Linux MMC 离线烧录
开发指南

版本号: 3.0

发布日期: 2025-06-20

深圳市宇芯数码技术有限公司yxsj_01

版本历史

版本号	日期	制/修订人	内容描述
1.0	2021-10-13	AWA0332	初始版本
1.1	2021-11-23	AWA0332	更新描述
1.2	2021-11-29	AWA0332	更新标题格式和错别字
1.3	2021-12-1	AWA0332	转成 markdown 格式, 修复错误，增加说明
1.4	2022-07-2	AWA1832	集成 NOR 方案
1.5	2023-05-26	AWA1767	增加 A523 相关平台
1.6	2024-09-1	AWA2256	增加 T536、MR536 相关平台，修正错别字
1.7	2024-10-23	AWA2256	增加 V821 平台，修改已经失效的文件路径，增加设备树存储配置
2.0	2024-11-1	AWA2256	将 NOR 方案转移至《Linux_NOR 离线烧录 _ 开发指南》
2.1	2025-3-22	XAA0312	新增 A537/A333 平台
2.2	2025-5-6	XAA0329	新增 H726 平台
3.0	2025-06-20	AWA2256	修改适用范围为所有产品，支持通过配置文件生成烧录器固件

版权所有 © 珠海全志科技股份有限公司。保留一切权利

目　录

版权所有 © 珠海全志科技股份有限公司。保留一切权利

插　图

深圳市宇芯数码技术有限公司yfxy_01

版权所有 © 珠海全志科技股份有限公司。保留一切权利

1 前言

1.1 文档简介

该文档介绍 sunxi 平台 SD/eMMC 离线烧录的使用方法，展示如何生成离线烧录的 img，用于给烧录器厂家进行离线烧录。

1.2 目标读者

- U-Boot 开发/维护人员
- eMMC 驱动开发/维护人员

1.3 适用范围

文档适用于所有产品。

1.4 相关术语介绍

1.4.1 硬件术语

离线烧录：

通常指 “裸片烧录”，芯片在未贴板之前，搭配相应的适配座放在编程器上进行烧录，与之对应的是在线烧录。

版权所有 © 珠海全志科技股份有限公司。保留一切权利

2 烧录器固件包生成方法

说明：烧录器固件是专门生成的一个固件，生成这个固件后，可以直接交给烧录器厂家, 用于离线烧录。烧录器固件打包命令如下：

```
source build/envsetup.sh   //配置环境变量
pack -w                    //打包非安全固件
pack -w -s                 //打包非安全固件
或
./build.sh pack_raw        //根目录下打包非安全固件
./build.sh pack_secure_raw //根目录下打包安全固件
```

生成 MMC 烧录器固件 xxx_programmer.img。

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3 烧录器固件包配置方法

3.1 eMMC 存储分布说明

- boot0_b 为 boot0 的备份，bootpacket_b 为 bootpacket 的备份。
- toc0 存放安全 boot0，toc0_b 存放安全 boot0 的备份。
- bootpacket 分区里面一般包含 uboot、atf 以及 optee 的 bin 文件，toc1 存放安全的 boot-packet。

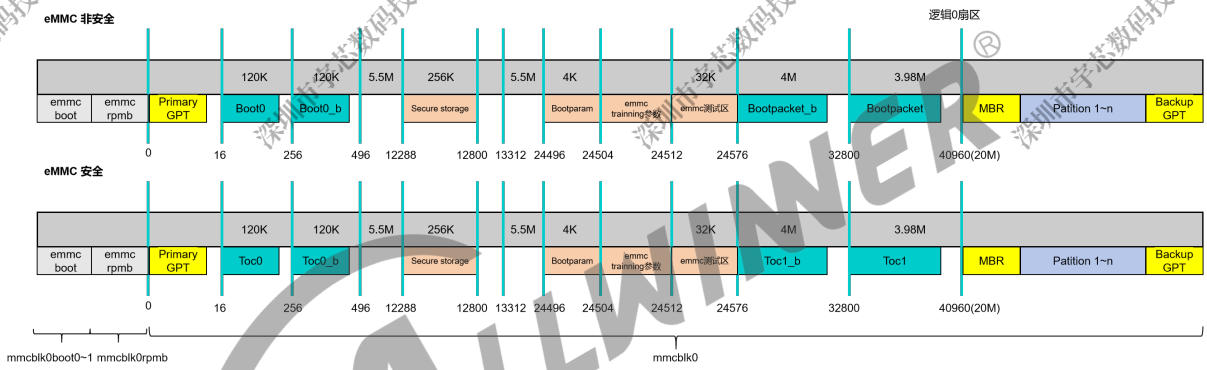


图 3-1: eMMC 存储分布图

3.2 配置文件说明

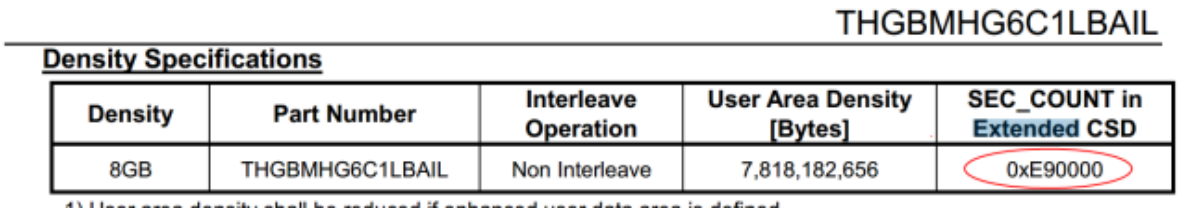
生成烧录器固件时会使用以下格式的配置文件：device/config/chips/xxx/configs/default/programmer_img.cfg

```
flash_total_sectors=15269888
logic_offset=40960

[normal]
;item=img_name       img_filename,       flash_offset_sector
item=boot0,          boot0_sdcard.fex,     16
item=boot0_b,        boot0_sdcard.fex,     256
item=bootpkg_b,      boot_package.fex,     24576
item=bootpkg,        boot_package.fex,     32800

[secure]
;item=img_name       img_filename,          flash_offset_sector
item=toc0,          toc0_sdcard.fex,         16
item=toc0_b,        boot0_offline_secure.fex,    256
item=toc1_b,        toc1.fex,              24576
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

```
item=toc1,          toc1.fex,          32800
```

- flash_total_sectors 表示实际 eMMC flash 的大小，需要根据实际 eMMC 的 datasheet 填写，如下图所示。



THGBMHG6C1LBAIL

Density Specifications

Density	Part Number	Interleave Operation	User Area Density [Bytes]	SEC_COUNT in Extended CSD
8GB	THGBMHG6C1LBAIL	Non Interleave	7,818,182,656	0xE90000

图 3-2: eMMC-datasheet 扇区大小

- logic_offset 表示逻辑 0 扇区地址。
- item 表示物理分区以及要烧写的镜像文件和扇区地址，[normal] 对应非安全固件，[secure] 对应安全固件。
- boot0_b 为 boot0 的备份，bootpkg_b 为 bootpkg 的备份，安全同理。
- toc0_b 存放支持安全方案烧录器的 boot0，第一次启动时 uboot 会将其恢复成安全 boot0 的备份。
- 以 # 和; 开头的行将被忽略
- 镜像文件的打包路径请查看.buildconfig 中LICHEE_PACK_OUT_DIR环境变量
- 对于逻辑分区，请修改 device/config/chips/xxx/configs/default/sys_partiiton.fex。

3.3 UBOOT 存储配置方法

注意要确保 uboot 中存储配置与 programmer_img.cfg 一致，否则可能无法读写对应分区。

1. 在 tina 根目录下执行./build.sh uboot_menuconfig 查看 uboot 图形化配置界面
2. 检查 uboot 中的存储配置是否需要修改
3. 执行./build.sh uboot_saveconfig 将修改保存到目前正在使用的 uboot defconfig 文件中

版权所有 © 珠海全志科技股份有限公司。保留一切权利

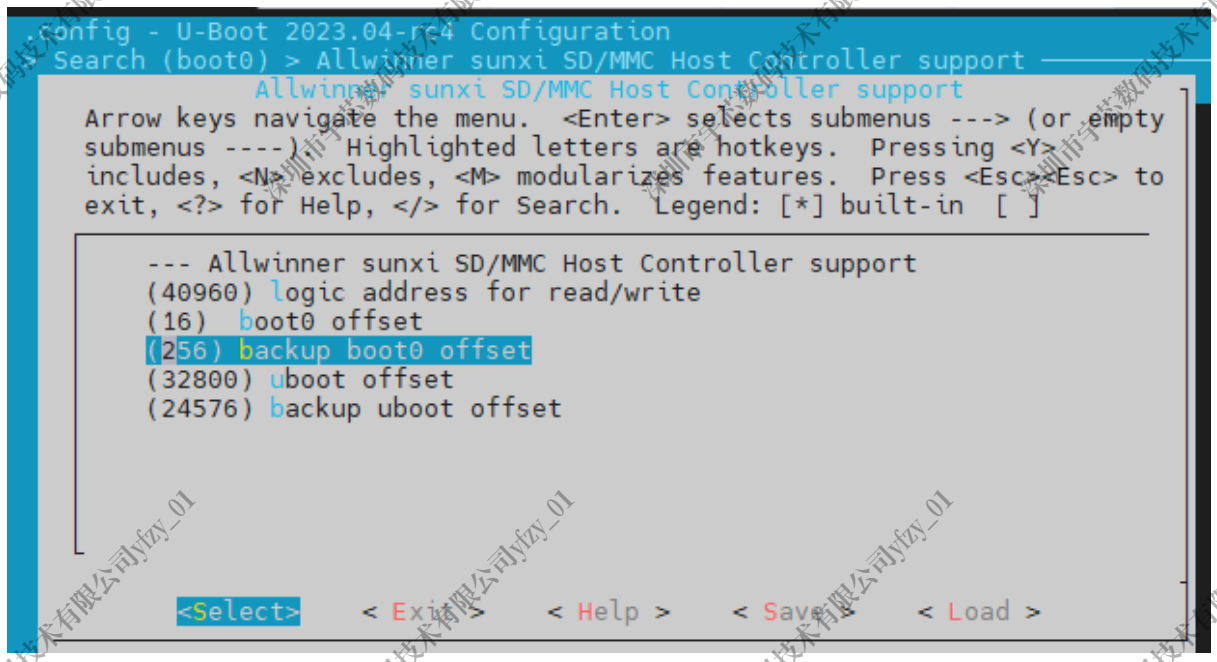


图 3-3: UBOOT 存储配置

3.4 SPL 存储配置方法

注意要确保 spl 中配置的 uboot 镜像地址与 programmer_img.cfg 一致，否则可能无法加载 uboot。

- 查看默认配置：{sdk}/brandy/brandy-2.0/spl/include/sunxi_flashmap.h

版权所有 © 珠海全志科技股份有限公司。保留一切权利

include > C sunxi_flashmap.h > ...

```
32
33  #ifdef CFG_SDMMC_UBOOT_OFFSET
34  #define DEFAULT_UBOOT_START_SECTOR_IN_SDMMC    \
35      CFG_SDMMC_UBOOT_OFFSET
36  #else
37  #define DEFAULT_UBOOT_START_SECTOR_IN_SDMMC (32800)
38  #endif
39
40  #ifdef UBOOT_START_SECTOR_IN_UFS
41  #define DEFAULT_UBOOT_START_SECTOR_IN_UFS    \
42      UBOOT_START_SECTOR_IN_UFS
43  #else
44  #define DEFAULT_UBOOT_START_SECTOR_IN_UFS (32800)
45  #endif
46        You, 16分钟前 • K1:sunxi:P2:mmc: support uboot offset config ...
47
48
49  #ifdef CFG_SDMMC_UBOOT_BACKUP_OFFSET
50  #define DEFAULT_UBOOT_BACKUP_START_SECTOR_IN_SDMMC    \
51      CFG_SDMMC_UBOOT_BACKUP_OFFSET
52  #else
53  #define DEFAULT_UBOOT_BACKUP_START_SECTOR_IN_SDMMC (24576)
54  #endif
```

图 3-4: SPL 默认存储配置

- 修改默认配置：{sdk}/brandy/brandy-2.0/spl/board/xxx/mmc.mk

board > mr153 > M mmc.mk

You, 3分钟前 | 2 authors (chengweipeng and others)

```
1
2   #
3   #config file for sun8iw22
4   #
5   #stroage
6   FILE_EXIST=$(shell if [ -f $(TOPDIR)/board/$(PLATFORM)/common.mk ]; then echo yes; else echo no; fi;)
7   ifeq (x$(FILE_EXIST),xyes)
8   include $(TOPDIR)/board/$(PLATFORM)/common.mk
9   else
10  include $(TOPDIR)/board/$(CP_BOARD)/common.mk
11  endif
12
13  MODULE=mmc
14  CFG_SUNXI_SDMMC =y
15  CFG_SDMMC_UBOOT_OFFSET=32800
16  CFG_SDMMC_UBOOT_BACKUP_OFFSET=24576
```

图 3-5: 修改 SPL 存储配置

版权所有 © 珠海全志科技股份有限公司。保留一切权利

4 补充说明

4.1 update_mbr 工具说明

根据分区配置文件，更新主引导目录文件 sunxi_mbr.fex ，sunxi_gpt.fex 及分区的下载文件列表 dlinfo.fex, 使用方法如下:

update_mbr sys_partition.bin 4 sunxi_mbr.fex dlinfo.fex 15269888 40960 0

参数:

```
--sys_partition.bin----分区表的二进制文件
--分区表的备份个数，一般spinor为1，其他为4
--sunxi_mbr.fex------生成mbr格式分区表的文件
--dlinfo.fex----------下载文件信息表
--15269888----------该flash的大小，比如eMMC的USR分区的大小，单位为sector
--40960-------------------该flash的逻辑地址，单位为sector，注意旧版本单位为Mbytes
--0--------------------代表flash的类型，1是spinor flash，0是非 spi nor类型
```

输出文件:

```
sunxi_mbr.fex
dlinfo.fex
sunxi_gpt.fex
sunxi_gpt_head.fex
Sunxi_gpt_pte.fex
```

4.2 programmer_img 工具说明

生成 mmc 介质的烧录器固件。

4.2.1 生成物理分区

使用方法如下：programmer_img cfg_create ${out_img} programmer_img.cfg normal

参数说明：

```
--cfg_create-----使用配置文件生成烧录器固件
--${out_img}------为生成的烧录器镜像文件
--programmer_img.cfg------配置文件
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

```
--normal-----------------代表固件类型，normal是非安全固件，secure是安全固件
```

备注：

1. 读取配置文件文件。
2. 解析逻辑扇区偏移，生成物理扇区大小的文件，将整个文件填充 0。
3. 将各物理分区文件写到相应的位置。

4.2.2 生成逻辑分区

使用方法如下：programmer_img sys_partition.bin sunxi_mbr.fex ${out_img} ${in_img} sunxi_gpt.fex 参数说明：

```
--sys_partition.bin----分区表的脚本文件
--sunxi_mbr.fex------为mbr格式的分区表文件
--${out_img}------为已生成物理分区的烧录器镜像文件
--${in_img}-----为固件备份分区文件
--sunxi_gpt.fex----为gpt格式的分区表文件
```

备注：

1. 读取已生成物理分区的烧录器镜像文件。
2. 解析分区表中 mbr 分区大小，写 MBR/GPT 分区表到镜像文件。
3. 解析分区表中各逻辑分区信息，写分区文件到镜像文件。

4.3 dragonsecboot 工具说明

- 根据指定的 keys 生成 toc0 文件。
- 根据指定的 keys 和 cnfbase 生成 toc1 文件。
- 根据配置文件生成 keys。
- 按配置文件的配置进行打包生成目的文件。

使用方法如下：

```
dragonsecboot -toc0 <cfg_file> <keypath> <version_file>
dragonsecboot -toc1 <cfg_file> <keypath> <cnfbase> <version_file>
dragonsecboot -key <cfg_file> <keypath>
dragonsecboot -pack <cfg_file>
```

参数说明：

```
-toc0----表示要生成 toc0 文件
-toc1----表示要生成 toc1 文件大小
-key----表示要生成 key
-pack----表示进行打包
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

```
cfg_file----配置文件
keypath----key 的路径
cnfbase----输入的 cnf_base.cnf 文件
version_file----固件防回滚配置文件
```

4.4 signature 工具说明

对 MBR 指定要进行签名的分区文件进行签名，使用方法如下：

```
signature <sunxi_mbr_file> <dlinfo_file>
```

参数说明：

```
sunxi_mbr_file: 输入的 MBR 文件
dlinfo_file: 输入的分区下载列表文件
```

4.5 sigbootimg 工具说明

在输入文件的后面添加一个证书，并输出一个带证书的文件，使用方法如下：sigbootimg --image <input_img> --cert <cert_file> --output <output_img>

参数说明:

```
input_img: 输入的文件或镜像
cert_file: 文件或镜像对应的证书
output_img: 带证书的文件或镜像
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

5 验证方法

5.1 eMMC 方案自测验证方法

5.1.1 检查生成烧录器的固件

- 使用 hexdump 工具查看 boot0_sdcard.fex 是否在第 16 个扇区, 备份是否在第 256 个扇区（特殊的如 A50 平台在 384 个扇区）。
- 使用 hexdump 工具查看 boot_package.fex 是否在第 32800 个扇区，备份是否在第 24576 个扇区。

5.1.2 自测验证方法

只要将 programmer 固件包中的数据按 1：1 的方式拷贝到 eMMC 上即可完成离线烧录操作，如下图。

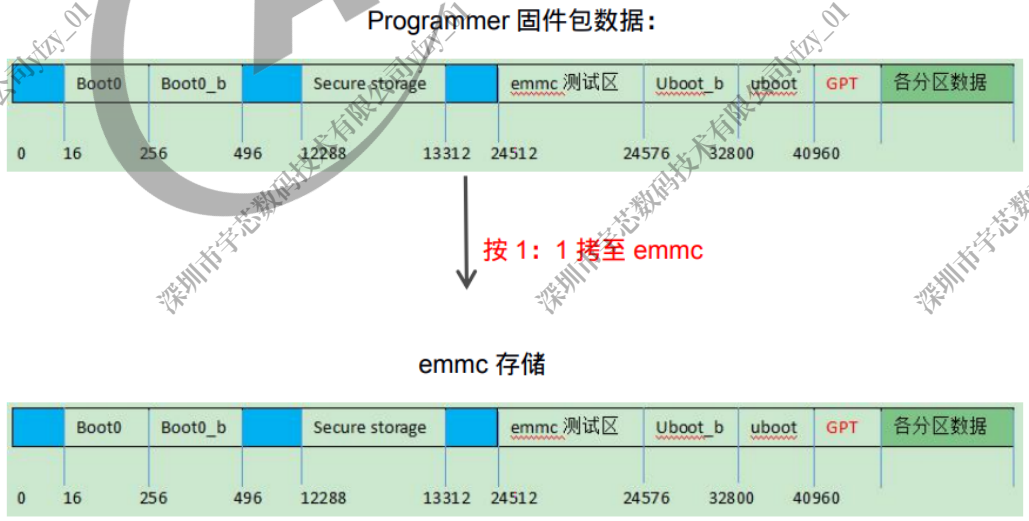


图 5-1: programmer 到 flash 数据分布

Programmer 固件包中蓝色区域为填充区域，以 0 填充，且 boot0 等非蓝色区域其实际数据大小一般都不会超过图中分配的大小，所以剩下的空间也以 0 填充，所以在将 programmer 固件包数

版权所有 © 珠海全志科技股份有限公司。保留一切权利

据拷到 eMMC 存储的时候，请做一些预处理，即拷贝时如遇到为 0 的连续大片区域时，则跳过，不拷贝，这样会提高拷贝效率。

5.1.3 制作特制的卡启动固件

1. 修改 sun50iw3p1.dtsi 文件。

   - 把 mmc2=&sdc2 注释掉。
   - 把sdc0：sdmmc@04020000{2.....}整体（包括 {} 里面的内容）移动到sdc2：sdmmc@04022000前面。

2. 重新编译 linux。
3. 执行命令:pack —重新打包生成卡启动固件:sun50iw3p1_android_y2_uart0.img。

注意：这一步只有在验证的时候使用，正式量产时不需要这一步，sun50iw3p1.dtsi 保持未修改前的状态。

5.1.4 烧录 eMMC

1. PC 启动 adb。
2. 然后输入以下命令。

   - mkdir /mnt/sd
   - mount /dev/block/mmcblk0p1 /mnt/sd
   - cd /mnt/sd
   - time dd if=sun50iw3p1_android_y2_card0_programmer.img of=/dev/block/mmcblk1

3. 等待命令执行完毕，这个时间会比较长。完成后关机。

5.1.5 重新开机

拔掉卡，重新开机，如果能够正常启动，说明烧录器固件是正常的。

5.2 烧写后无法启动的调试方法

- 对比烧写进去镜像文件是否在相应的位置，详细查看对应存储介质的存储分布图。
- 使用 TigerDump 工具 dump 相应的文件。
- 使用 SDC0 启动到内核的命令行，使用 DD 命令进行读取 flash 中的镜像文件。

版权所有 © 珠海全志科技股份有限公司。保留一切权利

6 FAQ

6.1 如何查看 eMMC 的 USR 分区的大小?

解释说明：由于分区表中最后一个 udisk 分区的大小就是其他分区用完所剩余的 flash 空间的大小，因此需要根据实际 eMMC flash 的大小，生成合适的分区表。

查看方法：找到某款 emmc 物料的数据手册，usr 分区的大小在图中的红框位置。根据下图，可知 8G 的 eMMC 的大小为 0xE90000。

7.4 Extended CSD Register

The Extended CSD register defines the eMMC properties and selected modes. It is 512 bytes long.
The most significant 320 bytes are the Properties segment, which defines the eMMC capabilities and cannot be modified by the host. The lower 192 bytes are the Modes segment, which defines the configuration the eMMC is working in. These modes can be changed by the host by means of the SWITCH command.

R: Read only
W: One time programmable and not readable.
R/W: One time programmable and readable.
W/E: Multiple writable with value kept after power failure, H/W reset assertion and any CMD0 reset and not readable.
R/W/E: Multiple writable with value kept after power failure, H/W reset assertion and any CMD0 reset and readable.
R/W/C_P: Writable after value cleared by power failure and HW/ rest assertion (the value not cleared by CMD0 reset) and readable.
R/W/E_P: Multiple writable with value reset after power failure, H/W reset assertion and any CMD0 reset and readable.
W/E/_P: Multiple writable with value reset after power failure, H/W reset assertion and any CMD0 reset and not readable

[Table 26] Extended CSD Register

Name	Field	Size (Bytes)	Cell Type	CSD slice	CSD Value			
					8GB	16GB	32GB	64GB
Properties Segment								
Reserved[1]		6	-	[511:506]	-			
Extended Security Commands Error	EXT_SECURITY_ERR	1	R	[505]	0x00			

图 6-1: eMMC usrspace

Sleep Notification Timeout	SLEEP_NOTIFICATION_TIME	1	R	[216]	0x07			
Sector Count	SEC_COUNT	4	R	[215:212]	0xE90000	0x1D1F000	0x3A3E000	0x747C000
Secure Write Protect Information	SECURE_WP_INFO	1	R	[211]	0x01			
Minimum Write Performance for 8bit at 52MHz	MIN_PERF_W_8_52	1	R	[210]	0x00			

图 6-2: eMMC SectorCount

6.2 如何编译支持安全方案烧录器的 boot0?

解释说明：由于芯片默认是非安全的芯片，因此无法启动安全固件，所以必须要制造一个非安全的 boot0(即 boot0_offline_secure.fex)，负责把芯片烧写成安全的芯片，然后复位芯片，接着芯片可以从安全的固件启动。

版权所有 © 珠海全志科技股份有限公司。保留一切权利

编译方法：到 spl 目录执行以下命令，{CHIP} 为平台名称

```
make distclean
make p={CHIP}
make offline_secure_mmc
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

著作权声明

版权所有 ©2025 珠海全志科技股份有限公司。保留一切权利。

本文档及内容受著作权法保护，其著作权由珠海全志科技股份有限公司（“全志”）拥有并保留一切权利。

本文档是全志的原创作品和版权财产，未经全志书面许可，任何单位和个人不得擅自摘抄、复制、修改、发表或传播本文档内容的部分或全部，且不得以任何形式传播。

商标声明


、 
、 
、 
（不完全列举）均为珠海全志科技股份有限公司的商标或者注册商标。在本文档描述的产品中出现的其它商标，产品名称，和服务名称，均由其各自所有人拥有。

免责声明

您购买的产品、服务或特性应受您与珠海全志科技股份有限公司（“全志”）之间签署的商业合同和条款的约束。本文档中描述的全部或部分产品、服务或特性可能不在您所购买或使用的范围内。使用前请认真阅读合同条款和相关说明，并严格遵循本文档的使用说明。您将自行承担任何不当使用行为（包括但不限于如超压，超频，超温使用）造成的不利后果，全志概不负责。

本文档作为使用指导仅供参考。由于产品版本升级或其他原因，本文档内容有可能修改，如有变更，恕不另行通知。全志尽全力在本文档中提供准确的信息，但并不确保内容完全没有错误，因使用本文档而发生损害（包括但不限于间接的、偶然的、特殊的损失）或发生侵犯第三方权利事件，全志概不负责。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文档未以明示或暗示或其他方式授予全志的任何专利或知识产权。在您实施方案或使用产品的过程中，可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。全志不承担也不代为支付任何关于获取第三方许可的许可费或版税（专利税）。全志不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。