

奉加微电子

# PhyPlusKit 使用说明

Version 2.3

## 版本控制信息

版本/状态	作者	起止日期	备注
V2.0	DMC	7/05/2018	文档初稿
V2.3	ZQ	8/21/2018	1. 加入 2.6 Flash Write Hex Merge 功能描述 2. 加入 3.2.4 Hex Merge 烧写示例
V2.3.2d	NHL	9/05/2018	1. 1.1 模块组成部分有小修改 2. 加入 2.7 命令行模式和 2.8MAC 地址 3. 加入 3.6 命令行模式下的烧写和合并操作
V2.3.7f	HXR	11/6/2019	1. BOOT 和 APP 支持中文路径 2. Uart 不连接时 chip version 的自选设置 3. Uart 连接时的 setting 设置 4. 脱机烧录 1M flash 烧录文件的合成
V2.3.8a	HXR	22/6/2019	1. 擦除功能中添加 Preserve 的地址区域保留功能 2. Preserve 模式中的指定多地址段的擦除功能
V2.3.8b	HXR	28/6/2019	1. 增加 PHY_fct_Mode 功能
V2.3.8c	HXR	9/8/2019	1. 在 2.7 中增加命令行模式的功能 2. 修改 3.6 中命令行模式下的烧写操作
V2.4.1a	HXR	19/9/2019	1. 脱机烧录两个 hexf 文件的合成 2. 支持外挂 flash 的烧写（IMG 页面）
V2.4.2c	HXR	7/21/2020	1. 支持单线烧录功能 2. Security boot
V2.4.5a	HXR	9/17/2020	3. 支持 hexf 解析来做保留擦除

## 目录

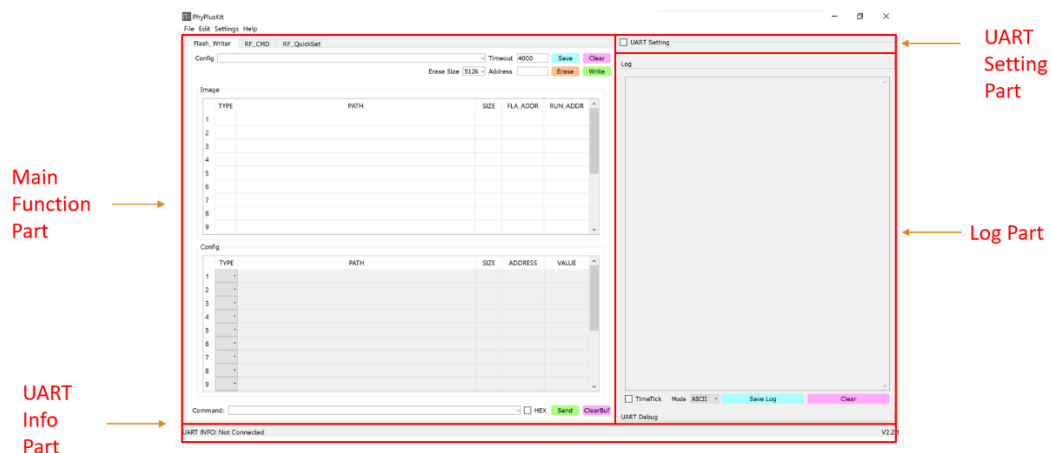
1. 模块介绍.....	4
1.1 模块组成.....	4
1.2 模块详述.....	5
1.2.1 主功能区.....	5
1.2.2 串口设置.....	5
1.2.3 日志显示.....	6
1.2.4 串口信息.....	6
2. 功能详述.....	7
2.1 UART 连接与设置.....	7
2.2 Flash 烧写 .....	8
2.3 RF Command 命令发送 .....	11
2.4 RF_QuickSet 快捷命令 .....	12
2.5 Multi_FlashWriter 多串口烧写 .....	13
2.6 Flash_Writer HEX Merge .....	14
2.7 命令行模式.....	16
2.8 MAC 地址.....	18
3. 功能示例.....	19
3.1. UART 速率获取和更新 .....	19
3.1.1. UART 速率更新 .....	19
3.1.2. UART 速率获取 .....	20
3.2. Flash 烧写 .....	21
3.2.1 仅有 hex 的烧写 (推荐此模式).....	21
3.2.2 仅有 Image 烧写.....	24
3.2.3 有 image 和 config 的烧写 .....	28
3.2.4 用 HexMerge 进行烧录 .....	30
3.3 RF Command 使用 .....	33
3.3.1 RF Command TX .....	33
3.3.2 RF Command RX .....	34
3.4 RF QuickSet 使用 .....	37
3.5 Multi-FlashWriter 使用 .....	40
3.6 命令行模式下的烧写和合并操作.....	42
3.6.1 仅烧写.....	42
3.6.2 仅合并.....	43
3.6.3 合并后烧写.....	44
3.7. Flash 烧写的新增配置 .....	44
3.7.1 BOOT 和 APP 支持中文路径 .....	44
3.7.2 Uart 不连接时的自选设置.....	45
3.7.3 Uart 连接时的烧写配置.....	46
3.7.4 脱机烧录 1M flash 烧录文件的合成.....	48
3.7.5 Preserve 模式的地址段保留和多地址段的擦除 .....	53

3.7.6	PHY_fct_Mode 功能 .....	54
3.7.7	支持外挂 flash 的烧写功能 .....	56
3.7.8	HEXF 解析的保留擦除模式 .....	57
3.7.9	DWC 连接 .....	57
3.7.10	Security Boot.....	60
3.7.11	block check .....	63

# 1. 模块介绍

## 1.1 模块组成

本软件由四部分组成，主功能区、串口设置区、日志显示区和串口信息区。另外本软件支持命令行模式，在命令行模式下可以通过添加相关的参数调用来直接执行一些命令。命令行模式的相关介绍详见 2.6 和 3.6 节。



(1)主功能区为 Flash\_Writer、RF\_CMD、RF\_QuickSet 四个部分。

- Flash\_Writer 进行 image 文件的烧写；
- RF\_CMD 用来发送 HCI 协议的命令；
- RF\_QuickSet 用来发送快捷的测试命令；

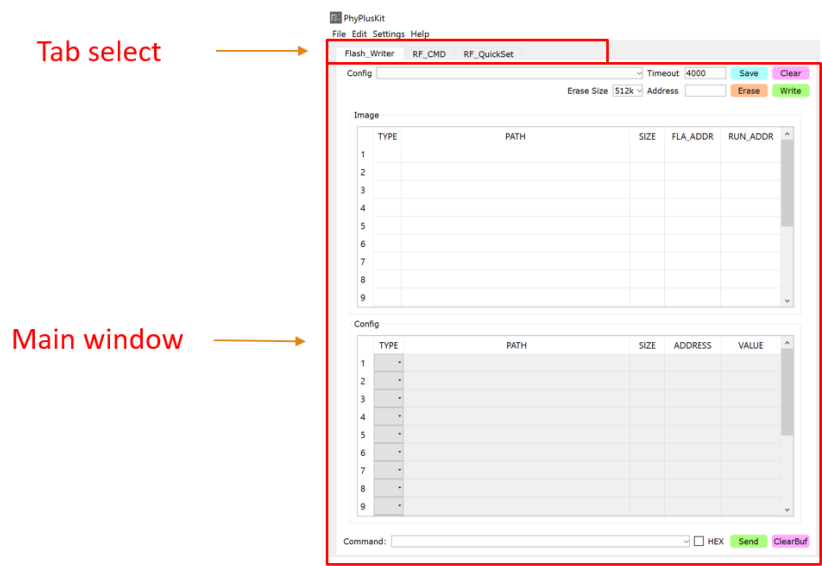
(2)串口设置区，UART 对串口进行设置和连接；

(3)日志显示区，可以对用户的操作及设备的通讯信息进行打印和保存。

(4)串口信息区，可以对串口的实时设置进行显示。

## 1.2 模块详述

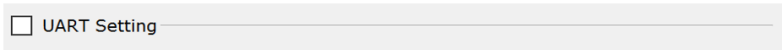
### 1.2.1 主功能区



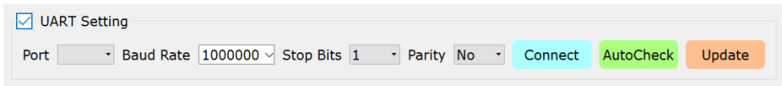
用户可以通过 Tab Select 切换功能；并在主窗口（Main Window）位置进行相应操作。

### 1.2.2 串口设置

Hide UART setting



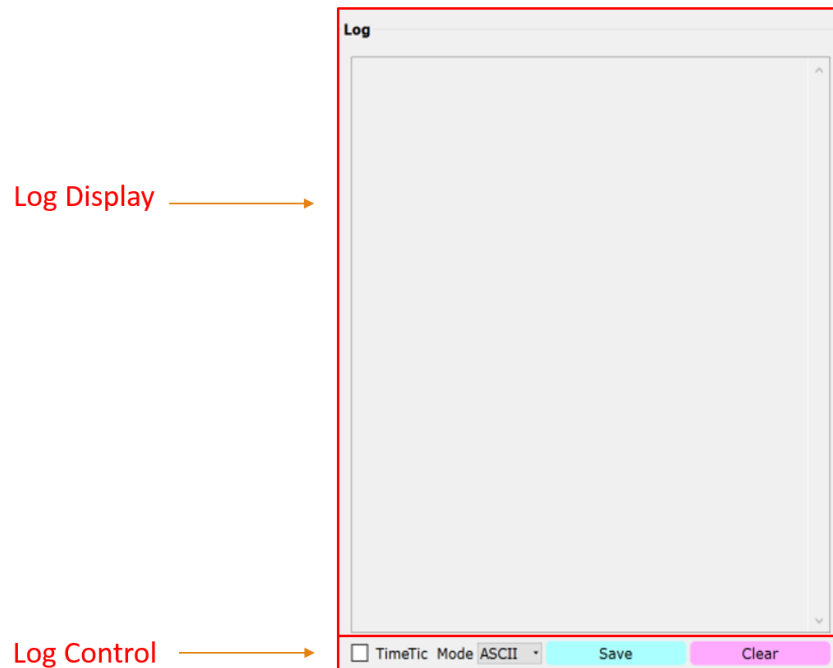
Show UART setting



在串口设置区可以隐藏或显示设置参数，并进行连接/断开操作。

- 可进行任意波特率、停止位、校验位的串口连接；
- 可自动检测下位机所设置的通信波特率；
- 可向下位机发送波特率更改命令，修改通信波特率；

### 1.2.3 日志显示



- 日志显示（Log Display）：打印出用户操作以及串口通讯信息；
- 日志控制（Log Control）：可以开启对不同 RX 内容的解析支持，以及时间戳，也可直接保存 log 内容到任意位置；

### 1.2.4 串口信息

#### Not Connect UART

UART INFO: Not Connected V1.2.0

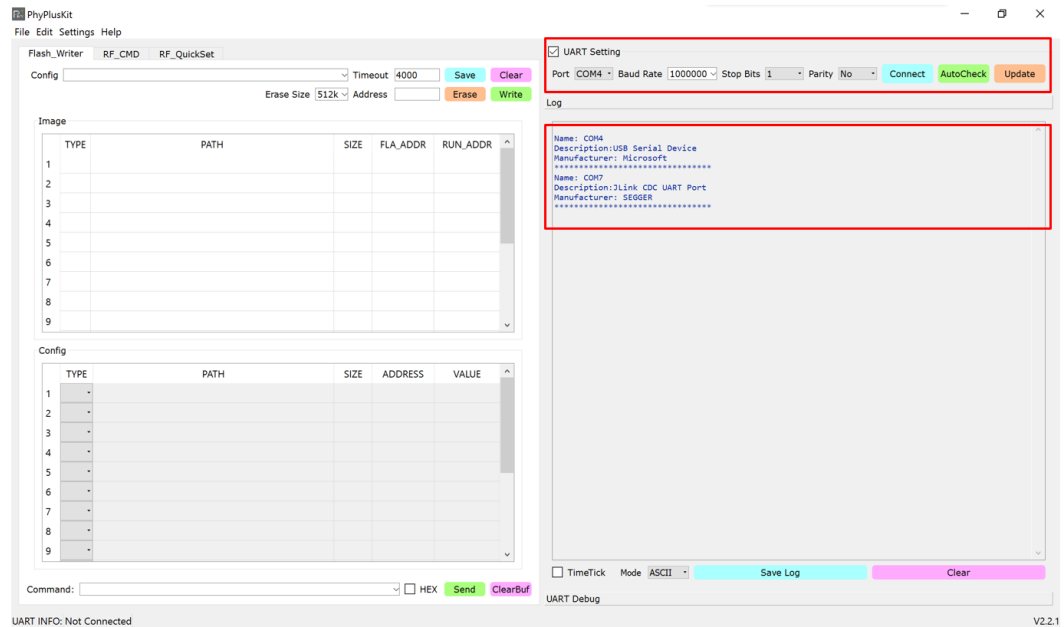
#### Connect UART

UART INFO: Port: COM4, Baudrate: 250000, StopBits: 1, Parity: No Parity V1.2.0

在串口信息区 显示串口的设置信息，以及当前的版本号。

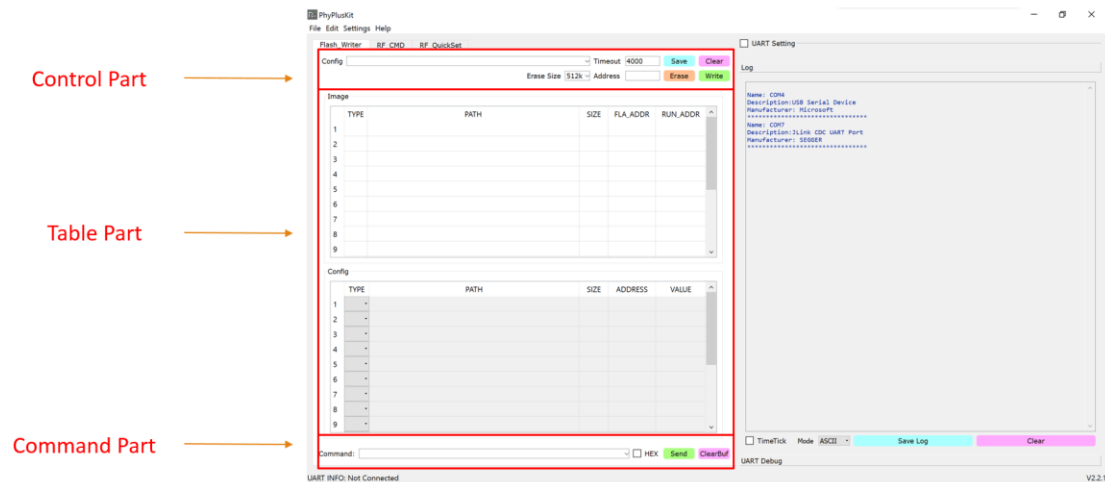
## 2. 功能详述

### 2.1 UART 连接与设置



- 点击 Port 下拉框，可获取所有可用的串口名并在 Log 中输出可用串口的信息；
- 可以设置串口波特率、停止位和校验位；
- 点击 Connect 按钮，即可按自定义设置连接所选串口，并在 UART Info 中显示当前串口设置参数信息；
- 点击 AutoCheck 按钮，可自动发送命令检测下位机通信波特率；
- 点击 Update 按钮，可向下位机发送波特率修改命令，同时修改软件通信波特率

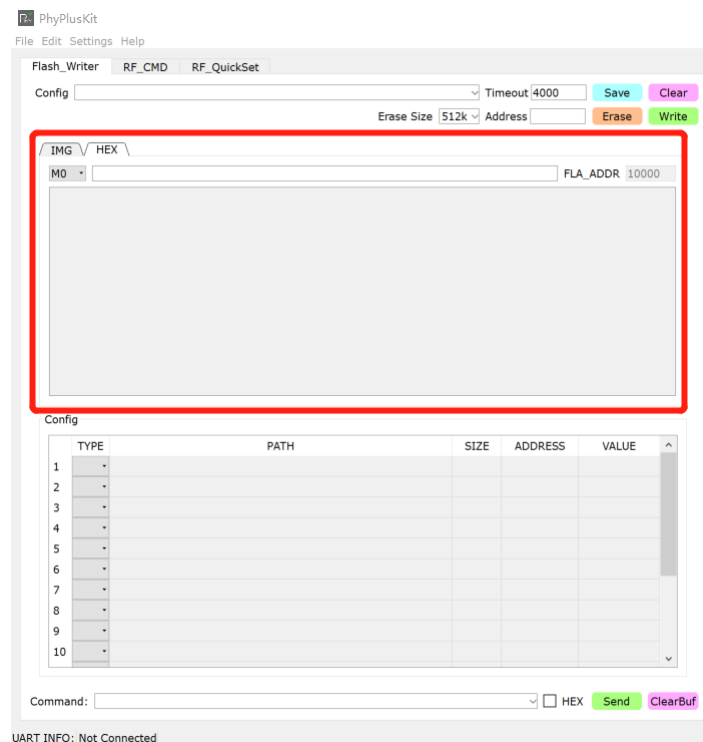
## 2.2 Flash 烧写



### 1. 表格（Table Part）

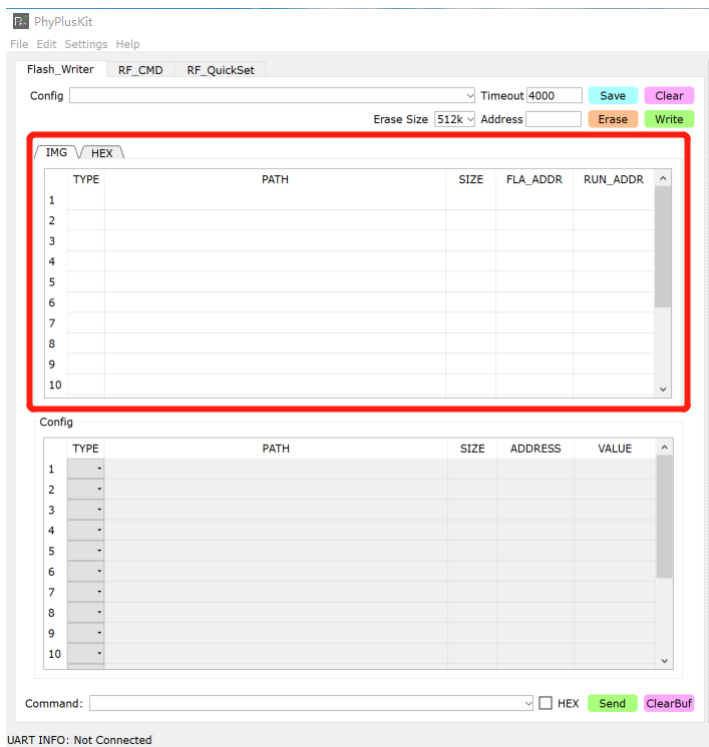
#### (1) image / HEX table

HEX: (建议使用 HEX 文件烧写)



- 在输入框中双击选择 Hex 文件，会自动分析 HEX 中的数据，并根据用户设置的起始 Start Flash Address，自动生成要烧写的 Flash Address；
- 起始 Start Flash Address，可在 Settings -> Configuration -> Flash Writer 设置。
- Hex 文件中必须存在主程序的运行地址 Run Address，默认为 0x1FFF4000，用户可在 Settings -> Configuration -> Flash Writer 中进行设置；

Image:



- 在表格区域（image）可以双击当前行 PATH 列的单元格，添加所要烧写的 bin 文件；
- 在对应行中填写所要烧写的 FLA\_ADDR、RUN\_ADDR；

(2) config table

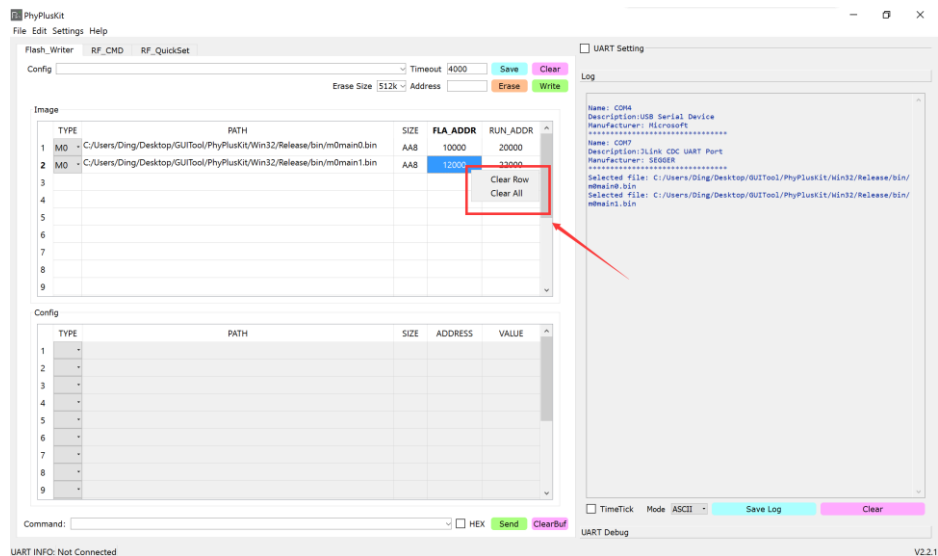
- 在表格区域（config）点击当前行 TYPE 列对应的下拉框，可以选择 AT 或 MT 模式，对应功能如下所示

模式 (TYPE)	功能	可编辑列
AT	从文件的读取要写 FLASH 寄存器的值	PATH, SIZE, ADDRESS
MT	从单元格 VALUE 的读取要写寄存器的值	ADDRESS, VALUE

- 若选择的为 AT 自动模式，可双击所对应的 PATH 单元格，选择存放寄存器值的文件。在 ADDRESS 单元格中填写寄存器起始地址。  
若选择的为 MT 手动模式，在 ADDRESS 单元格中填写，要写的寄存器起始地址，并在 VALUE 单元格中填写，要往对应寄存器的写的值。

寄存器写值的起始地址为 ADDRESS 所填写的地址，如文件有多行寄存器的值，则对应起始地址逐次+4，每次写 4bytes 的值。

注意： 表格中右键，可清除行内容，或清除整个表格内容



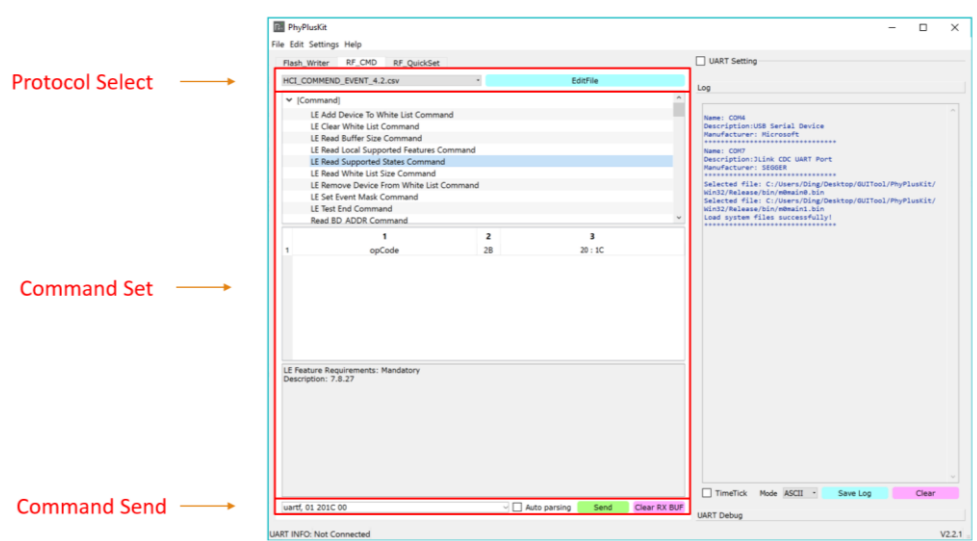
## 2. 控制（Control Part）

- 在烧写 bin 文件前，需要先进行擦除操作；可点击 Erase 按钮就行擦除；擦除成功后，会在 log 中输出“erase #ok”；
- 擦除成功后，即可进行烧写操作，点击 Write 按钮可将当前所选 bin 文件烧写到相应的地址；默认交互等待时间为 4000ms，用户可根据使用情况在 Timeout 中修改等待时间。
- 当前的 bin 文件配置信息和 Timeout 设置可以进行保存；在 config 中输入要保存的文件名，并点击 Save 按钮即可保存当前设置；
- 可以在 config 下拉列表中，可以看到之前保存的配置信息；选中即可加载。
- 点击 Clear 可清空，当前表格内容；

## 3. 命令（Command Part）

- 在 Command 中可以发送任意命令，点击 Send 发送命令；
- ClearBuf 按钮用于清空交互缓存，当串口接收出现错误时，可以清空缓存再次进行命令发送或烧写操作；
- Command 下拉列表中，会保存上次所发送的命令；

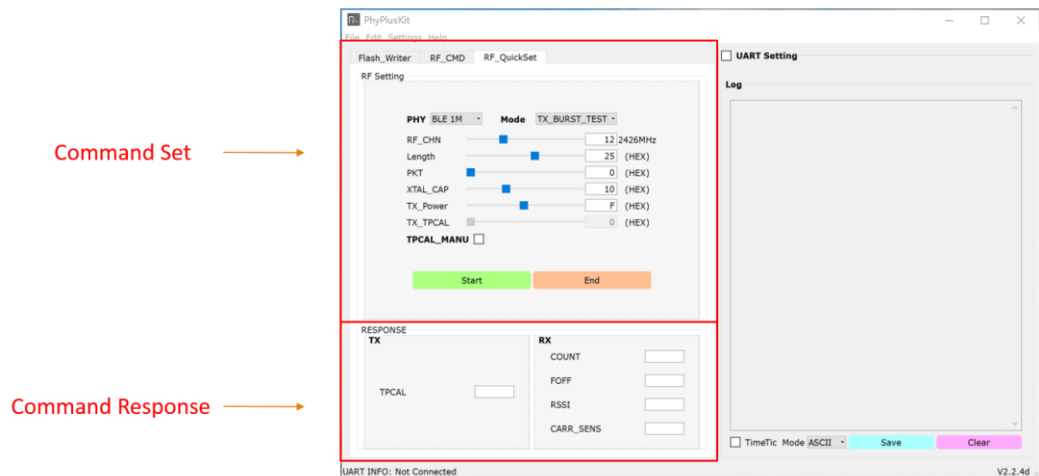
## 2.3 RF Command 命令发送



1. 协议选择（Protocol Select）
  - 在下拉列表中，可以选择不同版本的协议进行加载；
2. 命令设置（Command Set）
  - 在上方的树型列表中选择要发送的命令种类（Command, Event, etc.）；以及具体的命令；
  - 在中间的表格中，可对命令字段进行配置；
  - 下方显示命令的注释内容；
3. 命令发送（Command Send）
  - 在命令设置中配置好参数后，即可在 Command Send 看到组合好的命令内容，点击 Send 即可发送命令；
  - 在 Command 输入框中，可以输入自定义的命令内容；命令格式如下：

命令	作用
uarta, "abcd" (abcd 可替换为任意 ASCII 字符)	发送 ASCII 命令；
uarth, 01 02 (01 02 可替换任意 HEX 值)	发送 HEX 命令；
uartf, 01 0C6C	发送自动组合的命令；

## 2.4 RF\_QuickSet 快捷命令



### 1. 命令设置（Command Set）

- 在 RF Setting 中，可以选择相应的 PHY 硬件设备模式（BLE 1M, BLE 2M, BLE 500K, BLE 125K, ZigBee），以及相应的命令模式：TX（BURST\_TEST, Single Tone, Modulation），RX（BURST\_TEST, AUTO），设置命令参数 Frequency, Length, PKT, TX\_Power, TX\_TPCAL（校准值）；
- 点击 Start 按钮，可按序列发送相应命令；
- 点击 End 按钮，即可得到相应的相应结果 TX（TPCAL），RX（COUNT, FOFF, RSSI, CARR\_SENS）；

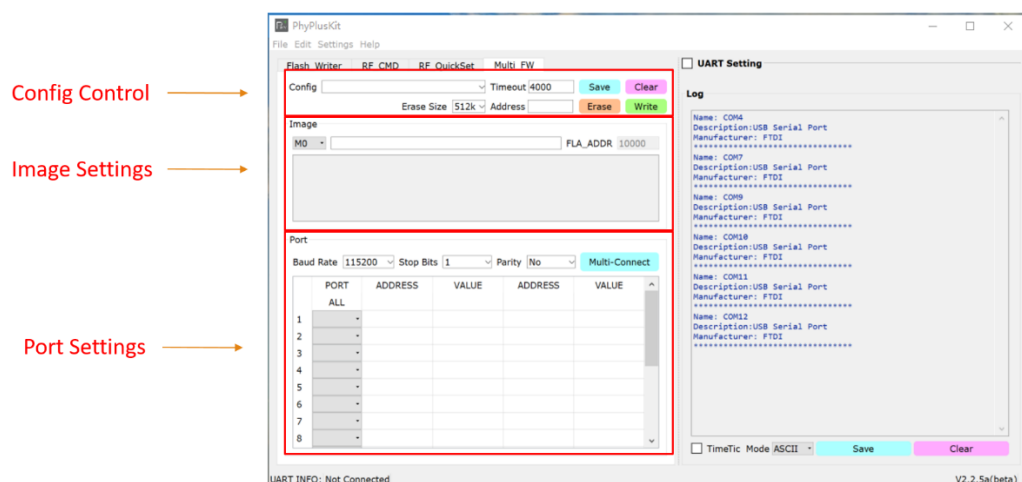
#### Note:

- BLE 模式下，Frequency 为 2402 ~ 2482MHz，步进 2MHz，分 0 ~ 40 个频道；ZigBee 模式下，Frequency 为 2405 ~ 2485MHz，步进 5MHz，分 0 ~ 16 个频道；
- 默认 TPCAL 为自动获取，若想手动修改 TPCAL 值，可以勾选 TPCAL\_MANU 选择框修改 TX\_TPCAL 的值；

### 2. 相应结果（Command Response）

- 在 RESPONSE 中，可以显示命令获取到的结果；

## 2.5 Multi\_FlashWriter 多串口烧写



### 1. Config Control

- 此处可以选择并载入保存过的相关配置文件，文件内容包括（和芯片交互的 timeout 值、要烧录的 HEX 文件路径、UART PORT 参数、要写入的 Flash 值）。
- 此处可以配置要擦出的 Flash 大小，以及对对应擦除的 Flash 地址（默认为 512K，擦除全部 Flash 内容）。
- Save 按钮保存配置，clear 清除表格内容，Erase 擦除 Flash，write 烧写 Flash

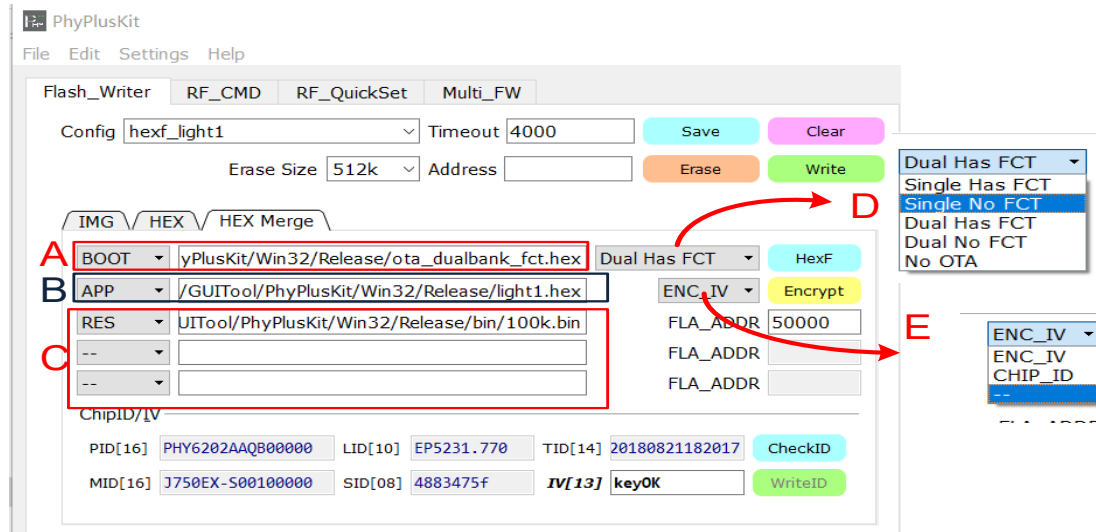
### 2. Image Settings

- 在输入框中双击选择 Hex 文件，会自动分析 HEX 中的数据，并根据用户设置的起始 Start Flash Address，自动生成要烧写的 Flash Address；
- 起始 Start Flash Address，可在 Settings -> Configuration -> Flash Writer 设置。
- Hex 文件中必须存在主程序的运行地址 Run Address，默认为 0x1FFF4000，用户可在 Settings -> Configuration -> Flash Writer 中进行设置；

### 3. Port Settings

- 此处对 PORT 连接进行配置，选择合适的波特率、停止位、校验位，并在表格内的下拉框中选择待连接 PORT 端口；
- 表格中 ALL 对应一行的 Flash 地址烧写，会对下面每一行所选的 PORT 生效，而每个 PORT 对应行的 Flash 地址烧写，仅对当前行（PORT）生效；
- 设置好 PORT 连接信息后，点击 Multi-Connect 进行连接操作

## 2.6 Flash\_Writer HEX Merge



### 1. HEX Merge

- A: 用于输入 ota\_boot.hex 文件，需要通过 D 选择框选择不同的 OTA\_BOOT 模式。目前一共支持 5 种模式。不同的模式 OTA\_BOOT 对 flash 地址的映射有所区别。参考下表 flash mapping
- B: 用于输入 APP 程序的 hex 文件，通过 E 选择框可以选择不同的加密方法。
  - E:[ENC\_IV] 模式是使用 IV 输入的 Identify Vector 对 app 文件进行加密保护。
  - E:[CHIP\_ID] 模式是使用唯一的芯片的 chip id 对 app 文件进行加密。  
IV=function (chipid.TID,chipid.SID)
  - E:[-]模式，不采用加密模式。
- C: 用于输入 resource 文件，目前支持 binary 格式和 hex 格式。可以在 FLA\_ADDR 输入资源文件的 flash 存储地址。
- [HexF] 按键用于一键产生\*.hexf 文件，该文件是 A, B, C 多个 hex 文件的合并产生，可以用于直接烧录。该文件输出路径为 app 文件目录。
- [Encrypt] 按键用于产生 app 文件的加密文件\*.hexe, 该文件输出路径为 app 文件目录。

### 2. ChipID/IV

- CheckID button 用于检测当前芯片的出厂 ID，需要连接 UART。如果检测结果未[EMPTY], 则会激活 WriteID button
- 对于没有烧录出厂 ID 的芯片，可以通过 WriteID button 进行烧录。需要填写相应的 PID, LID, MID, TID, SID
- IV: 用于输入文件需要使用的 Identify Vector, 13Byte。如果图中的加密方式选择为 CHIP\_ID,则这部分会自动从 CHIP ID 中产生 IV。

**Table 1 flash mapping**

	512K版本 (Dual bank) (Has FCT)		512K版本 (Single bank) (Has FCT)		512K版本 (Dual bank) (No FCT)		512K版本 (Single bank) (No FCT)		512K NO OTA	
Reserved By PhyPlus	00000~01fff	8k	00000~01fff	8k	00000~01fff	8k	00000~01fff	8k	00000~01fff	8k
1st Boot info	02000~03fff	8k	02000~03fff	8k	02000~03fff	8k	02000~03fff	8k	02000~03fff	8k
FCDS	04000~04fff	4k	04000~04fff	4k	04000~04fff	4k	04000~04fff	4k	04000~04fff	4k
UCDS	05000~08fff	16k	05000~08fff	16k	05000~08fff	16k	05000~08fff	16k	05000~09fff	20k
<a href="#">2nd Boot info</a>	09000~09fff	4k	09000~09fff	4k	09000~09fff	4k	09000~09fff	4k		
OTA bootloader	0a000~11fff	32k	0a000~11fff	32k	0a000~11fff	32k	0a000~11fff	32k		
FCT App	12000~2ffff	120k	12000~2ffff	120k		0k		0k		
App Bank0	30000~4ffff	128k	30000~4ffff	128k	12000~31fff	128k	12000~31fff	128k	0A000~29fff	128k
App Bank1	50000~6ffff	128k		0k	32000~51fff	128k		0k		
NVM	70000~7ffff	64k	50000~7ffff	192k	52000~7ffff	184k	32000~7ffff	312k	2A000~7ffff	344k

## 2.7 命令行模式

在命令行模式下，可以通过后接指定的参数来运行本程序以进行烧入指定的 hex（或 hexf）文件，或者将几个指定的文件合并成 hexf 文件（参照 2.6 节相关内容）。另外也可以选择既执行合并又执行烧写（实际执行顺序为先合并后烧写）。如果在命令行下不加参数打开本程序则会直接进入 GUI 模式。

本程序所支持的命令行参数如下（注意：参数除了 -a 要跟在 -r 后面外（具体参见 -a 参数的说明），其他参数理论上没有顺序先后的要求，另外每一个参数和它的缩写形式完全等价，可任意进行相互替换）：

参数	缩写	值	说明
--combine	-c	无	用来指示程序进行 hex 文件合并生成 hexf 文件，该参数不需要跟随一个值。 <b>生成的 hexf 文件默认根据 app 的文件名来确定</b> ，即如果 app 文件为 app.hex，则合并后的 hexf 文件和 app 在同一个目录并且文件名为 app.hexf 该选项和 -w 选项必须至少存在一个（可以同时存在）
--boot	-b	OTA Bootloader 文件 hex 格式	只能包含一个 OTA Bootloader 程序，如果为 No OTA 模式则该参数不是必要参数，其他情况下合并生成 hexf 文件时该参数为必要参数 参数为字符串，可以包含路径，路径有空格需要把整个字符串通过双引号包起来，和选项之间通过空格符号分开 例如：--boot ota.hex 以及 -b "c:\data app\ota.hex"
--app	-p	App 文件 hex 格式	只能包含一个 APP 固件程序，该参数是必要参数 参数为字符串，可以包含路径，路径有空格需要把整个字符串通过双引号包起来，和选项之间通过空格符号分开 例如：-p app.hex
--res	-r	Resource 文件 bin 或 hex 格式	可以包含不止一个 resource 文件（但是至多 3 个），该参数不是必要参数 参数为字符串，可以包含路径，路径有空格需要把整个字符串通过双引号包起来，和选项之间通过空格符号分开 如果为 bin 格式文件则需要在其后用 -a 选项指定它的起始地址（16 进制），如果为 hex 格式文件则不需要（如果 bin 文件后没有接起始地址则会报错并退出） 例如：-r res.bin -a 70000（即从 0x70000 地址开始写入）或者 -r res.hex

--addr	-a	bin 格式的 Resource 文件的写入起始地址	<p>起始地址为 <b>16 进制格式</b>，数值为 Flash 从 0 开始的偏移地址，数据大小为文件大小，本程序会对数据长度以及地址的合法性做检查，若为非法地址或者出现地址重叠则会提示错误，例子见上</p> <p>(hex 文件自带地址信息所以不需要该选项来对其起始地址进行设定)</p>
--mode	-m	合并模式	<p>合并模式需要为以下 5 个中的一个：SH (Single Has FCT), SN (Single No FCT), DH (Dual Has FCT), DN (Dual No FCT), NO (No OTA), 若为其他值则报错，该参数是必要参数</p> <p>例：-m DH</p>
--enc	-e	加密模式	<p>该参数不是必要参数，<b>若无该参数则默认为不加密</b></p> <p>如果有该参数，则它的值必须为以下两种情况：chip（表示用 chip id 加密）或 iv_xx...xx (xx...xx 为 13 位的 iv 值，若不足则在右端补 0，超过则截取左端 13 位作为 iv 值)，若不是这两种情况则报错</p> <p>例：-e chip 或者 -e iv_1234567890123</p>
--write	-w	写入的文件 hex 或 hexf 格式	<p>该选项和 -c 选项必须 <b>至少存在一个</b>（可以同时存在），若同时存在则先执行 -c 选项进行合并再执行该选项进行写入，即可以同时使用这两个选项来进行合并后写入</p> <p>参数为字符串，可以包含路径，路径有空格需要把整个字符串通过双引号包起来，和选项之间通过空格符号分开</p> <p>例如：-w target.hexf</p> <p><b>注意：在写入开始前程序默认会进行一次擦除操作！</b></p>
--uart	-u	更新波特率	<p>该参数不是必要参数，若无该参数，默认波特率为：115200；波特率值设置有：1500000、1000000、500000、250000、115200、76800、38400、9600 等，根据 uart 配置可进行修改（<b>v2.3.8c 目前只支持 PHY6212 在波特率 1500000、1000000 下的烧写</b>）</p> <p>例：-u 500000</p>
--Run	-R	Base run address (1FFF4000--PHY6202, 1FFF4800--PHY6212)	<p>该参数是必要参数，需要修改该值对不同的芯片类型的固件进行烧写；通过该命令进行烧写工具中 configuration—base run address 中的修改，对 hex 文件中的 bin 文件进行烧写的起始地址</p> <p>例：-R 1FFF4000（对应 6202 的芯片），-R 1FFF4800（对应 6212 的芯片）</p>

--Port	-P	获取的 Port 的 name	多个 uart 与主机相连，获取多个 uart 的 COM 口的名称，对指定的 COM 口的开发板进行固件烧写 例如：-P COM3
--config	-f	配置文件.csv	该参数不是必要参数，主要是用来烧写 MAC 地址，在 csv 文件中设置多行 12 位 mac 地址，通过 4000 和 4004 地址进行 mac 地址值的烧写（备注：有其他配置需求，也可自行添加，获取对应的写入地址 address 和 value） 例如：-f *.csv
--line	-l	配置文件的 line 值设定	该参数不是必要参数，配置文件中可输入多行信息，每次烧写，可以通过改变 line 的值来改变 mac 地址或者其他参数设置 例如：-l 3（这里是英文字符 l（小写 L），不是数字 1）
--help	-h 或 -?	无	显示帮助

## 2.8 MAC 地址

ChipID/IV

PID[16] 1234567890123456

LID[10] 1234567890

TID[14] 12345678901234

CheckID

MID[16] 1234567890123456

SID[08] 12345678

IV[13]

WriteID

MAC[6] 31-32-33-34-35-36

Hex[xx-xx-xx-xx-xx-xx]

WriteMAC

在 ChipID/IV 栏的下方有一个上图中红框标出的 MAC 地址栏，该栏可以查看设备当前的 MAC 地址，同时也可以写入新的 MAC 地址（仅限原来 MAC 地址为空时）。

如果要查看当前的 MAC 地址，需要在连接后点击上图中的 CheckID，如果 MAC 地址已经被设定好，则 MAC 地址栏会显示当前的 MAC 地址，如果 MAC 地址未被设定，则 MAC 地址栏会变为可编辑状态，同时右侧的 WriteMAC 按钮也会变为可用。

如果要为 MAC 地址为空的设备设定新的 MAC 地址，需要先点击 CheckID 进行检查，当提示 MAC 地址为空时，可以在已经变为可编辑的 MAC 地址栏里面填写新的 MAC 地址（格式为 xx-xx-xx-xx-xx-xx，并且要求为 16 进制，否则会写入失败）后点击 WriteMAC 以写入新的 MAC 地址。

## 3. 功能示例

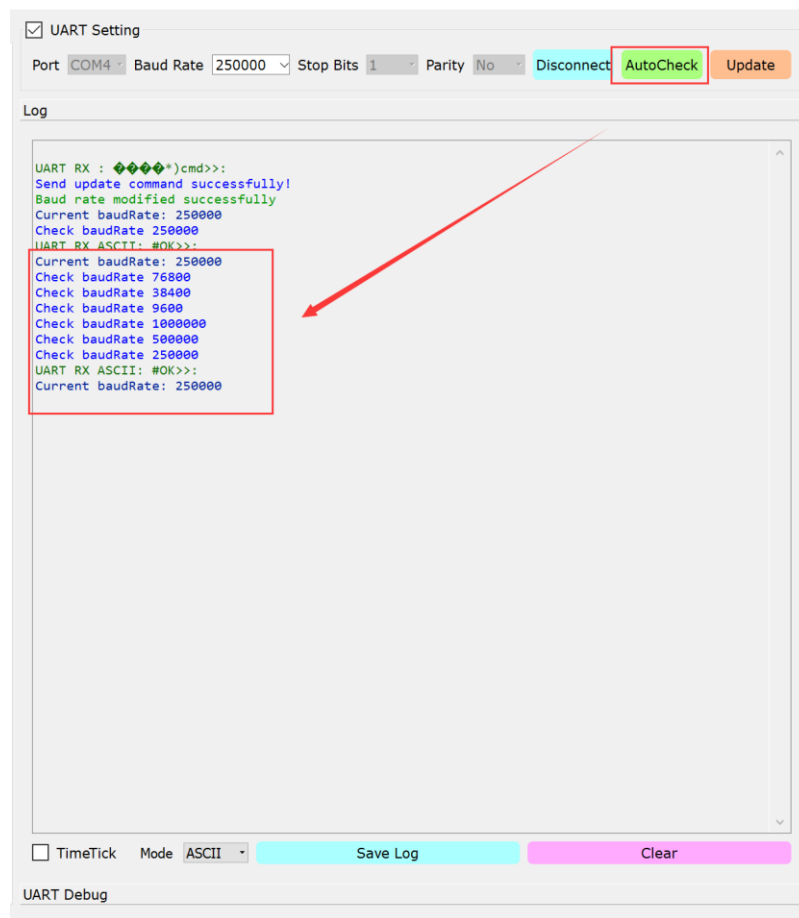
### 3.1. UART 速率获取和更新

#### 3.1.1. UART 速率更新



1. 选择下位机需要修改的波特率（如 250000）；
2. 点击 Update 按钮发送命令；
3. Log 提示更改成功；

### 3.1.2. UART 速率获取



1. 点击 AutoCheck 按钮，自动检测下位机波特率；
2. 待 UART 打印 OK 后，显示当前下位机波特率 25000；

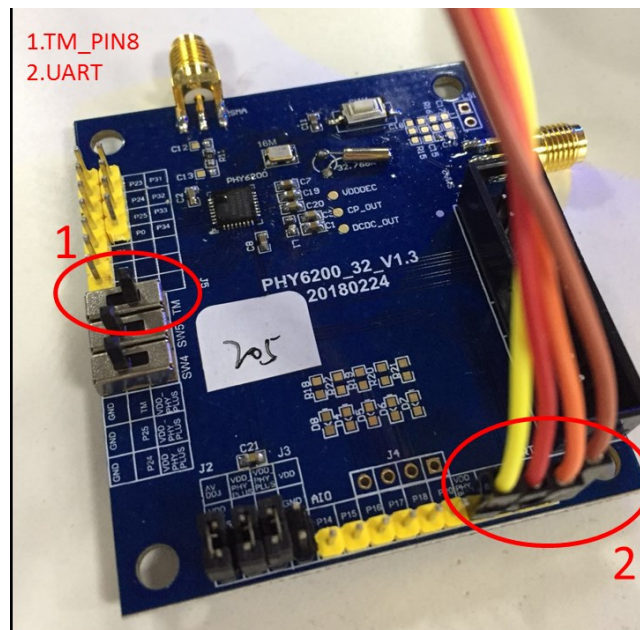
## 3.2. Flash 烧写

### 3.2.1 仅有 hex 的烧写 (推荐此模式)

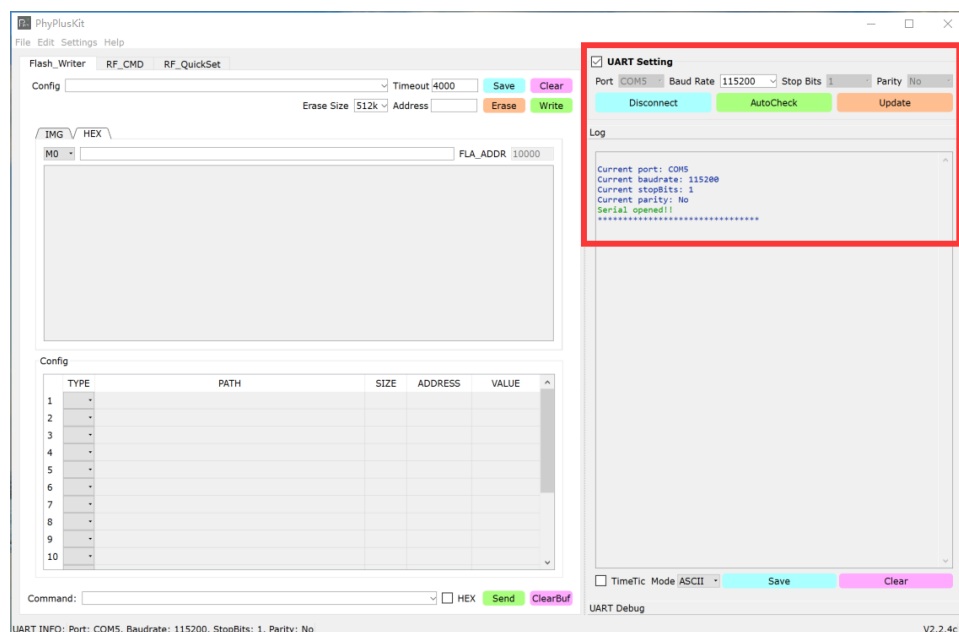
使用 uart 进行 flash 的烧写操作，pin8 拉高后上电，即为 uart 接收命令状态，uart 配置为波特率：115200，8bit, 1 bit stop, None parity, no flow control;

步骤：

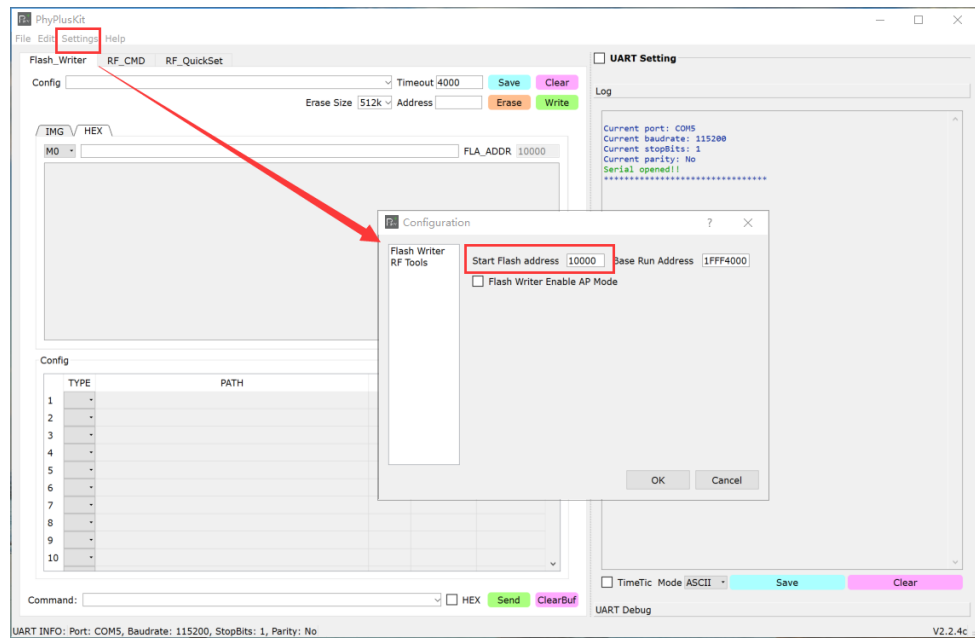
1. 准备软件和工具，连接硬件、TM (pin8)拉高，如下图所示



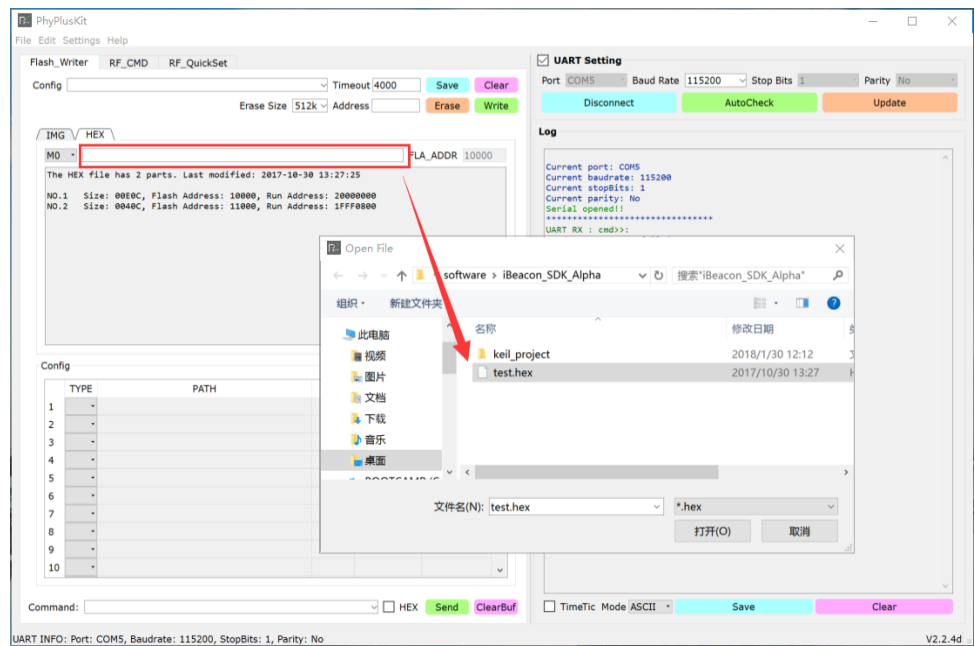
2. 打开 PhyPlusKit.exe，在 Uart Tab 中配置参数（115200，8bit, 1 bit stop, None parity, no flow control），Connect



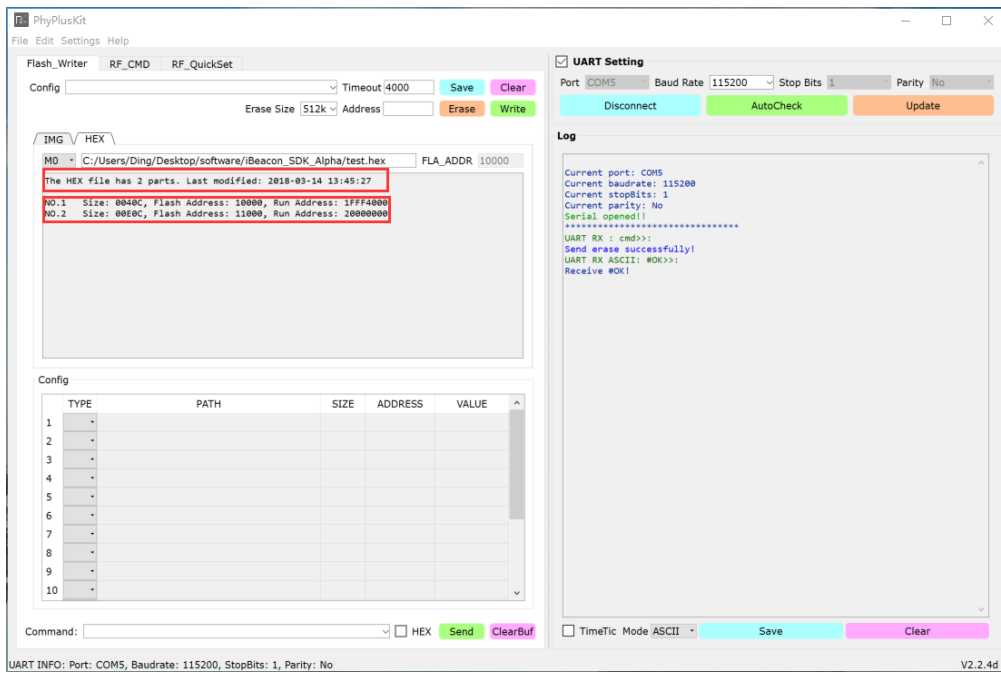
### 3. 设置起始 Flash 地址 Start Flash Address



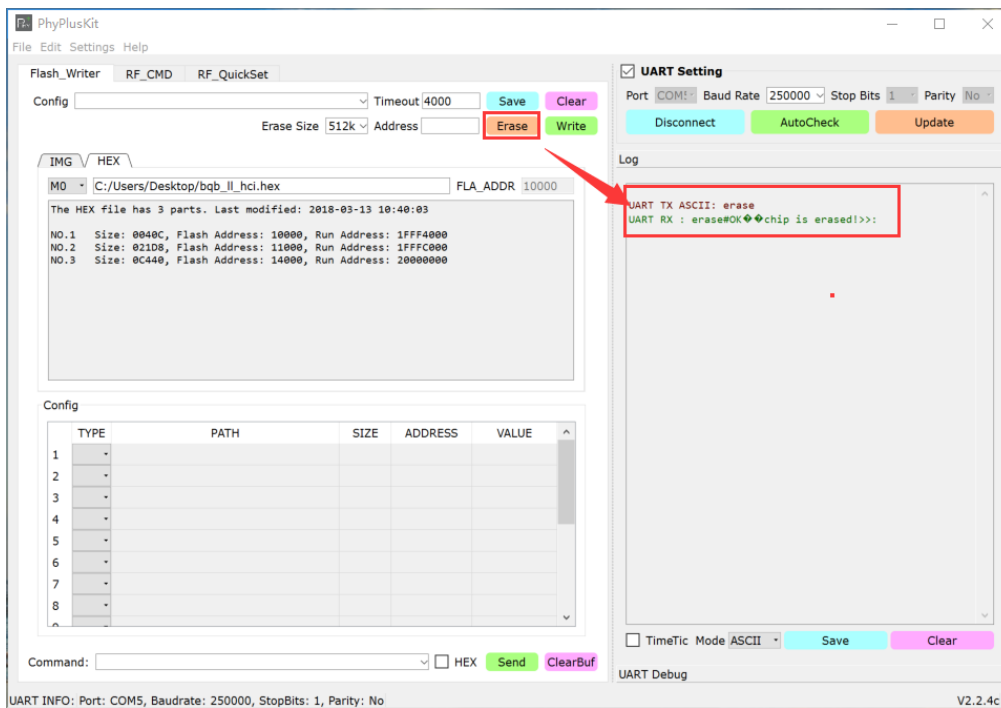
### 4. 双击输入框选择 Hex 文件



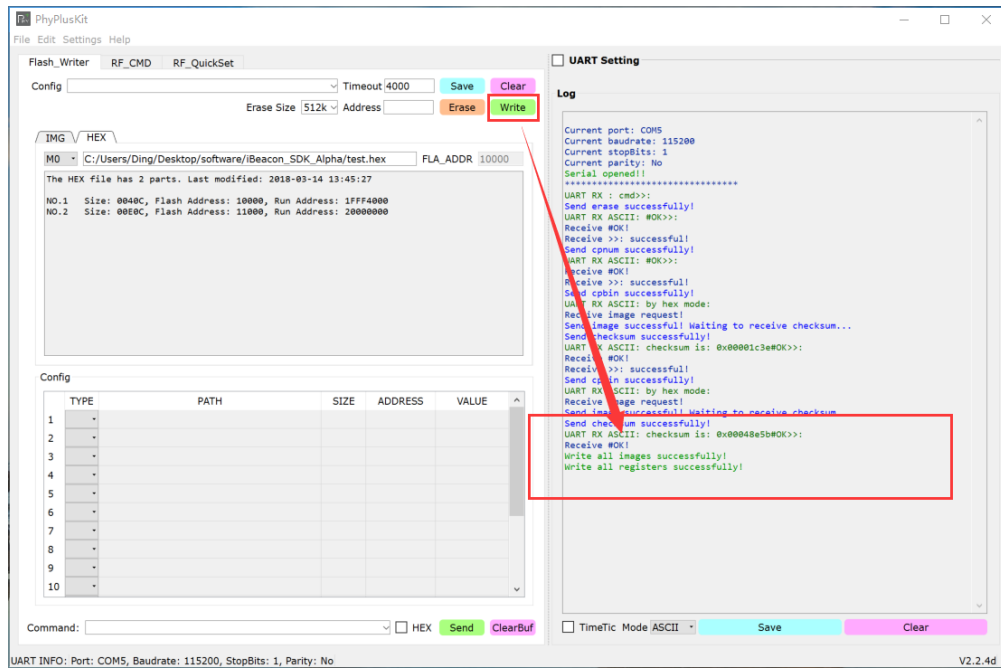
程序自动解析 HEX 文件中的数据文件，并显示上次修改时间



## 5. 烧写前先进行擦除，点击 erase 按钮



## 6. 点击 Write 按钮，自动烧写 HEX



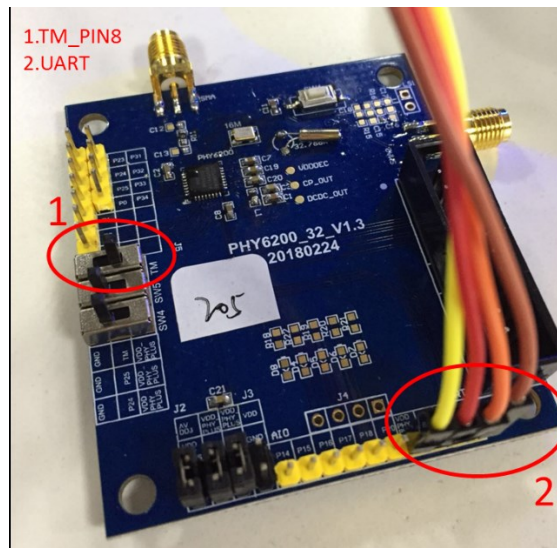
7. 烧写成功后，TM\_pin8 拉低，reset 或者重新上电，上电即为 boot 模式

### 3.2.2 仅有 Image 烧写

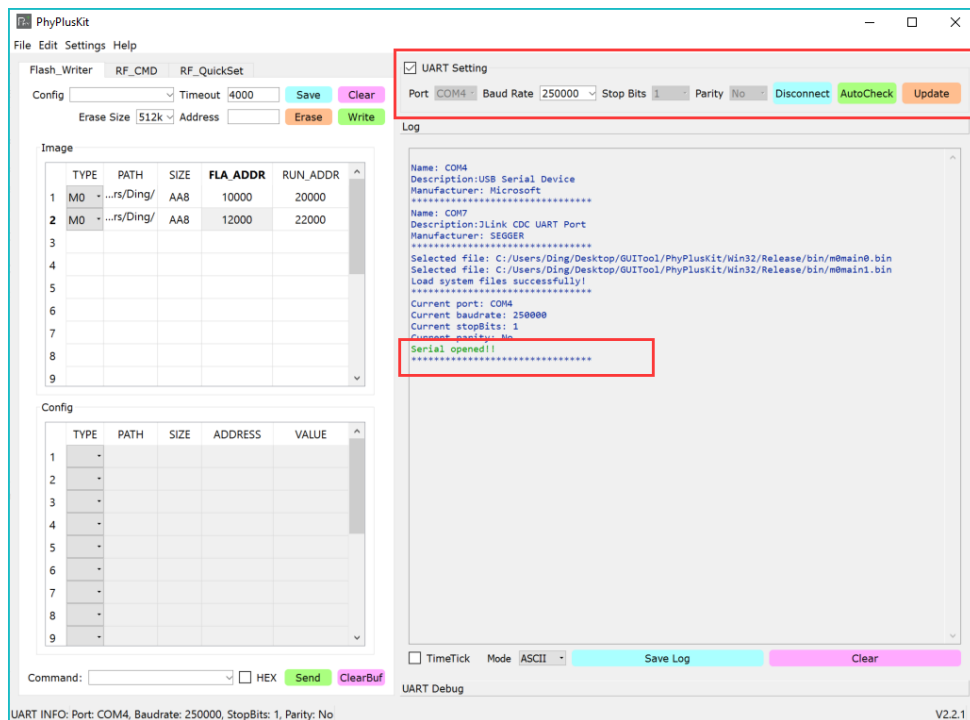
使用 uart 进行 flash 的烧写操作，pin8 拉高后上电，即为 uart 接收命令状态，uart 配置为波特率：115200，8bit，1 bit stop，None parity，no flow control；

步骤：

1. 准备软件和工具，连接硬件、pin8(TM)拉高，如下图所示



2. 打开 PhyPlusKit.exe，在 Uart Tab 中配置参数（115200，8bit，1 bit stop，None parity，no flow control），Connect

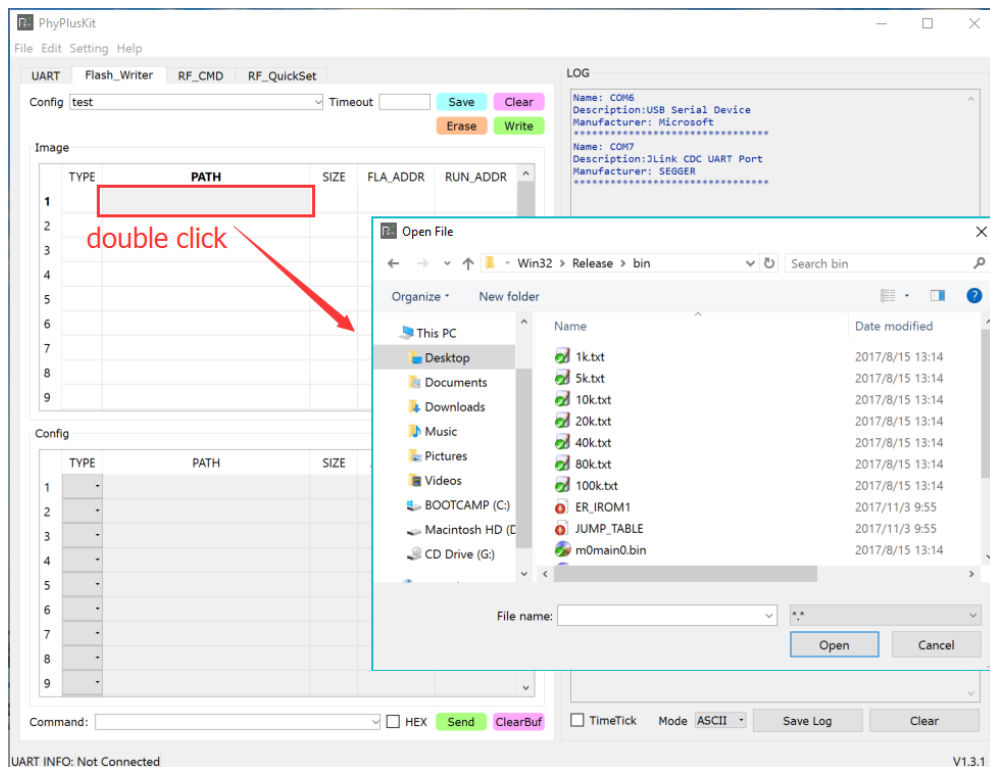


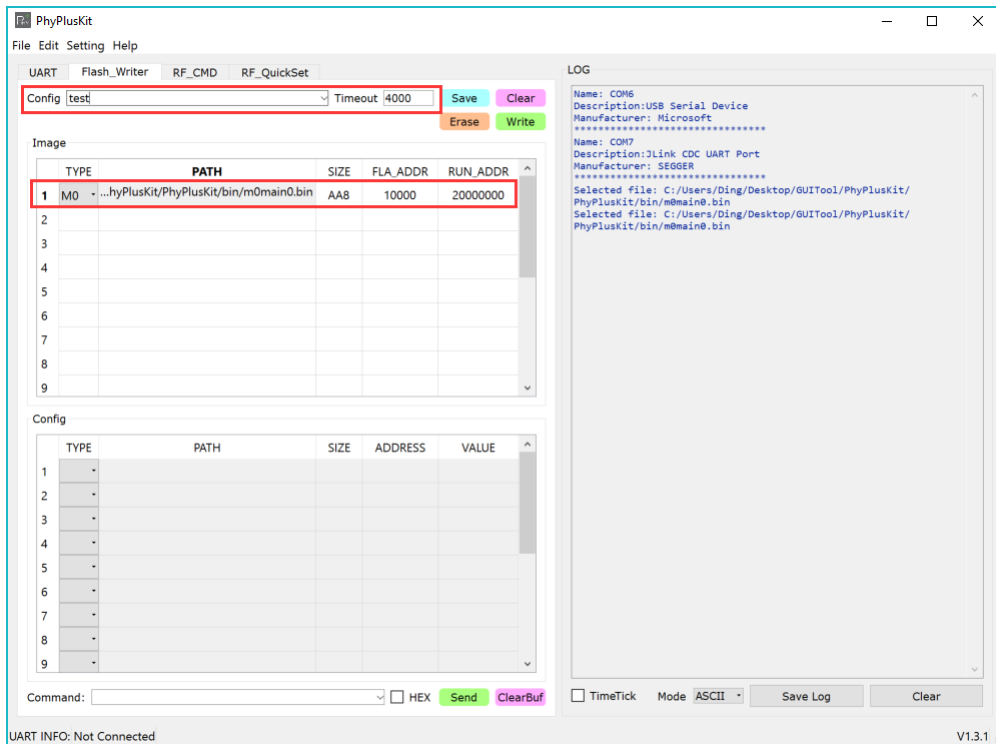
3. 连接成功后，选择 bin 文件（在 PATH 列 双击单元格），type(M0)，配置 fla\_addr 和 run\_addr，配置信息如下：

1) fla\_addr:flash 偏移地址，建议 0x10000

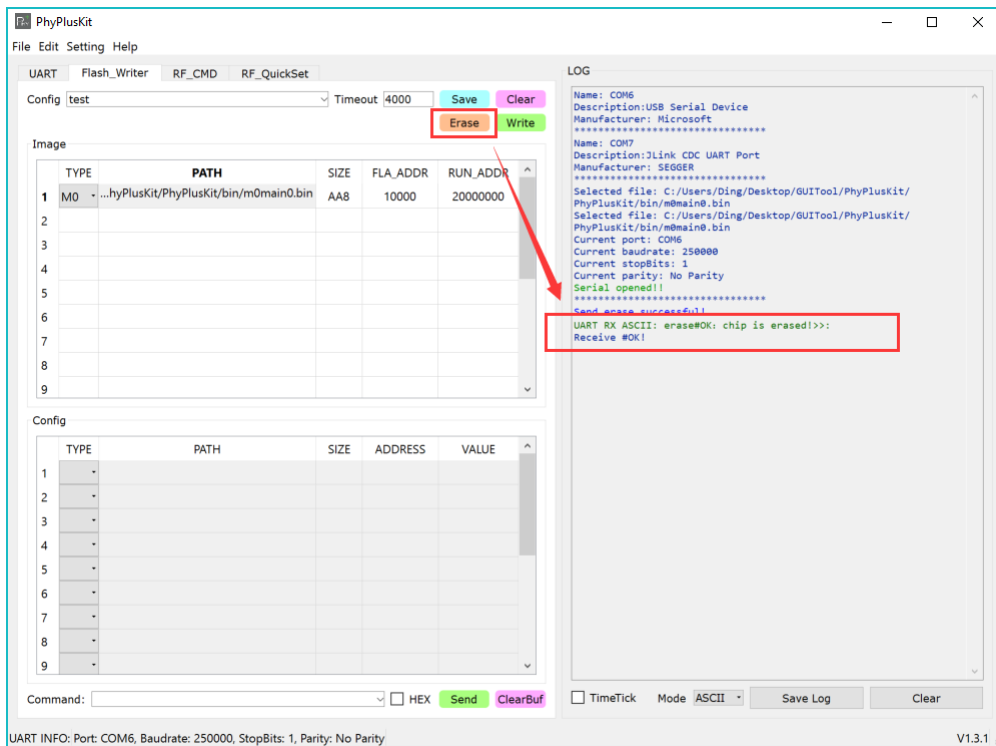
2) run\_addr:APP 运行地址，默认地址:0x1FFF4000

可以 save 此配置，用户自己定义 config name，连续使用。

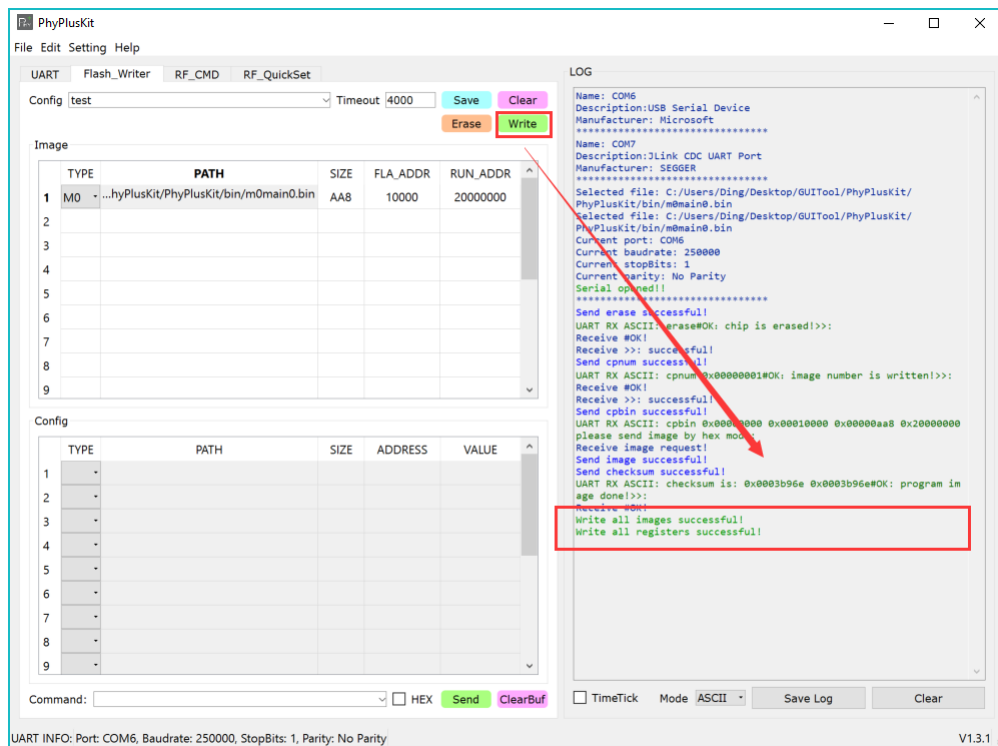




#### 4. 烧写之前先进行 erase



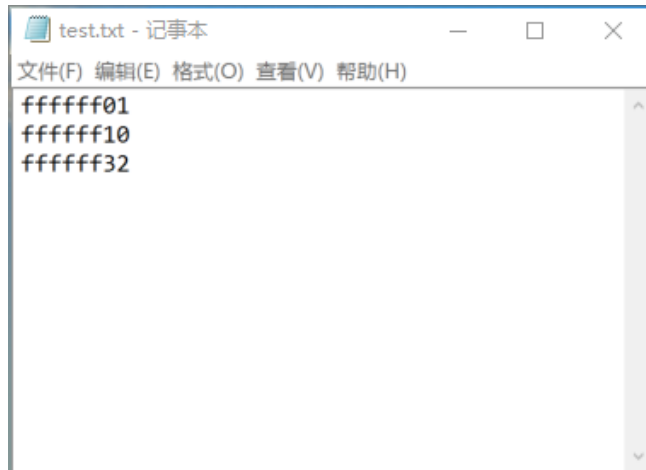
#### 5. 然后 write



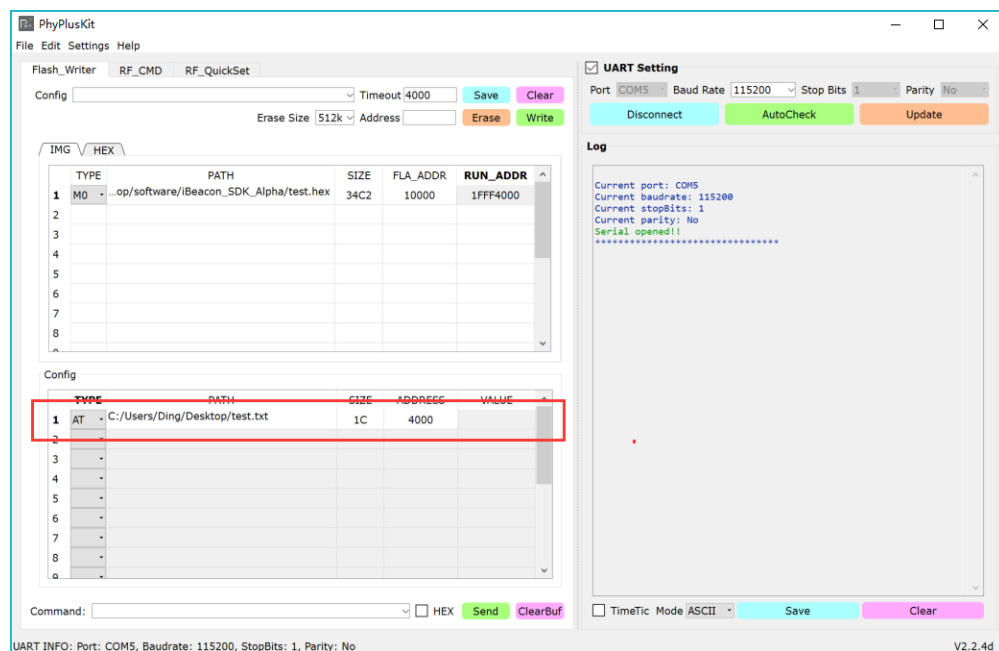
6. 烧写成功后，TM\_pin8 拉低，reset 或者重新上电，上电即为 boot 模式

### 3.2.3 有 image 和 config 的烧写

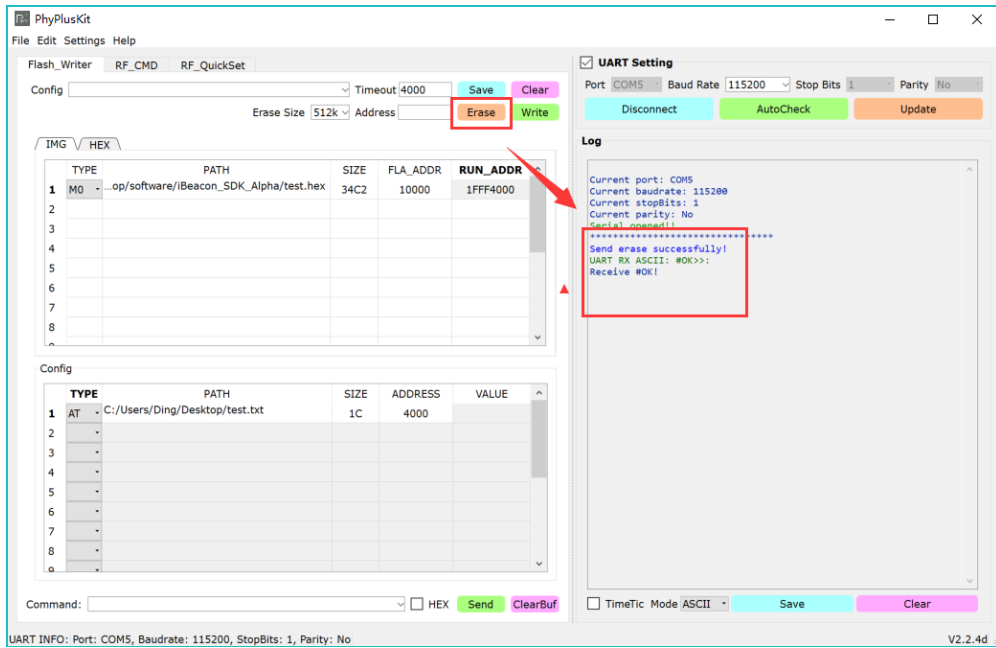
1. 重复上述步骤 1-5.
2. 选择 AT 模式读取存放寄存器值的文件，并填写起始地址  
起始地址如：4000  
文件内容如：



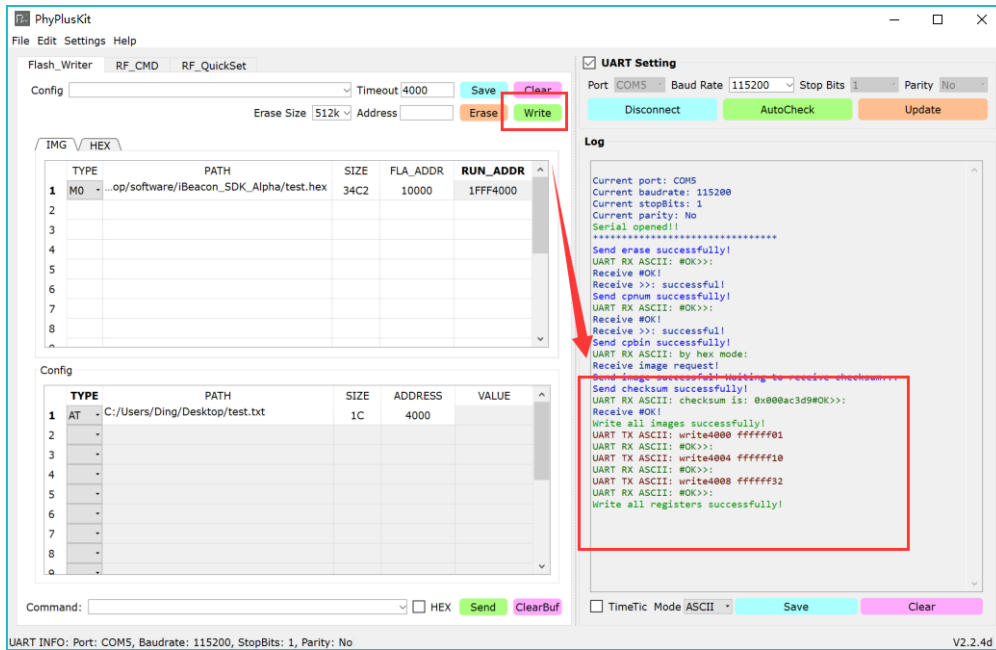
文件每行代表要写入的数值，起始地址由用户指定，每行对应写入的地址在起始地址基础上加 4 个单位。



3. 点击 erase 进行擦除

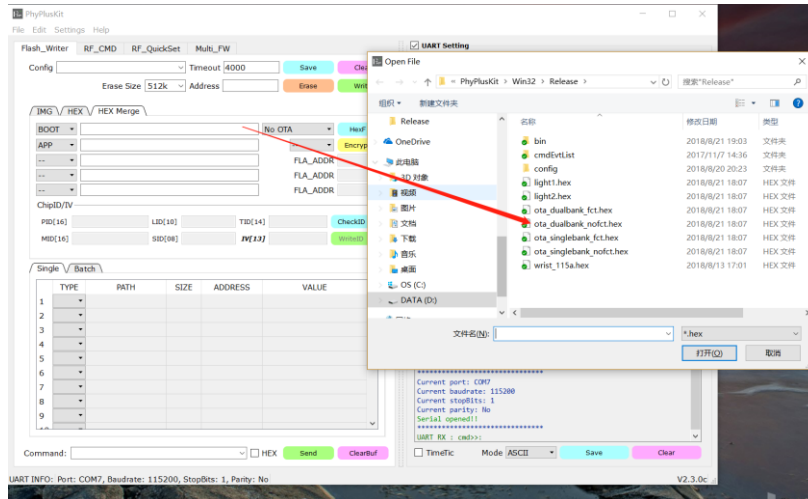


#### 4. 点击 write 烧写

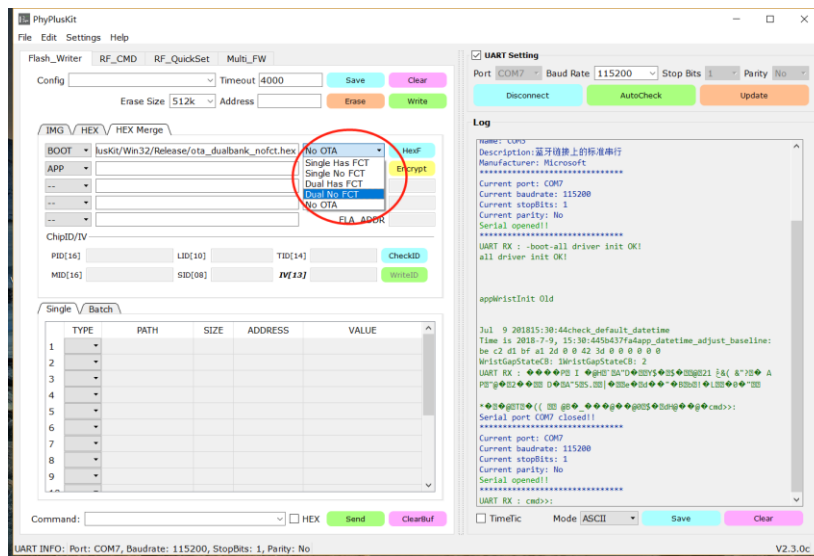


## 3.2.4 用 HexMerge 进行烧录

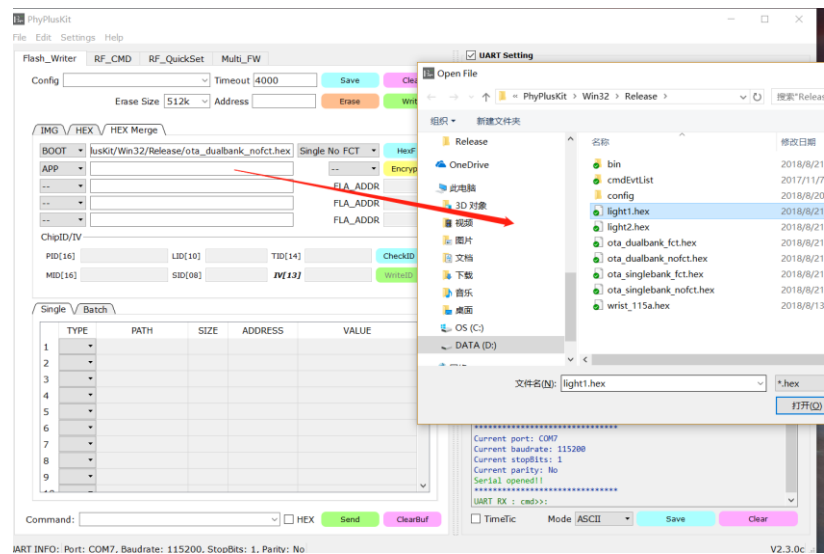
1. 双击 BOOT 文件输入框，选择 ota\*.hex 文件。



2. 选择合适的 OTA\_BOOT 模式

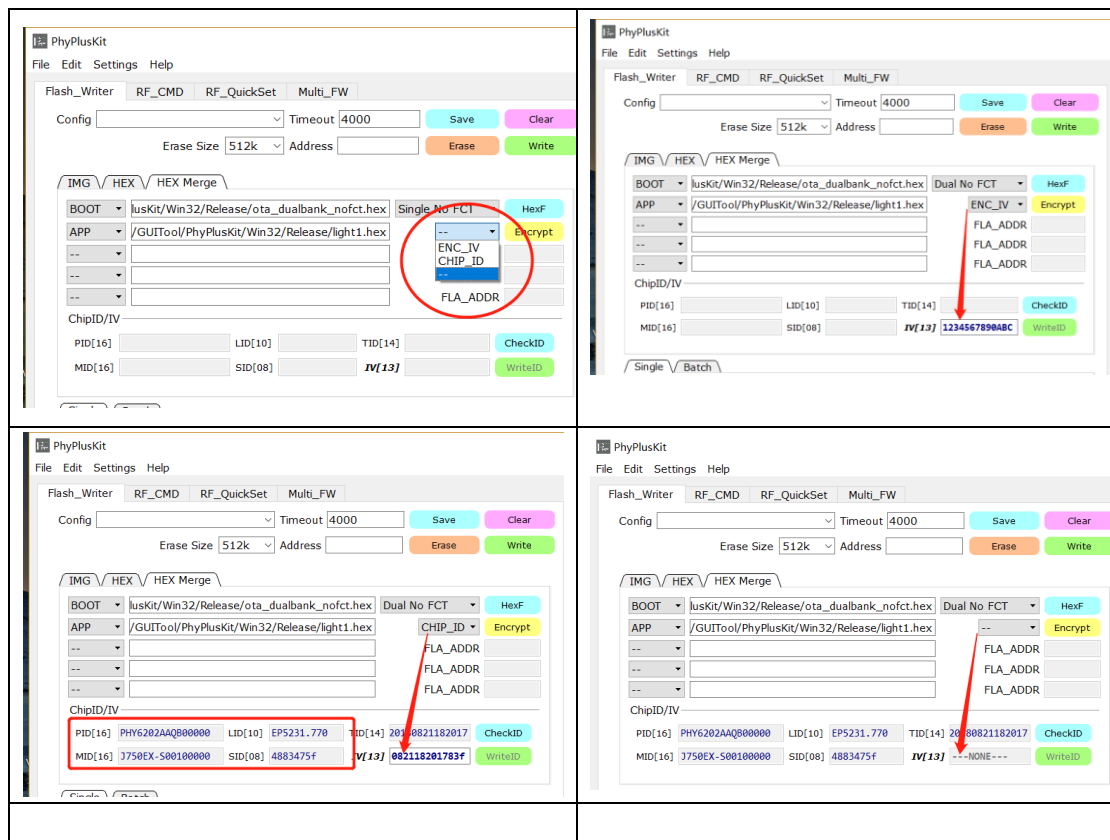


3. 双击 APP 文件输入框，选择 app\*.hex 文件。

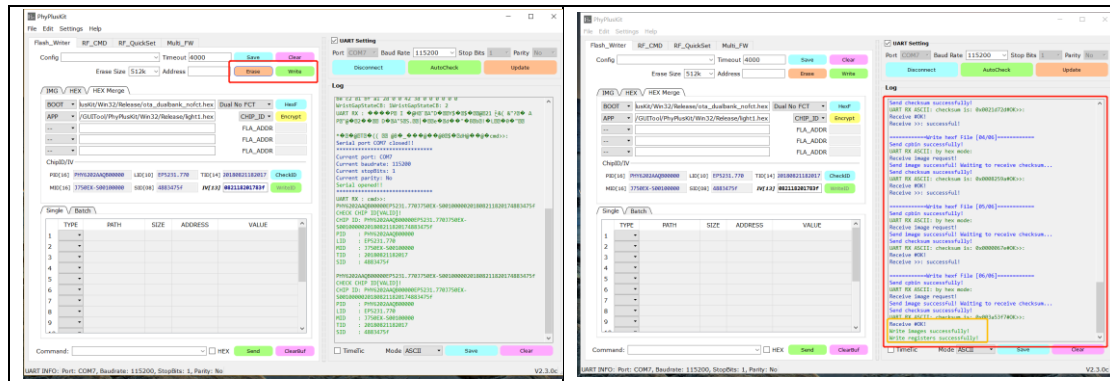


4. 选择 APP 文件的加密方式：

- a) ENC\_IV: 需要手动输入 IV
- b) CHIP\_ID: 自动查询当前芯片的 chipID, 自动计算 IV
- c) NO\_ENC: 不需要加密。



5.先点击 erase，再点击 write 进行烧录。



烧录完成，\*.hexf 以及\*.hexe 文件以及相应的产生在 app 的目录下。

cmd	2018/8/21 13:03	文件夹	
cmdEvtList	2017/11/7 14:36	文件夹	
config	2018/8/20 20:23	文件夹	
batchConfigList_compact.csv	2018/7/9 10:23	Microsoft Excel ...	2 KB
config.ini	2018/8/20 17:46	配置设置	1 KB
light1.hex	2018/8/21 18:07	HEX 文件	83 KB
light1.hexe	2018/8/21 20:26	HEXE 文件	83 KB
light1.hexf	2018/8/21 20:26	HEXF 文件	169 KB
light2.hex	2018/8/21 18:07	HEX 文件	209 KB
ota_dualbank_fct.hex	2018/8/21 18:07	HEX 文件	84 KB

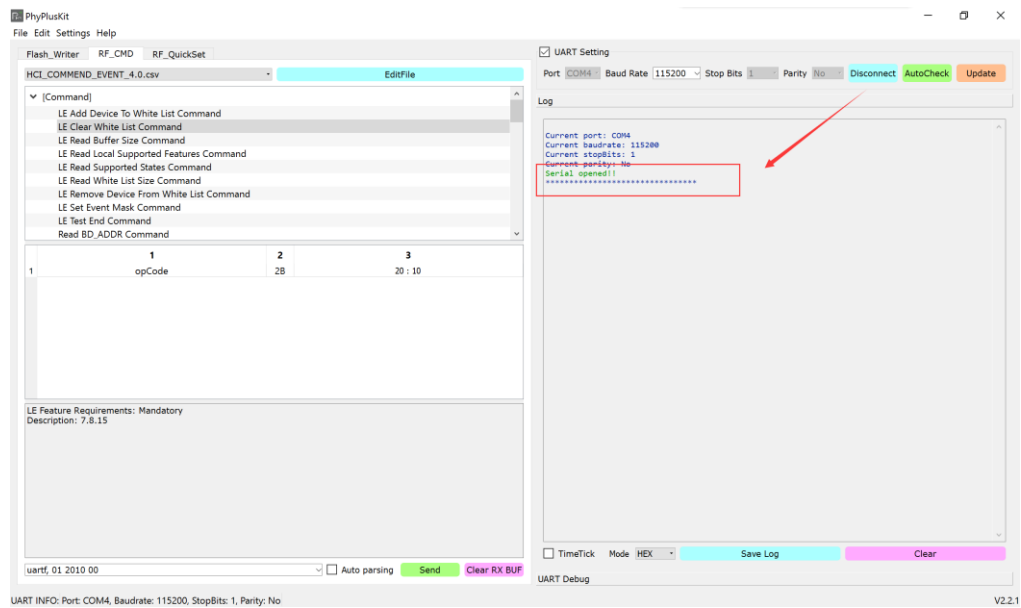
\*.hexf 文件为多个 hex 文件的合并输出，可以通过 PhyPlusKit 直接烧录。

\*.hexe 文件是 app\*.hex 文件的加密输出，也是之后 ota 升级的加密文件。

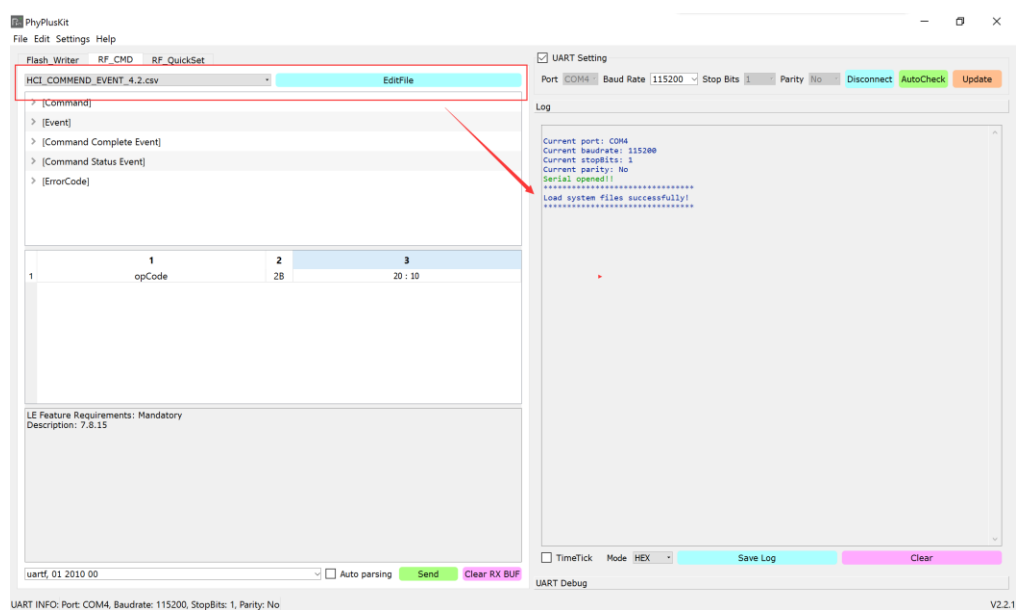
## 3.3 RF Command 使用

### 3.3.1 RF Command TX

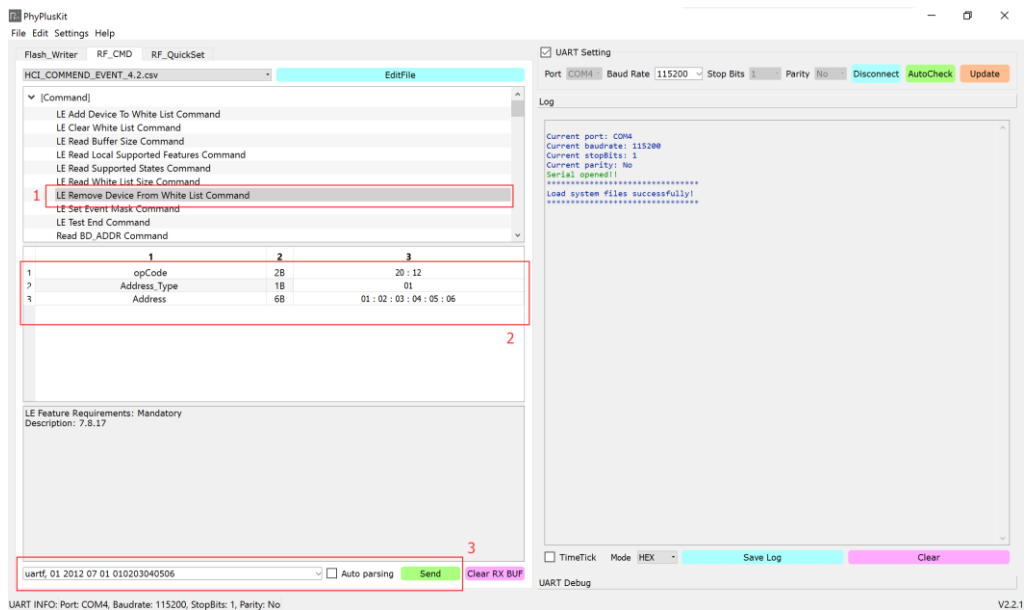
1. uart 配置为波特率: 115200, 8bit, 1 bit stop, None parity, no flow control;



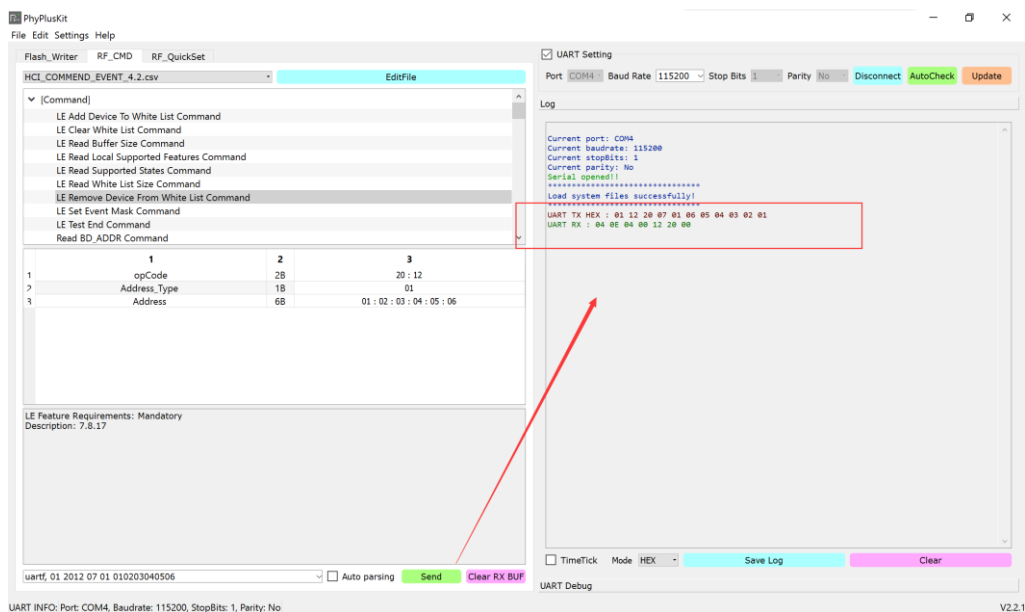
2. 选择要载入的协议文件



3. 选择要发送的命令，并在表格中配置命令内容

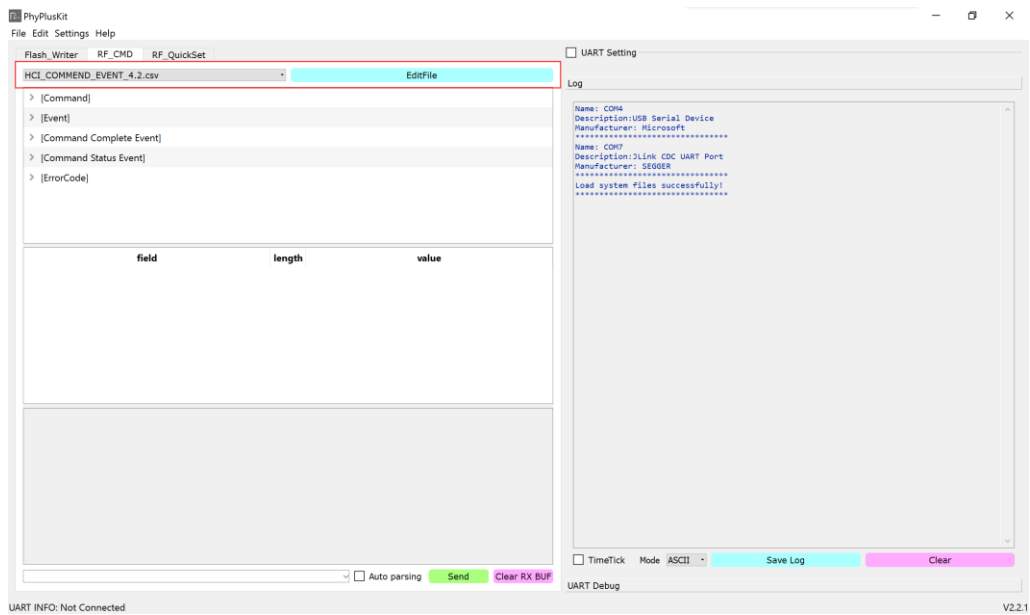


#### 4. 点击 send 发送命令

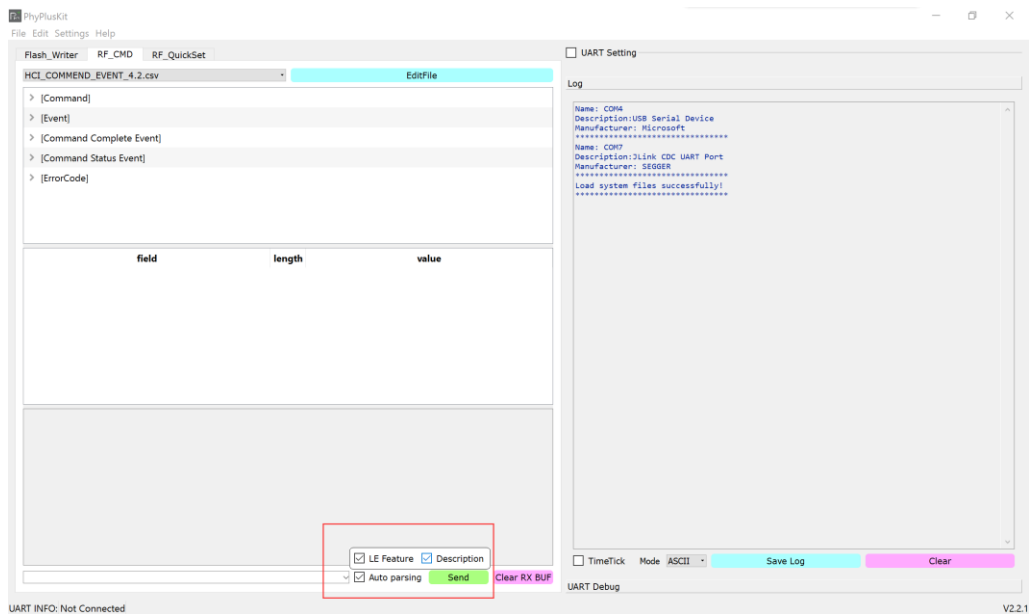


### 3.2.2 RF Command RX

#### 1. 先加载要解析的协议文件



2. 勾选 自动解析 (Automatic Parsing), 并根据需要选择悬浮窗中的设置



3. 在 Log 中会输出自动解析后的包内容

PhyPlusKit

File Edit Settings Help

Flash\_WriterRF\_CMDRF\_QuickSet

HCI\_COMMAND\_EVENT\_4.2.csvEditFile

[Command]

LE Add Device To White List Command

LE Clear White List Command

LE Read Buffer Size Command

LE Read Local Supported Features Command

LE Read Supported States Command

LE Read White List Size Command

LE Remove Device From White List Command

LE Set Event Mask Command

LE Test End Command

Read BD\_ADDR Command

	1	2	3
1	opCode	2B	20 : 02

LE Feature Requirements: Mandatory  
Description: 7.8.2

uartf\_01 2002 00

☒ Auto parsing

Send

Clear RX BUF

UART Setting

PortCOM4Baud Rate115200Stop Bits1ParityNoDisconnectAutoCheckUpdate

Log

UART TX HEX : 01 10 20 00

UART RX :

Data\_Type: Event

Data\_Name: Command Complete Event

LE\_Feature: Mandatory

Description: 7.7.14

Data\_Code: 06

Data\_Length: 04

Num\_HCI\_Command\_Packets: 00

Command\_Opcode: 10 20

Return\_Parameter:

Status: Success

UART TX HEX : 01 02 20 00

UART RX :

Data\_Type: Event

Data\_Name: Command Complete Event

LE\_Feature: Mandatory

Description: 7.7.14

Data\_Code: 06

Data\_Length: 07

Num\_HCI\_Command\_Packets: 00

Command\_Opcode: 02 20

Return\_Parameter:

Status: Success

HC\_LE\_ACL\_Data\_Packet\_Length: 10 00

HC\_Total\_Num\_LE\_Data\_Packets: 0C

☐ TimeTickModeASCII

Save Log

Clear

UART Debug

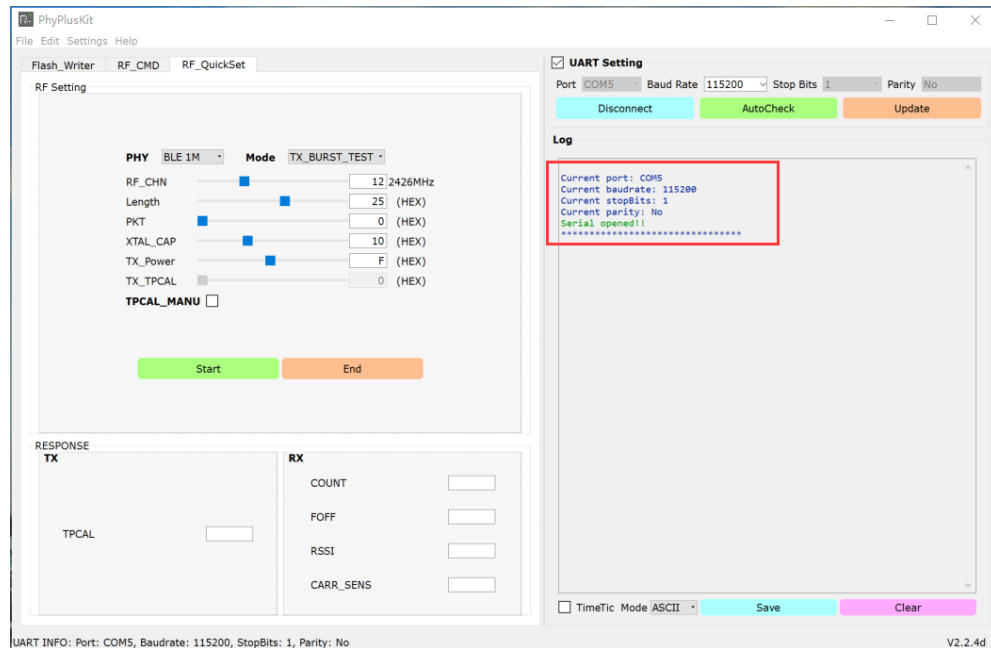
UART INFO: Port: COM4, Baudrate: 115200, StopBits: 1, Parity: No

Release

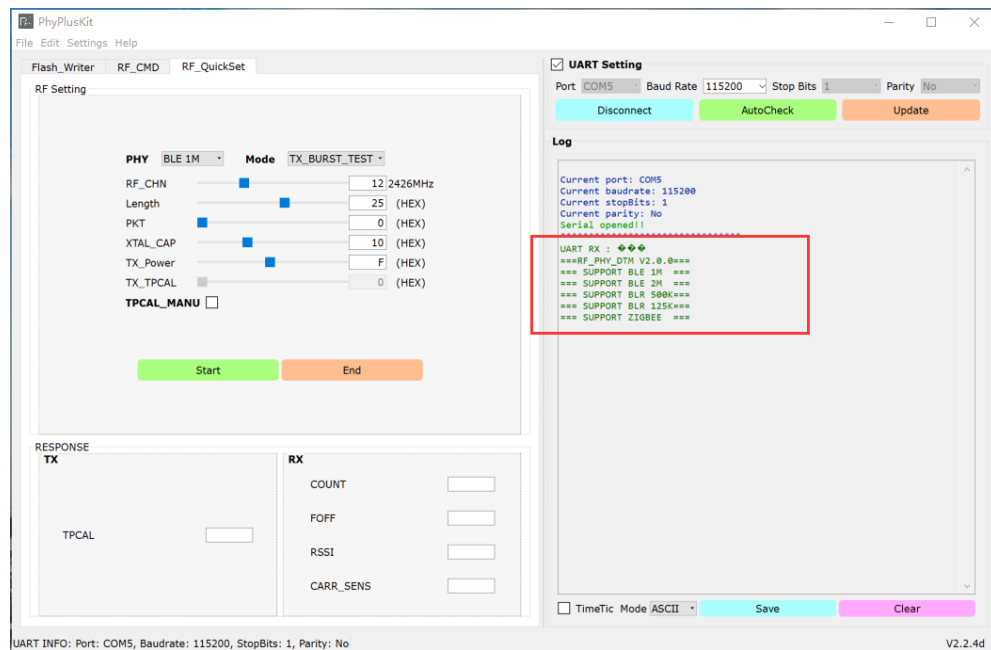
V2.2.1

## 3.4 RF QuickSet 使用

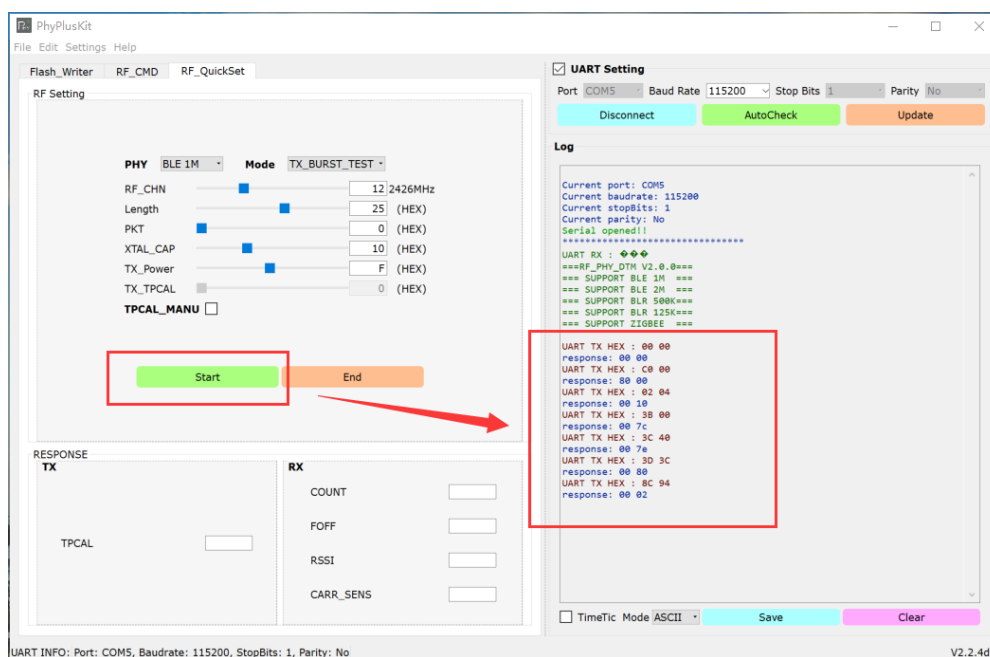
1. uart 配置为波特率: 115200, 8bit, 1 bit stop, None parity, no flow control;



2. 将 tab 切换到 QuickSet, reset 开发板, 看到开发板处于 DTM 模式



3. 配置好 TX 参数后，即可点击 **Start** 发送序列命令。



**PHY:** 用于设置 TX/RX 物理层类型，BLE1M, BLE2M,BLE500K,BLE125K,ZIGBEE

**MODE:** TX\_BURST\_TEST, 发射固定间隔的 BLE 数据包。  
TX\_SINGLE\_TONE, 发射单音信号，用于检测频偏和发射功率和相位噪声  
TX\_MODULATION, 发射连续的调制信号  
RX\_BURST\_TEST, 进入 RX 解调模式，统计接收到的数据包个数  
RX\_AUTO, 每 1000 个数据包间隔自动统计接收到正确数据包个数。

**RF\_CHN:** 用于设置 RF Frequency,  
对于 BLE,  $RF\_FREQ = RF\_CHN * 2 + 2400$   
对于 ZIGBEE,  $RF\_FREQ = RF\_CHN * 5 + 2400$

**Length:** TX packet Length 单位是 BYTE

**PKT:** TX packet 的类型, 0-> prbs9, 1-> 11110000, 2-> 10101010, 3-> prbs15

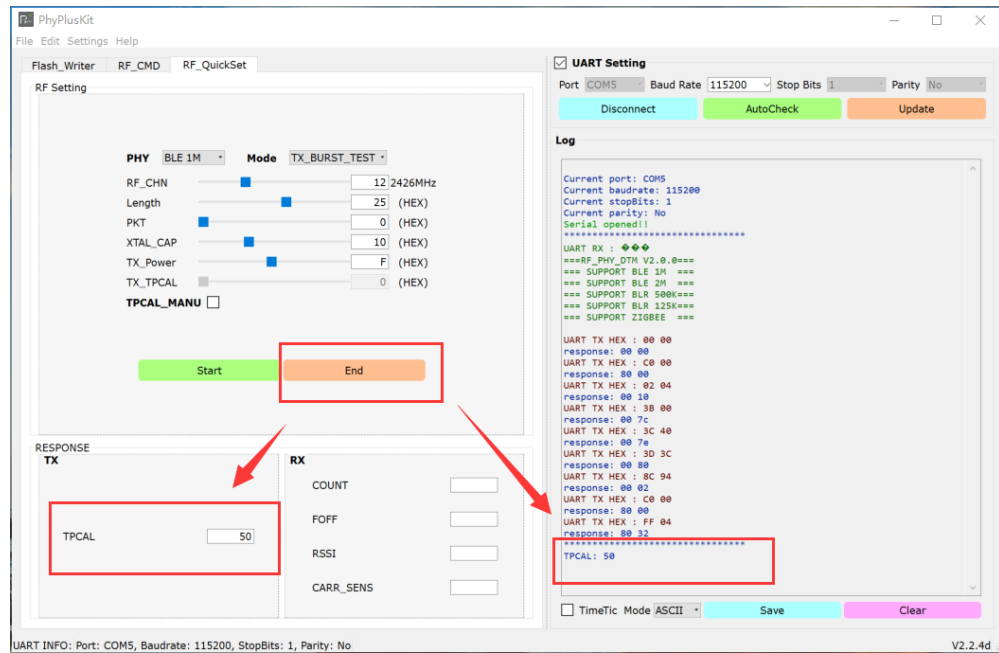
**XTAL\_CAP:** 用于调整芯片内部的 CAP loading, 改变 RF 的 Frequency Offset。

**TX\_Power:** 用于调整 RF 的发射功率，配置范围是 [0-0x1f] 该值与发射功率成正比。  
参考值 0x0A → 0dBm

**TPCAL\_MANU:** 影响 TX 的发射性能，如果不勾选 **Manual**，芯片内部会进行自动的 calibration。如果勾选了 **Manual**，需要手动填写 TPCAL 值。建议不勾选 **Manual**

**点击 START，改动的参数才能生效**

#### 4. 点击 End，结束测试，并获取相应参数（TPCAL）



当测试模式为 TX 时，点击 End 会自动更新 TPCAL 的结果。

当测试模式为 RX 时，更新下列参数：

COUNT：实际接收到的正确 Packet 个数。

FOFF：RX\_PHY 频率偏移估计值（KHZ）。

RSSI：接收信号强度的估计值(dBm)

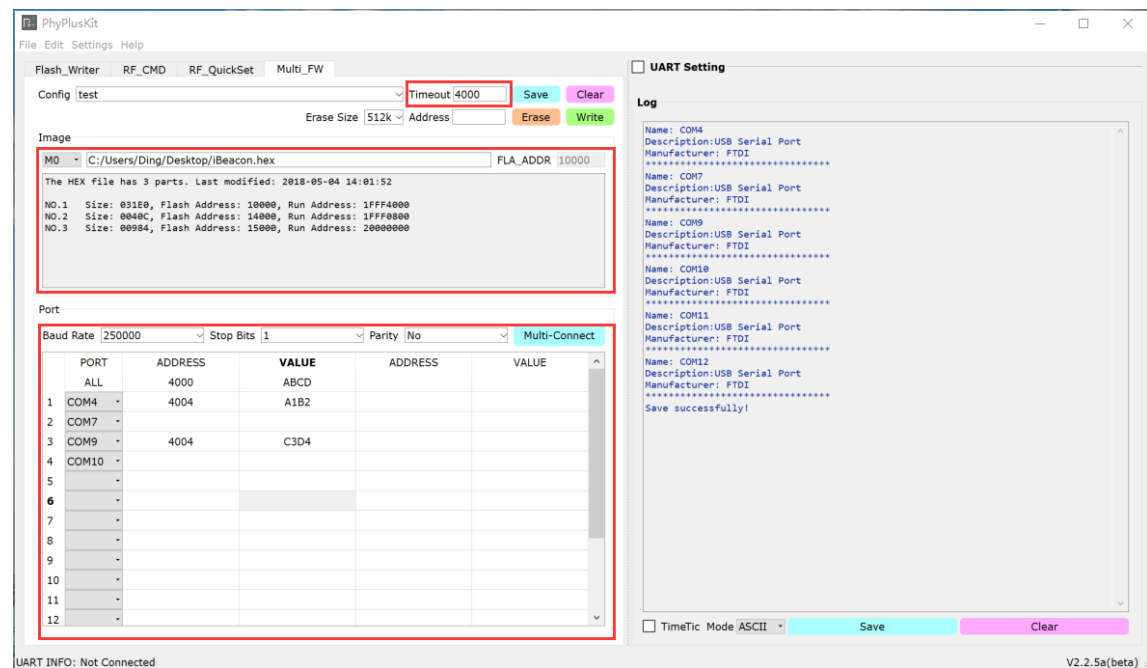
CARR\_SENS: 信号质量的估计值。

RX\_BURST\_TEST 模式，统计时间从点击 Start 到点击 End 为止。

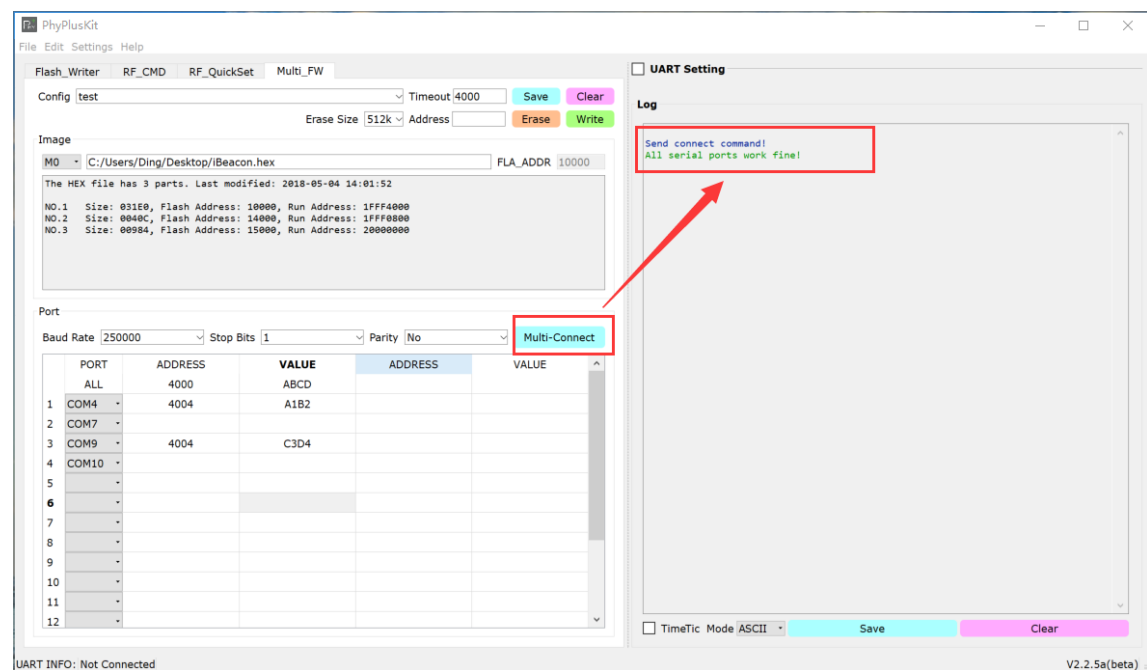
RX\_AUTO 模式，统计时间是 1000 个 Packet Interval。自动更新。点击 End，退出 RX\_AUTO 模式。

## 3.5 Multi-FlashWriter 使用

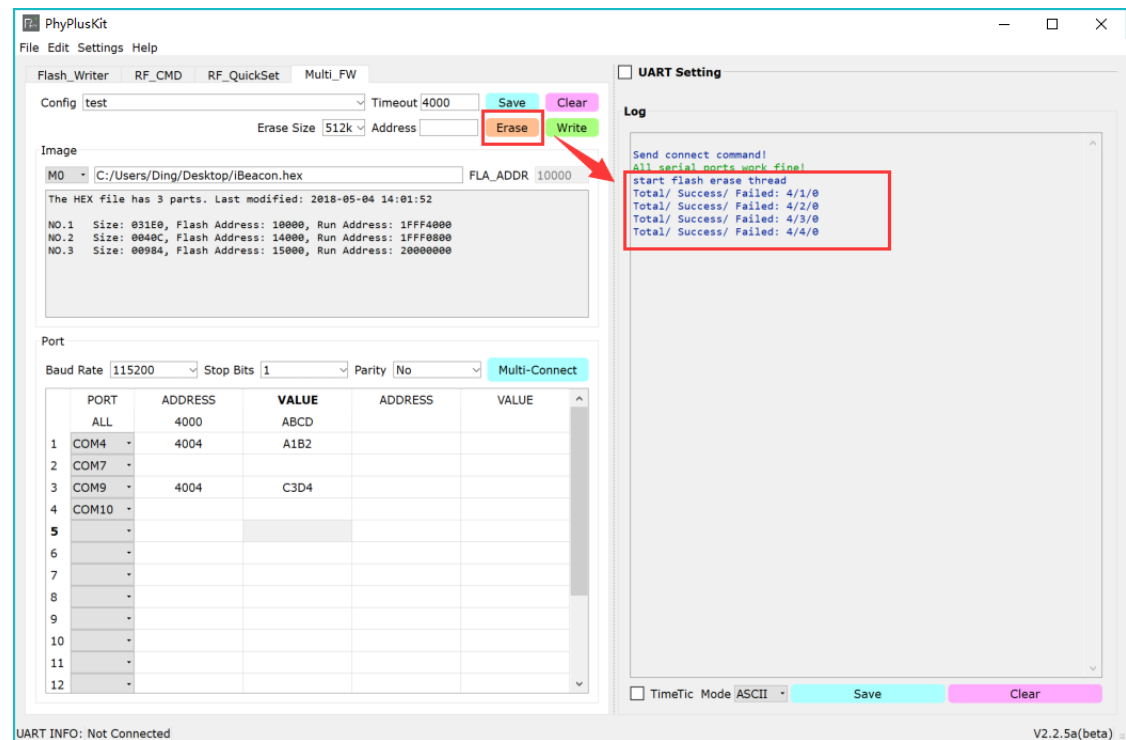
1. 设置合适的 Timeout 值（默认为 4000ms），选择待烧录的 HEX 文件并核对烧录的（Flash Address, Run Address），选择要烧录的 PORT 端口，并根据需要填写指定地址的值内容。



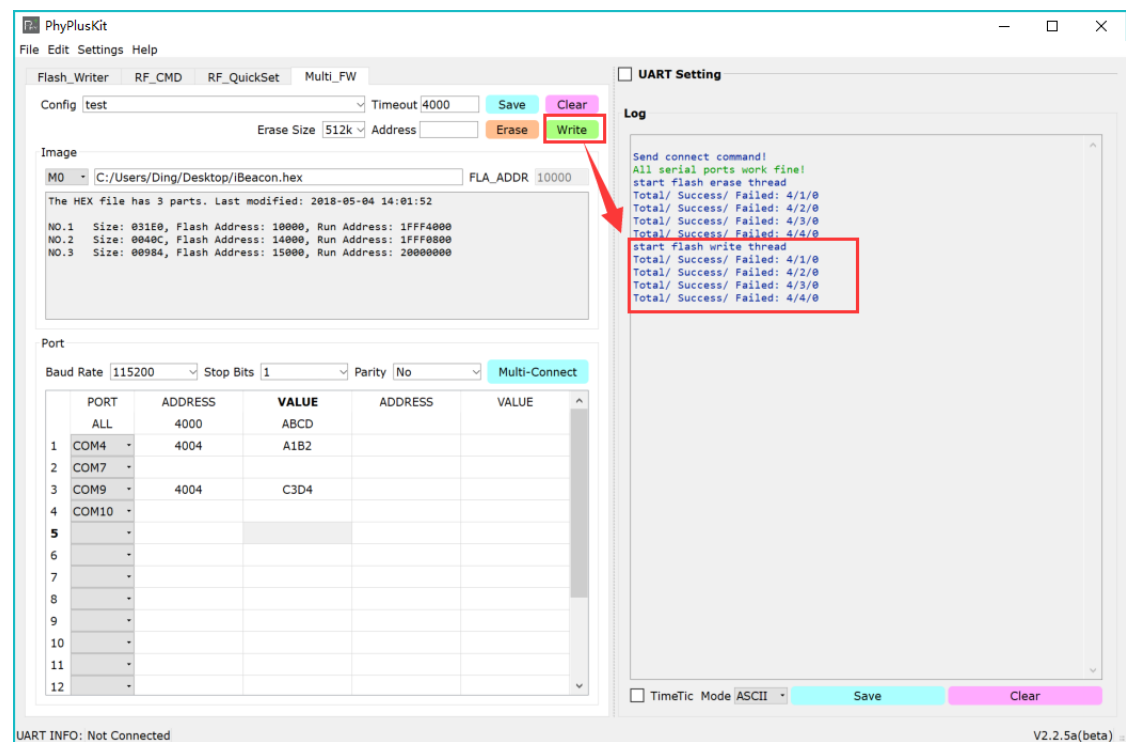
2. 待烧录的端口设置好后，点击 Multi-Connect 进行端口连接



3. 端口连接正常后，点击 **Erase** 按钮进行擦除（返回 总数/ 成功/ 失败的统计个数）



4. 设备擦除正常后，点击 **Write** 按钮进行烧写（返回 总数/ 成功/ 失败的统计个数）



## 3.6. 命令行模式下的烧写和合并操作

注意：

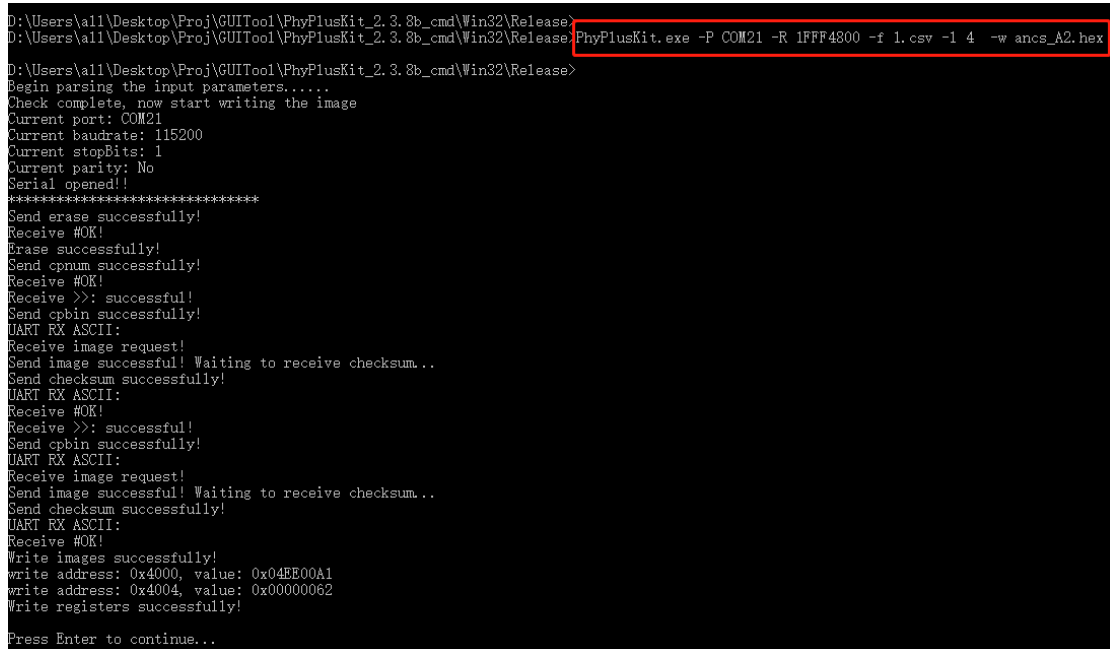
1. PhyPlusKit.exe 烧写工具、烧写固件、配置文件.csv 等需要在同一目录下
2. log 文件的保存时以追加的形式写入的。烧写过程会保存在一个 log 文件中，默认产生的 log 文件名为 zPhyPlusKit.log，与烧写工具在同一路径下。

### 3.6.1 仅烧写

1. 指令：PhyPlusKit.exe -P COM21 -R 1FFF4800 -f 1.csv -l 4 -w ancs\_A2.hex （hex 文件烧写）

描述：指定 Uart 口为 COM3，该芯片属于 PHY6212，设置 run address 为 1FFF4800，将 1.csv（指定行数，第 4 行）和 ancs\_A1.hex（和主程序在同一目录）写入到芯片中（芯片需要通过串口和电脑连接），写入前会自动执行擦除。（注意：TM 拉高）

效果截图



```
D:\Users\all\Desktop\Proj\GUITool\PhyPlusKit_2.3.8b_cmd\Win32\Release>
D:\Users\all\Desktop\Proj\GUITool\PhyPlusKit_2.3.8b_cmd\Win32\Release>PhyPlusKit.exe -P COM21 -R 1FFF4800 -f 1.csv -l 4 -w ancs_A2.hex
D:\Users\all\Desktop\Proj\GUITool\PhyPlusKit_2.3.8b_cmd\Win32\Release>
Begin parsing the input parameters.....
Check complete, now start writing the image
Current port: COM21
Current baudrate: 115200
Current stopBits: 1
Current parity: No
Serial opened!!
*****
Send erase successfully!
Receive #OK!
Erase successfully!
Send cpnum successfully!
Receive #OK!
Receive >>: successful!
Send cpbin successfully!
UART RX ASCII:
Receive image request!
Send image successful! Waiting to receive checksum...
Send checksum successfully!
UART RX ASCII:
Receive #OK!
Receive >>: successful!
Send cpbin successfully!
UART RX ASCII:
Receive image request!
Send image successful! Waiting to receive checksum...
Send checksum successfully!
UART RX ASCII:
Receive #OK!
Write images successfully!
write address: 0x4000, value: 0x04EE00A1
write address: 0x4004, value: 0x00000062
Write registers successfully!
Press Enter to continue...
```

2. 指令：PhyPlusKit.exe -P COM21 -R 1FFF4800 -f 1.csv -l 4 -w ancs\_A2.hexf （hexf 文件的烧写）

描述：指定 Uart 口为 COM3，该芯片属于 PHY6212，设置 run address 为 1FFF4800，将 1.csv（指定行数，第 4 行）和 ancs\_A1.hexf（和主程序在同一目录）写入到芯片中（芯片需要通过串口和电脑连接），写入前会自动执行擦除。

效果截图

```

D:\Users\all\Desktop\Proj\GUITool\PhyPlusKit_2.3.8b_cmd\Win32\Release>
D:\Users\all\Desktop\Proj\GUITool\PhyPlusKit_2.3.8b_cmd\Win32\Release>
D:\Users\all\Desktop\Proj\GUITool\PhyPlusKit_2.3.8b_cmd\Win32\Release>PhyPlusKit.exe -P COM21 -R 1FFF4800 -f 1.csv -l 4 -w ancs_A2.hexf
D:\Users\all\Desktop\Proj\GUITool\PhyPlusKit_2.3.8b_cmd\Win32\Release>
Begin parsing the input parameters.....
Check complete, now start writing the image
Current port: COM21
Current baudrate: 115200
Current stopBits: 1
Current parity: No
Serial opened!!
*****
Send erase successfully!
Receive #OK!
Erase successfully!
Send cpnum successfully!
Receive #OK!
Receive >>: successful!

```

```

Receive #OK!
Receive >>: successful!

=====Write hexf File [01/03]=====
Send cpbin successfully!
UART RX ASCII:
Receive image request!
Send image successful! Waiting to receive checksum..
Send checksum successfully!
UART RX ASCII:
Receive #OK!
Receive >>: successful!

=====Write hexf File [02/03]=====
Send cpbin successfully!
UART RX ASCII:
Receive image request!
Send image successful! Waiting to receive checksum..
Send checksum successfully!
UART RX ASCII:
Receive #OK!
Receive >>: successful!

=====Write hexf File [03/03]=====
Send cpbin successfully!
UART RX ASCII:
Receive image request!
Send image successful! Waiting to receive checksum..
Send checksum successfully!
UART RX ASCII:
Receive #OK!
write address: 0x4000, value: 0x04EE00A1
write address: 0x4004, value: 0x00000062
Write registers successfully!

Press Enter to continue...

```

### 3.6.2 仅合并

指令: PhyPlusKit.exe -c -p wrist\_115a.hex -r E:\test\test\bin\Debug\test.bin -a 70000 -m NO -e chip

描述: 执行合并命令, 模式为 No OTA, app 文件为 wrist\_115a.hex, resource 文件为 E:\test\test\bin\Debug\test.bin, 其写入起始地址为 0x70000, 加密方式为 chip id 加密

效果截图:

```

E:\PhyPlusKit1\Win32\Release>
E:\PhyPlusKit1\Win32\Release>
E:\PhyPlusKit1\Win32\Release>PhyPlusKit.exe -c -p wrist_115a.hex -r E:\test\test\bin\Debug\test.bin -a 70000 -m NO -e chip
E:\PhyPlusKit1\Win32\Release>
Begin parsing the input parameters.....
Merge option is set, ready to merge hex files
Checking the merge mode
No OTA mode is set, skipping the check of OTA Boot file
Checking APP file
Checking the binary resource files and addresses
Checking the encryption mode
Check complete, now start merging the input files
*****
Current port: COM3
Current baudrate: 115200
Current stopBits: 1
Current parity: No
Serial opened!!
*****
123456789012345612345678901234561234567890123412345678
CHECK CHIP ID[VALID]!
CHIP ID: 123456789012345612345678901234561234567890123412345678

```

### 3.6.3 合并后烧写

指令：PhyPlusKit.exe -c -b ota.hex -p wrist\_115a.hex -r E:\test\test\bin\Debug\test.bin -a 70000 -m DH -e iv\_1234567890123 -w wrist\_115a.hexf

描述：执行合并命令，模式为 Dual Has FCT，boot 文件为 ota.hex，app 文件为 wrist\_115a.hex，resource 文件为 E:\test\test\bin\Debug\test.bin，其写入起始地址为 0x70000，加密方式为 iv 值加密且 iv 为 1234567890123。

效果截图：

```

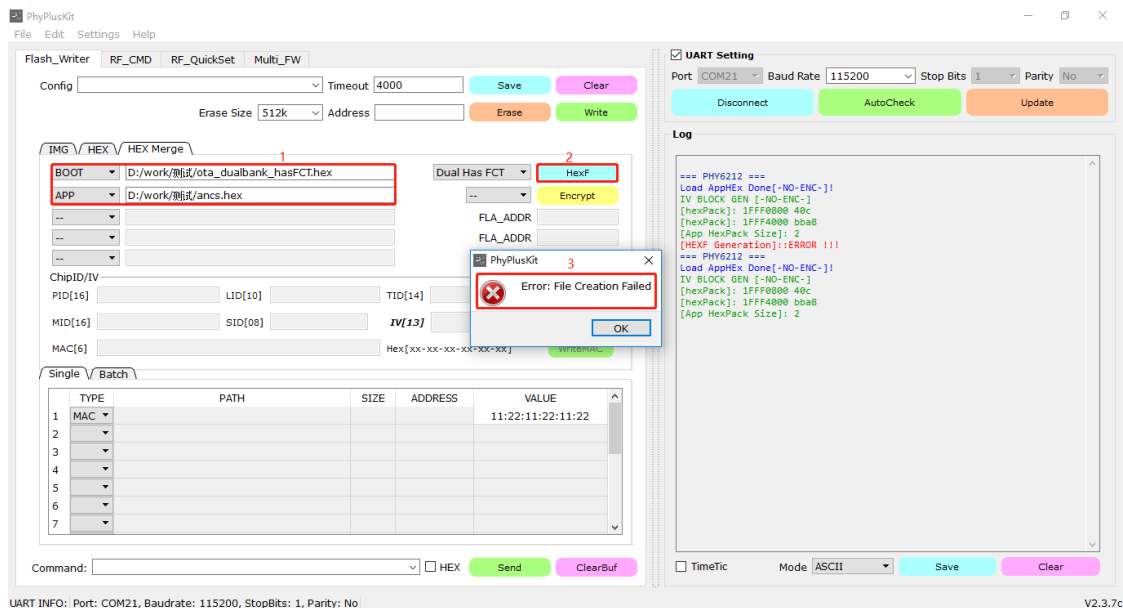
E:\PhyPlusKit1\Win32\Release>PhyPlusKit.exe -c -b ota.hex -p wrist_115a.hex -r E:\test\test\bin\Debug\test.bin -a 70000 -m DH -e iv_1234567890123 -w wrist_115a.hexf
E:\PhyPlusKit1\Win32\Release>
Begin parsing the input parameters.....
Merge option is set, ready to merge hex files
Checking the merge mode
Checking OTA Boot file
Checking APP file
Checking the binary resource files and addresses
Checking the encryption mode
Check complete, now start merging the input files
Current port: COM3
Current baudrate: 115200
Current stopBits: 1
Current parity: No
Serial opened!!
*****
Start Hex Encrypt.....
The HEX file has 3 parts. Last modified: 2018-08-13 17:01:49
IV:1234567890123
#0 size = 818
#1 size = 18000
#2 size = 7240

```

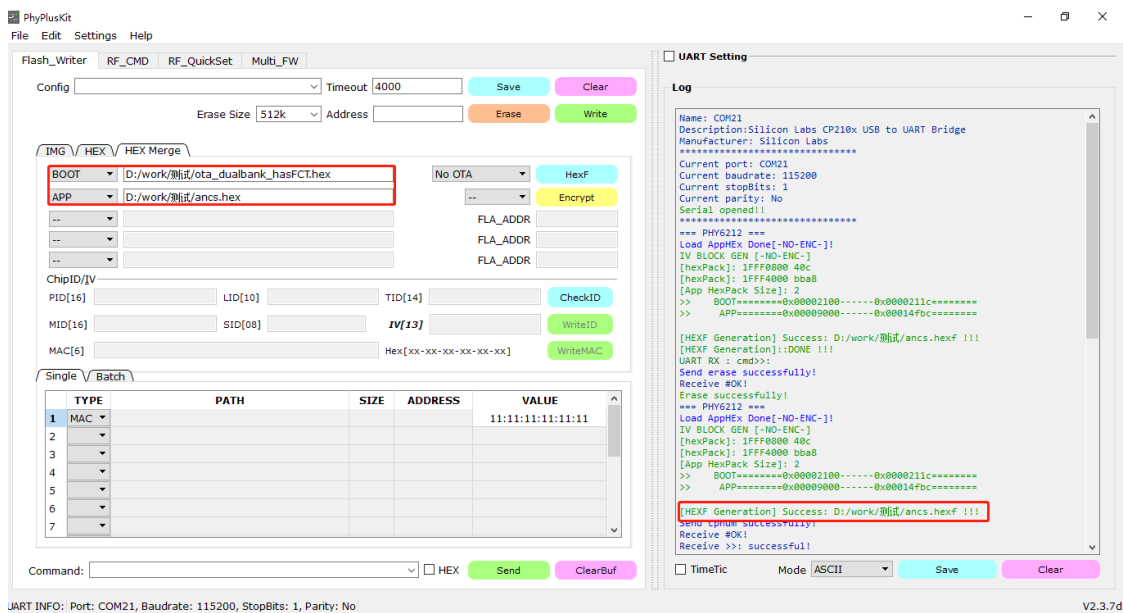
## 3.7. Flash 烧写的新增配置

### 3.7.1 BOOT 和 APP 支持中文路径

V2.3.7c 及以前的版本，不支持中文路径，具体表现如下：

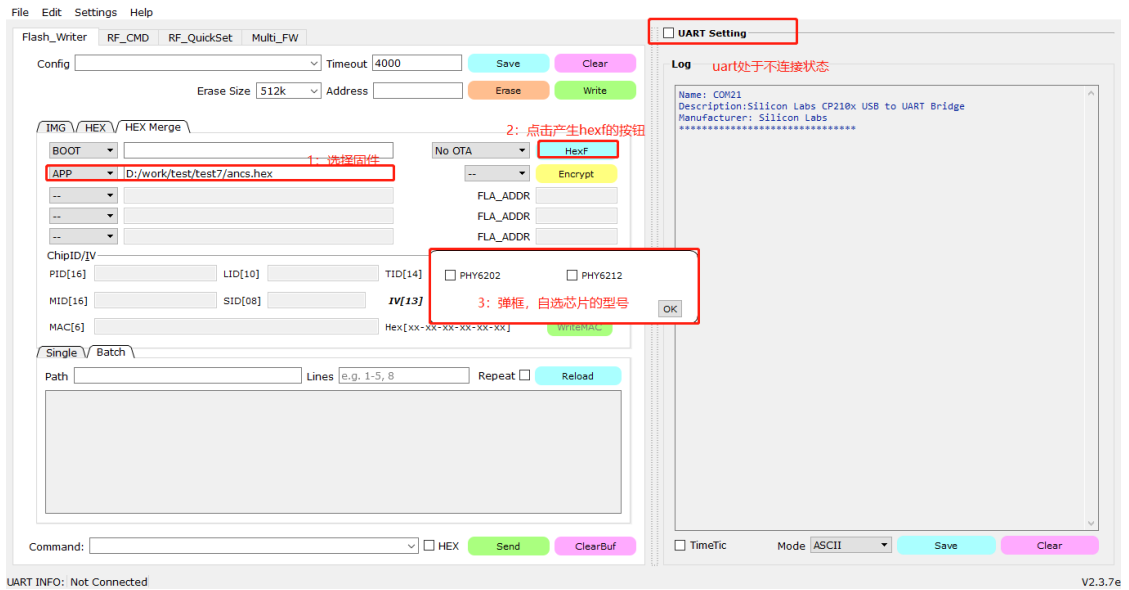


如图所示，BOOT 和 APP 的 File 路径中存在中文字符，就会报错：“File Creation Failed”，在接下来的 v2.3.7d 及之后的版本将支持中文路径，不影响 hexf 文件的产生及烧写过程。

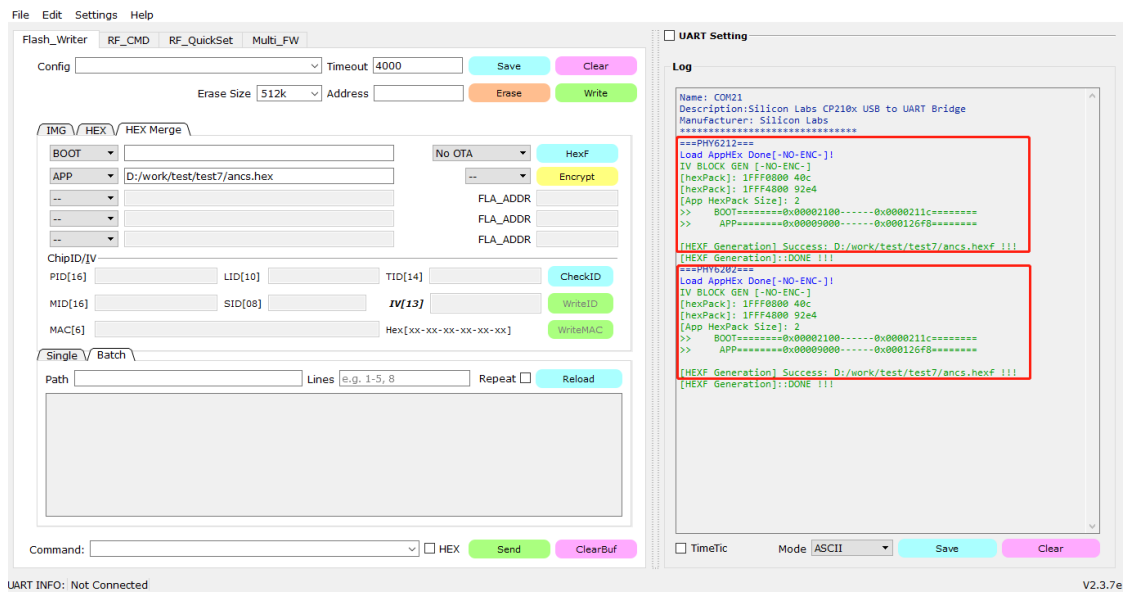


### 3.7.2 Uart 不连接时的自选设置

V2.3.7e 版本开始，Uart 不连接时由芯片型号的默认设置修改为自选设置，具体体现如下图：

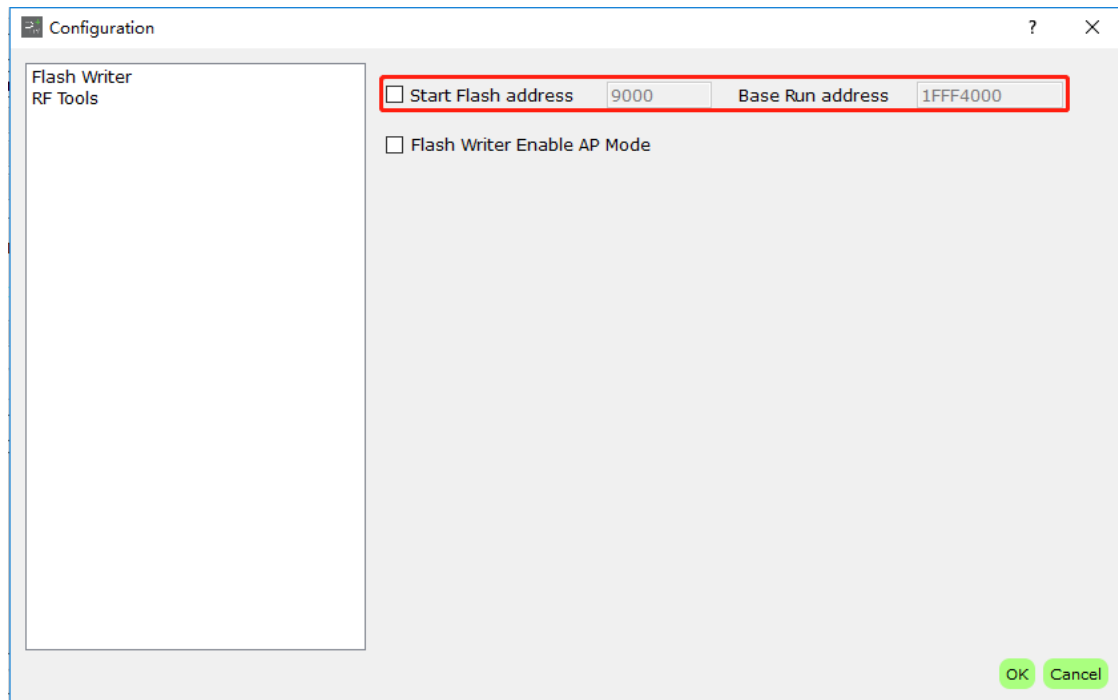


- Uart 处于不连接状态
- 加载应用固件.hex 文件
- 点击 HexF 按钮，弹出选择对话框
- 选择对应的芯片型号，点击 OK 按钮后，即可对应产生所加载应用固件的 hexf 文件

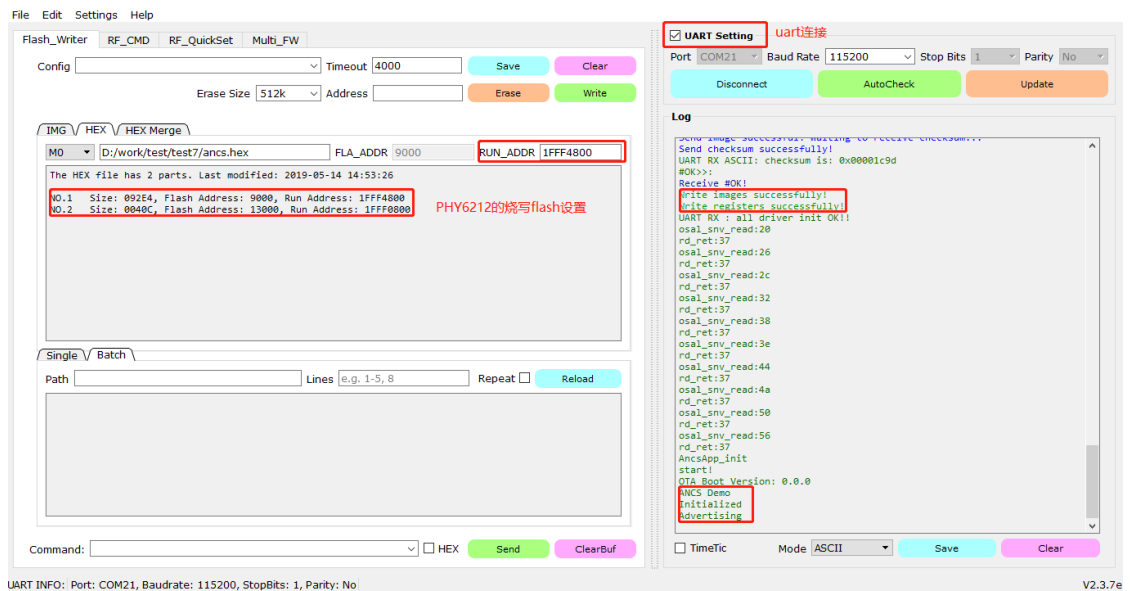
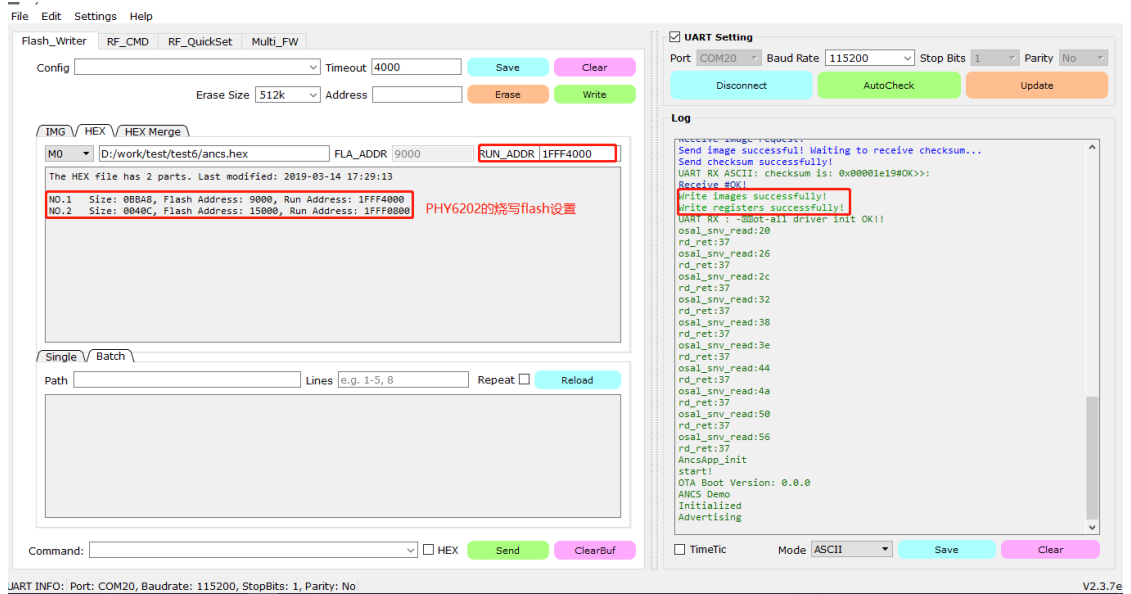


### 3.7.3 Uart 连接时的烧写配置

uart 连接时，芯片型号可自动识别。



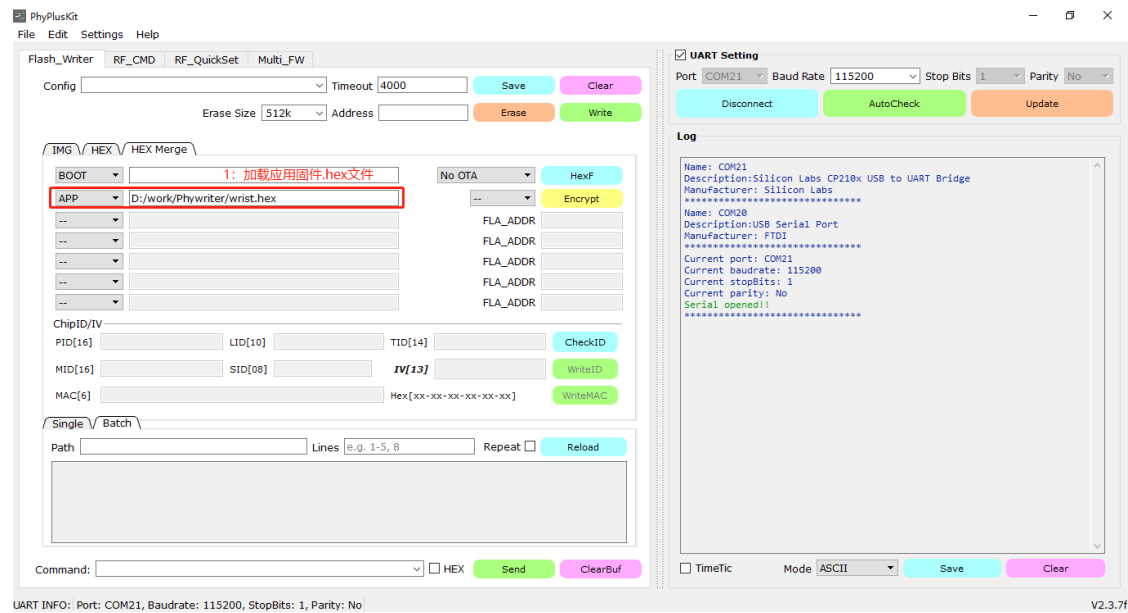
1. setting->configuration 配置  
Start Flash address 和 Base Run address 前设置勾选框，不勾选时默认设置：Start Flash address: 9000，Base Run address: 1FFF4000；勾选，可根据需要修改参数。
2. 烧写的 flash 配置：  
PHY6202 → Base Run address:1FFF4000  
PHY6212 → Base Run address:1FFF4800  
在 HEX 标签页添加 RUN\_ADDR 的输入框，可以进行 flash 配置的修改，不需要打开 configuration 进行修改设置，方便用户操作。



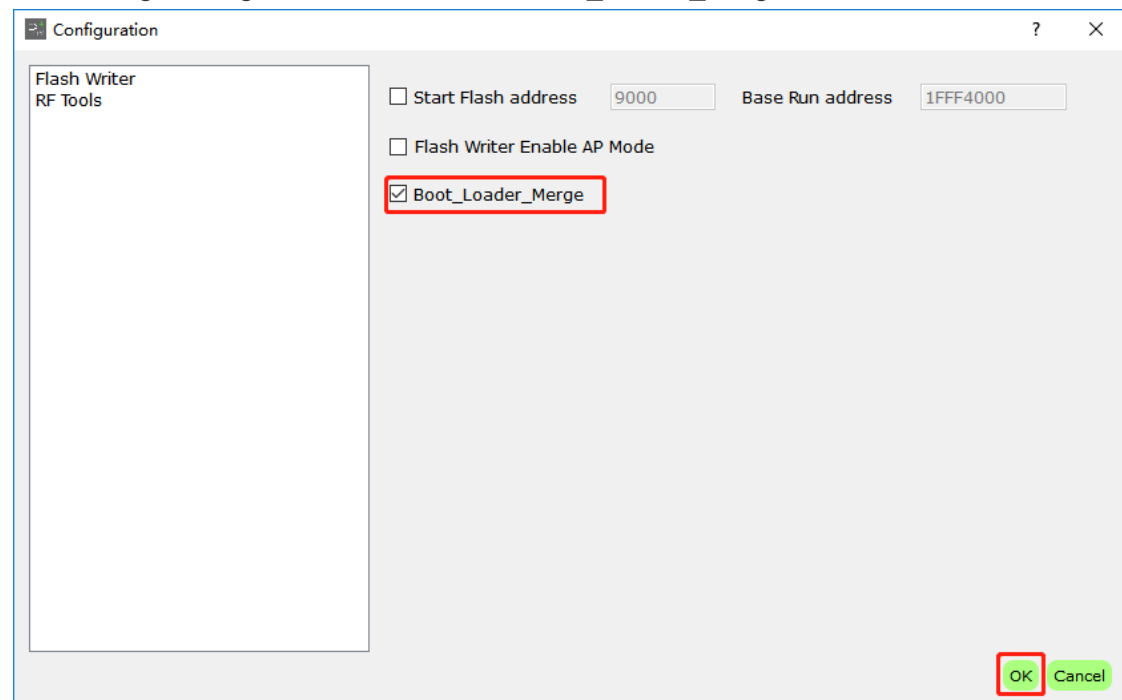
### 3.7.4 脱机烧录 1M flash 烧录文件的合成

一、通过 HEXMerge 页面的 HEXF 按钮将.hex 应用固件生成的.hexf 文件与烧录引导文件\*.hexf 合并，该功能从 v2.3.7f 版本开始。

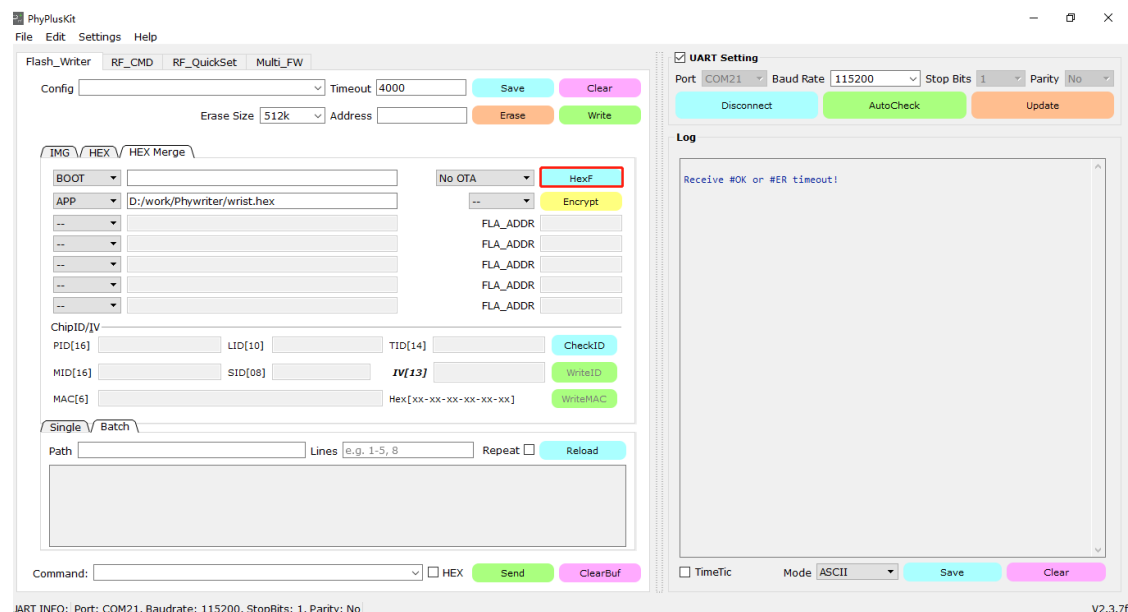
1.加载需要烧录的应用固件.hex 文件



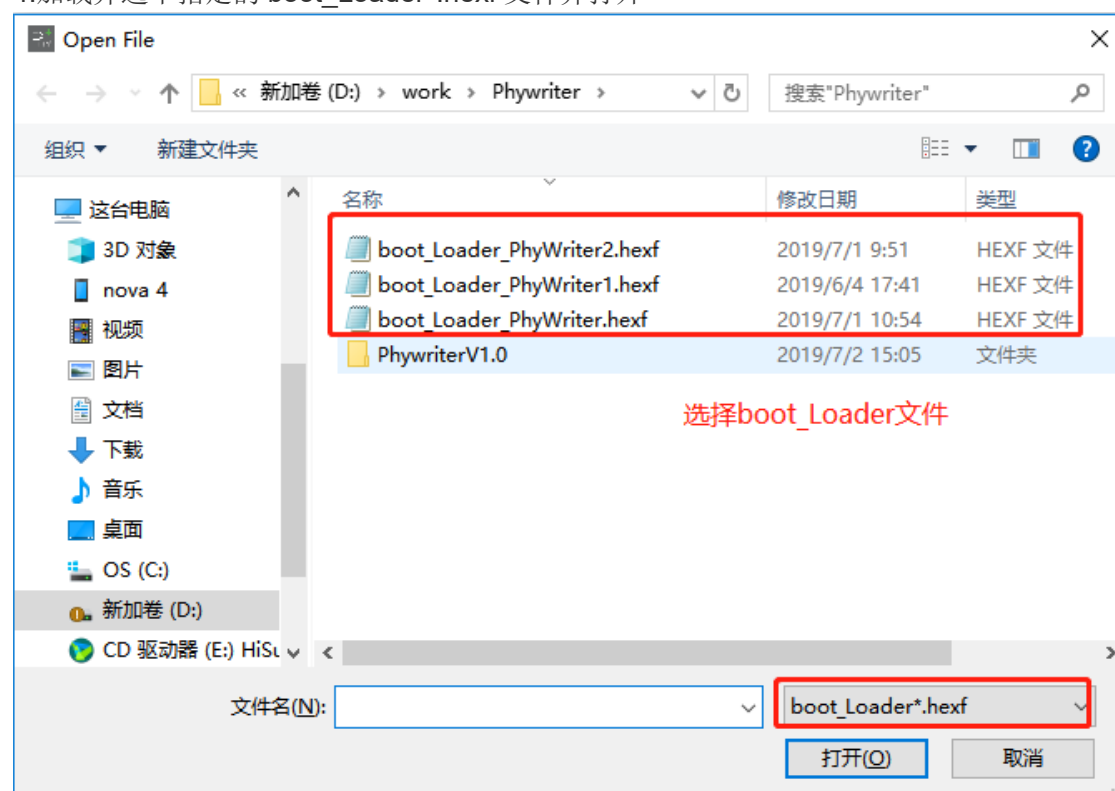
2.在 setting->configuration 的配置中勾选 Boot\_Loader\_Merge 勾选框



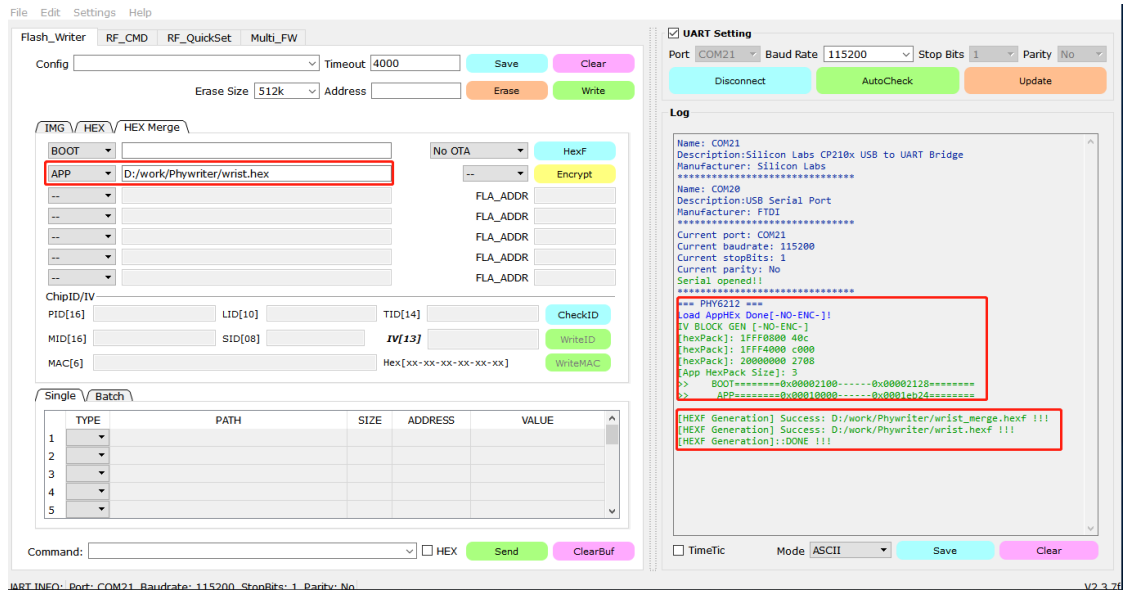
3.点击 HexF 按钮



4.加载并选中指定的 boot\_Loader\*.hexf 文件并打开



5.生成\*merge.hexf 文件（加 bootloader）和.hexf（不加 bootloader）文件。（截取两个\*merge.hexf 文件的初始位置以示说明，两个 hexf 文件 merge 成功）



```

1  :02000004FFFFFC
2  :102100000200000000A00000601D00000040FF1F52
3  :0C21100068BD000000C040000008FF1F68
4  :02000004FFFFFC boot文件初始位置
5  :10A000003868FF1FD540FF1F00000000000000005F
6  :10A01000000000000000000000000000000000040
7  :10A020000000000000000000000000000000000030
8  :10A030000000000000000000000000000000000020
9  :10A040000000000000000000000000000000000010
10 :10A0500000000000000000000000000000000000000
11 :10A0600000000000000000000000000000000000F0
12 :10A0700000000000000000000000000000000000E0
13 :10A0800000000000000000000000000000000000D0
14 :10A0900000000000000000000000000000000000C0
15 :10A0A00000000000000000000000000000000000B0
16 :10A0B00000000000000000000000000000000000A0
17 :10A0C0000348854600F01CF8004800472956FF1F4A
18 :10A0D0003868FF1F0648804706480268064B1A4248
19 :10A0E00001D106480047FEE7FEE7FEE7FEE7FEE790
20 :10A0F000DD43FF1FA8F0004001000000C140FF1F2A
21 :10A10000064C0125064E05E0E36807CC2B430C3CCA
22 :10A1100098471034B442F7D3FFF7D6FFCC5CFF1F4B
23 :10A12000FC5CFF1F30B58C180278401C13071B0F16
24 :10A1300001D10378401C120906D10278401C03E0CB
538 :10C150000000000000000000000000000000000DF
539 :10C160000000000000000000000000000000000FF01FF01FF01FFCF
540 :04C17000910000003A boot文件的末尾及应用
541 :020000040000FA 固件的初始位置
542 :1021000003000000000000000000000000000040FF1FAD
543 :1021100008C001000C0400000008FF1FC40100DF
544 :08212000082700000000000002068
545 :020000040001F9
546 :1000000068400020D540FF1FDD40FF1FDF40FF1F7D
547 :10001000000000000000000000000000000000E0
548 :10002000000000000000000000000000000000E140FF1F91
549 :10003000000000000000000000000000000000E340FF1FE540FF1F3C
550 :100040000000000000000000000000000000000B0
551 :100050000000000000000000000000000000000A0
552 :10006000000000000000000000000000000000090
553 :10007000000000000000000000000000000000080
554 :10008000000000000000000000000000000000070
555 :10009000000000000000000000000000000000060
556 :1000A000000000000000000000000000000000050
557 :1000B000000000000000000000000000000000040
558 :1000C0000348854600F04CFD00480047C5D2FF1F9D
559 :1000D000684000200448804704480047FEE7FEE7E8
560 :1000E000FEE7FEE7FEE7FEE7D570FF1FC140FF1FFA

```

注：只有在 setting->configuration 的配置中勾选 Boot\_Loader\_Merge，才能触发两个 hexf 的 merge 功能。（重新打开 PhyPlusKit.exe 需要重新进行勾选设置）

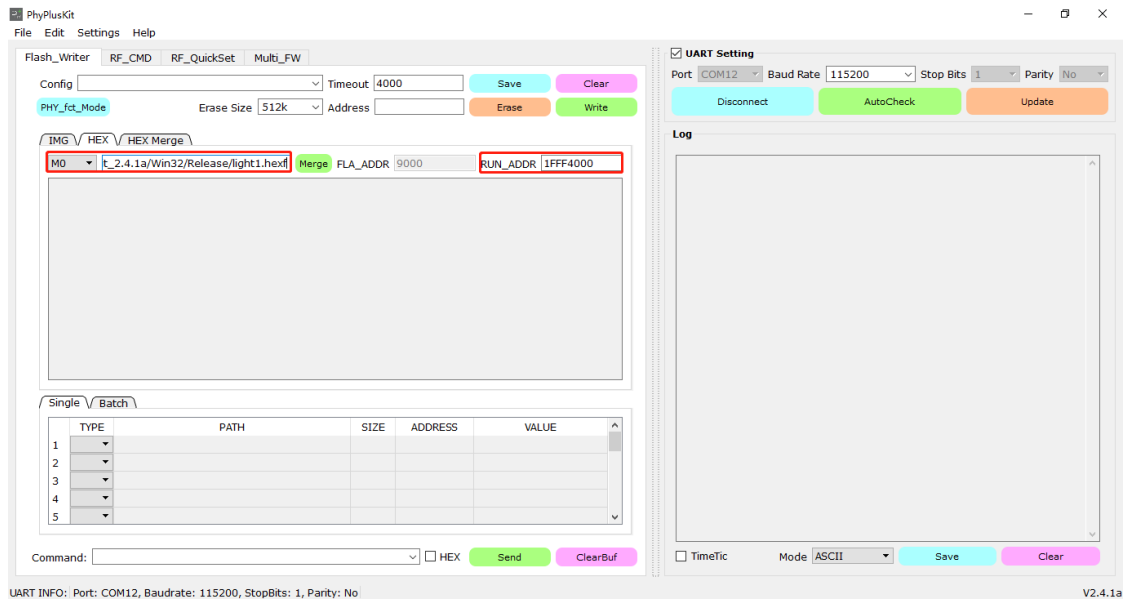
二、通过 HEX 页面的 merge 按钮将两个.hexf 文件进行合成（V2.4.1a 版本）

注意：M0 后的固件的格式必须是.hexf，.hex 文件与.hexf 的 merge 过程详见内容一。1.错误提示：

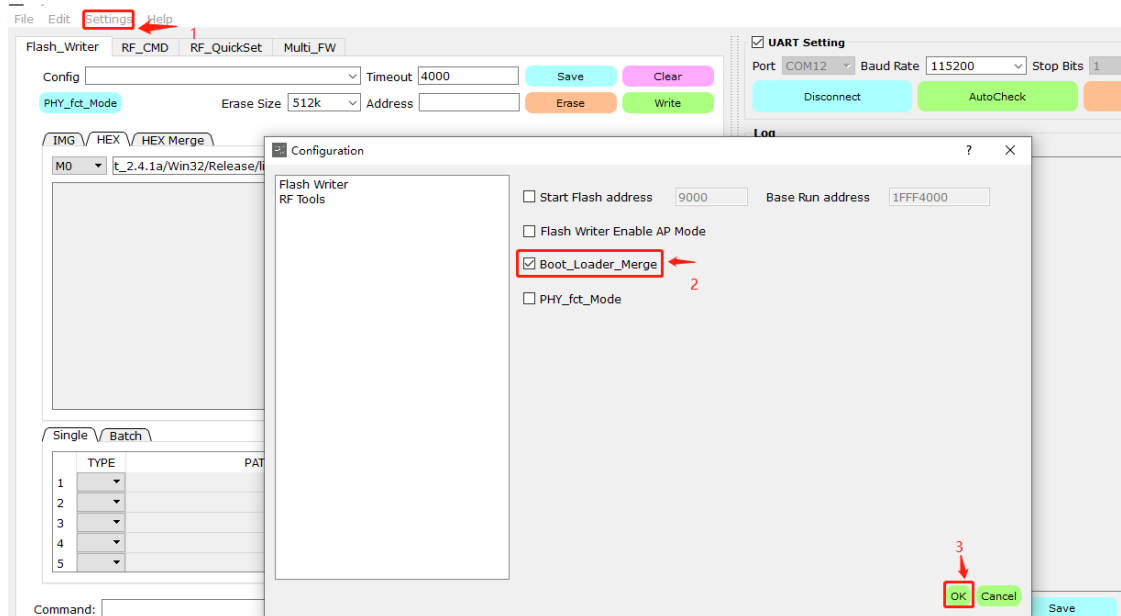
- 1) HEX 页面不加载固件，点击 merge 按钮，提示：The merge file is empty.
- 2) HEX 页面加载.hex 格式固件，点击 merge 按钮，提示：The merge file is not valid.
- 3) HEX 页面加载.hexf 格式固件，没有添加合成文件，点击 merge 按钮，提示：[HEXF Merge] Fail:+加载的文件路径

2.两个.hexf 文件合成步骤

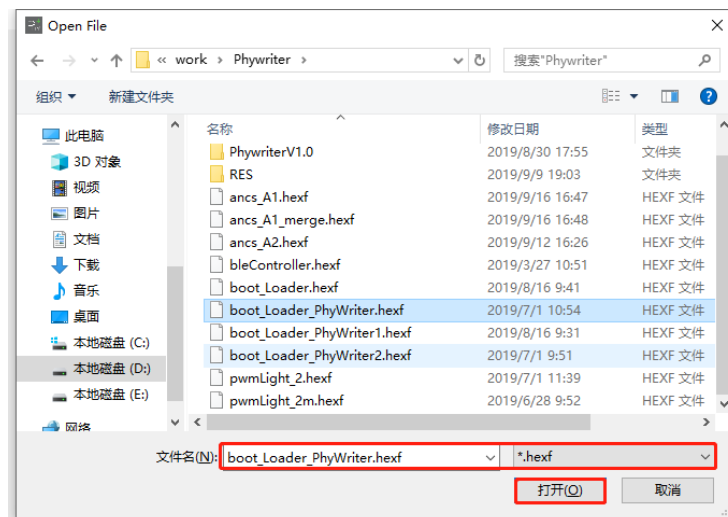
- 1) 在 HEX 页面加载应用固件.hexf 文件，并对应修改固件运行的 RUN\_ADDR，PHY6202--1FFF4000，PHY6212--1FFF4800



2) setting—configuration 进行 Boot\_Loader\_Merge 勾选框设置，勾选后，点击 OK



3) 点击 merge 按钮，弹出需要 merge 的.hexf 文件



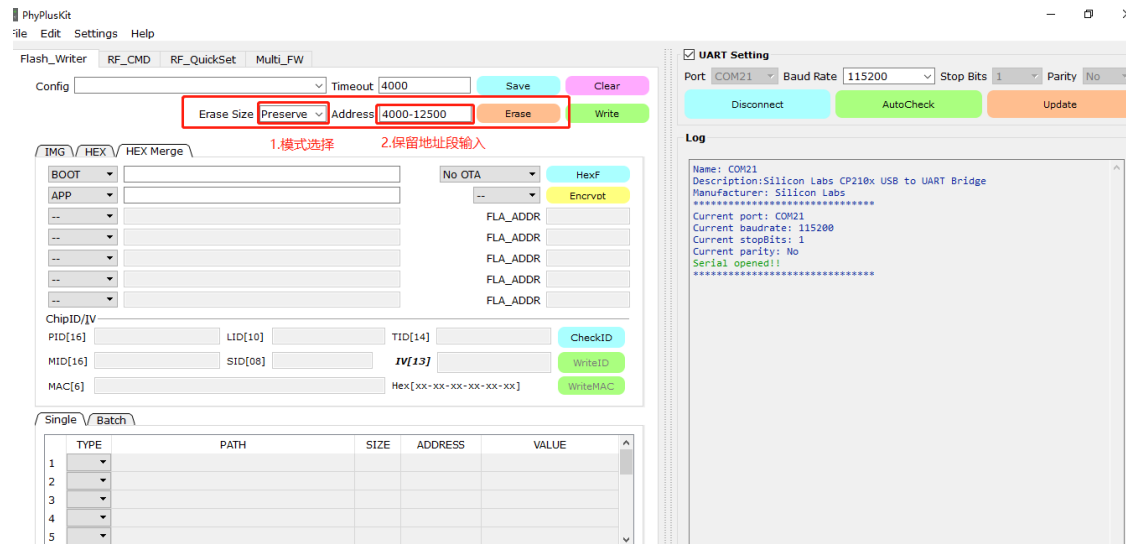
4) 点击打开后，界面 LOG 提示：[HEXF Merge] Success:+加载的固件路径\_merge.hexf

### 3.7.5 Preserve 模式的地址段保留和多地址段的擦除

V2.3.8a 开始，在 Erase 擦除功能中添加 Preserve 模式，可以实现某一地址段的 flash 区域保留和多个指定地址段的直接擦除功能。具体表现如下：

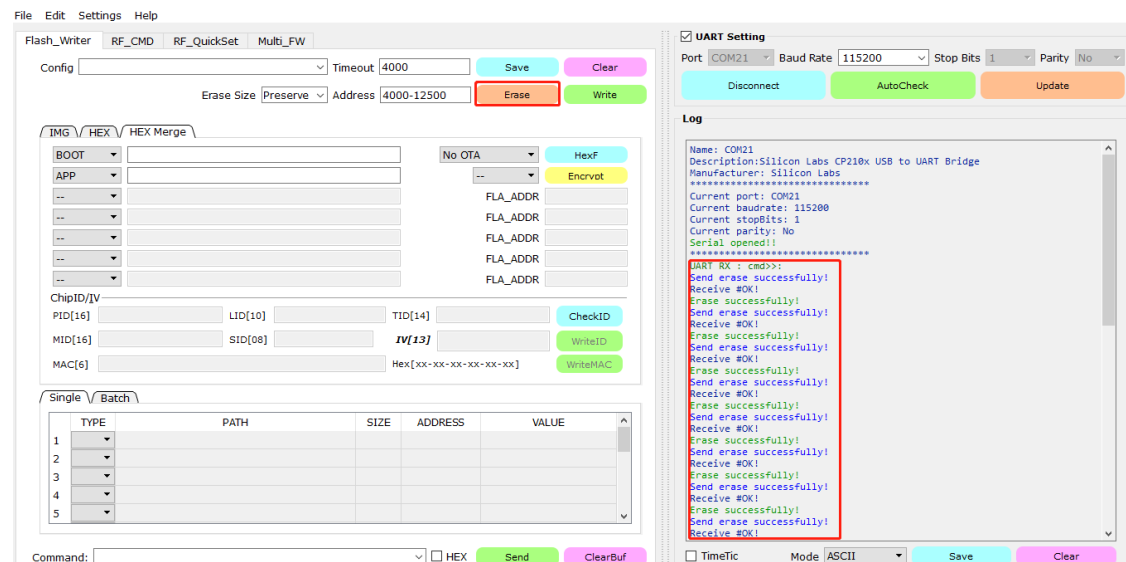
### 1. 某一地址段的保留功能 (Preserve)

### A. Preserve 模式的选择和保留地址段的输入（例如：4000-12500 如下图）



### B. TM 拉高，reset 开发板

C. 点击 Erase 按钮，擦出成功，4000-12500 之间的地址段被保留



## 2.Preserve 模式的多地址段擦除功能（例如：2000~5000,8000~12500）

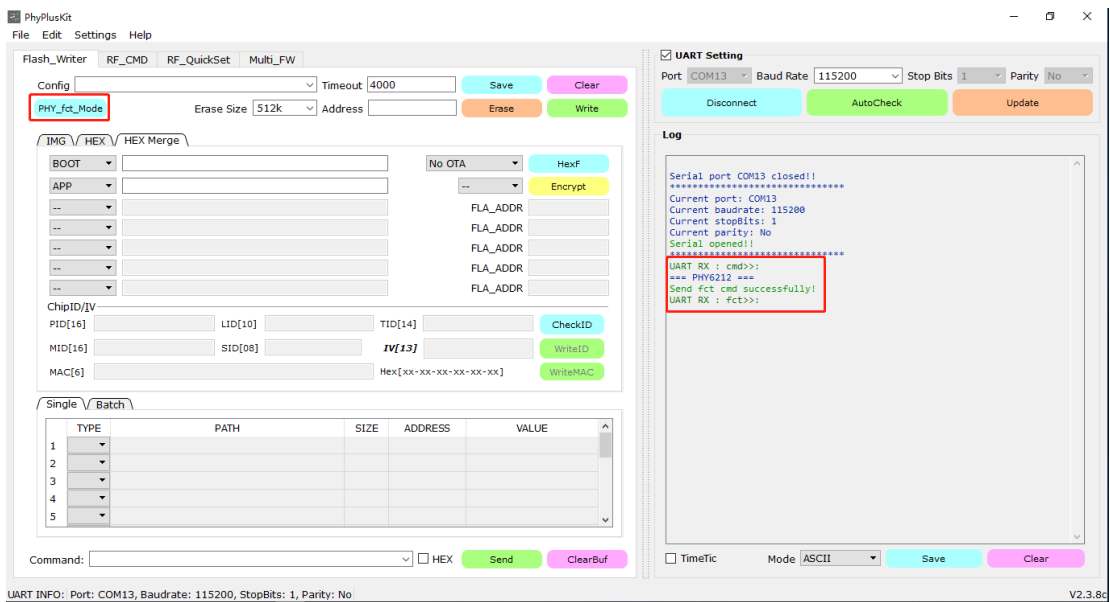
备注：多个地址段中间“,”必须是英文字符输入。只有输入的多个地址段的 flash 内容会被擦除

### 3.7.6 PHY\_fct\_Mode 功能

V2.3.8b 版本开始，PhyPlusKit 支持 **PHY6212 产品**的 FCT 模式。当芯片进入 FCT mode 时，无法在烧写模式下去读写寄存器等，可以保证程序的安全性。具体操作流程如下：

具体支持 FCT 模式有两条路径：

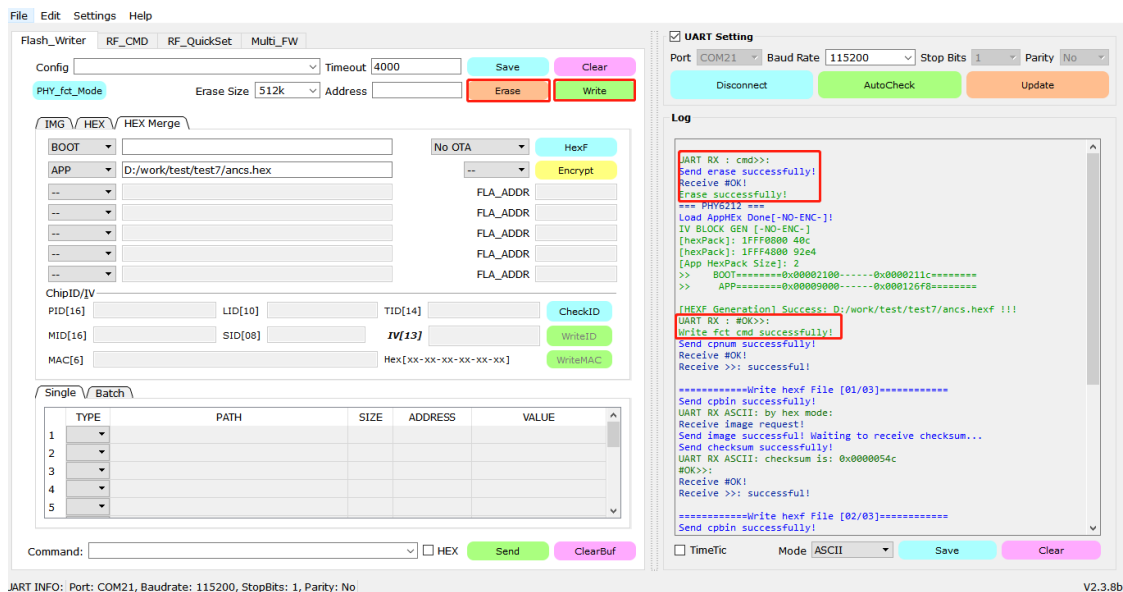
1. 主界面的 PHY\_fct\_Mode 按钮，方便用户随时点击进入 FCT 模式。  
具体步骤：
  - a. TM 拉高，reset 开发板，log 返回 UART RX : cmd>>:
  - b. 点击 PHY\_fct\_Mode 按钮，log 打印：
    - a) === PHY6212 ===
    - b) Send fct cmd successfully!
  - c. TM 处于拉高状态，reset 开发板，log 返回 UART RX : fct>>:则进入 FCT 模式



2. 通过 setting--configuration 中进行 FCT 模式的配置：  
具体步骤如下：
  - a. 点击 setting—configuration，进入配置文件的设置



- b. 勾选 PHY\_fct\_Mode 勾选框，点击 OK，这个设置会一直保存
- c. TM 拉高，reset 开发板，返回：UART RX : cmd>>:
- d. 点击 Erase 按钮，成功进行 erase 操作
- e. 点击 Write 按钮进行固件烧写，芯片进入烧写模式，在 log 信息会打印显示：
  - a) UART RX : #OK>>:
  - b) Write fct cmd successfully!
- f. 此时芯片已进入 FCT 模式，可以通过 reset 开发板进行验证，返回 UART RX : fct>>:



### 3. 退出 FCT 模式的操作：

- a) Erase Size 为 512K 时
- b) 点击 Erase 按钮进行擦除
- c) Reset 开发板
- d) 成功退出 fct 模式，返回 UART RX : cmd>>:

注：PHY\_fct\_Mode 模式目前只支持 PHY6212 芯片产品。

方式 1：方便用户快速进入 FCT 模式，点击按钮就可触发并进入 FCT 模式

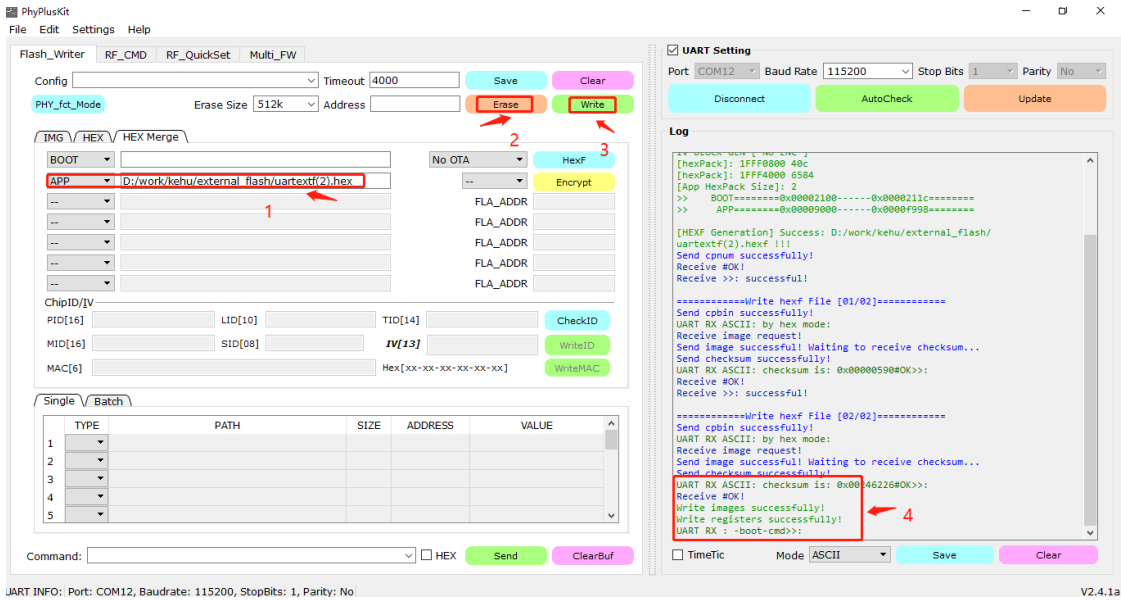
方式 2：方便用户多次使用 FCT 模式，并能一直保持触发。注：每次重新打开软件，需要重新进行配置勾选才能保持触发 FCT 模式

### 3.7.7 支持外挂 flash 的烧写功能

V2.4.1a 版本的主要更新功能就是支持外挂 flash 的烧写。（主要针对 PHY6202 产品）

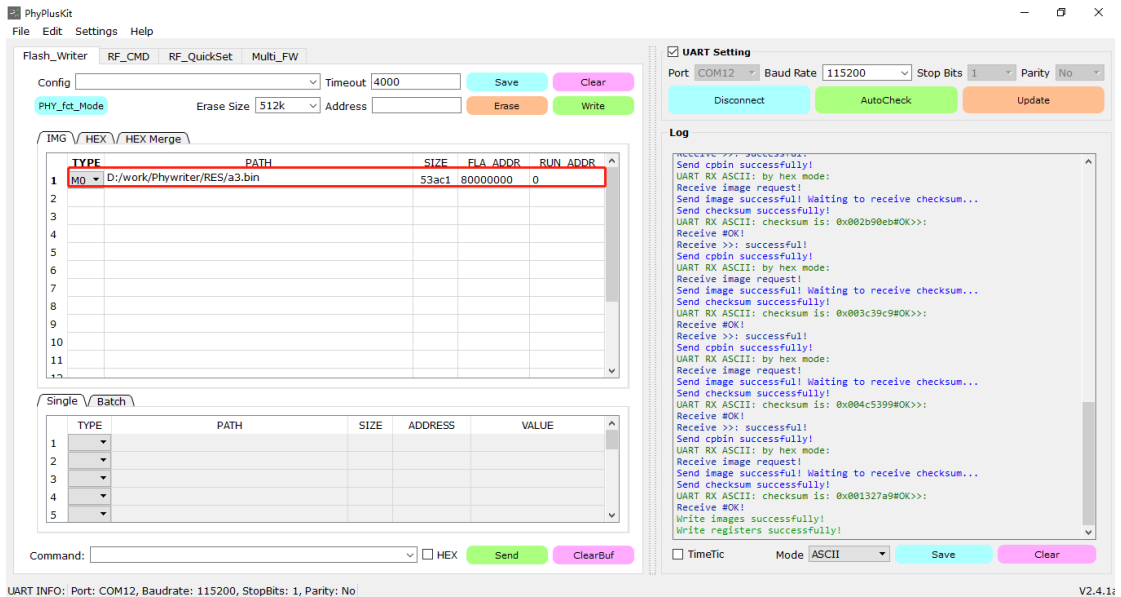
#### 1. 烧写支持外挂 flash 的 boot 文件--uartextf.hex 文件

TM 拉高，reset，进行 erase 和 write 操作；烧写成功后，TM 拉低，reset，Log 打印：UART RX：-boot-cmd>>:



#### 2. 外挂 flash 的烧写

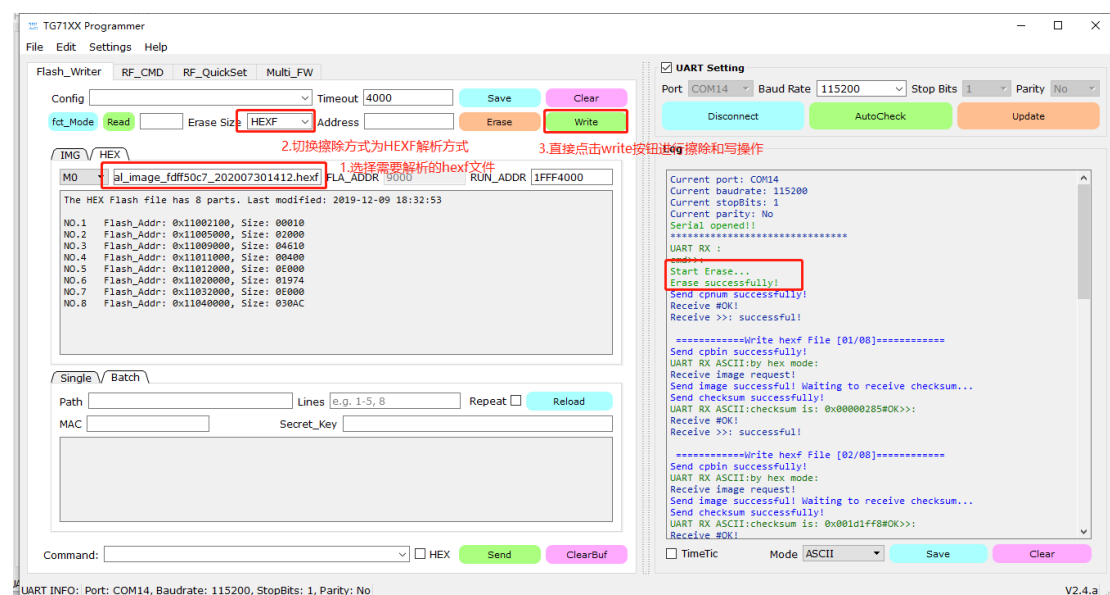
IMG 页面：Path 中双击加载\*.bin 文件；size 是加载的文件大小，自动生成；  
FLA\_ADDR：外挂 flash 的 flash address 是以 80000000 为基准，可以自行增加数值；  
RUN\_ADDR：可任意填写数值（也可支持 1M 波特率进行烧写）



## 3.7.8 HEXF 解析的保留擦除模式

V2.4.5a 开始，在 Erase 擦除功能中添加 HEXF 擦除模式，通过解析需要写入 flash 的 hexf 文件内容，进行选择性的擦除需要写入的 flash 的地址段，其余位置不做擦除操作，具体的操作步骤如下：

1. 选择对应需要解析的 hexf 文件，注意一定要是 hexf 文件，可以正确解析 flash address，如果没有选择，会有对应的提示，格式选择错误，则擦除方式将不会生效
2. 选择 HEXF 解析的擦除方式
3. 点击 Write 按钮进行擦除和写的操作，注意，这里不需要手动点击 erase 按钮，直接点击 Write 按钮即可



具体的操作流程见上图，log 区域中会有 Start erase... 等擦除操作的提示。

## 3.7.9 DWC 连接

V2.4.2c 版本以后的主要更新功能之一就是支持 DWC 连接并烧写。

PHY6202/PHY6212 产品均是通过 TM(TM=1)拉高，reset 开发板的方式进入 cmd>>: 烧写模式；PHY6222/PHY6220 系列产品进入烧写模式的两种方式：

- (1)TM=1，reset 开发板进入 cmd>>:

TM 拉高，reset 开发板的烧写方式具体操作和使用可以参考 3.2 节。

- (2)TM=0 (TM 引脚拉低或没有 TM 引脚)，在 9600bps 下发送特定的 uart 指令序列进行 DWC 连接，通过 reset 开发板或重上电的方式捕捉 cmd>>:进入烧写模式。

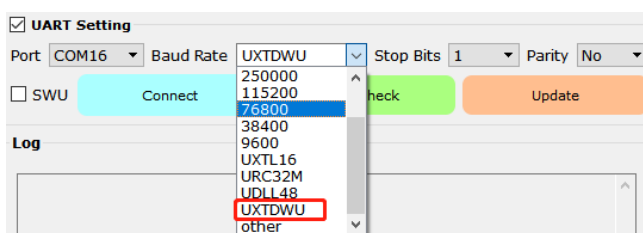
根据发送的 DWC 指令不同，可分为单线（SWU）和双线（DWU）两种连接方式。

- ①DWU(双线连接):

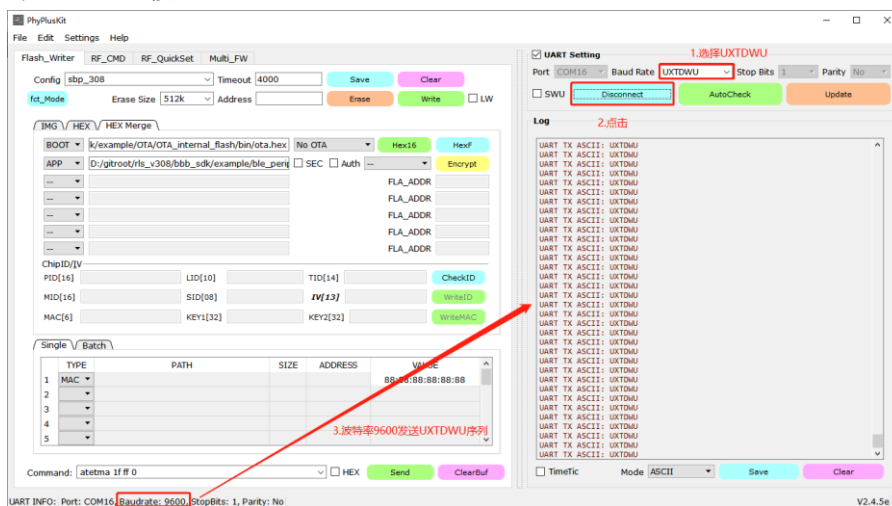
DWU 连接说明：uart 串口在波特率 9600bps 下间隔 50ms 内连接发送 “UXTDWU” 序列，发送过程中，reset 或重上电，芯片捕捉到 cmd>>:即为连接成功并切换波特率至 115200bps，进入烧写模式。

DWU 使用及操作如下：

a. 如图所示，选择对应的“UXTDWU”下拉框

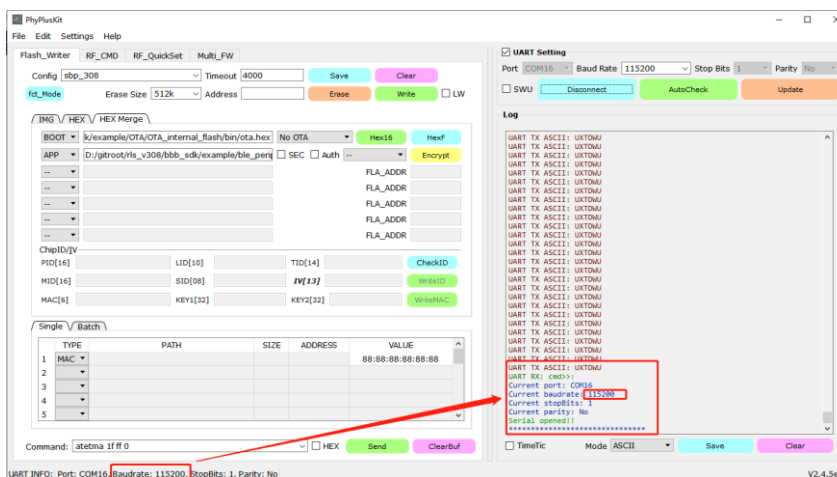


b. 点击 Connect 按钮



点击后，串口就会按需发送对应的 UXTDWU 序列，通过工具下方的 uart info 显示可以看出 dwc 序列是以 9600bps 发送的。

c. reset 或者重上电开发板，芯片捕捉到 cmd>>:即视为 dwc 连接成功



进入烧写模式后，详细的烧写流程操作可参照 3.2 小节。

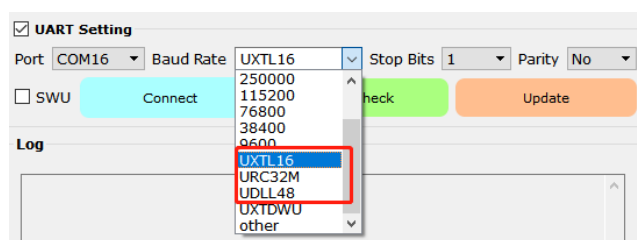
②SWU(单线连接):

SWU 连接说明：uart 串口在波特率 9600bps 下间隔 50ms 内连接发送“UXTL16”、“URC32M”或“UDLL48”序列，发送过程中，reset 或重上电，芯片捕捉到 cmd>>:即为连接成功并切换波特率至 115200bps，进入烧写模式。

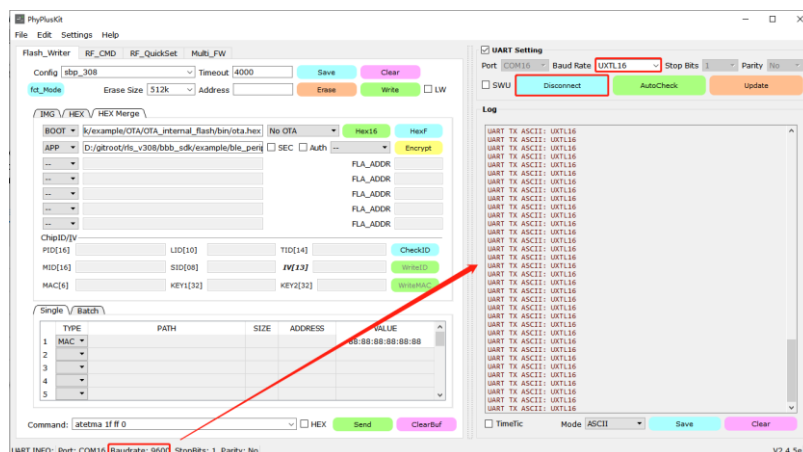
三种连接方式和操作是一样的，不同的是连接后切换的系统时钟（hclk）不同，“UXTL16”方式切换至 hclk:XTAL16M，“URC32M”方式切换至 hclk:RC32M，“UDLL48”方式切换至 hclk:DLL48M。

下面以“UXTL16”为例，详细介绍单线连接及烧写的操作流程。

a. 如图所示，选择对应的“UXTL16”下拉框

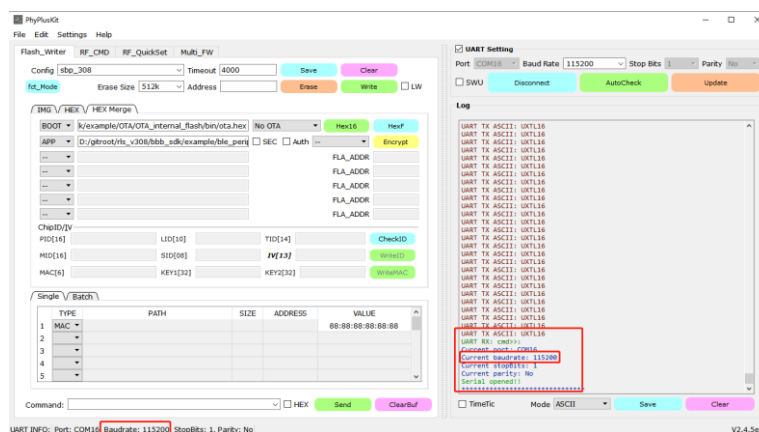


b. 点击 Connect 按钮



点击后，串口就会按需发送对应的 UXTL16 序列，通过工具下方的 uart info 显示可以看出 dwc 序列是以 9600bps 发送的。

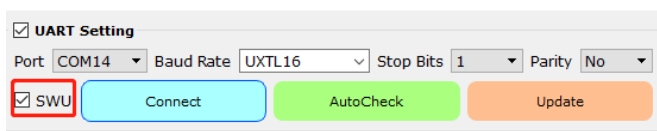
c. reset 或者重上电开发板，芯片捕捉到 cmd>>:即视为 dwc 连接成功



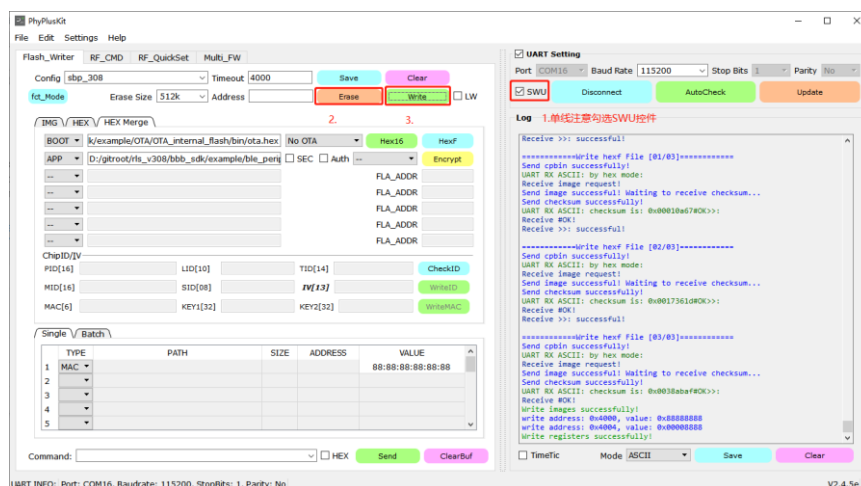
d. 单线烧写功能

单线烧写是以 P10 口来分别完成数据的发送和接收。需要注意单线烧写的地方是，uart 串口发送数据时，除了接收芯片发送的返回值外，还多了串口自身发送后 loop back 的数据，需要进行处理，工具中通过 SWU 控件进行 loop back 数据的处理，管控单线烧写功能数据的正常收发。

**注意：**上位机同时兼容单线和双线烧写模式，通过勾选 SWU 控件进行单双线区分和数据处理，勾选时是单线处理模式，不勾选默认双线烧写，需要注意此勾选控件的选择。

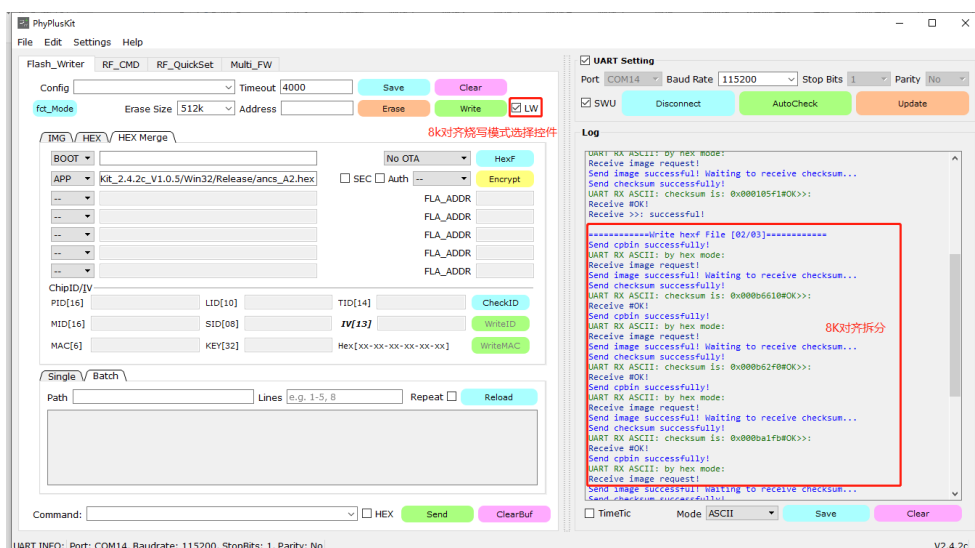


选择烧写的 APP 固件，先擦除再烧写，烧写流程详情可参照 3.2 小节



以上就是 DWC 连接的整个内容。

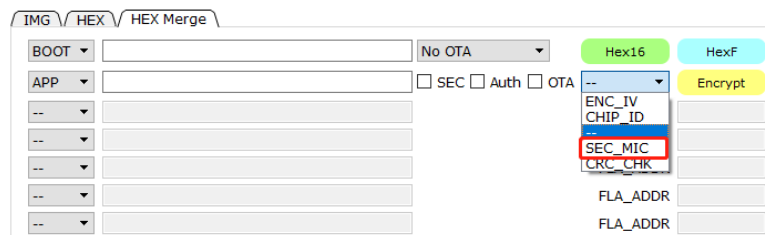
考虑各个 flash 的烧写速度的不同，kit 工具同时支持 8k 对齐烧写模式，通过 LW 控件进行选择，勾选就可以进行 8k 对齐模式进行烧写，具体详见下图：



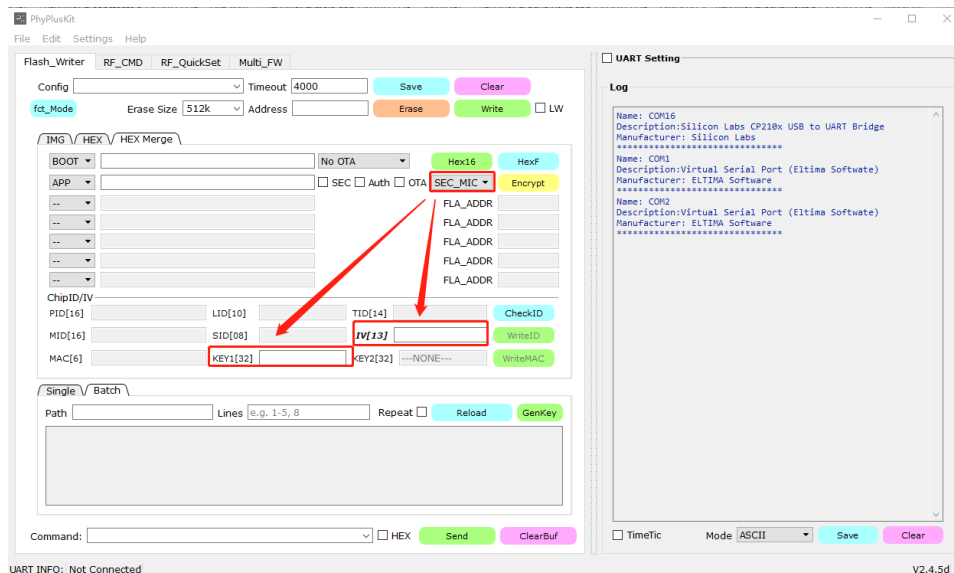
Security boot

## 3.7.10 Security Boot

V2.4.5a 版本之后，PhyPlusKit 更新的另一功能就是 security boot 功能，支持加密 boot 模块功能，根据 aes\_ccm 算法对应用固件的 image partition data 进行加密处理，走安全启动模式。该功能模块主要在下图中选择 SEC\_MIC 模式中进行支持。选择对应的 SEC\_MIC 表单才能走 security boot 功能。具体选择如下图：

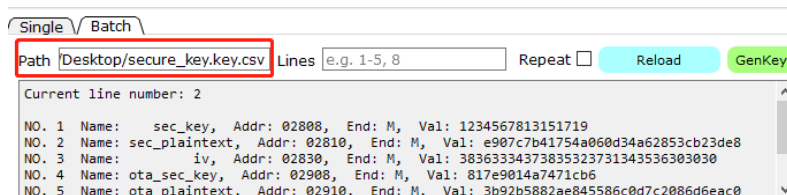


选择 SEC\_MIC 控件模式后，IV 和 Key 编辑框处于可编辑键入状态。

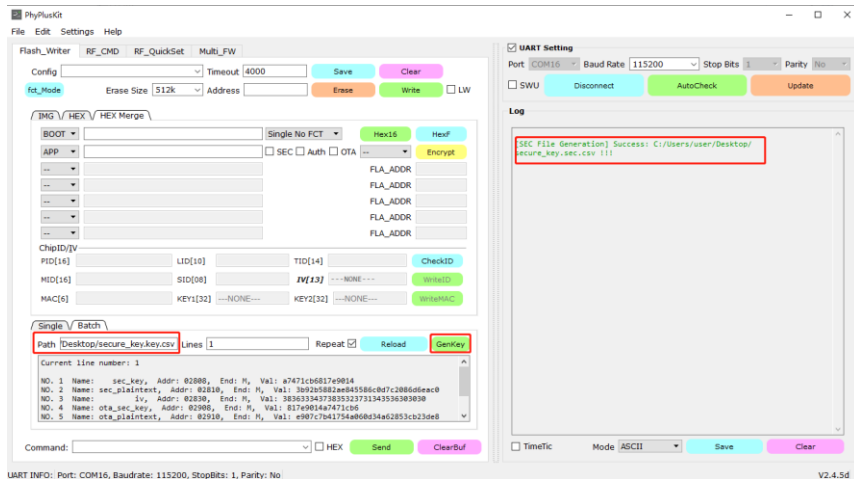


首先需要客户提供一系列 flash key 和 efuse key 组合产生加密用的 g\_sec\_key，即对应键入至图中的 KEY1[32]位置，具体操作如下：

- 1) security boot 加密 g\_sec\_key 的产生方法如下：
  - a. Batch 页面双击加载\*.key.csv 文件(注意要导入.key.csv 文件类型，否则会报错)
  - b. 对应显示所提供的 flash key 等内容，如下图：

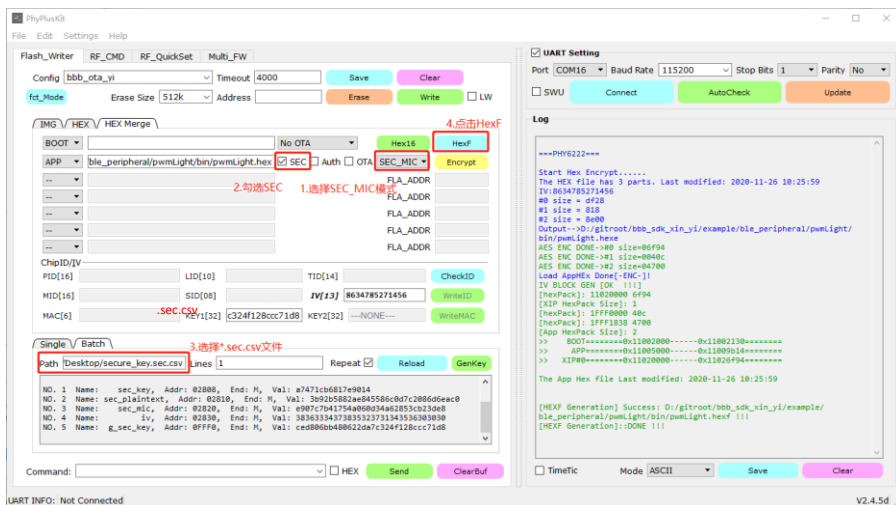


- c. 点击 GenKey 按钮，会对应产生当前显示的 flash key 处理后的\*.sec.csv 文件，可以根据填写的 Lines 值，产生对应行的数据(\*.sec.csv 文件)。（注意只产生一行数据，lines 配置是根据行数配置产生对应选择行的\*.sec.csv）



## 2) Security boot hexf merge 过程

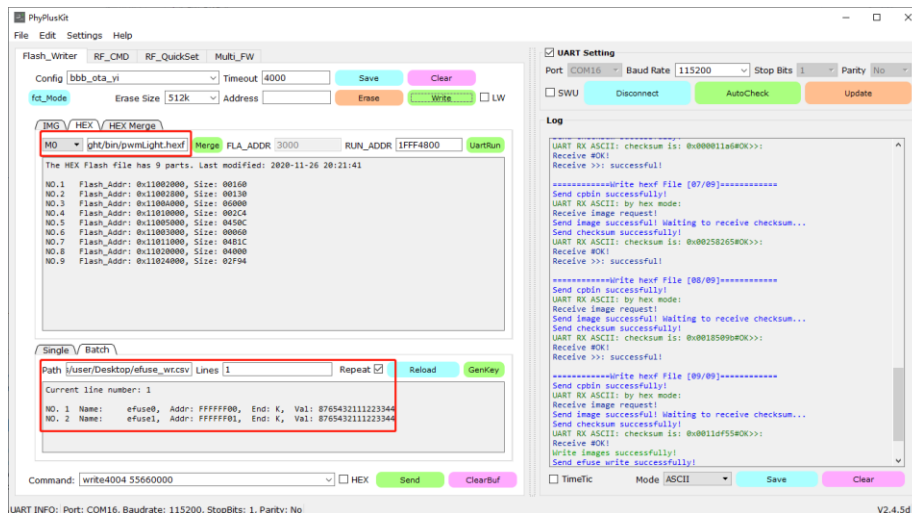
- 选择 SEC\_MIC 模式
- SEC 控件选择 (AUTH 控件仅用于 PHY6220 系列产品, 仅用于 bootloader 中对 image info 进行加解密)
- Batch 页面中双击选择 1) 中 GenKey 产生的 \*.sec.csv 文件
- 点击 HexF 按钮 (注意: 不连接 uart 时根据对应的芯片型号选择后产生; 连接 uart 之后必须处于烧录模式即 cmd>> 模式下点击 HexF 按钮才能正确产生, 详见 3.7.2 介绍)



KEY1[32]和IV[13]中的值是通过解析\*.sec.csv 文件键入的, 不需要手动输入。

## 3) PhyPlusKit 烧写过程

Security boot 烧写除了上述的配置产生的.hexf 密文, 还需要烧写对应产生加密的 g\_sec\_key 密钥对应的 efuse key, 具体操作如下:

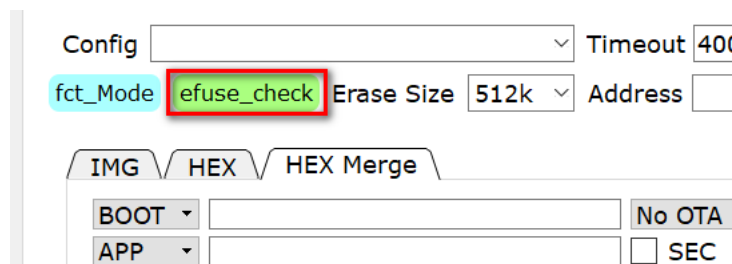


烧写成功后，重新上电，即可完成安全启动步骤

脱机烧录器烧写就需要提供上述 2) 步骤产生的 hexf 文件和对应的 efuse key 的三元组\*.csv 文件即可。

## 3.7.11block check

连接成功后，点击下图按钮，可以查看当前 efuse block 的使用情况以及是否会走加密启动的判断。



按钮

下图为点击按钮后的 log 输出：



判断过程为：1.输出 efuse lock status

2. 依次判断 4 个 block 的状态

3. 读 block 的数据，判断是否被使用（全为 0 代表没有被使用过）
4. 根据 efuse lock status 判断当前 block 的状态
5. 依次判断 efuse0, 1, 2, 3
6. 根据 g\_efuseflg 是否被置起判断当前是否会走 security boot