

**<Project Short Name>**

**<Project Full Name>**  
**<Customer>**

# Software Design Description

# Software Design Description

Company Logo

Department

---

| Version | Document maturity<br>(draft / valid) | Date of Issue<br>(200x-MM-DD) | Author/Owner | Check/Release | Description     |
|---------|--------------------------------------|-------------------------------|--------------|---------------|-----------------|
| 0.1     | draft                                |                               |              |               | Initial version |
|         |                                      |                               |              |               |                 |

**TABLE OF CONTENTS**

**1 INTRODUCTION .....5**

1.1 PURPOSE AND SCOPE .....5

1.2 DEFINITIONS, TERMINOLOGY AND ABBREVIATIONS .....5

1.3 REFERENCES.....5

**2 DEVELOPMENT CONTEXT .....6**

2.1 SYSTEM DESCRIPTION .....6

2.2 DEVELOPMENT AND DESIGN CONSTRAINTS .....6

    2.2.1 *General Constraints*.....6

    2.2.2 *System Constraints*.....6

    2.2.3 *HW Constraints*.....6

    2.2.4 *SW Constraints* .....6

**3 SOFTWARE ARCHITECTURE.....7**

3.1 FUNCTIONAL ARCHITECTURE.....7

3.2 SOFTWARE ARCHITECTURE.....7

    3.2.1 *Interfaces* .....7

    3.2.2 *Module Break Down* .....7

    3.2.3 *OS Description / Usage* .....7

        3.2.3.1 *General*.....7

        3.2.3.2 *Performance and Timing Considerations*.....7

3.3 INITIALIZATION & SYSTEM EVENTS.....7

3.4 DATA FLOW .....8

3.5 CONTROL FLOW .....8

**4 MODULE DESCRIPTION.....9**

4.1 MODULE <NAME> .....9

    4.1.1 *Function <Name>* .....9

**TABLE OF FIGURES**

ERROR! NO TABLE OF FIGURES ENTRIES FOUND.

# 1 Introduction

## 1.1 Purpose and Scope

State the purpose and aim of the document. If applicable, it may also be an abstract at this point, which summarizes to contents in a nutshell.

State for which project, department or group, etc. this document applies.

## 1.2 Definitions, Terminology and Abbreviations

Adapt and expand the below abbreviation table as necessary for your purpose. Further add in this section any definitions which the document is based on and which are necessary to make the contents clear.

The following element types are defined for use in the data tables:

For variables:

- P = pass parameter (argument)
- R = return value of a function
- G = global variable
- S = static variable
- L = local variable

For constants:

- D = define value
- C = constant value
- E = EEPROM / FLASH value which will be reloaded (into RAM) at system startup
- \* = an asterisk has to be added after every constant type if it is subject to calibration
- \*\* = two asterisk have to be added after every EEPROM or FLASH value if it is subject to self learning or other adaptive mechanisms.

Table of abbreviations used in this document

| Abbreviation | Description   |
|--------------|---------------|
| TBD          | To be defined |
|              |               |
|              |               |
|              |               |

## 1.3 References

- /1/ Reference to any document quoted in this document
- /2/ Reference to any document quoted in this document
- /3/

## 2 Development Context

*You can name your design method here, and remember: using tools for visualization/normalization is **only part** of a complete design method. Yet their output is very helpful and can be placed either in this document or as additional documents into the SW project folder (right behind this document).*

*Generally only constraints are listed in this chapter. In the next chapter, where the design choices are given in detail, you can reference these constraints as a reason for your particular design choice below.*

### 2.1 System Description

*Give a general description of the complete system here; preferably in form of a commented drawing. This is only for an overview and can be left out or be the same as used in the software requirements specification.*

### 2.2 Development and Design Constraints

#### 2.2.1 General Constraints

*What are the general constraints implied by the development process and what are the impacts on the SW architecture and module design?*

*List things such as sacrificing a part of the design phase (e.g. no preliminary design and therefore all SW in one module/task) and therefore sacrificing reusability to fulfill tight schedules.*

*What are the general constraints deriving out of the design method itself?*

*Describe the effects of the development platform (design tool, test tool) or a new design method (e.g. object-oriented design on the (future) architecture in terms of module breakdown, module definition, module relationship. **This can be a reference to the software requirements document or the requirements database if the quality of these items is sufficient (to fulfill the demands for traceability!).***

#### 2.2.2 System Constraints

*Are there system requirements influencing design choices?*

*E.g. design choices resulting of timing constraints (can be referenced if already defined in the specification), protocols or interfaces to HW which will be available only late in the V-cycle (leading to good encapsulation of that functionality for easy late changes), behavior enforced by safety concept.*

*Don't forget to describe also modules compensating non-available system components (for prototypes) which will not be present in the final version. **This can be a reference to the software requirements document or the requirements database if the quality of these items is sufficient (to fulfill the demands for traceability!).***

#### 2.2.3 HW Constraints

*Are there any constraints on the software architecture deriving from limited hardware resources?*

*E.g. implementation of functionality which would be easier solved with additional HW (but also with higher costs like low-pass filters for instance), constraints deriving from restrictions of low-cost  $\mu$ -controllers (small ROM/RAM, limited interrupt abilities,...), possible changes of the CPU, small ROM size leading to optimized modules difficult to maintain or reuse, small RAM size enforcing memory overlap. The assignment of resources to the software i.e. RAM, ROM and runtime has to be agreed with the SW domain and fixed here. **This can be a reference to the software requirements document or the requirements database if the quality of these items is sufficient (to fulfill the demands for traceability!).***

#### 2.2.4 SW Constraints

*Are there any constraints from the SW ?*

*E.g. usage of a specific Operating System or reuse of some parts of SW from other products (e.g. 'Basic Software') can influence the SW architecture (in general a problem of reusability), limitations of (self written) OS (like no multi-tasking in round-robin OS). **This can be a reference to the software requirements document or the requirements database if the quality of these items is sufficient (to fulfill the demands for traceability!).***

## 3 Software Architecture

### 3.1 Functional Architecture

*Describe how the software is divided into functional subparts, as e.g. terms, data pre-processing, fire logic, etc. This is focusing on a high level functional design, which should be described and documented here. Drawings from design tools can be included here. The use of standard design methods like UML is encouraged. Define your own sub-chapters as appropriate.*

### 3.2 Software Architecture

*Describe how the SW architecture is divided into subparts. Show how these subparts are distributed into modules / tasks / interrupts. For large architectures you can use the following subchapters or create subchapters of your own choice fitting your design method best.*

*In any case the distribution of the single modules on tasks & interrupts and the careful description of synchronous activities (activated by timers) **must** be part of this design document. This can be done for instance with the next two subchapters.*

*Again commented graphics are more descriptive than written prose.*

#### 3.2.1 Interfaces

*Describe the public interfaces of the software modules / components..*

#### 3.2.2 Module Break Down

*Describe the module structure of the software, the used libraries and general header files, etc.*

#### 3.2.3 OS Description / Usage

##### 3.2.3.1 General

*If you use an existing OS describe the **usage** of the OS **only** (i.e. all used services like semaphores, queues, SW timers...). If you also must create the OS (or new services for an existing OS) describe them here. Yet the type (round-robin, pre-emptive multitasking...) of the OS should always be described here. You can add additional subchapters if appropriate.*

##### 3.2.3.2 Performance and Timing Considerations

*Define here how the software is called, i.e. at which frequency, describe how long it may be running, if there is a possibility of preemption or canceling, etc. During the design phase the designer has to assess the performance of the software architecture in terms of tasks duration, accuracy of treatments, etc.*

*For better traceability references to performance requirements defined in the software requirements specification should be added.*

### 3.3 Initialization & System Events

*Describe how the SW initialization is made and how the system events are managed.*

*Global events not being part of the specification (because their necessity showed up only during design phase) must also be described here. If there is a centralized management of system events this must result in a module of its own.*

*For example: What must be initialized on HW Reset event and also on system events such as engine on/off, key on/off and whatever global events which are defined in the software requirements specification or in this document.*

### 3.4 Data Flow

*Graphical or textual description on how software modules exchange data related to the control flow. These flows should define the **complete** communication between modules.*

### 3.5 Control Flow

*Graphical or textual description on how software modules invoke each other (e.g. synchronous versus asynchronous calls, linear versus parallel execution or call graph).*

## 4 Module Description

*Describe each module in terms of interface and body (one subchapter per module):*

- description of exported operations (name, parameters, return values)
- description of exported data
- description of private operations
- description of private data
- SW internal flow

### 4.1 Module <Name>

*Copy this sub-section and the subsection of the function description as often as it is needed*

#### 4.1.1 Function <Name>

**Prototyping:**

|  |                 |
|--|-----------------|
| C - prototype:                         | Part of module: |
| <i>Describe the c-prototyping here</i> |                 |

**Referenced C functions:**

|   |                 |
|---|-----------------|
| Referenced C - function:                                    | Part of module: |
| <i>Name any functions which are called by this function</i> |                 |
|   |                 |

**Executed by (time period):**

*State how this function is called, e.g. cyclic every xx ms, etc.*

**Outline of the function:**

*Give a short abstract of what the function is doing.*

**Data description:**

*Describe the data in the below tables. There is one for the variables and one for parameters. Use the definitions in the definition section above for the data types.*

**Variables:**

| Type | Name | Description | I/O | Data Type | Value Range | Resolution & Unit |
|------|------|-------------|-----|-----------|-------------|-------------------|
|      |      |             |     |           |             |                   |
|      |      |             |     |           |             |                   |
|      |      |             |     |           |             |                   |

# Software Design Description

Company Logo

Department

|  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|
|  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|

## Parameters:

| Type | Name | Description | I/O | Data Type | Value Range | Resolution & Unit |
|------|------|-------------|-----|-----------|-------------|-------------------|
|      |      |             |     |           |             |                   |
|      |      |             |     |           |             |                   |
|      |      |             |     |           |             |                   |

## Body:

*Visualize the inputs and outputs to the function. Put in the control flow chart or any other helpful drawings and add any further description to describe the function as detailed as possible.*