

S32K14X SAFETY SOFTWARE INTRODUCTION

AMP GPIS AE
DEC 2019



SECURE CONNECTIONS
FOR A SMARTER WORLD

CONFIDENTIAL AND PROPRIETARY

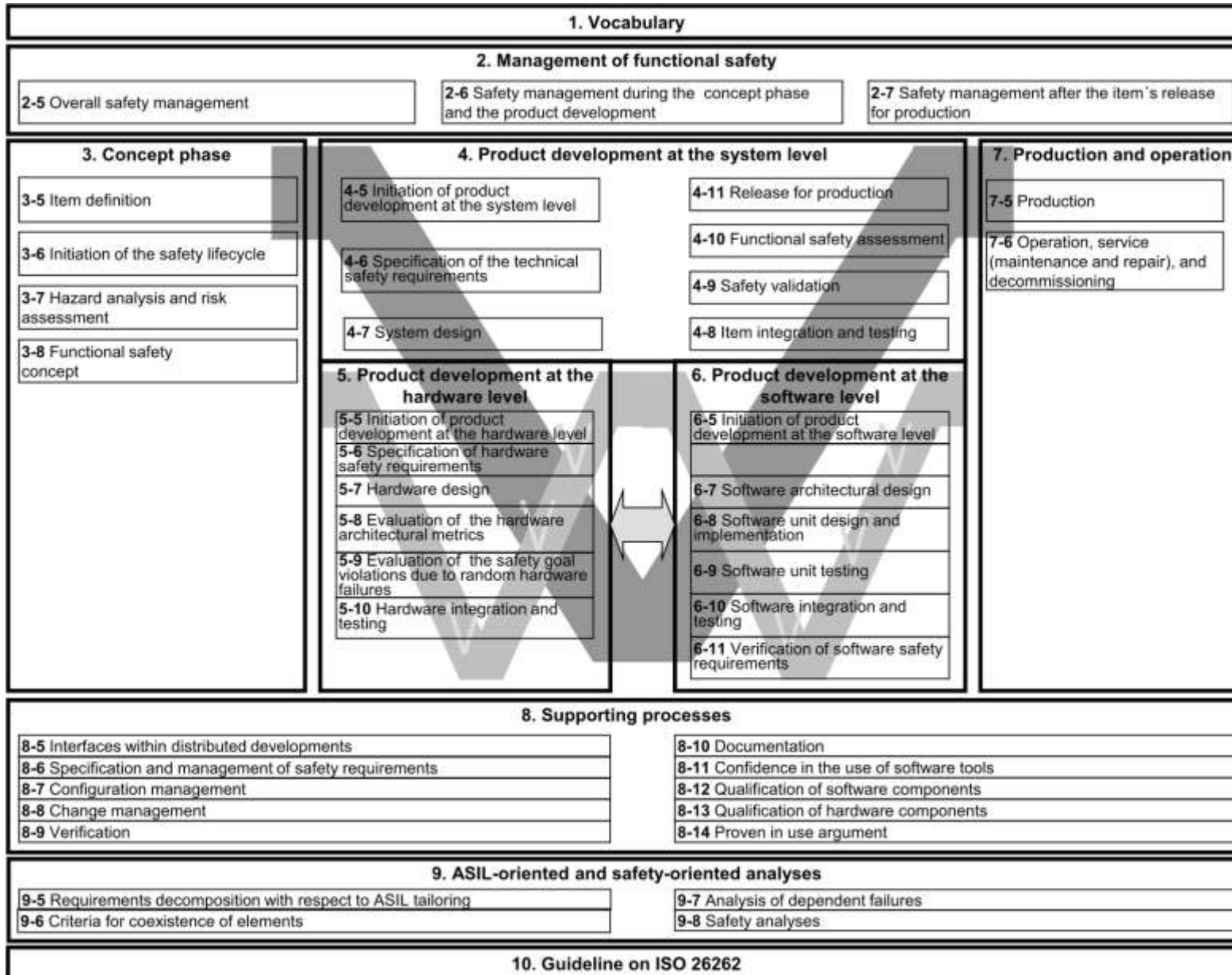


Agenda

- Functional Safety Background
- S32K1xx Safety features overview
- S32K1xx Safety documents request
- S32K14x Safety software overview

FUNCTIONAL SAFETY BACKGROUND

ISO26262 Overview



What is Functional Safety (功能安全)?

ISO 26262-1 (Vocabulary)

- Absence of **unreasonable risk** due to **hazards** caused by **malfunctioning behavior** of **E/E systems**.

Hazard Analysis and Risk Assessment (HARA)

- Identify and categorize the hazards that can be triggered by malfunctions in the system
- The Risk Assessment is carried out using three criteria
 - Severity – how much harm is done?

Class	S0	S1	S2	S3
Description	No injuries	Light and moderate injuries	Severe and life-threatening injuries (survival probable)	Life-threatening injuries (survival uncertain), fatal injuries

- Exposure – how often is it likely to happen?

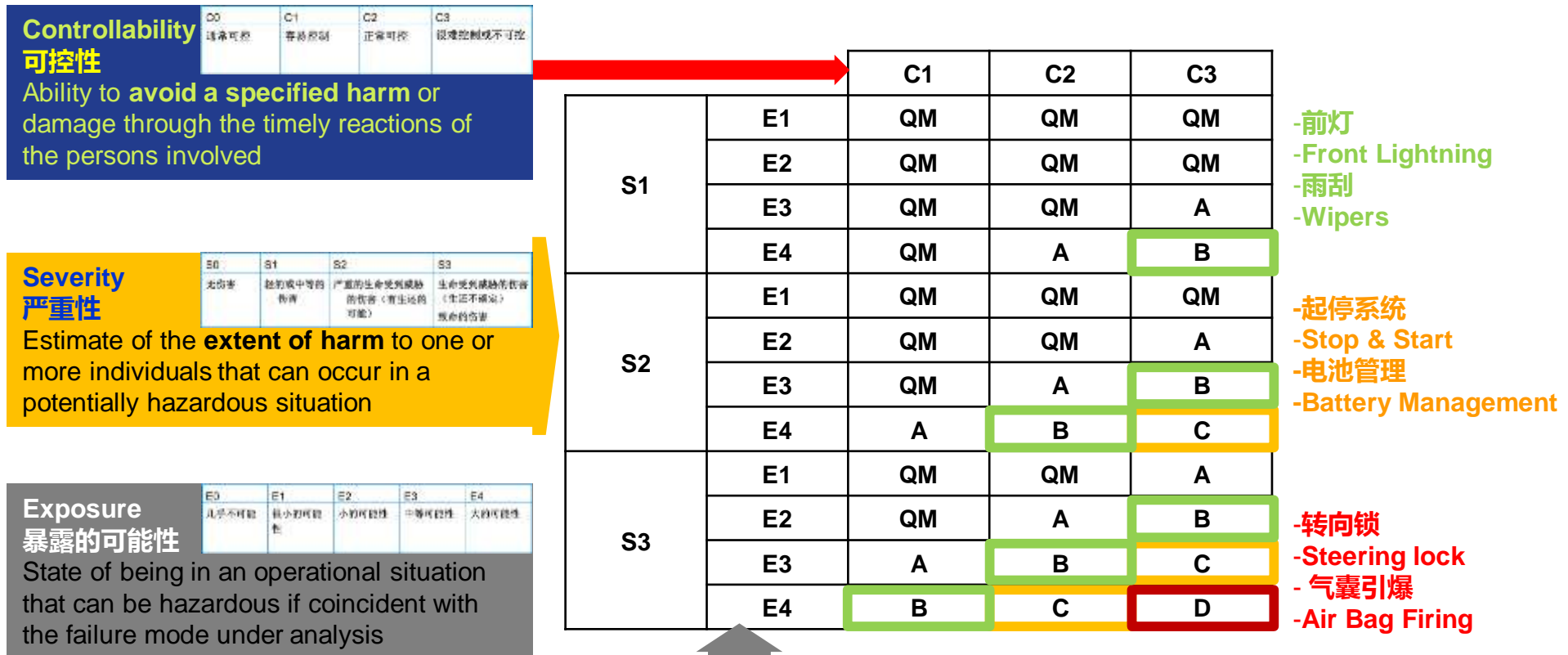
Class	E0	E1	E2	E3	E4
Description	Incredible	Very low probability	Low probability	Medium probability	High probability

- Controllability – can the hazard be controlled?

Class	C0	C1	C2	C3
Description	Controllable in general	Simply controllable	Normally controllable	Difficult to control or uncontrollable

Automotive Safety Integrity Level (汽车安全性等级)

- One of four levels to specify the item's or element's necessary requirements of ISO 26262 and safety measures to apply for avoiding an unreasonable residual risk, with D representing the most stringent and A the least stringent level.



Target Metrics for ASIL

- Associate the following target metrics to each **safety goal**
 - Single-point fault metric (SPFM)

Table 4 — Possible source for the derivation of the target “single-point fault metric” value

	ASIL B	ASIL C	ASIL D
Single-point fault metric	≥90 %	≥97 %	≥99 %

- Latent-fault metric (LFM)

Table 5 — Possible source for the derivation of the target “latent-fault metric” value

	ASIL B	ASIL C	ASIL D
Latent-fault metric	≥60 %	≥80 %	≥90 %

- Probabilistic Metric for random Hardware Failures (PMHF)

Table 6 — Possible source for the derivation of the random hardware failure target values

ASIL	Random hardware failure target values
D	$<10^{-8} \text{ h}^{-1}$
C	$<10^{-7} \text{ h}^{-1}$
B	$<10^{-7} \text{ h}^{-1}$

Reference ISO 26262-5:2011

Example – FMEDA for S32K144 SRAM

- FMEDA lists the failure rates and the safety measures required to achieve ASIL-B on S32K144
- Safety Manual includes the detailed description of SM_111:
 - The ECC SRAM reporting has to be enabled by the software application (in LMEM module), before the safety application starts.
 - The Error Injection Module (EIM) allows you to induce single-bit and multi-bit inversions on read data when accessing the System RAM. Due to ECC correction mechanism, an error in ECC could directly violate the safety goal. ECC shall be checked once within the FTTI.

6	Targeted Safety Integrity Level	SW EIM check executed within FTTI	EDC_ECC Code	How much of Memory is used	SCST(Sw)	Assuming 50% safe and 50% dangerous failures
7				Array		
8	ASIL B & SIL2 (90%)	TRUE	TRUE	100%	TRUE	TRUE
9		Diagnostic Coverage of EIM (SW check)	Diagnostic Coverage of ECC for d.c.model		RUN SCST within FTTI	Assuming a dangerous failure fraction (nSAFE) of
10		DC = 90.0%	DC = 99.0%		TRUE	50%
11		[SM_111]	Diagnostic Coverage of ECC for transient faults		Diagnostic Coverage of SCST for PRAM Controller d.c. model	
12			DC = 99.0%		DC = 0.0%	

Volatile Memory - System RAM (ISO 26262-5 D.1: Volatile Memory) Product: S32K144

Single-Point Fault Metric	≥ 96.90%	ASIL B & SIL 2 requires a single-point fault metric ≥ 90%
Latent Fault Metric	≥ 79.90%	ASIL B & SIL 2 requires a latent fault metric ≥ 60%
Safe Failure Fraction	≥ 96.90%	ASIL B & SIL 2 requires a single-point fault metric ≥ 90%
$\lambda_{SPF} + \lambda_{RF}$ (ISO26262), λ_{DU} (IEC61508)	6.12678E-09	ASIL B & SIL2 require a $\lambda_{SPF} + \lambda_{RF}$ or λ_{DU} of $\leq 1e^{-7} h^{-1}$

Key Vocabulary

- Safety Goal (安全目标; 1.108): Top-level safety requirement as a result of the hazard analysis and risk assessment.
 - MCU runs software correctly.
- Safe State (安全状态; 1.102): Operating mode of an item without an unreasonable level of risk.
 - MCU in reset state.
- Safety Mechanism (安全机制; 1.111): Technical solution implemented by E/E functions or elements, or by other technologies, to detect faults or control failures in order to achieve or maintain a safe state.
 - MCU initial check and periodic runtime check of internal modules.

SEooC in ISO26262

- 8.3.1 How to deal with microcontrollers in the context of ISO 26262 application
 - Microcontrollers are an integral component of modern E/E automotive systems. They can be developed as a safety element out of context (SEooC, see clause 9).

- 9.1 Safety element out of context development
 - The automotive industry develops generic elements for different applications and for different customers. These generic elements can be developed independently by different organizations. In such cases, assumptions are made about the requirements and the design, including the safety requirements that are allocated to the element by higher design levels and on the design external to the element.
 - Such an element can be developed by treating it as a safety element out of context (SEooC). An SEooC is a safety-related element which is not developed for a specific item. This means it is not developed in the context of a particular vehicle.

S32K1XX SAFETY FEATURES



MCU ASIL-B safety functions

	Description	Functional Safety Achievement
Software Execution Function	Read instructions out of flash, buffer these within instruction cache, execute instructions, read data from the SRAM or flash, buffer these in data cache, process data, write result data into SRAM or data flash	By MCU safety mechanisms
Input / Output functions	Appl. Dependant I/O functions	By system level safety measures
Not Safety related functions	For example debug	

- Safety approach (“goal”) for ASIL-B General market MCU: provide SW Execution Function (SWEF) - developed as SEooC
 - This relates to:
 - Power supplies
 - Clocks generation
 - Core platform (CM4, Cache ...)
 - Busses - XBAR
 - Memories – NVM, SRAM
 - Does not relate to:
 - I/O Peripherals – Digital and analogue
 - Communication peripherals
 - Peripheral Bus Bridge – AIPS

S32K1xx MCU – ASIL B Safety Features

Safety Hardware

- Core platform (core, DMA, cache ...), Buses - XBAR
- ECC in Flash & RAM
- Power & Clock Monitoring
- Watchdog, MPU, CRC, register protection
- Diversity of safety levels
 - LPSPi or LPSCI v FlexIO: alternate communication paths / high parallelisation
 - Analogue input monitoring: signal measuring via completely independent system resources (references, peripherals) and monitoring protection schemes

Safety Software

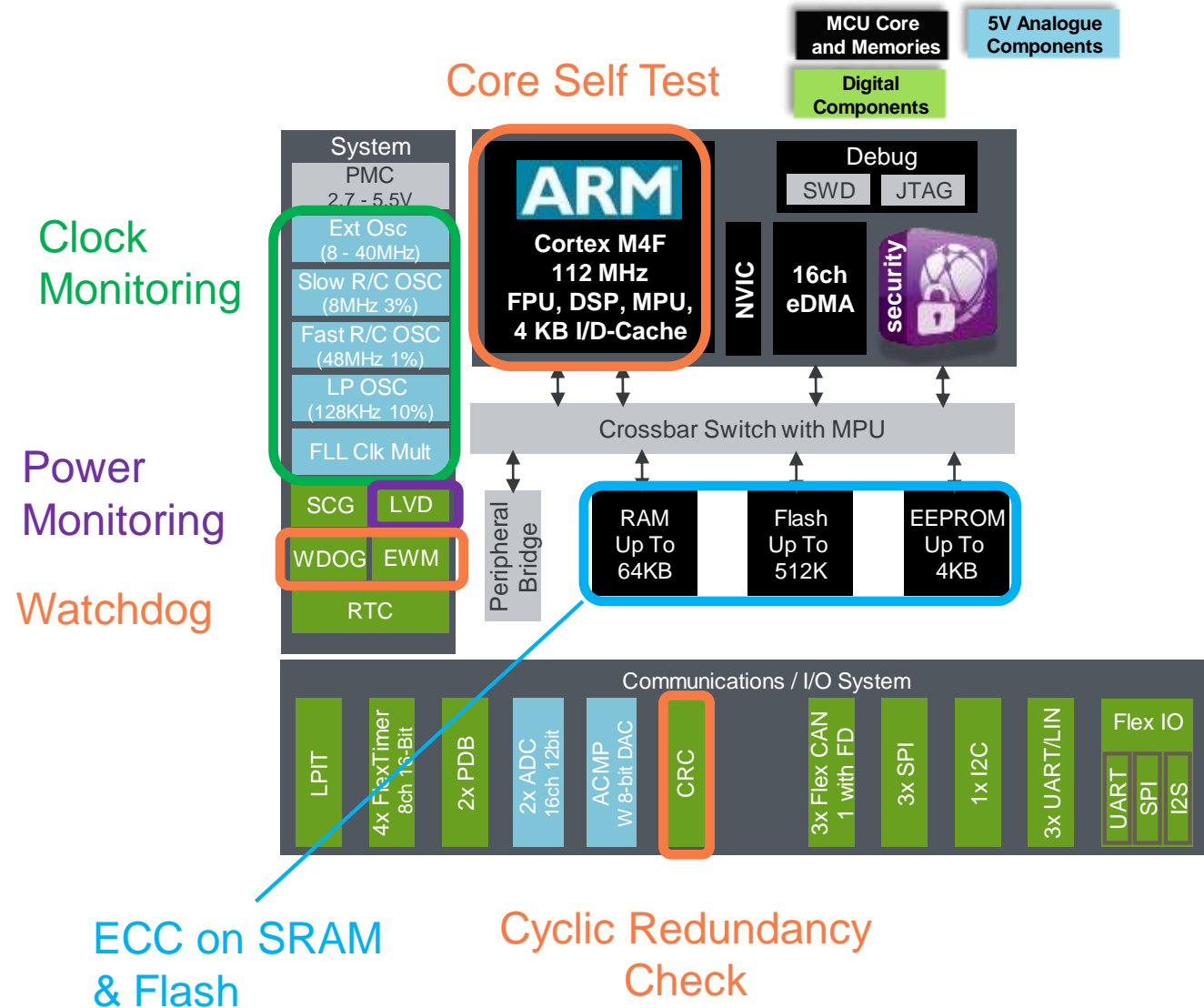
- S32K core self-test SW library
- Functional Safety Software Demo

Safety Process

- ISO 26262 development process

Safety Support

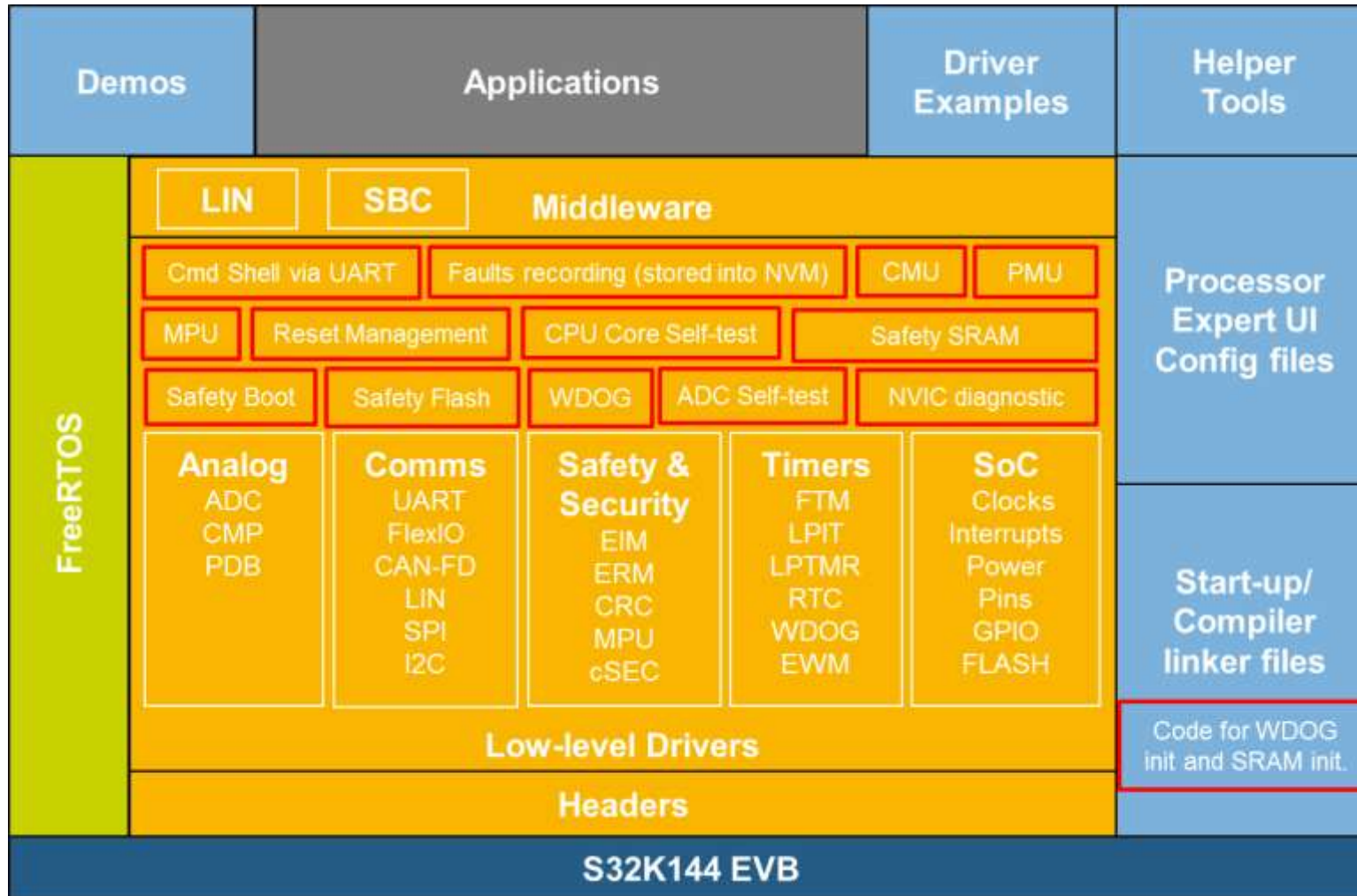
- FMEDA
- Safety manual
- Technical support



S32K1 Core Self-Test software

- Cortex-M Core Self-Test Library: Structural Core Self-Test Library (SCST) is a safety measure against permanent faults in the cores
- Developed for detecting hardware permanent faults in a core by means of executing machine op-codes with fixed set of operands and comparing their execution results
- This library is considered as Safety Element out of Context and was developed according to **ASIL B**
- SCST library provides tests to achieve the claimed diagnostic coverage (analytically estimated)

S32K14x Safety Software Block Diagram



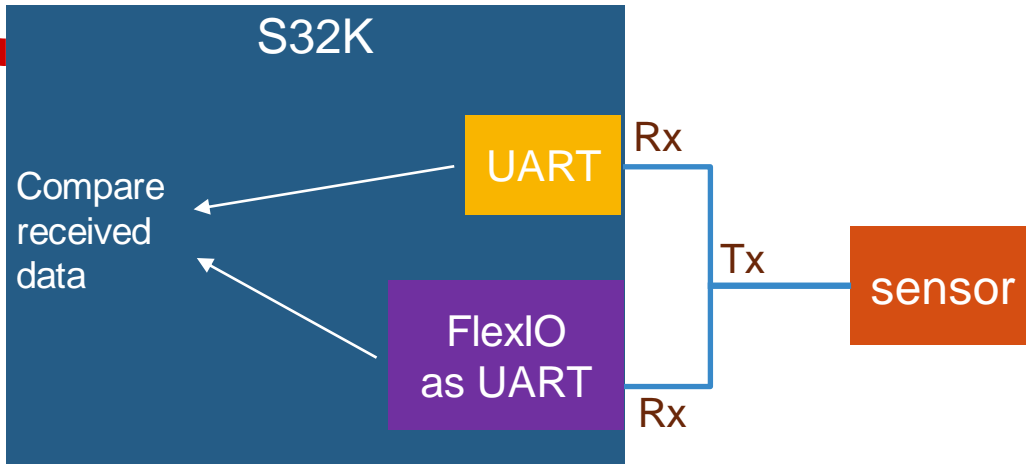
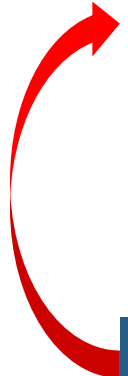
- An S32DS example projects based on S32K14x SDK.
- It demonstrates how to implement the safety features according to S32K14x safety manual recommendations.
- Safety Software User Guide includes details of software features.



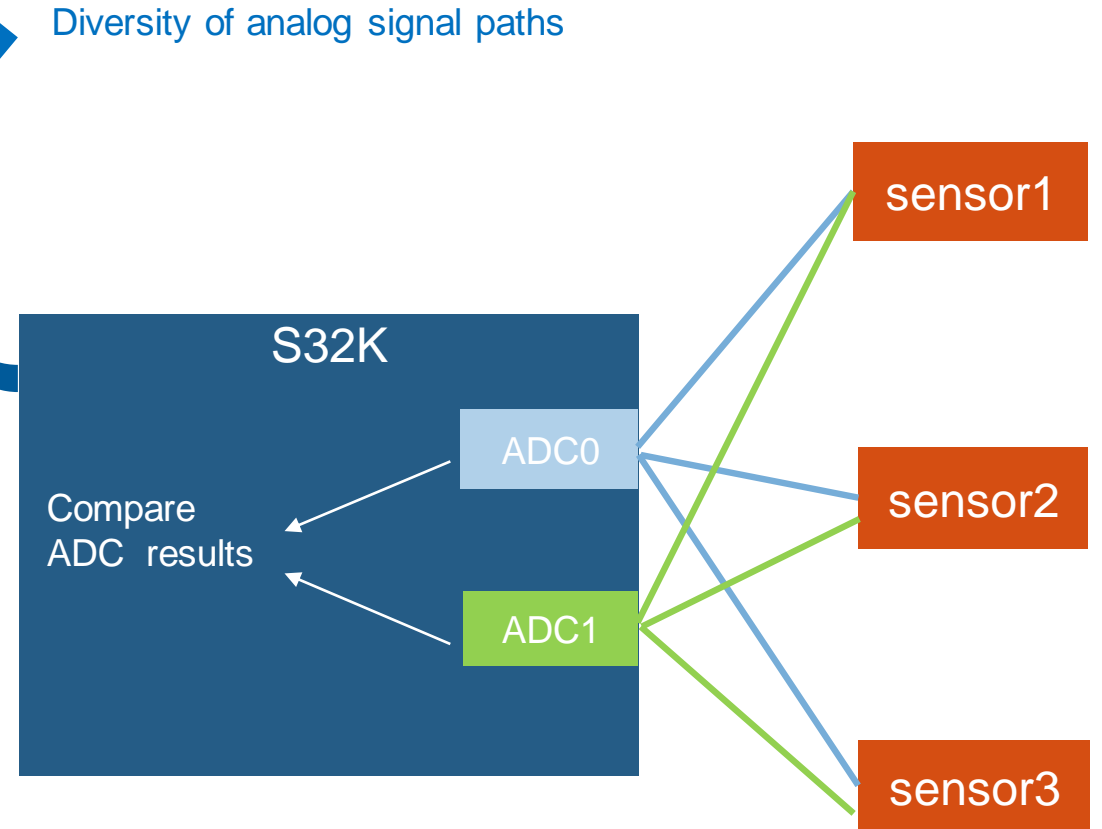
Safety measures on peripherals

Differentiating features

- end-to end CRC to detect data corruption
- Diversity of communication channel
- Internal voltages routed to ADC



Example: Emulate communication I/F (SPI, UART) using FlexIO. The UART and FlexIO's UART receive same data from sensor. This can detect failure at UART module.

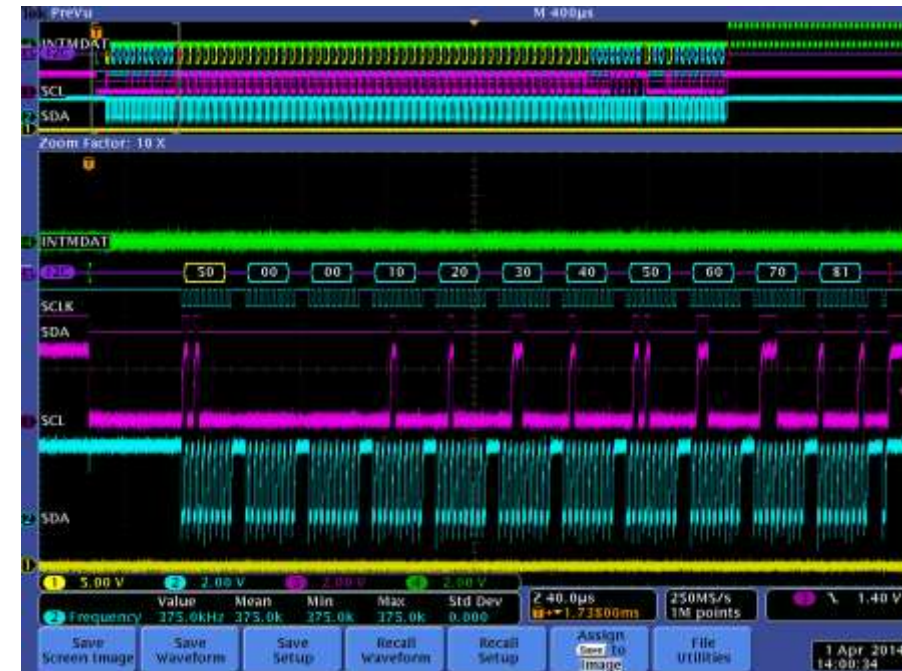


Example: dual ADC
Both ADCs receive the same sensor node

Example: FlexIO for redundant communication channels

“FlexIO” — **Flexible input and output peripheral**

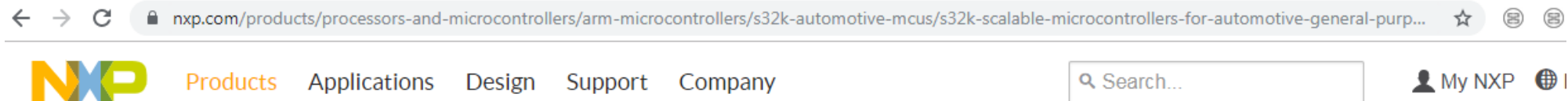
- Highly configurable module providing a wide range of functionality including:
 - Emulation of a variety of communication protocols: UART, I2C, SPI, I2S, etc.
 - Flexible 16-bit timers with support for a variety of trigger, reset, enable and disable conditions
- Creates an interlink between GPIO method of software emulation and exact hardware peripheral module
- Can continue operating under debug / stop modes
- Support of polling/interrupt/DMA (RX/TX) operation
- Low - medium software/CPU overhead
- The FlexIO peripheral was initially introduced on the NXP Kinetis KL43 family
- Multiple App Notes available – search FlexIO application notes on NXP.com



FUNCTIONAL SAFETY DOCUMENTS



Access Functional Safety Documents – Step 0: NXP Website



Processors and Microcontrollers > Arm Microcontrollers > S32K Automotive MCUs > S32K General Purpose MCUs

S32K: Scalable Microcontrollers for Automotive General Purpose and high-reliability industrial

Follow [Email] [Share]

OVERVIEW	DOCUMENTATION	TOOLS & SOFTWARE	BUY/PARAMETRICS	PACKAGE/QUALITY	TRAINING & SUPPORT
----------	----------------------	------------------	-----------------	-----------------	--------------------

Filter By | [Show All](#)

Filter by keyword

Recommended Documentation (3)

Data Sheet (1)

Application Note (23)

Application Note Software (4)

Recommended Documentation (3)

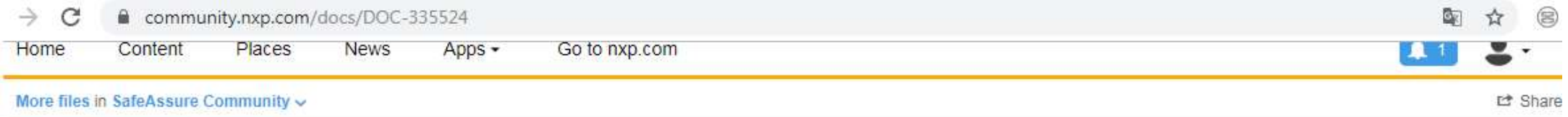
Name/Description	Type	Modified Date
S32K1xx MCU Family Fact Sheet (REV 4) PDF 702.7 kB S32K_FLYR [English, 中文, 日本語]	Fact Sheet	14 Oct 2019
S32K1xx MCU Family - Reference Manual (REV 11) PDF 12.7 MB S32K1XXRM [English]	Reference Manual	16 Jul 2019

Supporting Information (1)

Name/Description	Modified Date
Functional Safety documents AVAILABLE Require access to the SafeAssure NDA group (REV 1) URL 1kb SAFEASSURE-NDA-DOCS [English]	28 May 2019



Access Functional Safety Documents – Step 1: NDA request form



SafeAssure NDA group - request form

Like • 0

File uploaded by [Toño Hernández](#) on Oct 24, 2017 • Last modified by [Toño Hernández](#) on Jul 3, 2019

Version 7



Request your access to our [SafeAssure NDA group](#) to get safety documents and receive expert support for your functional safety applications.

1. Download the our NDA request form (.pdf file)
2. Fill all blank spaces and handwrite your signature
3. Send it back (as attachment) to safeassure@nxp.com























We will process your request and reply back in less than 10 days.

Access Functional Safety Documents – Step 2

- Visit SafeAssure NDA group and download the documents: Safety Manual and FMEDA

community.nxp.com/docs/DOC-335435

32-bit, based on ARM® S32 MCUs

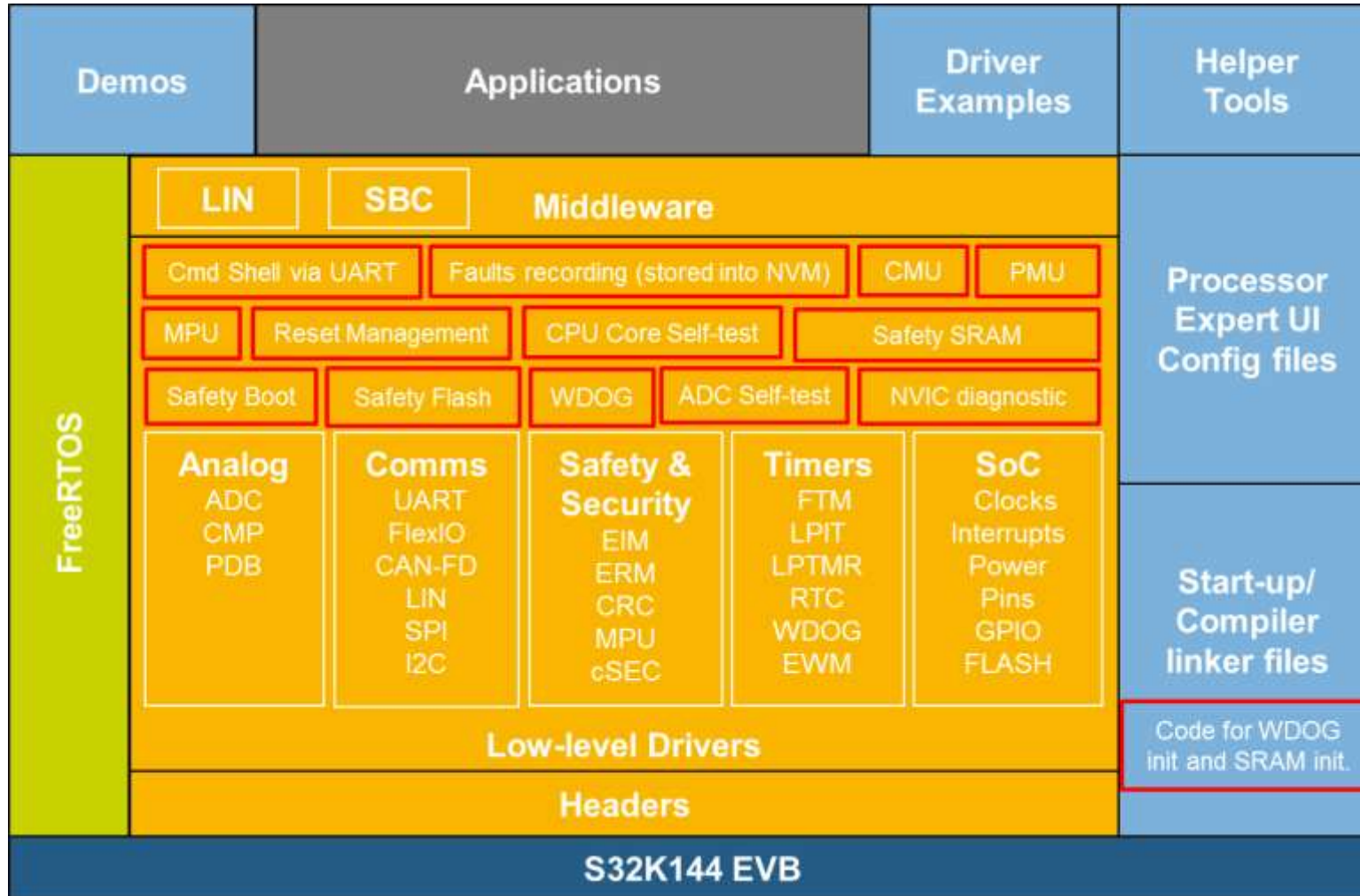
	Safety Manual	SEooC Standardized FMEDA	Analysis Report	Assessment / Confirmation Measure Report	PPAP
	S32K1xx 				
	S32K142 	S32K142 	S32K142 	S32K142 	S32K142 
S32K14x 	S32K144 	S32K144 	S32K144 	S32K144 	S32K144 
	S32K146 	S32K146 	S32K146 	S32K146 	S32K146 
	S32K148 	S32K148 	S32K148 	S32K148 	S32K148 



S32K14X SAFETY SOFTWARE DEMO



S32K14x Safety Software Block Diagram



The S32K14x safety software implements the following safety features:

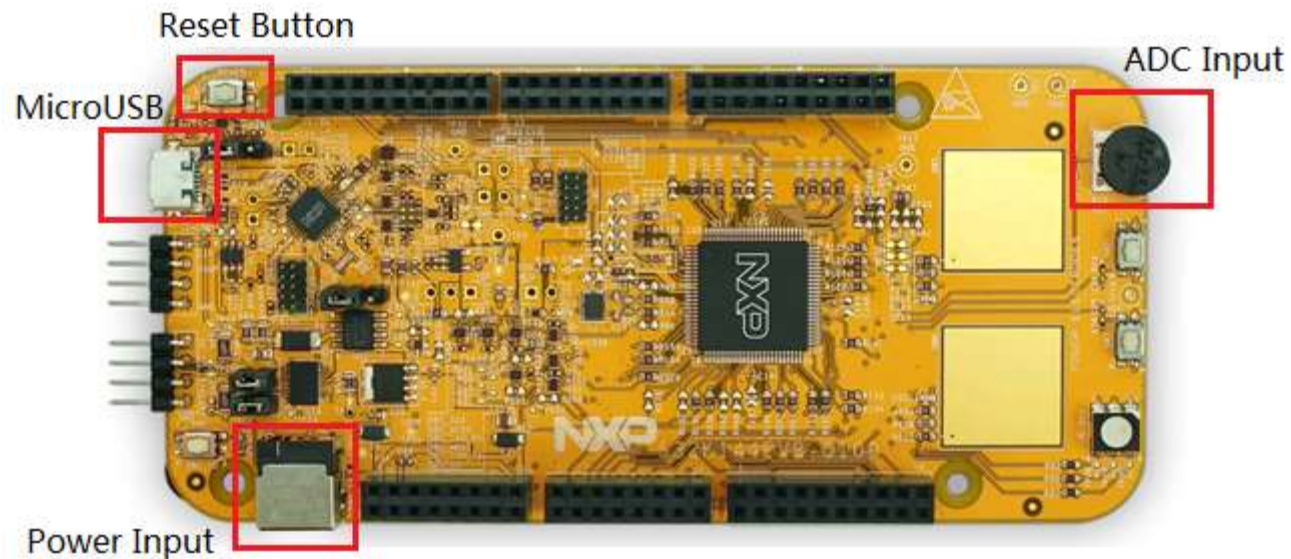
- Core self-test
- Power supply monitor
- Clock monitor
- Flash protection and error handling
- SRAM error handling
- MPU error handling
- WDOG test and error record
- Safety Boot
- Stack overflow monitor
- NVIC diagnostic
- ADC diagnostic
- Register data integrity check (PORT registers CRC test)
- CRC module driver



Safety Software Demo Environment

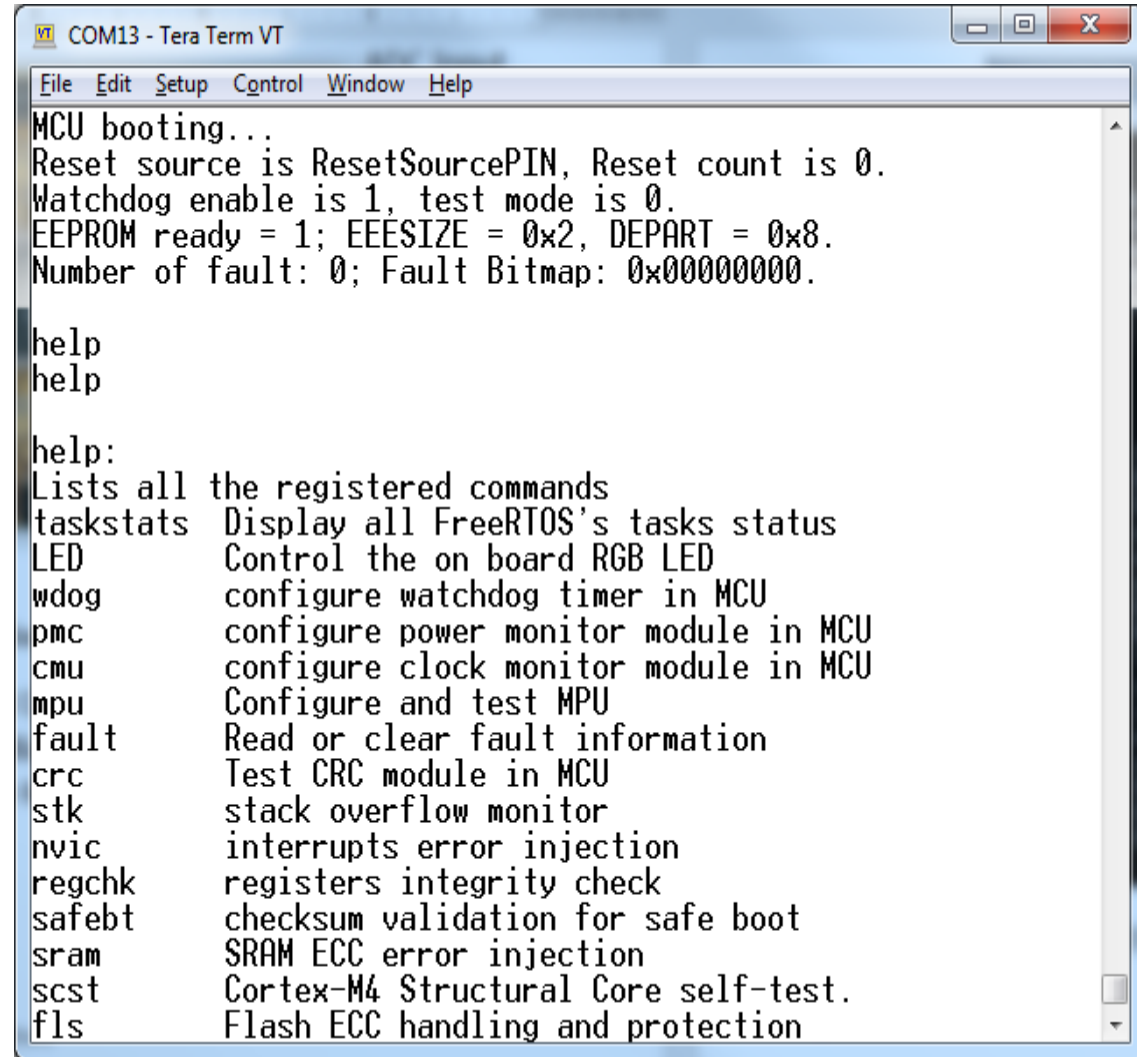
The safety software is based on an S32DS project with following features:

- S32K14x SDK 3.0.0;
- FreeRTOS with CLI command-line shell is integrated in the project;
- MCU faults injection and diagnostic can be triggered by UART commands from PC;
- Faults are saved in MCU EEPROM. (All data Flash and FlexRAM are used for EEPROM)
- One task (100ms interval) in FreeRTOS to run diagnostic API functions;



UART Tool on PC to Control the Demo

- Connect the S32K144 EVB and the PC with a micro USB cable;
- Use a UART command-line tool to inject faults and view diagnostic information.



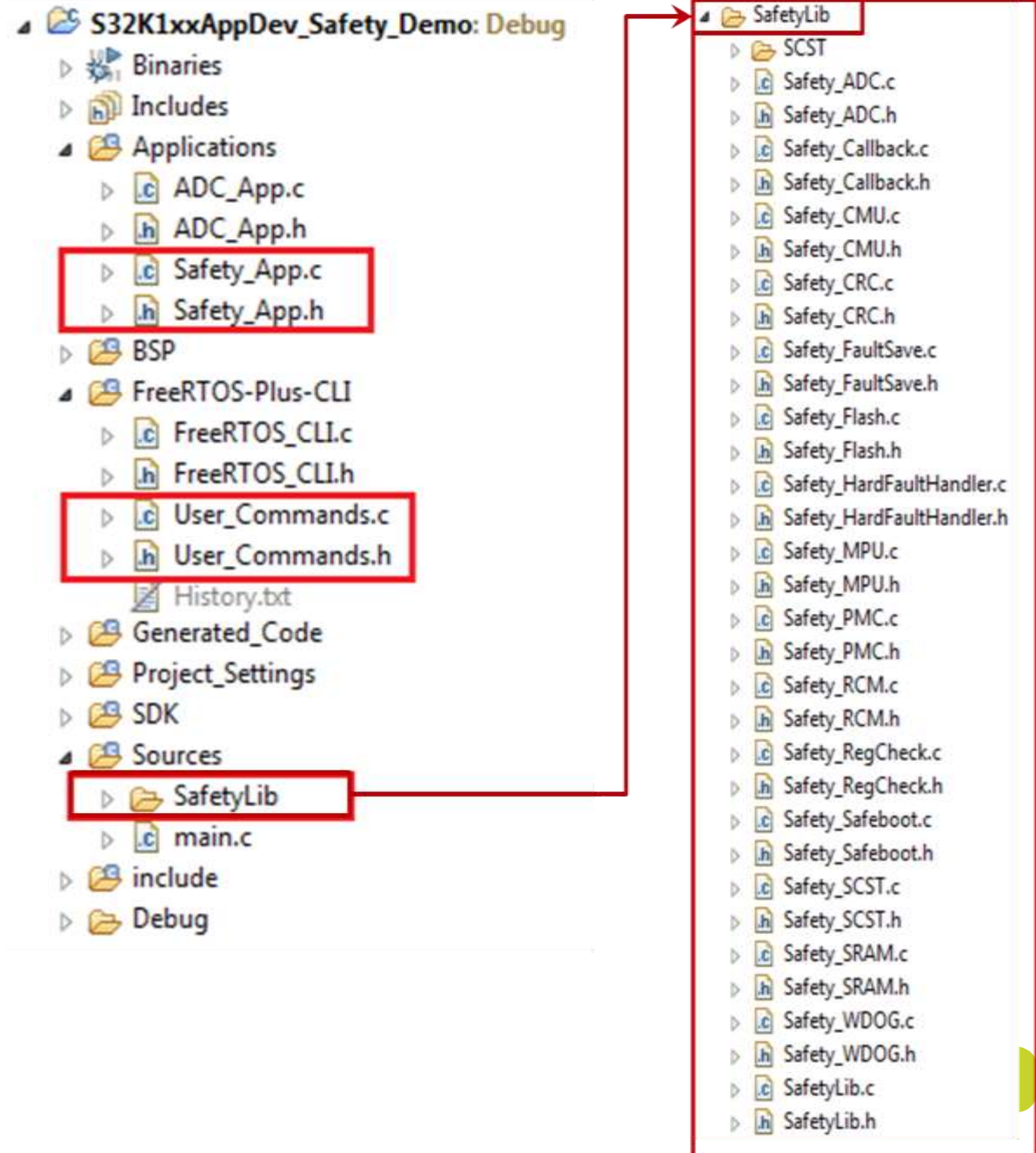
```
COM13 - Tera Term VT
File Edit Setup Control Window Help
MCU booting...
Reset source is ResetSourcePIN, Reset count is 0.
Watchdog enable is 1, test mode is 0.
EEPROM ready = 1; EEESIZE = 0x2, DEPART = 0x8.
Number of fault: 0; Fault Bitmap: 0x00000000.

help
help

help:
Lists all the registered commands
taskstats Display all FreeRTOS's tasks status
LED Control the on board RGB LED
wdog configure watchdog timer in MCU
pmc configure power monitor module in MCU
cmu configure clock monitor module in MCU
mpu Configure and test MPU
fault Read or clear fault information
crc Test CRC module in MCU
stk stack overflow monitor
nvic interrupts error injection
regchk registers integrity check
safebt checksum validation for safe boot
sram SRAM ECC error injection
scst Cortex-M4 Structural Core self-test.
fls Flash ECC handling and protection
```

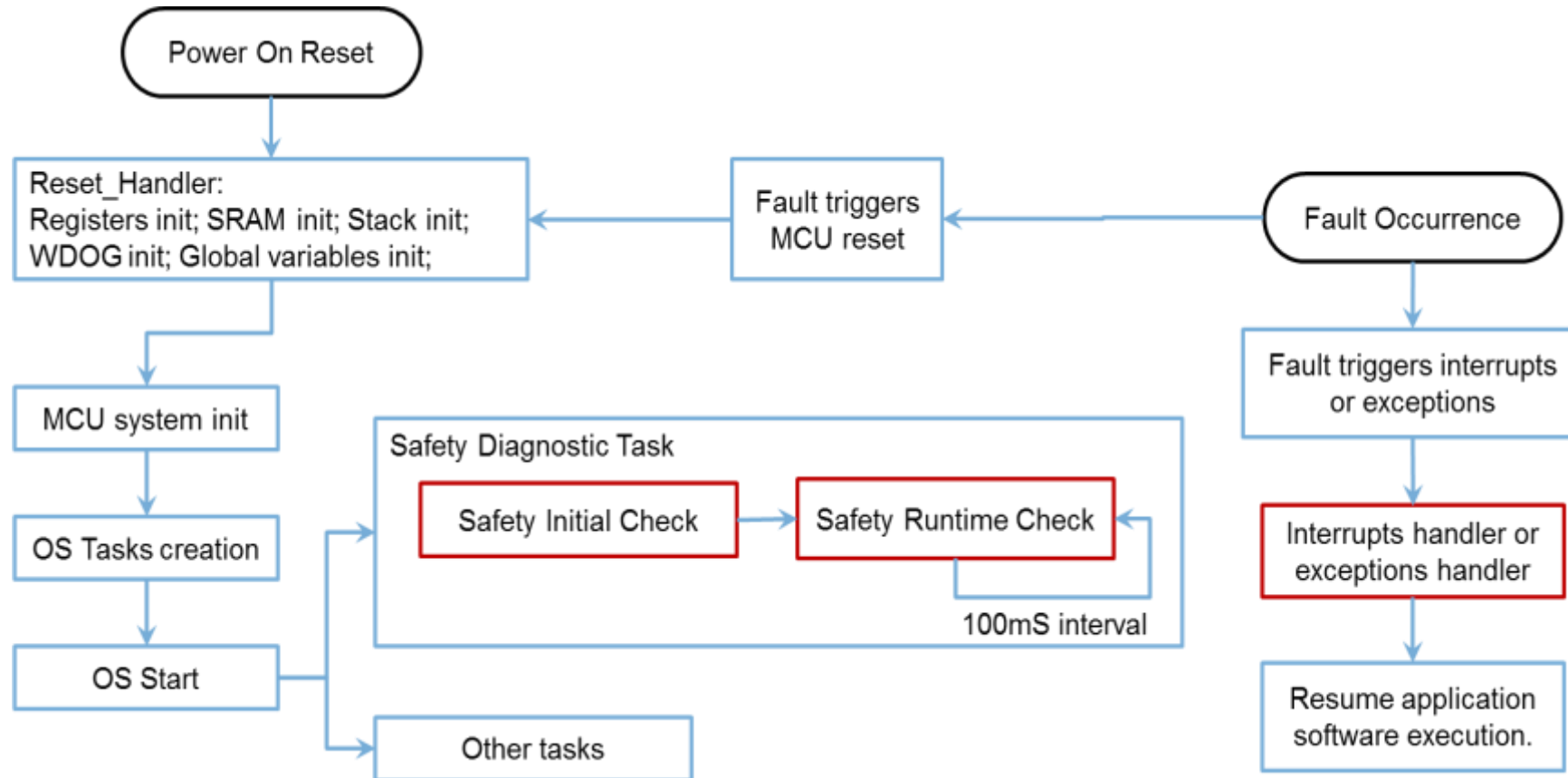
Files and Paths Structure

- The folder “SafetyLib” includes all the safety software library source code files.
- The file “Safety_App.c” demonstrates how to use the safety software in the application software.
- The file “User_Commands.c” shows how to inject faults and verify the faults detection.



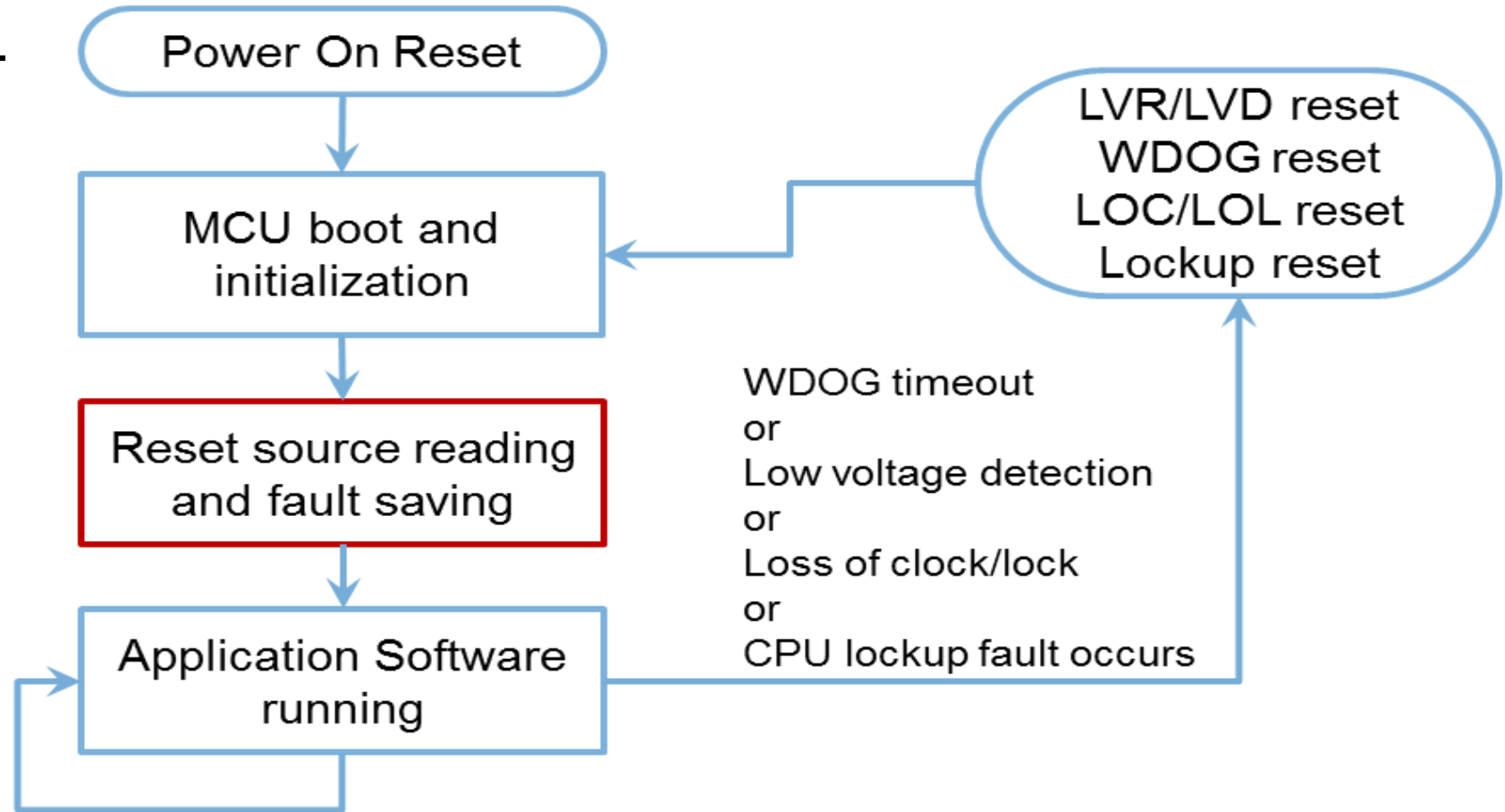
Software Overall Structure

- Safety initial check, Safety runtime check, exceptions handler;



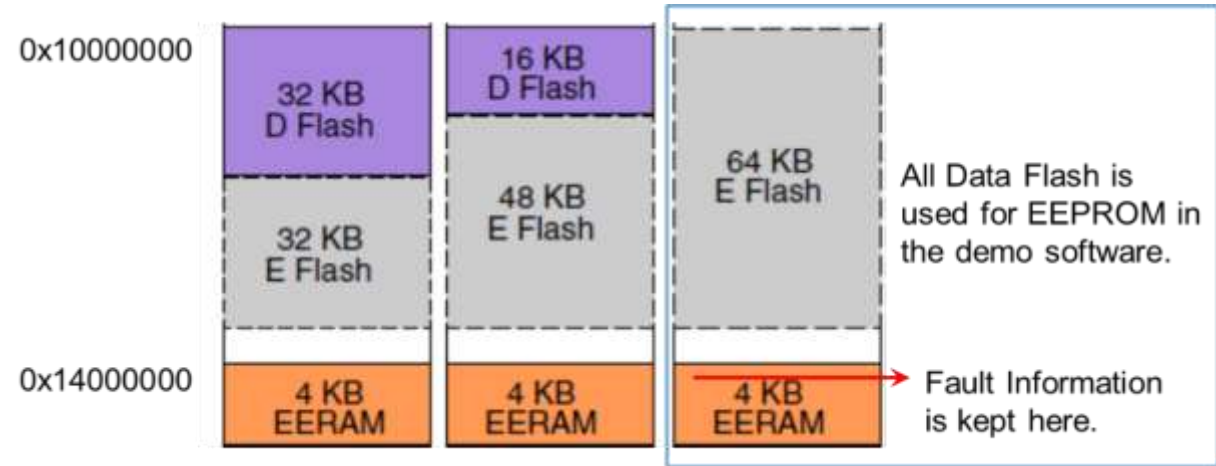
Safety Software Component – Reset Management

- Reset source is recorded.
- Some resets are interpreted as faults.



Safety Software Component – Fault Information Management

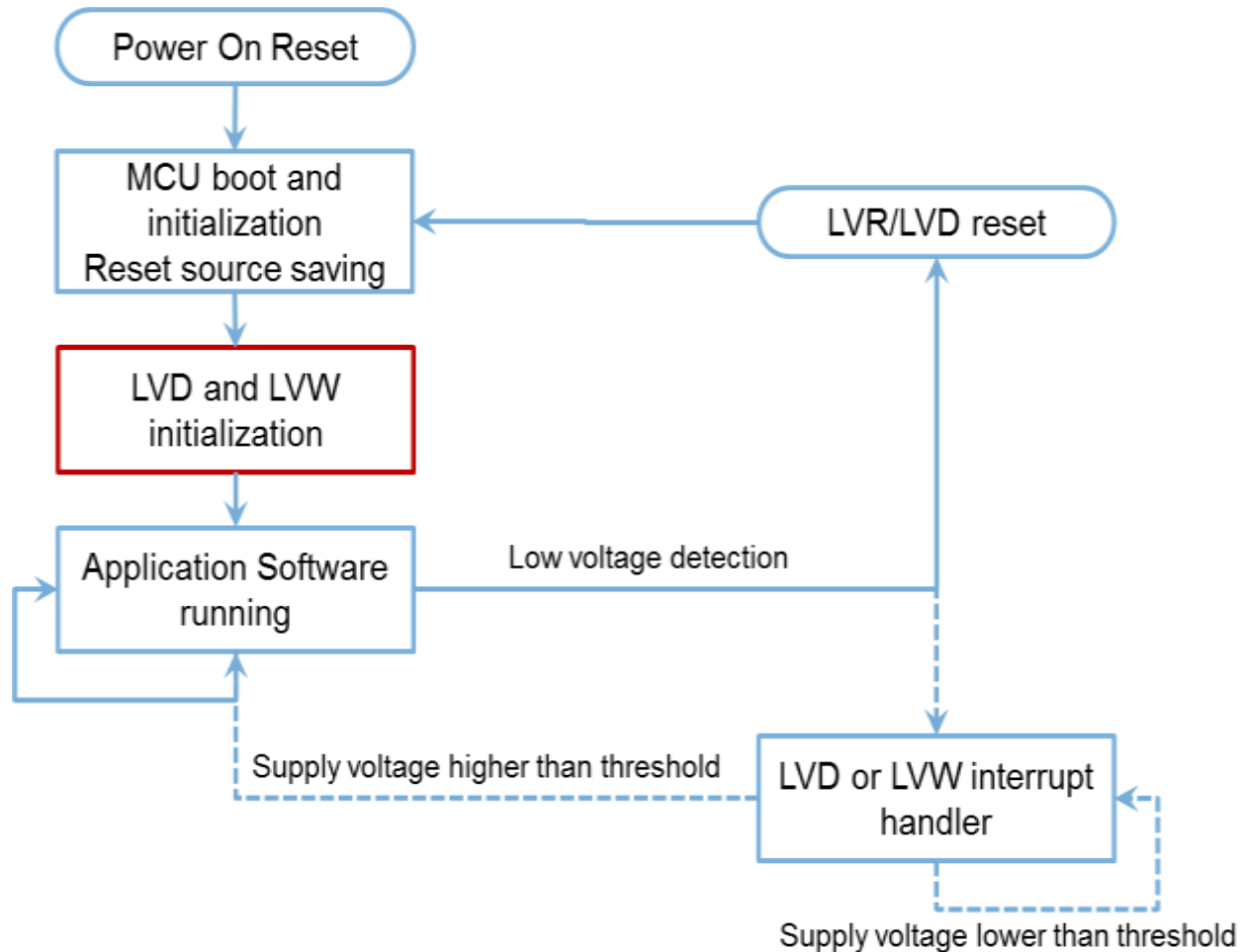
- Faults information is saved in EEPROM;
- Fault type, PC value, Fault Data address are recorded;
- Each fault can be cleared;
- Total number of faults are configurable;



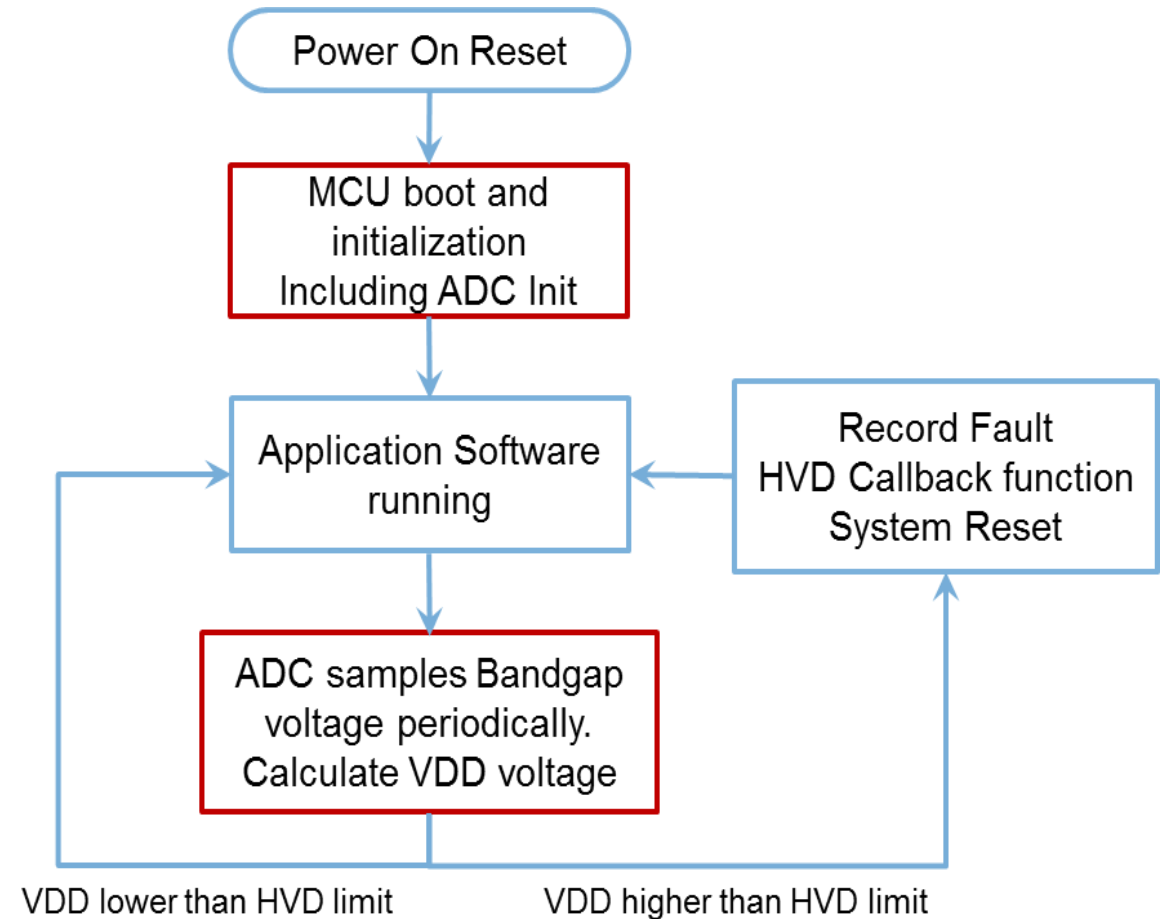
Fault 0	Fault Type (32bit)	Fault PC Addr (32bit)	Fault Data Addr (32bit)
Fault 1	Fault Type	Fault PC Addr	Fault Data Addr
Fault 2	Fault Type	Fault PC Addr	Fault Data Addr
.....
Fault 9	Fault Type	Fault PC Addr	Fault Data Addr

Safety Software Component – Power Supply Monitor (LVD/HVD)

- LVD is supported by HW.

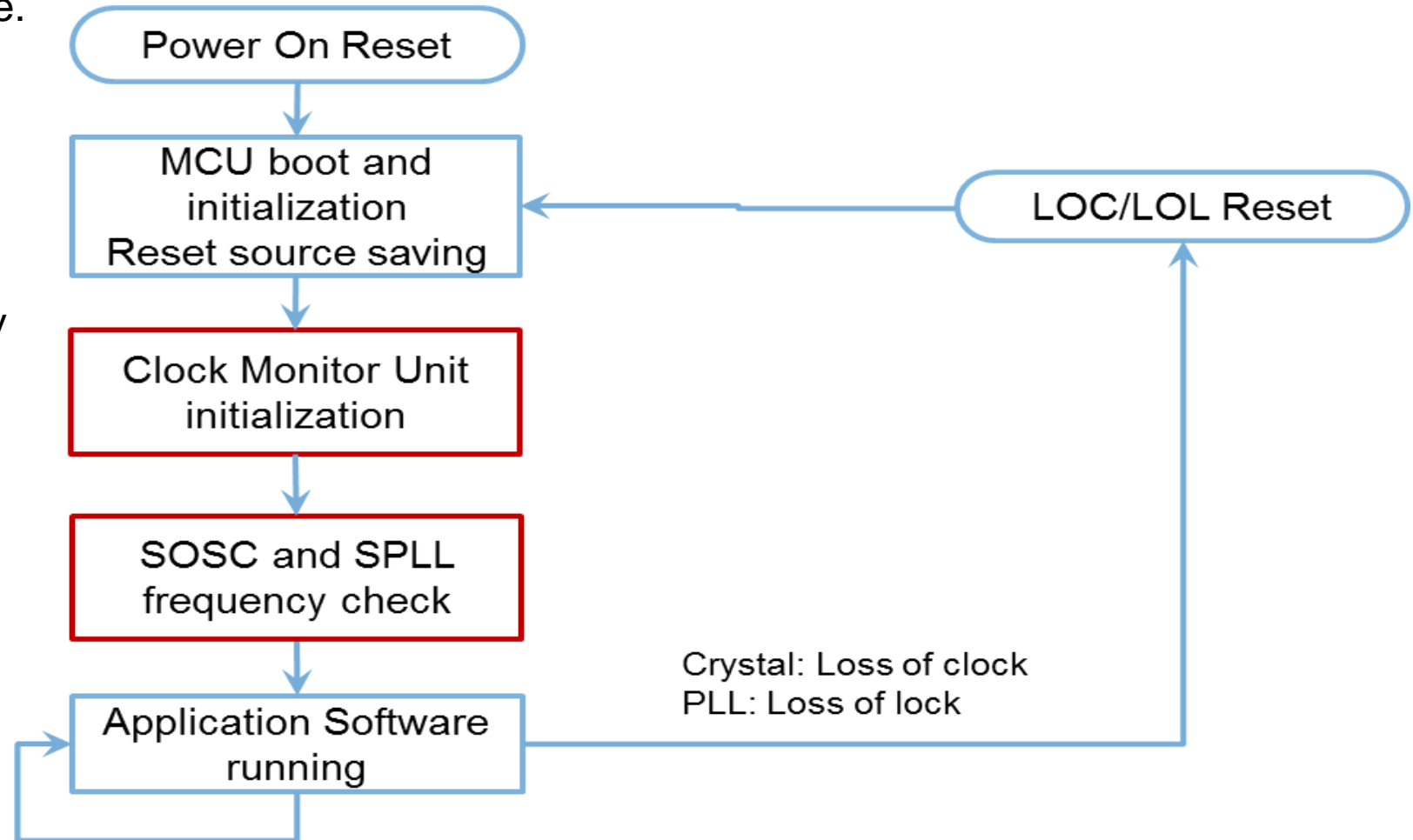


- HVD is supported by SW.



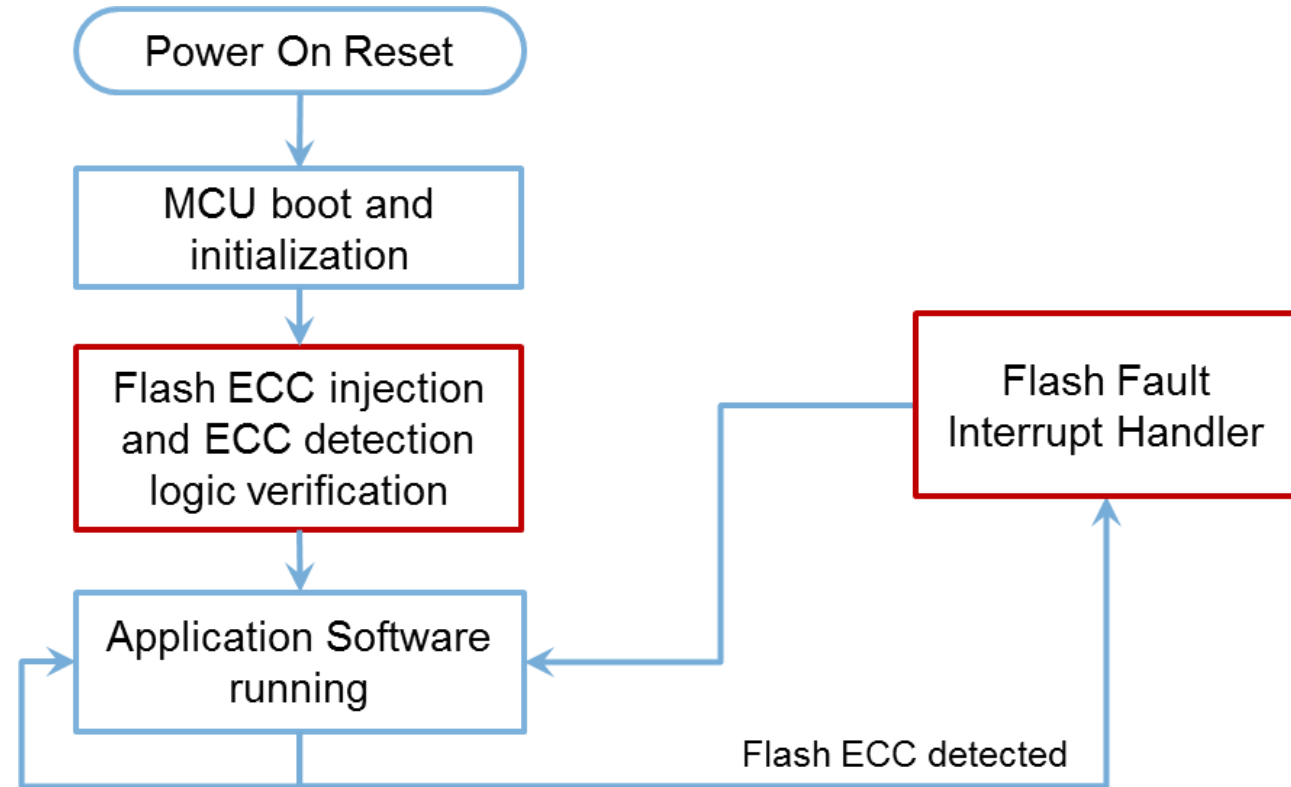
Safety Software Component – Clock Monitor (CMU)

- CMU monitors system clock using IRC clock as reference.
- Loss of clock fault causes MCU to reset.
- Software uses timer to verify SOSC and SPLL frequency is in desired range.



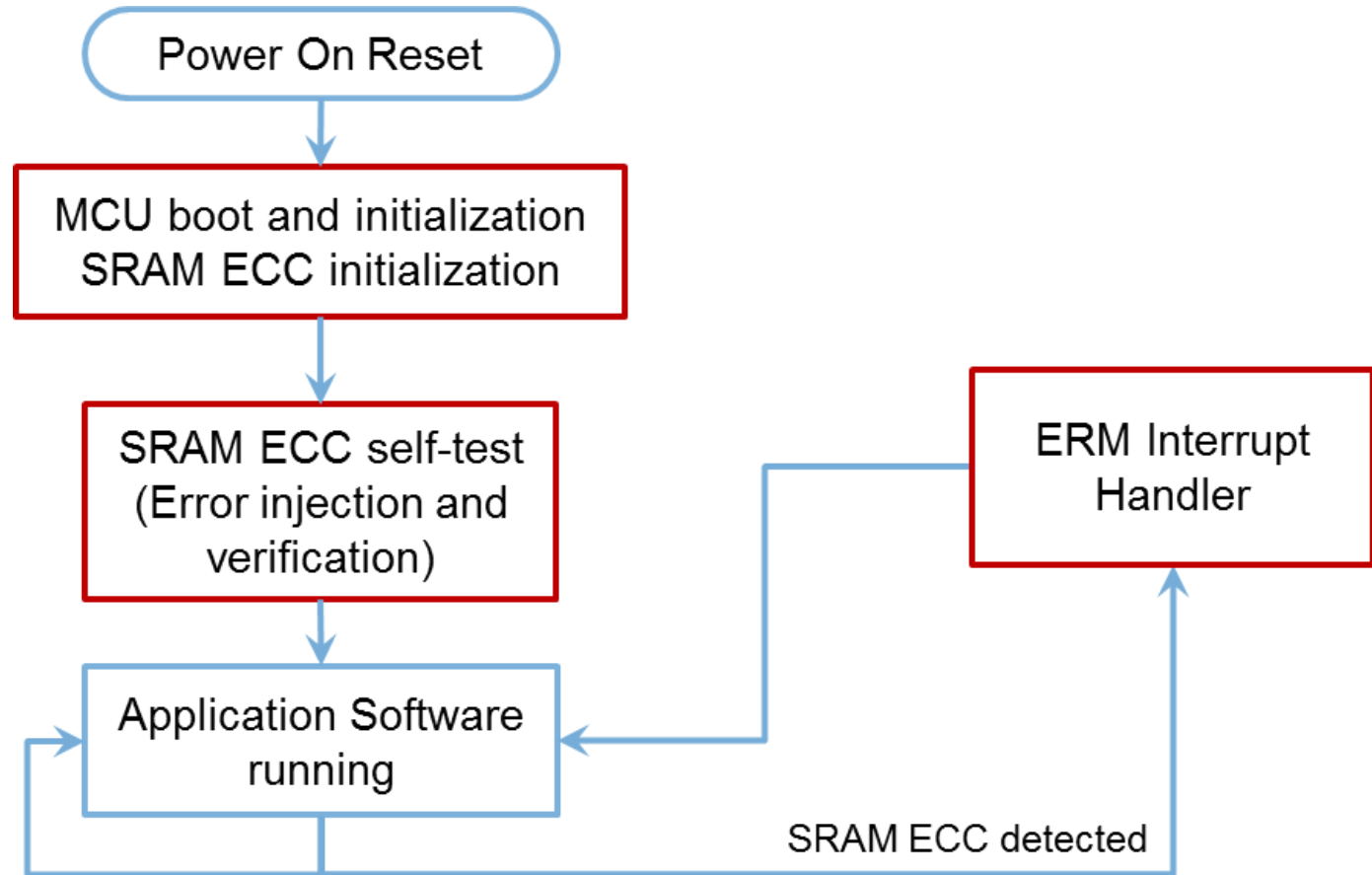
Safety Software Component – Flash ECC

- Flash ECC mechanism is checked during MCU initialization.
- Flash ECC error is reported through Flash fault interrupt handler.



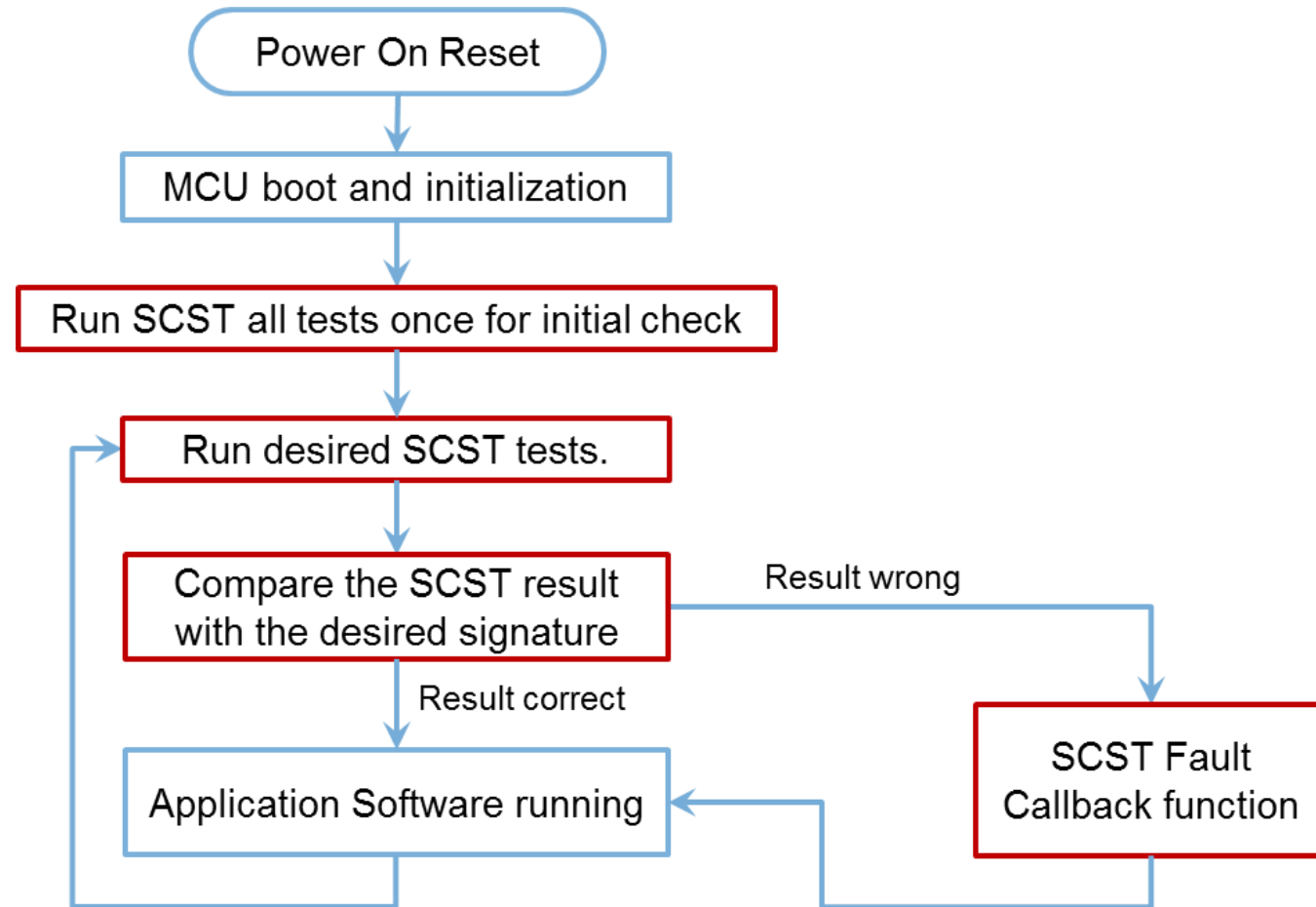
Safety Software Component – SRAM ECC

- SRAM ECC mechanism is checked during MCU initialization.
- SRAM ECC error is reported through ERM module.



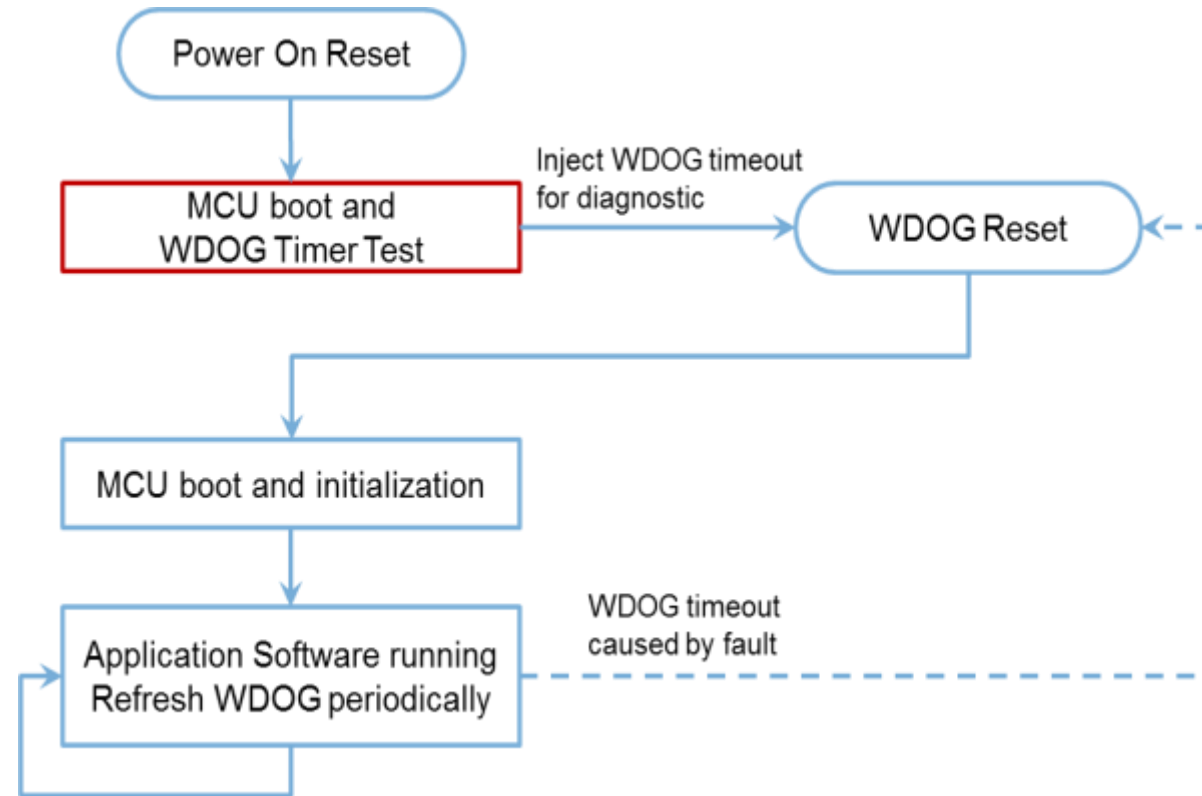
Safety Software Component – SCST (core self-test)

- CPU executes the specific sequences of instructions and the results are compared with expected values.
- Total 44 tests can be grouped as following:
 - Exceptions handling;
 - ALU operations verification;
 - CPU registers check;
 - Branch instructions verification;
 - Load and store tests;
 - MAC commands tests;
 - Double and SIMD saturation commands tests;
 - CPU status flags check;
 - CPU fetch unit test;
- Customer can run any single test or run a batch of tests.



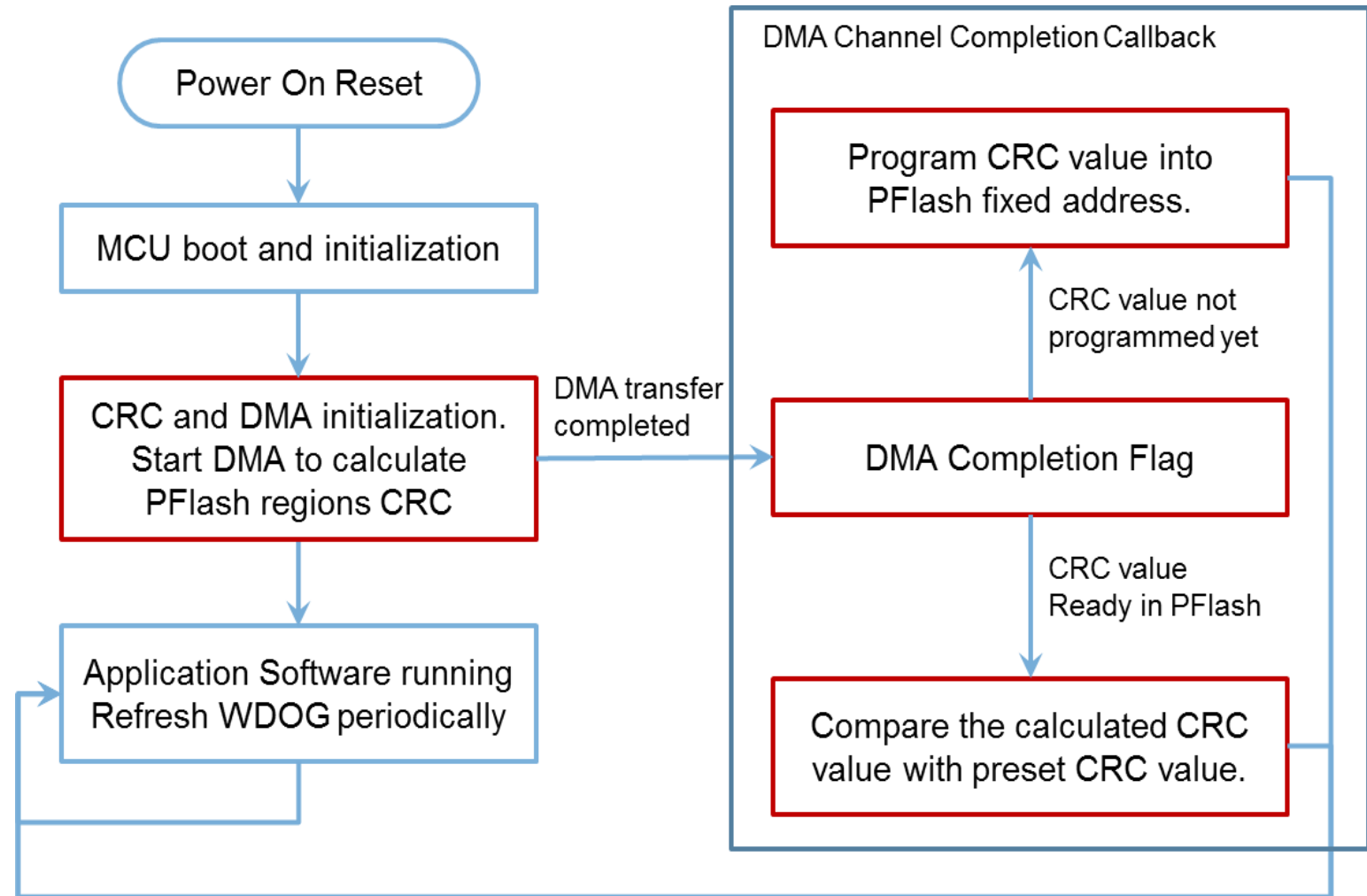
Safety Software Component – WDOG Test

- MCU software initialization process for POR is different from that of other resets;
- WDOG should be tested after each POR according to safety manual requirement.



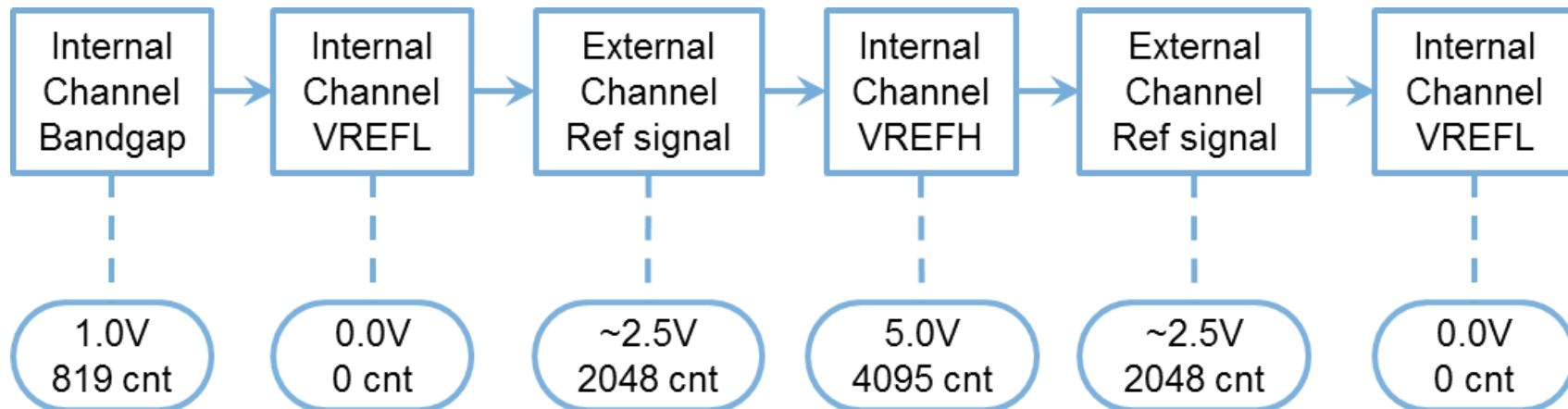
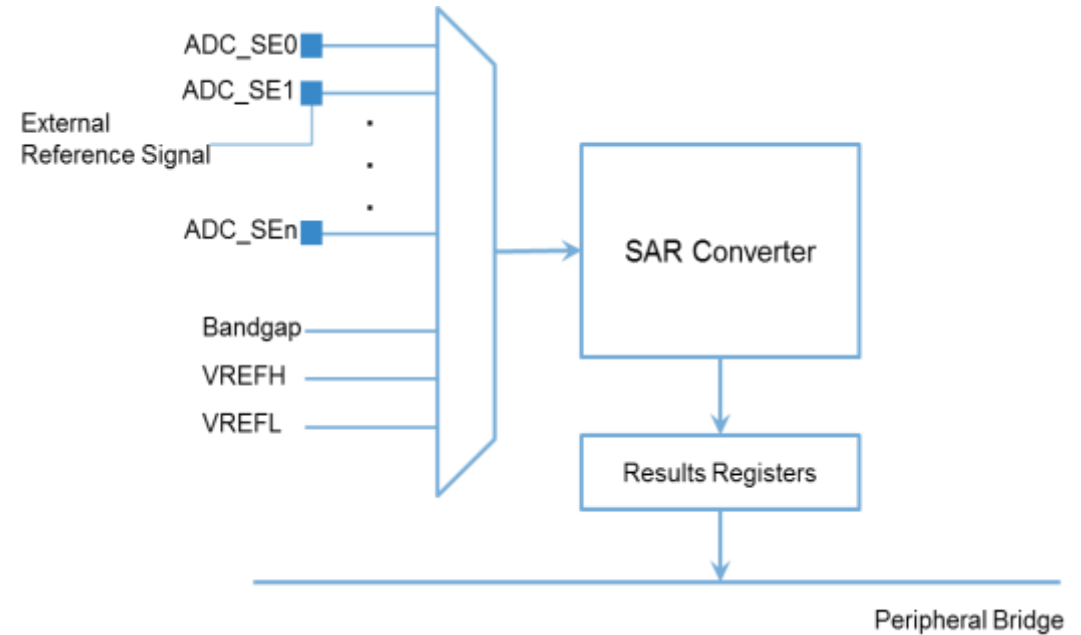
Safety Software Component – Safety Boot

- Use CRC module to calculate CRC value of the specified flash regions;
- This operation can assure the integrity of customer's firmware;



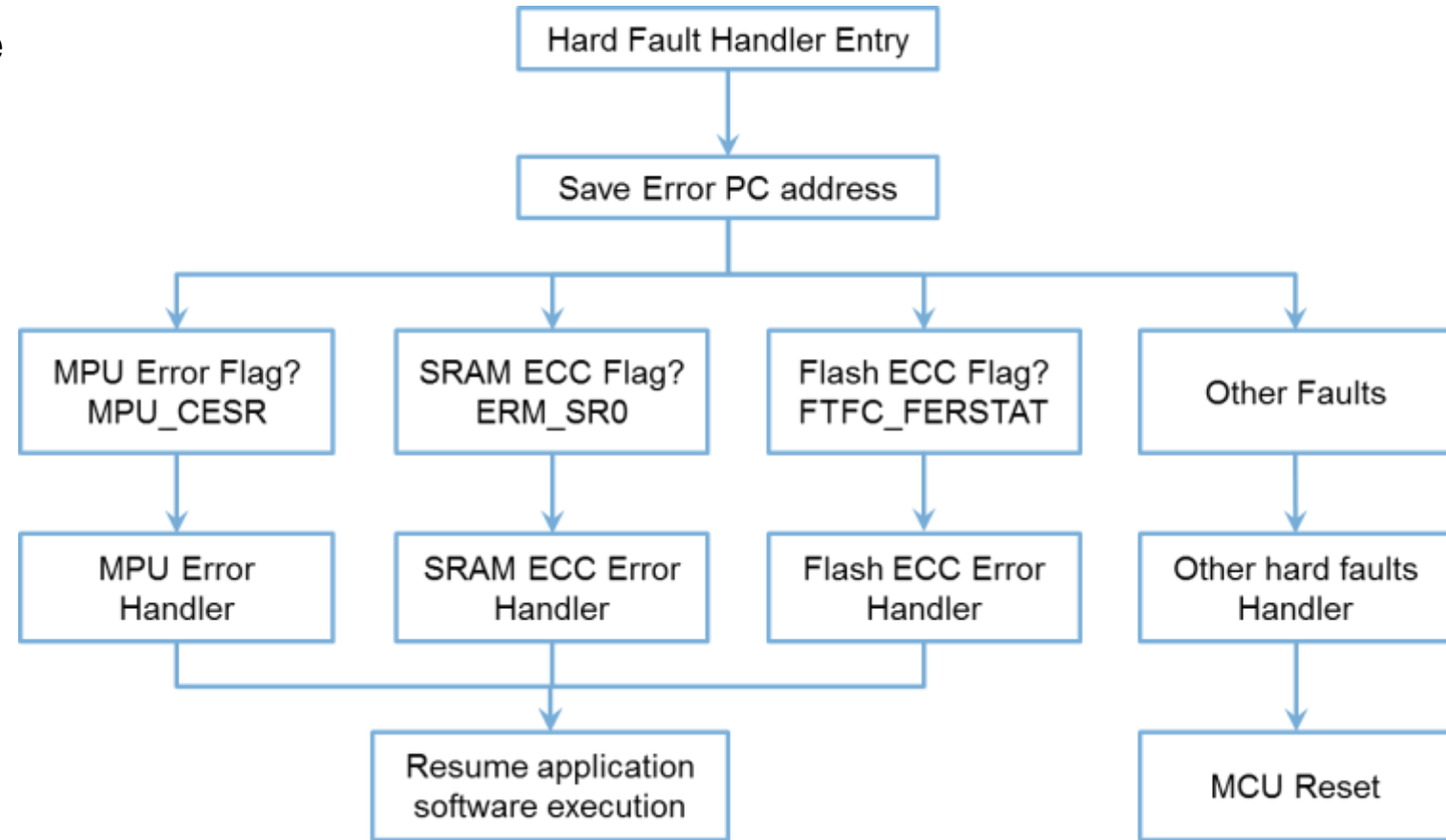
Safety Software Component – ADC Self-Test

- Use ADC to sample signals at specific voltages and verify its conversion results;
- ADC Sampling different channels to check gate stuck faults;



Safety Software Component – Hard Fault Handling

- Hard Fault Handler will handle the following exceptions:
 - MPU violation error;
 - SRAM ECC error;
 - Flash ECC error;
 - Other hard faults;



Software Integration

- SDK Components settings import;
- Safety software source code import;
- Linker file modifications;
- Symbol and Paths include settings in S32DS;
- Call Safety software APIs:
 - Call safety initial check during MCU init;
 - Call safety runtime check in periodic task;



Safety Software Performance

Code size for safety features:

- SCST code size is ~60KB;
- Non-SCST safety software code size is ~6.8KB;

CPU Loading:

- Safety features initialization takes total 3.5ms;
 - SRAM ECC initialization takes 0.71ms;
 - WDOG reset test lower byte takes 0.93ms;
 - WDOG reset test higher byte takes 0.93ms;
 - Safety features initialization takes 1.64ms;
 - Safe Boot takes 8.62ms, done by DMA, generate CRC for 183K Flash;
- Safety Monitor Task takes 1.68ms in every 100mS cycle. The task includes:
 - WDOG refreshing
 - Power Supply HVD monitor
 - Port PCR register check
 - CPU SCST test (1.2ms)



SECURE CONNECTIONS
FOR A SMARTER WORLD