Software Architectural Design (SWAD)

|  |  |  |
| --- | --- | --- |
| **OP’nSoft Project information** | | |
| **Project ID** | **Project Name** | **Project Manager** |
| [Type Project ID] | [Type Project Name] | [Type Project Manager] |
| **Field of application:** [Type field of application] | | |
| **Customer Name** | **Project Start Date** | **OP’nSoft project ID** |
| [Type Customer Name ] | [Type Project Start Date ] | [Type OP’nSoft project ID] |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Name | Function | Signature |
| Edited by |  |  |  |
| Reviewed by |  |  |  |
| Reviewed by |  |  |  |
| Reviewed by |  |  |  |
| Approved by |  |  |  |

# Template Revision History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Version | Author | Section | Description / Task ID |
|  |  |  |  |  |
|  |  |  |  |  |

# Document Revision History

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Date  **(yyyy-dd-mm)** | Version  (x.y) | Status | Author | Section | Description / Task ID |
|  |  | Choose an item. |  |  |  |
|  |  | Choose an item. |  |  |  |
|  |  | Choose an item. |  |  |  |
|  |  | Choose an item. |  |  |  |

# Table of Contents

[Template Revision History 2](#_Toc148021961)

[Document Revision History 2](#_Toc148021962)

[Table of Contents 3](#_Toc148021963)

[1 Introduction 5](#_Toc148021964)

[1.1 Purpose and Scope 5](#_Toc148021965)

[1.2 Review and Approval 5](#_Toc148021966)

[1.3 Referenced documents 5](#_Toc148021967)

[1.4 Applicable documents 6](#_Toc148021968)

[1.5 Glossary & Abbreviations 6](#_Toc148021969)

[2 Architecture definition strategy 8](#_Toc148021970)

[2.1 Tailoring 8](#_Toc148021971)

[2.2 Naming convention 8](#_Toc148021972)

[2.3 List of available software components at OP’nSoft 8](#_Toc148021973)

[2.4 Resource consumption criteria 8](#_Toc148021974)

[2.5 Architecture design evaluation 8](#_Toc148021975)

[2.5.1 Evaluation Criteria 8](#_Toc148021976)

[2.5.2 Software Architecture evaluation method 9](#_Toc148021977)

[2.6 Standard software architecture requirements 10](#_Toc148021978)

[2.7 Requirements allocation 10](#_Toc148021979)

[3 Product overview 11](#_Toc148021980)

[3.1 Global software use case 11](#_Toc148021981)

[3.2 Global software view 11](#_Toc148021982)

[3.3 ECU / hardware interfaces 11](#_Toc148021983)

[4 Software static view 13](#_Toc148021984)

[4.1 Software Environment 13](#_Toc148021985)

[4.2 Basic Software 13](#_Toc148021986)

[4.2.1 Illustration 13](#_Toc148021987)

[4.2.2 Basic Software Component description 14](#_Toc148021988)

[4.2.3 Basic software interfaces 15](#_Toc148021989)

[4.3 Applicative software 16](#_Toc148021990)

[4.3.1 Illustration 16](#_Toc148021991)

[4.3.2 Software components description 16](#_Toc148021992)

[4.3.3 Applicative software internal interfaces 17](#_Toc148021993)

[5 Software dynamic view 18](#_Toc148021994)

[5.1 States 18](#_Toc148021995)

[5.1.1 Illustration 18](#_Toc148021996)

[5.1.2 Description 18](#_Toc148021997)

[5.2 Wake up & sleep sequences 19](#_Toc148021998)

[5.2.1 Wake up sequences 19](#_Toc148021999)

[5.2.1 Sleep sequences 19](#_Toc148022000)

[5.3 Use cases 20](#_Toc148022001)

[5.3.1 Illustrations 20](#_Toc148022002)

# Introduction

## Purpose and Scope

**Purpose:** The purpose of the Software Architectural Design (SWAD) is to describe the Software architecture of the project[Type Project Name]**.**

The Software Architectural Design contains project specific information and is part of the OP’nSoft software architectural design process. Hence, this document cannot be seen independently because further information, valid for all projects, is not part of this plan, but part of the process.

**Scope**: This Software Architectural design is valid for the project [Type Project Name]and affects the Software Development activities and their supplementary work products. It aims to:

* Describe the software architecture
* Provide a high-level software design
* Provide a high-level interface specification

Software design specification and interface specification are completed by the software detailed design.

## Review and Approval

The Software Architectural Design is intended for all project members involved in the development and should be modified/adapted according to the project organization needs.

After the initial creation or any update afterwards, the document needs to be approved by all the following parties below:

* Software Product Owner
* Software Architect
* Software Integration Engineer

## Referenced documents

<In the table below, list the all the technical documents, norms, standards, etc., that were used to define this document>

|  |  |  |  |
| --- | --- | --- | --- |
| **Document name** | **Version** | **Date** | **Storage path and link** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

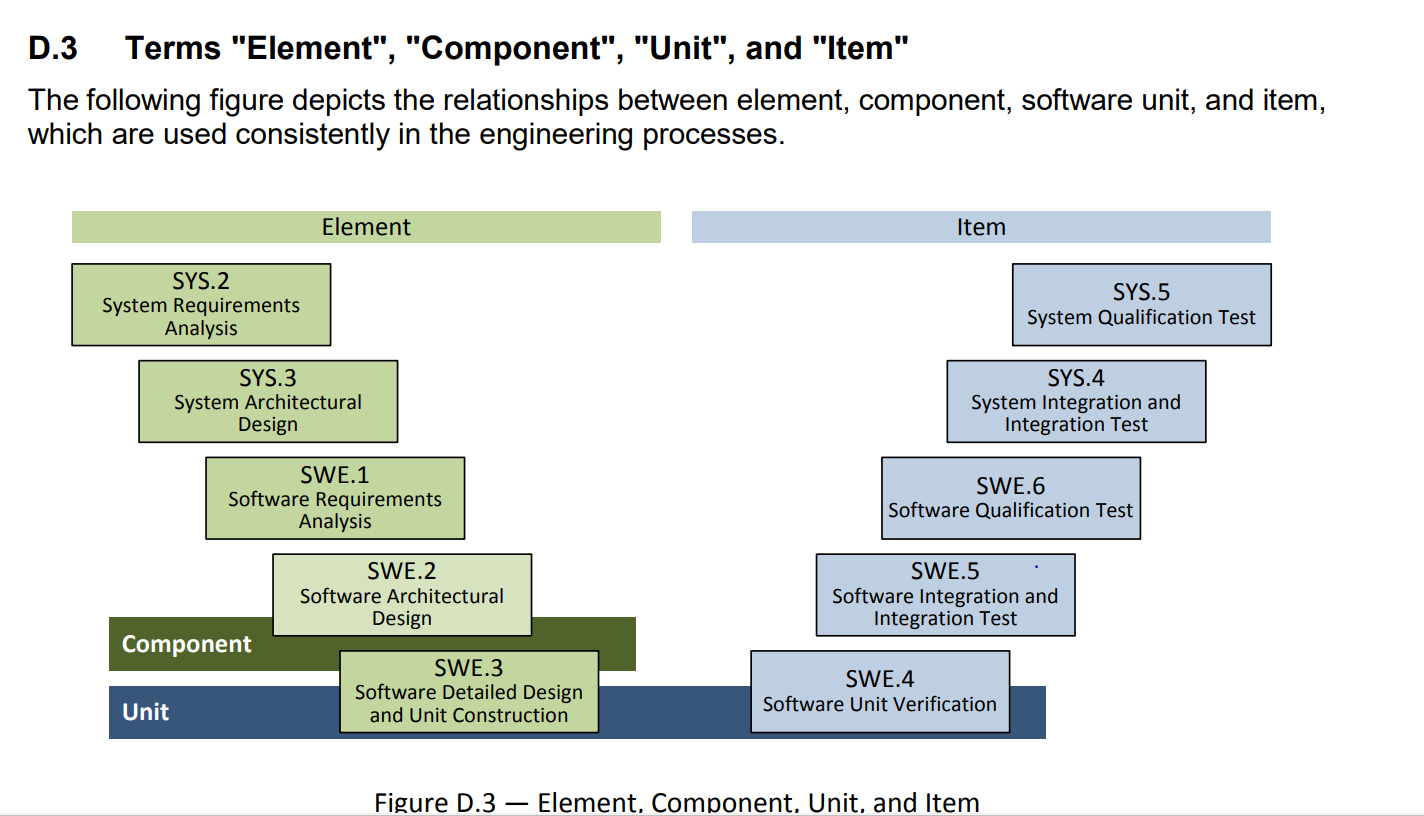
## Applicable documents

<In the table below, list the all the project specific documents that are cited in this document or that were used to define this document>

|  |  |  |  |
| --- | --- | --- | --- |
| **Document name** | **Version** | **Date** | **Storage path and link** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Glossary & Abbreviations

|  |  |
| --- | --- |
| **Term** | **Definition** |
| **Elements** | Elements are all structural objects on architectural and design level on the left side of the "V". Such elements can be further decomposed into more fine-grained sub-elements of the architecture or design across appropriate hierarchical levels |
| **System elements** | A member of a set of elements that constitutes a system. A system element is a discrete part of a system that can be implemented to fulfill specified requirements. A system element can be hardware, software, data, humans, processes (e.g., processes for providing service to users), procedures (e.g., operator instructions), facilities, materials, and naturally occurring entities (e.g., water, organisms, minerals), or any combination. (ISO/IEC 15288:2015) |
| **Software Element** | Result of the decomposition of the architecture on software level: The software is decomposed into elements of the software architecture across appropriate hierarchical levels down to the software components (the lowest level elements of the software architecture). |
| **Software Component** | Software-Components (SW-C) are architectural elements that provide and/or require interfaces and are connected to each other through the Virtual Functional Bus to fulfill architectural responsibilities. The Software Component is the central structural element used when building a system at the VFB-level. A SW-C has well-defined ports, through which the component can interact with other Software Components.  A Software Component has a formal description defined by the Software Component template. Software Components can be abbreviated as SW-Cs.  A SW-C contains a Software Component Description (SW-CD) and the implementation.  In Automotive SPICE V3.0 the term "software component" is used for the lowest level elements of the software architecture for which finally the detailed design is defined. A software "component" consists of one or more software "units". |
| **Unit** | Part of a software component which is not further subdivided. |



<In the table below, list and describe the abbreviations that are used in the document>

|  |  |
| --- | --- |
| **Abbreviation** | **Description** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Architecture definition strategy

## Tailoring

Sections 3, 4 and 5 of this document can be tailored according to the project needs. Some view may be added or deleted as long as all Software elements, interfaces and dynamic behavior are defined an described with enough details.

## Naming convention

Once current naming convention is tested and validated, it will be integrated in this section.

Naming convention should help identify hierarchical level of the item.

## List of available software components at OP’nSoft

Insert link to OP’nSoft platform catalog

## Resource consumption criteria

Tailor the following table with project specific criteria

|  |  |
| --- | --- |
| **Resource type** | **Criteria** |
| **CPU** |  |
| **RAM** |  |
| **ROM** |  |
| **External EEPROM** |  |
| **Internal EEPROM** |  |
| **Data Flash** |  |
| **…** |  |

## Architecture design evaluation

This evaluation describes how the chosen software architecture has been evaluated and justified based on alternative software architecture approaches and/or evolutions of software architecture designs over several improvement iterations.

### Evaluation Criteria

Tailor the following table with project specific criteria

|  |  |  |  |
| --- | --- | --- | --- |
| **Technical requirements** | **Maintainability requirements** | **Organizational requirements** | **Business requirements** |
| * Performance (response time, cycle time, flow, deadline, resource use, etc.), * Functional Safety, * Cybersecurity, * COTS (Commercial off the shelf) elements with defined interfaces, * Stability, * Efficiency, * etc. | * Usability, * Simplicity, * Maximum cohesion and minimum coupling * Reusability, * Testability, * Modifiability, * Upgradability, * Modularity, * Encapsulation needs, * etc. | * Responsibilities and work distribution within the organization, * Parallelization of development activities, * Collaboration with external partners, * Integration of third-party elements, * etc. | * Costs, * Portability (re-use, platform, modular kit, legacy interfaces, etc.), * Scalability, * etc. |

### Software Architecture evaluation method

It has to be ensured that all relevant parties and all necessary competences are involved in the agreement on the selection of the final software architecture. Any weaknesses with respect to the selected evaluation criteria shall either be eliminated or justified.

One of the 3 sections below have to be selected and elaborated for the project, the 2 other sections have to be removed

1. **Development of alternative solutions**

If the project is about a completely new product development, a few potential solutions for the software architecture shall be described at least to an abstraction level that allows the identification of the main differences between the architecture solutions, and the evaluation of the criteria for each architecture solution. It has to be argued that the solutions chosen for further elaboration (to result in the chosen architecture) shall cover the project needs.

The proposed and refined solutions are evaluated against the evaluation criteria selected in section 2.5.1, and a decision shall be made and argued:

* Selection/confirmation of one/the proposed solution as the chosen architecture for further development or
* Rejection of the previously proposed solution(s) and step back to architecture development.

1. **Iterative Architecture Development of Alternative Solutions**

Several solution variants come up during the development of a software architecture (potentially also in the context of other comparable projects other than this one). A variant can be a completely different architecture or differ from an already identified solution only in a few aspects or viewpoints. Consequently, the evaluation against the selected criteria in section 2.5.1 can take place several times during development. Such iterative steps shall be documented here.

1. **Carry Over and Adaptation of an Existing Architecture**

This approach is particularly relevant for platform development projects. Although only one solution approach is used for several projects, it has to be assured that this approach is suitable for this specific project and valid according to the chosen evaluation criteria from section 2.5.1.

In any case, it has to be ensured that all relevant parties and all necessary competences are involved in the agreement on the selection of the final software architecture. Any weaknesses with respect to the selected evaluation criteria shall either be eliminated or justified.

## Standard software architecture requirements

Standard requirements for software architecture are defined in (TBD)

## Requirements allocation

Define here where the components requirements and interface requirements are stored, so that they be reviewed and accessed with this document.

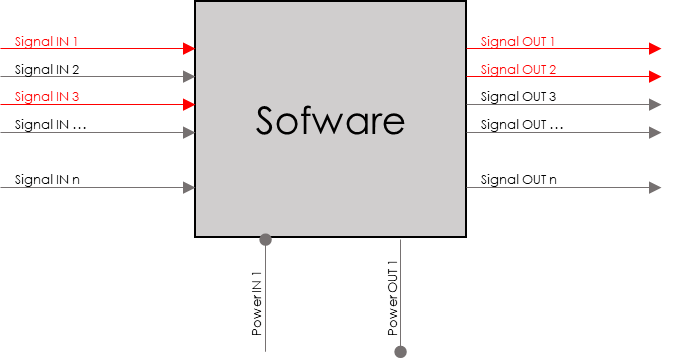
# Product overview

## Global software use case

In this section, adapt and describe the global use case made by the system to the software.

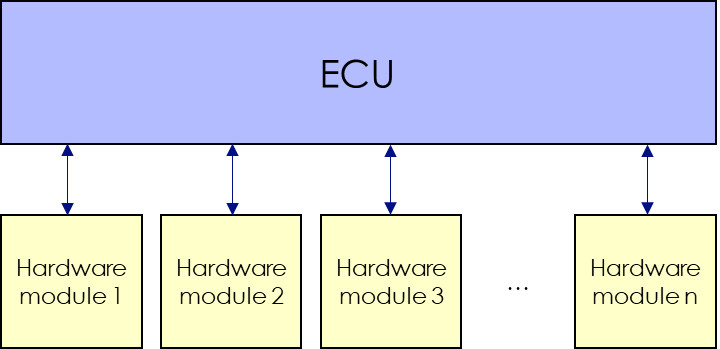
## Global software view

In this section, list all the signal inputs and outputs of the software:



## ECU / hardware interfaces

In this section, list all the hardware modules that will interface with the ECU as shown below:



In the table below, list the interfaces between ECU and hardware or insert link to system’s document defining hardware/ECU interfaces.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Interface name** | **Sender** | **Receiver** | **Format** | **Size** | **Resolution** | **Frequency** | **Description** |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

# Software static view

## Software Environment

Describe the overall OS or Software Platform in which this architecture has to fit (ex: Autosar classic version X with optional features F1 and F3), as well as the technologies to be used. (Be sure to define the versions and options used, so that all Components can refer to the same version of the OS.)

## Basic Software

### Illustration

If the project uses AUTOSAR for basic software, please delete the “Non-AUTOSAR basic software” section

If the project does not use AUTOSAR for basic software, please delete the “AUTOSAR basic sofware” section

**AUTOSAR basic software**: please delete the components that will not be used in the project from the image below:

**Application Layer**

**Complex Drivers**

**Runtime Environment**

**I/O Hardware Abstraction**

**Communication Drivers**

**I/O Drivers**

**Communication Hardware Abstraction**

**Communication Services**

**Wireless Communication Drivers**

**Wireless Communication Hardware Abstraction**

**Off Board Communication Services**

**Crypto Drivers**

**Crypto Hardware Abstraction**

**Crypto Services**

**Memory Drivers**

**Memory Hardware Abstraction**

**Memory Services**

**Microcontroller Drivers**

**Onboard Device Abstraction**

**System Services**

**Microcontroller**

Non-**AUTOSAR basic software**: please represent the basic software components that will be used in the project below:

**Application Layer**

**Runtime Environment**

**Microcontroller**

### Basic Software Component description

For each basic software component, duplicate the table below and fill it with component specific information.

|  |  |
| --- | --- |
| **<Component name>** | |
| **Role** | Describe briefly the SW component and its role. Include the name of the High-Level element it is part of. |
| **Internal/reused/external** | Indicate if the component is developed internally / reused / externally sourced |
| **Owner**  *(Not applicable for internally developed components)* | Add the name of the owner of the component |
| **Technical constrains**  *(Not applicable for internally developed components)* | Add known technical constraints that are imposed to the rest of the Software because of this external sourcing. |
| **Interfaces**  *(Not applicable for AUTOSAR components)* | Name here interfaces that this component either uses or exposes (they should be defined in the section below). When the Component is expected to only use part of the Interface (for instance because the interface has several parts), capture here which parts of the interface are to be used. (This will help with integration and tests.)  Note: Shared Memory shall be considered as an interface. |
| **ASIL level** | QM, A, B, C, or D. |
| **Level of reuse** | Fully reused without change, reused with small modifications OR “new or reused with significant changes”. This is mandated for Safety (ISO26262) considerations. Explicitly mention if this Software Component is reused from an existing one, and if this Component is to be for other Software Architects to reuse. Where very similar Software Components are available, try and reuse them or propose evolutions to them for additional flexibility to allow reuse. Where they are not reused, make sure to justify why here. |
| **<Hardware device x>** | In case the Software Component owns / manages a Hardware Device, describe it here. |
| **License** | Capture the required license for this Software component content. This information is an input to be treated as a requirement for the choice of the component content. Solution selection for this component shall match the required license, this license information shall be consistent with the project IP plan content if existing. |
| **Real time context** | If the Component is going to have significant runnable requirements (ex: an extra runnable on AUTOSAR with limited resources, or tasks in a real-time OS without a more advanced SW Framework), then document these here. |
| **Version**  *(Applicable only to reused/external component)* | What (overall) version does this document specify? |

### Basic software interfaces

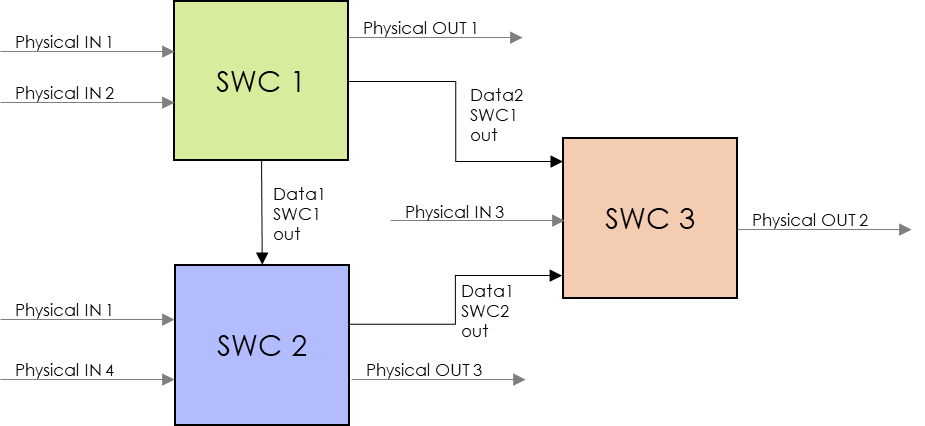
In the table below, describe the interfaces between the runtime environment(s) of the basic software and the applicative software.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Interface name** | **Sender** | **Receiver** | **Format** | **Size** | **Resolution** | **Frequency** | **Description** |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

## Applicative software

### Illustration

In this section, illustrate the applicative software components and their internal/external interfaces as show in the image below:



### Software components description

Each Software Component which has been introduced in the section above is detailed here.

Make sure to approach Components in a black-box manner (i.e. design their interactions and interfaces, but do NOT design their internal state machine and design – this shall be left for the Software Detailed Design activity).

Note that Software Requirements are mapped to Software Components and interfaces in a separate document, which has to be read alongside the present document – see section 2.6.

For each applicative software component, duplicate the table below and fill it with component specific information.

|  |  |
| --- | --- |
| **<Component name>** | |
| **Role** | Describe briefly the SW component and its role. Include the name of the High-Level element it is part of. |
| **Internal/reused/external** | Indicate if the component is developed internally / reused / externally sourced |
| **Owner**  *(Not applicable for internally developed components)* | Add the name of the owner of the component |
| **Technical constrains**  *(Not applicable for internally developed components)* | Add known technical constraints that are imposed to the rest of the Software because of this external sourcing. |
| **Interfaces** | Name here interfaces that this component either uses or exposes (they should be defined in the section below). When the Component is expected to only use part of the Interface (for instance because the interface has several parts), capture here which parts of the interface are to be used. (This will help with integration and tests.)  Note: Shared Memory shall be considered as an interface. |
| **ASIL level** | QM, A, B, C, or D. |
| **Level of reuse** | Fully reused without change, reused with small modifications OR “new or reused with significant changes”. This is mandated for Safety (ISO26262) considerations. Explicitly mention if this Software Component is reused from an existing one, and if this Component is to be for other Software Architects to reuse. Where very similar Software Components are available, try and reuse them or propose evolutions to them for additional flexibility to allow reuse. Where they are not reused, make sure to justify why here. |
| **<Hardware device x>** | In case the Software Component owns / manages a Hardware Device, describe it here. |
| **License** | Capture the required license for this Software component content. This information is an input to be treated as a requirement for the choice of the component content. Solution selection for this component shall match the required license, this license information shall be consistent with the project IP plan content if existing. |
| **Real time context** | If the Component is going to have significant runnable requirements (ex: an extra runnable on AUTOSAR with limited resources, or tasks in a real-time OS without a more advanced SW Framework), then document these here. |
| **Version**  *(Applicable only to reused/external component)* | What (overall) version does this document specify? |

### Applicative software internal interfaces

In the table below, describe the interfaces between the software components of the applicative software. For each interface, define the “sender” (maximum 1 sender per interface) and the “receiver(s)”.

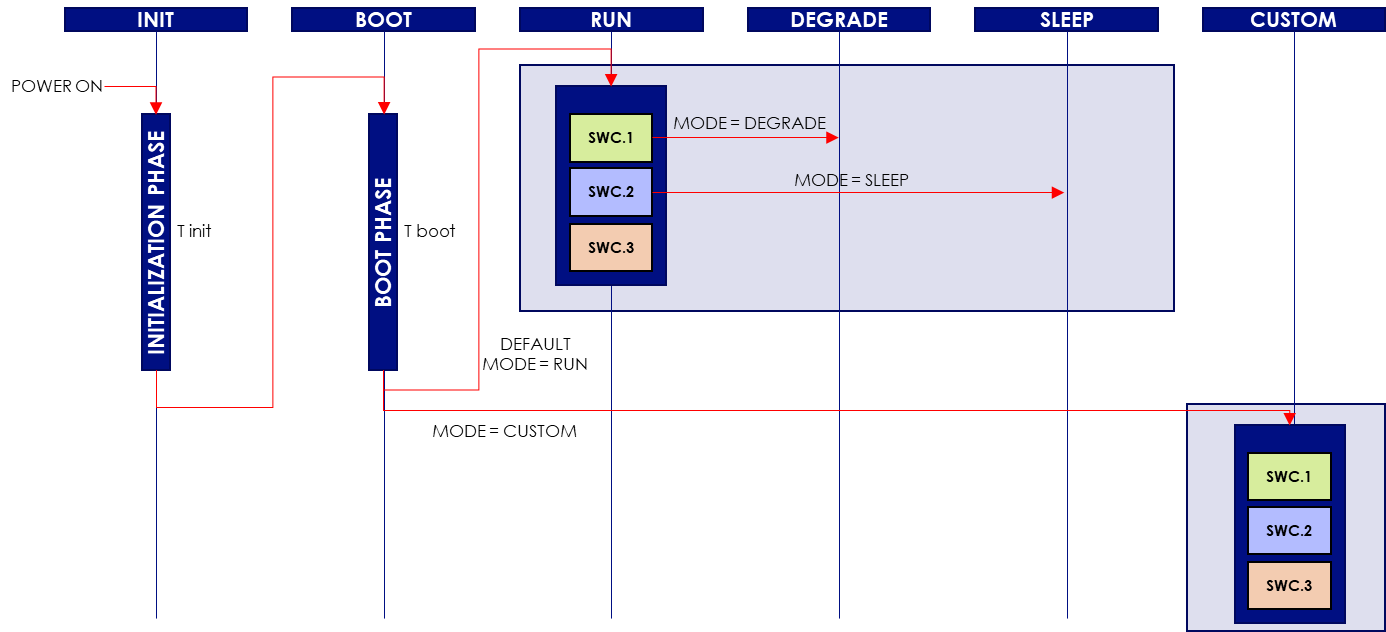
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Interface name** | **SWC roles** | | | **Format** | **Size** | **Resolution** | **Frequency** | **Description** |
| **SWC 1** | **SWC 2** | **SWC 3** |
|  | Choose an item. | Choose an item. | Choose an item. |  |  |  |  |  |
|  | Choose an item. | Choose an item. | Choose an item. |  |  |  |  |  |
|  | Choose an item. | Choose an item. | Choose an item. |  |  |  |  |  |
|  | Choose an item. | Choose an item. | Choose an item. |  |  |  |  |  |
|  | Choose an item. | Choose an item. | Choose an item. |  |  |  |  |  |

# Software dynamic view

## States

### Illustration

In this section, illustrate the different states of the software with a transition diagram such as the example below:



### Description

Fill the table below with the different states and their description:

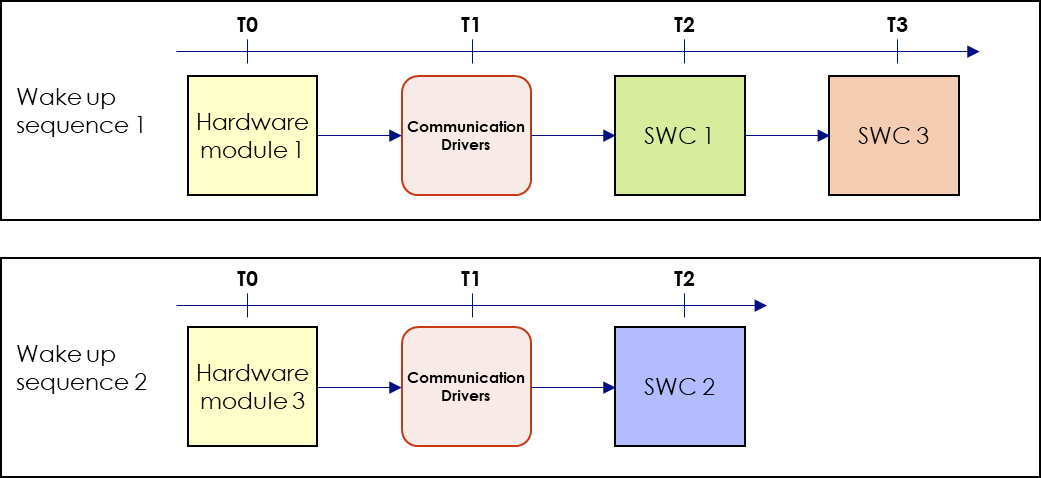
|  |  |
| --- | --- |
| **State** | **Description** |
| INIT |  |
| BOOT |  |
| RUN |  |
| DEGRADE |  |
| SLEEP |  |
| CUSTOM |  |
| … |  |

Fill the table below with information about all the possible states transitions:

|  |  |  |  |
| --- | --- | --- | --- |
| **Transition (state x > state y)** | **Condition** | **Timing** | **Error management** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Wake up & sleep sequences

### Wake up sequences

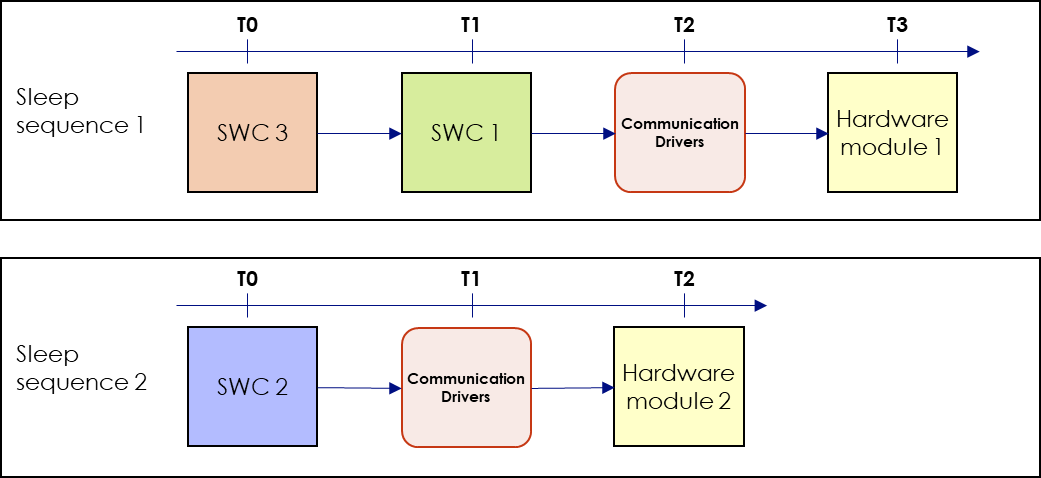
In this section, illustrate all the possible wake up sequences as shown in the example below:

Detail all the wake up sequences in the table below :

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Wake up sequence n°** | **Initial state** | **Module name + action + timing** | **Module name + action + timing** | **Module name + action + timing** | **…** | **Total estimated timing** | **Max timing** |
| 1 |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |
| … |  |  |  |  |  |  |

### Sleep sequences

In this section, illustrate all the possible sleep sequences as shown in the example below:



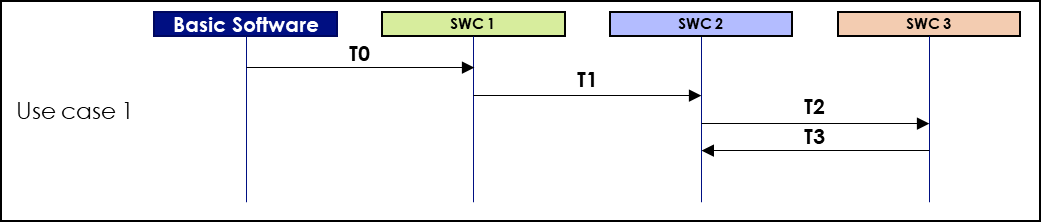
Detail all the sleep sequences in the table below:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sleep sequence n°** | **Initial state** | **Module name + action + timing** | **Module name + action + timing** | **Module name + action + timing** | **…** | **Total estimated timing** | **Max timing** |
| 1 |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |
| … |  |  |  |  |  |  |

## Use cases

### Illustrations

In this section, illustrate all the possible use cases as shown in the example below:



Detail all the use cases in the table below:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Use case n°** | **Initial state** | **Basic SW** | **SWC 1** | **SWC 2** | **…** | **Total estimated timing** | **Max timing** |
| 1 |  | Action + timing | Action + timing | Action + timing |  |  |  |
| 2 |  | Action + timing | Action + timing | Action + timing |  |  |  |
| 3 |  | Action + timing | Action + timing | Action + timing |  |  |  |
| 4 |  | Action + timing | Action + timing | Action + timing |  |  |  |
| … |  | Action + timing | Action + timing | Action + timing |  |  |  |