

Controlled separation between standard & safety software functions in ECUs

Kai Niestroj (Sensor-Technik Wiedemann GmbH)

In addition to standard functionalities to control machinery, ECUs have also to deal with an increasing number of functional safety requirements. To prevent potential hazard to human and machine, development according to EN13849 [4] and EN61508 [5] is required.

Rising complexity of software applications increases the development efforts. It has to be ensured that safety critical functionalities are not adversely affected by standard functionalities, which are considered as non-safety related.

The EN61508-3 [6] describes technical measures to prevent those negative side-effects of software and reduces the effort for certification of non-safety software modules.

Why do we have to use harmonised standards?

The machinery directive 2006/42/EC [1] introduces identical requirements for safety of machinery in every country within the European Economic Area (EEA). This directive will be converted into national laws, i.e. the Federal Republic of Germany converts the machinery directive into the Product Safety Act (ProdSG) [2]. In §4 of the ProdSG [2] is a reference to harmonised standards, which are published in the official Journal of the European Communities. There, for instance you will find the standards of EN13849 [4] respectively EN62061 [3].

The EN61508 [5] is not listed as a harmonized standard. In fact the EN61508 [5] is a generic standard. EN62061 [3] for example is derived from EN61508 [5]. In EN13849 [4] you are able to find references to the generic standard [5].

All mentioned standards above have the common aim, to reduce risks and hazards to humans and machines. To fulfil ProdSG [2] **Fehler! Verweisquelle konnte nicht gefunden werden.**, harmonised standards like EN62061 [3] or EN13849 [4] will be relevant. These standards are describing requirements on how to design and implement safety ECUs for mobile machinery.

For system designers it is important to observe the ProdSG [2]. If safety requirements of one harmonized standard are achieved, Prod SG [2] is fulfilled.

That is very important, because in mobile machinery you have to control potential hazards to human and machinery, too.

Used Terminology

After a successful risk assessment, you obtain, depending on the applied standard, either a performance level (PL) according to EN13849 [4] or a Safety Integrity Level (SIL) according to EN62061 [3].

For comparison between different risk measures, refer to Figure 1. Generally speaking, the higher the number or letter, the more must be contributed to risk reduction.

When using the terminology safe, requirements of the safety level for SIL 2 and/or PL d. are observed.


EN 61508 EN 62061	EN ISO 13849	see EN ISO 13849
-	PL a	Low contribution to risk reduction
SIL 1	PL b PL c	
SIL 2	PL d	
SIL 3	PL e	High contribution to risk reduction
SIL 4	-	

Figure 1: Compare of risk grades

The external certification agency is a further assurance to meet the high demands.

When using the terminology standard, internal corporate guidelines are applied for developing software components.

Standard functions are functions that fulfill functional requirements of the overall system. An error of a standard functionality could lead to a machine downtime.

If executed standard code does not affect the execution of safety relevant code, it can be identified as non-interfering.

Use-case of non-interference software

If you want to develop a highly flexible, modular, and complex safety control system, it is already evident in the planning phase, to offer solutions that provide non-interfering standard code in communication drivers like CANopen, and J1939, as well as in libraries for Mainboard (MB), and Expansion Board (EB).

A further difficulty is that concrete applications of ECUs are not known. That is the reason, why we have to find a general solution for components with non-interfering software on a single CPU system.

Techniques for achieving non-interference between software components on a single CPU

The standard EN61508-3 [6] describes solutions to control the non-interference between standard and safety functions.

The requirement 7.4.2.9 of EN61508-3 [6] states, that software components certified for different SIL ratings, can be executed on a single CPU:

The requirement of EN61508-3 provided that: *'Where the software is to implement safety functions of different safety integrity levels,*

(S) then all of the software shall be treated as belonging to the highest safety integrity level, unless adequate independence between the safety functions of the different safety integrity levels can be shown in the design.

It shall be demonstrated either

(T) that independence is achieved by both in the spatial and temporal domains, or

(W) that any violation of independence is controlled. The justification for independence shall be documented.'

(S) The entire module functions are certified according to EN61508-3 [6] SIL2, this means an extremely high burden of certification as well as a very large effort in software maintenance for example on changes in non-interference software components.

(T) Independence through spatial and temporal separation. With this method, the absence of interference can be ensured. In that case requirements for non-interference of software components can be fulfilled.

(W) Control, if breaching independence (non-interference), we do not know for which applications our ECUs are used for. Based on this fact, we cannot fulfill this requirement.

Integrating non-interference code as it is described in (T), leads to reduced certification and maintenance work and reduced expense of application development.

Through the combined separation temporal as well as spatial, a controlled flow of data between functions which have different safety significance, can take place.

Technical concept

A 32bit CPU (TriCore) was already been defined as an embedded platform. We have a single CPU architecture, so the safety-related and standard functions require the same resources such as CPU, ROM, and RAM. Merely a Memory Protection Unit (MPU) is provided by Infineon. A complete temporal and spatial separation is not given by the TriCore and so a own designed mechanism is needed.

In addition, the design needs to meet the safety requirements, according to the relevant standards.

First question is how to get a temporal separation? It is not given only by using a TriCore processor. Therefore, we need to consider the system architecture of the ECU. We have provided a watchdog controller at our HW design of the ECU. Additional to the watchdog controller, a task system is needed to prioritize the available tasks.

Second question is how to get a spatial separation? The spatial separation can be ensured through a proprietary memory protection mechanism.

Figure 2 shows the software system architecture, freely programmable in C. The green block, denoted as Safety Layer API is presenting the memory protection layer.

Based on the C application level the safe and standard functions will be conducted through the Safety Layer (SL) API. After correct execution of the code, all functions are passed to the Hardware Abstraction Layer (HAL). Each HAL function has its own SL function.

Before executing HAL functions, the additional safety checks and MPU configurations are executed in the SL-API. The illustrated STW task system has the same features like an RTOS task system, which is used by the application code. The task system is elementary needed for the temporal separation of safe and standard code.

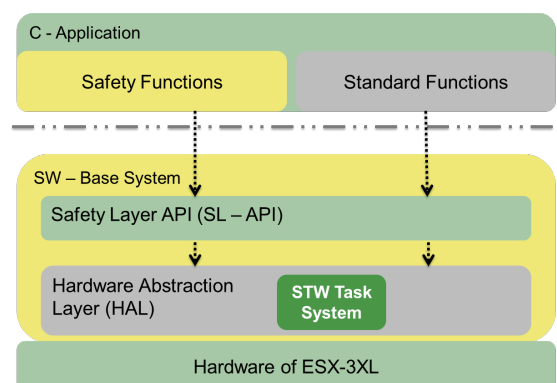


Figure 2: System architecture

The linkage of the STW task system with the safety layer is shown in Figure 3.

Global or static functions and data can be stored in three different memory areas with different security levels and each level has different memory access rights.

- ✓ System level
- ✓ Safety level
- ✓ Standard level

The system level is only reserved for internal BIOS functionalities and has access rights to almost all memory areas and CPU registers. All Interrupts will be executed in the system safety level. System functions have Read/Write access to system, safety and standard data.

The safety level has access to protected global data and also access to application data that affects safety relevant functionality.

Safety relevant data must be protected against access from the standard components of the application. Code that will be executed in this safety level, usually implements safety relevant functionality (for example handling of ECU outputs) and should undergo test and documentation procedure of the required safety standards. In this level you will have read access to all memory areas and write access to protected and standard data. System data cannot be accessed from this level.

The standard level has access to application data that is not used for safety relevant functionalities. Code executed in this standard level has no write access to protected and system data. Therefore it can be changed without affecting safety relevant functionality. Standard functions have read access to all memory areas, but write access only to standard data level.

The task system has two types of mechanism, safe tasks and standard tasks.

Managing different code classes and granting write permissions are its key properties. A task system also administrates available memory.

Task stack and static data of safety relevant tasks are write-protected; this is the memory protection unit (MPU).

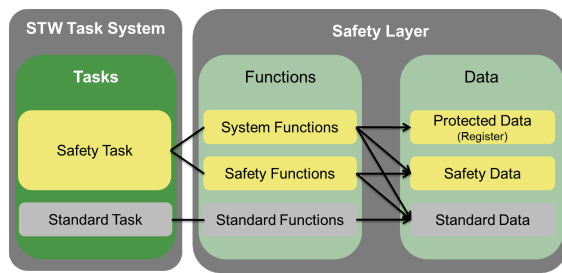


Figure 3: Interaction of memory protection and STW task system

Hardware has to be initialized the right way to meet safety and standard requirements. For example an output defined as safe, shall be only accessed by safe functions. Following properties are known by the safety layer:

- ✓ SL-API is an additional software layer, which is simply placed on top of the HAL.
- ✓ SL-API is directly coupled to the STW task system and uses the MPU of the TriCore CPU to refuse writing on global data, if the executed function does not have the required writing permission for the task.
- ✓ SL-API allows the application to interact between safe and non-interfering standard functionalities.

SL-API is essential to implement RTS Codesys Safety SIL2 on TriCore platforms with the existing single CPU system.

Codesys V3

Codesys is a software platform especially designed to fulfill different requirements of modern industrial automation projects. The IEC 61131-3 development system is the heart of Codesys.

It offers integrated, user-friendly solutions to support development tasks. In Codesys several PLC programming languages are

available. In according to the relevant standards EN62061 [3] and EN13849 [4], the languages can be divided into 2 categories:

- LVL - Low Variability Language
 - Function Block Diagram (FBD)
 - Ladder diagram (LD)
- FVL - Full Variability Language
 - Instruction List (IL)
 - Sequential Function Chart (SCF)
 - Structured Text (ST)

Codesys Safety SIL2 is based on Codesys V3.

The system architecture of Codesys runtime system is shown in Figure 4. Compared to C system architecture, the SW - Base System is now the lowest software layer in Codesys RTS. On SW – Base System the STW-Shell is placed.

The Codesys Safety SIL2 Core is embedded into the STW-Shell.

The STW-Shell provides the interface between Codesys Safety SIL2 Core, SW - Base System, and the IEC - Application. Safety and functional requirements of SW - Base System, Codesys Safety SIL2 Core, and IEC - Application must be implemented in the STW-Shell. It is not allowed to implement Codesys RTS specific requirements into the SW - Base System.

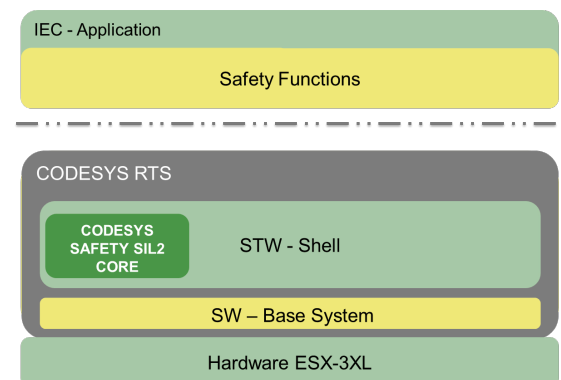


Figure 4: System architecture of Codesys RTS

Currently the entire IEC - Application has to be developed according to the harmonized standards.

What are the benefits with Codesys Safety SIL2?

In accordance with TC5 [7], chapter 4.1, safety-related applications can be subdivided into three parts:

- Basic level (according to [3] Fehler! Verweisquelle konnte nicht gefunden werden.)
- Extended level (according to [3])
- System level (according to [5])

The programming languages for application development are limited. If implementing an IEC – Application in basic or extended level, TC5 [7] allows the programming languages FBD and LD.

FBD and LD are languages with limited scope (LVL), according to EN62061 [3] chapter 3.2.49 are only the graphic languages of IEC61131-3 [8] qualified. However, if suitable coding guidelines applied, almost any programming language can be classified as LVL and therefore be used for basic or extended level. Coding guidelines used for LVL classification have to be approved by safety assessment. Are the coding guidelines approved, certification can take place in accordance with EN62061 [3], without regarding EN61508 [5].

In basic and extended level you have to consider the linking restrictions, and as an additional requirement only pre-certified modules are allowed.

The extended level is usually used for the preparation of pre-certified function libraries. The advantage of both levels are, that tests according to EN62061 [3] are required.

The system level allows the use of FVL and especially ST. But due to EN 62061 section 6.11.3.1.1 there is a reference to EN61508 [5] and must therefore be certify according to EN61508 [5].

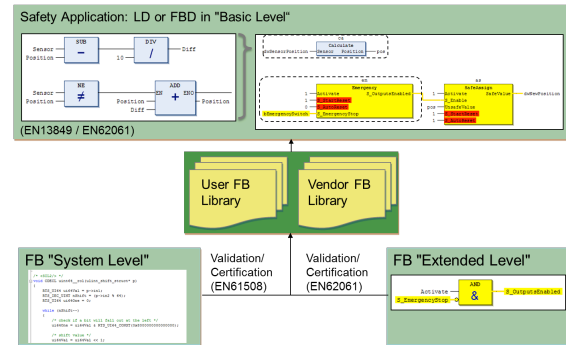


Figure 5: Program structure

In Figure 5 the program structure of a safety IEC - Application is shown.

As mentioned, complex libraries can be pre-certified by using the system level, by using the extended level, and in accordance to the used harmonized standards. After their certification, they can be easily merged and categorized as basic level. This means that only integration tests are required.

Another advantage of pre-certified libraries are, they can be used for other safety related IEC – Applications. Thereby the expenses for other application projects can be reduced.

Figure 6 shows the example of a graphical data flow for developing an IEC - Application. The data flow shows comprehensible the separation between safe and standard functions. Reviews became much more user-friendly.

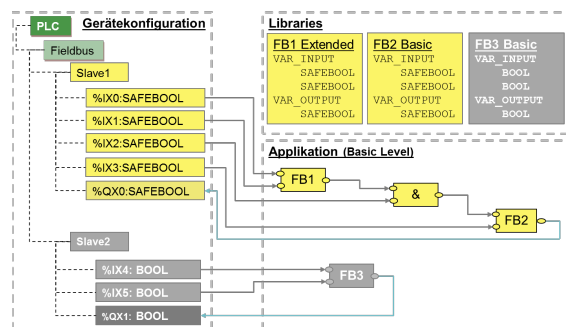


Figure 6: Example data flow

Codesys offers in addition to the compiler checks, a tool to check the source code based on predefined rules. To obtain information on potential problems, errors can be detected and already removed before the field tests will begin.

Summary:

By the developed mechanisms between the safety layer and the STW task system, we can ensure that the temporal and spatial separation of software functions can be guaranteed.

Therefore we have met the requirement 7.4.2.9 from the standard [6], to provide the mechanism of non-interference software.

In C - Applications this mechanism can be used to design applications in compliance with the safety requirements.

This allows the system designer during the entire machinery lifespan to focus on safe software modules. Standard components will not have any influence to the safe components (non-interference).

By the memory protection Codesys Safety SIL2 could be implemented on one single CPU.

The entire IEC code must be designed after the safety aspect of the relevant harmonized standards.

However, a simplified certification can be reached, if pre-certified libraries are used. At basic level designed IEC – Applications only integration tests for validation are required. The provided Codesys analysis tool ensures that applied predefined rules are checked.

Kai Daniel Niestroj

Sensor-Technik Wiedemann GmbH

Am Bärenwald 6

D-87600 Kaufbeuren

Phone: +49 8341-95 05-0

Fax: +49 8341-95 05-55

E-Mail: info@sensor-technik.de

Website: <http://www.sensor-technik.de/>

References

- [1] Machinery directive 2006/42/EC
- [2] Produktsicherheitsgesetz ProdSG
http://www.gesetze-im-internet.de/englisch_prodsg/englisch_prodsg.html#p0023 [Nov. 2011]
- [3] EN62061 [Okt. 2005]
- [4] EN ISO 13849 Safety of machinery – Safety-related parts of control systems [Jul. 2007]
- [5] EN61508 Functional safety of electrical/electronic/programmable electronic safety-related systems [Apr. 2010]
- [6] EN61508-3 Functional safety of electrical/electronic/programmable electronic safety-related systems Software requirements [Apr. 2010]
- [7] PLCopen - Technical Committee 5 Safety Software - Technical Specification Part1 - Concepts and Function Blocks [Jan. 31, 2006]
- [8] IEC61131-3 Programmable controllers – Part 3: Programming languages [Dec. 2002]