Software Detailed Design – Component [insert component name] (SWDD)

|  |  |  |
| --- | --- | --- |
| **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 |
| 26/10/2023 | 0.1 |  |  |  |
|  |  |  |  |  |

# 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](#_Toc149232970)

[Document Revision History 2](#_Toc149232971)

[Table of Contents 3](#_Toc149232972)

[1 Introduction 5](#_Toc149232973)

[1.1 Purpose and Scope 5](#_Toc149232974)

[1.2 Review and Approval 5](#_Toc149232975)

[1.3 Referenced documents 5](#_Toc149232976)

[1.4 Applicable documents 6](#_Toc149232977)

[1.5 Glossary & Abbreviations 6](#_Toc149232978)

[2 Detailed design definition strategy 8](#_Toc149232979)

[2.1 Document adaptation to the project 8](#_Toc149232980)

[2.2 Naming convention 8](#_Toc149232981)

[2.3 Resource consumption criteria 8](#_Toc149232982)

[2.4 Detailed design evaluation 8](#_Toc149232983)

[2.4.1 Evaluation Criteria 8](#_Toc149232984)

[2.4.2 Detailed design evaluation method 9](#_Toc149232985)

[2.5 Detailed Design requirements 10](#_Toc149232986)

[3 Software architecture overview 11](#_Toc149232987)

[4 Component overview 12](#_Toc149232988)

[4.1 Description 12](#_Toc149232989)

[4.2 External interfaces 12](#_Toc149232990)

[5 Static view 13](#_Toc149232991)

[5.1 Illustration 13](#_Toc149232992)

[5.2 Units description 13](#_Toc149232993)

[5.3 Interfaces between units 14](#_Toc149232994)

[6 Component dynamic view 15](#_Toc149232995)

[6.1 States 15](#_Toc149232996)

[6.1.1 Illustration 15](#_Toc149232997)

[6.1.2 Description 15](#_Toc149232998)

[6.2 Use cases 16](#_Toc149232999)

[6.2.1 Illustrations 16](#_Toc149233000)

# Introduction

## Purpose and Scope

**Purpose:** The purpose of the Software Detailed Design (SWDD) is to describe the Software Detailed Design of the Software components defined in the Software Architecture Design (SWAD) of the project[Type Project Name]**.**

The Software Detailed Design contains project specific information and is part of the OP’nSoft software Detailed design and unit construction 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 Detailed 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 detailed design of the component
* Describe the software units of the components
* Describe interfaces of the software units
* Describe dynamic behavior of the software units

## 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 Development Engineer
* Software Quality 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** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

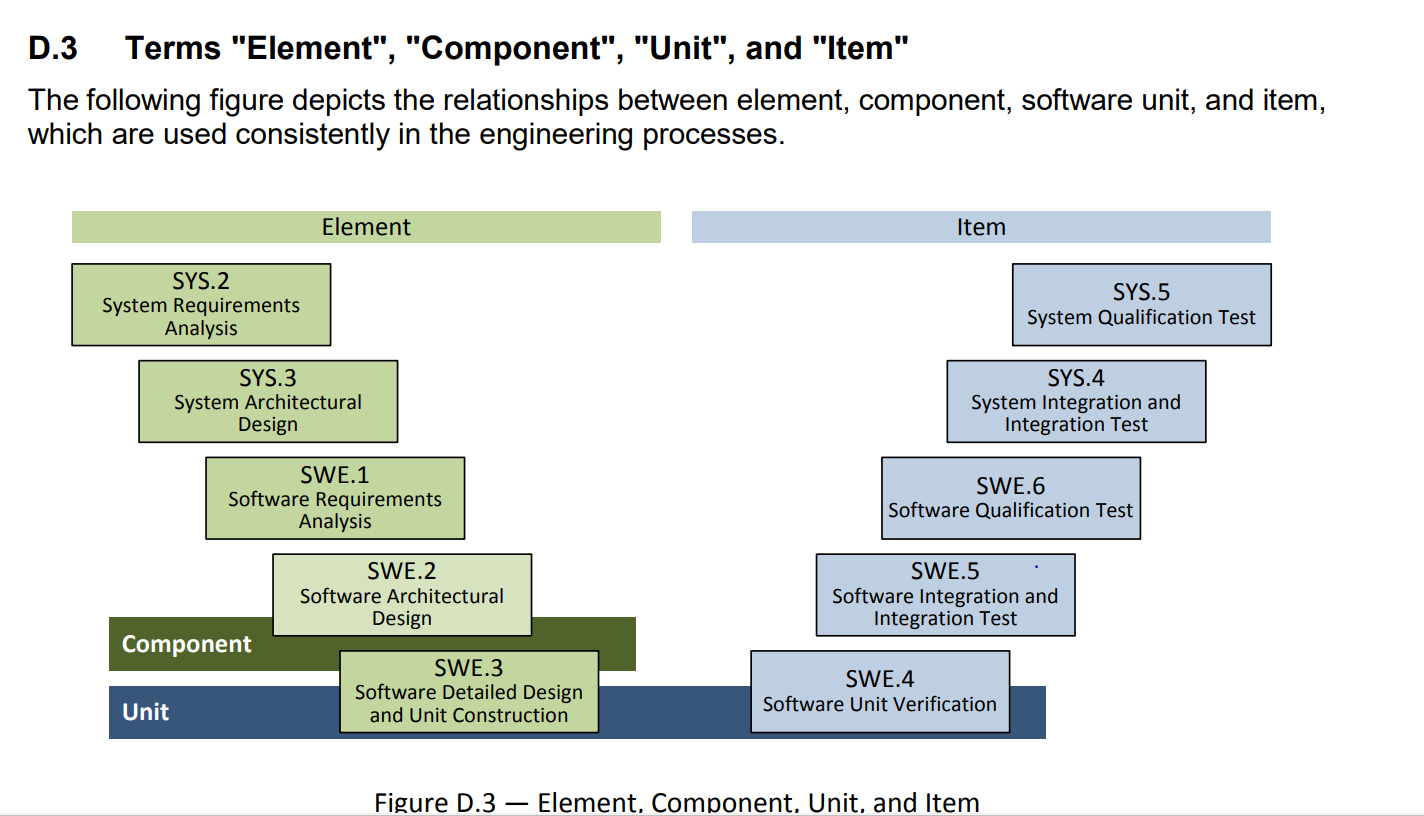
## Applicable documents

<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 V3.1 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** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Detailed design definition strategy

## Document adaptation to the project

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 and 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.

## Resource consumption criteria

Tailor the following table with project specific criteria

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

## Detailed design evaluation

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

### Evaluation Criteria

Tailor the following table with project specific criteria

|  |  |  |  |
| --- | --- | --- | --- |
| Non-functional technical requirements | **Maintainability requirements** | **Organizational requirements** | **Business requirements** |
| * Performance (response time, sample time, cycle time, flow, deadline, resource use, etc.), * Functional Safety, * Cybersecurity, * COTS (Commercial off the shelf) elements with defined interfaces, * Stability, * Efficiency, * Expandability, * Reliability, * 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. |

Some of these aspects are in contradiction to each other so that in most cases the finally selected detailed design is a compromise between these criteria.

### Detailed design 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 detailed design. 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 (e.g. for development of a completely new component):**

Several potential solutions for the software detailed design are described at least up to an abstraction level that allows the identification of the main differences between the detailed designs and that allows the evaluation in terms of interoperability, interaction, criticality, technical complexity, risks and testability for each of the potential solutions. Based on this first evaluation at least one of the proposed solutions is elaborated further by defining software units, their interfaces and their dynamic behavior. It has to be ensured that the proposed solutions which are chosen for further elaboration are able to cover the required needs of the project. Finally, these proposed and refined solutions are evaluated based on the evaluation criteria mentioned in section 2.4.1 and a decision is made:

* Selection/Confirmation of one/the proposed solution as the used detailed design for further development or
* Rejection of the previous proposed solution(s) and step back to detailed design development.

1. **Iterative Detailed Design Development**

Several solution variants come up during the development of a software detailed design (potentially also in the context of other comparable projects other than this one). A variant can be a completely different detailed design or differ from an already identified solution only in a few aspects or viewpoints. Consequently, the evaluation against the selected criteria in section 2.4.1 can take place several times during development. Such iterative steps shall be documented here.

1. **Carry Over and Adaptation of an Existing Detailed design (e.g. for platform development)**

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 2.4.1. identified weaknesses during the evaluation should be eliminated or the consequences of the weaknesses in the chosen detailed design have to be made transparent.

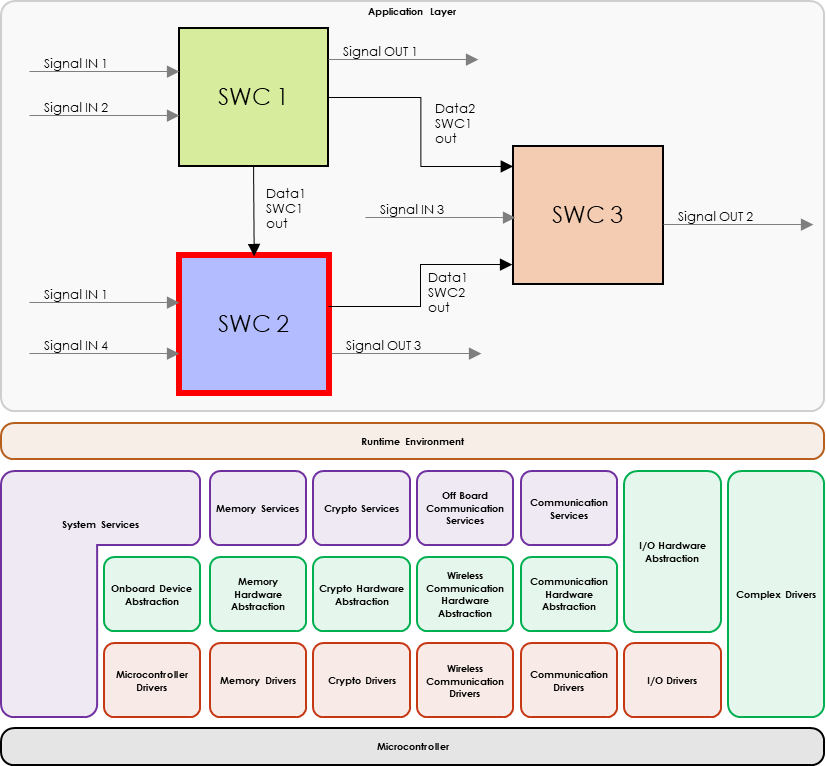
In any case, it has to be ensured that all relevant parties and all necessary competences are involved in the agreement on the selection of the final software architecture. Any weaknesses with respect to the selected evaluation criteria shall either be eliminated or justified.

## Detailed Design requirements

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

# Software architecture overview

In this section, insert the view of the basic and applicative software architecture from the Software Architecture Design document, and highlight the software component that will be designed in this document as shown below:

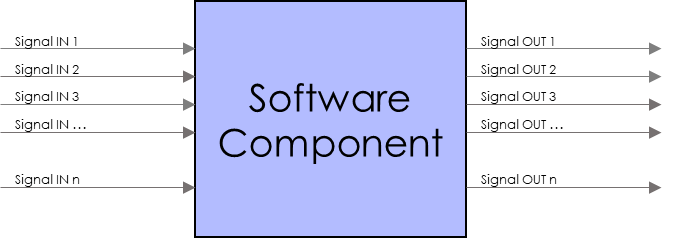


# Component overview

## Description

In this section, describe the role of the software component.

## External interfaces

In this section, illustrate all the signal inputs and outputs of the software component as shown below:

For each external interface, duplicate the table below and fill it with each interface information. If some interfaces that cover the same requirements can be regrouped, be sure to clarify it in this section.

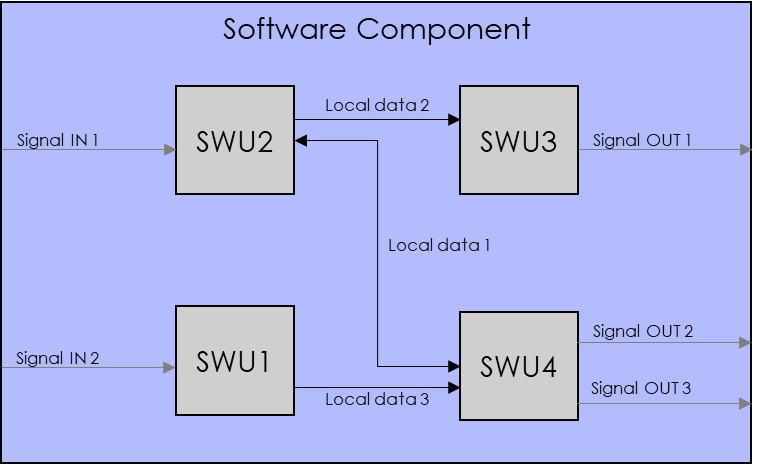
|  |  |
| --- | --- |
| **<Interface name>** | |
| **Type** |  |
| **Physical value range** |  |
| **Format** |  |
| **Unit** |  |
| **Gain** |  |
| **Size** |  |
| **Resolution** |  |
| **Frequency** |  |
| **Error values** |  |
| **Offsets** |  |
| **Initial values** |  |
| **Physical and logical data description** |  |
| **Quality information** |  |

|  |  |
| --- | --- |
| **INT-ID** |  |
| **REQUIREMENT ALLOCATION** |  |

# Static view

## Illustration

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



## Units description

For each unit represented in section 5.1, duplicate the table below and fill it with each unit information:

|  |  |
| --- | --- |
| **<Unit name>** | |
| **Role** | Describe briefly the SW unit and its role. |
| **Internal/reused/external** | Indicate if the unit is developed internally / reused / externally sourced |
| **Owner**  *(Not applicable for internally developed unit)* | Add the name of the owner of the unit |
| **Technical constrains**  *(Not applicable for internally developed unit)* | Add known technical constraints that are imposed to the rest of the component because of this external sourcing. |
| **Description** | Describe the role of the unit function. |
| **Interfaces** | Name here interfaces that this unit either uses or exposes (they should be defined in the section below). When the unit 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. |
| **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 Unit is reused from an existing one, and if this Unit could be reused by other Software Development Engineer. Where very similar Software Units 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. |
| **Version**  *(Applicable only to reused/external unit)* | What (overall) version does this document specify? |

|  |  |
| --- | --- |
| **SWU-ID** |  |
| **REQUIREMENT ALLOCATION** |  |

## Interfaces between units

For each interface between the units, duplicate the table below and fill it with each interface information. If some interfaces that cover the same requirements can be regrouped, be sure to clarify it in this section.

|  |  |
| --- | --- |
| **<Interface name>** | |
| **Type** |  |
| **Sender** |  |
| **Receiver(s)** |  |
| **Physical value range** |  |
| **Unit** |  |
| **Gain** |  |
| **Format** |  |
| **Size** |  |
| **Resolution** |  |
| **Frequency** |  |
| **Error values** |  |
| **Offsets** |  |
| **Initial values** |  |
| **Physical and logical data description** |  |
| **Quality information** |  |

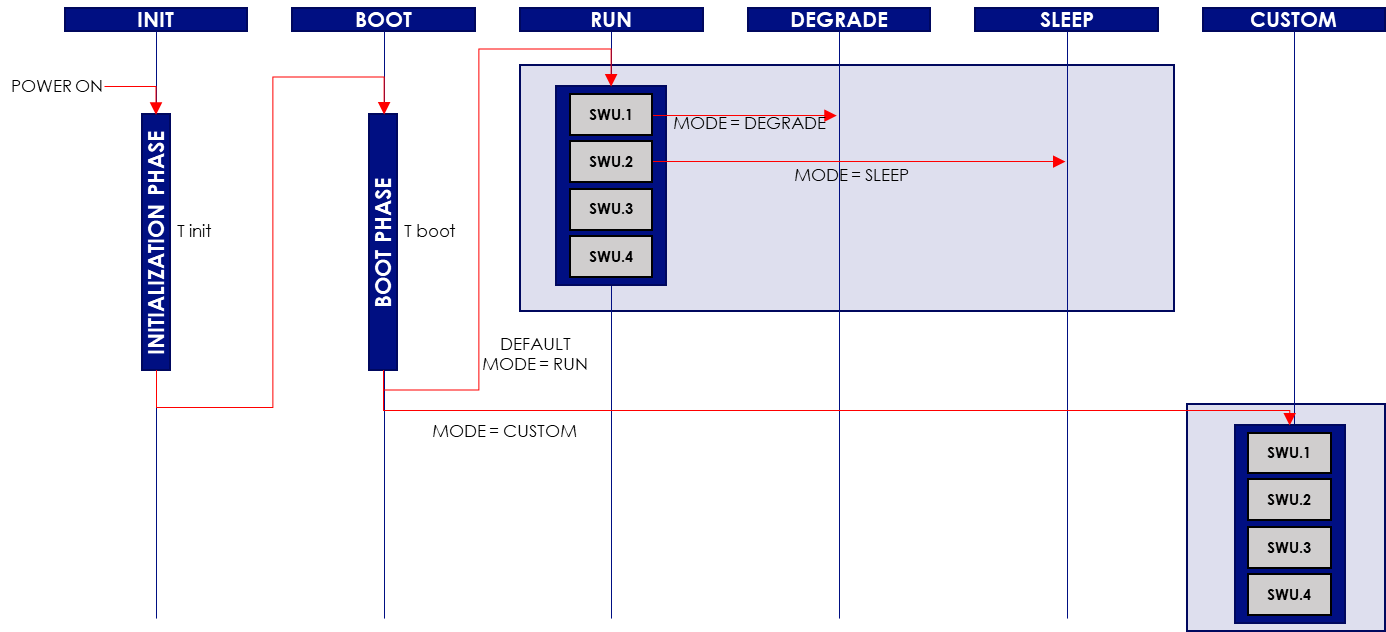
|  |  |
| --- | --- |
| **INT-ID** |  |
| **REQUIREMENT ALLOCATION** |  |

# Component dynamic view

## States

### Illustration

In this section, illustrate the different states of the software component 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 |  |
| … |  |

|  |  |
| --- | --- |
| **STATE-ID** |  |
| **REQUIREMENT ALLOCATION** |  |

### Transitions

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

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

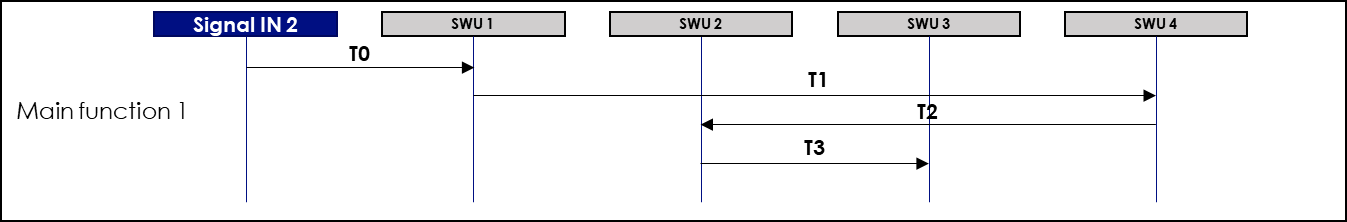
|  |  |
| --- | --- |
| **TRANS-ID** |  |
| **REQUIREMENT ALLOCATION** |  |

Fill the table below with the role of each software unit for each state transition:

| **Transition (state x > state y)** | **SWU1** | **SWU2** | **SWU3** | **…** |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## Main functions of the software component

### Illustrations

In this section, illustrate all the main functions as shown in the example below:

Detail all the main functions in the table below:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Main Functions n°** | **Initial state** | **Signal IN** | **SWU 1** | **SWU 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 |  |  |  |

|  |  |
| --- | --- |
| **FUNC-ID** |  |
| **REQUIREMENT ALLOCATION** |  |