




K-Solution PRBMD02 Bluetooth Tiny 5.0 Module User Guide

[Home](#) » [K-Solution](#) » K-Solution PRBMD02 Bluetooth Tiny 5.0 Module User Guide 

Contents

- [1 K-Solution PRBMD02 Bluetooth Tiny 5.0 Module](#)
- [2 Introduction](#)
- [3 Efuse Key](#)
 - [3.1 Format parsing of efuse key in Csv file](#)
 - [3.2 Efuse key programming note](#)
- [4 Security boot Key generation](#)
- [5 Security Boot](#)
- [6 Operation flow](#)
- [7 Flash Mapping](#)
- [8 Documents / Resources](#)
- [9 Related Posts](#)

KSolutions

K-Solution PRBMD02 Bluetooth Tiny 5.0 Module



Introduction

The Security Boot function introduced in this article is mainly for PRBMD02. It mainly introduces the programming of the Efuse Key involved in the secure boot, the method of obtaining the key, the secure boot process, and the specific operation method of the corresponding mode (No OTA/Support OTA) to perform the secure boot.

Efuse Key

- One of the keys to the realization of the secure boot function is the use of the refuse key.
- **Note:** the efuse key can only be written once and cannot be changed once written.

Efuse API

Efuse has a total of 4 blocks, the main uses and enumeration lists are as follows:

EFUSE_BLOCK_0	0	efuse key for security boot
EFUSE_BLOCK_1	1	Used as efuse key for OTA security boot app
EFUSE_BLOCK_2	2	Future use
EFUSE_BLOCK_3	3	Future use

efuse_lock(EFUSE_block_t block)	Lock data written to efuse block
efuse_read(EFUSE_block_t block, buf)	Read defuse block data
efuse_write(EFUSE_block_t block buf, us)	With fuse block data

Efuse key programming

- The realization of the security boot function requires the programming of the efuse key, and the programming of the efuse key must be performed in the programming mode (cmd>>:).

Efuse key programming operation

PhyPlusKit.exe and the programmer tool both parse and program the efuse block key by means of csv triples. The specific csv file format is as follows (shown in the table):

- No OTA

#efuse0
FFFFFFF00-K
8765432111223344

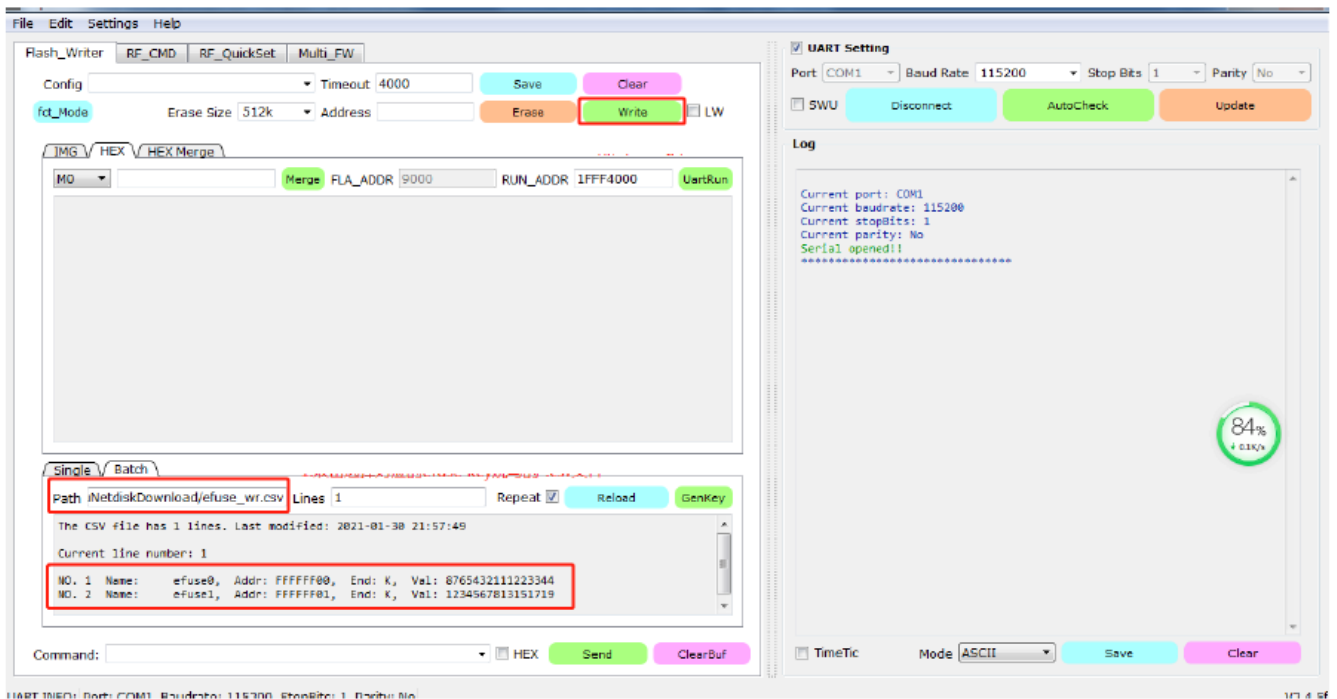
No OTA mode and security boot only need to program efuse block0 (ROM security boot). The tool operation steps corresponding to efuse key programming are as follows:



- Supports OTA

#efuse0	#efuse1
FFFFFFF00-K	FFFFFFF01-K
8765432111223344	1234567813151718

Support OTA mode and security boot need to program the two blocks of efuse block0 and block1. The tool operation steps corresponding to efuse key programming are as follows:



Format parsing of efuse key in Csv file

- **1st line:** Name is marked, starting with “#” as the name identification; the name of the efuse key is efuse0, efuse1, efuse2, efuse3 according to the value of the efuse block;
- **2nd line:** write address and port; the efuse key write port is fixed to K, and the write address is FFFFFFF0, FFFFFFF1, FFFFFFF2, FFFFFFF3 according to the block value;
- **3rd line:** the write value is the corresponding programmed efuse block value (64bit).

Efuse key programming note

- The use key must be programmed in the programming mode (cmd>>:).
- Fuse block can only be programmed once and not changed and needs to be managed by the user
- The programmed efuse block value must be an odd check value, for example 8765432111223344, the number of bits set to 1 is an odd number, which meets the requirements. If you enter a value that does not meet the conditions, an error message will appear!

Security boot Key generation

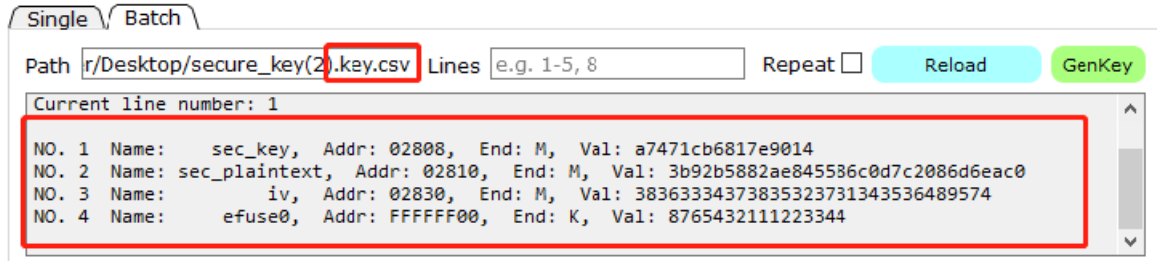
Security boot is the process of encrypting the App program by using the aes_ccm algorithm and decrypting the boot when restarting. Here we mainly introduce how to obtain the secret keys g_sec_key and g_ota_sec_key used for encryption and decryption:

g_sec_key generation process

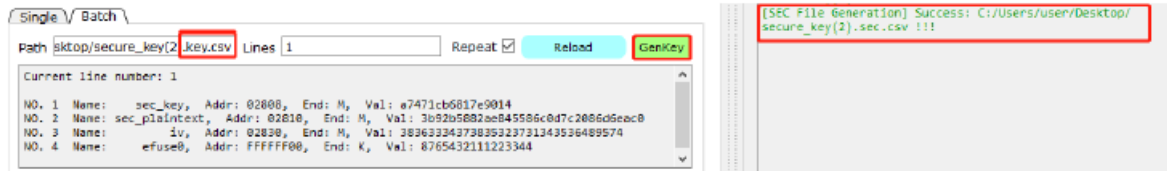
- g_sec_key is the secret key used for encryption and decryption by ROM security boot APP (No OTA). The following describes in detail how to generate g_sec_key by using PhyPlusKit.exe tool.
- The PhyPusKit.exe tool generates g_sec_key mainly by parsing the *.key.csv file. The specific content of the *.key.csv file is set as follows (table display):

#sec_key	#sec_plaintext	#iv	#efuse0
2808-M	2810-M	2830-M	FFFFFF00-K
a7471cb6817 e9014	3b92b5882ae845586c0 d7c2086d6eac0	3836333437383532373 1343536303030	8765432111223344

Use the PhyPlusKit.exe (starting from v2.4.5e) tool to generate g_sec_key The method is as follows:



- Double-click to load the above user-defined *.key.csv file on the Batch page (note that the .key.CSV file type must be imported, otherwise an error will be reported)



Clicking the GenKey button will generate the *.sec.csv file processed by the use key and flash key currently displayed on the current line. The data of the corresponding line (*.sec.csv file can be generated according to the Lines value filled in. (Note that only one row of data is generated, the configuration of the line is to generate *.sec.csv corresponding to the selected row according to the number of rows configuration) The *.sec.csv file generated by clicking the GenKey button will generate g_sec_key accordingly.

g_ota_sec_key generation process

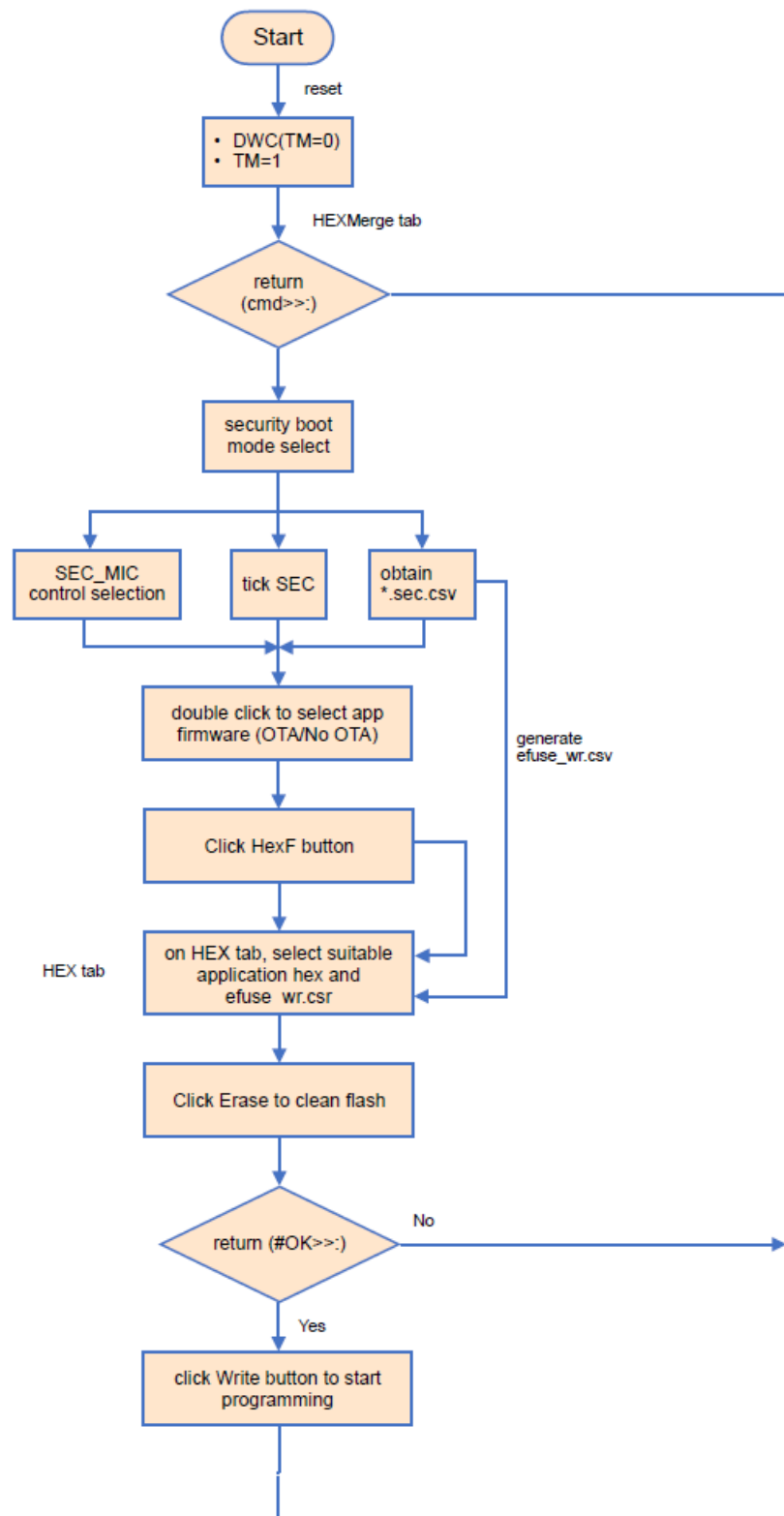
- g_sec_key is the key used for encryption and decryption by ROM security boot OTA (Support
- OTA); g_ota_sec_key is the key used for encryption and decryption by OTA security boot APP(Support OTA). Generate g_sec_key and g_ota_sec_key.
- The PhyPusKit.exe tool mainly generates g_sec_key and g_ota_sec_key by parsing the *.key.csv file. The specific content of the *.key.csv file is set as follows (table display):

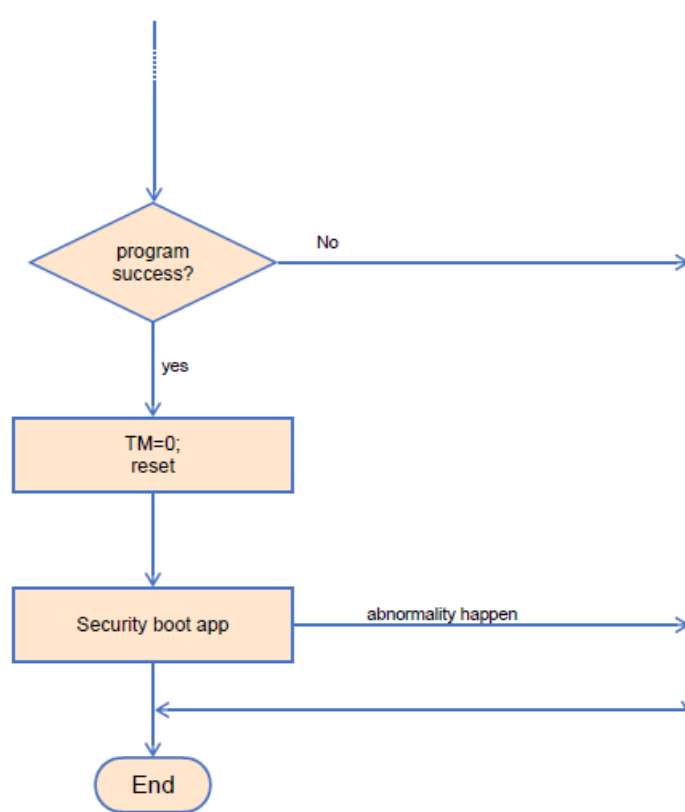
#sec_key	#sec_plaintext	#iv	#efuse0	#ota_sec_key	#ota_plaintext	#efuse1
2808-M	2810-M	2830-M	FFFFFF00-K	2908-M	2910-M	FFFFFF01-K
a7471cb68	3b92b5882ae	38363334373	876543211	817e9014a	e907c7b41	123456781
17e9014	845586c0d7c	83532373134	1223344	7471cb6	754a060d3	3151718
	2086d6eac0	3536303030			4a62853cb	
					23de8	

- The method of generating g_sec_key and g_ota_sec_key by using PhyPlusKit.exe (starting from v2.4.5e) can refer to the generation process of g_sec_key in Section 3.1, but the content of the *.key.CSV file is different.
- The *.sec.CSV file corresponding to the same operation method will generate g_sec_key and g_ota_sec_key correspondingly.
- Note that while the *.sec.CSV file is generated above, the efuse_wr.csv file is generated to be used as the efuse key programming file. The details of the refuse key programming and efuse_wr.csv file have been introduced in detail in Section 2.2.

Security Boot

The above three sections have described in detail the key acquisition process required for security boot encryption and decryption. Here, the use of security boot tools will be introduced. The specific process is as follows:





Operation flow

1. After PHY6252/PHY6222 is powered on, re-power on through DWC connection (TM=0)/TM=1 (pull TM high), Reset the development board, enter programming mode, and return to cmd>>:
2. On the HEXMerge page, the tool selects the corresponding SEC_MIC and SEC controls, and the secret key *.sec.csv file required by the Security boot process can be obtained in Section 3.
3. Select the application firmware to be programmed, including No OTA/Support OTA mode and click the next button to generate the corresponding ciphertext hex file
4. Switch to the HEX page, select the hexf file and efuse_wr.csv file generated above
5. Click the Erase button to send the erase command, after success, click the write button to program the firmware and efuse.
6. After the flash and efuse are successfully programmed, power on again (TM=0) or TM pulls down the reset PHY622X, the application runs, and the entire security boot process ends.

ROM Security Boot

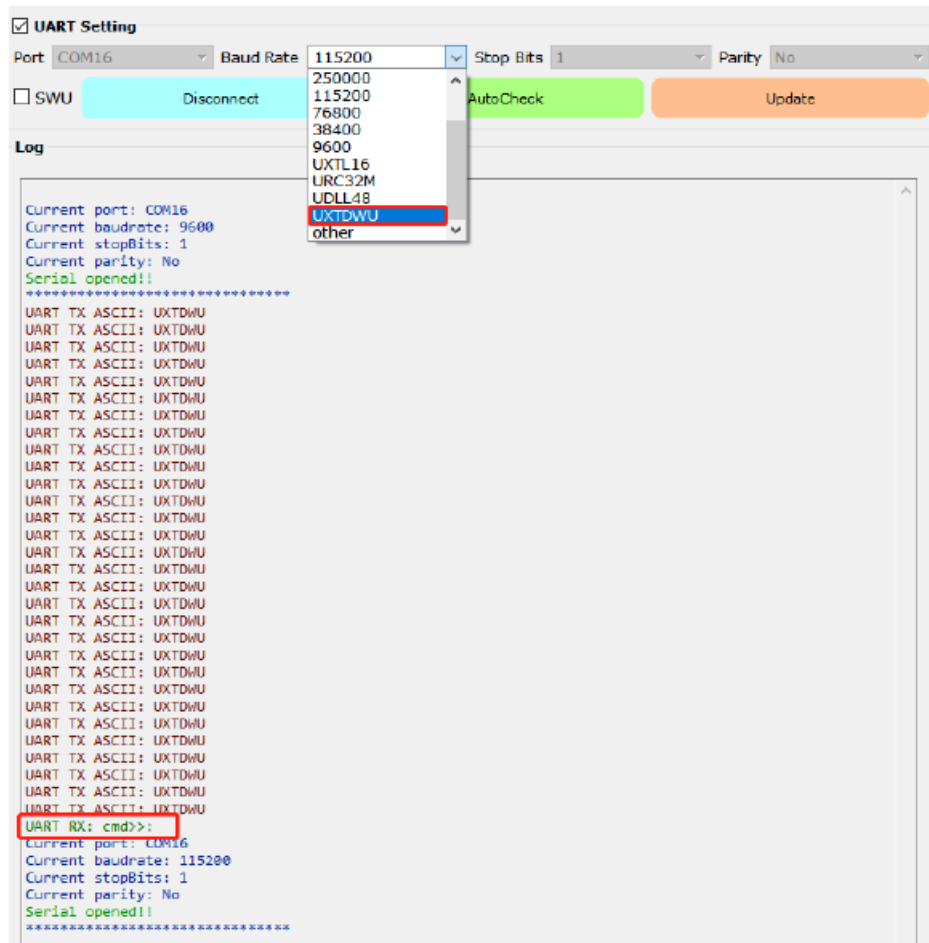
The ROM Security boot process is the encrypted boot process of No OTA. PhyPlusKit.exe tool V2.4.5e version, support secure boot function, this function module i supported in selecting SEC_MIC mode. Select the corresponding SEC_MIC form to use the secure boot function.

The operation steps are as follows:

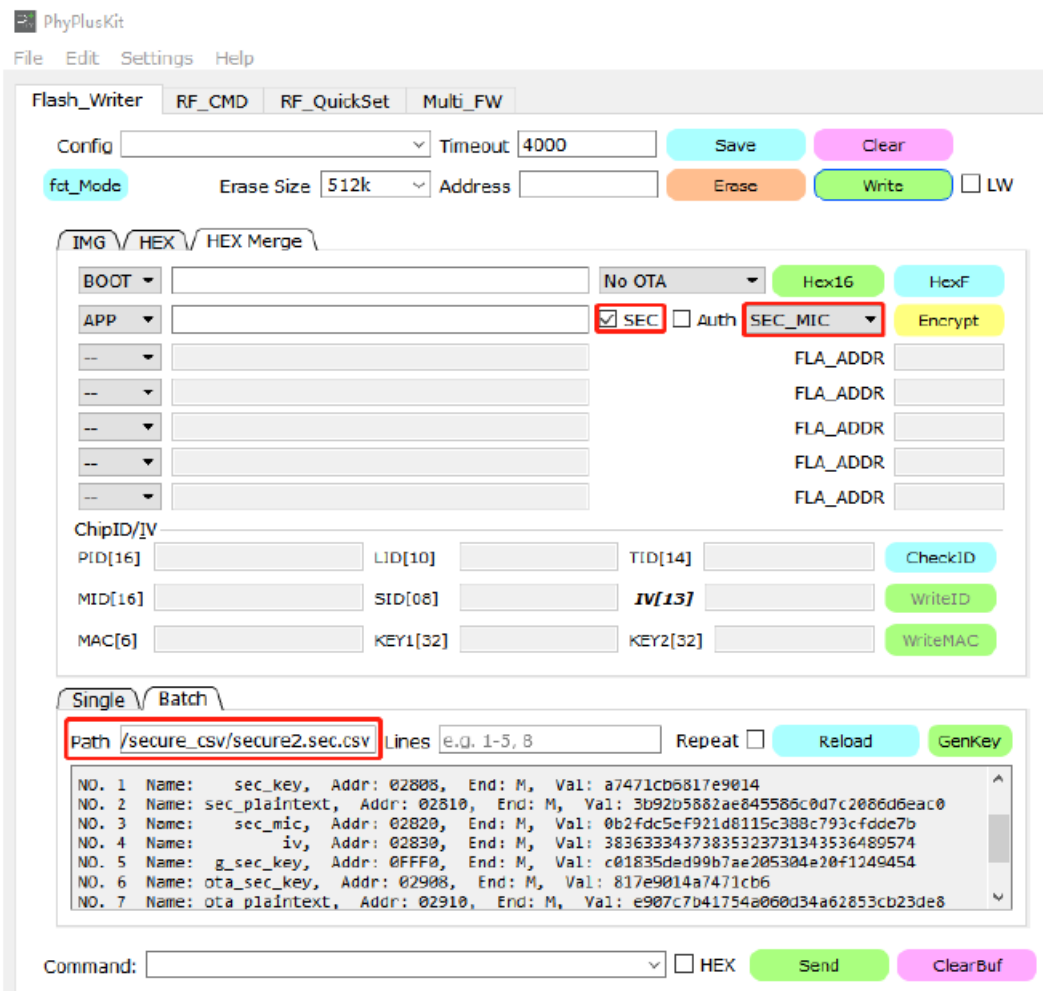
- After PHY6252/PHY6222 is powered on, re-power on through DWC connection (TM=0)/TM=1 (pull TM high), Reset the development board, enter programming mode, and return to cmd>>:

The following figure shows the PHY6252 (TM=0) entering the programming mode through the two-wire DWC

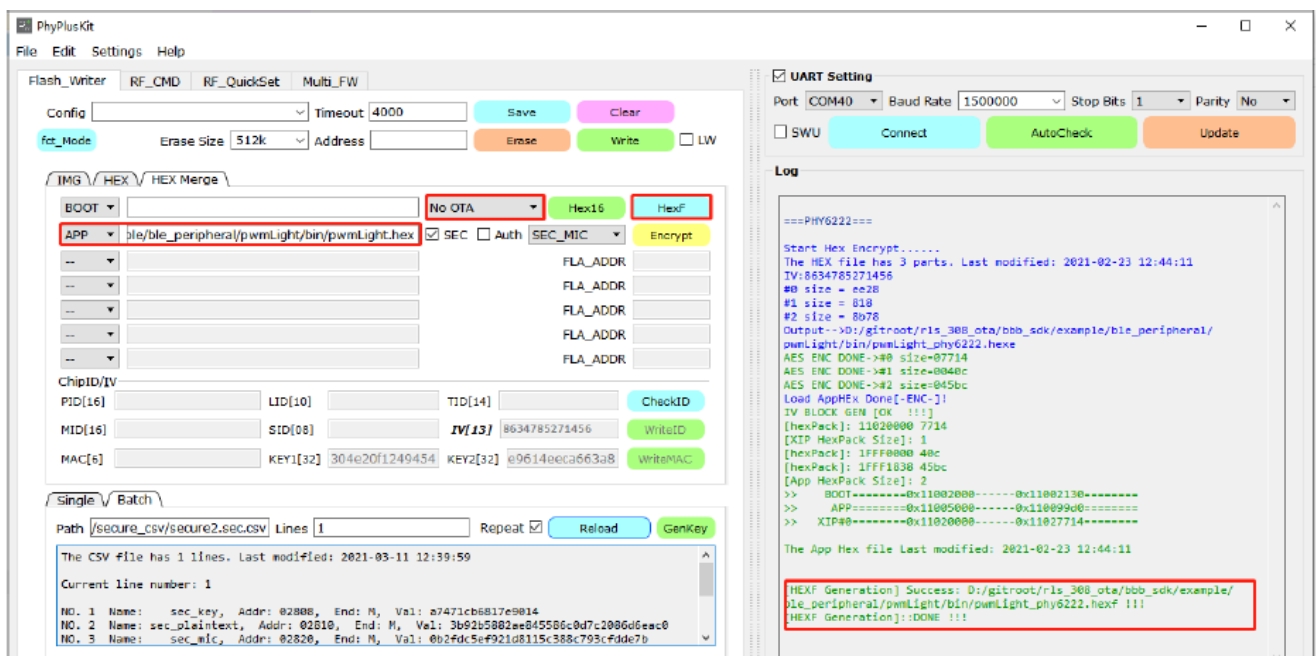
connection:



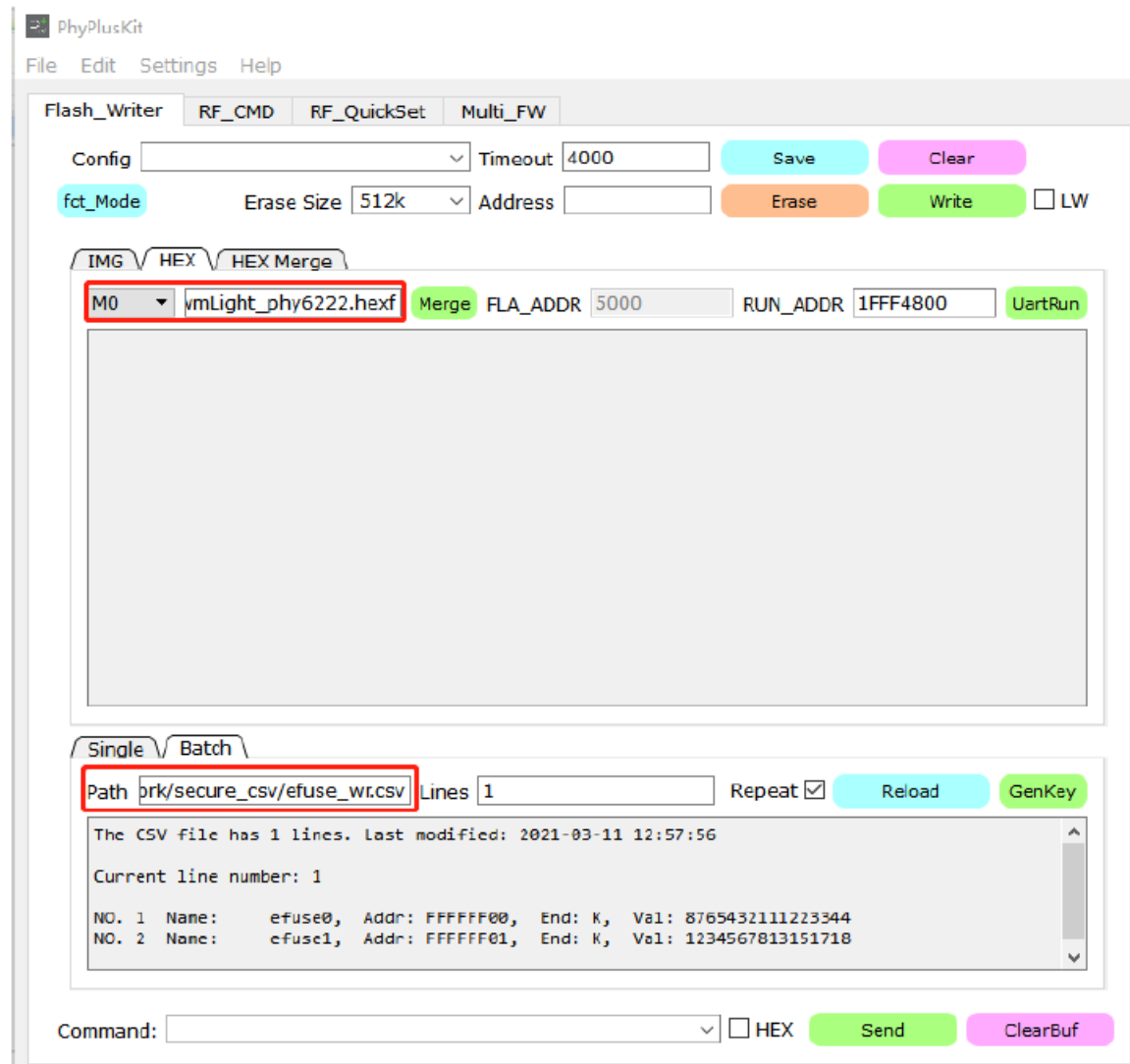
- On the HEXMerge page, select the SEC_MIC mode and check the SEC control, double-click on the Batch page to select the *.key.csv file and generate the corresponding *sec.csv file.



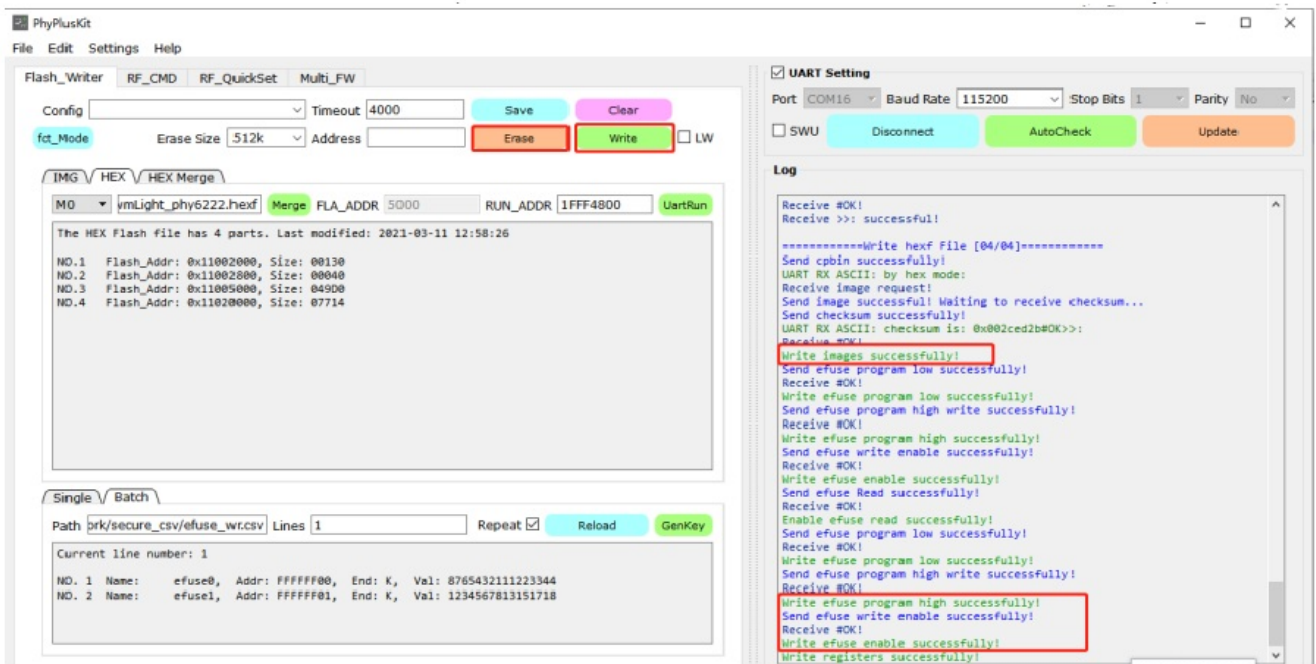
- Double-click to select the application firmware (No OTA), click the next button to generate the corresponding hex file



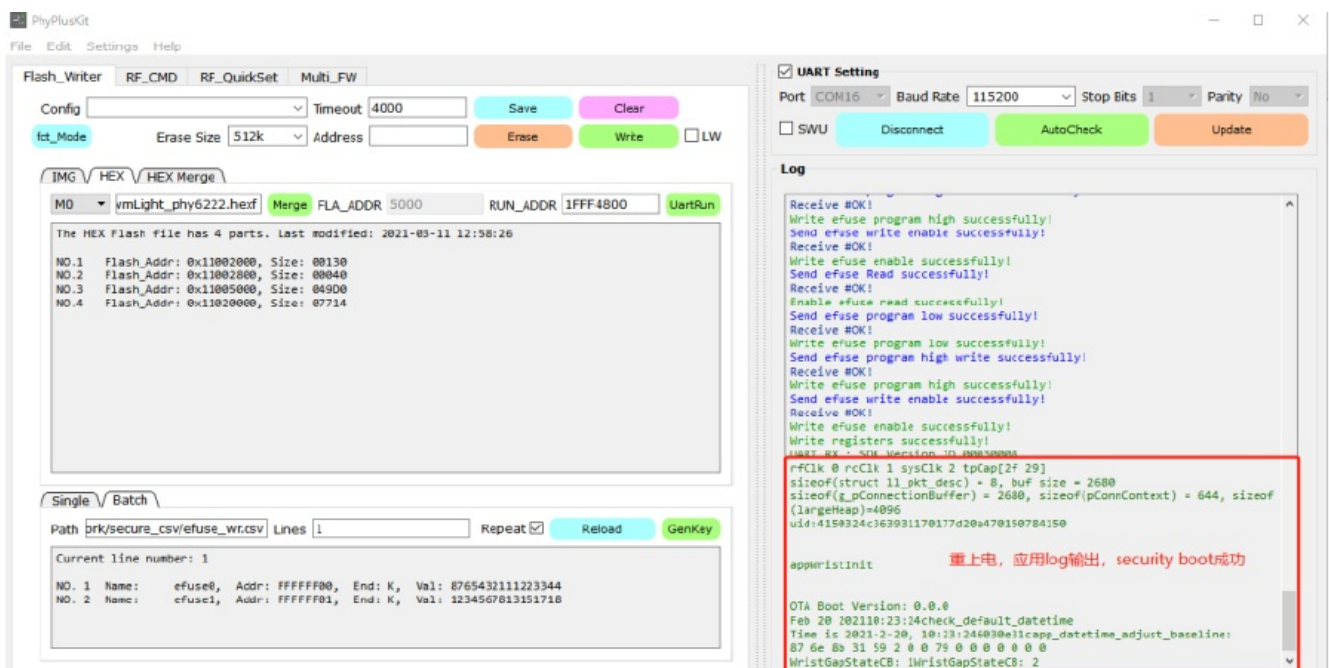
- Switch to the HEX page, select the corresponding hexf file and the efuse_wr.csv file generated by GenKey



- Click the Erase button, after the success of the firmware and efuse key programming



- After the firmware and efuse are successfully programmed, re-power on (TM=0) / TM is pulled low, reset (TM=1), the security boot process goes through, you can jump to the application and complete the ROM security boot process



OTA Security Boot

- The OTA Security boot process is the encrypted boot process of Support OTA. For the specific process and steps, please refer to No OTA mode:
- The difference is that you need to select the ota.hex file and the corresponding single no fct mode, as shown below:

Flash_Writer RF_CMD RF_QuickSet Multi_FW

Config Timeout 4000

fct_Mode Erase Size 512k Address ☐ LW

IMG HEX HEX Merge

BOOT k/example/OTA/OTA_internal_flash/bin/ota.hex Single No FCT

APP ple/ble_peripheral/pwmLight/bin/pwmLight.hex ☒ SEC ☐ Auth SEC_MIC

-- FLA_ADDR

-- FLA_ADDR

-- FLA_ADDR

-- FLA_ADDR

-- FLA_ADDR

ChipID/IV

PID[16] LID[10] TID[14]

MID[16] SID[08] IV[13] 8634785271456

MAC[6] KEY1[32] 1795427e4da595 KEY2[32] e9614eeca663a8

Single Batch

Path /secure_csv/secure2.sec.csv Lines 1 Repeat ☒

```

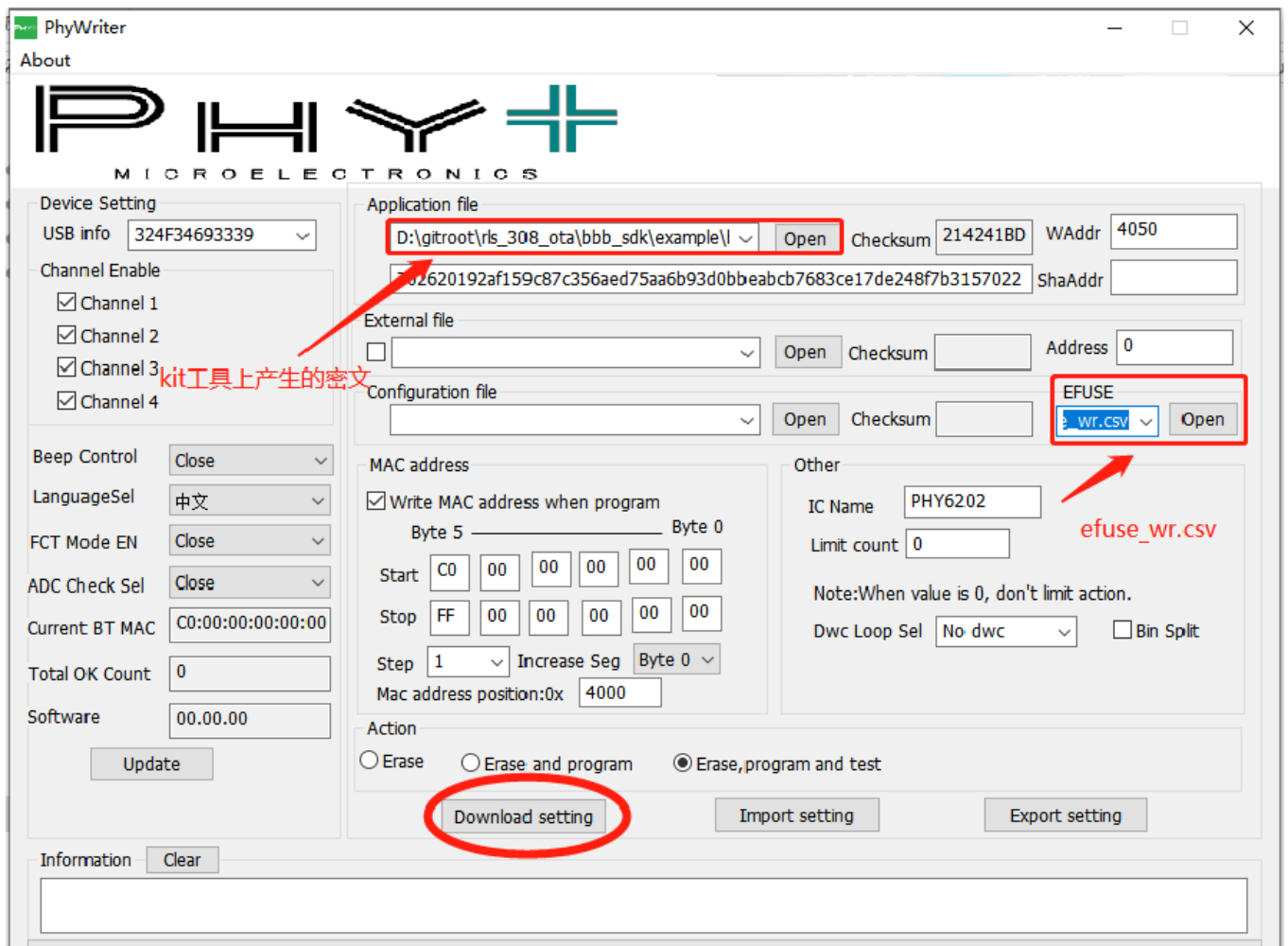
NO. 4 Name: iv, Addr: 02830, End: M, Val: 38363334373835323731343536489574
NO. 5 Name: g_sec_key, Addr: 0FFF0, End: M, Val: 9e7b6c35fff32680954795427e4da595
NO. 6 Name: ota_sec_key, Addr: 02908, End: M, Val: 817e9014a7471cb6
NO. 7 Name: ota_plaintext, Addr: 02910, End: M, Val: e907c7b41754a060d34a62853cb23de8
NO. 8 Name: ota_sec_mic, Addr: 02920, End: M, Val: 07de8c3eb2c3703b762532241cc7ce03
NO. 9 Name: g_ota_sec_key, Addr: 0FFF1, End: M, Val: 0d90351b146426b8ade9614eeca663a8

```

Command: ☐ HEX

Note: The offline programmer security boot only needs to provide the hexf file generate in step c above and the triple *.csv file of the corresponding efuse key generated in step b.

The configuration is as follows:



Flash Mapping


No OTA Mode Flash Mapping

Flash Mapping No OTA						
	256KB Flash			512KB Flash		
Reserved	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

Support OTA Flash Mapping

Single Bank OTA						
	256KB Flash			512KB Flash		
Reserved	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

Documents / Resources

	<p>K-Solution PRBMD02 Bluetooth Tiny 5.0 Module [pdf] User Guide PRBMD02, Bluetooth Tiny 5.0 Module, PRBMD02 Bluetooth Tiny 5.0 Module</p>
---	--