



■ PHY6252/6222

Security Boot User Guide

Version 1.0

Author: HXR

Security: Internal / Public

Date: 2020.12

PhyPlus



Revision History

Revision	Author	Date	Description
V1.0	HXR	2020.12.22	Draft

目录

1	简介	1
2	Efuse Key	1
2.1	Efuse API	1
2.2	Efuse key 烧写	1
2.2.1	Efuse key 烧写操作	1
2.2.2	Efuse key 烧写注意项	3
3	Security boot 秘钥产生	3
3.1	g_sec_key 产生过程	3
3.2	g_ota_sec_key 产生过程	3
4	Security Boot	4
4.1	流程详解	6
4.2	ROM Security Boot	6
4.3	OTA Security Boot	8
5	Flash Mapping	9
5.1	No OTA Mode Flash Mapping	9
5.2	Support OTA Flash Mapping	10

1 简介

本文介绍的 Security Boot 功能主要是针对 PHY6252/PHY6222 产品。主要介绍 security boot 中涉及到的 Efuse Key 的烧写、秘钥的获取方法及 security boot 流程和对应模式（No OTA/Support OTA）走安全启动的具体操作方法。

2 Efuse Key

Security boot 功能实现的其中关键之一就是 efuse key 的使用，**注意**：efuse key 只能写入一次，且一经写入就不可更改。

2.1 Efuse API

Efuse 总共有 4 块，主要用途和枚举列表如下：

EFUSE_BLOCK_0	0	用作 security boot 的 efuse key
EFUSE_BLOCK_1	1	用作 OTA security boot app 的 efuse key
EFUSE_BLOCK_2	2	暂未使用
EFUSE_BLOCK_3	3	暂未使用

efuse_lock(EFUSE_block_t block)	用来 lock efuse block 写入的数据
efuse_read(EFUSE_block_t block, buf)	读取 efuse block 的数据
efuse_write(EFUSE_block_t block buf, us)	写入 efuse block 数据

2.2 Efuse key 烧写

Security boot 功能的实现需要进行 efuse key 的烧写，烧写 efuse key 必须在烧录模式（cmd>>:）下进行。

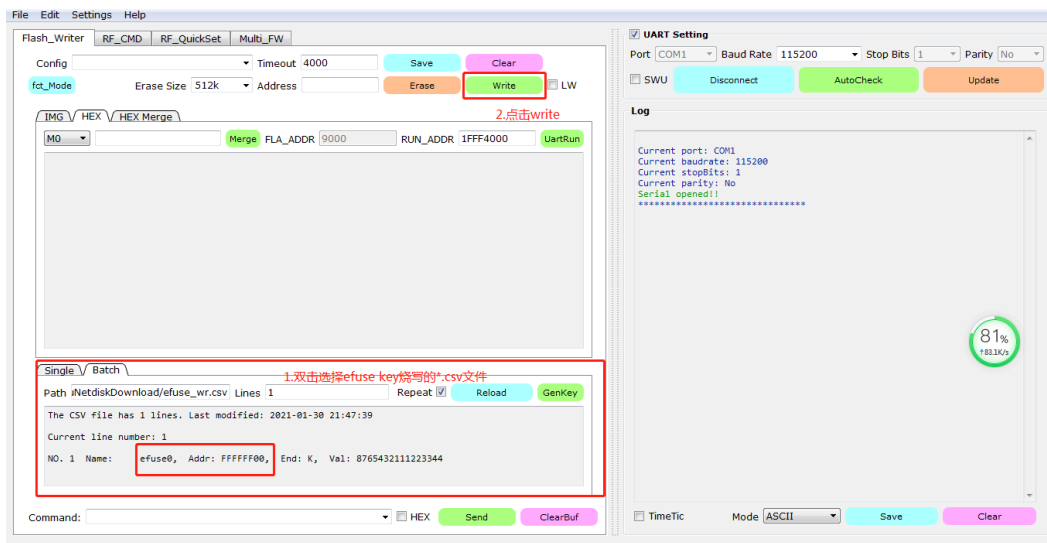
2.2.1 Efuse key 烧写操作

PhyPlusKit.exe 和烧录器工具，均通过 csv 三元组的方式解析并烧写 efuse block key，具体 csv 文件的格式如下（表格显示）：

1) No OTA

#efuse0
FFFFFF00-K
8765432111223344

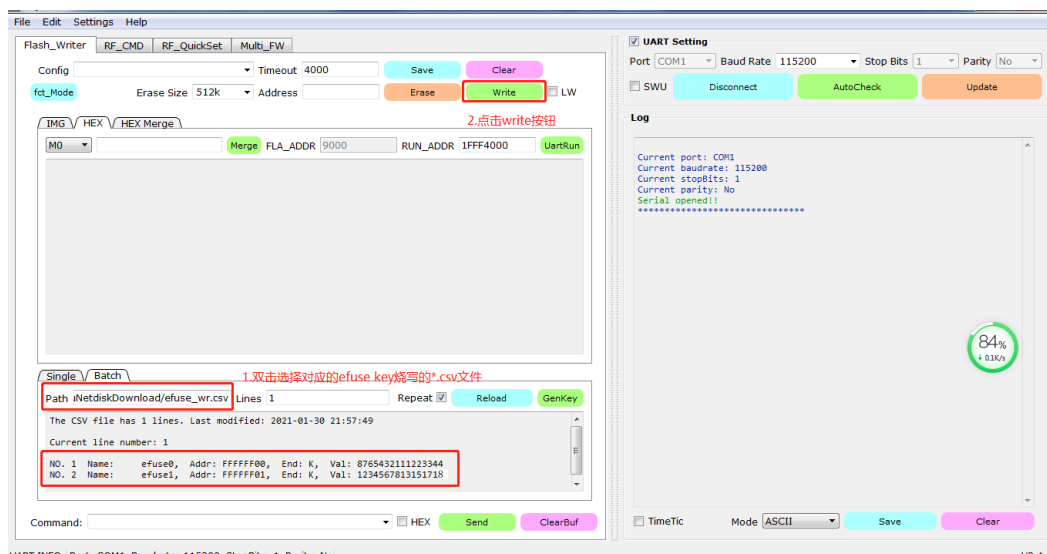
No OTA 模式走 security boot 只要进行 efuse block0 的烧写（ROM security boot）。对应 efuse key 烧写的工具操作步骤如下图：



2) Support OTA

#efuse0	#efuse1
FFFFFFF0-K	FFFFFFF1-K
8765432111223344	1234567813151718

Support OTA 模式走 security boot 需要进行 efuse block0, block1 两个 block 的烧写。对应 efuse key 烧写的工具操作步骤如下图：



Csv 文件中 efuse key 的格式解析：

第一行：Name 标注，以“#”开头作为 name 标识；efuse key 的 name 根据 efuse block 值依次为 efuse0,efuse1,efuse2,efuse3；

第二行：写入地址和端口；efuse key 写入端口固定为 K，写入 address 根据 block 值依次为 FFFFFFF0, FFFFFFF1, FFFFFFF2, FFFFFFF3；

第三行：写入值，就是对应烧写的 efuse block 值（64bit）。

2.2.2 Efuse key 烧写注意项

- 1) 烧写 efuse key 必须在烧录模式 (cmd>>) 下进行。
- 2) Efuse block 都只能烧写一次且不更改, 需要用户自己管理
- 3) 烧写的 efuse block 值必须是奇校验值, 例如: 8765432111223344, bit 置 1 的个数是奇数, 既满足要求, 如果输入不符合条件的数值, 会出现报错提示!!

3 Security boot 秘钥产生

Security boot 是利用 aes_ccm 算法对 App 程序进行加密, 重启时进行解密 boot 的过程。这里主要介绍用来加解密的秘钥 g_sec_key 和 g_ota_sec_key 的获取方法:

3.1 g_sec_key 产生过程

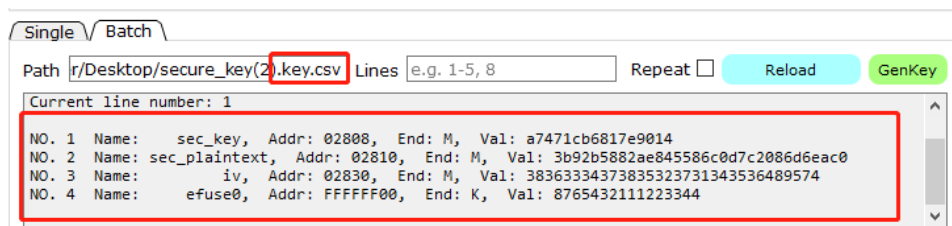
g_sec_key 是进行 ROM security boot APP (No OTA) 用来进行加解密的秘钥, 下面详细介绍通过使用 PhyPlusKit.exe 工具来产生 g_sec_key。

PhyPusKit.exe 工具主要通过解析 *.key.csv 文件的方式来产生 g_sec_key。具体的 *.key.csv 文件内容设置如下 (表格显示):

#sec_key	#sec_plaintext	#iv	#efuse0
2808-M	2810-M	2830-M	FFFFFF00-K
a7471cb6817e9014	3b92b5882ae845586c0d7c2086d6eac0	38363334373835323731343536489574	8765432111223344

使用 PhyPlusKit.exe (v2.4.5e 版本开始) 工具产生 g_sec_key 方法如下:

- a. Batch 页面双击加载上述用户自定义的 *.key.csv 文件(注意要导入 *.key.csv 文件类型, 否则会报错)



- b. 点击 GenKey 按钮, 会对应产生当前显示 current line 的 efuse key 和 flash key 处理后的 *.sec.csv 文件, 可以根据填写的 Lines 值, 产生对应行的数据 (*.sec.csv 文件)。(注意只产生一行数据, lines 配置是根据行数配置产生对应选择行的 *.sec.csv)



点击 GenKey 按钮产生的 *.sec.csv 文件会对应产生 g_sec_key。

3.2 g_ota_sec_key 产生过程

`g_sec_key` 是进行 ROM security boot OTA（Support OTA）用来进行加解密的密钥；
`g_ota_sec_key` 是进行 OTA security boot APP（Support OTA）用来进行加解密的密钥，下面详细介绍通过使用 `PhyPlusKit.exe` 工具来产生 `g_sec_key` 和 `g_ota_sec_key`。

`PhyPlusKit.exe` 工具主要通过解析*.key.csv 文件的方式来产生 `g_sec_key` 和 `g_ota_sec_key`。具体的*.key.csv 文件内容设置如下（表格显示）：

#sec_key	#sec_plaintext	#iv	#efuse0	#ota_sec_key	#ota_plaintext	#efuse1
2808-M	2810-M	2830-M	FFFFFF00-K	2908-M	2910-M	FFFFFF01-K
a7471c b6817e 9014	3b92b5882ae8 45586c0d7c20 86d6eac0	383633343738 353237313435 36303030	8765432 1112233 44	817e90 14a747 1cb6	e907c7b41754 a060d34a6285 3cb23de8	123456 781315 1718

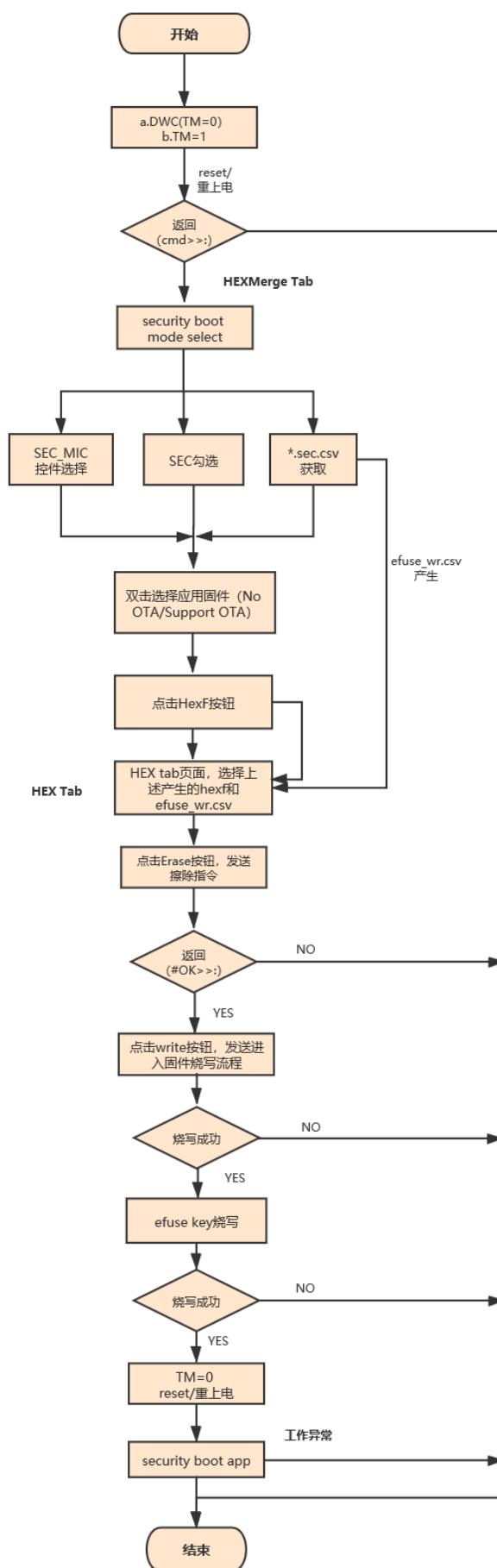
使用 `PhyPlusKit.exe`（v2.4.5e 版本开始）工具产生 `g_sec_key` 和 `g_ota_sec_key` 方法可参照 3.1 小节 `g_sec_key` 的产生过程，只是*.key.csv 文件内容不同。

相同操作方法对应产生的*.sec.csv 文件会对应产生 `g_sec_key` 和 `g_ota_sec_key`。

注意上述产生*.sec.csv 文件的同时产生 `efuse_wr.csv` 文件用来作为 `efuse key` 的烧写文件，具体关于 `efuse key` 的烧写和 `efuse_wr.csv` 文件详解已在 2.2 小节作详细介绍。

4 Security Boot

上述 3 节已经详细介绍了 `security boot` 加解密需要的密钥获取过程，这里会介绍 `security boot` 的工具使用，具体流程如下。



4.1 流程详解

1. PHY6252/PHY6222 上电后，通过 DWC 连接（TM=0）重上电/TM=1（将 TM 拉高），Reset 开发板，进入烧写模式，返回 cmd>>:
2. HEXMerge 页面，工具选择对应的 SEC_MIC 和 SEC 控件，Security boot 过程需要的秘钥 *.sec.csv 文件的获取可详见 3 节
3. 选择需要烧写的应用固件，包括 No OTA/Support OTA 模式点击 HexF 按钮，产生对应的密文 hexf 文件
4. 切换至 HEX 页面，选择上述生成的 hexf 文件和 efuse_wr.csv 文件
5. 点击 Erase 按钮，发送擦除命令，成功后，点击 write 按钮进行固件和 efuse 的烧写过程
6. 烧写 flash 和 efuse 成功，重上电（TM=0）或者 TM 拉低 reset PHY622X，应用程序跑起来,整个 security boot 流程结束。

4.2 ROM Security Boot

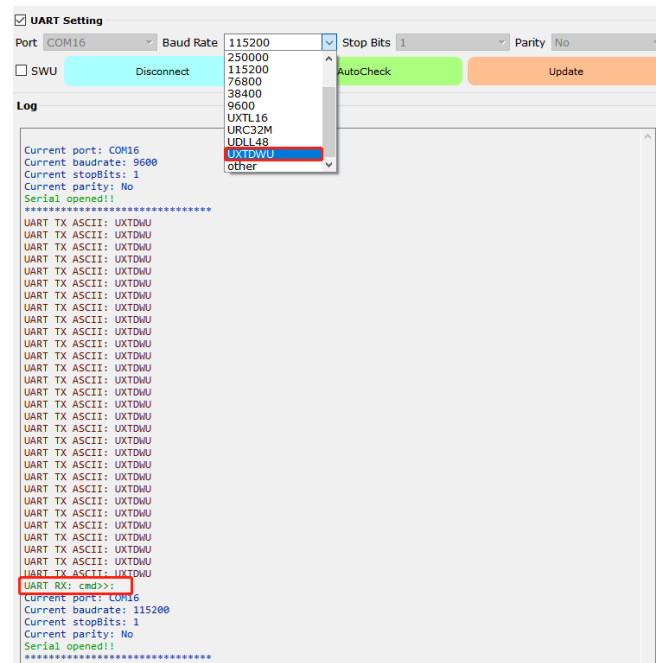
ROM Security boot 过程即 No OTA 的加密启动过程。

PhyPlusKit.exe 工具 V2.4.5e 版本开始，支持 security boot 功能，该功能模块在选择 SEC_MIC 模式中支持。选择对应的 SEC_MIC 表单才能走 security boot 功能。

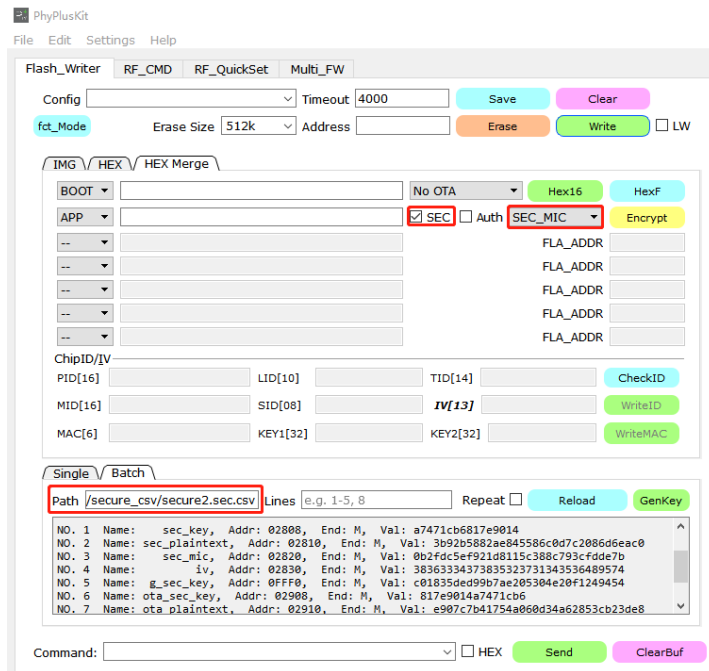
操作步骤如下：

- a. PHY6252/PHY6222 上电后，通过 DWC 连接（TM=0）重上电/TM=1（将 TM 拉高），Reset 开发板，进入烧写模式，返回 cmd>>:

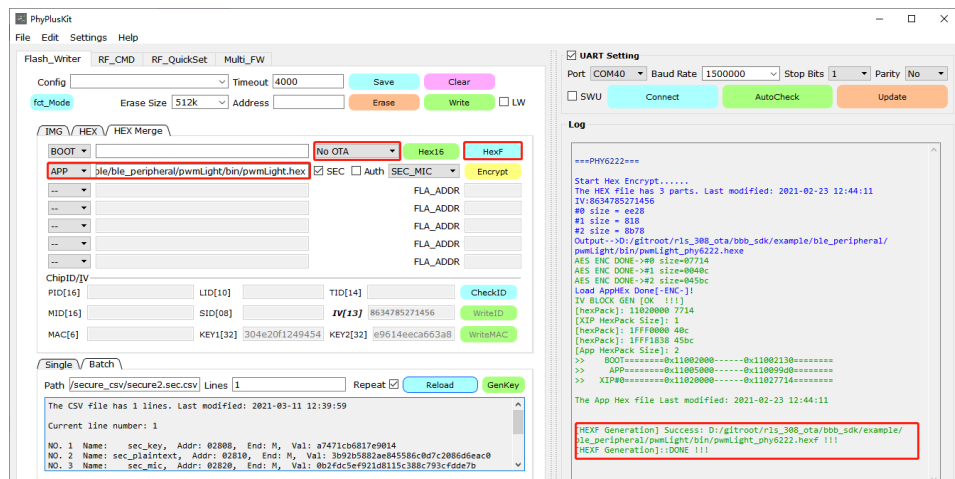
下图是 PHY6252(TM=0)通过双线 DWC 连接进入烧写模式的图示：



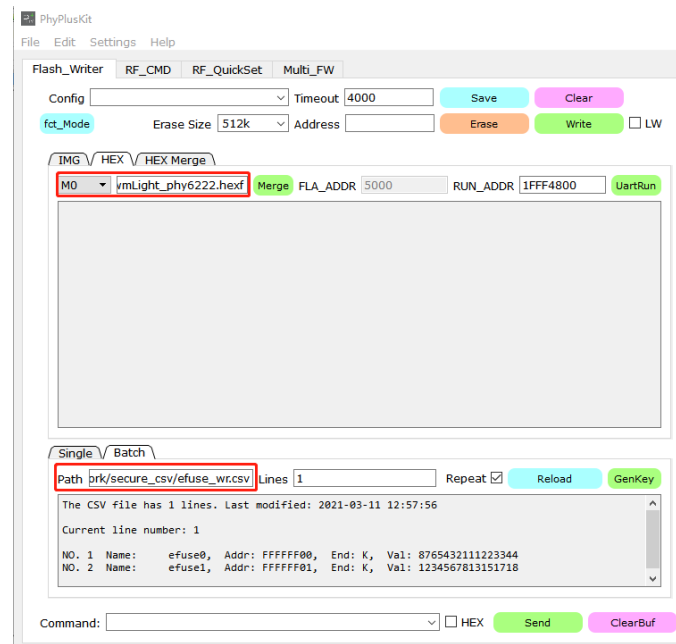
- b. HEXMerge 页面下，选择 SEC_MIC 模式并勾选 SEC 控件,Batch 页面中双击选择 *.key.csv 文件并产生对应的*.sec.csv 文件



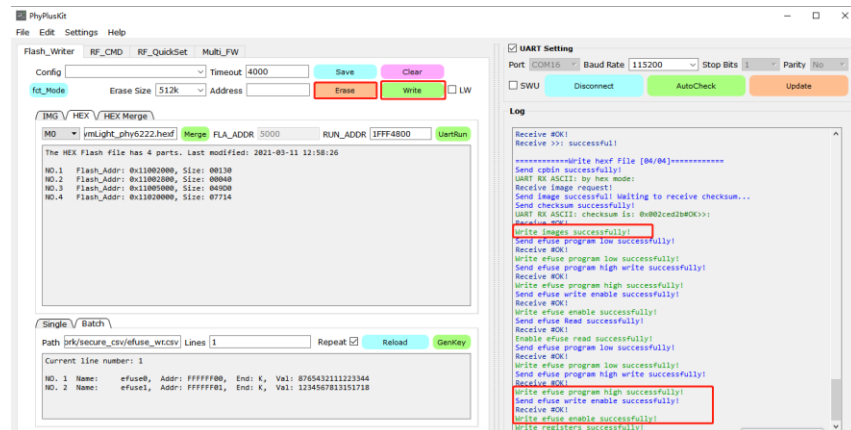
- c. 双击选择应用固件（No OTA），点击 HexF 按钮，产生对应的 hexf 文件



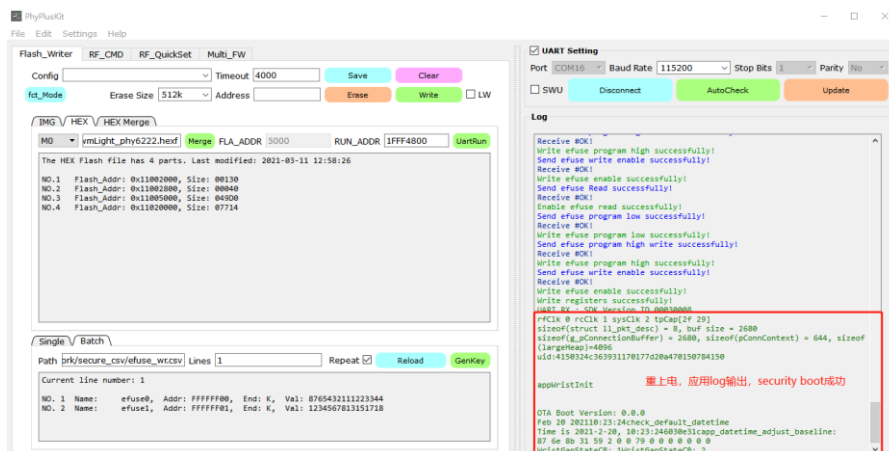
- d. 切换至 HEX 页面，选择对应的 hexf 文件和 GenKey 产生的 efuse_wr.csv 文件



e. 点击 Erase 按钮，成功后进行固件和 efuse key 的烧写



f. 固件和 efuse 烧写成功后，重新上电（TM=0）/TM 拉低，reset（TM=1），security boot 流程走通即可跳转至应用程序，完成 ROM security boot 过程

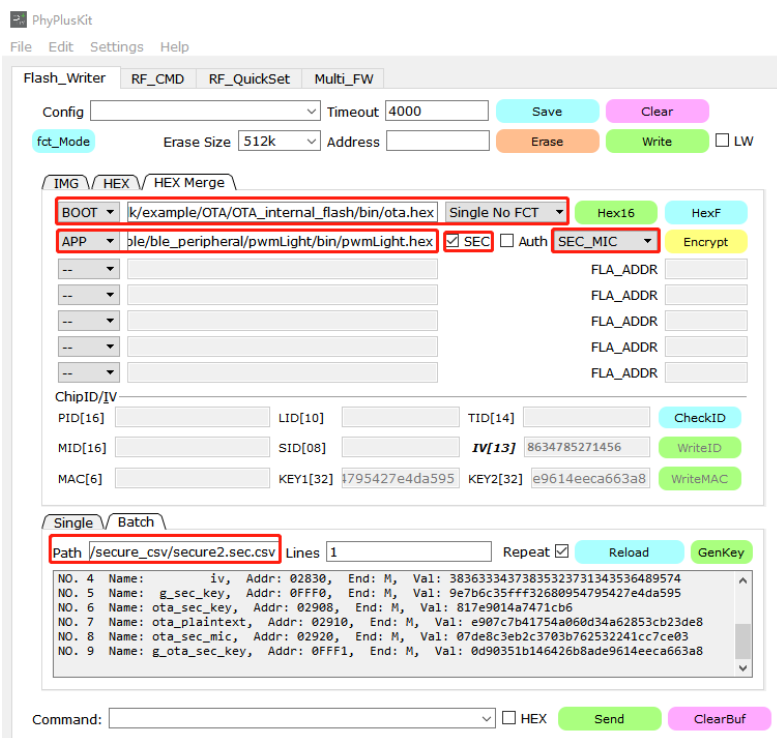


4.3 OTA Security Boot

OTA Security boot 过程即 Support OTA 的加密启动过程。

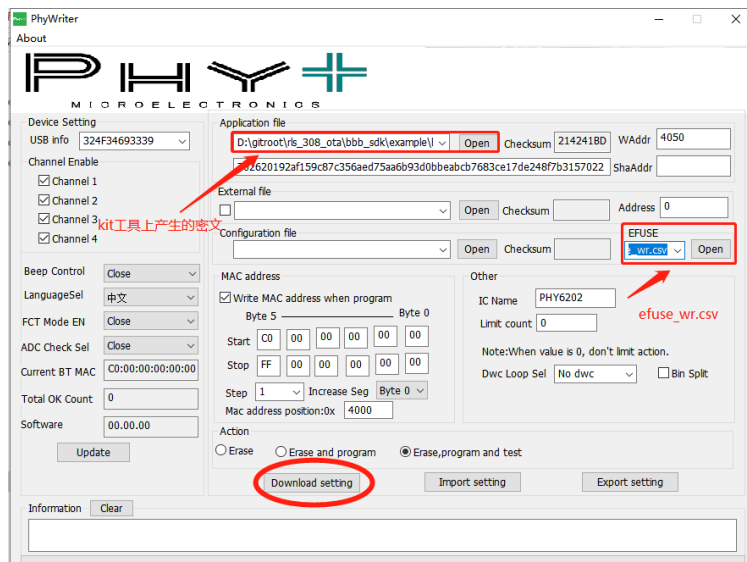
具体的流程和步骤可参考 No OTA 模式：

不同的是需要选择 ota.hex 文件和对应的 single no fct 模式，如下图：



备注：脱机烧录器 security boot 就只需要提供上述 c 步骤产生的 hexf 文件和 b 步骤产生对应的 efuse key 的三元组*.csv 文件即可。

具体的配置如下：



5 Flash Mapping

5.1 No OTA Mode Flash Mapping

Flash Mapping No OTA						
	256KB Flash			512KB Flash		
Reserved By PhyPlus	0	1FFF	8	0	1FFF	8
1st Boot info	2000	2FFF	4	2000	2FFF	4
FCDS	4000	4FFF	4	4000	4FFF	4
App Bank	5000	1FFFF	108	5000	1FFFF	108
XIP	20000	3BFFF	112	20000	33FFF	80
FS (UCDS)	3C000	3DFFF	8	34000	35FFF	8
Resource	3E000	3FFFF	8	36000	7FFFF	296
FW Storage	40000	3FFFF	0	80000	7FFFF	0

5.2 Support OTA Flash Mapping

Single Bank OTA						
	256KB Flash			512KB Flash		
Reserved By PhyPlus	0	1FFF	8	0	1FFF	8
1st Boot info	2000	2FFF	4	2000	2FFF	4
2nd Boot info	3000	3FFF	4	3000	3FFF	4
FCDS	4000	4FFF	4	4000	4FFF	4
OTA Bootloader	5000	10FFF	48	5000	10FFF	48
App Bank	11000	1FFFF	60	11000	1FFFF	60
XIP	20000	3BFFF	112	20000	33FFF	80
FS (UCDS)	3C000	3DFFF	8	34000	35FFF	8
Resource	3E000	3FFFF	8	36000	7FFFF	296
FW Storage	40000	3FFFF	0	80000	7FFFF	0