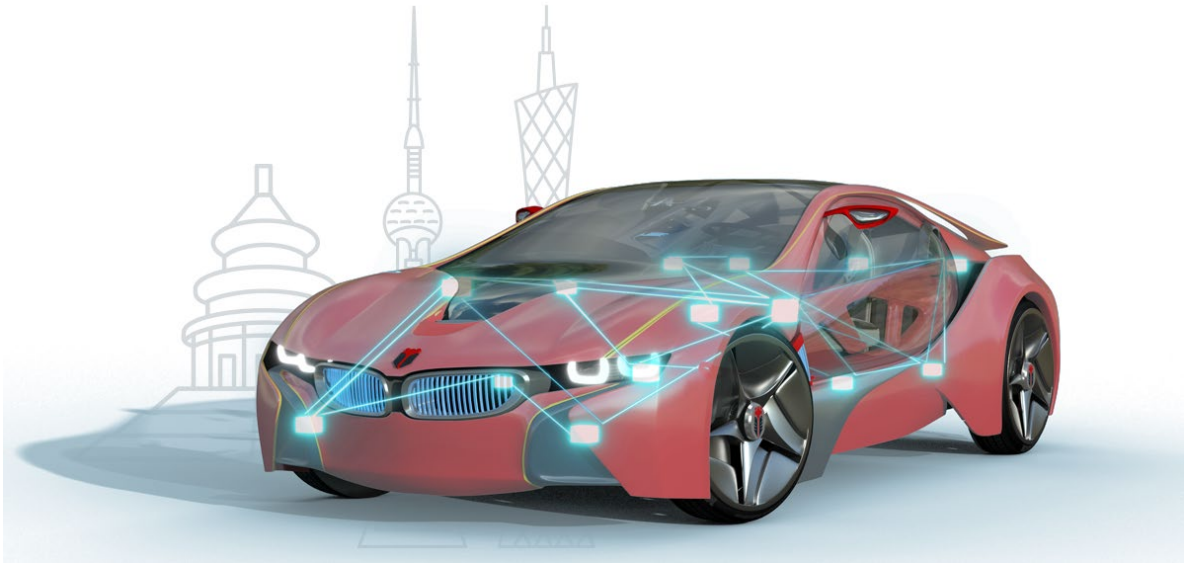
恩智浦 S32K3 SAF 和 SCST 功能安全库的应用

APPLICATION OF NXP S32K3 SAF AND SCST SAFETY LIBRARIES



1 SAF 功能概述(OVERVIEW OF SAF)

S32K3 safety software 是由 NXP 官方提供的软件包，用于帮助客户实现基于 S32K3 芯片的汽车电子控制器产品的功能安全要求。S32K3 safety software 满足 S32K3xx Safety Manual 的要求的大部分芯片层级的功能安全需求，其中包括了 SAF (safety software framework) / SPD (safety peripheral driver) / SCST (core self-test code) / RTD (Real-Time Drivers)内容。

The S32K3 safety software is an official software package provided by NXP, designed to help customers meet functional safety requirements for automotive electronic control unit products based on the S32K3 microcontroller. The S32K3 safety software fulfills most chip-level functional safety requirements specified in the S32K3xx Safety Manual, which includes SAF (Safety Application Framework), SPD (Safety Peripheral Driver), SCST (Core Self-Test Code), and RTD (Real-Time Drivers).

S32 安全软件框架在硬件安全层和服务安全层提供了多个软件模块，具体如下：

- BIST：上电自检管理模块，包含了逻辑自检（LBIST）和内存自检（MBIST）。
- eMCEM：可扩展微控制器错误管理模块，其中集成了 S32K3 芯片的 FCCU 模块驱动功能。
- Mode Selector：模式选择模块，mSel 会根据错误源信息进行安全分析，并根据分析结果来决定 MCU 运行状态。
- sBoot：安全启动模块，负责检查上电初始化完毕后各个安全相关的配置寄存器的配置值是否正确。
- SquareCheck：二次检查模块，用于针对硬件机制进行故障注入测试。
- SW Recovery：软件恢复模块，负责在检测出关键故障时执行芯片的功能性复位。

The S32 Safety Software Framework provides multiple software modules at both hardware safety layer and service safety layer, as follows:

- BIST: Power-on self-test management module, including Logic Built-In Self-Test (LBIST) and Memory Built-In Self-Test (MBIST).
- eMCEM: Extensible Microcontroller Error Management module, which integrates the FCCU (Fault Collection and Control Unit) module driver functionality of S32K3 microcontroller.
- Mode Selector: Mode Selection Module. mSel performs a safety analysis based on the error source information and decides the MCU operation state based on the analysis result.
- sBoot: Secure Boot module, responsible for verifying the correct configuration values of safety-related registers after power-on initialization.
- SquareCheck: Secondary check module, used for fault injection testing of hardware mechanisms.

- SW Recovery: Software Recovery module, responsible for executing functional reset of the microcontroller when critical faults are detected.

1.1 集成功能安全库后的启动流程概述(OVERVIEW OF STARTUP SEQUENCE AFTER SAFETY LIBRARIES INTEGRATION)

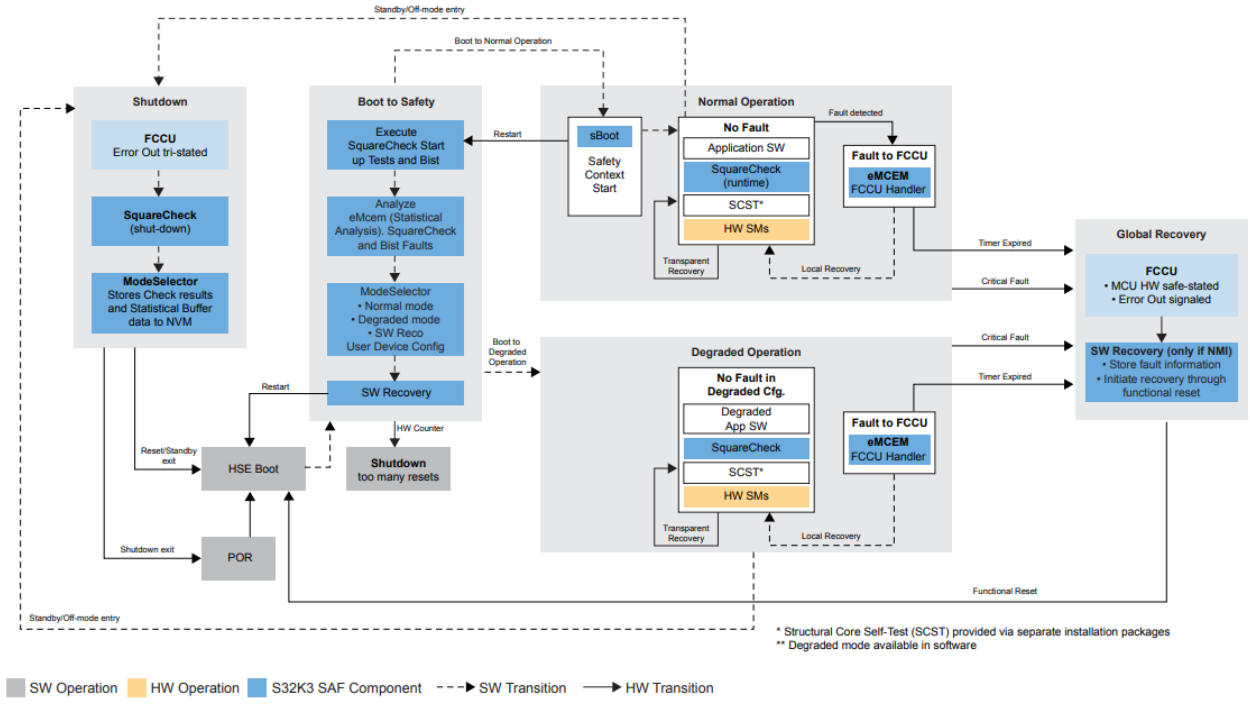


图 1.1 启动流程框图

Figure 1.1 Block diagram of the startup process

1.1.1 POR

POR，即 Power On Reset 是芯片复位类型中的破坏性复位（Destructive Reset）复位类型的一种。MCU 在该阶段正式启动，如果开启了 HSE 功能，则 MCU 进入 HSE 的 Boot 阶段；如果 HSE 功能关闭则直接进入 Boot to Safety 阶段。

POR (Power On Reset) is a type of Destructive Reset in the microcontroller's reset categories. During this stage, the MCU formally starts up. If HSE (Hardware Security Engine) functionality is enabled, the MCU enters the HSE Boot phase; if HSE functionality is disabled, it directly enters the Boot to Safety phase.

1.1.2 HSE BOOT

MCU 在上电时会通过 HSE（若使能）进行一系列信息安全相关的加解密、证书或密钥的解析校验工作。

During power-up, the MCU performs a series of information security-related operations through HSE (if enabled), including encryption/decryption, certificate verification, and key parsing validation.

1.1.3 BOOT TO SAFETY

在该阶段下首先 MCU 会执行 BIST 自检，并通过 mSel 模块来获取 Bist 自检结果。

同时 mSel 模块也会被用来调用 sCheck 的上电自检功能，通常包括 RAM、FLASH 及总线相关的 ECC 故障检测、时钟自检、FCCU 通道自检、CRC 自检等功能。

当自检完毕后 mSel 根据整个自检结果来决定是否进行 MCU 状态模式切换，如图中所示有 Normal 模式和 Degraded 模式。

In this phase, the MCU first performs BIST (Built-In Self-Test) and obtains the BIST results through the mSel (Mode Selector) module.

Meanwhile, the mSel module is also used to invoke the power-on self-test functions of sCheck (Square Check), which typically includes ECC fault detection for RAM, FLASH, and bus-related components, clock self-test, FCCU channel self-test, and CRC self-test functionalities.

After the self-test completion, mSel determines whether to switch the MCU state mode based on the overall self-test results, with Normal mode and Degraded mode available as shown in the figure.

Normal Mode

该模式下会执行全部的应用任务功能，在进入前还会使用 sBoot 模块来对一些外设配置进行校验，若不正确的话会调用 sReco 模块进行功能性复位。若检测通过则会启动 OS（若有）执行 MCU 的周期性任务，而对于 SAF 包则会执行一些特定的周期自检，例如 SCST 自检、sChech 周期检测等等。

In this mode, all application task functions are executed. Before entering this mode, the sBoot (Secure Boot) module is used to verify peripheral configurations. If the verification fails, the sReco (Software Recovery) module will be called to perform a functional reset. If the verification passes, the OS (if present) will be started to execute MCU's periodic tasks. As for the SAF package, it will perform specific periodic self-tests, such as SCST (Safety Core Self-Test) and sCheck (Square Check) periodic detection.

Degraded Mode

在该模式下会执行一些功能的裁剪操作，该行为完全由用户根据实际需要进行定制化开发，并且这种降级模式可以设置多种，用来应对在不同故障情况下启用不同的安全任务。

In this mode, certain functionality trimming operations are performed, which are entirely customized by users based on their actual requirements. Multiple degraded modes can be configured to handle different safety tasks in various fault scenarios.

1.1.4 SHUTDOWN

sCheck 模块同时也设计了可以在下电阶段进行检测的功能，在自检过程中产生的故障可以通过下次上电进行恢复。自检完毕后的结果可以通过 mSel 模块存储进 NVM 中，以便在下次上电时读取结果。

The sCheck (Square Check) module is also designed with detection capabilities during power-down phase. Faults detected during self-test can be recovered in the next power-up cycle. After self-test completion, the results can be stored in NVM (Non-Volatile Memory) through the mSel (Mode Selector) module, allowing the results to be retrieved during the next power-up sequence.

2 SCST 概述(OVERVIEW OF SCST (STRUCTURAL CORE SELF-TEST))

SCST（Structural Core Self-Test）组件用于在运行时针对各种 MCU 内核子模块（如 Load/Store 单元、Forwarding 逻辑单元、浮点计算单元等）进行检测，以此判断 MCU 内核是否存在永久性内核故障，其诊断覆盖率通常可以达到 90%。其集成与编译器无关，检测代码使用汇编代码编写，无需昂贵的故障模拟工具，即可覆盖诊断所需的检测项。其可达到的安全等级为 ASIL B。

The SCST (Structural Core Self-Test) component is used to detect various MCU core sub-modules (such as Load/Store units, Forwarding logic units, Floating-point units, etc.) during runtime to determine whether permanent core faults exist in the MCU core. Its diagnostic coverage typically reaches 90%. The integration is compiler-independent, and the detection code is written in assembly language. It can cover all required diagnostic test items without expensive fault simulation tools. The achievable safety level is ASIL B.

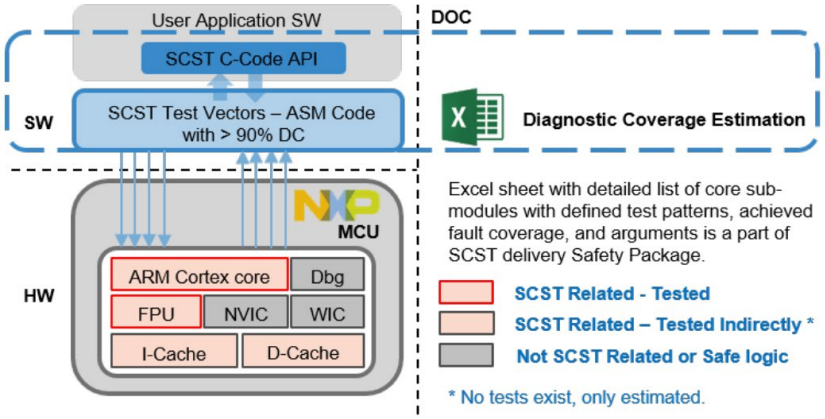


图 2.1 SCST 库材料概览

Figure 2.1 Overview of SCST library materials

3 集成问题指导(INTEGRATION ISSUES GUIDANCE)

本章节会以 SAF 集成过程中一个常见的问题为切入点，简单演示一下 SAF 的集成与问题排查过程。

This chapter demonstrates the SAF (Safety Application Framework) integration and troubleshooting process through a common issue encountered during SAF integration.

3.1.1 TCM 故障(TCM (TIGHTLY COUPLED MEMORY) FAULT)

在进行 SAF 包集成时比较常遇到的是 TCM 相关故障，例如在集成完之后正常上电，FCCU 模块报告出了 TCM 故障导致不断复位，但从代码上看没有什么异常，那这种情况要如何排查呢?

One of the most common issues encountered during SAF (Safety Application Framework) integration is TCM (Tightly Coupled Memory) related faults. For instance, after integration, during normal power-up, the FCCU (Fault Collection and Control Unit) module reports TCM faults causing continuous resets, while the code appears normal. How should we troubleshoot this situation?

首先是报错信息的查找，如下图:

First, let's look at the error information, as shown in the figure below:

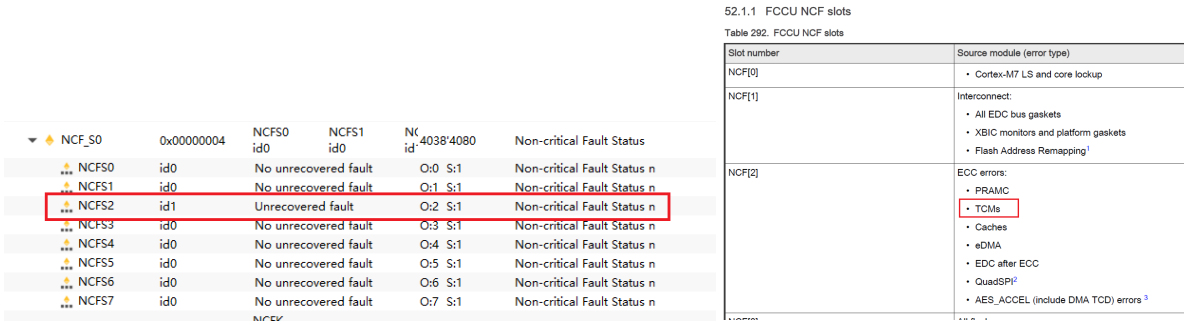


图 3.1 FCCU 故障信息

Figure 3.1 FCCU fault information

FCCU（Fault Collection and Control Unit）是 S32K3 芯片的故障收集单元，大部分芯片的故障异常信息都可以在这里读出。这里观察到通道二状态寄存器位置 1，后续的调试也能发现是在访问 TCM 地址时产生了故障。查看芯片手册可以知道，此时 TCM 部件产生了某种 ECC 故障，从而将信息上报到了 FCCU 中。

The FCCU (Fault Collection and Control Unit) is the fault collection unit of the S32K3 microcontroller, where most chip fault information can be read. Here, we observe that channel 2 status register is set to 1, and subsequent debugging reveals that a fault occurred when accessing the TCM (Tightly Coupled Memory) address. Referring to the chip manual, we can

determine that the TCM component generated an ECC (Error Correction Code) fault, which was then reported to the FCCU.

3.1.2 ECC 故障(ECC (ERROR CORRECTION CODE) FAULT)

这里先简单介绍一下 ECC 机制。对于内存有两种失效模式：

1. 非法访问，通常使用 MPU 进行内存管理进行保护

2. 内存损坏，一般通过 ECC 来进行诊断，ECC 的机制可由软件或硬件实现。

ECC 全称 Error Checking and Correcting，属于一种错误检查和纠正 算法 。 在数字电路中，最小的数据单位就是叫 “比特（bit）”，也叫数据 “位”，“比特” 也是内存中的最小单位，它是通过 “1” 和 “0” 表示数据高、低电平信号的。空间中的无线电磁干扰、电路噪声会导致内存与 CPU 在进行数据交互的时候发生 比特翻转（“0” 变为 “1”,“1” 变为 “0”), 典型的 ECC 算法一般可以做到 纠正单比特错误 和 检查 2 比特错误 。

Let's first briefly introduce the ECC mechanism. There are two failure modes for memory:

1. Illegal access, which is typically protected through memory management using MPU (Memory Protection Unit)

2. Memory corruption, which is generally diagnosed through ECC, and the ECC mechanism can be implemented in either software or hardware.

ECC (Error Checking and Correcting) is an error detection and correction algorithm. In digital circuits, the smallest data unit is called a "bit", which is also the smallest unit in memory, representing data through high and low level signals using "1" and "0". Wireless electromagnetic interference in space and circuit noise can cause bit flips ("0" changing to "1", "1" changing to "0") during data interaction between memory and CPU. Typical ECC algorithms can generally achieve Single-Bit Error Correction and Double-Bit Error Detection (SECDED).

ECC 的构成(Structure of ECC (Error Correction Code))

ECC 需要的 bit 位数量由数据段的比特数决定，8 位的数据需要 5 位的 ECC 位进行校验，数据位每增加一倍，ECC 只增加一位校验位。例如，32 位数据的 ECC 校验位数量位 7 位。

The number of required ECC bits is determined by the number of bits in the data segment. 8-bit data requires 5 ECC bits for verification, and for each doubling of data bits, only one additional ECC check bit is needed. For example, 32-bit data requires 7 ECC check bits.

ECC 编码逻辑单元会在写入时生成一个 7 位的 ECC 校验码；当数据通过 AHB 总线读取数据时，ECC 解码逻辑单元会使用相同的算法来计算读取数据的校验码，然后比较前后两个校验码是否相同。

The ECC encoding logic unit generates a 7-bit ECC check code during write operations. When data is read through the AHB (Advanced High-performance Bus) bus, the ECC decoding logic unit uses the same algorithm to calculate the check code for the read data, then compares it with the original check code to verify if they match.

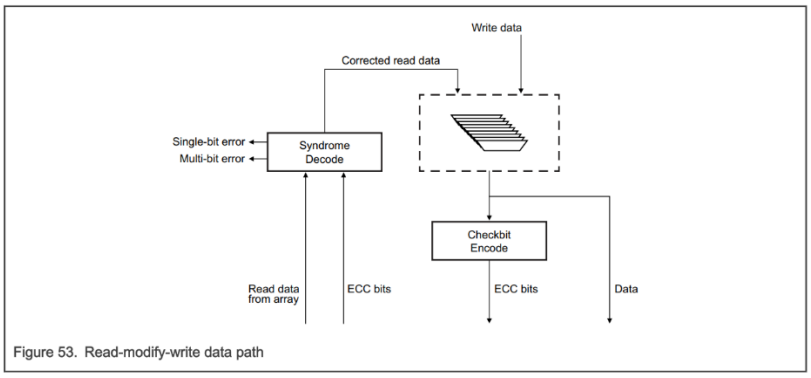


图 3.2 S32K3 芯片 ECC 机制校验路径

Figure 3.2 ECC Mechanism Verification Path of S32K3 Microcontroller

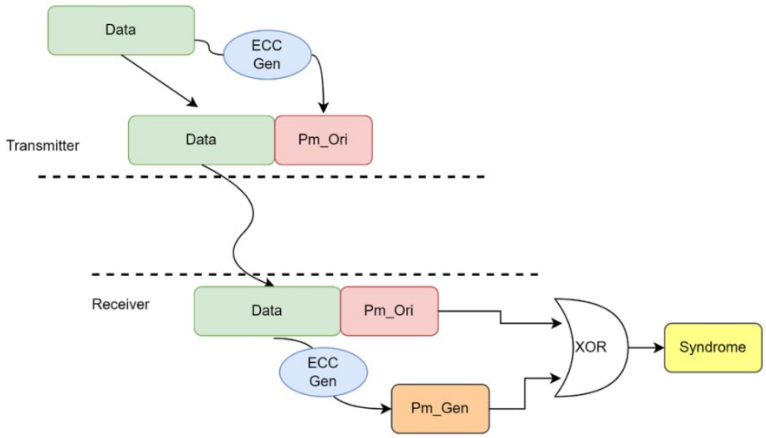


图 3.3 常规 ECC 校验机制

Figure 3.3 Conventional ECC Verification Mechanism

所以触发故障的原因是访问了 TCM 中带有 ECC 故障的地址，导致异常产生。

Therefore, the fault was triggered by accessing a TCM (Tightly Coupled Memory) address with an ECC (Error Correction Code) error, which resulted in the exception being generated.

3.1.3 TCM 内存初始化(TCM (TIGHTLY COUPLED MEMORY) INITIALIZATION)

类似 SRAM，TCM 内存在使用前也要进行初始化，否则会存在 ECC 故障。

Similar to SRAM, TCM (Tightly Coupled Memory) must also be initialized before use, otherwise ECC (Error Correction Code) faults may occur.

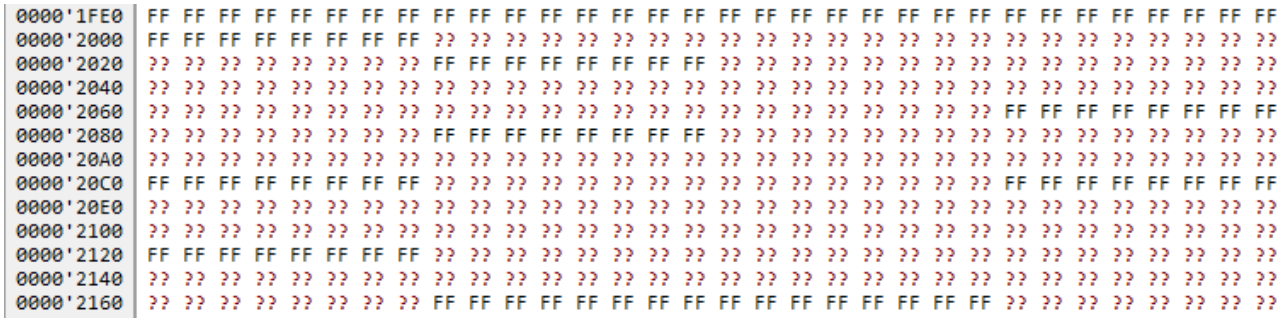


图 3.4 ECC 故障区域调试视图

Figure 3.4 Debug View of ECC Fault Region

下面是演示工程中的启动代码部分截图，可以观察启动代码会将__INT_DTCM_START 到__INT_DTCM_END 的部分进行写 0 操作，由 3.1.2 章节可以知道在执行写操作的过程中会刷新对应的 ECC 校验码，所以这个过程就是 TCM 的初始化步骤。

Below is a screenshot of the startup code from the demonstration project. We can observe that the startup code performs a zero-writing operation from __INT_DTCM_START to __INT_DTCM_END. As explained in section 3.1.2, the corresponding ECC check codes are refreshed during the write operation, which constitutes the initialization process of TCM.

```
RamInit:

    ldr  r0, =__RAM_INIT
    cmp  r0, 0

    beq SRAM_LOOP_END
    ldr r1, =__INT_SRAM_START
    ld
    subs    r2, r1
    subs    r2, #1
    ble SRAM_LOOP_END

    movs    r0, 0
    movs    r3, 0
SRAM_LOOP:
    stm r1!, {r0,r3}
    subs r2, 8
    bge SRAM_LOOP
SRAM_LOOP_END:

DTCM_Init:

    ldr  r0, =__DTCM_INIT
    cmp  r0, 0

    beq DTCM_LOOP_END

    LDR r1, =CM7_DTCMCR
    LDR r0, [r1]
    LDR r2, =0x1
    ORR r0, r2
    STR r0, [r1]

    ldr r1, =__INT_DTCM_START
    ldr r2, =__INT_DTCM_END

    subs    r2, r1
    subs    r2, #1
    ble DTCM_LOOP_END

    movs    r0, 0
    movs    r3, 0
DTCM_LOOP:
    stm r1!, {r0,r3}
    subs r2, #8
    bge DTCM_LOOP
DTCM_LOOP_END:
```

图 3.5 启动代码截图

Figure 3.5 Screenshot of Startup Code

最后我们只要通过调整链接脚本，将初始化范围覆盖到整个 TCM 区域，即可消除这个故障。

Finally, we can eliminate this fault by adjusting the linker script to extend the initialization range to cover the entire TCM (Tightly Coupled Memory) region.

3.1.4 总结(CONCLUSION)

在 SAF 功能安全包集成过程中时常会遇到类似的 ECC 问题，可能是 SRAM、TCM、FLASH 或者是其他部件，排查及解决思路大同小异。参考上述的分析步骤，逐步排查后大多数问题都能得到有效的解决。

During the integration of SAF (Safety Application Framework) safety package, similar ECC (Error Correction Code) issues are frequently encountered, which may involve SRAM, TCM, FLASH, or other components. The troubleshooting and resolution approaches are generally similar. By following the analysis steps mentioned above, most issues can be effectively resolved.

4 知从集成服务(ZC INTEGRATION SERVICES)

知从针对恩智浦 S32K3 系列 MCU 的 S32K3 safety software 可以提供相应的集成测试服务。针对 S32K3 Safety software 需要根据客户安全需求进行动态配置才能完全覆盖客户的应用需求的特点，我们可以根据不同的客户项目要求进行配置，最终满足客户功能安全部分的工程需求。

We provide integration testing services for NXP S32K3 series MCU's S32K3 safety software. Given that S32K3 Safety software requires dynamic configuration based on customer safety requirements to fully cover their application needs, we can perform configurations according to different project requirements, ultimately meeting the functional safety engineering needs of our customers.

目前，我们服务支持过如下芯片：

Currently, our services support the following microcontrollers:

编号	产品名称	芯片企业	芯片型号	编译器	调试器
No.1	S32K3 safety software	恩智浦	S32K322	Green Hills Compiler 2021.1	lauterbach(Trace32 R.2022.02) iSystem(IC5700,5000)
No.2	S32K3 safety software	恩智浦	S32K324	S32DS Version: 3.4	lauterbach(Trace32 R.2022.02) iSystem(IC5700,5000)
No.3	S32K3 safety software	恩智浦	S32K322	Green Hills Compiler 2021.1	lauterbach(Trace32 R.2022.02) iSystem(IC5700,5000)
No.4	S32K3 safety software	恩智浦	S32K342	Green Hills Compiler 2021.1	lauterbach(Trace32 R.2022.02) iSystem(IC5700,5000)
No.5	S32K3 safety software	恩智浦	S32K344	Green Hills Compiler 2021.1	lauterbach(Trace32 R.2022.02) iSystem(IC5700,5000)

提供 S32K3XX 功能

安全包集成测试服务

5 技术服务(TECHNICAL SERVICES)

5.1 功能介绍(Functional Description)

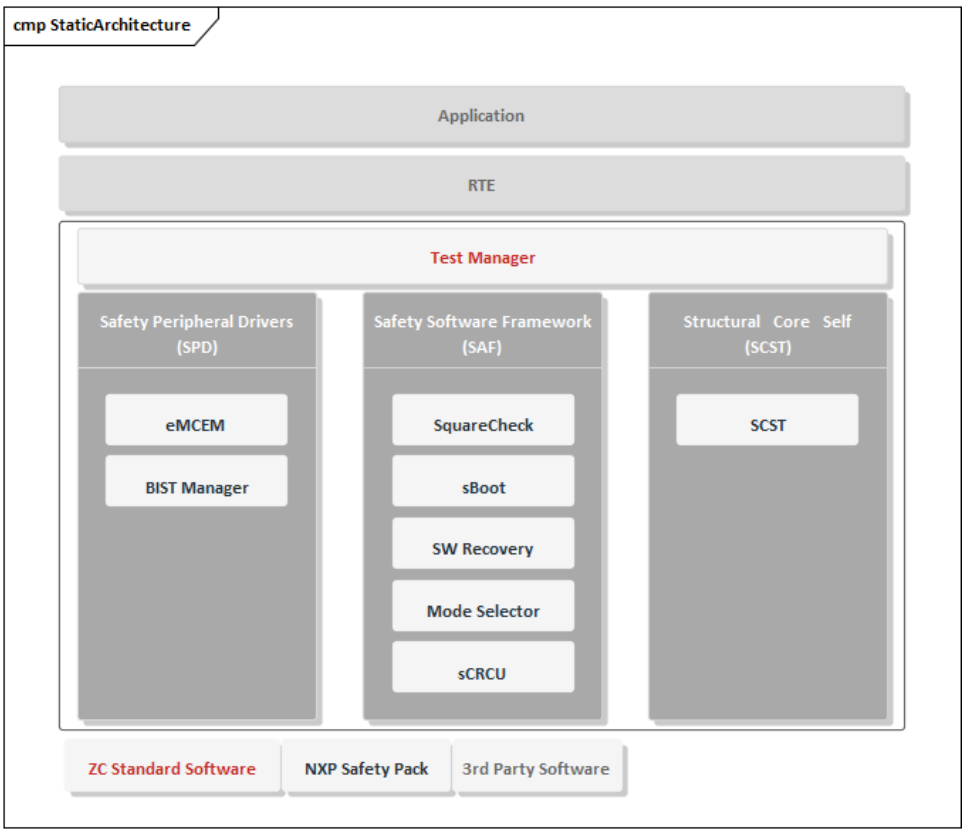


图 5.1 Safety Pack 软件架构图

Figure 5.1 Safety Pack Software Architecture Diagram

知从在进行 SAF&SCST 安全包集成的时候在上层还会增加一个 Test Manager 模块用于按启动流程调用各个自检模块的接口功能，在用户层面集成时只需要通过 Test Manager 接口即可执行整个 SAF&SCST 安全包功能，大大提高了集成效率。

During the integration of SAF & SCST safety package, a Test Manager module is added at the upper layer to call interface functions of each self-test module according to the startup sequence. At the user level integration, the entire SAF & SCST safety package functionality can be executed simply through the Test Manager interface, significantly improving integration efficiency.

5.2 知从 Test Manager 模块介绍（Introduction to ZC Test Manager Module）

Test Manager 模块是由知从在集成 Safety Pack 到实际工程中时添加的核心管理调度模块，用于管理 Safety Pack 的测试流程以及管理与应用层的接口。同时还配对了相应的知从木牛配置工具，可以快速配置需要的检测模块。

The Test Manager module is a core management and scheduling module added by ZC when integrating Safety Pack into actual projects. It is used to manage the Safety Pack test

process and handle interfaces with the application layer. Additionally, it is paired with ZC's MuNiu configuration tool, which facilitates quick configuration of required detection modules.

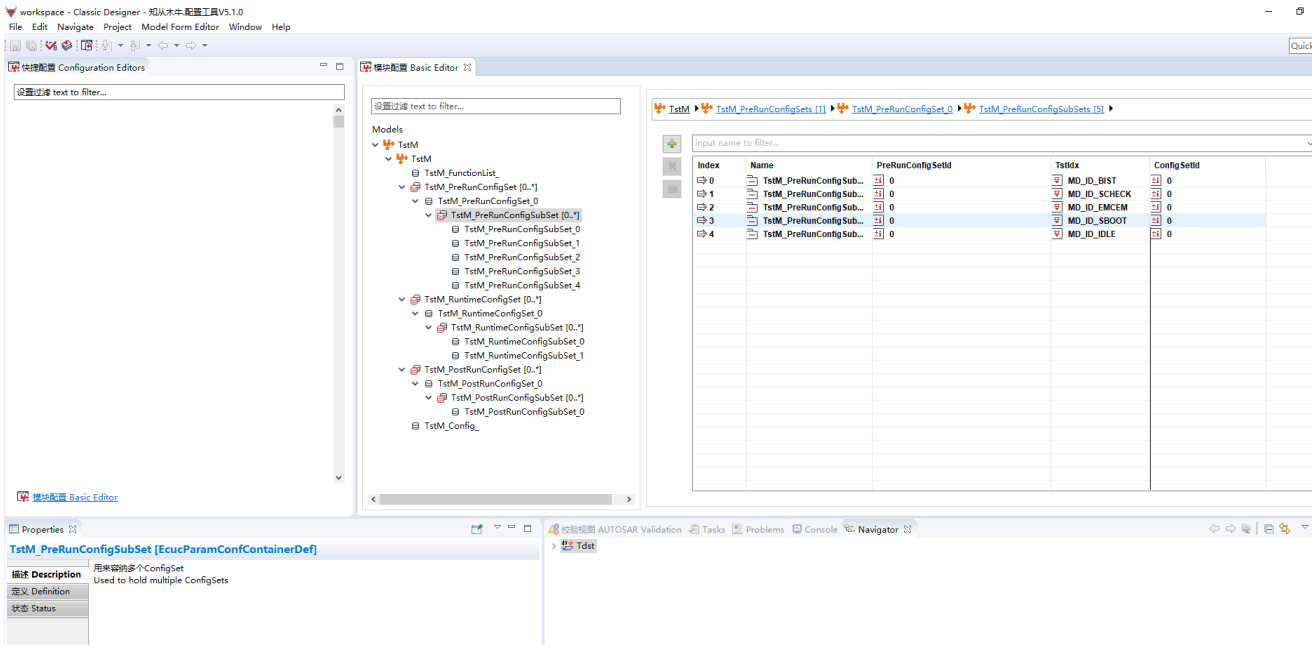


图 5.2 木牛配置工具界面

Figure 5.2 MuNiu Configuration Tool Interface