应用笔记

Application Note

文档编号：**AN1139**

G32R501 安全启动应用笔记

版本：**V1.0**

© 珠海极海半导体有限公司

1 引言

本应用笔记旨在介绍 G32R501 芯片安全启动的相关内容，包括安全启动应用程序的设计原则，以及如何使用 G32R5xx SDK 所提供的安全启动例程。

目录

1 引言 1

2 G32R5xx 安全启动概述 3

3 安全启动选项 4

4 如何设计安全启动的应用程序 5

4.1 Golden CMAC Tag 存储位置 5

4.2 待校验应用程序加密扇区设置的要求 5

4.3 以 MDK-ARM 环境为例，应用程序设计示例 5

5 G32R5xx 安全启动功能如何使用 10

5.1 安全启动的操作流程 10

5.2 Geehy_Bin 工具使用介绍 10

5.3 G32R5xx SDK 安全启动例程使用 12

6 版本历史 16

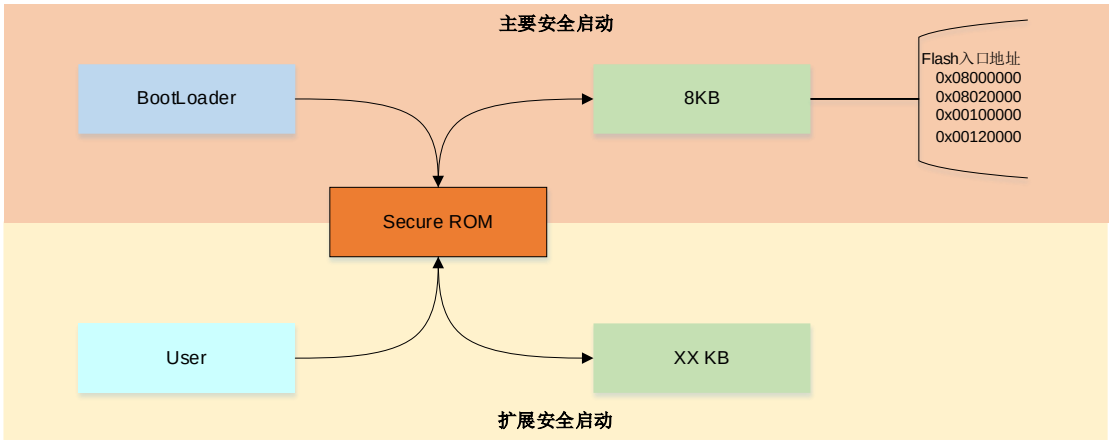
2 G32R5xx 安全启动概述

G32R5xx 的安全启动功能由芯片固化的 BootROM 代码实现，采用 128 位 AES-CMAC 身份验证算法，对存储在 Flash 中的用户固件的前 8KB 数据进行身份验证。只有在验证通过后，才会跳转执行用户应用程序。

由 BootROM 发起的针对用户固件前 8KB 进行的身份验证被称为“主要安全启动”。相应地，针对超出 8KB 范围的身份验证则称为“扩展安全启动”，该过程由用户应用程序代码选择性地发起。

其中，主要安全启动校验的所在 Flash 扇区，必须设置为 Zone1-ExeOnly 权限。扩展安全启动校验的所在 Flash 扇区，则可配置为 Unsecure、Zone1-Secure、Zone1-ExeOnly 三种权限均可。也就是说，进行 CMAC 身份验证的应用程序所在的 Flash 扇区，其加密权限不允许配置为 Zone2 区域的权限。

图 1 主要安全启动和扩展安全启动校验示意图



3 安全启动选项

对于 G32R5xx 的安全启动功能，仅支持从 CPU0 启动。表格 1 列出了安全启动的所有入口地址及其对应的 BootMode 值，以及在不同入口地址下 Golden CMAC Tag 必须存放的位置。

表格 1 安全启动选项配置表

Option	BootMode 值	Flash 入口点	Flash 块/扇区	Golden CMAC Tag 存储位置
0	0x0A	0x08000000	Bank 0, Sector 0(Busmatrix IF)	0x08000008
1	0x2A	0x00100000	Bank 0, Sector 0(ITCM IF)	0x00100008
2	0x4A	0x08020000	Bank 1, Sector 0(Busmatrix IF,DualBank Mode	0x08020008
3	0x6A	0x00120000	Bank 1, Sector 0(ITCM IF,BualBank Mode)	0x00120008

注意：安全启动对应的入口扇区，必须配置为 Zone1-ExeOnly 权限。

4 如何设计安全启动的应用程序

对于需要进行安全启动的应用程序，必须满足本章所描述的要求，以确保对应用程序进行正确的校验，从而保证其正常运行。

4.1 Golden CMAC Tag 存储位置

Golden CMAC Tag 值是用户存储在固件中的，用于进行 CMAC 身份验证的期望 Tag 值，其大小为 16 字节。对于主要安全启动，Golden CMAC Tag 的存储的起始地址必须位于安全启动对应入口地址之后的 8 个字节处。例如，如果安全启动校验的入口地址为 0x08000000，则 Golden CMAC Tag 的存储的起始地址应为 0x08000008。

对于用户希望自定义校验 Flash 扇区区间的扩展安全启动，Golden CMAC Tag 值存储的起始地址没有限制，但必须满足以下要求：

（1） 扩展安全启动的 Golden CMAC Tag 值不得位于主要安全启动校验范围内。

（2） 扩展安全启动的 Golden CMAC Tag 值存储的起始地址必须为 4 字节对齐。

（3） 扩展安全启动的 Golden CMAC Tag 值存储区域必须位于进行 CMAC 校验的范围内。例如，如果校验的是整个 Flash 的大小，则 Golden CMAC Tag 必须存储在 Flash 范围内。

4.2 待校验应用程序加密扇区设置的要求

当需要进行 CMAC 身份验证的程序所在的 Flash 扇区被设置为加密权限后，该区域只能被 CPU 执行程序，而不允许 CPU 读取该 Flash 扇区的数据。因此，如果需要 CPU 读取的数据存放在 Flash 中，则对应的 Flash 扇区不得设置为加密权限。

通常，CPU 需要从 Flash 读取的数据包括只读数据段（例如，使用 const 修饰的全局变量），以及具有非零初始值的全局变量的初始值，这些变量的初始值会存储在 Flash 区域，并在 MCU 启动时复制到 RAM 区域。因此，不允许设置为加密权限的代码区域包括：

（1） CONST（RO-DATA）：只读数据区域。

（2） RW：可读写数据段。具有非零初始值的全局变量在编译时会被存放到代码中并与其他数据一起存储，电源开启后会从 Flash 复制初始值到 RAM。

由此可知，唯一允许被加密的所存放在 Flash 的段是 text 段，即代码段。其他数据段所在的 Flash 扇区均不允许被加密。

4.3 以 MDK-ARM 环境为例，应用程序设计示例

根据前面 4.1 和 4.2 小节的介绍，本小节以 MDK-ARM 环境为例，阐述如何设计符合安全启动要求的应用程序。

4.3.1 Golden CMAC Tag 值存储位置设计示例

前面 4.1 小节介绍了 Golden CMAC Tag 值存储位置的要求，下面介绍的是在代码中如何去实现这些要求。

1. 主要安全启动 Golden CMAC Tag 值存储位置设计：

由于 ARM Cortex-M 架构的规定，起始地址的前 8 个字节必须存放 SP 和 PC 寄存器的值，因此 Golden CMAC Tag 值被规定存储在 SP 和 PC 寄存器位置之后，也就是入口地址偏移 8 个字节的后面。

此外，主要安全启动入口地址所在的扇区必须配置为 Zone1-ExeOnly 权限，为了确保中断向量表不被加密，其余的中断向量不能存放在入口扇区。

基于以上两点原因，设计了两个中断向量表的解决方案。第一个中断向量表用于存放 SP 和 PC 寄存器的值，以及 Golden CMAC Tag 值；第二个中断向量表则用于存放其余所有中断向量。

设计存放 Golden CMAC Tag 值的中断向量表，其代码示例如下：

```
const PINT __VECTOR_TABLE_CMAC[6] __VECTOR_TABLE_ATTRIBUTE = {
    [0]   = (PINT)(&__INITIAL_SP),    // Initial Stack Pointer
    [1]   = Reset_Handler,           // Reset Handler (Code Start)
    [2]   = (PINT)0xFFFFFFFF,        // CMAC Tag 0
    [3]   = (PINT)0xFFFFFFFF,        // CMAC Tag 1
    [4]   = (PINT)0xFFFFFFFF,        // CMAC Tag 2
    [5]   = (PINT)0xFFFFFFFF,        // CMAC Tag 3

};
```

该中断向量表必须存放到启动地址处。

另一个中断向量表则存放所有的中断向量，其存放位置不做要求，但是该向量表所在的扇区不允许设置为加密权限。其代码示例如下（节选）：

```
const PINT __VECTOR_TABLE[512] = {
    [0]   = (PINT)(&__INITIAL_SP),    // Initial Stack Pointer
    [1]   = Reset_Handler,            // Reset Handler (Code Start)
    [2]   = NMI_Handler,              // -14 NMI Handler
    [3]   = HardFault_Handler,        // -13 Hard Fault Handler
    [4]   = MemManage_Handler,        // -12 MPU Fault Handler
    [5]   = BusFault_Handler,         // -11 Bus Fault Handler
    [6]   = UsageFault_Handler,       // -10 Usage Fault Handler
    [7]   = SecureFault_Handler,      //  -9 Secure Fault Handler
    [11]  = SVC_Handler,              //  -5 SVCall Handler
    [12]  = DebugMon_Handler,         //  -4 Debug Monitor Handler
    [14]  = PendSV_Handler,           //  -2 PendSV Handler
    [15]  = SysTick_Handler,          //  -1 System Tick Handler

    //
    // Interrupts
    //
    [27]  = DCCOMP_Handler,           //  11 DCCOMP Handler
    [29]  = TIMER1_Handler,           //  13 CPU Timer 1 Handler
    [30]  = TIMER2_Handler,           //  14 CPU Timer 2 Handler
    [32]  = RAM_Handler,              //  16 RAM Handler
    [33]  = IPC_TE0_Handler,          //  17 IPC TE0 Handler
    [34]  = IPC_TE1_Handler,          //  18 IPC TE1 Handler
    [35]  = IPC_TE2_Handler,          //  19 IPC TE2 Handler
    [36]  = IPC_TE3_Handler,          //  20 IPC TE3 Handler
    [37]  = IPC_RF0_Handler,          //  21 IPC RF0 Handler
    [38]  = IPC_RF1_Handler,          //  22 IPC RF1 Handler

    ......
};
```

此处未展示全部的中断向量，详细代码可参考 G32R5xx SDK 对应的示例程序。

2. 扩展安全启动 Golden CMAC Tag 值存储位置设计：

用户可以对超出 8KB 的数据进行身份验证，用户自定义发起的身份验证属于扩展安全启动。根据前面所述的扩展安全启动 Golden CMAC Tag 值存储位置要求，以下是相应的代码示例：

```
const uint32_t cmac_all[] \
__attribute__((__used__, section(".ARM.__at_0x08008000"))) =
{
    0xFFFFFFFF,
    0xFFFFFFFF,
    0xFFFFFFFF,
    0xFFFFFFFF,

};
```

该数组旨在确保在编译生成的 bin 文件中占据一个特定位置，以防编译器把用于存放 Golden CMAC Tag 值的空间用于存放其他数据。其中，0x08008000 地址为 Golden CMAC Tag 的起始存放地址。

4.3.2 链接文件设计示例

4.2 小节介绍：编译出来的 bin 除了代码段可以被设置为加密权限外，其他的段均不允许设置为加密权限。因此，安全启动的应用程序，其链接文件必须重新设计，把不允许加密的段区域集中存放在固定的 Flash 扇区。

注意：请勿将不允许加密的段区域所在的 Flash 扇区设置为加密权限。

本小结的 sct 文件的示例，LR_SECURE_ROM 区域只存放了代码段以及第一个中断向量表数组，该区域所在的 Flash 扇区可以被设置为加密权限。LR_UNSECURE_ROM 所在的区域，则存放有只读数据段、__main 函数、RW 数据段等内容，该区域所在的 Flash 扇区是不允许被加密，如果该区域加密会导致程序无法正常执行。

以下是以 MDK-ARM 的 sct 链接文件为示例，其链接文件设计示例如下（节选）：

```
// This area allows users to be set with encryption permissions.
LR_SECURE_ROM __RO_BASE __RO_SIZE {
  ER_SECURE_ROM __RO_BASE __RO_SIZE {
   *.o (RESET, +First)
   .ANY (+RO)
   .ANY (+XO)
  }
}

// This area does not allow users to be set with encryption
permissions.
LR_UNSECURE_ROM __CPU0_UNSECURE_ROM_BASE __CPU0_UNSECURE_ROM_SIZE {
  ER_UNSECURE_ROM __CPU0_UNSECURE_ROM_BASE __CPU0_UNSECURE_ROM_SIZE {
    *(InRoot$$Sections)
    .ANY (+CONST)
  }

  RW_RAM_CPU0_ITCM __CPU0_ITCM_BASE __CPU0_ITCM_SIZE {
    .ANY (itcm.instruction)
    .ANY (itcm.ramfunc)
    .ANY (+RW +ZI)
  }

#if __HEAP_SIZE > 0
  ARM_LIB_HEAP __HEAP_BASE EMPTY __HEAP_SIZE {   ; Reserve empty
region for heap
  }
#endif

  ARM_LIB_STACK __STACK_TOP EMPTY - __STACK_SIZE {   ; Reserve empty
region for stack
  }

}
```

文件全部内容请参考 G32R5xx SDK 对应的安全启动例程的 sct 文件。

5 G32R5xx 安全启动功能如何使用

5.1 安全启动的操作流程

要实现安全启动，在设计应用程序时，除了遵循章节 4 所介绍的设计原则外，还需按照以下操作流程进行。

（1） 配置安全启动的起始扇区权限为 Zone1 ExeOnly 权限。

（2） 配置芯片的 CMACKEY 值。芯片的 CMACKEY 值存储在 OTP 区域，地址为 0x0810B020，长度为 128 字节。芯片出厂时，CMACKEY 值默认为全 FF。

（3） 对用户编译生成的应用程序的原始 bin 文件，根据用户设置的 CMACKEY 值，使用 Geehy_Bin 工具生成带有 Golden CMAC Tag 值的 bin 文件。

（4） 把带有 Golden CMAC Tag 值的 bin 文件烧录下载到芯片。

（5） 根据表格 1 的安全启动选项配置表，设置所需要的 BootMode 值（在调试阶段，建议使用仿真 BootMode 值设置为安全启动模式）。

（6） 按复位键或者上电复位，执行应用程序。

5.2 Geehy_Bin 工具使用介绍

用户在编译应用程序生成原始 bin 文件后，可以使用 Geehy_Bin 工具生成带有 Golden CMAC Tag 值的 bin 文件，以便用于安全启动。

本小节只介绍如何使用 Geehy_Bin 工具生成带有 Golden CMAC Tag 值的 bin 文件，更多有关该工具的使用信息，请参阅《AN1131_G32R501 Tool 用户手册》文档。

5.2.1 key.txt 和 cmd 文件

使用 Geehy_Bin 工具，生成带有 Golden CMAC Tag 值的 bin 文件，需要用户提供带有 CMACKEY 值的文本文件，以及主要安全启动或者扩展安全启动的校验长度信息。在 G32R5xx SDK 中，在 utilities/geehy_tool 目录下已经提供了这些文件的示例，有 key.txt、test_major.cmd、test_extend.cmd 三个文件。

- key.txt 文件。

该文件用于指定用户设置的 CMACKEY 值，示例内容如下，为芯片默认的 CMACKEY 值。

```
0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
```

注意：若修改了芯片的 CMACKEY 值，该文件的内容必须和用户设置的芯片内部的 CMACKEY 值对应上。

- test_major.cmd 文件

test_major.cmd 文件用于指定主要安全启动的入口地址，和校验长度，不过主要安全启动的校验长度是固定的 0x2000。test_major.cmd 文件的示例内容如下：

```
/* CPU1 Flash sectors */

/* HEX directive required by HEX utility to generate the golden
CMAC tag */
/* with one entry that represents all the allocated flash memory */
ROMS
{
  FLASH: o=0x08000000 l=0x2000, fill = 0xFFFF /* If fill not
specified, then default is all 0s */
}
```

其中，o=0x08000000 是用于指定主要安全启动校验的入口地址，该值必须与用户设置的 BootMode 值所对应的入口地址一致。l=0x2000 则用于指定主要安全启动的校验长度，需要注意的是，主要安全启动校验长度都是固定的 0x2000。

- test_ extend.cmd 文件

test_extend.cmd 文件用于指定用户自定义校验的起始地址和校验长度，文件示例内容如下：

```
/* CPU1 Flash sectors */

/* HEX directive required by HEX utility to generate the golden
CMAC tag */
/* with one entry that represents all the allocated flash memory */
ROMS
{
  FLASH: o=0x08000000 l=0x10000, fill = 0xFFFF /* If fill not
specified, then default is all 0s */
}
```

其中，o=0x08000000 是用于指定扩展安全启动（即用户自定义校验）的起始地址，l=0x10000 则用于指定扩展安全启动的校验长度，对于扩展安全启动指定的校验起始地址和校验长度，必须和应用程序设置的校验起始地址和长度一一对应。另外，扩展安全启动的校验范围是可以包含主要安全启动的 0x2000 的校验范围。

5.2.2 Geehy_Bin 工具的命令使用

- 生成主要安全启动的 bin 文件命令

Geehy_Bin.exe test_major.cmd -m --load_image --cmac=key.txt project.bin -

tag=0x08000008 -o project_cmac_major.bin

其中，-tag=0x08000008 用于指定运算生成的 Golden CMAC Tag 值存放的起始地址，对于主要安全启动，Golden CMAC Tag 值存放的起始地址都是入口地址偏移 8 字节的位置。

- 生成扩展安全启动的 bin 文件命令

  Geehy_Bin.exe test_extend.cmd -e --load_image --cmac=key.txt project_cmac_major.bin -tag=0x08008000 -load=0x08000000 -o project_cmac_extend.bin

  对于扩展安全启动，用户使用-tag 参数所指定存放 Golden CMAC Tag 值的起始地址，必须要和应用程序设置存放该值的地址一一对应。另外，其存储的位置不允许存放在主要安全启动的校验范围内。

扩展安全启动由用户选择性发起，如果同时存在主要安全启动和扩展安全启动，用户必须首先生成主要安全启动的 bin 文件，然后再使用该 bin 文件生成同时存在主要和扩展安全启动的 bin 文件。

5.3 G32R5xx SDK 安全启动例程使用

G32R5xx SDK 已经提供了一个安全启动的示例应用程序。该安全启动应用程序，包含主要安全启动 8KB 的校验示例，以及对超出 8KB 范围的代码如何进行校验的示例。

该示例程序位于：driverlib\g32r501\examples\eval\boot\boot_ex3_cpu0_secure_flash。

该示例假设进行身份验证的 Flash 扇区已经正确配置了加密权限（入口地址扇区必须设置为 Zone1 EXEONLY 权限），然后使用的是默认的 CMACKEY 进行身份验证。对于如何设置 Flash 扇区权限以及修改 CMACKEY，请参考 DCS 模块相关例程和介绍。

下面以 MDK-ARM 环境为例，介绍如何运行该示例程序。

5.3.1 生成带有 Golden CMAC Tag 值的 bin 文件

极海官方提供了工具，用于生成带有 Golden CMAC Tag 值的 bin 文件，下面介绍如何使用 Geehy 工具生成该 bin 文件。

（1） 由 hex 文件生成不带 Golden CMAC Tag 值的 bin 文件

由于安全启动的应用程序，其 sct 文件有多个加载域，使用 MDK-ARM 自带生成 bin 文件的工具无法生成一个完成的 bin 文件，而是分散的 bin 文件。所以借助 hex2bin 工具生成 bin 文件。如图 2 所示：

图 2 生成不带 Golden CMAC Tag 值的 bin 文件

```
D:\geehy_bin>hex2bin.exe project.hex
hex2bin v2.5, Copyright (C) 2017 Jacques Pelletier & contributors

Allocate_Memory_and_Rewind:
Lowest address:   08000000
Highest address:  080417FF
Starting address: 08000000
Max Length:       268288

Binary file start = 08000000
Records start     = 08000000
Highest address   = 080417FF
Pad Byte          = FF
```

（2） 使用 Geehy 工具生成带有 Golden CMAC Tag 值的 bin 文件

首先需要生成主要安全启动的 Golden CMAC Tag 值的 bin 文件，命令如下：

Geehy_Bin.exe test_major.cmd -m --load_image --cmac=key.txt project.bin -tag=0x08000008 -o project_cmac_major.bin

图 3 生成带有 Golden CMAC Tag 值的 bin 文件

```
D:\geehy_bin>Geehy_Bin.exe test_major.cmd -m --load_image --cmac=key.txt
project.bin -tag=0x08000008 -o project_cmac_major.bin
Tag值:0B5A988D69B5281E98180B5D2C528883
D:\geehy_bin\project_cmac_major.bin generated successfully!
```

最后使用生成的带有主要安全启动的 Golden CMAC Tag 值的 bin 文件，去生成扩展安全启动的 Golden CMAC Tag 值的 bin 文件。命令如下：

Geehy_Bin.exe test_extend.cmd -e --load_image --cmac=key.txt project_cmac_major.bin -tag=0x08008000 -load=0x08000000 -o project_cmac_extend.bin

图 4 生成扩展安全启动的 Golden CMAC Tag 值的 bin 文件

```
D:\geehy_bin>Geehy_Bin.exe test_extend.cmd -e --load_image --cmac=
key.txt project_cmac_major.bin -tag=0x08008000 -load=0x08000000 -o
 project_cmac_extend.bin
Tag值:B185E8DCC10D4804261C2993ADA7A8A0
D:\geehy_bin\project_cmac_extend.bin generated successfully!
```

必须先去生成带有主要安全启动的 Golden CMAC Tag 的 bin 文件，然后再使用该 bin 文件生成带有扩展安全启动的 Golden CMAC Tag 的 bin 文件。

注：本小节所使用的工具和对应的文件可以从 G32R5xx SDK 中获取。

5.3.2 修改启动模式为安全启动

芯片出厂的默认启动方式是没有安全启动的，需要用户自己去配置启动方式为安全启动模式。用户配置 Boot 模式，有修改 OTP 的方式，和模拟启动方式。修改 OTP 的方式只能修改一次，在调试阶段建议使用模拟启动方式修改 Boot 模式。下面介绍如何使用模拟启动的方式修改启动模式为安全启动。

（1） 首先使用支持 Jlink 调试器连接到开发板上。

（2） 使用 Jlink Commander 命令行工具去连接芯片，连接成功后如图 5 所示。

图 5 Jlink Commander 成功连接 G32R501 窗口

```
Connecting to target via SWD
Found SW-DP with ID 0x6BA02477
DPIDR: 0x6BA02477
CoreSight SoC-400 or earlier
Scanning AP map to find all available APs
AP[4]: Stopped AP scan as end of AP map has been reached
AP[0]: AHB-AP (IDR: 0x84770001)
AP[1]: AHB-AP (IDR: 0x84770001)
AP[2]: AHB-AP (IDR: 0x84770001)
AP[3]: APB-AP (IDR: 0x54770002)
Iterating through AP map to find AHB-AP to use
AP[0]: Core found
AP[0]: AHB-AP ROM base: 0xE00FF000
CPUID register: 0x630FD241. Implementer code: 0x63 (ARM China)
Feature set: Mainline
Cache: L1 I/D-cache present
Found Cortex-M52 r0p1, Little endian.
FPUnit: 8 code (BP) slots and 0 literal slots
Security extension: not implemented
CoreSight components:
ROMTbl[0] @ E00FF000
[0][0]: E000E000 CID B105900D PID 000F5D24 DEVARCH 47702A04 DEVTYPE 00 ???
[0][1]: E0001000 CID B105900D PID 000F5D24 DEVARCH 47711A02 DEVTYPE 00 DWT
[0][2]: E0002000 CID B105900D PID 000F5D24 DEVARCH 47701A03 DEVTYPE 00 FPB
[0][3]: E0000000 CID B105900D PID 000F5D24 DEVARCH 47701A01 DEVTYPE 43 ITM
[0][5]: E0041000 CID B105900D PID 000F5D24 DEVARCH 47754A13 DEVTYPE 13 ETM
[0][6]: E0003000 CID B105900D PID 000F5D24 DEVARCH 47700A06 DEVTYPE 16 ???
[0][7]: E0042000 CID B105900D PID 000F5D24 DEVARCH 47701A14 DEVTYPE 14 CSS600-CTI
[0][8]: E0046000 CID B105900D PID 001BB9BA DEVARCH 47710A55 DEVTYPE 55 ???
I-Cache L1: 4 KB, 64 Sets, 32 Bytes/Line, 2-Way
D-Cache L1: 4 KB, 32 Sets, 32 Bytes/Line, 4-Way
Memory zones:
  Zone: "Default" Description: Default access mode
Cortex-M52 identified.
J-Link>
J-Link>_
```

注意：所使用的 Jlink Commander 必须是已经支持了 Cortex-M52 内核的版本，本文演示所使用的版本是 V7.96h。另外，所使用的 Jlink 调试器也需要支持调试 Cortex-M52 内核才可以。

（3） 把启动方式配置为入口地址为 0x08000000 的安全启动方式，即 BootMode 值为 0x0A。在 Jlink Commander 命令行工具上输入命令：

w4 0x50020000 0x5AFFFFFF

w4 0x50020004 0xFFFFFF0A

w4 0x50020008 0xFFFFFFFF

5.3.3 下载和运行例程

我们可以使用 JFlash 工具下载前文生成的带有主要和扩展安全启动的 Golden CMAC Tag 值的 bin 文件。

注意：G32R5xx 芯片有 DCS 模块的加密保护，需要解密之后才能对 Flash 进行编程，如何解密请参考 DCS 相关模块介绍。

下载完程序之后，按下复位按键（请不要进行断电，否则前面配置的启动模式将会丢失），然后观察开发板上的 LED 指示灯的闪烁情况。根据 LED 闪烁情况判断是否校验通过，如下：

- LED2 和 LED3 关闭，则安全启动失败。
- LED2 或 LED3 闪烁，则安全启动通过，用户扩展 CMAC 校验失败。
- LED2 和 LED3 闪烁，则安全启动通过，且用户扩展 CMAC 校验通过。

6 版本历史

表格 2 文件版本历史

日期	版本	变更历史
2025.01	1.0	新建

声明

本手册由珠海极海半导体有限公司（以下简称“极海”）制订并发布，所列内容均受商标、著作权、软件著作权相关法律法规保护，极海保留随时更正、修改本手册的权利。使用极海产品前请仔细阅读本手册，一旦使用产品则表明您（以下称“用户”）已知悉并接受本手册的所有内容。用户必须按照相关法律法规和本手册的要求使用极海产品。

1、权利所有

本手册仅应当被用于与极海所提供的对应型号的芯片产品、软件产品搭配使用，未经极海许可，任何单位或个人均不得以任何理由或方式对本手册的全部或部分内容进行复制、抄录、修改、编辑或传播。

本手册中所列带有“®”或“™”的“极海”或“Geehy”字样或图形均为极海的商标，其他在极海产品上显示的产品或服务名称均为其各自所有者的财产。

2、无知识产权许可

极海拥有本手册所涉及的全部权利、所有权及知识产权。

极海不应因销售、分发极海产品及本手册而被视为将任何知识产权的许可或权利明示或默示地授予用户。

如果本手册中涉及任何第三方的产品、服务或知识产权，不应被视为极海授权用户使用前述第三方产品、服务或知识产权，也不应被视为极海对第三方产品、服务或知识产权提供任何形式的保证，包括但不限于任何第三方知识产权的非侵权保证，除非极海在销售订单或销售合同中另有约定。

3、版本更新

用户在下单购买极海产品时可获取相应产品的最新版的手册。

如果本手册中所述的内容与极海产品不一致的，应以极海销售订单或销售合同中的约定为准。

4、信息可靠性

本手册相关数据经极海实验室或合作的第三方测试机构批量测试获得，但本手册相关数据难免会出现校正笔误或因测试环境差异所导致的误差，因此用户应当理解，极海对本手册中可能出现的该等错误无需承担任何责任。本手册相关数据仅用于指导用户作为性能参数参照，不构成极海对任何产品性能方面的保证。

用户应根据自身需求选择合适的极海产品，并对极海产品的应用适用性进行有效验证和测试，以确认极海产品满足用户自身的需求、相应标准、安全或其它可靠性要求；若因用户未充分对极海产品进行有效验证和测试而致使用户损失的，极海不承担任何责任。

5、合规要求

用户在使用本手册及所搭配的极海产品时，应遵守当地所适用的所有法律法规。用户应了解产品可能受到产品供应商、极海、极海经销商及用户所在地等各国有关出口、再出口或其它法律的限制，用户（代表其本身、子公司及关联企业）应同意并保证遵守所有关于取得极海产品及/或技术与直接产品的出口和再出口适用法律与法规。

6、免责声明

本手册由极海“按原样”（as is）提供，在适用法律所允许的范围内，极海不提供任何形式的明示或暗示担保，包括但不限于对产品适销性和特定用途适用性的担保。

极海产品并非设计、授权或担保适合用于军事、生命保障系统、污染控制或有害物质管理系统中的关键部件，亦非设计、授权或担保适合用于在产品失效或故障时可导致人员受伤、死亡、财产或环境损害的应用。

如果产品未标明“汽车级”，则表示不适用于汽车应用。如果用户对产品的应用超出极海提供的规格、应用领域、规范，极海不承担任何责任。

用户应该确保对产品的应用符合相应标准以及功能安全、信息安全、环境标准等要求。用户对极海产品的选择和使用负全部的责任。对于用户后续在针对极海产品进行设计、使用的过程中所引起的任何纠纷，极海概不承担责任。

7、责任限制

在任何情况下，除非适用法律要求或书面同意，否则极海和/或以“按原样”形式提供本手册及产品的任何第三方均不承担损害赔偿责任，包括任何一般、特殊因使用或无法使用本手册及产品而产生的直接、间接或附带损害（包括但不限于数据丢失或数据不准确，或用户或第三方遭受的损失），这涵盖了可能导致的人身安全、财产或环境损害等情况，对于这些损害极海概不承担责任。

8、适用范围

本手册的信息用以取代本手册所有早期版本所提供的信息。

©2025 珠海极海半导体有限公司 – 保留所有权利