

## HPM BootROM OTA 操作说明

1.BootROM OTA 大致介绍	4
2.XPI NOR 启动镜像布局	5
3.BootROM OTA 操作流程	5
3.1 FLASH 分区规划, 修改 SEC_IMG_OFFSET.	5
3.2 构建工程	6
3.3 编译固件	6
3.4 HPMicro_Manufacturing_Tool 编辑烧录固件	8
3.4.1 编辑固件	8
3.4.2 烧录固件	12
3.4.3 批量烧写	15
3.5 OTA 升级操作	16
3.5.1 镜像编辑生成	16
3.5.2 OTA 包制作生成	17
3.5.3 OTA 升级	17
4.量产批量烧录	18
5.注意事项	20

版本:

日期	版本号	说明
2024-6-24	2.0	第二版, 基于 HPMicro_Manufacturing_Tool 工具

## 1.BootROM OTA 大致介绍

HPM 全系列 BootROM 支持高效的 OTA 方案;必须要说明的是,此方案只限 XPI NOR 启动模式。BootROM 在启动时,判断 OTP 中的 SEC\_IMG\_OFFSET 是否为 0,如果为 0,说明只有一份镜像;如果不为 0,则说明存在第二份镜像。BootROM 通过对比两份镜像中的 SW\_VERSION 大小,选择 SW\_VERSION 更大的一份作为最新的 IMAGE 执行,若两份 SW\_VERSION 相等,则认为第一份为最新的 IMAGE 执行。

## 2.XPI NOR 启动镜像布局

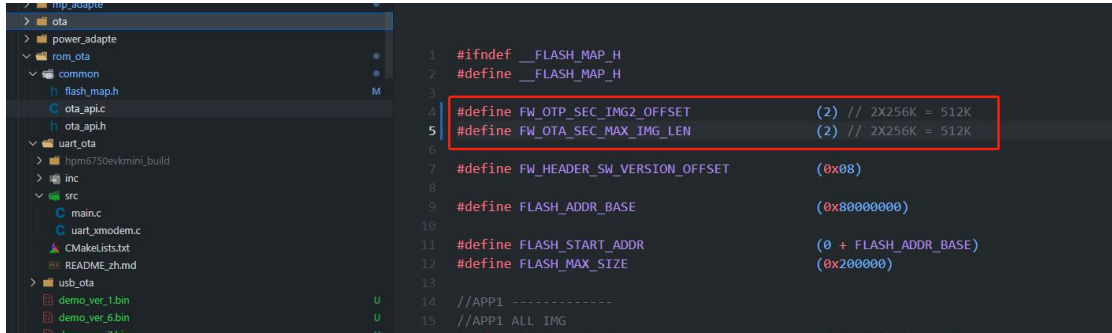


## 3.BootROM OTA 操作流程

### 3.1 FLASH 分区规划, 修改 SEC\_IMG\_OFFSET.

用户根据自己的项目 FLASH 分区规划，调整 SEC\_IMG\_OFFSET。设置最大固件大小为 SEC\_IMG\_OFFSET \* 256k。如 SEC\_IMG\_OFFSET=2,则最大固件大小 512K。

参考 demo 中的 flash\_map.h 中的，SEC\_IMG\_OFFSET 对应修改如下：

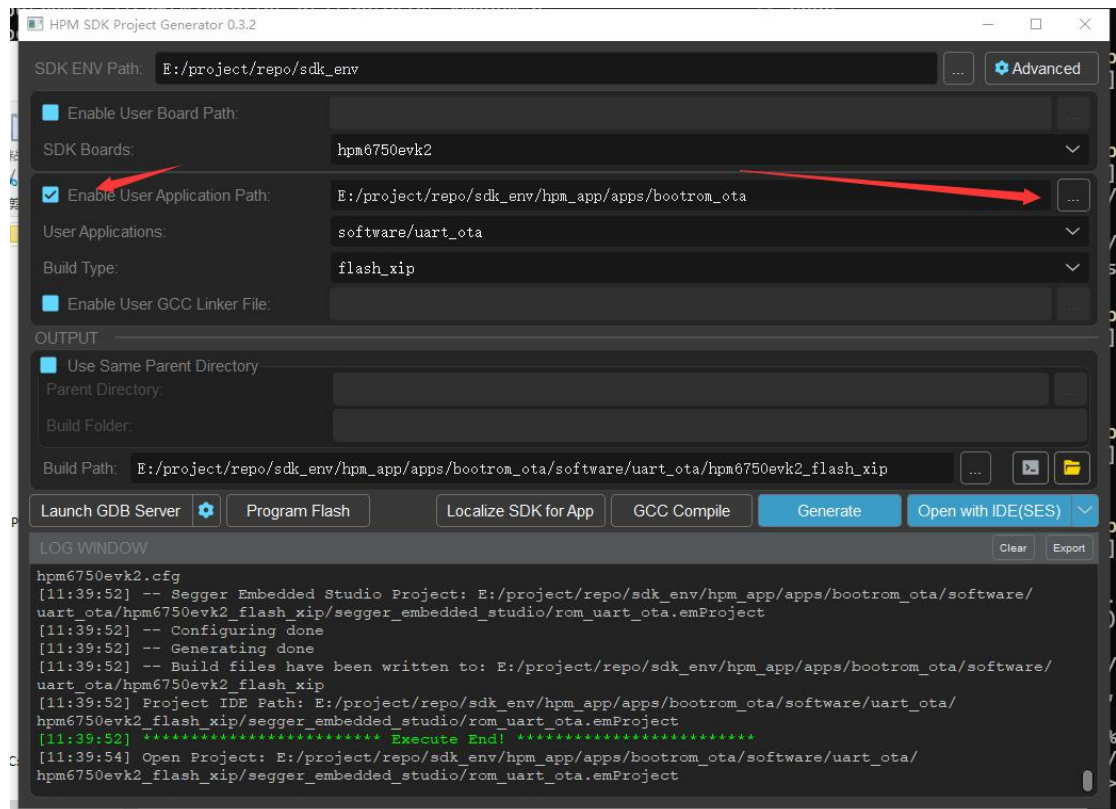


```

1  #ifndef __FLASH_MAP_H
2  #define __FLASH_MAP_H
3
4  #define FW_OTP_SEC_IMG2_OFFSET (2) // 2X256K = 512K
5  #define FW_OTA_SEC_MAX_IMG_LEN (2) // 2X256K = 512K
6
7  #define FW_HEADER_SW_VERSION_OFFSET (0x08)
8
9  #define FLASH_ADDR_BASE (0x8000000)
10
11 #define FLASH_START_ADDR (0 + FLASH_ADDR_BASE)
12 #define FLASH_MAX_SIZE (0x200000)
13
14 //APP1 -----
15 //APP1 ALL IMG

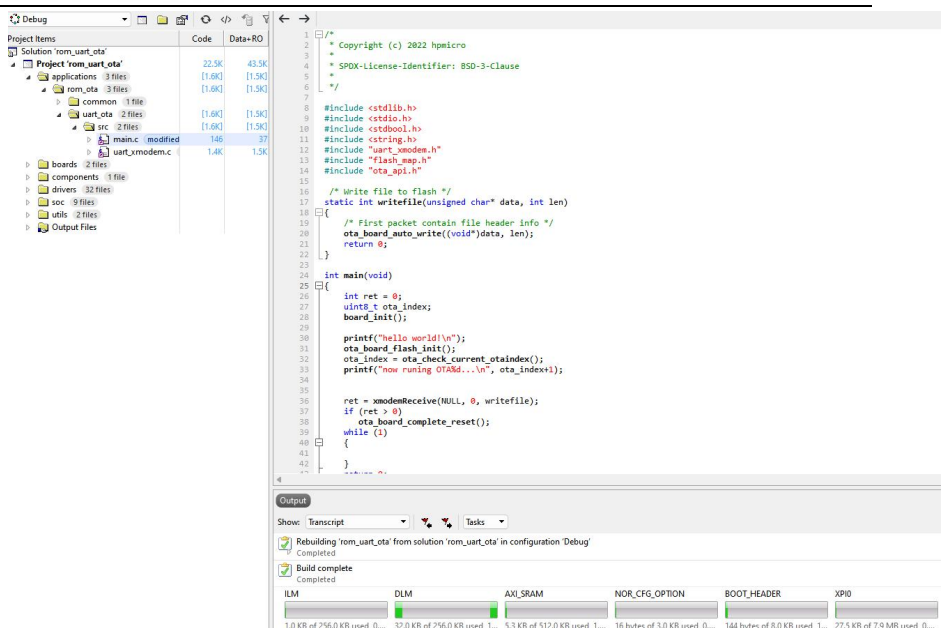
```

## 3.2 构建工程



当前使用的 linker 文件是官方默认的，可根据自己实际项目去调整。

## 3.3 编译固件



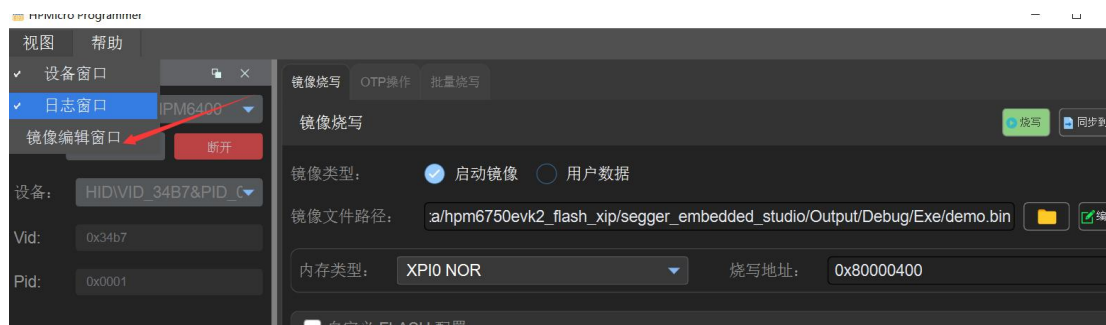
« segger\_embedded\_studio » Output » Debug » Exe

名称	修改日期	类型	大小
demo.bin	2023-03-10 15:12	BIN 文件	
demo.elf	2023-03-10 15:12	ELF 文件	17
demo.ind	2023-03-10 15:12	IND 文件	
demo.map	2023-03-10 15:12	Linker Address ...	

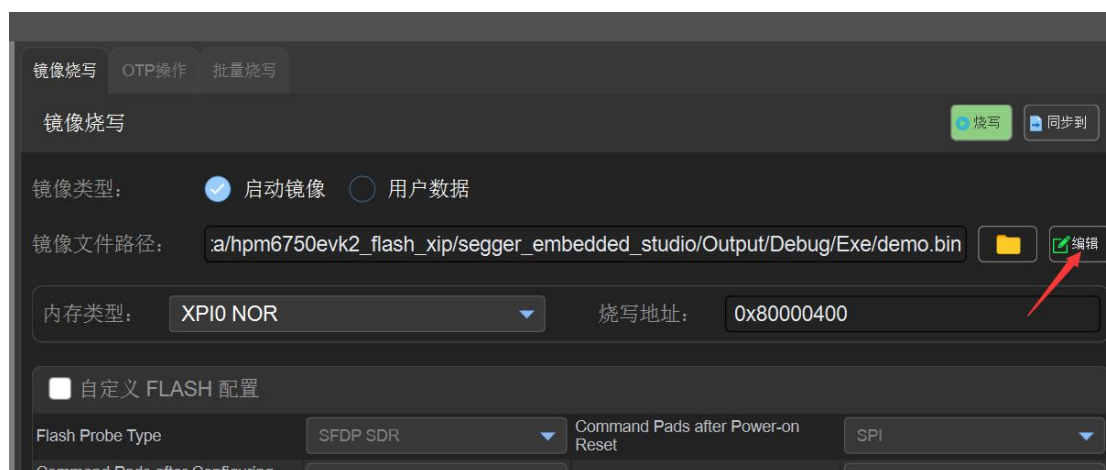
### 3.4 HPMicro\_Manufacturing\_Tool 编辑烧录固件

当前使用的 mfg tool 版本为 0.3.3，编辑固件可从两个入口进入，分别是：

a. 视图->镜像编辑窗口 (如果是单纯的编辑固件，可从此进入)



b. 镜像烧写窗口->编辑 (如果既要编辑又要烧录固件，从此进入)



#### 3.4.1 编辑固件

固件编辑窗口布局和镜像分区布局一致，每个分区的详细介绍可从手册中查阅。

(以下从镜像烧录窗口->编辑，进入编辑固件窗口)





EXIP BLOB: 如果要加密固件(密文固件), 启用 EXIP BLOB, 设置相关参数。

XPI Flash 配置选项: XIP Flash 启动, 启用此选项, 设置 Flash 相关参数。

如果是官方默认的 flash\_xip 类型及 linker 文件, 在固件头部已经包含, 可不设置。

固件容器头: 固件容器头参数设置。

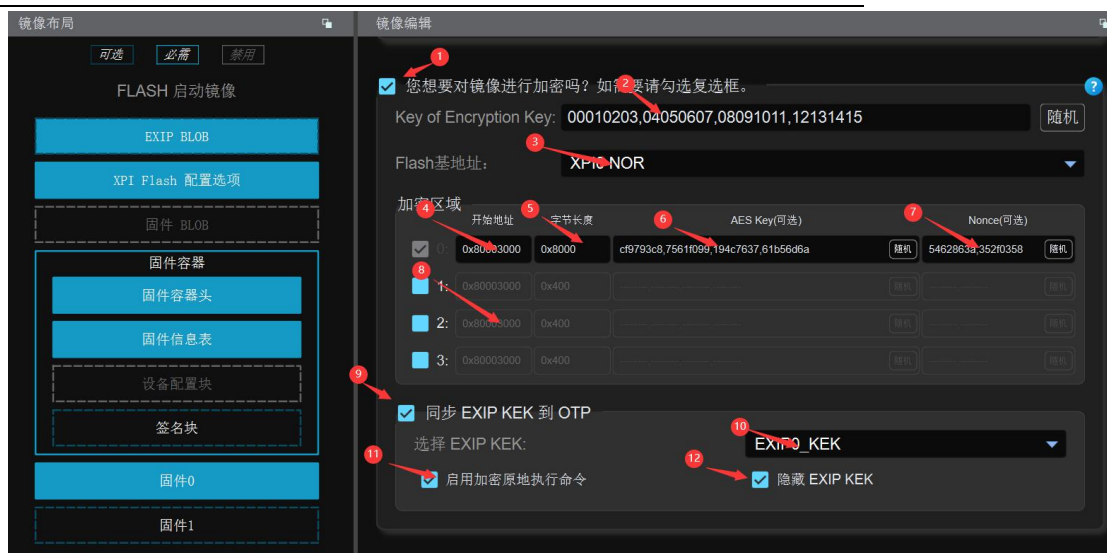
固件信息表: 固件信息表参数设置。

固件 0: 固件 0 参数设置。

生成: 生成编辑后的固件。

## A. EXIP BLOB 介绍

如果要使用 exip 加密固件, 启用 EXIP BLOB。



1. 启用 EXIP BLOB
2. KEK 密钥, 32 位 16 进制数
3. Flash 基地址: 默认 XIP0 NOR
4. 加密区域 1 开始地址, 注意: 启动镜像的头部可不被加密, 否则无法启动。如果是默认的 flash\_xip/flash\_sdram\_xip linker 文件, 默认固件开始地址为: 0x80003000
5. 加密区域 1 的加密字节长度。  
例如:  $2(\text{SEC\_IMG\_OFFSET}) \times 256\text{k} = 512\text{k} - 0\text{x}3000 = 0\text{x}7\text{d}00$ ; 由于需要 1K 对齐, 所以我们修改为 0x8000
6. 加密 AES key 设置(可选), 如果不手动设置, 建议点击随机, 安全级别高
7. 加密 AES Nonce 设置(可选), 如果不手动设置, 建议点击随机, 安全级别高
8. 其它加密区域设置。最大支持 4 个区域加密。设置同加密区域 1 类似
9. 同步 EXIP KEK 到 OTP。如果要同时写入 OTP EXIP KEK 等信息, 启用此选项即可。
10. 选择 EXIP KEK 是 EXIP0 或 EXIP1, 根据自己的硬件使用情况来选择
11. 启用加密原地执行命令; flash\_xip/flash\_sdram\_xip 必须启用此选项
12. 隐藏 EXIP KEK。启用此选项后, 烧录到 OTP 中的 KEK 信息将不可读。  
为了安全, 烧录了 KEK 后, 在出厂时一定要隐藏 EXIP KEK。

## B. 其他配置

已检测到固件信息并已自动添加到配置中！

固件类型: **XiP** Hash 类型: **None**

固件首地址相对于容器首地址的偏移量: **0x2000**

加载地址: **0x80003000** 入口点地址: **0x80003000**

已检测到 XPI Flash 选项并已自动添加到配置中！

典型 Flash Word 配置: **Channel A, Pin group0** 地址: **0x0000**

Flash Word: **0xfcf90002,0x00000000,0x00000000,0x00000000**

已检测到偏移量并已自动添加到配置中！

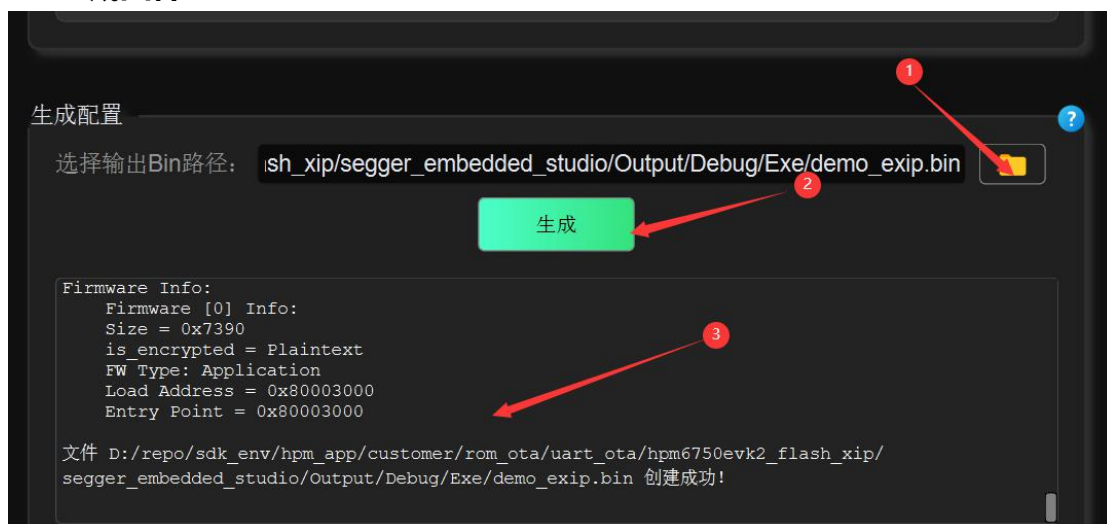
固件容器首地址相对于镜像地址偏移量: **0x0c00**

软件版本(可选): **0** FUSE 版本(可选): **0**

1. Hash 类型选择，固件签名 hash 类型选择。如果用户要 OTA 固件，建议启用其中一项，这样可确保固件完整性及合法性校验。
2. 固件信息表基于镜像的偏移 offset。如果是 RAM Offset，默认 auto 即可。如果是 XIP Offset,偏移值为固件首地址减去容器首地址。如果使用默认的 flash\_xip/flash\_sdram\_xip linker 文件，默认固件首地址:0x3000-容器首地址 0x1000 = 0x2000
3. XPI Flash 配置在镜像中的地址，如果使用默认的 flash\_xip/flash\_sdram\_xip linker 文件，默认此地址为：0x000（在我们界面，默认是从 0x80000400 开始烧录，由于默认没有添加 EXIP BLOB 功能，那么 xpi 配置默认也在 0x80000400，因此这边的地址是 0，当我们增加了 EXIP BLOB 并生成固件后，这边就会变成 0x400）
4. 固件容器头基于镜像的偏移 offset。如果使用默认的 flash\_xip/flash\_sdram\_xip linker 文件，默认 offset 为:0xc00（在我们界面，默认是从 0x80000400 开始烧录，由于默认没有添加 EXIP BLOB 功能，那么固件容器头默认在 0x80001000，因此这边的地址是 0xc00，当我们增加了 EXIP BLOB 并生成固件后，这边就会变成 0x1000）

- 软件版本号(可选), 如果启用了 SES\_IMAGE\_OFFSET, 可设置此软件版本号来设置当前固件版本号。  
其余参数默认即可。

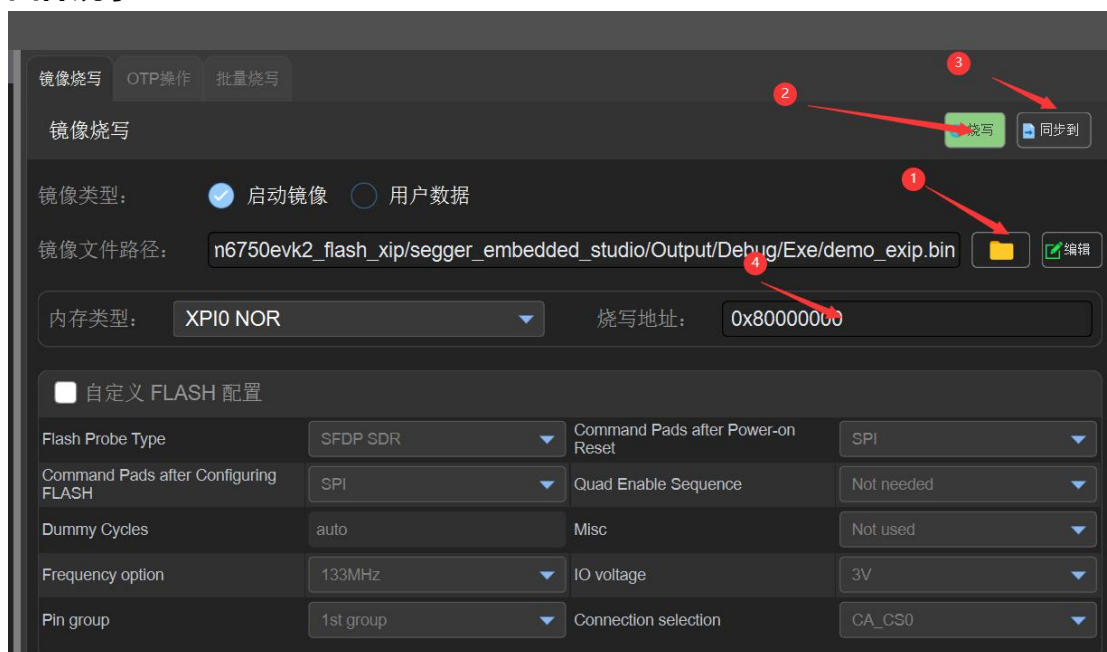
### C. 生成固件



- 选择生成固件的路径, 并保存, 这里保存为 demo\_exip
- 点击生成按钮, 即可生成固件。
- 生成固件的 log。

### 3.4.2 烧录固件

#### 固件烧录

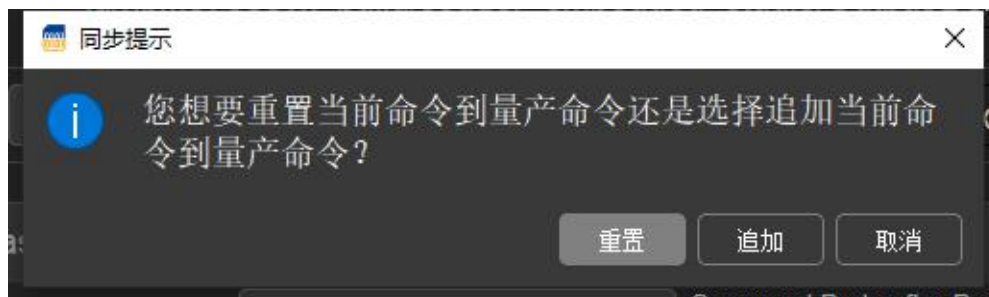


1. 烧录镜像路径。选择你要烧录的镜像。
2. 烧写固件。点击烧写即可。
3. 如果除了烧录镜像，还有其它 OTP 写入等操作，可同步到量产命令后，一次写入完成。
4. 这边可以看到烧写地址由 0x80000400 变更为了 0x80000000，此时，我们再点击编辑后，之前的偏移地址发生了改变

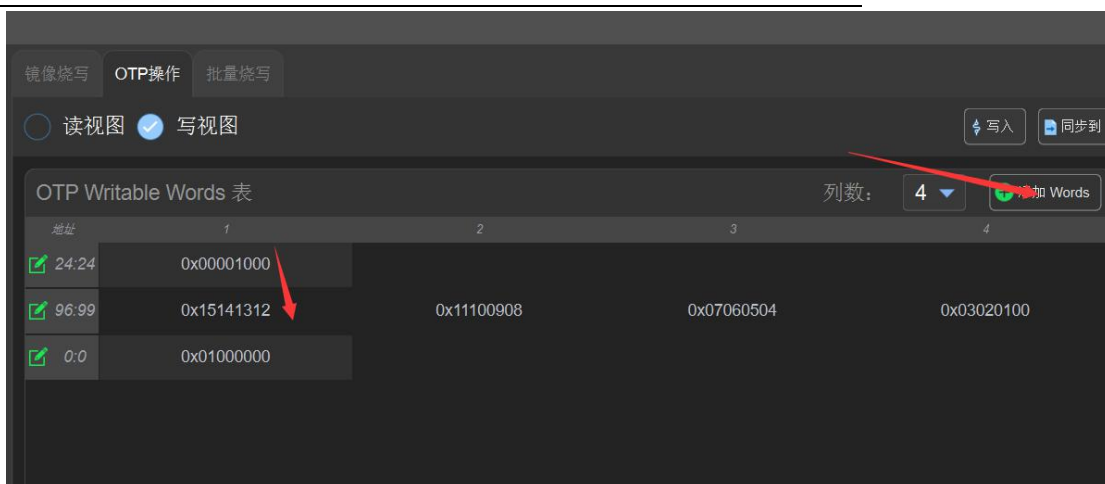


### OTP 操作

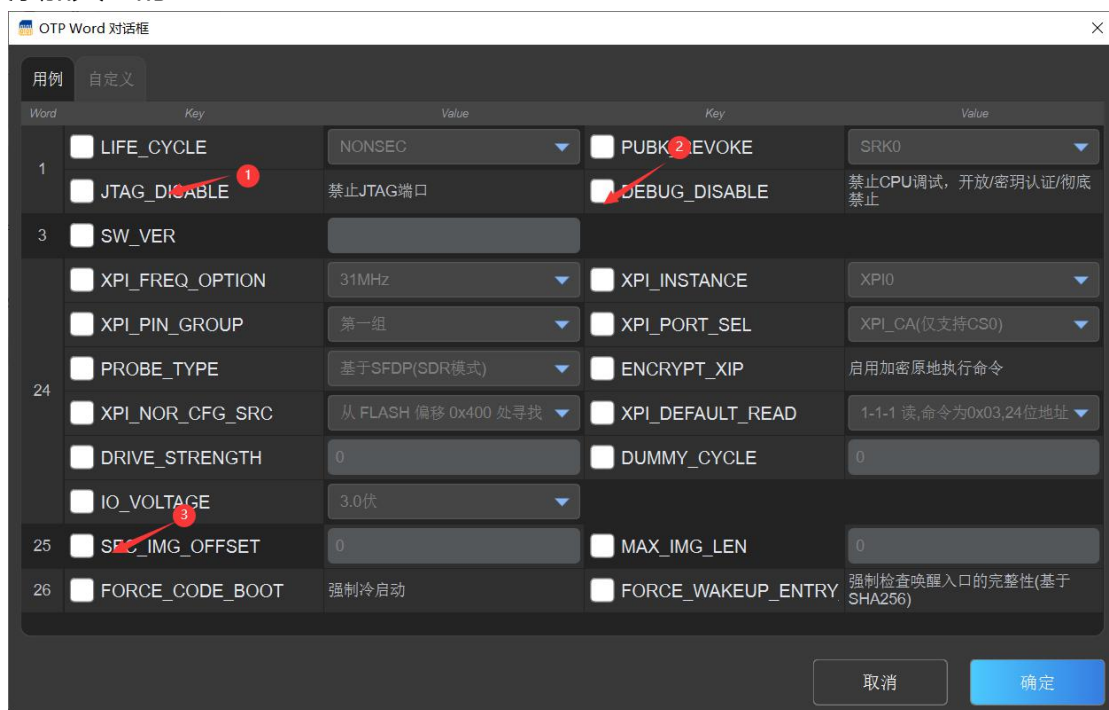
在上面基础上，我们点击同步，这里我先将固件烧录重置到量产命令中，接下来设置 OTP 命令。



切到 OTP 操作界面操作，并切到写试图。



1. 如果在镜像编辑界面下，开启了 EXIP\_BLOB，且开启了：同步 EXIP KEK 到 OTP。则在当前 OTP 写视图可看到对应的 OTP Word。具体含义查阅手册。
2. 添加其它的 OTP Word.



1. 禁止 JTAG 端口使能开关。
2. 禁止 DEBUG 使能开关。彻底禁止。
3. SEC\_IMG\_OFFSET，第二份镜像 offset。

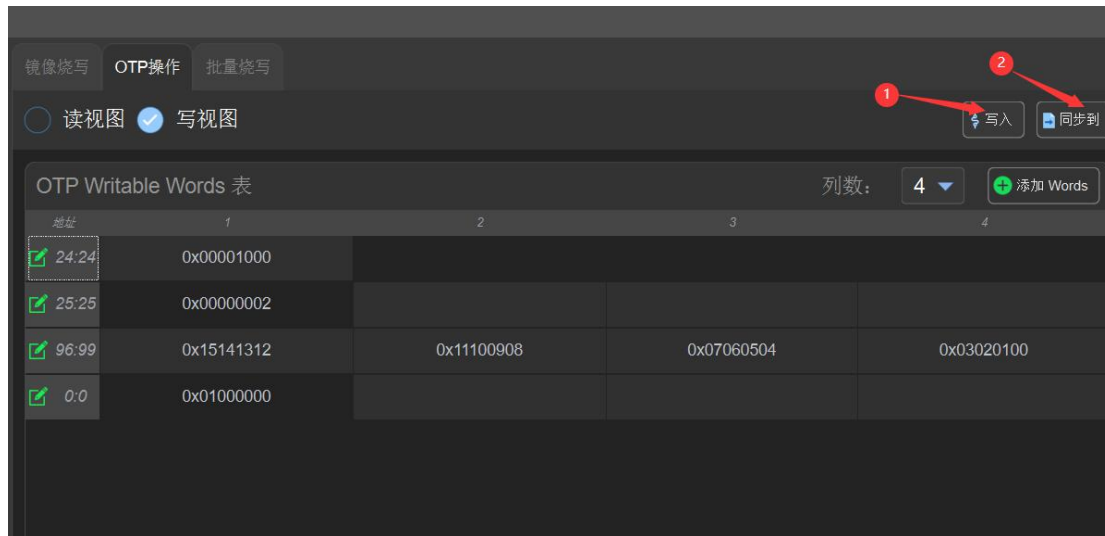
上图中 2 表示：2x256k=512k。

注意: OTP 操作不可逆, 写入之后无法修改, 一定要提前规划好自己的 FLASH 固件大小。谨慎操作。

注意：写入的 SEC\_IMG\_OFFSET 必须和 flash\_map.h 中的



FW\_OTP\_SEC\_IMG2\_OFFSET 相同。否则固件启动异常。

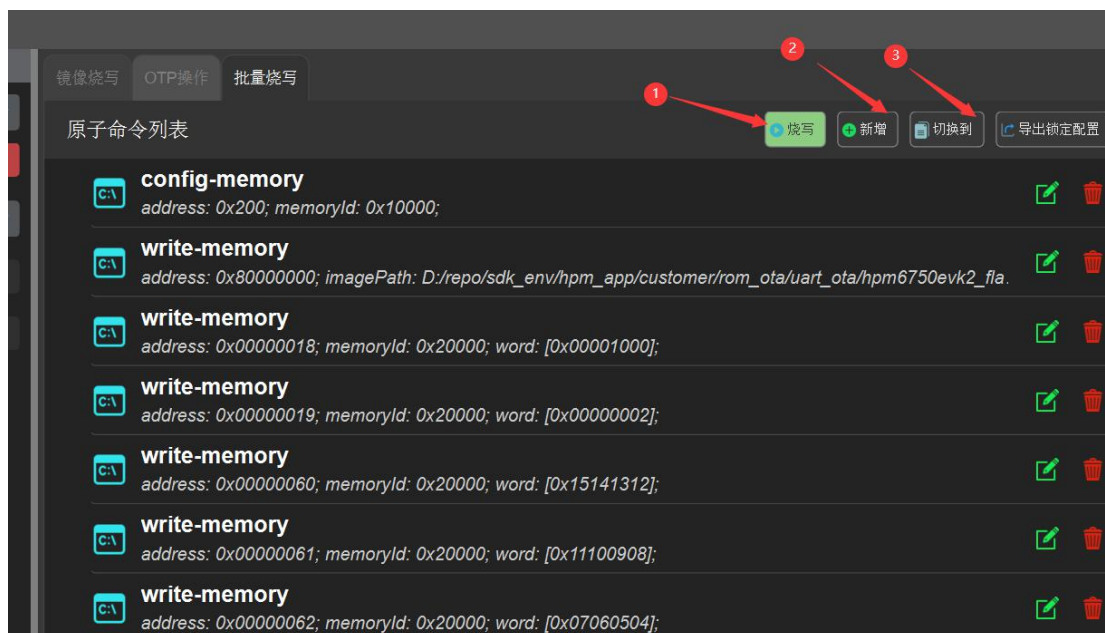


1. 设置 OTP 写入 word 后，可直接写入到芯片 OTP 中。
2. 同样，可以同步到量产命令中。

因为刚刚我已经同步了固件烧录的量产命令，在这里我选择追加 OTP 操作到量产命令中。

### 3.4.3 批量烧写

当固件及 OTP 都设置完成后，可通过批量烧录窗口一次性全部写入。



1. 批量烧录，一次性全部写入到设备中。点击烧写即可。

2. 新增批量烧录指令。
3. 切换到量产工具。当确认均设置完成，且验证通过后，可切换到量产工具批量生产。量产工具，可以一拖多的烧录，且是傻瓜式全自动的模式。


当我们烧录完成后，使用串口工具打开，显示如下，此时软件版本显示为 0，并且当前执行的是 APP1 IMG。当前测试固件支持 UART 和 USB OTA 功能，方便我们后续使用 UART 或者 USB 进行升级固件。

```

xpi1:      400000000Hz
femc:      166666666Hz
display:    74250000Hz
cam0:      59400000Hz
cam1:      59400000Hz
jpeg:      200000000Hz
pdma:      200000000Hz
=====
hpm_sdk: 1.6.0

-----
$$\  $$\ $$$$$$\  $$\  $$\  $$\
$$ |  $$ |$$ __$$\ $$$$  $$$ | \_|
$$ |  $$ |$$ |  $$ |$$$$\  $$$ |$$\ $$$$$$\  $$$$$$\  $$$$$$\
$$$$$$$ |$$$$$$$ |$$$$\$$ $ $ |$$ |$$ ____|$$ __$$\  $$ __$$\
$$ __$$ |$$ ____/  $$ \$$$ $ $ |$$ |$$ /  $$ |  \_|$$ /  $$ |
$$ |  $$ |$$ |      $$ |\$ /$$ |$$ |$$ |  $$ |  $$ |  $$ |
$$ |  $$ |$$ |      $$ | \/$$ |$$ |\$$$$$$\  $$ |  \$$$$$  |
\_|  \_| \_|      \_|  \/$$ |$$ |\$$$$$$\  $$ |  \$$$$$  |
\_|  \_| \_|      \_|  \/$$ |$$ |\$$$$$$\  $$ |  \$$$$$  |
-----
hello world!
sw_version:0
ota1 version:0, ota2 version:65535
now runing OTA1...
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

```



## 3.5 OTA 升级操作

### 3.5.1 镜像编辑生成

当我们需要升级固件时，在 OTA 之前我们需要先将编译生成的镜像加密，加密流程参考

**EXIP BLOB 介绍**，跟之前相比，我们这里修改软件版本大于之前的版本即可，比如之前

是 0，所以我这边假设设置个 4，为了方便跟之前的版本区别，我们这边创建的生成的名字



在后面加个 v4, 然后点击生成。生成的位置我们放在 tool 下, 方便使用 pack\_img 工具。

已检测到 XPI Flash 选项并已自动添加到配置中!

典型 Flash Word 配置: Channel A, Pin group0 地址: 0x0000

Flash Word: 0xfc90002,0x00000000,0x00000000,0x00000000

已检测到偏移量并已自动添加到配置中!

固件容器首地址相对于镜像地址偏移量: 0x0c00

软件版本(可选): 4 FUSE 版本(可选): 0

生成配置

选择输出Bin路径: D:/repo/sdk\_env/hpm\_app/customer/rom\_ota/tool/demo\_exip\_v4.bin

生成

需要注意: 镜像文件必须是原始的明文固件, 不要用密文固件再加密, 可以通过判断上面地址是否是 0 还是 0x400 判断, 如果是 0x400, 则是密文固件

### 3.5.2 OTA 包制作生成

使用 pack\_img 工具打包, 并生成对应 sign 后缀文件。在我们的 OTA demo 中, 会解析打包生成的头部信息并烧录到 flash 中, 其中 1 表示对加密固件打包, 0 表示对明文固件打包

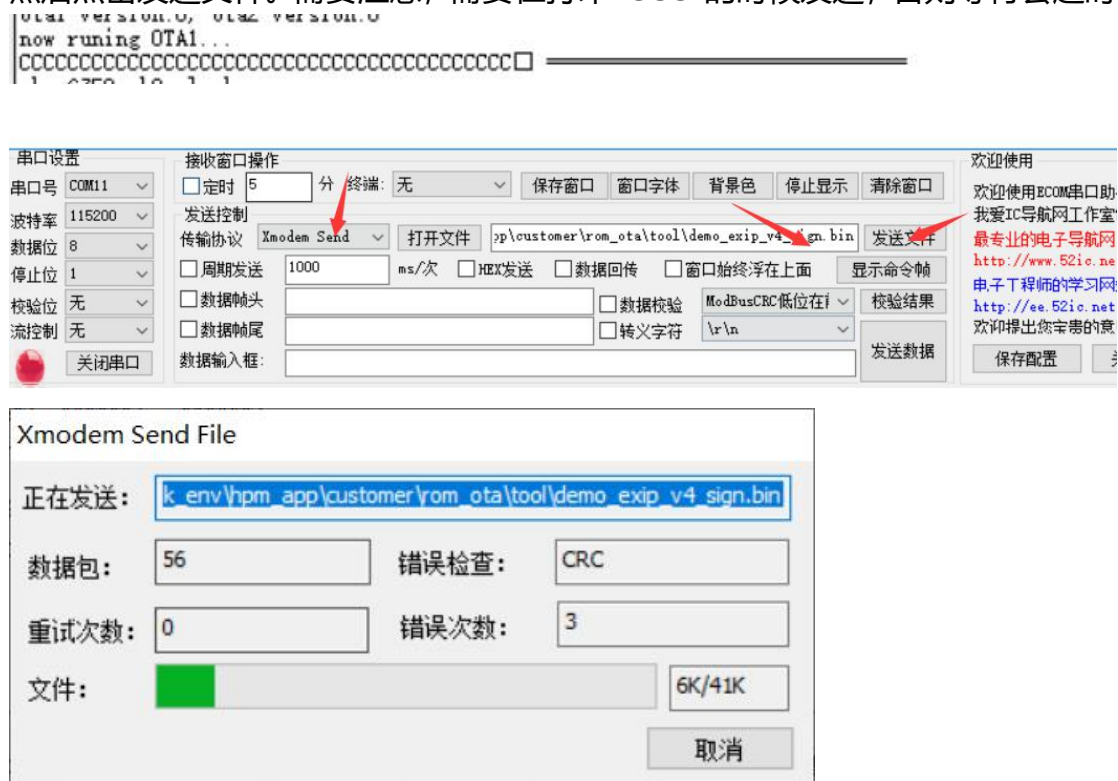
```
PS D:\repo\sdk_env\hpm_app\customer\rom_ota\tool> python .\pack_img.py 1 .\demo_exip_v4.bin
0: .\pack_img.py ,1: 1 ,2: .\demo_exip_v4.bin
fota_len: 41936
fota_checksum: 4585906
type: 1 , sign and ota success!
Generated file : .\demo_exip_v4_sign.bin ,SUCCESS!
pack .\demo_exip_v4.bin -> .\demo_exip_v4_sign.bin Success!!!
PS D:\repo\sdk_env\hpm_app\customer\rom_ota\tool> |
```

### 3.5.3 OTA 升级

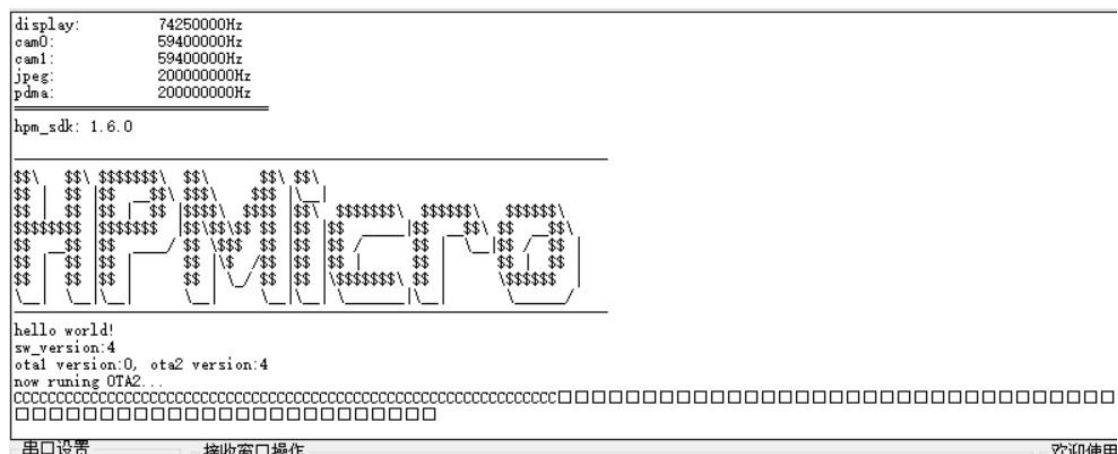
当前固件支持基于串口 xmodel 传输升级固件。

固件启动，选择签名 OTA 升级包，如：demo\_exip\_v4\_sign.bin 升级固件。

然后点击发送文件。需要注意，需要在打印 CCC 的时候发送，否则等待会超时

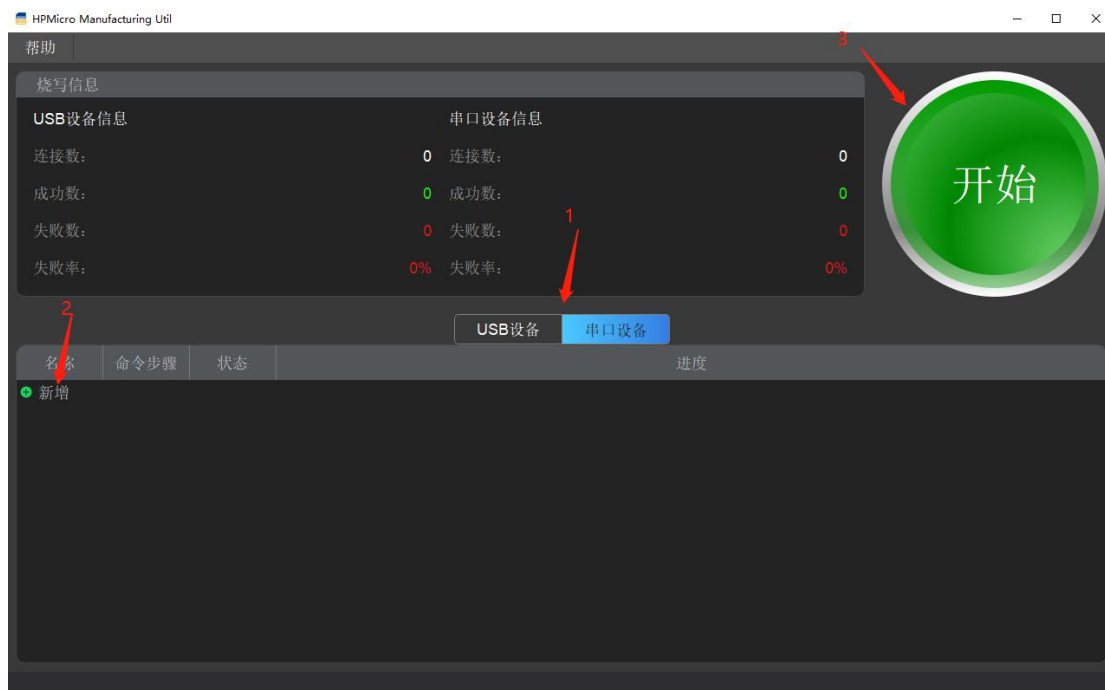
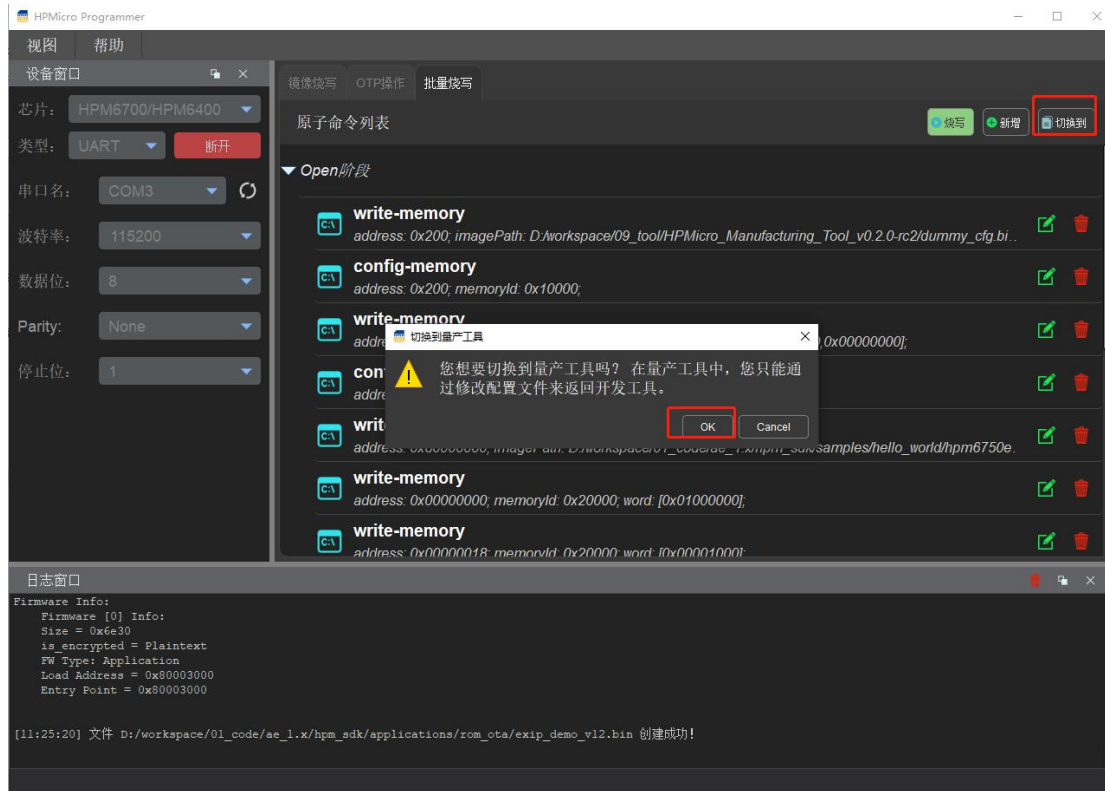


Ota 成功以后自动复位并跳转到最新固件



## 4.量产批量烧录

当批量烧写验证通过后，要量产批量烧录时，可切到量产烧录界面批量烧录。

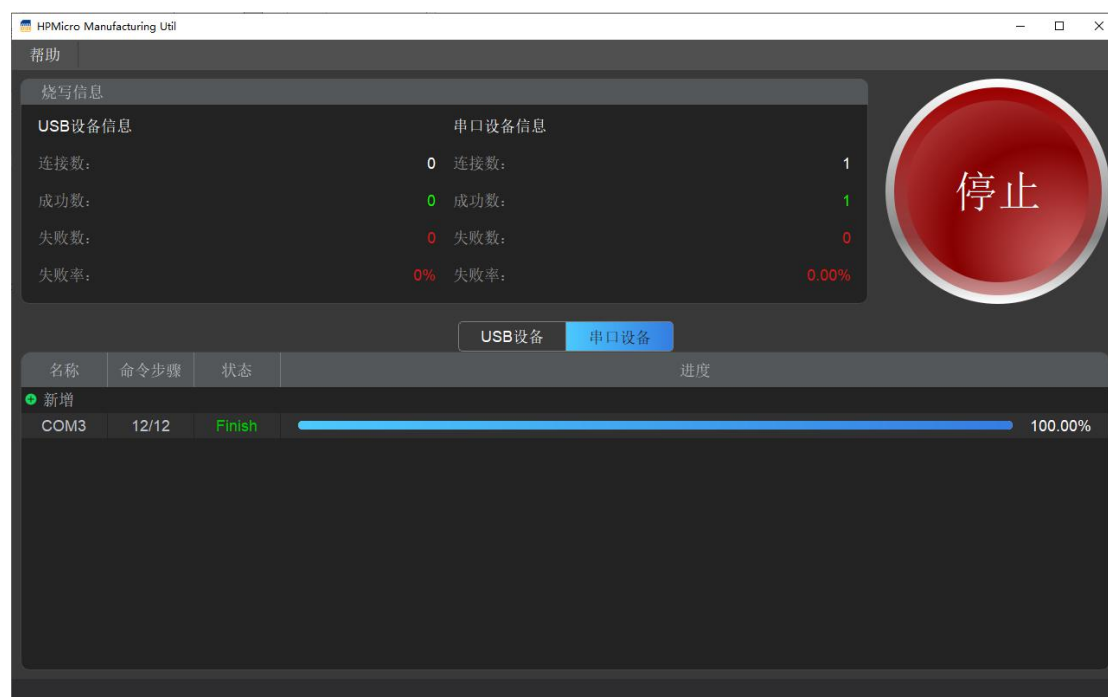


1. 选择是 USB 设备还是串口设备。
2. 如果是串口设备，需要新增串口设置。



3. 点击开始后，插入设备即可全自动烧录。

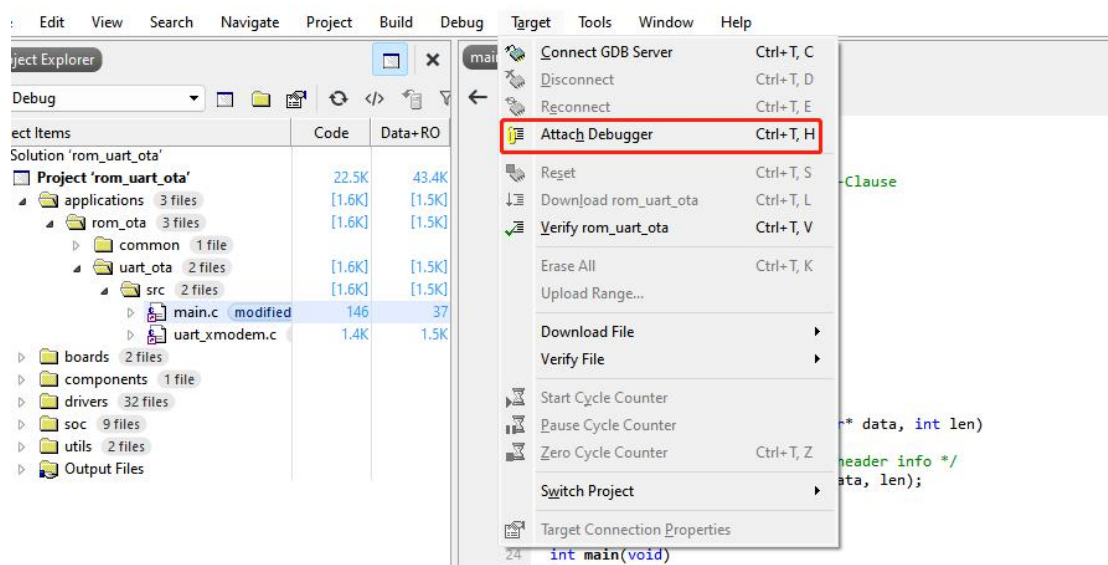
支持一拖多，且傻瓜式自动化烧录。



## 5.注意事项

1. 启用了 SEC\_IMAGE\_OFFSET 后，再次 DEBUG 仿真调试会受限(因为每

次编译生成的 sw\_version=0,启动会默认跳转到高版本运行), 为了能正常 DEBUG 仿真, 可先烧录 IMG2 sw\_version=0 的固件, 后 DEBUG 仿真。或者使用 Attach Debugger 的方式调试。



2. 启用了 EXIP\_BLOB(EXIP)后, 再次 DEBUG 仿真调试受限, 为了能正常 DEBUG 仿真, 可使用 RAM debug 的方式仿真。建议在量产开启了 Security Flash 后, 锁死 DEBUG 仿真。

3. 启用了 EXIP\_BLOB(EXIP)后, 烧录了明文固件, 不能正常启动。

4. 未启用 EXIP\_BLOB(EXIP), 烧录了加密固件, 不能正常启动。

5. KEK 已经写入 OTP 后不可更改。故要管理好自己的 KEK 密钥。