Project „Safety Function for Temperature Monitoring System “

Safety requirement specification

Modification history:

Version	Date	Modification	Creator	Auditor
1.0	2020-05-04	Creation of SRS	SaS	DS
1.1	2020-05-27	Revision of the requirements	SaS	DS
1.2	2021-03-16	Minor corrections	SaS	DS
1.3	2021-04-06	Formal corrections	SaS	DS
1.4	2023-05-07	SRS for „Safety Function for Temparature Monitoring System“	Kaushik	

Content

1.0 Introduction

A safety-relevant software function for safe temperature monitoring is to be developed. In addition, some auxiliary functions for the conversion of the units and the output of the temperature are to be programmed.

2.0 Safety requirements

2.1 Standards

[SRS1]	A safety relevant software function for safe temperature monitoring shall be developed according to IEC 61508:2010
[SRS2]	The safety relevant software function for safe temperature monitoring shall be designed to meet SIL3 requirements
[SRS3]	The system shall have the following system operating states based on the data read from the two temperature sensors A. Normal State – Temperature range is OK B. Safe State – Temperature range exceeded the limits C. Dangerous State – Failure of temperature sensors
[SRS4]	The system shall be developed such that it is compatible across multiple hardware
[SRS5]	Hardware dependent functions shall be informed to customer through user documentation
[SRS6]	The temperature sensor data read from hardware shall be checked for plausibility. Hint: • Checksum shall be implemented to ensure the data read from sensor is plausible • The sensor data shall have a accuracy of upto 4 decimal places
[SRS7]	The system shall be designed to be capable of handling implausible temperature sensor values which might arise due to hardware issues
[SRS8]	The sensors module shall be implemented as 1oo2 system
[SRS9]	The sensor data is considered as faulty if the discrepancy between the values read from two sensors exceeds the allowed limit Hint: The discrepancy limit shall be decided by the customer since it depends on the sensor hardware specification
[SRS10]	The system shall monitor the temperature data under the **monitorTemp** function
[SRS11]	The **monitorTemp** function shall have the following prototype Parameters: • TempFormat (char) – C (Celcius) or F (Fahrenheit) • min_Temp (float) • max_Temp (float) • max_DisCr (float) • TempS1 (float) • TempS2 (float) Return (uint8): • 7 – temperature range OK • 5 – function error • 3 – temperature range limits exceeded, alarm
[SRS12]	The system shall detect the discrepancy between the temperature values read from the two sensors under the **checkTemp** function
[SRS13]	The system shall display the current operating state under the **displayTemp** function
[SRS14]	The **checkTemp** function shall be implemented as a sub routine within **monitorTemp**

[SRS15]	The **displayTemp** function shall be implemented as a sub routine within **monitorTemp**
[SRS16]	The return value of **monitorTemp** function shall be updated according to the return value of **checkTemp** function
[SRS17]	The **checkTemp** function shall have the following prototype Parameters: • min_Temp (float) • max_Temp (float) • max_DisCr (float) • TempS1 (float) • TempS2 (float) Return (uint8): • 7 – temperature range OK • 5 – function error • 3 – temperature range limits exceeded, alarm Hint: The accuracy of temperature data shall be limited to 2 decimal places
[SRS18]	The **displayTemp** function shall output the temperature values based on the **TempFormat** parameter mentioned in [SRS11]
[SRS19]	The **displayTemp** function shall compute the mean value (Temp) of the two-sensor data (i.e., TempS1 and TempS2) Hint: This computed value shall be passed to **calcC2F** and **calcC2F** for converting the data as per user request
[SRS20]	The **calcC2F** sub function shall be implemented to convert the temperature data from Celcius (C) to Fahrenheit (F)
[SRS21]	The **calcC2F** sub function shall have the following prototype and convert the temperature data using the below formula Parameter: • Current temperature (float) Return (float): • Converted temperature Formula: • Fahrenheit = (9/5) * Celcius + 32
[SRS22]	The **calcF2C** sub functions shall be implemented to convert the temperature data from Fahrenheit (F) to Celcius (C)
[SRS23]	The **calcF2C** sub function shall have the following prototype and convert the temperature data using the below formula Parameter: • Current temperature (float) Return (float): • Converted temperature Formula • Celcius = (Fahrenheit – 32) * (5/9)
[SRS24]	The hardware specific dependencies implemented in **displayTemp** function shall be documented in the user documentation to fulfil [SRS5]
[SRS25]	The **displayTemp** function shall have the following prototype Parameters: • TempFormat (char)

	• Temp (float) • TempOK (uint8) Return (void)
[SRS24]	The **displayTemp** function shall output the system state as shown below Current Temperature in selected format: ____ F or C Temperature in another format: ____ F or C Temperature range: OK or ERROR or ALARM
[SRS25]	The system shall be implemented using C programming language Hint: The software shall be implemented such that it is modular and it can be extended to implement future requirements
[SRS26]	The maximum execution time of **monitorTemp** function shall be 5ms
[SRS27]	The software shall be tested according to IEC 61508:2010 Hint: Offline tests and periodical tests shall be planned accordingly

2.3 Specification of the requirement for the safety software

Note: Only the required technical safety characteristics of the product, not of the project, must be described in compliance with the safety integrity level. Organizational issues of the project are described in the safety plan.

Which SIL does the software have to fulfill?
[SRS2]

What is the safe state? (If there are several operating modes, there may be different safe states)
[SRS3]

Is the range of application of the software to be kept flexible or bound to a fixed hardware?
[SRS4], [SRS5]

Requirement for detection, indication and handling of faults in the programmable hardware. (for hardware-specific programming)
[SRS6], [SRS7]

Requirement for detection, indication and handling of faults of the sensors. Are the sensors multi-channel?
[SRS8]

Requirement for detection, display and handling of actuator faults. Are the actuators multi-channel?
[SRS9]

Requirement to detect, display and handle faults of the software. Self-monitoring?
[SRS10], [SRS11]

Possibility for testing (verification/validation) of the products? Periodical online tests / offline tests
[SRS27]

Does the software have to be updateable?
[SRS25]

What is the internal processing time?
[SRS26]

What is the fault reaction time?
[SRS26]

What system of units is intended? (metric / angloamerican / both)
[SRS20], [SRS21], [SRS22], [SRS23]

Setup/Operator interface with exclusion of "pathological cases". ("pathological cases" are input combinations that do not occur during normal operation but are possible under fault conditions).
[SRS6]

Requirement for independence between functions (multiple requirements possible).
[SRS14], [SRS15]

Evaluation of the sensors: In which way should the sensor values be evaluated?

[SRS9]

Programming language selection: Which programming language should be used?

[SRS25]

What is the necessary accuracy for sensor readout?
[SRS6]

What is the necessary accuracy for the evaluation of the temperature values within the software function?

[SRS17]

What are possible (remaining) dependencies between software sub-functions?

[SRS18]

Selection and justification of techniques and measures for the software requirement specification. (see 03_Prüfversion_Fehlervermeidung_Software_Vorlage_V00_1.docx, Tabelle A.1)

3.0 Functional description

Internal states and their dependencies (e. g. Petri-net, function block diagram)
[SRS3]

Layout display message

[SRS24]

4.0 Verification of the SRS

- Who shall perform the verification?

Company:	CKN GmbH
Name, first name:	Wallace David
expertise:	Strategic Planning
phone / e- mail:	wallace@de.ckn.com

- Are the specification of the software's safety requirements adequately met in terms of functionality, safety integrity, performance, and all other safety planning requirements?
  Yes

- Does the validation plan for the software safety aspects meet the specification of the software safety requirements?
  Yes

5.0 Note

This template is **not considered complete, but applicable**. The template contains the requirements of IEC 61508 for the development of software functions. The template is to be regarded as alive, i.e. during the development process, possibly issues may be discovered, which are not contained in the template. These should be added to the template at the correct place, in order to be able to consider them during next use. The more diverse software is created by means of this template, the more completely and exactly can the development process for software be realized in the future.