

WS63V100 EFUSE

使用指南

文档版本 02

发布日期 2024-06-27

前言

概述

本文档主要描述了 WS63V100 的客户预留 EFUSE 位域的使用方法。

产品版本

与本文档对应的产品版本如下。

产品名称	产品版本
WS63	V100

读者对象




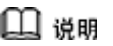
本文档主要适用于以下人员：

- 技术支持工程师
- 软件开发工程师

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
 危险	表示如不避免则将会导致死亡或严重伤害的具有高等级风险的危

符号	说明
	害。
 警告	表示如不避免则可能导致死亡或严重伤害的具有中等级风险的危害。
 注意	表示如不避免则可能导致轻微或中度伤害的具有低等级风险的危害。
 须知	用于传递设备或环境安全警示信息。如不避免则可能会导致设备损坏、数据丢失、设备性能降低或其它不可预知的结果。 “须知”不涉及人身伤害。
 说明	对正文中重点信息的补充说明。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害信息。

修改记录

文档版本	发布日期	修改说明
02	2024-06-27	更新 “3 burnttool 烧写 efuse_cfg.bin 说明” 章节内容。
01	2024-04-10	第一次正式版本发布。
00B02	2024-03-15	<ul style="list-style-type: none">更新 “1 概述” 章节内容。更新 “2 软件编程接口使用指导” 章节内容。
00B01	2024-02-22	第一次临时版本发布。

目 录

前言i

1 概述1

2 软件编程接口使用指导2

3 burnttool 烧写 efuse_cfg.bin 说明4

3.1 生成 efuse_cfg.bin4

3.2 烧录流程5

1 概述

EFUSE 是一种可编程的存储单元，由于其只可编程一次的特征，多用于芯片保存 Chip ID、密钥或其他一次性存储数据。WS63 提供了两种使用方式：通过软件驱动接口直接读写用户预留的 128bit EFUSE 空间；通过烧写工具操作整个 2048bit 空间。

2 软件编程接口使用指导

eFuse 模块提供的接口及功能如下：

头文件路径: include\driver\efuse_user.h

- uapi_efuse_user_read_bit: 从用户预留的 eFuse 空间中读取一位。
- uapi_efuse_user_read_buffer: 从用户预留的 eFuse 空间中读取多个字节，进入提供的缓冲区。
- uapi_efuse_user_write_bit: 向用户预留 eFuse 空间中的对应 bit 写 1。
- uapi_efuse_user_write_buffer: 从提供的缓冲区向用户预留的 eFuse 空间写入多个字节。

示例：

步骤 1 按照 buffer 写 efuse 值。

```
uint8_t efuse_data[8] = {0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88};
uint32_t byte_number = 1;
uint16_t length = 8;
uint32_t ret;
// 第1byte开始写入8个字节
ret = uapi_efuse_user_write_buffer(byte_number, efuse_data, length);
if (ret != 0) {
    // 异常处理
}
```

步骤 2 按照 buffer 读取 efuse 值。

```
uint8_t efuse_data[8] = {0};
uint32_t byte_number = 1;
uint16_t length = 8;
```

```
uint32_t ret;  
// 第1byte开始读取8个字节  
ret = uapi_efuse_user_read_buffer(byte_number, efuse_data, length);  
if (ret != 0) {  
    // 异常处理  
}
```

步骤 3 按照 bit 读取 efuse 值。

```
uint8_t bit_pos = 1;  
uint8_t value;  
uint32_t byte_number = 1;  
uint32_t ret;  
// 读取第一个byte的bit1的值, 存入value中 (0 or 1)  
ret = uapi_efuse_user_read_bit(byte_number, bit_pos, &value);  
if (ret != 0) {  
    // 异常处理  
}
```

步骤 4 按照 bit 写入 efuse 值。

```
uint8_t value;  
uint32_t byte_number = 1;  
uint32_t ret;  
// 向第一个byte的bit1写入1  
ret = uapi_efuse_user_write_bit(byte_number, bit_pos);  
if (ret != 0) {  
    // 异常处理  
}
```

----结束

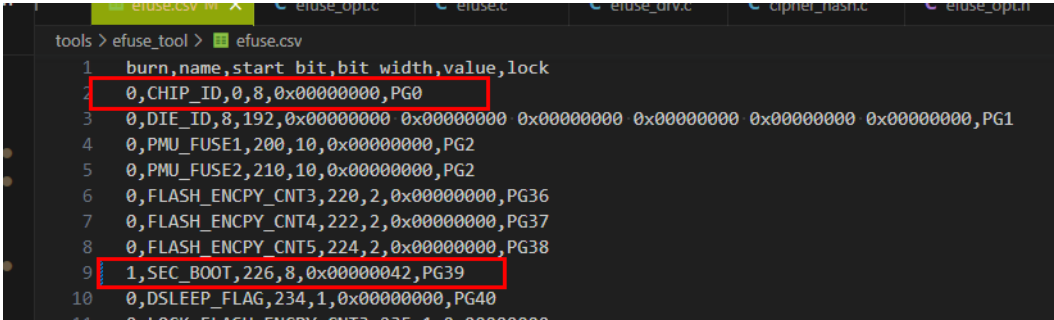
3 burntool 烧写 efuse_cfg.bin 说明

3.1 生成 efuse_cfg.bin

3.2 烧录流程

3.1 生成 efuse_cfg.bin

1. 数据准备



```
tools > efuse_tool > efuse.csv
1 burn,name,start bit,bit width,value,lock
2 0,CHIP_ID,0,8,0x00000000,PG0
3 0,DIE_ID,8,192,0x00000000 0x00000000 0x00000000 0x00000000 0x00000000,PG1
4 0,PMU_FUSE1,200,10,0x00000000,PG2
5 0,PMU_FUSE2,210,10,0x00000000,PG2
6 0,FLASH_ENCPY_CNT3,220,2,0x00000000,PG36
7 0,FLASH_ENCPY_CNT4,222,2,0x00000000,PG37
8 0,FLASH_ENCPY_CNT5,224,2,0x00000000,PG38
9 1,SEC_BOOT,226,8,0x00000042,PG39
10 0,DSLEEP_FLAG,234,1,0x00000000,PG40
```

如上图的 csv 表格，第一行是表头，表格可以根据需要自行按行添加。表格各个字段说明如下：

- burn: 0: 当前位域不烧写；1: 当前位域烧写。
- name: 当前 efuse 位域名字。
- start_bit: 位域起始 bit 位。
- bit_width: 当前位域 bit 长度。
- values: 烧写值。
- lock: 锁定位。

示例：

0,CHIP_ID,0,8,0x00000000,PG0 : chip_id 从 bit0 开始，长度是 8bit，不烧写

1,SEC_BOOT,226,8,0x00000042,PG39: SEC_BOOT 从 bit226 开始, 长度为 8bit, 烧写值为 0x42

2. 烧写文件生成

使用如下编译命令才会生成 efuse_cfg.bin, 并打包到最终的 frwpkg 镜像包中:

- 编译完整功能版本 (注: 此版本包含产测镜像, 烧录后首先进入产测模式)

```
./build.py -c ws63-liteos-app -def=PACKET_MFG_BIN
```

- 编译 APP 版本, 增加 efuse 参数:

```
./build.py -c ws63-liteos-app -def=SUPPORT_EFUSE
```

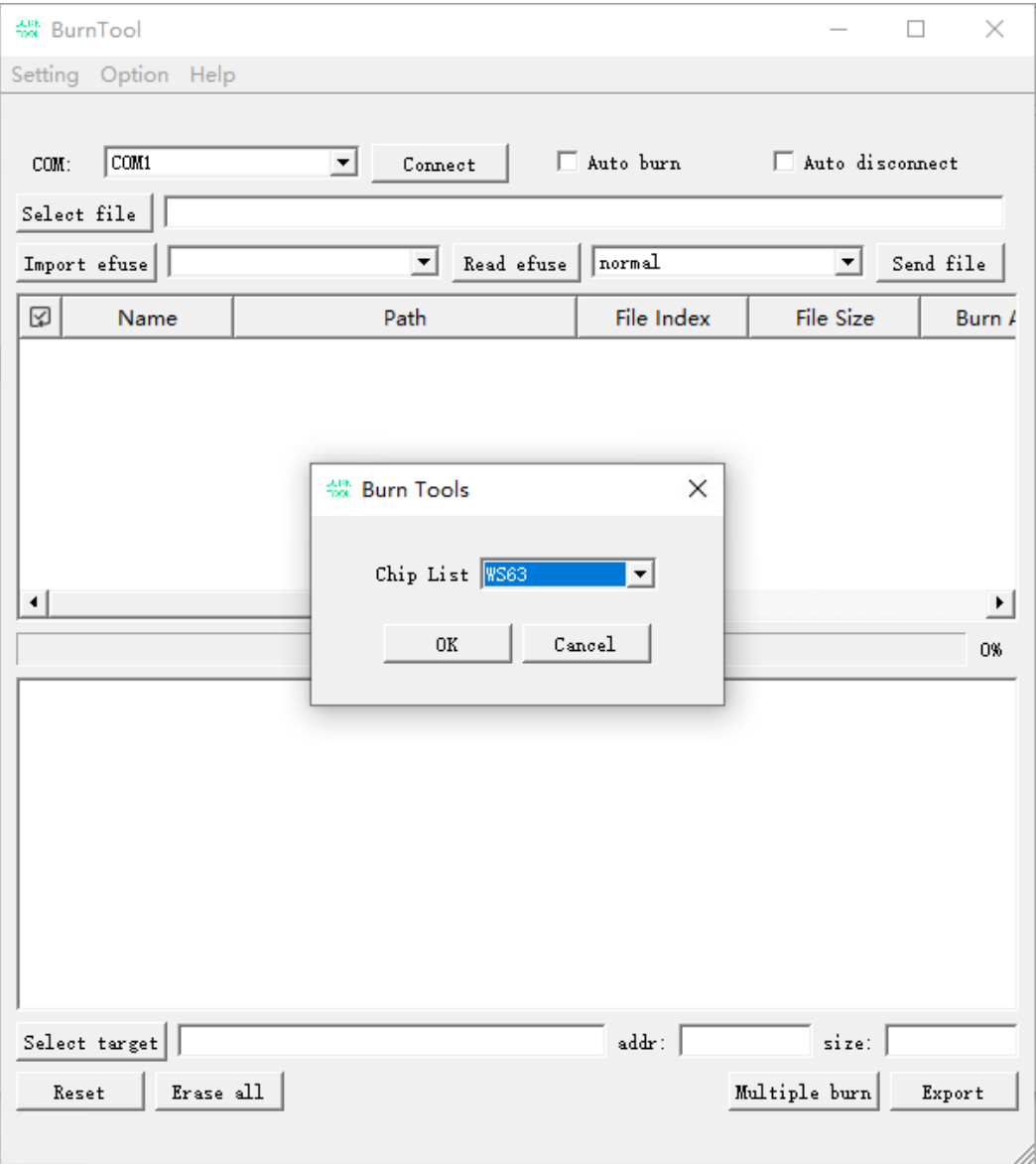
注意: 执行 ./build.py -c ws63-liteos-app, 未增加 -def=SUPPORT_EFUSE 参数, efuse_cfg.bin 不被打包到 frwpkg 中

3.2 烧录流程

准备烧写工具 “BurnTool” 通过 BurnTool 工具烧写镜像。具体步骤如下:

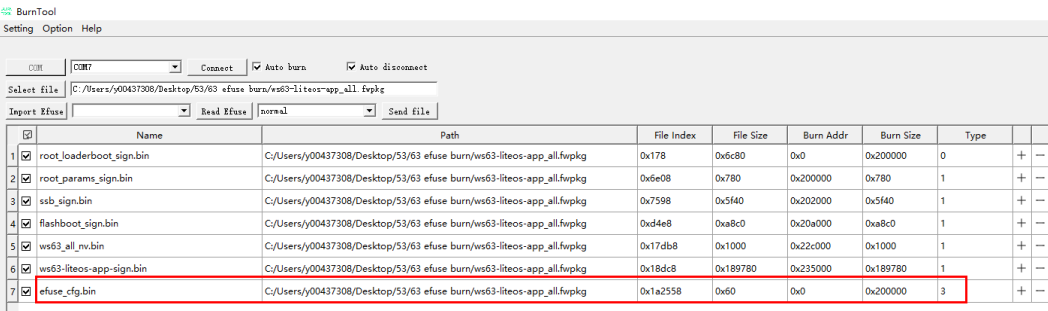
- 步骤 1 在 BurnTool 界面中, 单击 “Option” 按钮, 选择 “Change chip”, 从 “Chip List” 下拉菜单中选择 “WS63”, 并单击 “OK” 即可, 如图 3-1 所示。

图3-1 BurnTool 更换芯片示例



步骤 2 在 BurnTool 界面中，单击“COM”按钮选择 PC 机串口（串口选择，请参考开发板使用指南）；单击“Select file”按钮，选择各产品编译生成的固件包，并单击“OK”，如图 3-2 所示。efuse_cfg.bin 为待烧录数据，烧录类型为 3。

图3-2 烧录文件选择示例



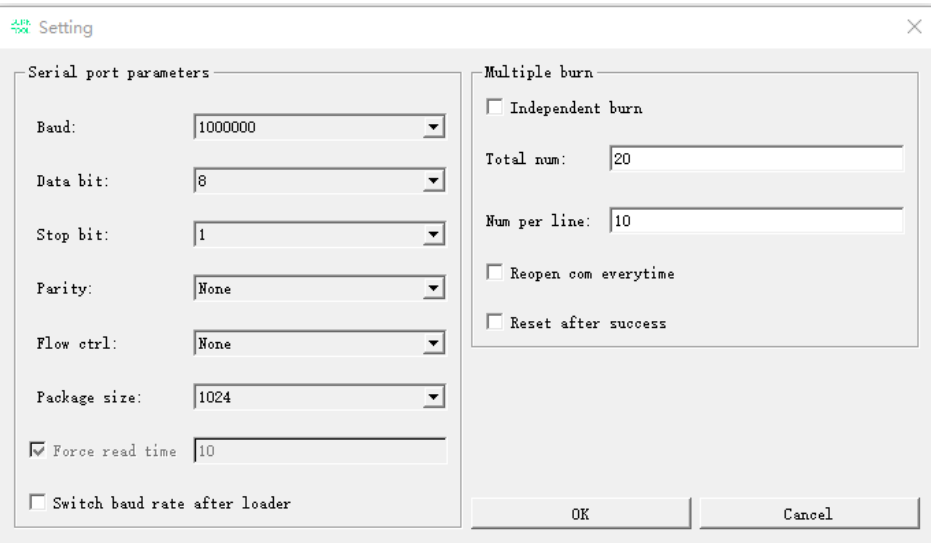
步骤 3 勾选 “Auto burn” 以及 “Auto disconnect” 选项；

选择 “Setting” → “Settings”，配置串口参数，默认配置如图 3-3 所示，baud 配置为 1000000。

说明

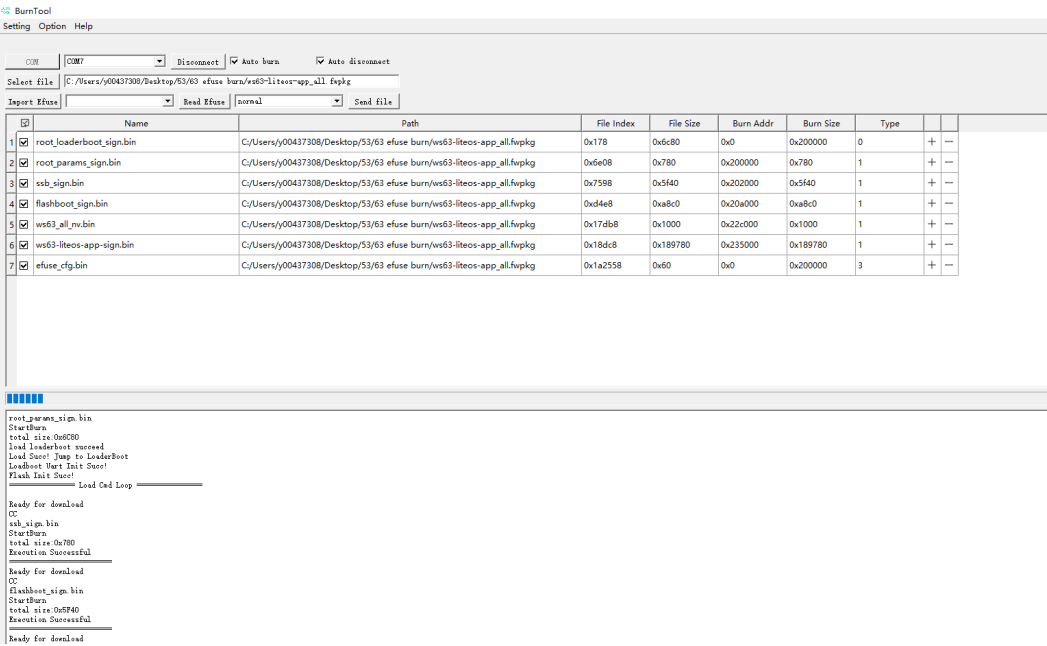
Force Read Time：定时读取的时间，以毫秒为单位。勾选时为定时读取串口，不勾选时为事件触发读取串口。适用于不勾选该选项无法正常烧录的场景。

图3-3 串口设置示例



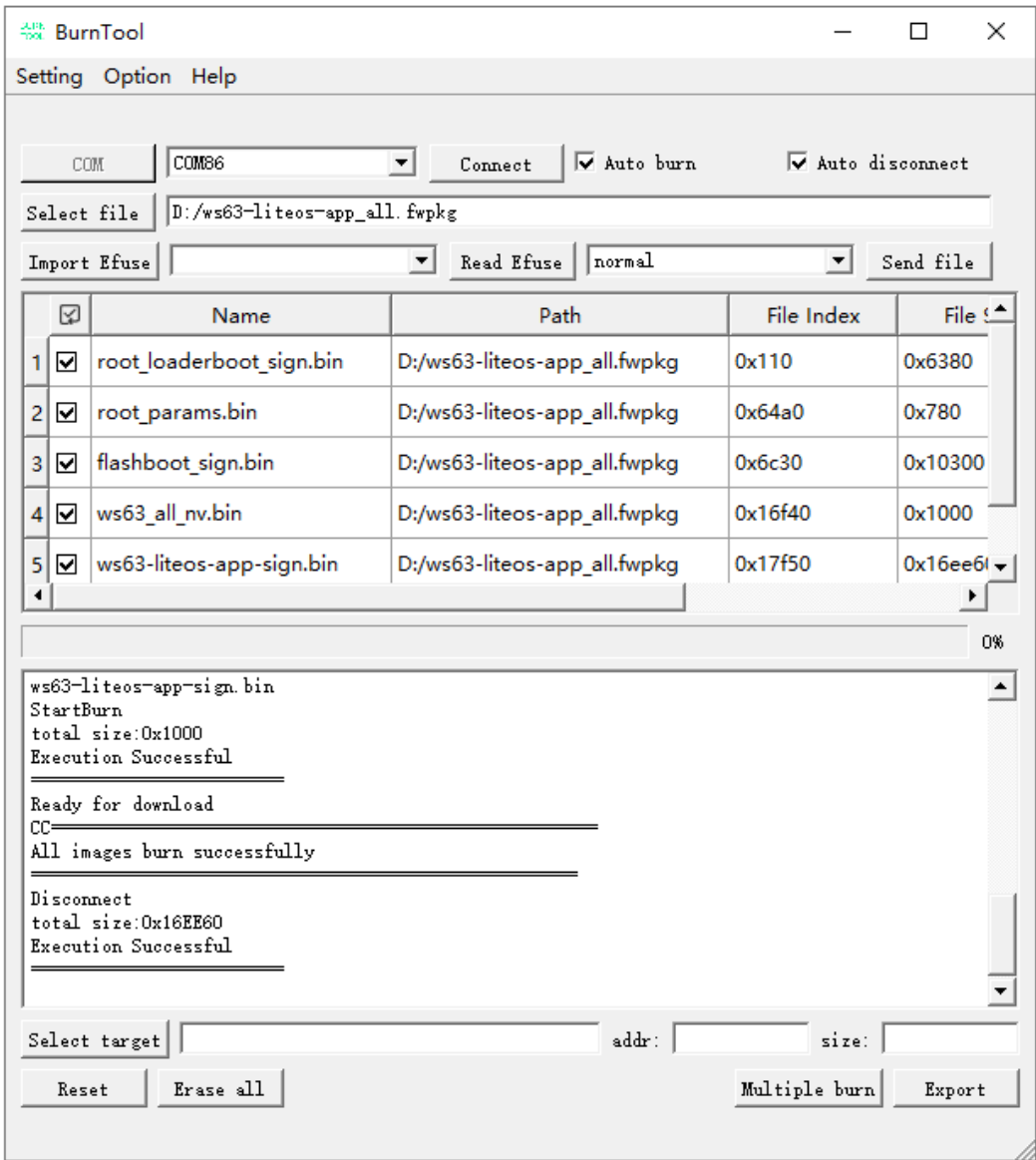
选择目标串口号并单击 “Connect” 按钮（单击后 “Connect” 变为 “Disconnect”），复位单板。自动烧录效果如图 3-4 所示。

图3-4 自动烧录示意图



等待传输完成后结束烧写，烧写完成会出现 “All images burn successfully”。烧写完成效果如图 3-5 所示。

图3-5 烧写完成示意图



说明

在速率不理想的外部状态下，若多次出现烧写镜像失败的情况，请拷贝至本地烧写

---结束