White Paper[PRELIMINARY- Sept. 15, 2020]

A guideline to Safety Certifications in the Industrial, CAV and Heavy-Duty segment.

A practical roadmap to the use of Infineon Aurix iLLD and Hightec PXROS in the IEC 61508 Safety context.

Introduction

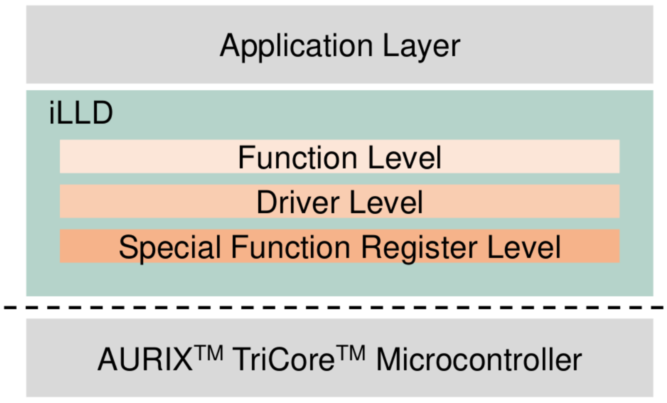
The purpose of this document is to explain the process of integrating iLLDs Aurix software drivers to enable a project compliant with IEC 61508 (or other similar Safety International Standards referred by the Construction, Agriculture, Off-highway and Heavy duty industries).
Similarly, the same approach has been exploited for Automotive “non-AUTOSAR” designs.

Infineon Aurix is a very powerful CPU capable of solving projects, both in the automotive and industrial, fields, due to its sophisticated peripherals. Complex architectures can be designed based on the configuration and combination of these devices.

To simplify the use of peripherals Infineon provides a set of related drivers, collected in two types of libraries: **iLLD** and **MCAL**. We will explain briefly what these drivers are, what are the basic principles of IEC 61508 involved in the integration, and when iLLD should be suggested for specific project types.
Finally we will list benefits and constraints of the proposed approach.

What is iLLD

iLLD stands for “Infineon Low Level Driver” package, and provides functions, drivers and structures that allow three levels of abstraction. The iLLD is a single **open source software** package offering driver functions for the AURIX™ family.


Application Layer
iLLD
Function Level
Driver Level
Special Function Register Level
AURIX™ TriCore™ Microcontroller

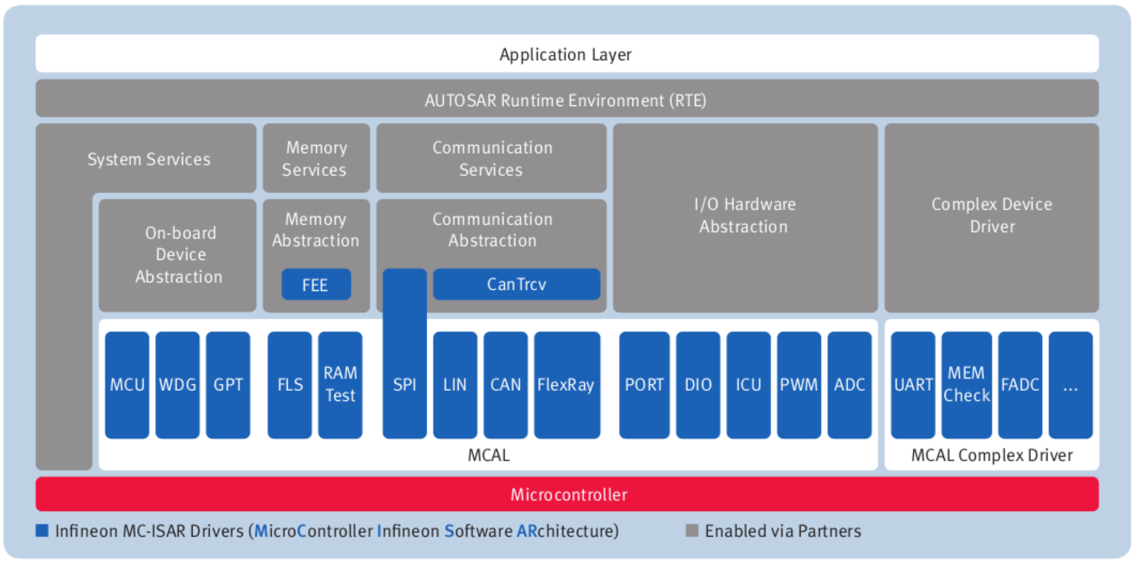
- Special Function Register Level: Access register bitfields by name
- Driver Level: Combines register configurations to be easily executed using function calls
- Function Level: Initialization, configuration, start and stop of any AURIX™ peripheral

In summary ,iLLD exposes an API (Application Program Interface) that enables the use of peripherals.

What is MCAL

Infineon provides the low-level drivers based on the **AUTOSAR** Microcontroller Abstraction Layer (MC-ISAR) standard, designed for the automotive world relying on ISO 26262 safety standard.

Below the MCAL structure is reported. It is comparable with iLLD in terms of goals, but it clearly shows an higher level of complexity.



MC-ISAR	MicroController – Infineon Software ARchitecture
MC-ISAR:	MCU, WDG, GPT, SPI, PORT, DIO, ICU, PWM, ADC
MC-ISAR COM Basic:	CAN, CanTrcv, LIN
MC-ISAR COM Enhanced:	FlexRay, Ethernet
MC-ISAR MEM:	FLASH, FEE
MC-ISAR MCAL CD:	UART, MEMCheck, FADC, ect. for TriCore™

Designing in IEC 61508 context

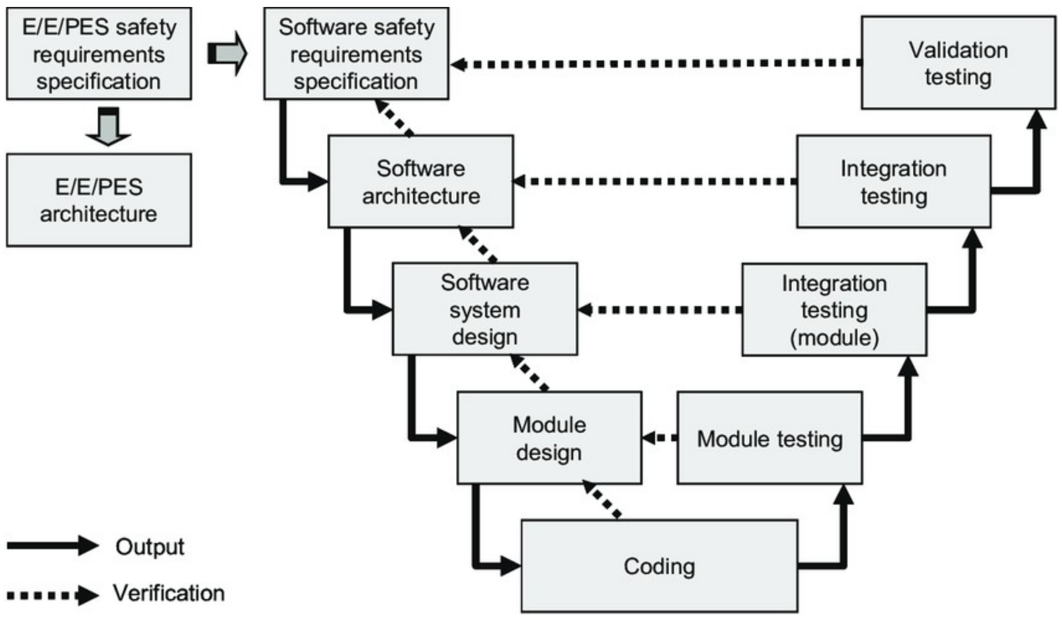
The IEC 61508 is a general international standard: **Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems**.

This standard provides a well-defined development process aimed to design electronic products that work correctly (or fail in a controlled and predictable way, for the matter). It is widely adopted in the Industrial sectors, and a number of derived international standards are related to it. This paper takes IEC61508 as a reference, but the principles explained here can be applied to most non-AUTOSAR electronic designs including Automotive.

A team must decide to include a driver library by evaluating its impact in terms of safety.
During the architecture phase, when all Safety Functions are "translated" into Safety Requirements, it has to be estimated whether it is better to include a "certified" library, (i.e. MCAL) or a non certified one (i.e. iLLD).

As a further step, in the latter case the team has to evaluate: is it required to certify the new library, or rather avoid this by encapsulating it in a "software architecture" that intrinsically protects it?
These are some recurrent questions emerging during the first design phase in a Safety context.

The proposed iLLD integration approach follows the last path (encapsulation) which, at the end, takes also in consideration the cost impact.



ILLD integration with PXROS-HR: task encapuslation

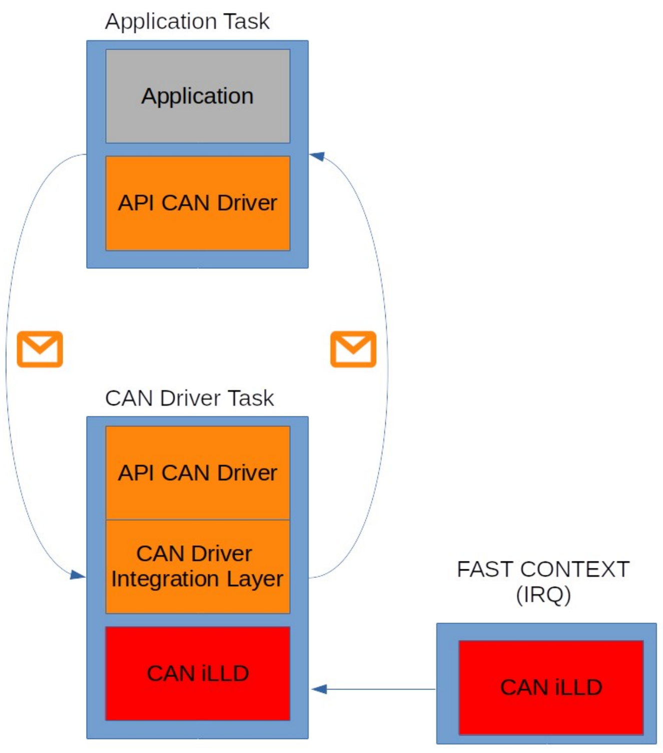
PXROS-HR (Portable eXtendible Realtime Operating System - High Reliability) is a realtime operating system for embedded systems.

In comparison to other realtime operating systems, PXROS-HR has some special features, such as:

- Hardware-assisted memory protection by a Memory Protection Unit (MPU)
- Permission concept for Tasks
- Object-based architecture
- No interrupt locks within the kernel

- Encapsulated Tasks, assisted by hardware memory protection
- Possibility of reloading and debugging Tasks during runtime
- Message-based data interchange between Tasks

This paper focuses on **Encapsulated Tasks, assisted by hardware memory protection**, because by encapsulating a iLLD driver inside a task we are isolating it from other processes (tasks). In case of failure, the system detects it and can define the action for that event. This is why we are going to include an iLLD driver (for instance the CAN driver) in a task by interfacing it with the application.



According to the presented guideline, the iLLD **Driver Integration Layer** and correspondent **API** must be specifically developed in function of the driver itself and taking into account the application requirements.

For example it shall contain the "send" and "receive" functions that the application will use. On the other side, the hardware configuration and any function required to: capture a message, store it, check it, send the result, are commissioned to the iLLD itself.

Further, the application is separated by the driver (different task), and it is allowed to access to driver's data only via intrinsic PXROS safety mechanism such as the **mailbox**.

This described approach (**task encapsulation**) permits to take under control the driver.

- Orange: Bluewind expertise engineering
- Red: Infineon
- Blue: Hightec PXROS-HR

Safety implication and guideline

We start for example from the CAN driver. The design of the CAN iLLD **Driver Integration Layer** and **API** depends on the safety requirements established during the SSR (Software Safety Requirements) specification. For example, how will the application detect if the CAN channel isn't working properly? Which kind of function is needed? And how can one test it?

These questions occur during the Design and Module Specification, where the Safety Manager must decide if these constraints can be satisfied by using the iLLD specific driver "as it is" (so as a "**black box**") or if it necessary to proceed to its validation (i.e.: applying unit test for all functions, applying coding rules, MISRA rules,...).

That decision should be made after an FMEA analysis of the driver, which gives the rational of the selected test of the black box.

As a first step, it is necessary to identify:

- which are the fail mode of the driver
- if the task encapsulation covers the worst case
- a set of test for the driver considered a **black box** item

As further step one must verify the task driver interface (**API** CAN Driver in the picture above).

Usually this steps are required:

- static analysis and code coverage of the integration layer (i.e. codestyle and MISRA rules)
- execution of unit tests for the driver integration layer (an automatic test tool is recommended)
- integration test of the interface (API Driver), for instance by using the fault injection technique

Conclusions: benefits and constraints

In general the use of iLLD in a IEC 61508 development has to be evaluated according to the SIL (Safety Integrity level), to the safety functions, and to the project investments trade-offs.

Obvious immediate benefits are low or zero costs for driver licenses.

Much more meaningful, is the opportunity to design simpler, more maintainable code, in a less expensive project environment. This is due to the lack of the complexity of the AUTOSAR framework.

On the other hand, the integration of a non certified driver set like iLLD may imply more engineering work for the analysis, documentation and test process.

The choice between the iLLD, MCAL or other libraries is not normally a black-or-white decision, but there is a suggested process to follow in order to take the best direction.

The parameters to consider are:

- **Feasibility**: starting from the Safety Requirements, can one assume that a driver is a **black box** inside a protected task?
- **Time-to-market**: is it better to buy a certified library in order to speed up the safety analysis?
- **Coding**: is it better to use a complex firmware (certified) or something more under the developer control?
- **Cost**: which is the break even? Generally this parameter depends on the number of safety drivers to be taken into account , the expected volume production, the team skills (AUTOSAR, Safety, stc).

As a conclusion, based on past projects and current experiences, the use of iLLD is a good solution that can save time and cost, in the context of industrial projects (or automotive projects non-AUTOSAR).

References:

- Infineon TriCore_neu.pdf
- Infineon iLLD manual
- AURIXTM TC2xx Microcontroller Training.pdf
- IEC 61508-3
- Hightec PXROS manual

About Bluewind

Bluewind, an independent engineering company, provides world-class products, engineering and software solutions in the domains of electronics, safety critical applications, and connected devices. As a qualified PDH partner for AURIX™–32-bit multi-core TriCore. Bluewind is actively involved in designing next generation products using Infineon technologies in the Industrial, Automotive and Medical industries.

About Infineon

Infineon is a globally leading semiconductor player. AURIX™ is Infineon's brand new family of microcontrollers serving exactly the needs of the Industrial & Automotive industry in terms of performance and safety. Infineon's products are found everywhere today, and together with our customers, we are increasing the safety and the quality in the Industrial, CAV and Heavy-Duty segment.

About Hightec

HighTec EDV Systeme is a privately owned company since establishment in 1982 and the world's largest commercial open source compiler vendor. HighTec ensures independence for the future and the most reliable and secure tools for embedded software development. The HighTec compiler is portable and always available for the latest chip revisions for our supported architectures ahead of general release.

Bluewind Srl

Via della Borsa, 16/A - 31033
Castelfranco Veneto (TV) - Italy
+39 0423 723431 - info@bluewind.it