**Software Detail Design Template**

**CPDP 1.0**

**Process Document**

**<Control Number>**

**Revision:** **<0.0>**

**Release Date: 08Jul22**

**Status: *Draft***

**Prepared By:**

**Rahma Fkaier**

**PREFACE**

The template conforms to the Institute of Electrical and Electronics Engineers (IEEE) Standard for Systems and Software Engineering - Project Management Plans, IEEE Standard 16326:2009, for format and content.

*ISO/IEC/IEEE 16326:2009 provides normative content specifications for project management plans covering software projects, and software-intensive system projects. It also provides detailed discussion and advice on applying a set of project processes that are common to both the software and system life cycle as covered by ISO/IEC 12207:2008 (IEEE Std 12207-2008) and ISO/IEC 15288:2008 (IEEE Std 15288- 2008), respectively. The discussion and advice are intended to aid in the preparation of the normative content of project management plans. ISO/IEC/IEEE 16326:2009 is the result of the harmonization of ISO/IEC TR 16326:1999 and IEEE Std 1058-1998.*

Table of Contents

[1 Overview 6](#_Toc108012585)

[1.1 Purpose 6](#_Toc108012586)

[1.2 References 7](#_Toc108012587)

[1.3 Definitions 8](#_Toc108012588)

[1.4 Realization constraints and targets 10](#_Toc108012589)

[2 Detailed SW Design 11](#_Toc108012590)

[2.1 Traceability matrix 11](#_Toc108012591)

[2.2 External interaction 11](#_Toc108012592)

[2.2.1 Required interfaces 12](#_Toc108012593)

[2.2.2 Provided Interfaces 12](#_Toc108012594)

[2.3 SW Component internal breakdown 13](#_Toc108012595)

[2.4 General Data description 16](#_Toc108012596)

[2.4.1 EEPROM DATA 16](#_Toc108012597)

[2.4.2 Constants 16](#_Toc108012598)

[2.4.3 SW component's Global variables 16](#_Toc108012599)

[2.5 Design decisions and reasoning 17](#_Toc108012600)

[2.5.1 Alternatives 17](#_Toc108012601)

[2.6 Design solutions for general features 17](#_Toc108012602)

[2.6.1 Power management handling 17](#_Toc108012603)

[2.6.2 Diagnosis interaction 17](#_Toc108012604)

[2.6.3 Error Handling 17](#_Toc108012605)

[Major events 17](#_Toc108012606)

[2.7 Propagation delay – worst case computation 17](#_Toc108012607)

[2.8 SW Component integration 18](#_Toc108012608)

[2.8.1 Integration context / constraints 18](#_Toc108012609)

[2.8.2 Include files 18](#_Toc108012610)

[2.8.3 Initialization 18](#_Toc108012611)

[2.8.4 Interrupts 18](#_Toc108012612)

[2.8.5 Real time scheduler task 18](#_Toc108012613)

[2.8.6 Other entry points 18](#_Toc108012614)

[2.8.7 Power mode transition job 18](#_Toc108012615)

[2.8.8 SW Component configuration 19](#_Toc108012616)

[2.8.9 Interface mechanisms 19](#_Toc108012617)

[2.8.10 SW Platform or Basic SW configuration 20](#_Toc108012618)

Figure Index

[Figure 1 - External Interaction 11](#_Toc108014363)

[Figure 2 - Functional Decomposition 13](#_Toc108014364)

[Figure 3 - State Machine 14](#_Toc108014365)

[Figure 4 - Activity Diagram 15](#_Toc108014366)

[Figure 5 - Sequence Diagram 15](#_Toc108014367)

**Table Index**

[Table 1 - Project Reference Table 7](#_Toc109912478)

[Table 2 - Definitions and Abbreviations 8](#_Toc109912479)

[Table 3 - Document Change History 9](#_Toc109912480)

[Table 4 - Template Change History 9](#_Toc109912481)

[Table 5 - <Required Function1 Name> 12](#_Toc109912482)

[Table 6 - Required Events 12](#_Toc109912483)

[Table 7 - <Provided Function Name> 13](#_Toc109912484)

[Table 8 - Provided Events 13](#_Toc109912485)

[Table 9 - <Function name> 14](#_Toc109912486)

[Table 10 - EEPROM DATA 17](#_Toc109912487)

[Table 11 - Constants 17](#_Toc109912488)

[Table 12 - Global variables 17](#_Toc109912489)

[Table 13 - Shared variables description 18](#_Toc109912490)

[Table 14 - Error Handling 19](#_Toc109912491)

[Table 15 - Major Events 19](#_Toc109912492)

[Table 16 - Init Functions 20](#_Toc109912493)

[Table 17 - Scheduler Tasks 20](#_Toc109912494)

[Table 18 - Other Entry Tasks 20](#_Toc109912495)

[Table 19 - Power mode Transition services 20](#_Toc109912496)

[Table 20 - Component configurations 21](#_Toc109912497)

[Table 21 - List of Timers 22](#_Toc109912498)

[Table 22 - Controlled RAM 22](#_Toc109912499)

# Overview

## Purpose

This document describes the Software (SW) ComponentsSW DetailedDesign using the principles of Structured Design (SD).

The “Individual Circuit Status Reporting”

Each of the Module’s input will report their status via polling a constantly updating “Analog Data Manager.” The data manager updates al the stored analog values and reports them back to the respective “.c” file to be filtered and saved.

The Analog Data manager also handles the channel switching of the ADC to ensure that each channel is polled in a timely manner. It does this by moving a mux selection through the carousel of data counts.

## References

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

Table 1 - Project Reference Table

## Definitions

| **Abbreviation or Acronym** | **Description** |
| --- | --- |
| SW | Software |
| SD | Structured Design |
| R4J | Requirements for JIRA |
| RE-Tool | Requirements Engineering Tool (such as Doors, R4J…) |

Table 2 - Definitions and Abbreviations

**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 | Initial Version | Alex White | 23Jun23 |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Table 3 - Document Change History

**Template Change History**

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

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Version** | **Description** | **Author(s)** | **RevisionDate** | **Approved By (Title)** | **Approved By (Name)** | **Approved Date** |
| 0.1 | Initial Template Version | HassanArghavani | 27Apr2021 | N/A | N/A | N/A |
| 0.2 | Updated changes based on the meeting and the date format | Hassan Arghavani | 16Jun2021 | N/A | N/A | N/A |
| 0.3 | Changed References and Definitions to subchapters  ChangedRevision on the title page to a field that will update universally by following new instructions | Joshua Hartwig | 15Jul2021 |  |  |  |
| 0.4 | Updated Headings for more consistent spacing | Joshua Hartwig | 22Jul2021 |  |  |  |
| 0.5 | Update content for SW Detailed design template | Rahma Fkaier | 05Jul2022 |  |  |  |

Table 4 - Template Change History

## Realization constraints and targets

<Place the name of the software component in place of the “Desired Software Component Name”>

*The aim of this document is to describe the design of the <Desired software Component Name>.*

# Detailed SW Design

## Traceability matrix

<If covered by RE-tool, this matrix could be deleted>

|  |  |
| --- | --- |
| ID | Satisfying design element |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## External interaction

*<Add a block diagram to explain Software Component external interactions. This is an example of the 400w Inverter ON/OFF control component.>*

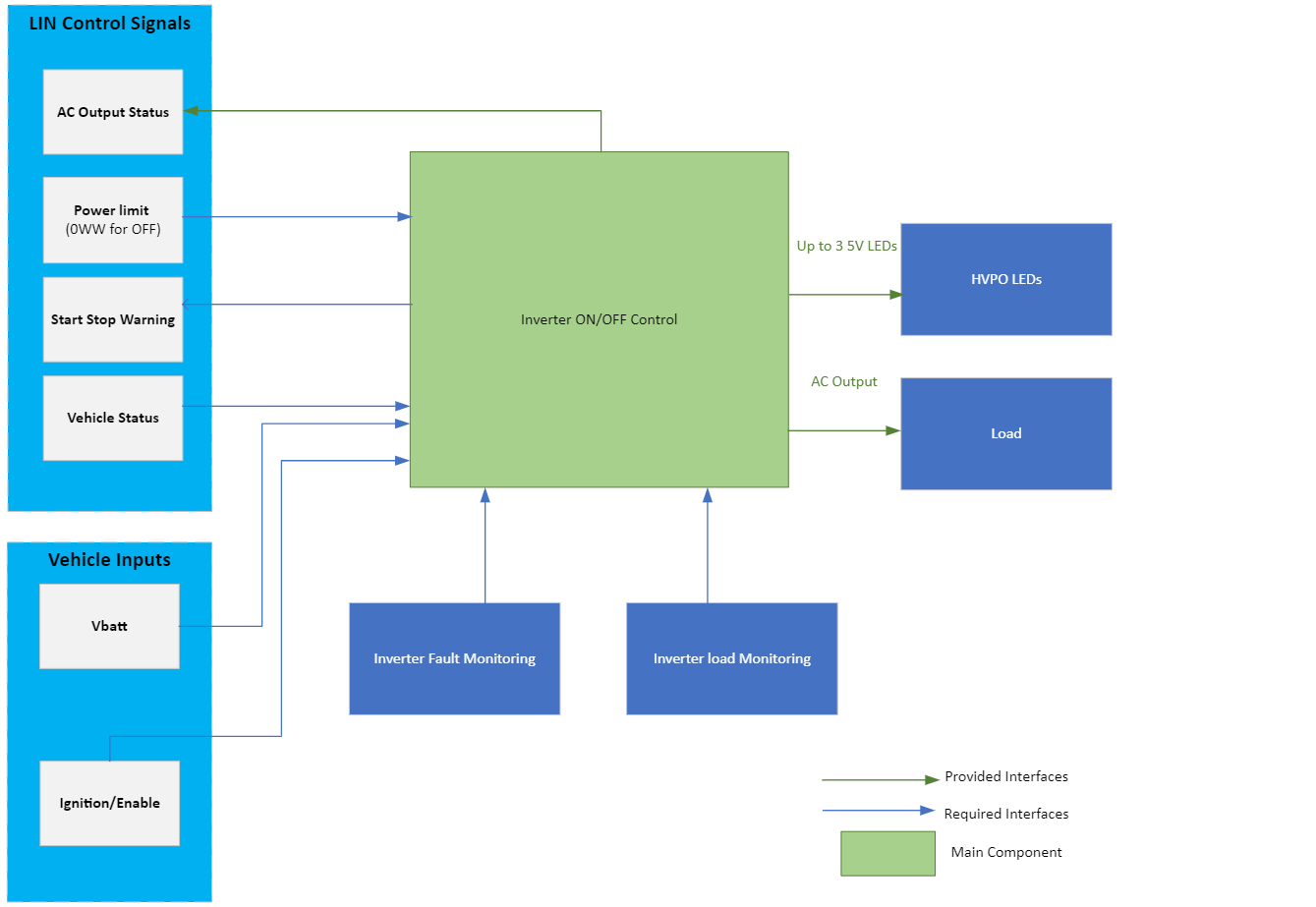


Figure 1 - External Interaction

### 2.2.1 Required interfaces

*<Detail the required interfaces of your software component provided by other external software components >*

#### Required services

*<In this section, capture all external functions, one at a time. Create more tables if there are more functions to capture>*

*WDT\_HAL.c,Tsk\_Feed\_WDT*

|  |  |
| --- | --- |
| **Description** | Clears the watchdog timer |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | Watchdog timer is cleared |

Table 5 – Tsk\_Feed\_WDT

*VehicleModeManager.c, Tsk\_VehicleModeManager*

|  |  |
| --- | --- |
| **Description** | Manages the vehicle mode through a set of if and else statements to determine what to do |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | Vehicle mode should be set |

Table 6 – Tsk\_VehicleModeManager

*Analog\_Data\_Manager.c, Tsk\_Refresh\_Latest\_Analog\_Data*

|  |  |
| --- | --- |
| **Description** | Refreshes the latest analog data by checking if running then if it is updates the values |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | Latest analog data should be refreshed |

Table 7 – Tsk\_Refresh\_Latest\_Analog\_Data

*Ref2p5\_Sensor.c, Tsk\_Ref2p5\_Sensor\_Filter*

|  |  |
| --- | --- |
| **Description** | Passes the analog signal result through the filter |
| **Parameter 1**  [in] | Void |
| **Return Value** | Void |
| **Post condition** | Data should be passed through the filter |

Table 5 – Tsk\_Ref2p5\_Sensor\_Filter

*SubSystemManager.c,Tsk\_SubSystemManager*

|  |  |
| --- | --- |
| **Description** | Commands the sub system to go awake or sleep depending on if else logic |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | Sub system should be given a command |

Table 5 – Tsk\_SubSystemManager

*SubSystemDriver.c, Tsk\_SubSystemDriver*

|  |  |
| --- | --- |
| **Description** | Wakes up or puts the module to sleep depending on its current state |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | Module should be woken or sleep |

Table 5 – Tsk\_SubSystemDriver

*PowerPath\_Interface.c, Tsk\_PowerPath\_Manager*

|  |  |
| --- | --- |
| **Description** | Checks the state of a fault, the vehicle mode, and trailer presense, then decides if power needs to be supplied based on the state of these 3 |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | Power should be supplied or not based on state of components |

Table 5 – Tsk\_PowerPath\_Manager

*PowerPathSequenceManager.c, Tsk\_PowerPath\_Sequence\_Manager*

|  |  |
| --- | --- |
| **Description** | Controls the power path based on the states of other components |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | Power path should be controlled |

Table 5 – Tsk\_PowerPath\_Sequence\_Manager

*PowerPathDriverDispatcher, Tsk\_PowerPathDriverDispatcher*

|  |  |
| --- | --- |
| **Description** | Dispatches driver commands to the different drivers |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | Driver commands should be dispatched |

Table 5 – Tsk\_PowerPathDriverDispatcher

*Battery\_Driver.c, Tsk\_AuxSwitched5V\_Driver*

|  |  |
| --- | --- |
| **Description** | Enables or disables the battery pin based on the state of the driver |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | Battery pins should be enabled or disabled |

Table 5 – Tsk\_AuxSwitched5V\_Driver

*Qswitch\_Driver.c, Tsk\_Qswitch\_Driver*

|  |  |
| --- | --- |
| **Description** | Disables enables or changes the state of the qswitch driver based on the state of other components |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | State of the q switch driver should change |

Table 5 – Tsk\_Qswitch\_Driver

*Boost\_Driver.c, Tsk\_Boost\_Driver*

|  |  |
| --- | --- |
| **Description** | Enables, disables, or resets the boost pins based on the command recieved |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | Boost driver pins will be changed |

Table 5 – Tsk\_Booster\_Driver

*Battery\_Sensor.c, Tsk\_Battery\_Sensor\_Filter*

|  |  |
| --- | --- |
| **Description** | Passes the Battery sensor result through the filter |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | Data should be passed through the filter |

Table 5 – Tsk\_Battery\_Sensor\_Filter

*Qswitch\_Sensor.c, Tsk\_Qswitch\_Sensor\_Filter*

|  |  |
| --- | --- |
| **Description** | Passes the Qswitch sensor result through the filter |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | Data should be passed through the filter |

Table 5 – Tsk\_Qswitch\_Sensor\_Filter

*Temperature\_Sensor.c, Tsk\_Temperature\_Sensor\_Filter*

|  |  |
| --- | --- |
| **Description** | Passes the temperature sensor result through the filter |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | Data should be passed through the filter |

Table 5 – Tsk\_Temperature\_Sensor\_Filter

*Battery\_State\_Monitor.c, Tsk\_Battery\_State\_Monitor*

|  |  |
| --- | --- |
| **Description** | Checks and stores the states of the battery |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | Variables should be saved that contain the states of the battery |

Table 5 – Tsk\_Battery\_State\_Monitor

*Aux5V\_State\_Monitor.c, Tsk\_Aux5V\_State\_Monitor*

|  |  |
| --- | --- |
| **Description** | Checks and stores the states of the Aux5V |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | Variables should be saved that contain the states of the Aux5V |

Table 5 – Tsk\_Aux5V\_State\_Monitor

*Qswitch\_State\_Monitor.c, Tsk\_Qswitch\_State\_Monitor*

|  |  |
| --- | --- |
| **Description** | Checks and stores the states of the Qswitch |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | Variables should be saved that contain the states of the Qswitch |

Table 5 – Tsk\_Qswitch\_State\_Monitor

*Boost\_State\_Monitor.c, Tsk\_Boost\_State\_Monitor*

|  |  |
| --- | --- |
| **Description** | Checks and stores the states of the boost |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | Variables should be saved that contain the states of the boost |

Table 5 – Tsk\_Boost\_State\_Monitor

*PowerPath\_State\_Monitor.c, Tsk\_PowerPath\_State\_Monitor*

|  |  |
| --- | --- |
| **Description** | Checks and stores the states of the PowerPath |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | Variables should be saved that contain the states of the PowerPath |

Table 5 – Tsk\_PowerPath\_State\_Monitor

*Light\_Control\_Interface.c, Tsk\_LightControl\_Manager*

|  |  |
| --- | --- |
| **Description** | Sets the light control command based on the state of the other components |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | Command for light control should be configured |

Table 5 – Tsk\_LightControl\_Manager

*Light\_Control.c, Tsk\_Light\_Control*

|  |  |
| --- | --- |
| **Description** | Gets conditions for the light control and then sets the rules for the turn trailer and vehicle lights |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | Rules for the lights should be set |

Table 5 – Tsk\_Aux5V\_State\_Monitor

*Light\_Output\_Driver.c, Tsk\_Light\_Output\_Driver*

|  |  |
| --- | --- |
| **Description** | Updates current light output states, updates the commands, and the drives the light outputs |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | Commands and output states should be updated and the light outputs should be driven |

Table 5 – Tsk\_Light\_Output\_Driver

*Light\_Sense\_Monitor.c, Tsk\_Light\_Sense\_Monitor*

|  |  |
| --- | --- |
| **Description** | Updates the light sense data |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | Light sense data should be updated |

Table 5 – Tsk\_Light\_Sense\_Monitor

*Light\_Load\_Monitor.c, Tsk\_Light\_Load\_Monitor*

|  |  |
| --- | --- |
| **Description** | Reads light output data then the states and loads the states into the data table |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | Data should be loaded into the table |

Table 5 – Tsk\_Light\_Load\_Monitor

*Light\_Request\_Handler.c, Tsk\_Light\_Request\_Monitor*

|  |  |
| --- | --- |
| **Description** | Decodes all light request signals |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | Light requests should be decoded |

Table 5 – Tsk\_Light\_Request\_Monitor

*Light\_Activity\_Monitor.c, Tsk\_Light\_Activity\_Monitor*

|  |  |
| --- | --- |
| **Description** | Figures out the current states of the light by figuring out its activity |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | Current light activity will be known |

Table 5 – Tsk\_Light\_Activity\_Monitor

*VehicleRequestMonitor.c, Tsk\_VehicleRequest\_Monitor*

|  |  |
| --- | --- |
| **Description** | Sets the vehicle request state based on the last light activity |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | Vehicle request state will be set |

Table 5 – Tsk\_VehicleRequest\_Monitor

*TurnLight\_Activity\_Monitor.c, Tsk\_TurnLight\_Activity\_Monitor*

|  |  |
| --- | --- |
| **Description** | Figures out the state of the left and right turn signal |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | Turn signals states will be known |

Table 5 – Tsk\_TurnLight\_Activity\_Monitor

*Trailer\_Present\_Monitor.c, Tsk\_Trailer\_Present\_Monitor*

|  |  |
| --- | --- |
| **Description** | Checks if the trailer is present and sets a value based on feedback |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | Value will be set that states what state the trailer is in |

Table 5 – Tsk\_Trailer\_Present\_Monitor

*Light\_Short\_Circuit\_Monitor.c, Tsk\_Light\_Short\_Circuit\_Monitor*

|  |  |
| --- | --- |
| **Description** | Checks for a short circuit and then updates data table for monitor |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | Short circuit data table should be updated |

Table 5 – Tsk\_Light\_Short\_Circuit\_Monitor

*Light\_Retry\_ability\_Manager.c, Tsk\_Light\_Retry\_Ability\_Monitor*

|  |  |
| --- | --- |
| **Description** | Attempts to restart/retry the lights |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | Lights may be reset |

Table 5 – Tsk\_Light\_Retry\_Ability\_Monitor

*VehicleModeMonitor.c, Tsk\_VehicleModeMonitor*

|  |  |
| --- | --- |
| **Description** | Figures out the state the vehicle is in |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | Vehicle state will be returned |

Table 5 – Tsk\_VehicleModeMonitor

#### Required Events

*<In this section, capture all external events, one at a time. Insert new line if there are more events to capture>*

|  |  |  |
| --- | --- | --- |
| **Events** | | |
| Name | Source | Connection type |
| *<Event Name>* | *<Sender>* | *<sender-receiver>* |

Table 6 - Required Events

### 2.2.2 Provided Interfaces

*<Detail the provided interfaces by your software component to external software components >*

#### Provided Services

*<In this section, capture all provided functions, one at a time. Create more tables if there are more functions to capture>*

*Module\_Mode\_Manager.c, Change\_COS\_Tasklist\_To\_VehicleMode\_TskList*

|  |  |
| --- | --- |
| **Description** | Changes cos task list to vehicle mode task list |
| **Parameter 1**  [in] | void |
| **Return Value** | void |
| **Post condition** | Task list should be changed to vehicle mode task list |

Table 5 – Change\_COS\_Tasklist\_To\_VehicleMode\_TskList

#### Provided Events

*<In this section, capture all external events, one at a time. Insert new line if there are more events to capture>*

|  |  |  |
| --- | --- | --- |
| **Event** | | |
| Name | Receiver | Notification function |
| *<Event Name>* | *<Receiver>* | *<Condition to send event >* |

Table 8 - Provided Events

## SW Component internal breakdown

*<Add description of the software component main features>*

#### Functional Decomposition

*<Overview of functions and their dependencies shown by a Static Function Tree>*

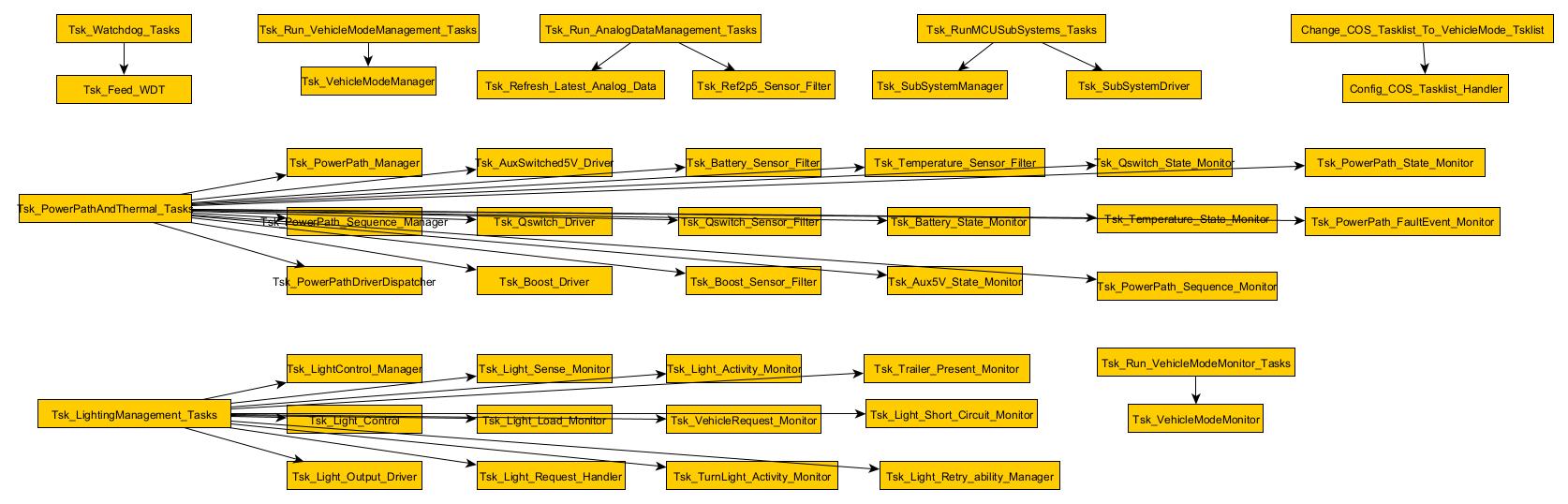


Figure 2- Functional Decomposition

#### Function Description and Dynamic Behavior

*<In this section, capture all static functions, one at a time. Create more tables if there are more functions to capture>*

*Tsk\_Watchdog\_Tasks*

|  |  |
| --- | --- |
| **Description** | Calls function Tsk\_Feed\_WDT which feeds the watchdog timer |
| **Parameter 1** | void |
| **Return Value** | void |
| **Precondition** | none |
| **Post condition** | The watchdog timer will be fed |
| **Error Conditions** | None |

Table 9 – Tsk\_Watchdog\_Tasks

HVPO\_SWDD\_VehicleMode\_0001

*Tsk\_Run\_VehicleModeManagement\_Tasks*

|  |  |
| --- | --- |
| **Description** | Runs the vehicle mode management tasks |
| **Parameter 1** | void |
| **Return Value** | void |
| **Precondition** | none |
| **Post condition** | Vehicle mode management tasks will be ran |
| **Error Conditions** | None |

Table 9 – Tsk\_Run\_VehicleModeManagement\_Tasks

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

HVPO\_SWDD\_VehicleMode\_0002

*Tsk\_Run\_AnalogDataManagement\_Tasks*

|  |  |
| --- | --- |
| **Description** | Runs the analog data management tasks |
| **Parameter 1** | void |
| **Return Value** | void |
| **Precondition** | none |
| **Post condition** | Analog data management tasks will be ran |
| **Error Conditions** | None |

Table 9 – Tsk\_Run\_AnalogDataManagement\_Tasks

Ref{PFE-49 }

HVPO\_SWDD\_VehicleMode\_0003

*Tsk\_RunMCUSubSystems\_Tasks*

|  |  |
| --- | --- |
| **Description** | Runs the Sub system tasks manager and driver |
| **Parameter 1** | void |
| **Return Value** | void |
| **Precondition** | none |
| **Post condition** | Sub system manager and driver tasks run |
| **Error Conditions** | None |

Table 9 – Tsk\_RunMCUSubSystems\_Tasks

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

*Tsk\_PowerPathAndThermal\_Tasks*

|  |  |
| --- | --- |
| **Description** | Calls the tasks for state monitors sensor filters manager drivers related to powerpath and thermal |
| **Parameter 1** | void |
| **Return Value** | void |
| **Precondition** | none |
| **Post condition** | Functions for tasks will be called and their tasks should be carried out |
| **Error Conditions** | None |

Table 9 – Tsk\_PowerPathAndThermal\_Tasks

*Tsk\_LightingManagement\_Tasks*

|  |  |
| --- | --- |
| **Description** | Calls the functions for tasks relating to lighting, including managers monitors handlers and controls |
| **Parameter 1** | void |
| **Return Value** | void |
| **Precondition** | none |
| **Post condition** | Functions should be called that do tasks related to lighting |
| **Error Conditions** | None |

Table 9 – Tsk\_LightingManagement\_Tasks

*Tsk\_Run\_VehicleModeMonitor\_Tasks*

|  |  |
| --- | --- |
| **Description** | Calls the Tsk\_VehicleModeMonitor function |
| **Parameter 1** | void |
| **Return Value** | void |
| **Precondition** | none |
| **Post condition** | Tsk\_VehicleModeMonitor should be ran |
| **Error Conditions** | None |

Table 9 – Tsk\_Run\_VehicleModeMonitor\_Tasks

*<Add a Function Activity diagrama sequence diagram or a state Machine diagram according to the need to better explain the function’s algorithm >*

#### Macros

*None*

## General Data description

*<Description of data which is produced and used by this SW Component  
Module global Variables – Data which exists globally, but only for use inside this SW Module   
Incl. Variables, Constants, for local use*  
***Global shared Data are described in the required or provided SW interface depending to the owner of a variable****.>*

### 2.4.1 EEPROM DATA

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **EEPROM DATA blocks** | | | | | | | | | |
| **Data Block Name** | **Size [Byte]** | **RAM mirror**  **[Yes/ No]** | **Write cycles** | **Check-**  **sum**  **[YES/NO]** | **No. of Multi-blocks** | **Variable / Structure name** | **Definition location  [Filename]** | **Type** | **Length** |
| (…) | (…) | (…) | (…) | (…) | (…) | (…) | (…) | (…) | (…) |

Table 10- EEPROM DATA

### 2.4.2 Constants

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Data Type** | **DataRange** | **Default Value** | **Description** |
| (…) | (…) | (…) | (…) | (…) |

Table 11- Constants

### 2.4.3 SW component's Global variables

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Memory section** | **Description** | **Shared** |
|
| <Inverter\_OnOff\_Command> | <e\_Inverter\_OnOff\_Command> | <RAM> | <Inverter ON/OFF command  INVERTER\_ONOFF\_COMMAND\_OFF=0  INVERTER\_ONOFF\_COMMAND\_ON=1> | <Check shared Variable’s description> |

Table 12- Global variables

<Shared Variables description>

|  |  |  |
| --- | --- | --- |
| **Update mechanism** | **Meaning** | **Special attention** |
| 0 | Background task level (only available in high power) | May have high or low priority! |
| 1 | Task level M  (Only available in high power, for SW-platform M is equal to FG1) | Check which task priorities are defined in this project |
| 2 | Task level N  (Only available in high power, for SW-platform N is equal to FG2) | Check which task priorities are defined in this project |
| L | Used in low power |  |
| H | Used in high power |  |
| A | Service provided to other SW-components | An external service might be in use everywhere and on any priority level |
| P | Service provided to other SW-subcomponents which belong to this SW-component design | An external service might be in use everywhere and on any priority level |
| S | Initialized at Init task (IT is allowed) |  |
| D | Init in startup hook (IT are disabled) |  |
| A | Atomic access (not interruptible, not good for reusability) | Depending to the µC and compiler! |
| P | Protected from interruption or/and tasks preemption |  |
| NP | Protection not needed |  |

Table 13 - Shared variables description

## 2.5 Design decisions and reasoning

### 2.5.1 Alternatives

<Brief description of possible alternatives and why a specific design solution was chosen.   
Possible improvements>

## Design solutions for general features

<This chapter is optional.  
It should describe detailed design solutions for general features of our products.>

### 2.6.1 Power management handling

*<Details which are not provided by the SW Architecture description>*

### 2.6.2 Diagnosis interaction

### 2.6.3 Error Handling

|  |  |
| --- | --- |
| **Error description** | **Reaction** |
| (…) | (…) |

Table 14 - Error Handling

### Major events

|  |  |
| --- | --- |
| **Major events** | **Reaction** |
| Cold reset |  |
| Warm reset |  |
| Sleep/wake-up |  |
| Ignition ON/OFF |  |
| Cranking |  |
| Over/Under voltage |  |
| Shorting pin |  |
| Communication disconnection |  |
| Cold reset |  |
| Warm reset |  |
| Sleep/wake-up |  |
| Ignition ON/OFF |  |

Table 15- Major Events

## Propagation delay – worst case computation

*<Performance data about the SW component  
List functions which are called in a task and provide following information:  
Worst case execution time, with short description of the worst-case scenario, and how the value has been deduced (measured/estimated/calculated etc.)  
Max. time spent in protected section, i.e. between SuspendAllInterrupts() and ResumeAllInterrupts() or EnterProtectedSection() and LeaveProtectedSection().   
Max. Stack consumption (measured/estimated/calculated) Integration Manual>*

## SW Component integration

### 2.8.1 Integration context / constraints

### 2.8.2 Include files

*<List of included files>*

#include"VehicleMode.h"

#include"COS\_TaskList\_Handler.h"

#include"WDT\_HAL.h"

#include"Analog\_Data\_Manager.h"

#include"Battery\_Driver.h"

#include"Battery\_Driver\_Interface.h"

#include"Battery\_Sensor.h"

#include"Qswitch\_Sensor.h"

#include"Boost\_Sensor.h"

#include"Ref2p5\_Sensor.h"

#include"Temperature\_Sensor.h"

#include"Qswitch\_Driver.h"

#include"Qswitch\_Driver\_Interface.h"

#include"Boost\_Driver.h"

#include"Boost\_Driver\_Interface.h"

#include"Light\_Activity\_Monitor.h"

#include"Light\_Control.h"

#include"Light\_Control\_Interface.h"

#include"Light\_Request\_Handler.h"

#include"Light\_Retry\_ability\_Manager.h"

#include"Light\_Short\_Circuit\_Monitor.h"

#include"Trailer\_Present\_Monitor.h"

#include"TurnLight\_Activity\_Monitor.h"

#include"Light\_Output\_Driver.h"

#include"Light\_Sense\_Monitor.h"

#include"VehicleRequestMonitor.h"

#include"Wakeup\_Event\_Handler.h"

#include"SubSystemDriver.h"

#include"SubSystemManager.h"

#include"VehicleModeManager.h"

#include"VehicleModeMonitor.h"

#include"PowerPath\_FaultEvent\_Monitor.h"

#include"PowerPath\_Interface.h"

#include"PowerPathDriverDispatcher.h"

#include"PowerPathMonitor.h"

#include"PowerPathSequenceManager.h"

#include"PowerPathSequenceMonitor.h"

#include"Boost\_State\_Monitor.h"

#include"Battery\_State\_Monitor.h"

#include"Aux5V\_State\_Monitor.h"

#include"Qswitch\_State\_Monitor.h"

#include"Light\_Load\_Monitor.h"

#include"Temperature\_State\_Monitor.h"

#include"SFR\_Refresh.h"

### 2.8.3 Initialization

*<All function bellow shall be called on initialization>*

|  |  |  |
| --- | --- | --- |
| Call Order | Service | Call constraints linked to another module |
|  |  |  |

Table 16 - Init Functions

### 2.8.4 Interrupts

*<If described in the SW Architecture Document, just put the reference here>*

### 2.8.5 Real time scheduler task

*<Add scheduler tasks>*

|  |  |  |
| --- | --- | --- |
| **Name** | **Service** | **Task level** |
|  | *<Function name>* | *<called by Task name>* |

Table 17- Scheduler Tasks

### 2.8.6 Other entry points

|  |  |  |
| --- | --- | --- |
| **Name** | **Service** | **Task level** |
|  | *<Function name>* | *<called by Task name>* |

Table 18 - Other Entry Tasks

### 2.8.7 Power mode transition job

|  |  |  |  |
| --- | --- | --- | --- |
| Call order | Transition | Service | Call constraints linked to another module |
|  | *<Power transitionMode>* | *<Function name>* | *<called by Task name>* |

Table 19 - Power mode Transition services

### 2.8.8 SW Component configuration

*<Default Value = Value which is used when EEPROM is corrupt or after init when EEPROM is empty*

|  |  |
| --- | --- |
| *Mem Type* | *Meaning* |
| *Stat ROM* | *Configuration item which can be changed statically, i.e. re-building the software* |
| *Stat. Flash ROM* | *Configuration item which can be changed statically, i.e. flashing ROM partially* |
| *Dyn* | *Configuration item which can be changed dynamically, e.g. by diagnosis* |

*>*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Mem Type** | **Data Type** | **Data Range** | **Default Value** | **Description** |
| (…) | (…) | (…) | (…) | (…) | (…) |

Table 20- Component configurations

### 2.8.9 Interface mechanisms

*See chapter*

### 2.8.10 SW Platform or Basic SW configuration

*<This tables list all resources needed and provided by the SW component in its environment.   
  
If the SWP is used, these data will be entered in the XML file that describes this VFB component, using the configuration tool provided by the SW Platform.>*

|  |  |  |
| --- | --- | --- |
| **TIMER** | | |
| **Fast Timer** | | |
| Name | Notification Function | |
| *<Timer Name>* | *<Callback of the Timer when it’s elapsed>* | |
| **Slow Timer** | | |
| Name | Resolution | Notification Function |
| *<Timer Name>* | *<Timer resolution>* | *<Callback of the Timer when it’s elapsed>* |

Table 21 - List of Timers

|  |  |
| --- | --- |
| **CONTROLLED RAM** | |
| Largest Data Byte Size | Initialization Callback Function |
| *<Data size>* | *<Callback Function Name>* |

Table 22- Controlled RAM