Electric Vehicle Strategies

Implementing electric vehicle strategies

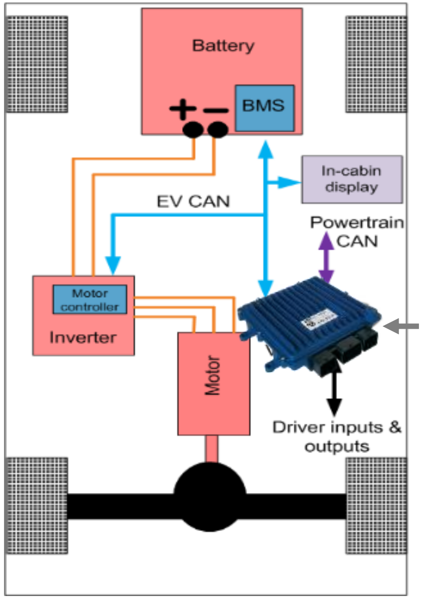
Outline

- Overview
- Software architecture design
- Functional requirements
- Baseline Strategy
- EV Controls
- Miscellaneous information
- Example Applications

Pi Innovo

Overview (1/2)

- Pi Innovo has developed control systems for electric vehicle architectures
- Simulink® strategies for these applications provide an efficient starting point for other electric vehicle projects
- The strategies form a starting point for prototyping as well as production activities



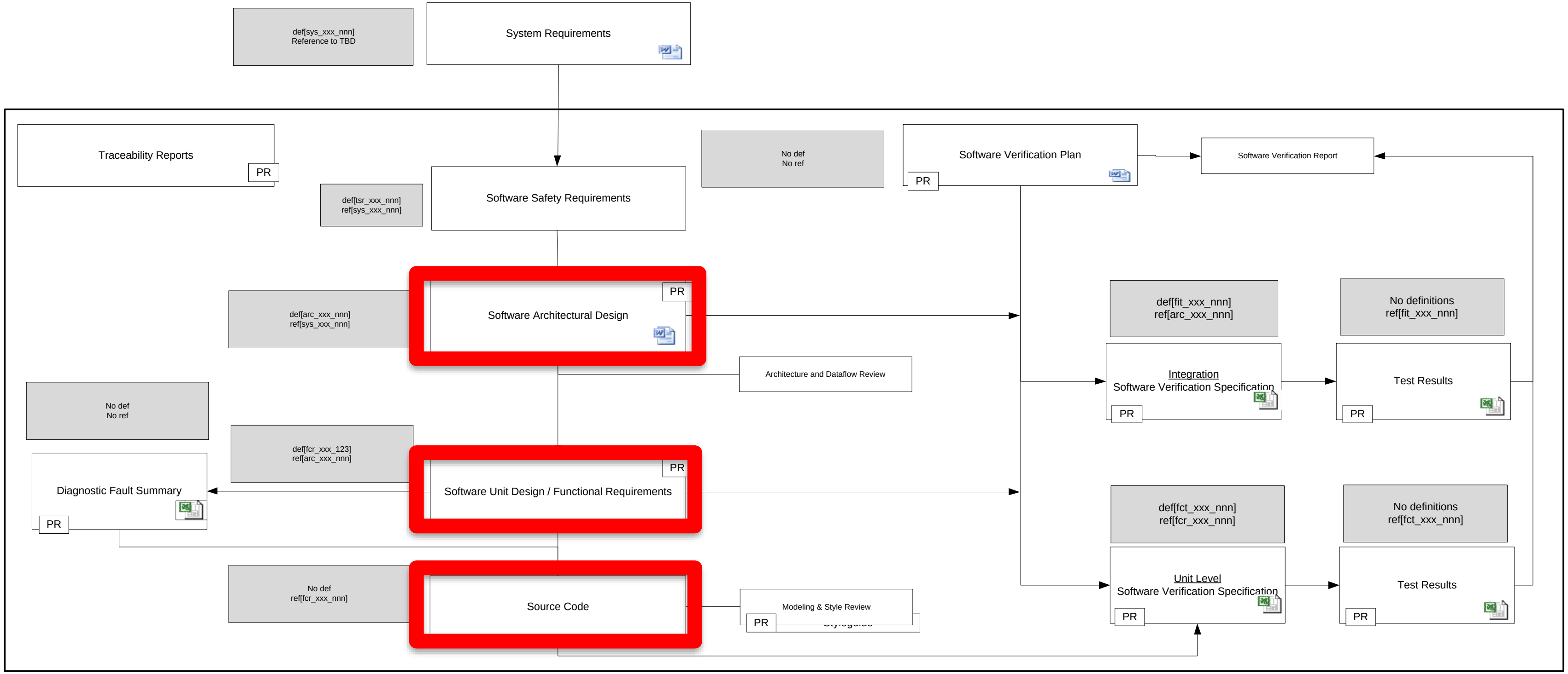
M560 as
a. VCU
b. CCS

Electric Vehicle Architecture

Pi Innovo

Overview (2/2)

Document Map



Note: Items appended to PR tag are considerations made during the peer review process.
Note: Items appended to a work product are considerations made in the authoring of the work product.

PR Peer Review

Software architecture design

- Top level architecture is contained in two documents
  - Control Architecture Document (~45 pages)
  - Functional Block Diagram
  - Architecture documentation describes the high level functionality, requirements and interfaces

7.8 MCS: Motor Control Strategy

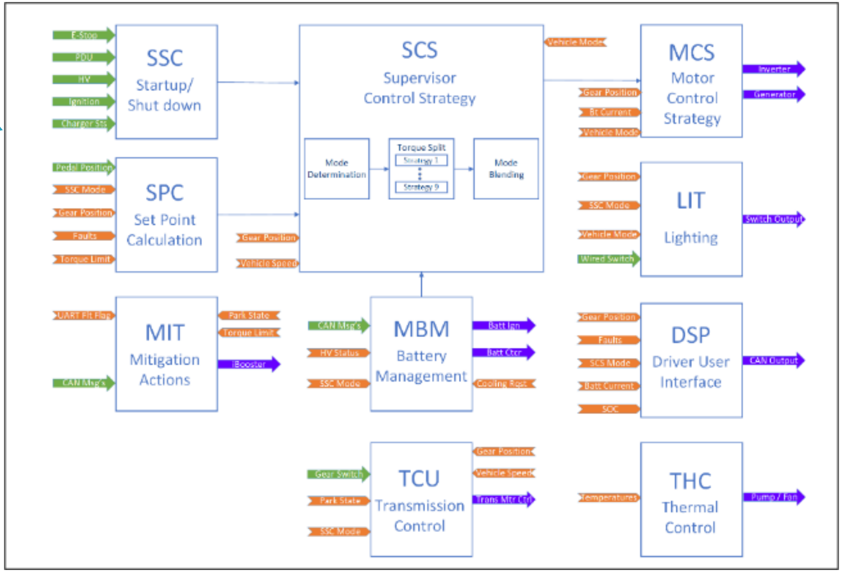
The MCS feature requests the SCS desired traction and generator motor torque after applying the necessary sign modifications from the MCM and the GCM respectively.

The torque sign is needed to be computed to account for the various vehicle states. For example, if it is intended that the vehicle move in reverse, then although SCS requests a positive torque (as this is traction torque to the wheels), the MCS is responsible to switch the sign of the torque request.

The MCS shall also set the Control mode of the MCM and GCM along with computing other CAN commands, which includes the Counter Torque Limit, Speed limit and checksum to interface with the Traction and Generator Motor Controller.

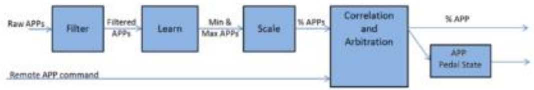
7.8.1 Requirements

Def Tag	Ref Tag	Requirement
def[arc_mcs_001]	ref[reqsr_chg_001] ref[reqsr_chg_002]	The MCS shall set the MCM and GCM's Control Mode depending on the Motor state and the Active Flags
def[arc_mcs_002]	def[reqsr_xxx_xxx]	The MCS shall apply a sign modification to the • SCS desired motor and generator torque request

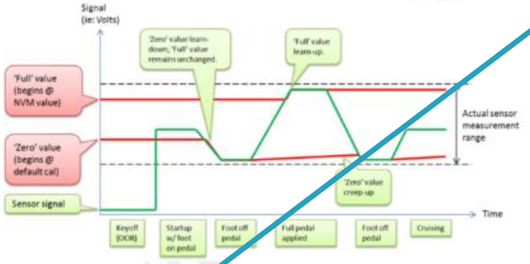


Pi Innovo

Functional Requirements



An example use-case for APP signal range learning is shown below. This shows normal, intended operation for the first-ever startup. No components are failed, and the pedal's physical end-stops do not change or shift during this use case.



Requirements Tags

5. Functional Requirements

5.1 General Requirements

Def Tag	Ref Tag	Requirement
def[fcr_ipr_014]	ref[sad_ipr_003]	Before processing of an analog raw input value, the IPR subsystem shall apply a LPF, with a calibratable cut off frequency to the raw values of analog inputs.
def[fcr_ipr_007]	ref[sad_ipr_004]	The IPR subsystem shall compute the Rate of change of the of raw voltage value for the following signals:

5.4.3 Conversion to pedal position

Def Tag	Ref Tag	Requirement
def[fcr_ipr_722]	ref[sad_ipr_001]	The feature shall contain a calibration for each pedal position sensor to indicate if the transfer function increases or decreases with pedal position.
def[fcr_ipr_723]	ref[sad_ipr_004]	A processed pedal position for an increasing transfer function shall be computed as $APP = \frac{APP_{meas} - APP_{zero}}{APP_{full} - APP_{zero}}$ APP Interpreted accelerator pedal position (pct) APP_{meas} Accelerator pedal position sensor measured signal APP_{zero} Zero accelerator pedal position sensor voltage value. APP_{full} Full accelerator pedal position sensor voltage value
def[fcr_ipr_724]	ref[sad_ipr_006]	A processed pedal position for an decreasing transfer function shall be computed as $APP = \frac{APP_{meas} - APP_{full}}{APP_{zero} - APP_{full}}$

5.4.4 Pedal Arbitration

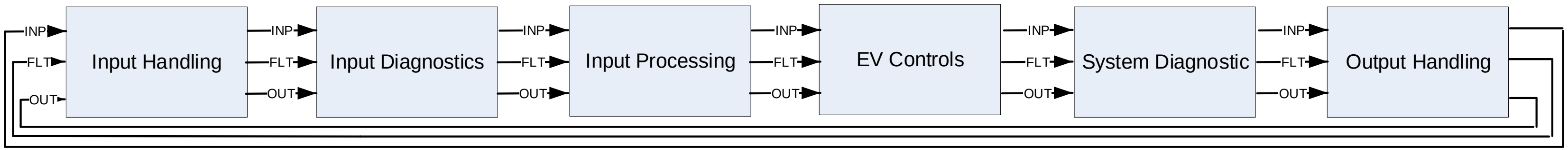
The feature assumes there will be two sensors at the accelerator pedal, and therefore the requirements given above will be executed on the signals from both sensors, independently. The requirements in this section are use the results from the two sensors to create a single arbitrated Accelerator Pedal value.

Def Tag	Ref Tag	Requirement
def[fcr_ipr_727]	ref[sad_ipr_009]	If there is no correlation fault and both the pedal positions are valid, the arbitrated pedal position shall be calculated from a weighted average of the two pedal signal positions. The weight shall be calibratable.

16 Software Components: Varying size and complexity 8-15 pages each

Pi Innovo

Baseline Strategy (1/4)



Subsystem	Name	Description
INP	Input Handling	Read/receive hardware channels and CAN messages
IDG	Input Diagnostics	Out of range diagnostics on raw inputs
IPR	Input Processing	Conversion of raw inputs to engineering units
EV Controls	Calculations	Electric Vehicle control features and calculated values
SDG	System Diagnostics	Output and rationality diagnostics
OUT	Output Handling	Set/send hardware channels and CAN message

Pi Innovo

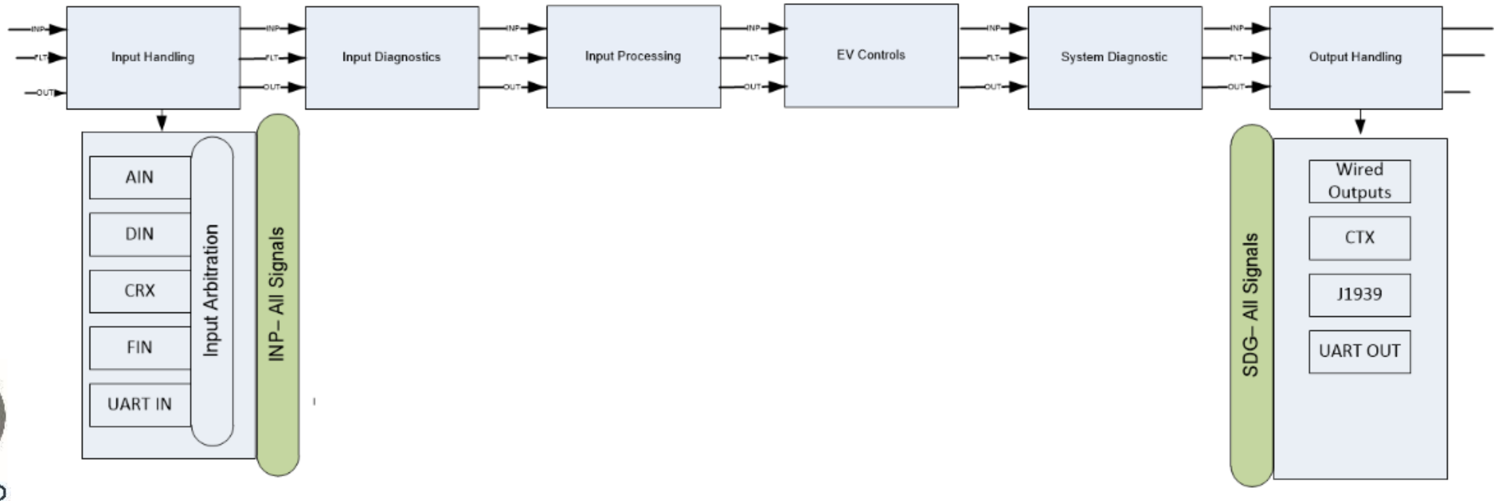
Baseline Strategy (2/4)

INP Input Handling

- This feature is responsible for reading all input signals
- Inputs include:
  - Analog signals from external sensors and OpenECU internal monitors
  - Digital signals from external sensors and OpenECU internal monitors
  - CAN input messages from all CAN buses

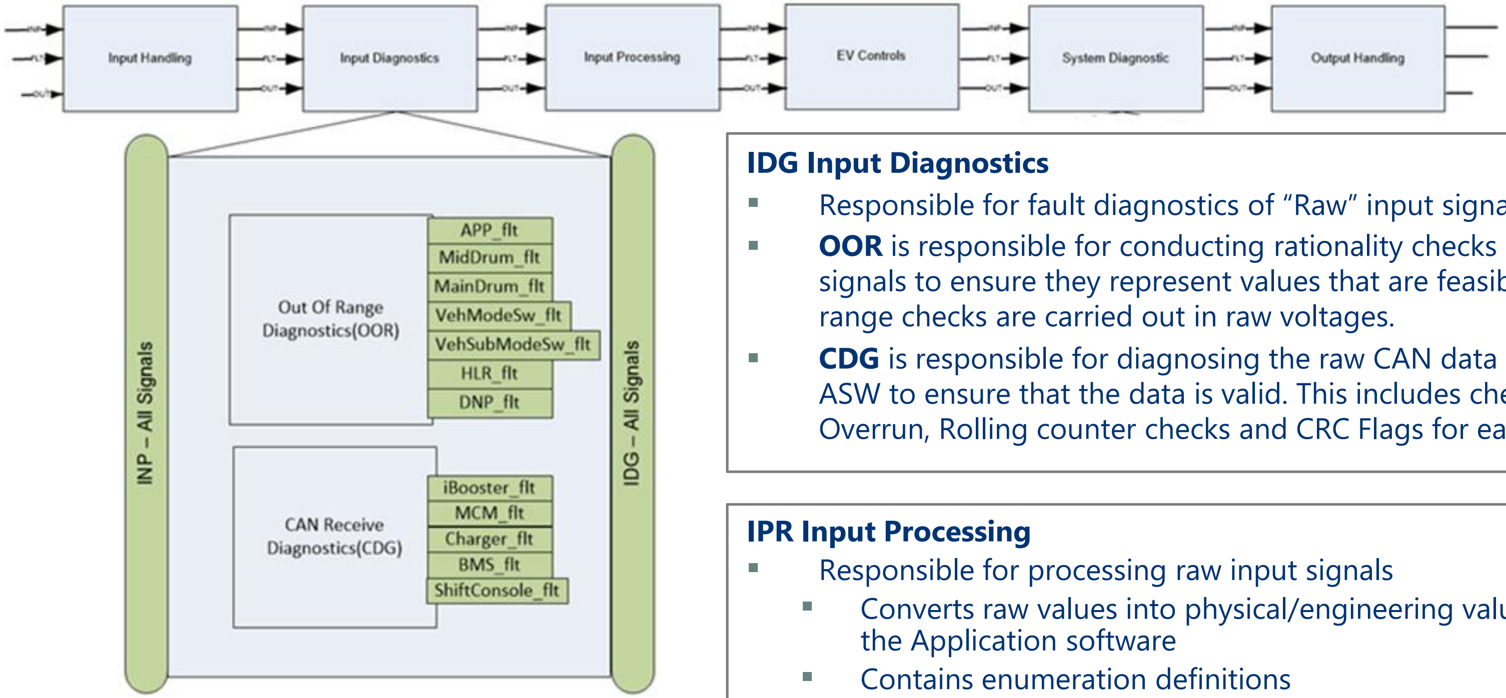
OUT Output Handling

- Responsible for receiving commands from other subsystems via the output bus
  - Set output blocks of wired components to the correct value
  - Transmit CAN messages to ECUs on CAN network
  - Transmit UART messages to the secondary micro
  - Set internal signals of the VCU



Pi Innovo

Baseline Strategy (3/4)



IDG Input Diagnostics

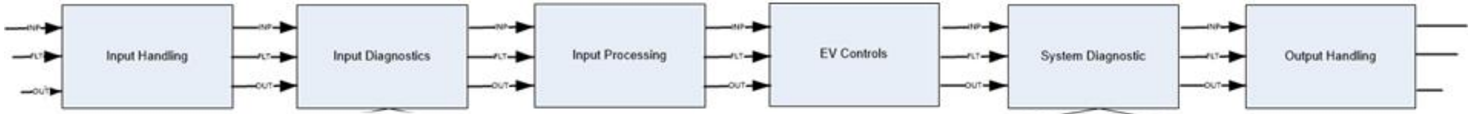
- Responsible for fault diagnostics of “Raw” input signals
- **OOR** is responsible for conducting rationality checks on all input signals to ensure they represent values that are feasible. These range checks are carried out in raw voltages.
- **CDG** is responsible for diagnosing the raw CAN data as read by the ASW to ensure that the data is valid. This includes checks like Overrun, Rolling counter checks and CRC Flags for each message

IPR Input Processing

- Responsible for processing raw input signals
  - Converts raw values into physical/engineering values for use by the Application software
  - Contains enumeration definitions
  - Contains an adaptive algorithm for APP interpretation
  - Contains conversion from one physical unit to another
  - Contains peak detection algorithms for signals like the H-Bridge current

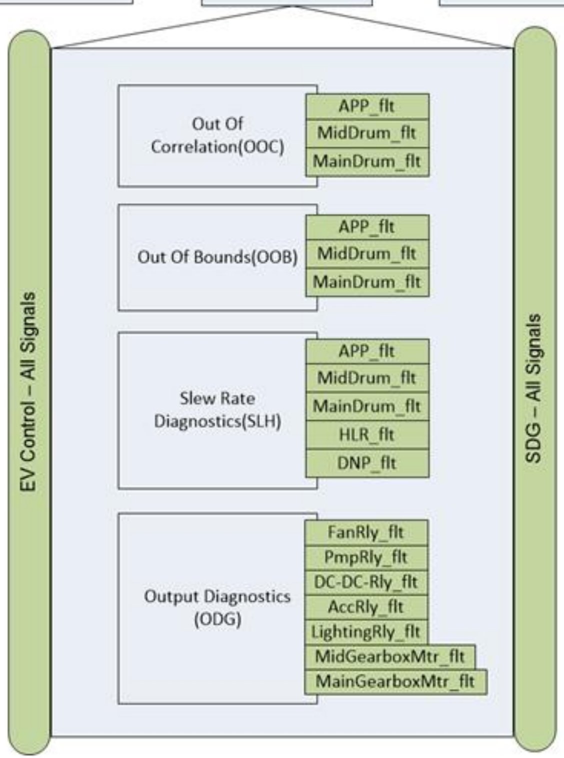
Pi Innovo

Baseline Strategy (4/4)



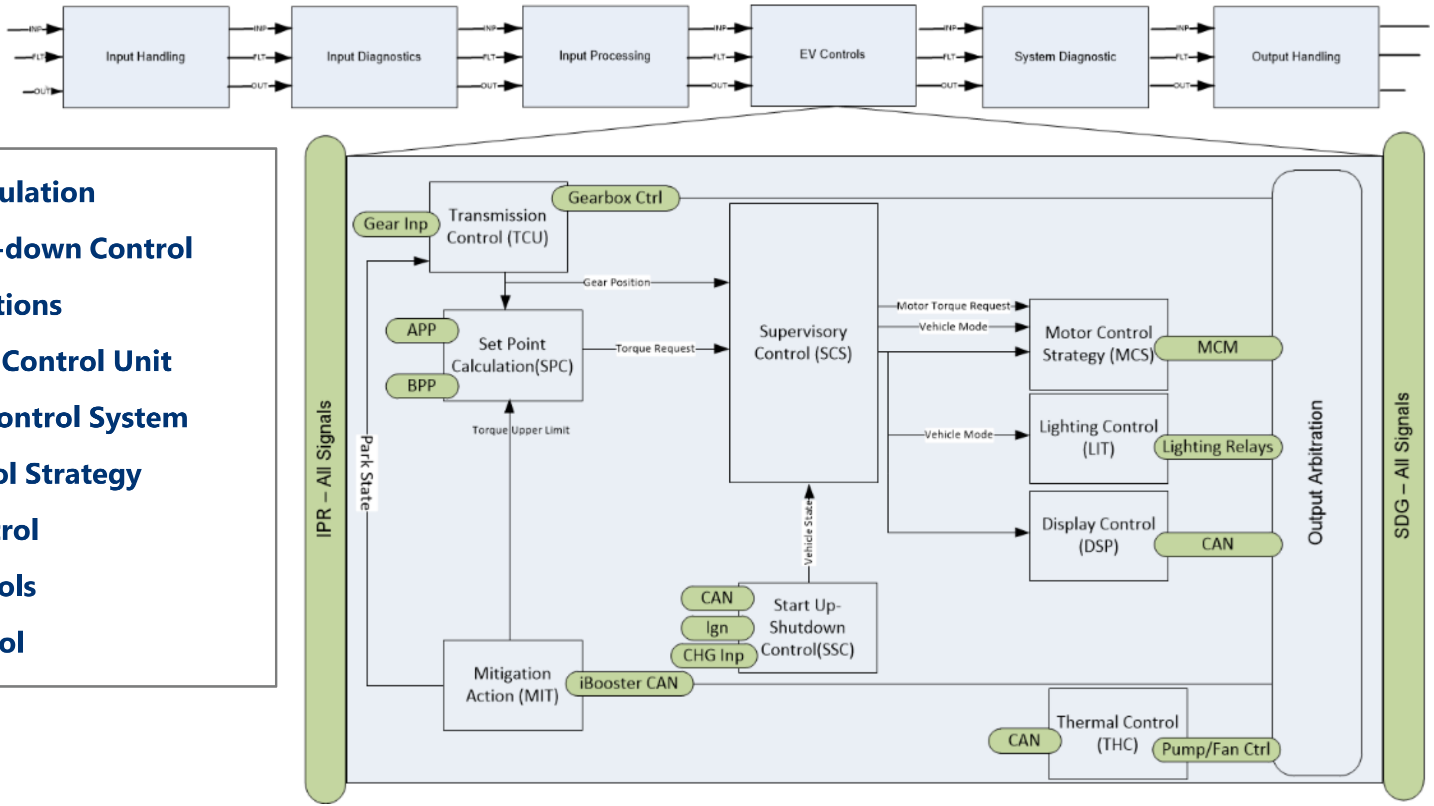
SDG System Diagnostics

- **OOC:** check for any mismatches between the same measurement values that is acquired from two independent sources
- **OOB:** To ensure feasibility of the physical values (engineering units) values of input signals/variables
- **SLH:** To diagnose abnormal high slew-rates for analog signals
- **ODG:** To diagnose faults on output pins of the ECU



Pi Innovo

EV Controls



SPC: Set Point Calculation

SSC: Start-up Shut-down Control

MIT: Mitigating actions

TCU: Transmission Control Unit

SCS: Supervisory Control System

MCS: Motor Control Strategy

THC: Thermal Control

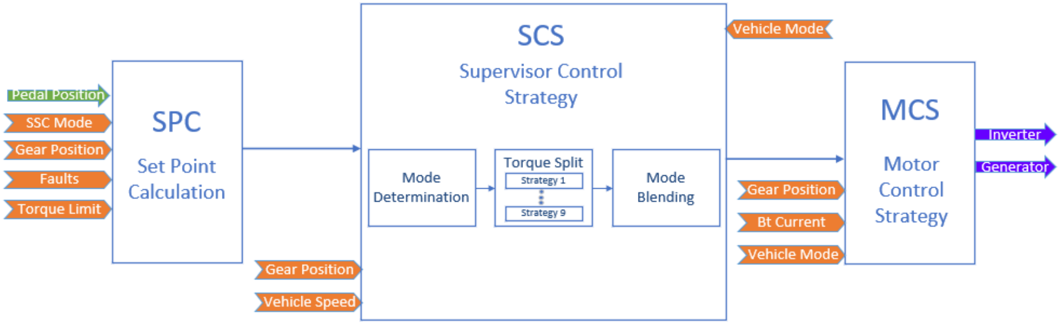
LIT: Lighting Controls

DSP: Display Control

Pi Innovo

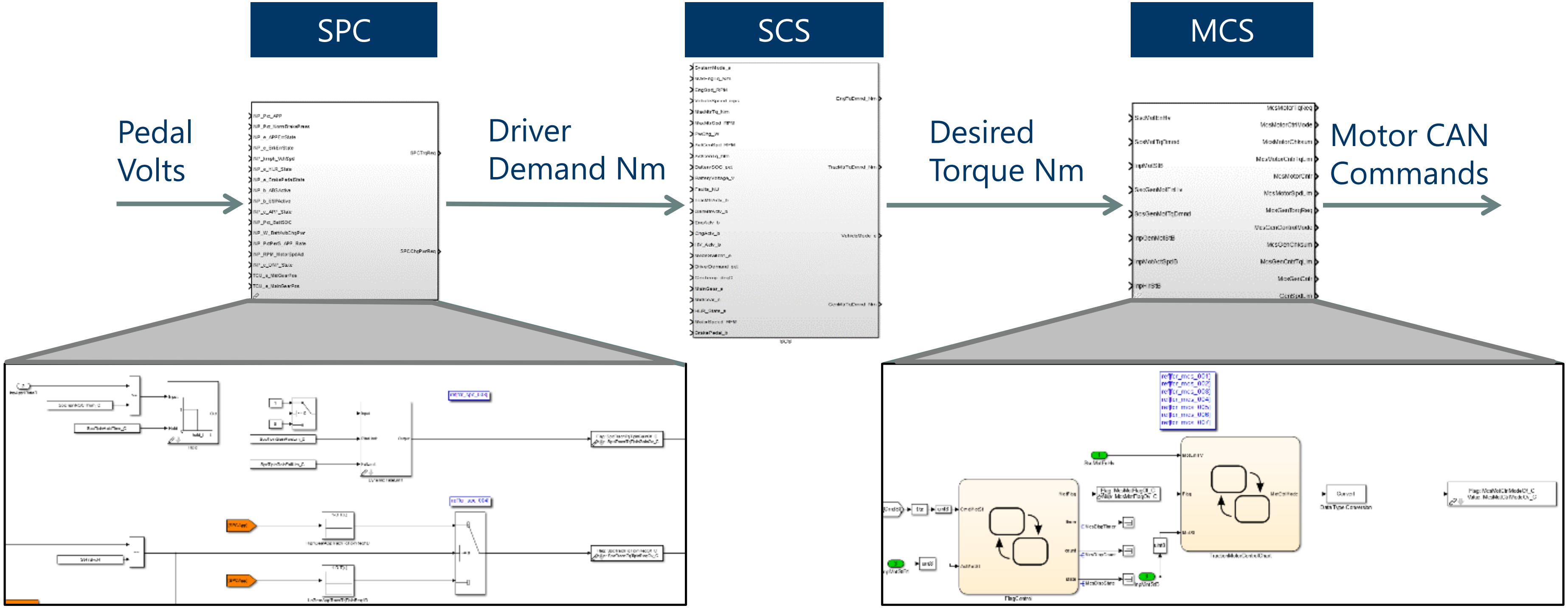
EV Controls – Sample use-case: Logic

- SPC = Reads pedal analog inputs, considers gear position, brake pedal, faults and creates a final driver demanded torque request
- SCS = Torque arbitration and Vehicle Mode determination. Implements multiple torque strategies to Creates a final torque request based on multiple sources
- MCS= Units that create final torque request to the Traction Motor(s)



Pi Innovo

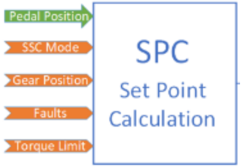
EV Controls - Sample use-case: Simulink



Pi Innovo

EV Controls - Features

SPC Set Point Calculation



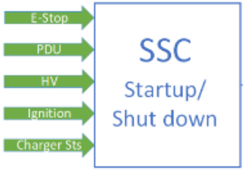
- SPC uses the accelerator pedal position signal and converts it into a torque percentage request
  - Via individual look up tables depending on switch states
- Collects torque requests from multiple sources, combining them to create a single overall driver demand

MIT Mitigation Actions

- Used to address functional safety requirements of the vehicle in the longitudinal direction
- If vehicle operation mode is unsafe set to degraded state by limiting upper Torque % limit and command Brake controller
- This feature can be tied in with Torque security

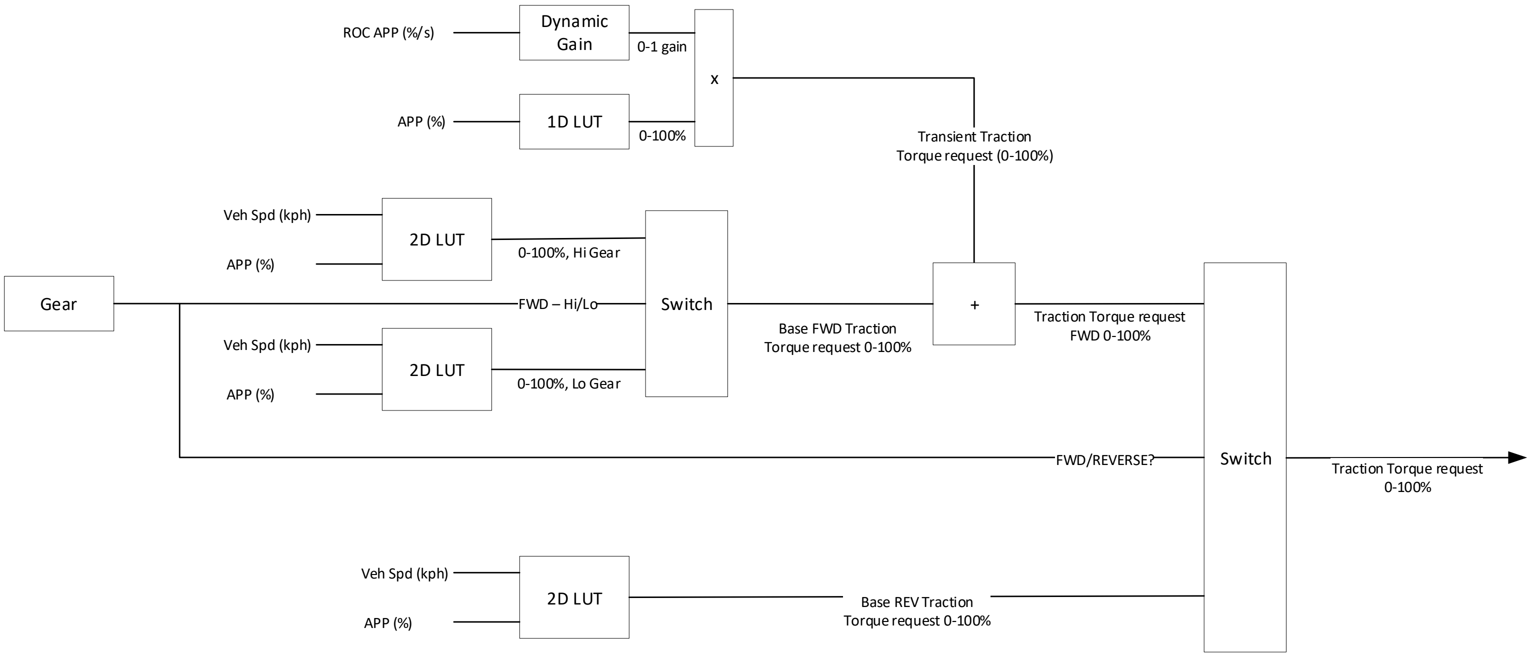
SSC Startup Shutdown Sequencing

- Consists logic acting as software enabler for MCU relays
- It provides startup and shut down routines for the EV
  - It includes hard shutdown (emergency stop push) or a soft shutdown
- SSC broadcasts startup/shutdown state of the vehicle
- Also handles the power supply hold feature



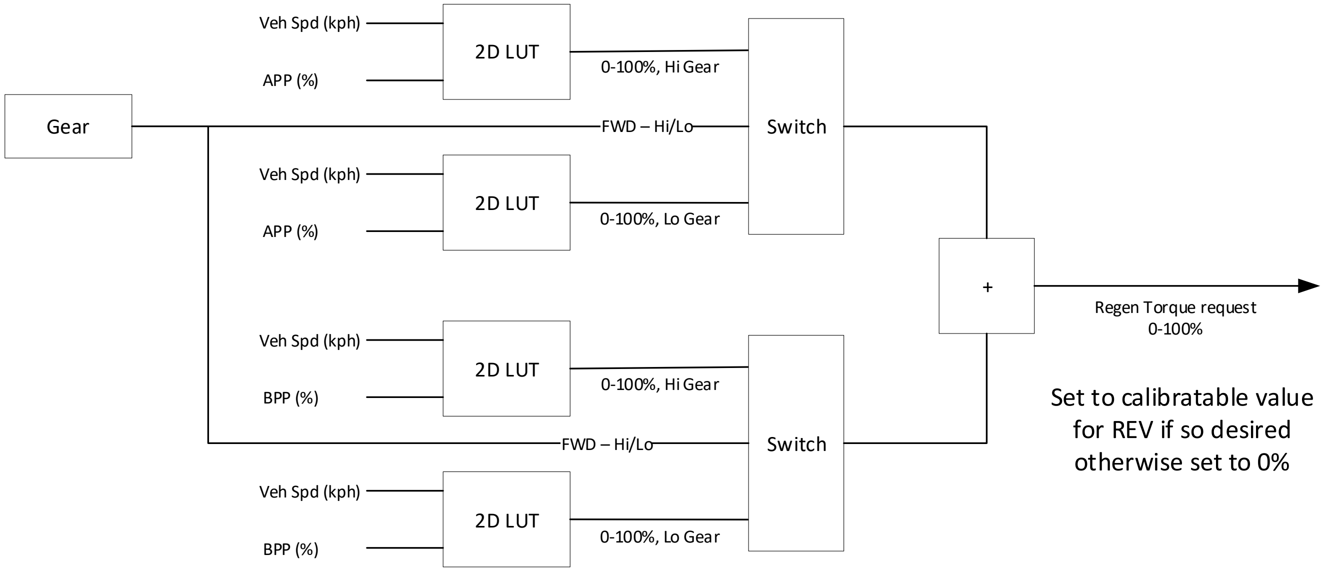
Pi Innovo

EV Controls – SPC: Traction Torque request



Pi Innovo

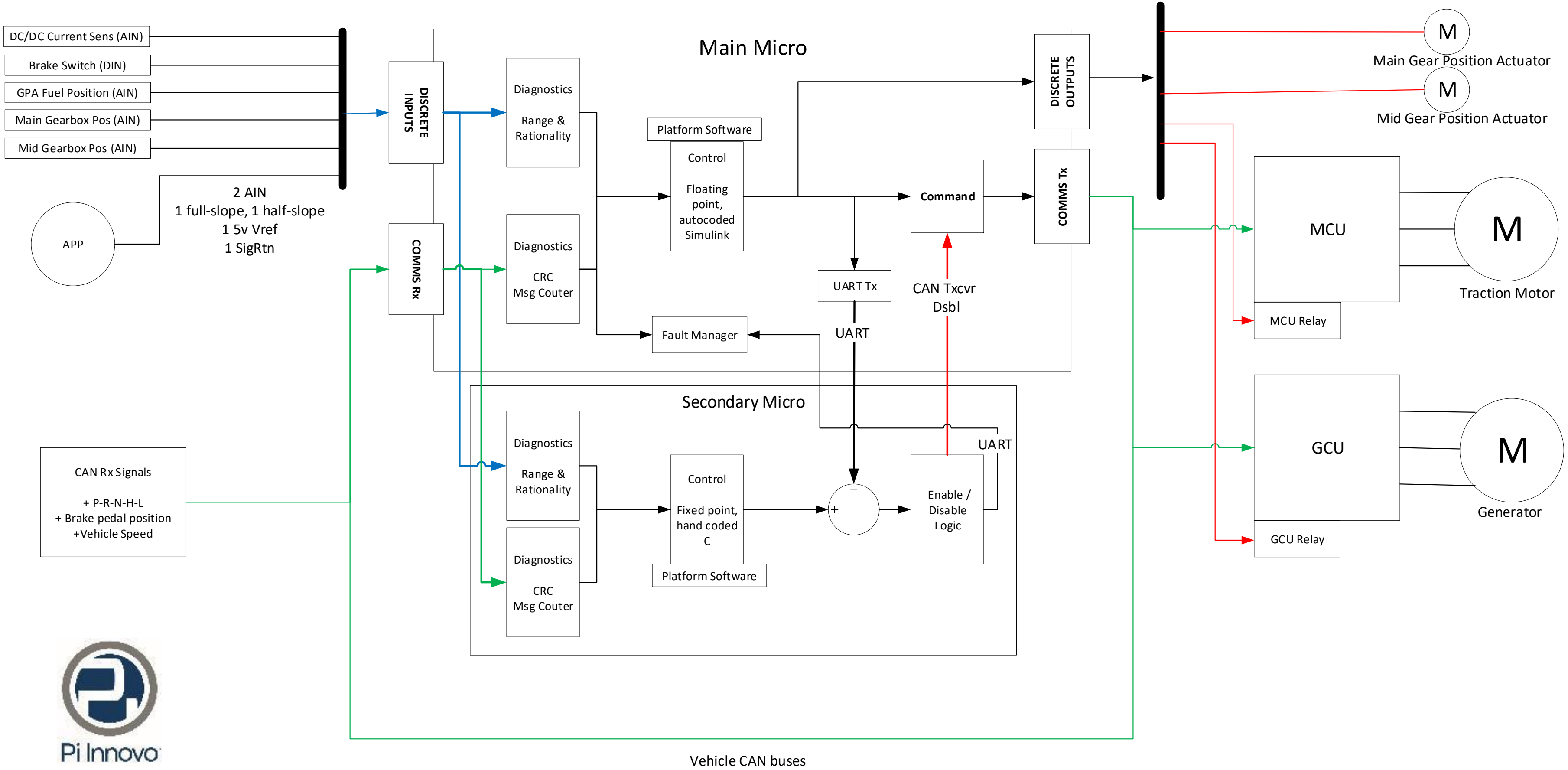
EV Controls – SPC: Regen Torque request



Set to calibratable value for REV if so desired otherwise set to 0%

Pi Innovo

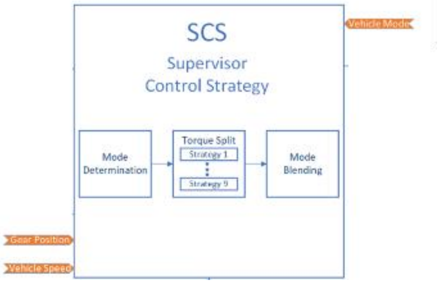
EV Controls – MIT: Torque Security (sample use-case)



EV Controls – Features

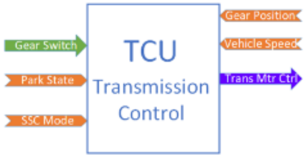
SCS Supervisory Control Strategy

- It is responsible for computing the torque demand from the traction motor
- SCS takes into account the driver demand, vehicle conditions and faults on existing components to compute the vehicle mode the EV should be in
- When there is transition from one vehicle mode to the other then motor torque blending takes place to improve drivability
- The SCS is divided into four subsystems namely
  - Driver Final Demand
  - Mode Determination
  - Torque Calculations
  - Mode Blending



TCU Transmission Control Strategy

- Responsible for computing electrical commands to engage the driver requested vehicle gear state
- Also indicates to the rest of the features in the Application Software the actual gear state of the vehicle
- The vehicle has Mid and Main gear box
  - 3 gear positions for Mid gearbox (High, Low and Neutral)
  - 5 gear position for Main gearbox (High, Low, Neutral, Park and Reverse)

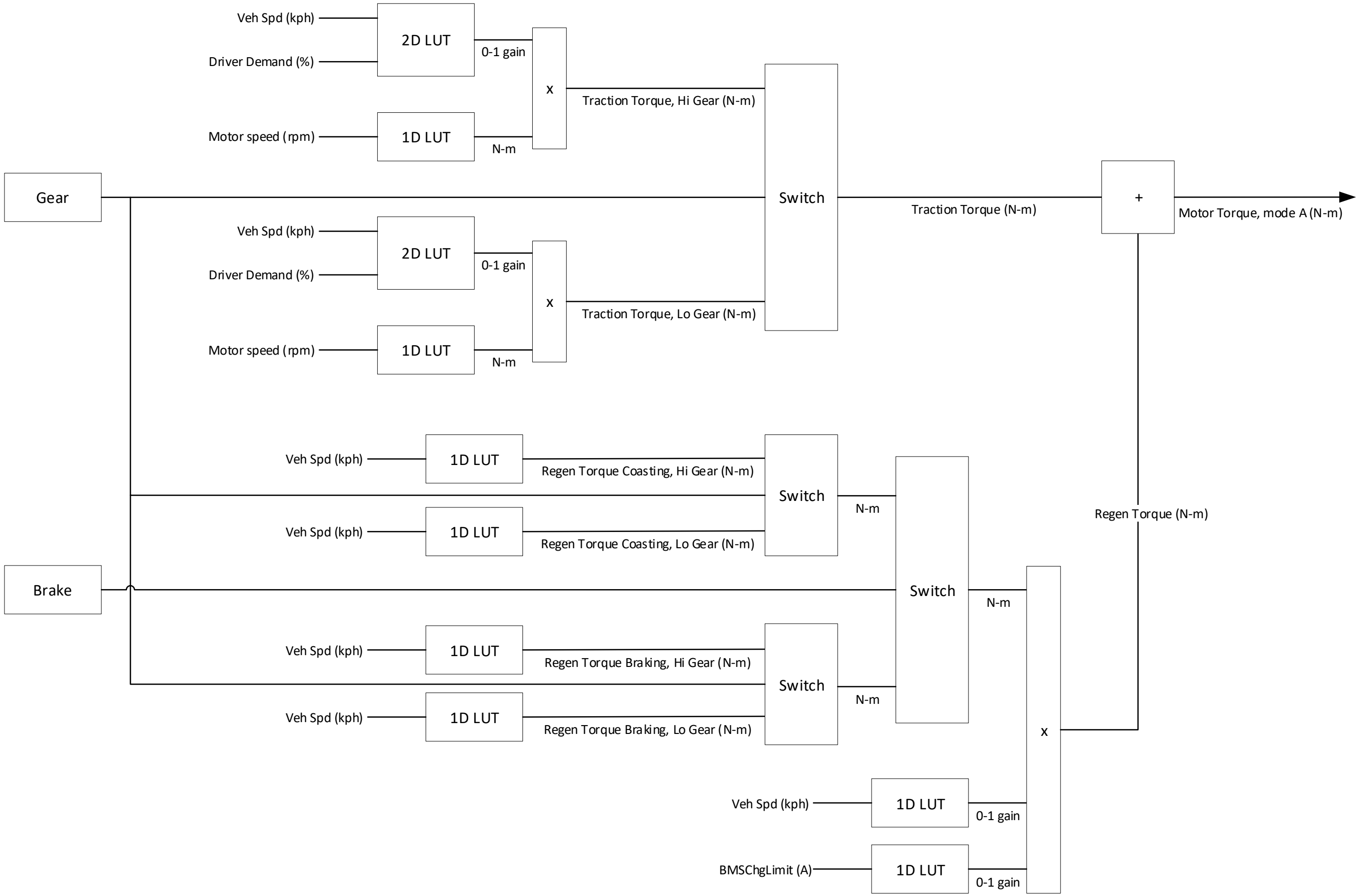


MCS Motor Control Strategy

- MCS is comprised of a control strategy that commands torque to the electric motor
- Motor 1 (Mot1) also known as the Traction Motor
- This feature process all the values that are required to drive the motor

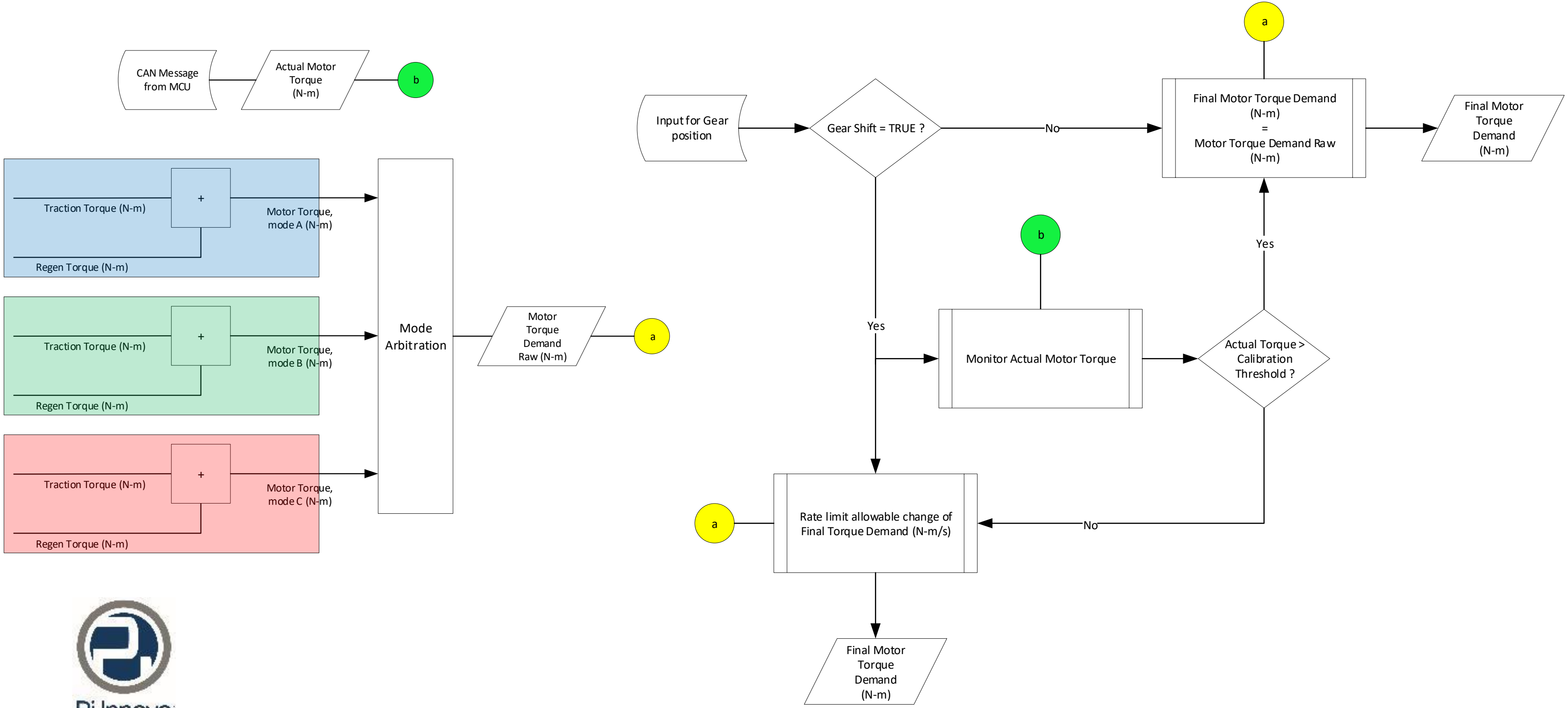
Pi Innovo

EV Controls – SCS: Torque Demand (per mode)



Pi Innovo

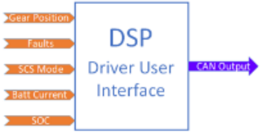
EV Controls – SCS: Torque blending



EV Controls – Features

THC Thermal Control Strategy

- Responsible for thermal management of electric components in the powertrain
  - Traction motor and traction motor inverter
- It receives temperatures of each component and decides if the cooling system needs to activated or not
  - Controls coolant pump and cooling fan with 2 independent relays (Note: It does not control the speeds of coolant pump or fan)



DSP Display Strategy

- This feature interfaces via CAN with the display module
- It is responsible to communicate necessary information required as per the HMI section of the system

LIT Lighting Control Strategy

- Responsible for handling logic pertaining to actuating lights on the EV
- It is anticipated that the logic is a simple “if-else”
  - For e.g. if it is detected that the brakes are depressed then the brake lights are on *OR* if the vehicle is in reverse gear then the reverse beeper is actuated.

Pi Innovo

Miscellaneous information

Variable naming convention

Prefix — Descriptive Section

Din IgnKeyDbnc _C

Variable Type

Named item type	Type letter
Displayable signals	(no type letter)
Calibration scalars	C
Calibration maps	M
Constants	SC
Arrays	CA

```
### Build information:
..Strategy memory:
   602592 bytes of strategy/code memory used
  1756704 bytes remaining
..Calibration memory:
    70836 bytes of calibration memory used (rough indication, includes Simulink support data)
    61259 bytes remaining
..Workspace memory:
   264902 bytes of workspace/displayable memory used, including
      188 bytes of adaptive data, and
     4278 bytes of diagnostic trouble code data, and
     8192 bytes of model stack
   259385 bytes remaining
```

CPU utilization < 50% on M560 (SPC5764)

Pi Innovo

Pi Innovo is the expert partner for the design and development of innovative electronics systems to the automotive, transportation, defense, industrial, and aviation industries. Our uniquely adaptable business engagements, based on Pi Team services and OpenECU products, enjoy a strong reputation for delivering the highest quality results , providing outstanding value for our clients.

OpenECU is a wide range of adaptable, field-ready products and intellectual property designed to accelerate electronics system development. The philosophy behind OpenECU is the creation of modular, reusable technology that is implemented to volume production standards and is fully "open" to custom configuration, adaption and further development.

ACCURATE TECHNOLOGIES
sales@accuratetechnologies.com
www.accuratetechnologies.com