# **TG7100C** 量产指导

# 烧录篇

版本*: 1.1*

# 目录

# 1 概述

本文主要介绍 TG7100C 在量产烧录环节中，如何生成量产烧录固件并使用烧录工具进行量产烧录。如果需要在量产环节做 RF 产测，请参考《TG7100C 量产指导-RF 产测篇》。总的来说，TG7100C 量产准备工作如下：

- 准备烧录固件
- 准备烧录模组或者芯片
- 准备烧写板
- 准备烧录软件

# 2 准备烧录固件

关键字：whole_img.pack, whole_flash_data.bin, efusedata.bin, efusedata_mask.bin

## 2.1 量产固件的类型

为了和量产紧密配合，开发工具 AliGenieFlashTool 具备生成量产烧录包的功能。

开发人员根据需求配置完成后，点击 Create&Download 按钮，即可在工具根目录下对应芯片的 create 文件夹，生成 whole_img.pack 和 whole_flash_data.bin。

以 TG7100C 芯片开发为例，其生成目录是 chips/tg7100c/img_create_iot。AliGenieFlashTool 具体使用说明文档在 AliGenieFlashTool 的 docs 目录下。

1. whole_flash_data.bin 文件是二进制文件，按照分区表，排布了所有要烧录的镜像相关文件，其内容和 Flash 中数据布局完全一致，whole_flash_data.bin 的构成如图所示：


图 2.1: whole_flash_data.bin 构成

从图中可以看出，whole_flash_data.bin 包含了所有要烧录的 bin 文件，并且已经按照分区表位置排放好。该文件可以直接使用 Flash 编程器或者批量烧写工具的单文件模式烧录到 Flash 的 0 地址起始位置。

此文件包含了不同固件之间的 padding，导致文件较大，其缺点就是烧录时间长。

2. whole_img.pack 是一个压缩文件，它不仅包含了各种要烧录的文件，还包含了要烧录文件的配置信息，压缩包构成如下图所示：

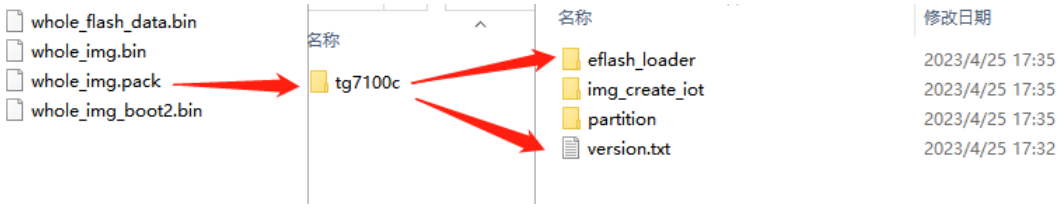


图 2.2: whole_img.pack 构成

在生产的时候，可以从量产烧录工具界面导入该开发包，量产工具会自行完成解压，并根据配置文件，分别烧录要烧写的文件，烧写速度快。

## 2.2 生成普通固件

对于普通固件，Dev Cube 界面按照如下设定，然点击 Create&Download 按钮即可。


图 2.3: 烧录配置界面

## 2.3 生成包含 MFG 的固件

如果需要在生产阶段进行 RF 产测，则使用 AliGenieFlashTool 烧录时需勾选”MFG Bin” 选项，并选择相应芯片目录下的产测固件。以 TG7100C 为例，其固件所在的目录是 tg7100c/builtin_imgs/mfg/tg7100c_mfg_v2.62/gu/mfg_gu_-25fcb4f72_40m.bin。该目录中存放了两类固件，一类是普通的固件 (没有 autoboot 后缀)，一类是烧录后自启动的固件 (带有 autoboot 后缀)。

自启动的 RF 测试固件在烧录完成后，会默认进入产测模式。产测完成后，芯片再次启动进入用户应用程序固件。

普通的 RF 测试固件在烧录完成，启动后默认进入用户应用程序固件。此时如需进入产测模式，则需要应用程序调用 IOT SDK 中的 API 实现，如果使用了 IOT SDK 的 CLI，则可直接在 CLI 输入 mfg 命令即可。

勾选 ”MFG Bin” 选项的界面示意图如下。点击 Create&Download 按钮，即可在生成的 whole_img.pack 和 whole_-flash_data.bin 中包含 RF 产测固件。

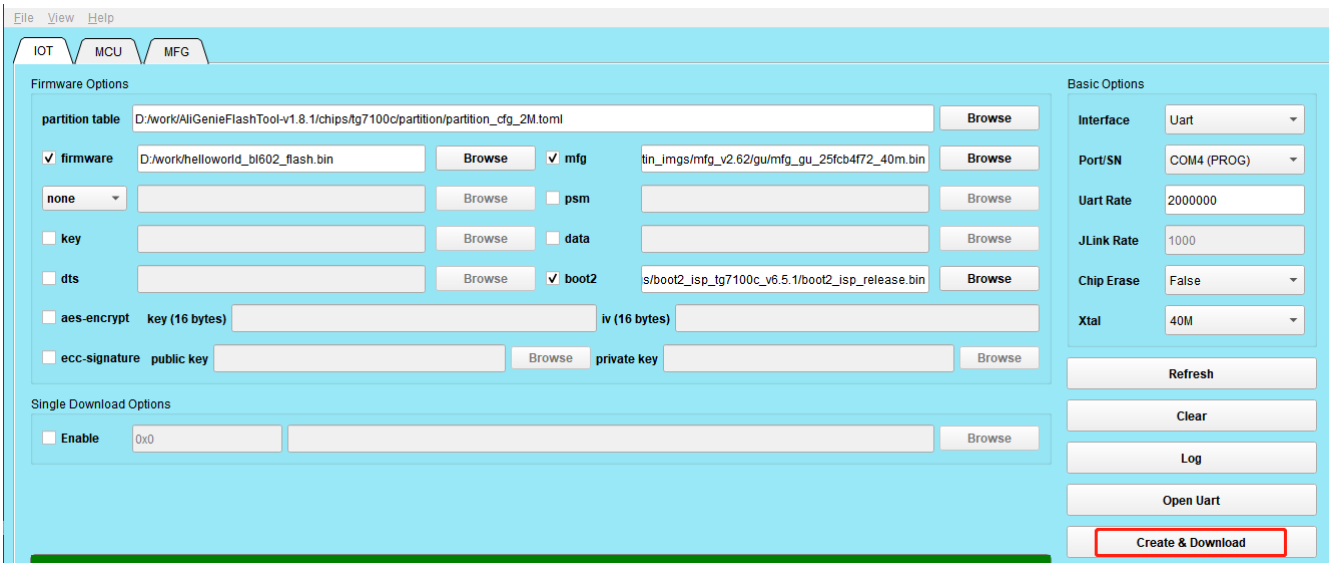


图 2.4: 烧录配置界面

## 2.4 生成包含 MFG 和 Media 的固件

如果在量产阶段需要烧录五元组信息，则使用 AliGenieFlashTool 烧录时需勾选 Media 选项，并选择 media.bin，这样生成的量产烧录包 whole_img.pack 中就包含需要烧录五元组文件的信息。

实际量产环节中，批量烧写工具在开始烧录之前，会先获取五元组信息，然后生成 media.bin 替换原有的文件，然后再烧录 Flash。

勾选 ”Media” 选项的界面示意图如下：

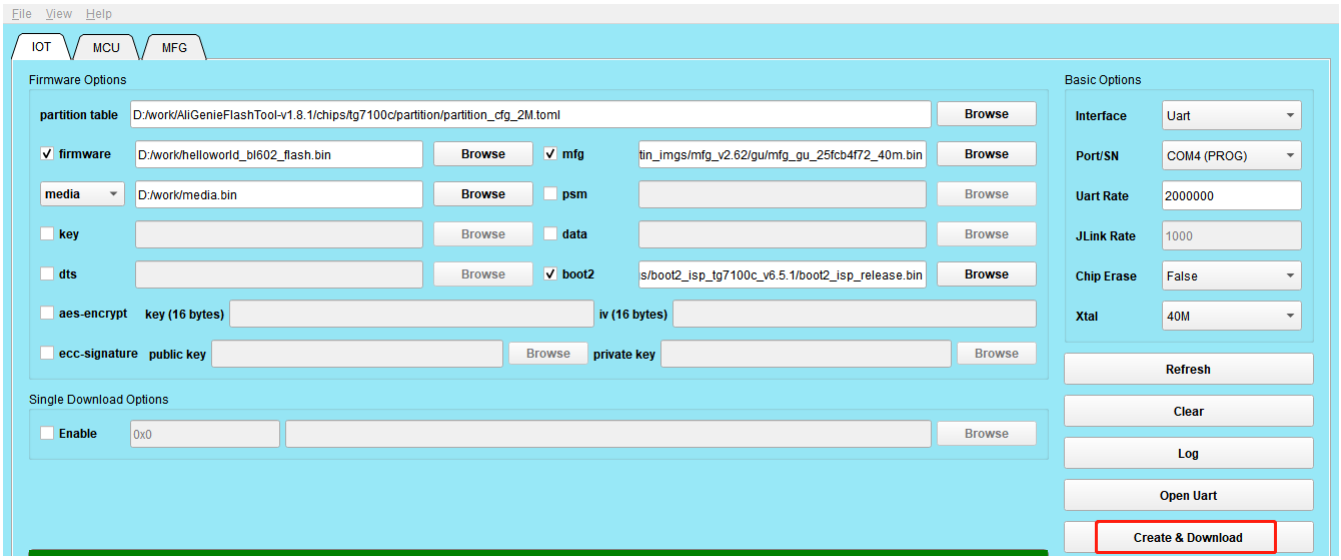


图 2.5: 烧录配置界面

---

注解：客户如果使用 whole_flash_data.bin 进行量产，需要根据实际情况，自行解决动态更改五元组信息的问题。

方法一：将 whole_flash_data.bin 中原始的五元组信息替换为当前五元组信息生成的 bin 文件。

方法二：烧录完 whole_flash_data.bin 后，再次将五元组信息 bin 文件烧录到 Flash, 从而将真正的五元组信息烧录到 Flash 中。

---

## 2.5 生成加密的固件

如果用户需要加密固件，则使用 AliGenieFlashTool 烧录时，需勾选 AES-Encrypt 并填写相应的 AES Key 和 IV。在生成加密镜像的同时，也会生成包含对应加密密钥的文件 efusedata.bin。

加密的密钥 (也是解密的密钥) 需要写入到芯片的 efuse 中，此时，whole_img.pack 已经包含要烧写的 efuse 密钥数据。


图 2.6: 烧录配置界面

---

注解：如果用户是使用 TG 的量产工具，通过导入 whole_img.pack 方式进行烧写，则无需额外操作，烧录工具会自动判断并将 efuse 烧写。

如果客户是自研整套工具，不是导入或者解压烧录包 whole_img.pack，而是直接拿到 whole_flash_data.bin 进行烧写 Flash，则此时需要将 efuse_bootheader 目录下的 efusedata.bin 烧录到 efuse 中，并根据 efusedata_mask.bin 进行回读校验。

---

# 3 准备烧录模组或者芯片

关键字：Boot 引脚 (GPIO8), Reset 引脚, RXD(GPIO7), TXD(GPIO16), VCC 电源，GND

烧录芯片或者模组需要引出 Boot 引脚，Reset 引脚，TXD，RXD 以及电源。在烧录过程中，这些引脚的作用如下：

- Boot 引脚：控制芯片启动方式。引脚为高电平时，芯片从 UART/SDIO 启动。引脚为低电平时，芯片从 Flash 启动。
  - 烧写 Flash/Efuse 时，Boot 引脚需要是高电平
  - 进行产测时，Boot 引脚需要是低电平
  - 如果是 SDIO 透传模组，没有 Flash，则 Boot 引脚可以固定为高电平
- Reset 引脚：控制芯片复位。在烧录或者产测开始之前，通过 Boot 引脚和 Reset 引脚，复位芯片，确保芯片进入正常的启动模式。防止因为上电时序或者供电问题导致的芯片启动不确定性。
- TXD RXD 引脚：用于 UART 烧录通信。

# 4 准备烧写板

为了达到更稳定的烧写效果，我们建议使用 USB 转串口烧写板 (带有 DTR 和 RTS 引脚)。通过 USB 转串口的 DTR 控制 Boot 引脚，RTS 控制 Reset 引脚，在烧录之前对芯片启动方式进行一次控制，使得烧录过程更稳定可靠。

芯片上电时，会根据 bootpin 引脚的电平高低决定启动方式。当 bootpin 引脚电平为低时，芯片会从 flash 启动，当 bootpin 引脚电平为高时，芯片会从 uart 启动，进入烧写模式，工具可以通过 uart 与芯片通信完成烧录流程。据此，博流提供的烧写工具会在开始烧写流程之前，通过控制 uart 的 DTR 和 RTS 引脚（先控制 DTR 将芯片的 bootpin 引脚电平拉高，再控制 RTS 将芯片重新 reset），自动控制芯片进入烧写模式，从而实现芯片自动烧写。

烧写板连接要求：

- VDD_OUT：3.3V, 与芯片/目标板 VCC 电源相连
- GND：与芯片/目标板 GND 相连
- UART_TXD：与芯片/目标板 RXD 引脚相连
- UART_RXD：与芯片/目标板 TXD 引脚相连
- UART_RTS：与芯片/目标板 RST 引脚相连，用于控制芯片的 Reset
- UART_DTR：与芯片/目标板 bootpin 引脚相连，用于控制芯片从 UART 或者 Flash 启动

# 5 准备烧录软件

## 5.1 使用量产烧录工具

批量烧录工具，支持 1 托 12 烧写、支持串口和 Jlink 两种通信方式、支持通信速度可配置。工具有三种烧录模式，用户可以根据实际需要进行选择。

### 5.1.1 单文件模式烧写

- 支持烧写 flash 和 efuse，根据实际需求配置烧录方式
  - 烧写 flash 时，选择需要烧录的 bin 文件以及填写对应的烧录地址
  - 烧写 efuse 时，只需选择对应的烧写文件
  - flash 和 efuse 都需烧写时，先烧写 flash，再烧写 efuse
- 选择的烧录文件可以是 AliGenieFlashTool 工具生成的 whole_flash_data.bin 和 efusedata.bin，也可以是用户自己需要烧写的文件
- 由《准备烧录固件》章节可知，单文件模式烧录 whole_flash_data.bin 的缺点是烧录文件大，速度慢，优点是简单灵活

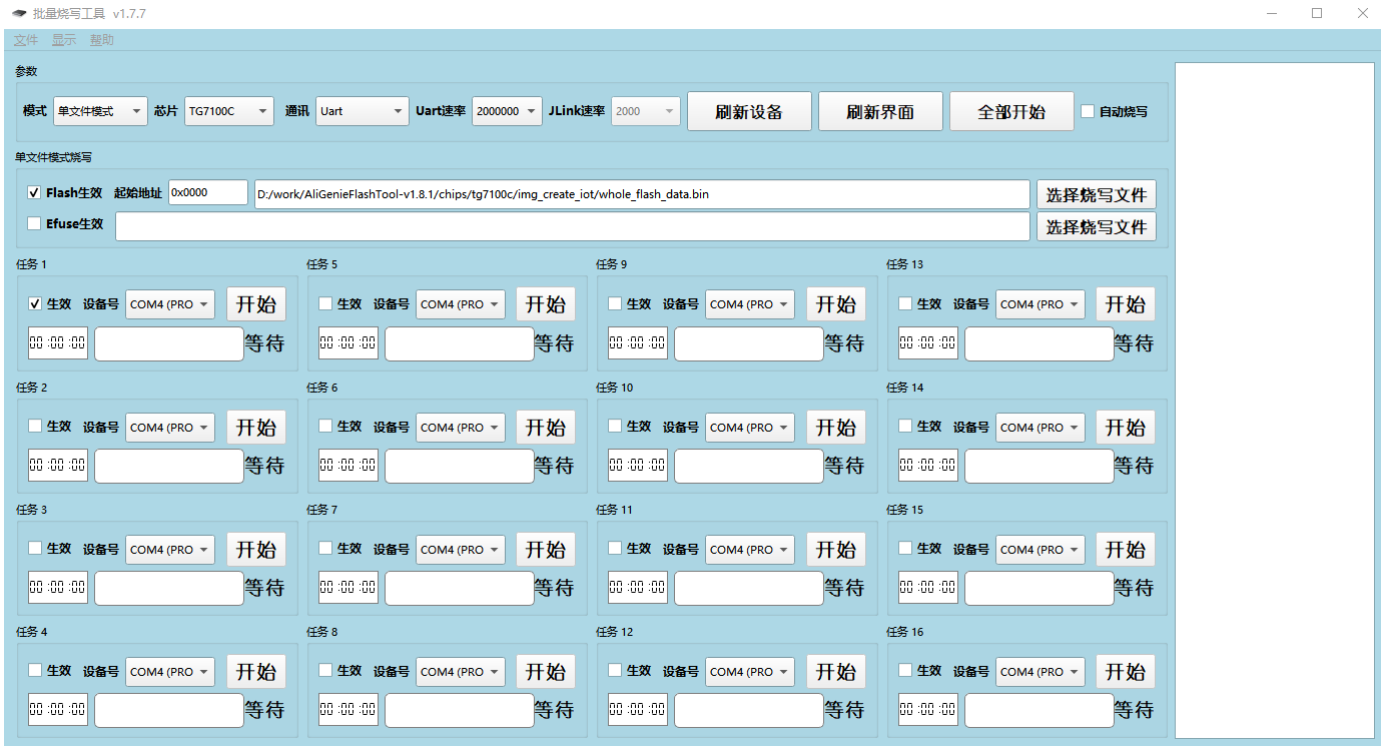


图 5.1: 单文件模式烧录

### 5.1.2 开发模式烧写

开发模式解决了单文件模式烧写速度慢的问题。它是根据 AliGenieFlashTool 生成的量产烧录包 whole_img.pack，按照烧录文件和对应的烧录地址，逐个进行烧录。烧写速度快，而且不需要客户额外配置，烧录包中包含了烧录所需要的各种文件和配置，只需导入烧录包 whole_img.pack 即可。因为烧录包中包含了烧录所需要的各种文件和配置。

- 配置烧录方式
- 导入 AliGenieFlashTool 工具生成的 whole_img.pack
- 选择需要烧写的设备号，点击“全部开始”按钮启动烧录，烧录工具根据烧录包中的烧录地址，逐个烧录文件

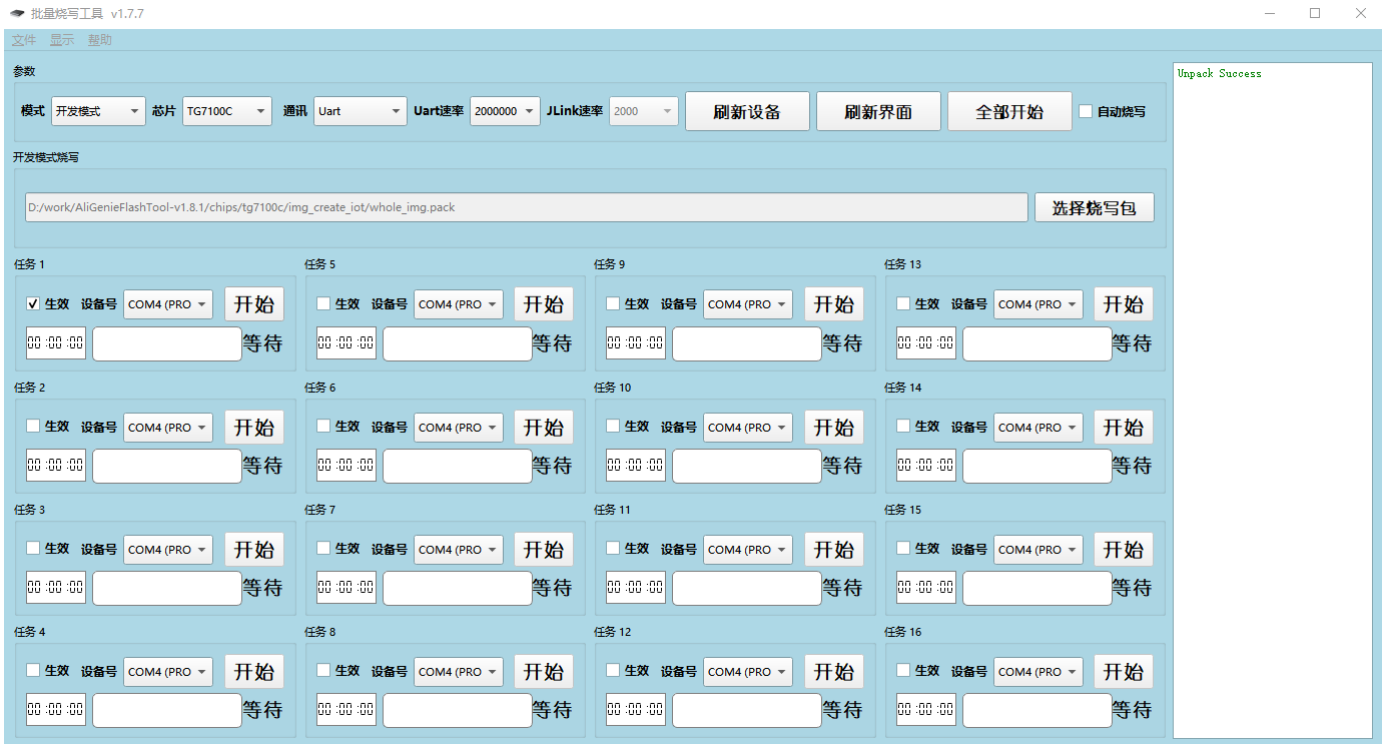


图 5.2: 开发模式烧录

### 5.1.3 动态模式烧写 (适用于预烧录五元组信息)

开发模式烧写具有操作简单的优点，若客户需要烧录三 (五) 元组信息，或者需要对镜像按照一机一密这种方式进行加密，那么每次烧录的文件 (或者部分文件) 都是不同的，需要动态进行改变，则使用动态模式烧写。

- 选择动态模式烧写时，烧写工具会自动打开一个动态服务程序，该程序用于动态修改要烧写的文件 (该程序位于工具目录 file_service 文件夹下)
- 导入 AliGenieFlashTool 工具生成的 whole_img.pack 烧写文件包
- 导入 CSV 文件
- 选择需要烧写的设备号，点击全部开始按钮开始烧录
- 在烧录前，批量烧写工具的每个任务会和动态服务程序通信，发送生成动态文件所需的信息 (TaskID、ChipID 等) 给动态服务程序，动态服务程序将结果反馈给批量烧写工具，批量烧写工具判断文件正确后才开始烧录


图 5.3: 动态模式烧录

以烧录五元组为例，在 AliGenieFlashTool 生成量产烧录包 whole_img.pack 时，需勾选 Media 选项，生成的 pack 中包含要烧录的 media.bin。

实际烧录时，烧录任务会先发送信息给动态服务程序，动态服务程序从包含五元组信息的 excel 文件中，获取一组未使用的五元组，生成新的 media.bin，替换原有的 media.bin，然后将结果返回给烧写任务。烧写任务在获得正确的结果后进行烧录，这样烧写工具就可以将真正包含五元组信息的 media.bin 文件烧录到 Flash。

## 5.2 自研烧写工具

如果客户是自研整套工具，则可以根据在线编程协议 (ISP)，通过 UART 接口，将程序下载到 RAM 运行。然后再次和应用程序通信，完成对 Flash 和 efuse 的烧写，或者实现其它的功能。