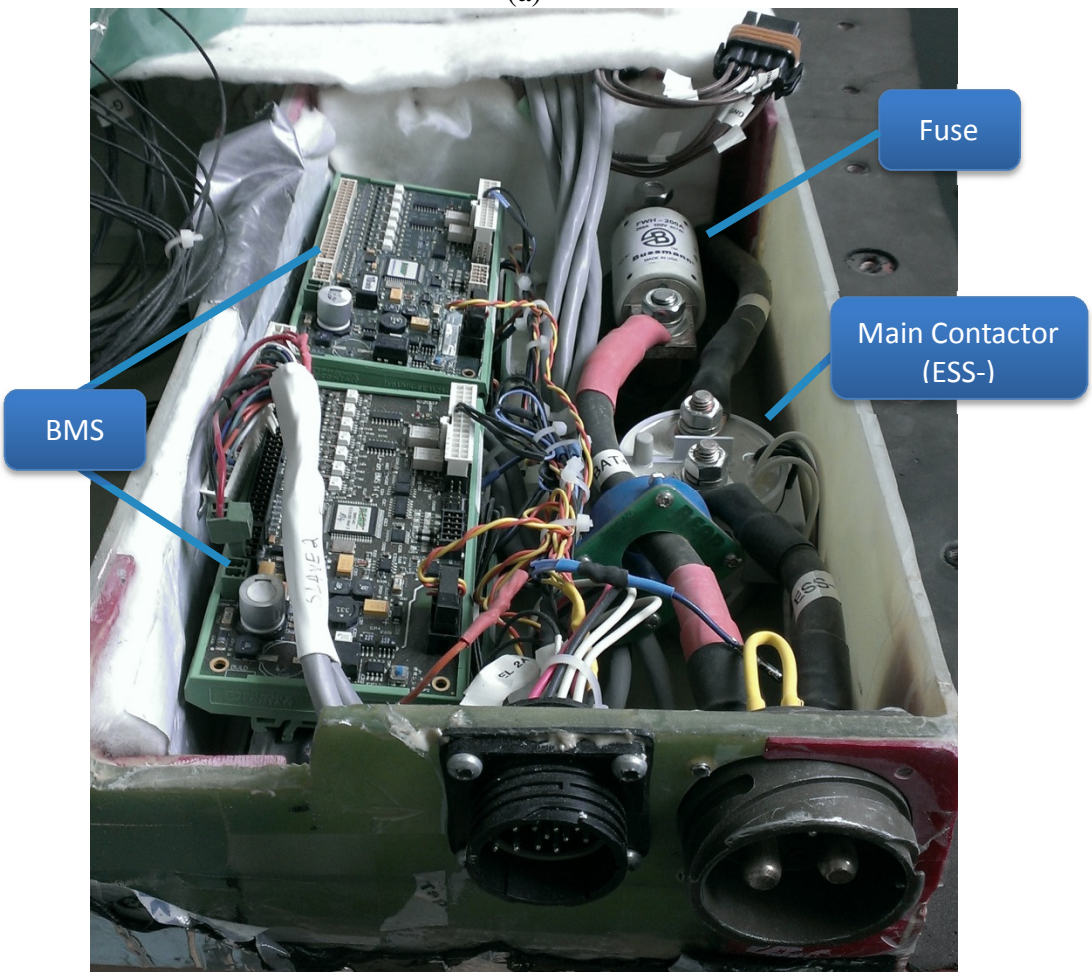


Figure 10. BMS14C: 1 module.



(a)



(b)

Figure 11. In-vehicle BMS layout (a) Front ESS (b) Underside ESS.

2.7.2 On-board charger: 5 Brusa NLG513 3.3kW

BRUSA chargers were required by the competition organizers. This charger is robust and has very flexible configuration settings making it versatile enough to accommodate the needs of 16 universities each having a unique architecture. For example, it is programmable for different output voltages and currents or charging time, and is compatible with a 120 V_{ac} input as well as 240V_{ac}. Moreover, multiple charging phases can be programmed and are used in UOIT's charging strategy to coordinate the operation of the 5 independent BRUSA chargers in the prototype. Table 7 details the characteristics of 1 BRUSA charger, along with the specifications of the UOIT EcoCAR prototype charging profile [13]. Figure 12 displays an individual charger and the layout on the engine craddle, while Figure 13 shows the SAE J1772 charging plug located behind the front licence plate.

Table 7. BRUSA NLG513 3.3kW Specifications [13].

	1 module	**Final Charger Design**
# of modules	1	5
$V_{in\ rms\ ac\ max}$	230 V	230 V
$V_{in\ rms\ ac\ min}$ (at reduced output power)	≥ 120 V (1 kW)	≥ 120 V (1 kW)
Efficiency	90 to 93%	90 to 93%
$I_{out\ dc\ max}$	12.5 A (@ 260 V)	58.5 A (@ 282 V)
$V_{out\ dc\ max}$	520 V (@ 6.25 A)	394.8 V (@ 41.8 A)
Max. output power	3.3 kW	16.5 kW

(a)

(b)

Figure 12. (a) Brusa NLG513 3.3kW. (b) 3 of 5 Brusa NLG513 3.3kW Integrated.

Figure 13. SAE J1772 Charging Inlet.

2.7.3 Motor and motor controller: Delphi S-10EV and MES-DEA TIM600

The electric traction motor is controlled by a traction inverter module (TIM), also referred to as motor controller. Only the combination of a motor and gearbox assembly with differential can be deemed useful. Its gear ratio and power rating need to match the application. A few choices were available, with caveats listed below.

- GM e-drive (GMT101X), denied use (110 kW)
- EcoStar (Siemens) drive – heavier but only rated at 67 kW
- Delco System 110's (S10-EV) – rated 85 kW @ 350 A
- MES-DEA – 30 kW continuous, peak unknown, no gearbox
- UQM Powerphase 125 - no matching gearbox, unobtainium price

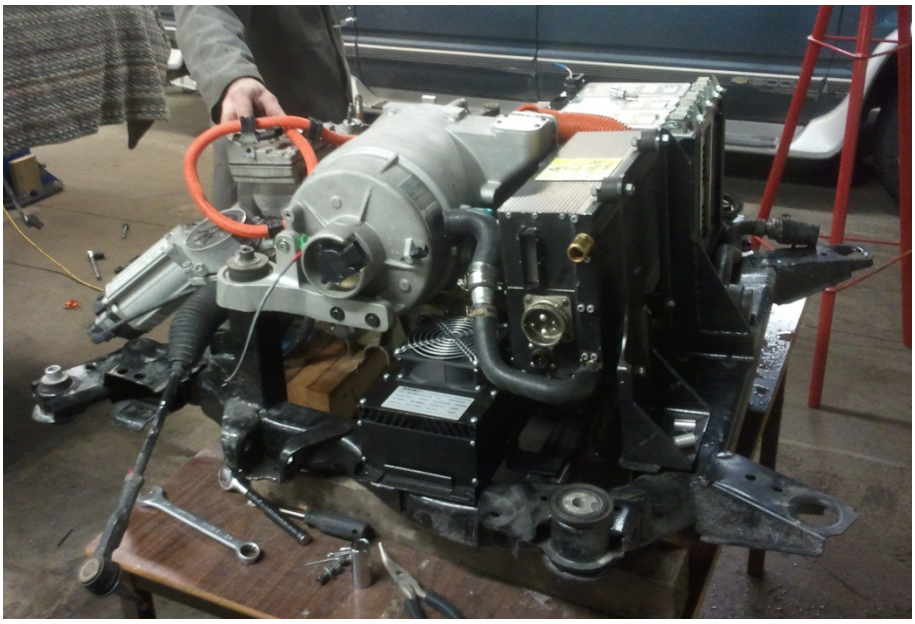
In the end, the S10-EV overdriven to 400 A as per the EV1 specifications was chosen for its compactness, lowest weight and package size plus availability. The team selected the MES-DEA TIM600 which is designed to run asynchronous induction machines, can be fully calibrated and auto-tune according to the motor's dynamic characteristics. It accepts torque commands directly from the acceleration and brake pedals, along with digital inputs for gear selection and ignition position. Regenerative and hydraulic braking can also be easily blended with a brake pedal potentiometer.

Table 8. Delphi S10-EV Delco System 110 @ 400 A, 240 V_{rms} [14].

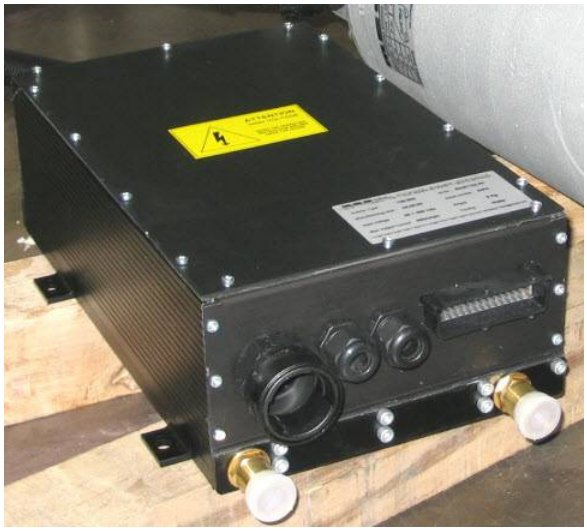
Type	3-phase induction motor
Power	102 kW (137 hp) @ 7000 rpm
Torque	150 Nm (110 ft/lbs) @ > 0 rpm
Transmission	Gear reducer and differential

Table 9. MES-DEA TIM600 [15].

Service voltage	12 V
$V_{in\ dc}$	80-450 V_{dc}
$I_{out\ ac\ nom}$	266 A_{rms}
$I_{out\ ac\ max}$	400 A_{rms}
f_{out}	0-500 Hz
Maximum power (400 A @ 400 V_{dc} input)	160 kW (213 hp)
Weight	16.5 lbs (7.5 kg)
Size (mm)	150 x 212 x 346
Liquid cooled (min. flow)	10 L/min 50/50 EG



(a)



(b)

Figure 14. (a) Delphi S10-EV with inverter, chargers and DC/DC. (b) MES-DEA TIM600.

2.7.4 Vehicle Integration Module: MotoTron ECM-5554-112-C00-M

The constantly growing complexity of automotive controllers forces engineers to develop innovative programming solutions in order to efficienty design robust embedded software. A model-based programming approach seems to be a solution to overcome this challenge. For that reason, the competition offered a choice between 2 rapid prototyping

embedded controllers employing this coding technique, to use as a vehicle supervisory control module. This controller is referred to as a vehicle integration module (VIM) throughout this thesis, since it is far more than just a supervisory controller. It also acts as a CAN message emulator, a high voltage control module (HVCM), a communication bridge between new powertrain components and the rest of the subsystems inherited from the production VUE, and is thus the central controller of the powertrain. Table 10 compares a MotoTron to a MicroAutoBox from the 2 competition sponsors, Woodward MotoHawk Control Solutions and dSPACE respectively.

Table 10. Comparison MotoTron vs MicroAutoBox [16][17]

	MotoTron	**MicroAutoBox**
Model	ECM-5554-112-C00-M	ds1401
# of Pins	112	156
Processor (MHz)	80 MHz	800 MHz
Memory (Mb)	2	8
CAN ports	3	4
Inputs	33 analog/digital	16 analog and 16 digital
Outputs	8 Logic Level (5V) 14 Low Side Driver (up to 12V and 7A)	8 analog and 10 digital (5V or 12V)
Contactor Compatibility (low side drivers sinking up to 5 A)	Yes	No (External I/O board required, 5000$)
Cost (controller, harness, software)	$2,000-$4,000	$30,000-$35,000
Automotive Production Grade	Yes	No

Even though the dSPACE controller has more inputs and outputs (I/Os) and processing power, the MotoTron was selected for its better automotive packaging allowing the team to directly mount it in the engine compartment, along with its capabilities for controlling the battery and charger contactors. Also, 3 CAN ports are sufficient to interact with all the vehicle communication networks. The controller is shown in Figure 15. It should be noted that a less powerful MotoTron, ECM-0555-80, was first selected and used during the second year of the competition, but was updated to the more powerful ECM-5554-112 during year 3.

Figure 15. MotoTron Controller ECM-5554-112-0904-C00-M [16].

2.7.5 Other subsystems

2.7.5.1 DC/DC: MES-DEA 400 V – 1000 W

The 2 main criteria used to select the DC/DC controller were: whether it was automotive grade and had a power output greater than the estimated requirement of at least 400 W. Meeting both criteria, being affordable and even leaving enough extra power for unexpected power draw made the MES-DEA 1 kW a logical choice for the vehicle's DC/DC converter. The specifications of this converter are listed in Table 11, and it is shown in Figure 16. Also, a second unit was later purchased to independently supply the energy consumed by the battery thermal management subsystem.

Table 11. MES-DEA DC/DC Converter 1 kW Specifications [18].

	MES-DEA
$V_{in\ dc}$ (V)	190-400
$V_{out\ dc}$ (V)	13.3-14.4
Efficiency (%)	>90%
$I_{out\ dc\ max}$ (A)	70
Maximum power output (W)	1000
Weight (kg)	3.6

Figure 16. MES-DEA DC/DC Converter 1 kW [18].

2.7.5.2 Datalogger: Vector CANlog4

The competition required the teams to record specific variables from the vehicle CAN buses while competing in dynamic events, thus the presence of a datalogger was mandatory. As discussed latter in Section 3.7, dataloggers can be either standalone by having embedded memory or can constantly stream the information to a computer hard drive. Non-standalone loggers were allowed during the second year, but not for the final competition. Consequently, the selected datalogger had to be standalone. 2 devices were considered: a CANcaseXL log and a CANlog4, both made by Vector. Due to its two CAN ports limitation, two separate CANcaseXL log would have been required to record information from the 3 high-speed CAN buses, i.e. GMLAN HS, GMLAN PTE and UOIT added bus, and also if necessary the single-wire CAN bus, i.e. GMLAN SW. Therefore, despite its higher cost and more complex programming method, the more powerful and versatile CANlog4 was selected.

Table 12. CANlog4 Specifications [19].

CAN ports	5
• **Baby board: HS CAN**	3
• **Baby board: Single-wire CAN**	1
• **Wake-up: HS CAN**	1
External flash card	64 Mb
Programming language	Log Task Language (LTL)
Programmable LEDs	4
Status LEDs	2
I/O expansion boards	Yes

Figure 17. CANlog4 [19].

2.7.5.3 Ground Fault Interrupt: Bender IR470LY

This unit is connected to the high voltage and low voltage buses to verify that there are no leakage current in the electrical system. The unit distinguishes between low and high leakage scenarios by setting flags that send signals to the VIM so that the appropriate actions can be taken. If a low leakage current is detected the motor controller can be placed in an emergency mode that limits the output current of the motor and a warning LED is turned on to inform the user. If a high leakage current is detected the HVCM in the VIM sends a signal to open the main contactors, disabling the motor controller output, to stop the vehicle. A ground fault interrupter of the Bender brand was mandatory according to the competition requirements. They are specifically designed to detect ground faults in floating (ungrounded) AC systems. The Bender IR470LY (series 470) [20] employed is shown in Figure 18.

Figure 18. Bender IR470LY [20].

2.7.5.4 Custom instrument cluster screen

Some off the shelf screens were first considered for integration to the dashboard near the instrument panel cluster (IPC), such as the Woodward embedded display series, the RM Display 1001, and even an upgraded version of the navigation system of the vehicle. However, having a customizable screen able to display a broad variety of information being read from the CAN bus was a priority. That's why the team selected an in-house LCD screen and controller designed, implemented and programmed in C code by the author. Furthermore, the small size of the screen allowed it to be integrated directly in the IPC. Examples of information displayed are the battery voltage, current, temperature, state-of-charge (SOC), errors and warnings. It was also planned to display the car ready signal and GFI status on it, but re-using the chime and lights of the IPC was successfully accomplished and more intuitive for the driver. This screen as it was being programmed on a test bench, before integration in the IPC, is shown in Figure 19.

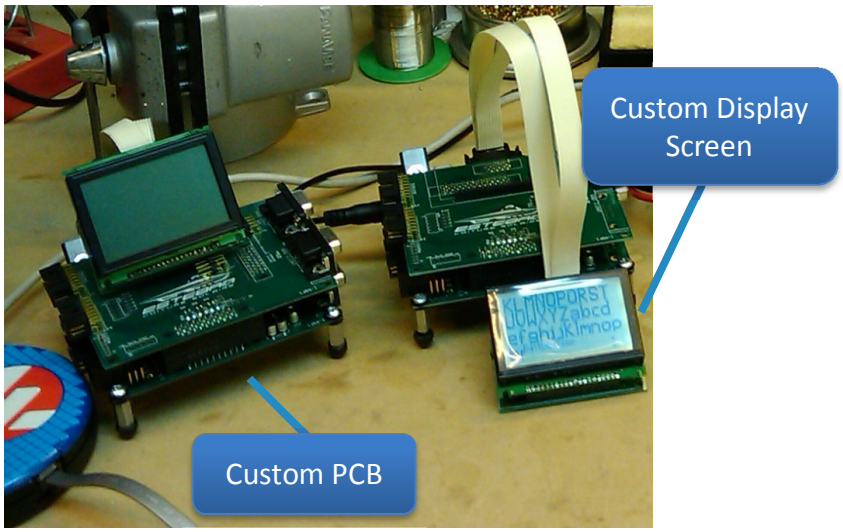


Figure 19. Dashboard Graphical User Interface.

2.7.6 Summary of Selected Subsystems

Table 13 lists the key devices selected to implement UOIT's new architecture into the 2009 Saturn VUE.

Table 13. Main EV subsystems selected.

Components	Models
Battery Cell	94 Kokam SLPB 160460330
Battery Management System (BMS)	7 REAP BMS14C
Charger	5 BRUSA NLG513
Electric Motor	Delco S10-EV
Motor Controller	MES-DEA TIM600
Vehicle Integration Module	MotoTron ECM-5554-112-C00-M
2 DC/DCs	MES-DEA 1kW
Datalogger	Vector CANlog4
Ground Fault Interrupt (GFI)	Bender IR470LY
Dashboard Graphical User Interface	In-house

2.8 Summary

The different powertrain configurations and practical fuel choices for implementation in today's vehicles to increase fuel economy and reduce GHG emissions were considered by EcoCAR organizers. The well-to-wheel energy use and GHG production were then studied and analysed by the UOIT team for several different architectures. The modeling and simulations of the 3 most promising ones lead to the selection of a full function electric vehicle. The last section gave an overview of the sophisticated hardware required by modern electric vehicles, along with a list of the key electrical components selected for implementation in the prototype designed. In the subsequent chapter, the theory and vehicular application of the controller area network (CAN) communication protocol is presented.

Chapter 3

CAN for Vehicles

3.1 Introduction

The powertrain architecture selection process, the specialized hardware found in a typical electric car and a description of the components selected for implementation on the UOIT EcoCAR are described in the preceding chapter. The goal of this section is to focus on establishing a robust in-vehicle communication network not only to connect these controllers, but also integrate with ECUs in today's vehicles. This efficient and robust field communication protocol is termed the controller area network, also known as CAN bus [29]. This section constitutes one of the few non-confidential documents available that details CAN bus for vehicles. It covers the fundamentals of the communication protocol theory, along with hands-on application useful to most CAN users. In summary, an overview on CAN protocol fundamental theory, a description of the hardware required to create a CAN environment, list of devices supporting CAN available on the market, a rigorous definition of the nomenclature used to accurately define CAN messages and CAN signals, different CAN bus configurations, a variety of techniques to use identifiers, and a description of automotive applications.

3.2 Overview

3.2.1 History

Robert Bosch GmbH started developing the controller area network protocol in 1983, but only officially released it at the Detroit's Society of Automotive Engineers (SAE) congress in 1986 [32]. At that time, the automotive industry was looking for a reliable in-vehicle communication system. CAN was developed for use in industrial environments and for in-vehicle networks, but was not the only networking technology competing in the race for the automotive market. Volkswagen Group was developing the A-bus, while another European group was developing the French Vehicle Area

Network (VAN). In 1992, Mercedes-Benz built the first car integrating CAN bus helping CAN to finally win this networking technology race in the mid-nineties [32].

Since 1996, the on-board diagnostics II (OBD-II) standard is a mandatory vehicle diagnostics standard [56]. Five communication protocols are included in that standard, CAN being the most important. With the arrival of the CAN-based protocols DeviceNet and CANopen in the mid-nineties, CAN rapidly found applications in factory automation. Since 2000, this robust field bus kept expanding to other industries, such as elevators and forklifts; connections between subsystems in ships, flight status sensors and navigation systems in aircrafts; factory and building automation, automatic door control for monorails; different types of industrial controls; medical systems; as well as laboratory and operating room automation. Some of the important CAN features are its multi-master capability; its built-in error detection and correction capability; as well as its unique fault confinement [22].

3.2.2 Properties of CAN Protocol

3.2.2.1 Basic Concepts

CAN bus is a serial communication protocol level supporting real time systems with a high reliability. It handles the detection of collisions, the detection of errors, the retransmission of corrupted messages and the prioritization of sent and received messages. The identifier length can be 11 bits (standard format) or 29 bits (extended format) while the data length can vary between 0 and 8 bytes. The bitwise arbitration using the identifier (ID) gives a static message priority to the protocol. This property is possible because of the binary logic used by the protocol, either dominant (logical 0) or recessive (logical 1). When a dominant and a recessive bit are simultaneously transmitted, the dominant bit supersedes the recessive one [22], see Appendix B.

This protocol uses a producer-consumer multi-master message model, instead of the more common client-server model. Its principal characteristic is the interpretation of the identifier in a message [25]. It does not inform about the destination of the message, but rather indicates where it was sent from; in other words, providing the source of the data.

CAN messages can be transmitted periodically, on request, or on a state change at a uniform and fixed bit rate up to 1 Mb/s, within a CAN bus or CAN network. As explained in Section 3.2.3, the speed, or bit rate, can have different values in different networks. A similar concept applies for the message format: standard or extended.

The CAN communication protocol shows configuration flexibility by requiring no hardware or software modification for any node when a new node is connected to the CAN network. The quantity of nodes on such a network is theoretically unlimited; however, delay times and electrical loads will place an empirical limit on a bus having too many nodes. Sleep mode and wake-up are available options for each node to reduce power consumption [41].

Since the reliability of a communication system is paramount, each node accomplishes powerful safety tasks such as error detection, signalling and self-checking. Each node is able to identify the difference between temporary errors and permanent failures of a node. In the event of a permanent failure, the other nodes on the network automatically command the faulty node to switch off [23].

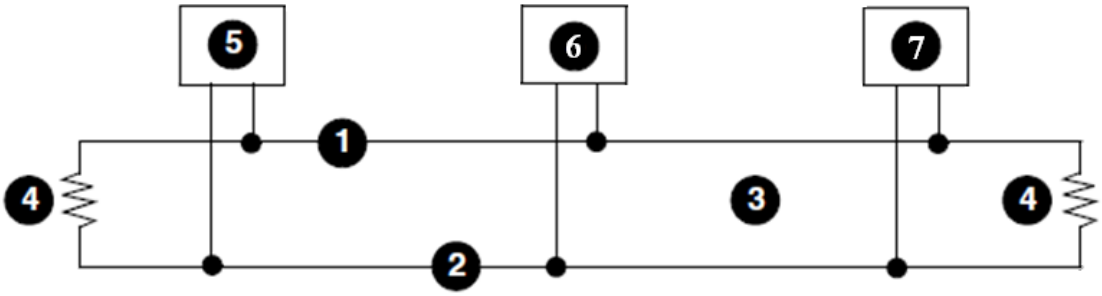
3.2.2.2 Properties

In summary, the priorities of the CAN communication protocol are [22]:

- Configuration flexibility
- Prioritization of messages (bitwise arbitration using the identifier)
- Simultaneous reception by multiple nodes with time synchronization
- Multi-master system
- Guarantee of latency time
- Error detection and signalling (by each node)
- Automatic retransmission of corrupted messages (once the bus is idle again)
- Error distinction (between temporary errors and a permanent failure of a node)
- Fault confinement (automatic switch off of defective nodes)

3.2.3 Physical Characteristics

The CAN specification does not define how the single channel which carries bits is implemented. A common way to implement a CAN bus is by using a single wire and a ground. However, the most typical implementation and the main focus of this work is using 2 twisted differential wires, CAN high and CAN low, with 2 termination resistors of 120 ohms each. Figure 20 shows a typical CAN bus topology when two differential wires are used [25].

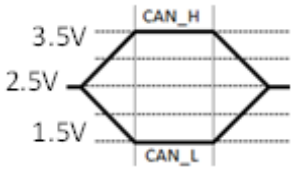
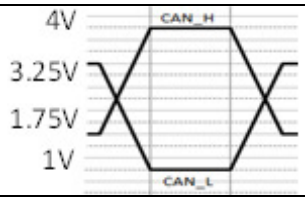
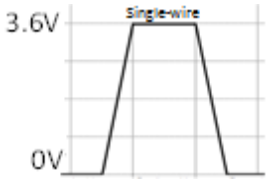


#	Description
1	CAN High wire (CANH)
2	CAN Low wire (CANL)
3	Voltage between CANH and CANL
4	Termination resistors (120Ω)
5	Node (i.e. ECU #1)
6	Node (i.e. ECU #2)
7	Node (i.e. ECU #3)

Figure 20. Topology

Depending on the configuration of the network and the environment, the transmission distance can reach up to 1 km. Table 14 summarizes 3 common specifications of the protocol.

Table 14. Differences between Low Speed and High Speed.

Parameters	ISO 11898-2	ISO 11898-3	SAE J2411
Name	high-speed	fault tolerant (low-speed)	single-wire
Baud rate	up to 1Mb/s	up to 125kb/s	33.3kb/s (up to 83.3kb/s)
Number of wires	2	2	1
Dominant bit level Recessive bit level			
Termination	2 resistors of 120Ω	Each node needs to terminate both CAN lines individually	9,09kΩ load resistor
Length	Limited by busload and data rate 40m @ 1Mb/s 100m @ 250 kb/s 1km @ 50kb/s	Limited by busload and data rate	
Number of nodes	Limited by busload and data rate	up to 32	up to 32
Maximum bandwidth	2000 msg/s @ 250 kb/s		
	4000 msg/s @ 500 kb/s		
	8000 msg/s @ 1 Mb/s		

As further explained in Appendix E, the bus load is associated with CAN transceivers and depends on internal resistors inside the transceiver and line capacitance, whereas the maximum speed and length are dependent on the acknowledgment bit, since it has to travel from the receiving ECUs to the transmitting one in a timely manner. The field of applications of each of these 3 specifications is broad. Section 3.8.1 details applications in the automotive industry. To assure robust arbitration when designing a network, the bandwidth should ideally be kept below 70% of the maximum bandwidth [26]. Bits travel on the bus using a non-return to zero (NRZ) bit encoding and decoding. This means the bit level is maintained over a full bit time and changes at the following bit time if a complementary bit is transmitted [31].

For synchronization purposes, a maximum of 5 consecutive bits of equal logic level is allowed before the insertion of a complementary bit [22]. This technique is called bit stuffing. As explained previously, the 2 bit levels are defined as dominant and recessive, and are associated with the logic level in Table 15.

Table 15. Bit logic level.

Bit	Logic Level
Dominant	0
Recessive	1

The physical waveforms on a CAN bus associated with these 2 logic levels can be found in Appendix B and the standard connectors used for CAN buses are covered in Appendix D.

3.2.4 Data Transmission

There are four different frame types [22]:

- Data Frame (standard or extended)
- Remote Frame (standard or extended)
- Error Frame (generated by the protocol manager)
- Overload Frame (generated by the protocol manager)

The data frame can be divided in 7 subsections [22]:

- Start of Frame
- Arbitration Field
- Control Field
- Data Field
- CRC Field
- ACK Field
- End of Frame

3.2.4.1 Difference between Data Frame and Remote Frame

The remote frame is a data frame having a payload equal to 0, it carries no data. It is used to request data from specific nodes. Another difference between these 2 frames is the value of the remote transmission request (RTR) bit in the arbitration field, dominant for data frame and recessive for remote frames. Figure 21 and Figure 22 detail the general structure of data frames and remote frames, translated from [25]. Additional info on the different frame types and formats can be found in Appendix B on CAN theory.

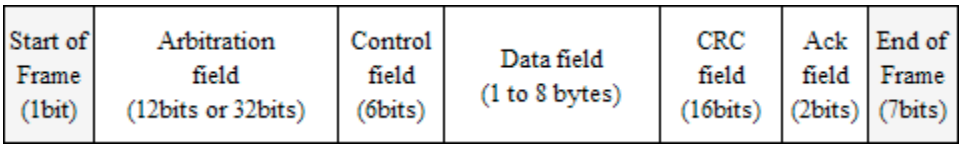


Figure 21. Data frame general structure [25].

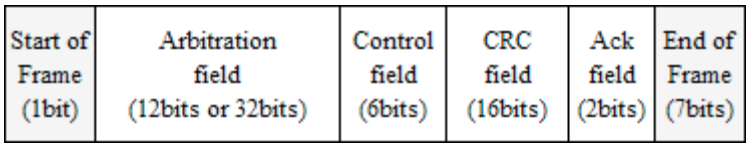


Figure 22. Remote frame general structure [25].

3.2.5 CAN Standards for Vehicle Applications

Table 16 groups the 3 different subdivisions of the high-speed CAN specification for vehicle applications SAE J2284, and the specification of the single-wire version SAE J2411 [50]. These specifications are based on the CAN definition of the physical and data link layers of the open system interconnection (OSI) reference model. However, they define them more precisely, i.e. with more restrictions. For example, these SAE specifications are specific to using non-shielded wires.

Table 16. SAE specifications for CAN buses applications in vehicles [50].

Name	Description	Status
SAE J2284-1	High-speed CAN for vehicle applications @ 125kb/s	Issued: 07-Mar-2002
SAE J2284-2	High-speed CAN for vehicle applications @ 250kb/s	Issued: 07-Mar-2002
SAE J2284-3	High-speed CAN for vehicle applications @ 500kb/s	Revised: 02-Mar-2010
SAE J2411	Single-wire CAN for vehicle applications @ 33kb/s	Issued: 14-Feb-2000

SAE J2284-3, providing a higher baud rate, and SAE J2411, presenting a cheaper solution, seem to be more common in vehicles then SAE J2284-1 and SAE J2284-2. Being the only CAN specification updated and re-published by the Society of Automotive Engineers since their first release, it is safe to assume that SAE J2284-3 will continue to be used for automotive applications in the foreseeable future.

3.3 CAN Transceiver and Protocol Controller

It is not really necessary to know all the details in the transmission management of error frames, overload frames, the start of frame (SOF), the acknowledgement (ACK) and the end of frame (EOF). Most CAN users, for instance application developers, only need to understand the general idea of the arbitration field (standard or extended identifier (ID)), the control field (encoding method of the data length code (DLC)) and the data field (endianness) [26].

The CAN protocol controller takes care of the details, such as applying the protocol specifications. It automatically handles the protocol and errors. However, a CAN transceiver is required to convert the bus voltage to a logic level value and to carry out the opposite conversion. Even though most microcontrollers supporting CAN have an integrated CAN protocol controller improving performance, they still need CAN transceivers, sometime called line drivers. Nevertheless, commercial controllers have these two devices integrated into their circuitry. Figure 23 summarizes these 3 cases and Table 17 gives examples of different products on the market.

Additional theory explaining at an electrical level how the bidirectional analog/digital conversion is achieved inside a CAN transceiver chip is covered in Appendix E.

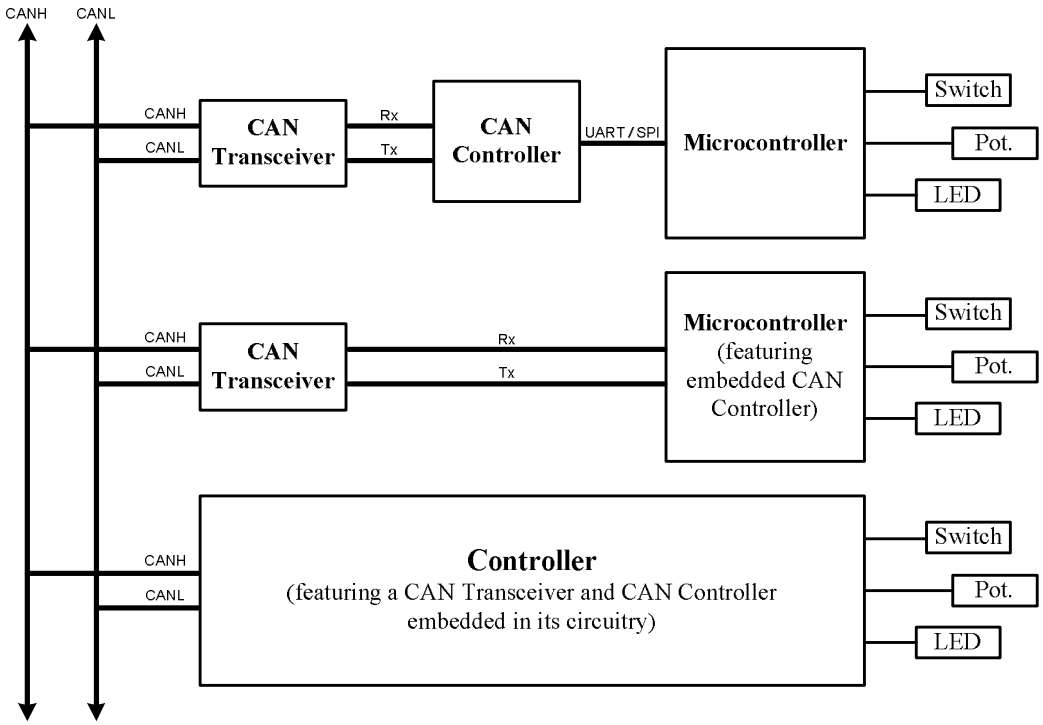


Figure 23. CAN Transceiver, CAN Protocol Controller and Controllers.

Table 17. Commercial devices.

CAN Transceivers	CAN Protocol Controllers	Microcontrollers with CAN	Commercial Controllers
High-Speed	**High-Speed**	**Microchip**	**MotoHawk**
MCP2551 (HS)	MCP2515 (HS)	dsPIC33FJ256GP710	ECM-5554-112
TJA1041 (HS)	SJA1000 (HS)	PIC24HJ256GP610A	ECM-0555-080
TJA1050 (HS)		**Atmel**	ECM-0565-128
MAX3050 (HS)		AT90CAN128	**National Instruments**
MAX3057 (HS)		AT89C51CC03	PCI/PXI/PCMCIA
ATA6660 (HS)		**Freescale Semiconductor**	**Vector**
Fault Tolerant		MPC5554	CANlog4
TJA1054 (LS)		MPC555	CAN Case XL
MC33388 (LS)		MPC565	**Others**
Single Wire			CAN-AC2-PCI
TLE6255G (SW)			CANview USB

3.4 Use of Identifiers

In a CAN network, a technique sometimes used to easily identify a device is the division of the identifier into two fields: the device identifier and the message identifier. This can be useful when several similar devices are connected on the same network. Table 18 shows an example of a Tritium WaveSculptor [58] configuration using a 6-bit device identifier of "0b100 00 xxxx" and 5-bit message identifiers.

Table 18. Device and message identifiers.

Motor Controller TX ID: 0x400 + Message Identifier															
Device ID	Msg ID	ID	Device ID		Msg ID		Msg Name	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
			Bit 10	Bit 5	Bit 4	Bit 0										
32	0	0x400	0b100 000		0 0000		Id. Info.	8	Tritium ID				Serial Number			
32	1	0x401	0b100 000		0 0001		Status Info.	8	Limit Flags		Error flags		Act. Motor		Reserved	
32	2	0x402	0b100 000		0 0010		Bus Measur.	8	Bus Voltage				Bus Current			
32	3	0x403	0b100 000		0 0011		Velocity	8	Motor Velocity				Vehicle Velocity			
32	4	0x404	0b100 000		0 0100		Phase Current	8	Phase A Current				Phase B Current			

Some devices use the first byte of the data field as a sub-identifier, often referred to as a mode. When the identifier is not already divided in two, the first byte might also be called message identifier. This method usually requires more coding in the controller to decode the information. Table 19 details a battery management system (BMS) message having a fixed identifier of 600 and using sub-IDs.

Table 19. Sub-identifiers.

BMS TX ID: 0x600 + Sub-Identifier											
Sub ID	ID	Msg Name	DLC	Byte 0 Sub-ID	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0	0x600	Problem Flags	7	0x00	Error Cause		Active Error		Warning		
1	0x600	Master Measur.	8	0x01	Current		Voltage		SOC	Temp. Master	
2	0x600	Temperature	7	0x02	Average		Min		Max		

These techniques are not only useful to rapidly structure custom CAN buses, but are also standardized and used as a basis for communication protocols based on CAN.

3.5 CAN Message Definition

This section defines some common terms used to discuss CAN messages. A proper definition and understanding of these expressions is essential when creating custom CAN databases to interconnect controllers using different transmission (display) format. This is likely to be the case when converting a vehicle to electric drive using generic controllers from multiple suppliers.

3.5.1 Message and Signal

Each identifier is assigned a maximum of 8 bytes (data field) referred to as a message, regardless of the division and order of the data. A message is usually broken into signals, which are subdivisions of the data field, and therefore of the message [26][46][47]. The only rule in creating a signal is: it has to be made of consecutive bits. Its length varies from one bit to a few bytes, and can overlap 2 bytes of the data field even if it is a 2 bit signal. There are no rules concerning the length of signals; however, most controllers do not allow signals longer than 32 bits. If a custom microcontroller is used, for instance a 16 bit microcontroller, signals should not exceed 16 bits to avoid extra coding for storage and arithmetic operations. Table 20 shows examples of a few messages and their signals.

Table 20. Message and signal examples.

Data Field (Message)							
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Mode = 0	Info	Errors	Passives	Actives	Warnings	Limits	
Mode = 1	Batt. Status		Batt. Current		Batt. Voltage		
Mode = 2	State of Charge						
Mode = 3	Avg. Cell Temp.		Max. Cell Temp.		Min. Cell Temp.		

3.5.2 Mapping and Positioning of Signals

The bit order, or bit order of significance, within a byte is always increasing from right to left as specified by the CAN communication protocol [22]. In other words, the most significant bit (msb) is always at the leftmost position of a byte and the least

significant bit (lsb) at the rightmost position. The byte numbering, defined as the number assigned to each byte of the data field, is also fixed, but is increasing from left to right [45][46][47][48]. The first byte sent is defined as byte 0 and the last byte is defined as byte 7. These concepts are illustrated in Table 21.

Table 21. Byte numbering and bit order.

Data Field (Message)							
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
msb ... lsb	msb ... lsb	msb ... lsb	msb ... lsb	msb ... lsb	msb ... lsb	msb ... lsb	msb ... lsb

Knowing that the bit order of significance and byte numbering are fixed, to map a CAN message and to place its signals in the data field the following 4 pieces of information are required:

- Byte order of significance (endianness)
- Bit numbering
- Message progression (and overall bit numbering)
- Start bit of signals

It should be noted that the length of a signal is needed for its definition, but not to position it within the message.

3.5.2.1 Byte Order (Endianness)

When discussing byte order, 2 methods of sending the bytes of a message exist, little-endian and big-endian [47]:

- **Little-endian** means sending the least significant byte (LSB) first and the following bytes in increasing order of significance. (Intel)
- **Big-endian** means sending the most significant byte first (MSB) first and the following bytes in decreasing order of significance. (Motorola)

These expressions are often referred to as Intel and Motorola respectively, since Intel processors use the little-endian method and Motorola processors use the big-endian one. Table 22 explains this nomenclature.

Table 22. Byte ordering.

Endianness	Referred to as	Byte order							
		Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Little-endian	Intel	LSB	...	...	...	...	...	...	MSB
Big-endian	Motorola	MSB	...	...	...	...	...	...	LSB

In summary, the expression little-endian and big-endian define the byte ordering of data, LSB to MSB and MSB to LSB respectively. In other words, they inform on the ordering of significance in the data field of a CAN message, but give no information on the numbering associated to each bit stuffed within a byte of the data field.

3.5.2.2 Bit Numbering

The bit numbering, indexing of bits within a byte, can be defined by the following 2 methods [47][49]:

- Decreasing from left to right (Sawtooth)
- Decreasing from right to left (Sequential or Monotone)

Table 23. Bit numbering.

Bit Numbering	Referred to as	Byte							
		msb	...	...	...	...	...	...	lsb
		Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit
Decreasing from left to right	Sawtooth	7	6	5	4	3	2	1	0
Decreasing from right to left	Sequential (Monotone)	0	1	2	3	4	5	6	7

The bit numbering, also called bit counting, is independent of the bit order, which is referring to the msb to lsb bit progression within a byte and is fixed in CAN communication as explained previously.

3.5.2.3 Message Progression

Now that the byte ordering, bit ordering, byte numbering and bit numbering are defined, the bit numbering progression from one byte to another, called message

progression, must be defined. The message progression assigns a bit number from 0 to 63 to each bit in the message. The message progression can be [47][46]:

- Forward (numbering from the first byte)
- Backward (numbering from the last byte)

In other words, regardless of the bit numbering technique used, forward means the bit numbering begins from the first byte sent (bit 0 is in byte 0), and backward means the bit numbering begins from the last byte sent (bit 0 is in byte 7). In order to send and receive CAN data in a coherent manner, it is important for CAN users to know the bit number of each bit stuffed in a message, i.e. the overall bit numbering. This results in 4 possibilities, expanded from references [47][48][49]:

Table 24. Message progression.

Msg Progression	Bit Numbering	Data Field (Message)							
		Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Forward	Sawtooth	7...0	15...8	23...16	31...24	39...32	47...40	55...48	63...56
	Sequential	0...7	8...15	16...23	24...31	32...39	40...47	48...55	56...63
Backward	Sawtooth	63...56	55...48	47...40	39...32	31...24	23...16	15...8	7...0
	Sequential	56...63	48...55	40...47	32...39	24...31	16...23	8...15	0...7

Terminology used to define the bit numbering is discussed next. When representing 64 bits over the 8 bytes of the data field, the expression sawtooth seems to not only define the numbering within a byte, but also the message progression for the forward message progression. In a forward sawtooth message, the bit numbering increases from right to left while the message progression is from left to right creating discontinuities in the overall bit numbering. This expression kind of loses its sense for the big-endian byte order, since the bit numbering increases from right to left and the message progression is also from right to left creating no discontinuities as implied by the word “sawtooth”. The overall bit numbering of a backward sawtooth message has no discontinuities and looks like a forward “sequential” message starting the counting from the end of the message.

Being inconsistent with the continuous and discontinuous overall bit numbering, the bit numbering expressions are inadequate to visualize the overall bit numbering. This is one of the reasons why it is essential to specify the message progression in a message to avoid confusion from the user when defining the overall bit numbering.

3.5.2.4 Start Bit of Signals

As shown in Table 20, signals are variables, or pieces of data, contained by a message. They are the actual information, i.e. they can represent a voltage, a current, a state of charge, a temperature, a speed, a status, errors, warnings, etc. The start bit of a signal is usually its lsb when discussing CAN bus, however this is not always the case. Therefore, when positioning a signal in a message, the start bit used must be specified as the:

- lsb of the signal
- msb of the signal

If a signal is defined using the lsb as its start bit, the bit-wise progression is towards the left, but the byte-wise progression depends on the byte order.

3.5.2.5 Display Formats

This section details 6 display formats for messages using the terminology explained in the above sections. In an attempt to further clarify these display formats, a column describing visually the bit-wise and byte-wise progression is added to Table 25. It summarizes the 6 common display formats in CAN communication, expanded from [47][45][49].

Table 25. Display formats.

Display Format	Byte Order	Bit Numbering	Msg Progression	Start Bit of signals	Bit progression (from the start bit to full length)
Intel Standard	Little-endian	Sawtooth	Forward	lsb	bit-wise: to the left byte-wise: to the right
Intel Sequential	Little-endian	Sequential	Forward	lsb	bit-wise: to the left byte-wise: to the right
Motorola Forward lsb	Big-endian	Sawtooth	Forward	lsb	bit-wise: to the left byte-wise: to the left
Motorola Forward msb	Big-endian	Sawtooth	Forward	msb	bit-wise: to the right byte-wise: to the right
Motorola Backward	Big-endian	Sawtooth	Backward	lsb	bit-wise: to the left byte-wise: to the left
Motorola Sequential	Big-endian	Sequential	Forward	msb	bit-wise: to the right byte-wise: to the right

Table 26 is a compact form of Table 25 using the 3 main CAN display formats.

Table 26. Compact form of the 3 main CAN display formats.

Display Format	Byte Order	Bit Numbering	Msg Progression	Start Bit of signals	Bit progression (from the start bit to full length)
Intel Standard	Little-endian	Decreasing from left to right	Forward	lsb	bit-wise: to the left byte-wise: to the right
Motorola Forward lsb	Big-endian				bit-wise: to the left byte-wise: to the left
Motorola Backward			Backward		

3.5.2.6 Examples

Representation in the 3 main display formats

In this example, adapted from reference [45], a signal is represented in the 3 main display formats and is defined by its start bit and length. The data length code (DLC) of the message must be greater or equal to 7 and it is assumed that the start bit is the least significant bit of the signal in each of display formats.

Specifications: Start bit (lsb) = 11, Length = 9 bits.

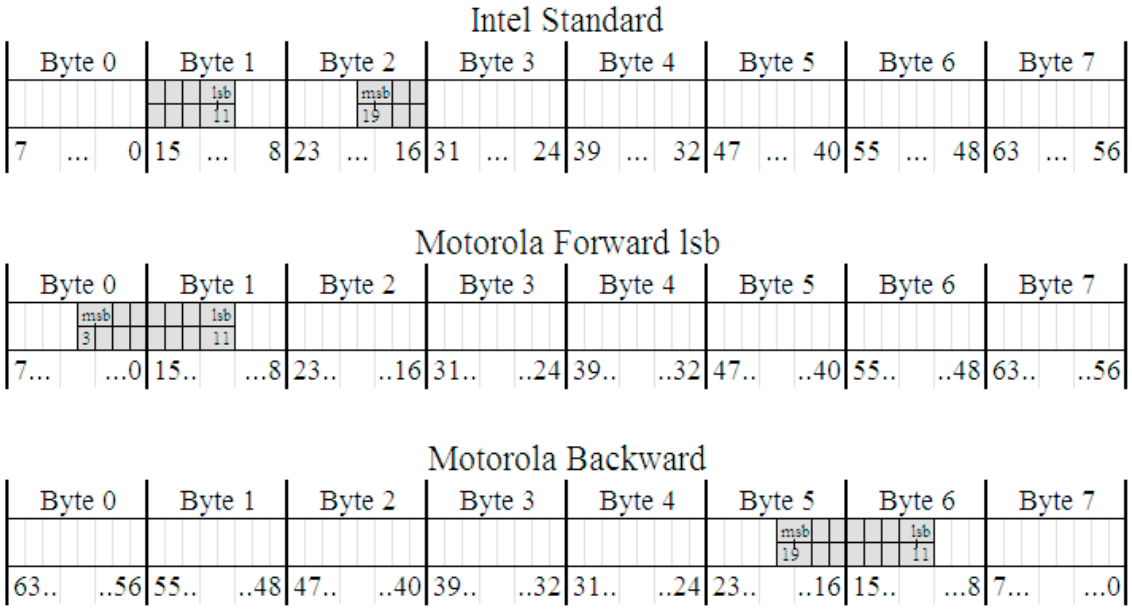


Figure 24. Representation of a signal in the 3 main CAN display formats.

This means that three controllers using the same signal specifications, but having three different display formats will send their signal in a completely different space of the data field.

Conversion from Intel Standard

As shown by the previous example, in order to decode a signal, it is essential to know its display format and apply an appropriate conversion when reading it with a controller using a different format. This example shows how a signal sent from controller using an Intel Standard display format is read by two controllers, one using a Motorola Forward lsb display format and the other Motorola Backward.

Specifications for Intel Standard: Start bit (lsb) = 11, Length = 9 bits.

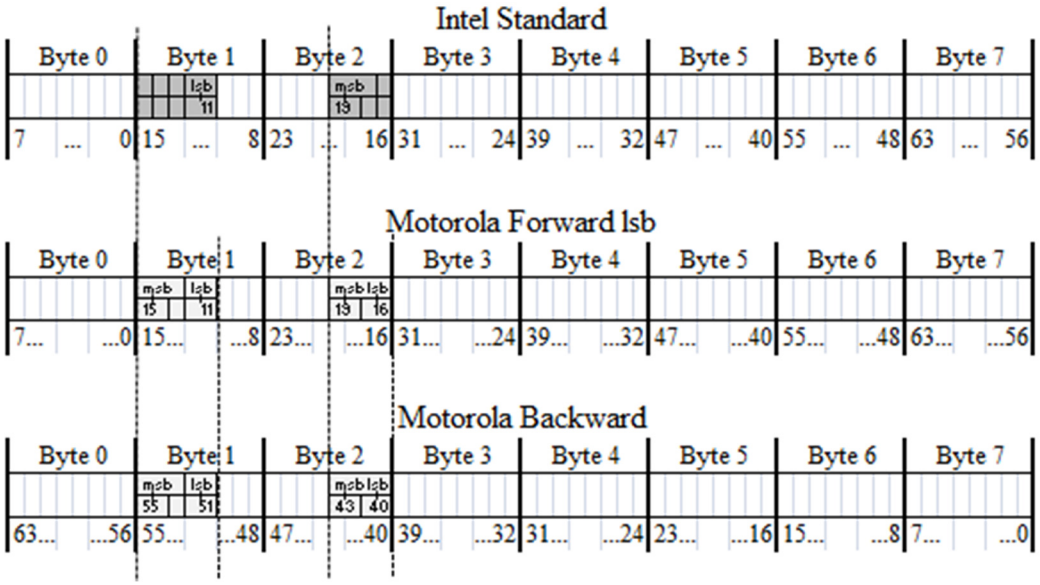


Figure 25. Conversion from Intel Standard.

Figure 25 shows how the signal in the Intel Standard display format is represented in the other two formats. In both cases, since the byte order is big-endian instead of little-endian, the signal must be rebuilt by concatenation of two signals. According to the little-endian byte ordering, the most significant part of the signal is in Byte 2 and the least significant part in Byte 1. Therefore, the concatenation is: signal in Byte 2 + signal in Byte 1. In the case of the Motorola Forward lsb, the start bit does not need to be converted, since it is using a forward message progression like the Intel Standard does. However, a conversion needs to be applied to the start bit for Motorola Backward, because the message progression used is backward. The graphical representation shown in Figure 25 is an easy method to find the new start bit in a different display format.

Specifications for Motorola Forward lsb: Start bit (lsb) = 11, Length = 9 bits
Specifications for Motorola Backward: Start bit (lsb) = 51, Length = 9 bits

For the most significant part of the signal, the start bit is always the first bit of the following byte and the length is the remainder of the total length not used by the least significant part of the signal. Thus, a detailed expression of the specifications for each display format is:

Specifications for Motorola Forward lsb:

[Start bit (lsb) = 16, Length = 4 bits] + [Start bit (lsb) = 11, Length = 5 bits]

Specifications for Motorola Backward:

[Start bit (lsb) = 40, Length = 4 bits] + [Start bit (lsb) = 51, Length = 5 bits]

The graphical representation of the conversion from Motorola Forward lsb and from Motorola Backward can be found in Appendix F.

3.5.3 Message and Signal Definitions

The difference between messages and signals was established in Section 3.5.1. This section describes in detail both messages and signals, and how to filter them in order to receive and send CAN messages.

3.5.3.1 Message Definition

A CAN message is defined by the following characteristics:

- Message name
- ID
- ID mask
- DLC (payload size)
- Payload filter (payload value)
- Payload mask
- Periodic interval (rate in ms)

The message name is the name of the message associated with a data frame or remote frame meeting the filtering characteristics: ID, ID mask, payload filter and payload mask. The data length code (DLC) can be between 0 and 8, and describes how many bytes of the data field are used for a given message. Messages can be sent when triggered by an event, such as a remote frame, or can be transmitted periodically. The periodic interval value is commonly expressed in milliseconds.

Filtering

CAN communication follows a producer-consumer model where all the nodes are masters. Messages give no information on the destination (consumer), only on the source (producer) [30]. Therefore, each node must read all the messages on the bus. When a message is sent (produced) by a node, it is up to the other nodes on the network to decide whether to receive it (consume) or to ignore it. In other words, nodes need to sort, or filter, the messages they are interested in receiving.

When using microcontrollers having CAN integrated features, also called CAN chips, filtering is done by hardware using CAN buffers and by software using a custom dispatcher. In microcontrollers without specific CAN buffers, sorting is accomplished by software only and is less efficient.

The sorting process has the two following filtering stages:

- ID filtering (using the ID and the ID mask)
- Payload filtering (using the payload filter and the payload mask)

In the first case, messages are filtered according to their identifier. The filter is the identifier mask defining which bits of the identifier to care about. The second type of filtering is used when a portion of the data field is used as a sub-identifier. Messages are sorted using a mask on the payload, named payload mask, indicating which bits of the data field to care about. Usually, the payload mask defines which bits of the data field to use as sub-ID allowing the demultiplexing of signals in messages using modes, as first explained in Section 3.4. The payload filter is more likely the sub-ID of a message, but

could be a specific payload value. It should be noted that the ID, is to the ID mask, what the payload filter is to the payload mask. Since the payload filtering is used to filter the sub-IDs of a specific ID, the ID filtering is processed first. When sub-IDs are not used, the payload mask is an all-pass filter. Table 27 presents an example of an ID filtering process while Table 28 details a payload filtering example using sub-IDs, based on reference [26].

Table 27. ID filtering example.

Filtering Configuration	Binary Value	Results
ID = 400	0b100 0000 0000	
ID mask = 7E0	0b111 1110 0000	1 = care, 0 = don't care
ID acceptance filter	0b100 000x xxxx	x = any value allowed
Incoming Message		
ID = 403	0b100 0000 0011	
ID acceptance filter	0b100 000x xxxx	Message accepted
ID = 404	0b100 0000 0100	
ID acceptance filter	0b100 000x xxxx	Message accepted
ID = 600	0b110 0000 0000	
ID acceptance filter	0b100 000x xxxx	Message ignored

Table 28. Payload filtering example.

Filtering Configuration	Binary Value	Results
ID = 600	0b110 0000 0000	
ID mask = 7FF	0b111 1111 1111	1 = care, 0 = don't care
ID acceptance filter	0b110 0000 0000	Only ID = 600 accepted
Payload filter = 01 00...00	0b00000001 00000000 ... 00000000	
Payload mask = FF 00...00	0b11111111 00000000 ... 00000000	1 = care, 0 = don't care
Payload acceptance filter	0b00000001 xxxxxxxx ... xxxxxxxx	x = any value allowed
		Sub-ID = 01 in byte 0
Incoming Message		
ID = 400	0b100 0000 0000	
ID acceptance filter	0b110 0000 0000	Message ignored
Payload = 01 00 ... 00	-	
Payload acceptance filter	-	-
ID = 600	0b110 0000 0000	
ID acceptance filter	0b110 0000 0000	ID accepted
Payload = 03 00 ... 00	0b00000011 00000000 ... 00000000	
Payload acceptance filter	0b00000001 xxxxxxxx ... xxxxxxxx	Message ignored
ID = 600	0b110 0000 0000	
ID acceptance filter	0b110 0000 0000	ID accepted
Payload = 01 00 ... 00	0b00000001 00000000 ... 00000000	
Payload acceptance filter	0b00000001 xxxxxxxx ... xxxxxxxx	Message accepted

3.5.3.2 Signal Definition

A CAN signal is described by the following characteristics [28][47][48]:

- Signal name
- Start bit
- Length (in bits)
- Byte order (little-endian or big-endian)
- Data type (signed, unsigned, float, double, boolean, etc.)
- Units (mV, A, ºC, km/h, etc.)
- Scale (also called factor or gain)
- Offset
- Range (minimum and maximum)

Scaling

Signals are often converted before being transmitted over the CAN network. The purpose of the scale and offset of a signal is to convert the raw value of a signal to its engineering value. The physical value, or unit value, is the signal value having a scientific or physical meaning by being associated to a unit. The raw value, or bus value, is the signal value transmitted over the CAN bus. The scale and offset are adjusting the raw value of the signal to obtain its physical value expressed in the desired unit. The following basic linear equation is used to express the relation between the physical value and raw value, based on [46][48]:

$$physical\ value = (scale)(raw\ value) + offset$$
$$y = mx + b$$

3.6 Bus Configurations

The basic configuration of a CAN bus is the bus topology. Figure 26 illustrates this concept using devices found on an electric vehicle CAN network.

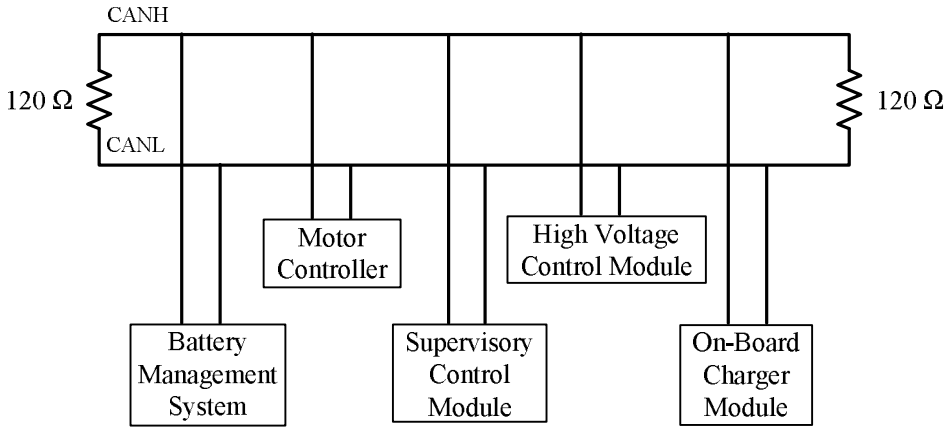


Figure 26. Bus topology.

In this topology, a main bus of 2 wires exists and devices are connected to it in parallel. The length of the wires connecting each device to the bus can vary. While this technique works well for small networks, electrical noise can be experienced on large networks, since every single branch connected to the main bus acts as an antenna.

In order to reduce the electrical noise, the two 120 Ω termination resistors should be placed next to the 2 farthest controllers leaving only short leads between the resistors and the controllers, thus eliminating the two branches acting as antennas. This method is still considered a star topology. It retains simplicity and flexibility for the addition or removal of devices on the bus. Figure 27 gives an example of such a topology.

A bus topology can be defined as nodes connected by passive links through a single cable allowing transmission in both directions [40]. Hubs are generally used on the main bus to easily add and remove devices. They can be passive or active. Since the length of a CAN bus is short in automotive applications, hubs do not need to regenerate or amplify the bus signal, therefore only passive hubs will be considered. Active hubs require power and act as repeaters.

A star topology is a network where all the nodes are connected to a central one [40]. The addition of a hub on a bus configuration creates a node to which several other nodes are connected. This distribution node is therefore a centralized point for these other nodes modifying the architecture of the network. When a hub is inserted to a bus structure, the network architecture becomes a combination of the bus and star topologies. Networks using more than one topology are called hybrid networks. A CAN network using one or more hubs, as shown on Figure 28, is considered to have a hybrid star-bus topology with a bus as the network backbone. This topology is sometime referred to as a cascaded star topology. A disadvantage of this technique is the addition of more branches, and therefore more noise in the network.

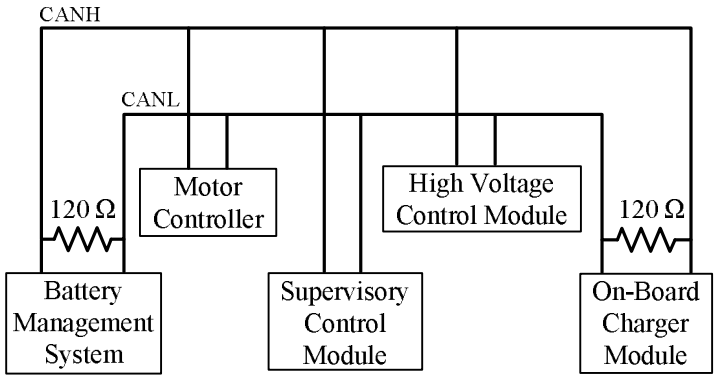


Figure 27. Bus topology minimizing noise.

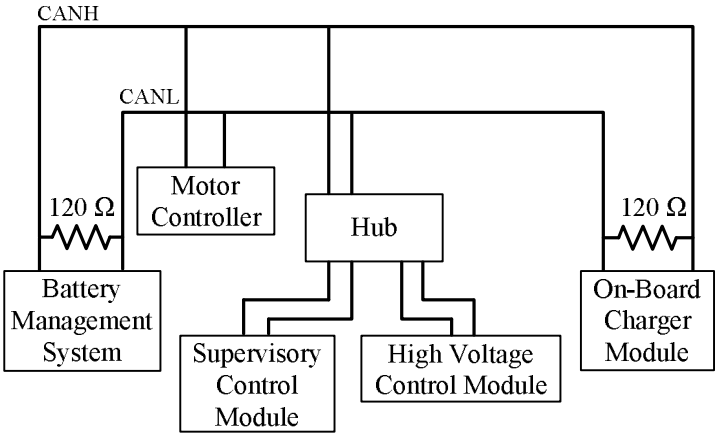
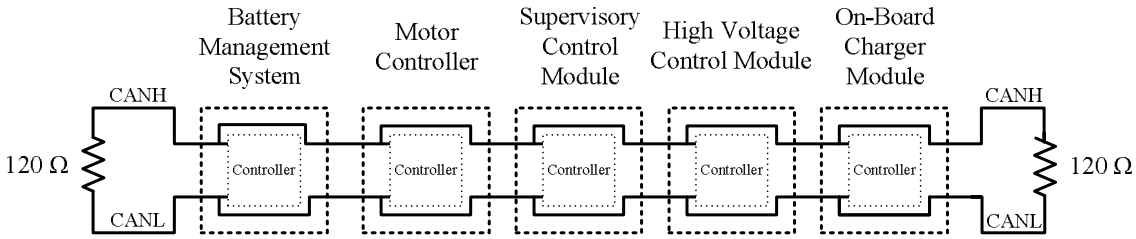


Figure 28. Hybrid star-bus topology.

To reduce the electrical noise to a minimum, another variation of the bus topology known as a daisy chain can be used. This method brings the main bus directly to every device on the CAN bus removing the branches, and therefore the antenna effect induced by them. In this configuration, the devices have an internal parallel connection design and each device acts as a hub for the following device in the chain. The expression daisy chain topology can be viewed as a bus topology with short studs. However, the expression is appropriate regarding the method used to create the wiring harness for the configuration and has an influence on the hardware of the devices connected to the bus. The absence of hubs in this configuration makes it more difficult to add and remove devices and makes the overall design more complex. Figure 29 illustrates this topology.



Note: The resistors can be internally integrated to the farthest devices' circuits.

Figure 29. Daisy chain with short studs.

Another method of creating a daisy chain is to simply twist two wires together and screw them in place into a single slot of a connector as shown on Figure 30. However, these types of connectors are not found in vehicles for reliability reasons.

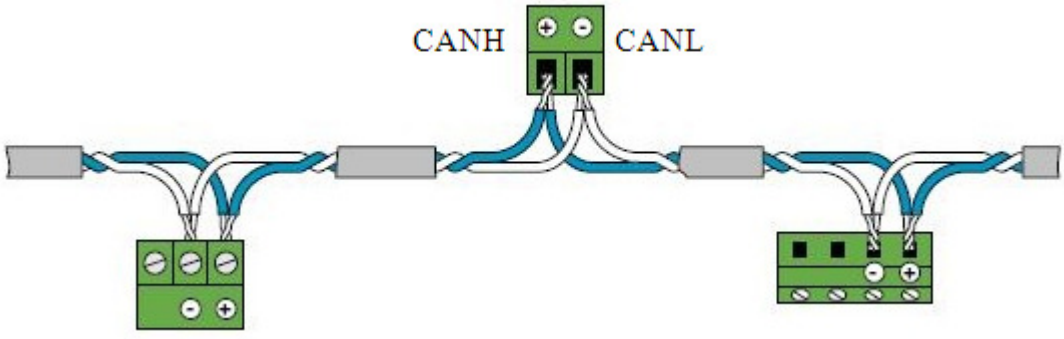


Figure 30. Daisy chain with twisted wires in the connector [52].

3.7 Commercial Devices

Some commercial devices are listed under "commercial controllers" in Table 17 where CAN transceivers, CAN protocol controllers, microcontroller with CAN and commercial controllers are enumerated. This section focuses not only on commercial controllers, but also on commercial devices. For the purpose of this document, commercial controllers are considered commercial devices, but the opposite is not true. A commercial device is the hardware portion of a CAN tool and requires a software application to interface and interact with the user. It is important to distinguish commercial devices from their software applications, since some sophisticated software can be used with different hardware while other less elaborate software is specific to a device.

Commercial devices for CAN can be categorized in 4 major groups:

- Scopes
- Loggers
- Controllers
- Hardware In the Loop (HIL)

Scopes are basically CAN to USB gateways allowing one to visualize the identifier and raw 8 bytes of data in a CAN message. A logger's purpose is to log traffic on a CAN bus. Some can be programmed and operate without constant computer interaction using embedded memory, in other words they are standalone devices with memory; whereas other devices have no memory but still allow logging through a software application by using the computer's memory. As discussed in Section 3.3, controllers are programmable microcontrollers with circuitry allowing communication on one or several CAN buses. Hardware in the loop simulation devices are machines able to interact in real-time with an external system and update the measurements in a model running on a computer, or on the device itself, to test real-time embedded systems [59].

The list of commercial devices available on the market is extensive. Some of the most common ones are summarised in Table 29 with their features, software and vendor, information is extracted from [64][65][66][67][68][69][70][71][72].

Table 29. Commercial devices.

Commercial Devices	Features					Software Applications (not exclusively)	Vendor
	Scope	Standalone Logger	Not Standalone Logger	Controller	HIL		
CANView USB	x		x			RM CAN-Device Monitor	RM
Leaf Light HS	x		x			Kvaser CanKing	Kvaser
CANcaseXL	x		x			Vector tools: CANoe	Vector
CANcardXL	x	x	x			Vector tools: CANalyser	Vector
neoVI	x	x	x	x*		Vehicle Spy	Intrepid
CANLog 4		x		x*		G.I.N. Configuration Program	Vector
CANcaseXL Log	x	x	x			Vector tools: CANoe	Vector
MotoTron	x			x		MotoHawk: Simulink, MotoTune	Woodward
Mico-Autobox	x		x	x		Control Desk, Simulink	dSPACE
PCI/PXI/PCMCIA	x		x		x	Labview: Simulink, NI VeriStand	NI
CAN-AC2-PCI	x		x		x	Matlab Simulink: xPC Target	Softing
dSPACE	x		x		x	Control Desk	dSPACE

*Limited capabilities.

3.8 Applications for Vehicles

So far, this chapter has covered the theory involved in the CAN bus protocol, the hardware required to support it, its bus configurations, how to define CAN messages in a non-ambiguous manner and introduced a few devices supporting CAN. Representing a rugged serial bus, CAN has a wide range of applications from providing a network in building and factory automation, to connecting controllers in ships, to its use in aircrafts for navigation systems and sensors, to elevators, forklifts and numerous other applications in industrial controls [24][29]. CAN bus is also used as an in-vehicle

communication network by most vehicle manufacturers in North America, Europe and Asia [24]. In this section, the focus is primarily on CAN bus as an intra-vehicle communication environment. In addition, Appendix H compares CAN to LIN, a cheaper communication system for vehicles.

3.8.1 Types of Applications

There are usually multiple CAN buses in vehicles. Up to 5 CAN buses have been seen, for example 4 high-speed buses and a single wire one. The types of applications differ whether it is a single-wire bus or a high-speed one.

3.8.1.1 Low Speed Applications

As defined in Table 14, 33.3 kb/s is the speed of a single-wire CAN communication system. Transmission rate is the primary limit of single-wire buses. Fault tolerant buses and high-speed buses configured with a low baud rate can also be used for non-time critical purposes; however, requiring only one wire and being slightly cheaper to implement than high-speed and fault tolerant buses, single-wire buses are generally used for these applications. Table 30 provides a list of examples of low speed applications, expanded from reference [24].

Table 30. Low speed applications.

Dashboard	Instrument cluster panel
	Cabin temperature controls
	Light sensor
	Steering wheel electronics
	Entertainment controls
	Infotainment controls
	Air Conditioning controls
Door control	Mirror
	Window
	Door locks
Seat control	Seat position
	Seat heater
	Occupancy sensor
Roof control	Interior lights
	Visor lights
	Moon roof
Engine control	Fans
	Non-time critical sensors

3.8.1.2 High Speed Applications

For applications requiring near real-time communications, high-speed CAN buses are used. With the complexity of car controls constantly growing, it is not infrequent to find more than one high-speed bus in a car. Examples of applications using high-speed CAN buses are enumerated in Table 31, based on reference [24].

Table 31. High speed applications.

ECU programming
Diagnostic interface
Engine management
Electric motor controller
High voltage battery management
Adaptive Brake System (ABS)
Body controller
Accident avoidance system
Fuel system

3.8.1.3 Diagnostic Interface and ECU Programming

There are 2 other key applications of CAN in vehicles [24]:

- Diagnostic interface
- ECU programming

Most electronic control units (ECUs) save diagnostic information that can be sent on a CAN bus as diagnostic trouble codes (DTCs) to other ECUs and a diagnostic tester. The diagnostic tester generally connects to the vehicle's On-Board Diagnostic II (OBD II) port to read DTCs present on the CAN network in order to make a diagnostic.

Calibration of variables in ECUs and software updates are inevitable when developing a vehicle. ECUs can usually be programmed through a CAN bus to simplify these tasks, since not every ECU of a vehicle is easily accessible and/or removable. This programming method has the advantage of requiring no modifications to the vehicle's hardware and can generally be done using the OBD II port.

3.8.1.4 Gateways between CAN buses

As mentioned earlier in this section, it is not uncommon to find multiple CAN buses interconnecting electronic control units (ECUs) within a vehicle. The need for having multiple CAN environments comes from the constantly increasing number of ECUs requiring more information, thus using more bandwidth [24]. When the bandwidth of a CAN network is saturated, it is common practice to add a new CAN bus to allow the addition of new controllers and features. These new ECUs present on the new CAN environment might need information transmitted on another CAN bus. This is why, most ECUs have multiple CAN ports, i.e. 2 to 4, and some ECUs are used as gateways between different CAN buses. Gateways are also required to interconnect ECUs from CAN buses using different high speed frequencies or to interconnect ECUs from a single-wire bus to a high-speed one.

3.9 Summary

This chapter provided an overview on CAN protocol fundamental theory; a description of the hardware required to create a CAN environment; lists of devices supporting CAN available on the market; a rigorous definition of the nomenclature used to accurately define CAN messages and CAN signals. Different CAN bus configurations; as well as different techniques for using identifiers; and numerous automotive applications are covered. In other words, it was shown how the CAN communication protocol allows ECUs to interact with one another. The next chapter probes deeper into these interactions by explaining the control strategy managing the powertrain ECUs selected by the UOIT EcoCAR team, now implemented in the EV prototype.

Chapter 4

Electrical and Control System Integration Strategy

4.1 Introduction

The in-vehicle communication technology linking on-board ECUs through a robust network was covered in the previous chapter. The control strategy for integrating these additional components to the baseline production vehicle is detailed in the current chapter. Removal and addition of ECUs is discussed first, along with their layout on the CAN communication network. An overview of the general control strategy is then presented, followed by the control details of the main powertrain modules. Additionally, the safety control strategy is explained, followed by a description of the on-board diagnostics system.

4.2 Conversion Methodology

The EcoCAR challenge suggested competing teams follow the GM validation process diagram shown in Figure 31 [8].

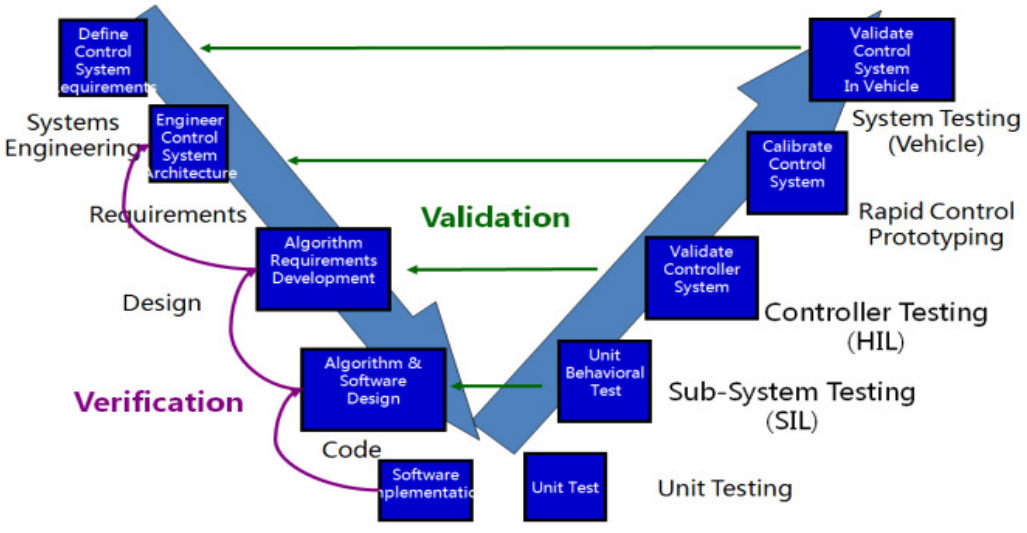


Figure 31. Validation Process Diagram [8].

Even though the general approach of the UOIT team can be explained using this validation model, the conversion methodology is also accurately represented by Figure 32.

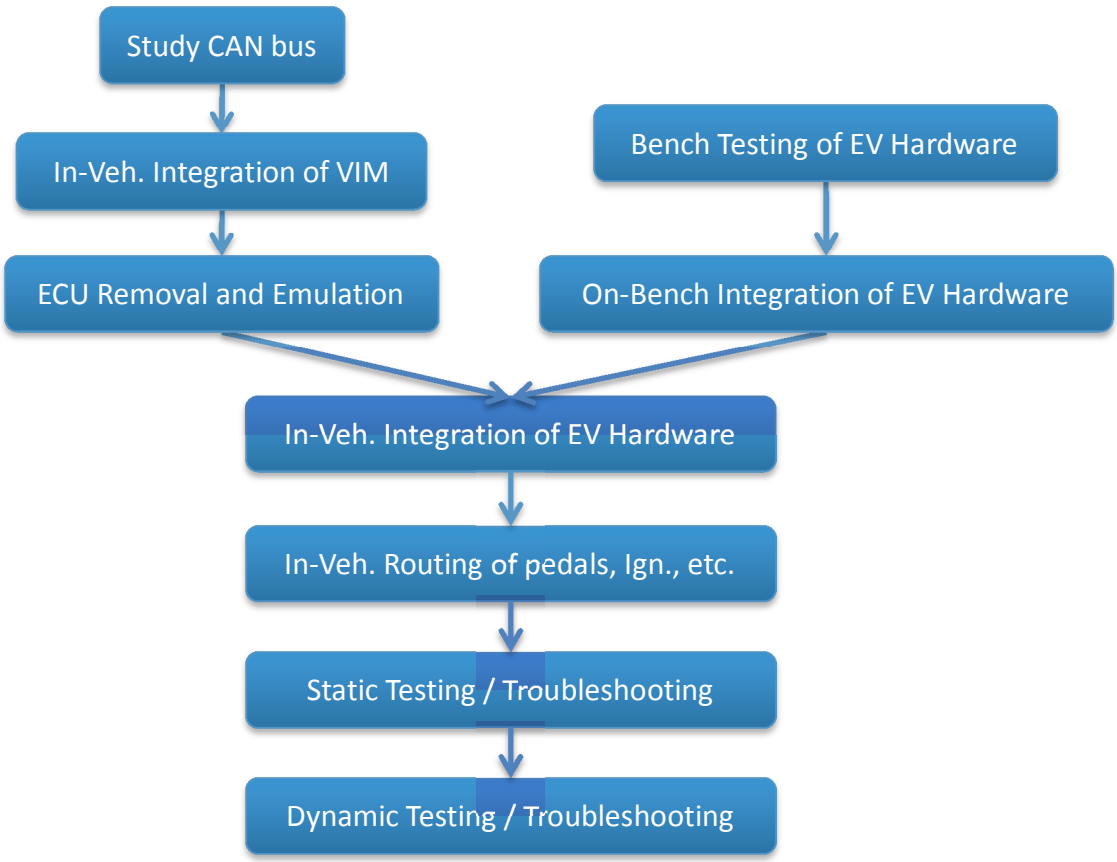


Figure 32. Conversion Methodology.

During the conversion process, 2 phases took place in parallel, i.e. one on the original vehicle and one on the test bench, and were followed by the vehicle integration phase.

In the vehicle phase, the CAN buses were studied message by message, signal by signal and even bit by bit to learn the communication pattern between the original ECUs. Then, the author started communicating with the car using the VIM, and finally removed and emulated the ECUs.

In the test bench phase, the author tested and calibrated all the powertrain components, such as the battery, battery management system, motor, motor controller, chargers and vehicle integration module, first individually, then integrated.

When both phases were completed, the integrated powertrain on the test bench was taken and installed as is in the car. The test bench custom control panel and terminal blocks fixed on a piece of wood were first used, but quickly the pedals and other controls were re-routed to control the car from the driver seat. Static and dynamic testing followed. This methodology resulted in an efficient vehicle conversion, while minimizing the risk of controls errors. The outcome of this controls conversion process is discussed in more details in Section 6.2.1.

4.3 Electronic Control Units

4.3.1 Original ECUs

The network architecture of the hybrid 2009 Saturn VUE consists of 3 controller area networks (CAN buses) using the higher-layer protocol called General Motors Local Area Network (GMLAN). This protocol is also compliant with 2 SAE specifications for CAN buses applications in vehicles: SAE J2284-3 (high-speed CAN for vehicle applications at 500kb/s) and SAE J2411 (single-wire CAN for vehicle applications at 33kb/s). A brief description of the 3 CAN buses present on the vehicle follows:

- **HS GMLAN**

  High-speed GMLAN is the main dual wire CAN bus on GM's vehicles and provides a communication network for most chassis and powertrain modules (i.e. ECM, TPIM, TCM, EBCM, EPS, BCM, CGM, OnStar, BPCM, FSCM, DLC). Data is transmitted at 500 kb/s and is compliant with the SAE J2284-3 standard.

- **PTE GMLAN**

  GMLAN powertrain expansion is also a dual wire CAN bus based on the SAE J2284-3 specification, thus having a baud rate of 500 kb/s. It is a secondary high-speed network to accommodate the data flow increase caused by the components of the hybrid powertrain (i.e. ECM, APM, TPIM, EBCM, CGM, CIS, DLC).

- **SW GMLAN**

  Single-wire GMLAN, also called low-speed, utilizes a transmission rate of 33 kb/s and is compliant with the SAE J2411 standard. ECUs not requiring near real-time communication, generally chassis modules (i.e. IRC/NAV, IPC, CGM, VTD, RFA, ECC, BCM, AOS, SDM, ROS, SDARS, OnStar, ACCM, DLC), are found on this network.

Table 32 enumerates the ECUs present on the high-speed CAN buses and summarizes their main functions, whereas Table 33 introduces those on the single-wire one.

Table 32. List of ECUs on HS & PTE.

ECUs on High-Speed (HS) & Powertrain Expansion (PTE)					
Acronym	**HS**	**PTE**	**SW**	**Name**	**Description**
APM		x		Auxiliary Power Module	Manages the 12V auxiliaries
BPCM	x			Battery Pack Control Module	Controls the lithium battery
BCM	x	x	x	Body Control Module	Main controller
CIS		x		Chassis Inertial Sensor	Inertial/Yaw Sensor
CGM	x	x	x	Communication Gateway Module	Gateway between HS, PTE and SW buses
OBD-II port (DLC, ALDL)	x	x	x	On-board diagnostics II (Data Link Connector, Assembly Line Diagnostic Link)	Not an ECU. SAE J1962 defines the connector used for the OBD-II interface. Give access to all CAN buses for vehicle diagnostics.
EBCM	x	x		Electronic Brake Control Module	Automatic Braking System, Electronic Stability Control
ECM	x	x	x	Engine Control Module	Control throttle position, spark plug and fuel injection
EPS	x			Electric Power Steering	Amplify steering effort according to driving conditions
FSCM	x			Fuel System Control Module	Control fuel pump and fuel level sensor
OnStar	x		x	OnStar system	Communication for security, remote diagnostics, hands free calling and navigation
TCM	x			Transmission Control Module	Automatic transmission
TPIM • HCP • MCPA • MCPB	x x x x	x x x x		Traction Power Inverter Module • Hybrid Control Processor • Motor Control Processor A • Motor Control Processor B	High voltage and motor controller

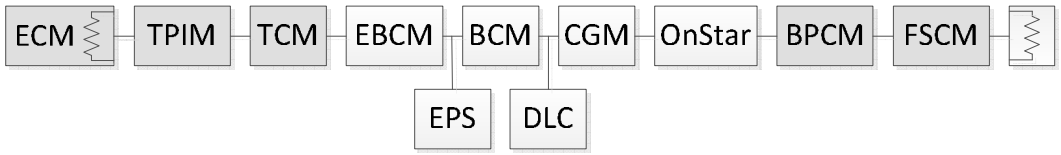
*Dark grayed ECUs were removed from the vehicle, see Section 4.3.2 and Table 35.

Table 33. List of ECUs on GMLAN SW.

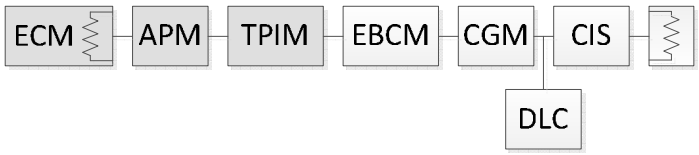
ECUs on GMLAN Single-Wire (SW)		
Acronym	**Name**	**Description**
BCM	Body Control Module	Main controller
CGM	Communication Gateway Module	Gateway between HS, PTE and SW buses
OBD-II port	On-board diagnostics II	Not an ECU. SAE J1962 defines the connector used for the OBD-II interface. Give access to all CAN buses for vehicle diagnostics
OnStar	OnStar system	Communication for security, remote diagnostics, hands free calling and navigation
ACCM	Air Conditioning Compressor Module	Control air conditioning
AOS	Automatic Occupant Sensor	Occupant seat sensors
ECC	Electronic Climate Control	Cabin climate control panel
IPC	Instrument Panel Cluster	Control the dashboard
IRC/ NAV	Integrated Radio Chassis / Navigation	Radio and GPS
RFA	Remote Function Actuator	Remote control door lock receiver
ROS	Roll Over Sensor	Roll over detection for airbag deployment
SDARS	Satellite Digital Audio Radio Service	XM Radio
SDM	Sensing and Diagnostic Module	
VTD	Vehicle Theft Deterrent	

The physical interconnections between the ECUs described in Table 32 and Table 33 are shown on a layout diagram in Figure 33. The ECUs on the 2 high-speed buses utilize a hybrid daisy chain topology, this occurs when the majority of components are daisy chained together with a few connected in a star topology. Nevertheless, the single-wire bus is split in 2 buses using a star topology, which are daisy chained together by the body control module (BCM). HS and PTE GMLAN are both terminated by a 120 Ω resistor at

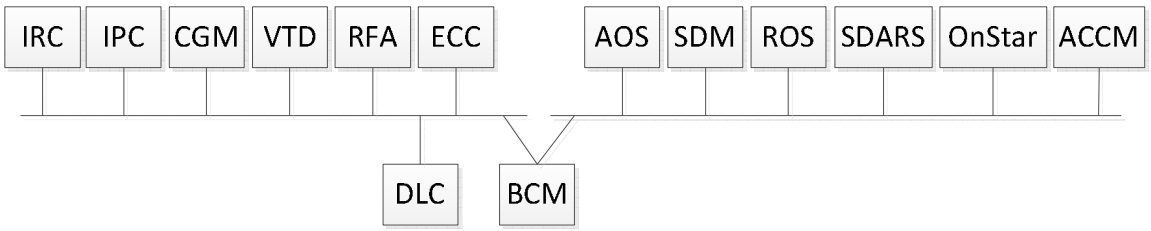
both ends of their CAN bus. The ECM terminates both networks at one end and independent resistors of the same value adjust the impedance at the other end. The 120 Ω resistor for the HS network is located near the fuel tank and the one for the PTE can be found underneath the vehicle at the front passenger seat position.



(a) HS GMLAN



(b) PTE GMLAN



(c) SW GMLAN

Figure 33. Stock CAN buses: (a) GMLAN HS, (b) GMLAN PTE, (c) GMLAN SW.

*Dark grayed ECUs were removed from the vehicle, see Section 4.3.2 and Table 35.

4.3.2 ECU changes

Among the ECUs present in the Saturn VUE some became obsolete after the conversion, others were mounted on components being removed. For example, the TPIM unit contained the HCP, MCPA and MCPB, thus to remove the TPIM module three ECUs had to be removed at once. It is important to understand the function of each ECU on the baseline vehicle to determine which ones are not required by an electric powertrain conversion. An understanding of the additional modules is also fundamental when converting a vehicle. This sub-section compares the original ECUs to those

replacing them in order to explain the identification process of which ECUs to remove. The stock ECUs and their related hardware are compared to their replacements in Table 34.

Table 34. Hardware Changes.

Hardware Changes	
Saturn VUE	**UOIT EcoCAR**
Engine and Electric Motor	Electric Motor with integrated gear box
ECM	TIM
TCM	
TPIM	VIM (HVCM)
Fuel tank	Li-Ion Battery
NiMH Battery	
BPCM	BMS
FSCM	
	VIM (SCM)
Alternator	2 DC/DCs
APM	
	Charger
	GFI
	Datalogger
	ICS
	Chiller
	Heater

The baseline is a 2-mode hybrid, this constitutes an internal combustion engine and 2 electric motors which are mechanically coupled through a planetary gear box controlled by the TCM. As explained in Section 2.7, an induction motor controlled by a power inverter controller propels the converted vehicle. This not only simplifies the powertrain, but also renders the stock components obsolete for the converted vehicle. Therefore, the entire powertrain was removed along with its controllers, i.e. ECM, TCM and TPIM.

For energy storage, the hybrid utilized a 63.22 L fuel tank and a 2.3 kWh NiMH battery pack. These are managed by a fuel system control module (FSCM) and a battery pack control module (BPCM), respectively. Both systems were replaced by a 83.5 kWh lithium polymer battery pack and a battery management system (BMS). A vehicle integration module (VIM), including a high voltage control module (HVCM) and a supervisory control module (SCM), was also added to fulfill the tasks executed by the TPIM and BPCM not covered by the motor controller and BMS, respectively. The HVCM controls the high voltage contactors, whereas the VIM manages the vehicle integration.

In a hybrid, the battery pack is charged by a combination of regenerative braking and engine power. An on-board charger is added as the primary charging device for the electric vehicle. Regenerative braking is available through the traction inverter module (TIM).

A typical vehicle powers its ancillaries with a 12V lead acid battery charged by an alternator coupled to the engine. A DC to DC converter (DC/DC), reducing the high voltage from the traction battery to 12V, is utilized to charge a 12V battery and power the accessories. As a result, the auxiliary power module (APM) integral to the TPIM block was also removed. A second DC/DC converter is added to satisfy the power requirements of the battery thermal management system, which is only enabled at temperature extremes. On the baseline vehicle, the battery pack has a smaller liquid thermal management system consuming less energy, thus avoiding the need for extra power on the 12V system.

A datalogger is added to record data from the HS and UOIT buses. Even though the dashboard cluster screen is replaced by a custom one managed by a proprietary controller, the instrument panel cluster (IPC) is still needed to manage other functionalities of the dashboard. Although it was not necessary for the conversion, the baseline radio was replaced by a combination of navigation and rear seat entertainment system.

All the removed ECUs, i.e. APM, BPCM, ECM, FSCM, TCM, TPIM (HCP, MCPA, MCPB), were on the HS and PTE GMLAN. The SW GMLAN remained unmodified. Table 35 lists the removed and remaining stock ECUs.

Table 35. Removed and Remaining ECUs.

Removed ECUs		Remaining ECUs	
GMLAN HS	**GMLAN PTE**	**GMLAN HS**	**GMLAN PTE**
BPCM	APM	BCM	BCM
ECM*	ECM*	CGM	CIS
FSCM	TPIM	EBCM	CGM
TCM		EPS	EBCM
TPIM (HCP, MCPA&B)		OBD-II port	OBD-II port
		OnStar	

*Including the 120Ω termination resistor for each bus.

Table 36 gives an overview of the modifications made related to the ECU architecture.

Table 36. Main ECUs After Conversion.

Original		Modified	
Removed ECUs	**Retained ECUs**	**Added ECUs**	**Added Components**
APM	BCM	BMS	for Li-Ion battery
BPCM	CGM	Motor Controller (TIM)	for S10-EV induction motor
ECM	EBCM	Charger (OBCM)	J1772 power inlet
FSCM	EPS	VIM (SCM / HVCM)	Terminations
TCM	OnStar	DC/DC for Accessory	120V power outlet (center console)
TPIM (HCP, MCPA&B)		DC/DC for Thermal Management	Battery Chiller, and Heater
		Datalogger	CAN ports in the glove compartment
		Instrument Cluster Screen (ICS)	LCD Screen
			GFI

An extra CAN bus, named UOIT CAN bus, was first added for segregation. Nevertheless, having no overlap in CAN identifiers and being far from the maximum loading capacity of the PTE bus, the additional ECUs from the UOIT bus and the ECUs from the original PTE bus were combined for simplicity into one CAN bus, now referred to as UOIT CAN bus. Technically, the migration was never fully completed. Some ECUs are still allocated to the CAN port #3 of the VIM, instead of CAN port #2 on which the PTE bus is connected. The VIM program needs to be updated accordingly. However, the CAN network of the vehicle is not affected, since these 2 CAN ports of the VIM were physically connected together as a temporary solution. In other words, they physically communicate over the same wires, even though the signals originate from different ports. Table 37 lists the ECUs present on the 2 high-speed buses and Figure 34 shows their physical layout in the respective network. A hybrid topology is used for both CAN buses leaving the stock ECUs in their original configuration, while the additional ECU's are connected to their CAN bus using a star configuration. External 120 Ω terminations were added to compensate for the ones removed with the ECM.

Table 37. Modified HS and UOIT buses.

GMLAN HS	UOIT CAN
BCM	BCM
CGM	CIS
EBCM	CGM
EPS	EBCM
OBD-II port	OBD-II port
OnStar	
	BMS
	Motor Controller
	Charger
VIM (SCM / HVCM)	VIM (SCM / HVCM)
	Datalogger
	Instrument Cluster Screen

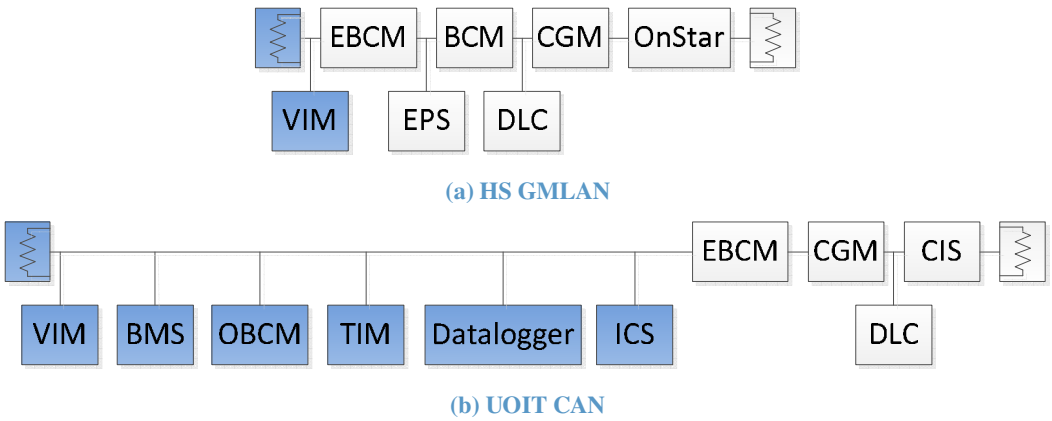


Figure 34. Modified CAN buses: (a) GMLAN HS, (b) UOIT CAN bus.

4.4 Control Strategy

4.4.1 Powertrain Control Strategy

The FFEV provides some simplification in the controls strategy to ensure functionality and operation of the vehicle. Complex controls to switch between various modes of operation are not required. The single charge depletion mode isolates most of the controls complexity in the BMS coding to ensure safe charging procedures, and the implementation of various algorithms for internal safety systems are used to protect users and promote extended battery range and life. The control system strategy consists of charge depletion mode, regenerative braking charge assist, and charge mode. Complementary to these modes, the controls strategy is classified based on the states of vehicle operation, which dictates what control system architecture is necessary to fulfill the requirements of each state.

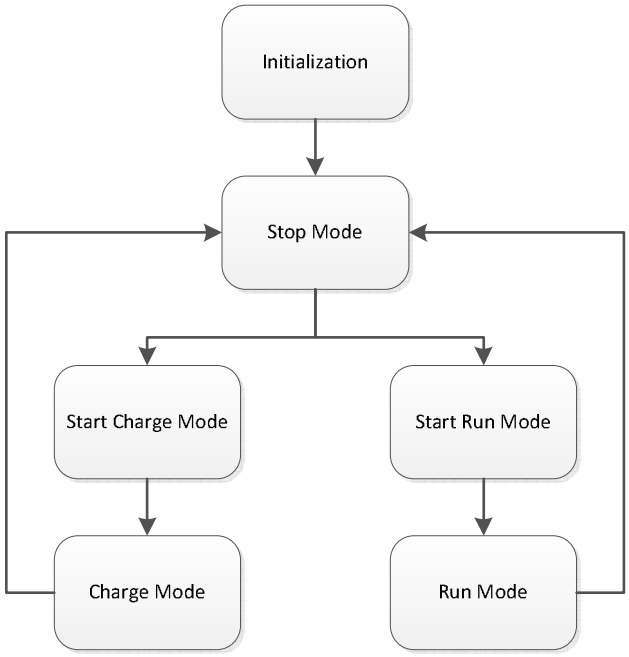


Figure 35. HV Control Module flow diagram.

The flow diagram provided in Figure 35 describes at a high level the high voltage control module (HVCM); the detailed Simulink Stateflow can be found in Appendix J Figure 95. The points gathered in Table 38 give a brief description of the states of the vehicle operation and their control functionality, whereas Table 39 details the power states of the main controllers.

Table 38. Description of Powertrain Control Strategy by Operational Modes.

Operational Modes				Description
Normal Mode	OFF			Main contactors are open, controllers are in "sleep" mode, HV discharge circuit is active to remove remaining energy from the motor controller
	ACC			Accessory controllers are functional, main contactors are open, discharge circuit is still active.
	ON	Ignition OFF		Instrument cluster turns on (all display screens are functional, main contactors are open, discharge circuit is still active).
		Ignition Starting		Transition state. The car must be in park and the driver must press the brake pedal while cranking the ignition key to start the vehicle. This starts the precharge cycle (closes the precharge contactor and the battery negative terminal contactor).
		Ignition ON	Regen Disable	Main contactors are closed, motor controller output is enabled, car is ready to drive, the car is in charge depletion mode.
			Regen Enable	MotoTron sends regen enable signal and brake pedal ON/OFF signal to the motor controller.
Charging Mode				Charging is enabled when the charge port detects an AC source plugged in. At this time the main contactors are closed, charging contactor is closed, motor output (traction) is disabled.

Table 39. Power States of Main Controllers by Operational Modes.

Modes	PRND	BMS	MotoTron	TIM	Motor
OFF	P, R, N, D	-Powered -HV output disabled	-Powered -Functional	-Not Powered -HV output disabled	Not powered
ACC	P, N			-Powered -HV output disabled	
ON + Ignition OFF	P, N				
ON + Ignition ON	P, N	-Powered -HV output enabled			
	R, D			-Powered -HV output enable	Powered
Charge	P, R, N, D	-Powered -HV output disabled	-Powered or not -Functional or not	-Powered or not -HV output disabled	Not powered

4.4.2 Discharge Control Strategy

Having only one energy source, the UOIT discharge control strategy is charge depleting 100% of the time, to nearly full depletion. Normal operation limits are between 5% and 95% SOC, with a reduced power mode below 5% SOC. Regenerative braking mode is a normal function of the motor controller and a regenerative brake blending was calibrated for an intuitive brake feel.

The final configuration of the ESS contains 94 cells, nominally 83.5 kWh. The BMS maintains the cells in a 95% to 5% state-of-charge (SOC) window, thus a 90% ΔSOC gives 75.15 kWh useable energy. When the battery hits 5% SOC, the TIM is switched into a progressively reduced power mode controlled by CAN messages between the VIM and TIM. A built-in PWM safety cut-off to stay within voltage limits is available in the BMS and could have been used to trigger and control the TIM reduced power mode. However, the team preferred having the flexibility of implementing their custom program in the VIM for a smoother integration. It should be noted that the PWM safety cut-off could be implemented as a second safety in case the VIM fails. A full depletion strategy, within the aforementioned SOC window, gives UOIT's electric vehicle a range of over 400 km. Chapter 6 discusses this topic in detail.

Regenerative braking, controlled by the motor controller, is blended with the conventional hydraulic braking system. The brake blending algorithm is proprietary to the TIM. The hydraulic pressure is controlled by the displacement of the brake pedal and the electronic brake control module (EBCM), whereas the motor controller manages the amount of regenerative braking to apply according to calibration values. For example, a low regenerative braking is applied while coasting to emulate an internal combustion engine (ICE) feel. Also, regenerative braking is reduced at high SOC to avoid overcharging the lithium battery which would result in triggering a protection mechanism.

Another aspect of the control strategy for discharging the battery is the emulation of the typical creep torque of an ICE. A built-in function of the motor controller was calibrated to emulate creep torque when having the shift lever in the drive position.

4.4.3 High voltage schematic

The high voltage schematic of the UOIT EcoCAR architecture is detailed in this section and shown in Figure 36.

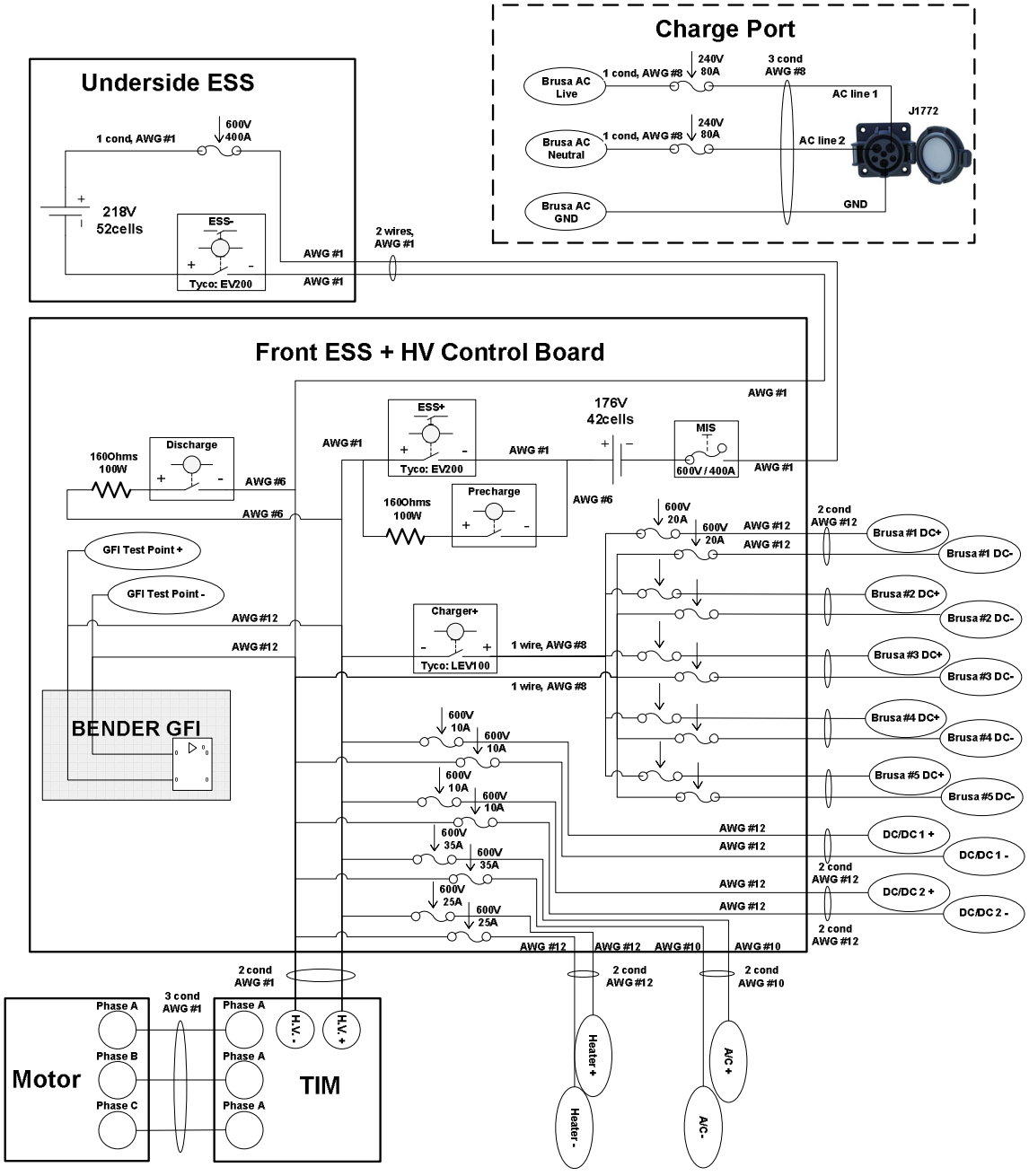


Figure 36. High Voltage Schematic.

The 52 cell underside battery pack is connected to the one underhood by 1 gauge wire. A normally open Tyco EV200 contactor is connected to the underside negative terminal, which is also the most negative line of both battery packs combined. Therefore, this contactor is considered the main negative contactor of the powertrain. The main positive contactor is located in the front ESS. The underside ESS positive terminal is protected by a 600 V / 400 A fuse. The fuse and the contactor provide a safe unpowered outlet when the module is disconnected from the vehicle for maintenance and when the vehicle is turned off.

Containing most of the high voltage control hardware, the front battery pack, located in the engine bay, employs a more complex wiring strategy. The front battery negative terminal is connected to the underside ESS positive terminal using a 1 gauge wire. A manual interrupt switch (MIS), having a built-in fuse of 600 V / 400 A, is also on this high voltage line. The precharge circuit and the battery main positive contactor, in a parallel configuration, are connected to the front battery positive terminal. The other side of the main positive contactor is the main positive HV line, whereas the main negative HV line is directly connected to the underside ESS contactor. The following devices are connected to this HV line: TIM, chargers, discharge circuit, GFI, DC/DC 1, DC/DC 2, A/C, Heater.

The TIM is directly connected by 1 gauge wire to the main HV lines and utilizes the two 600 V / 400 A battery fuses as protection. It gets connected and disconnected from the battery by the main contactors, however the motor is not yet powered when high voltage is sensed at the TIM's input. The traction is enabled by the VIM controlling many of the TIM inputs and sending CAN commands. The 3-phase induction motor is hooked directly via three short and shielded 1 gauge wires.

The vehicle HV inlet is compliant with the SAE J1772 standard and located behind the front license plate which is mounted on a sprung hinge. From the power inlet, the 240 V_{ac} flows to a breaker box splitting the input power between the 5 BRUSA chargers, using 8 gauge wire, individually protected by 240 V / 15 A breakers, however only the

240 V_{ac} / 80 A breaker is shown on Figure 36. In addition to the charger internal fast-acting 15 A fuse, the DC output of each charger is protected by 600 V / 20 A fuses and merged using a positive and a negative bus bar. From the positive bus bar, 8 gauge wire is connected to a normally open Tyco LEV100 contactor independently managing the charger's logic and the HV DC connection between the chargers and the vehicle's HV line. As detailed in Appendix I, 3 operation modes are available on BRUSA chargers: auto, master/booster and CAN. Due to using 5 BRUSA chargers, a master/booster configuration was not possible. Although the CAN identifiers used by each BRUSA charger could have been modified using RS232 communication as shown in Appendix I, CAN control of multiple chargers would have required designing complex controls within the VIM. However, this would have allowed the VIM to monitor and control each BRUSA charger individually and dynamically. For simplicity and as a competition time saving measure, the auto mode was therefore chosen and each charger was configured through RS232 communication for autonomous operation. The control strategy for this is covered in Section 4.5.2.

The discharge circuit is designed to have the same performance as the precharge circuit, thus employs similar hardware, i.e. a RY4S-UL-DC12V normally closed relay from IDEC (normally open for the precharge circuit), a 160 Ω / 100 W resistor and 6 gauge wire. There is no protection required for the GFI, since the HV leads are only probes and no current is drawn. Both DC/DCs utilize 600 V / 10 A fuses and 12 gauge wire. The air conditioning is connected to 600 V / 35 A fuses through 10 gauge wire, whereas the heater uses 12 gauge wire and 600 V / 25 A fuses.

4.5 Control Law

A MotoTron controller, model ECM-5554-112, is used in order to integrate new systems within the GM donated SUV. It emulates messages required by the stock controllers remaining on the CAN network. The MotoTron emulates these along with providing a communications bridge to the CAN networks for the added HV systems. Thus, the vehicle has 3 CAN buses: HS GMLAN (High Speed General Motors Local Area Network), PTE (Power Train Expansion) GMLAN and SW (Single wire) GMLAN.

As explained in Section 4.3.2, it was planned to implement an extra CAN bus for the HV systems, but for wiring simplicity the UOIT CAN bus was merged to the PTE GMLAN. The combined bus also doubles as the supervisory (safety) controller. In addition to CAN communication, RS232 is also used, but only to upload and download configuration files and data to or from the added controllers. These include: the REAP BMS system, the MES-DEA TIM 600 induction motor controller and BRUSA NLG513 chargers, along with a Vector CANlog4 data logging device. As explained in Section 4.4, the MotoTron outputs also command the high voltage contactors and set the precharge sequence, thus it acts as an integration module and a high voltage controller, so it is referred to as a vehicle integration module (VIM).

4.5.1 Battery Management System

This system consists of 7 units responsible for monitoring 14 cells each using a master-slave configuration as shown in Figure 37. To save space, only 4 modules are depicted on the schematic. Table 40 describes the inputs and outputs of a REAP BMS unit.

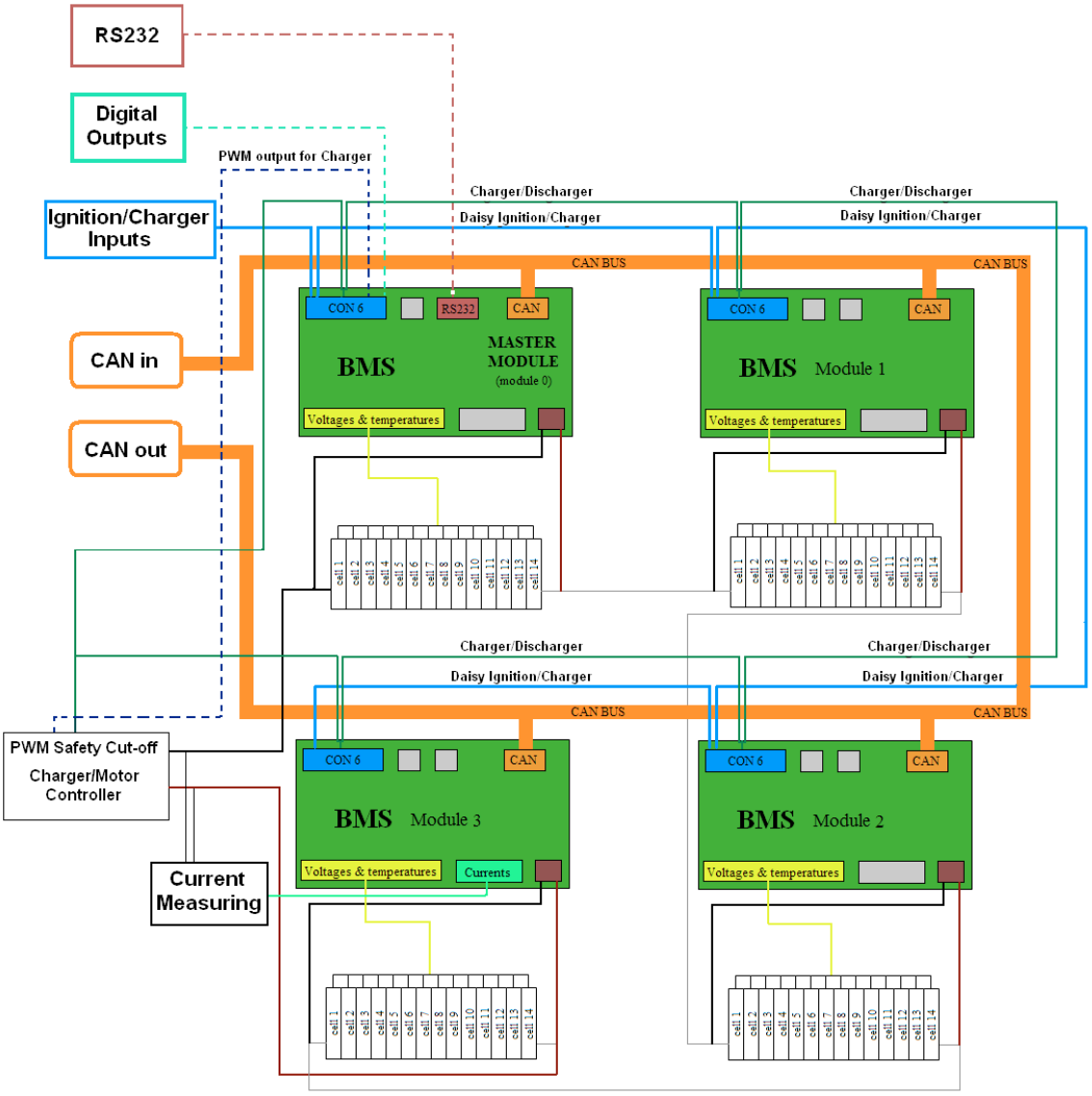


Figure 37. BMS I/Os and Master/Slave Configuration [12].

Table 40. BMS I/Os Description [12].

RS232	Configuration of each BMS's software
CAN	-Normal mode: Broadcast data to the MotoTron -Charge mode: Broadcast commands to the charger and data to the MotoTron
Ignition/Charger Input	-Ignition input: When car is in ON and ignition ON mode -Charger input: When the charger is connected / When AC is detected at the charger inlet
Current Measuring	Measure the current entering or exiting the battery
PWM safety cut-off*	-Normal mode: Shut off the motor controller output -Charge mode: Shut off the charger output
Digital Outputs**	N/A

*PWM safety cut-off not implemented in UOIT's design, see Section 4.4.2.

**Not used.

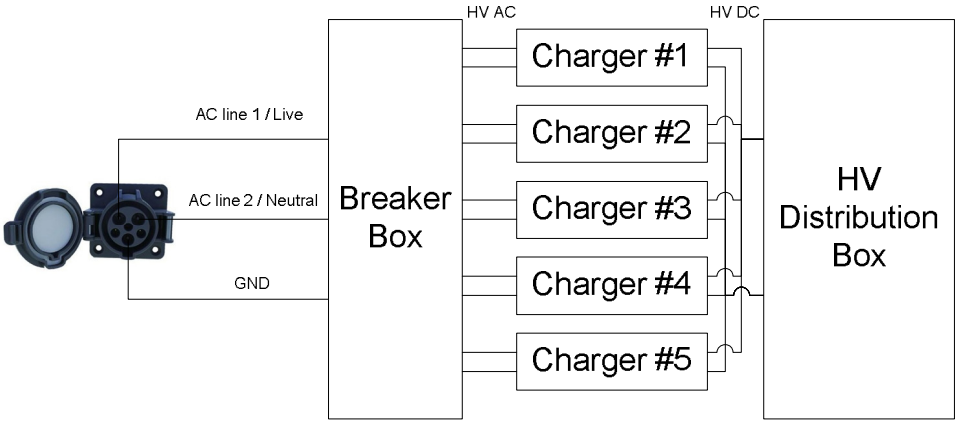
Each BMS unit monitors the voltage and temperature of its cells to ensure that they are within set ranges. Each cell is connected to a BMS module through a lead protected by a 125 V / 1 A fuse. This allows the BMS system to balance the battery pack. The system achieves balancing via a flying capacitor scheme, and all modules are networked via CAN. A temperature sensor is fitted between every 2 cells, and cell voltages are individually monitored and balanced. When an individual cell voltage lies outside the specified range, a signal is sent by the BMS to the MotoTron to open the main contactors and disable the motor controller. When an individual cell temperature exceeds the specified range the BMS sends a signal to the MotoTron to limit the output current of the motor controller. A hall effect current sensor, iSensor L 1.1 400 from REAP System, is place on the main HV line to monitor the current entering and exiting the ESS. SOC estimations are made, and the CAN messaging is heard by the MotoTron controller which can take actions like limiting current to / from the TIM or chargers or disabling the system.

Similar to the HVCM, the BMS has 2 main modes: normal and charging mode. These modes are controlled using a hardware connection from the HVCM to each BMS unit. The normal mode of the BMS can be subdivided into: ignition off and ignition on. When the ignition is off cell balancing occurs. When the ignition input is activated, the BMS expects current to be drawn. As soon as AC is detected at the charging inlet, the

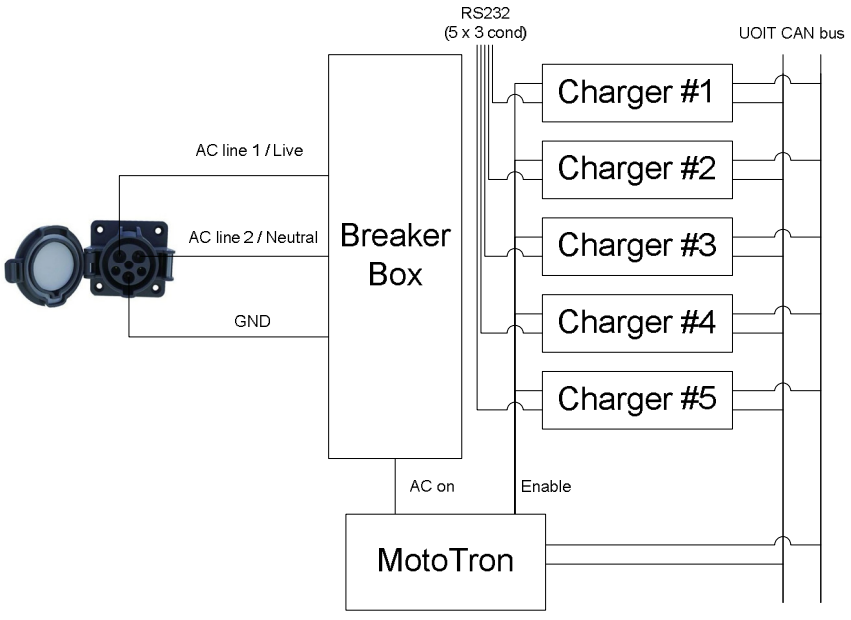
VIM exits the run mode described in Figure 35 regardless of the shift lever position and vehicle speed. The VIM activates the charger input of the BMS, enables the chargers, closes the contactors and starts charging once the ignition key is in the OFF or ACC position, and all the other charging conditions are met. It is possible to charge with the ignition key ON, if it has not been cranked in the current key cycle. In other words, if the driver cranked the ignition key, or attempted to crank it while charging, it has to be turned back to ACC or OFF before charging is enabled again. Additional safety could be combined to the ignition key logic and AC detection using the shift lever position and the vehicle speed. The BMS also provides regular battery information and updates to the VIM using the modified CAN bus whether it is in charging or normal mode.

4.5.2 On-Board Charger Module

The UOIT Full Function Electric Vehicle can be equipped with up to 5 BRUSA NLG513 onboard chargers connected in parallel as shown in Figure 38. Their inputs and outputs are detailed in Table 41.



(a) High Voltage Parallel Configuration.



(b) I/Os.

Figure 38. Chargers (a) High Voltage Parallel Configuration (b) I/Os.

Table 41. OBCM I/Os Description [13].

RS232	Configuration of each charger's software
CAN	-Rx: Receive command from the BMS to drive the charge cycle -Tx: Broadcast data to the MotoTron
Voltage sensors	Measure the battery voltage
Current sensors	Measure and limit the battery charging current
Temperature sensors	Internal and external temperatures
PWM safety cut-off*	Input controlled by the BMS to shut off the charger's HV output
Charger enabled output	Enable signal is sent to the BMS and TIM when charger is ready to charge using the MotoTron

*PWM safety cut-off not implemented in UOIT's design, see Section 4.4.2.

This charging system consists of five BRUSA chargers connected in parallel and each charger has an individual charging profile. Initially, it was planned to connect the chargers in a master-slave configuration to allow the BMS to send commands over the UOIT CAN bus to the master, which would then control the slaves. However, being limited to 3 chargers, this configuration was changed to a parallel one when additional chargers were added to facilitate a faster charge time to meet VTS charging expectations.

The maximum charging power each BRUSA can deliver is 3.3 kW, therefore a maximum charging power of 16.5 kW can be supplied to the batteries, thus 47.4 A_{dc} continuous at the nominal battery voltage of 347.8 V_{dc}. The acceptable grid voltage range is single phase 100-250 V_{ac}, and current limits can also be set to avoid tripping breakers. The power plug on the vehicle at present is a SAE J1772 80 A, and adaptable via the power cord to whatever supply is available for charging. The AC pilot line is constantly read by the VIM, however the PWM pilot line limiting the power flow is held static and fixed to 100% using a diode and resistor.

Each charger unit monitors the grid voltage and current, the battery voltage, branch output current and internal temperature. In addition to CAN messages, a hardwire output signal indicates whether or not the unit is ready for charging. Due to the parallel configuration, the total charging current is the summation of all 5 charger current outputs and is only monitored by the BMS current sensor. Therefore, the output current of each

unit must be limited to calibrated values and adjusted for each charging phase. Table 42 specifies the voltage of each charging phase.

Table 42. Charging Phases.

Phase 1: Preconditioning (V)	$V \leq 282$ (3.0V*94cells)	0.1 C
Phase 2: Constant Current (V)	$282 < V < 394.8$	Max. Charging Current Available
Phase 3: Constant Voltage (V)	$V \geq 394.8$ (4.2V*94cells)	0.07 C

When the battery state of charge is very low, i.e. a voltage under 282 V, the first phase slowly charges the battery to a safe voltage level using a low constant current (0.1C). Once the safe voltage threshold is reached, the charger enters the constant current phase, and uses the maximum charge current available. For the cell technology used, the nominal battery limit is 1.0 C without cooling. When the battery reaches the maximum voltage of 394.8 V, the last mode is started. In this phase, the voltage remains constant and the current slowly drops until it tappers off (0.07C). Moreover, the chargers are programmed to turns off sequentially to increase the accuracy during the last charging phase without reducing the charging time. The control logic in the chargers is static and programmed via RS232 communication and using the supplier's software. The VIM only enables and disables the chargers, along with controlling the contactors by monitoring the CAN messages sent by the charger configured to turn off last and on first.

4.5.3 Motor controller

The available I/Os on the TIM are shown in Figure 39. For clarity, the high voltage wires and sensors are omitted. Nevertheless, among the available controls I/Os, several were not required by the control strategy used for the powertrain, thus their pin numbers were dashed. An exhaustive description of the TIM's I/Os is found in Table 43.

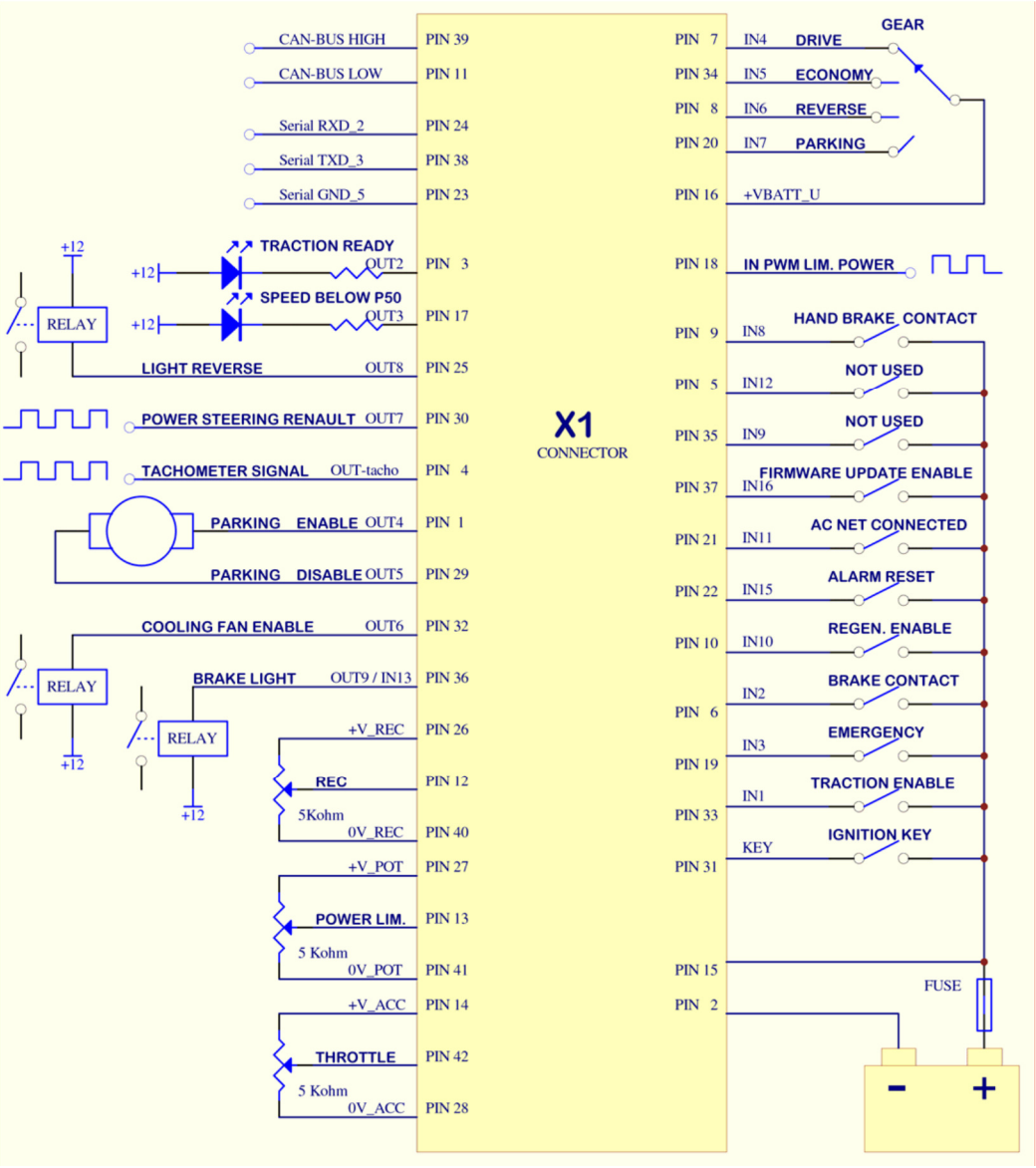


Figure 39. TIM I/Os [15].

Table 43. TIM I/Os Description [15].

RS232	Configuration of TIM's software
CAN	-Rx: Receive commands from the MotoTron -Tx: Broadcast data to the MotoTron
Cooling fan enable	Cooling fan activated when needed
Brake pedal	Directly connected to the stock brake pedal
Accelerator pedal	Directly connected to the stock accelerator
Gear selector (shifter lever)	Park, Reverse, Drive and Economy mode
Hand brake contact	Activated when hand brake engaged
AC net connected	Charger connected
Regen enable	Always enable (PWM safety cut-off is used if battery at full charge)
Brake contact	Activated when brake pedal is pressed
Emergency	Shut off HV output when activated
Traction enable	Activated when on drive or reverse gear
Ignition key	Activated by the stock ignition key
Voltage sensors	Measure the battery voltage
Current sensors	Measure and limit the battery charging/discharging current
Temperature sensors	Measure internal and external temperatures
PWM safety cut-off (PWM lim. power)*	Input controlled by the BMS to shut off the TIM's HV output
Light reverse**	Activated when on reverse gear

*PWM safety cut-off not implemented in UOIT's design, see Section 4.4.2.
**The reverse light built-in function of the inverted is not used, but the vehicle reverse and brake lights are dynamically emulated by the VIM.

The vehicle acceleration and brake pedals have their respective potentiometers directly hardwired to the motor controller input pins, throttle and power limiter. The accelerator position commands the torque output of the motor when the gear selector is not in PARK position. The brake pedal position governs the regenerative braking amount blended to the hydraulic braking when regenerative braking is enabled and brake contact is activated.

The VIM controls these 2 inputs along with the GEAR selector, HAND BRAKE CONTACT, AC NET CONNECTED, EMERGENCY, TRACTION ENABLE and IGNITION KEY. The TRACTION ENABLE input of the TIM is the main enabler of the powertrain and, as explained in Section 4.4, is managed by the HVCM residing inside the VIM.

Similar to the BMS and charger, an RS232 interface is used to program and calibrate the motor controller; the CAN communication provides information on the motor and the drive itself; and sensors read the battery voltage and current along with the temperature of the TIM.

4.6 Safety Control Strategy

4.6.1 Acceleration Fault Management

Throughout the years of design, a lot of concerns were raised by the EcoCAR competition judges about fault mitigation of the acceleration pedal. The accelerator has 2 potentiometers of opposite travel logic to determine its position. One is connected to and managed by the motor controller's throttle input and the second one is linked directly to the supervisory controller which has fault management capabilities. The TIM has a basic built-in fault detection and management (short to ground or power supply), and arbitrates between control signal inputs if there is any conflicts on the throttle or brake (favoring the brake). Furthermore, it sends the pedal status on the UOIT CAN bus. The VIM is also programmed to detect and mitigate faults to ground or power on the acceleration pedal potentiometer directly connected to it.

The supervisory controller prevents unintended acceleration by detecting faults determined by the redundancy between the pedal status sent from the TIM and the second potentiometer locally monitored. Therefore, the acceleration pedal is read by 2 devices, TIM and VIM, having fault mitigation safety features able to open the main contactors to prevent unintended acceleration.

4.6.2 Cell Protection

Another major safety concern on electric vehicles is the cell voltage and temperature of the lithium battery cells. Here again, redundancy is used. The BMS monitors for over-charge, over-discharge, and temperature conditions. It sends this information over CAN to the VIM controlling the charge and main contactors, along with the signal enabling

and disabling the charger high voltage DC output. The BMS built-in charging and discharging relay switches could also be used as additional safety, but were not implemented in UOIT's prototype. The VIM reads the information provided by the BMS over CAN bus and calculates the delta between the minimum and maximum cell voltage under load conditions to detect defective cells. It activates the TIM's emergency mode when a defective cell is established to limit the output current of the motor controller, thus reducing the stress on the cell. A similar safety feature exists in the VIM for cell temperature. As it is explained in Section 4.6.5, there are 2 levels of safety: reduced current and OFF (HV contactors open).

4.6.3 Voltage, Current and Temperature Management

The powertrain has 4 main controllers: the BMS, charger, TIM and MotoTron. The first 3 monitor voltage, current and temperature and can enter different safety modes depending on the values sensed. They also send the information to the VIM which can take additional actions when abnormal situations are detected.

4.6.4 High Voltage Interlock

HV interlock lines are installed on the HV hardware, i.e. all the HV box lids, HV connectors/cables, HV controllers and E-Stops, and are monitored by the VIM. Figure 40 shows the interlock configuration of the EV prototype designed by UOIT.

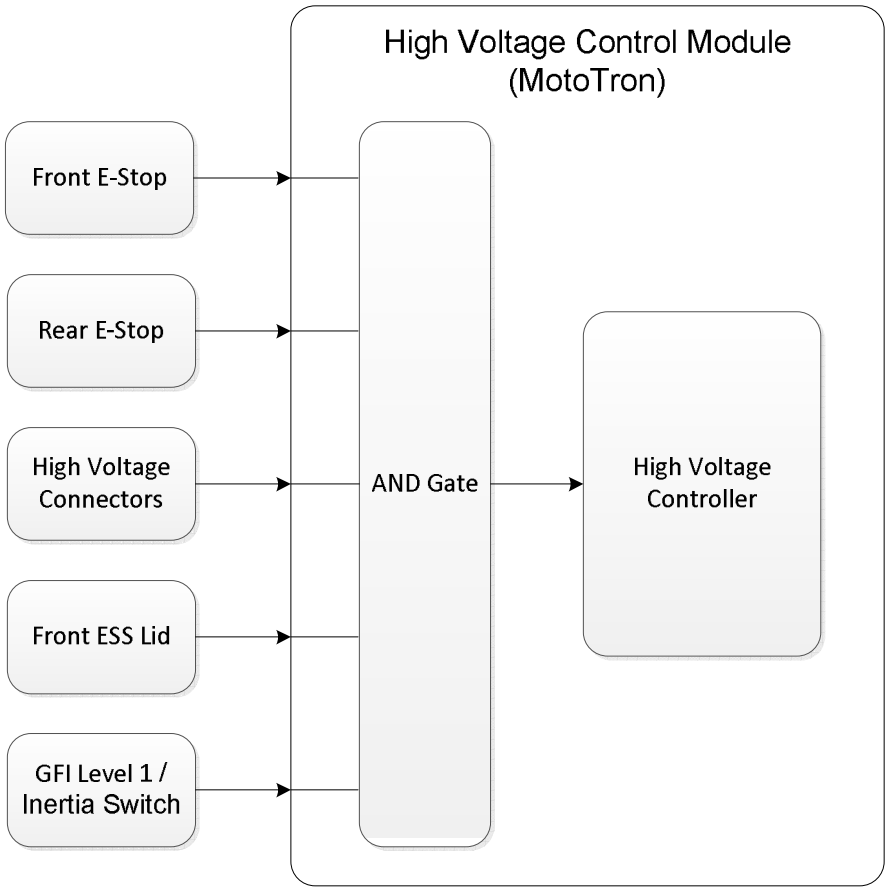


Figure 40. Interlock configuration.

When one of these HV trace lines is not a closed circuit, the VIM triggers the opening of the main contactors and deactivates the traction enable signal to the motor controller. The main HV contactors are equipped with a feedback relay switch, also monitored by the VIM, to insure they are in the desired state. The TIM broadcasts the traction enable status on the UOIT CAN bus. Since the motor controller already has built-in safety features designed to disable the motor power input, it was judged unnecessary to incorporate it to the feedback verification made by the VIM. However, it could be implemented for extra safety.

4.6.5 Ground Fault Interrupt and Inertia Switch

The ground fault interrupt (GFI) unit is connected to the high voltage and low voltage buses to verify that there is no leakage current in the electrical system. The unit

distinguishes between low and high leakage scenarios by setting flags that send signals to the MotoTron so that the appropriate actions can be taken. If a low leakage current is detected the motor controller is placed in an emergency mode that limits the output current of the motor and an orange warning LED is turned on to inform the user. If a high leakage current is detected the MotoTron sends a signal to open the main contactors, disabling the motor controller output, to stop the vehicle. In this case, a red GFI LED is also turned on in the instrument cluster.

An inertia switch, from First Technology, is calibrated to get triggered when the vehicle sustains an impact greater than 8 G's. It acts as a safety feature used to open the main contactor relays in the event of a collision to protect passengers from high voltage exposure. In other words, it is a resettable crash switch that disconnects the power in the high voltage distribution box during a significant collision. The specifications on this component are reviewed in Table 44.

Table 44. Resettable Crash Sensor Specifications.

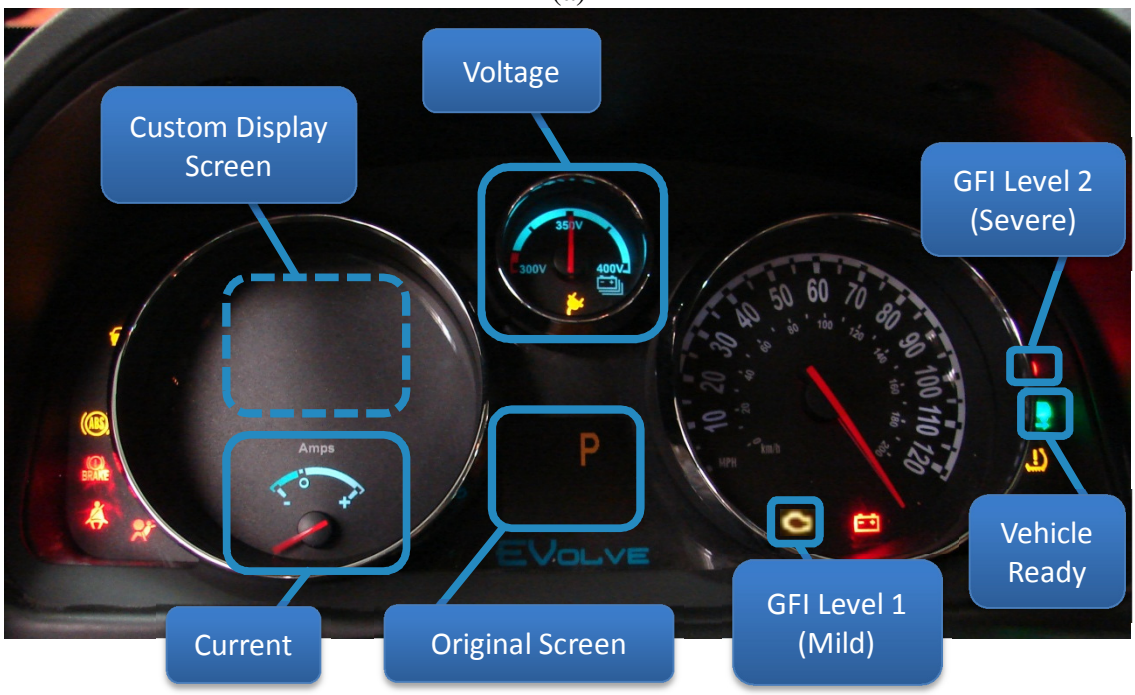
Response to impact	360º
Rated load	10 A
Calibration range	8-30 G's
Resettable	Manually
Inertia Mechanism	Magnet restrained mass

4.7 On-board Debugging and User Interfaces

4.7.1 Driver Notifications

The instrument panel cluster, or dashboard, informs the driver through gauges, lights and chimes on the vehicle status. The information required by someone driving an electric vehicle is different than the one needed for driving a conventional vehicle. To address this need, the instrument cluster was customized by installing a new layout back panel, modifying the logic behind multiple LEDs, gauges and chimes, and also integrating an in-house screen within the original dashboard. Figure 41 shows a picture of the stock cluster and one of the new layout. A detailed model of the instrument panel cluster is shown in Appendix K.

(a)



(b)

Figure 41. Instrument Cluster (a) Before (b) After (without the custom screen).

The speedometer remains unmodified, the fuel gauge now displays the battery voltage, as an approximation of its state-of-charge (SOC), and the empty light turns on at low SOC. A bidirectional ampmeter gauge was created, using the original 12 V battery needle, to indicate the current flow while monitoring the amount of regenerative braking.

The yellow oil pressure light was replaced by a red LED for ground fault interrupt alarms and any issues related to the powertrain. A chime is also triggered when the GFI

light is activated to cue the driver via visual and audible indicators for critical situations. For a lower current leakage, the malfunction indication light (MIL), or commonly called the check engine light, is enabled and no chime is audible. A green vehicle ready light replaces the yellow oil change indication light. The vehicle ready light indicates that the main HV contactors are closed, in other words that the powertrain is enabled.

The stock screen occupying the center of the cluster, situated between the main 2 gauges, could not be reprogrammed or modified due to its embedded firmware. Instead, a custom one was installed at the hybrid gauge position on the left. The in-house screen controller communicates with the MotoTron over the UOIT CAN bus and enhances the information displayed to the driver. It is located behind the instrument cluster. The PRND, maximum and minimum cell voltage, SOC based on energy consumption, battery temperature and expected range are examples of variables displayed on this backlit LCD screen.

4.7.2 Troubleshooting Interfaces for Engineering Development

Engineering a prototype having the complexity of an electric vehicle requires a lot of debugging and good troubleshooting tools. In the design of the UOIT EcoCAR, the 4 programs interfacing with the 4 main controllers of the powertrain, i.e. the MotoTron, BMS, charger, TIM, were key to the project's success. A specialized program was provided by each manufacturer except for the BMS.

Being the integration module, the MotoTron is connected to all CAN buses and in communication with most modules. Therefore, it was used as a diagnostics tool via its user interface MotoTune. This gives access to any desired variable of the controller for monitoring, calibrating and testing. Having a visual layout similar to an Excel spreadsheet, variables could be dragged and dropped easily to create displays to diagnose the vehicle. Three main interfaces were created and can be found in Appendix G, Figure 79 to Figure 81, the BMS supervisory, VIM and HVCM interfaces respectively.

Troubleshooting of the 3 other devices was possible using RS232 serial communication. The BMS can be programmed using a generic serial terminal; however, the UOIT team had more success employing the program HyperTerminal. From the serial port, 3 modes can be accessed: monitoring of the cell voltages and general errors, parameter settings and configuration mode. Screen captures can be found in Appendix G Figure 75 to Figure 78. The serial communication has the disadvantage of being limited to only 1 BMS unit at a time. Therefore, each module needs to be accessed individually to be monitored, calibrated and configured. Most of the variables are monitored through MotoTune and the calibration and configuration are infrequent. Additionally, BMS modules need to be configured individually, but the calibrations in the master module are automatically shared with the slaves.

Similar to the BMS, each BRUSA unit has to be programmed individually using RS232 communication and the specialized program BRUSA Charger Star NLG5. It allows programming the 3 charging phases and sets multiple thresholds, such as the nominal battery voltage, charging and mains current, time limit and maximum voltage output to name only a few. Figure 69 to Figure 71 found in Appendix G represent the 3 charging modes programmed into the last unit to switch off, for the cascading shut off sequence.

The traction inverter module uses a RS232 interface named Powertrain Inverter Series. This interface was very useful for motor calibration and behaviour analysis. For example, graphs can be plotted in near real-time and motor data can be recorded for post analysis as shown in Appendix G, Figure 82. Relevant calibration values programmed in the UOIT EcoCAR traction inverter can also be found in Appendix G, Table 69 and Table 70. An auto-tuning feature is also available to automatically tune the motor with the motor controller. Additionally, approximately 300 parameters can be modified to customize this drive. Unfortunately, these parameters have poor documentation making it difficult to calibrate properly. In conclusion, the physical connections required to enable the communication of these 4 controller interfaces were made accessible from the front passenger seat through the glove compartment for convenience.

4.8 Summary

The ECU architecture of the original and modified vehicle was presented. The control strategy implemented by the high voltage controller and the 3 main powertrain controllers were detailed. A more in-depth explanation of the safety control strategy was provided. A description of the on-board diagnostics system and troubleshooting tools developed was reviewed. The next chapter addresses the implementation of this control strategy via programming the VIM.

Chapter 5

Vehicle Integration Module

5.1 Introduction

The current chapter details the controller at the center of the electrical and control system integration strategy described in the previous chapter. An overview of the MotoTron controller is presented, followed by an explanation of the model-based program created. The encoders and decoders required for a successful vehicle integration are detailed along with the high voltage controller. This chapter concludes by describing how the vehicle integration module links the main ECUs through CAN communication.

5.2 Overview of the MotoTron

As described in the previous chapter, removing several important ECUs from a vehicle requires the addition of new ones. It is unlikely that all the new controllers will be compatible with one another or with the remaining stock ones. The purpose of a programmable controller like the MotoTron is to integrate all these ECUs together as a functioning system by interconnecting them and rebuilding the dismantled system. Its multiple CAN ports allow the MotoTron to communicate with controllers spread throughout different CAN buses and also act as a gateway between them. Some CAN signals need to be translated before being sent back on the destination CAN bus, while others must be emulated. Emulated signals are sometime required to insure proper functioning of original ECUs which expect communication from removed ones. Such a controller also has the task of controlling the additional devices. This is accomplished by controlling their inputs and supervising their outputs. These could be hardwired as well as transmitted information. A detailed list of the MotoTron I/Os and their functionality can be found in Appendix J, while a list of the relevant ports follows:

- Analog inputs
- Discrete inputs
- Discrete outputs

- PWM outputs
- Low side outputs (current sinking drivers)
- CAN communication ports

MotoTron is a controller series made by Woodward MotoTron Control Solutions, thus a few controllers with different features could have been chosen as the vehicle integration module. As a reference, MotoHawk is a the software development tool, which uses the Matlab Simulink MotoHawk library for model-based design and the MotoTune program for flashing and calibrating MotoTron controllers. The software versions used to program the VIM are in Appendix J. Table 45 compares the MotoTron controller used for the second year of the competition to the one chosen in the final design.

Table 45. Comparison between ECM-0555 and ECM-5554.

	ECM-0555	ECM-5554
Processor	Freescale MPC555, 40 MHz	Freescale MPC5554, 80 MHz
Flash memory	448 kB	2 MB
RAM memory	26 kB	64 kB
# of CAN ports	2	3
Analog inputs	19	33
TTL level outputs	8	8
Low side outputs	4 (contactors) 16 (others)	4 (contactors) 19 (others)
Cost (with harness)	1000 $	1300 $

Even though the ECM-0555 was powerful enough for the second year competition, it was missing an important feature to accomplish the prototype's final design, a 3rd CAN port. Being more powerful overall and having the required extra CAN port are the main reasons why the ECM-5554 was chosen over the ECM-0555 in the final year and the unit is displayed in Figure 15, Section 2.7.4. Also, the VIM pinout can be found in Appendix J.

5.3 Model-based design

5.3.1 Top Level Subsystem

The top level subsystem of the VIM model contains the standard MotoTron blocks required to prepare the model for code generation and to configure the basic function blocks. Extra blocks are also added to initialize functions related to the program modeled. Appendix J, Figure 84, represents the top level subsystem of the VIM.

The MotoTron controller series has many different controllers sharing the same Simulink library, thus a target definition block must be added, shown by item #1 in Figure 84, Appendix J. This block is used to set up the model as a MotoHawk project and to specify the target, i.e. the MotoTron device. As mentioned in the previous section, the MotoTron chosen for the VIM is the ECM-5554-112-0904-C00-M. Along with the target, the compiler for the Matlab Real-Time Workshop (RTI) needs to be defined. As a reference, the compiler required by this MotoTron is the gcc-powerpc-eabisps 4.4, identified by item #2 in Appendix J, Figure 84.

The main power relay and main switch can be found in the wiring harness used to power and access the MotoTron inputs and outputs. The main switch is a button for the user to manually turn on and off the controller, whereas the main power relay is controlled by the MotoTron and needs to be initialized in the top level subsystem. This initialization block is marked by item #3 in Appendix J, Figure 84.

The non-virtual subsystem named “Foreground” contains the whole program of the VIM. A virtual subsystem is only a subsystem grouping blocks for visual convenience, whereas a non-virtual subsystem in Simulink is a subsystem triggered by a function-call. These function-calls can be triggered by actions or at periodic time intervals. The latter can also be termed function-call subsystem or triggered subsystem. For MotoHawk projects, each non-virtual subsystem located in the top level must be triggered by a MotoHawk entry point trigger block to allow the compiler to generate compatible c-code. Multiple non-virtual subsystems can run at different rates and interact with one another. However, it was elected to design the VIM’s program in a single subsystem

triggered every 5 ms. The MotoTron trigger block and the triggered subsystem are represented by item #4 in Appendix J, Figure 84.

The 4 items mentioned are required for any MotoTron controller regardless of its purpose. Using CAN communication and fault management, additional definition blocks related to the programmed logic are present in the top level subsystem. A CAN definition block is added for each of the 3 CAN buses and a fault manager definition block is require for the acceleration and brake pedal fault detection and mitigation.

5.3.2 Model Structure

The main program of the VIM is found within the "Foreground" non-virtual subsystem and is arbitrarily divided into 18 virtual subsystems. Note, this does not mean non-virtual subsystems can not be used, but rather means their use was not deemed necessary considering that the main program, the "Foreground", is evaluated every 5 ms. Consequently, every sections of the main program is evaluated at a 5 ms periodic rate, i.e. the CAN message emulation, HVCM, I/O decoding and actuating, fault management, etc. For example, the acceleration pedal potentiometer is read every 5 ms and the fault manager outputs an updated status indicating whether or not the accelerator is faulty within 5 ms; in other words, before the next accelerator potentiometer value is read. Appendix J, Figure 85, shows the core model of the VIM.

Most of these subsystems are for CAN communication. The emulated ECUs on high speed and powertrain expansion GMLAN have their own subsystems and so do the added UOIT powertrain controllers. The I/Os subsystem contains the hardware dependent blocks reading the MotoTron input pins connected to sensor outputs and actuating the MotoTron output pins connected to actuators. The high voltage controller manages the high voltage strategy using a Stateflow controller. PRND_IGN decodes the transmission shift lever position and the ignition key position. Pedal_Faults detects and mitigates faults on the acceleration and brake pedals, whereas the "NewFeatures" subsystem controls other features such as the front radiator fans.

5.4 Encoders and Decoders

5.4.1 Motor Controller Controls

As seen in section 4.5.3, numerous inputs are required by the motor controller to operate. Where the acceleration and brake pedals are directly wired to it, the other features are controlled by the VIM. Using 0 to 5V TTL outputs, the MotoTron ports couldn't be directly connected to the motor controller. Therefore, a voltage translator was added between the VIM and the motor controller to achieve voltage boost to 12V. Figure 42 is a high level schematic of the controls configuration between the VIM and TIM using a voltage translator and the UOIT CAN bus.

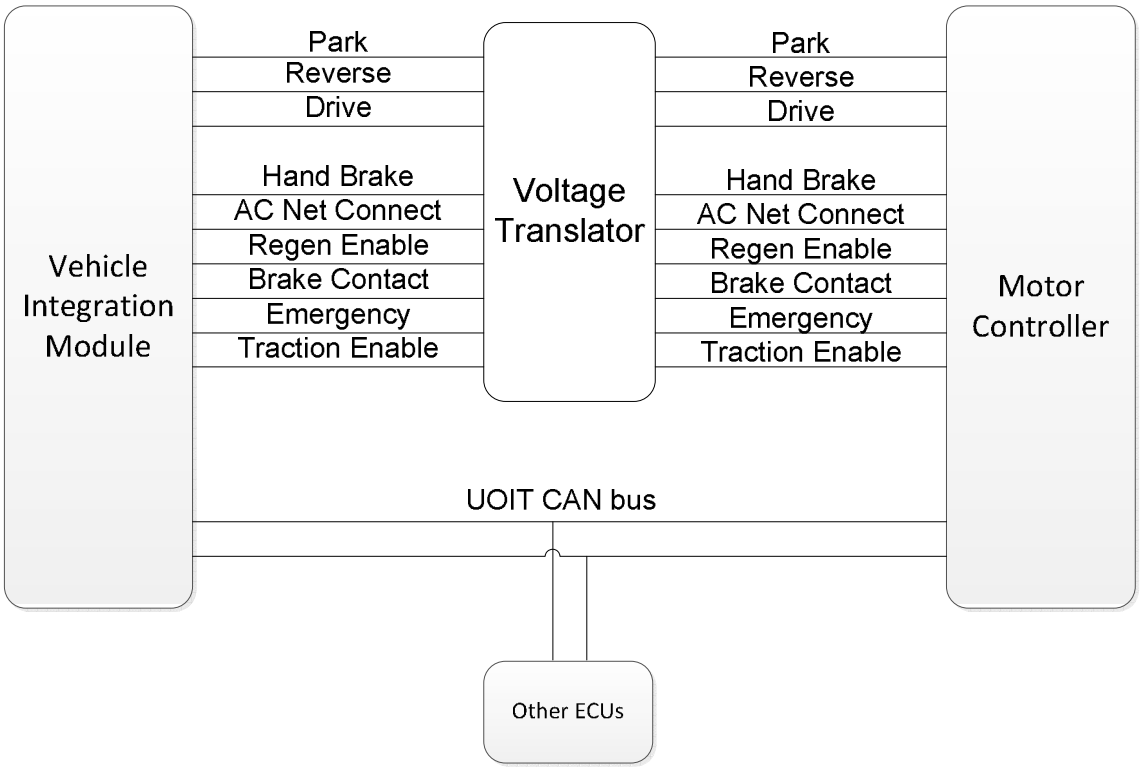


Figure 42. Motor Controller Controls.

5.4.2 PRND and Ignition

5.4.2.1 PRND

The shift lever is connected through a cable to the motor transmission switch (encoder) which also mechanically takes the motor out of park (pawl locking clutch

lever) according to the PRND lever position. In order to vary the transmission signal in the controls, the output of the encoder originally connected to the S10 engine control module (ECM) is rerouted to the custom vehicle integration module (VIM). The VIM decodes the shift lever position, translates the position for numerous dynamically emulated GMLAN signals and generates the hardwired inputs required by the motor controller.

The logic of the encoder is depicted by Figure 43, and Table 46 explicitly details its logic. A complex encoding system is used to avoid unintended gear shift. Two of the A, B, C and D encoder inputs need to be high to enable a specific gear, along with the ground sensor. Additionally, a park/neutral switch needs to be low to enable reverse or drive modes.

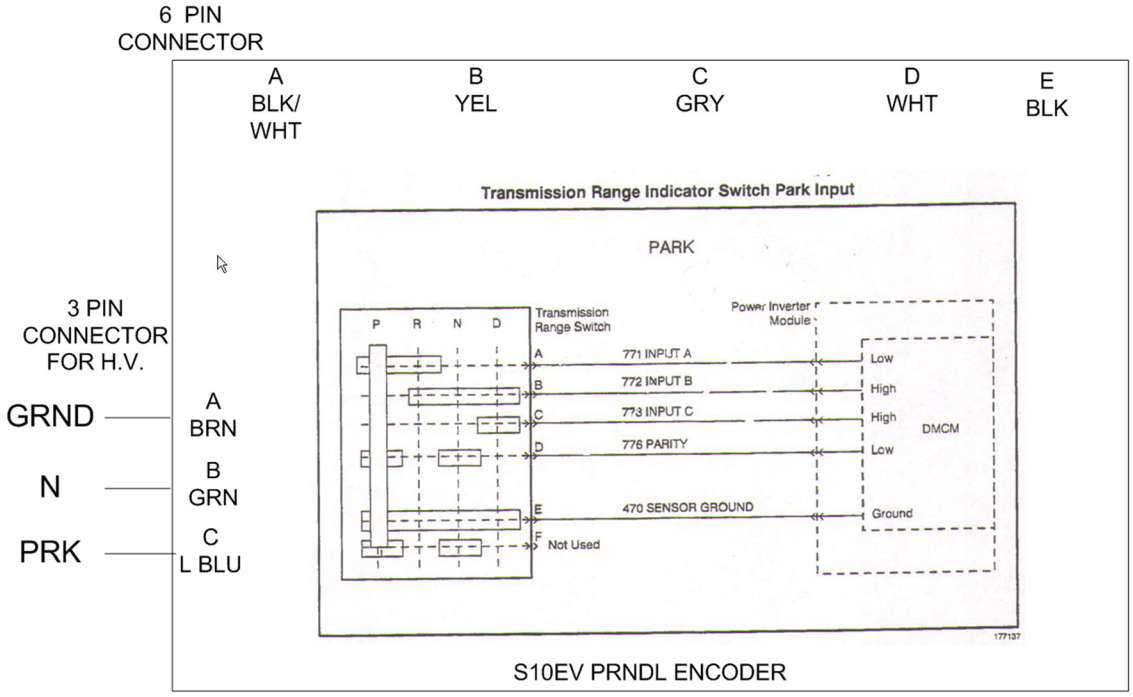


Figure 43. Motor Shift Lever Encoder.

Table 46. Transmission Shift Lever Encoder Logic.

	P	R	N	D
A	1	1	0	0
B	0	1	1	1
C	0	0	0	1
D	1	0	1	0
Gnd Sensor	1	1	1	1
Park/Neutral	1	0	1	0

Several modeling techniques, such as look up tables and Stateflow diagrams, could have been used to program the transmission decoder in the VIM; however, simply using model-based logic gates was judged adequate for this task. Figure 86 in Appendix J shows the memory access blocks of the A, B, C and D inputs filtered by logic gates generating the park, reverse, neutral and drive output signals. These are used to update the motor controller inputs (pins 7, 8 and 20 of Figure 39, respectively DRIVE, REVERSE and PARKING) and dynamically emulate GMLAN signals. It should be noted that for simplicity reasons the VIM ignores the ground signal and the park/neutral switch signals, represented at the bottom of Figure 43, in the transmission decoder logic degrading the robustness of the input redundancy verification during the decoding process. However, these decoded values control the motor controller, but do not control the gearbox. The gears are mechanically unlocked by the shift lever. This additional logic was omitted in the program for time saving reasons and does not affect the prototype functionality, however should be programmed in order to use the transmission encoder to its full potential. It would also create a safer vehicle by reducing the risk of having a mismatch between the mechanical lock released in the transmission and the condition the inverter believes the vehicle is in. In addition to the encoding safety, a mechanical latch in the shift lever box prevents the PARK position to be engaged while the vehicle is in motion to protect the gearbox. This feature is reused form the stock vehicle and is complementary to pressing the brake pedal to engage the PARK position.

Once the PRND signals are decoded, 4 TTL (transistor-transistor logic) outputs of the VIM are used to update the motor controller. Before being sent as CAN signals, the decoded shift lever signals needed to go through a second signal processing phase. After

scrutinizing the GMLAN database and considering the removed ECUs, 4 CAN messages were found to require the PRND information and all 4 were using a different encryption. Appendix J, Figure 87, illustrates the shift lever translator model making the appropriate conversion for each CAN message.

As the vehicle controls complexity increases, additional CAN messages must be transmitted between controllers leading to extra CAN buses. Sending multiple redundant CAN signals becomes a waste of bandwidth and vehicle manufacturers ought to avoid this type of inefficient situation, but it probably results because new hardware is developed piecemeal and is deployed over existing configurations.

5.4.2.2 Ignition

The ignition key has 4 positions: off, accessory (ACC), on and cranking. Each of these positions are decoded by the body control module (BCM) which activates different 12V power modes and CAN signals depending on the ignition key position. The on state voltage is connected to the motor controller ignition key input (pin 31 of Figure 39). Nevertheless, none of the BCM outputs could be deciphered to detect when the driver actually cranks the ignition to start the powertrain. To overcome this issue, an additional wire was tapped to the ignition potentiometer and connected to the vehicle integration module. The VIM reads the ignition potentiometer and feeds the converted signal to the high voltage control module. Figure 89 and Figure 90 in Appendix J shows the hardware dependant input block, debouncing logic, calibrations, converted signal and debugging probes (hooks).

5.4.3 Acceleration and Brake Pedals

The acceleration and brake pedals both have 2 potentiometers, one connected to the motor controller and the other one to the vehicle integration module. Originally, these potentiometers were all wired directly to the engine control module for torque determination and fault management. In the UOIT prototype, the fault management is done in 2 different ECUs, i.e. TIM and VIM, and the torque is only determined in the TIM. The motor controller has built-in fault management functions and will enter an

emergency mode when a fault is detected. Under normal conditions, it manages the vehicle acceleration torque demand and the regenerative braking blend. The supervisory control module of the VIM utilizes these potentiometers to dynamically emulate GMLAN signals, and also for fault detection. Under faulty conditions, it updates the emergency hardwired input of the motor controller (pin 19 of Figure 39, EMERGENCY). Therefore, the vehicle enters an emergency mode even though the gas and brake pedal potentiometer connected to the VIM are not used directly to control the motor torque. This conservative strategy was implemented to in an attempt to make the prototype being designed as safe as possible. The acceleration and brake pedal decoder is modeled in Appendix J, Figure 91, and the fault management model is shown in Figure 93.

5.4.4 Radiator Fans

In the original vehicle, the 2 fans on the front radiators were controlled by the ECM. The latter being removed from the vehicle, these controls were rerouted to the VIM which drives the stock fan controller by using its PWM input to adjust the fan speed according to the shifter lever position and the motor controller cooling needs. The intended design was to enable the fans and vary their speed based on temperature sensors. At present, however, when the vehicle is in drive or reverse, the fans are simply turned on to their maximum speed. Therefore, the control logic could be improved by implementing a more complex thermal management algorithm based on temperature feedback which would optimize energy savings. The basic solution implemented is represented in Figure 92, Appendix J.

5.4.5 Speedometer

The instrument cluster was modified to reflect the needs of an electric vehicle. The emulation of CAN signals was used to take control of the instrument cluster gauges and indicator lights. Features controlled by the VIM are:

- Speedometer
- Current gauge

- State-of-charge gauge
- GFI and vehicle ready lights
- Custom screen

The remaining stock ECUs on board the vehicle expect a speed signal, which originally emanated from the ECM and used wheel speed sensor values broadcast by the electronic brake control module (EBCM). The VIM has the 2 following options to determine the vehicle speed: process the speed sensor values or use the speed value sent by the motor controller. Both options were explored; however, experiencing conversion inaccuracies in the speed sensor value processing, lead to using the velocity output provided by the motor controller in km/h. In this case, the VIM just acts as a gateway between the UOIT bus and the HS GMLAN since no scaling or offset is required.

The 12V battery gauge was modified as a current gauge. The VIM receives the battery current from the BMS and TIM. However, the value from the BMS was judged more representative. This value is converted in the VIM to use the full range of the 12V battery gauge and allow negative values for regenerative braking. The conversion is necessary as the instrument cluster expects a CAN signal having a scaled value of 0 to 30V, while the VIM wants to send a current ranging from -200 to 200 A.

The fuel gauge is now a state-of-charge gauge on the UOIT prototype. The SOC received by the VIM from the BMS is a value between 0 and 100%. Having the same range as the original GMLAN signal, the SOC can be dynamically emulated on the HS GMLAN without any conversion.

As discussed in section 4.6.5 and 4.7.1, the GFI indication lights are using the check engine light (MIL) and the oil pressure light, for any level 2 and level 1 faults evaluated. The oil change indication light is now used as the vehicle ready light. However, the LED colour was changed to green to make the notification intuitive for the driver. The vehicle ready indication light illuminates when the high voltage control module (HVCM) enables the powertrain's traction. The vehicle ready signal is also hardwired to the

traction enable input on the motor controller (pin 33 of Figure 39). A representation of the customized faceplate of the instrument panel is shown on Figure 44. In addition to the modifications aforementioned, most of the original backlight white LEDs were replaced by light blue ones and a green one was used for the ready light. Figure 44 shows the custom faceplate of the instrument cluster. Additional figures can be found in Appendix K.

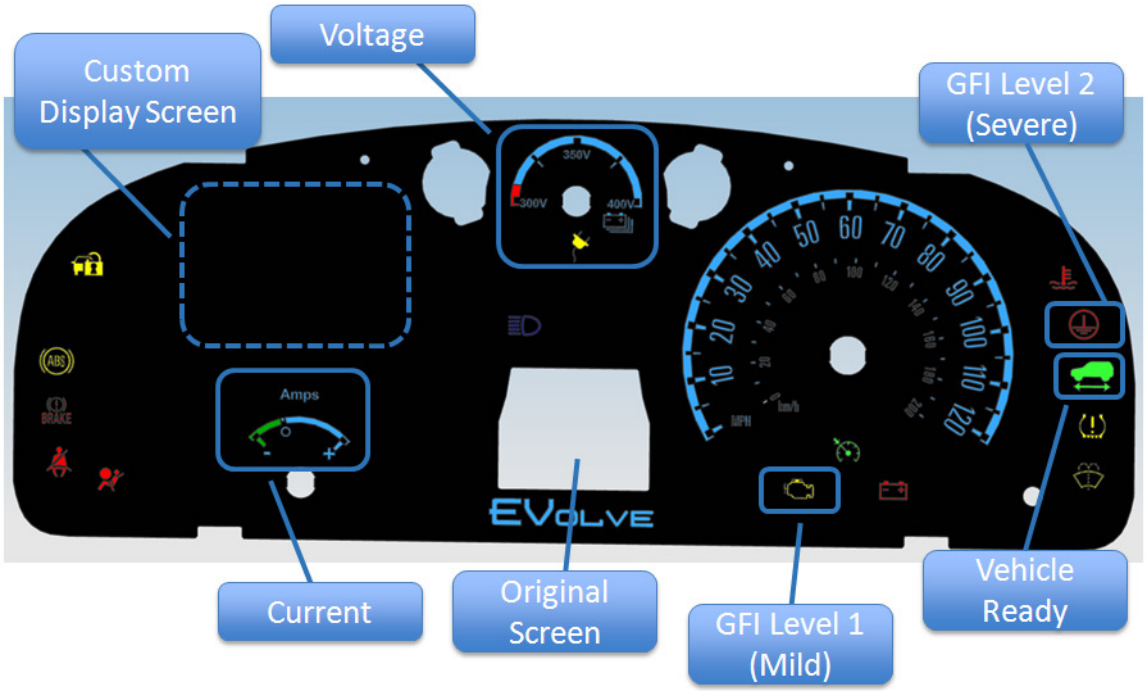


Figure 44. Custom Faceplate of the Instrument Cluster.

5.4.6 Diagnostics and Safety

5.4.6.1 Diagnostics

Several diagnostics run at different intervals in a vehicle, often at start-up. They verify the proper functioning of most sensors, actuators and controllers of the vehicle. When a diagnostic fails, a diagnostic trouble code (DTC) is generated on the CAN bus and logged. These DTCs are convenient to quickly know the status of a vehicle when serviced.

Requiring a considerable programming time, creating custom diagnostics to determine the status of the electric powertrain was not in the scope of this vehicle

conversion. However, obsolete CAN messages reserved for DTCs could have been reused to populate on the OBDII port the status of the BMS, TIM, OBCM and VIM, along with the added sensors and actuators. Each controller sends its status to the VIM able to emulate corresponding DTCs. For example, the motor controller sends its status to the VIM. The VIM, acting as a translator and a gateway, could emulate the motor controller status by reusing existing DTCs reserved for the engine.

For the added sensors and actuators not monitored or diagnosed by any of the added ECUs, complete diagnostics would have had to be programmed in the VIM. For the purpose of this prototype, diagnostics and DTC emulation is not managed and integrated by the VIM. The status of each added controller is available for troubleshooting using CAN on the MotoTune interface of the VIM and RS232 communication on controller specific software directly wired to each of them. For convenience, the connectors required to troubleshoot these ECUs were gathered in the glove compartment.

5.4.6.2 Safety

Having left unmodified the sensing and diagnostic module (SDM) and stock crash sensor, along with the single-wire GMLAN, the airbags of this prototype will deploy in the event of a crash. As explained in Section 4.6.5, an additional inertia switch was added and is used as an interlock to disable the high voltage powertrain when triggered. The airbag CAN messages could also have been used by the VIM as redundancy to this additional crash sensor, but was not programmed due to time constraint.

5.5 HVCM

The high voltage control module modes, which were detailed when discussing the powertrain control strategy in section 4.4.1, are programmed in the VIM mainly using a Simulink development tool named Stateflow. Figure 45 shows the high level inputs and outputs of the HVCM subsystem of the VIM Simulink model. The detailed Simulink model and Stateflow can be found in Appendix J.

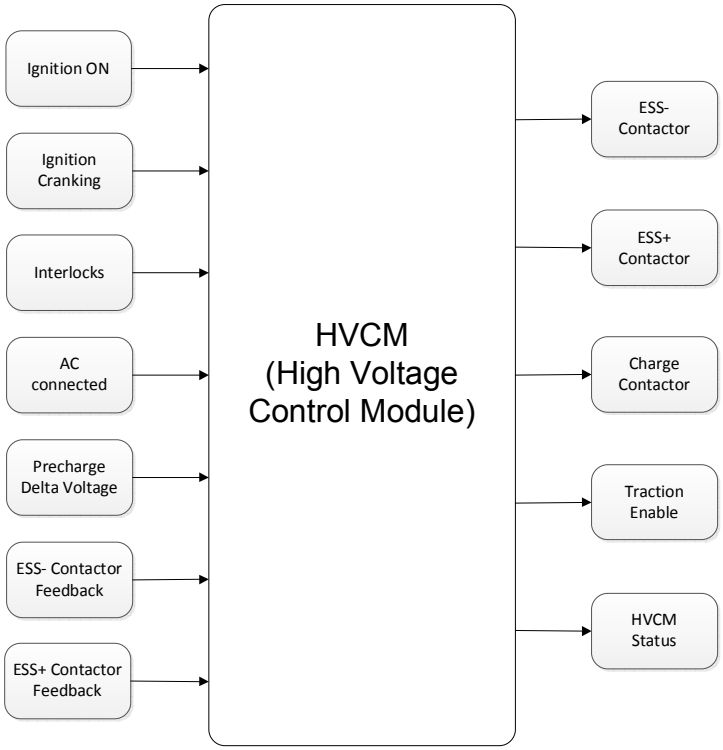


Figure 45. HVCM I/Os.

5.5.1 Inputs

In Stateflow, the inputs are used as conditions to enter and exit states, whereas the outputs are action variables updating the high voltage hardware of the powertrain. It is important to distinguish the difference between the ignition ON (ignition key) and ignition cranking (start pulse) signals. The ignition ON signal is 1 when the key is either in the ON or cranking positions, while the ignition cranking signal is only 1 when the key is held in the cranking position. As explained in section 4.6.4, the interlock signal is generated using an "and" gate employing the vehicle's hardwired interlocks as inputs. The AC connect signal equals 1 when either high voltage is detected at the charging inlet or the SAE J1772 charging connector is physically plugged-in. The precharge delta voltage signal turns to 1 after the precharge is completed, i.e. when the voltage difference between the voltage read at the BMS and the one measured at the motor controller is less than 10 volts. Interlock ESS- and ESS+ are feedback signals from the vehicle's main contactors indicating whether they are opened or closed.

5.5.2 Outputs

The ESS-, ESS+ and charge contactor outputs are used to ground the negative side of each contactor's coil using low side outputs of the MotoTron. These outputs are rated for sinking up to 7 A which is enough to sustain the current peak created by the coils of these high voltage contactors. As discussed previously, the traction enable signal is fed to the motor controller and is activated when the high voltage powertrain is ready to power the motor, i.e. when the ESS- and ESS+ contactors close after the key was cranked and the high voltage precharge circuit is complete.

5.6 CAN Communication

5.6.1 Emulated CAN Messages

Among the 8 main ECUs removed from the vehicle, 5 of these were still expected to be sending CAN messages either on GMLAN HS or PTE by the remaining controllers. A summary of the emulated ECUs is provided in Table 47. ECUs expecting inputs were determined by looking in the CAN database provided by General Motors checking if the remaining controllers were normally receiving messages from the removed ones.

Table 47. Emulated ECUs.

Removed ECUs	Expected CAN messages	
	GMLAN HS	GMLAN PTE
APM		**X**
BPCM	**X**	
ECM	**X**	**X**
FSCM		
TCM	**X**	
HCP	**X**	**X**
MCPA		
MCPB		

In an attempt to avoid as many issues as possible related to removing ECUs, all CAN messages typically transmitted by the remaining ECUs were broadcast, unless the original ECU expecting the message was removed from the vehicle. Most emulated signals are sent statically meaning that a fixed value for each signal was chosen by scrutinizing the original dynamic vehicle CAN bus traffic, and is now constantly being transmitted. Common examples of these signals are validity and fault active bits being emulated to 0, letting other ECUs believe everything is working properly. However, some CAN signals need to be updated frequently and are referred to as dynamic ones. The list of the dynamically emulated CAN signals can be found in Table 48.

Table 48. Dynamically Emulated CAN Signals.

ECUs	Signals	New function
BPCM	HV battery voltage	
	State-of-charge	
	Temperature	
	Auxiliary voltage limits (13.6V)	
	HV battery current	
ECM	Engine run active	
	Vehicle speed	
	Accelerator actual position	
	Brake pedal status	
	Transmission shift lever position	
	Engine oil change indication on	Ready light
	Check engine indication on (MIL)	GFI Level 1 light
	Engine oil pressure low indication on	GFI Level 2 light
	Fuel level percentage	HV battery voltage
	12V battery gauge	HV battery current
	Cruise Control (off)*	
	Air conditioning compressor command (off)*	
	Air conditioning refrigerant high side fluid pressure (off)*	
HCP	Transmission shift lever position	
TCM	Transmission shift lever position	

* The dynamic emulation of the cruise control and air conditioning was intended, but was never implemented. At present, static emulation keeps these 2 functionalities disabled.

5.6.2 UOIT Added ECUs

As shown by Figure 34 in Chapter 4, the UOIT CAN bus physically interconnects the battery management system (BMS), on-board battery module (OBCM), traction inverter module (TIM), datalogger and instrument cluster screen (ICS) to the vehicle integration module (VIM). Each of these controllers communicates directly only with the VIM. Even though they are on the same network, they do not exchange CAN messages

directly, but indirectly through the VIM acting as a bridge between them. The main reason for this resides in the control strategy chosen for the 3 main powertrain controllers, i.e. BMS, OBCM and TIM. The best solution was to use the VIM to integrate them individually to the vehicle architecture and use RS232 communication to define the initial settings of the devices. The motor controller is the only added ECU configured to receive as well as transmit CAN messages. The BMS and OBCM were configured to only broadcast messages, while the datalogger and ICS were set to only receive messages. The next sub-sections describe in detail the communication between the VIM and each of these controllers. Also, a complete list defining their CAN messages can be found in Appendix L.

5.6.2.1 Motor Controller

The motor controller was the only main powertrain ECU to receive CAN messages as it was only able to function as desired with the CAN communication enabled and receiving periodic CAN commands. Three CAN messages need to be sent at 135 ms intervals to the motor controller for proper functioning. The first one specifies on the maximum voltage and current discharge rate while driving. The second one focuses on regenerative braking and motor activation using these 4 signals: maximum regenerative voltage and current, regenerative braking enable and traction enable. The latter 2 signals are also provided by hardwired inputs. The arbitration between the CAN messages and the hardwired inputs is done as an OR gate, i.e. as soon as one of the values, whether CAN or hardwired, is a logic 1 the motor controller consider the signal to be a logic 1. More testing needs to be done on the inverter to better understand the arbitration between the CAN messages and the discharge/charge limits programmed using the RS232 communication. However, it is seems logical to the author that the CAN messages would have priority, since they are dynamically sent. The third message defines the maximum acceptable voltage while using an AC charger. For the UOIT prototype, the maximum regenerative voltage and the maximum charge voltage were set to the same value, 395V which is 4.2 V / cell.

Two CAN messages are broadcast at a 12 ms rate by the motor controller. One contains boolean variables for the status of all the inputs, outputs and alarms present, whereas the other one provides the input battery voltage, vehicle speed, motor speed and motor torque.

5.6.2.2 Battery Management System

Numerous CAN messages are sent by the BMS to the VIM, from a battery metrics summary, to detailed individual cell information. A status and problem frame is also sent along with commands for a charger. Due to the presence of 5 on-board chargers, the built-in charging commands broadcast on the network by the BMS could not be used, and a custom charging strategy had to be employed instead. Furthermore, operational commands can be received by the master, but are not used since the BMS module's basic settings were configured manually via RS232 communication.

The summary message periodically broadcast by the master gives an overview of the battery by providing the:

- State-of-charge
- State-of-health
- Maximum temperature
- Maximum cell voltage
- Minimum cell voltage

In addition, the delta cell voltage between the maximum cell and the minimum cell was computed by the VIM. It is a useful indicator to determine how well balanced the battery pack is. After a request by the competition organizers, this value was also recorded with the datalogger. Even though the overall current going in and out of the battery is not present in this summary, it is sent by the master on a dedicated CAN message. Unfortunately, the total battery voltage is simply not provided by this BMS. Being an important metric to monitor when using a battery pack, the VIM computes the battery voltage by adding the voltage of every single lithium cell. To accomplish this computation, the VIM needs to read 28 additional messages from the UOIT CAN bus.

28 different messages would typically require the same quantity of CAN IDs. However, the BMS manufacturer decided to use the first byte, of the 8 data bytes available in CAN messages, as a sub-ID to minimize the use of IDs. In other words, the first byte of the payload is used to identify the content of the 7 remaining data bytes. Each BMS board sends information on a predefined CAN ID, for a total of 7 different identifiers, and uses sub-IDs to differentiate messages. It should be noted that the cell voltages are sent using 2 bytes, and only 7 bytes remain in each message. Therefore, some cell voltages are split between 2 CAN messages and need to be concatenated by the VIM when received.

In theory when concatenating to a single variable, 2 signals of 1 byte received from 2 different CAN messages, it is recommended to verify that both signals contain a new value before overwriting both bytes of the existing variable. Otherwise, a mismatch could occur between the high and low byte of the merged variable. However, considering the order of voltage magnitude of the high byte, it was determined to change slowly enough compared to the low byte of the same variable that such a verification could be neglected.

The temperature values, measured by the NTC thermistors (negative temperature coefficient) inside the battery module and connected to the BMS boards, are transmitted using the same sub-ID system. Each BMS module also sends a status and problem message. The information expressed uses the boolean flags listed below.

Status flags:

- Start-up
- Control charge
- Control discharge
- Equalize cells
- Monitor battery
- Do active control
- Cooling on
- Heating on
- Do shut down

Problem flags:

- Overvoltage problem
- Undervoltage problem
- Temperature too high for charging
- Temperature too high for discharging
- Hardware overvoltage problem
- Hardware undervoltage problem
- Charge current too high
- CAN response failure
- CAN receiver buffer overflow
- Configuration error (may be set during start-up)
- Doing start-up (is always on during start-up)

5.6.2.3 On-Board Charger

Similar to the BMS, even though it can receive messages such as commands from a BMS, it is not part of the control strategy and the on-board charger only transmits information. Since most of the troubleshooting was done manually using a RS232 communication, only 1 message from the exhaustive list of CAN messages sent by the charger, at a 100 ms rate, was used. It provides the AC voltage and current coming into the charger and the DC voltage and current coming out of the charger, i.e. charging the lithium battery. Although it was intended to program each of the 5 chargers to communicate with the VIM, only the one shutting off last in the cascade sequence, as explained in Section 4.5.2, ended up being programmed due to competition time constraints.

5.6.2.4 Datalogger and Instrument Cluster Screen

The datalogger and the ICS are grouped under the same sub-section, since the ICS only displays some of the datalogger's recorded CAN messages and does not require any additional messages from the VIM.

For the EcoCAR competition, it was mandatory for each team to provide a log of specific signals recorded at 1 Hz after each competition event. Having designed and built their own high voltage battery pack, the UOIT EcoCAR team was asked by the organizers to record 3 signals in addition to those defined in the rules. The list of recorded signals is:

- Motor temperature (ºC)
- Vehicle speed (km/h)
- Motor speed (rpm)
- Motor torque (Nm)
- Acceleration pedal position (%)
- Brake pedal (on/off)
- Transmission shift lever position (P, R, N or D)
- Battery current (A)
- Battery voltage (V)
- Battery temperature (ºC)
- State-of-charge (%)
- Battery maximum cell voltage (mV) – (only UOIT)
- Battery minimum cell voltage (mV) – (only UOIT)
- Battery maximum delta cell voltage (mV) – (only UOIT)

Here's the list of signals also read and displayed by the instrument cluster screen:

- Transmission shift lever position (P, R, N or D)
- State-of-charge (%)
- Battery current (A)
- Battery voltage (V)

5.7 Summary

As indicated in the present chapter, the vehicle integration module, a MotoTron controller, plays a key role in the electrical and control system integration strategy. A brief overview of the controller hardware and software was given. Its various features were identified and described, such as encoders, decoders, high voltage control module, CAN signal emulator and gateway function. The final results of UOIT's electrified vehicle prototype are summarized in the next chapter.

Chapter 6

Tests and Results

6.1 Introduction

The previous chapter explained the functionalities of the vehicle integration module. The objective of the present chapter is to highlight the vehicle results obtained by the UOIT EcoCAR team with a focus on the author's accomplishments. The validation, calibration and testing methodology, followed by the UOIT EcoCAR team general results, are presented with the individual results provided by the author.

6.2 Validation, Calibration and Testing

6.2.1 Controls Development Process

The vehicle integration module (VIM) is able to interact with the battery management system, the motor controller and the charger. The approach used to integrate each new ECU to the vehicle was to prove that the component was communicating and reacting as expected with the VIM and via its RS232 interface in normal and critical situations on a test bench. Tests were conducted on all inputs and outputs of the component to better understand the subtleties of each feature and confirm integrity. The motor and its controller were tested on the bench using a 400 V lead acid battery pack and an in-house control panel to manage the motor controller inputs. Each module of the BMS was tested and configured using a smaller capacity Li-ion battery pack borrowed from a different project. At this stage of the conversion, the vehicle's Li-ion battery was not fully assembled, and having safer and more stable battery chemistry, the 400 V lead acid battery was used to test the charger. The VIM inputs and outputs were verified mainly with a multimeter, switches and contactors, whereas the CAN ports were tested by establishing communication with the main powertrain controllers.

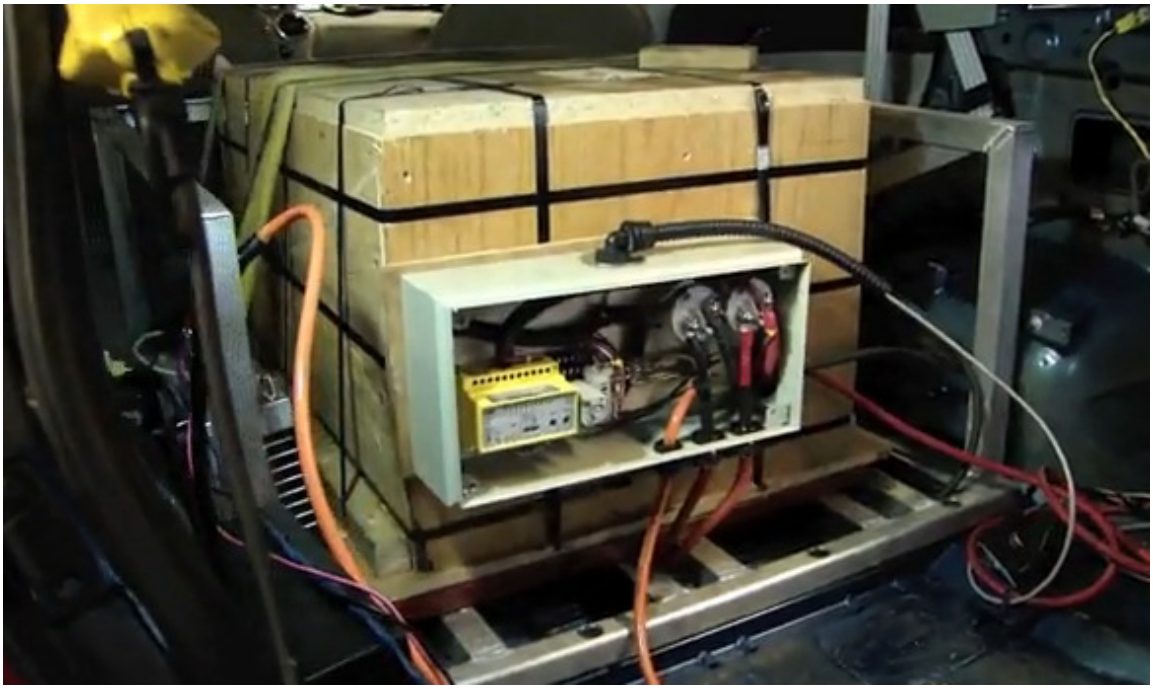


Figure 46. Temporary lead-acid battery box.

Once the independent testing of each module was completed, they were integrated and tested two by two on the test bench. For instance, the main powertrain components were tested as follow: the BMS and the VIM, then the charger and the VIM, and finally the TIM and the VIM. Methodically, each interacting set of controllers were tested 2 by 2, then 3 by 3 and so forth. After numerous tests, the system was ready to be migrated systematically into the vehicle.

The migration from the test bench to the vehicle was initiated by integrating the VIM. Two of the VIM's main tasks, once installed in the vehicle, were to act as a gateway between the high speed and powertrain GMLAN buses (the latter being also used as a custom CAN bus), plus to emulate all the removed modules, i.e. Engine Control Module (ECM), Fuel System Control Module (FSCM), Battery Pack Control Module (BPCM), Traction Power Inverter Module (TPIM) and Accessory Power Module (APM). Unfortunately, the emulation of the removed ECUs could not be tested outside of the vehicle. Therefore, the first tests performed were to connect the VIM to the HS and the PTE GMLAN buses to establish communication with the vehicle, and

thereafter verifying if the main car modules would still operate properly after the systematic removal of ECUs.

It was difficult to know exactly what to expect, since the emulation program of the VIM does not respond exactly like the car modules in every situation. As mentioned previously, some CAN messages are emulated dynamically, whereas others are only static. There are so many possible scenarios that it is very difficult to determine with certainty whether a CAN signal needs to be static or dynamic. However, the numerous hours studying the GMLAN dictionary and logs of the unmodified CAN buses paid off, since the car turned on without any indication of warnings on the instrument cluster the first time. Also, the presence of diagnostic trouble codes (DTC) was checked by the team's GM mentor using a NeoVI and the GMLAN database. Again, no unexpected trouble codes were found.

The shift lever, acceleration and brake pedals were rerouted to the VIM and dynamically emulated with success. Proper emulation of the brake pedal and shift lever was shown by functioning brake and reverse lights, along with the dynamic update of the PRND position on the instrument cluster display. Power steering was enabled by the emulation of the shift lever signal and a few other specific signals. The power steering also worked immediately. Unfortunately, a modification made under competition pressure in the VIM programming during the second year competition events disabled that functionality for the following few months until the controls team spent an evening troubleshooting the problem. It is also important to mention that the turning signals and hazards were not impacted by the conversion. The car ready and GFI lights were implemented in one of the VIM software release during the last year of design updates.

By converting the vehicle into an EV a number of control modules were no longer required for standard vehicle operation and needed to be removed. The fuel system control module (FSCM) was judged to be the easiest module to emulate, since it only broadcast CAN messages and did not receive any. It was therefore selected to be the first ECU to be removed from the vehicle and emulated by the VIM. This emulation was

required to ensure no error codes were passed into the vehicle's ECUs. Through successful emulation and removal of the FSCM, the remaining modules were also removed, i.e. the stock battery pack control module (BPCM) followed by the engine assembly where the TPIM, ECM, TCM and APM were mounted.

The second key step for the migration from the test bench to the vehicle was to hook the temporary 400 V lead acid battery pack (without BMS) to the accelerator, brake pedal, motor controller, and the MotoTron controller (VIM). Also, for this step, the motor and its controller were connected and installed in the engine bay. At that point, the VIM was integrated to the vehicle, but only used as an emulator and a diagnostic tool via MotoTune. There was no main high voltage controller or distribution box. The car ignition was still connected to the stock ECUs, but not to the VIM or the other added powertrain controllers. The in-house control panel was used to drive the motor controller inputs, such as the acceleration and brake pedals, ignition, traction enable and regenerative braking enable. With the vehicle off the ground, this is when the team saw the front wheels of their electric vehicle prototype spin for the very first time. A few minutes after, the acceleration and brake pedals were routed to the terminal blocks while the other signals were still originating from the control panel.

Figure 47. Set-up when the wheels spun for the first time.

As mentioned previously, most of the components added to the vehicle functioned properly virtually the first time when migrating from the test bench to the vehicle. This can be attributed to an efficient test bench connected to the VIM and was realized during previous months by following a meticulous debugging process. Debugging on a complete test bench is much more efficient and easier to follow, because one can verify each module individually and analyze its reaction to known stimuli as opposed to trying to debug it in a complex network of controllers. The methodology employed has helped improve the control system and ensured proper functioning and communications between the vehicle's major components: Vehicle Integration Module (VIM), Battery Management System (BMS), On-Board Charger Module (OBCM) and Traction Inverter Module (TIM).

6.2.2 Performance Testing at UOIT

The previous section described the process followed and tests performed on the prototype before and during the integration. This section focuses on some of the tests realized with the integrated vehicle.

The methodology used to safely test the vehicle was as follows: the motor tuning and controls were tested with the vehicle off the ground, thereafter short stints of mild driving, acceleration and braking were done, followed by longer tests at higher speeds, maximum power output and sustained duration running. For safety, the Vehicle Integration Module (VIM) was generally re-flashed with the vehicle off the ground to avoid any unintended acceleration, and preliminary verifications performed then.

The motor tuning was one of the first powertrain calibrating attempts done on the integrated vehicle. Having it on a hoist with the wheels free spinning, the motor was initially tested with the same in-house control panel used for preliminary verification and debugging on the test bench, before controlling it via the VIM. The team ran the inverter's built-in auto-tuning function a few times until successful completion of the test sequence. After connecting the acceleration pedal to the inverter, instead of using the potentiometer of the in-house control panel, the vehicle was drivable. Unfortunately, the

auto-tuning feature allowed a lot of leeway in the setup and two S10-EV motors were eventually damaged in the following months. Hours were then spent trying to understand the poorly translated and utterly incomplete inverter manual by modifying each major control parameter individually. Section 6.2.4.1 explains how the final calibrations were obtained during dyno-testing.

Once the creep torque calibration variable was found among the several parameters under study, a small acceleration was tuned to provide an intuitive drive feel before pressing the gas pedal while engaging the drive or reverse gear. This simulates a torque present on automatic transmission internal combustion engine vehicles, making the drivability of this electric vehicle a little more intuitive.

Most of the dynamic testing was performed in the shop's parking lot, such as short stints of mild driving, braking and even hard acceleration as the team's confidence in the hardware increased. Thereafter, the vehicle was driven on public roads near the garage for longer periods to test the vehicle at higher power demand. However, 0-60 mph accelerations could not be evaluated accurately in the limited sized parking lot, or safely on public roads, and were left for the end of the third year during performance testing at the EPA lab and the final competition.

6.2.3 Performance Testing at Year 2 Competition

After a few days of last minute modifications, the full-function electric vehicle of the UOIT team passed the qualifying static and dynamic safety technical inspections and was ready to compete in the dynamic events of that second year's competition. The event took place at the GM Yuma Proving Ground, a facility that provided a great opportunity to realize dynamic testing in a controlled and safe environment. Furthermore, extra monitoring instruments were added to the vehicle, provided by the organizers, i.e. a GPS antenna, a display gauge and a battery current sensor interconnected to a small MotoTron controller via a dedicated CAN bus. The list of dynamic events performed is as follow:

- Acceleration: 0-60 mph
- Acceleration: 50-70 mph

- Stopping Distance: 60-0 mph
- AVL Drive (drive quality test)
- Autocross (serpentine cone-lined course)
- Towing (1500 lbs load up a 3.5% grade at 45 mph for 15 miles)
- Lane Change @ vehicle safe handling limits

Even though a temporary Li-ion battery pack was used and installed in the cargo space of the vehicle as shown in Figure 48, the control strategy and most of the powertrain devices in place were that from the final design. Therefore, the results of these dynamic events gave the team a good preview of the performance achievable by the final version of their prototype.

Figure 48. Temporary Li-ion battery pack.

6.2.4 Performance Testing at the Environmental Protection Agency

In March 2011, the UOIT EcoCAR team and the competing universities were given the opportunity to test their prototypes for 3 days at the National Vehicle and Fuel Emission Laboratory (EPA). Although the general test methodology followed at the EPA facility was similar in some ways to what was normally done at the university garage, the additional instrumentation, dynamometers and certified drivers provided the unique advantage of testing in a controlled environment. The UOIT test schedule at EPA is detailed in Table 49.

Table 49. Test Schedule at EPA.

Day 1	Motor Controller Calibration
	Charge Monitoring (to ~ 100% SOC)
Day 2	Range test (without regen.) (to ~ 0% SOC)
	Charge Monitoring (~ 0% to ~ 30% SOC)
Day 3	Regenerative Braking Calibration
	Battery Current Monitoring

6.2.4.1 Day 1

Initially, the vehicle underwent experiments on the dynamometer with different motor controller tuning parameters that had been initially set at the university garage. Different steady state speed power consumption and acceleration runs were performed for each parameter under study. After analysis, the team converged on a set of motor tuning calibrations for subsequent tests. Also, a number of full throttle runs were performed to evaluate the vehicle acceleration performance.

Once the inverter calibration was completed, the vehicle was unstrapped from the dynamometer and was brought to the charging station where the battery pack was charged in preparation for the full depletion test occurring the next day. Due to some remaining cell imbalance, the battery pack was charged to an estimated state of charge (SOC) of 96%, instead of 100%. The AC input voltage, current and power of the charger,

along with their DC output values were measured and logged to study the charger efficiency, total battery capacity and charge profile.

6.2.4.2 Day 2

The second day, a full discharge to approximately 0% of the pack capacity was sought while running a variety of drive schedules. To accomplish this level of discharge, different drive cycles and steady speed driving were selected by EPA and UOIT team in consultation. The EPA crew was mainly interested in gauging performance nominally at 100%, 80%, 50% and 20% states of charge. The UOIT team did not expend effort in developing unique drive cycles, rather tried to run the standard EPA cycles as much as they were allowed. The US06 was not permitted, although it would have been indicative and helpful to test the vehicle on a more demanding drive cycle and prove the robustness of the vehicle. However, the team was allowed to run the vehicle at sustained speeds of 60 mph. The schedule devised to perform this battery depletion test is shown in Table 50.

Table 50. Drive Schedule Sequence.

Sequence	Drive Schedule
# 1	UDDS #1
# 2	UDDS #2
# 3	UDDS #3
# 4	UDDS #4
# 5	60 mph (until 65% SOC)
# 6	UDDS #5
# 7	Highway FTP #1
# 8	Highway FTP #2
# 9	UDDS #6
# 10	60 mph (until 30% SOC)
# 11	UDDS #7
# 12	UDDS #8
# 13	60 mph (until depletion)

Similar to the day prior, the battery was again charged upon dynamic test completion. Due to time constraints caused by the interdiction of charging overnight, the vehicle was only charged to approximately 30% SOC.

6.2.4.3 Day 3

For the last day of testing, an additional current clamp was installed to verify earlier results, but no significant difference was found. However, the focus on that 3rd day of testing was on calibrating the regenerative braking parameters to obtain a smooth blending with the hydraulic brake system and a smooth deceleration while coasting similar to an internal combustion engine vehicle.

To achieve a natural throttle response the team engaged the help of the professional EPA driver in tuning a progressive regenerative braking level on the fly to a point where the car naturally coasts to a stop as the driver releases the throttle and with very light application of the brakes, i.e. before the hydraulics activate. This makes the car almost drive itself down the UDDS "hills" to a stop, giving a very natural braking performance while recharging the battery.

The testing program during day 3 ended about 1 hour prematurely while the vehicle was being run with more aggressive REGEN braking levels. The ground fault indicator (GFI) went off while the car was holding during a 30 second stop sign pause about halfway through a UDDS cycle. The subsequently scheduled highway cycle with REGEN, the last in the test sequence planned for that day, was thus not run. No quick fix was found to trace the problem, so testing ended.

6.2.5 Vehicle Integration at Year 3 Competition

Unfortunately, for several different technical reasons, the UOIT team did not successfully complete the safety scrutineering at the final EcoCAR competition, thus was not allowed to compete in any of the dynamic events. However, during the week of

competition lots of work was accomplished on the vehicle. As an example, with 2 new battery modules just finished and assembled overnight, prior to shipping the vehicle to the competition at Milford Proving Ground, the team had to spent most of the competition integrating them to the vehicle. Beside the battery, the integration involved a more powerful MotoTron controller, a junction box interconnecting most of the controls wires for the conversion and a new high voltage distribution box.

Even though the high voltage control module operation was programmed in the new MotoTron controller, and had been tested first through Simulink simulation and then on a test bench, it had never been operated with the new high voltage distribution box. Since the controls connected to the junction box needed to be functional first, debugging of the HV distribution box was kept for last. Therefore, a lot of work was spent routing signals from one terminal block to another, along with the BMS installation and configuration. This required testing and replacing some cell inline fuses needed for cell voltage monitoring, connecting and routing the fuse strip board cables to the BMS, etc.

No dynamic performance tests were achieved during the final competition, since the vehicle was not driven. The UOIT team ran their vehicle for the last time after the EPA event in road testing of the motor controller's tuning parameters prior to the final competition event, and before the temporary Li-ion battery pack situated in the vehicle cargo space was disassembled and rebuilt into the intended battery pack design.

6.3 EcoCAR Project Results

6.3.1 Competition Result Overview

UOIT placed 6th at the second year competition and was one of the few teams with a running vehicle. Similarly, UOIT was one of the few teams having their prototype ready for dynamometer testing and taking advantage of the sophisticated EPA facility. Extra dynamometer time was even given to the team for a lack of usage from the other universities. Unfortunately, a very risky and lengthy step remained to be achieved: the

transition from the temporary Li-ion battery pack stored in the trunk of the SUV to the final battery packs, one located underneath the vehicle and the other in the engine bay.

Delays in the assembling process combined with the short period of time between the testing at EPA and the final competition resulted in shipping an unfinished vehicle to the event. Numerous hours were spent at the competition at the Milford Proving Ground finishing the two battery pack modules, leaving minimal time for troubleshooting the newly made high voltage distribution module and its high voltage controller, the vehicle integration module.

When the organizers gave the ultimatum to the team, the ignition of their newly modified vehicle was turned on prior to an exhaustive testing of each component in the new high voltage system, resulting in the explosive failure of the main precharge resistor. Therefore, the UOIT full function electric vehicle prototype did not run during the very last competition of EcoCAR The NeXt Challenge. Not competing in the dynamic events severely affected the UOIT score card. After the static events taking place in Washington DC, the team ended up finishing in 13^{th} place of 15 entries.

6.3.2 Environmental Protection Agency Testing Results

6.3.2.1 Day 1

On the first day of testing, steady state and acceleration runs were performed with different inverter tunings. It was discovered that the vehicle exhibited a surging at around 35 mph, i.e. a noticeable torque peak seen by the vehicle occupants and the supervisory team. This caused the wheels to slip on the dynamometer rolls. 0-60 mph data indicated acceleration times of around 19 seconds, possibly faster but roll slippage is difficult to quantify. A key conclusion drawn from the testing was that the supply voltage from the temporary battery pack was inadequate for the inverter which was struggling to get much past 200 V_{ac}. The inverter requires at least 360 V_{dc} on the DC mains to attain the motor's rated 240 V_{RMS}. For this reason, the number of cells in the final pack was raised to that planned in the initial design. That fact had long been known and was designed into the final pack configuration more than a year prior to these tests.

During the previous year, the team damaged two S10-EV motors overheating them while doing on-road testing. To avoid damaging a third motor, an infrared temperature sensor was added to the motor housing in order to measuring the actual motor temperature in addition to the windings. Therefore, the motor temperature was closely monitored while performing the inverter calibration at EPA. To the team's delight, the motor overheating issues struggled with throughout the previous year appear to have been overcome through a fine tuning of the motor controller calibrations.

6.3.2.2 Day 2 and Day 3

A major concern of the EcoCAR organizers was that UOIT's full-function electric vehicle would not meet the competition range requirement. A complete depletion of the 74.6 kWh temporary Li-ion battery was therefore undertaken on the second day at EPA. For this test, the regenerative braking was deactivated, since its calibration was only planned for the next day. Table 51 shows the detailed results of each drive schedule, i.e. the number of miles driven, cumulative energy, energy for a schedule, energy consumption per mile, the extrapolated range and the estimated state-of-charge (SOC). Table 52 summarizes the results found in Table 51 and displays an extrapolated range of 222.7 miles for one charge (96% to 4.8% SOC) with an enabling regenerative braking and with the temporary Li-ion battery pack. It should be observed that the extrapolated range is more conservative than the total miles driven of 232.967 miles.

On the third day, regenerative braking was tested and calibrated. It should be noted that a second current clamp was added for more accuracy. As explained in Section 6.2.4.3 and reflected in Table 54, a ground fault went off during the UDDS cycle #10 and the subsequent highway cycle with REGEN activated could not be run. Therefore, a combined estimation of UDDS #9 and #10 was made and can be found on the last row of Table 53. The REGEN calibrations were changed on the fly within a single UDDS cycle. Therefore, the energy consumption in Table 53 for UDDS #9 and 10 is a mix of different calibrations of REGEN. However, these calibrations mainly affected the drivability, thus the energy savings achieved from one calibration to another were similar. That is why

the results of UDDS #9 and #10 were still compared to the energy consumption of the cycles ran without REGEN the day before. An estimation of the energy consumption for the final parameters is also present in the aforementioned table.

In Table 54, the energy savings with REGEN activated was estimated at 8% for the highway, and 0% for the steady speed of 60 mph since no brake are applied. In summary, activating regenerative braking increases the extrapolated range by 44.8 km for a maximum range of 402 km, as stated in Table 55. Figure 49 illustrates some results of the energy regenerated. One should note the smooth transition in REGEN current every time the vehicle comes to a stop.

Table 51. EPA Energy Consumption Results – Battery Depletion.

	Drive Schedule	Miles Driven	Cumulative Energy Use (KWh)	Energy for Schedule (KWh)	Consumption (Wh/mi)	Extrapolated Range (mi)	Estimated SOC %	
Battery Depletion	UDDS #1	7.443	2.874	2.874	386.1	198.18	SOC @ Start:	96.0
	UDDS #2	7.450	5.688	2.814	377.7	202.60		92.6
	UDDS #3	7.437	8.440	2.752	370.0	206.80		89.2
	UDDS #4	7.442	11.260	2.820	378.9	201.95		85.9
	60 mph	46.118	26.236	14.976	324.7	235.66		82.6
	UDDS #5	7.430	29.020	2.784	374.7	204.23		64.7
	Highway	10.243	32.032	3.012	294.1	260.24		61.4
	Highway	10.244	34.984	2.952	288.2	265.56		57.8
	UDDS #6	7.435	37.728	2.744	369.1	207.35		54.3
	60 mph	60.533	56.663	18.935	312.8	244.64		51.0
	UDDS #7	7.433	59.324	2.661	358.0	213.76		28.5
	UDDS #8	7.407	62.039	2.715	366.5	208.78		25.3
	60 mph	46.352	76.526	14.487	312.5	244.85		22.1
	Total miles driven	232.967		Average:	347.2	222.7	SOC @ End:	4.8

Table 52. EPA Energy Consumption Results – Battery Depletion Summary.

	Averaged Drive Schedule	Consumption (Wh/mi)	Extrapolated Range (mi)	Estimated ΔSOC %
W/O REGEN	UDDS	372.6	205.36	
	Highway	291.1	262.88	
	60 mph	316.3	241.93	
	Battery Depletion Schedule	347.2	222.7	91.2

Table 53. EPA Energy Consumption Results – Regenerative Braking Activated.

	Drive Schedule	Miles Driven	Cumulative Energy Use (KWh)	Energy for Schedule (KWh)	Consumption (Wh/mi)	Extrapolated Range (mi)	Estimated SOC % (Start of Schedule)
Clamp #1	UDDS #9	7.420	2.639	2.415	325.5	235.12	~30
	Final tune regen		2.788				
	UDDS #10 @ GFI	3.797	3.850	1.063	279.9	273.45	~27
	UDDS #9/10 estimate	7.420		2.117	285.3	268.22	
Clamp #2	UDDS #9	7.420	2.723	2.435	335.4	233.24	~30
	Final tune regen		2.828				
	UDDS #10 @ GFI	3.797	3.896	1.068	281.2	272.09	~27
	UDDS #9/10 estimate	7.420		2.133	287.5	266.18	
Avg.	UDDS #9/10 estimate	7.420		2.125	286.4	267.2	

Table 54. EPA Energy Consumption Results – Regenerative Braking Activated Summary.

	Estimated Drive Schedule		Consumption (Wh/mi)	Extrapolated Range (mi)	Estimated SOC % (Start of Schedule)
REGEN	UDDS	(with REGEN averaged from both clamps)	286.4	267.20	
	Highway	(consumption with regen (-8%))	267.8	285.73	
	60 mph		316.3	241.93	
	Battery Depletion Schedule	(with regen engaged)	305.3	250.7	

Table 55. Influence of Regenerative Braking on the Range.

	w/o REGEN	w REGEN
Range	356.3 km (222.7 mi)	401.1 km (250.7 mi)

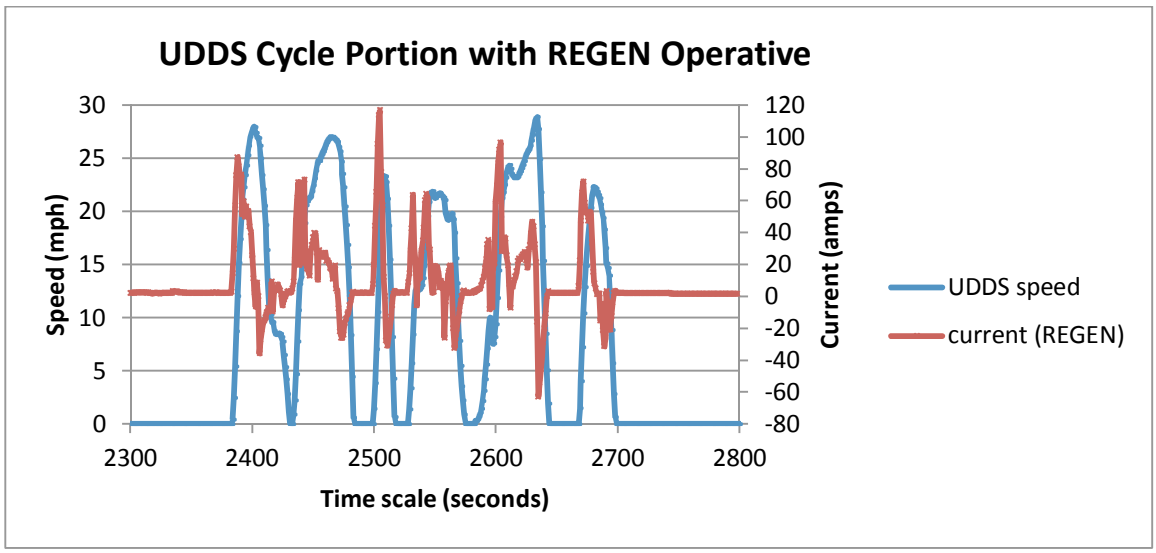


Figure 49. UDDS Cycle Portion with REGEN Activated.

6.3.2.3 Ground Fault Interrupt Issue

Upon inspection back at our shop it turned out the main fuse had blown, so there was no ground fault present rather the indicator was triggered by a loss on continuity on the battery circuit. This feature is a little known function of the ground fault interrupt unit used from Bender. The blown fuse was traced to a shorted IGBT leg in the inverter. Since the vehicle was not under any high demand situation through the UDDS cycle, it is theorized that increased high voltage spikes in REGEN caused the IGBT to avalanche. To remediate this problem, the high voltage bus capacitance was increased, the high voltage leads were re-routed for reduced inductance, and the inverter signal lines were isolated from ground noise by floating its supply and opto-isolating signal lines. Noise on the vehicle ground had been a consistent problem and may have caused false triggering of the IGBT taking away all dead-time and causing shoot-through. Figure 50 shows the inverter apart with the shorted IGBT on the bench.

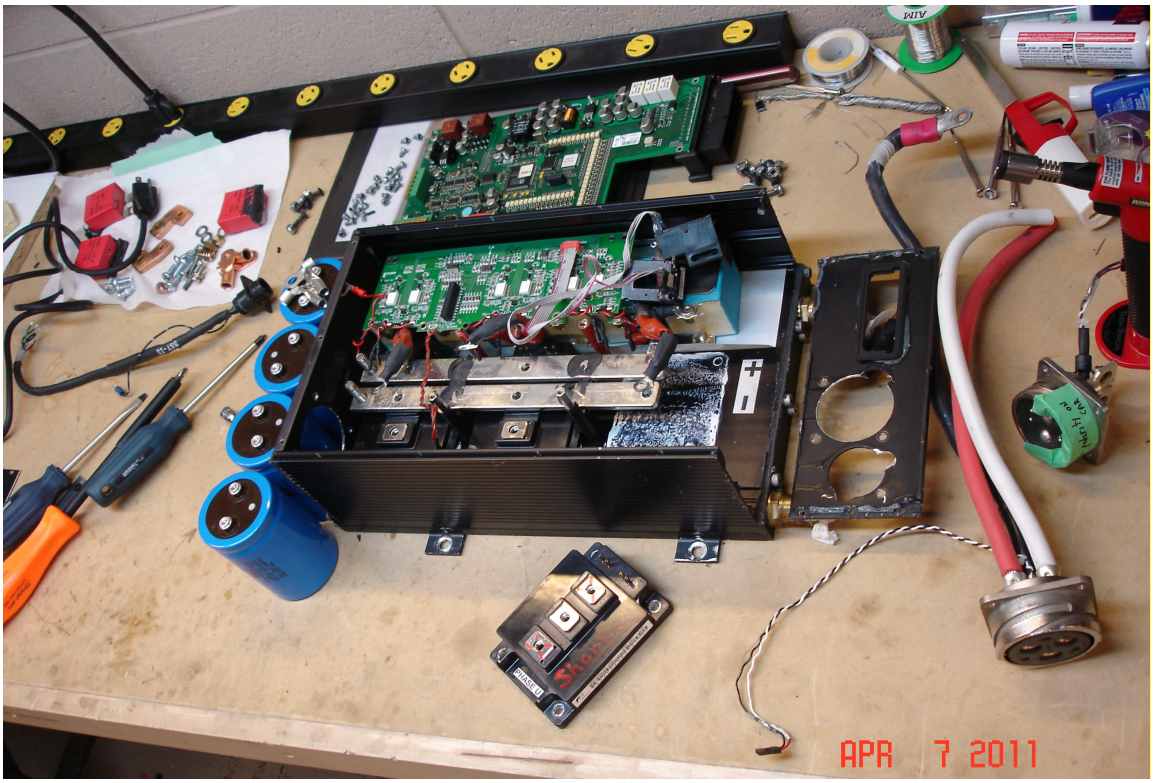


Figure 50. Inverter with shorted IGBT.

In any case, noise issues had been at the root of proper RS232 communications with the inverter, and frequent loss of communications with the device while recording data. It is believed (but not proven) that the majority of ground plane noise came from capacitive coupling between the battery plates and the aluminum boxes they were contained in. The noise on the HV is thus transmitted to the ground, thus signal lines pick up ground reference problems. This noise problem was experienced with the temporary Li-ion battery using an aluminum box and located in the trunk of the vehicle. With the new battery box, such coupling was to be minimized due to the non-metallic composite material employed in construction and the shorter high voltage leads.

6.3.2.4 State-of-Discharge

Since the charger input was instrumented, it was possible to verify at least in part the cyclic efficiency of this process. The state of charge is not an easy quantity to verify accurately because it depends on temperature and remaining "surface charge" on the cells from the last direction of current flow. However, from the battery depletion test, the

team now has a fair idea of the applicable discharge curve and can estimate SOC to better than 5% even in the midranges. The BMS offers accuracy of individual cell voltages down to 1 mV, and temperatures to 0.1°C. Figure 51 is the experimentally derived voltage / SOC curve of the temporary battery pack based on the EPA results.

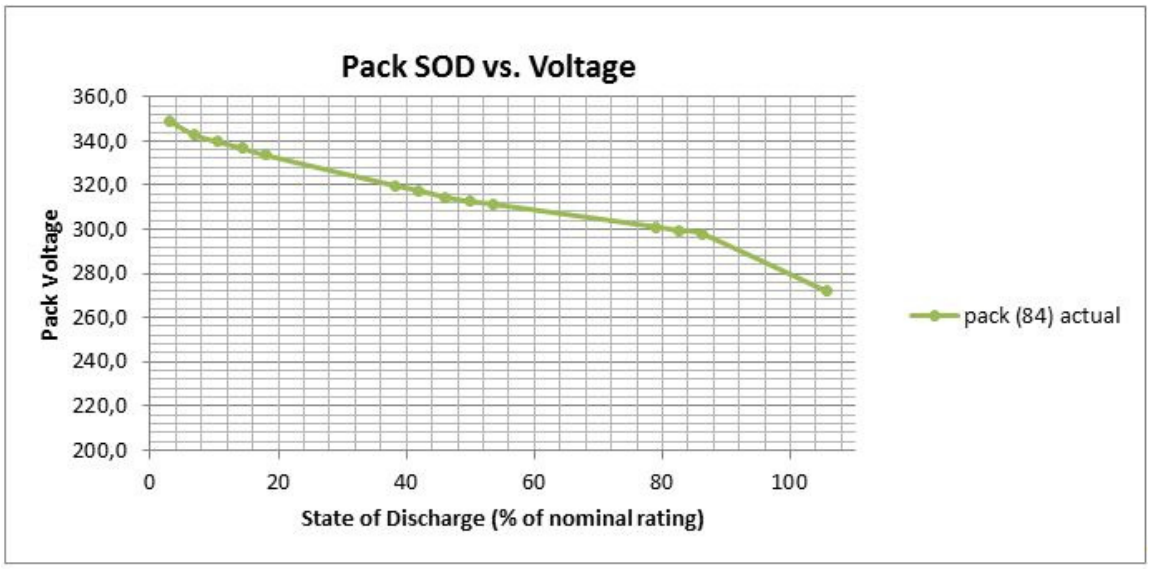


Figure 51. Experimentally derived battery pack energy vs. voltage curve.

6.3.2.5 Summary

The dynamometer results were encouraging, and in line with the prediction made from PSAT simulations for energy consumption and range by the team in the first year of the competition. Although the temporary Li-ion pack was not fully balanced before testing at EPA, it managed to deliver slightly more energy than the nominal 74.6 kWh expected. 375 km (over 230 mi) were covered through the various drive cycles, thus easily demonstrating competition requirements for autonomous range were being met. With the data subsequently gathered from the two UDDS cycles run with REGEN active, projections were made for energy consumption in final tune. This computes to approximately 178 Wh/km (286 Wh/mi) for the UDDS cycle, and 167 kW/km (268 Wh/mi) on the highway cycle. If the entire test sequence were re-run with REGEN active, a 400 km (250 mi) range should have resulted. Not to mention that these results were based on a temporary battery pack. An additional 10 cells were installed in the final

battery modules to raise the voltage, but also provide for approximately 8.9 kWh extra energy capacity, thus 83.5 kWh total. Using a 55% UDDS and 45% HW cycle mix and regenerative braking, the range with the temporary battery was 442 km, whereas the estimation for the final one is 482 km. Table 56 reflects the performance data collected at EPA, and projections thereupon for the year 3 competition. Since no dynamic events were accomplished at that competition, except for the vehicle mass, these projections are considered as the final performance results of the UOIT EcoCAR project.

Table 56. Performance projection based on EPA Results.

Parameter		Performance (EPA)	Estimated @ Competition
Battery Capacity (kWh)		74.6	83.5
Battery Charging Efficiency (AC to DC) (%)		87.5	87.5
UDDS Consumption (Wh/km)		178	182
Highway FTP (Wh/km)		167	171
Steady 60 mph (Wh/km)		197	202
Fuel Consumption	(L/100 km)	2.39	2.45
	(MPGGe)	98.3	96.0
GHG (g/km)		138	142
PEU (Wh/km)		213	218
Acceleration	0-60 mph (s)	19.0	14.0
	50-70 mph (s)		9.0
Braking Distance (m)		N/A	44
Weight (kg)		2061	2112
EV Range (55% UDDS + 45% HW) (km)		442	482

Further, the BMS, signal emulator, datalogger and dashboard displays were all performing nominally. No mechanical issues were uncovered, and only an unrated relay in the charger circuit was discovered at technical inspection. With a replacement already in hand, this concern was resolved immediately.

CAN communication issues with the motor controller were discussed in Section 5.6.2.1 and were still present at EPA, since messages were only being read from the inverter and none were sent. This issue was fixed before the final competition by sending CAN commands from the VIM to the TIM enabling proper functionality of the inverter's CAN communication feature. This lack of functionality restricted the motor information

the team was able to record during testing at EPA, however it did not affect the results of the targeted tests, such as the energy consumption determination, motor tuning and regenerative braking calibration, and was fixed for the final competition.

6.3.3 Vehicle Technical Specification

Even though the vehicle did not partake in any of the dynamic events at the final competition, the vehicle technical specification (VTS) met all the competition requirements as summarized in Table 57. The 0-60 mph and 50-70 mph accelerations, along with the 60-0 mph braking, had been tested at the second year one and EPA. The performance values for the final design were extrapolated from the data collected at EPA and can also be found in Table 56. The vehicle can seat as many passengers as the stock vehicle configuration and has an increased cargo space through an in-floor compartment. The UF weighted FE and GHG emissions were calculated extrapolating the results recorded at EPA. A range of 482 km was estimated. UOIT's vehicle scored 100% at the towing capacity event at year 2 competition. Using the same motor and inverter with a larger battery and no significant mass increase, the towing capacity of the vehicle should not be affected. At the final competition, the vehicle mass and ground clearance were measure at 2130 kg (or 2112 kg) and 178 mm, respectively. The starting time of the vehicle is only limited by the high voltage precharge circuit calibrated to last less than 2s. In summary, the vehicle performance met all the EcoCAR competition requirements.

Table 57. UOIT Team VTS.

Specification	Competition		Team
EcoCAR	**Production VUE**	**Competition Requirement**	**VTS**
Accel 0-60 mph	10.6 s	≤14 s	14 s
Accel 50-70 mph	7 s	≤10 s	9 s
UF Weighted FE	8.3 L/100 km (28.3 mpgge)	7.4 L/100 km (32 mpgge)	2.45 L/100km (96 mpgge)
GHG emissions	246 g/km	217 g/km	142 g/km
Towing Capacity	680 kg (1500 lb)	≥680 kg @ 3.5% 20 min @ 72 km/h (45 mph)	≥680 kg @ 3.5% 20 min @ 72 km/h (45 mph)
Cargo Capacity	0.83 m^3	Height: 457 mm (18") Depth: 686 mm (27") Width: 762 mm (30")	Stock configuration + in-floor compartment
Passenger Capacity	5	≥4	5
Braking 60-0 mph	38 m- 43 m (123 -140 ft)	< 51.8 m (170 ft)	44 m (144 ft)
Mass	1758 kg (3875 lb)	≤ 2268 kg (5000 lb)	2112 kg
Starting Time	≤ 2 s	≤ 15 s	≤ 2 s
Ground Clearance	198 mm (7.8 in)	≥178 mm (7 in)	178 mm (7 in)
Range (UF=.971)	> 580 km (360 mi)	≥ 320 km (200 mi)	482 km (300 mi)

6.3.4 Consumer Acceptability

The UOIT prototype was designed to improve consumer acceptability by maintaining the stock features of the vehicle and adding new ones. The vehicle can seat 5 occupants. The cargo space was maintained and an in-floor trunk compartment was even created and carpeted to make use of the free space left by the removal of the stock NiMH battery. Having no engine and an electric powertrain, this vehicle is quieter than the original one. The audible noises now come from the electronic brake control module pump when activated and the rolling sound of the tires against the road. A DVD player with a screen integrated to each front seat head rest was installed for rear seat passenger entertainment. For infotainment, the double DIN radio was replaced by a touch screen navigation system (NAV). The custom screen, back panel and LEDs create a customized instrument cluster with relevant EV information providing immediate feedback to the driver, such as speed, SOC, estimated range, battery current and voltage, motor and cell temperature, along with vehicle ready and GFI lights. As a maintenance feature, the on-board diagnostic port (OBDII) still works for diagnosing the remaining stock controllers

(ECUs), however DTCs for the new high voltage powertrain controllers were not programmed. These ECUs can be diagnosed through CAN and RS232 communication connectors located in the glove compartment with the datalogger. Another hidden characteristic of this prototype is the use of automotive grade and environmentally resistant electrical components and connectors.

6.4 Thesis Results

Where section 6.2 and 6.3 detailed the overall results of the EcoCAR project at UOIT, this section focuses on the author's accomplishments. After studying the CAN dictionary of the vehicle, a list of ECUs to disconnect and CAN messages to emulate was made. Then, ECUs were individually removed from the vehicle taking care to rebuild the multiple CAN bus network, after each removal. Also, selected CAN messages were successfully emulated by the VIM.

The integration of the high voltage and controls of the VIM, BMS, charger and motor controller went smoothly on the test bench as well as the transition to the vehicle. Safety features were successfully implemented in the VIM, such as the acceleration and brake pedal fault management, cell protection, HV interlock system, inertia switch and voltage/current/temperature management. The instrument panel cluster was customized to display the vehicle powertrain status, i.e. ready and ground fault interrupt lights, relevant information about the FFEV on an in-house screen, and stock gauges were reprogrammed to suit the needs of a FFEV. Also, the cluster background panel was modified accordingly by another team member.

In addition to the stock OBDII CAN bus access under the dashboard and on the left of the steering wheel, RS232 and CAN access points to every powertrain EV controller were made available through the glove compartment. Troubleshooting interfaces were created to diagnose the vehicle using variables in the VIM. The VIM inputs and outputs were all tested first to insure proper functioning of the controller leading to a successful test bench integration phase. In the 3rd year competition, the VIM model-based program was translated to migrate to a more powerful MotoTron controller, i.e. from the ECM-

0555 to the ECM-5554. A voltage translator was also added with success to link the VIM and motor controller controls signals.

The 3 main tasks of the VIM, i.e. emulation, decoding sensors and controlling actuators, along with managing the HVCM, function properly. A functional power steering, the customized instrument cluster features and the absence of DTCs are examples of the successful emulation of CAN messages. The VIM makes decisions based on the acceleration and brake pedals, ignition potentiometer and shift lever position values, along with actuating the radiator fans. The subsystem of the VIM responsible of controlling the HV module actuates as intended the main contactors, precharge and discharge relays, charger contactor and the traction enable signal sent to the motor controller. To accelerate the integration process and troubleshooting of the vehicle, terminal blocks were used instead of automotive connectors to route the wires between the VIM and the other controllers.

6.5 Summary

The validation, calibration and testing methodology was presented with specific examples carried out both at UOIT and EPA. The overall results of the vehicle performance and the team's success at the EcoCAR competition were then discussed. Finally, results of projects lead by the author were described. The next chapter concludes by highlighting the author's contributions to the UOIT EcoCAR project, recommending future work to be considered and presenting a closing argument.

Chapter 7

Conclusion

7.1 Thesis Summary

The primary goal of this thesis was the electrical design and implementation of a full function electric vehicle powertrain architecture based on GM's existing pre-production 2009 2-mode hybrid Saturn VUE. In chapter 2 the architecture selection process that led the UOIT EcoCAR team to choose a full electric powertrain for their vehicle prototype, along with listing the main powertrain controllers selected was described. Chapter 3 probed deeper in the theory behind the most important in-vehicle communication protocol used today, the controller area network (CAN). The electrical and control system integration strategy was then detailed in chapter 4. An exhaustive study of the model-based algorithm programmed in the vehicle integration module (VIM), which is the controller at the heart of this integration strategy, was provided in chapter 5. Chapter 6 discussed the tests and results of the vehicle conversion to full electric propulsion both at a general team level and individually for the author's specific accomplishments and contributions.

7.2 Main Conclusions

The first main conclusion is that the tight packaging of the 83.5 kWh in-house built Li-ion battery in an existing chassis was the primary accomplishment of the UOIT EcoCAR team. The competition organizers were among the skeptics thinking this could not be achieved with a battery having enough capacity to meet the range requirement of the competition. The UOIT team proved them wrong by exceeding the competition range requirement when doing a complete battery depletion test at the EPA facility. Building and packaging a battery of this magnitude while dealing with an existing chassis came with challenges, such as cooling, electrolyte leaks, high voltage control, and cell monitoring and balancing. From an architectural standpoint, it seems less challenging to build an EV from the ground up to accommodate a battery able to provide a range greater

than 400 km. However, low sales volumes of electric vehicles generally preclude developing bespoke chassis designs, thus UOIT demonstrated what lies in the realms of possibility.

The end results of the VIM integration prove that the model-based design approach and upgraded controller used for rapid prototyping were adequate. It provided a fast programming method; the flexibility to calibrate variables on the fly; compatibility with a broad range of sensors and actuators; the capability of communicating on multiple CAN buses; and was even left with enough controller resources for future additions to the control system, if required.

Results also indicated it is entirely possible to remove a considerable number of ECUs, mainly powertrain ones, from a vehicle while avoiding DTC generation if a proper CAN emulation method is employed and the physical CAN buses are reconstructed. The combination of having the GM CAN dictionary alongside the in-depth analysis of the vehicle CAN logs made an accurate static and dynamic CAN message emulation possible. Not being provided with the CAN database, static emulation would have potentially been possible by studying CAN logs before and after an ECU removal and putting the missing signals back on the CAN buses with constant values to avoid DTCs. However, it would have made it difficult and time consuming to gain control of the instrument panel cluster requiring dynamic emulation.

Furthermore, establishing CAN communication between generic controllers from different suppliers can require a series of trials if one does not understand the terminology defining CAN messages and signals, or as it is too often the case, incomplete documentation and poor terminology employed in the controller specification sheets.

A preliminary analysis made at the final competition on the precharge circuit resistor failure pointed towards a component selection with insufficient (unspecified) surge current rating, however an in-depth analysis might come to a different conclusion.

The flawless migration of the electric powertrain from the test bench to the vehicle proved the benefits of a step by step methodology for designing, testing and troubleshooting engineering systems. In the engineering world, a transition of this importance comes rarely without major engineering issues to solve.

The choice of the MES-DEA TIM600 as a motor controller caused the team several problems and challenges. The poor documentation resulted in poor calibrations, damaging two S10-EV motors and even one of the motor controller IGBTs, even though days were spent analyzing the device specifications. The preclusion of disabling the CAN receiving command while keeping the status transmission activated caused delays in the control strategy implementation and forced the author to find temporary solutions to some issues for the second year competition and testing at EPA. Vehicle speed determination and motor information recording and monitoring were affected.

The importance of creating and respecting a predefined timeline cannot be emphasised enough. According to the author, a couple months of delay in the battery build process and the inconvenient timing of the testing at the EPA facility resulted in a lack of time for thoroughly testing the new controls and high voltage systems. Consequently, the UOIT EcoCAR prototype was not ready on time to compete in dynamics events at the final competition, resulting in component failure and an overall 13^{th} place finish.

The last results from dynamic testing were gathered at EPA a few months prior. An impressive range of over 400 km was demonstrated on a dynamometer with a temporary Li-ion battery pack of 74.6 kWh; while the final battery has a capacity of 83.5 kWh and an estimated range of 482 km with REGEN activated. Also, the vehicle 0-60 mph acceleration is estimated at 14s, limited primarily by the drive motor assembly permitted to the team. The available ETX-101 unit is the preferred choice for continued work.

7.3 Original Contributions

The list below organizes some of the author's specific contributions successfully implemented in this research and development project:

- Non-confidential document on electric vehicle conversion.
- In-vehicle implementation of the vehicle integration module (VIM) and its model-based algorithm, including:
  - Dynamic and static CAN message emulation of removed ECUs.
  - Acting as a CAN message gateway, such as sending messages to be displayed on the in-house IPC screen or recorded by the datalogger.
  - High voltage distribution box control through the high voltage control module.
  - Decoding and management of stock sensors, such as acceleration and brake pedals, ignition key and transmission shift lever.
  - Fault management of acceleration and brake pedal inputs.
  - Actuation and control of the radiator fans.
- Bench testing of the added powertrain electrical subsystems, i.e. VIM, BMS, Li-ion battery, TIM, motor, charger.
- In-vehicle electrical integration of the EV powertrain subsystems.
- High voltage design of the powertrain architecture and Li-ion battery packs.
- Modifications to the vehicle CAN buses (GMLAN HS and PTE), including:
  - Fixing of the broken daisy chain when removing stock ECUs.
  - Migration from daisy chains to hybrid configurations, i.e. combination of star configuration and daisy chain.
  - Addition of several ECUs to the CAN networks.
- Gaining control of the dashboard by properly emulating CAN messages, for example:
  - Ready and GFI lights.
  - Speed, SOC and current gauges.
- Design and implementation of an in-house instrument cluster screen with CAN capabilities.

- Description of an unambiguous terminology to explicitly define CAN messages and map their signals.
- Implementation of a strategy to control five BRUSA chargers connected in parallel.
- Implementation of 2 GFI fault levels.
- Explanation of the removal and emulation of ECUs.
- Explanation of the electrical aspects of a vehicle conversion to electric.

7.4 Recommendation

7.4.1 Future Work

Even though the UOIT EcoCAR team spent countless hours developing their EV prototype, mainly due to competition deadlines some engineering issues were left unresolved, temporarily fixed or ignored. As described in chapter 6, after dynamic testing of their prototype at EPA, the team undertook major modifications to the high voltage system by upgrading the Li-ion battery, high voltage distribution box and control module, along with the migration to a more powerful controller for the VIM. Unfortunately, the totality of these changes could not be accomplished in time for the final competition and the vehicle was left in a non-running state. Future research and development should focus on completing this migration phase to bring the vehicle back to a drivable state. Below is a list of main tasks to accomplish in an attempt to reach that objective and also improve the vehicle:

- Precharge circuit analysis by troubleshooting the HV distribution box and HVCM. (Verify the polarity of the battery main contactors in particular, if any.)
- Further testing on the motor controller calibrations determined at EPA, and tuning if necessary – now with a higher voltage battery pack.
- Addition of proactive on-board diagnostics for the EV powertrain controllers, sensors and actuators to reduce the need for reactive troubleshooting when an issue or failure arises, and tend towards a certified vehicle that is diagnostics compliant.
- Fix battery cell leaks – incorporate updated cell module packaging.

- Verify terminal block signal routing.
- Fix the 12V battery draining problem that exist on the stock hybrid Saturn VUE, but ignored by the team.
- Improve the power reduction request controlling the inverter when a ground fault occurs.

7.4.2 Best Practices

The list following enumerates the hands-on personal best practices gained by the author over the years, while working on developing vehicle prototypes, such as the EcoCAR project and 2 prior solar car projects. They are recorded here for the benefit of readers contemplating similar endeavours.

- Hardware is as important as software.
- A good wiring harness drives good results, in other words a bad harness equals bad results.
- Learn how to make a proper wiring harness (soldering, crimping, wire lengths, loom, etc.).
- A need for software testing time in the schedule once the hardware is built.
- The mechanical integration of the prototype should be finished about 2 months prior to a competition to give the controls and high voltage teams enough time to test and troubleshoot their systems.
- A lack of testing equals a lack of results.
- Stick to the timeline. If it does not exist, create one.
- The important is not to have the best or perfect design for the competition, but a reliable prototype.
- Make priorities and do not waste time on little things, in other words do not let the perfect be the enemy of the good.
- At one point, but not too early, the team need to stop “cuddling” their car and push it to its limits, more likely resulting in parts breaking. This step is essential to learn and test the robustness of the prototype, however thorough debugging and calibrating need to be completed first.

- Skip no steps in the design, integration, testing or calibration phase.
- Log all CAN buses in the different ignition key modes before disassembly.
- Insure the car can be debugged conveniently. For example, put all the communication connectors at the same location, i.e. the glove compartment.
- Know your teammates and learn the best way to deal with each of them considering their strengths and weaknesses.
- Know your personal limits, when you are productive and when to rest.
- When building a prototype, the design is in constant evolution. Wires should not be shortened until the system has been thoroughly tested and the design proven to be final. For example, a lot of pressure was put on the author to cut the VIM wiring harness by 2 meters, but the team was relieved to have the extra length when the cables needed to be rerouted to accommodate the front battery pack.
- Test the equipment regardless of whether it is new or used. For example, the brand new CANlog4, a $2000 device, had 2 defective CAN transceivers and a few of the BMS boards, reused from another project, had blown surface mount fuses.
- Archive a new version of the software each time a small improvement is made.
- Make release notes for each software version.
- The use of an online software release management tool might be useful when working within a group of programmers.
- Keep the same file names when creating new releases to avoid bugs, only modify the folder name.
- Avoid spaces and special characters in file and folder names, along with long names (< 32 bits).
- Keep a compiled version of the working software release to avoid having to rebuild it again, i.e. the .dll and .srz files for MotoTron controllers. Compiling and building a program can be time consuming.
- Ideally, always flash 2 controllers to have a spare in case the flashing of one fails and leaves the other controller in a useless state.
- The high voltage lead and the controls lead should be 2 different electrical engineers to spread the work load.

- Once embedded system software is proven reliable, it will stay reliable, whereas hardware reliability decreases over time.

7.5 Conclusion

This thesis documents the state of the art in comprehending architecture choices, components selection and controls development in an electric vehicle conversion designed for competition. This thesis was also written in the hope of providing a solid high voltage and controls basis to students, engineers and even hobbyists working on green vehicle prototypes, such as electric vehicles, to successfully accomplish their own vehicle conversion. This manuscript should also be one of the starting points for future students involved in the continuation of the UOIT EcoCAR project if the latter is re-launched one day.

References

[1] EcoCAR: The NeXt Challenge, (page consulted in October 2013), [Website], http://www.ecocar2.org/ecocarchallenge

[2] Miguelangel Maduro. "WELL-TO-WHEEL GREENHOUSE GAS EMISSIONS AND ENERGY USE ANALYSIS OF HYPOTHETICAL FLEET OF ELECTRIFIED VEHICLES IN CANADA AND THE U.S.", University of Ontario Institute of Technology, 2010.

[3] Leon Zhou, Jeremy Wise, Shaun Bowman, Curran Crawford, Zuomin Dong. "Design, Modeling and Hardware Implementation of a Next Generation Extended Range Electric Vehicle", University of Victoria, 2010-2011.

[4] Argonne National Laboratory, (page consulted in April 2014), [Website], http://www.transportation.anl.gov/modeling_simulation/PSAT/

[5] Greg Rohrauer, Pierre Hinse, Helen Qin, Mike Maduro. "Year 1 UOIT Pre-Competition Report - UOIT EcoCAR Report 4", University of Ontario Institute of Technology, 2009.

[6] Hybrid Center, (page consulted in October 2013), [Website], http://www.hybridcenter.org/hybrid-center-how-hybrid-cars-work-under-the-hood-2.html

[7] Greg Rohrauer, Pierre Hinse, Helen Qin, Mike Maduro. "Year 1 UOIT Final Technical Report - UOIT EcoCAR: A Full-function Electric Vehicle Design with +400KM Range", University of Ontario Institute of Technology, 2009.

[8] Greg Rohrauer, Pierre Hinse, Helen Qin, Mike Maduro, Joseph Brennan, Gavin Clark, Hugo Provencher. "EcoCAR Challenge Year 2 Progress Report 3", University of Ontario Institute of Technology, 2010.

[9] Gavin Clark, Samantha Hazell. "Year 3 Technical Presentation" [Presentation], University of Ontario Institute of Technology, June 2011.

[10] Axsen, Jonn, Burke, Andrew, Kurani, Ken. "Batteries for Plug-in Hybrid Electric Vehicles (PHEVs): Goals and State of Technology circa 2008", Institute for Transportation Studies, University of CA, Davis, May 2008.

[11] "Cell Specification Data – SLPB 160460330", Kokam Co., Ltd.

[12] REAPsystems Ltd. "Manual release 1.02 BMS 12C s/w version 0.12", www.reapsystems.co.uk.

[13] BRUSA. "USER's MANUAL – Battery Charge NLG5 – 02.2003-NLG5xx_108_bis Nr24.doc", www.brusa.biz

[14] Rating plate on the Delphi S10-EV motor.

[15] MES-DEA. Traction Inverter Module – TIM300/TIM400/TIM600 – User Manual – Installation guide of Electric Powertrain MES-DEA", version 7.2 ENG, MES-DEA Alternative Energy Division.

[16] Woodward. "ECM-5554-112-0902-C/F – Engine Control Modules – Calibratible / Flash", datasheet 36757, MotoHawk Control Solutions.

[17] dSPACE. "MicroAutoBox II", [Website], http://www.dspace.com/en/pub/home/products/our_solutions_for/flexray_development_with_ds/dspace_products_flexray_applic/microautobox_flexray.cfm.

[18] MES-DEA. "DC-DC CONVERTER 1000W - 100V-400V / 12VDC Output, Feb 11th 2005.

[19] G.i.N mbH. "CANlog 4 User Manual", Vector.

[20] Bender. "IR470LY User Manual", www.bender.org.

[21] Greg Rohrauer, Hugo Provencher, Gavin Clark, Joseph Brennan. “Year 3 Progress Report 3 Revision: H”, University of Ontario Institute of Technology, April 2011.

[22] Bosch. “CAN Specification”, Version 2.0, Robert Bosch GmbH, 1991.

[23] Kadionik, Patrice. “Le bus CAN”, École Nationale Supérieure Électronique Informatique & Radiocommunications Bordeaux, 2001.

[24] Held, Gilbert. “Inter- and intra-vehicle communications”, Auerbach Publications, 2008.

[25] Provencher, Hugo. “Introduction au protocole de communication CAN” [Presentation], ELE4202 Commande des processus industriels, Département de génie électrique, École Polytechnique de Mtl, Automne 2009.

[26] MotoHawk Training. “CAN (Controller Area Network)” [Presentation], Woodward MotoTron Control Solutions, 28 October 2008.

[27] Schultz, Katrina. “GMLAN In-Vehicle Communication Networks” [Presentation], General Motors, 1 October 2010.

[28] Linton, Becky. “CAN Communication for NI Dynamic Testing Software (Hardware in the Loop)” [Presentation], National Instruments.

[29] http://www.can-cia.org/, visited on 13th December, 2011.

[30] CAN in Automation. “Controller Area Network (CAN)” [Website], http://www.can-cia.org/index.php?id=systemdesign-can

[31] CAN in Automation. “CAN physical layer” [Website], http://www.can-cia.org/index.php?id=systemdesign-can-physicallayer

[32] CAN in Automation. “CAN history” [Website], http://www.can-cia.org/index.php?id=systemdesign-can-history

[33] CAN in Automation. “CANopen” [Website], http://www.can-cia.org/index.php?id=systemdesign-canopen

[34] CAN in Automation. “DeviceNet” [Website], http://www.can-cia.org/index.php?id=48

[35] CAN in Automation. “CAN made easy Basic information on the CAN physical and data link layer”, CiA, retrieved December 13th,2011, from www.can-cia.org/pg/can/additional/can_basics_print.pdf.

[36] Passemard, Michel. “Atmel Microcontrollers for Controller area Network (CAN)”, Atmel Corporation, 4069A-CAN-02/04, http://www.atmel.com [Website].

[37] National Instruments. “FlexRay Automotive Communication Bus Overview” [Website], http://zone.ni.com/devzone/cda/tut/p/id/3352, visited on 28th March, 2012.

[38] FlexRayTM. “About FlexRay™” [Website] http://www.flexray.com, visited on 28th March, 2012.

[39] www.freescale.com/files/microcontrollers/doc/fact_sheet/FLEXCOMMS YSTM4FS.pdf [Website].

[40] http://en.wikipedia.org/wiki/Network_topology, visited on 10th December, 2011.

[41] Microchip. “MCP2551 High-Speed CAN Transceiver” [datasheet], Microchip Technology Inc., 2007, DS21667E, [Website] http://www.microchip.com/.

[42] Microchip. “dsPIC33FJXXXGPX06/X08/X10 Data Sheet High-Performance, 16-Bit Digital Signal Controllers” [datasheet], Microchip Technology Inc., 2007, DS70286A, http://www.microchip.com/ [Website].

[43] Philips Semiconductors (NXP Semiconductors). “TJA1050 High speed CAN transceiver” [datasheet], Product specification, 22nd October 2003, www.nxp.com/documents/data_sheet/TJA1050.pdf

[44] Tech Note, “CAN Data Field Format – Intel, Motorola Forward and Motorola Backward” [Website], www.warwickcontrol.com, visited on 13th December, 2011.

[45] T. McLaughlin, Richard. “CAN Data Formats and X-Analyser Functions” [Presentation], 2011, Warwick Control Technologies, retrieved December 13th, from www.testing-expo.com/europe/05txeu_conf/pres/mclaughlin.pdf.

[46] Vector Academy, “Introduction to CANoe / DENoe” [video], v7.0, 2009-03-10, CANoe v7.0, Vector CANtech Inc.

[47] Vector CANdb++ Online Help, “Mapping and Position Numbering of Signals in Frames” [Software], Vector CANdb++ Editor, Version 3.0.62 (SP6).

[48] Matlab Product Help. “CAN Unpack” [Software], CAN Unpack :: Block Reference (Vehicle Network Toolbox™), Library: CAN Communication, Matlab 2009b.

[49] http://www.race-technology.com/wiki/index.php/CANInterface/ByteOrdering, visited on 13th December, 2011.

[50] SAE, “Vehicle Architecture For Data Communication Standards” [Website], http://www.sae.org/servlets/works/documentHome.do?comtID=TEVEES12, visited on 30th March 2012.

[51] National Instruments, (visited in February 2011), website: http://www.ni.com/pxi/ [Website].

[52] Wiring Controlsoft RS-485 networks, KeyMaster Systems, Hardware Installation Guidelines, Controlsoft, 2002, retrieved from www.powerrichsystem.com/Downloads/RS-485wiringNetworks.pdf

[53] ISO Committee Draft. “Road vehicles – Controller area network (CAN) – Part 1: Data link layer and physical signalling”, ISO/CD 11898-1, ISO/TC 22SC 3 N, October 1999.

[54] ISO Committee Draft. “Road vehicles – Controller area network (CAN) – Part 2: High-speed medium access unit”, ISO/CD 11898-2, ISO/TC 22SC 3 N, September 1999.

[55] ISO Committee Draft. “Road vehicles – Controller area network (CAN) – Part 3: Fault tolerant medium access unit”, ISO/CD 11898-3, ISO/TC 22SC 3 N, 2001-06-07.

[56] Wikipedia. “On-board diagnostics” [Website], http://www.en.wikipedia.org/wiki/On-board_diagnostics, visited on 28th March, 2012.

[57] Microchip. “Section 21. Enhanced Controller Area Network (ECANTM)” [datasheet], Microchip Technology Inc., 2007, DS70185A, http://www.microchip.com/ [Website].

[58] Tritium. “WaveSculptor CAN Bus Communications Protocol Specification” [Datasheet], 20 August 2007, http://www.tritium.com.au/index.html [Website].

[59] Provencher Hugo. “Embedded Real-Time System Development for Electric Vehicles”, Faculty of Engineering and Applied Sciences, UOIT, ENGR 5910G Embedded Real-Time Control Systems, 2011-02-16.

[60] CAN in Automation. “Safety in field bus systems” [Website], http://www.can-cia.org/index.php?id=50

[61] ON Semiconductor. “MMBZ15VDLT1G, MMBZ27VCLT1G, SZMMBZ15VDLT1G, SZMMBZ27VCLT1G 40 Watt Peak Power Zener Transient Voltage Suppressors” [Datasheet], Semiconductor Components Industries, LLC, 2012, http://www.onsemi.com/pub_link/Collateral/MMBZ15VDLT1-D.PDF

[62] NXP Semicondutors. “PESDxS2UAT series Double ESD protection diodes in SOT23 package” [Datasheet], 2004 February 18, http://www.nxp.com/documents/data_sheet/PESDXS2UAT_SER.pdf

[63] Vector. “Basic Vehicle Networks – An Introduction to CAN” [Webinar], Vector CANtech Inc., 2012.

[64] RM Michaelides software & elektronik. “Connecting PCs to mobile or stationary CAN networks” [Website], http://www.rmcan.com/index.php?id=73&L=1

[65] Kvaser Advanced CAN Solutions. “Home\Products\CAN\USB” [Website], http://www.kvaser.com/en/products/can/usb.html

[66] Vector. “VN1600” [Website], http://www.vector.com/vi_vn1600_en.html

[67] Vector. “CANlog 3 and CANlog 4” [Website], http://www.vector.com/vi_canlog_en.html

[68] New Eagle. “Welcome to New Eagle Learning Center” [Website], http://www.neweagle.net/support/wiki/index.php?title=Main_Page

[69] dSPACE. “MicroAutoBax II” [Website], http://www.dspace.com/en/inc/home/products/hw/micautob.cfm

[70] National Instruments. “Controller Area Network (CAN)” [Website], http://www.ni.com/can/

[71] Softing. “CAN bus Interface Card: CAN-AC2-PCI” [Website], Softing your connection to excellence, http://www.softing.com/home/en/automotive-electronics/products/can-bus/interface-cards/can/pci-2.php

[72] Intrepid Control Systems, inc. “neoVI FIRE : 6x CAN, 4x LIN - neoVI RED : 2x CAN, 2x LIN” [Website], http://intrepidcs.com/neovifire/index.html

[73] Vector. “Introduction to CAN” [Website], http://www.vector-elearning.com/vl_einfuehrungcan_portal_en.html

[74] Vector. “Vector Training Worldwide” [Website], http://www.vector.com/vi_training_en.html

Appendix A

Detailed Performance Simulation Results

Table 58. Powertrain Specification of the Full Electric [5]

Components	Size
Motor (GMT101X)	110 kW peak
Battery	90 cells, 240 Ah, Li poly
Total pack energy	80 kWh
Max available energy	80 kWh (100% DOD)
Vehicle Curb Weight	2139 kg

Table 59. Simulation Results of the Full Electric [5]

Parameters	Result
Accel 0-60 mph	9.5 s
Accel 50-70 mph	6.2 s
UDDS	191.9 Wh/km*
HW	232.6 Wh/km*
Combined	215.4 Wh/km* (96.4 mpgge)
US06	319.8 Wh/km*
Towing on grade	747.7 Wh/km*
Range (combined cycle)	444 km, 276 mi

* Charging efficiency 90%, upstream energy use included

Table 60. Powertrain Specification of the EREV-60 [5]

Components	Size
Motor (GMT101X)	110 kW peak
Generator (E10)	1 liter, 2 cyl, 17.5 kW
Battery	160 cells, 40 Ah, Li-poly
Total pack energy	24 kWh
Available energy	19.2 kWh (80% DOD)
Vehicle Curb Weight	2043 kg

Table 61. Simulation Results of the EREV-60 [5]

Parameters	Result	
Accel 0-60 mph	8.7 s	
Accel 50-70 mph	5.6 s	
UDDS	250.3 Wh/km* CD, 37.6 mpgge CS	
HW	235.1 Wh/km* CD, 53.9 mpgge CS	
Combined	241.4 Wh/km* CD, 45.6 mpgge CS	75.4 mpgge
US06	389.2 Wh/km* CD, 45.5 mpgge CS	
Towing on grade	729.1 Wh/km*	
All Electric Range	95.2 km, 59.2 mile	UF=0.739

* Charging efficiency 90%, upstream energy use included

Table 62. Powertrain Specification of the PHEV-30 [5]

Components	Size
Front Motor (BAS)	9.7 kW peak
Rear Motor (Magna)	51 kW peak
Battery	80 cells, 40 Ah, Li-poly
Total pack energy	12 kWh
Available energy	9.6 kWh (80% DOD)
Vehicle Curb Weight	2027 kg

Table 63. Simulation Results of the PHEV-30 [5]

Parameters	Result	
Accel 0-60 mph	8.9	
Accel 50-70 mph	5.2	
UDDS	264.0 Wh/km* CD, 31.5 mppge CS	
HW	239.8 Wh/km* CD, 36.9 mppge CS	
Combined	246.1Wh/km* CD, 35.6 mppge CS	60.6 mppge
US06	394.8 Wh/km* CD, 25.1 mppge CS	
Towing on grade	746.1 Wh/km*	
All Electric Range	46.7 km, 29.0 mile	UF=0.512

* Charging efficiency 90%, upstream energy use included

Appendix B

Additional CAN Theory

1. Arbitration Mechanism

As explained in Section 3.2.2.1, when a dominant bit (0) and a recessive bit (1) are simultaneously transmitted, the dominant bit overrides the recessive one. This is the fundamental principle under the bitwise arbitration mechanism using identifiers. Figure 52 shows an example of arbitration by using 3 nodes that attempt to send a message simultaneously and gain access to the CAN bus.

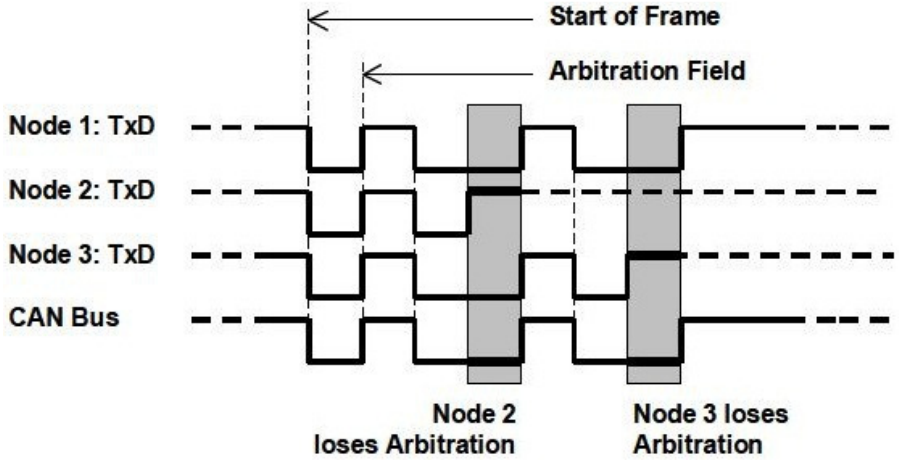


Figure 52. Arbitration mechanism [36].

The grey area on the left overlaps the 3rd bit of the arbitration field of each signal. Node 1 and 3 have a dominant bit (0) while node 2 is attempting to send a recessive one (1). Since the 3rd bit of node 2's identifier is overridden, node 2 realizes that one or several messages of higher priority are trying to access the bus. Thus, it will stop transmitting and re-attempt to send its message as soon as the bus is free again.

A similar scenario happens to node 3 on the 6th bit of the identifier shadowed by the right grey area. Trying to send a recessive bit while node 1 sends a dominant one, node 3 ceases sending its message. Node 1 wins the arbitration, thus gets access to the bus and finishes transmitting its message. As detailed under Appendix E, by sending a dominant

bit, physically node 1 is pulling-up CANH and pulling-down CANL, while node 3 is trying to maintain both CANH and CANL at 2.5V and loses arbitration. Node 3 detects the voltage difference on the line, then gives up and relinquishes till the bus is idle again.

During the arbitration field, whenever a node is attempting to transmit a bit that is different than the physical result on the CAN bus, which is illustrated as the bottom signal on Figure 52, the node in question stops transmitting and awaits the end of frame before re-attempting to send it once more. From another perspective, the node winning the arbitration is identical to the result on the CAN bus [36].

2. Frame format: Standard or Extended

The main subdivision in the arbitration field is the identifier. The CAN specification 2.0 offers 2 possibilities regarding the length of the identifier, standard (11 bits) or extended (29 bits) [22]. These are referred to as the frame format. The substitute remote request (SRR) bit is a recessive bit in the remote transmission request (RTR) position of the extended format. In the extended format, the identifier extension (IDE) bit is recessive and part of the arbitration field while it is dominant and part of the control field in the standard format.

Table 64. Identifier length.

Frame Format	Identifier Length
Standard	11 bits
Extended	29 bits

3. Frame type: Data or Remote

As explained previously, a device can request another device to send back a specific data frame by sending a remote frame. Such a frame contains no data and is recognized by the RTR bit located at the end of the identifier in the arbitration field [22].

Table 65. RTR values.

Frame Type	RTR	Logic Level
Data	Dominant	0
Remote	Recessive	1

4. Format and Type Summary

This section depicts in detail the 4 possible combinations of frames discussed above.

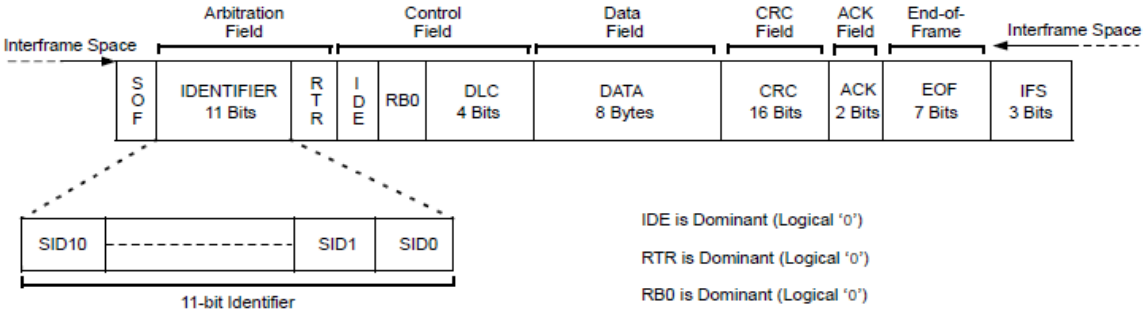


Figure 53. Standard data frame [57].

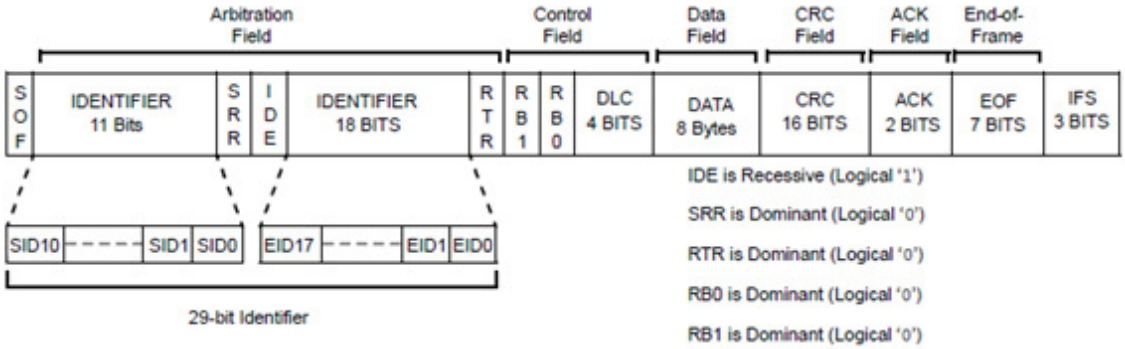


Figure 54. Extended data frame [57].

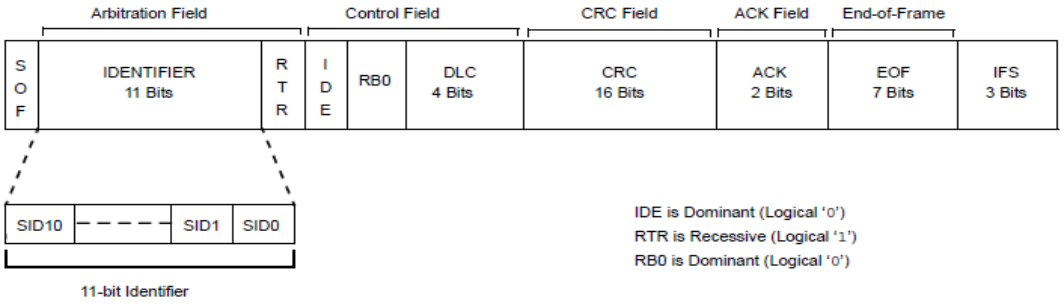


Figure 55. Standard remote frame [57].

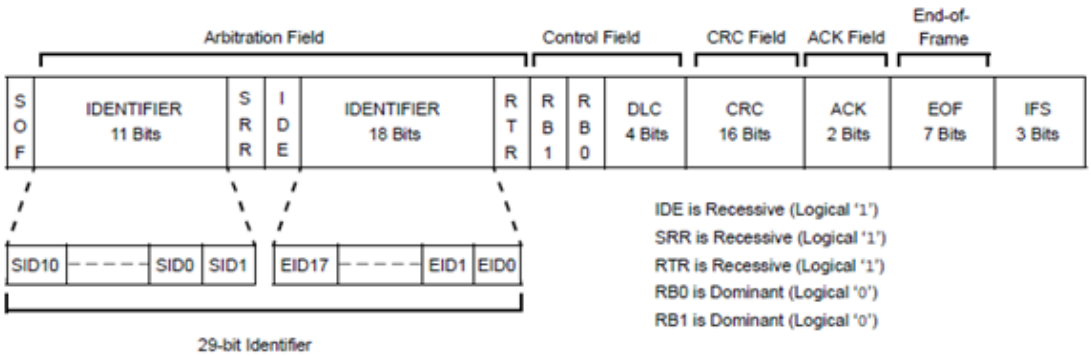


Figure 56. Extended remote frame [57].

5. Error frame

An error frame has the two following fields: error flag and error delimiter [22]. The active error flag (6 dominant bits) and the passive error flag (6 recessive bits) are the 2 existing flags used [23]. It should be noted that the bit-stuffing rule does not apply to these flags. The way to recover from an error is by automatically re-transmitting the faulty message from the transmitter. Each node has an error counter determining whether it is in active error mode, passive error mode or bus off mode. When a station is in active error mode, it sends an active flag on error detection. The opposite applies for a node in passive error mode. In other words, by sending flags with dominant bits, nodes in active error mode have priority over the ones in passive error mode. The general structure of an error frame is illustrated in Figure 57.

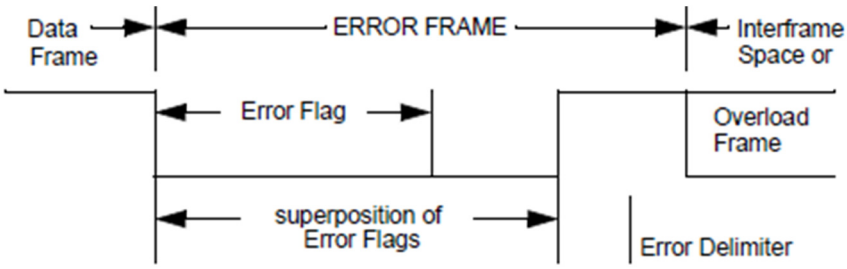


Figure 57. Error frame [22].

The station, or node, sending the first error flag sends 6 dominant bits when in active mode, breaking the bit stuffing rule. When the bit-stuffing error is detected, the other nodes, in either active or passive modes, start sending active and passive flags respectively. In this case, passive error flags are ignored, because they are superscribed by the active flags' dominant bits. As a consequence, the value of the bus becomes a superposition of error flags varying from 6 to 12 dominant bits (see Figure 57). This superposition does not affect the length of each nodes' error flag, but does affect the overall error active flag on the bus which can be of a maximum length of 2 active error flags. Using recessive bits, a passive error flag is much simpler and is always 6 recessive bits long regardless of the superposition of other passive error flags from other nodes that might overlap with the error delimiter. However, it can be overwritten by an active error flag [23].

The error delimiter is 8 recessive bits following an active error flag or a passive error flag. In the first case, there is a superposition of passive error flags and error delimiters that are overwritten by active error flags. Once the last active error flag is completely sent, after 6 to 12 dominant bits, the first recessive bit of the overall error delimiter appears. In the second case, no node is sending dominant bits resulting in the start of the overall error delimiter after exactly 6 recessive bits during the superposition of passive error flags [23].

6. Overload frame

There are 2 causes resulting in the generation of an overload frame [23]:

- A receiver requiring a delay before being able to decode the next data frame or remote frame.
- A node detecting a dominant bit in the first and second bit positions during the intermission field of the interframe space.

In both cases, an overload frame, represented in Figure 58, has the general form of an error frame containing active error flags. In other words, 6 to 12 dominant bits as superposition of overload flags and 8 recessive bits as the overload delimiter are sent.

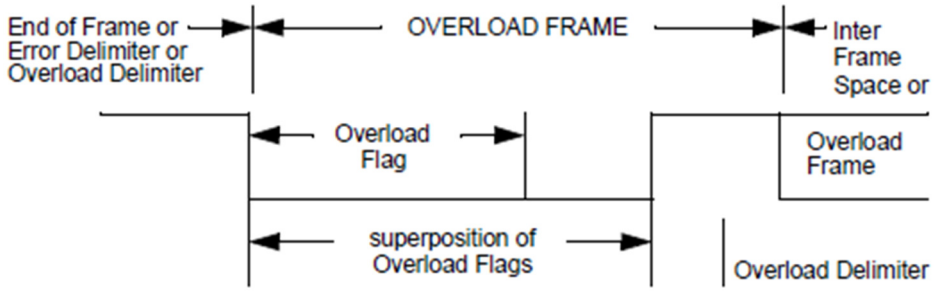


Figure 58. Overload frame [22].

Appendix C

CAN Physical Waveforms

Below is an example of a CAN high-speed transmission showing physical bits on a CAN bus, and therefore the NRZ coding method:

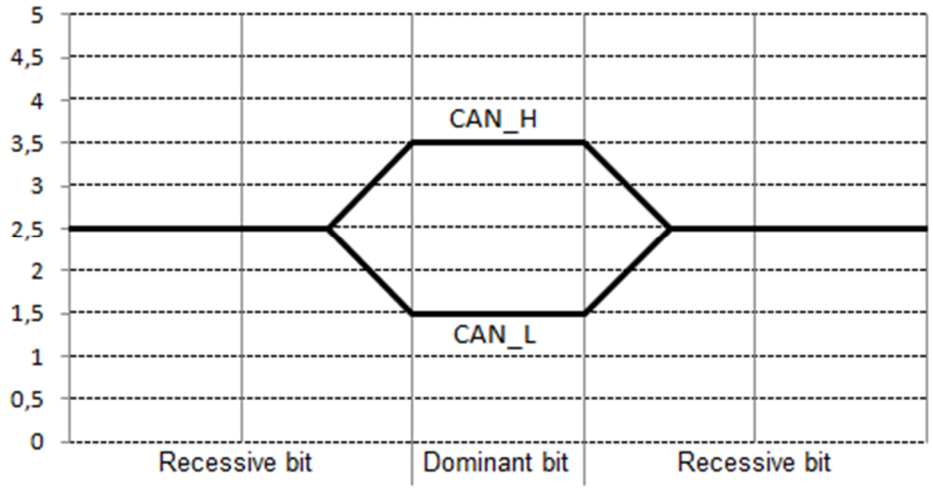


Figure 59. High-speed CAN bus waveform, ISO 11898-2.

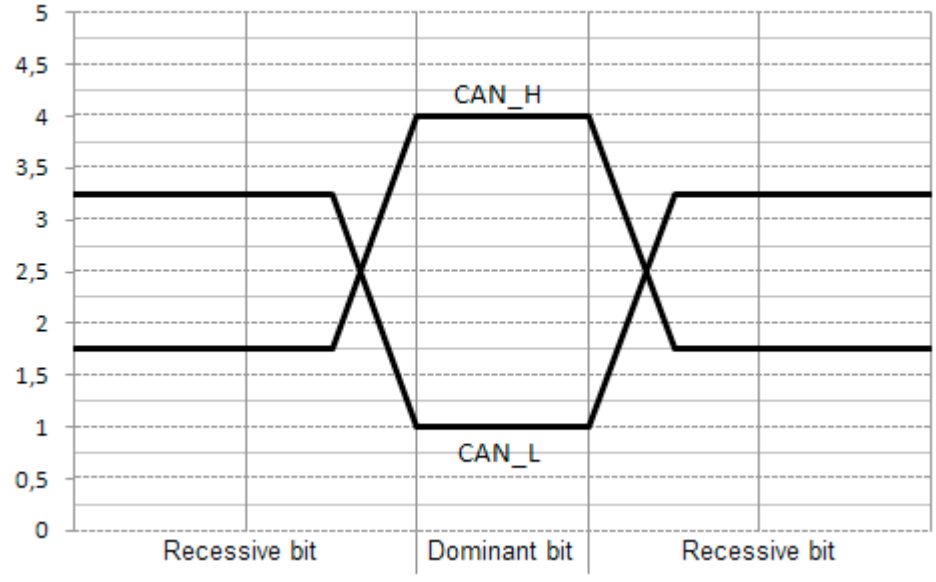


Figure 60. Fault tolerant CAN bus waveform, ISO 11898-3.

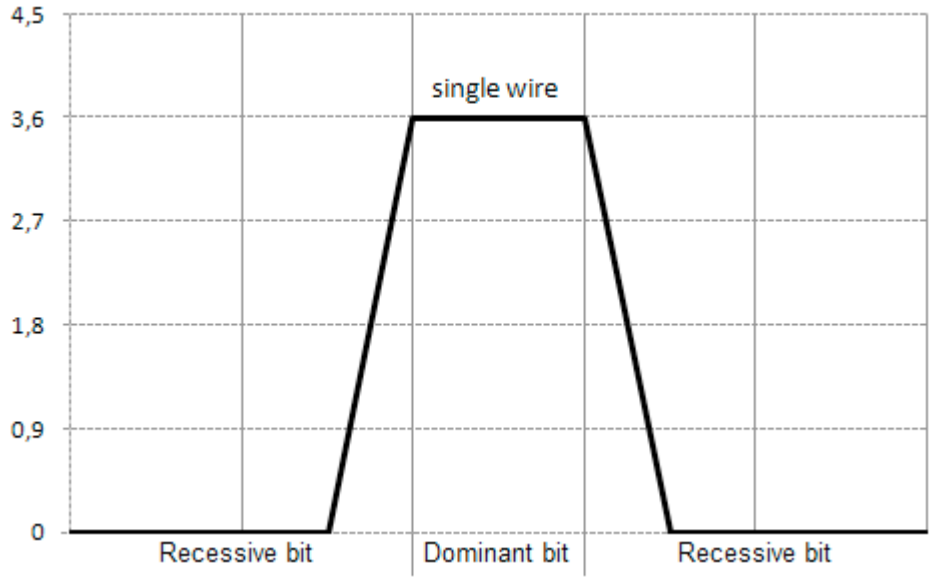
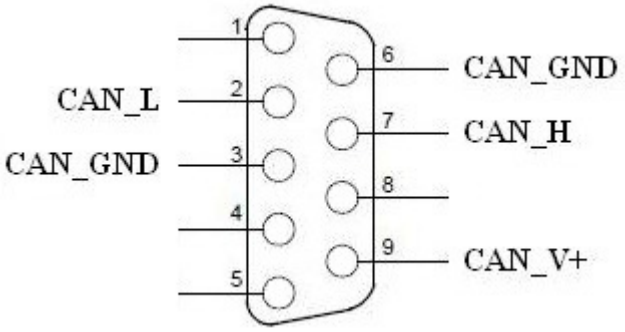


Figure 61. Single wire CAN bus waveform, SAE J2411.

Appendix D

Standard CAN Connectors

There are 2 standard types of connectors to access a CAN bus: DB9 and 5-pin M12, respectively Figure 62 and Figure 63.



Pin	Signals	Description
1		
2	CAN_L	Low dominant level
3	CAN_GND	Ground
4		
5		
6	CAN_GND	Ground (optional)
7	CAN_H	High dominant level
8		
9	CAN_V+	Power (optional)

Figure 62. DB9 connector.

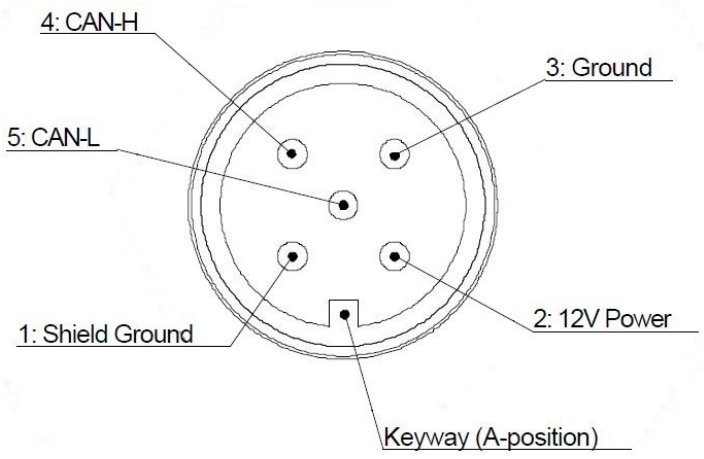
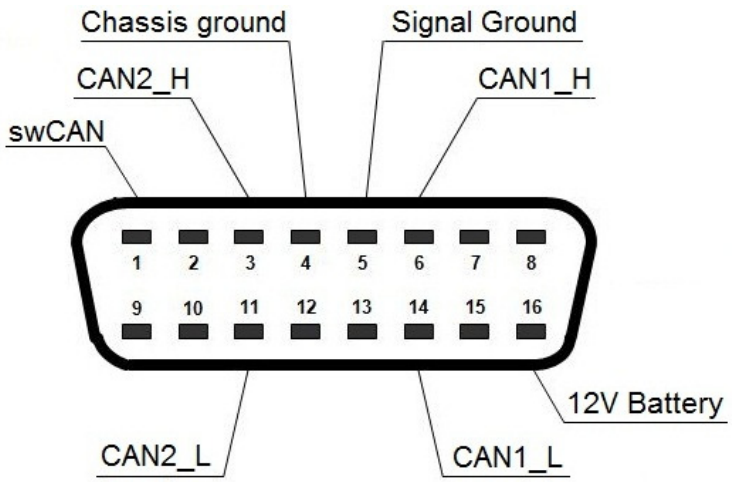


Figure 63. 5-pin M12 connector [58].

It is also interesting to note that in the automotive industry the connector used to access the CAN buses on a vehicle is the SAE J1962 (on-board diagnostics OBDII) connector as shown in Figure 64, based on reference [50].



Pin	Description
1	Single wire CAN
2	
3	CAN2 High
4	Chassis ground
5	Signal ground
6	CAN1 High
7	
8	
9	
10	
11	CAN2 Low
12	
13	reserved
14	CAN1 Low
15	
16	Battery +12V

Figure 64. SAE J1962 connector.

Appendix E

CAN Transceivers

Figure 65 represents the block diagram of a basic high-speed CAN transceiver, the MCP2551 manufactured by Microchip [41]. It is equivalent to the TJA1050 made by NXP Semiconductors, known as Philips Semiconductors before 2006 [43]. The TJA1041 is more sophisticated and will not be discussed in this paper, although its basic functionalities are similar to the MCP2551 and TJA1050. The digital signals are on the left of the figure, while the analog signals are on the right. The three primarily blocks in this diagram are the 2.5V voltage source, the driver control and the receiver both interfacing between the digital and analog world.

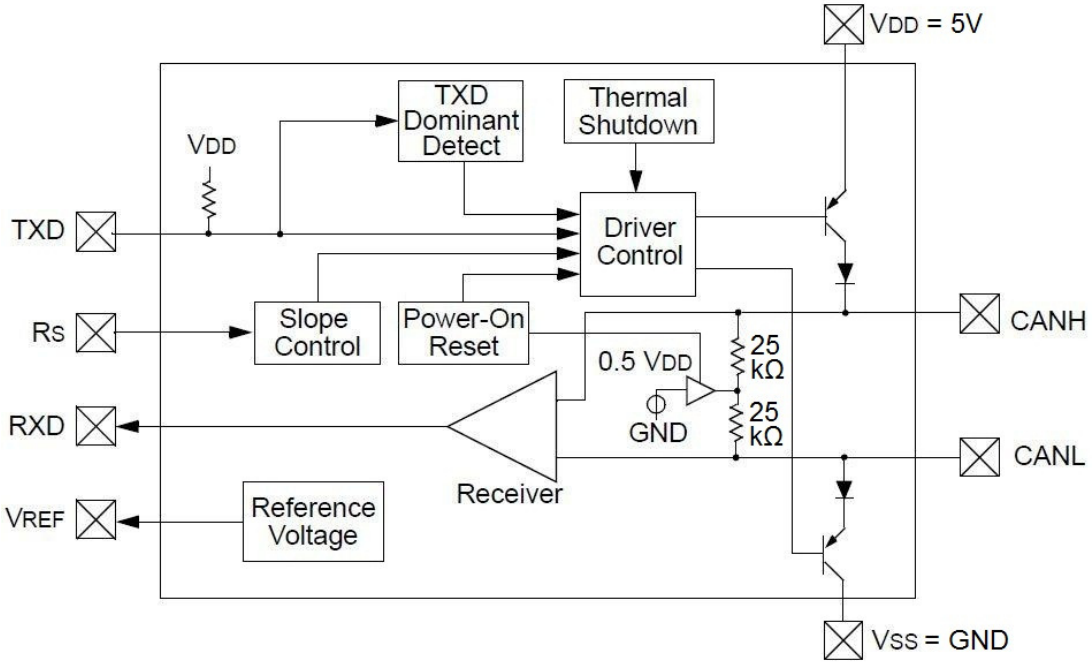


Figure 65. CAN transceiver MCP2551 block diagram [41].

The CANH and CANL voltages are determined by the driver control reacting to the digital transmit signal and by a voltage source of 2.5V. This voltage source is responsible for setting the CAN bus at 2.5V when a node is powered on. This value corresponds to the recessive state voltage. The 2.5V source is connected to each line through a high

impedance resistor, 25 kΩ in the case of the TJA1050 [43]. The value of these resistors contributes to the busload and therefore the maximum number of nodes a CAN environment can support. For example, a high-speed CAN bus on which each node uses a MCP2551 CAN transceiver can support up to 112 nodes [41]. This source can be turned off by the power-on reset block to disconnect the CAN transceiver from the CAN bus.

To obtain an analog dominant bit level, the driver control achieves the bus voltage by driving two transistors. One is connected between the voltage supply (V_{DD} = 5V) and the CANH wire and acts as a pull-up**,** while the other one connects the CANL line to the ground (V_{SS}) and acts as a pull-down. Even though the block diagram shows two outputs on the driver control, both transistors are conducting simultaneously and are off at the same time.

The voltage drop through such a transistor and a diode is about 1V to 1.5V depending on their impedance and the current going through them. Assuming a supply voltage of 5V, when the transistor connected to CANH is conducting, CANH is pulled-up from 2.5V to only 3.5V, due to a 1.5V drop between the 5V source and CANH, caused by the transistor and the diode. When CANH is pulled-up, CANL that was idling at 2.5V is pulled-down and reaches 1.5V due to a voltage drop of 1V across its transistor and diode. The diodes between the lines and the transistors are also protection against high-voltage transients.

The receiver is basically a discriminator circuit having as inputs the CANH and CANL wires, and for output the digital receive signal. A discriminator circuit compares the voltage between its two inputs and sets its output to 1 when both signal are the same, and to 0 when they are different. In other words, the digital output is based on differential voltage levels between the lines. Since interference is generally induced almost equally in both twisted wires, CANH and CANL, and knowing that the receiver reacts to a difference in voltage level between the two lines, this circuit provides high noise immunity.

Several other features are implemented in basic CAN transceivers [41]. The TXD dominant detect block is used for ground fault protection (equivalent to transmitting only dominant bits) on TXD input. The thermal shutdown block disables the driver control outputs controlling the transistors when they overheat. Transistor overheating can be caused when conducting an excessive current due to a short circuit on the bus. An external signal, Rs for Figure 65, can control the rise and fall times of CANH and CANL in order to reduce electromagnetic interference (EMI), also called radio frequency interference (RFI). This signal also controls the CAN transceiver sleep mode. In sleep mode, the discriminator, or receiver, operates at a lower current and the driver control is turned off.

Filters can be placed between the CAN transceiver and the CAN bus to help reduce noise. In the case of a single-wire CAN bus, EMI can be reduced by adding an inductor next to the CAN transceiver on the CAN wire. For a dual-wire CAN bus, high-speed CAN bus, a common mode choke and capacitor can be added to improve radiated emissions. In both cases, electrostatic discharge transient suppressor techniques, also known as ESD protection techniques, can be used to prevent permanent damage to the CAN transceiver related to undesired voltage transients on the bus. One of these techniques involve the connection of back-to-back zener diodes, such as mmbz27vclt1 [61] or pesd24vs2uat [62], as close to the CAN transceiver as possible between CANH and the ground, and CANL and the ground. Other techniques use ESD capacitors or metal oxide varistors (MOV) or other suppression devices, instead of or in addition to back-to-back zener diodes. Providing a good ground to CAN transceivers and connecting it only at only one end of the cable to avoid ground loops, using a twisted pair wires to reduce high frequency noise, and using shielded cables (twisted pair and ground) in high RF environments are efficient noise reduction solutions easy to implement.

Appendix F

Conversion of Display Format

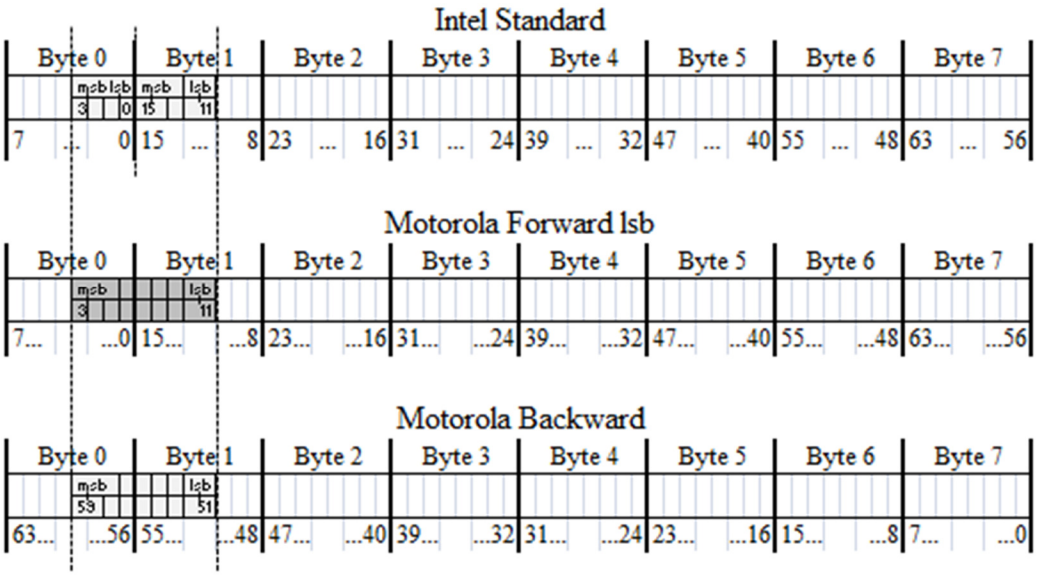


Figure 66. Conversion from Motorola Forward lsb.

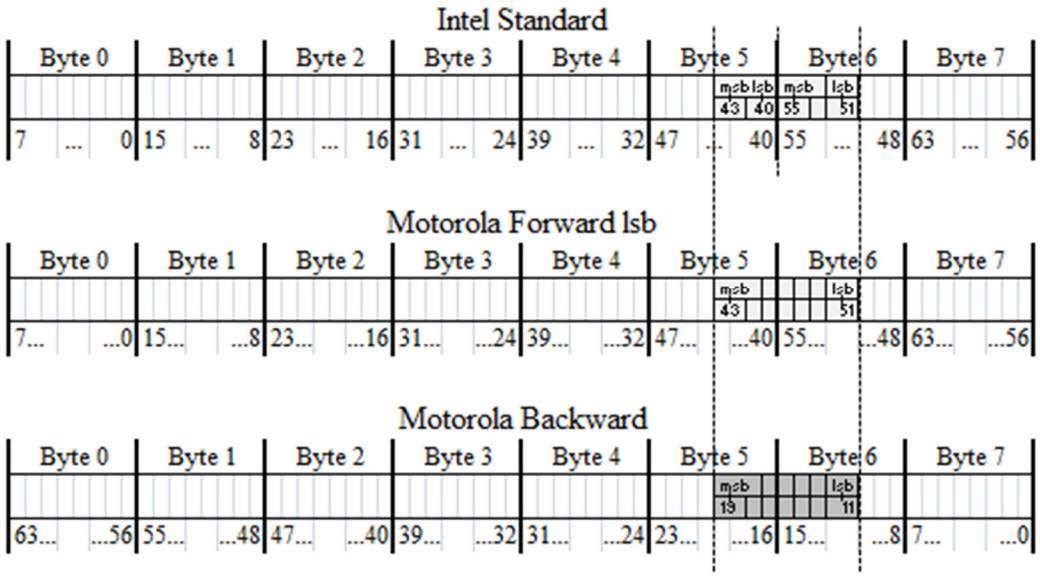


Figure 67. Conversion from Motorola Backward.

Appendix G

Post Office Analogy

As mentioned before, CAN messages are not destination oriented, but source based. However, most nodes need only a small subset of CAN messages travelling on the bus. Modules are required to filter the information transmitted on the bus to select the messages of interest. A relevant and simple analogy to understand the sorting of CAN messages is the post office analogy, adapted from reference [26].

Filtering, also called sorting, can be done by hardware and/or software. Microcontrollers featuring integrated CAN controllers typically have a hardware layer involving programmable buffers acting as a first filter. A custom dispatcher can be programmed in the target module's CAN chip software to refine the first sorting. The post office analogy terminology is defined in Table 66.

Table 66. Post Office Analogy

<table>
<tr><th colspan="2">Post Office</th><th colspan="2">CAN</th></tr>
<tr><td colspan="2">Letters</td><td colspan="2">CAN messages</td></tr>
<tr><td colspan="2">Postman</td><td colspan="2">Hardware/Software dispatcher</td></tr>
<tr><td rowspan="2">Mail box
(Desired letter)</td><td>Address
(of the sender)</td><td>ID Filtering
(ID + mask)</td><td rowspan="2">Slot
(Desired CAN msg)</td></tr>
<tr><td>Name
(of the sender)</td><td>Payload filtering
(Payload + mask)</td></tr>
<tr><td colspan="2">Doorbell</td><td colspan="2">Interrupt (or a function trigger)</td></tr>
</table>

Imagine a world where letters are CAN messages and the postman is the hardware and software dispatcher. In this world, the mail system works according to the sender's address and the name, instead of the destination information. Also, a house owns a mail box for each type of sender, in other words for each type of desired letter. Several people can send letters from the same address and a mail box is only limited to one sender's address, but may contain letters from numerous people from that address. A letter is

desired when its sender's address and name match the ones on the mail box. In this analogy, a desired letter represents a filtered CAN message, i.e. a CAN message having an identifier and a data field fulfilling the criteria of the identifier and payload filters. For priority mail, the postman notices the mail box, but delivers the urgent letter directly to the front door and rings the doorbell. This is the equivalent to setting a desired CAN message to an interrupt or a trigger function.

Appendix H

Comparing LIN and CAN

Table 67 compares major characteristics of LIN and CAN.

Table 67. Comparing LIN and CAN [24].

Feature	LIN	CAN
Network topology	Bus	Bus
Number of wires	2	1
Maximum data rate	20 kbps	1 Mbps
Communications method	UART based	Controller based
Network access	Master-initiated transmission	Non-destructive
Node support	1 master, up to 15 slaves	64-128; typically limited by physical layer or higher-layer protocol

The number of nodes a CAN environment can support shown in Table 67 is different from the details provided by Table 14. The literature does not agree on a specific limit of nodes allowed on a CAN bus. Some documents define the maximum number of nodes as 30 [23], others as 128 [24].The important fact to remember is that the quantity of nodes supported by a CAN bus is dependant and limited by its load and data rate.

Regardless of these 2 criteria, a CAN network can support more nodes than a LIN environment. Being controller based, CAN is a more expensive solution compared to a universal asynchronous receiver-transmitter based local interconnect network, i.e. UART based LIN. However, according to Table 67, CAN is 50 times faster, more efficient, more flexible and more robust than LIN [24]. Moreover, LIN is only designed for automotive applications, whereas CAN has numerous uses beyond vehicles.

The data rate of a LIN environment might be 50 times slower than a high-speed CAN bus, but is similar to a single-wire one. Having cheaper hardware, LIN is often used as an alternative to single-wire CAN to control functions in a localized area. However, single-wire CAN networks are not limited to localized areas. In such

applications, the LIN master node acts as gateway between the LIN network and a CAN environment to extend communications to other ECUs spread all over the vehicle. In other words, LIN is often used in vehicles as a sub-bus via a LIN master node to manage devices in a localized area, such as the roof or a seat, providing several complementary networks to the main CAN buses. Figure 68 illustrates a typical application of a LIN sub-bus in a vehicle, while Table 68 lists common LIN localized support areas, based on reference [24].

Table 68. Typical LIN localized support areas.

Doors
Mirror control
Mirror switch
Window lift
Door locks
Engine
Sensors
Small motors
Roof
Moon-roof controls
Light Sensor
Interior light
Visor lighting
Seat
Occupancy sensor
Seat position motor
Seat heater
Steering column
Wheel tilt/position control motor
Cruise control switches
Wiper control
Turn control

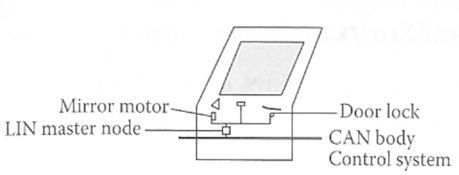


Figure 68. LIN sub-bus [24].

Appendix I

Detailed Subsystem Parameters

1. BRUSA Charger RS232 Interface
2. REAP BMS RS232 Interface
3. MotoTron CAN Interface
4. TIM600 MotorController RS232 Interface
5. CANLog4 USB Interface

1. BRUSA Charger RS232 Interface

1.1 Final Vehicle Design

The description of the interface fields is self-explanatory.

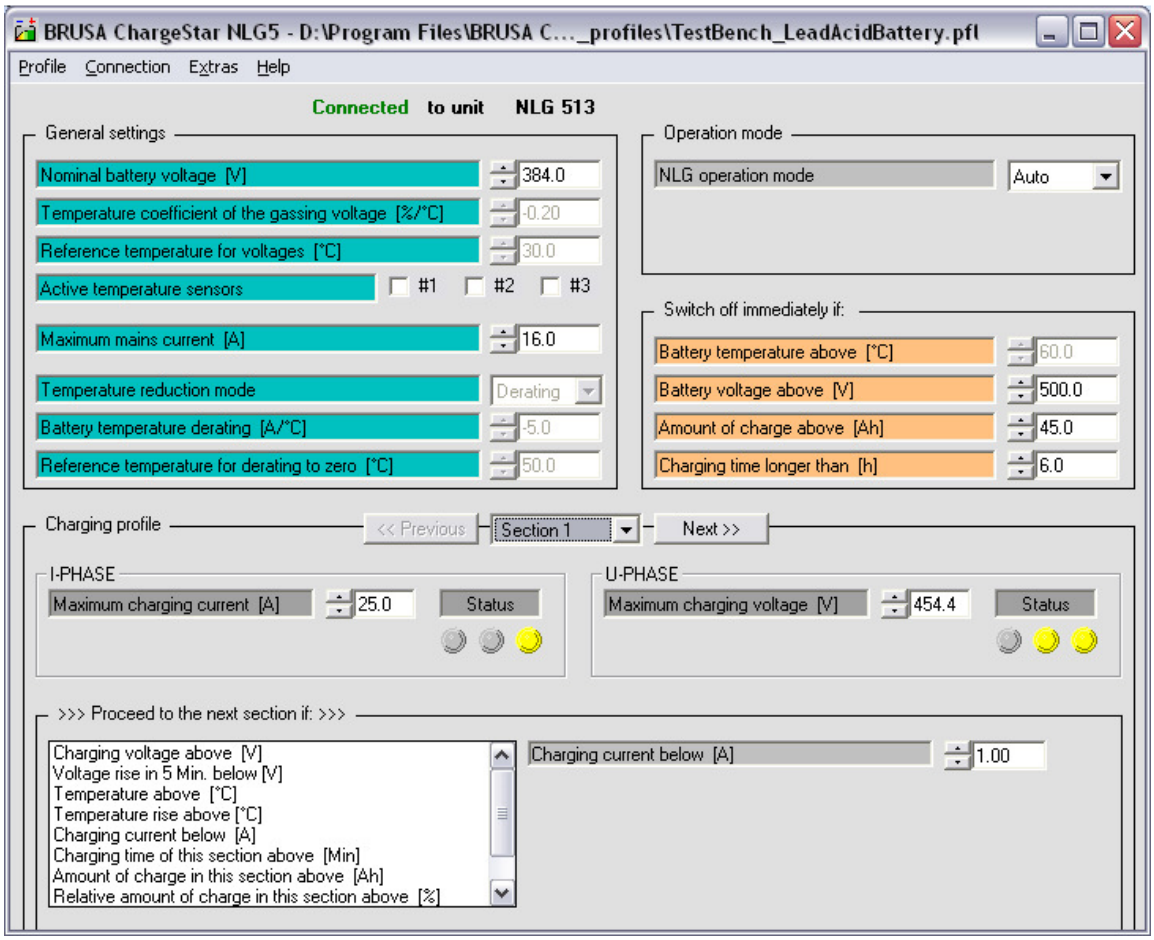


Figure 69. BRUSA Charging profile: Mode 1: Preconditioning.

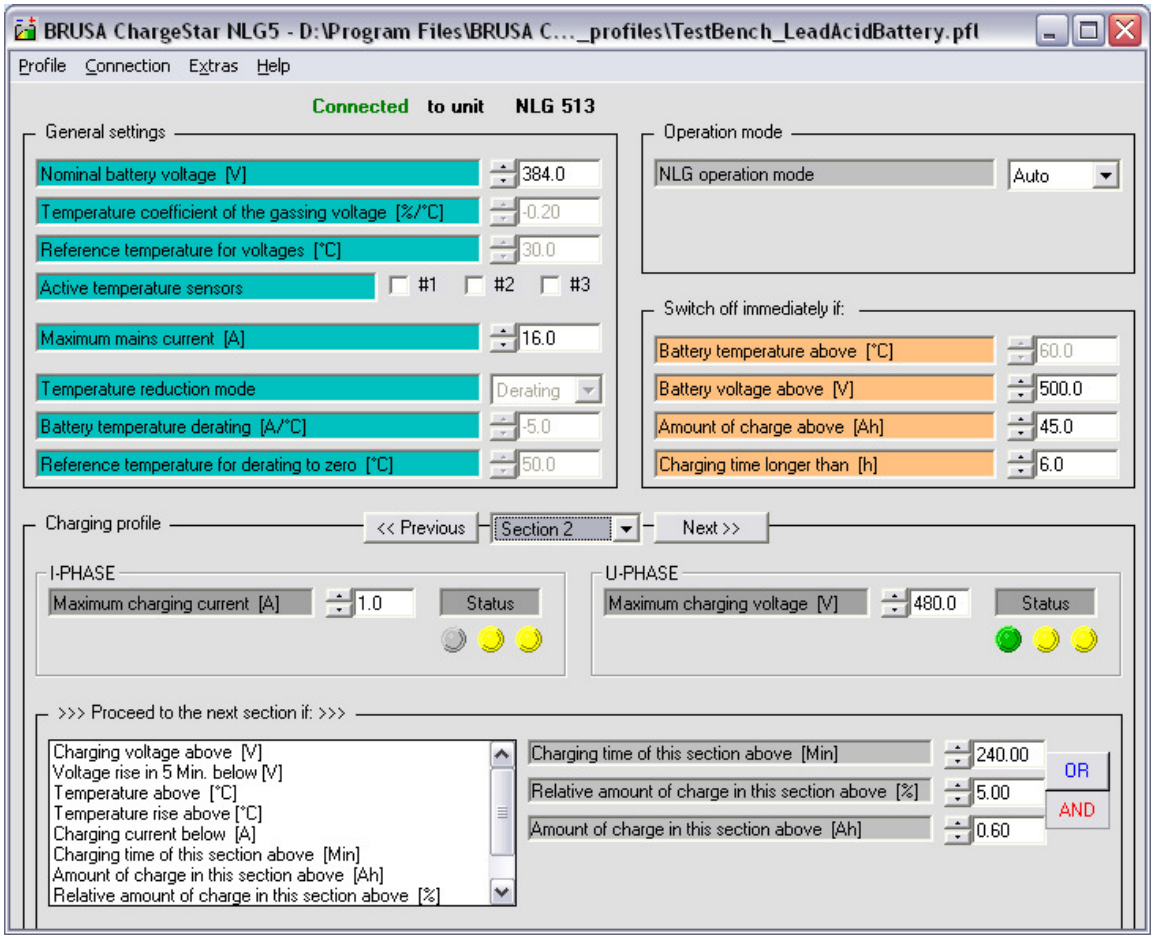


Figure 70. BRUSA Charging profile: Mode 2: Constant Current.

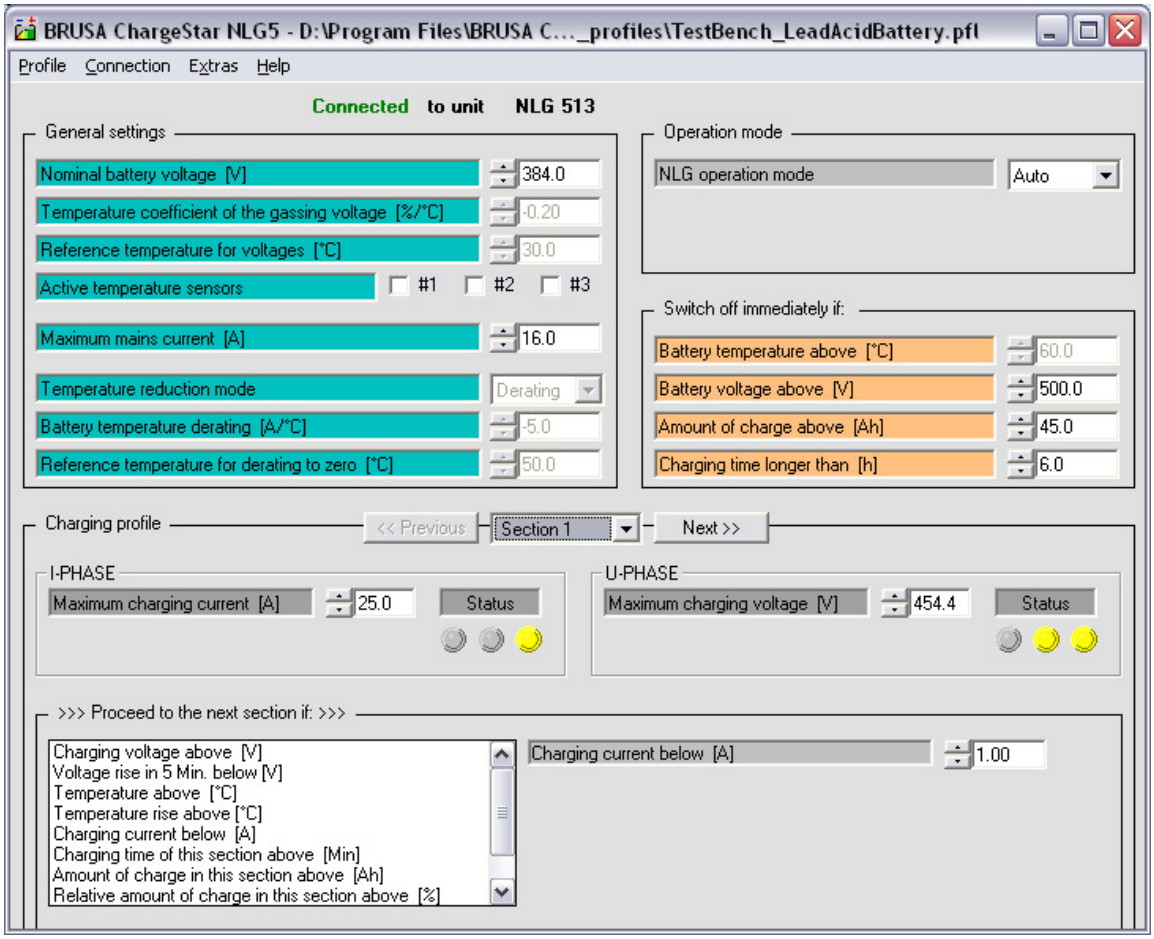


Figure 71. BRUSA Charging profile: Mode 3: Constant Voltage.

1.2 Booster Mode (Not Used)

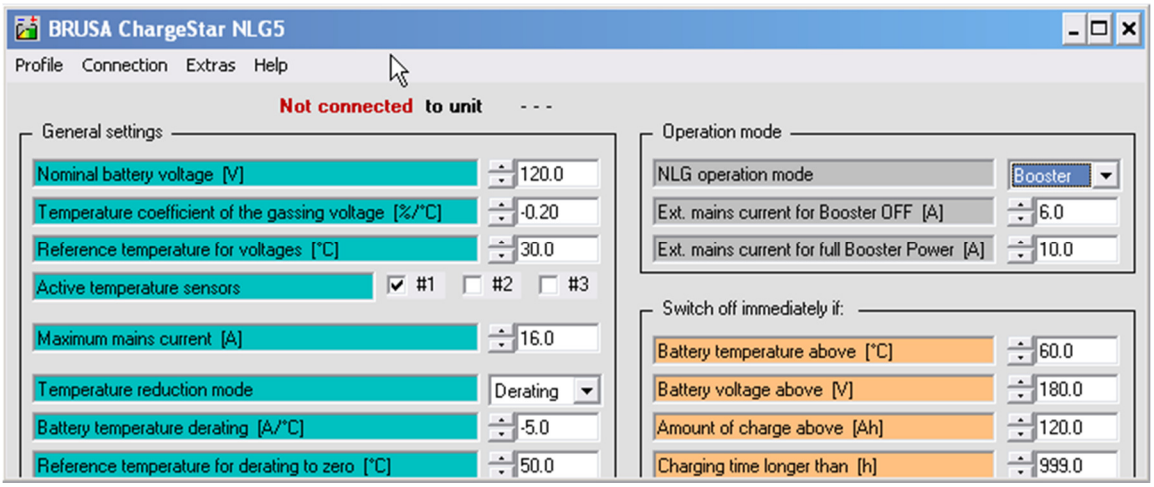


Figure 72. Booster Mode.

1.3 CAN Mode (Not Used)

Although it has not been tried by the author, as shown in Figure 73 and Figure 74, it appears that each BRUSA charger CAN identifiers can be configured allowing dynamic control of multiple chargers from a vehicile integration module.

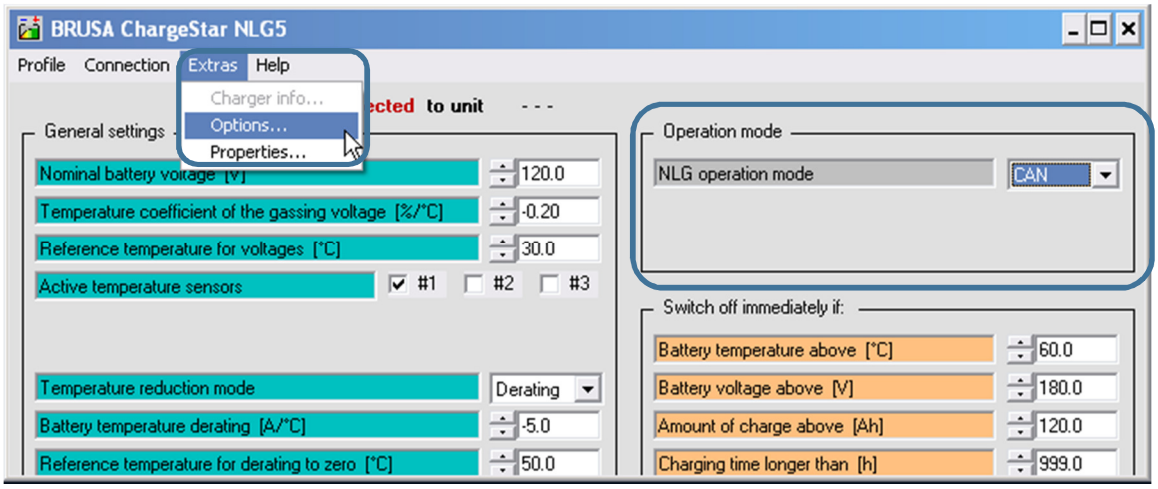


Figure 73. CAN Operation Mode and Extras Options.

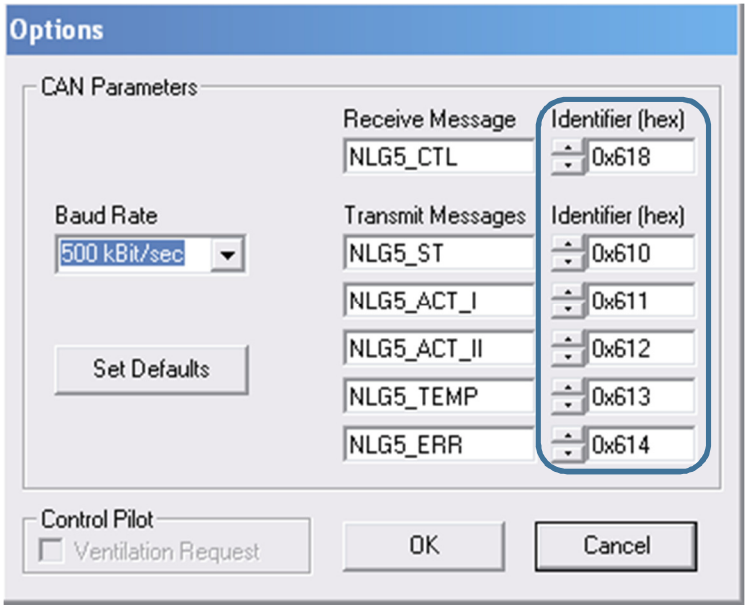


Figure 74. CAN Configuration Window.

2. REAP BMS RS232 Interface

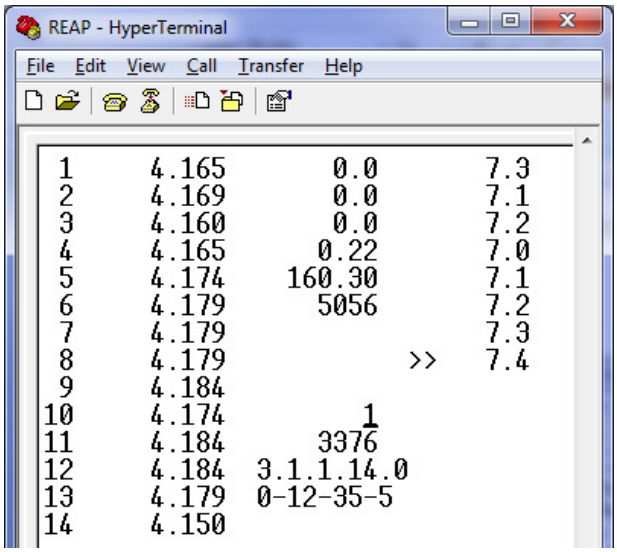


Figure 75. REAP BMS Screen Mode.

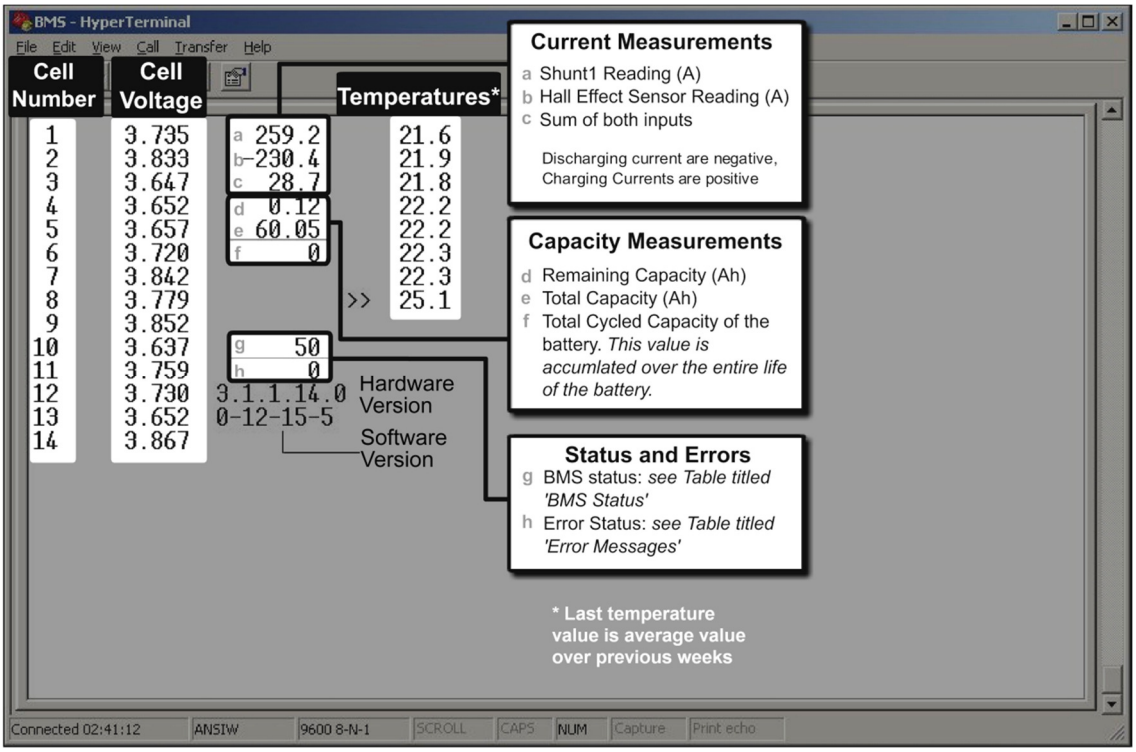


Figure 76. REAP BMS Screen Mode description.

Important values (V, I, SOC) and errors/warnings are displayed.

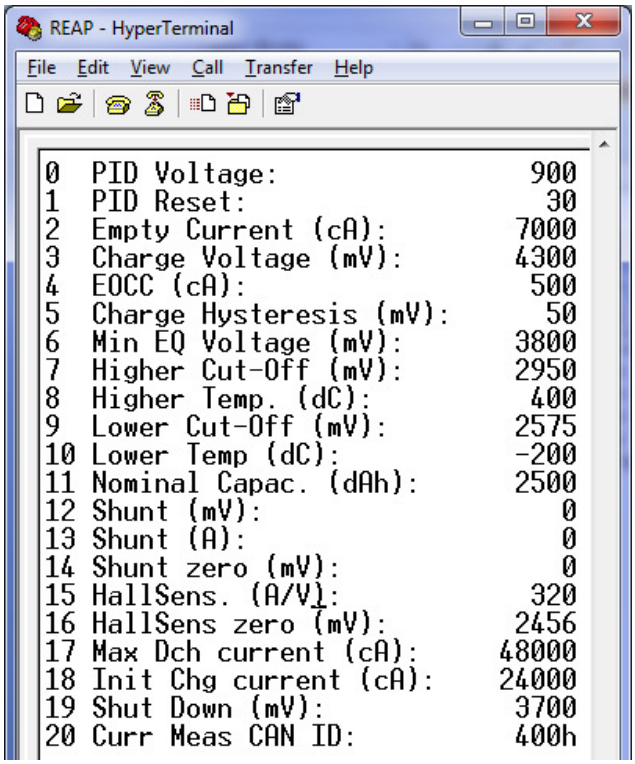


Figure 77. REAP BMS Parameters Setting Mode.

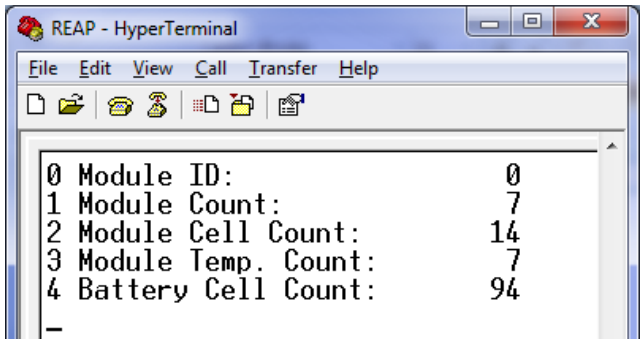
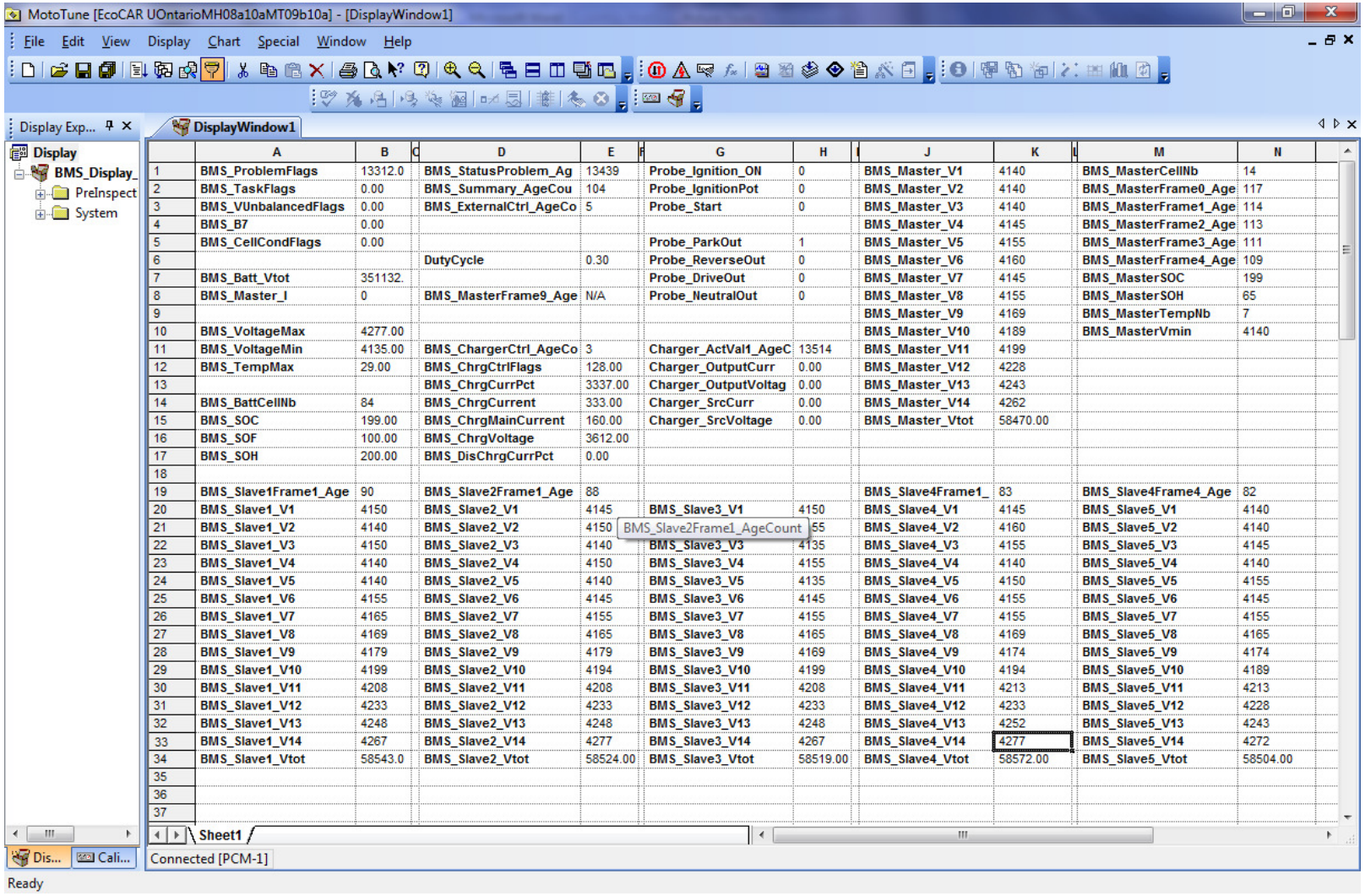


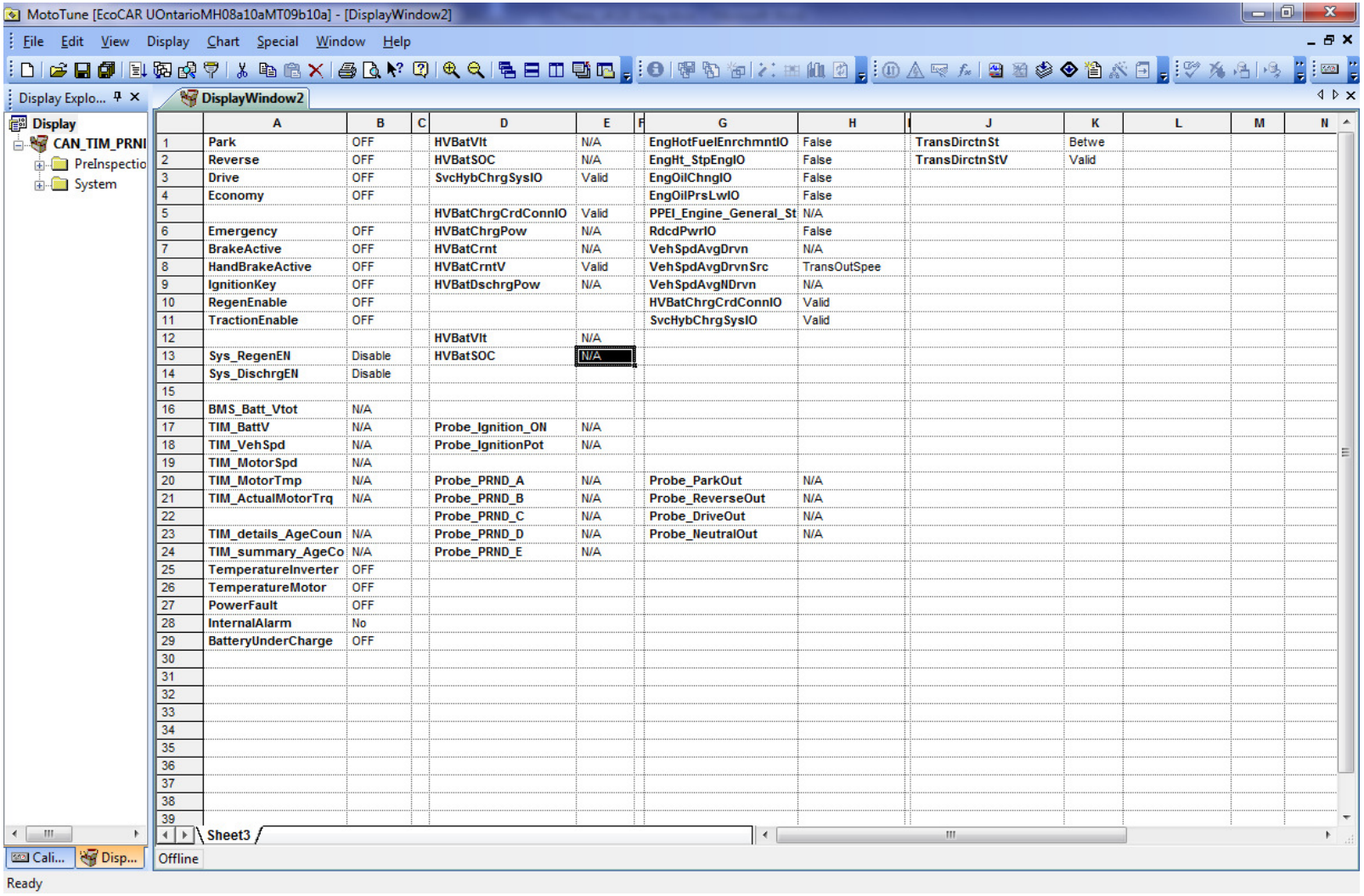
Figure 78. REAP BMS Configuration Mode.

3. MotoTron CAN Interface



	A	B	D	E	G	H	J	K	M	N
1	BMS_ProblemFlags	13312.0	BMS_StatusProblem_Ag	13439	Probe_Ignition_ON	0	BMS_Master_V1	4140	BMS_MasterCellNb	14
2	BMS_TaskFlags	0.00	BMS_Summary_AgeCou	104	Probe_IgnitionPot	0	BMS_Master_V2	4140	BMS_MasterFrame0_Age	117
3	BMS_VUnbalancedFlags	0.00	BMS_ExternalCtrl_AgeCo	5	Probe_Start	0	BMS_Master_V3	4140	BMS_MasterFrame1_Age	114
4	BMS_B7	0.00					BMS_Master_V4	4145	BMS_MasterFrame2_Age	113
5	BMS_CellCondFlags	0.00			Probe_ParkOut	1	BMS_Master_V5	4155	BMS_MasterFrame3_Age	111
6			DutyCycle	0.30	Probe_ReverseOut	0	BMS_Master_V6	4160	BMS_MasterFrame4_Age	109
7	BMS_Batt_Vtot	351132.			Probe_DriveOut	0	BMS_Master_V7	4145	BMS_MasterSOC	199
8	BMS_Master_I	0	BMS_MasterFrame9_Age	N/A	Probe_NeutralOut	0	BMS_Master_V8	4155	BMS_MasterSOH	65
9							BMS_Master_V9	4169	BMS_MasterTempNb	7
10	BMS_VoltageMax	4277.00					BMS_Master_V10	4189	BMS_MasterVmin	4140
11	BMS_VoltageMin	4135.00	BMS_ChargerCtrl_AgeCo	3	Charger_ActVal1_AgeC	13514	BMS_Master_V11	4199		
12	BMS_TempMax	29.00	BMS_ChrgCtrlFlags	128.00	Charger_OutputCurr	0.00	BMS_Master_V12	4228		
13			BMS_ChrgCurrPct	3337.00	Charger_OutputVoltag	0.00	BMS_Master_V13	4243		
14	BMS_BattCellNb	84	BMS_ChrgCurrent	333.00	Charger_SrcCurr	0.00	BMS_Master_V14	4262		
15	BMS_SOC	199.00	BMS_ChrgMainCurrent	160.00	Charger_SrcVoltage	0.00	BMS_Master_Vtot	58470.00		
16	BMS_SOF	100.00	BMS_ChrgVoltage	3612.00						
17	BMS_SOH	200.00	BMS_DisChrgCurrPct	0.00						
18										
19	BMS_Slave1Frame1_Age	90	BMS_Slave2Frame1_Age	88			BMS_Slave4Frame1_	83	BMS_Slave4Frame4_Age	82
20	BMS_Slave1_V1	4150	BMS_Slave2_V1	4145	BMS_Slave3_V1	4150	BMS_Slave4_V1	4145	BMS_Slave5_V1	4140
21	BMS_Slave1_V2	4140	BMS_Slave2_V2	4150	BMS_Slave2Frame1_AgeCount	55	BMS_Slave4_V2	4160	BMS_Slave5_V2	4140
22	BMS_Slave1_V3	4150	BMS_Slave2_V3	4140	BMS_Slave3_V3	4135	BMS_Slave4_V3	4155	BMS_Slave5_V3	4145
23	BMS_Slave1_V4	4140	BMS_Slave2_V4	4150	BMS_Slave3_V4	4155	BMS_Slave4_V4	4140	BMS_Slave5_V4	4140
24	BMS_Slave1_V5	4140	BMS_Slave2_V5	4140	BMS_Slave3_V5	4135	BMS_Slave4_V5	4150	BMS_Slave5_V5	4155
25	BMS_Slave1_V6	4155	BMS_Slave2_V6	4145	BMS_Slave3_V6	4145	BMS_Slave4_V6	4155	BMS_Slave5_V6	4145
26	BMS_Slave1_V7	4165	BMS_Slave2_V7	4155	BMS_Slave3_V7	4155	BMS_Slave4_V7	4155	BMS_Slave5_V7	4155
27	BMS_Slave1_V8	4169	BMS_Slave2_V8	4165	BMS_Slave3_V8	4165	BMS_Slave4_V8	4169	BMS_Slave5_V8	4165
28	BMS_Slave1_V9	4179	BMS_Slave2_V9	4179	BMS_Slave3_V9	4169	BMS_Slave4_V9	4174	BMS_Slave5_V9	4174
29	BMS_Slave1_V10	4199	BMS_Slave2_V10	4194	BMS_Slave3_V10	4199	BMS_Slave4_V10	4194	BMS_Slave5_V10	4189
30	BMS_Slave1_V11	4208	BMS_Slave2_V11	4208	BMS_Slave3_V11	4208	BMS_Slave4_V11	4213	BMS_Slave5_V11	4213
31	BMS_Slave1_V12	4233	BMS_Slave2_V12	4233	BMS_Slave3_V12	4233	BMS_Slave4_V12	4233	BMS_Slave5_V12	4228
32	BMS_Slave1_V13	4248	BMS_Slave2_V13	4248	BMS_Slave3_V13	4248	BMS_Slave4_V13	4252	BMS_Slave5_V13	4243
33	BMS_Slave1_V14	4267	BMS_Slave2_V14	4277	BMS_Slave3_V14	4267	BMS_Slave4_V14	4277	BMS_Slave5_V14	4272
34	BMS_Slave1_Vtot	58543.0	BMS_Slave2_Vtot	58524.00	BMS_Slave3_Vtot	58519.00	BMS_Slave4_Vtot	58572.00	BMS_Slave5_Vtot	58504.00
35										
36										
37										

Figure 79. BMS supervisory interface.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Park	OFF		HVBatVlt	N/A		EngHotFuelEnrchmntlO	False		TransDirctnSt	Betwe			
2	Reverse	OFF		HVBatSOC	N/A		EngHt_StpEnglO	False		TransDirctnStV	Valid			
3	Drive	OFF		SvcHybChrgSysIO	Valid		EngOilChnglO	False						
4	Economy	OFF					EngOilPrsLwlO	False						
5				HVBatChrgCrdConnlO	Valid		PPEI_Engine_General_St	N/A						
6	Emergency	OFF		HVBatChrgPow	N/A		RdcdPwrlO	False						
7	BrakeActive	OFF		HVBatCrnt	N/A		VehSpdAvgDrvn	N/A						
8	HandBrakeActive	OFF		HVBatCrntV	Valid		VehSpdAvgDrvnSrc	TransOutSpee						
9	IgnitionKey	OFF		HVBatDschrgPow	N/A		VehSpdAvgNDrvn	N/A						
10	RegenEnable	OFF					HVBatChrgCrdConnlO	Valid						
11	TractionEnable	OFF					SvcHybChrgSysIO	Valid						
12				HVBatVlt	N/A									
13	Sys_RegenEN	Disable		HVBatSOC	N/A									
14	Sys_DischrgEN	Disable												
15														
16	BMS_Batt_Vtot	N/A												
17	TIM_BattV	N/A		Probe_Ignition_ON	N/A									
18	TIM_VehSpd	N/A		Probe_IgnitionPot	N/A									
19	TIM_MotorSpd	N/A												
20	TIM_MotorTmp	N/A		Probe_PRND_A	N/A		Probe_ParkOut	N/A						
21	TIM_ActualMotorTrq	N/A		Probe_PRND_B	N/A		Probe_ReverseOut	N/A						
22				Probe_PRND_C	N/A		Probe_DriveOut	N/A						
23	TIM_details_AgeCoun	N/A		Probe_PRND_D	N/A		Probe_NeutralOut	N/A						
24	TIM_summary_AgeCo	N/A		Probe_PRND_E	N/A									
25	TemperatureInverter	OFF												
26	TemperatureMotor	OFF												
27	PowerFault	OFF												
28	InternalAlarm	No												
29	BatteryUnderCharge	OFF												
30														
31														
32														
33														
34														
35														
36														
37														
38														
39														

Figure 80. VIM interface.

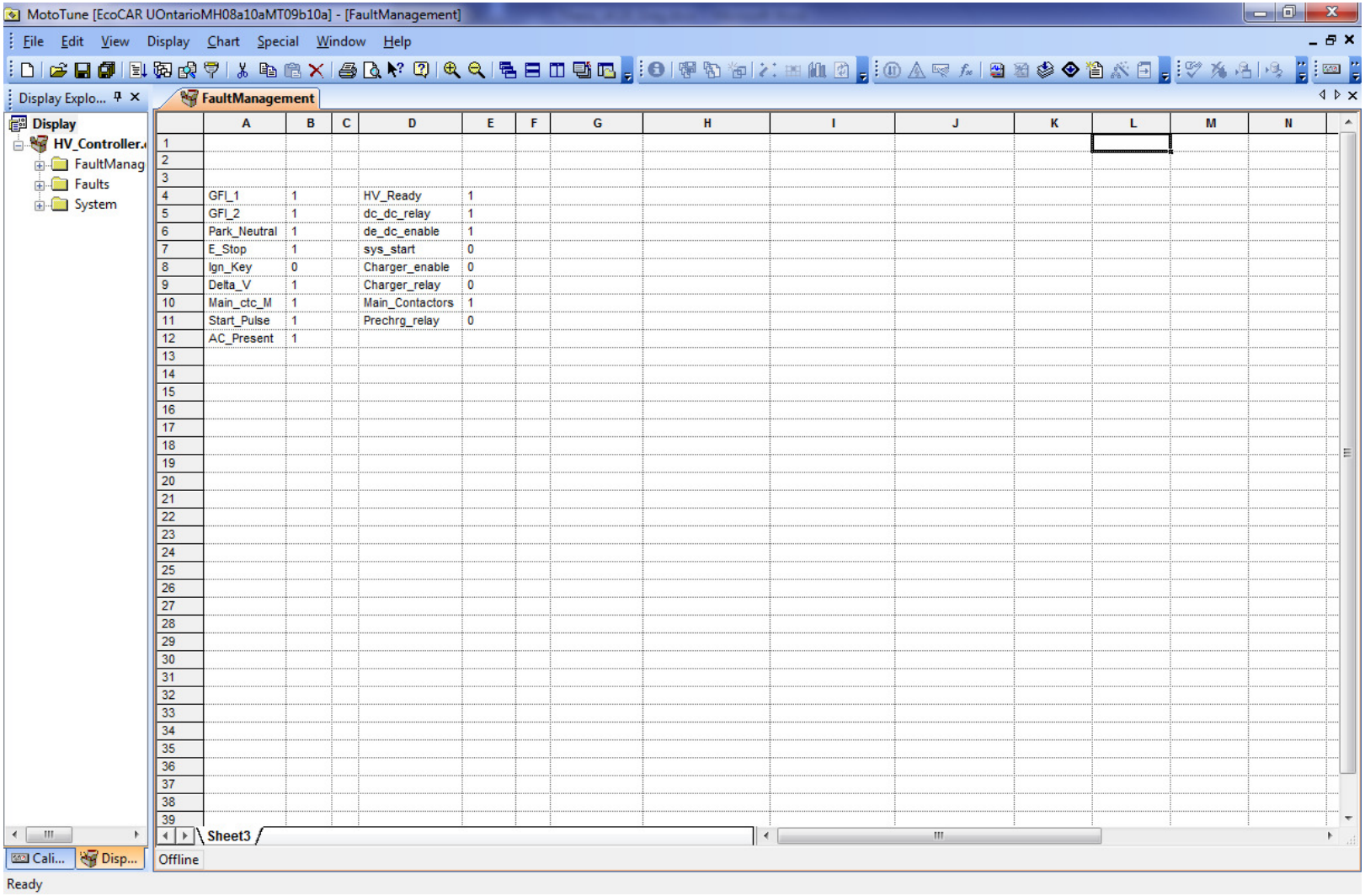


Figure 81. HVCM interface.

4. TIM600 Motor Controller RS232 Interface

Table 69. Enabling Keys.

P60	95
P99	82

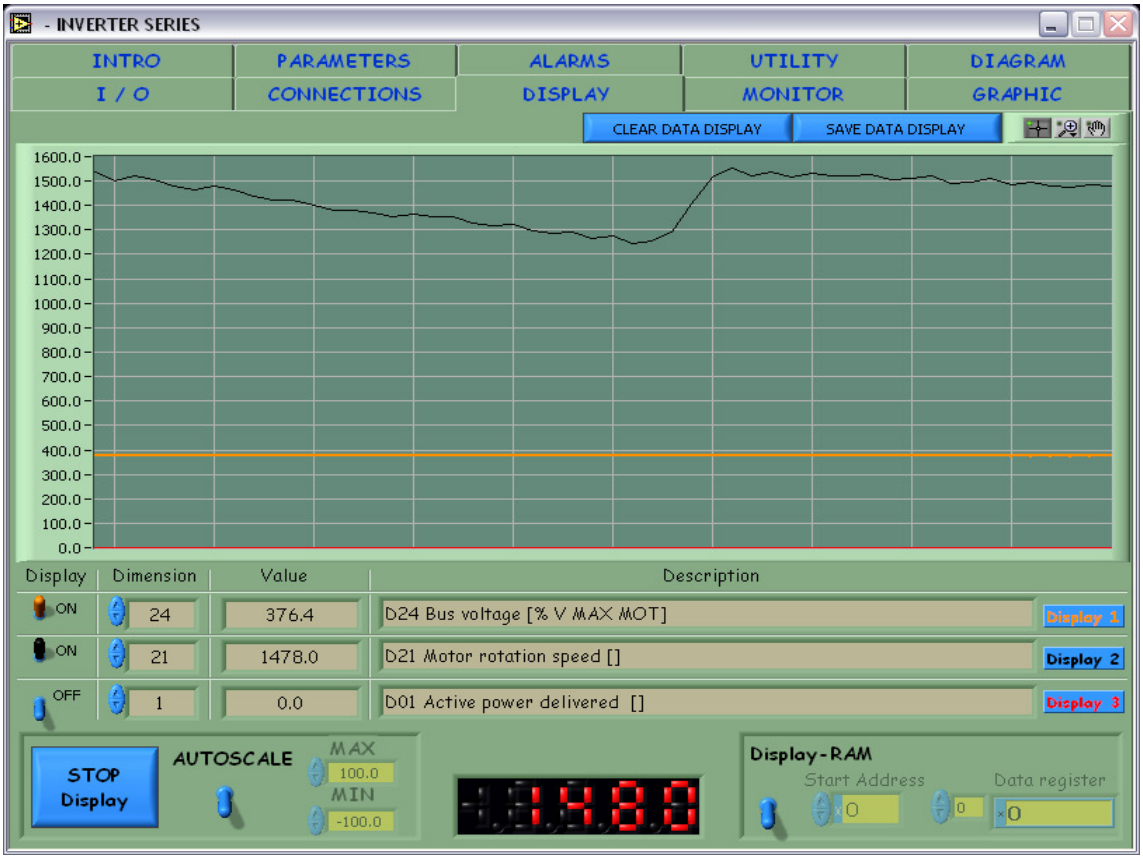


Figure 82. Display/Log Tab.

Table 70. Parameter list. 125Hz 155V.

Para-meter	Description	Range	UOIT EcoCAR Value	Default Value	Unit
P1	Accelerator pedal gain factor for 14 bit analog reference 1 (AN_INP_1)	+/- 400 = 100%	235,6	100,0	%
P2	Accelerator pedal offset factor for 14 bit analog reference 1 (AN_INP_1)	+/-16383 = 100%	-10698	0	%
P3	Gain factor for 14 bit analog reference 2 (AN_INP_2)	+/- 400 = 100%	100,0	100,0	%
P4	Offset for 14 bit analog reference 2 (AN_INP_2)	+/-16383 = 100%	0	0	%
P5	Gain factor for 14 bit analog reference 3 (AN_INP_3)	+/- 400 = 100%	100,0	100,0	%
P6	Offset for 14 bit analog reference 3 (AN_INP_3)	+/-16383 = 100%	0	0	%
P8	Motor potentiometer starting speed	+/-100.0	0,0	2,0	% n max
P9	Filter time constant for analog torque reference value	0-20	20,0	0,0	ms
P21	CW acceleration time	0.01-199.99	1,00	10,00	s
P22	CW deceleration time	0.01-199.99	1,00	10,00	s
P23	CCW acceleration time	0.01-199.99	1,00	10,00	s
P24	CCW deceleration time	0.01-199.99	1,00	10,00	s
P40	Current Limit	0-P103	150,0	200,0	% I nom AC
P42	Max torque in the positive direction of rotation	0-400	300,0	400,0	% Tnom
P43	Max torque in the negative direction of rotation	-400-0	-300,0	-400,0	% Tnom
P51	Max speed for alarm	0-125	120,0	120,0	% n max
P52	Min flux for alarm	0-100	0,9	2,0	% Phi nom
P54	Monitor sampling period	0-1999.9	20	1	ms

Para-meter	Description	Range	UOIT EcoCAR Value	Default Value	Unit
P55	Points memorized after monitor trigger	1-2000	100	1	-
P56	Monitor trigger level	-400	1,0	0,0	%
P60	Access Code for Reserved Parameters P61-P99	-	95	0	key = 95
P61	Nominal Motor Current (P53) divided by Nominal Inverter Current	1-100	66,0	100,0	% I nom
P62	Nominal Motor Voltage	100-1000	155,0	380,0	V nom motor
P63	Nominal Motor Frequency	10-800	125,0	50,0	Hz
P64	% nominal voltage at Max Speed	1.0-200.0	150,0	100,0	% Vnom AC
P65	Max RPM of Motor	50-30000	12000	2000	RPM (n max)
P67	Number of motor poles	0-12	4	4	Integer
P69	Encoder pulses per revolution	0-6000	128	1024	Integer
P91	Max motor temperature (if read with PT100)	0-150	70,0	130,0	Degree C
P95	Motor NTC or PTC resistance value for setting alarm	0-19999	4607	1500	ohms
P96	Thermal trip threshold on logic output 14	0-200 % P70	50,0	100,0	% of In
P99	Access key to TDE parameters	-	92	0	key = 82
P100	Value of acces key (P60) to reserved parameters	-	95	95	key = 95
P101	PWM frequency	2500-9000	7500	5000	Hz
P103	Drive Current Limit (per phase)	0-800.0	150,0	150,0	% I nom
P105	Bus voltage correction factor	80-120	94,8	94,8	%
P106	Minimum Battery Voltage	180-500	50,0	50,0	volts
P107	Maximum Battery Voltage	300-1200	420,0	420,0	volts
P113	Max drive current	0-2000	400,0	400,0	A
P115	Multiplication factor for motor PTC/NTC/PT100 analog reference value	0-200	200,01	200,01	-

Para-meter	Description	Range	UOIT EcoCAR Value	Default Value	Unit
P121	Acceleration time for tests 3 and 4 during auto-tune	0.3-1999.9	0,00	4,00	s
P123	Smart brake voltage cut-in level	300-850	750,0	750	volt
P163	Enable alarms	-100--100	-0,2	0,0	-
P200	Maximum torque in DRIVE	-	200,0	200,0	% Tnom
P201	Max RGEN torque in DRIVE	-	20,0	20,0	% Tnom
P202	Max RPM in DRIVE	-	12000	12000	RPM
P203	Time to reach torque value P200 in DRIVE	-	1,00	1,00	s
P204	Time to decrease the torque value from P200 to zero in DRIVE	-	0,50	0,50	s
P205	Time to reach the regenerative torque value P201 in DRIVE	-	2,00	2,00	s
P206	Time to decrease the regenerative torque value from P201 to zero in DRIVE	-	0,50	0,50	s
P207	Maximum torque in ECONOMY	-	100,0	100,0	% Tnom
P208	Max RGEN torque in ECONOMY	-	20,0	20,0	% Tnom
P209	Max RPM in ECONOMY	-	12000	12000	RPM
P210	Time to reach torque value P207 in ECONOMY	-	1,00	1,00	s
P211	Time to decrease the torque value from P207 to zero in ECONOMY	-	0,50	0,50	s
P212	Time to reach the regenerative torque value P208 in ECONOMY	-	2,00	2,00	s
P213	Time to decrease the regenerative torque value from P208 to zero in ECONOMY	-	0,50	0,50	s
P214	Maximum torque in REVERSE	-	150,0	150,0	% Tnom
P215	Max RGEN torque in REVERSE	-	20,0	20,0	% Tnom
P216	Max RPM in REVERSE	-	2000	2000	RPM

Para-meter	Description	Range	UOIT EcoCAR Value	Default Value	Unit
P217	Time to reach torque value P214 in REVERSE	-	1,00	1,00	s
P218	Time to decrease the torque value from P214 to zero in REVERSE	-	0,50	0,50	s
P219	Time to reach the regenerative torque value P215 in REVERSE	-	2,00	2,00	s
P220	Time to decrease the regenerative torque value from P215 to zero in REVERSE	-	0,50	0,50	s
P221	Maximum regenerative BRAKE torque	-	70,0	70,0	% Tnom
P222	Time to reach P221 BRAKE value	-	1,00	1,00	s
P223	Enable complementary BRAKE logic (0 = brake on)	-	0	0	logic
P224	Minimum Acceptable value of Accelerator pedal	-	2,0	2,0	-
P225	Maximum Acceptable value of Accelerator pedal	-	97,0	97,0	-
P226	Threshhold for (Throttle / Brake) transition point	-	20,0	20,0	% Accelerator
P249	Enable Analog Regenerative	0,1	0	0	logic
P250	Proportional gain control in overload	-	1,00	1,00	multiplier
P251	Maximum IGBT junction temperature in overload	-	125,0	125,0	C
P252	Velocity threshold switch (more or less filtered)	-	2,5	2,5	-
P253	Time filter for accelerator pedal	-	150,0	150,0	ms
P266	CAN Transmit enable TX0	-	1	1	logic
P267	Transmit message ID TX0	-	255	255	-
P268	Periodic transmission rate (TX0)	-	1000	1000	ms
P269	CAN Transmit enable TX1	-	1	1	logic
P270	Transmit message ID TX1	-	254	254	-
P271	Periodic transmission rate (TX1)	-	500	500	ms
P274	Receive Message ID to stop transmission	-	785	785	-

Para-meter	Description	Range	UOIT EcoCAR Value	Default Value	Unit
P275	Enable Rx message for Vmax Battery	0,1	0	0	logic
P276	Second order filter time constant for Bus Voltage	-	1,0	1,0	ms
P277	Time allowance for PWM signal inversion	-	1000	1000	7500=1 s
P278	Time to decrease the regenerative BRAKE torque to zero	-	0,50	0,50	s
C15	Meaning of programmable analog output 1 [-63 64] (current)	-	18	11	-
C16	Meaning of programmable analog output 2 [-63 64] (velocity)	-	24	4	-
C17	Meaning of analog input A.I.1 14 bit 0 = speed ref. 1 = torque ref. 2 = torque limit ref. [0 2]	0,2	1	0	-
C32	Motor thermal switch 'Block drive? [0 1]	-	0	1	-
C46	Enable temperature sensor of D26, 0=none, 1=Thermal switch,2=NTC, 3=TS+NTC	-	3	2	-
C48	CAN Baud rate, 0=1M,1=500K,2=250K,3=125K,4+100K,5+50K,6=20K,7= 10K	-	1	0	-
C52	CAN bus enable, 0 = disable, 1 = enable	-	0	0	-
C56	Type of Overload; 0=120% x 30 sec, 1=150% x 30s, 2=200% x 30 sec, 3=200% x 3 sec + 155% x 30 sec	0-3	1	3	-
C64	Enable Current Control	-	1	0	-
C69	Enable second order filter on speed governor	0-1	1	0	-
C74	Enable managing incremental encoder over time	-	1	0	-
C75	Disable autotuning starting from the defaults	-	1	0	-

5. CANLog4 USB Interface

```
System

 CAN1Timing=Timing500K
 CAN2Timing=Timing500K
 CAN3Timing=Timing500K
 CAN4Timing=Timing40K

 CAN1Output=OFF
 CAN2Output=OFF
 CAN3Output=OFF
 CAN4Output=OFF

Output
 LED1 = (1)
 LED2 = (0)
 LED3 = (0)
 LED4 = (1)

Recordfilter
 exclude CAN1 000 - 7ff
 exclude CAN2 000 - 7ff
 exclude CAN3 000 - 7ff
 exclude CAN4 000 - 7ff

 include CAN1 0FA CAN1 0FB CAN1 0FC

END
```

Figure 83. Log Task Language (LTL) Code Programmed in the CANlog4.

Appendix J

Vehicle Integration Module Models

1. VIM Port Description

Table 71. VIM Port description (HVCM and SCM).

			ECM5554			
Pin	**Colour**	**Used**	**Inputs**	**Outputs**	**Simulink**	**Description**
TIM: 20		EST1	EST1	EST1	Discrete Ouput	Park
TIM: 21		EST2	EST2	EST2	Discrete Ouput	Reverse
TIM: 22		EST3	EST3	EST3	Discrete Ouput	Drive
TIM: 23		EST4	EST4	EST4	Discrete Ouput	Brake
TIM: 24		EST5	EST5	EST5	Discrete Ouput	Traction En
TIM: 25		EST6	EST6	EST6	Discrete Ouput	Emergency
Moto: 3	Org	AN01	AN01		Analog Input	Ign pot
Moto: 4	Blk-wht	AN02	AN02		Analog Input	Brake pot
Moto: 5	Red-Blk	AN03	AN03		Analog Input	Accel pot
Moto: 6	Wht-Blk	AN05	AN05		Digital Input	Interlock: E-Stop
Moto: 7	Wht	AN06	AN06		Digital Input	Interlock: HV conn
Moto: 8	Wht-Blk	AN07	AN07		Digital Input	Interlock: Lid Front ESS
Moto: 9	Wht	AN08	AN08		Digital Input	Interlock: GFI Alarm 1
Moto: 10	Wht-Blk	AN09	AN09		Digital Input	Interlock: GFI Alarm 1
Moto: 11	Blu	AN10	AN10		Digital Input	Charger: AC_On
Moto: 16	Blu-Wht	AN15	AN15		Digital Input	Charger: Proximity
Moto: 18	Grn	AN22	AN22		Digital Input	Encoder: A
Moto: 19	Grn-Wht	AN23	AN23		Digital Input	Encoder: B
Moto: 20	Grn-Blk	AN24	AN24		Digital Input	Encoder: C
Moto: 21	Dk Grn	AN25	AN25		Digital Input	Encoder: D
Moto: 30	Red-Blk	LSO4	LSO4		Discrete Ouput	Contactor: ESS-, Precharge
Moto: 31	Red-Wht	LSO5	LSO5		Discrete Ouput	Contactor: ESS+, DC/DC1, DC/DC2
Moto: 32	Blu-Wht	LSO6	LSO6		Discrete Ouput	Contactor: Charger
Moto: 35		LSO9	LSO9		PWM Ouput	Radiator: Fans

2. VIM Simulink Model

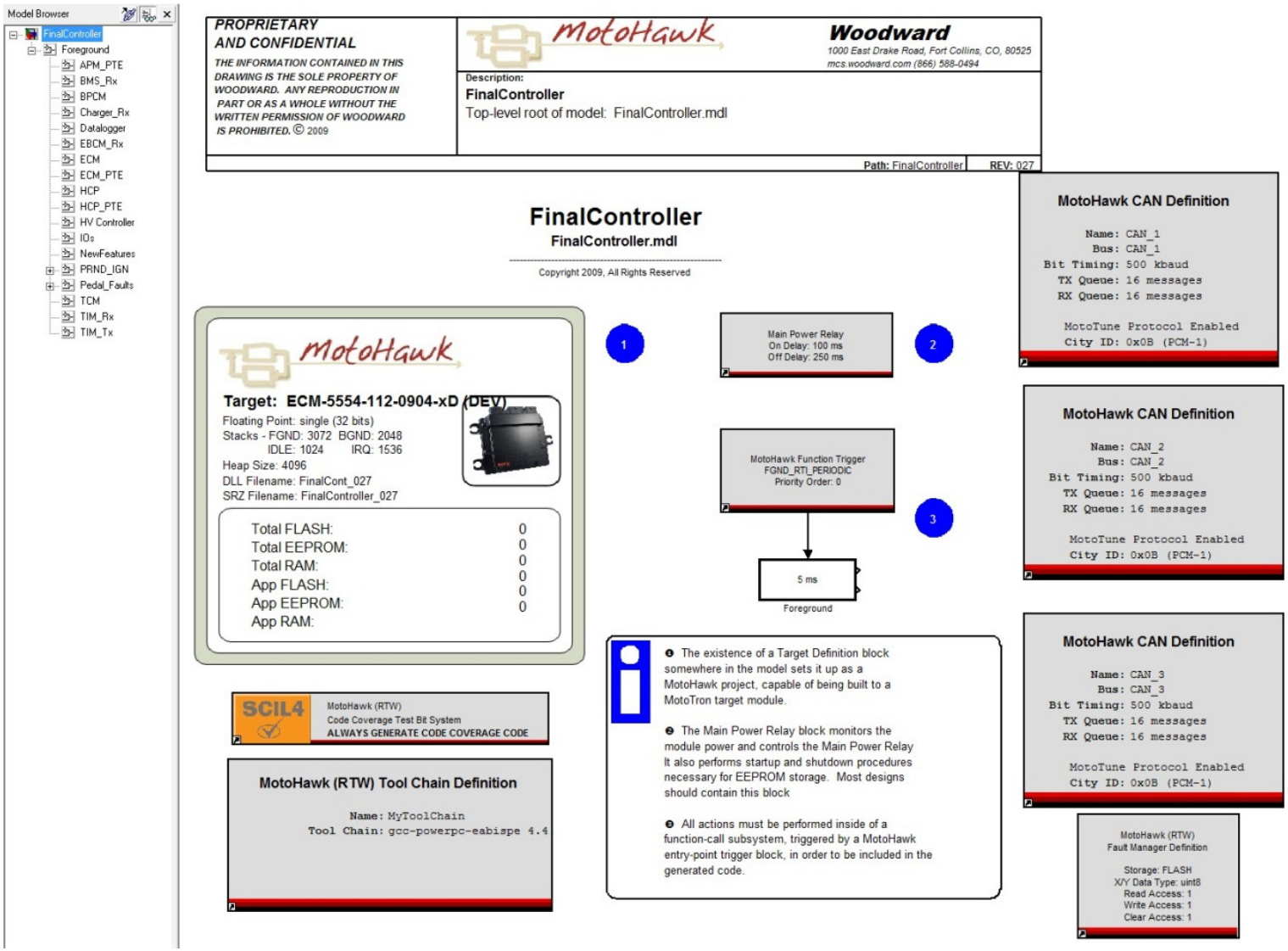


Figure 84. Top Level Subsystem.

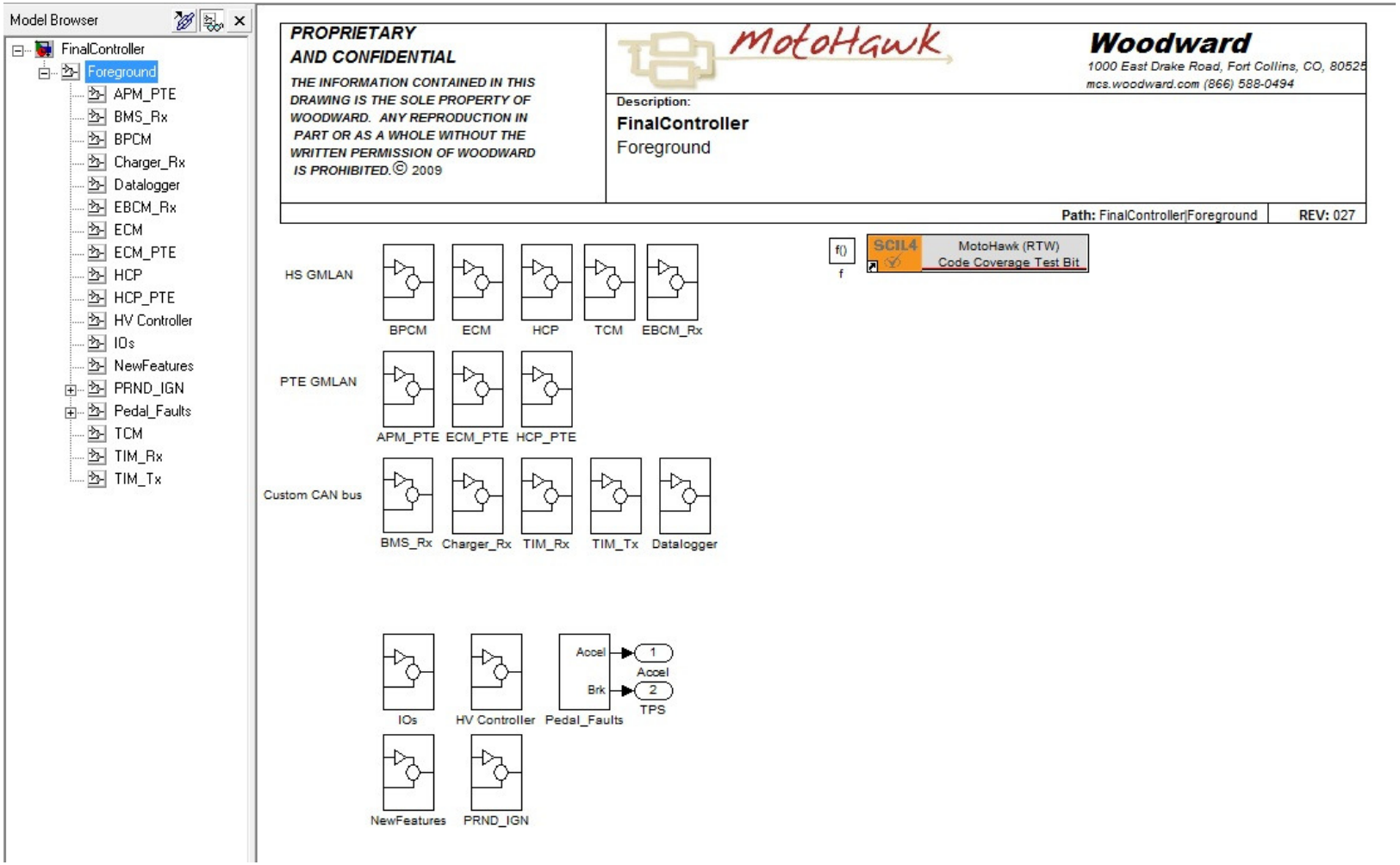


Figure 85. Main program.

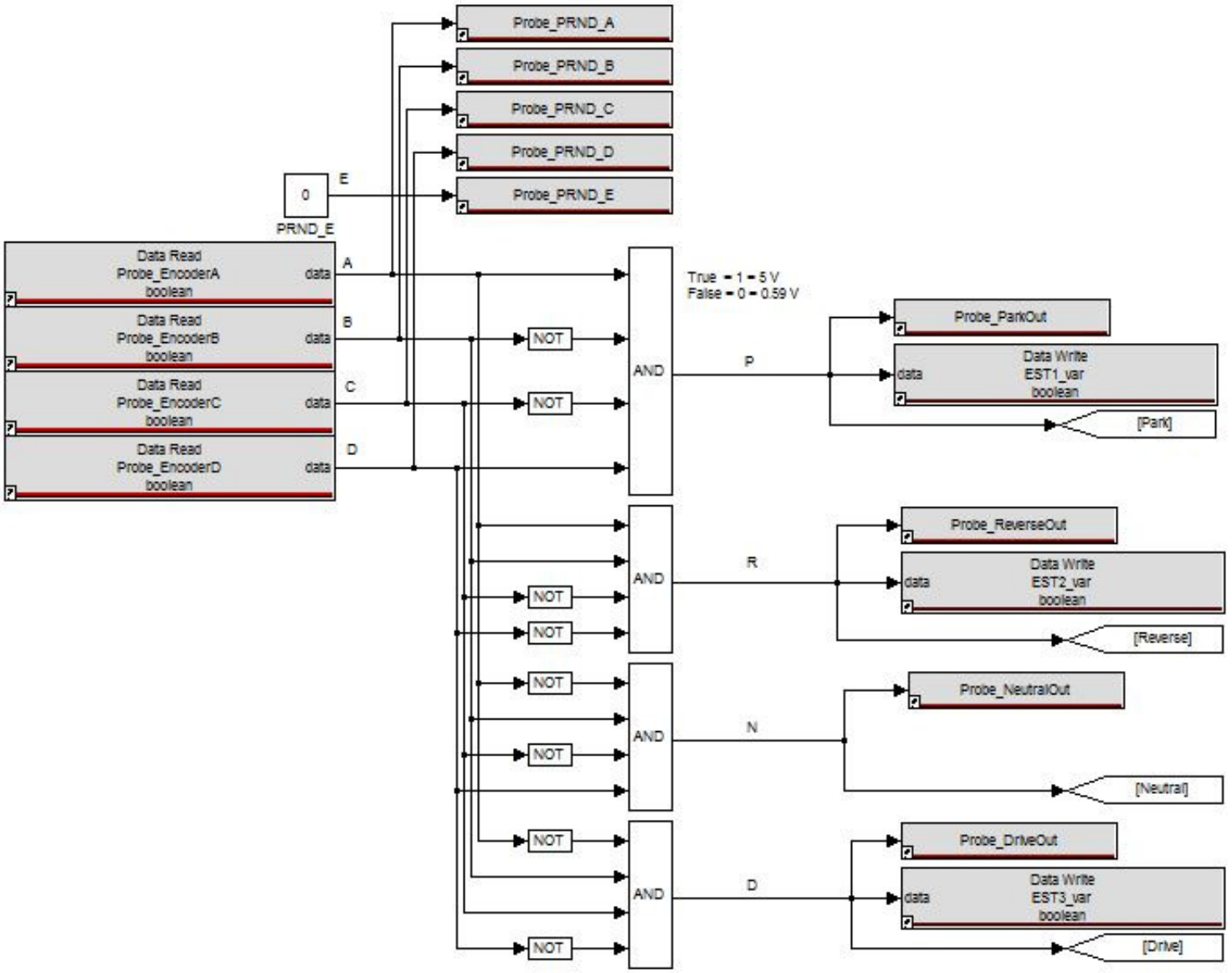


Figure 86. Shift Lever Decoder Model.

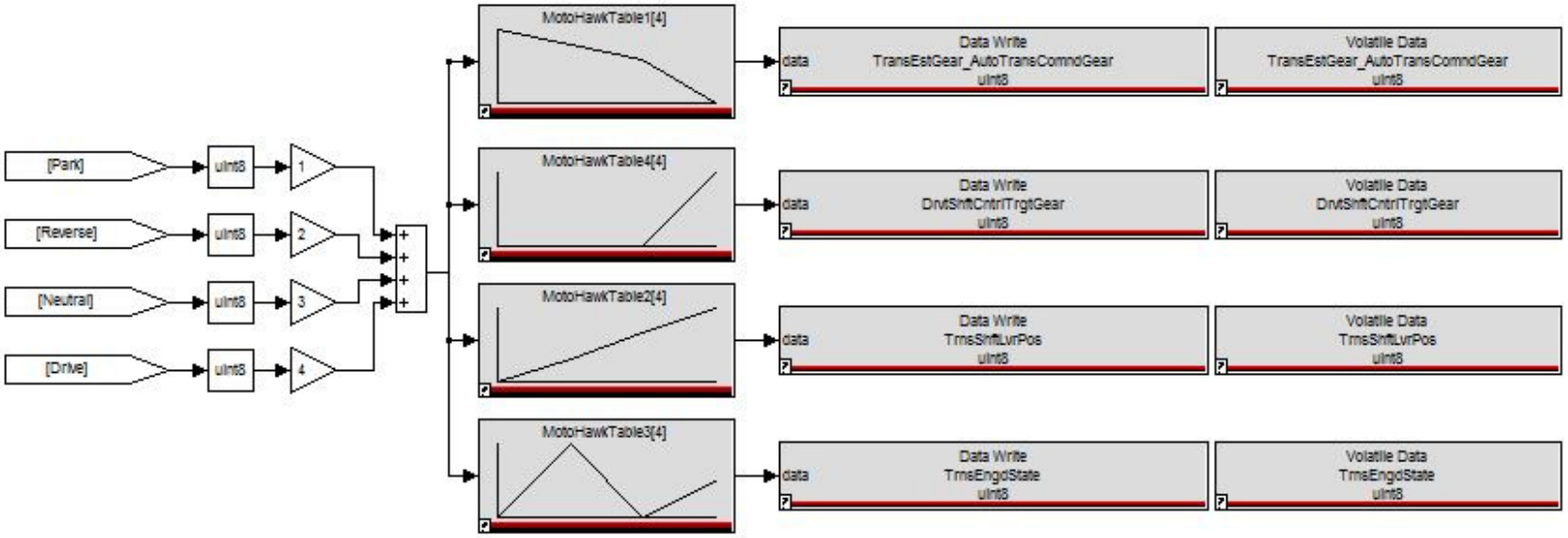


Figure 87. Shift Lever Translator Model.

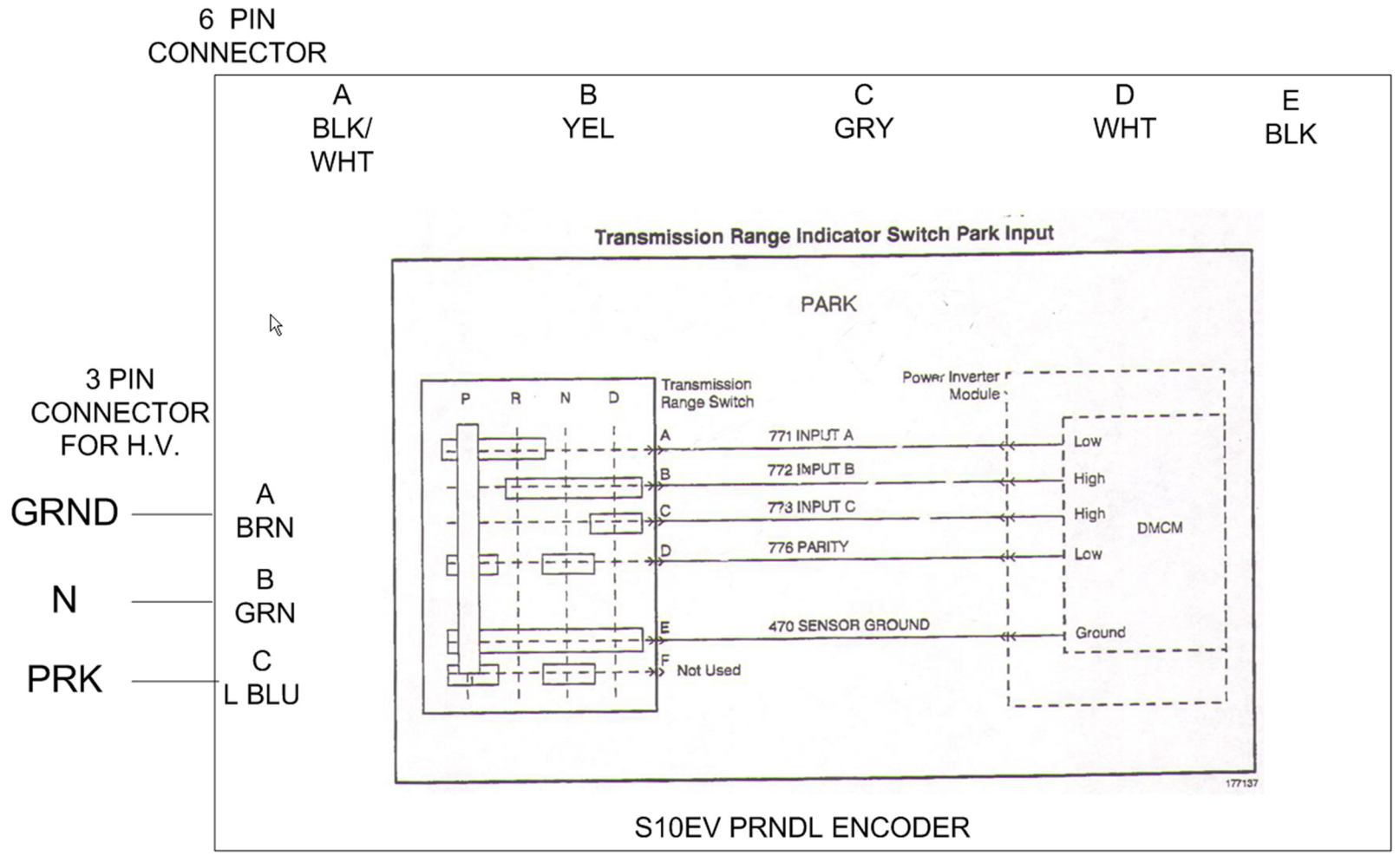


Figure 88. Schematic of States [service manual].

Note: In the UOIT vehicle, GRND (3 pin connector) and E (6 pin connector) are not used as a ground reference, but are powered with 5V instead. N and PRK (3 pin connector) are connected to the shift lever, whereas A, B, C and D (6 pin connector) are connected to the MotoTron analog inputs for decoding. Once decoded, the MotoTron discrete outputs are updated and the TIM sees the decoded value of the shift lever.

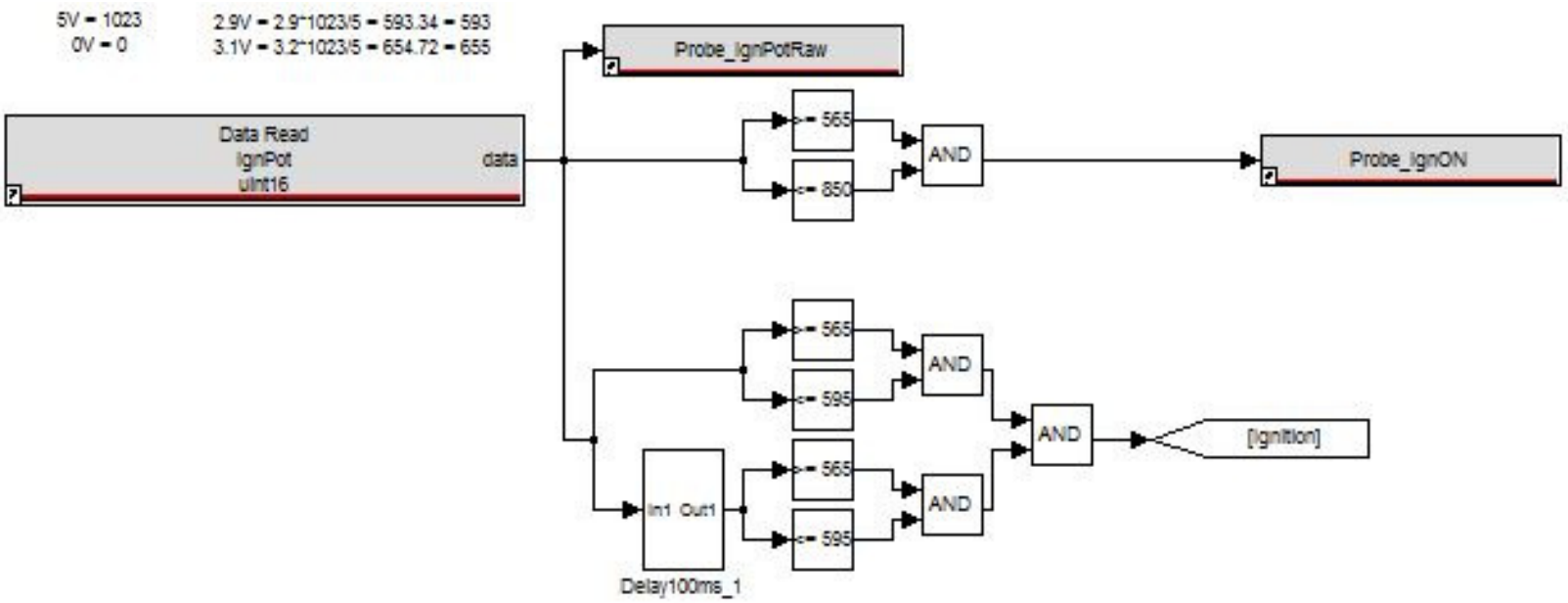


Figure 89. Ignition Key Decoder Model.

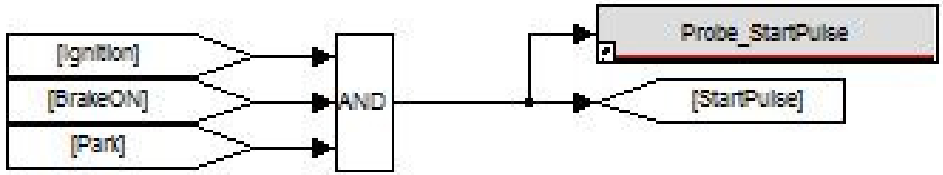
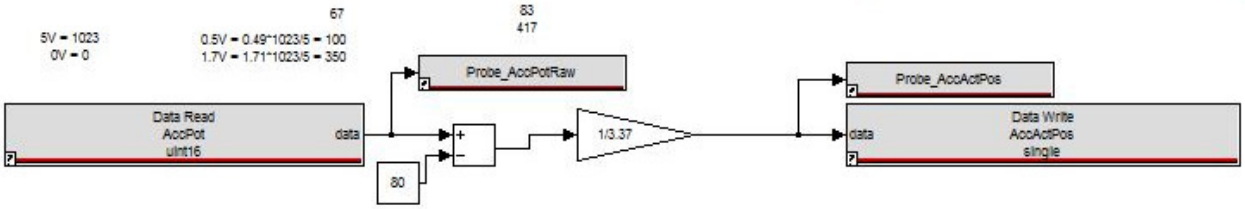
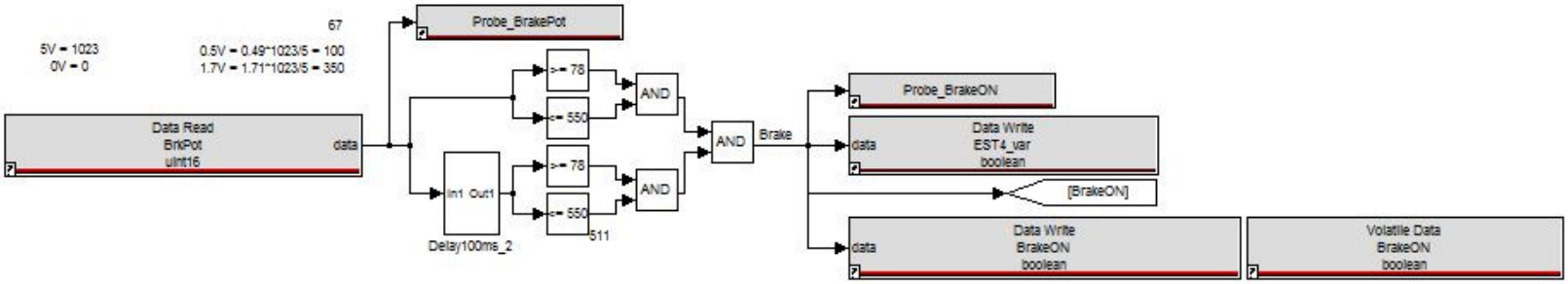


Figure 90. Ignition Cranking Conditions.



(a)



(b)

Figure 91. Pedal Decoder Model (a) Acceleration (b) Brake.

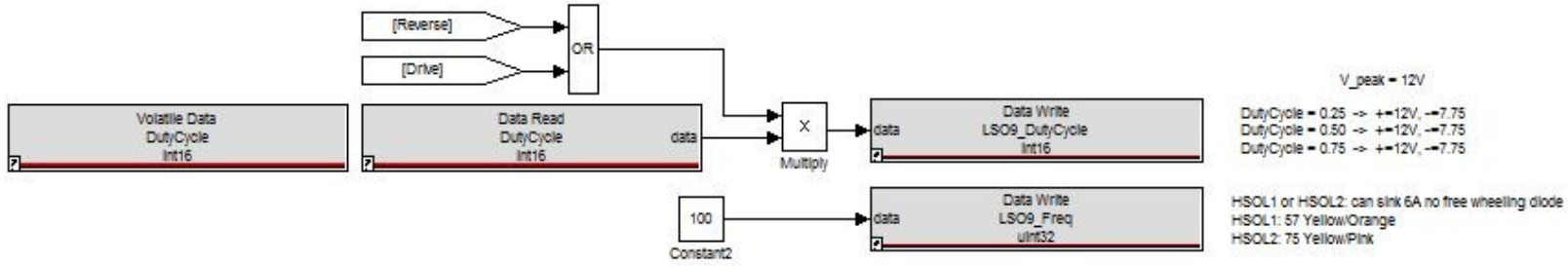


Figure 92. Radiator Fan Controls.

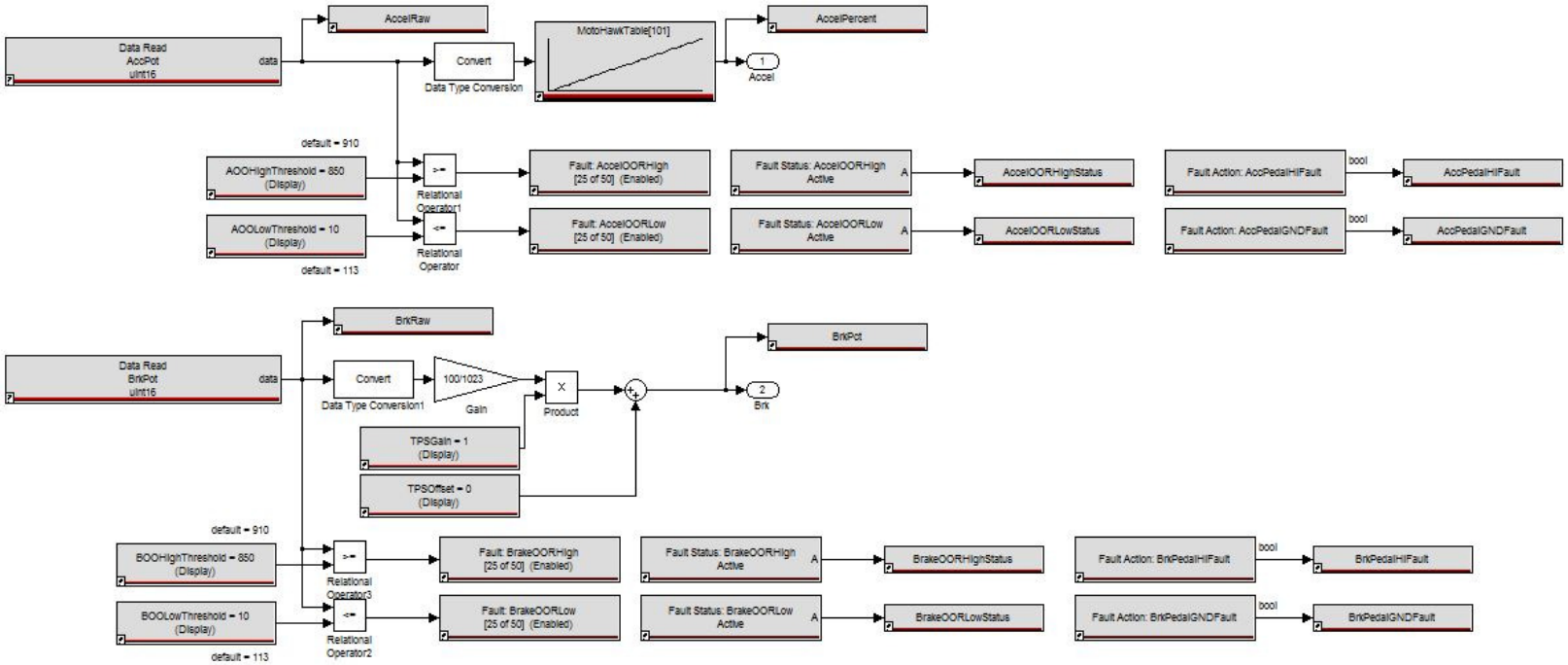


Figure 93. Pedals Fault Management Model.

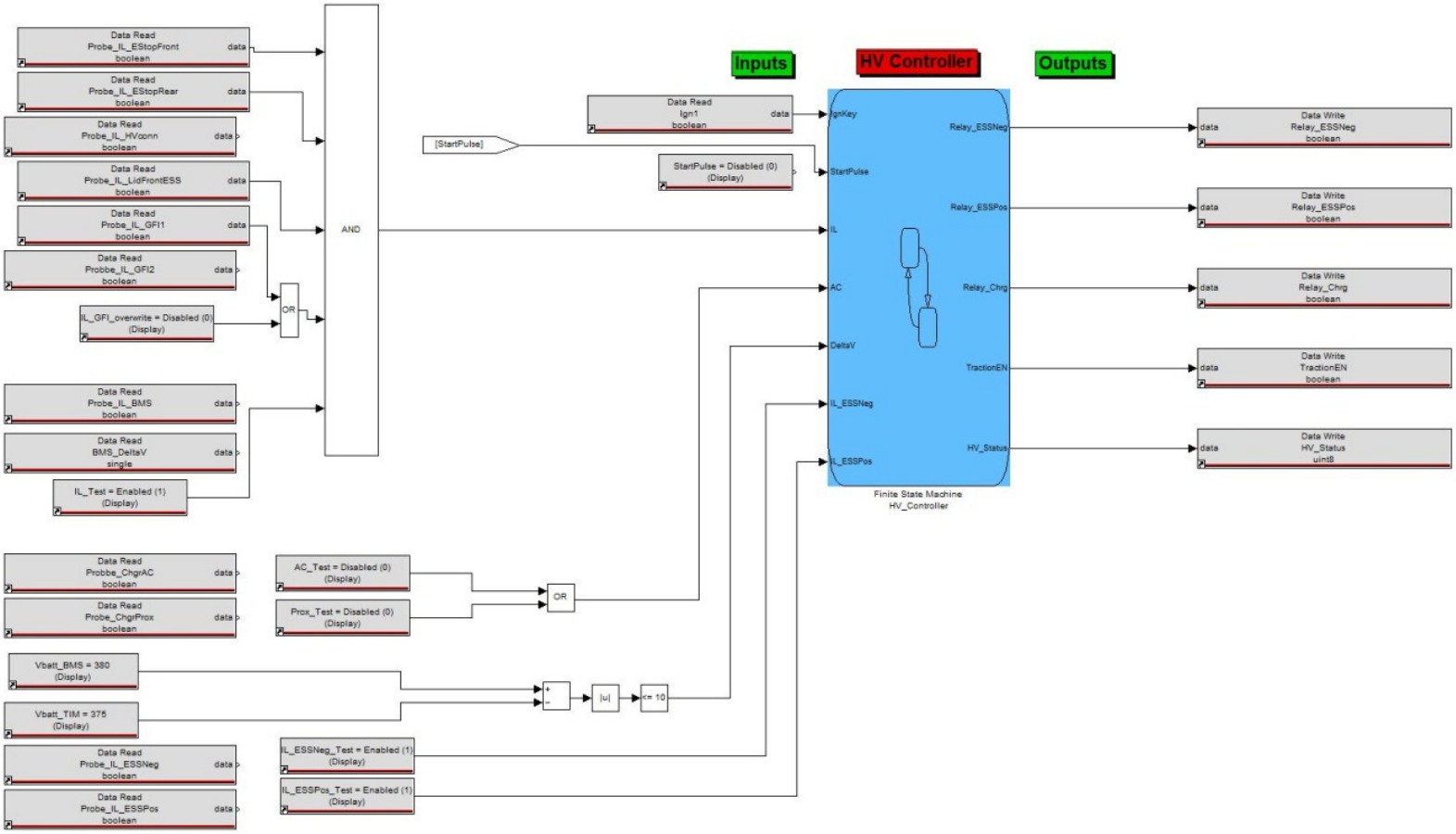


Figure 94. High Voltage Controller Inputs and Outputs.

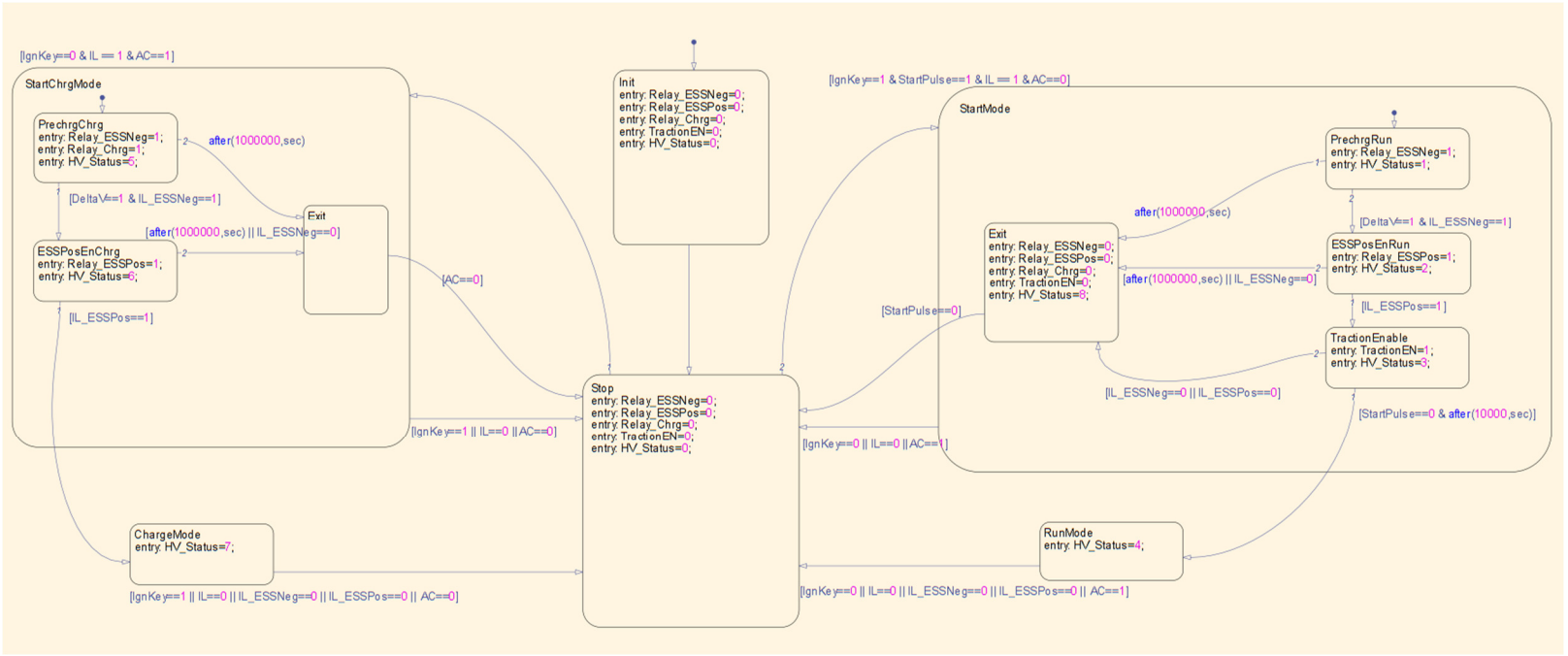


Figure 95. HV Control Module Stateflow.

3. Software Versions Used

The software versions used with Windows 7 to program the VIM are:

- Matlab/Simulink 2009b
- MotoHawk_2009b_ML7p5_to_ML7p9.exe
- MotoServerRuntime RELEASE 8.13.7.140.exe
- MotoTune RELEASE 8.13.7.140.exe
- gcc-powerpc-eabispe-4_4_0-SP1.exe
- kvaser_drivers_w2k_xp_w7.exe

Appendix K

Instrument Panel Cluster

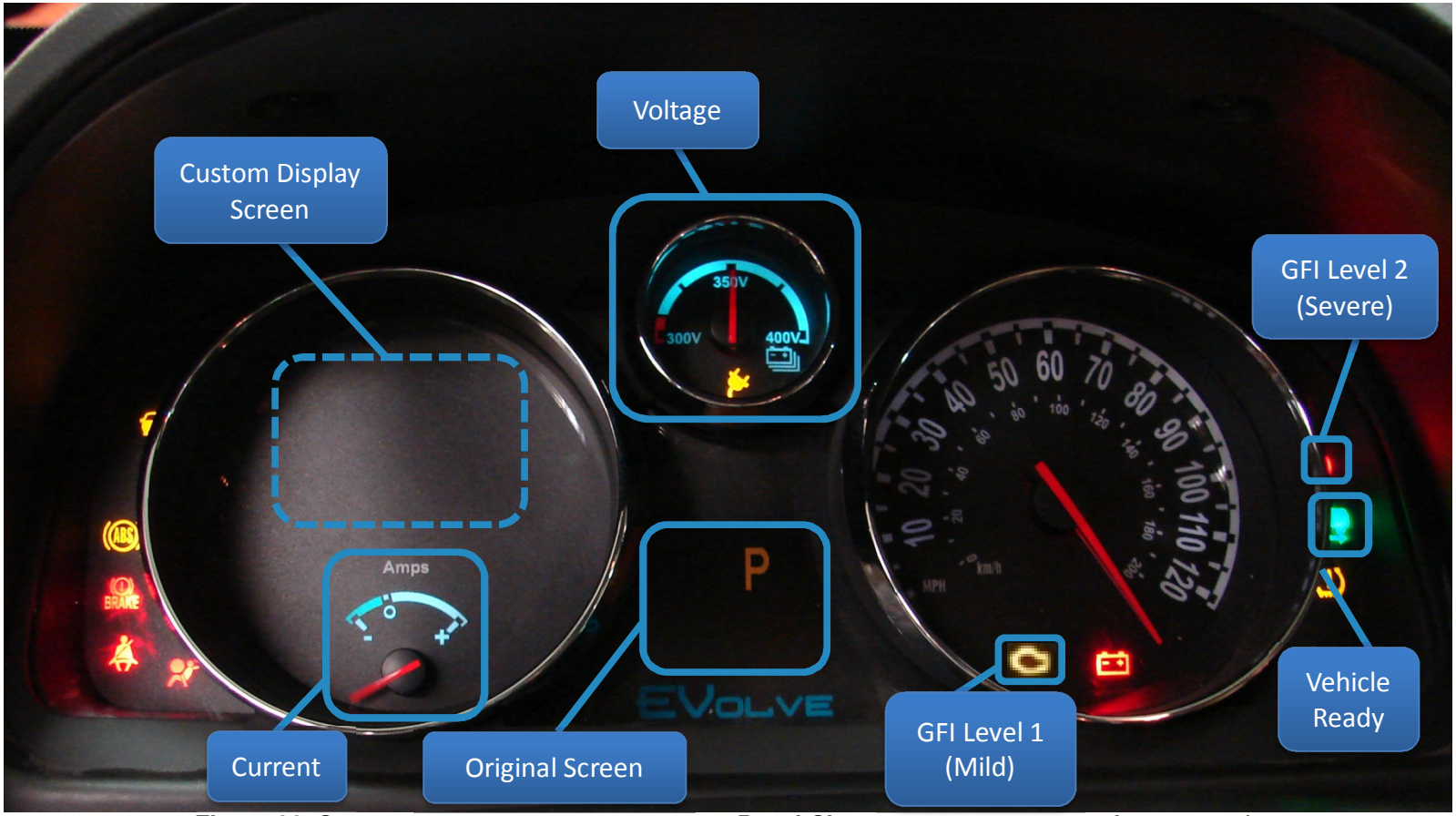


Figure 96. Custom Faceplate of the Instrument Panel Cluster (without the 2 display screens).

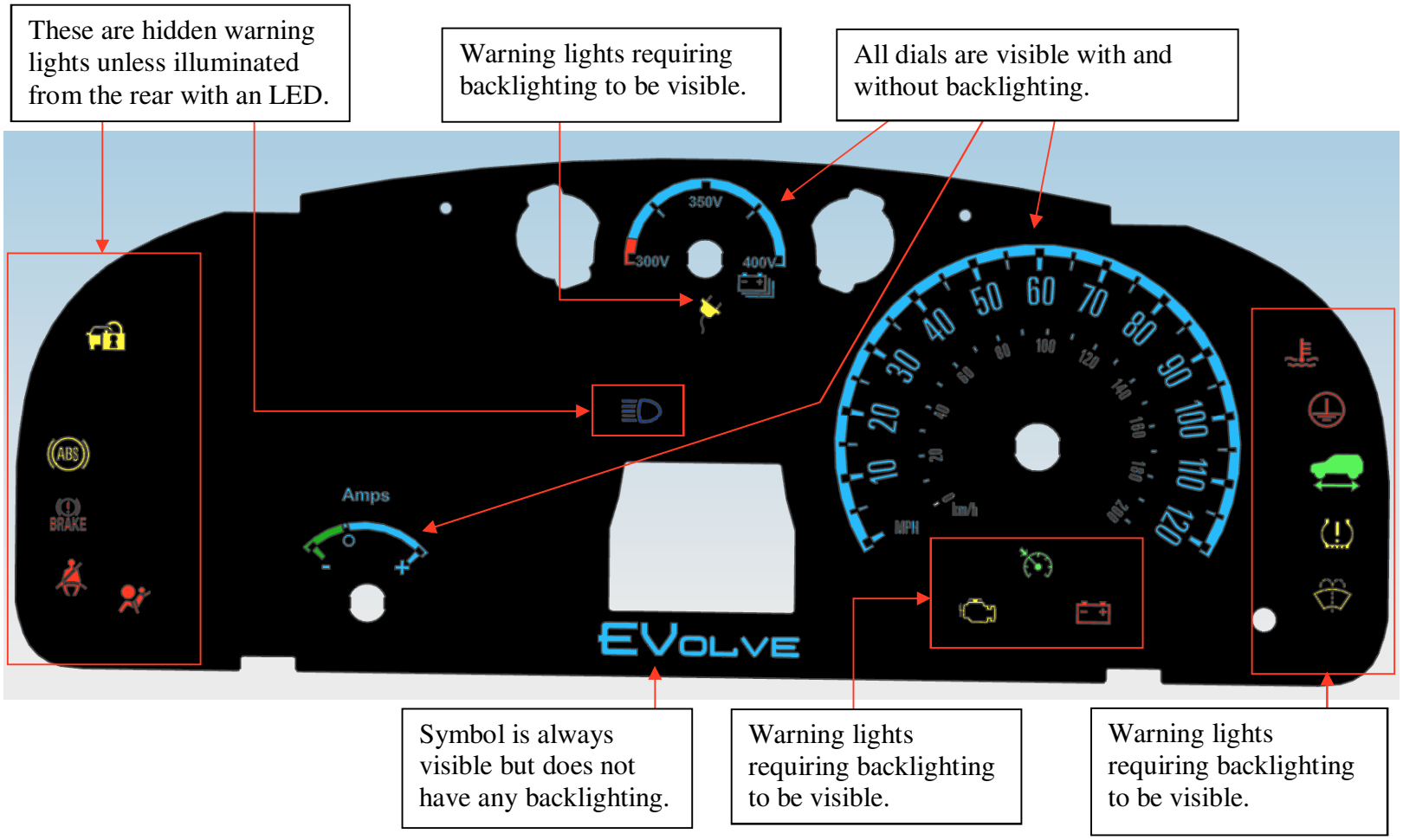


Figure 97. Custom Faceplate of the Instrument Panel Cluster (without the 2 display screens).

Figure 98 shows what the faceplate looks like when no warning lights are ON. All these areas are visible during the daytime when the lighting is not effective, and also when the backlight is ON at night. As for Figure 97, the 2 display screens are not represented on Figure 98.

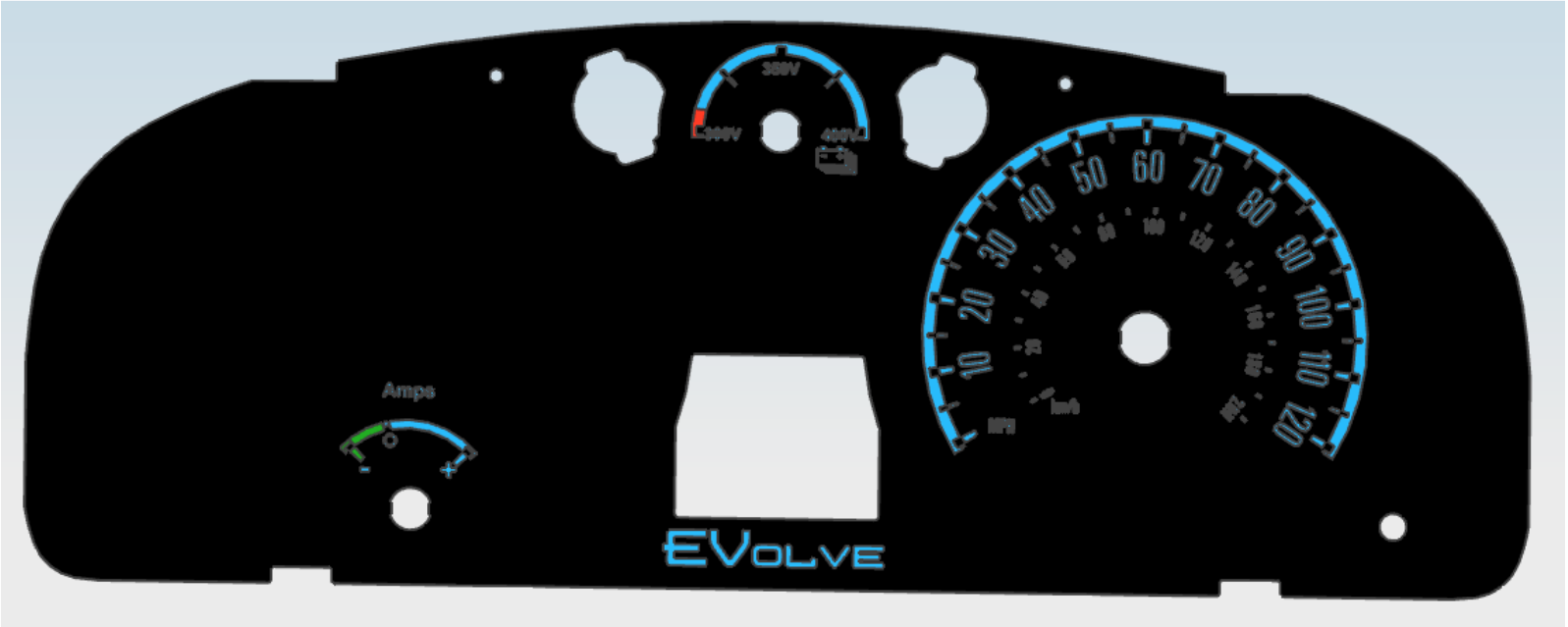


Figure 98. Custom Faceplate of the Instrument Panel Cluster – Warning Symbols Hidden (without the 2 display screens)

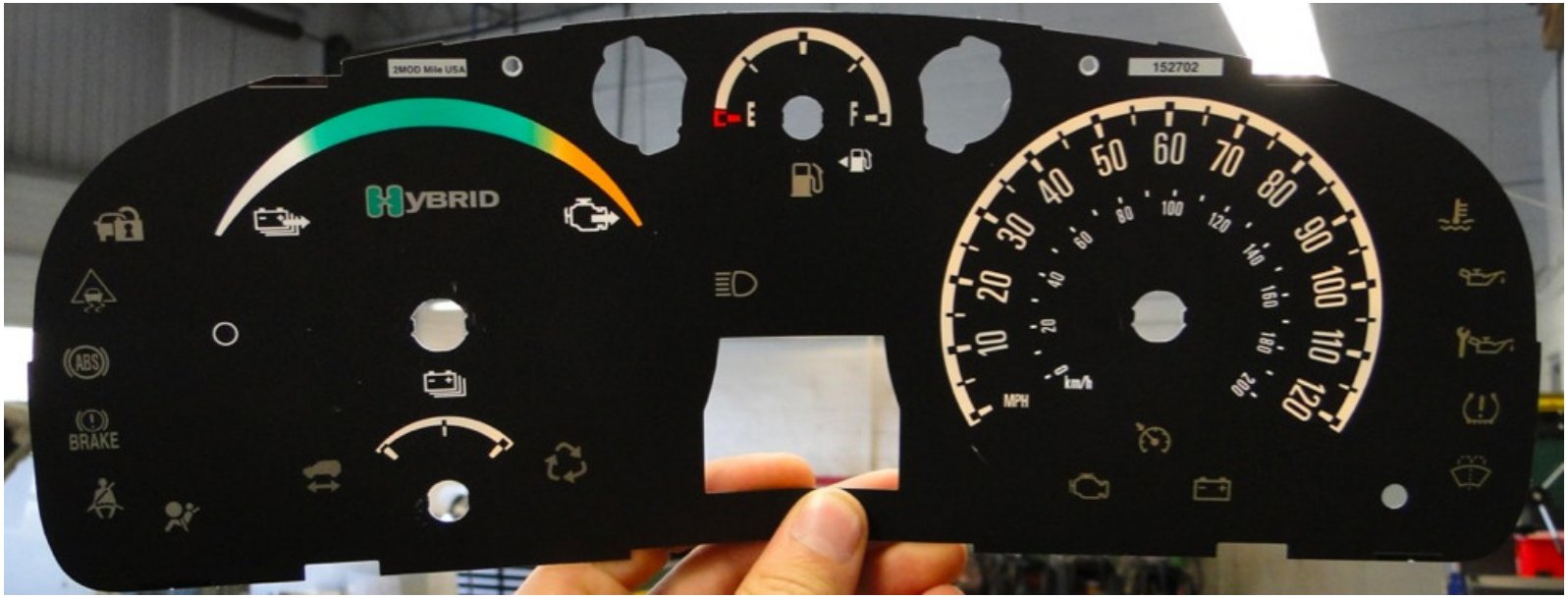


Figure 99. Stock Faceplate of the Instrument Panel Cluster.

Appendix L

UOIT CAN bus dictionary

Table 72. UOIT CAN bus Dictionary.

BMS Tx		TIM		BRUSA Tx		Datalogger Tx	
300h + Module ID	300	TIM Tx	0FEh		611h		0FAh
	301		0FFh				0FBh
	302	TIM Rx	302h				0FCh
	303		303h				
	304		304h				
	305						
3FFh	3FF						
580h + Module ID	580						
	581						
	582						
	583						
	584						
	585						
400h + Module ID	400						
	401						
	402						
	403						
	404						
	405						
500h	500						
600h	600						
618h	618						
630h	630						
650h + Module ID	650						
	651						
	652						
	653						
	654						
	655						
680h	680						
690h + Module ID	690						
	691						
	692						

	693			
	694			
	695			
790 + Module ID	790			
	791			
	792			
	793			
	794			
	795			