应用笔记

Application Note

AN1170

G32A1085 A1065 A1045 系列

FLASH 使用指南

版本：**V1.0**

1 引言

本应用笔记提供在 G32A1085 A1065 A1045 系列应用时 FLASH 的使用方法，如 FLASH 的 ECC 校验功能、FLASH 的应用流程及操作函数的具体使用流程。

目录

1 引言 1

2 芯片 FLASH 信息 3

3 ECC 功能 4

3.1 FLASH 的 ECC 功能 4

3.2 SRAM 的 ECC 功能 4

3.3 CAN SRAM 的 ECC 功能 4

3.4 ECC 注意事项 4

4 FLASH 解锁 6

4.1 FLASH 解锁 6

4.2 选项字节解锁 6

5 FLASH 擦写读 8

5.1 FLASH 擦除 8

5.2 FLASH 编程 10

5.3 SDK 关键函数说明 11

6 FLASH 保护 13

6.1 FLASH 读/写保护 13

6.2 选项字节编程 14

7 FLASH 注意事项 16

7.1 建议一 16

7.2 建议二 17

7.3 建议三 17

8 版本历史 18

2 芯片 FLASH 信息

下表罗列了 G32A1085 A1065 A1045 系列不同芯片的 FLASH 的大小信息：

表格 1 存储器说明

存储器	G32A1085 最大字节	G32A1065 最大字节	G32A1045 最大字节	说明
PFlash	256KB	128KB	64KB	存储用户代码、常量数据
DFlash	32KB	32KB	16KB	数据存储区
SRAM	32KB	16KB	8KB	—
选项字节	24Bytes	24Bytes	24Bytes	配置主存储区读写保护、MCU 工作方式

3 ECC 功能

3.1 FLASH 的 ECC 功能

PFLASH 和 DFLASH 均带有 ECC 功能，纠一检二的 ECC 算法可用于校验数据的正确性，增强数据可靠性。此功能在复位后默认开启,可以将 ECCEN 失能来关闭 FLASH 的 ECC 检测。
开启后，若检测到不可纠正的 ECC 错误， 错误标志 DBFIFLG 置起。 若使能了 DBFIEN，则可产生中断。
发生不可纠正的 ECC 错误后，最后一个出错的地址将被记录在 FLASH_ECC_ADDR 寄存器中。清除 DBFIFLG 标志位时，出错地址会被一起清零。

3.2 SRAM 的 ECC 功能

SMS 提供针对 SRAM 的 ECC 校验，包括错误检测、单比特错误纠正、以及双比特错误检测。每个地址均附加额外的校验位。通过使能 ECC_EN 启用 ECC 功能时，每次写入操作都会由 SMS 在内部计算校验位。

对于每次读取操作，包括 RMW 读取操作，其 ECC 错误都会被报告并记录下来，最后一个出错的地址将被记录在 SMS_ECC_LOGn 寄存器中。复位后 SMS_INTRMn 的中断屏蔽位保持置 1，清除对应的中断屏蔽位，被报告的 ECC 错误将能产生一个可屏蔽的系统中断。

在配置 INSERT_ERR 后，SMS 可以在 SRAM 访问时插入错误位，用于诊断目的。可以配置注入单比特错误、双比特错误和三比特错误。

3.3 CAN SRAM 的 ECC 功能

CAN SRAM 支持 ECC 校验，包括单比特错误纠正、以及双比特错误检测，校验出错可以触发中断并支持查询出错地址。

通过使能 ECC_EN 启用 CAN SRAM ECC 校验和全局 ECC 中断，在每次访问 CAN SRAM 时，将会检测存在的 ECC 错误并将其记录下来，错误标志 BEC_FLAG 或 BEU_FLAG 置起，最后一个出错的地址将被记录在 ERR_ADDR 寄存器中。在置位 BEC_INT_EN 或 BEU_INT_EN 后，产生对应的 ECC 错误将会产生一个中断。

3.4 ECC 注意事项

数据校验功能开启后，每次写入数据时，ECC 控制器会根据写入的 32 位数据生成对应的 7 位 ECC 数据，每次读取数据时会对 39 位数据进行数据校验，写入数据时不会进行校验，数据校验结果异常时会置位状态寄存器中的相关标志位，并将产生一个中断请求。

1.开启 ECC 错误中断，在中断里将访问错误的 FLASH 地址进行擦除操作，即可恢复该 FLASH 地址的访问。

2. 产生 ECC 错误后，需要将数据擦除。为了数据不丢失，可用第二个 FLASH 扇区做数据备

份，当出现 ECC 错误后，从备份区域读取。即擦写完一个扇区后，再擦写第二个 FLASH 扇区。第二个扇区只有在上电时和出现 ECC 错误才会有读取的情况，需注意区分是读取哪个 flash 扇区出现 ECC 错误，擦除对应 FLASH 扇区即可。区分方法可以在读取该地址前做一个信号标记。

3. 可以在代码上增加一个程序功能，通过程序里隐藏的串口指令或其它通信接口指令执行指令擦除程序，这样既可以保护芯片不被恶意读取程序，也可以在需要程序更新的情况下恢复芯片，解除安全状态。

4 FLASH 解锁

4.1 FLASH 解锁

复位后，FLASH 控制器 FMC 被硬件锁定，不能直接进行写入操作，必须先按照正确的顺序向 FMC_KEY 写入对应的值来解锁 FMC。KEY 值如下：

- KEY1=0x4567 0123
- KEY2=0xCDEF 89AB

错误的写入顺序或错误的值均会使程序进入硬件错误，而且此时 FMC 将被锁定，所有 FMC 操作均无效，直到下次复位才能解除。用户也可以通过向控制寄存器 2（FMC_CTRL2）的 LOCK 位写 1 使软件锁定 FMC。

用户在每次 Flash 编程操作中必须按照“Flash 解锁_用户编程_Flash 上锁”的步骤进行，以避免在 Flash 编程操作结束后，因 Flash 未上锁而带来的用户代码/数据被意外修改等风险。

在 AUTOSAR Fls 中 FLASH 的解锁操作在底层 LLD 层的硬件操作函数内，且是按需解锁（擦除/写入/选项字修改）并在操作完成后马上锁定 FMC，避免出现 FLASH 无保护状态。主函数只做任务发起和触发执行，因此在 SDK 使用中不需要对解锁进行单独操作。

AUTOSAR Fls 解锁的调用链路：主函数发起任务→Fls_MainFunction () 调度→对应 LLD 层函数→函数内部执行解锁 + 硬件操作 + 锁定。

AUTOSAR Fls 解锁锁定执行流程

主函数 Fls_Erase→Fls_MainFunction()→调用 Fls_SectorErase()
↓
【解锁在此执行】Fls_SectorErase()内部：
向 FMC->KEYR 依次写入 0x45670123、0xCDEF89AB

REG_WRITE32(FMC_KEY_REG_ADDR, FMC_KEY_1);

REG_WRITE32(FMC_KEY_REG_ADDR, FMC_KEY_2);→ FLASH 解锁
↓
选扇区→启动擦除→忙等完成→错误检测
↓
REG_BIT_SET32(FMC_CTRL2_REG_ADDR, FMC_CTRL2_LOCK_MASK); → 立即锁定

4.2 选项字节解锁

默认状态下，选项字节始终是可以读且被写保护。要想对选项字节块进行写操作（编程/擦除）首先要在 FMC_OBKEY 中写入正确的键序列（与上锁时一样），随后允许对选项字节块的写操作，FMC_CTRL2 寄存器的 OBWEN 位标示允许写，清除这位将禁止写操作。

系统复位后选项字节默认处于锁定状态，只有进行正确的解锁操作才能修改选项字节。选项字节

解锁与闪存解锁不同点在 KEY 值写入的是 FMC_OBKEY 寄存器而不是 FMC_KEY 寄存器。KEY 值如下：

- KEY1=0x4567 0123
- KEY2=0xCDEF 89AB

用户需要特别注意的是每次修改选项字节的值后，需要强制加载（置位 OBLOAD）或重新上电才能使之生效。

在 AUTOSAR Fls 中选项字节的解锁操作在底层 LLD 层的硬件操作函数内，且是按需解锁（擦除/写入/选项字修改）并在操作完成后马上锁定选项字节，避免出现选项字节无保护状态。主函数只做任务发起和触发执行，因此在 SDK 使用中不需要对解锁进行单独操作。

5 FLASH 擦写读

5.1 FLASH 擦除

G32A1085 A1065 A1045 系列芯片支持页擦和片擦（可选 PFLASH 和 DFLASH）,在每次写入前均需要进行擦除操作，如果写入地址不为全 1 则会触发一个编程错误。

页擦除流程如下：

（1） 解锁 FLASH，检查 BUSYF 标志，确保没有正在进行的 FLASH 操作

（2） 置位 PAGEERA

（3） 往 FMC_ADDR 写入目标擦除页内的任意地址

（4） 置位 STA

（5） 等待 BUSYF 拉低，对擦除地址的值进行读校验

片擦除流程如下：

片擦可将 PFLASH 或 DFLASH 或二者的全部内容进行擦除。在使用时需要特别注意，以避免误操作导致重要数据丢失。

（1） 解锁 FLASH，检查 BUSYF 标志，确保没有正在进行的 FLASH 操作

（2） 置位 MASSERA

（3） 配置 MER_TYPE 选择片擦除区域

（4） 置位 STA

（5） 等待 BUSYF 拉低，对擦除地址的值进行读校验

注意事项

FLASH 单页擦除时间为 4ms 左右，多页擦除时间为：页数*单页擦除时间。片擦除时间为 10ms 左右，PFLASH 和 DFLASH 为独立接口，配置 MER_TYPE 位同时进行 PFLASH 和 DFLASH 擦除，擦除时间共 20ms。最小编程单元双字编程时间为 215us 左右。

以下是三点注意事项：

1.上电延时：芯片上电后需延时 100ms，待系统稳定后再访问 Flash。

2.看门狗处理：多页连续擦除时，需在擦除期间喂狗（清除看门狗计数），避免因操作时间过长而触发看门狗复位。

3.供电稳定性：擦写过程中需保证供电稳定，建议增加电压监测，供电电压不稳定或出现异常时停止 Flash 操作，否则会导致 Flash 异常，并且在下次上电后访问 Flash 地址时触发 ECC 错误。

在调用 Fls_SectorErase()进行擦除操作前，调用 Fls_GetStatus()接口确保 FLASH 处于空闲状态，否则擦除函数将直接返回 ERROR。

应用代码示例：

```
VAR( Std_ReturnType, AUTOMATIC ) u8RetVal = (Std_ReturnType)E_NOT_OK;
VAR( MemIf_StatusType, AUTOMATIC ) eRetVal = MEMIF_IDLE;
VAR( MemIf_JobResultType, AUTOMATIC ) eJobRetVal = MEMIF_JOB_FAILED;
/* Set the erase operation , erase 1KB Data Flash starting from address 0x0803E000 */
u8RetVal = Fls_Erase(0x3E000, 1024);
eJobRetVal = Fls_GetJobResult();
eRetVal = Fls_GetStatus();
if((E_OK == u8RetVal) && (MEMIF_JOB_PENDING == eJobRetVal) && (MEMIF_BUSY == eRetVal))
{
    /* Erase 1KB Data Flash starting from address 0x0803E000 */
    while (MEMIF_IDLE != Fls_GetStatus())
    {
        Fls_MainFunction();
    }
}
```

AUTOSAR Fls 擦除执行流程

主函数：Fls_Erase(0x3E000, 1024) → 发起擦除请求→保存任务参数→标记 s_jobResult=MEMIF_JOB_PENDING→模块状态置为 MEMIF_BUSY

↓

主函数：进入循环（模块状态为 BUSY）

↓

循环内：反复调用 Fls_MainFunction() → 检测到待执行擦除任务→调用 LLD 层 Fls_SectorErase()执行硬件擦除

↓

硬件擦除完成→Fls_MainFunction()更新 s_jobResult=MEMIF_JOB_OK→模块状态置为 MEMIF_IDLE

↓

循环条件不满足（MEMIF_IDLE == Fls_GetStatus()）→ 退出循环→操作执行完成

5.2 FLASH 编程

G32A1085 A1065 A1045 系列对 FLASH 存储区的写操作地址必须是 64 位对齐，否则 PAEF 置起。编程长度可通过 PROGLEN 配置，最小为 64 位（8 字节）数据。

为保证写入正确，需要在写入前检查目的地址是否已经被擦除，若未被擦除，则写入数据无效并将 FMC_STS 寄存器的 PEF 位置 1。若目的地址存在写保护，则写入数据无效并触发一个写保护错误（FMC_STS 的 WPEF 位置 1）。

编程流程如下：

（1） 解锁 FLASH，检查 BUSYF 标志，确保没有正在进行的 FLASH 操作。

（2） 置位 PG。

（3） 往 FMC_ADDR 写入编程的起始地址（注意双字对齐）。

（4） 配置编程长度 PROGLEN，并将待编程的数据依次写入 FLASH_PROG_DATAx（例如，欲写入数据 0x12345678_22345678，则先往 FLASH_PROG_DATA0 写入 0x22345678，再往 FLASH_PROG_DATA1 写入 0x12345678；若想写入的数据更多，依次写入后续的 FLASH_PROG_DATA 寄存器）。

（5） 置位 STA。

（6） 等待 BUSYF 拉低，对写入地址的值进行读校验。

注意事项

由于页擦除会清空页内的所有数据，因此如果当前擦除页还有其他有效数据，则需要先用缓存将数据读出，待擦除完成后再将数据写回。写入数据时需要确保 FLASH 处于空闲状态，否则写入函数直接返回 ERROR。

应用代码示例：

```
uint32_t dataNum = 0;
uint8 g_aaAppBufferForWriteB[1024];
for(dataNum = 0;dataNum < 1024;dataNum++)
{
    g_aaAppBufferForWriteB[dataNum] = dataNum;
}
```

```
/* Set the write operation , write 1KB Data Flash starting from address 0x0803E000 */
u8RetVal = Fls_Write(0x3E000, (const uint8 *)&g_aaAppBufferForWriteB[0], 1024);
if(E_OK == u8RetVal)
{
    while (MEMIF_IDLE != Fls_GetStatus())
    {
        Fls_MainFunction();
    }
}
```

AUTOSAR Fls 编程执行流程

主函数：Fls_Write(0x3E000, &g_aaAppBufferForWriteB[0], 1024) → 发起写入请求→保存任务参数→标记 s_jobResult=MEMIF_JOB_PENDING→模块状态置为 MEMIF_BUSY
↓
主函数进入循环（模块状态为 BUSY）
↓
循环内：反复调用 Fls_MainFunction() → 检测到待执行写入任务→调用 LLD 层 Fls_SectorWrite()执行按双字对齐的编程
↓
硬件写入完成→Fls_MainFunction()更新 s_jobResult=MEMIF_JOB_OK→模块状态置为 MEMIF_IDLE
↓
循环条件不满足（MEMIF_IDLE == Fls_GetStatus()）→ 退出循环→写入操作执行完成

5.3 SDK 关键函数说明

Fls_MainFunction()：AUTOSAR Fls 模块的后台主函数，内部完成所有 FLASH 硬件操作的实际执行，执行流程如下：

① 检查全局任务状态 s_jobResult 是否为 MEMIF_JOB_PENDING（有任务待执行）；

② 若有任务，根据 s_jobOpnType（任务类型：擦/写/读）调用对应的 LLD 层硬件函数：
- 擦除：调用 Fls_SectorErase()（解锁 FLASH→扇区擦除→锁定 FLASH）；
- 写入：调用 Fls_SectorWrite()（解锁 FLASH→逐双字编程→锁定 FLASH）；
- 读取：调用 Fls_SectorRead()（直接内存映射读取，无需解锁）；
- 对比：调用 Fls_SectorCompare()（读 FLASH+逐字节对比）；
- 空白检查：调用 Fls_SectorBlankCheck()（读 FLASH+校验 0xFF）；

③ 硬件操作完成后，根据 LLD 层返回结果更新 s_jobResult；

④ 更新 FIs 模块状态为 MEMIF_IDLE（空闲），表示任务执行完成；

⑤ 根据任务执行结果，调用用户配置的任务执行完成通知或任务执行错误通知

6 FLASH 保护

6.1 FLASH 读/写保护

通过配置选项字节，可实现对 FLASH 的读/写保护。

读保护

通过配置选项字节 READPROT 的值可以为存储块添加读保护。读保护有三个等级，分别是：等级 0、等级 1、等级 2，具体情况如下：

表格 2 读保护级别区别

类别	READPROT	描述
等级 0	0xAA	主存储区、选项字节可擦、写、读。
等级 1	除了 0xAA 和 0xCC 的其它值	用户模式：允许对主存储区、选项字节擦、写、读。 Debug、SRAM 运行：除了片擦操作，禁止访问主存储区；选项字节可擦、写、读。当等级修改为 0 时，会先执行主存储区片擦。
等级 2	0xCC	无法 Debug、SRAM 运行，允许对主存储区进行擦、写、读，允许读取选项字节，允许写入读保护等级外的选项字节，不允许擦除选项字节。

写保护

可通过配置写保护选项字节 PWRP 及 DWRP 的值来实现对主存储块对应的页进行写保护控制。写保护开启后，主存储区对应页的内容使用任何方式都不能被修改。对于 PFLASH，写保护的基本单位是 16 页(8KB)；对于 DFLASH，写保护的基本单位是 2 页(1KB)。

表格 3 主存储区写保护 WRPx 功能描述

产品容量	功能描述
G32A1085 A1065 A1045	WRPx 中的每一个 bit 位控制主存储区 8KB（16 页）或 1KB（2 页）地址的写保护 0：开启写保护 1：未开启写保护 WRP0：PFLASH 写保护配置第 0-127 页 WRP1：PFLASH 写保护配置第 128-255 页（仅 G32A1085 和 G32A1065） WRP2：PFLASH 写保护配置第 256-383 页（仅 G32A1085） WRP3：PFLASH 写保护配置第 384-511 页（仅 G32A1085） WRP4：DFLASH 写保护配置第 0-15 页 WRP5：DFLASH 写保护配置第 16-31 页 WRP6：DFLASH 写保护配置第 32-47 页（仅 G32A1085 和 G32A1065） WRP7：DFLASH 写保护配置第 48-63 页（仅 G32A1085 和 G32A1065）

注意：Flash 读/写保护配置是互相独立的，解除写保护不会强制丢失主存储区的内容，而是原样保留。

6.2 选项字节编程

选项字节为用户提供了一些可供选择的功能，它主要由 12 个可配置的字节和对应的补码组成。上电后，选项字节区将被加载到 FMC_OBCS 和 FMC_WRTPROT0/1 寄存器中（选项字节只有每次被重加载到 FMC 寄存器后才会生效）。

擦选项字节

选项字节支持擦除功能，正确的选项字节擦除（或选项字节写入操作）结束后，FMC_STS 寄存器的 OCF 将会被置位，若开启了 OCIE 中断则将触发一个操作完成中断。
擦除步骤：

（1） 解锁 FLASH，检查 BUSYF 标志，确保没有正在进行的 FLASH 操作。

（2） 检查 BUSYF 标志确保没有正在进行的 FLASH 操作

（3） 解锁选项字节区域，等待 OBWEN 置位

（4） 置位 OBE

（5） 置位 STA

（6） 等待 BUSYF 拉低，对擦除地址的值进行读校验

写选项字节

选项字节的 12 个可配置字节均支持写入功能。

编程步骤：

（1） 检查 BUSYF 标志确保没有正在进行的 FLASH 操作

（2） 解锁选项字节区域，等待 OBWEN 置位

（3） 置位 OBP

（4） 往 FMC_ADDR 写入编程的起始地址（注意双字对齐）

（5） 配置编程长度 PROGLEN，并将待编程的数据依次写入 FLASH_PROG_DATAx（硬件保证写入数据互补）

（6） 置位 STA

（7） 等待 BUSYF 拉低，对写入地址的值进行读校验

AUTOSAR Fls 写选项字节执行流程

在 AUTOSAR Fls 中无单独的选项字节擦除指令，针对选项字节的修改添加了一个函数接口：Fls_WriteOptionByte(SourceAddressPtr)来执行完整的选项字节解锁擦除编程流程，需要完成擦

除时，可以通过编程默认值来完成擦除操作，程序执行后需要芯片重新上电，配置才起作用

应用代码示例：

```
uint32_t g_OptionByteEraseBuf[6] = {
0xFFFF55AA, // [0] 0x1FFFF800-0x1FFFF803：READPROT=0xAA(无读保护)、
UOB=0xFF(默认)、补码位占位
0xFFFFFFFF, // [1] 0x1FFFF804-0x1FFFF807：Data0/Data1=0xFF(默认)
0xFFFFFFFF, // [2] 0x1FFFF808-0x1FFFF80B：WRP0/WRP1=0xFF(无写保护)
0xFFFFFFFF, // [3] 0x1FFFF80C-0x1FFFF80F：WRP3/WRP2=0xFF(无写保护)
0xFFFFFFFF, // [4] 0x1FFFF810-0x1FFFF813：WRP5/WRP4=0xFF(无写保护)
0xFFFFFFFF // [5] 0x1FFFF814-0x1FFFF817：WRP7/WRP6=0xFF(无写保护)
};

Fls_WriteOptionByte(g_OptionByteEraseBuf); //  触发擦除+恢复默认
```

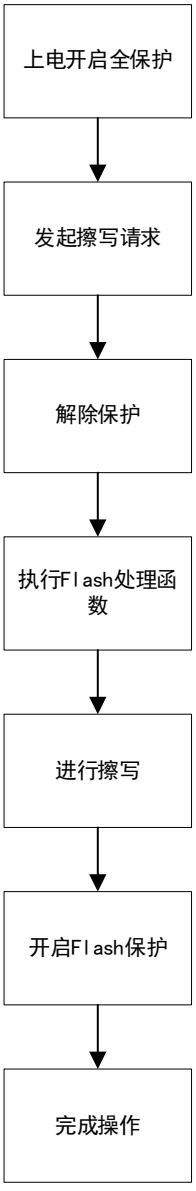
7 FLASH 注意事项

7.1 建议一

未操作 FLASH 时将 FLASH 配置为全保护状态，擦写操作前解除保护，擦写操作完成后再配置为全保护。 例如对于有升级功能的应用，上电时配置 BOOT 和 APP（整个 FLASH 区）为全保护状态， 在升级时配置为只保护 BOOT 区， 升级完成后再配置为全保护状态。

FLASH 保护执行流程

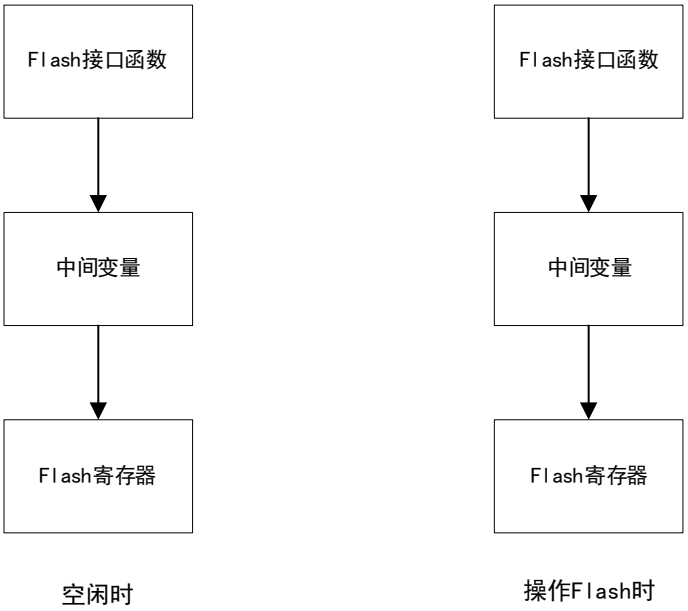
图 1 Flash 保护流程图



7.2 建议二

通过修改底层驱动以及配合应用层逻辑， 将原 FLASH 底层接口函数修改为不直接操作寄存器，而是通过变量指向寄存器进行操作，默认状态下，该变量不指向寄存器，只有在需要操作 FLASH 时，才将变量指向寄存器进行操作。该操作逻辑如下图所示：

图 2 Flash 寄存器操作流程



7.3 建议三

升级场景下，FLASH 擦写函数动态加载至 RAM 执行。对于有升级功能的应用，整个 ROM（FLASH）中不保留 FLASH 擦写接口函数，在升级时使用 CAN 诊断报文将 FLASH 擦写接口函数放到 RAM 中执行，升级完成后清除 RAM 中的 FLASH 擦写接口函数。

升级时通过 CAN 诊断将擦写函数传输至 RAM，执行后清除，避免 ROM 中静态函数被误调用。

8 版本历史

表格 4 文件版本历史

日期	版本	变更历史
2026.5	1.0	● 初始版本

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。