



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.

- A wide range of electronic applications must comply with basic safety requirements to prevent serious hazards including:
  - Human or animal death or injury
  - Environmental damage
  - Process destruction or devaluation
  - Secondary factors
    - Electronic device reliability or malfunction
    - Customer dissatisfaction
- Safety standards
  - Development – legislative & executive national and international bodies
  - Appliances – globally-recognized testing labs

### Application benefits

- Accelerate user software development and certification processes
- Ensure compliance with safety standards



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 (like 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.

- STM32 MCUs and MPUs support:
  - Safety of household appliances – IEC 60730 & IEC 60335 (Class B level)
  - Industrial safety – IEC 61508 (SIL – up to SIL3 solutions)
- Systematic failures integrity (hardware/software lifecycle maintenance)
  - Set up correct internal processes & procedures
    - Common rules collected in ST quality manuals, Standard Operating Procedures (SOPs), specific tools & analysis (Manufacturing, operational procedures, design, materials, production testing, quality management, software development, documentation, field feedback, issue tracking, etc.)
  - Correct application of all the rules and procedures and their compliance with standards
    - Confirmed by regular audits & certifications
- Integrity against random failures (hardware)
  - Specific hardware and software methods of dealing with unpredictable failures
    - Standard diagnostic software library offer
    - Safety-related documentation (e.g. STM32U0 Safety Manual)



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.

## Safety concepts

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)

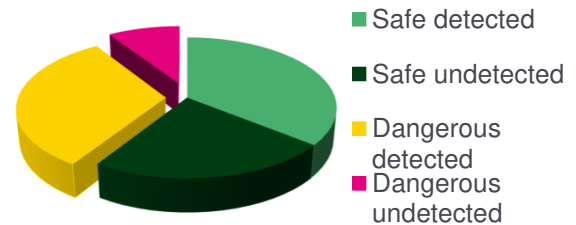
- Identification of random failures

- 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)

Failure ratio pie graph



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 of 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.

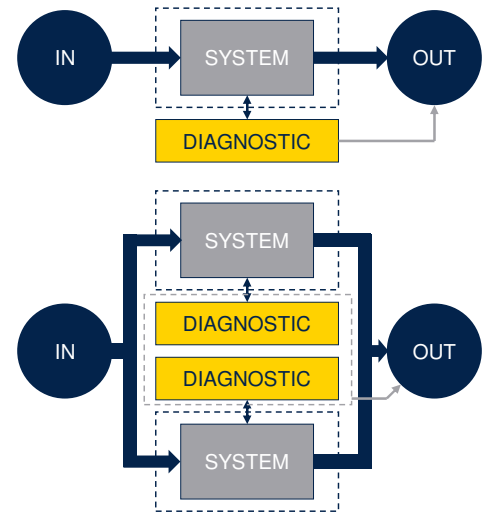
From a cross-product point of view, using ISO 26262 terminology, we can recognize single-point, latent or common types of failure causes.

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



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)



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.

## Diagnostic responsibilities

- The vendor's focus: generic parts of the component
  - A component is considered "out of context" when the concrete safety task is unknown in advance
  - Diagnostic coverage of local components
    - Increase possible ratio of detected dangerous errors (DC)
  - Crucial, commonly used and area heavy parts (CPU, clock system, RAM, Flash memory)
    - With the biggest significance and influence on the overall safety budget
- The user's focus: application-specific parts
  - Component integrated at target application is identified with concrete safety task
  - Identification of microcontroller-specific parts involved in the task
    - Input & outputs, converters, interfaces, interrupts, and communication peripherals
  - Appliance of redundancy and other diagnostic methods just on these specific parts
    - Redundancy (multiple channels, data & communication handling - protocols, CRC, ECC, parity)
    - Logical checks (valid ranges, trends, response, combinations, timing, process flow order)



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.

## STM32U0 Safety features

The following slides are devoted to features present in STM32U0 dedicated for safety support.

## Hardware safety features (1)

- Specific hardware features for detecting random failures

Standard ARM Cortex®-M0+ core system exceptions	
Goal	Capture unpredictable software or system behavior or malfunction
Method	Handling system interrupts (hardfault, memmanage, busfault, usagefault, NMI)
Standard ARM Cortex®-M0+ Memory Protection Unit (MPU)	
Goal	Capture unpredictable software behavior or malfunction due to software bugs
Method	Programming MPU zones to: <ul style="list-style-type: none"><li>➢ Enforce privilege rules</li><li>➢ Separate software processes</li><li>➢ Enforce access rules to memory-mapped resources</li></ul>
Independent and Window watchdogs	
Goal	Monitoring correct software timings and flows
Method	Apply correct techniques for handling watchdogs timeouts <ul style="list-style-type: none"><li>➢ See our specific application notes</li></ul>



STM32U0 microcontrollers feature specific hardware for efficient diagnostic testing and to quickly react to failures with the potential to cover a wide range of 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 by the diagnostics reported above is marginal, since the STM32U0 is not explicitly designed for specific use in safety applications.

## Hardware safety features (2)

- Specific hardware features for detecting integrity failures

ECC	
Goal	Correct single errors and detect double errors by 8 bits added for each 64-bit word in flash
Method	Implementation of SECDED schemes and interrupt generation on errors detection
Parity	
Goal	Detect single errors on SRAM
Method	Implementation of parity schemes on SRAM (1 bit each 8), interrupt generation on errors detection

Memory integrity failure could lead to unpredictable results. The Error Correcting Code (ECC) is the most common technique used to detect data corruption on memory devices. The ECC calculation is performed by hardware. It applies 8 additional redundant bits to each 64-bit word making so called Hamming distance between valid values of the stored word data. This basic commonly used distance, known as SECDED, enables Single Error Correction or Double Error Detection upon each data value. In this case, the corrections are done automatically, and interrupt is generated on errors detection.

Parity only enables Single Error Detection (SED).

Both ECC and parity aim to detect bit flips that may occur while data is stored in memory.



## Hardware safety features (3)

Hardware 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 (CSS) 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 and LSE
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



12

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

Note that the STM32U0 features two Clock Security System units:

- One, called CSS, monitors the High-Speed External (HSE) oscillator
- The other, called LSECSS, monitors the Low-Speed External (LSE) oscillator.

In case of failure, an automatic switching to the clock provided by an internal oscillator, respectively HSI and LSI, is achieved.

## Hardware safety features (4)

<b>Power supply supervisor (Power-On Reset, Power-Down Reset and Programmable Voltage Detector)</b>	
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
<b>Locking mechanism of configuration registers</b>	
Goal	Preventing any accidental change of critical configurations (peripherals, system)
Method	Control locking registers and bits, configuration under specific conditions only
<b>Handling protocols at communication peripherals</b>	
Goal	Fast hardware calculation and verification of CRC checksum on given set of data
Method	Built up additional redundancy above communicated data
<b>Break input for timers collecting selected system errors</b>	
Goal	Fast control of timer outputs generating timing signals
Method	Put all timer outputs into predefined state



Nevertheless, all the tests dedicated to the detection of failures listed here are not sufficient to achieve higher safety levels.

This is why additional functional software self tests, additional checks and techniques must be added to comply the safety standards requirements. User must ensure that each software testing loop must be fully completed within 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 SIL3
    - X-CUBE-CLASSB : software solutions to achieve IEC60730-1/IEC60335-1 Class B 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*

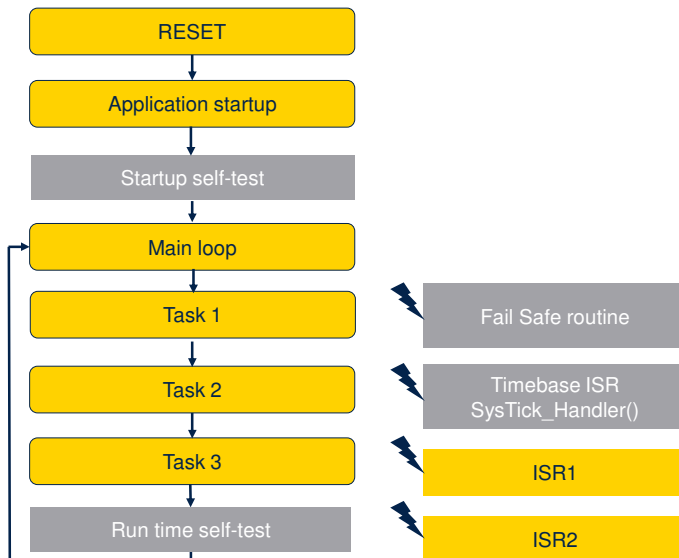


This slide lists the software checks included in the ST self-test firmware solution with a brief summary of why they are applicable.

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 could be not available for free download. Users should ask their local ST representative for the firmware.

## Firmware integration example



- Four basic firmware blocks:
  - Startup self-test  
Optional, initial single run overall test
  - Run time self-test repeated periodically
  - Time-base interrupts to synchronization and clock measurement
  - Fail-safe procedure called at fault detection

In principle, startup self-test procedures are performed once the system is initialized prior the application entry to main loop.

Runtime self-test is then scheduled periodically within main task loop together with the other application tasks.

The watchdog timeout is usually refreshed upon each completion of the run time test if everything goes correctly. ST firmware provides certified testing modules for testing the CPU, memories and the clock system. The other testing modules must be implemented by end user in accordance with the application design and its safety task.

The startup test should check the system overall including all the memory areas within a single run while run time test can perform the tests per part at partial steps. Both tests are synchronized by time-base ticks derived from timer interrupts.

The interval required to complete the test depends mainly on

configurable size of the memory areas under test. At run time, it depends on frequency of the task calls, and sizes of blocks tested in a single step.

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®-M0+ (Core)
  - Power control (PWR)
  - Flash memory (FLASH)
  - Cyclic Redundancy Check (CRC)
  - Independent Watchdog (IWDG)
  - System Window Watchdog (WWDG)



Safety is spread over the full STM32U0 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:
  - UM3261 Safety Manual for STM32U0 microcontrollers series, including instructions about STM32U0 use in the framework of IEC 61508 and other safety standards
  - Class B FW is described at specific STM32U0 associated user guide

*(\*) Associated firmware and documentation is under certification process*



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

# Thank you

© STMicroelectronics - All rights reserved.

ST logo is a trademark or a registered trademark of STMicroelectronics International NV or its affiliates in the EU and/or other countries.

For additional information about ST trademarks, please refer to [www.st.com/trademarks](http://www.st.com/trademarks).

All other product or service names are the property of their respective owners.



Thanks for attending this presentation.