文档版本	
发布日期	

TG7100C Efuse 及 Flash 加解密使用说明

天猫精灵

芯片开放社区
Open Chip Community

TG7100C

Efuse 及应用程序加解密使用说明

版本：*1.0*

Contents

1 概述

TG7100C 带有 1024Bits(128 字节) 的 Efuse，可以作为 RF 功率校准、频偏校准，用户 MAC 地址或者用户数据的存储区域，同时也是 Flash 上用户代码加密密钥和签名公钥 Hash 的存储区域。每个区域都带有读写保护功能，用户可以根据自身的读写保密需求进行设定。

2 Efuse 功能说明

TG7100C 带有 1024Bits(128 字节) 的 Efuse 主要功能分布如表所示,Base Address 为 0x40007000。

表 2.1: 功能分布

Offset	用途	备注
0x00	保留	
0x04	mac_addr[31:0]	Customer MAC Address2(Slot2)，用户可以在生产环节烧录自己的 MAC
0x08	mac_addr_crc[5:0] mac_addr[47:32]	Customer MAC Address2(Slot2)，用户可以在生产环节烧录自己的 MAC
0x0C	保留	
0x10	保留	
0x14	mac_addr[31:0]	TG MAC Address(Slot0), 出厂自带的 MAC 地址
0x18	mac_addr_crc[5:0] mac_addr[47:32]	TG MAC Address(Slot0), 出厂自带的 MAC 地址
0x1C	ECC Public Key Hash (SHA256)	如不使用 ECC 签名功能，该位置可以留给用户自行使用
0x20	ECC Public Key Hash (SHA256)	
0x24	ECC Public Key Hash (SHA256)	
0x28	ECC Public Key Hash (SHA256)	

下页继续

表 2.1 – 续上页

0x2C	ECC Public Key Hash (SHA256)	
0x30	ECC Public Key Hash (SHA256)	
0x34	ECC Public Key Hash (SHA256)	
0x38	ECC Public Key Hash (SHA256)	
0x3C	XIP(Flash) AES Key	Flash AES Key 的低 128Bits 如不需要使用 Flash 应用程序代码加密功能，该位置可以留给用户自行使用
0x40	XIP(Flash) AES Key	
0x44	XIP(Flash) AES Key	
0x48	XIP(Flash) AES Key	
0x4C	XIP(Flash) AES Key	Flash AES Key 的高 128Bits 如不需要使用 Flash 应用程序代码加密功能或者使用加密的密钥只有 128Bits(AES-128) 该位置可以留给用户自行使用
0x50	XIP(Flash) AES Key	
0x54	XIP(Flash) AES Key	
0x58	XIP(Flash) AES Key	
0x5C	Sec Engine AES Key	128 Bits 应用数据 AES Key 如不需要使用加密引擎加密用户数据，该位置可以留给用户自行使用
0x60	Sec Engine AES Key	
0x64	Sec Engine AES Key	
0x68	Sec Engine AES Key	
0x6C	mac_addr[31:0]	Customer MAC Address1(Slot1)，用户可以在生产环节烧录自己的 MAC
0x70	mac_addr_crc[5:0] mac_addr[47:32]	Customer MAC Address1(Slot1)，用户可以在生产环节烧录自己的 MAC
0x74	保留	
0x78	保留	
0x7C	Read&Write Lock Config	
0x7C[Bit30]	Read Lock 0x5C-0x6C	
0x7C[Bit29]	Read Lock 0x4C-0x5C	
0x7C[Bit28]	Read Lock 0x3C-0x4C	
0x7C[Bit23]	Write Lock 0x5C-0x64	

下页继续

表 2.1 – 续上页

0x7C[Bit22]	Write Lock 0x4C-0x5C	
0x7C[Bit21]	Write Lock 0x3C-0x4C	
0x7C[Bit20]	Write Lock 0x2C-0x3C	
0x7C[Bit19]	Write Lock 0x1C-0x2C	
0x7C[Bit13]	Write Lock 0x64-0x6C	

在该表格中，RF 功率校准，频偏校准等参数的写入，需要配合 RF 测试固件和测试仪器一起使用，因此在这里不做介绍，对应位置被标记为“保留”。

其它功能位置如表格中所示，主要分为 MAC 地址存储区，ECC 签名公钥 Hash 存储区，Flash 加密密钥存储区，加密引擎硬件密钥存储区。没有预留给客户自定义的参数存储区，如果对应的功能用户不需要使用，则该区域可以拿来作为用户自定义的参数存储区。

地址 0x1C-0x3C 共 32 字节 256Bits，用来存储 ECC 签名的公钥 Hash，使用的是 SHA256, 在使能 ECC 签名的时候，Bootrom 会读取该位置验证公钥的合法性，然后进行签名的验证，当使能 ECC 签名的时候，需要在此位置烧录签名公钥的 Hash，并且该位置只能设置写保护，不能设置读保护，因为该位置需要被 Bootrom 读取，如果不使能 ECC 签名，则该位置可以用作用户参数存储区。

地址 0x3C-0x4C 是预留 Flash AES 加密密钥的低 128Bits,地址 0x4C-0x5C 是预留 Flash AES 加密密钥的高 128Bits,如果用户不使能 Flash 用户程序加密功能,则地址 0x3C-0x5C 可以用作客户自定义的参数存储区,如果用户使用 AES128 加密，则地址 0x3C-0x4C 用来存储 128Bits 的解密密钥。地址 0x4C-0x5C 可以用作客户自定义的参数存储区，如果客户使用 AES192/AES256 加密，则地址 0x3C-0x5C 只能全部用来存放加密密钥，当然存放密钥时候，需要把对应的位置设置为读保护和写保护，以防止密钥的泄露和篡改。

地址 0x5C-0x68 是预留 Sec Engine AES 加密密钥，目前只支持 128Bits，Sec Engine 加密引擎主要用于用户程序在实时通信时，对通信数据进行加密，Sec Engine 加密引擎支持使用软件 AES Key 加密（例如 AES 的 Key 是通过 HTTPS 方式来自于云端），也支持使用硬件 AES Key 加密（就是预先存放在 Efuse 中的 AES 加密密钥）。如果客户需要使用 Efuse 存储预先设定的通信加密密钥，则在烧录完密钥后，设定该区域的读保护和写保护，以防止密钥的泄露和篡改。反之，如果客户不需要使用 Efuse 存储预先设定的通信加密密钥，该区域可以用作客户自定义的参数存储区。

3 自定义 Efuse 区域使用接口及注意事项

Efuse 由于其自身的特点，因此在读写操作上需要有一些事项需要留意。

1. Efuse 读写之前需要把系统时钟切换到晶振时钟
2. Efuse 的读写必须是 4 字节地址对齐

标准驱动 tg7100c_mfg_efuse.c 中提供了用户自定义区域数据读写的统一接口，在该接口中实现了系统时钟的切换和恢复。

3.1 烧写 Efuse

```
int8_t mfg_efuse_write_pre(uint32_t addr,uint32_t *data,uint32_t countInword)
```

该函数主要用于将用户数据存储到对应的 Efuse 寄存器，此时烧写并未真正的进行，用户可以在烧写之前，多次调用该函数，将所有的数据全部存储到对应的 Efuse 寄存器。

```
入口参数：
addr：要读取的地址偏移
data：读取到的数据存放的缓冲区指针
countInword：读取数据的长度（按照字计算）
返回参数：
0：成功，-1：失败
```

```
int8_t mfg_efuse_program(void)
```

该函数主要用于将 Efuse 寄存器中的数据，烧录到 Efuse 存储实体。0：成功，-1：失败。

例如，在不使用 ECC 对应用数据进行签名的情况下，可以把 0x1C-0x3C 用作用户自定义参数存储区，假设要对 0x1C 地址烧写 4 字节数据，0x2C 地址烧写 8 字节数据，函数调用如下：

```
mfg_efuse_write_pre(0x1C,buf1,1);
mfg_efuse_write_pre(0x2C,buf2,2);
mfg_efuse_program();
```

3.2 读取 Efuse

```
int8_t mfg_efuse_read(uint32_t addr,uint32_t *data,uint32_t countInword,uint8_t reload)
```

该函数主要用于从 Efuse 中读取已经烧录的数据。

```
入口参数：
addr：要读取的地址偏移
data：读取到的数据存放的缓冲区指针
countInword：读取数据的长度（按照字计算）
reload：是否要重新加载 efuse 中最新的数据，一般都要为 1
返回参数：
0：成功，-1：失败
```

例如, 要从 0x1C 地址处读取 8 个字节，函数调用如下 mfg_efuse_read (0x1C,buf,2,1)。对应读保护的区域进行读操作，得到的结果将是 0。

对于 MAC 地址，RF 校准参数等，tg7100c_mfg_efuse.c 也提供了相应的 API，但是原则上这些参数的写入，都是由产线产测固件实现的，标准的 SDK 中已经实现了对应参数的读操作并融入到 SDK 的功能中，这写参数用户不需要关心，故文档不再对这些参数的读写做介绍。此外，由于 Efuse 只有写入一次的特点，往往即使是用户自定义的参数，也是在生产阶段写入到 Efuse 中的，用户只要在应用程序中读取即可。

量产烧录工具支持在生产阶段，将客户私有的数据写入到 Efuse 中，具体可参见量产工具的介绍文档。

4 Flash 加密功能

Flash 加密支持 AES CTR 模式。密钥格式支持 AES-128、AES-192、AES-256 等。

5 通过 Dev Cube 工具使用 Flash 加密功能

1. 设定加密密钥，生成加密的镜像并烧写镜像：

Dev Cube 左上角的“View”菜单选择“IOT”界面，在通常使用 Dev Cube 工具时，需要勾选分区表、Boot2、Firmware 等选项，并点击“Create&Download”按钮，工具会先生成烧写文件，再将生成的烧写文件烧写到 flash 的对应地址中。

当使用 flash 加密功能时，需要在 Dev Cube 界面勾选“AES-Encrypt”，并在“Key”和“IV”中填写 AES 加密的 key 和 iv(Dev Cube 工具为了降低使用的复杂度，目前只支持 AES-128)。点击“Create&Download”按钮，工具会根据设定的 key 和 iv 生成对应的加密镜像 (生成的加密镜像文件同样位于 Dev Cube 工具目录的“tg7100c/img_create”文件夹下)，并将加密后的文件写入到 flash 的对应地址中。见下图：

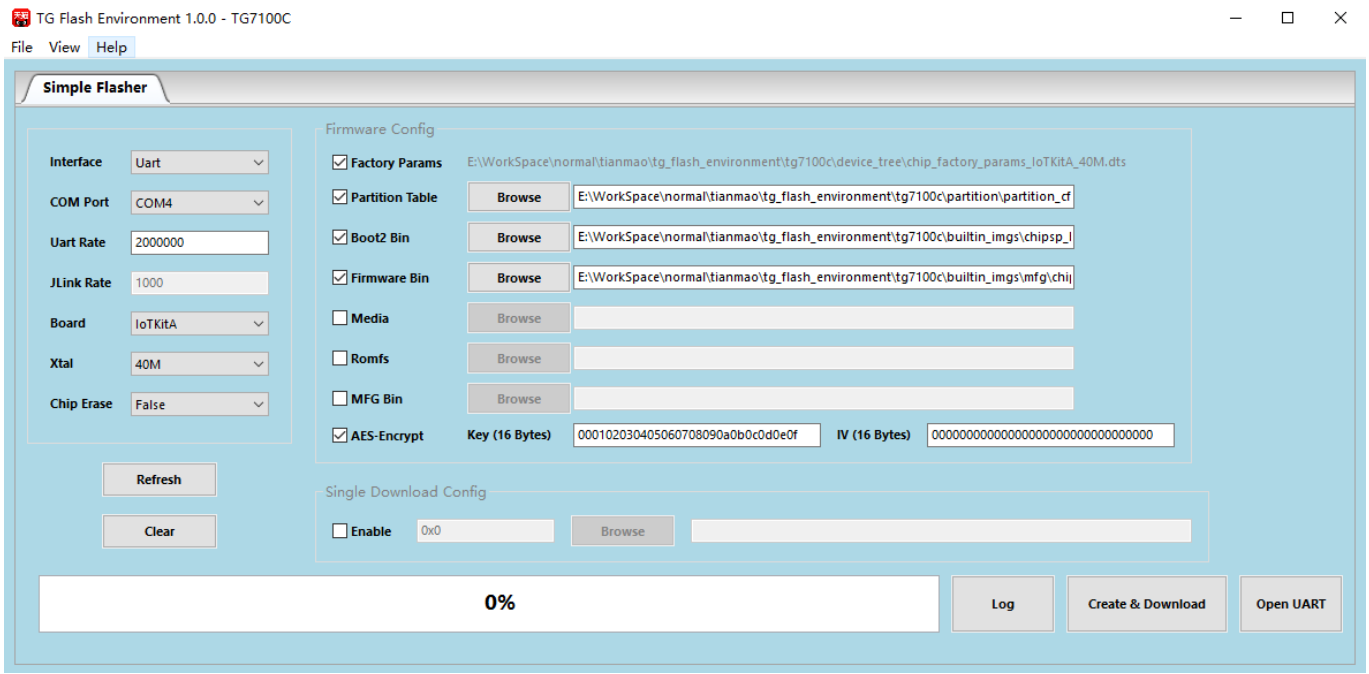

图 5.1: 烧写加密镜像

2. 向 efuse 中写入解密 key，使芯片能正常从 flash 中 boot 起来：

flash 中写入已加密的固件，因为固件内容已加密，此时无法直接从 flash boot 起来，所以需要向 efuse 中写入解密 key

并使能 Flash 解密功能，才能成功从 flash boot 起来。

向 efuse 中写解密 key 需要在 Dev Cube 左上角的“View”菜单选择“MCU”界面，再选择“Security”窗口。在刷新出来的界面中，“AESMode”选择“AES128”，“AES Key”填入之前加密固件使用的 key(“AES Key”右侧的“Write Lock”和“Read Lock”勾选框，当“Write Lock”勾选时，efuse 中写加密 key 的区域在本次写 key 操作后再次写入将会无效，当“Read Lock”勾选时，efuse 中写加密 key 的区域读不到写入的 key 值，只能读到 0)，见下图。

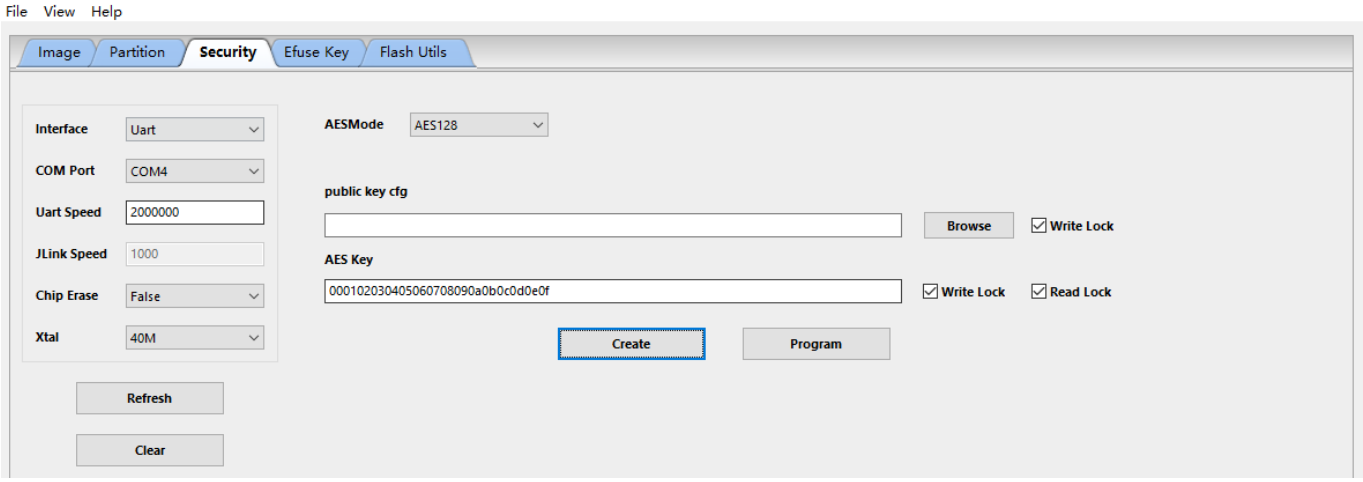

图 5.2: 烧写 efuse key

点击“Create”，当弹出“Create Success”时，带有解密 key 值的 efuse 文件创建成功 (efuse 文件位于 Dev Cube 工具目录的“tg7100c/efuse_bootheader/efuse_data.bin”)。接着点击“Program”，工具会将生成的 efuse 文件写入芯片的 efuse 中。

向 efuse 中写解密 key 完成后，再次复位从 flash 启动，应用程序就可以成功 boot 起来。

注解：在烧写加密镜像和 efuse key 时，一定是先烧写加密固件，再烧写 efuse key，如果先烧写 efuse key，则之后的加密固件会烧写失败。因为写过 efuse key 后，芯片使能了 flash 解密功能，烧写工具无法正常给芯片发送烧写 flash 命令。

---

6 批量烧写工具如何处理 Flash 加密功能

客户在量产环节，只要将“tg7100c/img_create”目录下的生产烧录文件包 whole_img.pack(此时文件包的镜像应该是加密的，并且包含存有加密密钥的文件) 给到量产烧录工具，烧录工具可自行完成加密镜像的烧写和解密密钥的烧写。批量烧写工具的使用细节参考《批量烧写工具使用手册》。

如果客户每个芯片使用的加密密钥不同，也就是“一机一密”模式，也是可以通过量产工具对生产烧录文件包 whole_-img.pack(此时文件包的镜像应该是没有加密的) 进行加密并烧写。