System Design for Mechatronics

Team 28, Controls Freaks
Abhishek Magdum
Dharak Verma
Jason Surendran
Laura Yang
Derek Paylor

January 19, 2023

Revision History

Date	Version	Notes
January 18, 2023	1.0	First Version

Contents

List of Tables

List of Figures

1 Introduction

The goal of this capstone project is to simulate the vehicle controls system for the McMaster Formula Electric Vehicle, as stated in the Problem Statement and Goals documentation. While taking into account the Requirements, Hazard Analysis, and Verification and Validation Plan, readers of this document and members of the McMaster Formula Electric team should be able to:

2 Purpose

The purpose of this System Design document is to demonstrate the components of the system and will serve as a guide for the implementation of the control system, which was originally established in the problem statement and goals document. The document examines the software requirements stated in the SRS document and builds on them through further explaining how the user interacts with the system Furthermore the hardware, electrical, and communication components of the system will be examined in related to how they interact with each other and the software. This document should also contextualize the tests which will be conducted as specified in the Vnv documents.

The following objectives will be achieved:

- Allow quarter scale Formula 1 style electric car to operate in basic driving conditions
- The control system will need to be compartmentalized into subsystems: Driver Interface, Governor, Vehicle Dynamics, Motor Interface, and Battery Monitor
- Development will be conducted in a model-based environment via MATLAB/Simulink
- All subsystems will need to be tested separately and in conjunction as specified in the VnV Plan and Report

3 Scope

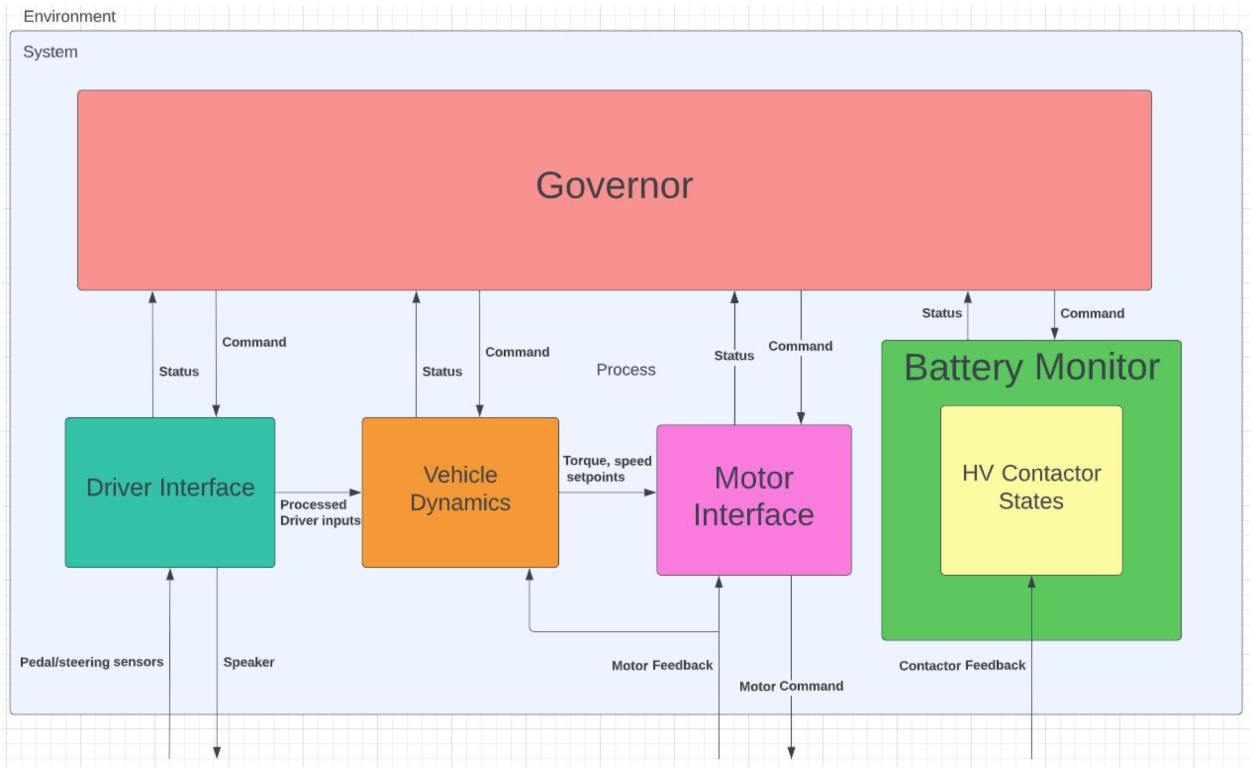


Figure 1: System context figure, depicting the boundary between the control system and external environment

4 Project Overview

4.1 Normal Behaviour

Normal behaviour for this system constitutes the vehicle's hardware and embedded systems functioning as intended, with the vehicle in regular driving scenarios (parked, idling, driving). Mechanical vehicle hardware (chassis, brakes, suspension, etc.) have not failed or yielded. Electrical vehicle hardware (sensors, PCBs, etc.) is powered and is producing necessary outputs. Vehicle controllers (computers) are operating and running their software. Communication systems (CAN SPI busses, etc.) are transmitting data between the controllers as expected.

4.2 Undesired Event Handling

Undesired events constitute a violation of the aforementioned normal operating conditions (Section 4.1). Our system is capable of monitoring the vehicle's systems only through the analog and digital signals it receives. Therefore, we are able to monitor the voltage of driver input sensors (pedals, steering) for unexpected/error behavior, as well as monitoring if CAN communication from the BMS or AMK motor kit is lost. If such errors are detected, the control system will respond accordingly, which in most cases means coasting down (enforcing 0 acceleration from the motors) and allowing the driver to use the mechanical brakes and steering to bring the vehicle to a stop. If an undesired event happens that the control system has no visibility of (eg. mechanical failure, or more likely an electrical failure outside of our input sensors), the vehicle relies on fail-safe procedures for which there are extensive guidelines from SAE, who verify these systems through comprehensive vehicle inspections at competition events.

4.3 Connection Between Requirements and Design

The following requirements from the Software Requirements Specification were the most influential on system design:

NFR10	If the vehicle is in motion while an unsafe condition occurs, the control system shall zero the motor torque commands until a system reset.
Rationale	The vehicle's "safe state" should be to allow the driver to coast or brake to a safe location, without issuing any positive/negative torque command from the motors.

NFR5	The control system modules shall incorporate modularization principles, including encapsulation and information hiding.
Rationale	Having modules and sub-modules interact through well-defined inputs/outputs, while hiding their implementation, facilitates maintainability by allowing module implementation updates to not drive downstream changes.

MSR4	Mode Selection shall adhere to Activation Sequence rule FSAE EV.10.4.2 ["Traction System Active"] • Definition – High Voltage is present outside of the Accumulator Container • Tractive System Active must not be possible until both: (1) GLV System is Energized; (2) Shutdown Circuit is Closed
Rationale	Competition rule
Likelihood of Change	Very unlikely - Competition rule

MSR5	Mode Selection shall adhere to Activation Sequence rule FSAE EV.10.4.3 [”Ready to Drive”] • Definition – the Motor(s) will respond to the input of the APPS • Ready to Drive must not be possible until: (1) Tractive System Active; (2) The brake pedal is pressed and held to engage the mechanical brakes; (3) The driver performs a manual action to initiate Ready to Drive such as pressing a specific button in the cockpit
Rationale	Competition rule
Likelihood of Change	Very unlikely - Competition rule

(Note: mention of ”Mode Selection” no longer applies - this module has since been renamed to ”Governor”)

These four requirements heavily inform the software system’s architecture (see Section 3) of several ”Layer 1” modules that encapsulate a specific subsystem’s functions, with a supervisory controller at ”Layer 2”, the Governor, that monitors the state of each subsystem, and issues commands if state changes are needed. For example, MSR4 (determining ”Tractive System Active”) is implemented by the Motor Interface and Battery Monitor reporting to the Governor that they are both are in a running state. MSR5 (determining ”Ready To Drive”) is implemented when Tractive System Active is valid, as well as the Driver Interface reporting to the Governor that the driver has requested the motors to respond. In cases such as NFR10, an ”unsafe condition” would be detected by a Layer 1 system (eg. a motor error), report this to the Governor, and the Governor can command all the subsystems accordingly.

5 System Variables

5.1 Monitored Variables

Variable	ptCAN_AMKActualValues1_Right
Description	First CAN message received from the right-rear motor. See Figures 2 & 3, an excerpt from the AMK motor documentation, for a breakdown of the signals within this message.
Units	N/A - see Figures 2 & 3

Variable	ptCAN_AMKActualValues2_Right
Description	Second CAN message received from the right-rear motor. See Figure 4, an excerpt from the AMK motor documentation, for a breakdown of the signals within this message.
Units	N/A - see Figure 4

Variable	ptCAN_AMKActualValues1_Left
Description	First CAN message received from the left-rear motor. See Figures 2 & 3, an excerpt from the AMK motor documentation, for a breakdown of the signals within this message.
Units	N/A - see Figures 2 & 3

Variable	ptCAN_AMKActualValues2_Left
Description	Second CAN message received from the left-rear motor. See Figure 4, an excerpt from the AMK motor documentation, for a breakdown of the signals within this message.
Units	N/A - see Figure 4

Variable	DI_V_SteeringAngle
Description	Reading from the steering angle sensor (potentiometer).
Units	Voltage, remapped through a ADC

Variable	DI_V_BrakePedalPos
Description	Reading from the brake pedal sensor (potentiometer).
Units	Voltage, remapped through a ADC

Variable	DI_b_DriverButton
Description	Reading from the driver 'vehicle start' button (digital IO pin input).
Units	Boolean

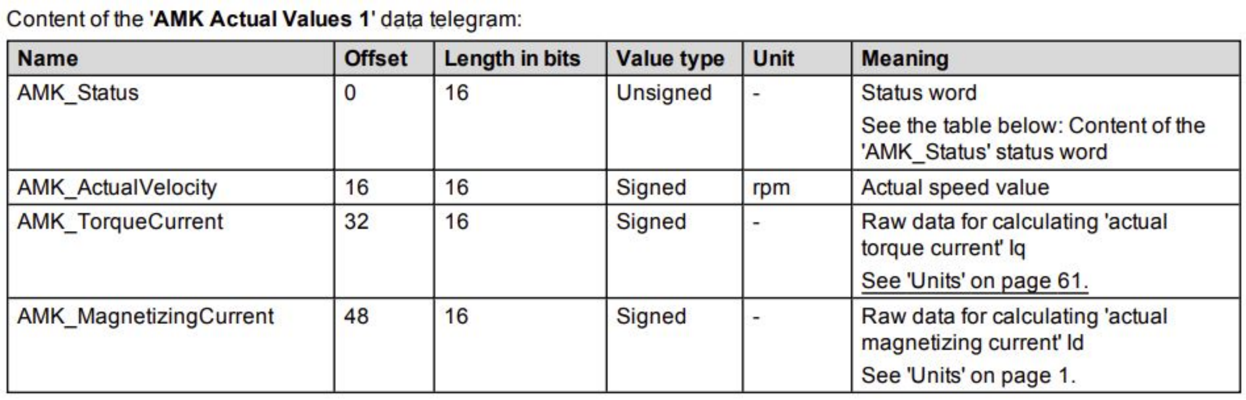
Variable	DI_V_AccelPedalPos1
Description	Reading from the first accelerator pedal sensor (potentiometer).
Units	Voltage, remapped through a ADC

Variable	DI_V_AccelPedalPos2
Description	Reading from the second accelerator pedal sensor (potentiometer).
Units	Voltage, remapped through a ADC

Variable	BM_b_prechrgContactorSts
Description	Received from the BMS through CAN. Indicates if the HV battery's precharge contactor is closed (true).
Units	Boolean

Variable	BM_b_HVposContactorSts
Description	Received from the BMS through CAN. Indicates if the HV battery's positive contactor is closed (true).
Units	Boolean

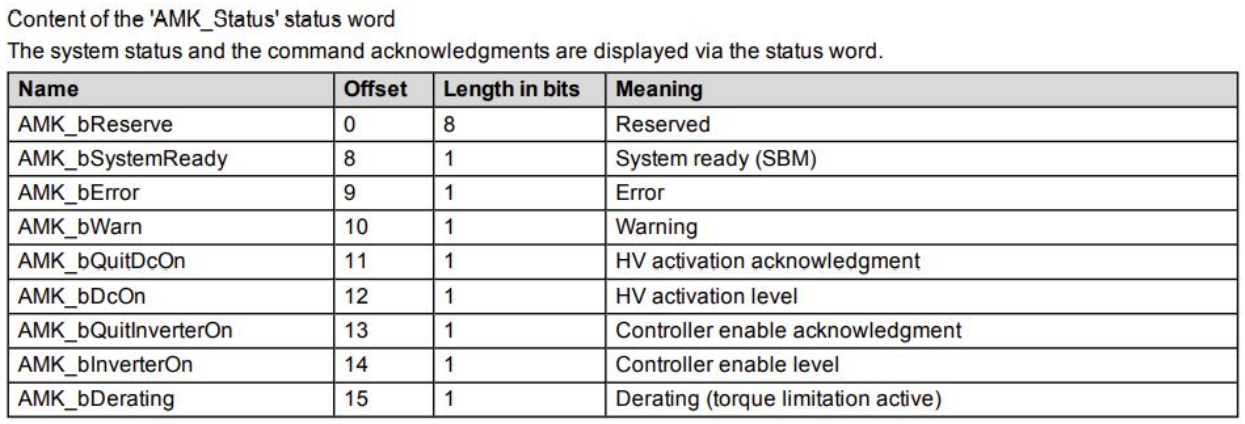
Variable	BM_b_HVnegContactorSts
Description	Received from the BMS through CAN. Indicates if the HV battery's negative contactor is closed (true).
Units	Boolean



Content of the **'AMK Actual Values 1'** data telegram:

Name	**Offset**	**Length in bits**	**Value type**	**Unit**	**Meaning**
AMK_Status	0	16	Unsigned	-	Status word See the table below: Content of the 'AMK_Status' status word
AMK_ActualVelocity	16	16	Signed	rpm	Actual speed value
AMK_TorqueCurrent	32	16	Signed	-	Raw data for calculating 'actual torque current' Iq See 'Units' on page 61.
AMK_MagnetizingCurrent	48	16	Signed	-	Raw data for calculating 'actual magnetizing current' Id See 'Units' on page 1.

Figure 2: AMK_ActualValues1 signals

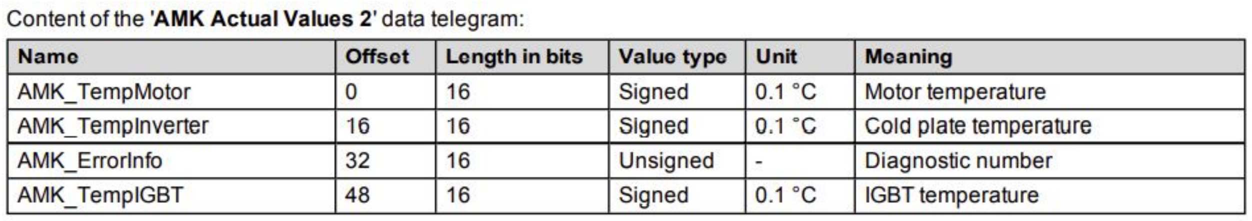


Content of the 'AMK_Status' status word

The system status and the command acknowledgments are displayed via the status word.

Name	**Offset**	**Length in bits**	**Meaning**
AMK_bReserve	0	8	Reserved
AMK_bSystemReady	8	1	System ready (SBM)
AMK_bError	9	1	Error
AMK_bWarn	10	1	Warning
AMK_bQuitDcOn	11	1	HV activation acknowledgment
AMK_bDcOn	12	1	HV activation level
AMK_bQuitInverterOn	13	1	Controller enable acknowledgment
AMK_bInverterOn	14	1	Controller enable level
AMK_bDerating	15	1	Derating (torque limitation active)

Figure 3: Data bits within AMK_Status



Content of the **'AMK Actual Values 2'** data telegram:

Name	**Offset**	**Length in bits**	**Value type**	**Unit**	**Meaning**
AMK_TempMotor	0	16	Signed	0.1 °C	Motor temperature
AMK_TempInverter	16	16	Signed	0.1 °C	Cold plate temperature
AMK_ErrorInfo	32	16	Unsigned	-	Diagnostic number
AMK_TempIGBT	48	16	Signed	0.1 °C	IGBT temperature

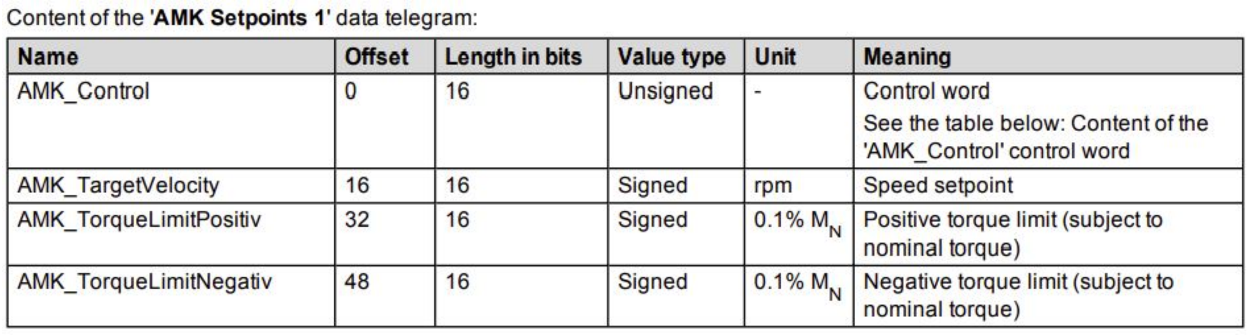
Figure 4: AMK_ActualValues2 signals

5.2 Controlled Variables

Variable	ptCAN_AMK_Setpoints1_Right
Description	CAN message sent to the right-rear motor. See Figures 5 & 6, an excerpt from the AMK motor documentation, for a breakdown of the signals within this message.
Units	N/A - see Figures 5 & 6

Variable	ptCAN_AMK_Setpoints1_Left
Description	CAN message sent to the left-rear motor. See Figures 5 & 6, an excerpt from the AMK motor documentation, for a breakdown of the signals within this message.
Units	N/A - see Figures 5 & 6

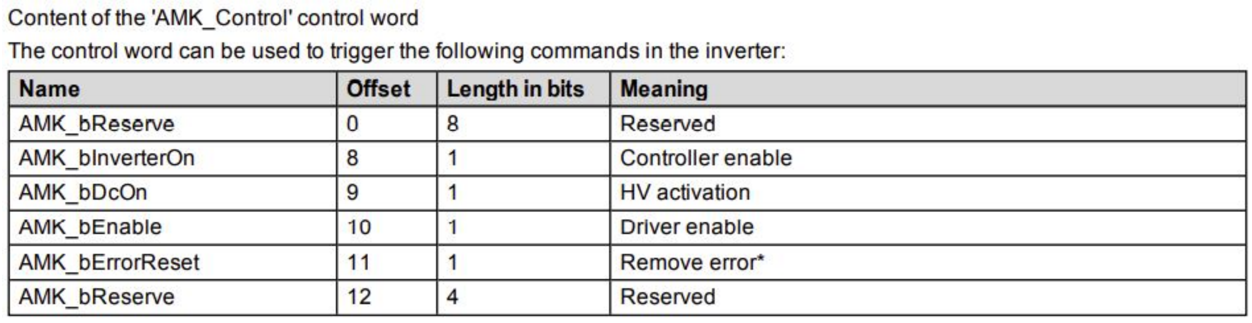
Variable	DI_b_driverSpeaker
Description	For controlling the driver speaker, sent to a digital output pin.
Units	Boolean



Content of the **'AMK Setpoints 1'** data telegram:

Name	Offset	Length in bits	Value type	Unit	Meaning
AMK_Control	0	16	Unsigned	-	Control word See the table below: Content of the 'AMK_Control' control word
AMK_TargetVelocity	16	16	Signed	rpm	Speed setpoint
AMK_TorqueLimitPositiv	32	16	Signed	0.1% M_N	Positive torque limit (subject to nominal torque)
AMK_TorqueLimitNegativ	48	16	Signed	0.1% M_N	Negative torque limit (subject to nominal torque)

Figure 5: AMK_Setpoints1 signals



Content of the 'AMK_Control' control word
The control word can be used to trigger the following commands in the inverter:

Name	Offset	Length in bits	Meaning
AMK_bReserve	0	8	Reserved
AMK_bInverterOn	8	1	Controller enable
AMK_bDcOn	9	1	HV activation
AMK_bEnable	10	1	Driver enable
AMK_bErrorReset	11	1	Remove error*
AMK_bReserve	12	4	Reserved

Figure 6: AMK_Control bits

5.3 Constants Variables

The current scope of the control system does not require the use of constants.

6 User Interfaces

Although we do not have a traditional software-based user interface, our control system still has connections and reliance upon the user, or in this case, the driver, through physical buttons located on the vehicle's steering wheel and inside the driver's cockpit.

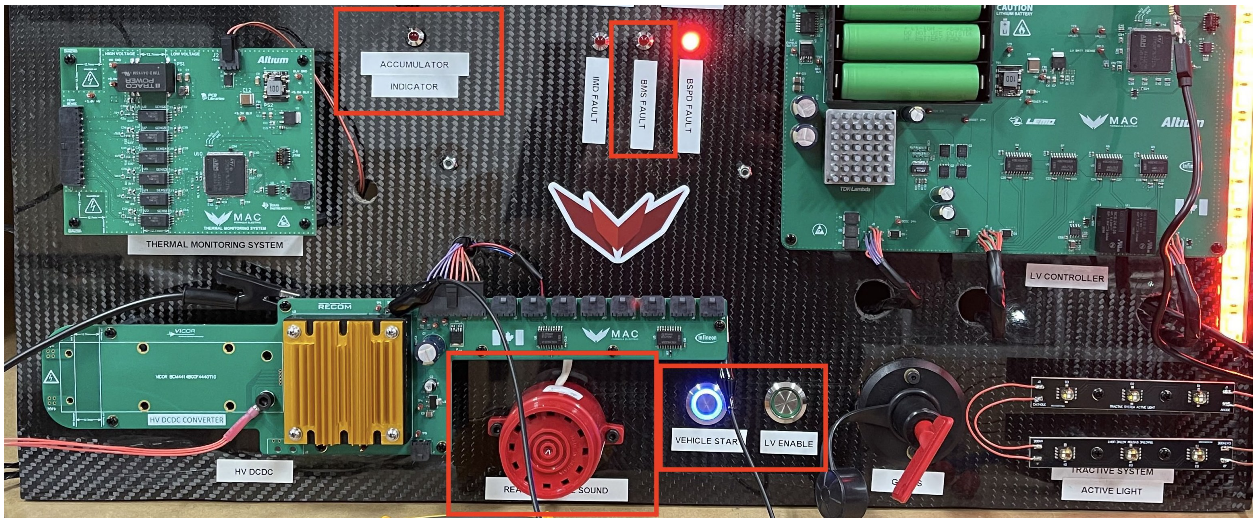


Figure 7: MAC FE LV Test Bench

Figure 3 was taken from the MAC FE LV test bench, on which the relevant hardware that will interface our control system with the user can be found. The first major component is the accumulator indicator, used to notify the user that the accumulator, or battery pack, is functional and ready for use. This signal comes from a vehicle controller and is based on the status of the high-voltage contactors provided by the Orion BMS. The second major component is the BMS fault LED, used to notify the user of any mismatched or out-of-order contactor sequencing faults caught by the BMS. We then have our physical push buttons labeled 'vehicle start' and 'LV enable'. Both switches act as blocks to prevent the vehicle from starting without user input, and are/will be accounted for in the control system accordingly. The last piece of hardware that could be considered a user interface is the ready-to-drive speaker, pictured at the bottom of figure 2 in red. This speaker is used to notify the driver and everyone in their surrounding that the vehicle is now indeed ready-to-drive and should be treated with caution. Other user interface elements not shown in Figure 3 include the vehicle's accelerator pedal, brake pedal, and steering wheel. These inputs to our control system are the primary interface with the user while the vehicle is in motion.

7 Design of Hardware

The design of physical, non-electrical, hardware is not relevant to this particular capstone project. This is in part due to the hardware, such as the steering, chassis, and all ECU and accumulator enclosures already having been designed and fabricated by members of MAC FE.

8 Design of Electrical Components

In terms of electrical components, we will be acquiring hardware from MAC FE as a vast majority of the hardware being controlled by our system is far outside our project budget and heavily specialized for the vehicle's application. Below is a summary of the hardware being acquired.

Hardware	Usage
Orion BMS	Off-the-shelf battery management system used to control contactor states, operational cell temperatures, and report data back to the control system via CAN.
AMK Motor Kit	Off-the-shelf dual motor kit, single source of propulsion for the vehicle, completely controlled by the control system.
Vehicle Start Push Button	Button to signal to the control system that the user is ready to initiate the startup sequence.
LV Enable Push Button	Button to enable the LV hardware on the vehicle, including the Front Controller ECU which hosts the control system.
Ready-to-Drive Speaker	Off-the-shelf speaker used to notify the user and surrounding personnel that all critical vehicle systems are functional.

9 Design of Communication Protocols

The design of the relevant communication protocols inside the control system are completely reliant upon the MAC FE embedded system and the AMK motor kit being able to receive, decode, and utilize the data sent over. Inside the system, we have defined multiple structures to consolidate and tag data appropriately and subsequently send it over to the embedded system for consumption. All the structures were defined as enumerations that contain model-specific state information, commands, and error codes. These enumerations are then auto-generated into C structures using the built-in autogeneration toolbox in Simulink. With these C structures, the MAC FE embedded system, which is solely written in C, can easily use the data being computed inside the control system in order to make sub-system-specific decisions.

Apart from the enumerations, we also utilize CAN communication to interface with the AMK motor kit. The MAC FE embedded system is still acting as a middleman in order to transmit the CAN frames, receive them, and then push them back into the control system, but our systems Motor Interface module prepares the CAN frame for the embedded system such that the embedded system does not need to decode or refactor the frame, but can simply transmit the output from the control system directly to the AMK motor controllers. These frames are using the CAN 2.0b standard and the data within each frame is formatted such that the motor controllers are able to decode them appropriately, shown as an example in Figure 4.

10 Timeline

Current Tasks which have been planned for implementation on hardware are as follows:

- Simscape / vehicle dynamics modelling for plant simulation (Laura, Jason): Due January 22
- CAN interfacing (Abhishek): Due January 27
- Potentiometer interfacing (Derek, Abhishek): Due February 3
- Simulation test case design (Laura, Jason): Due February 3

A Reflection

Given greater resources in the form of time, know-how and budget, we would be able vastly improve the testing loop in our control system's development. This could include purchasing industry-level vehicle modelling/simulation tools that would allow us to simulate transient signals such as component temperature, vehicle dynamics (speed, acceleration), and motor and battery characteristics. Such tools are much more accurate for simulation and logic testing when compared to our manually-created plant models, which are based on rudimentary physics and hard-coded timeseries signals. We would also be able to build an automated test infrastructure that runs through a series of predefined simulation scenarios and verifies that a change has not broken the logic of a required feature. In an even more ideal scenario, we would have access to the hardware-complete vehicle to perform in-vehicle testing and data acquisition, to tweak and calibrate driver input handling and motor commands as to improve driving characteristics.

When we first approached the task of creating a control system for the McMaster FSAE car, the initial thought was to develop and launch it in C code. Through our preliminary information gathering and consideration of the system's complexity, we quickly realized that this would be a major hindrance to implementation within the given time frame. We quickly moved to the framework of Simulink development, which is almost ubiquitous in automotive software development or prototyping. The rapid development workflow (design, implement, simulate, fix), auto-generated code and data visualization tools would allow for greater progress to be made by the end of the project. Beyond this decision, from the start we knew the system would require a series of layered modules, each controlling various subsystems of the car, and communicating their states and signals to each other as needed. This software architecture was primarily driven by industry experience among team members, and moreover this reflects and integrates well with the planned hardware systems/layout that the vehicle will be built with.