PRBMD02 Application Note

Security Boot User Guide

Disclaimer

Liability Disclaimer
K-Solution Consulting Co. Ltd reserves the right to make changes without further notice to the product to improve reliability, function or design. K-Solution Consulting Co. Ltd does not assume any liability arising out of the application or use of any product or circuits described herein.

Life Support Applications
K-Solution Consulting Co. Ltd's products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. K-Solution Consulting Co. Ltd customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify K-Solution Consulting Co. Ltd for any damages resulting from such improper use or sale.

1. 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 security boot, the method of obtaining the key, the security boot process and the specific operation method of the corresponding mode (No OTA/Support OTA) to perform the secure boot.

2. Efuse Key

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

2.1. 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

2.2. 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>>:)..

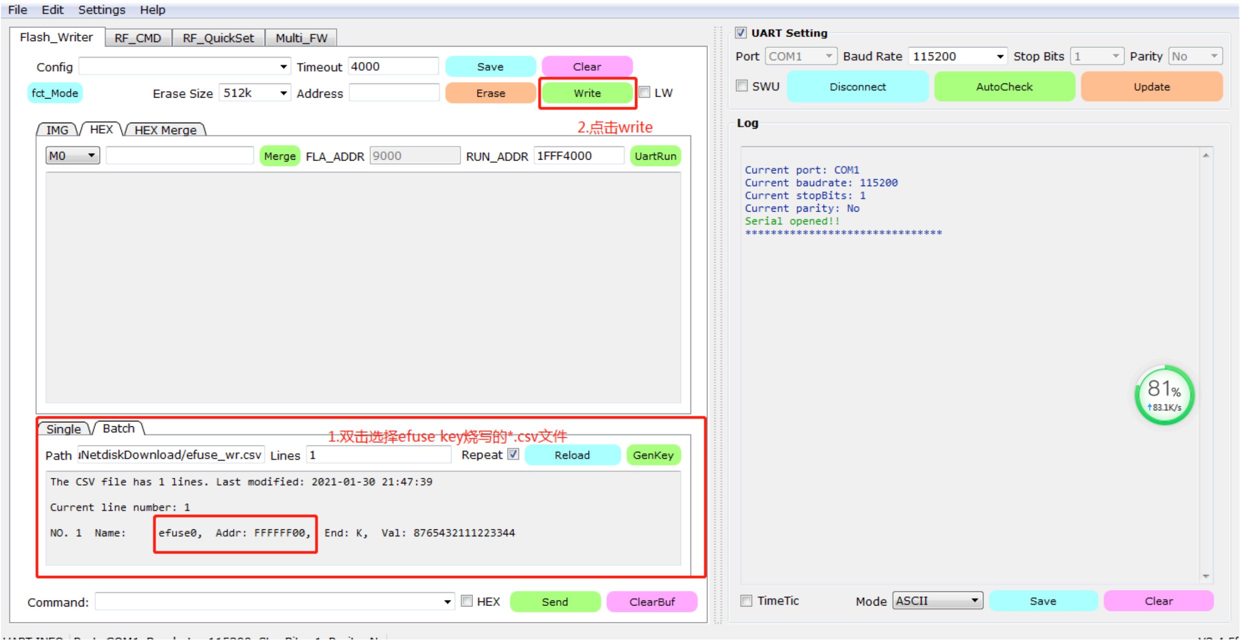
2.2.1. Efuse key programming operation

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):

a. No OTA

#efuse0
FFFFFF00-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:



Config
Timeout 4000
Erase Size 512k
Address
M0
FLA_ADDR 9000
RUN_ADDR 1FFF4000

2.点击write

1.双击选择efuse key烧写的*.csv文件

Path ıNetdiskDownload/efuse_wr.csv Lines 1 Repeat ☑

```
The CSV file has 1 lines. Last modified: 2021-01-30 21:47:39

Current line number: 1

NO. 1  Name:   efuse0,  Addr: FFFFFF00,  End: K,  Val: 8765432111223344
```

Command:

Port COM1 Baud Rate 115200 Stop Bits 1 Parity No

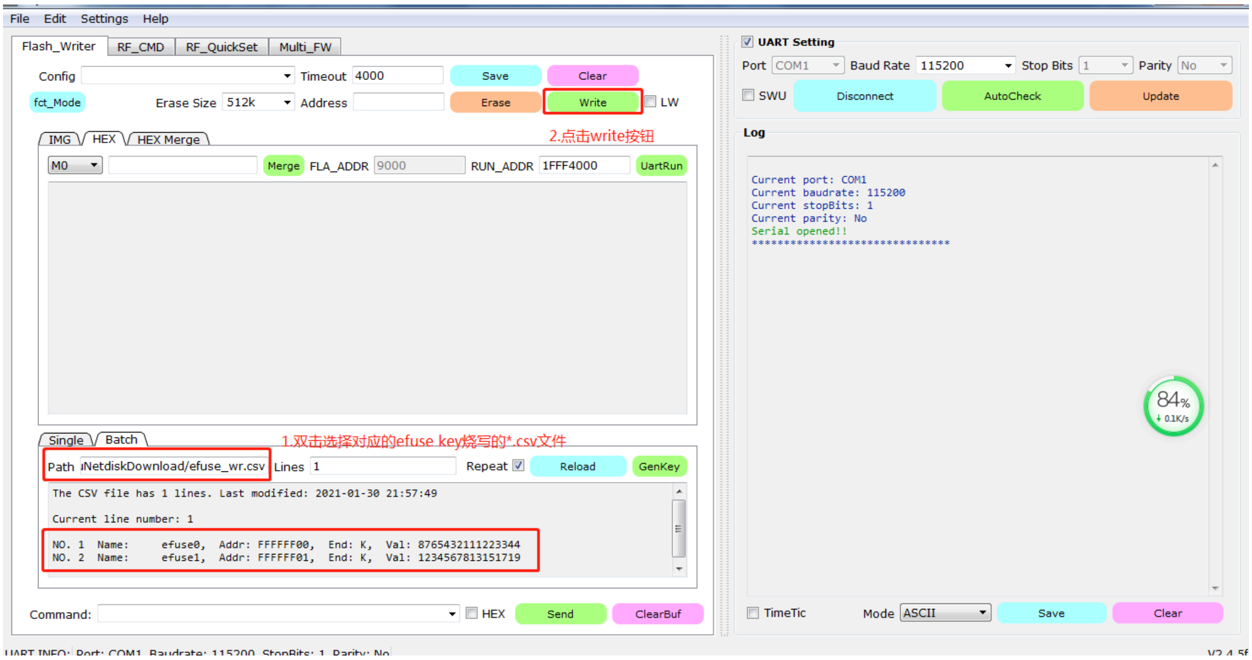
Mode ASCII

```
Current port: COM1
Current baudrate: 115200
Current stopBits: 1
Current parity: No
Serial opened!!
******************************
```

b. Supports OTA

#efuse0	#efuse1
FFFFFF00-K	FFFFFF01-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:



Timeout 4000

Erase Size 512k

Address

2.点击write按钮

M0

FLA_ADDR 9000

RUN_ADDR 1FFF4000

1.双击选择对应的efuse key烧写的*.csv文件

Path ıNetdiskDownload/efuse_wr.csv

Lines 1

Repeat ☑

```
The CSV file has 1 lines. Last modified: 2021-01-30 21:57:49

Current line number: 1

NO. 1  Name:    efuse0,  Addr: FFFFFF00,  End: K,  Val: 8765432111223344
NO. 2  Name:    efuse1,  Addr: FFFFFF01,  End: K,  Val: 1234567813151719
```

Port COM1

Baud Rate 115200

Stop Bits 1

Parity No

```
Current port: COM1
Current baudrate: 115200
Current stopBits: 1
Current parity: No
Serial opened!!
*******************************
```

Mode ASCII

UART INFO: Port: COM1 Baudrate: 115200 StopBits: 1 Parity: No

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 FFFFFF00, FFFFFF01, FFFFFF02, FFFFFF03 according to the block value;
- 3rd line: the write value is the corresponding programmed efuse block value (64bit).

2.2.2. Efuse key programming note.

- The efuse key must be programmed in the programming mode (cmd>>:).
- Efuse 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!

3. 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:

3.1. g_sec_key generation process

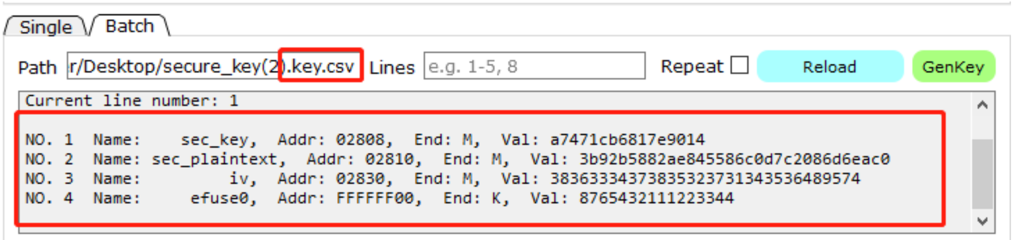
g_sec_key is the secret key used for encryption and decryption by ROM security boot APP (No OTA). The following describes in detail how to generate g_sec_key by using PhyPlusKit.exe tool.

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	3b92b5882ae845586c0d7c2086d6eac0	38363334373835323731343536303030	8765432111223344

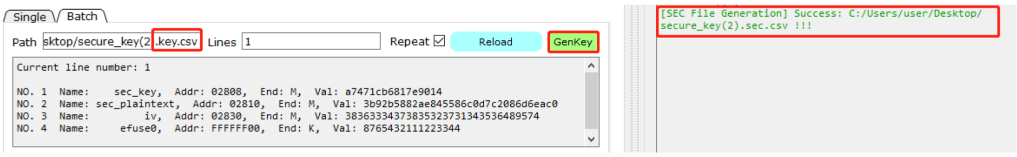
Use the PhyPlusKit.exe (starting from v2.4.5e) tool to generate g_sec_key The method is as follows:

- 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)



```
Current line number: 1

NO. 1  Name:    sec_key,  Addr: 02808,  End: M,  Val: a7471cb6817e9014
NO. 2  Name: sec_plaintext,  Addr: 02810,  End: M,  Val: 3b92b5882ae845586c0d7c2086d6eac0
NO. 3  Name:        iv,  Addr: 02830,  End: M,  Val: 38363334373835323731343536489574
NO. 4  Name:     efuse0,  Addr: FFFFFF00,  End: K,  Val: 8765432111223344
```

- Clicking the GenKey button will generate the *.sec.csv file processed by the efuse 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 lines configuration is to generate *.sec.csv corresponding to the selected row according to the number of rows configuration)



```
Current line number: 1

NO. 1  Name:    sec_key,  Addr: 02808,  End: M,  Val: a7471cb6817e9014
NO. 2  Name: sec_plaintext,  Addr: 02810,  End: M,  Val: 3b92b5882ae845586c0d7c2086d6eac0
NO. 3  Name:        iv,  Addr: 02830,  End: M,  Val: 38363334373835323731343536489574
NO. 4  Name:     efuse0,  Addr: FFFFFF00,  End: K,  Val: 8765432111223344
```

```
[SEC File Generation] Success: C:/Users/user/Desktop/
secure_key(2).sec.csv !!!
```

The *.sec.csv file generated by clicking the GenKey button will generate g_sec_key accordingly.

3.2. 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
a7471cb6817e9014	3b92b5882ae845586c0d7c2086d6eac0	38363334373835323731343536303030	8765432111223344	817e9014a7471cb6	e907c7b41754a060d34a62853cb23de8	1234567813151718

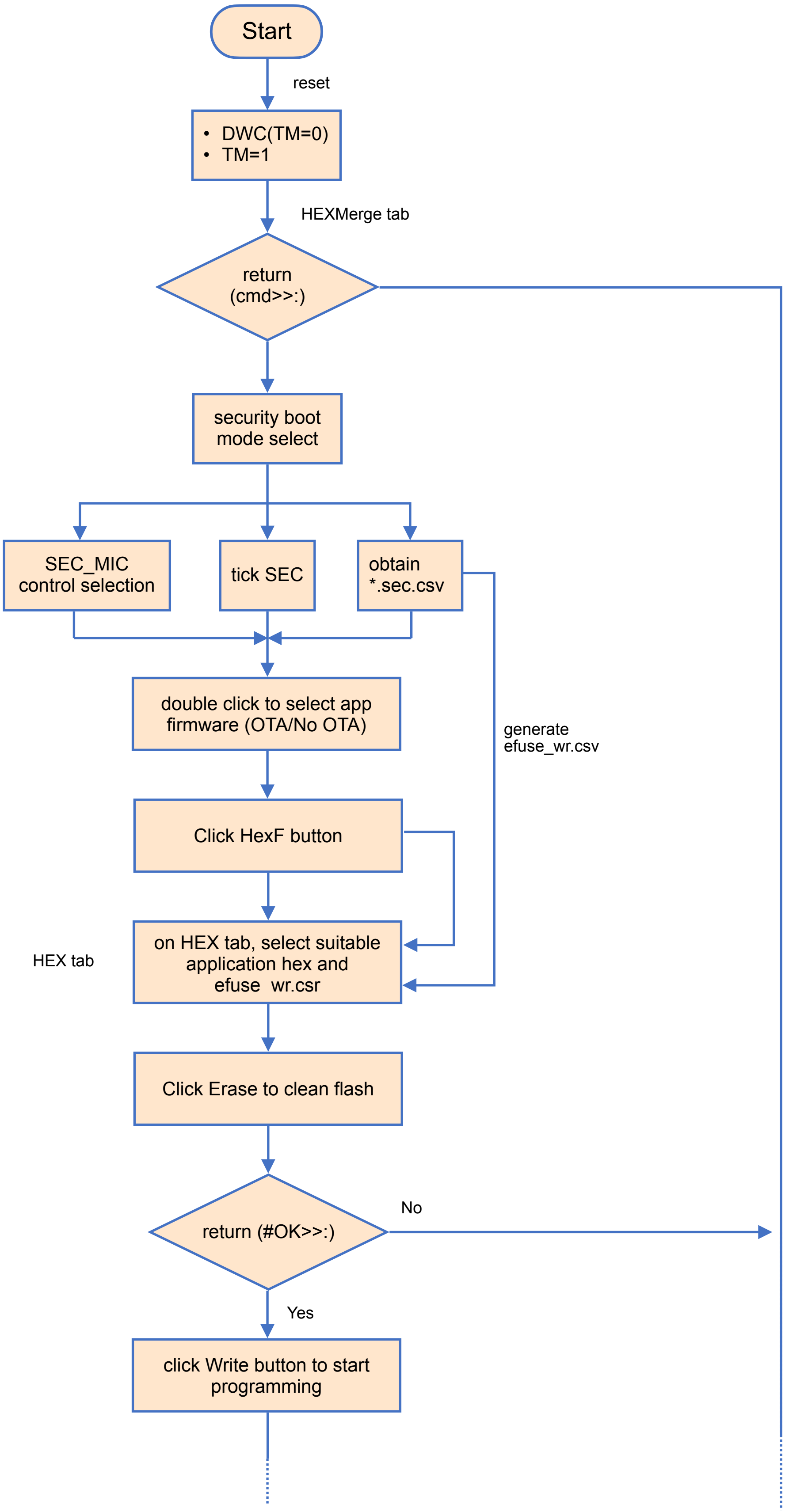
The method of generating g_sec_key and g_ota_sec_key by using PhyPlusKit.exe (starting from v2.4.5e) can refer to the generation process of g_sec_key in Section 3.1, but the content of the *.key.csv file is different.

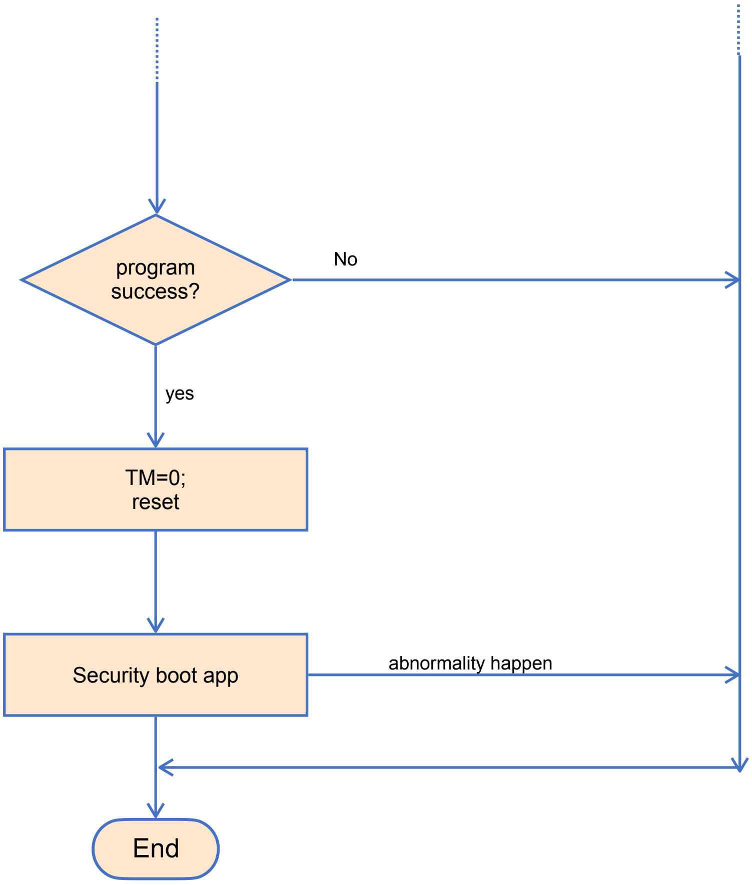
The *.sec.csv file corresponding to the same operation method will generate g_sec_key and g_ota_sec_key correspondingly.

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 efuse key programming and efuse_wr.csv file have been introduced in detail in Section 2.2.

4. 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:


Start
reset
• DWC(TM=0)
• TM=1
HEXMerge tab
return (cmd>>:)
security boot mode select
SEC_MIC control selection
tick SEC
obtain *.sec.csv
generate efuse_wr.csv
double click to select app firmware (OTA/No OTA)
Click HexF button
HEX tab
on HEX tab, select suitable application hex and efuse_wr.csr
Click Erase to clean flash
return (#OK>>:)
No
Yes
click Write button to start programming


program success?
No
yes
TM=0; reset
Security boot app
abnormality happen
End

4.1.Operation flow

I. 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>>:

II. 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.

III. Select the application firmware to be programmed, including No OTA/Support OTA mode and click the HexF button to generate the corresponding ciphertext hexf file

IV. Switch to the HEX page, select the hexf file and efuse_wr.csv file generated above

V. Click the Erase button to send the erase command, after success, click the write button to program the firmware and efuse

VI. 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.

4.2. ROM Security Boot

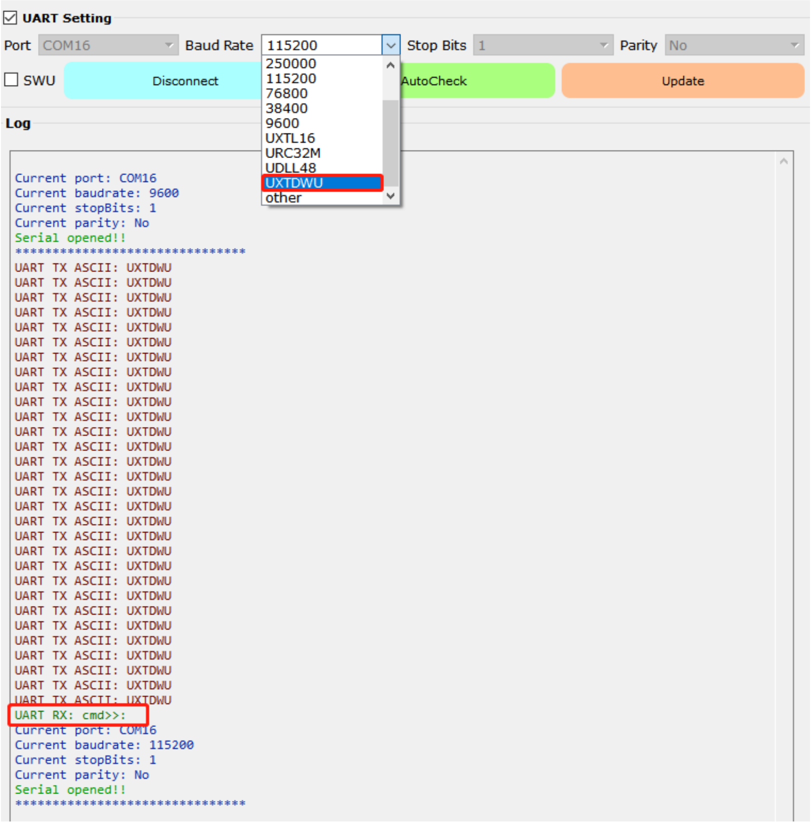
The ROM Security boot process is the encrypted boot process of No OTA.

PhyPlusKit.exe tool V2.4.5e version, support security boot function, this function module is supported in selecting SEC_MIC mode. Select the corresponding SEC_MIC form to use the security boot function.

The operation steps are as follows:

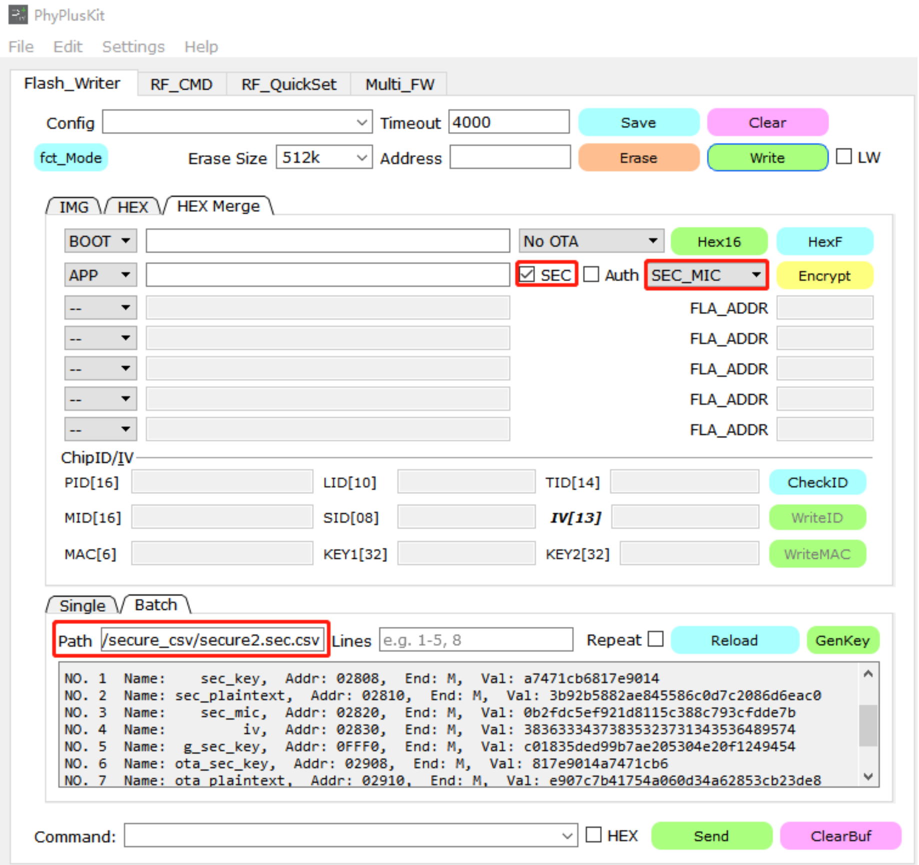
i. After PHY6252/PHY6222 is powered on, re-power on through DWC connection (TM=0)/TM=1 (pull TM high), Reset the development board, enter programming mode, and return to cmd>>:

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



```
Current port: COM16
Current baudrate: 9600
Current stopBits: 1
Current parity: No
Serial opened!!
*******************************
UART TX ASCII: UXTDWU
UART TX ASCII: UXTDWU
UART TX ASCII: UXTDWU
UART TX ASCII: UXTDWU
UART TX ASCII: UXTDWU
UART TX ASCII: UXTDWU
UART TX ASCII: UXTDWU
UART TX ASCII: UXTDWU
UART TX ASCII: UXTDWU
UART TX ASCII: UXTDWU
UART TX ASCII: UXTDWU
UART TX ASCII: UXTDWU
UART TX ASCII: UXTDWU
UART TX ASCII: UXTDWU
UART TX ASCII: UXTDWU
UART TX ASCII: UXTDWU
UART TX ASCII: UXTDWU
UART TX ASCII: UXTDWU
UART TX ASCII: UXTDWU
UART TX ASCII: UXTDWU
UART TX ASCII: UXTDWU
UART TX ASCII: UXTDWU
UART TX ASCII: UXTDWU
UART TX ASCII: UXTDWU
UART TX ASCII: UXTDWU
UART TX ASCII: UXTDWU
UART TX ASCII: UXTDWU
UART TX ASCII: UXTDWU
UART TX ASCII: UXTDWU
UART TX ASCII: UXTDWU
UART RX: cmd>>:
Current port: COM16
Current baudrate: 115200
Current stopBits: 1
Current parity: No
Serial opened!!
*******************************
```

ii. 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



Config

Timeout 4000

Erase Size 512k

Address

□ LW

BOOT | No OTA

APP | ☑ SEC □ Auth SEC_MIC

-- FLA_ADDR (×5)

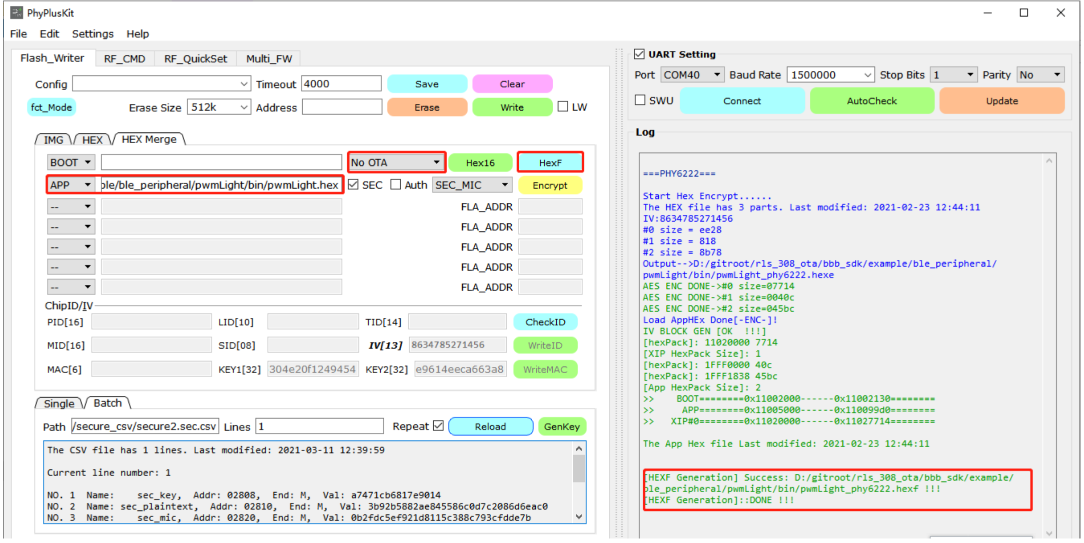
ChipID/IV: PID[16], LID[10], TID[14], MID[16], SID[08], *IV[13]*, MAC[6], KEY1[32], KEY2[32]

Path /secure_csv/secure2.sec.csv Lines e.g. 1-5, 8 Repeat □

```
NO. 1  Name:    sec_key,  Addr: 02808,  End: M,  Val: a7471cb6817e9014
NO. 2  Name: sec_plaintext,  Addr: 02810,  End: M,  Val: 3b92b5882ae845586c0d7c2086d6eac0
NO. 3  Name:    sec_mic,  Addr: 02820,  End: M,  Val: 0b2fdc5ef921d8115c388c793cfdde7b
NO. 4  Name:         iv,  Addr: 02830,  End: M,  Val: 38363334373835323731343536489574
NO. 5  Name:  g_sec_key,  Addr: 0FFF0,  End: M,  Val: c01835ded99b7ae205304e20f1249454
NO. 6  Name: ota_sec_key,  Addr: 02908,  End: M,  Val: 817e9014a7471cb6
NO. 7  Name: ota_plaintext,  Addr: 02910,  End: M,  Val: e907c7b41754a060d34a62853cb23de8
```

Command: □ HEX

iii. Double-click to select the application firmware (No OTA), click the HexF button to generate the corresponding hexf file



Config

Timeout 4000

Erase Size 512k

Address

□ LW

BOOT | No OTA

APP ble/ble_peripheral/pwmLight/bin/pwmLight.hex | ☑ SEC □ Auth SEC_MIC

-- FLA_ADDR (×5)

ChipID/IV: PID[16], LID[10], TID[14], MID[16], SID[08], *IV[13]* 8634785271456, MAC[6], KEY1[32] 304e20f1249454, KEY2[32] e9614eeca663a8

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

```
The CSV file has 1 lines. Last modified: 2021-03-11 12:39:59

Current line number: 1

NO. 1  Name:    sec_key,  Addr: 02808,  End: M,  Val: a7471cb6817e9014
NO. 2  Name: sec_plaintext,  Addr: 02810,  End: M,  Val: 3b92b5882ae845586c0d7c2086d6eac0
NO. 3  Name:    sec_mic,  Addr: 02820,  End: M,  Val: 0b2fdc5ef921d8115c388c793cfdde7b
```

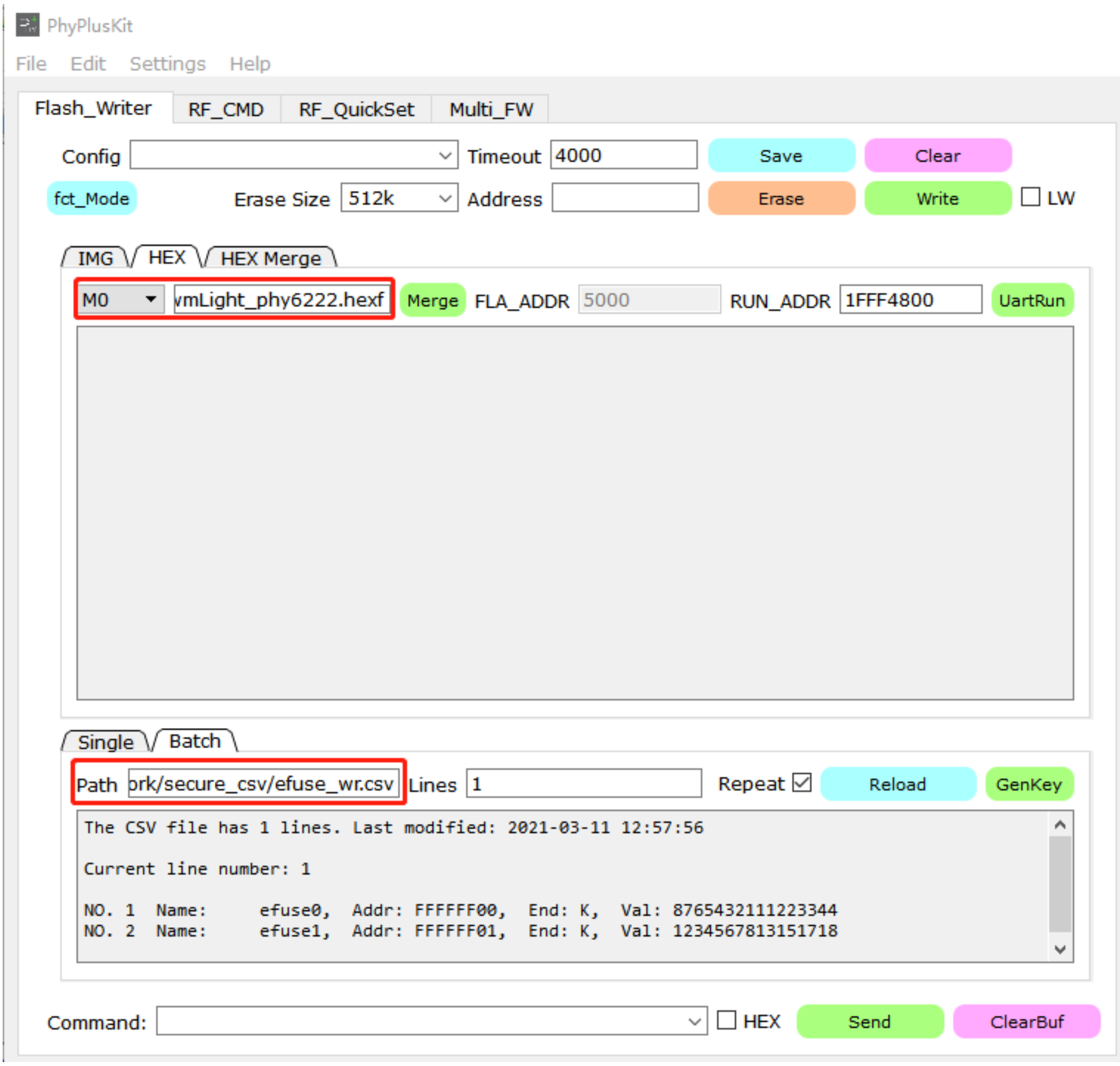
☑ UART Setting

Port COM40 Baud Rate 1500000 Stop Bits 1 Parity No

□ SWU

```
===PHY6222===

Start Hex Encrypt......
The HEX file has 3 parts. Last modified: 2021-02-23 12:44:11
IV:8634785271456
#0 size = ee28
#1 size = 818
#2 size = 8b78
Output-->D:/gitroot/rls_308_ota/bbb_sdk/example/ble_peripheral/
pwmLight/bin/pwmLight_phy6222.hexe
AES ENC DONE->#0 size=07714
AES ENC DONE->#1 size=0040c
AES ENC DONE->#2 size=045bc
Load AppHEx Done[-ENC-]!
IV BLOCK GEN [OK  !!!]
[hexPack]: 11020000 7714
[XIP HexPack Size]: 1
[hexPack]: 1FFF0000 40c
[hexPack]: 1FFF1838 45bc
[App HexPack Size]: 2
>>    BOOT========0x11002000------0x11002130========
>>     APP========0x11005000------0x110099d0========
>>   XIP#0========0x11020000------0x11027714========

The App Hex file Last modified: 2021-02-23 12:44:11


[HEXF Generation] Success: D:/gitroot/rls_308_ota/bbb_sdk/example/
ble_peripheral/pwmLight/bin/pwmLight_phy6222.hexf !!!
[HEXF Generation]::DONE !!!
```

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

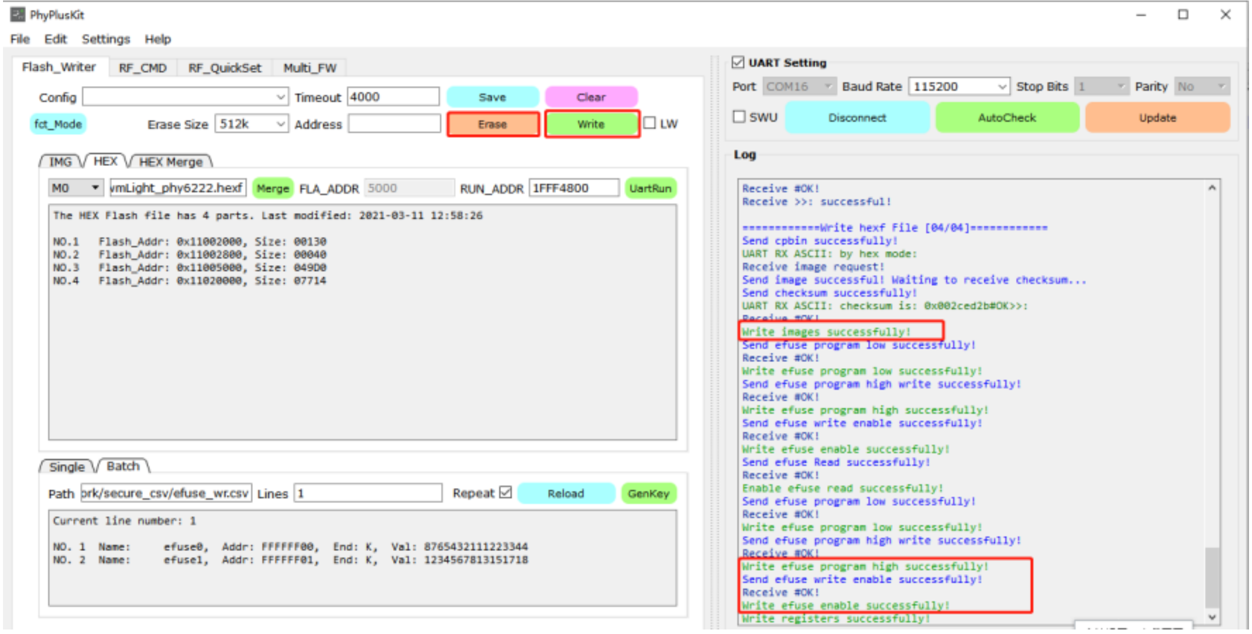


Config
Timeout 4000
Erase Size 512k
Address
M0 vmLight_phy6222.hexf
FLA_ADDR 5000
RUN_ADDR 1FFF4800
Path prk/secure_csv/efuse_wr.csv
Lines 1
Repeat ☑

```
The CSV file has 1 lines. Last modified: 2021-03-11 12:57:56

Current line number: 1

NO. 1  Name:    efuse0,  Addr: FFFFFF00,  End: K,  Val: 8765432111223344
NO. 2  Name:    efuse1,  Addr: FFFFFF01,  End: K,  Val: 1234567813151718
```

Command:

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



Config
Timeout 4000
Erase Size 512k
Address
M0 vmLight_phy6222.hexf
FLA_ADDR 5000
RUN_ADDR 1FFF4800

```
The HEX Flash file has 4 parts. Last modified: 2021-03-11 12:58:26

NO.1   Flash_Addr: 0x11002000, Size: 00130
NO.2   Flash_Addr: 0x11002800, Size: 00040
NO.3   Flash_Addr: 0x11005000, Size: 049D0
NO.4   Flash_Addr: 0x11020000, Size: 07714
```

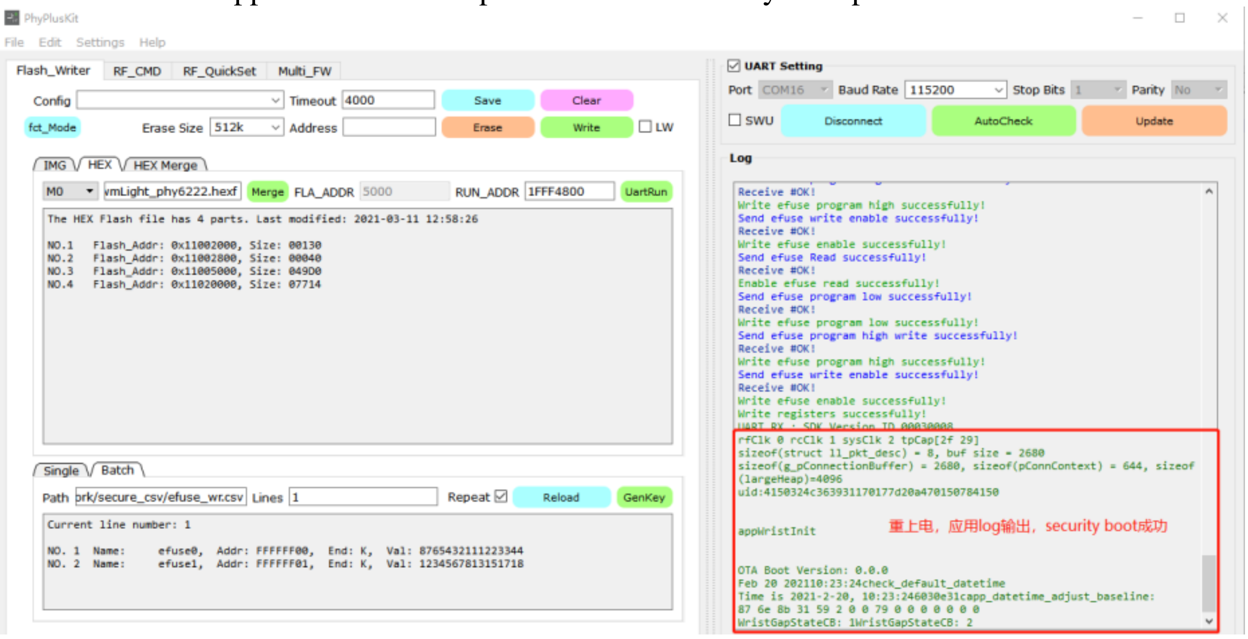
Path prk/secure_csv/efuse_wr.csv
Lines 1
Repeat ☑

```
Current line number: 1

NO. 1  Name:    efuse0,  Addr: FFFFFF00,  End: K,  Val: 8765432111223344
NO. 2  Name:    efuse1,  Addr: FFFFFF01,  End: K,  Val: 1234567813151718
```

Port COM16
Baud Rate 115200
Stop Bits 1
Parity No

```
Receive #OK!
Receive >>: successful!

============Write hexf File [04/04]============
Send cpbin successfully!
UART RX ASCII: by hex mode:
Receive image request!
Send image successful! Waiting to receive checksum...
Send checksum successfully!
UART RX ASCII: checksum is: 0x002ced2b#OK>>:
Receive #OK!
Write images successfully!
Send efuse program low successfully!
Receive #OK!
Write efuse program low successfully!
Send efuse program high write successfully!
Receive #OK!
Write efuse program high successfully!
Send efuse write enable successfully!
Receive #OK!
Write efuse enable successfully!
Send efuse Read successfully!
Receive #OK!
Enable efuse read successfully!
Send efuse program low successfully!
Receive #OK!
Write efuse program low successfully!
Send efuse program high write successfully!
Receive #OK!
Write efuse program high successfully!
Send efuse write enable successfully!
Receive #OK!
Write efuse enable successfully!
Write registers successfully!
```

vi. 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



Config
Timeout 4000
Erase Size 512k
Address
□ LW

M0 vmLight_phy6222.hexf FLA_ADDR 5000 RUN_ADDR 1FFF4800

```
The HEX Flash file has 4 parts. Last modified: 2021-03-11 12:58:26

NO.1   Flash_Addr: 0x11002000, Size: 00130
NO.2   Flash_Addr: 0x11002800, Size: 00040
NO.3   Flash_Addr: 0x11005000, Size: 049D0
NO.4   Flash_Addr: 0x11020000, Size: 07714
```

Path ork/secure_csv/efuse_wr.csv Lines 1 Repeat ☑

```
Current line number: 1

NO. 1  Name:     efuse0,  Addr: FFFFFF00,  End: K,  Val: 8765432111223344
NO. 2  Name:     efuse1,  Addr: FFFFFF01,  End: K,  Val: 1234567813151718
```

☑ UART Setting

Port COM16 Baud Rate 115200 Stop Bits 1 Parity No

□ SWU

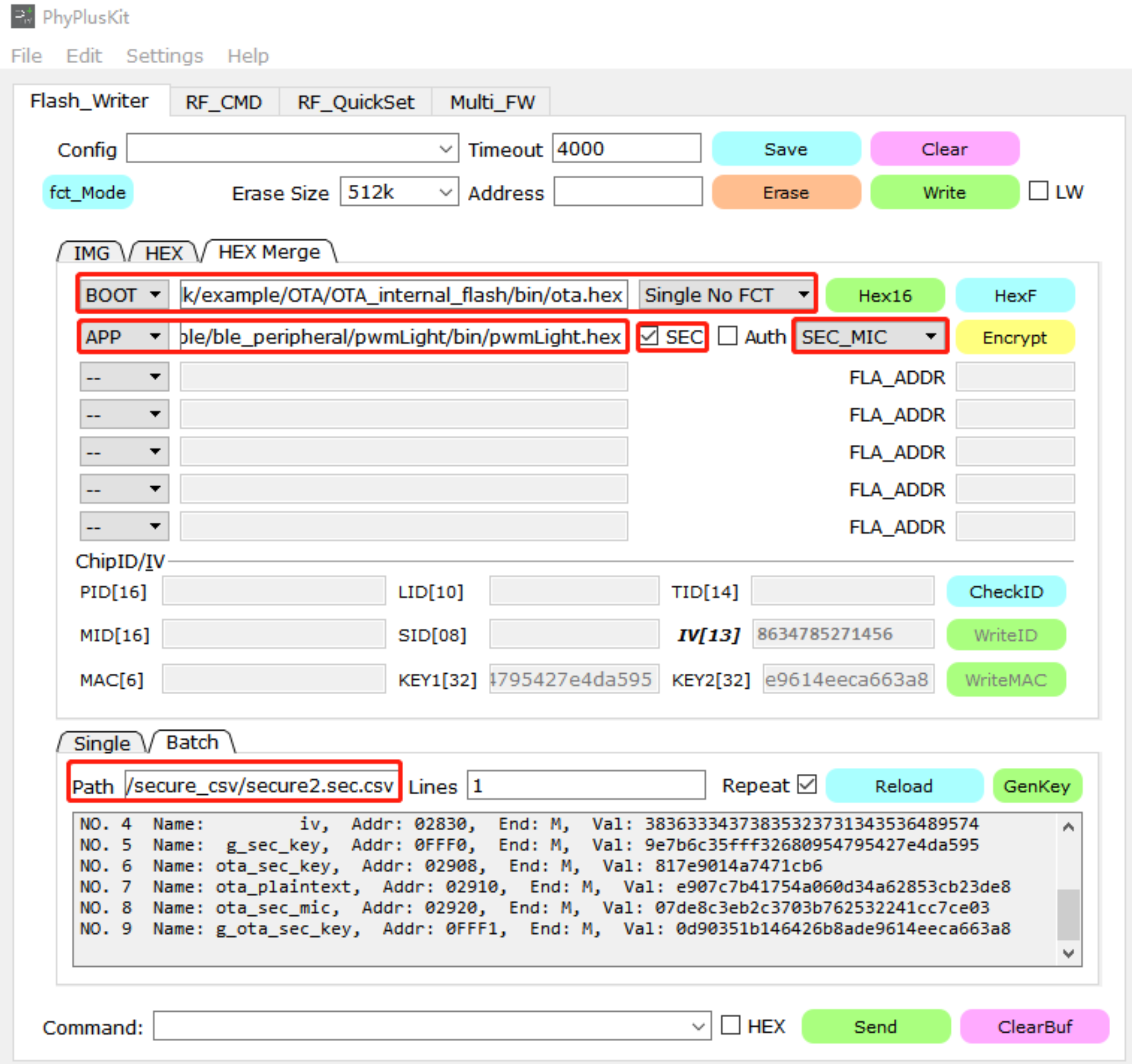
```
Receive #OK!
Write efuse program high successfully!
Send efuse write enable successfully!
Receive #OK!
Write efuse enable successfully!
Send efuse Read successfully!
Receive #OK!
Enable efuse read successfully!
Send efuse program low successfully!
Receive #OK!
Write efuse program low successfully!
Send efuse program high write successfully!
Receive #OK!
Write efuse program high successfully!
Send efuse write enable successfully!
Receive #OK!
Write efuse enable successfully!
Write registers successfully!
UART RX : SDK Version ID 00030008
rfClk 0 rcClk 1 sysClk 2 tpCap[2f 29]
sizeof(struct ll_pkt_desc) = 8, buf size = 2680
sizeof(g_pConnectionBuffer) = 2680, sizeof(pConnContext) = 644, sizeof
(largeHeap)=4096
uid:4150324c363931170177d20a470150784150


appWristInit


OTA Boot Version: 0.0.0
Feb 20 202110:23:24check_default_datetime
Time is 2021-2-20, 10:23:246030e31capp_datetime_adjust_baseline:
87 6e 8b 31 59 2 0 0 79 0 0 0 0 0 0 0
WristGapStateCB: 1WristGapStateCB: 2
```

重上电，应用log输出，security boot成功

4.7. 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:



Timeout 4000

Erase Size 512k

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

APP ble/ble_peripheral/pwmLight/bin/pwmLight.hex ☑ SEC □ Auth SEC_MIC

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

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

MAC[6] KEY1[32] 4795427e4da595 KEY2[32] e9614eeca663a8

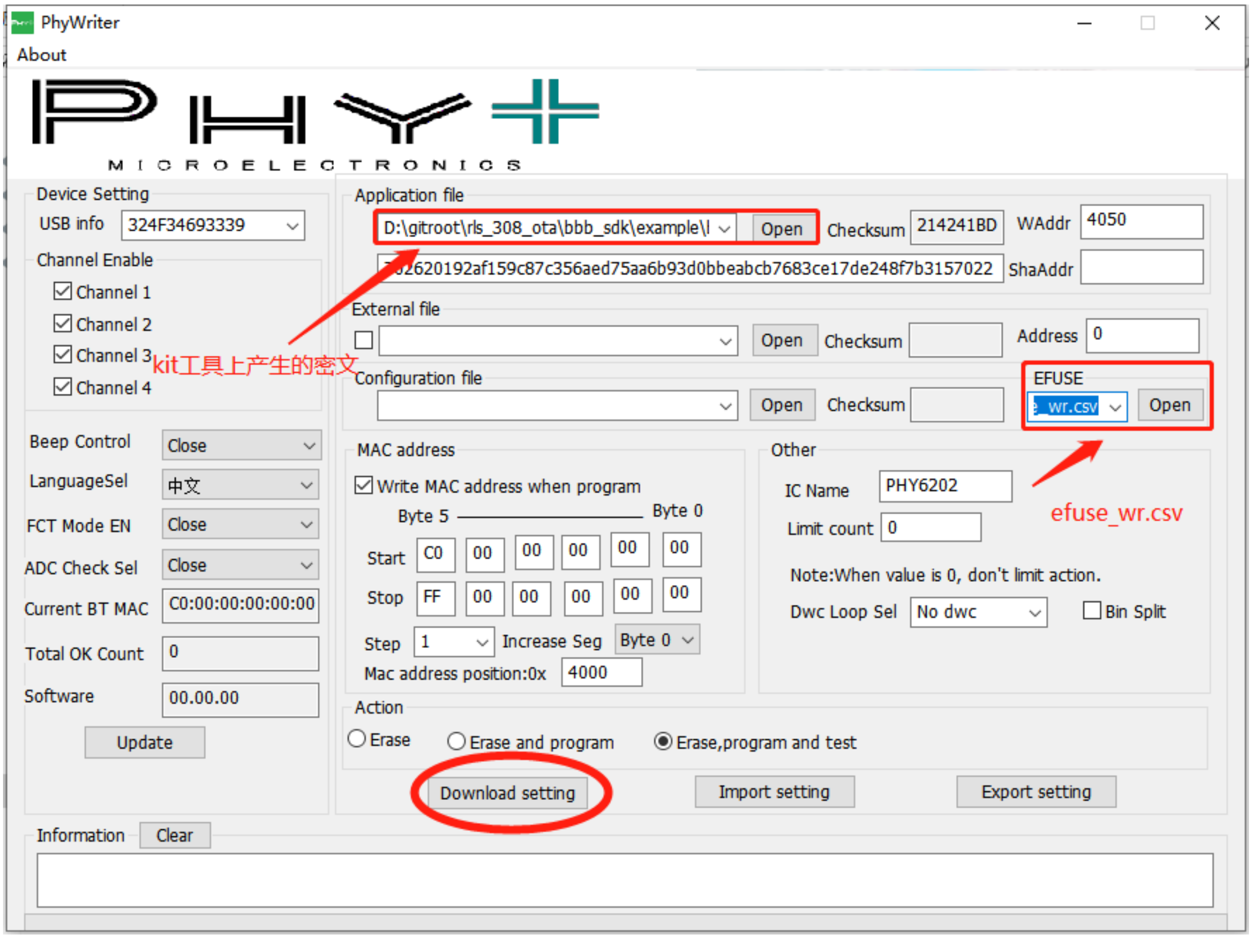
Path /secure_csv/secure2.sec.csv Lines 1 Repeat ☑

```
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 generated in step c above and the triple *.csv file of the corresponding efuse key generated in step b.

The configuration is as follows:



Device Setting

USB info 324F34693339

Channel Enable

☑ Channel 1

☑ Channel 2

☑ Channel 3

☑ Channel 4

Beep Control: Close

LanguageSel: 中文

FCT Mode EN: Close

ADC Check Sel: Close

Current BT MAC: C0:00:00:00:00:00

Total OK Count: 0

Software: 00.00.00

Application file: D:\gitroot\rls_308_ota\bbb_sdk\example\l

Checksum: 214241BD

WAddr: 4050

2620192af159c87c356aed75aa6b93d0bbeabcb7683ce17de248f7b3157022

ShaAddr:

External file: □

Address: 0

Configuration file:

EFUSE: e_wr.csv

MAC address

☑ Write MAC address when program

	Byte 5					Byte 0
Start	C0	00	00	00	00	00
Stop	FF	00	00	00	00	00

Step: 1 Increase Seg: Byte 0

Mac address position:0x 4000

Other

IC Name: PHY6202

Limit count: 0

Note:When value is 0, don't limit action.

Dwc Loop Sel: No dwc

□ Bin Split

Action

○ Erase ○ Erase and program ● Erase,program and test

kit工具上产生的密文

efuse_wr.csv

5. Flash Mapping

5.1. 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

5.2. 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