WHITE PAPER

Safety standards in the ARM® ecosystem

Developing systems for safety-related applications requires support on all levels of design, from basic hardware to application software

Lauri Ora January 2015

Introduction

Safety is an increasingly important topic for system developers and there is an ever widening range of relevant safety standards that system developers need to be aware of. The increasing use of electronics in cars, medical equipment and the Internet of Things (IoT) has led to new safety standards and new demands on developers of hardware and software. The increasing connectivity means devices must also be secure. Providing justification for the safety of an unsecure connected device will be extremely difficult, as potential attacks via the network connection can directly impact the safety functions. This means that both safety and security standards have to be considered for next generation designs.

Over the last few years, microcontroller designers have been providing additional features specifically to help develop safety-related systems, and that in turn is helping software developers tackle the range of standards they have to comply with. Support for virtualization in the hardware allows developers to use existing, proven code alongside new developments and new standards without compromising the safety of the overall system.

This paper will first explore some of the key industries with safety and security concerns. Following this, a brief introduction to some of the applicable functional safety and security standards is included. Finally the ARM® architecture and ecosystem support for functional safety and security applications is discussed.

Medical applications

The medical equipment industry has been adopting electronics based safety technologies with increasingly complex basic designs. Infusion pumps and pacemakers that keep patients alive are increasingly using safety-related semiconductor devices at the heart of their designs. As systems become connected — either to share data or for remote operation in 'telehealth' applications — the safety and security elements of the designs become even more important. This is resulting in a wide range of safety standards that have to be considered across the range of medical equipment development, alongside stringent approvals processes from organizations such as the FDA in the US, or the notified bodies operating within the framework of the European Medical Device Directive.

The IEC 60601 medical electrical equipment standard covers equipment such as EEG monitors, IV pumps, imaging systems, ECG devices, vital signs monitors, and other devices that connect directly to a patient. Devices and systems not directly connected to the patient are covered by IEC 61010, including measurement, control and laboratory systems.

For medical devices and systems, comprehensive risk management is an integral part of ensuring patient safety. ISO 14971 requires identification of hazards for all operating modes and fault scenarios. The hazard identification process needs to be comprehensive, including electronics and software within the device or system. Analysis of identified hazards is typically done through the use of a risk matrix, which provides a mapping from expected severity of harm and probability of occurrence to the overall risk associated with the hazard. The resulting risk for each hazard is then used to determine required countermeasures, which can be design-based, operational, or documentation related.

Software in medical devices and systems is also regulated with two different testing approaches. IEC 60601-1 Annex H treats software as a black box component of the system, with the functionality qualified and tested as part of the overall system.

For devices with more complex software IEC 62304 also applies. The standard classifies software in three categories, ranging from the less critical Class A, to Class C where a failure could result in death or serious injury. The standard outlines requirements at each stage of the software development lifecycle and defines the minimum activities and tasks that need to be performed to provide confidence that the software has been developed in a way that reduces the risks from potential malfunctions caused by software errors to a tolerable level.

Medical devices also often utilize legacy software, often referred to as software of unknown pedigree (SOUP) and rely on third party software and drivers for function such as internet connectivity, USB communication, or file management. These software elements also have to be tested and integrated as part of an IEC 62304 compliant process which can be complex and time consuming.

Industrial applications

Areas of industrial equipment design have had to deal with safety concerns for many years. Process equipment controlling critical operations that can fail, causing explosions, fires, environmental disasters and even fatalities have to be designed with safety and security in mind. One of the ways to address these risks is to use a risk based approach, which drove the development of the functional safety standards such as IEC 61508 and IEC 61511.

However, as more and more industrial equipment becomes connected to networks and to the Internet, there is more focus on the combined safety and security of these systems. Water pumps or electrical systems failing, either by accident or by malicious attack, can cause widespread disruption that is both costly and potentially disastrous. Ensuring that industrial systems are safe and secure is an increasingly vital consideration.

Fortunately the safety standards that have evolved over the last couple of decades have taken a broad approach to safety and so are equally applicable in today's networked world.

Automotive applications

Automotive design has been the driving force for safety in integrated silicon devices over the last 20 years. Standards such as ISO 26262 and AUTOSAR have been developed to ensure that systems, software and hardware can be developed and operated in a way which allows risks related to malfunctions to be mitigated and controlled.

Progress in automotive driver assistance capabilities and onward to semi-autonomous driving, whether by controlling brakes, steering or other safety-related parts of the car, is limited by the ability to demonstrate the safety of the electronics systems. Vehicle manufacturers and their suppliers are focusing their efforts to develop and validate these systems such that appropriate safety, quality, and reliability attributes can be demonstrated. Recalling a range of vehicles can be incredibly costly and severely damaging to the brand of the car maker. This is driving an understanding that component cost is less important than ensuring the safety systems are correctly architected and developed, giving semiconductor vendors the opportunity to innovate. This experience and innovation can then be transferred to other applications in medical and industrial markets.

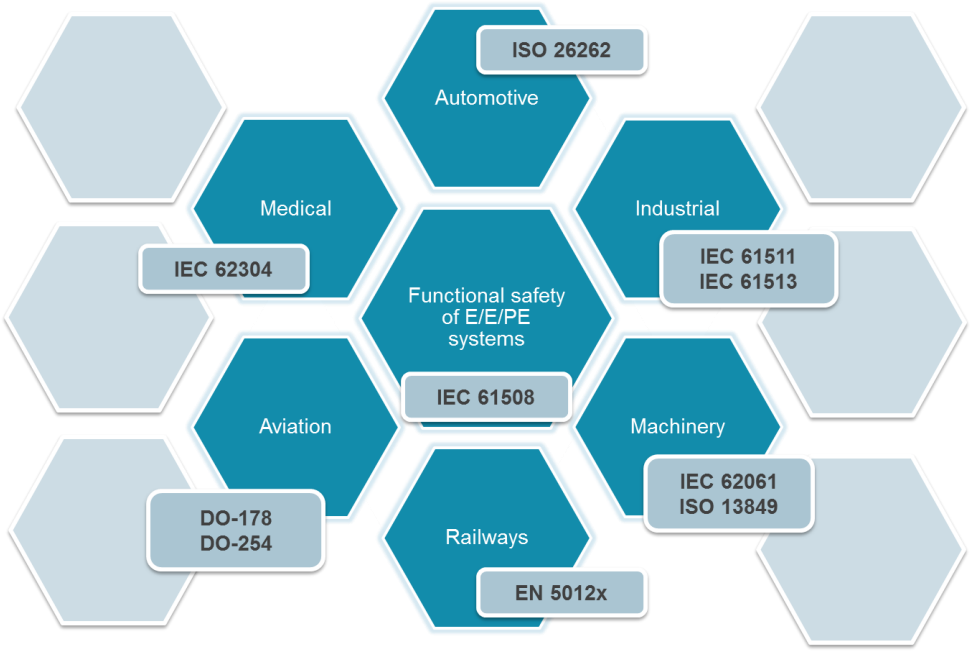

Figure 1: Main functional safety and related standards

IEC 61508 – Functional safety for electrical, electronic, and programmable electronic devices

IEC 61508 is designated as a basic safety publication in the IEC standardization framework, meaning that it defines generic requirements for functional safety across applications and industries. The standard was originally published in 2000, with the current second edition being from 2010.

The standard covers the complete safety life cycle through 16 phases in three groups covering the analysis, implementation and operation of a system. At the heart of the standard are the concepts of risk and safety function. Risk is defined as a function of the likelihood of a hazardous event and the severity of its consequences. The risk associated with the hazardous event is then reduced to a tolerable level by applying safety functions either in hardware and software or in other technologies, although only the electronic, electrical, and programmable electronic elements are covered by the standard. The risk management therefore relies on the functionality of such safety functions, hence 'functional safety'.

While IEC 61508 accepts that zero risk can never be reached, intolerable risks must be reduced. To do this, systems and operations must be designed for safety from the beginning. Engineering safety as an afterthought generally results in less effective and more complex solutions with an increased cost, compared to systems where appropriate safety engineering principles have been considered from the beginning.

ISO 26262 – Functional safety for road vehicles

ISO 26262 is an adaptation of the IEC 61508 safety standard aimed at automotive electric and electronic systems. It defines functional safety for automotive equipment and applies throughout the lifecycle of all automotive electronic and electrical safety-related systems. After several years of development by experts mostly from the automotive industry it was published in November 2011, followed by rapid adoption within the industry. The standardization activities to develop the second edition will start in late 2014, with the eventual publication of the second edition still being several years away.

The standard consists of nine key normative parts, and like IEC 61508, is a risk-based safety standard. Hazards caused by malfunctioning behavior of electronics or software need to be comprehensively identified, and the risks associated with each hazard need to be assessed using the method described in the standard. Safety measures are then defined to avoid or control systematic failures and to detect or control random hardware failures, or mitigate their effects. The robustness of safety measures is determined by the level of risk, as quantified by automotive safety integrity level, ASIL.

Through the use of ASILs, the ISO 26262 provides an automotive-specific risk-based approach for determining the safety requirements to address identified hazardous events. The ASIL therefore conveys information both about the level of risk, as well as the requirements for technical and process aspects of the designs.

Addressing the identified safety requirements requires a defined safety lifecycle that includes management, development, production, operation, service and decommissioning. The standard allows activities within each lifecycle phase to be tailored to the specific requirements of the project, provided that the tailoring follows the rules specified in the standard, and can be shown not to have a negative impact on the achievable safety integrity.

AUTOSAR

The key software standard for automotive is AUTOSAR (the AUTomotive Open System ARchitecture). This is an open, standardized software architecture jointly developed by automobile manufacturers, suppliers and tool developers. This open standard for automotive systems provides a basic infrastructure to help develop control software, user interfaces and management for all application domains.

The key goals include the standardization of basic systems functions, scalability to different vehicle and platform variants, transferability throughout the network, integration from multiple suppliers, maintainability throughout the entire product life-cycle and software updates and upgrades over the vehicle's lifetime. Applying AUTOSAR’s disciplined approach to software development brings a common understanding of the challenges faced by engineers developing safety-related systems and it enables them to integrate and reuse software programs.

Security

Security also has its own standards such as the Common Criteria defined within ISO/IEC 15408. The Common Criteria is a standard for information technology security evaluation, and is often applied to computer and embedded systems. The Common Criteria defines a framework for specifying security requirements for both functional and assurance aspects.

The security assurance requirements are captured in Evaluation Assurance Levels, EAL1 through EAL7, with EAL7 being the most stringent assurance level. The requirements of EAL involve design documentation, design analysis, functional testing, or penetration testing, with higher EALs requiring increasingly robust processes and methods to be used.

As a generic framework the Common Criteria can be used to define the security requirements in a consistent way, allowing different software and system vendors to communicate security requirements unambiguously. The Common Criteria framework also provides mechanisms to conduct security evaluations, and to audit developed systems against the defined security requirements.

Integration of safe and secure software with regular applications

The increasing need for both safety and security is changing the way systems are designed. One of the key considerations of safety design is separation. How to separate system elements so that the failure or breach in one part of the system does not have negative effects to another part of the system which may be running safety-related or secure functions?

Previously designers used separate processors or even separate system boards to keep safety-related software separate from regular application software. This reduced the likelihood that application software could have an adverse effect on the safety-related software. Designing and developing systems with full physical separation is inherently costly, due to physical and logical constraints such separation requires. Having a mechanism to reduce the level of physical separation is therefore often beneficial.

Increasingly these critical and non-critical elements can be combined in a single chip using multiple processor cores, or even on a single core, that support the ability to maintain that separation through partitions. Different partitions will be separated in terms of resources and timing. Partitioning also allows other software that doesn't need such stringent standard conformance to be added to a design more easily without increasing the hardware complexity.

Virtualization is one key mechanism for increasing the separation between two or more software partitions executing on the same underlying hardware. Virtualization allows the generation of virtual environments, which present the underlying system to different software partitions in isolation. Virtualization can therefore be used to separate safety-related and regular software by allocating software of different criticality to separate partitions. A hypervisor is typically used to control the resource allocation as part of the virtualization scheme.

For automotive system developers, virtualization through the use of hypervisors allows consolidation of electronic control units. Each software system can run in a separate partition as long as the hypervisor

controlling the resource allocation to partitions can sensibly share processing resources by time slicing or event triggering. Different partitions may also be used to run different operating systems, allowing combination of real-time safety-related software to co-exist with a rich operating system such as Linux on the same processor.

For example, a Tier-1 automotive supplier can deliver its carefully tested safety-related software for controlling the power steering. Additional software could be integrated by a vehicle manufacturer, perhaps to include chassis control over a CAN bus, without compromising the power steering. In this case the hypervisor would prioritize real-time interrupts from the steering system and switch execution to it whenever necessary. A necessary precondition for doing this is naturally that the hypervisor software itself is trusted and developed with stringent safety requirements.

Importance of hardware support for functional safety

The development of safety systems requires consideration of both hardware and software. Recent processors have additional features to assist in the implementation of safety-related designs. The ARMv7-R architecture includes features in the architecture such as user and privileged software operating modes with a Memory Protection Unit (MPU) and advanced error management, which help safety-related processor designers. In addition, implementations of the ARMv7-R architecture support lock-step processor configuration, which allows a high diagnostic coverage to be achieved for random hardware faults. These features were a key part of the original ARM® Cortex®-R processor family, starting with the Cortex-R4 in 2005 for automotive systems, industrial control, wireless baseband, hard disk drive controllers and other real-time applications.

ARM's Cortex-R processors implementing the ARMv7-R architecture support high clock frequencies with a deeply pipelined micro-architecture and hardware SIMD instructions for very high performance DSP functions. The processors were designed for fast, bounded and deterministic interrupt response with Tightly Coupled Memories (TCM) local to each core for fast-responding code and data and a Low Latency Interrupt Mode (LLIM) to accelerate interrupt entry. Bounded and deterministic response times are essential for most safety-related applications, as these typically also have strict real-time requirements.

Following on from the success of ARMv7-R architecture for real-time safety applications, the ARMv8-R architecture was announced in 2013. The ARMv8-R architecture profile is still targeted for 32-bit embedded applications, especially in automotive and industrial applications, while bringing a number of improvements over the ARMv7-R architecture.

The most important addition in the ARMv8-R architecture is an enhanced hardware-assisted hypervisor mode which provides an additional privilege level within the processor hardware. This privilege level determines what the virtualized software can and cannot do, with basic user tasks having the lowest level of privilege, zero or PL0. An operating system typically runs with higher privilege at PL1, managing the interrupt controller and configuring the attributes for regions of memory and peripheral address map to police user task accesses. Such policing is performed in a processor's hardware by the MPU.

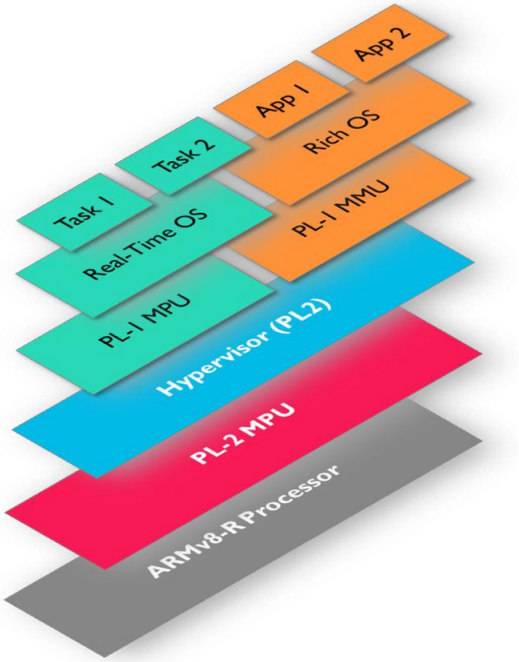

Figure 2: Second privilege layer in the ARMv8-R supports hardware assisted virtualization

The ARMv8-R architecture adds an additional level of privilege, PL2, for running a hypervisor that has exclusive control over a second-stage MPU which isolates regions of program, data and peripheral addresses between each virtualized OS guest and task in a 'sandbox'.

This supports the development of safety applications and the separation of operating systems and tasks. In addition it provides a separate mode for virtualization and the handling of critical safety and security events in the system. Real-time critical interrupts can be handled directly by the hypervisor, saving the overhead of switching to a guest OS.

The safety ecosystem

Designers of safety-related systems are faced with a number of challenges throughout the development process. In addition to the underlying hardware design, the designers need to consider the safety-related software that is being developed, and any supporting software tools that are required for the development of the system. Therefore the decision to use particular processor architecture for safety-related designs needs to be done with the overall safety requirements in mind.

ARM has been working with a number of partners to ensure that there is a robust ecosystem to support safety-related development activities on designs based on the ARM architecture. This includes work with semiconductor vendors, compiler and support tool vendors, and design consultancies.

A number of ARM partners have developed microcontroller products targeted for safety-related markets. Typical products are currently based on the ARMv7-R architecture, and include either the ARM Cortex-R4 or ARM Cortex-R5 real-time processor. These products are available from Texas Instruments with its Hercules range of multicore safety controllers, as well as Toshiba, Spansion, ScaleoChip, Renesas, STMicroelectronics, Infineon, Broadcom, LSI, Fujitsu and many others.

Software development tools are a vital element in the ecosystem as without the appropriate development tools developing applications for safety-related systems is challenging. When developing software for safety-related applications, the developers need to ensure that the compiler, for example, does not introduce errors into the generated object code which will be executed on the safety-related system.

Safety standards have different ways to describe requirements for software tool qualification. ISO 26262, for example, contains requirements for establishing a level of confidence for the use of software tools. The process of establishing sufficient confidence in the tools, such as compilers, typically requires additional software tool qualification activities to be conducted by the system developers. Compiler and analysis tool vendors supporting the ARM architecture have adopted a diverse range of approaches to facilitate such compiler qualification activities.

To address the requirements of tool qualification for various safety-related markets, ARM has developed the ARM Compiler Qualification Kit, which contains essential additional information about the ARM C/C++ compiler that can be used as part of tool qualification activities. The ARM Compiler Qualification Kit includes a Safety Manual for the compiler, describing the details of the compilation flow, together with information about possible failure modes of the compiler tool chain. Information about constraints of use is also included, with recommendations on applicable configuration options. Further, the ARM Compiler Qualification Kit includes detailed Defect Report, Test Report, and Development Process documents.

The ARM Compiler toolchain is further certified by TÜV SÜD, a recognized safety industry expert. The TÜV Certificate and the accompanying report confirm that the ARM Compiler version 5.04 fulfils the requirements for development tools classified T3 according to IEC 61508-3. This enables customers to use the ARM Compiler 5.04 for safety-related development up to SIL 3 (IEC 61508) or ASIL D (ISO 26262) without further qualification activities when following the recommendations and conditions documented in the Qualification Kit.

The ARM ecosystem for compilers is diverse, as in addition to C and C++ compilers from vendors such as Green Hills Software and IAR, AdaCore has an Ada compiler targeting the ARM architecture. Ada is a lesser known language that is well-suited for applications having stringent requirements for safety and security, such as avionics, industrial control, and railway control, as it has support for strong typing, concurrency, low-level programming, and mechanisms to support development of large-scale programs.

In addition to compilers, safety-related software development typically requires the use of debugging, testing and analysis tools. Examples include static analysis tools for demonstrating language subset conformance, such as MISRA C which is widely used within the automotive industry. Conformance to programming language subsets is a requirement in multiple functional safety standards, including IEC 61508 and ISO 26262.

Other program analysis tools that can be used to analyze worst case execution times (WCET) and maximum stack space usage of programs are also available for the ARM architecture.

ARM is also working closely with software vendors such as Green Hills Software, Mentor Graphics, eSOL, SYSGO and ETAS to develop various hypervisor implementations for the ARMv8-R architecture. For automotive safety system developers, these hardware and software solutions will permit consolidation of electronic control units via virtualization in order to save manufacturing cost and allow re-use of software between projects. Each software system can run in its own virtual machine with the hypervisor managing processing resources by time slicing, event triggering, or even scheduling between multi-processor cores.

Conclusion

The growth of the connected world is driving new demands on the development of safety-related systems. Combining both the safety requirements from different standards and demonstrating high levels of security is a challenge, and will remain a challenge in the future. However, advances in ARM architecture and ecosystem partner support for safety-related designs are helping developers overcome some of these key challenges. Processor features facilitating fault detection and control enable robust designs with reduced overhead at system level. The new ARMv8-R architecture allows easier consolidation of safety-related and other software, while maintaining strict separation. A growing number of development tools for the ARM architecture come with supporting safety information, reducing the effort required to qualify them for use in safety-related designs.

Safety engineering challenges cannot typically be solved in a bottom-up fashion, but these steps taken by ARM and the ecosystem partners are vital in ensuring that required product features and supporting information are in place to be used by the system designers and safety engineers. Ultimately this will result in reduced development effort, allowing the designers to focus on adding value to society through innovations in safety-related applications.

Join the discussion on safety, on the ARM Connected Community at http://community.arm.com/R5Safety

References

ISO 14971:2007. Medical devices. Application of risk management to medical devices.

ISO/IEC 15408 (series). Information technology – Security techniques – Evaluation criteria for IT security.

ISO 26262:2010. Road vehicles - Functional safety.

IEC 60601 (series). Medical electrical equipment.

IEC 61010 (series): Safety requirements for electrical equipment for measurement, control, and laboratory use.

IEC 61508:2010. Functional safety of electrical / electronic / programmable electronic safety-related systems.

IEC 61511:2003. Functional safety – Safety instrumented systems for the process industry sector.

IEC 62304:2006. Medical device software. Software lifecycle processes.