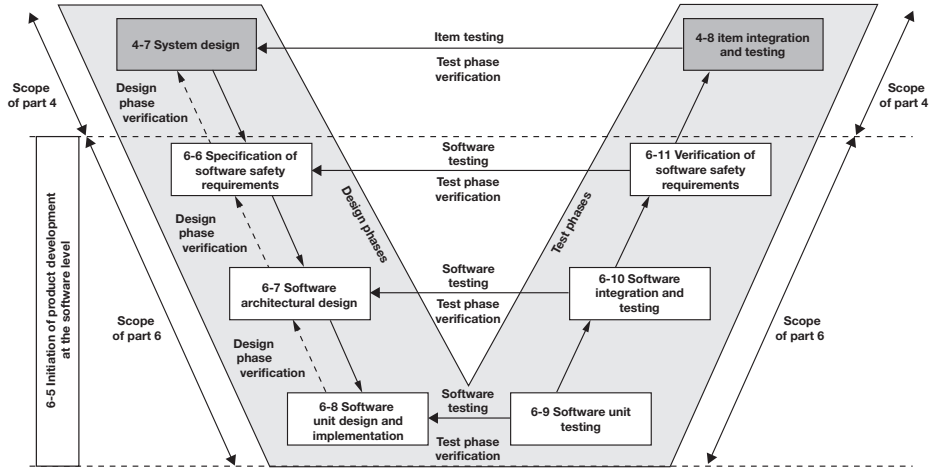
Foundational Software for Functional Safety

Jay Thomas
LDRA
Technical Development Manager

Siddharth Deshpande
Texas Instruments
Hercules MCU Lead Software Engineer

Functional safety is a concept which is used to manage risk in dynamic systems.

Functional safety involves active systems and provides processes to manage risk in the context of the system inputs and expected response. It is often applied in industries such as automotive, aerospace, industrial and medical devices where there are regulatory and commercial safety constraints. These constraints require the management of the software development life cycle (SDLC) as it pertains to risk. Example safety standards such as ISO 26262, IEC 61508, IEC 62304 and DO-178B/C aim to reduce risk by requiring functional safety elements to be examined at every stage of the SDLC – including requirement specification, design, implementation, verification, validation and deployment.



In certain development workflows this process is commonly represented by the V diagram, such as the one above from ISO 26262. In the next sections, we will examine functional safety in different domains and discuss similarities and differences. We will then look at real systems and how they can be partitioned and how you can help to ensure that the system can more readily meet functional safety requirements by looking at the individual components. This will provide the context to look at the SafeTI™ software framework – which can be used as a foundation for functional safety systems.

Introduction to Functional Safety

To begin – consider the definition of functional safety. Functional safety is a system-level concept where, when achieved, each of the underlying safety functions performs as expected even under conditions of failure modes of operation. When a functional safety system is governed by software, there are requirements that must be followed during development, verification, and deployment.

The origin of functional safety standards is the avionics industry. The complexity of aircraft systems has driven functional safety to other functional safety systems over the last decades (see figure below). DO-178B and now DO-178C generally provide the most specific guidance in terms of test activities required. In industrial control – IEC 61508 took many of the same principles and generalized them. Once they were generalized, other standards

such as ISO 26262 for automotive and EN 50128 for railway looked at the specific needs of those industries and made sure that they were accounted for in those specific standards.

These standards are industry specific, but there are some variations:

Avionics	DO-178B (First published 1992) / DO-178C
Industrial	IEC 61508 (First published 1998, Updated 2010)
Railway	EN 50128 (First published 2001)
Nuclear	IEC 61513 (First published 2001)

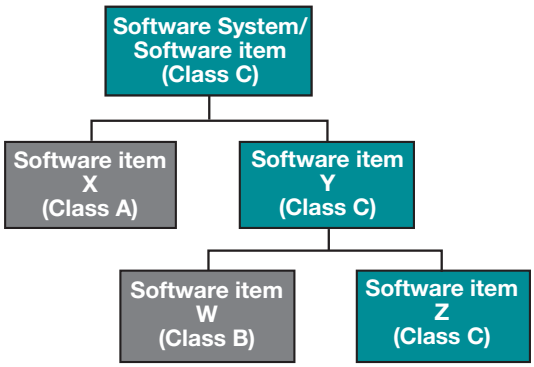
System Partition Principles

Functional safety can also involve system partitioning. Every system is composed of a set of subsystems. These subsystems each require risk management at different levels. For instance, some subsystems may cause a catastrophic event if they fail, while others will have no effect. The subsystems typically have dependencies on each other and can often be decomposed into trees of dependent software elements. An example is shown at the top of the next column. Software systems inherit the most stringent safety class – if a system depends on an unsafe subsystem – then it is unsafe.

This leads us to the Safety Element Out of Context / Compliant Item concepts of ISO 26262 and IEC 61508 respectively. These principles allow you to perform your safety activities on the subsystem level and make sure that hazards are contained in specific software items. For the system as a whole, this means down to the lowest level subsystems – the board support package (BSP) and driver layer.

Safety Classification Principles

- No adverse side effects caused by X and W
- No hazard contributing effect by X and W
- Z include all software system contributions to hazards
- Software system inherits "worst" safety class



These all should be designed to achieve the highest levels of safety. All other subsystems, including those that contain contributions to system hazard are connected to these low-level systems.

The SafeTI™ [1] software development process utilizes the Safety Element Out of Context (SEooC) concept. This concept allows users to develop systems independently with a view of assumed hazards and risks. Given the view of assumed risks and the contexts of functional safety standards – such as (i.e. ISO 26262-10:2012) these subsystems can then be integrated into a variety of target applications such as automotive braking and steering systems, industrial automation and drives, medical devices, aerospace control systems and many others.

The SEooC approach is particularly applicable for highly integrated, distributed systems as the safety of the overall application can be considered in the contexts of its parts – each of which can contribute to overall failure in different ways.

[1] *See www.ti.com/safeti for more information on SafeTI control solutions.*

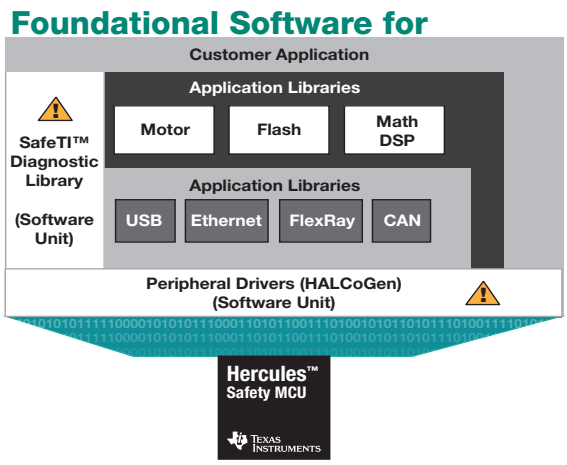


Figure 1. Software Stack built on top of Hercules™ Safety MCU

Figure 1 shows a typical software stack. The customer application is developed on top of low-level drivers and application libraries. Typically the software components of the software stack are developed by different vendors, which are then integrated by the system developers into a system or application.

For a functional safety system, it is very important to have a solid foundation with adherence to industry functional safety standards with which to start. TI provides a strong foundation for the software stack through HALCoGen (Hardware Abstraction Layer Code Generator) and the SafeTI™ Diagnostic Library. These software components provide low-level peripheral drivers, device initialization and diagnostic functions for Hercules MCUs. Since they have been developed by following the ISO 26262 and IEC 61508 safety standards, they can be used by the customer assist in the development of their functional safety systems. The software development process for these software components has been certified by TÜV NORD to meet ASIL D and SIL 3 levels of safety integrity. TÜV NORD is an internationally recognized and independent assessor of compliance to quality and safety standards.

A functional safety system designer is responsible for creating systems (and any hardware or software components incorporated in these systems) that meet all applicable safety, regulatory and performance requirements. When the system is submitted for functional safety certification, all the software components are assessed to determine compliance to functional safety standards. This is a monumental task for the system developer if they are to provide all the artifacts for all the software components coming from different vendors.

SafeTI™ compliance support packages (CSP) help make this task easier for the system developer. SafeTI™ Compliance Support Packages are developed according to TI's certified software development process and are available for HALCoGen and the SafeTI™ Diagnostic library.

These CSPs provide a helpful starting point for customers who need to provide similar evidence for their functional safety software.

SafeTI™ Compliance Support Package

The SafeTI™ Compliance Support Packages (CSPs) help make it easier for customers to comply with the functional safety standards. The LDRA tool suite is used in the development of the CSPs. The artifacts included in the CSPs can be classified into three categories:

Requirements and Design

- **Software Safety Requirements Specification**
  This document provides the purpose, scope and requirements of the software product being developed. It lists all the software requirements assumed during the development of the product. Each of the safety requirements is assigned an ASIL/SIL level based on the technical safety concept derived from the SEooC approach

- **Software Architecture/Design Document**
  This document provides an overview of the architecture and software design of the product being developed. It describes the overall structure, architecture style, constraints and the application programming interface (API) it offers.
- **Software Safety Manual**
  The Safety Manual provides information needed by system developer to assist in the integration of this software in their end safety system. This information will help the system developers to assist in the creation of a functional safety system which uses this software unit.

Test Reports

- **Detailed Static Analysis Report**
  This report provides a summary of the software quality metrics and the source code violations as per coding guidelines. TI utilizes a coding standard for static analysis that is primarily derived from MISRA C:2004 guidelines. The enforcement of this coding standard improves software quality by restricting usage of potentially dangerous language features. In addition, by following these guidelines, the resulting code will be more maintainable, clear, concise, and robust.

  Apart from this, quality metrics are also measured for the source code. These metrics are a subset of Hersteller Initiative Software (HIS) Quality metrics. HIS is a consortium of five major automotive manufacturers (Audi, BMW, DaimlerChrysler, Porsche and Volkswagen). This group has specified a fundamental set of metrics to be used in the evaluation of software. Comment density, cyclomatic complexity, procedure exit points, fan in/fan out, number of global variables and Halstead metrics are some of the measurables used for the evaluation of the quality of the software product
- **Detailed Dynamic Analysis Report**
  This report provides a summary of the structural coverage metrics (statement coverage, branch coverage, and modified condition/decision coverage (MC/DC coverage). LDRAunit® is used to perform dynamic analysis. It provides facilities to improve code efficiency and detect potential software defects. These structural coverage metrics help toevaluate the completeness of the test cases and demonstrate there is no unintended functionality. This report indicates the statements in the code, branch conditions, MC/DC conditions which are covered and not covered during the test run.
- **Test Matrix**
  This report provides detailed information about the results of the formal testing including information about the test cases, test case descriptions, input and output parameters, test results and requirements associated with each of the test cases.
- **Traceability Matrix**
  Traceability Matrix report provides bidirectional traceability between different phases of software development. This is a comprehensive report detailing the traceability between requirements, design, source code, and test cases. This matrix helps track safety requirements bi-directionally from specification to design to implementation to test cases.

Test Automation Unit

The CSPs feature a Test Automation Unit (TAU) tool that enables customers to re-execute unit-level test cases in their environment.

The default configuration is tested by TI, and the test results and code coverage reports included in the CSP are relevant to the default configuration. Providing such fixed test results and code coverage reports may not suit all customer needs. The TAU, coupled with configurable software provided by HALCoGen, therefore provides the flexibility to enable customers to execute and provide evidence for the unit test cases within their specific configuration. In so doing, the TAU utilizes LDRAunit® for performing dynamic analysis.

The TAU also provides infrastructure for the customers to add their own test cases for their particular configuration. The TAU generates the dynamic coverage and regression report for these selected test cases.

The Dynamic Coverage reports include a summary report and individual report for each of the source files. The summary report provides a summary of the structural coverage metrics at a file level whereas the individual report provides the coverage metrics at an API level. For each API within the source file, the code coverage metrics are obtained.

The TAU generates a regression report for each test sequence. In addition, a high-level summary report and individual reports are also generated.

The summary report provides a summary of the total number of test cases executed, detailing the number of test cases which have passed and failed.

The individual regression report provides detailed information about the test cases executed. The provided information includes a test case description, input and output parameters, and the test results for each test case within the sequence.

Conclusions

Developing functional safety-compliant software is a challenging task with risk having to be managed throughout the entire software life cycle.

SafeTI™ Compliance Support Packages (CSPs) and the LDRA tool suite can help simplify the development of functional safety software. The

SafeTI™ CSPs provide documentation, reports and a unit test capability to assist customers using the Hercules MCU software components to comply with functional safety standards such as IEC 61508 (Industrial Controls) and ISO 26262 (Automotive).

The CSPs can also be a helpful starting point for customers who need to provide similar evidence for their functional safety software. The CSPs help

reduce software validation efforts and provide work products that can assist with end system functional safety certification.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265