**Software Architecture Design**

**Stellantis**

**12V 2kW Inverter**

**<Control Number>**

**Revision:** **1.0.2**

**Release Date: 27Apr21**

**Status: *Draft***

**Prepared By:**

**Alina Barnett**

Table of Contents

[1 Overview 5](#_Toc128059991)

[1.1 Purpose 5](#_Toc128059992)

[1.2 References 6](#_Toc128059993)

[1.3 Definitions 7](#_Toc128059994)

[2 Architecture Overview 8](#_Toc128059995)

[2.1 Architecture Evaluation and Selection 8](#_Toc128059996)

[2.2 Software Design Decisions 8](#_Toc128059997)

[2.3 System Overview 8](#_Toc128059998)

[2.3.1 Context Diagram 8](#_Toc128059999)

[2.4 Software Architecture Component view 9](#_Toc128060000)

[3 Software Architecture Design 10](#_Toc128060001)

[3.1 Dynamic Architecture 10](#_Toc128060002)

[3.1.1 Modes of operation 10](#_Toc128060003)

[3.2 Static Architecture 20](#_Toc128060004)

[3.2.1 Component Interface Matrix 20](#_Toc128060005)

[3.2.2 State Machine Component 26](#_Toc128060006)

[3.2.3 RTOS Component 27](#_Toc128060007)

[3.2.4 Fault Manager Component 28](#_Toc128060008)

[3.2.5 Board to Board Communication (B2b Comm) Component 29](#_Toc128060009)

[3.2.6 Monitor Component 30](#_Toc128060010)

[3.2.7 DCDC Component 32](#_Toc128060011)

[3.2.8 Cooling Manager Component 33](#_Toc128060012)

[3.2.9 LIN Component 34](#_Toc128060013)

[3.2.10 Signals Component 35](#_Toc128060014)

[3.2.11 Fan Component 37](#_Toc128060015)

[3.2.12 EEPROM Component 38](#_Toc128060016)

[4 Platform Architecture 40](#_Toc128060017)

[4.1 Hardware and Software interaction 40](#_Toc128060018)

[4.2 Microcontroller pin functionality allocation 40](#_Toc128060019)

[4.2.1 Peripherals 40](#_Toc128060020)

[4.2.2 Clocks selections 40](#_Toc128060021)

[4.2.3 Recommended I/O Pin Function Allocation 40](#_Toc128060022)

[4.2.4 Interrupts 40](#_Toc128060023)

[5 Resource usage 41](#_Toc128060024)

[5.1 Task Management 41](#_Toc128060025)

[5.2 Memory Management 41](#_Toc128060026)

[6 General Information of Development 42](#_Toc128060027)

[6.1 IDE details 42](#_Toc128060028)

[6.2 Compiler details 42](#_Toc128060029)

[6.3 Project setup 42](#_Toc128060030)

[6.4 Application configuration details 42](#_Toc128060031)

Figure Index

**No table of figures entries found.**

**Table Index**

[Table 1 - Project Reference Table 6](#_Toc77262847)

[Table 2 - Definitions and Abbreviations 7](#_Toc77262848)

[Table 3 - Document Change History 8](#_Toc77262849)

[Table 4 - Template Change History 8](#_Toc77262850)

# Overview

## Purpose

<This document is the …>

## References

|  |  |  |
| --- | --- | --- |
| **#** | **Reference Document** | **Version number** |
|  |  |  |

Table 1 - Project Reference Table

## Definitions

| **Abbreviation or Acronym** | **Description** |
| --- | --- |
|  |  |

Table 2 - Definitions and Abbreviations

# Architecture Overview

## Architecture Evaluation and Selection

<This section describes the architecture evaluation and selection criteria>

## Software Design Decisions

<This section describes the design decisions. Ex: Libraries details, production/Testing support decisions, customizations, bootloader and application partitioning etc.,>

The primary software has three important elements: the core primary software project, the customer-specific LIN libraries, and the customer-specific header file which provides configuration details used to tune the hardware capabilities to the customer requirements. As the hardware has several variations differentiated from each other by differences in parts population, the correct software configuration is cross-checked between the hardware configuration header file and the reading of the hardware configuration ID resistor value. The core primary software project interfaces with the customer-specific LIN library with a defined set of Interfaces. The LIN library holds the logic for processing information from the primary project in the specific way the customer wants to receive it.

The primary software design consists of a 1ms tick RTOS scheduler that supports autonomous running including startup and shutdown sequences as well as fault handling. Time critical tasks are handled with interrupts. The architecture is layered to separate application specific modules, driver modules, and the hardware abstraction layer. Microchip Code Configurator tool was used to generate the files for peripherals containing the API for the hardware abstraction layer. The system is set up to monitor pertinent diagnostics throughout system operation.

The system was designed such that only one fault is registered at a time. This is because in the aftermath of certain events (like a DCDC shutdown) a series of faults could be triggered even though they are the result of a normal event. The decision was made to have one fault at a time that must be cleared in order to commence normal operation.

## System Overview

### Context Diagram

<Update the context diagram and high-level description. A context diagramis drawn to define and clarify the boundaries of the software system. It identifies the flows of information between the system and external entities. The entire software system is shown as a single process.>

Ex:

## Software Architecture Component view

### Component Diagram

State Machine

Fault Manager

B2b Comm

Monitor

DCDC

Cooling Manager

LIN

RTOS

Integrity Checks

NVM Manager

Signals

Fan

Tach

EEPROM Module

ADC Driver

Tach Capture

SCCP6

I2C1

PIN\_MGR

SCCP3

UART2

ADC

PWM

SCCP1

SCCP2

UART1

WATCHDOG

SCCP5

SCCP4

Microcontroller

### Component High-Level Descriptions

##### RTOS

This component includes the FreeRTOS source code as well as the task sequences. The RTOS component is what handles program flow and non-critical timing.

##### State Machine

The State Machine component is responsible for the system state and state transitions. The State Machine is the only module that knows the system state. All other modules merely report events or faults but do not determine what to do with those events or faults. The State Machine is notified of events or faults and handles the state transition, notifies any relevant RTOS tasks, updates mode flags, and updates primary-to-secondary command flags.

##### Fault Manager

The Fault Manager component handles the processing of a fault. If the State Machine is notified of a fault and there is not already a fault present in the system, the Fault Manager will be engaged to process the fault. Processing the fault consists of updating diagnostic information to later write to non-volatile memory, and then determining the type of fault and whether there is a retry/recovery strategy. The Fault Manager component orchestrates fault recovery and retry.

##### B2b Comm

The B2b Comm component handles the sending, receiving, and message processing of information between the primary and secondary boards.

##### Monitor

The Monitor component is responsible for periodic monitoring of signals for the purposes of diagnostics. This module utilizes the Signals component to monitor signals in the system for faults and events.

##### DCDC

The DCDC component handles the startup and shutdown of the DCDC, as well as the PWM control during operation.

##### Cooling Manager

The Cooling Manager component handles the application specific logic of the fan cooling. This includes determining the fan target speed, validation, and stuck fan.

##### LIN

The LIN module is the bridge between the primary core software and the LIN libraries that are specific to a certain customer. This module has API for sending and retrieving information over LIN.

##### Integrity Checks

<TODO>

##### NVM Manager

The NVM Manager handles data written in non-volatile memory.

##### EEPROM Module

The EEPROM Module facilitates reading from and writing to EEPROM.

##### Signals

The Signals component provides interfaces to the rest of the modules to retrieve signals information in raw ADC values, scaled engineering units, or flags (if applicable). This is also where other modules can access information sent over from the secondary board.

##### ADC Driver

The ADC Driver component provides ADC data to the Signals module.

##### Fan

The Fan component is responsible for overseeing the speed control of the fans. While the Cooling Manager oversees determining and commanding speed, the Fan module oversees controlling each fan to the commanded speed. This includes utilizing a PID module that controls the fan to the target speed using a PID control algorithm.

##### Tach

The Tach module handles the calculation of fan speed based on the tachometer data provided by the Tach Capture module.

##### Tach Capture

The Tach Capture module is responsible for collecting tachometer data from the fans and providing capture information to the Tach module.

# Software Architecture Design

## Dynamic Architecture

<This section describes the dynamicbehavior. Modes, states of the system, Fault handling. Switching between bootloader and application>

### Modes of operation

<Describe different modes of the software and specify the configuration regarding the modes. Ex: EOL Test, Production, FaultInjection. Specify the state machine, sequence diagram and Data flow of each mode.>

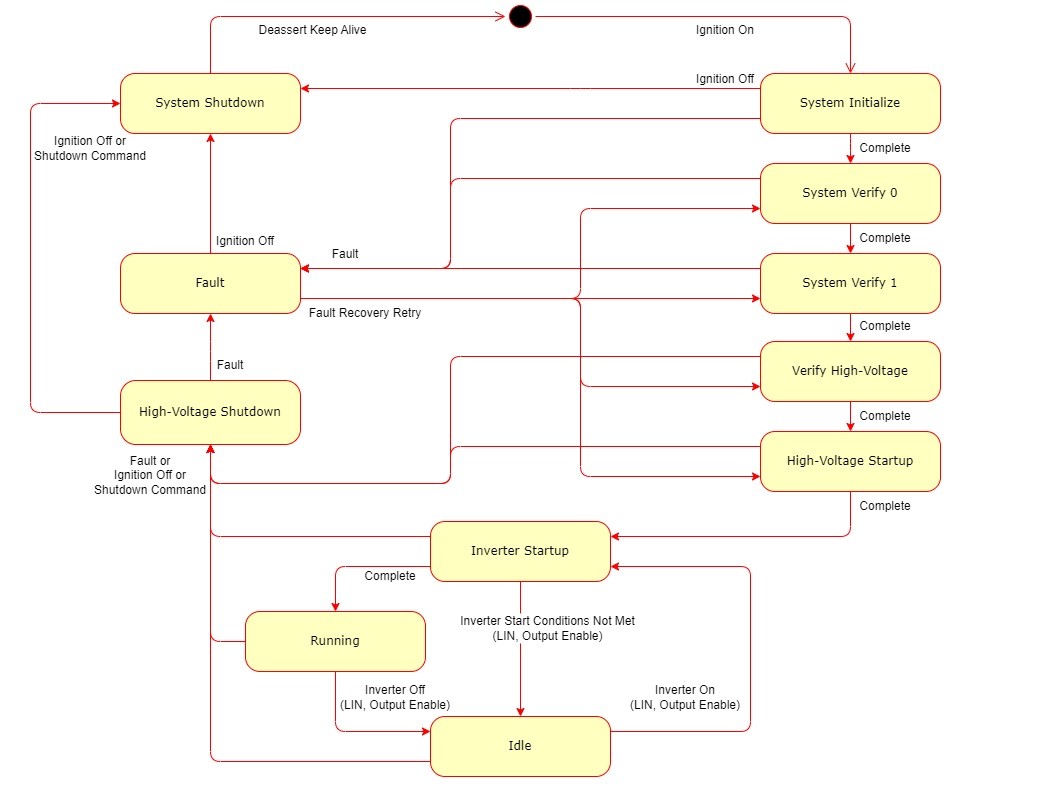
**Control flow:**

<Describe the control flow between operational modes>

#### Mode-1: Production

##### **State machine**

<Describe all the states of the software and transitions between the states. Start-up, normal, shutdown, fault handling etc., >

****

##### Sequence Diagram

###### State Machine Event Notify Sequence

A diagram of a function

Description automatically generated

###### State Machine Fault Notify Sequence

A diagram of a state machine

Description automatically generated

###### System Initialize Sequence

A screenshot of a computer program

Description automatically generated

###### System Verify 0 Sequence

A screenshot of a computer program

Description automatically generated

###### Primary Leakage Test Sequence

A screenshot of a computer program

Description automatically generated

###### System Verify 1 Sequence

A screenshot of a computer program

Description automatically generated

###### Monitor\_VbatMeasIntegrity Sequence

A screenshot of a computer program

Description automatically generated

###### Monitor\_ValidateNTCs Sequence

A screenshot of a computer program

Description automatically generated

###### High Voltage Verify Sequence

A screenshot of a computer

Description automatically generated

###### DCDC Startup Sequence

A screenshot of a computer program

Description automatically generated

###### DCDC Shutdown Sequence

A diagram of a computer system

Description automatically generated

###### DC-Link Leakdown Test Sequence

A screenshot of a computer program

Description automatically generated

###### DC-Link Plausibility Sequence

A screenshot of a computer

Description automatically generated

###### Process Fault Sequence

A screenshot of a computer program

Description automatically generated

###### Retry State at Time of Fault Sequence

A screenshot of a computer program

Description automatically generated

###### High-Voltage/System Shutdown (Ignition Off/Bootloader Request)

A screenshot of a computer program

Description automatically generated

###### Cooling Manager/Fan Viability

A screenshot of a computer application

Description automatically generated

##### Data Flow

<Describe the data flow between the components>

##### **TimingBehavior**

<Show the use cases where timing of the operation is critical. Ex. ADC reading, PWM updates and Interrupt handling, etc.,>

## Static Architecture

### Component Interface Matrix

#### Component Interface Matrix Table

The following shows the component interactions with required interface lists referenced by table values.

Column requires interfaces from Row

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | RTOS | State Machine | Fault Manager | B2bComm | Monitor | DCDC | Cooling Manager | LIN | Integrity Checks | NVM Manager | EEPROM Module | Signals | ADC Driver | Fan | Tach | Tach Capture |
| RTOS |  | [2.1](#_Interface_Matrix_Table_7) | [3.1](#_Interface_Matrix_Table_30) |  | [5.1](#_Interface_Matrix_Table_15) | [6.1](#_Interface_Matrix_Table_19) |  |  |  |  | [11.1](#_Interface_Matrix_Table_52) |  |  |  |  |  |
| State Machine | [1.2](#_Interface_Matrix_Table) |  | [3.2](#_Interface_Matrix_Table_8) | [4.2](#_Interface_Matrix_Table_12) | [5.2](#_Interface_Matrix_Table_16) | [6.2](#_Interface_Matrix_Table_20) | [7.2](#_Interface_Matrix_Table_23) |  |  |  |  |  |  |  |  |  |
| Fault Manager | [1.3](#_Interface_Matrix_Table_1) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| B2bComm | [1.4](#_Interface_Matrix_Table_2) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Monitor | [1.5](#_Interface_Matrix_Table_3) |  | [3.5](#_Interface_Matrix_Table_9) |  |  | [6.5](#_Interface_Matrix_Table_21) |  |  |  |  |  |  |  |  |  |  |
| DCDC | [1.6](#_Interface_Matrix_Table_4) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Cooling Manager | [1.7](#_Interface_Matrix_Table_5) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| LIN | [1.8](#_Interface_Matrix_Table_6) |  | [3.8](#_Interface_Matrix_Table_10) | [4.8](#_Interface_Matrix_Table_13) | [5.8](#_Interface_Matrix_Table_17) |  |  |  |  |  |  |  |  |  |  |  |
| Integrity Checks | [1.9](#_Interface_Matrix_Table_32) |  |  | [4.9](#_Interface_Matrix_Table_33) |  |  |  |  |  |  |  |  |  |  |  |  |
| NVM Manager | [1.10](#_Interface_Matrix_Table_26) |  | [3.10](#_Interface_Matrix_Table_11) |  |  |  |  |  |  |  |  |  |  |  |  |  |
| EEPROM Module | [1.11](#_Interface_Matrix_Table_34) |  |  |  |  |  |  |  |  | [10.11](#_Interface_Matrix_Table_35) |  |  |  |  |  |  |
| Signals | [1.12](#_Interface_Matrix_Table_31) |  |  | [4.12](#_Interface_Matrix_Table_14) | [5.12](#_Interface_Matrix_Table_18) | [6.12](#_Interface_Matrix_Table_22) | [7.12](#_Interface_Matrix_Table_24) | [8.12](#_Interface_Matrix_Table_29) |  |  |  |  |  |  |  |  |
| ADC Driver |  |  |  |  |  |  |  |  |  |  |  | [12.13](#_Interface_Matrix_Table_36) |  |  |  |  |
| Fan |  |  |  |  |  |  | [7.14](#_Interface_Matrix_Table_25) |  |  |  |  |  |  |  |  |  |
| Tach |  |  |  |  |  |  |  |  |  |  |  |  |  | [14.15](#_Interface_Matrix_Table_37) |  |  |
| Tach Capture |  |  |  |  |  |  |  |  |  |  |  |  |  |  | [15.16](#_Interface_Matrix_Table_38) |  |
| ADC |  |  |  |  |  |  |  |  |  |  |  |  | [13.17](#_Interface_Matrix_Table_39) |  |  |  |
| I2C2 |  |  |  |  |  |  |  |  |  |  | [11.18](#_Interface_Matrix_Table_49) |  |  |  |  |  |
| SCCP1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | [16.20](#_Interface_Matrix_Table_39) |
| SCCP2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | [16.21](#_Interface_Matrix_Table_51) |
| PWM |  |  |  |  |  | [6.22](#_Interface_Matrix_Table_48) |  |  |  |  |  |  |  | [14.22](#_Interface_Matrix_Table_50) |  |  |
| SCCP3 | [1.23](#_Interface_Matrix_Table_40) |  |  | [4.23](#_Interface_Matrix_Table_45) |  |  |  |  |  |  |  |  |  |  |  |  |
| SCCP4 | [1.24](#_Interface_Matrix_Table_41) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| SCCP5 | [1.25](#_Interface_Matrix_Table_42) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| SCCP6 | [1.26](#_Interface_Matrix_Table_43) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| UART1 | [1.27](#_Interface_Matrix_Table_44) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| UART2 |  |  |  | [4.28](#_Interface_Matrix_Table_46) |  |  |  |  |  |  |  |  |  |  |  |  |
| WATCHDOG | [1.29](#_Interface_Matrix_Table_47) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

##### Interface Matrix Table 1.2 – RTOS Required Interfaces from State Machine

|  |
| --- |
| **Interface Matrix –1.2** |
| StateMachine\_EventNotify |
| StateMachine\_IsModeActive |
| StateMachine\_FaultNotify |
| StateMachine\_Begin |
| StateMachine\_ProcessEvent |

##### Interface Matrix Table 1.3 – RTOS Required Interfaces from Fault Manager

|  |
| --- |
| **Interface Matrix –1.3** |
| FaultManager\_ProcessFault |
| FaultManager\_UpdateFaultRecord |

##### Interface Matrix Table 1.4 – RTOS Required Interfaces from B2bComm

|  |
| --- |
| **Interface Matrix –1.4** |
| B2b\_Receive |
| B2b\_Init |
| B2b\_ReInit |

##### Interface Matrix Table 1.5 – RTOS Required Interfaces from Monitor

|  |
| --- |
| **Interface Matrix –1.5** |
| Monitor\_ResetDcLinkStabilityTest |
| Monitor\_Run |
| Monitor\_SetFilter |
| Monitor\_Init |
| Monitor\_PrimaryLeakageTest |
| Monitor\_VbatMeasIntegrity |
| Monitor\_ValidateNTCs |
| Monitor\_DcLinkPlausibility |
| Monitor\_LV\_DcLinkLeakDownTest |
| Monitor\_HV\_DcLinkLeakDownTest |

##### Interface Matrix Table 1.6 – RTOS Required Interfaces from DCDC

|  |
| --- |
| **Interface Matrix –1.6** |
| DCDC\_Tick |
| DCDC\_Init |
| DCDC\_Shutdown |
| DCDC\_Startup |

##### Interface Matrix Table 1.7 – RTOS Required Interfaces from Cooling Manager

|  |
| --- |
| **Interface Matrix –1.7** |
| CoolingManager\_Tick |
| CoolingManager\_Init |

##### Interface Matrix Table 1.8 – RTOS Required Interfaces from LIN

|  |
| --- |
| **Interface Matrix –1.8** |
| LIN\_Driver\_CommRx |
| LIN\_Driver\_Service |
| LIN\_Init |
| LIN\_InverterEnable |

##### Interface Matrix Table 1.9 – RTOS Required Interfaces from Integrity Checks

|  |
| --- |
| **Interface Matrix –1.9** |
| CRC\_CCITT\_Calc |

##### Interface Matrix Table 1.10 – RTOS Required Interfaces from NVM Manager

|  |
| --- |
| **Interface Matrix –1.10** |
| NVM\_Init |
| NVM\_SaveRecord |

##### Interface Matrix Table 1.11 – RTOS Required Interfaces from EEPROM Module

|  |
| --- |
| **Interface Matrix –1.11** |
| EEPROM\_Init |

##### Interface Matrix Table 1.12 – RTOS Required Interfaces from Signals

|  |
| --- |
| **Interface Matrix –1.12** |
| Sig\_MainInputEnable |
| Sig\_MainInputDisable |
| Sig\_FanPowerEnable |
| Sig\_FanPowerDisable |
| Sig\_KeepAliveEnable |
| Sig\_KeepAliveDisable |
| Sig\_InitHwBuildID |
| Sig\_ValidHwBuildConfig |
| Sig\_OUTPUT\_ENABLE |
| Sig\_SecPowerEnable |
| Sig\_SecPowerDisable |
| Sig\_ClearSecSignals |

##### Interface Matrix Table 1.23 – RTOS Required Interfaces from SCCP3

|  |
| --- |
| **Interface Matrix –1.23** |
| SCCP3\_TMR\_Start |

##### Interface Matrix Table 1.24 – RTOS Required Interfaces from SCCP4

|  |
| --- |
| **Interface Matrix –1.24** |
| SCCP4\_TMR\_Start |

##### Interface Matrix Table 1.25 – RTOS Required Interfaces from SCCP5

|  |
| --- |
| **Interface Matrix –1.25** |
| SCCP5\_TMR\_Start |

##### Interface Matrix Table 1.26 – RTOS Required Interfaces from SCCP6

|  |
| --- |
| **Interface Matrix –1.26** |
| SCCP6\_TMR\_Start |

##### Interface Matrix Table 1.27 – RTOS Required Interfaces from UART1

|  |
| --- |
| **Interface Matrix –1.27** |
| UART1\_Initialize |

##### Interface Matrix Table 1.29 – RTOS Required Interfaces from WATCHDOG

|  |
| --- |
| **Interface Matrix –1.29** |
| WATCHDOG\_TimerClear |
| WATCHDOG\_TimerSoftwareEnable |

##### Interface Matrix Table 2.1 – State Machine Required Interfaces from RTOS

|  |
| --- |
| **Interface Matrix – 2.1** |
| RTOS\_GetHandle |
| RTOS\_SetStartupTaskReset |
| RTOS\_SetB2bTaskReset |
| xTaskNotifyIndexed |
| xTaskNotify |
| vTaskSuspend |
| vTaskResume |

##### Interface Matrix Table 3.1 – Fault Manager Required Interfaces from RTOS

|  |
| --- |
| **Interface Matrix – 3.1** |
| RTOS\_delay\_ms |

##### Interface Matrix Table 3.2 –Fault Manager Required Interfaces from State Machine

|  |
| --- |
| **Interface Matrix – 3.2** |
| StateMachine\_EventNotify |
| StateMachine\_RetryAtSystemVerify0 |
| StateMachine\_RetryAtSystemVerify1 |

##### Interface Matrix Table 3.5 – Fault Manager Required Interfaces from Monitor

|  |
| --- |
| **Interface Matrix – 3.5** |
| Monitor\_FaultRecOvertemp |
| Monitor\_FaultRecVbatOV |
| Monitor\_FaultRecVbatUV |
| Monitor\_FaultRecVbatProtOV |
| Monitor\_FaultRecVbatProtUV |
| Monitor\_FaultRecVignSupply |
| Monitor\_FaultRecOpenOE |
| Monitor\_FaultRecShortOE |

##### Interface Matrix Table 3.8 – Fault Manager Required Interfaces from LIN

|  |
| --- |
| **Interface Matrix – 3.8** |
| LIN\_SetFault |

##### Interface Matrix Table 3.10 – Fault Manager Required Interfaces from NVM Manager

|  |
| --- |
| **Interface Matrix – 3.10** |
| NVM\_UpdateCountDownFault |
| NVM\_UpdateCountUpFault |
| NVM\_UpdateFaultLifetime |
| NVM\_UpdateMostRecentIgnCycles |
| NVM\_UpdateMostRecentList |

##### Interface Matrix Table 4.2 –B2bComm Required Interfaces from State Machine

|  |
| --- |
| **Interface Matrix – 4.2** |
| StateMachine\_GetPriFlags |
| StateMachine\_FaultNotify |
| StateMachine\_EventNotify |

##### Interface Matrix Table 4.8 – B2bComm Required Interfaces from LIN

|  |
| --- |
| **Interface Matrix – 4.8** |
| LIN\_SetOutputPower |

##### Interface Matrix Table 4.9 – B2bComm Required Interfaces from Integrity Checks

|  |
| --- |
| **Interface Matrix – 4.9** |
| Set\_Sec\_ProgramMemChecksum |

##### Interface Matrix Table 4.12 – B2bComm Required Interfaces from Signals

|  |
| --- |
| **Interface Matrix – 4.12** |
| Sig\_HwBuildID |
| Sig\_UpdateSecSignals |

##### Interface Matrix Table 4.23 – B2bComm Required Interfaces from SCCP3

|  |
| --- |
| **Interface Matrix – 4.23** |
| SCCP3\_TMR\_Stop |
| SCCP3\_TMR\_Start |

##### Interface Matrix Table 4.28 – B2bComm Required Interfaces from UART2

|  |
| --- |
| **Interface Matrix – 4.28** |
| UART2\_SetDevice |
| UART2\_SetRxPacketLength |
| UART2\_Initialize |
| UART2\_Disable |

##### Interface Matrix Table 5.1 –Monitor Required Interfaces from RTOS

|  |
| --- |
| **Interface Matrix – 5.1** |
| Startup\_IsTaskReset |
| RTOS\_delay\_ms |

##### Interface Matrix Table 5.2 – Monitor Required Interfaces from State Machine

|  |
| --- |
| **Interface Matrix – 5.2** |
| StateMachine\_EventNotify |
| StateMachine\_IsModeActive |
| StateMachine\_FaultNotify |

##### Interface Matrix Table 5.8 – Monitor Required Interfaces from LIN

|  |
| --- |
| **Interface Matrix – 5.8** |
| LIN\_SetLidStatus |
| LIN\_InverterEnable |
| LIN\_GetOutputPowerLimit |
| LIN\_GetBootReqStatus |
| LIN\_SetACOutputActive |

##### Interface Matrix Table 5.12 – Monitor Required Interfaces from Signals

|  |
| --- |
| **Interface Matrix – 5.12** |
| Sig\_HwBuildID |
| Sig\_HasFeatureRCD |
| Sig\_HasFeatureAuxOutlet |
| Sig\_HasFeatureDiagLidSwitch |
| Sig\_VBAT\_Raw |
| Sig\_VBAT\_MovAvg |
| Sig\_VBAT\_WindowAvg |
| Sig\_VBAT\_PROTECTED\_Raw |
| Sig\_VBAT\_PROTECTED\_MovAvg |
| Sig\_VBAT\_PROTECTED\_WindowAvg |
| Sig\_VIGN |
| Sig\_VLOGIC |
| Sig\_AdcFiltered |
| Sig\_AdcBridgeTemp |
| Sig\_AcRmsV |
| Sig\_AcRmsA |
| Sig\_InputCurrentEst |
| Sig\_PriRuntimeShortFlag |
| Sig\_AuxOutletA |
| Sig\_AuxOutletPeakOC |
| Sig\_RCD |
| Sig\_RAIL\_5V8 |
| Sig\_ISO\_MON |
| Sig\_OUTPUT\_ENABLE |
| Sig\_ScaledOutputEnableADC |
| Sig\_DCLinkV\_FastFilter |
| Sig\_DCLinkV\_SlowFilter |
| Sig\_MainInputDisable |
| Sig\_MainInputEnable |
| Sig\_FanPowerDisable |
| Sig\_IgnResVDrop |
| Sig\_VdsOvervoltFlag |
| Sig\_CutoffDownstreamPower |

##### Interface Matrix Table 6.1 – DCDC Required Interfaces from RTOS

|  |
| --- |
| **Interface Matrix – 6.1** |
| Startup\_IsTaskReset |
| taskENTER\_CRITICAL |
| taskEXIT\_CRITICAL |

##### Interface Matrix Table 6.2 –DCDC Required Interfaces from State Machine

|  |
| --- |
| **Interface Matrix – 6.2** |
| StateMachine\_IsModeActive |
| StateMachine\_EventNotify |
| StateMachine\_FaultNotify |

##### Interface Matrix Table 6.5 – DCDC Required Interfaces from Monitor

|  |
| --- |
| **Interface Matrix – 6.5** |
| Monitor\_VBAT\_inStartupRange |

##### Interface Matrix Table 6.12 – DCDC Required Interfaces from Signals

|  |
| --- |
| **Interface Matrix – 6.12** |
| Sig\_HwBuildID |
| Sig\_DCLinkV\_SlowFilter |

##### Interface Matrix Table 6.22 – DCDC Required Interfaces from PWM

|  |
| --- |
| **Interface Matrix – 6.22** |
| PWM\_GeneratorEnable |
| PWM\_GeneratorDisable |
| PWM\_SoftwareUpdateRequest |
| PWM\_FaultModeLatchClear |

##### Interface Matrix Table 7.2 – Cooling Manager Required Interfaces from State Machine

|  |
| --- |
| **Interface Matrix – 7.2** |
| StateMachine\_EventNotify |
| StateMachine\_IsModeActive |

##### Interface Matrix Table 7.12 – Cooling Manager Required Interfaces from Signals

|  |
| --- |
| **Interface Matrix – 7.12** |
| Sig\_HwBuildID |
| Sig\_NTC\_AMBIENT |
| Sig\_Power |
| Sig\_FanInputV |
| Sig\_IsFanPowerEnabled |

##### Interface Matrix Table 7.14 – Cooling Manager Required Interfaces from Fan

|  |
| --- |
| **Interface Matrix – 7.14** |
| Fan\_Enable |
| Fan\_SetTargetSpeed |
| Fan\_GetCurrentSpeed |
| Fan\_ControlSpeed |
| Fan\_SaturatedHigh |
| Fan\_Stuck |
| Fan\_Disable |
| Fan\_Init |

##### Interface Matrix Table 8.12 – LIN Required Interfaces from Signals

|  |
| --- |
| **Interface Matrix – 8.12** |
| Sig\_HwBuildID |
| Sig\_LinEnable |
| Sig\_LinDisable |

##### Interface Matrix Table 10.11 – NVM Manager Required Interfaces from EEPROM Module

|  |
| --- |
| **Interface Matrix – 10.11** |
| EEPROM\_Read |
| EEPROM\_Write |

##### Interface Matrix Table 11.1 – EEPROM Module Required Interfaces from RTOS

|  |
| --- |
| **Interface Matrix – 11.1** |
| RTOS\_delay\_ms |

##### Interface Matrix Table 11.18 – EEPROM Module Required Interfaces from I2C2

|  |
| --- |
| **Interface Matrix – 11.18** |
| I2C2\_Initialize |
| I2C2\_MasterQueueIsFull |
| I2C2\_MasterWrite |
| I2C2\_MasterRead |

##### Interface Matrix Table 12.13 – Signals Required Interfaces from ADC Driver

|  |
| --- |
| **Interface Matrix – 12.13** |
| Adc\_SetWindowAvgSize |
| Adc\_GetQ12 |
| Adc\_GetRaw |
| Adc\_WindowAvg |
| Adc\_GetAvgExtraRes |
| Adc\_EngUnitShift |

##### Interface Matrix Table 13.17 – ADC Driver Required Interfaces from ADC

|  |
| --- |
| **Interface Matrix – 13.17** |
| ADC1\_SoftwareTriggerEnable |

##### Interface Matrix Table 14.15 – Fan Required Interfaces from Tach

|  |
| --- |
| **Interface Matrix – 14.15** |
| Tach\_Initialize |
| Tach\_GetCurrentSpeed |
| Tach\_Reset |

##### Interface Matrix Table 14.22 – Fan Required Interfaces from PWM

|  |
| --- |
| **Interface Matrix – 14.22** |
| PWM\_DutyCycleSet |
| PWM\_SoftwareUpdateRequest |
| PWM\_GeneratorEnable |
| PWM\_GeneratorDisable |

##### Interface Matrix Table 15.16 – Tach Required Interfaces from Tach Capture

|  |
| --- |
| **Interface Matrix – 15.16** |
| TachCapture\_Init |
| TachCapture\_GetCurrCap |
| TachCapture\_GetPrevCap |
| TachCapture\_HasNewCapture |
| TachCapture\_NotifyProcessed |
| TachCapture\_Reset |

##### Interface Matrix Table 16.20 – Tach Capture Required Interfaces from SCCP1

|  |
| --- |
| **Interface Matrix – 16.20** |
| SCCP1\_CAPTURE\_Start |
| SCCP1\_CAPTURE\_Data32Read |

##### Interface Matrix Table 16.21 – Tach Capture Required Interfaces from SCCP2

|  |
| --- |
| **Interface Matrix – 16.21** |
| SCCP2\_CAPTURE\_Start |
| SCCP2\_CAPTURE\_Data32Read |

### State Machine Component

#### Component Diagram

The State Machine component manages the state of the system, state transitions based on events/faults, command flags sent in messages to the secondary, and modes the system may be in. The State Machine module is designed to be the only module that knows the state of the system. Any other modules in the system notify the State Machine of events or faults. The State Machine component is notified of events or faults by other modules and handles the state transition, mode update, or command message flag update accordingly.

A screenshot of a computer

Description automatically generated

#### Provided Interface Details

|  |  |  |  |
| --- | --- | --- | --- |
| Interface Name | Description | Parameters | Returns |
| StateMachine\_Begin | Begin state machine, kicking off startup sequence | void | void |
| StateMachine\_ProcessEvent | Process event in state machine and take appropriate action. | Event id 8-bit integer | void |
| StateMachine\_FaultNotify | Register fault code with fault module, take actions in state machine to move to a fault state. | Fault id 8-bit integer | void |
| StateMachine\_EventNotify | Notify state machine task of event to process | Event id 8-bit integer | void |
| StateMachine\_IsModeActive | Check if a particular mode is active or not. | Mode id 8-bit integer | return true if mode is active, false if mode is inactive |
| StateMachine\_GetPriFlags | API for retrieving primary command flags for sending to secondary. | void | 16-bit primary flag bitfield |
| StateMachine\_RetryAtSystemVerify0 | Returns true if fault recovery should restart at system verify 0 | void | bool |
| StateMachine\_RetryAtSystemVerify1 | Returns true if fault recovery should restart at system verify 1 | void | bool |

### RTOS Component

#### Component Diagram

The RTOS component includes the RTOS itself, as well sequences in the individual tasks that run when unblocked or scheduled. If a fault or event causes the need to stop a sequence from continuing and be reset to start at the beginning of the sequence the next time the task is unblocked, this is handled by a reset flag that is checked at each step in the sequence. If the reset flag becomes set then any remaining steps in the sequence should be skipped and execution should be “reset” back to the beginning of the sequence.

A diagram of a diagram

Description automatically generated

#### Provided Interface Details

|  |  |  |  |
| --- | --- | --- | --- |
| Interface Name | Description | Parameters | Returns |
| RTOS\_GetHandle | Return RTOS task handle | task handle enum ID (uint8\_t) | TaskHandle\_t |
| RTOS\_SetStartupTaskReset | Reset RTOS startup task, taking any action specific for that state sequence to be reset. | uint8\_t state sequence to reset | void |
| RTOS\_SetB2bTaskReset | Set Reset B2b task flag to true | void | void |
| RTOS\_delay\_ms | Wrapper for vTaskDelay to enable other modules to use the RTOS delay functionality without having access to the full task control. | uint16\_t delay time in ms | void |
| Startup\_IsTaskReset | API for other modules to check if a startup sequence has been reset. | void | bool |
| xTaskNotifyIndexed | \*See FreeRTOS Documentation |  |  |
| xTaskNotify | \*See FreeRTOS Documentation |  |  |
| vTaskSuspend | \*See FreeRTOS Documentation |  |  |
| vTaskResume | \*See FreeRTOS Documentation |  |  |
| taskENTER\_CRITICAL | \*See FreeRTOS Documentation |  |  |
| taskEXIT\_CRITICAL | \*See FreeRTOS Documentation |  |  |

### Fault Manager Component

#### Component Diagram

When a fault happens in the system, it is processed and recorded in the Fault Manager component. The Fault Manager component deals with the fault based on its associated category (ignition latched, conditional, limited retry). If the fault is categorized as ignition latched, the fault is recorded, and the system does not leave the fault state for the duration of the ignition cycle. If the fault is conditional, the Fault Manager facilitates a wait period before checking a condition to be satisfied to retry operation. If the fault is limited retry, the Fault Manager facilitates a wait period before retrying. The Fault Manager also updates data to be stored in EEPROM for diagnostic purposes.

A diagram of a computer

Description automatically generated

#### Provided Interface Details

|  |  |  |  |
| --- | --- | --- | --- |
| Interface Name | Description | Parameters | Returns |
| FaultManager\_ProcessFault | Process fault by recording fault occurrence and take retry action depending on fault category (ignition latched, conditional retry, limited count retry) | Fault ID (uint8\_t), State ID of system at time when fault occurred (uint8\_t) | void |
| FaultManger\_UpdateFaultRecord | Update any fault statistics and record fault information to EEPROM during an orderly shutdown after loss of ignition or bootloader request. | void | void |
| FaultManager\_GetCurrentFaultID | Return current fault if a fault is present | Fault\_e fault ID | void |

### Board-to-Board Communication (B2b Comm) Component

#### Component Diagram

The Board-to-Board Communication component deals with sending messages to and receiving messages from the secondary. The primary software sends messages to the secondary every 1ms. The secondary responds to messages that the primary has sent. During startup, there is a startup message packet sent to the secondary containing the hardware build configuration number of the primary. The secondary responds to the startup packet with the secondary version and checksum. During running, the primary send message contains command flags for the secondary. The secondary message response contains signal and status information.

A diagram of a computer network

Description automatically generated

#### Provided Interface Details

|  |  |  |  |
| --- | --- | --- | --- |
| Interface Name | Description | Parameters | Returns |
| B2b\_Init | Initialize board to board module | void | void |
| B2b\_ReInit | Reinitialize board to board module | void | void |
| B2b\_Receive | Parse message from secondary board. Notify system of faults or events based on secondary flags if applicable. Retrieve secondary signals for primary monitoring. | void | void |

### Monitor Component

#### Component Diagram

The Monitor component runs through periodic diagnostic checks for each signal looking for faults and/or events. It also contains some diagnostic routines for startup sequences.

A diagram of a computer system

Description automatically generated

#### Provided Interface Details

|  |  |  |  |
| --- | --- | --- | --- |
| Interface Name | Description | Parameters | Returns |
| Monitor\_Init | Initialize monitor module including hardware build dependent variables. | void | void |
| Monitor\_Run | Run through each signal's monitoring routine checking for faults or events to be reported to the state machine. | void | void |
| Monitor\_SetFilter | Update the filtering value for cases when selective monitoring of signals is required. | void | uint32\_t monitor filter bitfield |
| Monitor\_ResetDcLinkStabilityTest | Reset DC Link stability test in the event that it is halted by a fault and will need to be retried from the beginning. | void | void |
| Monitor\_PrimaryLeakageTest | Perform test sequence to check for primary leakage (including validating test preconditions) | void | void |
| Monitor\_VbatMeasIntegrity | Perform Vbat measurement integrity test. | void | void |
| Monitor\_DcLinkPlausibility | Perform DC Link Plausibility test. | void | void |
| Monitor\_LV\_DcLinkLeakDownTest | Perform Low Voltage DC Link Leak down test. | void | void |
| Monitor\_HV\_DcLinkLeakDownTest | Perform High Voltage DC Link Leak down test. | void | void |
| Monitor\_ValidateNTCs | Check all NTCs for open or short circuit. Check for low ambient temperature (informs delayed fan start). | void | void |
| Monitor\_VBAT\_inStartupRange | Check that Vbat is within range for DCDC startup. | void | Fault ID (uint8\_t) |
| Monitor\_FaultRecOvertemp | Check for overtemperature fault recovery | void | bool, true for fault recovered |
| Monitor\_FaultRecVbatOV | Check for Vbat overvoltage fault recovery | void | bool, true for fault recovered |
| Monitor\_FaultRecVbatUV | Check for Vbat undervoltage fault recovery | void | bool, true for fault recovered |
| Monitor\_FaultRecVbatProtOV | Check for Vbat protected overvoltage fault recovery | void | bool, true for fault recovered |
| Monitor\_FaultRecVbatProtUV | Check for Vbat protected undervoltage fault recovery | void | bool, true for fault recovered |
| Monitor\_FaultRecVignSupply | Check for Vign supply fault recovery | void | bool, true for fault recovered |
| Monitor\_FaultRecOpenOE | Check for output enable open circuit fault recovery | void | bool, true for fault recovered |
| Monitor\_FaultRecShortOE | Check for output enable short circuit fault recovery | void | bool, true for fault recovered |

### DCDC Component

#### Component Diagram

The DCDC component handles startup and shutdown of the DCDC control, as well as the control algorithm during operation (including spread spectrum).

A diagram of a computer network

Description automatically generated

#### Provided Interface Details

|  |  |  |  |
| --- | --- | --- | --- |
| Interface Name | Description | Parameters | Returns |
| DCDC\_Init | Initialize DCDC module including hardware build dependent variables. | void | void |
| DCDC\_Tick | Update control parameters for DCDC module taking into account burst mode. | void | void |
| DCDC\_Startup | Startup DCDC module with soft start routine after verifying Vbat is within range. | void | void |
| DCDC\_Shutdown | Shutdown DCDC PWMs | void | void |

### Cooling Manager Component

#### Component Diagram

The Cooling Manager component handles the logic related to cooling the board via the fans. The Cooling Manager determines a target speed to command the fans to taking into account temperature, output power, and beat frequency avoidance. The Cooling Manager component also handles the viability test for each fan and oversees monitoring for a stuck or faulty fan.

A diagram of a computer

Description automatically generated

#### Provided Interface Details

|  |  |  |  |
| --- | --- | --- | --- |
| Interface Name | Description | Parameters | Returns |
| CoolingManager\_Init | Initialize Cooling Manager module including hardware build dependent variables. | void | void |
| CoolingManager\_Tick | Determine fan speed commands, control fans to commanded speeds, monitor for stuck fans, handle fan viability routines. | void | void |

### LIN Component

#### Component Diagram

HVPO\_SWAD\_Primary\_0001

The LIN component is the bridge between the primary software and the LIN libraries.

A diagram of a network

Description automatically generated

#### Provided Interface Details

|  |  |  |  |
| --- | --- | --- | --- |
| Interface Name | Description | Parameters | Returns |
| LIN\_Driver\_CommRx | LIN data reading from the buffers and process the LIN frame | void | void |
| LIN\_Driver\_Service | Run the Driver operations and commanding the inverter operation | void | void |
| LIN\_Init | Initialize LIN module | void | void |
| LIN\_InverterEnable | Do aggregated signals from LIN allow inverter to be enabled? | void | uint16\_t (true/false) |
| LIN\_SetFault | Set fault over LIN. Sends internal fault, LIN library decides how to display fault. | fault ID (uint8\_t), fault category ID (uint8\_t) | void |
| LIN\_SetLidStatus | Set lid status over LIN. | status (1 or 0, uint16\_t) | void |
| LIN\_GetOutputPowerLimit | Get output power limit communicated over LIN. | void | LIN output power limit in watts x10 |
| LIN\_GetBootReqStatus | Get boot request status | void | Boot request stats flag (uint8\_t) |
| LIN\_SetACOutputActive | Notify LIN of AC output active (1) or inactive (0) | uint16\_t active | void |

Ref:{[STLADT2KW-370](https://jira-cascoproducts.msappproxy.net/browse/STLADT2KW-370), [STLADT2KW-1227](https://jira-cascoproducts.msappproxy.net/browse/STLADT2KW-1227)}

### Integrity Checks

#### Component Diagram

A diagram of a network

Description automatically generated

HVPO\_SWAD\_Primary\_0002

#### Provided Interface Details

|  |  |  |  |
| --- | --- | --- | --- |
| Interface Name | Description | Parameters | Returns |
| Set\_Sec\_ProgramMemChecksum |  | uint32\_t received\_checksum | void |
| CRC\_CCITT\_Calc |  | void | bool |
|  |  |  |  |

Ref:{PFE-49 }

### NVM Manager Component

#### Component Diagram

The NVM Manager handles the Non-volatile memory including manager the fault record. The NVM Manager is the sole accessor of the EEPROM module read and write functions. The NVM Manager provides API for the fault record to be updated and saved to EEPROM.

#### Provided Interface Details

|  |  |  |  |
| --- | --- | --- | --- |
| Interface Name | Description | Parameters | Returns |
| NVM\_Init | Init NVM Manager | void | void |
| NVM\_SaveRecord | Write fault record to EEPROM and ignition cycles if ignOff is true | bool ignOff | void |
| NVM\_GetFaultRecordData | Get fault record data | uint16\_t fault buffer index, uint8\_t length, uint8\_t \*response buffer | void |
| NVM\_GetRecentFaultRecords | Get most recent faults list | uint8\_t length, uint8\_t \*response buffer | void |
| NVM\_UpdateFaultLifetime | Update lifetime count of fault in fault record | Fault\_e id | void |
| NVM\_UpdateCountUpFault | Increase count up/down value for fault id in fault record by specified amount | Fault\_e id, uint16\_t countUpIncr | void |
| NVM\_UpdateCountDownFault | Decrease count up/down value for fault id in fault record by specified amount | Fault\_e id, uint16\_t countDownIncr | void |
| NVM\_UpdateMostRecentList | Add fault to most recent faults list in fault record | Fault\_e currentFault | void |
| NVM\_UpdateMostRecentIgnCycles | Update most recent fault list ignition cycles since occurrence count | void | void |

### EEPROM Component

#### Component Diagram

The EEPROM module handles the read and write functionality to the EEPROM chip via I2C2.

A diagram of a network

Description automatically generated

#### Provided Interface Details

|  |  |  |  |
| --- | --- | --- | --- |
| Interface Name | Description | Parameters | Returns |
| EEPROM\_Write | Write to EEPROM | uint8\_t \*(dataToWrite),  uint16\_t dataLen, uint16\_t address | void |
| EEPROM\_Read | Read from EEPROM | uint8\_t \*(placeForReadData), uint16\_t dataLen, uint16\_t address | void |
| EEPROM\_Init | Initialize EEPROM Module | void | void |

### Signals Component

#### Component Diagram

The Signals component contains interfaces for the rest of the system to access signals. The Signals component makes signals from the ADC Driver available in either ADC counts or relevant units and scale. It also contains interfaces related to the hardware build configuration ID as well as any flags in the hardware build configuration.

A diagram of a computer

Description automatically generated

#### Provided Interface Details

|  |  |  |  |
| --- | --- | --- | --- |
| Interface Name | Description | Parameters | Returns |
| Sig\_InitHwBuildID | Read voltage influenced by hardware build resistor. Determine hardware build ID from voltage level. Initialize hardware build ID variables. | void | void |
| Sig\_HasFeatureRCD | Does this hardware build have the RCD feature? | void | bool |
| Sig\_HasFeatureAuxOutlet | Does this hardware build have the Aux Outlet feature? | void | bool |
| Sig\_HasFeatureDiagLidSwitch | Does this hardware build have the diagnostic lid switch feature? | void | bool |
| Sig\_ValidHwBuildConfig | Does the hardware build ID represent a valid configuration? | void | bool |
| Sig\_AdcFiltered | Returns filtered ADC value for specified channel | ADC channel (uint8\_t) | uint16\_t |
| Sig\_AdcRaw | Returns the unfiltered ADC value for specified channel | ADC channel (uint8\_t) | uint16\_t |
| Sig\_VBAT\_Raw | Returns only hardware filtered Vbat in volts x 100 | void | uint16\_t |
| Sig\_VBAT\_MovAvg | Returns moving average filtered (4 place) Vbat in volts x100 | void | uint16\_t |
| Sig\_VBAT\_WindowAvg | Returns window average filtered (25 place) Vbat in volts x100 | void | uint16\_t |
| Sig\_VBAT\_PROTECTED\_Raw | Returns only hardware filtered Vbat protected in volts x 100 | void | uint16\_t |
| Sig\_VBAT\_PROTECTED\_MovAvg | Returns moving average filtered (4 place) Vbat protected in volts x 100 | void | uint16\_t |
| Sig\_VBAT\_PROTECTED\_WindowAvg | Returns window average filtered (25 place) Vbat protected in volts x 100 | void | uint16\_t |
| Sig\_VLOGIC | Returns software filtered Vlogic in volts x 100 | void | uint16\_t |
| Sig\_VIGN | Returns hardware filtered Vign in volts x 100 | void | uint16\_t |
| Sig\_RAIL\_5V8 | Returns software filtered 5.8V rail in volts x 100 | void | uint16\_t |
| Sig\_ISO\_MON | Returns software filtered isolation monitor in volts x 100 | void | uint16\_t |
| Sig\_HW\_BUILD\_V | Returns software filtered hardware build voltage in volts x100 | void | uint16\_t |
| Sig\_OUTPUT\_ENABLE\_ADC | Returns software filtered output enable ADC value | void | uint16\_t |
| Sig\_ScaledOutputEnableADC | Returns output enable ADC value scaled by Vbat | void | uint16\_t |
| Sig\_NTC\_RECTIFIER | Returns software filtered NTC RECTIFIER value in degrees Celsius x 100 | void | uint16\_t |
| Sig\_NTC\_DCDC | Returns software filtered NTC DCDC value in degrees Celsius x 100 | void | uint16\_t |
| Sig\_NTC\_AMBIENT | Returns software filtered NTC AMBIENT value in degrees Celsius x 100 | void | uint16\_t |
| Sig\_NTC\_INPUT\_FET | Returns software filtered NTC input FET value in degrees Celsius x 100 | void | uint16\_t |
| Sig\_FanInputV | Returns fan input voltage in volts x100 | void | uint16\_t |
| Sig\_KeepAliveEnable | Enable Keep Alive | void | void |
| Sig\_KeepAliveDisable | Disable Keep Alive | void | void |
| Sig\_CutoffDownstreamPower | Disables Main Input, Fan Enable, and Secondary Power | void | void |
| Sig\_MainInputEnable | Enable Main Input | void | void |
| Sig\_MainInputDisable | Disable Main Input | void | void |
| Sig\_FanPowerEnable | Enable Fan Power | void | void |
| Sig\_FanPowerDisable | Disable Fan Power | void | void |
| Sig\_IsFanPowerEnabled | Is fan power enabled? | void | uint16\_t |
| Sig\_SecPowerEnable | Enable power to secondary board | void | void |
| Sig\_SecPowerDisable | Disable power to secondary | void | void |
| Sig\_LinEnable | Enable LIN | void | void |
| Sig\_LinDisable | Disable LIN | void | void |
| Sig\_DCLinkV\_FastFilter | Returns DCLink voltage in voltsx10,  With filter size 4 | void | uint16\_t |
| Sig\_DCLinkV\_SlowFilter | Returns DCLink voltage in voltsx10, with filter size 128 | void | uint16\_t |
| Sig\_AcRmsV | Returns AC output RMS voltage in voltsx10 | void | uint16\_t |
| Sig\_AcRmsA | Returns AC output RMS current in ampsx10 | void | uint16\_t |
| Sig\_Power | Returns AC output RMS Power in wattsx10 | void | uint16\_t |
| Sig\_AuxOutletA | Returns auxiliary outlet RMS current in ampsx10 | void | uint16\_t |
| Sig\_AuxOutletPeakOC | Returns auxiliary outlet peak overcurrent flag | void | uint16\_t |
| Sig\_AdcBridgeTemp | Returns bridge Temp value sent from secondary in ADC counts | void | uint16\_t |
| Sig\_BridgeTemp | Returns bridge temp value sent from secondary in degrees Celsius x100 | void | uint16\_t |
| Sig\_RCD | Returns RCD voltage in voltsx100 | void | uint16\_t |
| Sig\_UpdateSecSignals | Update secondary signals with information from secondary signal response packet | pointer to secondary response status packet struct | void |
| Sig\_ClearSecSignals | Clear secondary signals saved on primary | void | void |
| Sig\_OUTPUT\_ENABLE | Return output enable (1 or 0) | void | uint16\_t |
| Sig\_PriRuntimeShortFlag | Returns primary runtime short circuit flag | void | uint16\_t |
| Sig\_VdsOvervoltFlag | Returns Vds overvolt flag | void | uint16\_t |
| Sig\_InputCurrentEst | Returns input current estimate in amps | void | uint16\_t |
| Sig\_IgnResVDrop | Returns Ignition resistor voltage drop in volts x 100 | void | uint16\_t |
| Sig\_HwBuildID | Returns hardware build ID | void | uint16\_t |

### ADC Driver Component

#### Component Diagram

The ADC Driver component supplies information from the ADC channels to the Signals component. All active ADC channels are sampled every 1ms. Signals utilizing a moving average filter get a new value every 1ms, stored with optional extra resolution.

A white rectangular sign with black text

Description automatically generated

#### Provided Interface Details

|  |  |  |  |
| --- | --- | --- | --- |
| Interface Name | Description | Parameters | Returns |
| Adc\_SetWindowAvgSize | Set size of window average (must be synchronized with AC output frequency) | uint8\_t channel | uint16\_t |
| Adc\_GetQ12 | Get filtered Q12 ADC channel value | uint8\_t channel | uint16\_t |
| Adc\_GetRaw | Get unfiltered ADC channel value | uint8\_t channel | uint16\_t |
| Adc\_WindowAvg | Return window average for specified channel | uint8\_t channel | uint16\_t |
| Adc\_GetAvgExtraRes | Return filtered ADC value including extra resolution | uint8\_t channel | uint16\_t |
| Adc\_EngUnitShift | Return bit resolution of ADC (12-bit) plus extra resolution | uint8\_t channel | uint8\_t |

### Fan Component

#### Component Diagram

While the Cooling Manager component handles the application specific logic for cooling, the Fan module is responsible for controlling each fan to the commands of the Cooling Manager. The Fan component utilizes a PID controller to control the speed of each fan to a target. The current speed of the fan is calculated from the tachometer signals.

A diagram of a computer system

Description automatically generated

#### Provided Interface Details

|  |  |  |  |
| --- | --- | --- | --- |
| Interface Name | Description | Parameters | Returns |
| Fan\_Init | Initialize fan module including instances for each fan, stuck fan parameters, and the control update interval | FanStuckParams\_t \*stuck, uint16\_t interval | void |
| Fan\_SetTargetSpeed | Set target speed for specified fan | fan ID, uint16\_t target speed | void |
| Fan\_GetCurrentSpeed | Get current speed of specified fan | fan ID | uint16\_t current speed |
| Fan\_SaturatedHigh | Return true/false for if specified fan is saturated at the maximum duty cycle | fan ID | uint16\_t (true/false) |
| Fan\_Stuck | Return true/false if fan is determined to be stuck | fan ID | bool |
| Fan\_ControlSpeed | Control fans to target speed | fan ID, uint16\_t input voltage | void |
| Fan\_Enable | Enable fan PWMs | fan ID | void |
| Fan\_Disable | Disable fan PWMs | fan ID | void |

### Tach Component

#### Component Diagram

The Tach component is responsible for using the tach capture information and interpreting it in terms of fan speed.

A diagram of a computer

Description automatically generated

#### Provided Interface Details

|  |  |  |  |
| --- | --- | --- | --- |
| Interface Name | Description | Parameters | Returns |
| Tach\_Initialize | Initialize tach module with parameters from the fan module | Tach\_t \*tach, TachParams\_t \*params, uint8\_t id | void |
| Tach\_GetCurrentSpeed | Calculate current fan speed for this tach instance based on time between tach captures. Returns speed in RPMs | Tach\_t \*tach | uint16\_t |
| Tach\_Reset | Initiates a Tach Capture clearing of the capture variables | Tach\_t \*tach | void |

### Tach Capture Component

#### Component Diagram

The Tach Capture component is responsible for interfacing with the capture compare peripherals and making information from them available to the Tach component.

A screenshot of a computer

Description automatically generated

#### Provided Interface Details

|  |  |  |  |
| --- | --- | --- | --- |
| Interface Name | Description | Parameters | Returns |
| TachCapture\_Init | Initialize Tach Capture module | uint8\_t id | void |
| TachCapture\_GetCurrCap | Return current capture reading | uint8\_t id | uint32\_t |
| TachCapture\_GetPrevCap | Return previous capture reading | uint8\_t id | uint32\_t |
| TachCapture\_HasNewCapture | Return true if there has been a new tach capture since last processing of reading | uint8\_t id | bool |
| TachCapture\_NotifyProcessed | Notify the Tach Capture module the reading has been processed, setting the “new capture” flag to false | uint8\_t id | void |
| TachCapture\_Reset | Clear capture readings | uint8\_t id | void |

# Platform Architecture

## Hardware and Software interaction

<Show the picture and details of Hardware and software interactions/interfaces of each component>

## Microcontroller pin functionality allocation

### Peripherals

| **Module** | **Channels** | **Purpose** | **Comments** |
| --- | --- | --- | --- |
| ADC1 | 1-25 | Read Signals |  |
| CMP1 |  | Comparator tripped indicates Primary Runtime Short Event | comparator output directly fed into PWM module (PG2 Fault PCI) |
| CMP2 |  | Comparator tripped indicates Vds overvolt event | comparator output directly fed into PWM module (PG2 CL PCI) |
| TMR1 |  | RTOS tick interrupt handler |  |
| I2C2 |  | Communication with EEPROM chip |  |
| MCCP9 |  | Used as a 1ms timer interrupt to kick off round robin ADC sampling of the channels |  |
| OPA1 |  | Provide low source impedance to the ADC (used as a signal buffer) |  |
| PIN\_MGR |  | Set up GPIO |  |
| PWM | PG2 | Controls first phase of DCDC | PCI inputs for primary runtime short and Vds overvolt |
| PWM | PG3 | Controls second phase of DCDC | bypass enable for PCI actions |
| PWM | PG4 | Controls third phase of DCDC | bypass enable for PCI actions |
| PWM | PG5 | Controls fan 1 speed |  |
| PWM | PG7 | Controls fan 2 speed |  |
| SCCP1 |  | Tach capture for fan 1 |  |
| SCCP2 |  | Tach capture for fan 2 |  |
| SCCP3 |  | Used as 1ms Timer interrupt | B2b send message to secondary |
| SCCP4 |  | Used as 1ms Timer interrupt | Interrupt deferred processing for DCDC Task |
| SCCP5 |  | Used as 1ms Timer interrupt | Interrupt deferred processing for Monitor Task |
| SCCP6 |  | Used as 1ms Timer interrupt | Interrupt deferred processing for LIN Task |
| UART1 |  | LIN Communication |  |
| UART2 |  | B2b Communication |  |

### Clocks selections

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SL No** | **Clock Base** | **Scalar** | **Clock Source** | **Clock Type** | **Frequency** |
| 1 | XOSC (8MHZ) | x12 | PLL0 | PHI | 192 MHz |
| 2 | PHI | ÷3 | LIN\_CLK | UART Clock (CUT 2) | 64 MHz |
|  |  |  |  |  |  |

### Recommended I/O Pin Function Allocation

### Interrupts

| **IRQ No.** | **Vector Number** | **ISR Name** | **Priority Level** | **Details** |
| --- | --- | --- | --- | --- |
| 1 | 9 | \_T1Interrupt | 1 | RTOS tick handler |
| 6 | 14 | \_CCP1Interrupt | 1 | Tach Capture for Fan 1 |
| 11 | 19 | \_U1RXInterrupt | 1 | LIN Rx byte |
| 23 | 31 | \_CCP2Interrupt | 1 | Tach Capture for Fan 2 |
| 27 | 35 | \_U2RXInterrupt | 1 | B2b Rx byte |
| 28 | 36 | \_U2TXInterrupt | 1 | B2b Tx byte |
| 36 | 44 | \_CCT3Interrupt | 1 | B2b initiate send packet |
| 38 | 46 | \_MI2C2Interrupt | 1 | EEPROM read/write |
| 41 | 49 | \_CCT4Interrupt | 1 | Interrupt deferred DCDC Task |
| 44 | 52 | \_CCT5Interrupt | 1 | Interrupt deferred Monitor Task |
| 47 | 55 | \_CCT6Interrupt | 1 | Interrupt deferred LIN Task |
| 68 | 76 | \_PWM2Interrupt | 2 | DCDC spread spectrum update |
| 91 | 99 | \_ADCAN0Interrupt | 1 | ADC channel 0 conversion complete |
| 93 | 101 | \_ADCAN2Interrupt | 1 | ADC channel 2 conversion complete |
| 95 | 103 | \_ADCAN4Interrupt | 1 | ADC channel 4 conversion complete |
| 96 | 104 | \_ADCAN5Interrupt | 1 | ADC channel 5 conversion complete |
| 97 | 105 | \_ADCAN6Interrupt | 1 | ADC channel 6 conversion complete |
| 102 | 110 | \_ADCAN11Interrupt | 1 | ADC channel 11 conversion complete |
| 103 | 111 | \_ADCAN12Interrupt | 1 | ADC channel 12 conversion complete |
| 105 | 113 | \_ADCAN14Interrupt | 1 | ADC channel 14 conversion complete |
| 106 | 114 | \_ADCAN15Interrupt | 1 | ADC channel 15 conversion complete |
| 107 | 115 | \_ADCAN16Interrupt | 1 | ADC channel 16 conversion complete |
| 108 | 116 | \_ADCAN17Interrupt | 1 | ADC channel 17 conversion complete |
| 109 | 117 | \_ADCAN18Interrupt | 1 | ADC channel 18 conversion complete |
| 110 | 118 | \_ADCAN19Interrupt | 1 | ADC channel 19 conversion complete |
| 182 | 190 | \_CCT9Interrupt | 1 | Kick off sampling ADC channels round robin |

# Resource usage

## Task Management

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl. No** | **Task Name** | **Periodicity (ms)** | **Priority** |
|  | LIN Communication |  | 1 |
|  | LIN Driver Service |  | 1 |
|  | Startup Task |  | 2 |
|  | Shutdown Task |  | 2 |
|  | Fault Task |  | 2 |
|  | Cooling Task | 256ms | 3 |
|  | B2b Receive Task | 1ms | 4 |
|  | Monitor Task | 1ms | 4 |
|  | DCDC Task | 1ms | 4 |
|  | State Machine Task |  | 5 |

## Memory Management

<Specify all partitions of memory (Boot, Application, reserved, etc.,) and segments of partition (.text, .rodata, .usercode, etc.,)>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sl. No** | **Memory partition type** | **Memory segment** | **Memory range** | **Size (bytes)** | **Purpose** |
|  | **Bootloader** |  |  |  |  |
|  |  | Program(.text) |  |  |  |
|  |  | data |  |  |  |
|  | **Application** |  |  |  |  |
|  |  | Program(.text) |  |  |  |
|  |  | data |  |  |  |

# General Information of Development

## IDE details

MPLAB X IDE v6.05, DFP: dsPIC33CK-MP\_DFP v1.6.176

## Compiler details

XC16 v1.70

## Project setup

<Provide the details of boot/app/calibration project structures >

## Application configuration details

The following C macros are defined in project properties.

|  |  |
| --- | --- |
| **Macro** | **Purpose** |
| LITTLE\_ENDIAN | Used for parsing B2b messages correctly |
| FULL\_BUILD | Full build indicates a fully populated set of boards. Non full build would indicate partially populated boards without power components used for development. |
| \_\_PRIMARY\_\_ |  |
| BUILD\_CA | Build is specified because BUILD\_BA requires GPIO configuration change and a change in handling output enable. |
| \_\_bootloadable\_\_ |  |

The hardware build configuration is specified in a file called hwBuildConfig.h in the same directory level as the primary project. The hardware build configuration header file should contain the following definitions.

|  |  |  |  |
| --- | --- | --- | --- |
| **Macro** | **Value** | **Purpose** | **Module** |
| HBX\_BUILD\_ID | 1 | Stellantis 12V 2kW 120V is hardware build ID 1 | Signals |
| HBX\_AUX\_OUTLET\_CURR\_MAX | 3.6 | Auxiliary Outlet RMS Current Max in amps  \*Only applicable on units where HBX\_FEATURE\_AUX\_OUTLET = 1 | Monitor |
| HBX\_BURST\_MODE\_THRES | 300 | DC-Link voltage at which DCDC enters burst mode | DCDC |
| HBX\_BURST\_RECOVERY\_THRES | 290 | DC-Link voltage at which DCDC exits burst mode | DCDC |
| HBX\_DCDC\_BOOST\_RATIO\_MAX | 23 | Used to calculate max in DC-Link voltage plausibility range | Monitor |
| HBX\_DCDC\_BOOST\_RATIO\_MIN | 16 | Used to calculate min in DC-Link voltage plausibility range | Monitor |
| HBX\_DCDC\_TABLE\_PERIOD\_0 | 27 | Defines DCDC PWM period vs. DC-Link voltage interpolation table | SCSC |
| HBX\_DCDC\_TABLE\_PERIOD\_1 | 27 | Defines DCDC PWM period vs. DC-Link voltage interpolation table | DCDC |
| HBX\_DCDC\_TABLE\_PERIOD\_2 | 4 | Defines DCDC PWM period vs. DC-Link voltage interpolation table | DCDC |
| HBX\_DCDC\_TABLE\_PERIOD\_3 | 4 | Defines DCDC PWM period vs. DC-Link voltage interpolation table | DCDC |
| HBX\_DCDC\_TABLE\_PERIOD\_4 | 6 | Defines DCDC PWM period vs. DC-Link voltage interpolation table | DCDC |
| HBX\_DCDC\_TABLE\_PERIOD\_5 | 6 | Defines DCDC PWM period vs. DC-Link voltage interpolation table | DCDC |
| HBX\_DCDC\_TABLE\_VOLTAGE\_0 | 0 | Defines DCDC PWM period vs. DC-Link voltage interpolation table | DCDC |
| HBX\_DCDC\_TABLE\_VOLTAGE\_1 | 100 | Defines DCDC PWM period vs. DC-Link voltage interpolation table | DCDC |
| HBX\_DCDC\_TABLE\_VOLTAGE\_2 | 150 | Defines DCDC PWM period vs. DC-Link voltage interpolation table | DCDC |
| HBX\_DCDC\_TABLE\_VOLTAGE\_3 | 235 | Defines DCDC PWM period vs. DC-Link voltage interpolation table | DCDC |
| HBX\_DCDC\_TABLE\_VOLTAGE\_4 | 290 | Defines DCDC PWM period vs. DC-Link voltage interpolation table | DCDC |
| HBX\_DCDC\_TABLE\_VOLTAGE\_5 | 350 | Defines DCDC PWM period vs. DC-Link voltage interpolation table | DCDC |
| HBX\_DCLINK\_LEAKDOWN\_INITIAL\_MIN | 10 | Defines initial minimum DC-Link voltage used in high voltage and low voltage DC-Link Leakdown tests | Monitor |
| HBX\_DCLINK\_MAX | 330 | Defines DC-Link voltage threshold for DC-Link overvoltage fault | Monitor |
| HBX\_DCLINK\_MIN | 150 | Defines DC-Link voltage threshold for DC-Link undervoltage fault | Monitor |
| HBX\_DCLINK\_PLAUS\_MAX | 300 | Defines maximum bound the calculated upper DC-Link plausibility can be | Monitor |
| HBX\_DCLINK\_STAB\_RANGE | 3 | Defines maximum difference between one DC-Link voltage value to the next when monitoring for DC-Link voltage stability | Monitor |
| HBX\_FAN\_OFF\_MODE\_HYST\_WINDOW | 50 | Defines hysteresis in RPMs for turn on/turn off thresholds when cooling manager power vs speed interpolation table has the fans off in the first speed position. | Cooling Manager |
| HBX\_FAN\_TABLE\_POWER\_0 | 0 | Defines Cooling Manager power vs. speed interpolation table | Cooling Manager |
| HBX\_FAN\_TABLE\_POWER\_1 | 200 | Defines Cooling Manager power vs. speed interpolation table | Cooling Manager |
| HBX\_FAN\_TABLE\_POWER\_2 | 300 | Defines Cooling Manager power vs. speed interpolation table | Cooling Manager |
| HBX\_FAN\_TABLE\_POWER\_3 | 400 | Defines Cooling Manager power vs. speed interpolation table | Cooling Manager |
| HBX\_FAN\_TABLE\_POWER\_4 | 1400 | Defines Cooling Manager power vs. speed interpolation table | Cooling Manager |
| HBX\_FAN\_TABLE\_SPEED\_0 | 1000 | Defines Cooling Manager power vs. speed interpolation table | Cooling Manager |
| HBX\_FAN\_TABLE\_SPEED\_1 | 1000 | Defines Cooling Manager power vs. speed interpolation table | Cooling Manager |
| HBX\_FAN\_TABLE\_SPEED\_2 | 1000 | Defines Cooling Manager power vs. speed interpolation table | Cooling Manager |
| HBX\_FAN\_TABLE\_SPEED\_3 | 1000 | Defines Cooling Manager power vs. speed interpolation table | Cooling Manager |
| HBX\_FAN\_TABLE\_SPEED\_4 | 5400 | Defines Cooling Manager power vs. speed interpolation table | Cooling Manager |
| HBX\_FEATURE\_AUX\_OUTLET | 0 | Defines whether hardware has auxiliary outlet capability | Signals |
| HBX\_FEATURE\_DIAG\_LID\_SWITCH | 1 | Defines whether hardware has diagnostic lid switch | Signals |
| HBX\_FEATURE\_RCD | 0 | Defines whether hardware has RCD functionality | Signals |
| HBX\_HV\_DCLINK\_LEAKDOWN\_MAX | 10 | Defines maximum allowable DC-Link voltage Leakdown during DC-Link high voltage Leakdown test | Monitor |
| HBX\_INPUT\_CURR\_EST\_MAX | 170 | Defines maximum input current | Monitor |
| HBX\_LIMP\_POWER\_LIMIT | 400 | Defines power maximum during Limp mode | Monitor |
| HBX\_LV\_DCLINK\_LEAKDOWN\_MAX | 3 | Defines maximum allowable DC-Link voltage Leakdown during DC-Link low voltage Leakdown test | Monitor |
| HBX\_NOM\_OUTPUT\_VOLTAGE | 120 | Nominal output voltage | Monitor |
| HBX\_POWER\_LIMIT | 2000 | Power Limit | Monitor |
| HBX\_RMSV\_MAX | 130 | Maximum AC RMS voltage | Monitor |
| HBX\_RMSV\_MIN | 108 | Minimum AC RMS voltage | Monitor |
| HBX\_SYSTEM\_EFFICIENCY | 0.9 | System efficiency used in input current estimate calculation | Signals |
| HBX\_TURNS\_RATIO | 9 | \*Unused | Monitor |
| HBX\_VBAT\_MAX\_THRES | 16.1 | Maximum threshold used to determine battery overvoltage | Monitor |
| HBX\_VBAT\_MIN\_THRES | 10.9 | Minimum threshold used to determine battery undervoltage | Monitor |
| HBX\_VBAT\_STARTUP\_MAX\_THRES | 15.1 | Maximum threshold used to determine if battery voltage is in range to startup DCDC | Monitor |
| HBX\_VBAT\_STARTUP\_MIN\_THRES | 11.9 | Minimum threshold used to determine if battery voltage is in range to startup DCDC | Monitor |
| HBX\_WINDOW\_AVG\_SIZE | 25 | Window average size, represents first integer multiple in ms (three half cycles) | Signals |

**Document Change History**

The following is the document control for revisions to this document.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Version** | **Description** | **Author(s)** | **Revision Date** | **Approved By (Title)** | **Approved By (Name)** | **Approved Date** |
| 0.00 or A | Initial Version | Hassan Arghavani | Exp: 27Apr21 |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Table 3- Document Change History