Separating non-safety and safety software functions in ECUs

Kai-Daniel Niestroj

Tadano Faun GmbH utilize the ESX-3XL Safety controller with Codesys Safety SIL-2.

Author

Kai-Daniel Niestroj

Sensortechnik Wiedemann (STW)
Am Bärenwald 6
DE-87600 Kaufbeuren
Tel.: +49-8341-9505-0
Fax: +49-8341-9505-55
info@sensor-technik.de

Introduction

In mobile machines often non-safety and safety functions are required. In order to avoid separate ECUs, OEMs demand to use one single ECU for both kinds of tasks. The EN/IEC 61508-3 international specifies technical measures to prevent negative side-effects of software and describes how to reduce the effort for certification of non-safety software modules running in a safety ECU.

The EN/IEC 61508-3 international standard requires that in case of different safety integrity levels, all software parts "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." This can be achieved either by spatial and temporal domains or by double-checking the independency against violations. Of course, the justification needs to be documented.

Using the Tricore micro-controller

The 32-bit MCU by Infineon provides one core, which is used for the safety and non-safety software. Safety and non-safety software are using the same CPU and the same on-chip memory (ROM and RAM). Merely the micro-controller features a memory protection unit (MPU). The MCU does not provide a complete temporal and spatial separation of its resources. This means, this functionality has to be added by the ECU design.

A watchdog controller and a task system to prioritize the available tasks are needed. These add-ons fulfill the temporal separation requirements. In our ECU design, the spatial separation is ensured through proprietary memory protection mechanism.

Figure 1 shows the software architecture for the C programming application. The green block, denoted as Safety Layer (SL) API, is presenting the memory protection layer. Based on the C application level the safe and non-safe functions are conducted through the SL-API. After the execution there, 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 described STW task system, which is used by the application code, has the same features as an RTOS (real-time operating system) task system. The task system is fundamentally required for the temporal separation of safe and non-safe code. The linkage of the STW task system with the safety layer is shown in Figure 2.

Global or static functions and data can be stored in three different memory areas with different security levels (system, safety, and non-safety) and each level has different memory access rights. The system level is reserved for internal system functionalities and has access rights to 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 non-safety data. Safety-related data must be protected against access from the non-safe components of the application. Code that will be executed in safety level, usually implements safety-related functionality (for example handling of safety ECU outputs) and must be

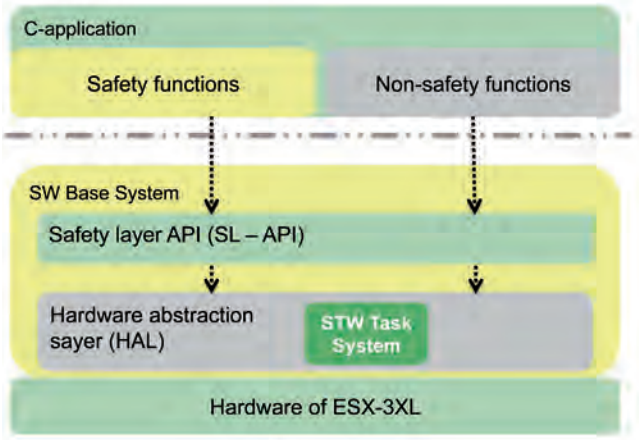

Figure 1: Software architecture

verified and documented according to the required safety standards. This level will have read access to all memory areas and write access to protected and non-safe data. System data cannot be accessed from this level.

The non-safety level has access to application data that is not used for safety-related functionalities. Code executed in this level has no write access to safety and system data. Therefore it can be changed without affecting safety-related functionality.

The task system differs two types of mechanism, safe tasks and non-safe tasks. Managing different code classes and granting write permissions are the key properties. The task system also administrates available memory. Task stack and static data of safety relevant tasks are write-protected; this is achieved by means of the on-chip MPU.

Safety requirements have to be taken into consideration for the configuration of the ECU hardware. For example, an output defined as safe, shall be only accessed by safety functions. The safety layer features the following behavior:

- 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 write permission for the task;
- SL-API allows the application to interact between safety and non-safety functionalities.

The SL-API was needed for the Codesys Safety SIL-2 runtime system (RTS) for the ESX-3XL modular Safety controller.

Using Codesys Safety SIL-2

Codesys by 3S is a software platform especially designed to fulfill different requirements of industrial automationprojects. TheIEC61131-3 ▷

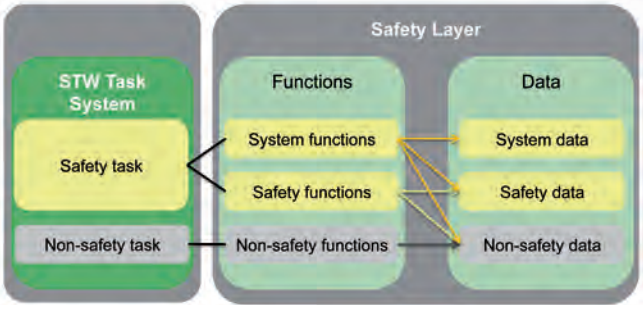

Figure 2: Interaction of memory protection and STW task system

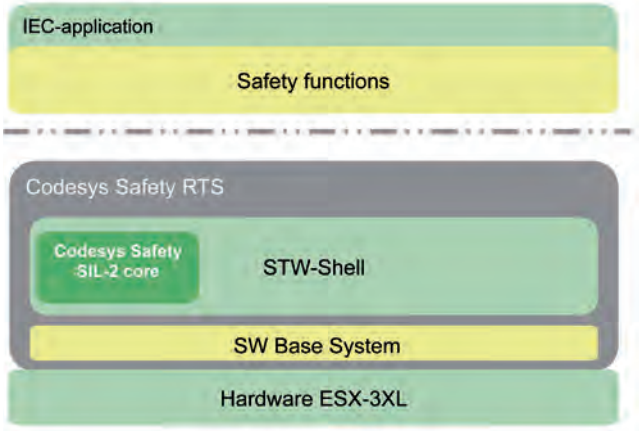

Figure 3: System architecture of the Codesys Safety RTS

development system is the heart of Codesys software approach. It offers integrated, user-friendly solutions to support development tasks. In Codesys several PLC programming languages are available. In accordance to the relevant standards EN 62061 and EN13849, the languages can be divided into two 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 SIL-2 is based on Codesys version 3. The system architecture of Codesys RTS is shown in Figure 3. The SW Base System is the lowest software layer in the RTS. Above the SW Base System the STW Shell is placed. The Codesys Safety SIL-2 Core (PLC core) is embedded into the STW Shell. The STW Shell provides the interface between the PLC core, the SW Base System, and the PLC application program (IEC Application). Safety and functional requirements of the SW Base System and the PLC core are considered in the STW Shell. The Codesys RTS specific requirements are all implemented in the STW Shell. The entire PLC application program has to be developed according to the harmonized safety standards.

In accordance with the PLCopen (nonprofit trade organization for IEC 61131) guidelines the safety-related applications can be subdivided into three levels:

- Basic level
- Extended level
- System level

When implementing PLC application software in basic or extended level, only graphic languages (LVL) are allowed including the FBD and LD languages. Non-graphical languages (FVL) are not qualified (for details see EN/IEC 62061 chapter 3.2.49). However, if suitable coding guidelines are applied, almost any programming language can be used for the basic or the extended level. Such coding guidelines have to be approved by safety assessment. If the coding guidelines are approved, certification can take place in accordance with EN/IEC 62061.

In the basic and the extended level the user has to consider 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 is that tests only according to EN/IEC 62061 are required. The system level allows the use of FVL and especially ST languages.

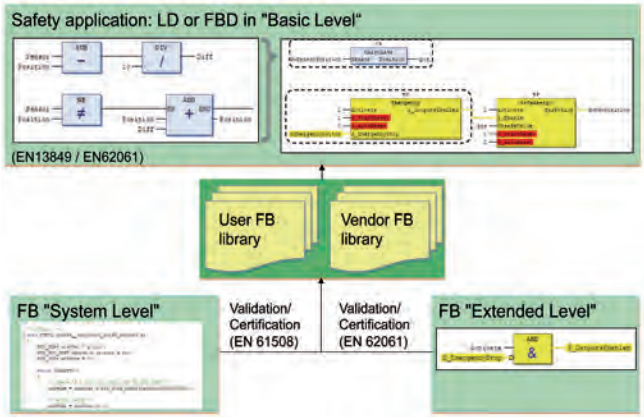

Figure 4: Program structure

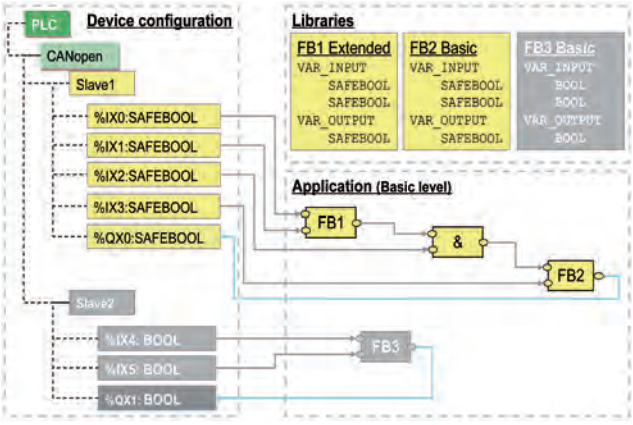

Figure 5: Data flow example

Links

www.3s-software.com

www.infineon.com

www.plcopen.org

www.sensor-technik.de

Figure 6: The ESX-3XL modular safety controller implements Codesys Safety SIL-2

If the system level is applied, the EN/IEC 62061 requires that the software must be certified according to EN/IEC 61508.

In Figure 4, the program structure of a safety PLC application program is shown. As mentioned-above, complex libraries can be pre-certified by using the system level or the extended level. After certification, they can be merged and categorized as basic level. This means that only integration tests are required. Another advantage is the reusability of pre-certified libraries. Thereby, the expenses for other application projects can be reduced.

Figure 5 shows a graphical data flow of an IEC application (PLCopen program). The data flow shows the separation between safe and non-safe functions in an easy and comprehensible format.

Codesys offers in addition to the compiler checks, a tool to check the source code based on pre-defined rules. Potential errors can be detected and already removed before the field tests begin. ◀

Summary

Sensor Technik Wiedeman GmbH (STW) has many years of experience in developing safety ECUs. A new mechanism for ESX-3XL ensures temporal and spatial independence of SW functions for different risk ratings. This mechanism can be used in applications, where safety requirements have to be fulfilled. System designers have to focus only on safety software modules during the entire machinery lifecycle. Non-safety software will not have any influence on safety functions. Codesys Safety SIL-2 is now available for ESX-3XL customers. An entire IEC-Application has to be designed in accordance with the safety standards. To make it easier for your application to reach certification, pre-certified libraries can be used. STW offers pre-certified libraries. For on basic-level designed PLC software only integration tests for validation are required.