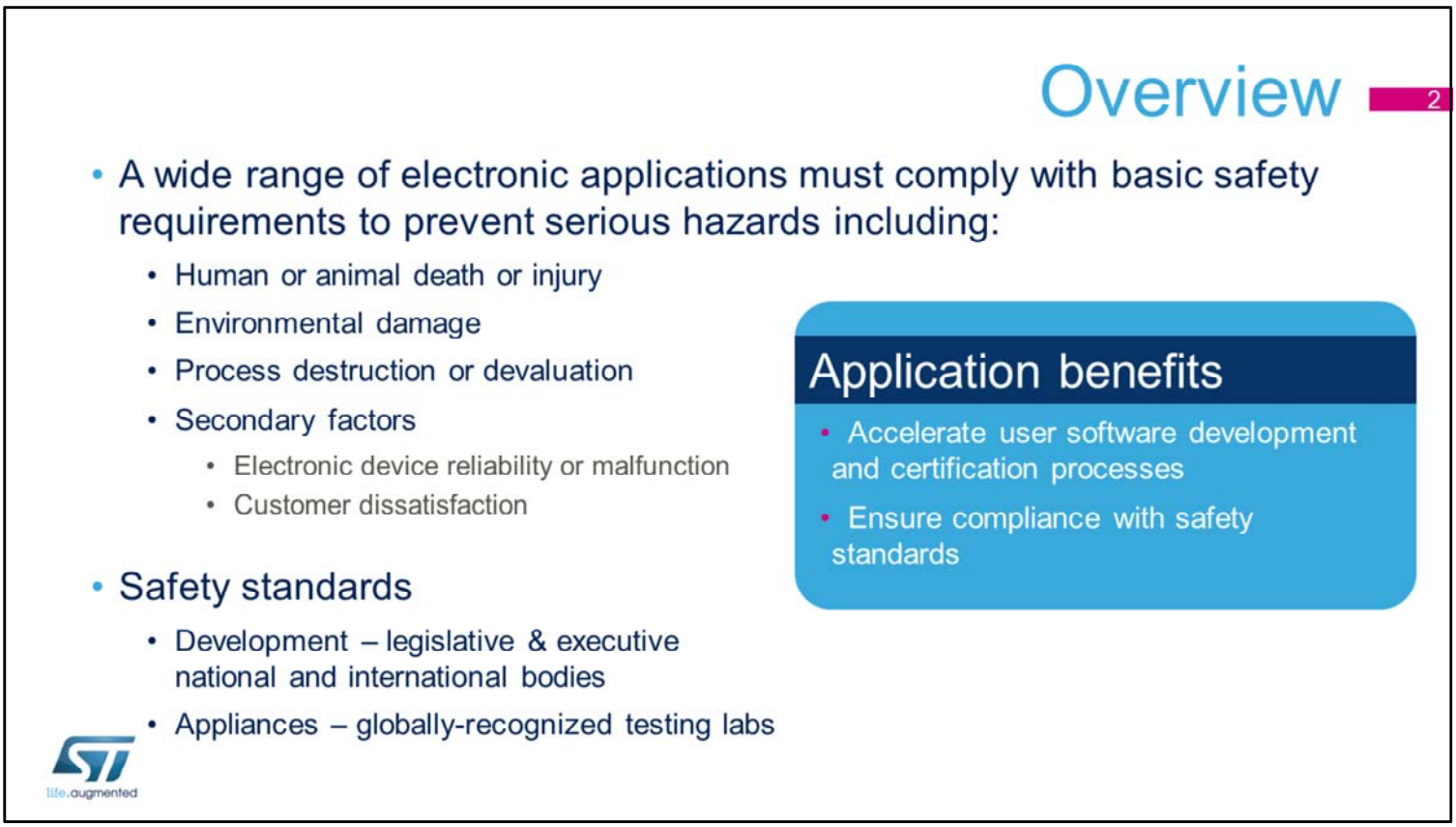
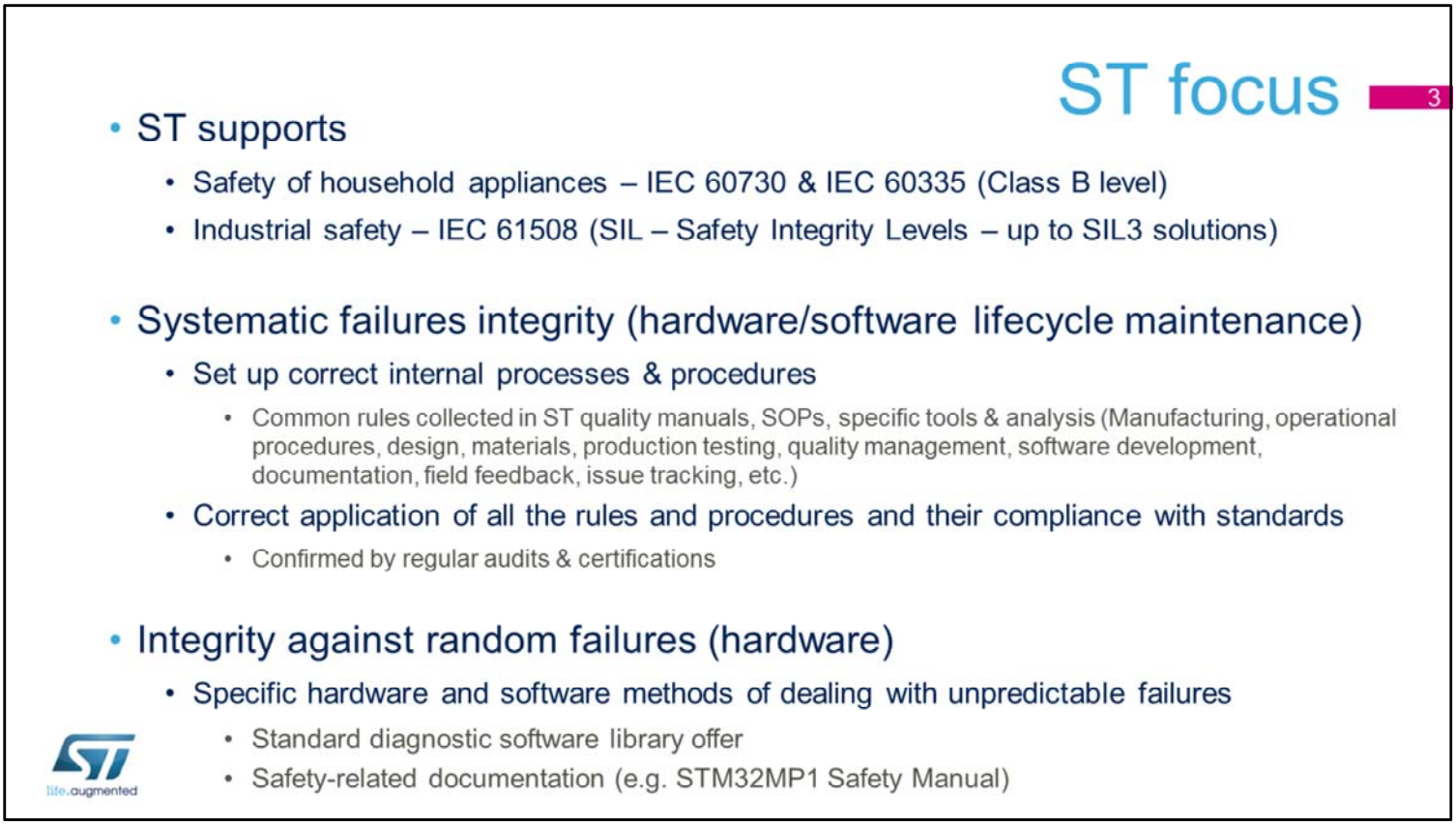
Hello, and welcome to this presentation of the STM32 safety support. It covers the requirements for compliance with safety standards and how STMicroelectronics helps customers targeting safety for their projects.



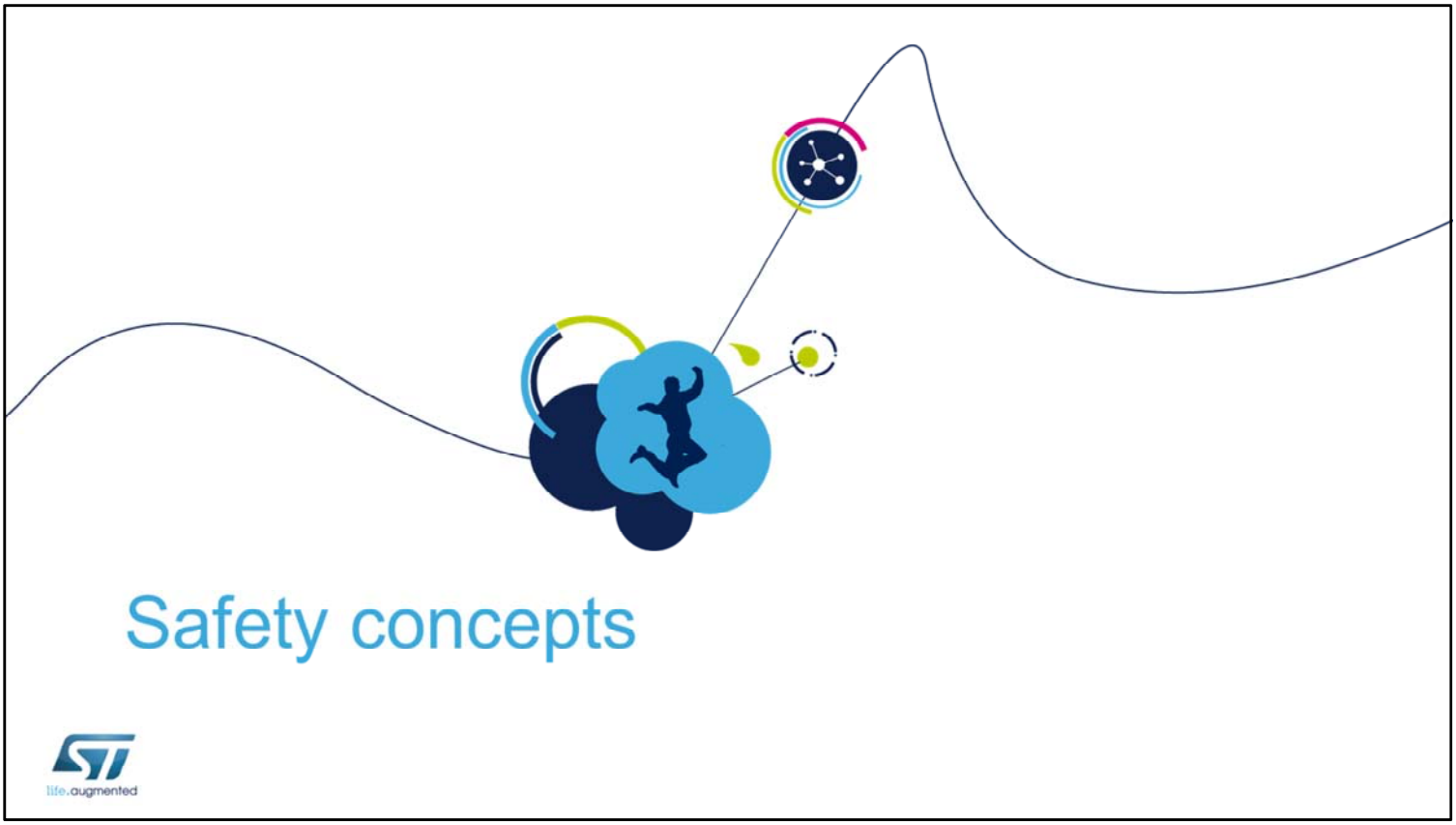
Safety requirements for electronic devices increase permanently as the use of electronic control systems expands into the huge range of human activities. The massive expansion of these devices requires their compliance with specific safety standards. The primary goal is to prevent human death or injury as well as environmental damage, but there are many other important factors at a lower level such as the devaluation of an industrial process including the loss of important data, connections, power or control and many others. The process for developing harmonized standards at both national and international levels is rather complex; sometimes involving completely opposite efforts (e.g. local market protection vs. its globalization). In any case, the main influencing factors come from field experience, market requirements, insurance issues, and the globalization of trade and business. The standards are produced by specific legislative and executive bodies while specific worldwide recognized testing houses

inspect and verify all the required appliances to ensure their compliance.
Applications targeting safety can benefit from the acceleration of software development. Efficient and early diagnostics using specific hardware features together with the application of proper hardware and software methods decrease the probability of hazardous events due to possible component malfunctions. Applying certain hardware design and manufacturing methods can even increase component reliability.



ST supports two basic general safety standards – a specific one targeting household appliances known as a “Class B” or “Class C” standard and a more common industrial standard targeting safety integrity levels called ”SIL”. The latter is a generic standard which produces a large number of derivative standards dedicated to different fields of application.
ST, in compliance with these standards, cares about both systematic and random failures. Systematic failures are predictable and their avoidance and monitoring are based on practical experience gained in the industry. Systematic failures can be avoided mainly by applying correct internal processes throughout a product’s lifecycle. These requirements are defined in specific internal quality documentation. Regular inspections and audits ensure that these internal rules are applied and comply with the recognized standards.
To ensure integrity against random failures, specific software

methods and hardware design techniques must be applied as described in the following slides.

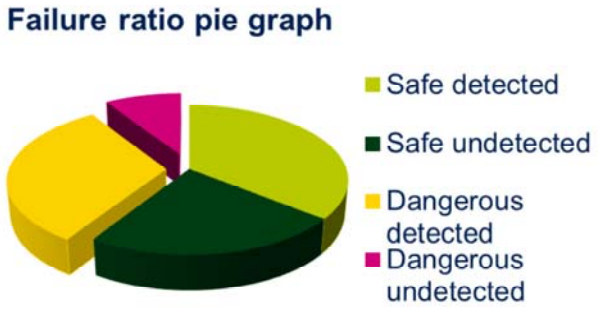


The next slides will give you an overview of the main safety concepts to be taken into account when working with microcontrollers.

Random failures methodology (1)

5

- Identification of random failures
  - Safe & Dangerous
  - Detected & Undetected
- Types of random failures
  - Permanent- component is permanently damaged
  - Transient - recovery can be possible
    - Soft-errors – identifiable by SW or HW tests or diagnostics
    - Transient - identifiable by fast HW tests or diagnostics exclusively
  - Cross-product failure criteria
    - Single-point failures (SPF) – immediate effect
    - Latent failures (LF) – dormant, can aggregate with another fault
    - Common causes of failure (CCF) – immediate effect, several components affected; possible destruction of complex safety structures (power, clock, temperature, timing)



life.augmented

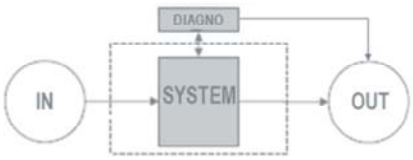
Not all random failures result in a hazardous event, and they may even be considered as safe from a safety point of view. Basically, safety standards require monitoring to detect dangerous failures that may be directly or indirectly related to safety and have the potential to cause a dangerous situation. Both safe and dangerous errors can either be detected or stay hidden and undetected by the system. The more often dangerous errors are discovered and prevented in time, the more the probability of a failure propagating into a hazardous event decreases. The time needed to detect dangerous errors and prevent hazardous events must fit into the overall Process Safety Time (PST) available which includes all the possible delays and reaction times for the system (e.g. on sensors or actuators). For quantification purposes, safety standards recognize a Safe Failure Fraction and Diagnostic Coverage. The Safe Failure Fraction, or SFF, is the ratio of the rate of safe failures, including the rate of detected dangerous failures, to the total

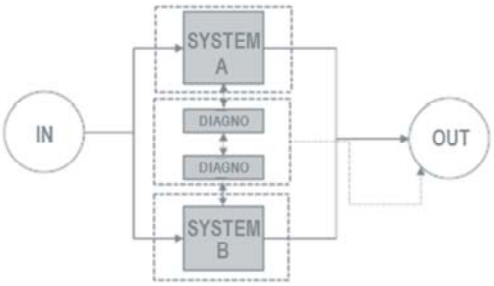
failure rate (safe failures as well as detected and undetected dangerous failures). The Diagnostic Coverage, or DC, is the ratio of the probability of detected dangerous failures to the probability all the dangerous failures. Random failures can cause permanent or recoverable errors. Hard failures cause permanent physical damage to the component and the system is no longer able to operate normally. If no compensation is possible, the system has to be put into a safe state (e.g. cutting power to actuators) until it is repaired. Random transient or soft errors can be correctable and some kind of recovery process can be applicable. In addition to being detected, these failures can also be compensated in certain cases. Soft-error failures can be managed by both hardware and software while transient failures need fast hardware methods exclusively. Software tests can never compensate for these temporary and short-lived errors efficiently as they are considerably slower and limited by their execution time.

From a cross-product point of view, using ISO 26262 terminology, we can recognize single-point, latent or common types of failure causes. Common causes of failure require a special focus especially as they can potentially destroy even quite complex safety structures.

Random failures methodology (2)

- Random failure control techniques
  - Detection
    - Diagnostics recognize an error
    - System is no longer able to continue normal operations
    - It has to fall into fail safe state or be recovered
  - Compensation (Hard Fault Tolerance (HFT) > 0)
    - Diagnostics is able to detect and identify the bad part
    - Next correct one is still available
    - System can still continue at normal operation
- Essential principle – REDUNDANCY
  - Diagnostics, comparison, identification, and voting





life.augmented

When random failures are detected and cannot be compensated for, especially after a dangerous error is detected, the system has to be stopped and placed into a safe state or go through a recovery process like reset, roll back or a specific check function.
Compensation methods usually allow the system to continue operating normally while using error-correction, passivation or masking functions. Generally, a sure voting process is used to identify the damaged part or incorrect data which is then replaced by the correct one. Standards recognize Hard Fault Tolerance (HFT), or the maximum number of errors which a system can absorb while it still can continue at normal operation.
In addition to specific functional testing, redundancy is the essential diagnostic principle here. Both detection and compensation techniques always require a sure level of redundancy to be efficient. Compensation is considerably more demanding than detection, as not only discrepancies

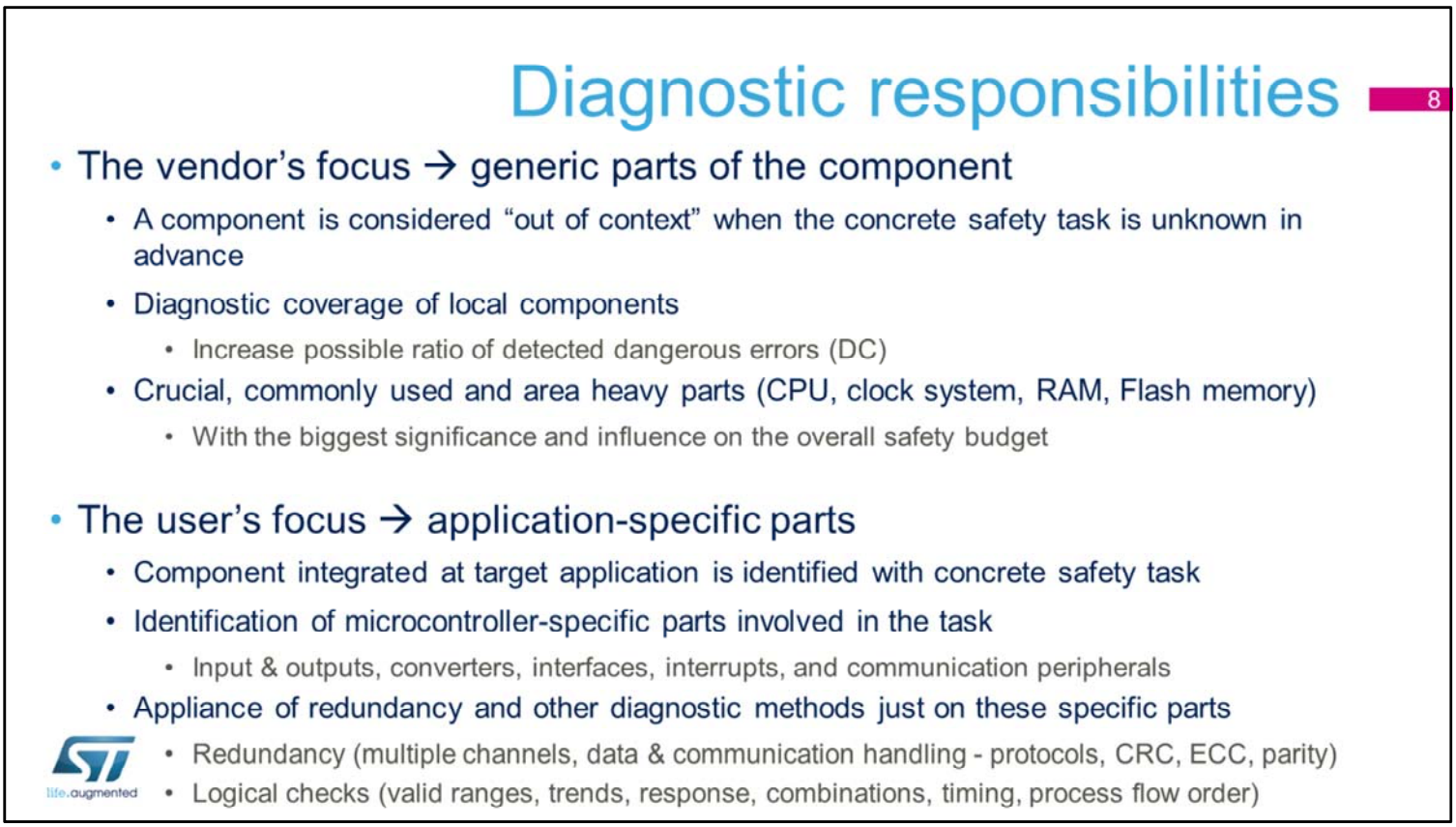
but the correct state has to be identified as well. To do so, specific comparison and voting mechanisms have to be additionally applied.

Random failures methodology (3)

- Redundancy techniques
  - Structural
    - Parallel identical structures like dual registers, memories, CPU, or MCUs with hardware comparators and voters.
  - Functional
    - Parallel asymmetrical hardware structures or different software methods are applied for a single task and their outputs are compared.
  - Temporal
    - The same method is implemented several times using the same hardware or software at different time slots and results are compared.
  - Informational
    - Added information is implemented at data level and evaluated for compliancy by hardware or software (parity, ECC, CRC, data protocols, or copies).

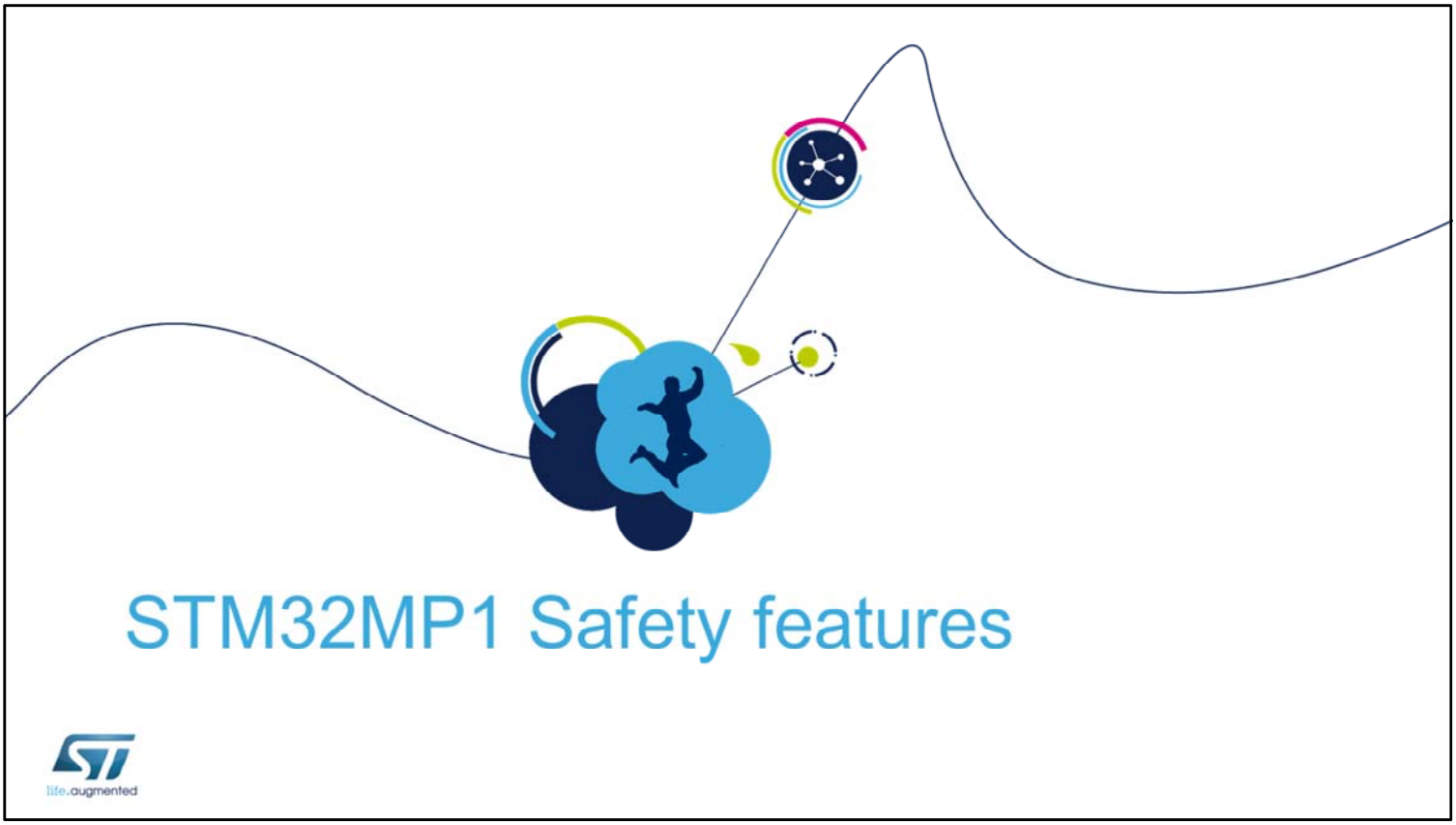
life.augmented

The required level of redundancy can be achieved using a wide range of different software or hardware methods and techniques. Some of them are listed here and others will be highlighted later in this presentation. The techniques can be usually achieved either by hardware or software, or a combination of both.

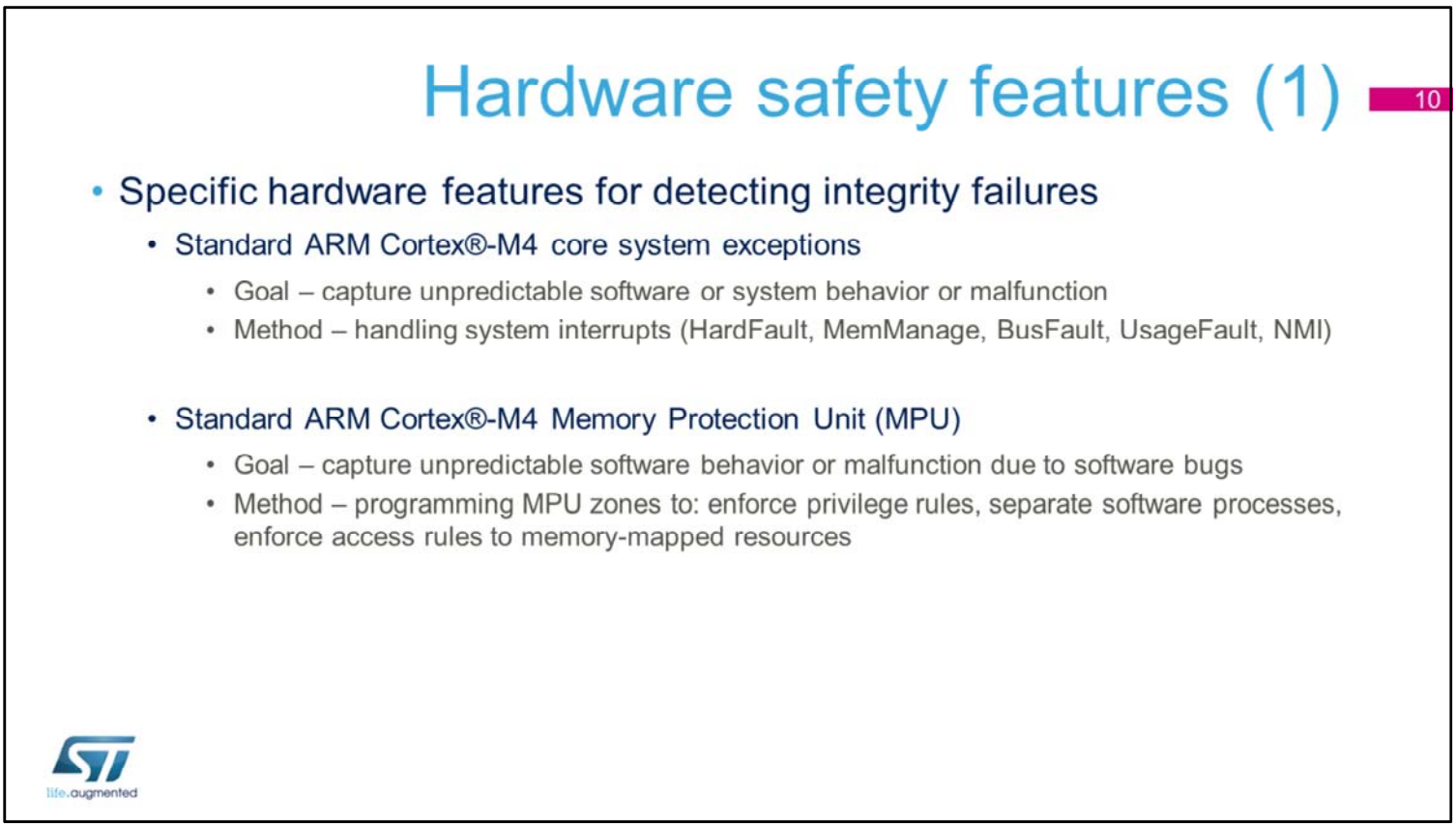


From a safety point of view, a microcontroller is a relatively complex programmable electronic component which has to comply with specific requirements determined by the applicable standards. In regards to support safety for a microcontroller, a vendor considers the product as a component "out of context" as its final application purpose and safety tasks are not known in advance. This is why we can speak about component "ready" or "suitable" for a determined common level of safety tasks. The effort is always to cover the component's overall reliability and fulfill the overall budget of diagnostic coverage defined by the standard for the given safety integrity level required by the final application. A complex component like a microcontroller can be considered as a set of partial components involved in various safety tasks, each with a different diagnostic coverage and weight in the overall component safety budget. An effective way to ensure the required overall safety budget has to be focused on crucial and generic parts of the

microcontroller especially those commonly used by most applications. Any small improvement in the safety of these fundamental and significant parts of the design always brings the biggest gain in the overall safety budget of the component, which is beneficial for each application.
Once a microcontroller is included in an application design and the safety task is specified, then the safety support can be deployed much more efficiently and cover just the very specific parts of the microcontroller involved in the required safety case. Many efficient methods can then be applied based on detailed knowledge of the application requirements, its design, the process and the equipment under control. Redundancy and knowledge of the system behavior are crucial principles applied either separately or together. Inputs and outputs can be multiplied or checked by feedback, tested for logical state, value or expected response in trends or time intervals. The processes can be monitored for correct timing and flow order. Correct decisions can be made based on the comparison of results coming from redundant and independent flows, analysis, calculations or data.



The following slides are devoted to features dedicated for safety support.



STM32MP1 microprocessors feature hardware solutions for diagnostic testing and to quickly react to failures with the potential to cover lower level safety applications. The hardware tests are autonomous with minimal or no software control. This is especially helpful in detecting transient errors and consumes the least amount of time of the overall process safety time.
Please note that overall contribution to MCU mitigation by above reported diagnostics is marginal, being STM32MP1 not explicitly designed for specific use in safety applications.

Hardware safety features (2)

- Specific hardware features for detecting integrity failures
  - ARM Cortex-M4 dedicated Window watchdog (WWDG1)
    - Goal – monitoring correct software timings and flows
    - Method – apply correct techniques for handling watchdog timeouts - see our specific application notes
  - ARM Cortex-A7 dedicated Independent watchdogs (IWDG1 and IWDG2)
    - Goal – monitoring correct software timings and flows
    - Method – apply correct techniques for handling watchdogs timeouts - see our specific application notes

life.augmented

STM32MP1 microcontrollers feature hardware solutions for diagnostic testing and to quickly react to failures with the potential to cover lower level safety applications. The hardware tests are autonomous with minimal or no software control. This is especially helpful in detecting transient errors and consumes the least amount of time of the overall process safety time.
Please note that the overall contribution to MCU mitigation using the above reported diagnostics is marginal, as the STM32MP1 is not explicitly designed for specific use in safety applications.

Hardware safety features (3)

- Specific hardware features for detecting integrity failures
  - Standard ARM® Cortex®-M7 Memory Management Unit (MMU)
    - Goal – detect unpredictable software behavior or malfunction due to software bugs
    - Method – programming MMU zones to: enforce privilege rules, separate software processes, and enforce access rules to memory-mapped resources
  - Enhanced Trust Protection Controller (eTPZC)
    - Goal – configuring Trust Zone access rights to peripherals
    - Method – programming eTPZC zones to implement an isolation scheme between ARM® A7 dedicated peripherals and ARM® M4 dedicated ones. This isolation scheme allows the coexistence of non-safety related application software execution on A7 CPU and safety-related application software execution on M4 CPU

life.augmented

The above reported hardware diagnostic can contribute in relevant way to improve the systematic safety integrity of the software solutions executed on STM32MP1.

Hardware safety features (4)

- HW CRC computation module
  - Goal – fast calculation of CRC checksum on given set of data (support of software methods)
  - Method - built up additional redundancy above a set of data (communication, memories)
- Clock security system for external clocks
  - Goal – detect malfunction of external clock
  - Method – automatic switch to internal clock, raise NMI interrupt
  - Separated CSS blocks are available for HSE
- Clock cross-reference measurement (monitoring of differences between two frequencies)
  - Goal – detect malfunction of clock system (support of software methods)
  - Method – reference frequency input is captured by another one at dedicated timer

life.augmented

This slide lists additional safety features dedicated to Checksum Redundancy Code (CRC) computation and clock control.

Hardware safety features (5)

- Power supply supervisor (Power-On Reset, Power-Down Reset and Programmable voltage detector)
  - Goal – safe thresholds to ensure correct function of all parts of the system
  - Method – interrupt to call emergency shutdown task or keeping the device under reset
- Handling protocols at communication peripherals
  - Goal – fast hardware calculation and verification of CRC checksum on given set of data
  - Method - built up additional redundancy above communicated data
- Break input for timers collecting selected system errors
  - Goal – fast control of timer outputs generating timing signals
  - Method – put all timer outputs into predefined state

life.augmented

Please note that additional software tests must be added when compensation or an additional check is required, for example to achieve a higher level of SIL. In this case, the user must ensure that the software testing period takes into account the process safety time.

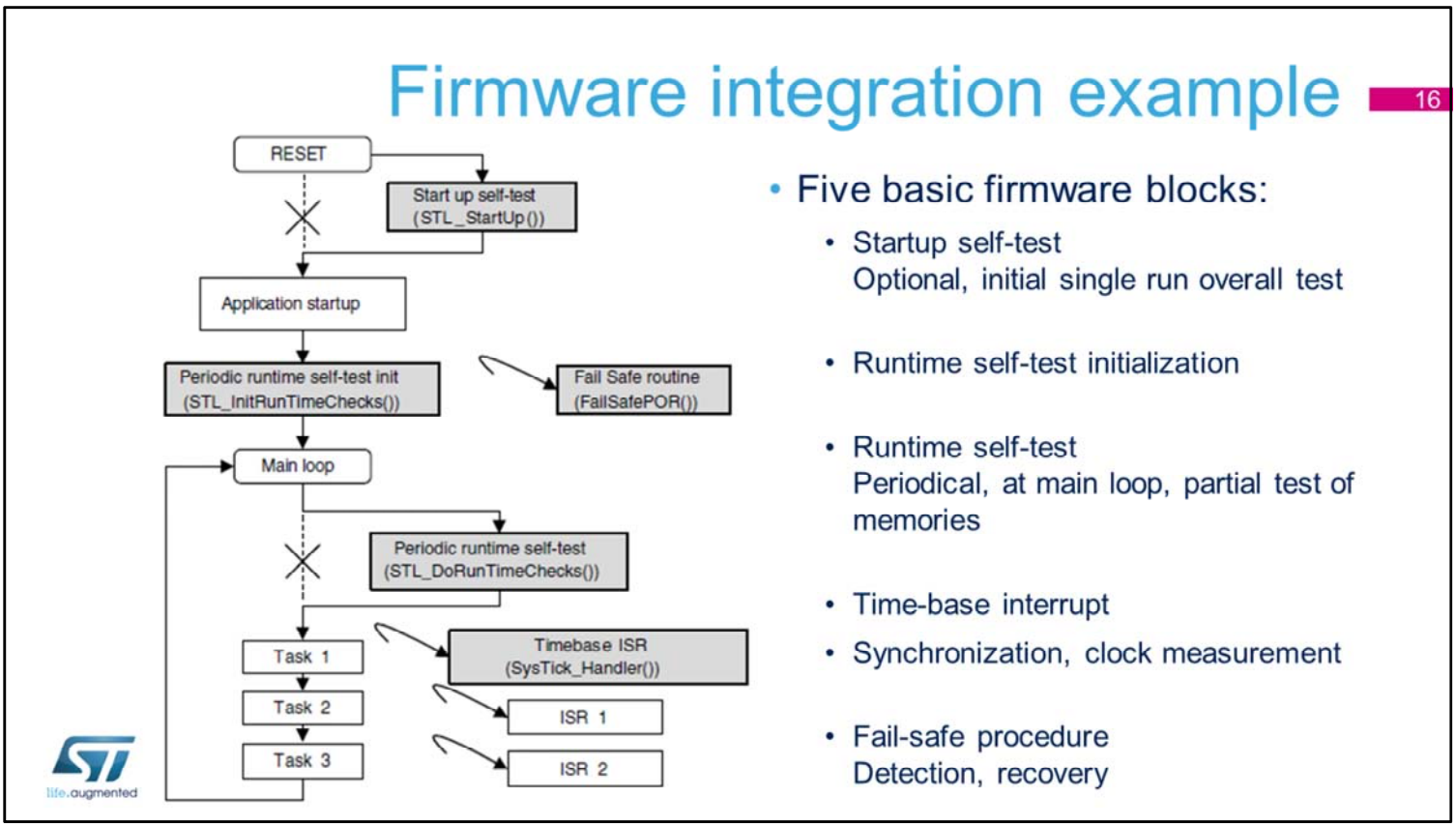
Firmware safety accessory checks

- Software checks improving the capability to detect random failures
  - Multiple software solutions are available on st.com website to address safety related projects by leveraging on standard, ST verified software checks:
    - X-CUBE-STL : software solution to achieve IEC61508 compatibility up to SIL2
    - X-CUBE-CLASSB/ STM32-CLASSB-SPL : software solutions to achieve IEC61730/IEC60335-1 ClassB certification

End user must check the availability of the software solutions for the specific STM32 series/part number by visiting st.com website or by contacting local ST representative

life.augmented

This slide lists the software checks included in the ST self-test firmware solution with a brief summary of why they are applicable. Generally, the firmware focuses on generic parts of the microcontroller based on in-depth knowledge of the design while packages dedicated to achieve SIL standards use more extensive testing methods proved by specific methodology for their efficiency. The packages are not available for free download. Users should ask their local ST representative for the firmware.



In principle, self-test procedures are included as an additional task when initializing the application main loop during system startup. This runtime self-test task provides periodic testing of the CPU, clock system, stack boundary, program flow and both volatile and non-volatile memories. The watchdog timeout is refreshed upon completion if everything goes correctly. The memory areas are tested step-by-step per parts within the task. The test is synchronized by time-base ticks derived from timer interrupts. The interval required to complete the test depends mainly on size of the memory areas under test, frequency of the task calls, and sizes of the blocks tested in a single step. Optionally, a one-time initial startup overall self-test can be additionally implemented at power-on or after application reset. Whenever a malfunction or discrepancy is found during these tests, the fail-safe routine is called. It should put the application into safe state and determine the next recovery possibilities.

Related peripherals

- Refer to these trainings linked to Safety topics
  - Reset and clock control (RCC)
  - ARM Cortex-M4 (Core)
  - ARM Cortex-A7 (Core)
  - Power control (PWR)
  - Flash memory (FLASH)
  - Cyclic Redundancy Check (CRC)
  - Independent Watchdog (IWDG)
  - System Window Watchdog (WWDG)

life.augmented

Safety is spread over the full STM32MP1 product range. You will find a detailed description of the aforementioned features in the different peripherals chapters.

References

- For more details, please refer to following sources and other presentations related to the peripherals focusing on safety:
  - Application note AN3307: Guidelines for obtaining IEC 60335 Class B certification in any STM32 applications
  - Application note AN4435: Guidelines for obtaining UL/CSA/IEC 60335 Class B certification in any STM32 application *

(*) Associated firmware and documentation could still be under certification process

life.augmented

For more details, please refer to dedicated documentation and contact your local ST representatives for the availability, status and possible delivery of the firmware and associated documentation.