AURIX TC3xx HSM

硬件安全模块

Training Material

面向新人的完整入门手册

版本 v1.0　2026 · 06

手册信息

手册名称	AURIX TC3xx HSM 培训材料
副标题	面向新人的完整入门手册
文档版本	v1.0
发布日期	2026-06-04
分发级别	
内容来源	Infineon 官方 AN + demo 工程源码 + 实战整理
核心参考	HSM_TS_V2.0_Rev2.0.pdf (AURIX TC3xx HSM Target Spec, 292 页)
适用读者	首次接触 AURIX HSM 的工程师 / Bootloader / 启动软件 / 安全系统设计者

如何使用本手册

本手册共 14 章 + 附录,约 1353 行。**按学习深度分 4 个层次,新人可按需精读:**

30 分钟速读	**只读这 5 章,新人必看** §1 是什么 → §3 整体架构 → §4.1-4.3 硬件 → §6.2 桥接握手 → §10.1-10.2 Debug 保护
半天深入	**加 4 章,搞清启动 + 桥接 + 命令** §5.1-5.3 启动流程 → §6.1 寄存器清单 → §8 命令集 → §9 Secure Boot 实战

2 天吃透	**全读,工具链 + OTA + 陷阱全覆盖** §7 工具链、§11 OTA、§12 陷阱、§14 附录
 **实战参考**	**用到再查,做项目时翻** §6.5-6.7 中断/传感器、§5.2.3 状态码、§13 Demo 速查

目 录

提示:本目录由 Word 域自动生成。如页码未更新,请选中后按 F9。

手册简介

AURIX TC3xx HSM 培训材料

面向新人的完整入门手册

内容基于 D:\workspace\HSM 目录下的 Infineon 官方文档、应用笔记 (AN) 与 demo 工程源码,结合实战经验整理而成。

最权威参考:Infineon `HSM_TS_V2.0_Rev2.0.pdf` —— AURIX TC3xx Hardware Security Module Target Specification Revision 2.0 (2016-03-04,292 页),本手册中所有带 TS §x.y 标注的细节都来自该文档。

官方速训:infineon-aurix-tc3xx-hardware-security-module-quick-training-en.pdf(V1.0 / 2020-09,11 页),重点是性能数据与典型应用示例(已并入本手册第 4、8 章)。

适合第一次接触 AURIX HSM 的工程师、Bootloader / 启动软件开发人员、安全系统设计者。

CH 01

第 1 章

HSM 是什么 —— 一句话回答

HSM = Hardware Security Module,芯片内部一个独立的小 CPU + 专属 RAM/Flash/加密硬件,专门负责"所有跟密钥、签名、安全启动、安全通信有关的活"。

它长在 AURIX TC3xx 内部,跟主 CPU(TriCore)在物理上隔离,代码、数据、内存各自独立,互相不可直接访问。Host(TriCore)想用 HSM 算个 AES、做一次签名验证、或者让 HSM 帮忙"确认一下我即将跑的固件没被改过",都得通过一组特殊的桥接寄存器去敲门。

一句话总结它的价值:把车规 ECU 的"信任根 (Root of Trust)"固化到硅片上,即便攻击者拿到整块板子,他也无法读取 HSM 内部的密钥或篡改 HSM 行为。

CH 02

第 2 章

为什么车规 MCU 一定要有 HSM

车上的 ECU 越来越像一台电脑,网关、ADAS、OTA 升级统统都需要它。攻击面也越来越大:

- 通过 OBD 接口、CAN 刷入恶意固件
- 通过 OTA 包下发假升级
- 通过 JTAG 调试口读出固件,反向工程密钥
- 物理拆片做侧信道攻击

如果没有可信硬件支撑,任何软件层面的"加密"都只是把钥匙放在客厅桌子上 —— 攻击者只要拿到设备主控权,就可以直接 dump。

EVITA-FULL、ISO 21434、UN R155 这些车规安全标准都明确要求:

- 密钥不出硬件(私钥烧入 HSM 内部专有 NVM/DFlash 区域,不进主 PFlash)
- 关键代码签名验证后才能跑(Secure Boot)
- 调试接口受控(生产后可以关掉调试,或必须经 HSM 授权才能开)
- 异常自检:电压/温度/时钟等传感器异常时,HSM 能主动把系统锁定/复位

AURIX TC3xx HSM 就是用来满足这些要求的硬件模块,Infineon 称之为"Security Island"。

2.1 典型应用场景(Quick Training Page 9 —— "Chip Tuning Protection")

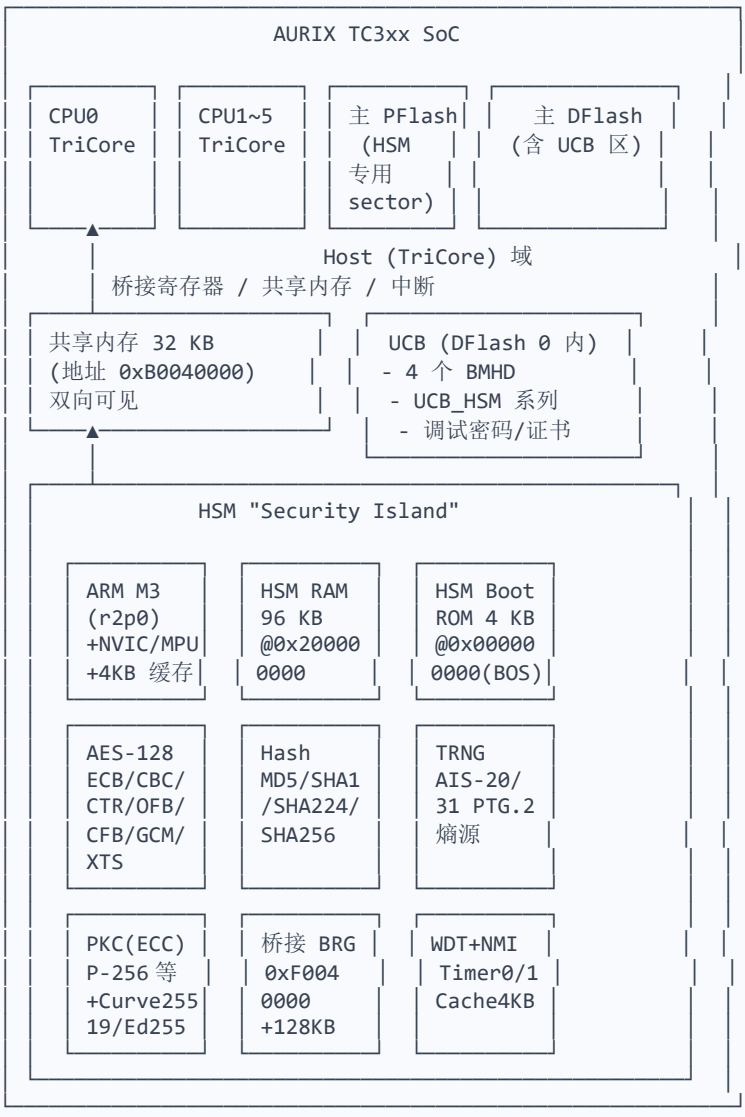
- Challenge-Response Authentication:更换 ECU 或更换微控制器时,需要 OEM 出具的私钥对随机 challenge 签名,设备端 HSM 验签通过后才解锁。
- Closed Debugger Interface:出厂后调试口默认锁死,任何 JTAG/DAP 接入都需要 HSM 授权。
- IP Protection:OEM 固件代码 / 标定数据 / 调校参数放在受 HSM 保护的存储区,即便物理取出芯片 flash 也读不出来。
- Tuning Protection:防止第三方刷写"性能包" / "解锁包" —— 调校数据必须经 HSM 验签后由 HSM 写入标定区,任何直接写 flash 的尝试都被阻止。

CH 03

第 3 章

AURIX TC3xx Security Island 整体架构

下面这张图,新人务必背下来:



关键点:

- HSM 是物理上隔离的子系统,有自己的内核、RAM、ROM、总线。

- Host 永远不能直接读/写 HSM 内部 RAM/ROM,只能通过桥接寄存器和 32 KB 共享内存交互。
- 共享内存(0xB0040000)从 Host 视角是普通内存,从 HSM 视角也是普通内存,但只有这一段是双向可视的。
- 调试接口分两层:HSM 自己可以锁(且只有 HSM 能解),TriCore 只能锁自己那一层。

CH 04

第 4 章

HSM 硬件结构(Cortex-M3 + 硬件加速器)

4.1 内核:ARM Cortex-M3 (r2p0)

项目	数值	来源
ISA	ARMv7-M(Thumb/Thumb-2 only,没有 ARM 模式)	TS §1.1
具体型号	ARM Cortex-M3 Revision r2p0	TS §2.2
主频	默认与 SPB 同频(典型 100 MHz);HSM 可通过 CLKCTRL.CLKDIV 降为 SPB/2、SPB/4 省电	TS §9.2.3
NVIC	10 个中断节点,8 级优先级(3 bit)	TS §2.3.1
SysTick	24-bit,参考时钟 = SPB/16(SPB=100 MHz 时即 6.25 MHz);10 ms 周期 = 重载值 0x0000F423 (62500-1)	TS §2.2.2
MPU	8 个 address region,带 sub-region,粒度 32 B – 4 GB,ARMv7-M PMSAv7 兼容;Region 7 优先级最高	TS §1.1 / §2.4
字节序	Little endian	TS §2.2.4
特权/用户模式	支持	TS §1.1
Sleepdeep	不支持 — 写 SCR.SLEEPDEEP=1 仅停 SysTick,效果等同普通 Sleep	TS §2.2.3
WIC(Wakeup Interrupt Controller)	未实现	TS §2.2.3
SEV 指令	不能 用来跨核同步 Host(SEV 跨 HSM-Host 不生效),必须走桥接寄存器	TS §2.2.1
SYSRESETREQ / VECTRESET	禁止设置 — 会导致未定义行为	TS §2.2.5

项目	数值	来源
(AIRCR)		
调试	DAP,默认启用,可通过 UCB(HSMDBGDIS / DBGIFLCK)禁用或永久锁定	TS §1.1

重要: HSM 内核是纯 M3,指令集和 TriCore 完全不一样。HSM 工程必须用 ARM 工具链(GNU Arm Embedded、HighTec ARM、Tasking ARM、IAR ARM 任选)独立编译,产出 .elf / .hex 后烧录到 HSM PFlash 区域。

核内中断分配(TS Table 2-2,异常号 0–25):

异常号	类型	说明
2	NMI	Watchdog Timer 中断(优先级 -2,固定最高)
16	Timer0	16-bit 定时器 0 溢出
17	Timer1	16-bit 定时器 1 溢出
18	TRNG	真随机数生成器
19	Bridge Service	桥接服务(Host → HSM 命令)
20	Bridge Error	桥接错误(ECC/Bus error 等)
21	Sensor	传感器异常(电压/温度)
22	External	外部中断(32 选 1,见 6.6 节)
24	PKC Ready	公钥引擎空闲
25	PKC Error	公钥引擎错误

AES 速度够快,不产生中断;WDT 用 NMI(优先级高于所有可编程 IRQ)。其他 7–15、23 号槽位保留或 disabled。

4.2 完整存储映射(HSM 视角,TS Table 2-1)

段	地址范围	大小	属性	用途
0	0x00000000 –	4 KB	B,C,MPU	HSM Boot ROM(BOS)

段	地址范围	大小	属性	用途
	0x00000FFF			
0	0x00001000 – 0x0FFFFFFF	-	B,C,MPU	Host Reserved
1	0x10000000 – 0x1FFFFFFF	256 MB	B,C,MPU	Host SRAMs(CPU5)
2	0x20000000 – 0x20017FFF	96 KB	B,C,MPU	HSM Local RAM
2	0x20018000 – 0x200FFFFF	-	B,C,MPU	Host Reserved
2	0x20100000 – 0x21FFFFFF	-	B,C,MPU	Host Reserved
2	0x22000000 – 0x23FFFFFF	32 MB	B,C,MPU	HSM Bit-banding Alias(映射到 0x20000000–0x200FFFFF,逐位访问)
3	0x30000000 – 0x3FFFFFFF	256 MB	B,C,MPU	Host SRAMs(CPU4)
4	0x40000000 – 0x41FFFFFF	32 MB	B,C,MPU	Host SRAMs(CPU3)
4	0x42000000 – 0x43FFFFFF	32 MB	B,C,MPU	Host Bit-banding Alias
5–7	0x50000000 – 0x7FFFFFFF	-	B,C,MPU	Host SRAMs(CPU2/1/0)
8	0x80000000 – 0x80FFFFFF	16 MB	B,C,MPU	Host PFlash 0–5 — HSM 用户代码 + 数据
8	0x81000000 – 0x8FFFFFFF	-	B,C,MPU	Host Reserved(burst)
9	0x90000000 – 0x9FFFFFFF	-	B,C,MPU	Host Reserved(burst)
A	0xA0000000 – 0xA0FFFFFF	16 MB	XN,C,MPU	Host PFlash 0–5(非缓存视角别名)
A	0xAF400000 – 0xAF400DFF	-	XN,C,MPU	Host UCB00 – UCB06

段	地址范围	大小	属性	用途
A	0xAF400E00 – 0xAF400FFF	512 B	XN,C,MPU	UCB07 + HSM Config Area(BOS 用的加密配置)
A	0xAF401000 – 0xAF405FFF	-	XN,C,MPU	Host UCB08 – UCB47
A	0xAFC00000 – 0xAFC1FFFF	128 KB	XN,C,MPU	HSM Data Flash 1(HSM 专属 EEPROM)
E	0xE0000000 – 0xE0000FFF	4 KB	XN,PRIV	HSM Reserved for ITM(实际未实现)
E	0xE0001000 – 0xE0001FFF	4 KB	XN,PRIV	HSM DWT(Data Watchpoint & Trace)
E	0xE0002000 – 0xE0002FFF	4 KB	XN,PRIV	HSM FPB(Flash Patch & Breakpoint,6 个 BP)
E	0xE000E000 – 0xE000EFFF	4 KB	XN,PRIV	HSM System Control Space(NVIC / SCB / SysTick / MPU)
E	0xE008A000 – 0xE008BFFF	8 KB	XN,PRIV	HSM Cache Control SFRs
E	0xE00FF000 – 0xE00FFFFF	4 KB	XN,PRIV	HSM ROM table(供调试器自动识别)
F	0xF0040000 – 0xF005FFFF	128 KB	-	HSM Bridge 区(含 DBGMEM 调试窗口 0xF0050000–0xF005FFFF)

新手必背:

- Host 访问 HSM 桥接寄存器用 0xF004xxxx。

- HSM 访问 Host 内存用 0xF0050000(64 KB 调试窗口,TS 称之为 DBGMEM;在 HSM 端配合 SAHBASE 寄存器重定位;与 HSM 软件代码中通常叫 SAHMEM 的是同一个窗口)。

- HSM 用户代码放 0x80000000(PFlash 0,16 MB)。

- HSM Config Area 在 0xAF400E00(512 B,被 BOS 用 mask 个性化密钥 AES-CBC 加密 + 16 字节 AES-CBC-MAC 校验)。

4.3 硬件加速器

模块	能力	关键点(均来自 TS)
AES-128	AES-128 only;支持 ECB / CBC / CTR / OFB / CFB / GCM / XTS	8 个 128-bit key slot,前 2 个是 use-only(不可改、只能用于加密/认证),5 个 context slot(其中 1 个保留给 PRNG)。AES 没有中断
Hash	MD-5、SHA-1、SHA-256、SHA-224(SHA-224 由 SHA-256 派生 + 少量软件参与)	SHA-256/SHA-224 支持 HMAC
PKC	ECC only — 256-bit 以内,支持 F(p) 与 F(2^m) 两类域;ECDSA、Curve25519、Ed25519;原始算子(ADDN, SUBN, MULTN, RED, DIVN, INV, MULT, INV2, RED2, EXP)	内部有 SCM(Shared Crypto Memory,1 KB = 256×32 bit) 和 TCM(Tightly Coupled Memory,1 KB,PKC 独占);TS 没说支持 RSA — 别再想 RSA
TRNG	真随机数源,数字噪声 + 数字后处理(DPP)	符合 BSI AIS-20/31 PTG.2(高强度);典型吞吐 260 kbit/s @ 33 MHz,360 kbit/s @ 100 MHz;Trng_CheckBuffer() 在 HSM 主循环持续维护熵池
Cache	4 KB,4-way set-associative,16-byte block,64 sets,LRU 替换	读分配 + 写回(write-back,无 write-through);命中 1 cycle;1 个写缓冲;lock/unlock、clean/invalidate 操作齐全;通过 0xE008A000 控制
Timer	2 × 16-bit 定时器(独立)	共享 HSM 系统时钟;溢出产生 IRQ 16/17
WDT	带 checkpoint 机制的看门狗	超时或 checkpoint mismatch 触发 NMI(不是普通 IRQ);有 WDT_CTRL/WDT_VAL/WDT_RELOAD/WDT_SRV/WDT_SRV_INI 5 个 SFR;HALT mode 下 WDT 暂停

4.3.1 HSM 加速器性能数据(Quick Training V1.0,Infineon 官方)

Hash 性能(每 512-bit 数据块所需时钟周期):

算法	Clock cycles / 512-bit	备注
MD-5	65	—
SHA-1	81	—
SHA-224	65	需要少量软件参与(由 SHA-256 派生)
SHA-256	65	—

65 cycles @ 100 MHz ≈ 理论 98 MB/s SHA-256 吞吐(Q-Training Page 5 标注)。

PKC 性能与能力:

项	数值
整数 / 二进制多项式位宽	up to 256 bit
存储容量	32 个值(整数或多项式)× 256 bit
ECDSA 签名速率	200 sign/s @ 100 MHz,曲线位宽 256
ECDSA 验签速率	100 verify/s @ 100 MHz,曲线位宽 256
基础算子	乘法;模加/减/乘/除/逆;仿射坐标点加 / 倍 / 标量乘
NIST 曲线支持	P-192 / P-224 / P-256 / K-163 / B-163 / K-233 / B-233
Brainpool 曲线	brainpoolP160r1 / P192r1 / P224r1 / P256r1
现代曲线	Curve25519 / Ed25519
域	F(p) 与 GF(2^m)

TRNG 性能(Quick Training Page 4):

- 符合 BSI AIS-20/31 标准(Quick Training 写的是 AIS-31,TS 写的是 AIS-20/31 PTG.2,本质一致)
- 典型吞吐 Rtyp ≈ 360 kb/s @ 100 MHz(TS 给的 33 MHz 数值是 260 kb/s)
- 适用:密钥生成、Challenge、blinding values、padding bytes

AES 模式全清单(Quick Training Page 3):ECB / CBC / CTR (32-bit counter) / OFB / CFB / GCM / XTS。

4.3.2 HSM DFlash 与 "Exclusive Access" 机制(Quick Training Page 7)

- HSM Data Flash 1(DF1) = 128 KB @ 0xAFC00000,专门给 HSM 存安全密钥、加密数据、单调计数器等。
- Round Robin 刷新:DF1 内容由 FEE 驱动按 Round Robin 算法循环刷新(类似 AUTOSAR NvM 的 wear-leveling),保证长寿命 + 抗单点失效。
- Exclusive Access 特性:HSMDX 位置位时,DF1 与 HSM PFlash 区域只允许 HSM 核读写,TriCore 与其他总线主全部访问被拒。这意味着 HSM 在更新密钥时,TriCore 仍能继续从 PFlash 取指执行用户代码——两侧完全独立,不影响实时性。

4.4 HSM 内存与 RAMKEEP 行为

- HSM 内部 RAM(HSMRAM,96 KB 左右):HSM 自己的运行空间。
- 关键开关 `HSMRAMKEEP`:在 UCB_HSMCFG 里,决定每次上电/系统复位后是否清空 HSMRAM。
- 0 = 清空(每次重启密钥消失,默认,最安全)
- 1 = 保留(可实现"会话密钥持续存在",但增加攻击窗口)

CH 05

第 5 章

启动流程(SSW × BOS × UCB × BMHD)

这是 HSM 最容易让人懵的部分。HSM 的启动和 TriCore 的启动是分开的,但最终 TriCore 能不能跳出 SSW 还取决于 HSM。

5.1 三个 ROM,三条线

启动模块	谁执行	位置	角色
Boot Firmware	CPU0(TriCore)	芯片 BOOT ROM 物理区(0xF0000000 起)	复位向量、做基础硬件初始化(Flash ramp-up、RAM init、Lockstep 配置),然后转交控制权到 SSW
SSW(Start-up Software)	CPU0(TriCore)	BootROM 内部	主启动软件,负责初始化 TriCore、读 UCB/BMHD、决定启动模式
BOS(Boot System)	HSM(Cortex-M3)	HSM BootROM(0x00000000–0x00000FFF,4 KB)	Infineon 出厂固化的 HSM 启动器;读 HSM Config Area、解密、装 AesKey0/AesKey1、4 步找启动 sector、跳到 User-OS

关键: BOS 是 Infineon 提供的不可改的底层固件(没有 patch 机制)。所有 HSM 用户代码都从 BOS 手里接过接力棒。

5.2 完整启动时序

⚠ 简化版时序图 —— 真实 BOS 内部远比这复杂(4 步 BOOTSEL fallback + AES-MAC 校验 + HAR 处理 + Config Area 解密)。详细流程见 §5.2.1,精确协议见 §5.2.3 HT2HSMS/HSM2HTS 状态码。

```
POR / 系统复位
    │
    ▼
Boot Firmware (基础硬件初始化)
    │ - 触发 Flash ramp-up
    │ - RAM 初始化(可配)
    │ - Lockstep 配置
    │ - 评估 UCB 状态后,转交控制权到 SSW
    │
    ▼
SSW (Start-up Software,跑在 CPU0)
    │ - 读 UCB_BMHD0~3,选一个有效的 BMHD
    │ - 读 UCB_HSMCOTP0
    │    ├─ HSMBOOTEN=0 → 不释放 HSM 复位,HSM 不参与启动
    │    └─ HSMBOOTEN=1 → 释放 HSM 复位,BOS 开始跑(与 SSW 并行)
    │ - 检查 PROCONHSMCFG.SSWWAIT
    │    ├─ SSWWAIT=0 → SSW 不等 HSM,直接跳到 BMHD.STAD(不安全)
    │    └─ SSWWAIT=1 → SSW **自旋等待** HSM 释放 Host
    │
    ├──────────── HSM 侧 ────────────┐
    │ BOS (HSM BootROM)              │
    │  - 校验 HSM Config Area        │
    │    (AES-CBC-MAC,见 §5.2.1)     │
    │  - 4 步 BOOTSEL fallback       │
    │    找 User-OS sector           │
    │    (见 §5.2.1 第 5 步)         │
    │  - 跳到 HSM PFlash 用户固件    │
    │  - HSM User-OS 跑 Secure       │
    │    Boot,完成后写 HSM2HTF=1     │
    │    释放 CPU0                   │
    └────────────────────────────────┘
    │
    ▼
SSW 收到 HSM 释放信号 → 跳到 BMHD.STAD
    │
    ▼
用户启动软件 (Application) + HSM 主循环
```

5.2.1 BOS 内部流程(HSM 侧细节,TS §11)

BOS 跑在 HSM CPU 上,这是 HSM 上电后第一段执行的代码,由 Infineon 固化、无 patch:

- ▸ 硬件自动阶段 — 读 BOS 自身的 vector table(BOS-ROM 偏移 0),初始化 SP 和 PC,跳到 BOS reset vector。
- ▸ Boot Initialization(图 11-3):
- ▸ 按 Table 11-1 配置 NVIC:除 Bridge Error(异常 20)和 NMI 外,其他异常全 disabled;WDT NMI 占异常 2(优先级 -2,最高)。
- ▸ 用 AES-CBC 计算 HSM Config Area (0xAF400E00 起 512 B,实际从 offset 0x10 起) 的 16 字节 MAC,与 Config Area 末尾存的 MAC 比对。

- 失败 → TRNG 用复位值起、HSM2HTS.STATUS_CODE = 0xFFFF、配 debug lock → 进 Usermode 错误处理。
- 成功 → 用 AES-ECB + BOS-ROM 中固化的 mask-individual key 解密 Config Area;启动 TRNG;把 AesKey0 和 AesKey1(use-only)装入 AES 硬件并 lock AES;锁 Config Area 读(永远禁用,只有 BOS 自己用);销毁 AES 关键上下文(防泄漏)。
- 决定走 Testmode 还是 Usermode — 读 HT2HSMS.REQ_MODE(bit 27:24 = 0x5 = Testmode,其他 = Usermode)。HT2HSMS 整体内容若无效(典型由 SSW 在释放 HSM 前没写)则默认 Usermode。
- Testmode 路径(只在产线 wafer test 时用,用户代码路径不常见):
- Testmode signature 错 → STATUS_CODE = 0xFF01。
- DLT 执行权限错 → 0xFF02。
- Testmode 被锁(写锁) → 0xFF03,sleep 退出。
- Usermode 路径(产品常用):
- 检查 AesKey0/1 是否已 lock;未 lock → STATUS_CODE = 0xF003,sleep 退出。
- 4 步找 User-OS 启动 sector:DMU_SP_PROCONHSMCBS.BOOTSEL0/1/2/3 每个 6 bit,值 0–27 编码为 HSM0X–HSM27X(物理 0–39 个 sector,实际只用低 28 个),地址 0x80000000 + 0x4000 * sector。BOS 按 Step 0→1→2→3 顺序试,每个候选 sector 检查:
- 读 vector table 头 8 字节(stack pointer + reset vector);
- 栈指针必须在 Internal RAM(0x20000000–0x20017FFF);
- reset vector 必须指向合法 HSM PFlash sector 范围;
- 顺便设 CHECK_OS_VEC_TABLE 标记,这一阶段允许 ECC 单错;
- 任何一个不过就跳下一步;4 步全失败 → STATUS_CODE = 0xF002(User-OS 复位向量/栈指针无效),sleep 退出。
- 找到合法 sector → 记下 SP、User-Code 起始、VTOR 基址,清 CHECK_OS_VEC_TABLE,更新 `HSM2HTS`: ACT_MODE = 01(Usermode Entry prepared)。
- 自旋等 Host:BOS 等 HT2HSMS.OS_ENTRY_ALLOWED(bit 4 = 1)才继续。
- HAR(Halt After Reset)检查:HT2HSMS.HAR_ENABLE(bit 31:28 = 0x5)被置位 + HSM 调试模式已开 → BOS 配置第一次 User-OS 指令前硬件 BREAK(便于调试器接管)。调试模式未开时 HAR 请求被忽略(BOS 不报错,只是不触发 BREAK)。

- 清理:BOS 把 startup 用过的 RAM 区(头 384 B)清零;清 cache;关 bridge interrupt;设 VTOR 指向 User-OS vector table;通过 ERRCTRL.ROMON=0 关 HSM Boot ROM;HSM2HTS 清零;跳到 User-OS reset vector。
- 异常处理 — 整个过程任何 Generic Exception 都会:
- 锁 Config Area read;
- 写 HSM2HTS.STATUS_CODE = 0xEE00 | isrNumber,ACT_MODE = 11;
- 进 Sleep,不再恢复。

Bus Fault → 0xC000;Bridge Error → 0xC000 | bridgeStatus(bridgeStatus 来自 ERRCTRL 的 13 bit)。

5.2.2 HSM Config Area 与 HSM 内存布局(TS §10 / §11.2.3.2)

- HSM Config Area:0xAF400E00–0xAF400FFF,512 B,前 16 字节是头(状态/版本),从 offset 0x10 起是加密的配置体,末尾 16 字节是 AES-CBC-MAC。BOS 启动时解密 + 校验 MAC;读锁永远开(用户 HSM 代码不能直接读),但 HSM User-OS 启动后可通过 cryptoInitBootParameters() 间接取到解密后的副本(hsmCfgData 结构)。
- BOS RAM Layout(HSM Local RAM 96 KB):
- 0x20000000–0x00018000+(头 384 字节):BOS 保留 — 128 B startup stack + 32 B stack 保护 + 156 B BOS variables + 68 B 每次启动被清零的暂存区。进 Usermode 前清零。
- 0x20000180 起:User-OS 自由使用,共约 95.6 KB。

5.2.3 HT2HSMS / HSM2HTS 协议(TS Table 11-3 / 11-4,Host ↔ BOS 通信的 32-bit 寄存器)

HT2HSMS(Host → HSM/BOS,Host 写、HSM 读):

位段	名称	含义
31:28	HAR_ENABLE	0x5 启用 Halt After Reset;其他 = 关闭(默认)
27:24	REQ_MODE	0x5 = 请求 Testmode;其他 = Usermode(默认)
23:5	RFU	保留,BOS 不读
4	OS_ENTRY_ALLOWED	1 = Host 允许 BOS 跳到 User-

位段	名称	含义
		OS(握手用)
0	CONT_VALID	1 = 上面这些位有效;0 = 整 HT2HSMS 无效(默认)

HSM2HTS(HSM/BOS → Host,HSM 写、Host 读):

位段	名称	含义
24	TM_CMD_FINISHED	Testmode 命令完成(仅 Testmode)
17:16	ACT_MODE	00 = Boot 过程中;01 = Usermode Entry Prepared;10 = Testmode;11 = 错误 / Sleep
15:0	STATUS_CODE	状态码(见下表)

STATUS_CODE 关键值(TS Table 11-4):

值	含义
0x0000	无错误
0xF001	检测到安全攻击
0xF002	User-OS 复位向量 / 栈指针无效
0xF003	AesKey0/1 锁定失败(Usermode)
0xF004	HSM Config Area 锁定失败
0xF005	CACHE CONFIG BOS bit 锁定失败
0xF006	PAU BOS bit 锁定失败
0xFF01	Testmode signature 无效
0xFF02	DLT 执行权限检查失败
0xFF03	Testmode 功能被锁
0xFFFF	HSM Config Area hash 无效
0xC000	Special Exception — Bus Fault
`0xC000 \	bridgeStatus(13 bit)`

值	含义
`0xEE00 |	isrNumber(8 bit)`

实战提示: 在 demo 里 HSM_0_ISR 经常读 HSM2HTS 后用 (cmd << 8) | status 形式上报;Host 端如果看到 0xEE03 这种值,要意识到这是 HSM 在告诉你它进了 NMI 异常 3(Hard Fault)。

5.2.4 前台 vs 后台 Secure Boot(TS §11.4)

当 PROCONHSMCFG.SSWWAIT=1 时,SSW 会等 HSM 释放 Host 才开始跑用户代码;HSM User-OS 有两种策略:

- Background(后台):HSM 启动后立即写 HSM2HTF.0=1,把 Host 释放;HSM 跑 Secure Boot 校验与 Host 用户代码同时进行(缩短启动时间)。Host 不会因 HSM 校验失败而卡死,但 Host 自己要轮询最终结果。
- Foreground(前台):HSM 跑完 Secure Boot 校验后,只在没检测到安全违规时才写 HSM2HTF.0=1。Host 用户代码被 SSW 一直按住不放,直到 HSM 确认安全。HSM 可选地向 HSM2HTS 写个随机 seed,SSW 用来搅乱功耗特征(防 SPA)。

5.3 关键 UCB 块

UCB 在 DFlash0 区域,TC3xx 一共约 25 个命名的 UCB(数量因芯片型号略异),每个 UCB 在物理上存在 ORIG + COPY 两份(共约 50 个物理块),内容必须一致,目的是防单点失效。SSW/HSM 读取时优先用 ORIG,ORIG 损坏时回退到 COPY。

UCB 名	作用	在 demo 中的体现
UCB_BMHDx_ORIG/COPY (x=0~3)	4 个 Boot Mode Header,SSW 启动时选用	Ifx_Cfg_SswBmhd.c
UCB_HSM_ORIG/COPY	HSM 调试/测试控制位 (HSMDBGDIS/DBGIFLCK/HSMTSTDIS...)	Ifx_Cfg_SswUcbHsm.c(默认 UNLOCKED)
UCB_HSMCOTP0_ORIG/COPY	HSM 启动核心配置 —— 启动 sector、EXCLUSIVE、HSMBOOTEN、SSWWAIT 等	demo 关键
UCB_HSMCFG_ORIG/COPY	HSM RAMKEEP、HSM 调试	-

UCB 名	作用	在 demo 中的体现
	password 等	
UCB_HSMDX_ORIG/COPY	HSM 调试扩展 / 证书 (PASSWORD/CERTIFICATE 模式)	-
UCB_DBG_ORIG/COPY	TriCore 自己的调试保护(password)	-
UCB_SWAP_ORIG/COPY	SWAP/OTA 切换标志	AP32404 demo

最关键的位 PROCONHSMCFG(在 UCB_HSMCOTP0 内):

位	含义	典型值
HSMBOOTEN	HSM 是否启动	1=启用,0=不启用(HSM 不工作)
SSWWAIT	SSW 是否等 HSM 释放	1=等(安全启动场景必须置 1)
HSMDX	HSM 数据段是否 HSM 独占	1=独占(host 无法读 HSM 私钥)
HSMRAMKEEP	HSM RAM 是否跨复位保留	0=清空(默认)
HSMENPINS	HSM 是否能 force pin	0/1
HSMENRES	HSM 是否能触发 reset	0/1
DESTDBG	Debug entry 是否破坏性	0/1
BLKFLAN	全部 Flash range 上是否允许某些函数	-

`DMU_SP_PROCONHSMCBS`(在 UCB_HSMCOTP0 内)控制 4 个 BOOTSEL:

位段	名称	含义
BOOTSEL0 (5:0)	第 1 优先选 sector	值 0–27 编码 HSM0X–HSM27X, 地址 0x80000000 + 0x4000 * sector
BOOTSEL1 (13:8)	第 2 优先选 sector	同上
BOOTSEL2 (21:16)	第 3 优先选 sector	同上
BOOTSEL3 (29:24)	第 4 优先选 sector	同上

BOS 按 BOOTSEL0 → 1 → 2 → 3 顺序试,任一成功就跳到那个 sector 的 User-OS;4 个全失败 STATUS_CODE=0xF002,死 Sleep。物理上 PFlash 0 的 0x80000000–0x8009FFFF 共有 40 个 16 KB sector(HSM0X–HSM39X),BOS 编码上只用前 28 个。

`DMU_SP_PROCONHSM`(在 UCB_HSM)控制 debug:

位	含义
HSMDBGDIS	1 = 禁 HSM 调试;BOS 在 startup 时按此位配 DBGCTRL.HSM
DBGIFLCK	1 = 把 HSM Config Area 锁成"写锁 + 不可再开 debug"

5.4 BMHD(Boot Mode Header)

BMHD 是 TriCore 启动模式的"配置单",不是 HSM 自己的,但理解 HSM 启动必须先懂它。

```
// AP32391 demo 里的 BMHD 配置
const Ifx_Ssw_Bmhd bmhd_0_orig = {
    0x00FE,           // BMI (Boot Mode Index): 0xFE = Internal start from Flash
    0xB359,           // BMHD ID (魔术数,必须是 B359)
    0xA0000000,       // STAD: 启动地址(用户代码第一条指令)
    0x31795570,       // CRC of above
    0xCE86AA8F,       // CRC inverse
    /* ... 大量 reserved 与 password ... */
    0x43211234        // confirmation word(必须)
};
```

要点:

- BMI = 0xFE = Internal start from Flash
- BMI = 0x00 = ASC BSL 模式(从 UART 引导)
- BMI = 0x3C / 0x3D / 0x3E / 0x3F = 备用 ABM
- password 8 个 word(共 256 bit)用于保护 UCB_BMHD,SSW 会做 password 校验:4 个 BMHD 任何一个 password 错,该 BMHD 就被 SSW 当作"无效"跳过,直到找到下一个有效的;不会让整片锁死,但该 BMHD 自身失去更新能力(要重写必须用 CPU 编程接口,且 SSW 仍可能继续走别的 BMHD)。所以工程上常说"UCB 写一次就要三思"

CH 06

第 6 章

Host ↔ HSM 桥接机制

这是 HSM 编程的核心,所有 HSM 命令都走这套机制。

6.1 桥接寄存器完整清单(TS Table 9-3 / 9-4)

基地址 0xF0040000 起,占 128 KB 地址空间(实际 SFR 只到 0xF0050000 多一点,DBGMEM 调试窗口占 0xF0050000–0xF005FFFF)。

读写权限速记:rr(read + write-1-reset)、rs(read + write-1-set)、rw(read + write)、ro(read-only)、w(write-only)、/(无访问);类型 (HSM/Host) 表示 HSM 视角权限 / Host 视角权限。例:rr/rs = HSM 可读 + 写 1 清;Host 可读 + 写 1 置。

Host 与 HSM 双方都可见的寄存器(Bridge Communication):

寄存器	偏移	类型 (HSM/Host)	用途
HSM_ID	0x08	ro/ro	模块 ID(读出 = 0x00BEC002,MODULE=0x00BE, TYPE=0xC0=32-bit, REV=0x02)
HT2HSMF	0x20	rr/rs	Host → HSM Flag:32 个 flag,Host 写 1 set,HSM 写 1 clear;用于触发 HSM 中断
HT2HSMIE	0x24	rw/–	HSM 中断使能(只有 HSM 能写,Host 只读)
HSM2HTF	0x28	rs/rr	HSM → Host Flag:HSM 写 1 set,Host 写 1 clear;用于触发 Host 中断
HSM2HTIE	0x2C	ro/rw	Host 中断使能(只有

寄存器	偏移	类型 (HSM/Host)	用途
			Host 能写)
HSM2HTIS	0x30	ro/rw	Host 中断路由选择:0 = 中断 0,1 = 中断 1(32 位分别映射)
HSM2HTS	0x34	rw/ro	HSM → Host Status(状态/响应数据)
HT2HSMS	0x38	ro/rw	Host → HSM Status(命令字、参数)

rr/rs 含义:HSM 视角是 rr(读 / 写 1 清),Host 视角是 rs(读 / 写 1 置)。

只有 HSM 可见的寄存器:

寄存器	偏移	用途
CLKCTRL	0x40	HSM 时钟分频 (00=SPB,01=SPB/2,10=SPB/4); Host 读不到,只能软件约定
DBGCTRL	0x60	Debug 控制:HSM 位开放 HSM 内部资源给 host 调试器,HOST 位开放 host 调试
PINCTRL	0x64	控制 2 个 pin 输出(OEN0/1 切换 normal/HSM 控,VAL0/1 驱动值,LOCK 后只读)
ERRCTRL	0x80	错误标志(Bus error / ECC / 外部调试器状态)
ERRIE	0x84	错误中断使能
ERRADDR	0x88	最后一次 bus error 的地址(只记最后一次)
EXTIF	0xA0	32 个外部中断标志
EXTIE	0xA4	32 个外部中断使能
SAHBASE	0xC0	Single Access to Host Memory 基址(高 16 位,低 16 位 =

寄存器	偏移	用途
		0xF0050000 起偏移)
SAHMEM	0x10000–0x1FFFF	64 KB 窗口;HSM 端读 Host 内存用,等效 DBGMEM
RSTCTRL	0xE0	复位控制(可触发 app reset / system reset)
RSTPWD	0xE4	复位密码(写 RSTCTRL 前先写对密码)
SENSIF	0xF0	10 个传感器标志
SENSIE	0xF4	10 个传感器中断使能
SENSAPPRST	0xF8	10 个传感器应用复位使能
SENSSYSRST	0xFC	10 个传感器系统复位使能

只有 Host 可见的寄存器:

寄存器	偏移	用途
HSMCTRL	0x1000	HSM 启动控制:EOMBT(End of MBIST Test)写 11b 释放 HSM 复位;00b = MBIST 中,HSM 保持 reset;01b/10b = 准备态
DBGBASE	0x1010	调试窗口基址(配合 DBGMEM)
HSMOTGB	0x1020	OCDS Trigger 总线控制(配 OTGB0/1 / Trigger Set)
DBGMEM	0x10000–0x1FFFF	64 KB 调试窗口;Host 调试器读 HSM 内部用(只对 HSM 调试已开有效)

6.2 单次握手(典型)

```
Host 端                                  HSM 端
──────                                   ──────
1. 把命令参数写到 Host 共享内存
```

```
   (地址 0xB0040080 起的 32 KB 区域)
2. 写 HT2HSMS = 命令号
3. 写 HT2HSMF = HSM_MODE_COMMAND
   → 触发 HSM 中断
                                        中断到达 → BridgeService_handler()
                                        4. 读 HT2HSMS → 得到命令号
                                        5. 写 SAHBASE = 目标 Host 地址
                                           后续用 0xF0050000 窗口访问
                                        6. 跑命令 (例如 AES_CMAC)
                                        7. 结果回写 Host 共享内存
                                        8. 写 HSM2HTS = (cmd << 8) | status
                                        9. 写 HSM2HTF → 触发 Host 中断
10. Host ISR 收到 HSM2HTF
11. 读 HSM2HTS 拿状态
12. 从 Host 共享内存读结果
```

6.3 共享内存窗口 SAHMEM

HSM 端访问 Host 内存时,不能直接用 0xB0040000,要用 0xF0050000 这个 64 KB 窗口,通过 SAHBASE 寄存器做"基地址重定位":

```
// 来自 HSM20_app bridge.h
inline uint32_t* HostWritePointer(uint32_t address)
{
    HSM_BRIDGE->SAHBASE.U = (address);
    return (uint32_t*)(((uint32_t)(address) & 0x0000FFFFu) | 0xF0050000u);
}
```

含义: 任何想访问 Host 内存的 HSM 代码,先把目标地址低 16 bit 写到 SAHBASE,然后用 0xF0050000 + 低 16 bit 就能访问。窗口只能开 64 KB,大块数据要分块处理。

6.4 Host 侧 API 风格(AP32391 demo)

demo 里 Host 端不是直接操作寄存器,而是用"宏"包了一层,调用时接近函数:

```
// HSM 收到命令
HSM_HT2HSMS = HSM_AesCMAC;          // 命令号
hsm.parameter[0] = (uint32)&cryptoBlock;  // 参数:输入数据地址
hsm.parameter[1] = dataLength;             // 参数:长度
hsm.parameter[2] = (uint32)&key;           // 参数:密钥地址
hsm.parameter[3] = (uint32)&resultBuf;     // 参数:输出地址
HSM_HT2HSMF = HSM_MODE_COMMAND;      // 触发中断
```

demo 中 hsm 是 32 个 word 的 HSM_INTERFACE_Type 结构,放在 .hsmCtrl 段,物理地址 = 0xB0040000 起的"控制区"。cryptoBlock 是 32 KB 的 .hsmBlock0 段,做"数据区"。

6.5 32 个外部中断源(TS Table 9-1,HSM 端用 `EXTIF`/`EXTIE` 控制)

#	源	SRN equivalent
0	GTM_TIM0_IRQ4	SRC_GTMTIM04
1	GTM_TIM0_IRQ5	SRC_GTMTIM05
2	GTM_TIM0_IRQ6	SRC_GTMTIM06
3	GTM_TIM0_IRQ7	SRC_GTMTIM07
4	GTM_ATOM0_IRQ2	SRC_GTMATOM02
5	GTM_ATOM0_IRQ3	SRC_GTMATOM03
6	GTM_MCS0_IRQ0	SRC_GTMMCS00
7	GTM_MCS0_IRQ1	SRC_GTMMCS01
8	DMA Error	SRC_DMAERR
9	RCU(Shutdown Trap)	无对应 SRN,直接连到 RCU
10	CCU60 SR0	SRC_CCU60SR0
11	CCU61 SR0	SRC_CCU61SR0
12	PSI5	SRC_PSI50
13–18	MCMCAN0 SR0–SR5	SRC_CAN0INT0– SRC_CAN0INT5
19–20	VADC CG0 SR0/SR1	SRC_VADCCG0SR0/SR1
21–22	VADC CG1 SR0/SR1	SRC_VADCCG1SR0/SR1
23	DSADC SRM3	SRC_DSADCSRM3
24	DAM0 Ready	SRC_DAMRI0
25	DAM0 Limit	SRC_DAMLI0
26	DAM1 Ready	SRC_DAMRI1
27	DAM1 Limit	SRC_DAMLI1
28	Ethernet	SRC_ETH
29	PMU0 End of busy(CPU)	SRC_PMUHOST

#	源	SRN equivalent
30	PMU0 End of busy(HSM)	无 SRN,直连 PMU0
31	CPU(仅 INT_SRB0.0)	SRC_GPSR00

应用场景:HSM 可以监听 CAN 总线(看 OTA 包)、ADC(看传感器异常)、GTM 定时器(做时间窗认证)等。

6.6 10 个传感器输入(TS Table 9-2,电压/温度/时钟异常)

#	源	用途
0	5.0V 欠压	EVR 异常
1	3.3V 欠压	
2	1.2V 欠压	
3	5.0V 过压	
4	3.3V 过压	
5	1.2V 过压	
6	DTS 过温	Die Temperature Sensor
7	DTS 欠温	
8	SPB 快速过频(SCU_CCUCONSM)	时钟攻击检测
9	SPB 欠/过频(SCU_CCUCON4)	

SENSIF/SENSIE/SENSAPPRST/SENSSYSRST 四组寄存器并行控制同一组源:每个传感器标志被 set 后,可分别触发 HSM 中断、Application Reset、System Reset。优先级: System Reset > Application Reset > Interrupt(高优先级动作会屏蔽低优先级)。

软件触发复位:先写对 RSTPWD 密码,再写 RSTCTRL(防误操作)。触发后 chip 复位原因记录在 SCU_RSTSTAT:bit 26 = HSM 触发的 system reset,bit 27 = application reset。

6.7 中断/复位触发控制(完整流程)

- Host → HSM 命令(典型 handshake):
- Host 写共享内存(参数 + 数据);
- Host 写 HT2HSMS(32-bit 命令/状态);

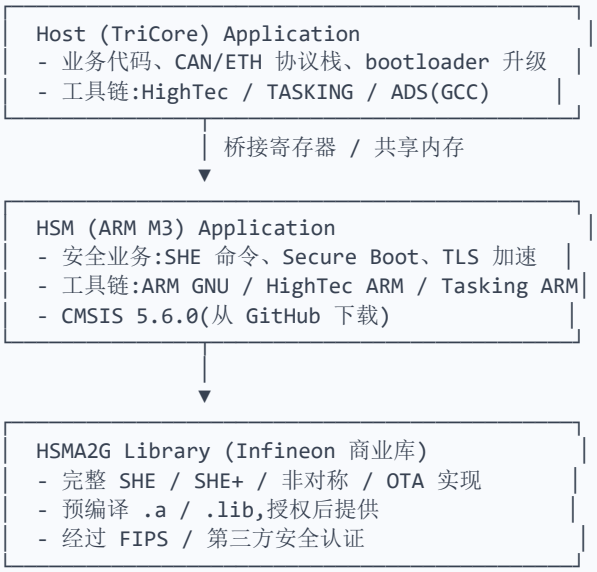
- Host 写 HT2HSMF 的 bit i 写 1 → 触发 HSM Bridge Service 中断(异常 19);
- HSM ISR 读 HT2HSMS,跑命令;
- HSM 写 HSM2HTS(结果);
- HSM 写 HSM2HTF 的 bit j 写 1 → 触发 Host 中断(走 HSM2HTIS 配置的 0/1 两条 SR 线之一);
- Host ISR 读 HSM2HTS,清 HSM2HTF 对应 bit。
- HSM → Host 通知(单边,例如 Secure Boot 完成):HSM 直接 HSM2HTF.0=1,Host 在 HSM_0_ISR 里感知。AP32391 demo 用此机制释放 CPU0。
- 中断处理顺序:先清 NVIC pending(进入 handler 时自动 / 软件清),后清 bridge flag;否则下一次中断不会被识别(pulse-triggered)。这是新人最常踩的坑。

CH 07

第 7 章

HSM 软件分层与工具链

7.1 三层结构



实战建议:

- 学习阶段:从 demo (HSM20_app + HSM_TC39xB_demo) 入手,理解桥接和命令。
- 产品阶段:用 Infineon 官方 HSMA2G 库(也叫 "CycurHSM")。里面已经实现 SHE、Secure Boot、OTA、HSM 主机驱动、Challenge-Response、Key Management。
- 量产阶段:网关 / ADAS / 动力域等 ECU 通常会集成 Vector MICROSAR Safe(基于 AUTOSAR 的 Security 套件)或 ETAS CycurHSM(基于 HSMA2G 封装)这类商业安全中间件,通过标准 API 接入即可,不必自己重写底层。

7.2 demo 工程的工具链

HSM20_app 目录下有显式的工具链说明:

- 编译器:ARM GNU(免费,CMSIS 工程友好)

- 调试器:Lauterbach Trace32 / PLS UDE / iSYSTEM winIDEA
- 链接脚本:HsmLinker.lsl / Lcf_Gnuc.lsl
- 烧录:TriBoard 自带 SEGGER J-Link,或用 UDE

7.3 调试 Host 端 HSM 通信

打开 AURIX_HSM Revealed_Training_Slides.pdf(项目根目录),里面展示了如何用 UDE / AURIX Development Studio 调试 HSM 内核、看桥接寄存器、观察 AES 寄存器、跟踪 PKC 运算。

CH 08

第 8 章

HSM 命令集速查

这是 AP32391 demo(HSM20_app/inc/hsmCmds.h)内置的命令集。HSMA2G 库的命令集更全(>100 条,SHE 标准 28 条 + 各种扩展)。

```
typedef enum {
    HSM_EncryptECB     = 0,    // AES-128 ECB 加密
    HSM_DecryptECB     = 1,    // AES-128 ECB 解密
    HSM_EncryptCBC     = 2,    // AES-128 CBC 加密
    HSM_DecryptCBC     = 3,    // AES-128 CBC 解密
    HSM_AesLoadKey     = 4,    // 把 Host 内存的 key 装入 HSM 内部 key slot
    HSM_AesLoadIV      = 5,    // 装 IV
    HSM_AesCMAC        = 6,    // CMAC 一次性计算(用 host 内存中数据)
    HSM_AesCMACOpt     = 7,    // 优化的 CMAC(走 cache)
    HSM_AesCmacGenSubKeys = 8, // CMAC 子密钥生成(K1/K2)
    HSM_TrngGetRandomNumber = 9,// 从 TRNG 读随机数
    HSM_PrngSetSeedNumber   = 10,// PRNG 注入种子
    HSM_PrngGetRandomNumber = 11,// PRNG 出数
    HSM_HashSha256     = 12,   // SHA-256 一次性计算
    HSM_HMACSha256     = 13,   // HMAC-SHA256
    HSM_ServiceWDT     = 14,   // 服务 HSM 看门狗
    HSM_InitDataFlash1 = 15,   // 初始化 HSM 专属 DFlash1
    HSM_CacheOperation = 16,   // 操作 HSM cache(预取/清理/旁路)
} HSM_CMD_Type;
```

注意:demo 命令集只覆盖了 TS 定义的子集。HSM AES-128 硬件实际还支持 CTR / OFB / CFB / GCM / XTS 5 种模式(TS §4.2),需要在 HSM 固件里加命令才能用;Hash 也支持 MD-5 / SHA-1 / SHA-224(SHA-224 由 SHA-256 + 软件派生)。

8.0.1 HSM 硬件加速器全命令族(TS §4 AES / §5 Hash / §6 PKC)

模块	TS 文档命令(实现需要 HSM 固件封装)
AES-128	ECB / CBC / CTR / OFB / CFB / GCM / XTS 7 种模式(§4.2);AES-CMAC 模式(走 K1/K2 子密钥,§4.2.7)
Hash	MD-5 / SHA-1 / SHA-224 / SHA-256(§5.1);HMAC 用 SHA-256
PKC	算子级:ADDN, SUBN, MULTN, RED, DIVN, INV, MULT, INV2, RED2, EXP(§6.5.1); ECC 点运算:PDBL, PADD, SMULT,

模块	TS 文档命令(实现需要 HSM 固件封装)
	CHECKAB, CHECKN, CHECKPXY(§6.5.2); Curve25519/Ed25519:SMULT25519, XRECOVER, SMULTED25519, CHECKVALID(§6.5.2.7–10); ECDSA:ECDSASIG, ECDSAVER(§6.5.2.11–12); F(2^m) 域版本有 ADDN2/SUBN2/MULTN2/PDBL2/PADD2/SMULT2/ECDSASIG2 等(§6.2.5/§6.5.1/§6.5.2)
WDT	服务(WDT_SRV)、初始化(WDT_SRV_INI)、重载值设置
Cache	全清(CACHE_AC)、清 set(CACHE_SC)、清 block(CACHE_BC)、lock/unlock、旁路

关键: HSM 没有 RSA 硬件(TS 没提)。要做 RSA 必须用 PKC ECC 算子软件模拟,慢、不安全、不推荐;量产请用 HSMA2G 商业库的 TLS/签名接口。

8.1 命令执行流程总览(`bridge.c`)

```
void BridgeService_handler(void)   // HSM 中断
{
    uint32_t flags = HSM_BRIDGE->HT2HSMF.U;   // 读 Host 请求

    switch(flags) {
        case HSM_MODE_GET_BASE_ADDRESS:       // Host 第一次握手:告诉 HSM 共享内存地址
            hostBaseAddress = (uint32_t*)HSM_BRIDGE->HT2HSMS.U;
            mode = HSM_MODE_GET_BASE_ADDRESS;
            break;

        case HSM_MODE_COMMAND:                // Host 发起命令
            if ((status & HSM_STATUS_BUSY) != HSM_STATUS_BUSY) {
                hostCmd = HSM_BRIDGE->HT2HSMS.U;   // 读命令号
                mode = HSM_MODE_COMMAND;
            } else {
                // 忙,告诉 Host 你现在忙
                HSM_BRIDGE->HSM2HTS.U = (hostCmd << 8) | status;
                HSM_BRIDGE->HSM2HTF.U = HOST_MODE_ERROR;
            }
            break;
    }
    status |= HSM_STATUS_BUSY;
    HSM_BRIDGE->HT2HSMF.U = flags;  // 清中断
}

void CheckForCommand(void)  // HSM 主循环调用
{
    if (HSM_STATUS_BUSY == (HSM_STATUS_BUSY & status)) {
        switch (mode) {
            case HSM_MODE_COMMAND:
```

```
                CommandProcessMulti(hostCmd);  // 派发到具体命令实现
                break;
            case HSM_MODE_IDLE:
                status &= ~HSM_STATUS_BUSY;
                HSM_BRIDGE->HSM2HTS.U = (hostCmd << 8) | status;
                HSM_BRIDGE->HSM2HTF.U = HOST_MODE_COMMAND;  // 通知 Host 完成
                break;
            // ...
        }
    }
}
```

CommandProcessMulti 内部是一个函数指针数组,根据 hostCmd 索引跳到 AesEncryptECB / AesCMAC / HashSha256 / ... 等具体实现。

CH 09

第 9 章

Secure Boot 实战(AP32391 demo 详解)

这是新人最该读透的 demo,在 AP32391_Secure_Boot_with_HSM_v1.3.3_SW 目录下。

9.1 目录布局

```
AP32391_Secure_Boot_with_HSM_v1.3.3_SW/
├── HSM_TC39xB_demo/                # TriCore 侧(host)
│   ├── 0_Src/AppSw/Tricore/
│   │   ├── Main/                    # Cpu0/1/2/3/4/5_Main.c
│   │   ├── App/                     # scheduler.c, msg/
│   │   ├── Cfg_Ssw/                 # BMHD、UCB HSM 配置
│   │   └── Cfg_Ssw/...
│   └── ...
├── HSM20_app/                       # HSM 侧(ARM M3)
│   ├── CMSIS/Include/               # 寄存器定义、core_cm3
│   ├── src/                         # main_app, bridge, cache, cryptoAes, ...
│   ├── inc/                         # hsmCmds, bridge, nvm_data
│   └── Makefile
├── hsma2g/                          # 商业库(本目录是 demo 简化版)
│   └── (空 — 商业版需向 Infineon 申请授权)
└── README.md
```

9.2 TriCore 侧启动流(Host)

- CPU0 上电 → 跑 SSW(Ifx_Ssw.c,不用改,Infineon 提供)。
- SSW 读完 BMHD/UCB,跳到用户启动代码 `Cpu0_Main.c`:

```
void core0_main (void)
{
    IfxScuWdt_disableCpuWatchdog(IfxScuWdt_getCpuWatchdogPassword());
    IfxScuWdt_disableSafetyWatchdog(IfxScuWdt_getSafetyWatchdogPassword());
    IfxCpu_emitEvent(&cpuSyncEvent);
    IfxCpu_waitEvent(&cpuSyncEvent, 1);  // 等其他 CPU
    scheduler();
}
```

- `scheduler()` (`App/scheduler.c`) 做几件事:
- 初始化 ASCLIN0(UART 给上位机用)
- 初始化 GPT12(1ms 定时器)

- 检查 DMU_SP_PROCONHSMCFG.B.HSMBOOTEN,如果 HSM 启用:
- 设置 HSM 中断路由 SRC_HSM0
- 调用 MSG_SecureBootFinished() 把"BOOTUP 消息就绪"标志置上(等 HSM 真的释放 CPU0 后,再由 MSG_TxMessage 通过 UART 发给上位机)
- 写 HSM_HT2HSMS = &hsm(共享内存基地址)
- 写 HSM_HT2HSMF = HOST_MODE_SET_BASE_ADDRESS(触发 HSM 中断,让 HSM 知道共享内存的物理地址)
- 进入主循环:ASCLIN0_CheckMsgPending() + 1ms/50ms/500ms 任务 + 处理 HSM 中断
- TriCore 端 HSM 中断处理 (`HSM_0_ISR`):

注:这是 CPU0(TriCore 侧)的中断,不是 HSM 自身的中断。HSM 写 HSM2HTF=1 后,该 ISR 触发。

```
void HSM_0_ISR(void) {
    timeLowFinish = STM0_TIM0.U;
    timeHighFinish = STM0_CAP.U;
    uint32 flags = HSM_HSM2HTF;
    lastCmd = HSM_HSM2HTS;  // 读 HSM 状态
    HSM_HSM2HTF = flags;   // 清中断(写回 flags)
    hsmInterrupt = true;
}
```

后续主循环检查 hsmInterrupt 标志,调用 MSG_TxMessage() 把结果通过 UART 报给 PC 上位机(为了 timing 测量)。

9.3 HSM 侧启动流(ARM M3)

- POR / 系统复位 → BOS(HSM BootROM)启动
- 读 UCB_HSMCOTP0_ORIG 和 UCB_HSMCOTP0_COPY(必须同时有效,或用 ORIG 优先)
- 校验 BOOTSEL0/1/2/3 选 HSM 启动 sector
- 跳到 HSM PFlash 用户固件入口
- HSM 用户固件入口 `main_app.c`:

```
void main_app(void)
{
    SystemInit();              // 初始化 ARM M3 核、SP、VTOR

    // 1. 跑 Secure Boot 校验
    if (E_CRYPTO_OK == CheckHashApplication()) {
        // 2. 读 CPU0 当前 PC(应该还在 BootROM)
        cpu0_PC = *(volatile uint32_t*)(0xF881FE08);
```

```
        // 3. 释放 CPU0(写 1 到 HSM2HTF)
        HSM_BRIDGE->HSM2HTF.U = 1;
    }
    // 4. 抓 STM 计时戳
    time1L = STM0_TIM0.U;
    time1H = STM0_CAP.U;
    // 5. 配 NVIC,使能 HSM 桥接中断
    HSM_BRIDGE->HT2HSMIE.U = 0xFFFFFFFF;
    NVIC_SetPriority(Bridge_IRQn, 0);
    NVIC_EnableIRQ(Bridge_IRQn);
    // 6. 写回 Host 共享内存:
    //    0xB0040010 起始:time0H, time0L, time1H, time1L + 2 保留 + Build info
    //    0xB0040090 起始:cpu0_PC
    // 7. SysTick 10ms 启动
    // 8. 初始化 TRNG(异步预热)
    while (1) {
        CheckForCommand();   // 监听 Host 命令
        Trng_CheckBuffer();  // 维护 TRNG 熵池
        // LED 心跳...
    }
}
```

- `CheckHashApplication()` 详解(secure boot 核心):

```
uint8_t CheckHashApplication(void)
{
    cryptoCheck = cryptoInitBootParameters();  // 从 HSM config 区域读 key、marker、地址
    if (E_CRYPTO_OK == cryptoCheck) {
        if ((0x01 & hsmCfgData.BootOptions) == 1) {  // BootOptions bit0: 是否"边校验边释放"
            CCU_RampDownSystemPll();    // 恢复默认时钟
            HSM_BRIDGE->HSM2HTF.U = 1;  // 提前释放 CPU0(优化启动时间)
        }
        cryptoCheck = CmacOnBoot();     // 跑 CMAC 校验 Host 代码
    }
    if (true == clockWasSet) CCU_RampDownSystemPll();
    return E_CRYPTO_OK;
}
```

- `CmacOnBoot()` 详解(AES-CMAC 校验 Host PFlash):

```
uint8_t CmacOnBoot(void) {
    if (hsmCfgData.BootMarker == 0xAA550000) {  // 魔数确认
        AES_DATA_Type *in = (AES_DATA_Type *)(hsmCfgData.BootMacStartAddress);
        rounds = AesCMAC_CalculateLastBlock(hsmCfgData.BootMacSize, in);
        HSM_AES->AESCTRL.U = AES_CTRL(OPC_WK, KEYNR_0, CVNR_0);
        HSM_AES->AESCTRL.U = AES_CTRL(OPC_WCV, KEYNR_2, CVNR_0);  // 0 IV
        if (rounds > 1) {
            AesCmacMove((uint32_t *)in, rounds-1, AES_CTRL(OPC_CBC_ENC, KEYNR_2, CVNR_0));
        }
        // 最后一轮
        HSM_AES->AESIN0.U = M_last.w[0]; // ...
        HSM_AES->AESCTRL.U = AES_CTRL(OPC_CBC_ENC, KEYNR_2, CVNR_0);
        while (HSM_AES->AESSTAT.B.BSY == true);
        // 写到 Host 共享内存 0xB0040000
        HSM_BRIDGE->SAHBASE.U = (uint32_t)Boot_ResponseAddress;
        AesGetResult(out);
        out += 8;
        *out++ = hsmCfgData.BootMacSize;
        *out = hsmCfgData.BootMacStartAddress;
```

```
        result = E_CRYPTO_OK;
    }
    return result;
}
```

核心思想:

- HSM 内部有配置区 hsmCfgData,生产时由 OEM 烧入主密钥 / 派生密钥 / 启动参数 (BootMarker、BootMacStartAddress、BootMacSize 等)。
- 校验时,HSM 用配置区里的 K1/K2(由主密钥派生,或运行时调用 AesCmacGenSubKeys 生成),对 Host PFlash 的指定段做 AES-CMAC。
- 校验通过 → 写 HSM2HTF = 1 释放 CPU0;失败 → 死循环(永远不释放,host 永远卡在 SSW)。
- 这种"密钥永远不离开 HSM"的设计,就是 SHE 标准的核心思想。

9.4 用 demo 实测一把(简要)

- 打开 HSM_TC39xB_demo 用 ADS / HighTec 编译,烧到 TC39xB TriBoard。
- 打开 HSM20_app 用 ARM GNU 编译,烧到 HSM PFlash 对应 sector。
- 上电,PC 上位机(通过 ASCLIN0 UART)能看到 secure boot 完成后 HSM 报回的:
- CMAC 结果
- 校验耗时(time0L/time0H/time1L/time1H)
- HSM 固件版本号
- CPU0 释放时的 PC
- 上位机发送 SERIAL_CMD_HSM + 命令号,触发 HSM 跑 AES / SHA / CMAC / TRNG 等。
- 上位机收到 MSG_TxMessage() 回包,带耗时测量,即可做 HSM 性能评估。

CH 10

第 10 章

Debug 保护(AP32399 / AP32562)

10.1 调试保护的几种模式

TC3xx 的调试保护按强度由低到高,大致分这几档(具体术语以 AN AP32399 / AP32481 为准):

模式	描述	解除方式
OPEN	调试接口完全开放,任何 DAP/JTAG 都能进	重新擦写 UCB_DBG
PASSWORD / PROTECTED	调试接口锁定,需输入 UCB_DBG 中预存的 password(由 OEM 设置)	在 UCB 解除流程中输入正确 password
CERTIFICATE	调试接口锁定,需 Infineon 签名的证书(基于 UCB_HSMDX 中的公钥)	OEM 出具的 CSR 送回 Infineon,Infineon 签名后回写,设备端 HSM 验签后开放
DESTRUCTIVE / DISABLED	永久禁用调试,且 debug entry 会自动擦除 HSM PFlash 与 DFlash 关键数据(DESTDBG=1 时的行为)	不可逆 —— 工厂烧录完、产线下线前最终一次确认

10.2 谁来锁?

锁的层级	谁锁	谁解
TriCore 自己的调试接口	TriCore 写 UCB_DBG	TriCore 自己解
HSM 自己的调试接口	HSM 写 UCB_HSM(HSMDBGDIS / DBGIFLCK 写到 DMU_SP_PROCONHSM)	只有 HSM 能解 — BOS 启动时读这俩位去配 DBGCTRL.HSM/HOST

关键:HSM 锁的,TriCore 永远解不了。这是 security 设计的根基 —— 哪怕攻击者拿到完整 TriCore 代码执行权,也关不掉 HSM 的保护。

10.3 HSM 调试工具(TS §12)

模块	能力	备注
FPB(Flash Patch & Breakpoint)	6 个硬件断点(FP_COMP0–5);不支持 flash patch(地址比较扩展到 32 bit 以支持 PFlash > 512 MB)	NUM_CODE2=0, NUM_LIT=2(占位), NUM_CODE1=6
DWT(Data Watchpoint & Trace)	4 个 trigger,可作 watchpoint 或 value compare(数据断点)	触发信号通过 OTGS 给 host OCDS
ROM Table	4 KB(0xE00FF000),自动暴露给调试器做模块识别	

注意: HSM 没有 ITM、ETM、TPIU —— 完全无 trace 能力(TS §12 明确)。要观察 HSM 行为只能靠:

- 6 个断点
- 4 个 watchpoint
- 桥接寄存器(HSM2HTS、桥状态位)
- 调试器读 HSM 全部 SFR/RAM/ROM 通过 DBGMEM 64 KB 窗口(0xF0050000 起)

HAR (Halt After Reset) 模式(TS §11.2.3 / §12.3.4):

- Host 在 HT2HSMS 里设 HAR_ENABLE = 0x5 + HSM 调试模式已开(BOS 在 startup 阶段会检查这俩条件)
- BOS 跳到 User-OS 第一指令之前插入硬件 BREAK
- 调试器接管 PC,可单步 / 设断点 / 看 User-OS 状态
- HSM 调试模式未开时,HAR 请求被静默忽略(不报错)
- 适合现场第一次跑新固件,或怀疑 HSM 死机时抓现场

HALT 模式下 HSM 各模块行为(TS §12.3.5):

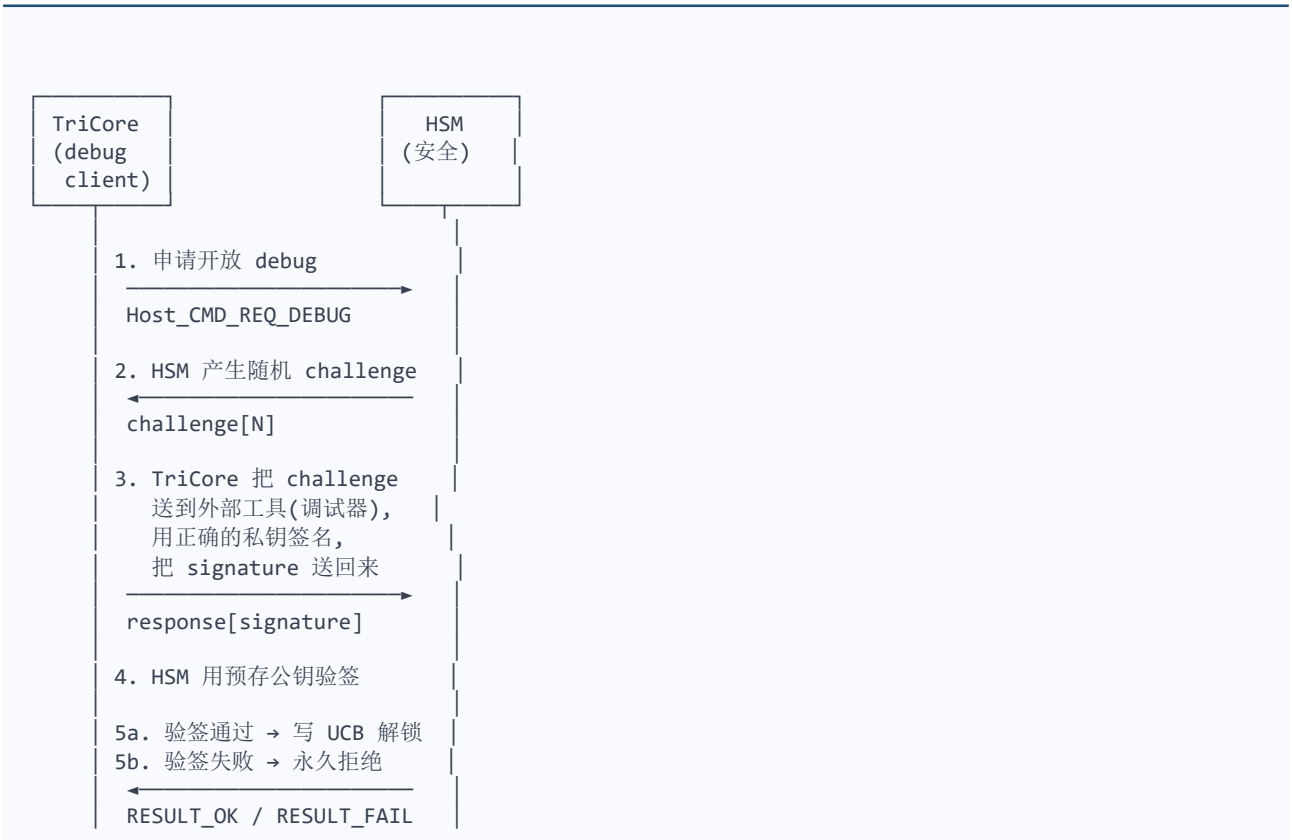
模块	HALT 时是否暂停
AES	不暂停 — 跑命令照常
WDT / TRNG / Timer / SysTick	暂停
PKC	TS 未明确,一般也暂停(指令流停了)

调试时钟(TS §12.3.2):HSM 主时钟 FCLK 暂停时,debug 模块仍由 free-running clock 供着,保证调试器即使在 HSM sleep 时也能访问 debug 寄存器。

SPB 总线冲突(TS §12.3.6,常见坑):调试器通过 DBGMEM 访问 HSM 时,SPB 总线被独占占用,HSM 这边如果同时发起对 host 资源的访问(包括 cache 缺失时),会触发 SPB 冲突 → 行为未定义。规避方法:调试器访问 HSM 时,HAL 端不能在访问 window 同时碰 host 资源;最稳妥是断点下在不会触发 cache writeback 的位置。

10.4 Challenge-Response(CnR)模式 —— AP32562 demo

在 PASSWORD 模式下,Host 端(被调试器)要重新开放调试,需要 HSM 介入做 Challenge-Response 协议:



应用场景: 现场设备,需要 RMA(返厂维修)或开发调试时,工程支持中心可以拿离线工具(用生产时烧入 HSM 的私钥)对设备做 CnR 协议,验证身份后开放调试,而不需要把密码写在文档里。

详细源码看 AP32562-Tutorial_CnR_Example_on_HSM_can_handle_debug_protection_v0.5.6/ 目录,里面 demo 包含:

- HSM_Demo_TC37A_AP32562_v0.5/:TriCore + HSM 双工程
- 提供:PC 端 Python 工具(算 signature) + 设备端 HSM 固件(验签)+ README

CH 11

第 11 章

OTA / SWAP 软件更新(AP32404)

11.1 SWAP 是什么

TC3xx 的 PFlash 划分 AB 两区(A 区 = 当前运行,B 区 = 备份)。OTA 升级时把新固件写到 B 区,写完后通过写 UCB_SWAP_ORIG 的标志位,通知 SSW 下次启动切换到 B 区。如果新固件跑挂了,SSW 会自动回滚到 A 区,这就是 A/B 备份(Automotive 行业称 A/B Swap,类似 Android 的 A/B 分区)。

11.2 升级流程概览

```
1. 旧版本固件在 A 区正常运行
2. 收到 OTA 包(经过 ECC 验签,HSM 硬件无 RSA)
3. 在 A 区控制下,擦 + 写 B 区
4. 写 UCB_SWAP,标记"下次启动从 B 区"
5. 系统复位 / POR
6. SSW 读 UCB_SWAP,跳到 B 区入口
7. 跑 Secure Boot 验证 B 区
8. 通过 → B 区成为新 A 区,继续运行
9. 失败 → 自动回滚到 A 区,旧版本继续运行
```

11.3 HSM 在 OTA 中的角色

- 验签 OTA 包:用 ECC 公钥验签(P-256),确认包是 OEM 签的(HSM 硬件无 RSA,大文件批量签名请用外部工具 + ECC)
- 保护密钥:OTA 包的解密密钥永远不会离开 HSM
- 完整性校验:新固件写入后,HSM 跑 CMAC/SHA 校验,确认 flash 内容没被篡改
- 写 UCB_SWAP:HSM 写比 TriCore 写更安全(因 HSM 在 UCB 写时也受 UCB 自身保护)

详细看 AP32404-Tutorial_AURIX_TC3xx_Using_SWAP_mechanism_for_software_updates_over_the_air_v1.3.0_SW/,里面有完整 demo + README。

CH 12

第 12 章

关键陷阱与最佳实践

12.1 死锁陷阱(AP32555 重点)

HSM 的几个"一不小心就 brick 板子"的场景:

场景	后果	应对
UCB_HSMCOTP0 写错 + confirmation 不匹配	芯片永久锁死	永远不要在不确定时一次性写 ORIG + COPY;先写 COPY 验证,稳定后再同步 ORIG
UCB password 写错	该 UCB 自身失去更新能力(SSW 会改用别的有效 BMHD)	写之前用工具算 CRC;第一次写 UCB 一定要留出调试通道
HSMBOOTEN=1 但 HSM PFlash 是空的	SSW 死循环	必须先烧 HSM 固件,再置 HSMBOOTEN=1
HSMRAMKEEP=1 但忘了清理安全数据	下次启动遗留密钥	安全敏感数据存 RAM 时,应用 HSMRAMKEEP=0,或显式 zeroize
烧 HSM 固件时 HSM 在运行	烧录失败或 HSM 错乱	必须先让 HSM 进入"reset / idle"状态才能更新 HSM PFlash
HSM WDT 没服务	HSM 复位 → 桥接通信中断	demo 里 HSM_ServiceWDT 命令要定期发
HSMENRES=0 但应用依赖 HSM 触发 reset	系统不能 reset	评估是否真要 HSM 触发 reset,通常给 HSM 触发能力
DESTDBG=1 误设 + 进 debug	HSM 擦 flash,NVM 密钥全毁	生产前反复测试;demo 默认 DESTDBG=0(非破坏)

12.2 cache 性能(AP32456)

- HSM 内部有 4 KB 4-way cache(TS §2.5.3,注意不是 8 KB,别再记错),16-byte block,64 sets,LRU 替换,主要用于加速 PFlash 数据读取(HSM 读 Host PFlash 是慢的)
- CMAC 大数据计算时,先 CACHE_BT 预取,再 AesCmacMove() 流水处理
- CACHE_AC 一次性全清;CACHE_SC 清某个 set;CACHE_BC 清某个 block
- mode=1 时旁路 cache(读 RAM 用)
- demo 中 cache.c 实现了 7 种 cache 操作命令

12.3 共享内存窗口(64 KB)限制

- 任何大于 64 KB 的数据块,HSM 必须分块处理,每次 SAHBASE = new_offset
- 实际上 demo 设计 32 KB 共享数据区(cryptoBlock[8192],每个 word 4 字节),刚好用满
- 真要做 TLS 握手或大文件签名,推荐用 HSMA2G 库(它内部已封装多块机制)

12.4 中断优先级

- HSM Bridge_IRQn 在 demo 里设为优先级 0(最高)
- Host 端 SRC_HSM0 优先级根据实时性需求配
- 不要让 HSM 中断被长时间屏蔽(否则 Host 不知道 HSM 已完成命令)

12.5 双核通信纪律

- 任何 HSM 命令,先检查 HSM_HSM2HTF 是否被 Host 端清掉
- Host 收到 HSM2HTF 中断后,必须先读 HSM_HSM2HTS 再清标志(否则丢响应)
- HSM 在 status & BUSY 时拒绝新命令,Host 要重试

12.6 工具链搭配

组合	适用
ADS(Eclipse + GCC)+ ARM GNU + UDE	全部免费,适合学习和小批量
HighTec TriCore + HighTec ARM + Trace32	中等成本,Infineon 官方推荐
TASKING TriCore + TASKING ARM +	高成本,大客户使用

组合	适用
iSYSTEM	

HSM 侧 必须用 ARM 工具链(因为 M3),不要尝试用 TriCore 工具链。

12.7 调试 HSM 自己的小技巧

- 看桥接寄存器:在调试器里看 0xF0040000 起 0x40 字节,就是 HSM 状态镜像
- 看 AES 状态:HSM 端调 HSM_AES->AESSTAT.B.BSY 轮询,debug 时直接读 0xE8000000 + AESSTAT_OFFSET(具体偏移见 hsm20_app/CMSIS/Include/component/aes.h)
- 看 SAHBASE:调试 HSM 访问 Host 内存有没有走对窗口
- LAUTERBACH:Trace32 有专门的 HSM 调试脚本,HSM_demo/AP32373/.../Lauterbach/ 下有样例
- 触发调试器异常:demo 在所有 ISR 错误处理里都写 HSM_BRIDGE->HSM2HTS = 0xdeadxxxx,调试时看到这个值就知道哪个 ISR 进了

CH 13

第 13 章

HSM Demo 工程速查表

Demo 目录	应用笔记	关键内容	学习优先级
AP32391_Secure_Boot_with_HSM_v1.3.3_SW	AP32391	完整 Secure Boot:HSM 校验 TriCore 代码	⭑ ⭑ ⭑ ⭑ ⭑ 必读
AP32373_HSM_Demo_examples_for_TC3xx_V3.0.1_SW	AP32373	17 个 demo:Crypto、HSM 启动、Key Loading、Debug 等	⭑ ⭑ ⭑ ⭑ ⭑ 必读
AP32562-Tutorial_CnR_Example_on_HSM_can_handle_debug_protection_v0.5.6	AP32562	Challenge-Response 模式 debug 保护	⭑ ⭑ ⭑ ⭑
AP32404-Tutorial_AURIX_TC3xx_Using_SWAP_mechanism_for_software_updates_over_the_air_v1.3.0_SW	AP32404	SWAP/OTA 升级	⭑ ⭑ ⭑ ⭑
AN_HSMDemo/HSM_BootBasic	经典	HSM 最简引导 (code_hsm.c 95 行)	⭑ ⭑ ⭑ ⭑
AN_HSMDemo/HSM_Demo_TC39xA	经典	旧版 TC39xA demo	⭑ ⭑
AURIX_HSM Revealed_Training_Demo_1.0	培训	配套幻灯片的代码	⭑ ⭑ ⭑

推荐学习路径

第 1 步(1~2 天):通读

- White_Paper_TC3xx_Family_Security_Island_v1.2.pdf — 整体框架
- AURIX_HSM Revealed_Training_Slides.pdf 前 50 页 — 基础概念
- AN_HSMDemo/HSM_BootBasic/src/code_hsm.c — HSM 最小代码

第 2 步(3~5 天):读 Secure Boot 完整 demo

- AP32391_Secure_Boot_with_HSM_v1.3.3_SW/README.md
- HSM_TC39xB_demo/0_Src/AppSw/Tricore/Main/Cpu0_Main.c
- HSM_TC39xB_demo/0_Src/AppSw/Tricore/App/scheduler.c
- HSM_TC39xB_demo/0_Src/AppSw/Tricore/App/msg/msg.c
- HSM20_app/src/main_app.c
- HSM20_app/src/bridge.c
- HSM20_app/src/cryptoCmac.c
- HSM20_app/src/cache.c
- HSM_TC39xB_demo/0_Src/AppSw/Tricore/Cfg_Ssw/Ifx_Cfg_SswBmhd.c
- HSM_TC39xB_demo/0_Src/AppSw/Tricore/Cfg_Ssw/Ifx_Cfg_SswUcbHsm.c

第 3 步(1 周):上手改 demo

- 改一改 CMAC 计算的输入数据
- 加一个自定义 HSM 命令
- 试着接 Lauterbach 调试
- 把 UCB 烧进去,看 UCB_HSMCOTP0 怎么生效

第 4 步(2 周):进阶

- Debug 保护(CnR 模式)
- SWAP / OTA
- HSMA2G 商业库使用(申请授权)
- 评估 EVITA-FULL / ISO 21434 合规

CH 14

第 14 章

附录:术语表 / 寄存器地址 / 参考文档

14.1 术语表

术语	全称	含义
HSM	Hardware Security Module	AURIX TC3xx 内部 ARM Cortex-M3 子系统
BOS	Boot System	HSM 的启动固件,Infineon 固化在 HSM BootROM 0x00000000(4 KB),不可改
Boot Firmware	-	TriCore 上电后跑的第一段固化代码(在 BOOT ROM),做硬件初始化后转 SSW
SSW	Start-up Software	TriCore 启动软件,固化在 BootROM,负责 BMHD/UCB 处理
Testmode	-	BOS 路径之一,产线 wafer test 用,HT2HSMS.REQ_MODE=0x5 触发
Usermode	-	BOS 路径之一,产品启动,HT2HSMS.REQ_MODE=其他
HAR	Halt After Reset	启动后第一指令前停住,等调试器接管;HT2HSMS.HAR_ENABLE=0x5 触发
HT2HSMS	Host to HSM Status	32-bit Host→HSM(BOS)通信字,含 REQ_MODE / HAR_ENABLE / OS_ENTRY_ALLOWED
HSM2HTS	HSM to Host Status	32-bit HSM(BOS)→Host 通信字,

术语	全称	含义
		含 ACT_MODE / STATUS_CODE / TM_CMD_FINISHED
UCB	User Configuration Block	DFlash 里的用户配置区(约 25 个命名 UCB,每个含 ORIG+COPY,共 50 份)
BMHD	Boot Mode Header	启动模式配置,共 4 个
BMI	Boot Mode Index	启动模式索引(0xFE=Internal Flash)
BOOTSEL0/1/2/3	-	DMU_SP_PROCONHSMCBS 里的 4 个 6-bit 字段,指 BOS 找 User-OS sector 的优先级
HSM Config Area	-	0xAF400E00–0xAF400FFF(512 B),BOS 用 mask-individual key 解密 + AES-CBC-MAC 校验
HSMCFG	HSM Configuration	UCB_HSMCOTP0 里的 HSM 配置位
HSMBOOTEN	HSM Boot Enable	是否启用 HSM 启动
SSWWAIT	SSW Wait for HSM	SSW 是否等 HSM 释放 Host
HSMDX	HSM Data eXclusive	HSM 数据段是否 HSM 独占
HSMRAMKEEP	HSM RAM Keep	HSM RAM 是否跨复位保留
HSMDBGDIS / DBGIFLCK	-	DMU_SP_PROCONHSM 里的 HSM 调试锁定位;BOS 读它们去配 DBGCTRL
DESTDBG	Destructive Debug	Debug 进入时是否擦 HSM 私钥
SHE	Secure Hardware Extension	HIS 早期车规安全规范,AES-128 + key slot 体系
EVITA	E-safety Vehicle Intrusion protected Applications	EU 项目,车规 HSM 完整规范
ISO 21434	-	2021 起的车规网络安全强制标准

术语	全称	含义
UN R155	-	UNECE 网络安全管理体系强制法规
PKC	Public Key Crypto	HSM 内部 ECC 硬件(TS 没 RSA,只有 ECC P-256 等 + Curve25519/Ed25519)
TRNG	True Random Number Generator	真随机数生成器(BSI AIS-20/31 PTG.2)
HSMA2G	HSM AURIX 2G 库	Infineon 商业安全库(也叫 CycurHSM)
BRG / Bridge	-	Host↔HSM 桥接器,基址 0xF0040000
SAHMEM / SAHBASE	Single Access to Host Memory	HSM 访问 Host 内存的 64 KB 窗口
DBGMEM	Debug Memory window	调试器访问 HSM 内部资源的 64 KB 窗口,地址上 = SAHMEM(0xF0050000)
EXTIF / EXTIE	External Interrupt Flag / Enable	32 个外部中断源
SENSIF / SENSIE	Sensor Interrupt Flag / Enable	10 个传感器(电压/温度/时钟异常)
FPB / DWT	Flash Patch & Breakpoint / Data Watchpoint & Trace	HSM debug 工具,6+4 trigger
DBGCTRL	Debug Control	HSM-side 控制 HSM/Host 调试开放
HSMCTRL.EOMBT	End of MBIST Test	Host 释放 HSM 复位的开关,写 11b

14.2 关键寄存器地址速查(完整)

```
[HSM 内存映射(HSM 视角,TS Table 2-1)]
HSM Boot ROM              0x00000000 (4 KB)
HSM Local RAM             0x20000000 (96 KB, 0x20017FFF)
HSM Bit-banding Alias     0x22000000 (32 MB → 0x20000000 范围)
```

```
HSM PFlash(代码+数据)     0x80000000 (16 MB, 也可从 0xA0000000 非缓存访问)
UCB00-UCB06               0xAF400000 - 0xAF400DFF
UCB07 + HSM Config Area   0xAF400E00 - 0xAF400FFF (512 B)
UCB08-UCB47               0xAF401000 - 0xAF405FFF
HSM Data Flash 1          0xAFC00000 (128 KB, 0xAFC1FFFF)
HSM 内核 NVIC / SCB       0xE000E000
HSM DWT                   0xE0001000 (4 KB)
HSM FPB                   0xE0002000 (4 KB)
HSM Cache Control         0xE008A000 (8 KB)
HSM ROM Table             0xE00FF000 (4 KB)
HSM Bridge 基址           0xF0040000

[HSM Bridge 寄存器块(0xF0040000 起的 SFR 偏移,TS Table 9-4)]
HSM_ID                    0xF0040008
HT2HSMF                   0xF0040020    rr/rs  Host→HSM Flag
HT2HSMIE                  0xF0040024    rw/-   HSM 中断使能
HSM2HTF                   0xF0040028    rs/rr  HSM→Host Flag
HSM2HTIE                  0xF004002C    ro/rw  Host 中断使能
HSM2HTIS                  0xF0040030    ro/rw  Host 中断路由选择
HSM2HTS                   0xF0040034    rw/ro  HSM→Host Status
HT2HSMS                   0xF0040038    ro/rw  Host→HSM Status
CLKCTRL                   0xF0040040    rw/-   时钟分频
DBGCTRL                   0xF0040060    rw/-   Debug 控制
PINCTRL                   0xF0040064    rw/-   2 个 pin 控制
ERRCTRL                   0xF0040080    rw/-   错误标志
ERRIE                     0xF0040084    rw/-   错误中断使能
ERRADDR                   0xF0040088    ro/-   错误地址
EXTIF                     0xF00400A0    rw/-   外部中断标志
EXTIE                     0xF00400A4    rw/-   外部中断使能
SAHBASE                   0xF00400C0    rw/-   HSM 访问 Host 内存基址
RSTCTRL                   0xF00400E0    rw/-   复位控制
RSTPWD                    0xF00400E4    w/-    复位密码
SENSIF                    0xF00400F0    rw/-   传感器标志
SENSIE                    0xF00400F4    rw/-   传感器中断使能
SENSAPPRST                0xF00400F8    rw/-   传感器应用复位
SENSSYSRST                0xF00400FC    rw/-   传感器系统复位
HSMCTRL                   0xF0041000    -/rs   Host 释放 HSM 复位(EOMBT=11b)
DBGBASE                   0xF0041010    -/rw   调试窗口基址
HSMOTGB                   0xF0041020    -/rw   OCDS trigger
DBGMEM                    0xF0050000 (64 KB,调试窗口)

[HSM 端外设]
HSM AES                   0xE8000000
HSM HASH                  0xE8000400
HSM PKC                   0xE8003C00
HSM Cache                 0xE008A000
HSM Timer                 0xEC000000
HSM WDT                   0xEC000100
HSM TRNG                  0xEC000200
HSM NVIC                  0xE000E000 (ARM 标准)

[Host 端 HSM 共享内存地址]
HSM 控制/参数区           0xB0040000 (0x80 字节,放 cmd/参数)
HSM 数据区                0xB0040080 (32 KB,放输入/输出数据)
```

14.3 本工程(workspace)内文档清单

文件	类型	用途

文件	类型	用途
White_Paper_TC3xx_Family_Security_Island_v1.2.pdf	白皮书	整体架构,新人第一份必读
AURIX_HSM Revealed_Training_Slides.pdf	培训幻灯片	配合 HsmMain demo,系统化教学
AP32349_HSM+_StartupV2.1.0.pdf	AN	HSM 启动完整流程
AP32373_HSM_Demo_examples_for_TC3xx_V3.0.1.pdf	AN	17 个 demo 总览
AP32391_Secure_Boot_with_HSM_v1.3.0.pdf	AN	Secure Boot 详细步骤
AP32399_AURIX_TC3xx_Debug_protection_with_HSM_v1.3.0.pdf	AN	Debug 保护完整说明
AP32404-Tutorial_AURIX_TC3xx_Using_SWAP_mechanism_for_softwa.pdf	AN	SWAP/OTA 教程
AP32456-Tutorial_AURIX_TC3xx_HSM_Cache_Usage_v1.0.4.pdf	AN	HSM Cache 使用细节
AP32470_HSM_Demo_Optimization_Code__V1.0.pdf	AN	HSM 性能优化技巧
AP32481-Tutorial_Debugger_Protection_Differences_in_AURIX_TC.pdf	AN	Debug 保护差异(TC2xx vs TC3xx)
AP32489-Tutorial_HSM_changes_in_AURIX_TC3xx_compared_to_AURI.pdf	AN	TC2xx → TC3xx HSM 迁移
AP32543-	AN	HSM 硬件支持细节

文件	类型	用途
Tutorial_How_AURIX_TC3xx_HSM_hardware_supports_secur.pdf		
AP32555-Tutorial_AURIX_TC3xx_Avoiding_device_lockups_v1.0.1.pdf	AN	避免死锁(必读)
AP32562-Tutorial_CnR_Example_on_HSM_can_handle_debug_protect.pdf	AN	CnR 模式 demo
AP32574-Tutorial_HSM_Performance_Figures_Summary_v1.0.1.pdf	AN	性能数据汇总

14.4 外部推荐资源

- Infineon 官网 infineon.com → AURIX → TC3xx → Documentation
- AURIX Development Studio (ADS): 免费 IDE,基于 Eclipse + GCC
- iSYSTEM winIDEA / Lauterbach Trace32 / PLS UDE: 商业调试器
- ETAS CycurHSM:商业安全中间件,基于 HSMA2G 库封装
- Vector MICROSAR Safe:Vector 的 AUTOSAR + Security 套件
- HSM demo GitHub:github.com/Infineon/AURIX_code_examples

14.5 网上优质博客(中文,适合新人)

- 沙与沫:英飞凌 TC3XX 单片机 HSM 内核的 Secure Boot 实现(腾讯云开发者)
- 美好生活:英飞凌 TC3XX 单片机 HSM 内核开发系列(CSDN)
- 代码收藏家:英飞凌 AURIX 系列 HSM 详解(物联沃)
- 鉴源实验室:HSM 技术浅述(CSDN)

结语

HSM 是 AURIX TC3xx 最值钱、也最容易让人懵的子系统。新人入门建议:先把 demo 跑通,把 HSM 当成一个"独立小黑盒"来用,而不是要彻底搞懂每个寄存器。等业务用上 Secure Boot / Debug 保护 / OTA 时,再回头精读对应 AN + 源码,事半功倍。

遇到问题先去 AP32391、AP32373 这两个最完整的 demo 里找答案,90% 的问题别人都踩过。

Good luck,新兵。