PhyPlusKit user guide

for PRBMD0x

Introduction

PRBMD0x module is cored with PhyPlus Bluetooth chip, and PHYPlus Kit is a multi-purpose software for PhyPlus chip, including firmware programming, and RF testing. This document is base on PhyPlus Kit user guide in Chinese from PhyPlus, and modified for PRBMD0x module. If there is any variance between PhyPlus Kit user guide and this document, please refer to hyPlus Kit user guide.

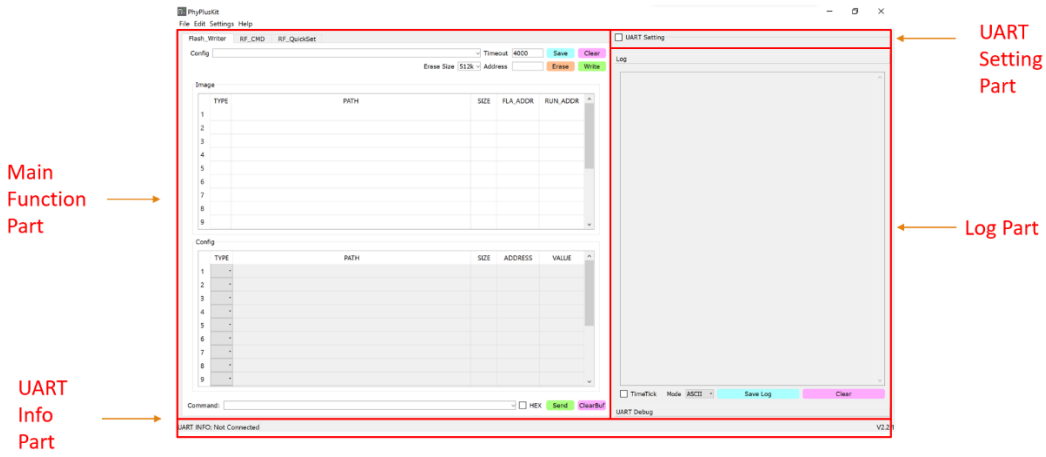
More information and related documents are located at our web site: www.k-sol.com.hk .

Content

1. Functional parts introduction

1.1. Functional parts

This software consists of four functional parts, the Main Function area, the serial port setting area, the log display area and the serial port information area. In addition, the software supports the command line mode, in the command line mode, you can directly execute some commands by adding relevant parameters to call. See Sections 2.6 and 3.6 for a description of the command-line mode.



⑴ Main Function area consists 4 parts: Flash_Writer、RF_CMD、RF_QuickSet.

- Flash_Writer: program image document；
- RF_CMD: send out HCI protocol command；
- RF_QuickSet: send out pre-set test comand.

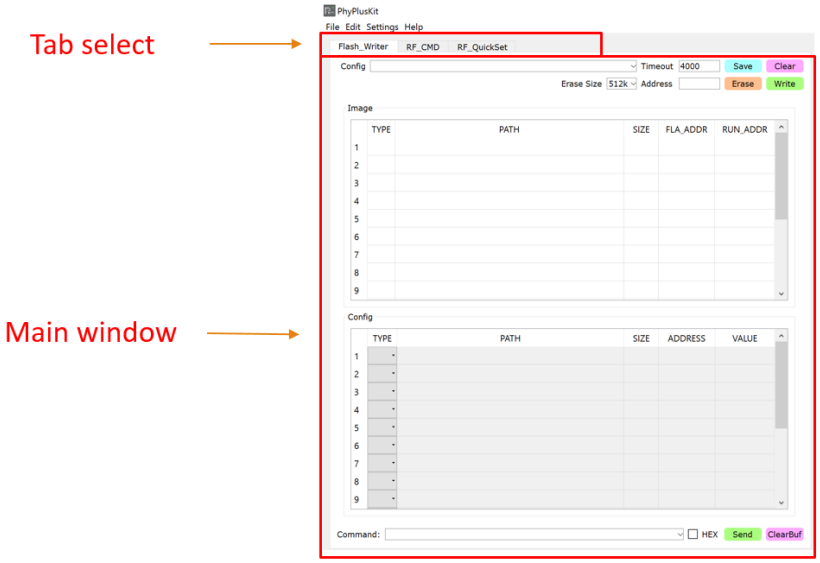
⑵ UART Setting part: set UART configuration and connection；

⑶ LOG part: display and print out user operation and communication;

⑷ UART Info part: Real time display UART configuration.

1.2. Functional part description

1.2.1. Main Function part



User can switch function by clicking different tab select.

1.2.2. UART configuration

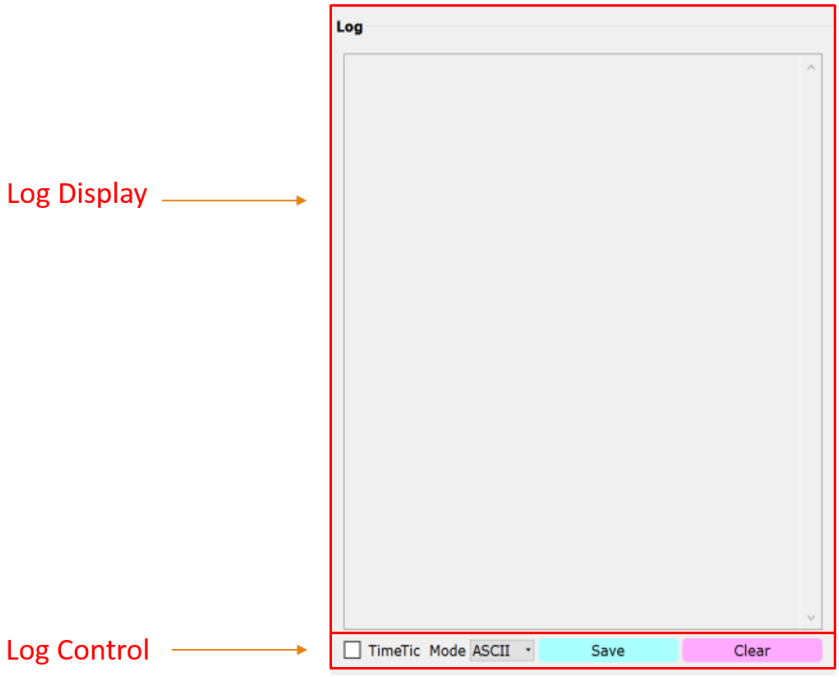
Hide UART setting

Show UART setting

In the UART port setting area, the setting parameters can be hidden or displayed, and the connection/disconnection operation can be performed.

- Serial connection with arbitrary baud rate, stop bit and parity bit；
- It can automatically detect the communication baud rate set by the lower computer；
- The baud rate can be changed to match with the device it is connected；

1.2.3. Log Display

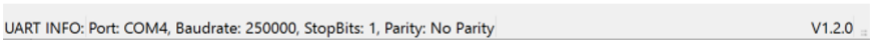


- Log Display: Print out user operation andUART port communication;
- Log Control: You can enable parsing support for different RX content, as well as timestamps, or directly save log content to any location;

1.2.4. UART info part

Not Connect UART

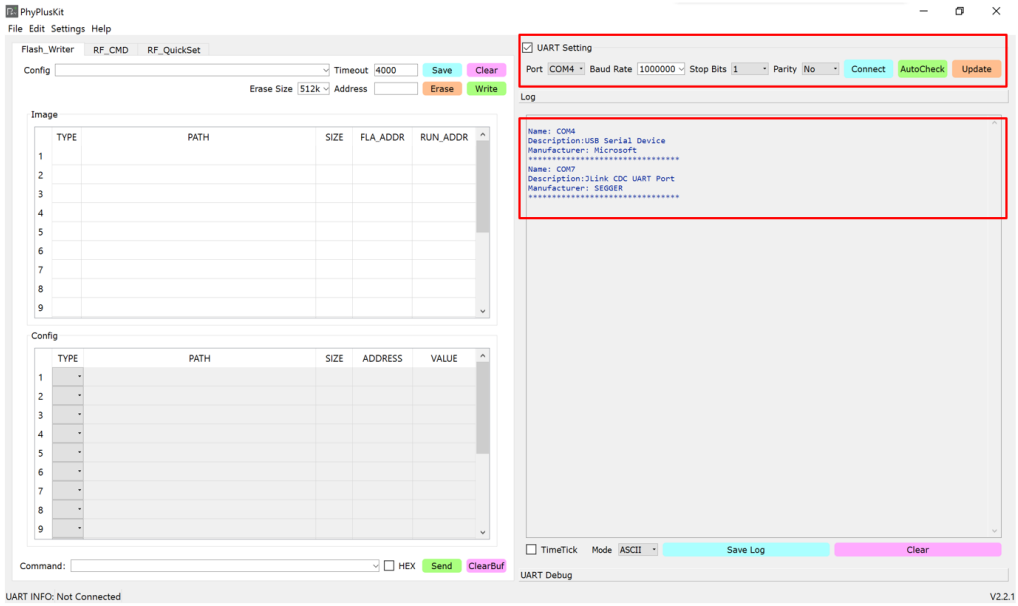
Connect UART



UART setting information and the current version number are displayed.

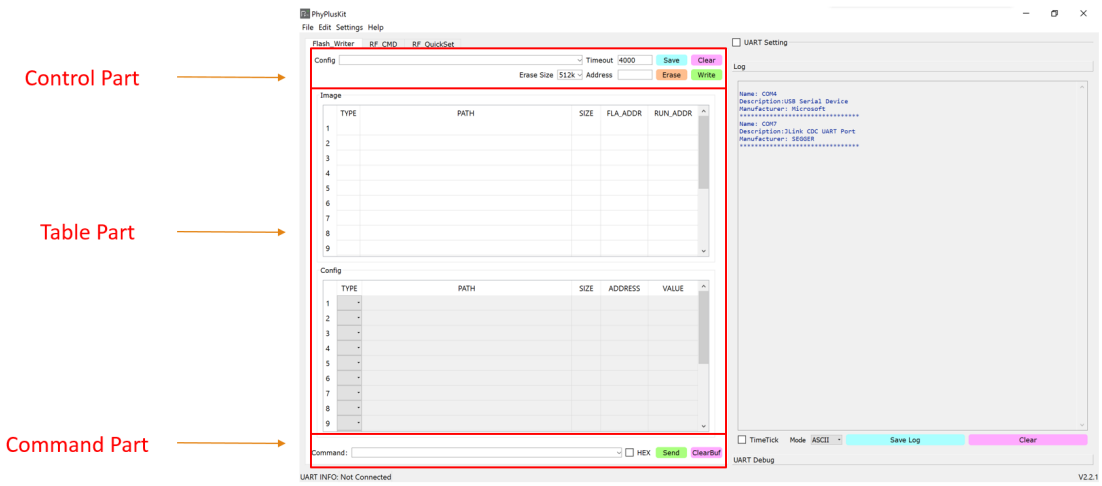
2. Feature description

2.1. UART connection and setting



- Click the Port drop-down box to get all available serial port names and output the information of the available serial ports in the Log;
- Able to set baud rate, stop bit and parity bit;
- Click the Connect button to connect the selected serial port according to the custom settings, and display the current serial port setting parameter information in UART Info;
- Click the AutoCheck button to automatically send commands to detect the communication baud rate of the lower computer;
- Click the Update button to send the baud rate modification command to the lower computer, and modify the software communication baud rate at the same time

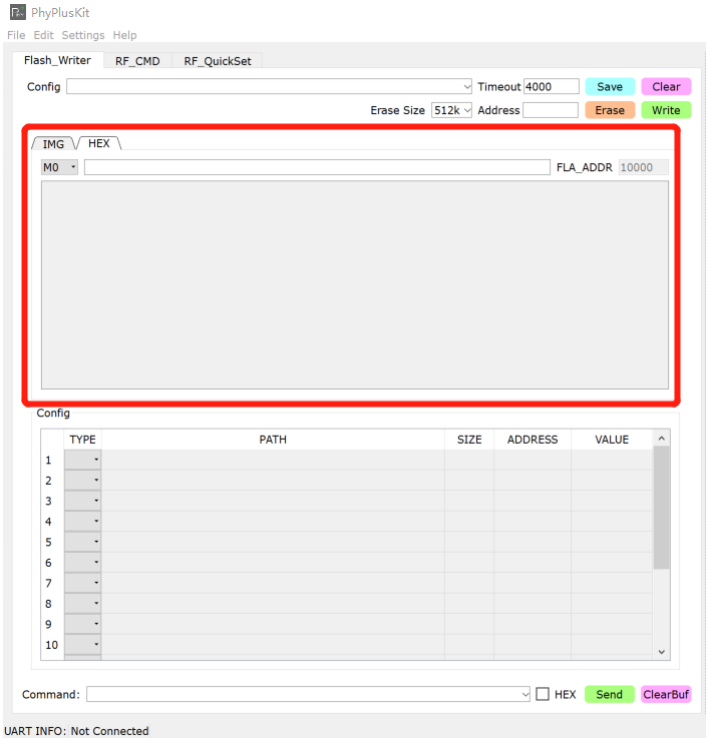
2.2. Flash Programming



1. Table （Table Part）

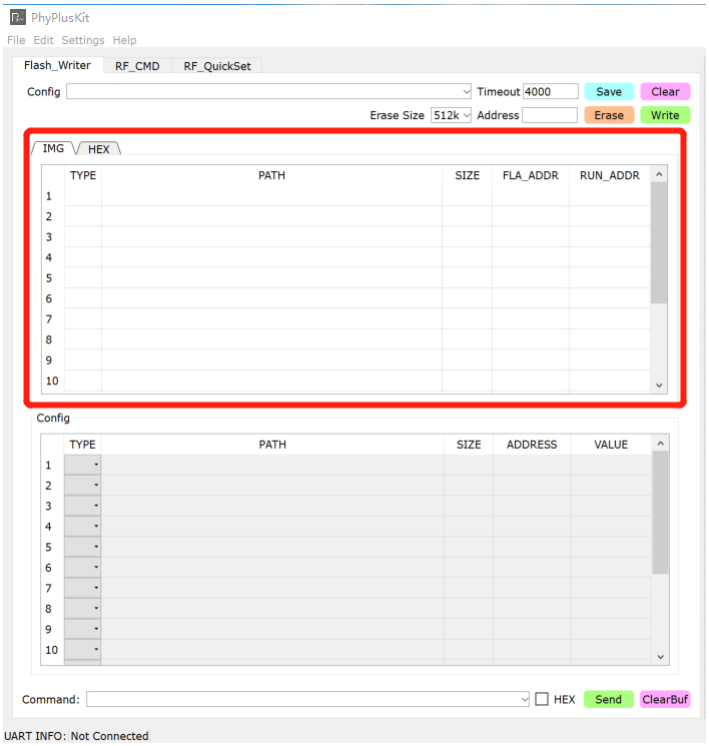
（1）image / HEX table

HEX: (Recommend to program with HEX file)



- Double-click the area to select the Hex file in the input box, the data in the HEX will be automatically analysed, and the Flash Address to be programmed will be automatically generated according to the starting Start Flash Address set by the user;
- Start Flash Address can be set up at:Settings -> Configuration -> Flash Writer.
- The Run Address of the main program must exist in the Hex file, the default is 0x1FFF4000, the user can set it in Settings -> Configuration -> Flash Writer ；

Image:



- In the Image, you can double-click the cell in the PATH column of the current row to add the bin file to be programmed. ;
- Fill in the FLA_ADDR, RUN_ADDR to be programmed in the corresponding line

(2) config table

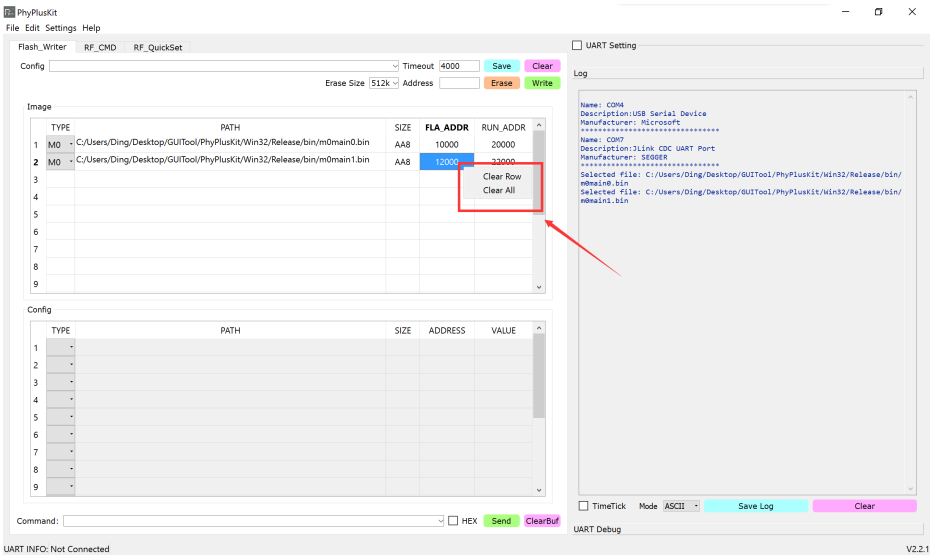
- In Config area, click the drop-down box corresponding to the TYPE column of the current row to select AT or MT mode. The corresponding functions are as follows:

Type	Feature	Programmable row
AT	Read from the file to write the value of the FLASH register	PATH, SIZE, ADDRESS
MT	Read the value of the register to be written from cell VALUE	ADDRESS, VALUE

- If the AT automatic mode is selected, double-click the corresponding PATH cell to select the file storing the register value. Fill in the register start address in the ADDRESS cell.
  If the MT manual mode is selected, fill in the starting address of the register to be written in the ADDRESS cell, and fill in the VALUE cell, the value to be written to the corresponding register.

  The starting address of the register write value is the address filled in by ADDRESS. If the file has multiple lines of register values, the corresponding starting address is +4 successively, and the value of 4bytes is written each time.

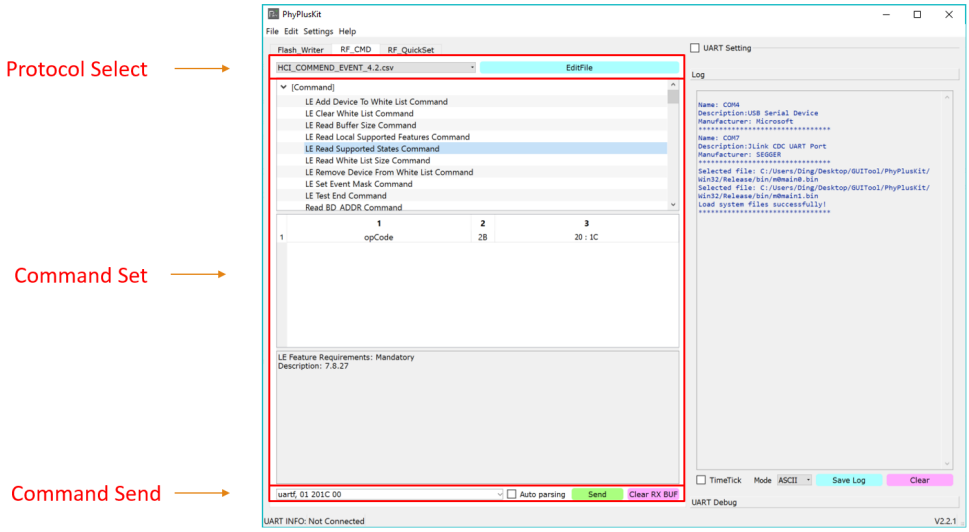
Note: Right-click in the table to clear the row content, or clear the entire table content



2. Control Part
   - Before writing the bin file, you need to erase it first; you can click the Erase button to erase it; after erasing is successful, "erase #ok" will be output in the log;
   - After the erasing is successful, the programming operation can be performed. Click the Write button to program the currently selected bin file to the corresponding address; the default interactive waiting time is 4000ms, and the user can modify the waiting time in Timeout according to the usage;
   - The current bin file configuration information and Timeout settings can be saved; enter the file name to be saved in config, and click the Save button to save the current settings;
   - You can see the previously saved configuration information in the config drop-down list; select it to load.
   - Click Clear to clear the current table content;

3. Command Part
   - Any command can be sent in Command, click Send to send the command;
   - The ClearBuf button is used to clear the interactive buffer. When there is an error in the serial port reception, the buffer can be cleared and the command can be sent or programmed again;
   - In the Command drop-down list, the last command sent will be saved.

2.3. RF Command



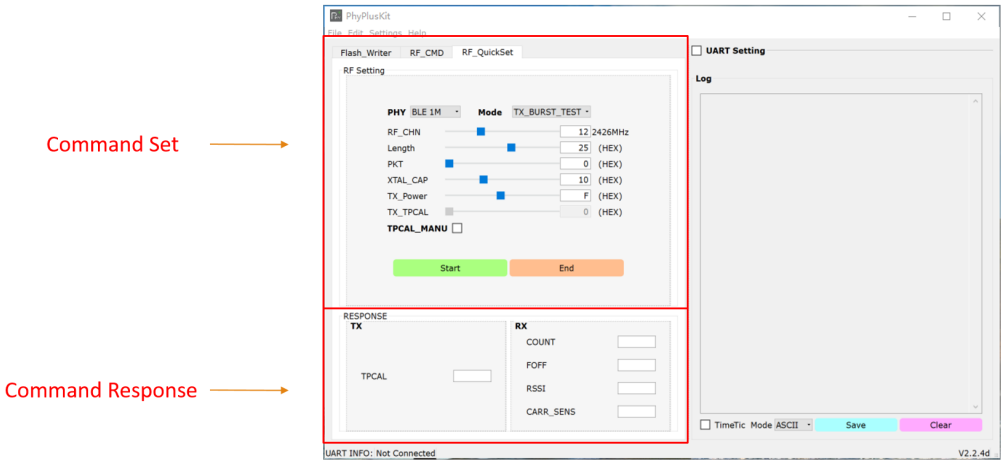
1. Protocol Select
   - In the drop-down list, different versions of the protocol can be selected to load

2. Command Set
   - Select the type of command (Command, Event, etc.) to be sent in the tree list above; and the specific command;
   - In the middle table, the command field can be configured;
   - The lower part display the details

3. Command Send
   - After configuring the parameters in the command settings, you can see the combined command content in Command Send, click Send to send the command;
   - In the Command input box, you can enter custom command content; the command format is as follows:

Command	Feature
uarta, “abcd” (abcd can be arbitrary ASCII character)	send ASCII commands;
uarth, 01 02 (01 02 can be arbitrary Hex code)	Send HEX command;
uartf, 01 0C6C	Send auto-combined commands;

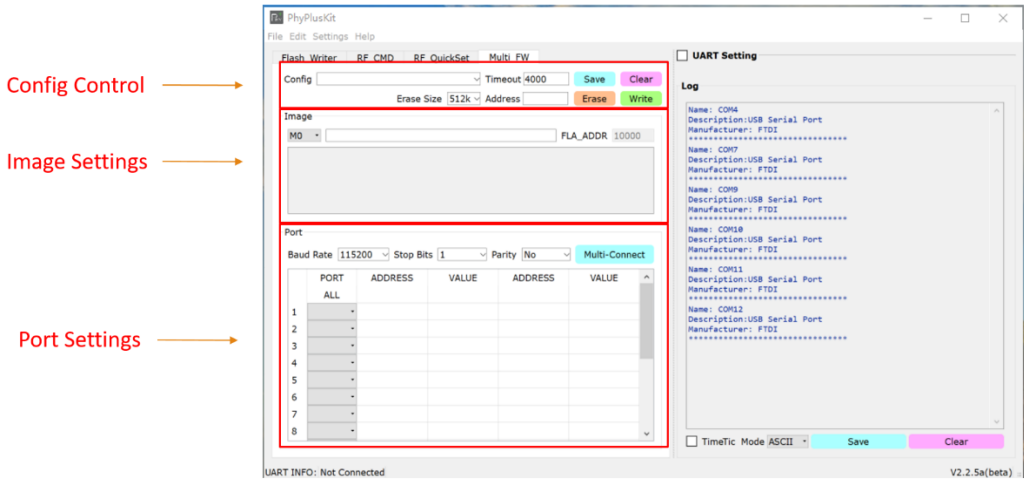
2.4. RF_QuickSet quick command



1. Command Set
   - In RF Setting, you can select the corresponding PHY hardware device mode (BLE 1M, BLE 2M, BLE 500K, BLE 125K, ZigBee), and the corresponding command mode: TX (BURST_TEST, Single Tone, Modulation), RX (BURST_TEST, AUTO)), set the command parameters Frequency, Length, PKT, TX_Power, TX_TPCAL (calibration value);
   - Click the Start button to send corresponding commands in sequence;
   - Click the End button to get the corresponding corresponding results TX (TPCAL), RX (COUNT, FOFF, RSSI, CARR_SENS);

   Note：
   - In BLE mode, Frequency is 2402 ~ 2482MHz, step 2MHz, divided into 0 ~ 40 channels; In ZigBee mode, Frequency is 2405 ~ 2485MHz, step 5MHz, divided into 0 ~ 16 channels;
   - The default TPCAL is obtained automatically. If you want to manually modify the TPCAL value, you can check the TPCAL_MANU selection box to modify the TX_TPCAL value;
2. Command Response
   - In RESPONSE, the result obtained by the command can be displayed;

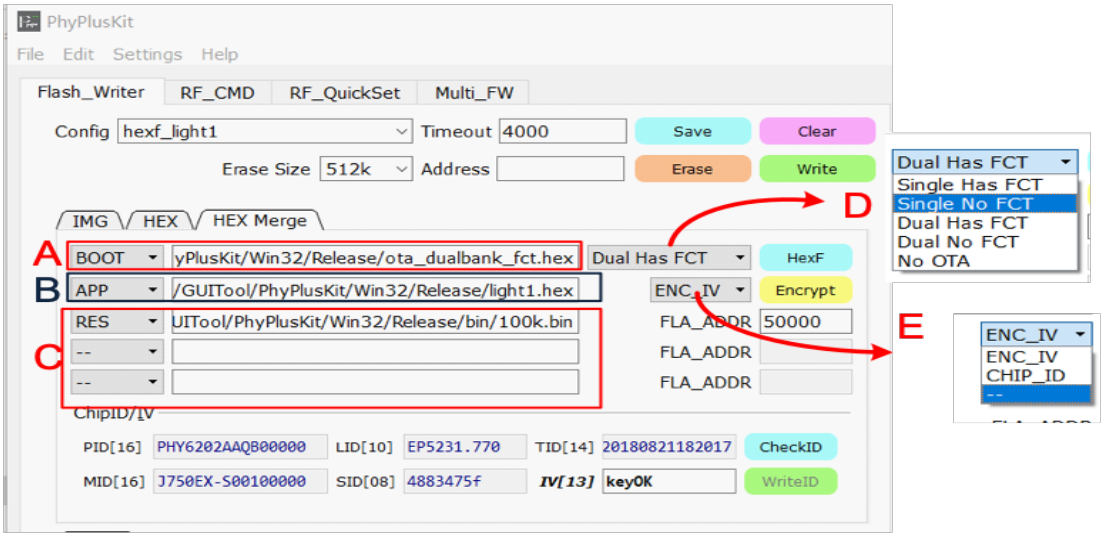
2.5. Multi_FlashWriter multi-UART programming



1. Config Control
   - Here you can select and load the saved related configuration files, the file content includes (timeout value interacting with the chip, HEX file path to be programmed, UART PORT parameters, Flash value to be written).
   - Here you can configure the size of the Flash to be erased, and the corresponding address of the Flash to be erased (the default is 512K, which erases all the Flash content).
   - Save button to save configuration, clear to clear table content, Erase to erase Flash, write to program Flash.
2. Image Settings
   - Double-click to select the Hex file in the input box, the data in the HEX will be automatically analyzed, and the Flash Address to be programmed will be automatically generated according to the starting Start Flash Address set by the user;
   - The starting Start Flash Address can be set in Settings -> Configuration -> Flash Writer.
   - The Run Address of the main program must exist in the Hex file, the default is 0x1FFF4000, the user can set it in Settings -> Configuration -> Flash Writer;

3. Port Settings
   - Configure the PORT connection here, select the appropriate baud rate, stop bit, parity bit, and select the PORT port to be connected in the drop-down box in the table;
     The flash address programming of the corresponding row of ALL in the table will take effect on the PORT selected in each row below.
     The flash address programming of the corresponding row of each PORT is valid only for the current row (PORT);
   - After setting the PORT connection information, click Multi-Connect to connect

2.6. Flash_Writer HEX Merge



1. HEX Merge
    - A: Used to input the ota_boot.hex file, you need to select a different OTA_BOOT mode through the D selection box. A total of 5 modes are currently supported. Different modes of OTA_BOOT have different mappings to flash addresses. Refer to the table below for flash mapping
    - B: It is used to input the hex file of the APP program, and different encryption methods can be selected through the E selection box.
        - ➢ The E:[ENC_IV] mode is to encrypt and protect the app file using the Identify Vector input by IV.
        - ➢ The E:[CHIP_ID] mode is to encrypt the app file with the unique chip's chip id.IV=function（chipid.TID,chipid.SID）
        - ➢ E:[--]mode, no encryption mode is used.
    - C:Used to input resource files, currently supports binary format and hex format. You can enter the flash storage address of the resource file in FLA_ADDR.
    - [HexF] button is used to generate a *.hexf file with one key, which is a combination of multiple hex files of A, B, and C, which can be used for direct burning. The output path of the file is the app file directory.
    - [Encrypt] button is used to generate the encrypted file *.hexe of the app file, and the output path of the file is the app file directory.

2. ChipID/IV
    - CheckID button is used to detect the factory ID of the current chip and needs to be connected to the UART. If the detection result is not [EMPTY], the WriteID button will be activated
    - For the chip without factory ID, it can be burned through the WriteID button. Need to fill in the corresponding PID, LID, MID, TID, SID
    - IV：Identify Vector to be used for the input file, 13Byte. If the encryption method in the figure is selected as CHIP_ID, this part will automatically generate the IV from the CHIP ID.

Table 1 flash mapping

	512k version (Dual bank) (Has FCT)		512k version (Single bank) (Has FCT)		512k version (Dual bank) (No FCT)		512k version (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
2nd Boot info	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. Command Line mode

In command line mode, you can run this program by followed by the specified parameters to burn the specified hex (or hexf) file, or combine several specified files into a hexf file (refer to Section 2.6 for related content). In addition, you can also choose to perform both merging and programming (the actual execution order is merging first and then programming). If you open the program without parameters on the command line, it will directly enter the GUI mode.

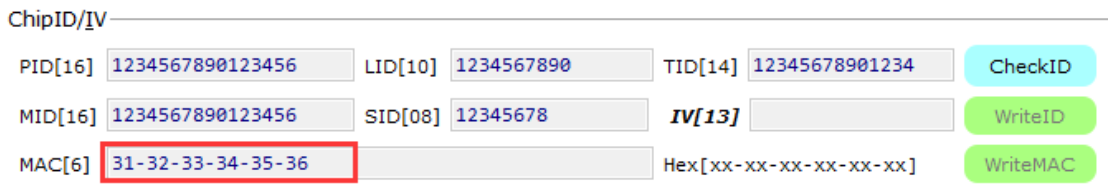
The command line parameters supported by this program are as follows (note: the parameters except -a should be followed by **-r** (see the description of the **-a** parameter for details), other parameters have no order in theory, and each parameter and its The abbreviated forms are completely equivalent and can be interchanged arbitrarily):

parameter	abbr.	value	Description
--combine	-c	NIL	Used to instruct the program to merge hex files to generate a hexf file. This parameter does not need to follow a value. The generated hexf file is determined according to the file name of the app by default, that is, if the app file is app.hex, the merged hexf file and the app are in the same directory and the file name is app.hexf. At least one of this option and the -w option must exist (can exist at the same time)
--boot	-b	OTA Bootloader document hex format	Only one OTA Bootloader program can be included. If it is No OTA mode, this parameter is not a necessary parameter. In other cases, this parameter is a necessary parameter when merging the hexf file. The parameter is a string, which can include a path. If the path has spaces, the entire string needs to be Enclose in double quotes and separate options with spaces Example :—boot ota.hex 以及 -b "c:\data app\ota.hex"
--app	-p	App document hex format	It can only contain one APP firmware program. This parameter is a necessary parameter. The parameter is a string and can include a path. If the path has spaces, you need to wrap the entire string in double quotation marks, and separate it from the options with a space symbol. Example: -p app.hex
--res	-r	Resource document bin/hex format	Can contain more than one resource file (but at most 3), this parameter is not necessary The parameter parameter is a string and can contain a path. If the path has spaces, the entire string needs to be enclosed in double quotes, and the options are separated by a space symbol. If it is a bin format file, you need to specify its starting address (hexadecimal) with the -a option after it. If it is a hex format file, it is not required (if the bin file is not followed by a starting address, it will report an error and exit) Example: -r res.bin -a 70000（write start from address 0x70000）or -r res.hex

--addr	-a	Starting address of bin format Resource document	The starting address is in hexadecimal format, the value is the offset address of Flash starting from 0, and the data size is the file size. This program will check the data length and the validity of the address. If it is an illegal address or the address overlaps, it will be Prompt error, see above for example (The hex file has its own address information, so this option is not needed to set its starting address)
--mode	-m	Mix mode	Mix mode needs to be one of the following five:SH (Single Has FCT), SN (Single No FCT), DH (Dual Has FCT), DN (Dual No FCT), NO (No OTA), if it is any other value, an error will be reported, this parameter is a necessary parameter Example：-m DH
--enc	-e	encryption mode	This parameter is not a necessary parameter. If there is no such parameter, the default is no encryption. If there is this parameter, its value must be one of the following two cases: chip (indicates encryption with chip id) or iv_xx…xx (xx…xx is 13 The iv value of the bit, if it is insufficient, add 0 to the right end, and if it exceeds, truncate the left end 13 bits as the iv value), if it is not in these two cases, an error will be reported Example: -e chip or -e iv_1234567890123
--write	-w	Write a document hex/hexf format	At least one of this option and the -c option must exist (can exist at the same time). If they exist at the same time, execute the -c option first to merge and then execute this option to write, that is, you can use these two options at the same time to write parameters after merging. It is a string and can contain a path. If the path has spaces, you need to wrap the entire string in double quotes, and separate it from the options with a space symbol Example: -w target.hexf **Note: Before writing starts, the program will perform an erase operation by default!**
--uarts	-u	update baud rate	Parameter is not a must, default baud rate is 115200 if no parameter. Available baud rate are: 1500000,1000000, 500000, 250000, 115200, 76800, 38400 and 9600. It is possible to adjusted base on UART configuration （v2.3.8c currently only supports PHY6212 programming at baud rates of 1500000 and 1000000） Example:-u 500000

--Run	-R	Base run address（1FFF4000--PHY6202，1FFF4800--PHY6212）	Parameter is NEEDED for programming firmware into different chip type, starting programming address of bin file (inside hex file) can be adjust through modifying **configuration—base run address** Example -R 1FFF4000（corresponding to 6202 chip），-R 1FFF4800（corresponding to 6212 chip）
--Port	-P	Obtain port name	Connect multiple UART to the host, obtain the names of the COM ports of multiple UART, and program the firmware for the development board of the specified COM port Example: -P COM3
--config	-f	Configuration document. csv	This parameter is not a necessary parameter, it is mainly used to program the MAC address, set multiple lines of 12-bit mac address in the csv file, and program the mac address value through the 4000 and 4004 addresses (Note: If there are other configuration requirements, you can also do it yourself Add, get the corresponding write address address and value) Example: -f *.csv
--line	-l	line value setting of configuration document	Parameter is not a MUST. Configuration file allows multiple lines of information. MAC address and other parameter can be modified by changing the LINE value in every programming. Example: -l 3（note: this is capital letter i）
--help	-h or -?	NIL	display play

2.8. MAC address



Below the ChipID/IV column, there is a MAC address column marked with a red frame in the figure above. This column can view the current MAC address of the device, and can also write a new MAC address (only when the original MAC address is empty) .

If you want to check the current MAC address, you need to click CheckID in the above picture after connecting. If the MAC address has been set, the MAC address bar will display the current MAC address. If the MAC address has not been set, the MAC address The column becomes editable and the WriteMAC button on the right becomes available.

If you want to set a new MAC address for a device whose MAC address is empty, you need

to click CheckID to check it first. When it prompts that the MAC address is empty, you can fill in the new MAC address in the MAC address field that has become editable (format xx-xx-xx-xx-xx-xx, and it is required to be in hexadecimal, otherwise the writing will fail) and then click WriteMAC to write the new MAC address.

3. Feature example

3.1. Obtain UART baud and update

3.1.1. Update UART baud

1. selected UART baud for the connected device（i.e.: 250000）；
2. Click Update button to send out the command；
3. Change success display in LOG；

3.1.2.Obtain UART baud

1. Click AutoCheck button, it will auto detect the baud of connected device;
2. After “UART RX ASCII: #OK>> ” is shown, then the current BAUD will be displayed.

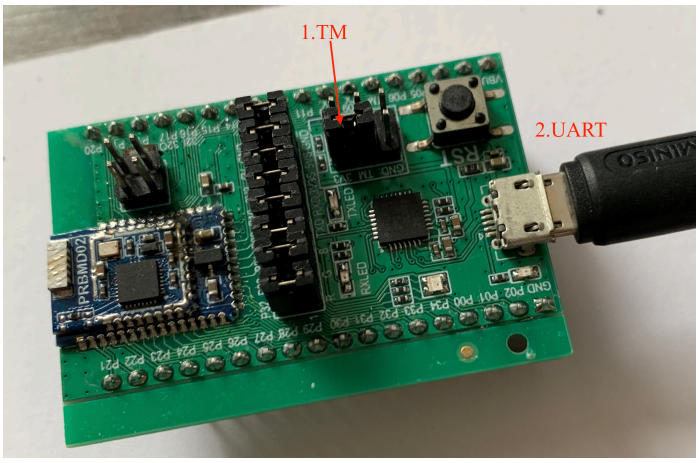
3.2. Flash programming

3.2.1.Hex only programming (recommended)

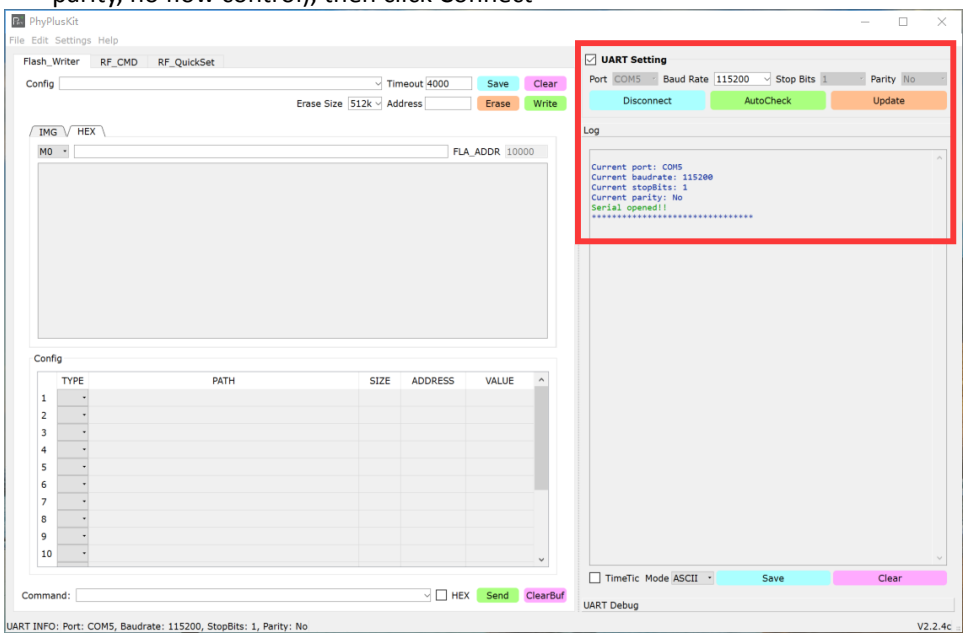
Use UART for flash programming operation, and power on (or press reset) after TM pin is pulled high, which is the state of UART receiving command, and UART is configured as baud rate: 115200，8bit, 1 bit stop, None parity, no flow control;

Steps:

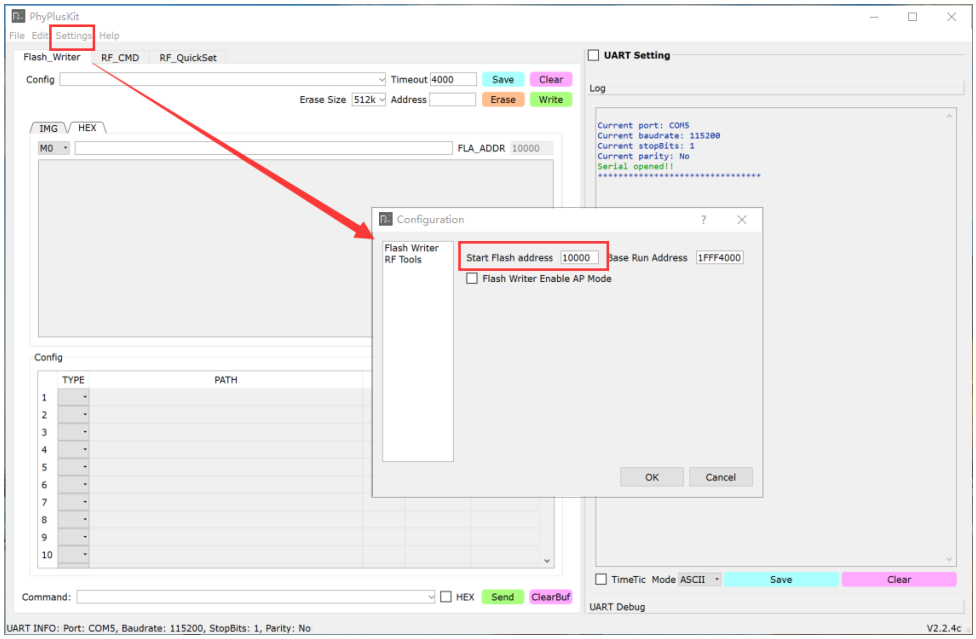
1. Prepare software and tools, connect hardware, pull TM (pin8) high, as shown below:



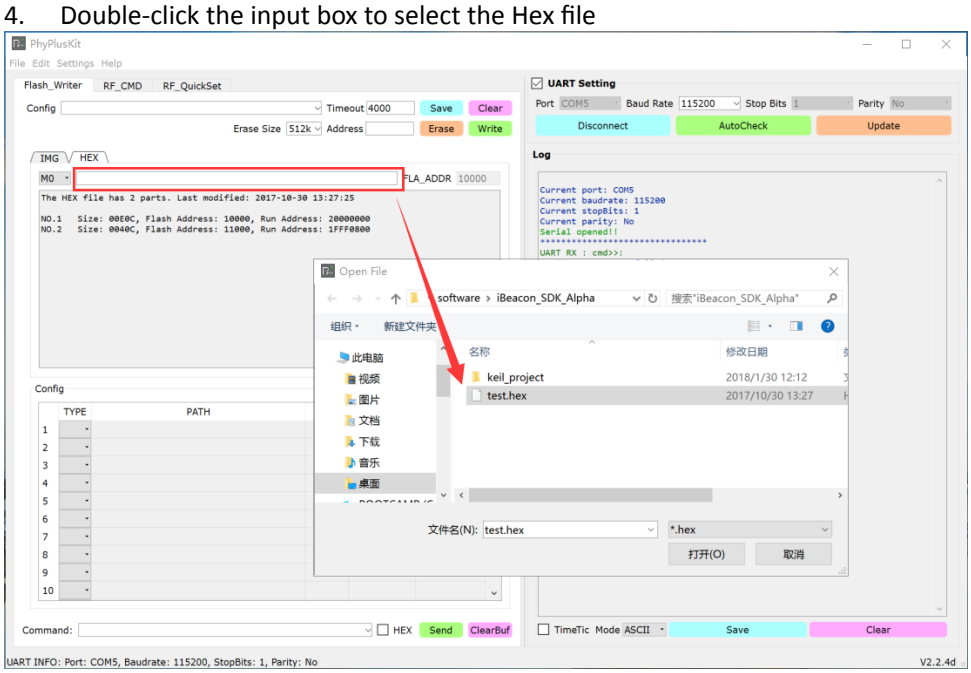
2. Run PhyPlusKit.exe, configure parameters in UART Tab (115200, 8bit, 1 bit stop, None parity, no flow control), then click Connect



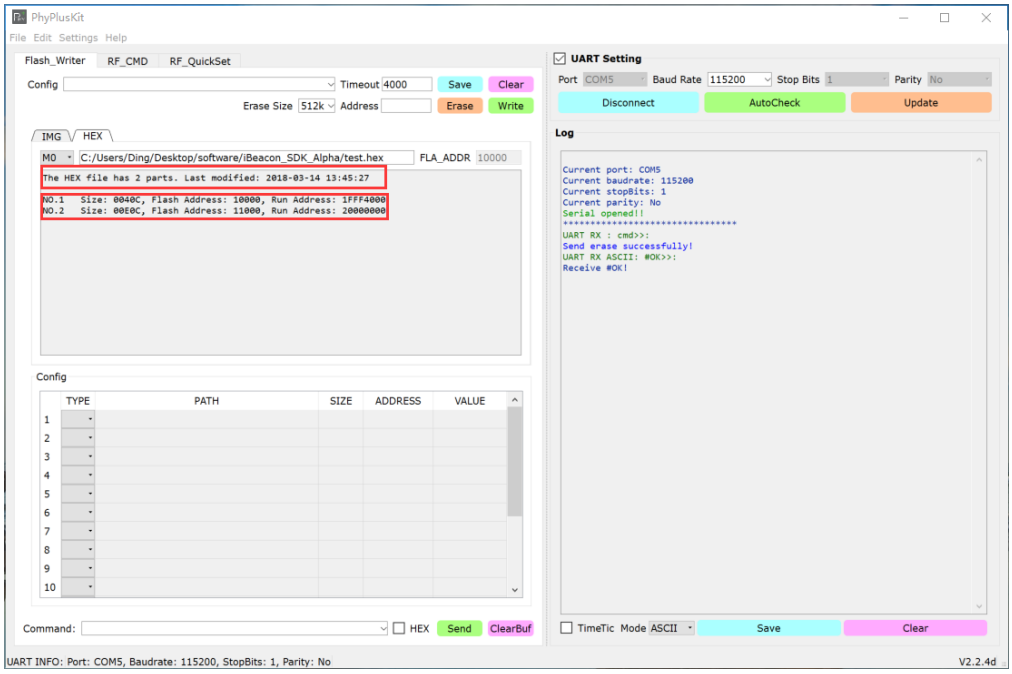
3. 设置起始Configure the Start Flash Address



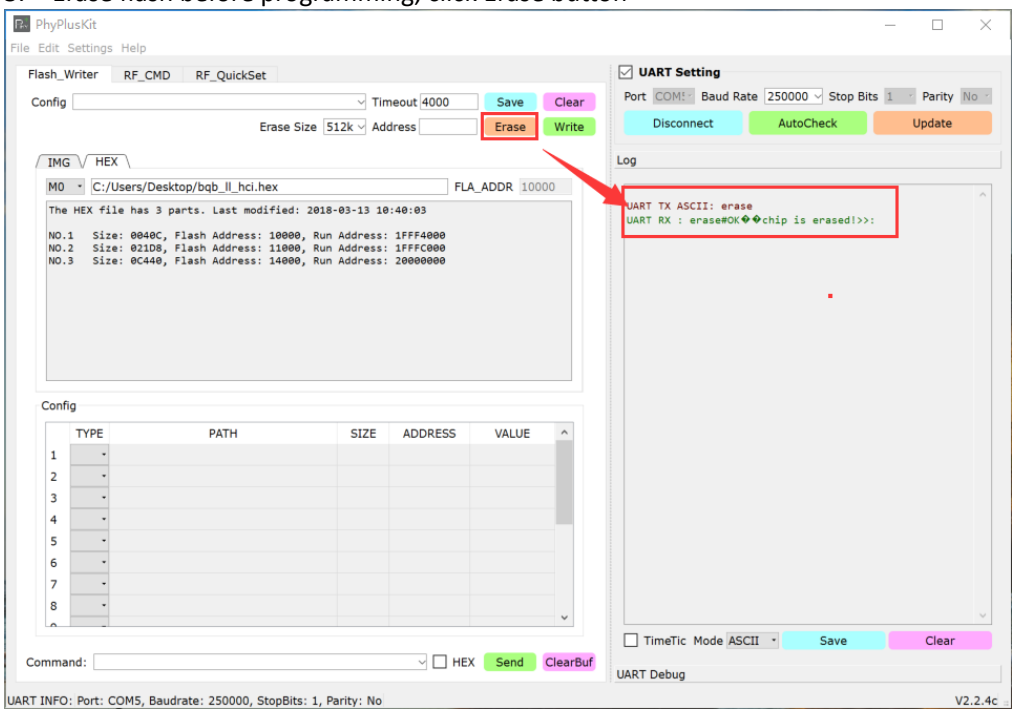
4. Double-click the input box to select the Hex file



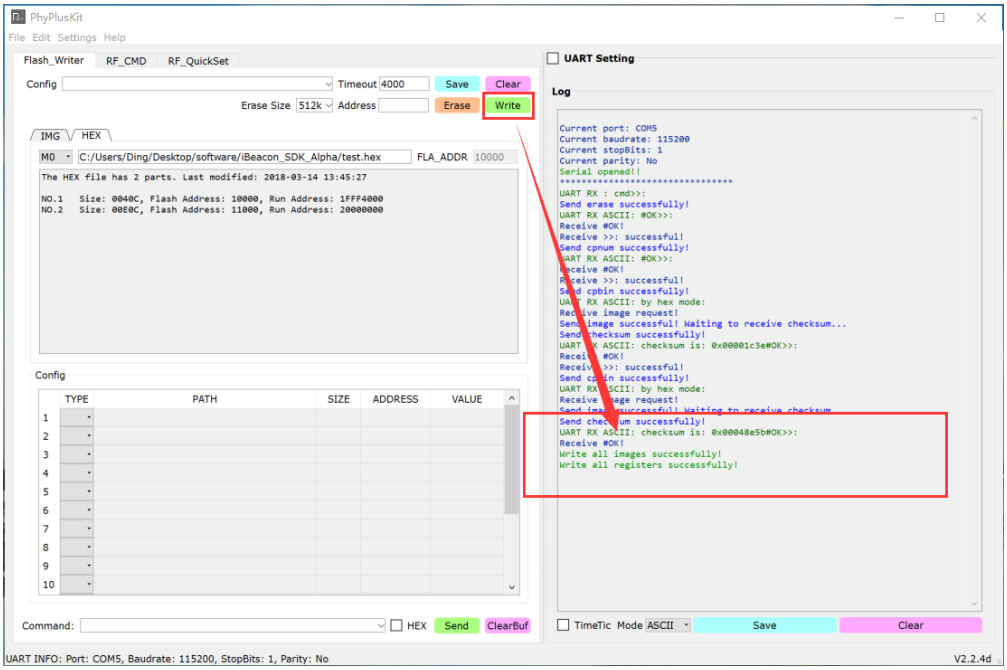
The program automatically parses the data files in the HEX file and displays the last modification time



5. Erase flash before programming, click Erase button



6. Click Write button to start programming of the HEX file



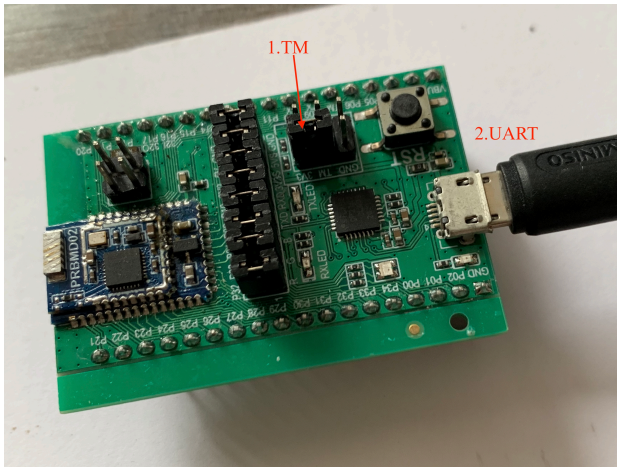
7. After programming successfully, pull-down TM pin, followed with reset, module will go to boot mode.

3.2.2. Image only programming

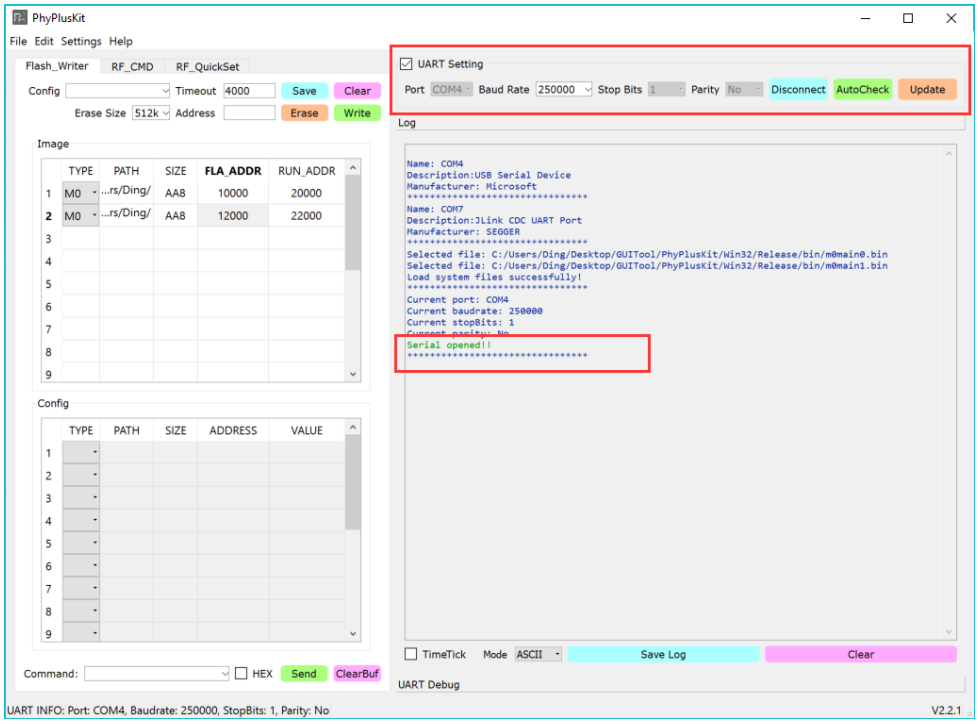
Use UART for flash programming operation, and power on (or press reset) after TM pin is pulled high, which is the state of UART receiving command, and UART is configured as baud rate: 115200，8bit, 1 bit stop, None parity, no flow control;

Steps:

1. Prepare software and tools, connect hardware, pull TM (pin8) high, as shown below:

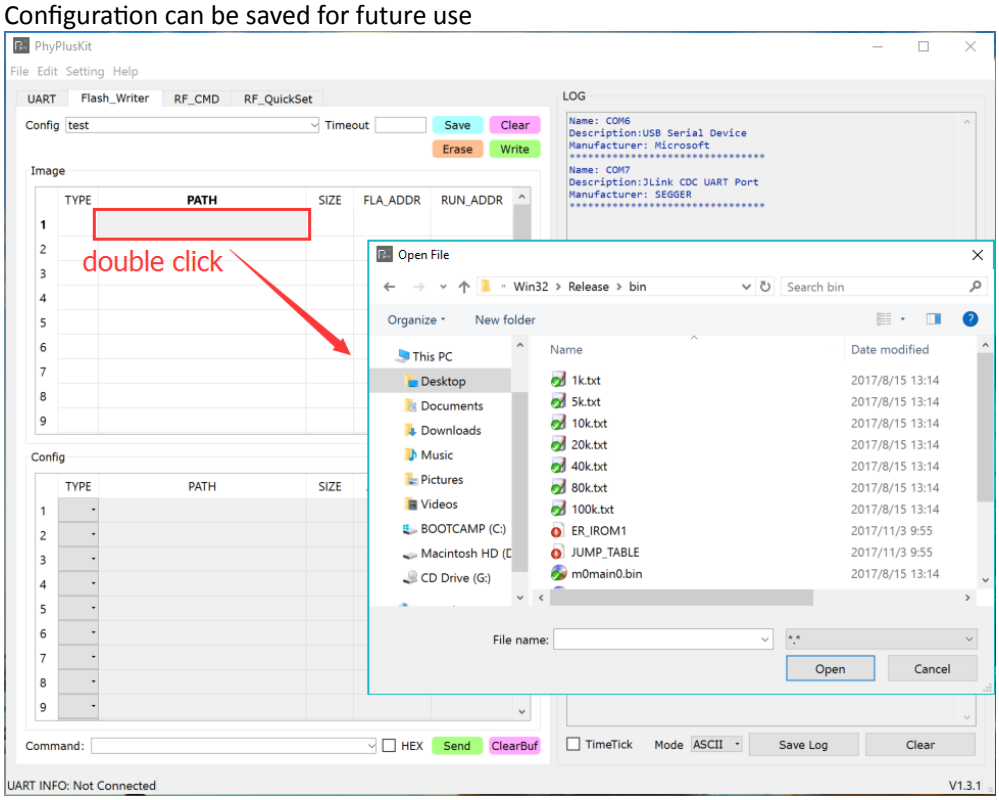


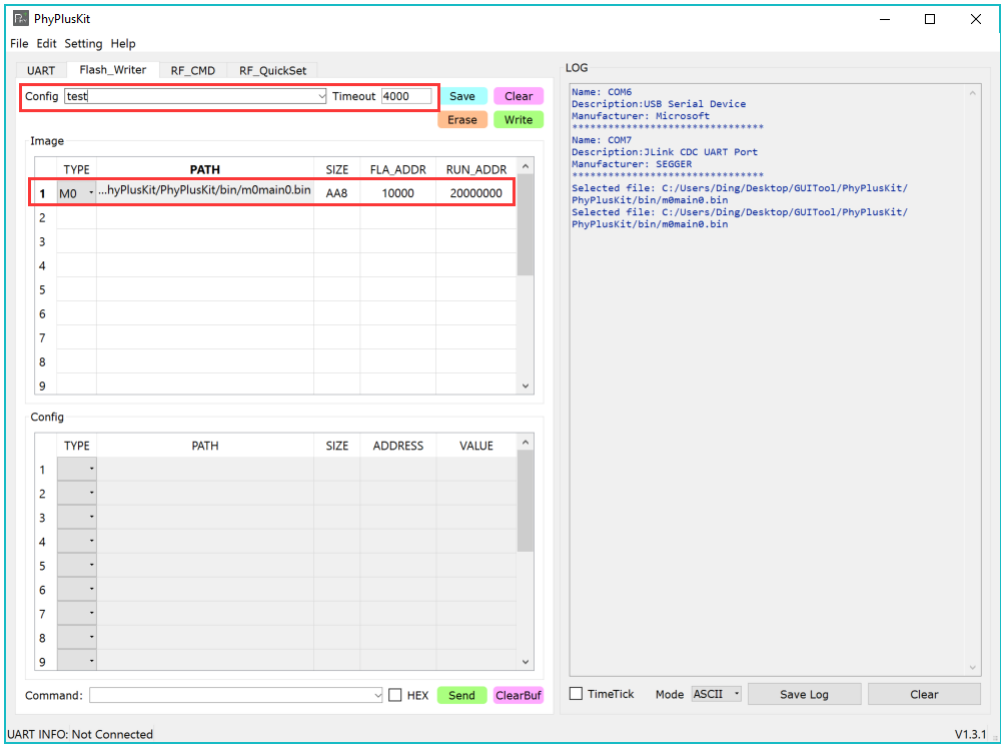
2. Run PhyPlusKit.exe, configure parameters in UART Tab (115200, 8bit, 1 bit stop, None parity, no flow control), then click Connect



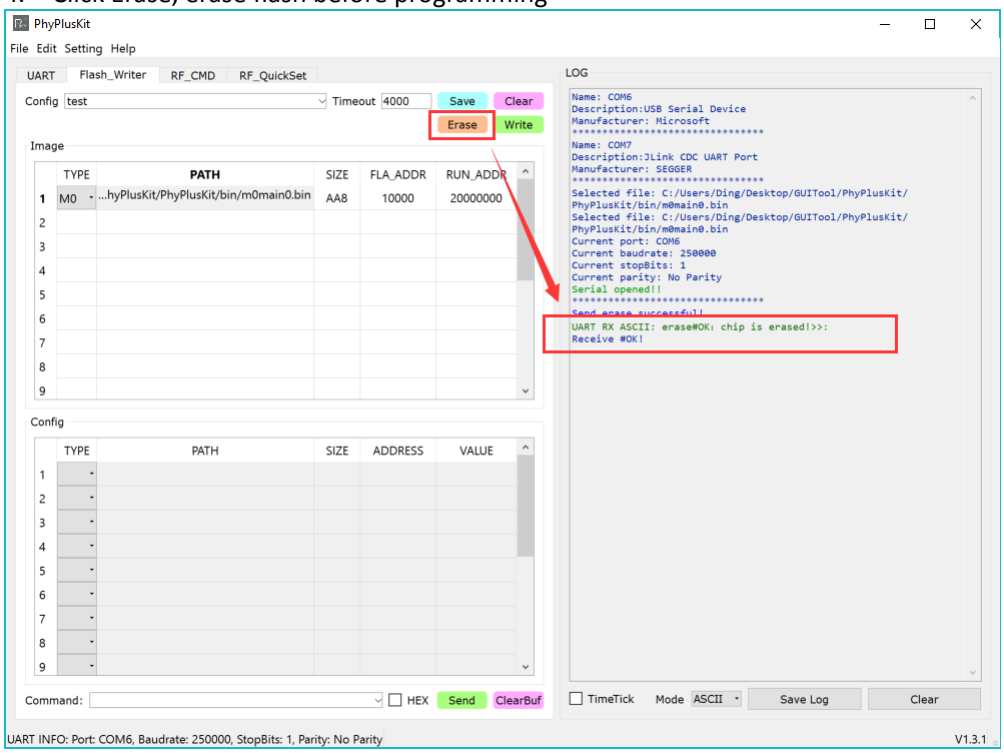
3. Select BIN file after connected. (Double click the cell in the PATH column)，type(M0)，Configure fla_addr and run_addr，The configuration information is as follows:
   1) fla_addr:flash offset address, suggest 0x10000
   2) run_addr:APP operation address, default address :0x1FFF4000

Configuration can be saved for future use

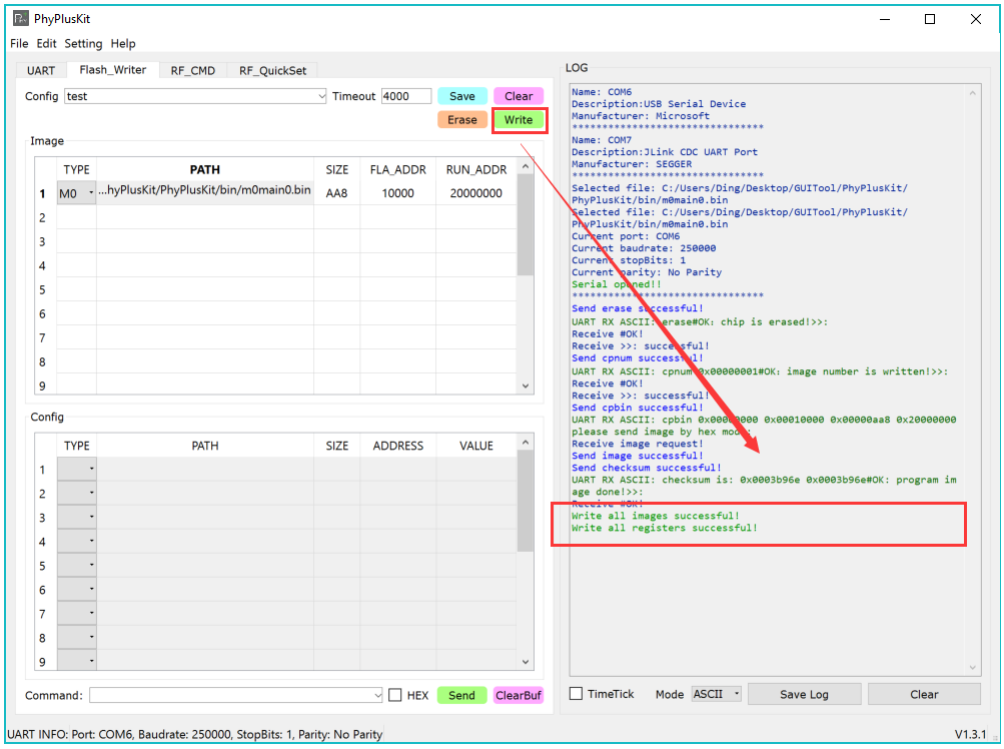




4. Click Erase, erase flash before programming



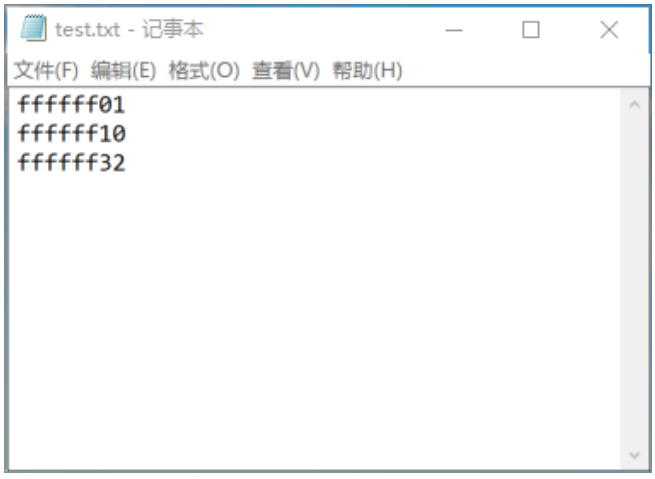
5. Click Write to start programming



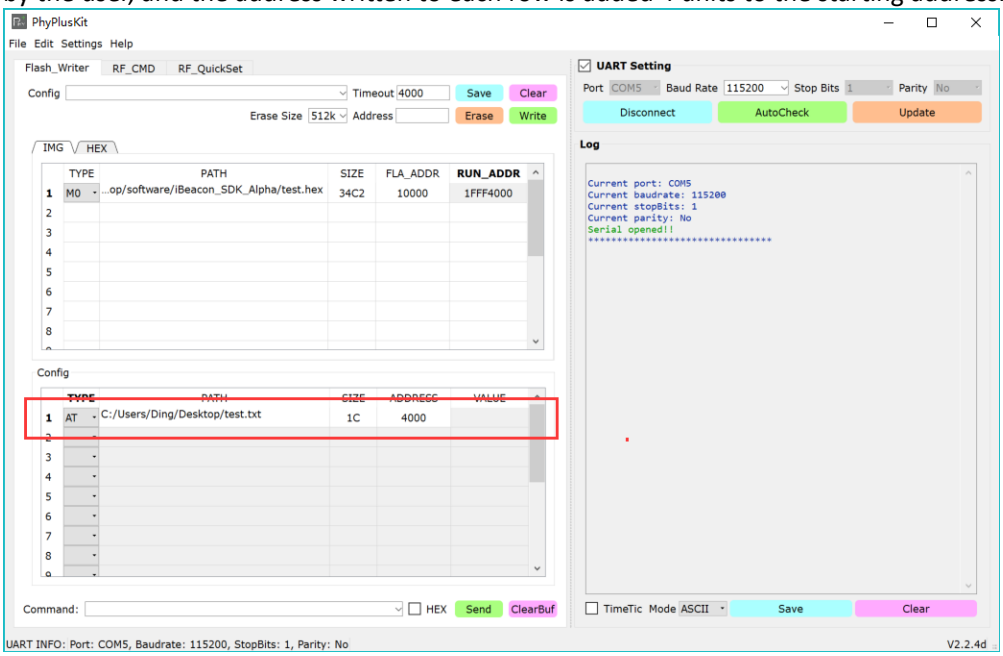
6. After programming successfully, pull-down TM pin, followed with reset, module will go to boot mode

3.2.3. Program image and config

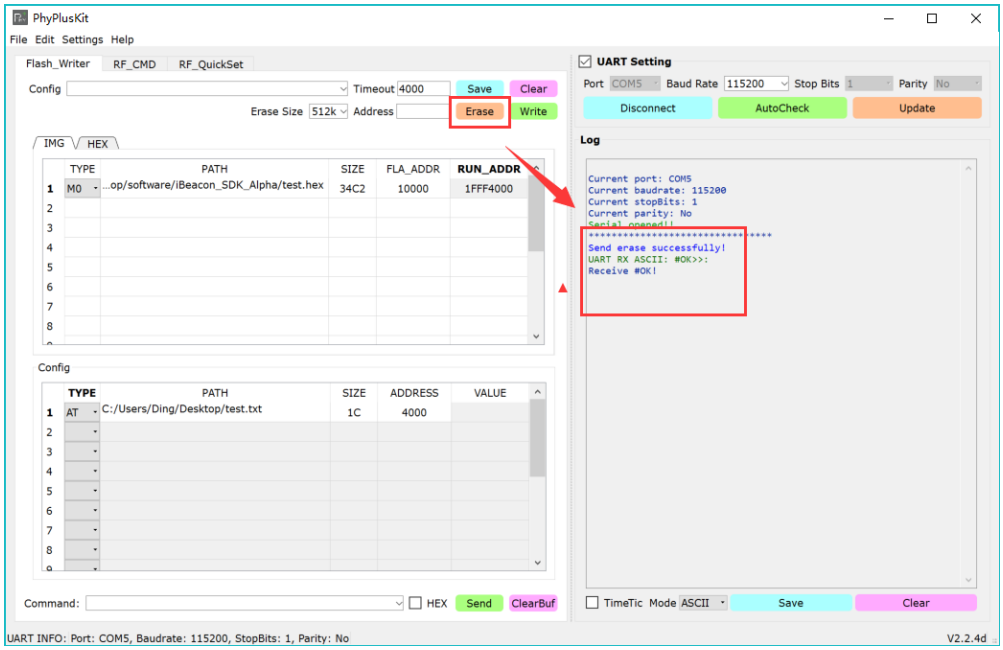
1. repeat the previous steps 1-5
   Select AT mode to read the file storing the register value, and fill in the starting address
   Starting address such as: 4000
   Document content:



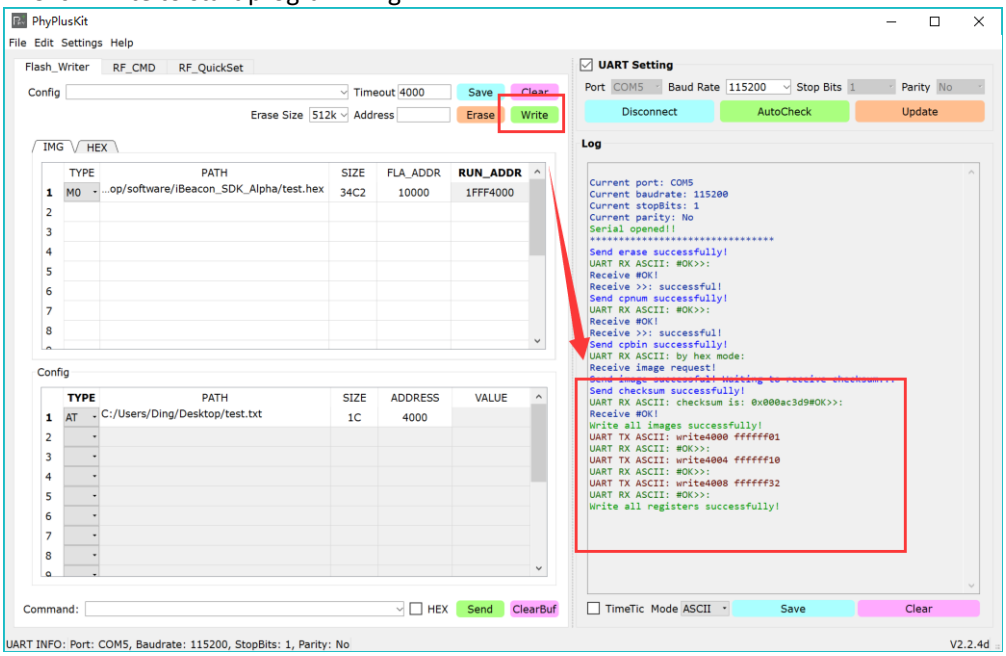
2. Each line in the document represents value to be written, the starting address is specified by the user, and the address written to each row is added 4 units to the starting address.



3. Click Erase to erase the flash

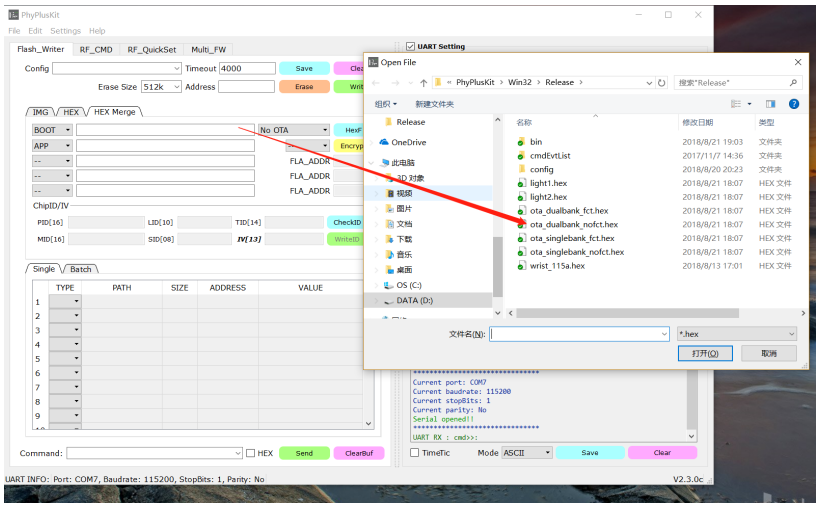


4. Click Write to start programming

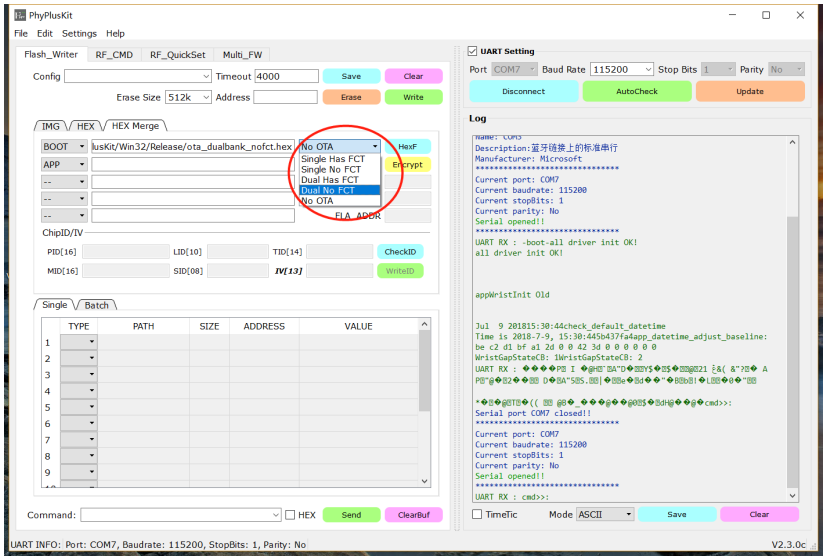


3.2.4. HexMerge programming

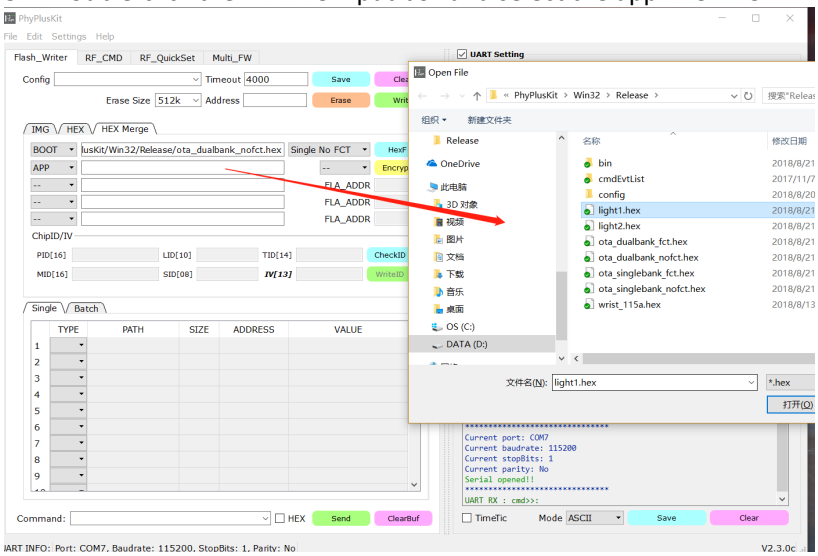
1. Double-click the BOOT file input box and select the ota*.hex file.



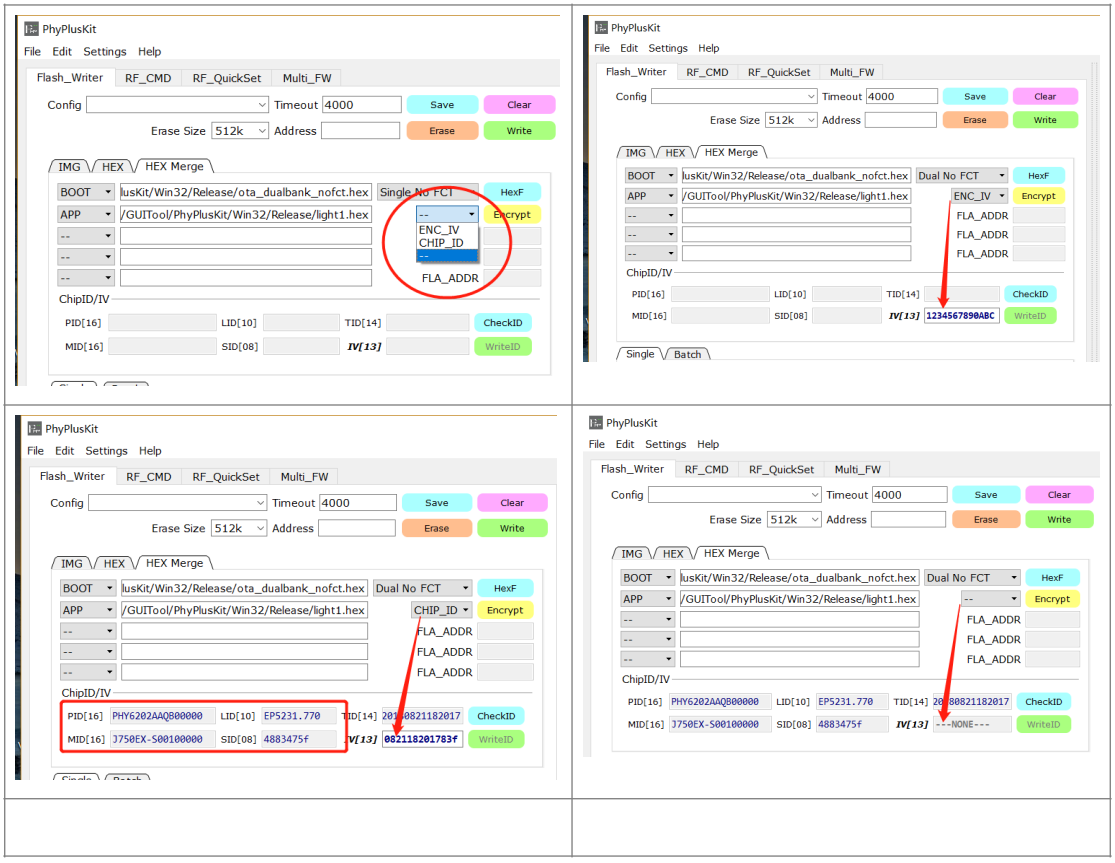
2. Select the appropriate OTA_BOOT mode



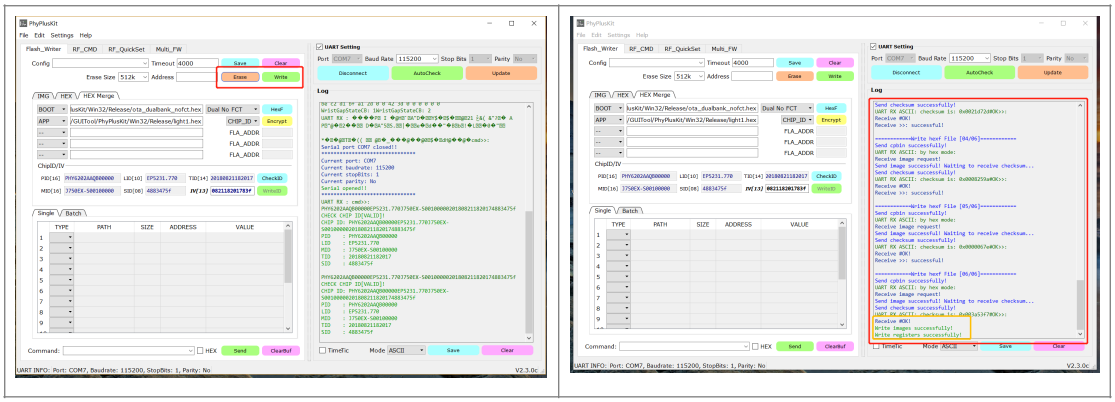
3. Double-click the APP file input box and select the app*.hex file.



4. Select the encryption method of the APP file:
   a) ENC_IV: input IV manually
   b) CHIP_ID: auto check the connected chip ID, auto calculate the IV
   c) NO_ENC: No encryption.



5. Click Erase to erase the flash, then click Write to program.



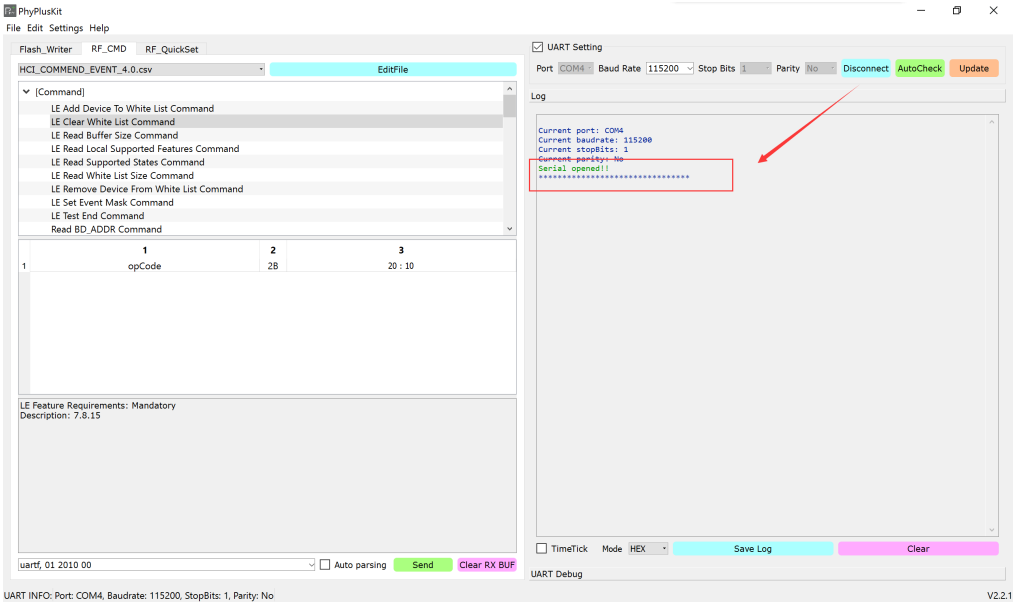
After the burning is completed, the *.hexf and *.hexe files and the corresponding files are generated in the app directory.

- The .hexf file is the combined output of multiple hex files, which can be directly burned by PhyPlusKit.
- The .hexe file is the encrypted output of the app*.hex file, and it is also the encrypted file of the ota upgrade later.

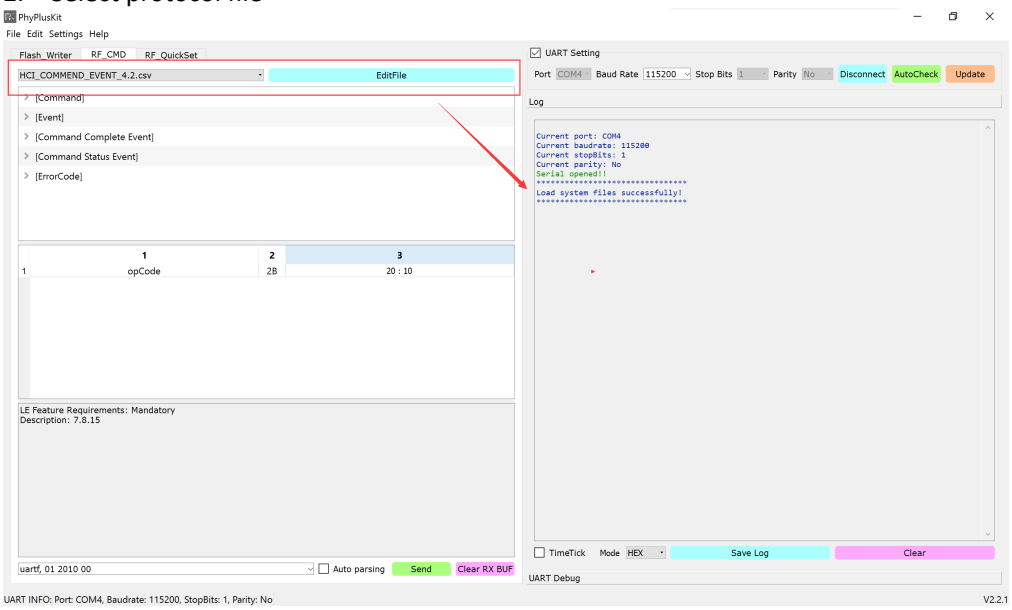
3.3. Using RF Command

3.3.1. RF Command TX

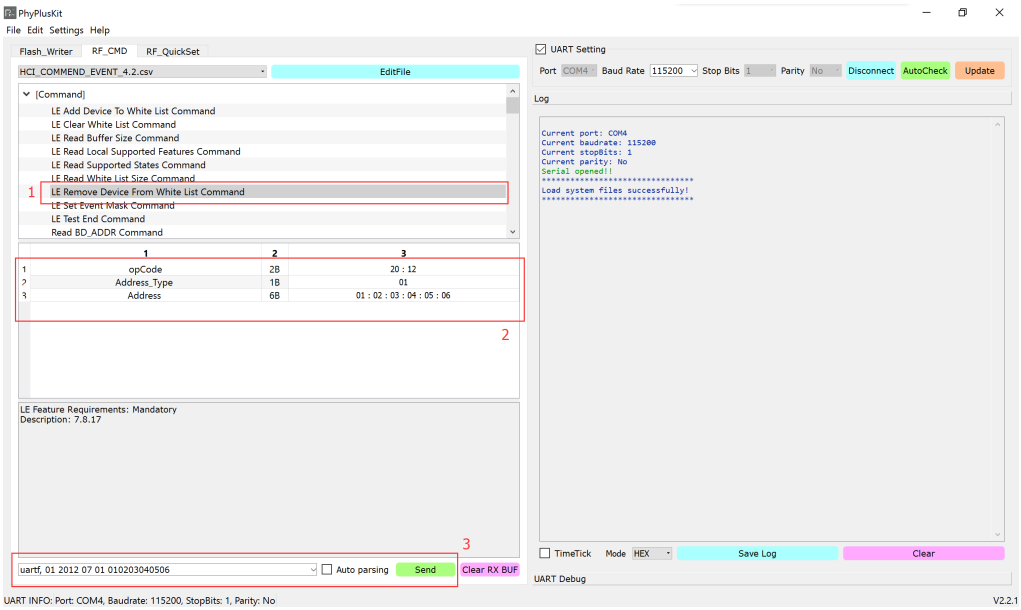
1. UART settings are baud rate: 115200, 8bit, 1 bit stop, None parity, no flow control;



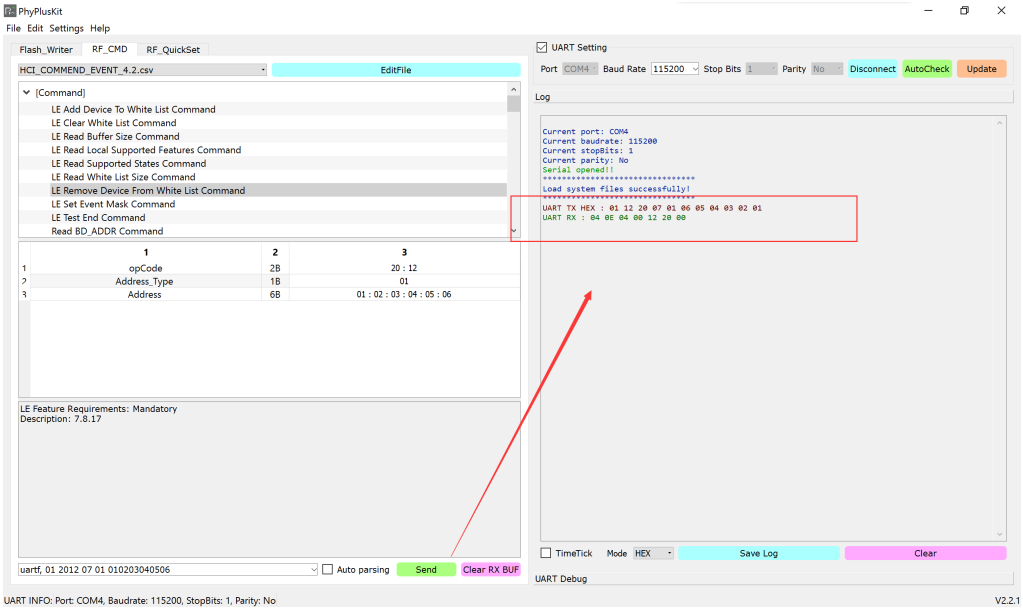
2. Select protocol file



3. Select the command to send and configure the command content in the form

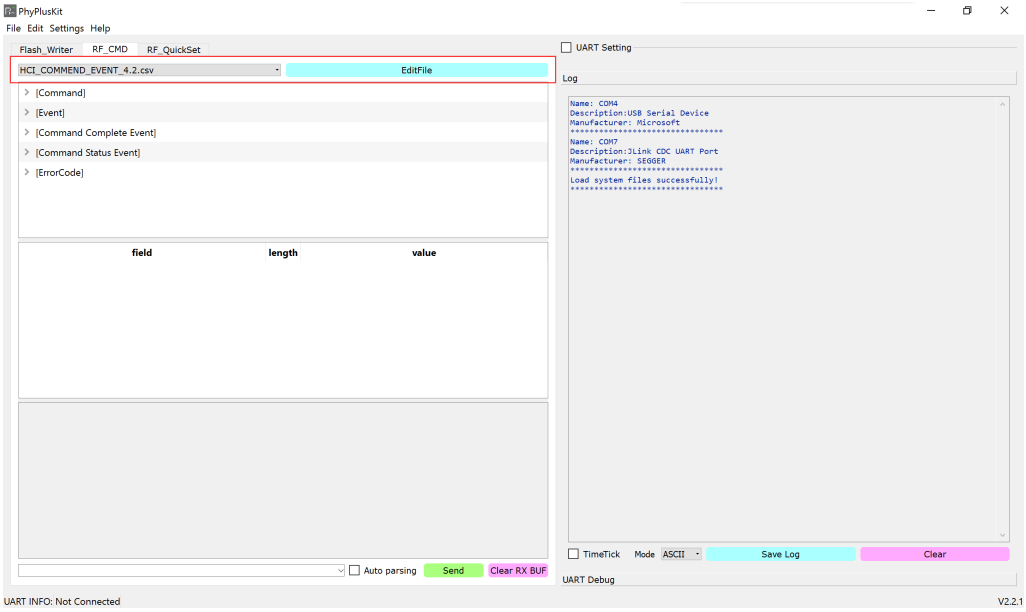


4. Click Send to send out command

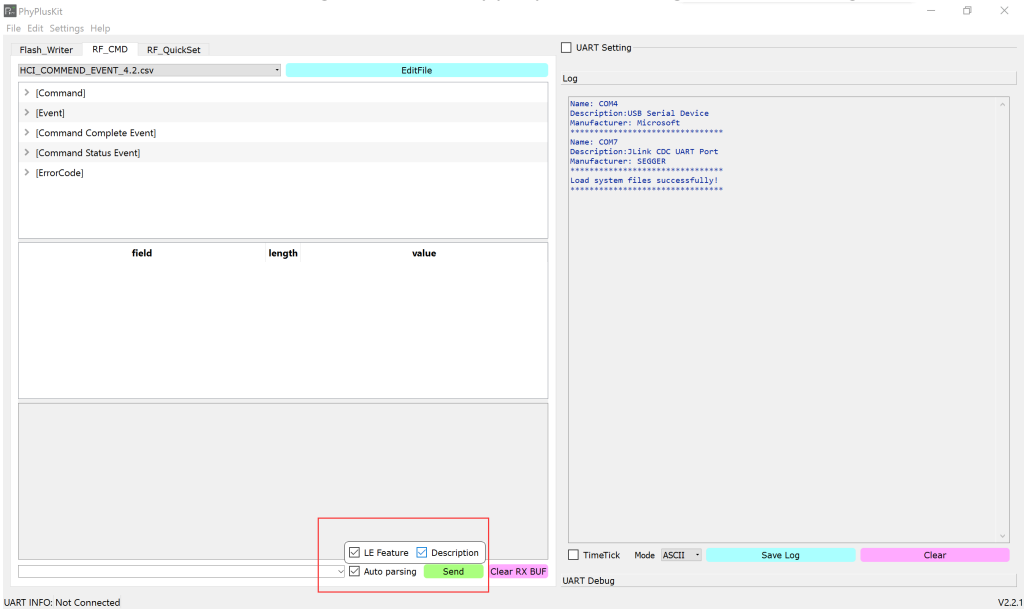


3.3.2. RF Command RX

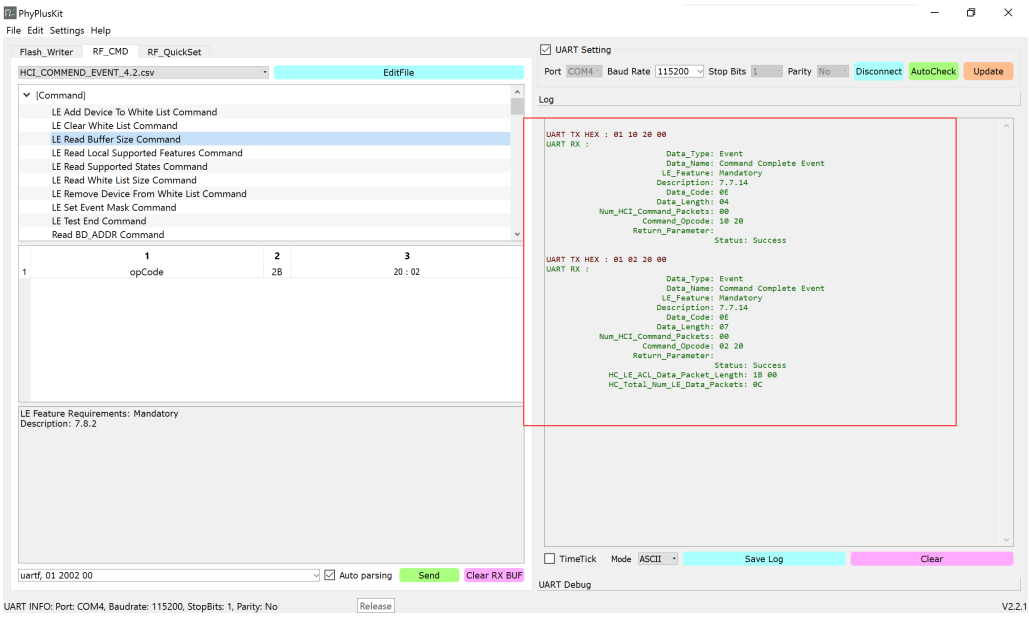
1. First load the protocol file to be parsed



2. Select Automatic Parsing, and select appropriate settings in the floating window

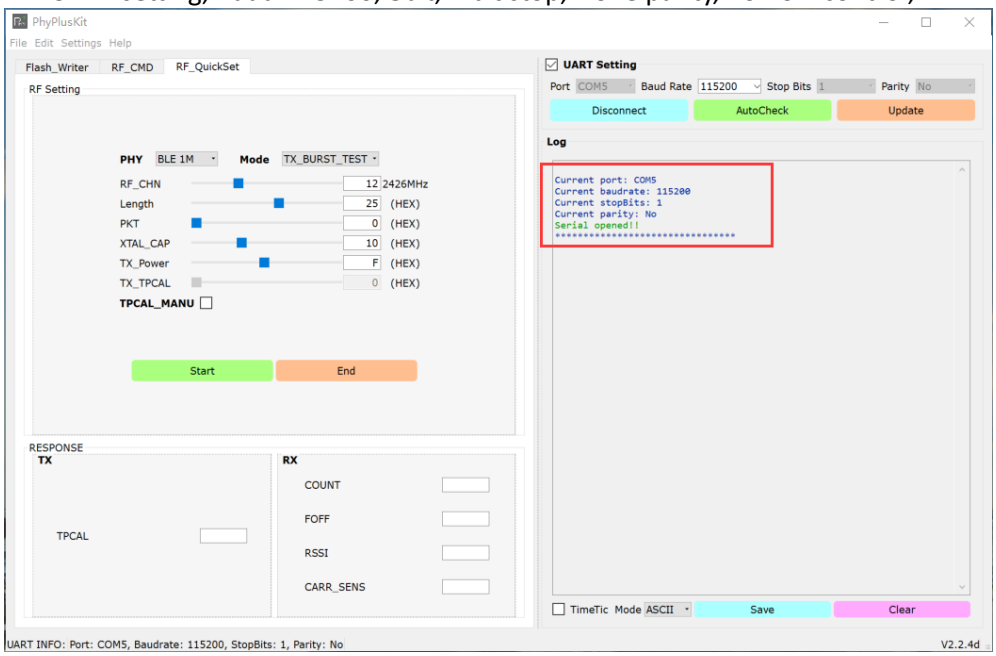


3. The automatically parsed package content will be output in the Log

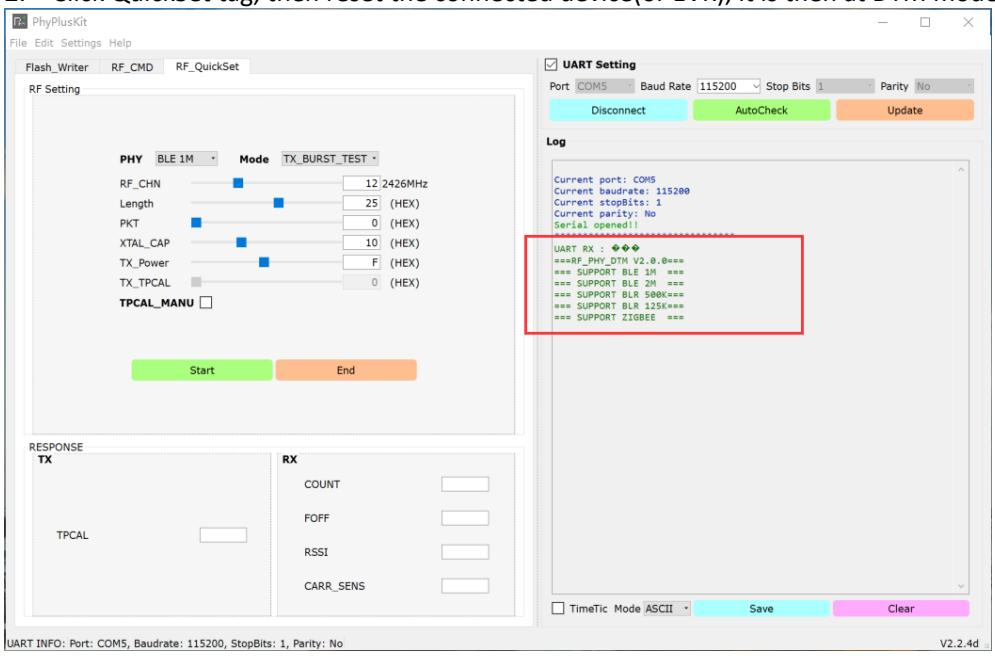

PhyPlusKit
File Edit Settings Help
Flash_Writer
RF_CMD
RF_QuickSet
HCI_COMMEND_EVENT_4.2.csv
EditFile
[Command]
LE Add Device To White List Command
LE Clear White List Command
LE Read Buffer Size Command
LE Read Local Supported Features Command
LE Read Supported States Command
LE Read White List Size 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
Port COM4
Baud Rate 115200
Stop Bits 1
Parity No
Disconnect
AutoCheck
Update
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: 0E
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: 0E
Data_Length: 07
Num_HCI_Command_Packets: 00
Command_Opcode: 02 20
Return_Parameter:
Status: Success
HC_LE_ACL_Data_Packet_Length: 1B 00
HC_Total_Num_LE_Data_Packets: 0C
TimeTick
Mode ASCII
Save Log
Clear
UART Debug
UART INFO: Port: COM4, Baudrate: 115200, StopBits: 1, Parity: No
Release
V2.2.1

3.4. Using RF QuickSet

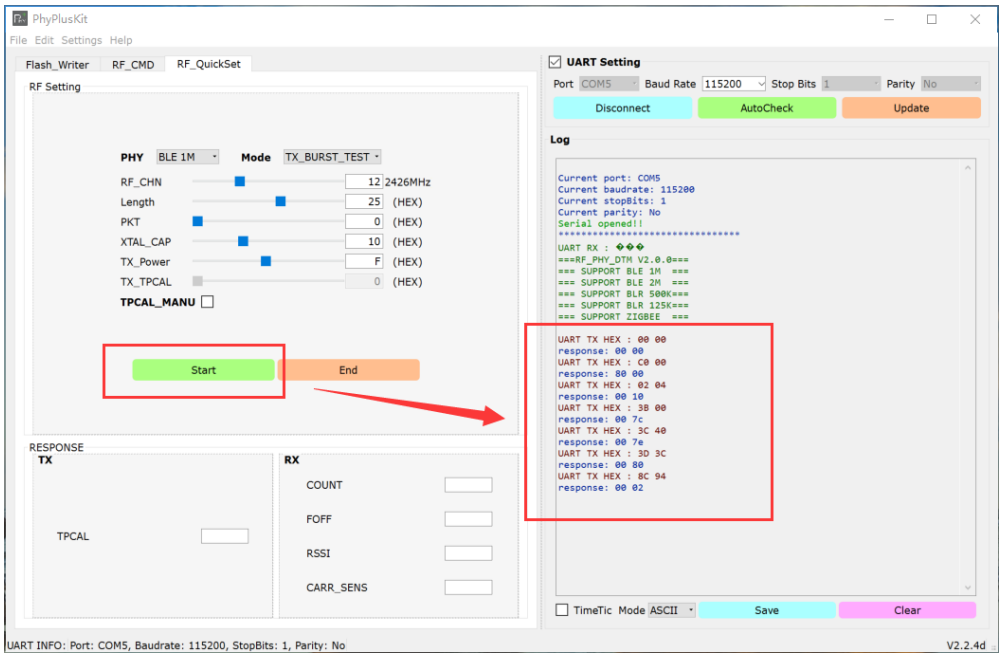
1. UART setting, Baud:115200, 8bit, 1 bit stop, None parity, no flow control;



2. Click QuickSet tag, then reset the connected device(or EVK), it is then at DTM mode



3. Configure the Tx parameter, click Start to send out the command/data.



PHY： Set the physical type:BLE1M, BLE2M, BLE500K, BLE125K or ZIGBEE

MODE： **TX_BURST_TEST**, Transmits BLE packets at regular intervals.
TX_SINGLE_TONE,Transmit single tone signal for frequency offset and transmit power and phase noise detection
TX_MODULATION, transmits a continuous modulated signal
RX_BURST_TEST, Enter RX demodulation mode, count the number of received packets
RX_AUTO,Automatically count the number of correct data packets received every 1000 data packet intervals.

RF_CHN: Set the RF Frequency，
For BLE，RF_FREQ=RF_CHN*2+2400
For ZIGBEE， RF_FREQ=RF_CHN*5+2400

Length: TX packet Length Unit=BYTE

PKT： TX packet type, 0-> prbs9, 1-> 11110000, 2->10101010,3-> prbs15

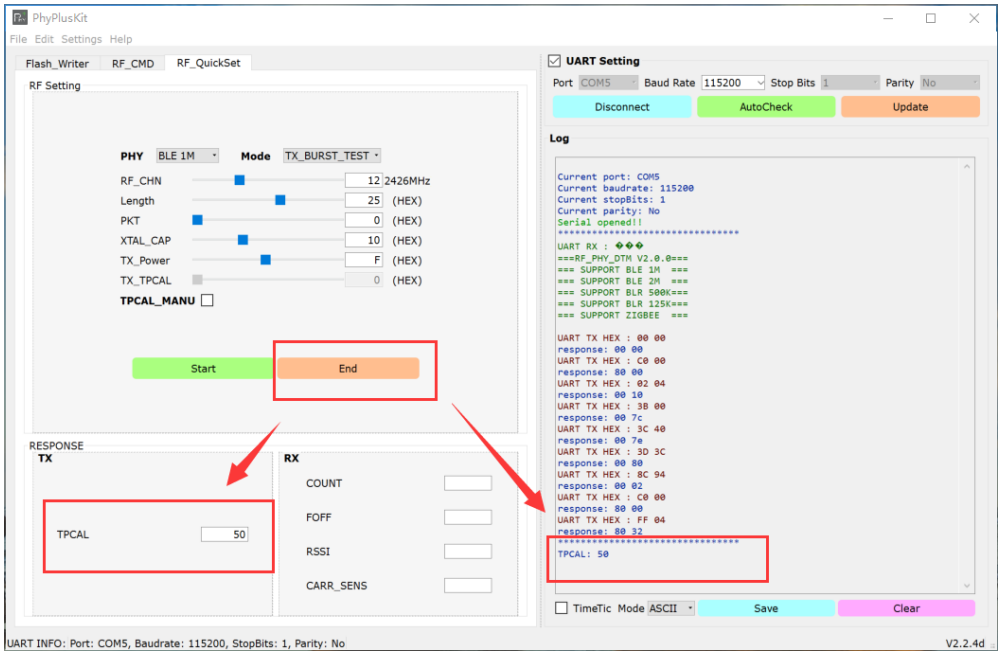
XTAL_CAP: Adjust core chip internal CAP loading, change RF的Frequency Offset。

TX_Power: Adjust the RF Tx power, the range is [0-0x1f], which is proportional to the power, when the value is 0x0A, the power is 0dBm

TPCAL_MANU: Adjust the TX efficiency. If not click Manual, core chip will implement internal automation calibration. If Manual is selected, user needs to fill in the TPCAL value. Manual is NOT recommended to be clicked.

All new parameter valid only after START is clicked

4. Click End to end the test, and obtain the corresponding parameter（TPCAL）



When the test mode is TX, clicking End will automatically follow the new TPCAL result.
When the test mode is RX, update the following parameters:

COUNT: The actual number of correct Packets received.

FOFF: RX_PHY frequency offset estimate （KHZ） 。

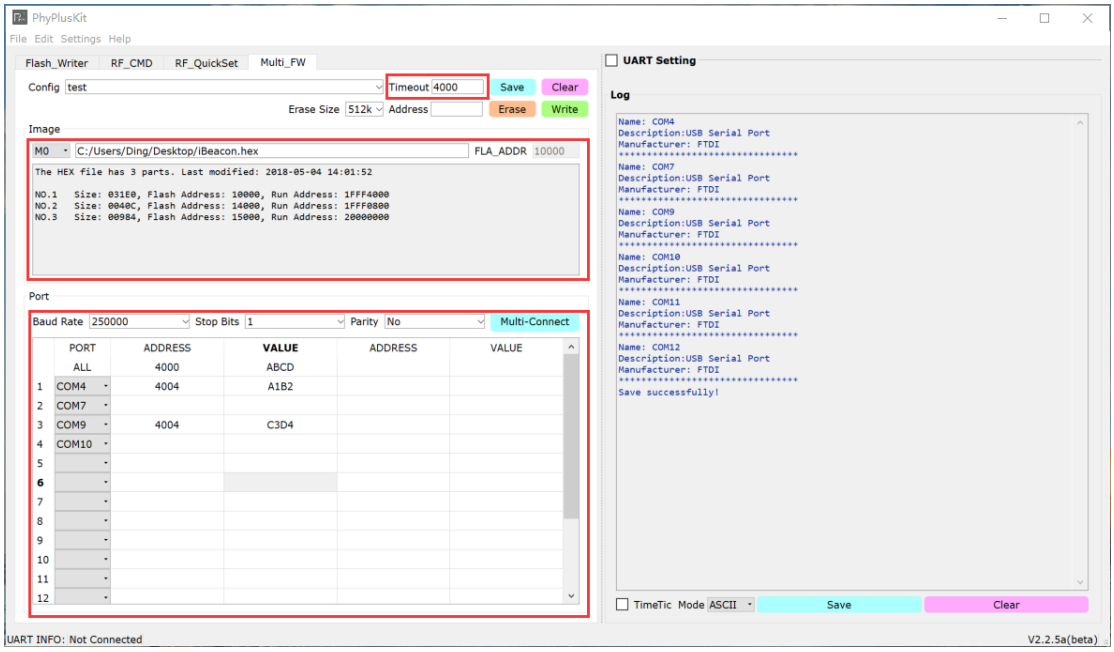
RSSI: Estimated value of received signal strength (dBm)

CARR_SENS: An estimate of signal quality.

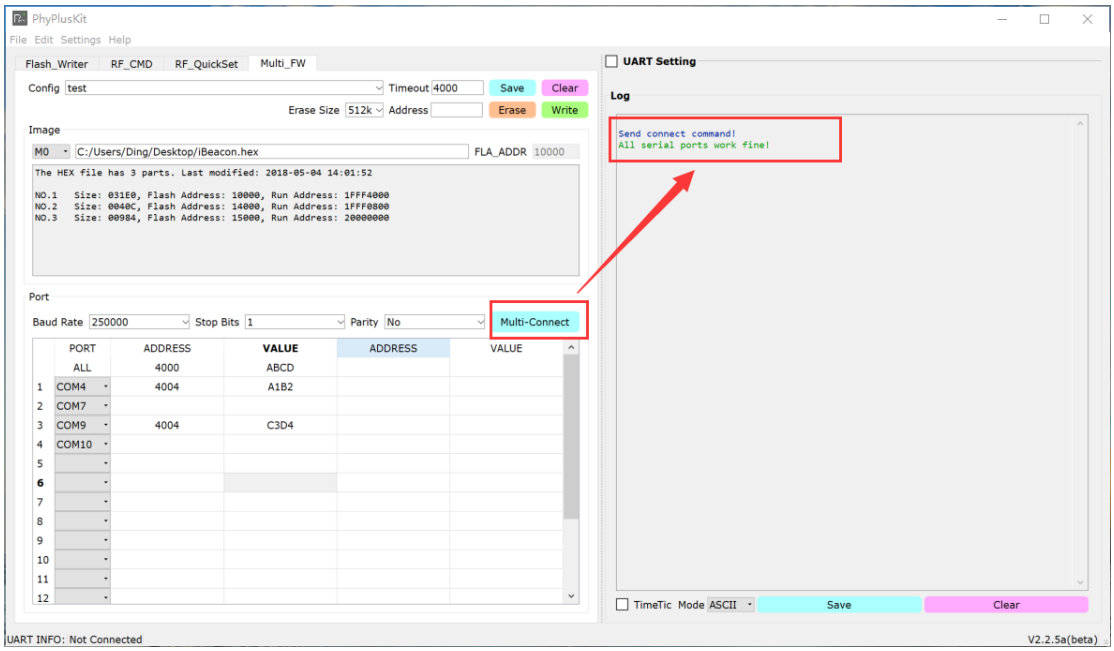
RX_BURST_TEST mode. The statistical time is from the click of Start to the click of End.
RX_AUTO mode. The statistical time is 1000 Packet Intervals. Automatic update. Click End to exit RX_AUTO mode.

3.5. Multi-FlashWriter

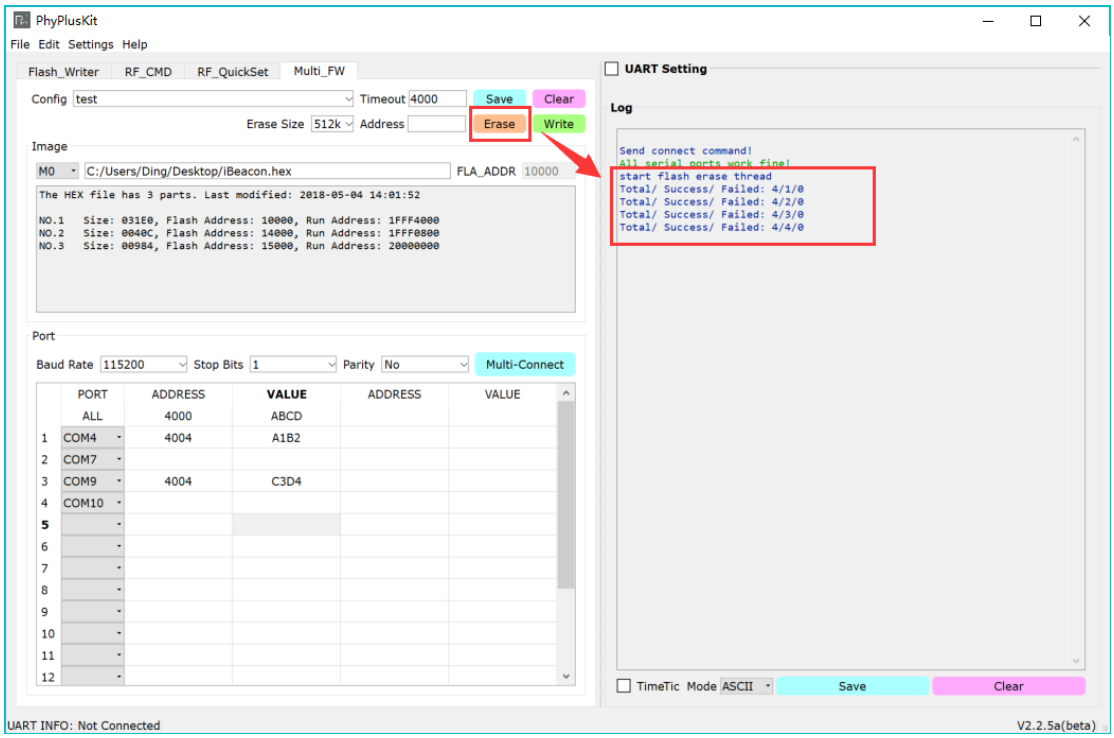
1. Set the appropriate Timeout value (the default is 4000ms), select the HEX file to be burned and check the burned (Flash Address, Run Address), select the PORT port to be burned, and fill in the value of the specified address as needed.



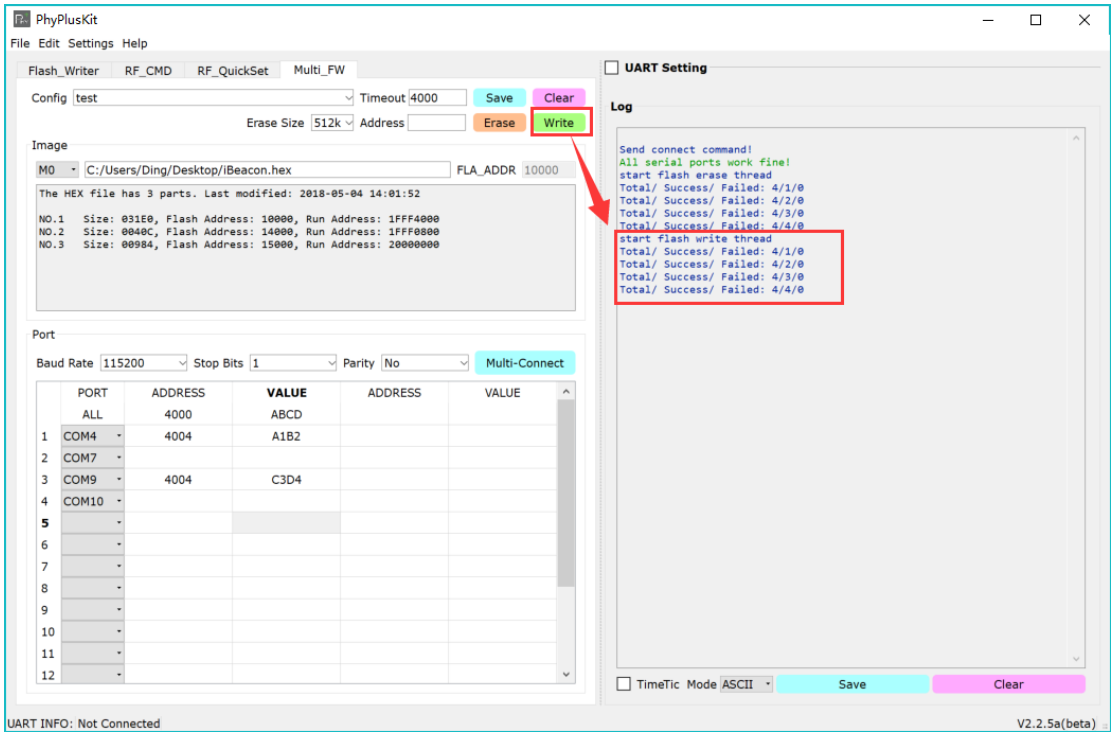
2. After the port to be burned is set, click Multi-Connect to connect the port



3. After the port is connected normally, click the Erase button to erase (return the total number/ success/failure count)



4. After the device is erased normally, click the Write button to program (return the total number/success/failure count)



3.6. Programming and mixed operation under command line

note:

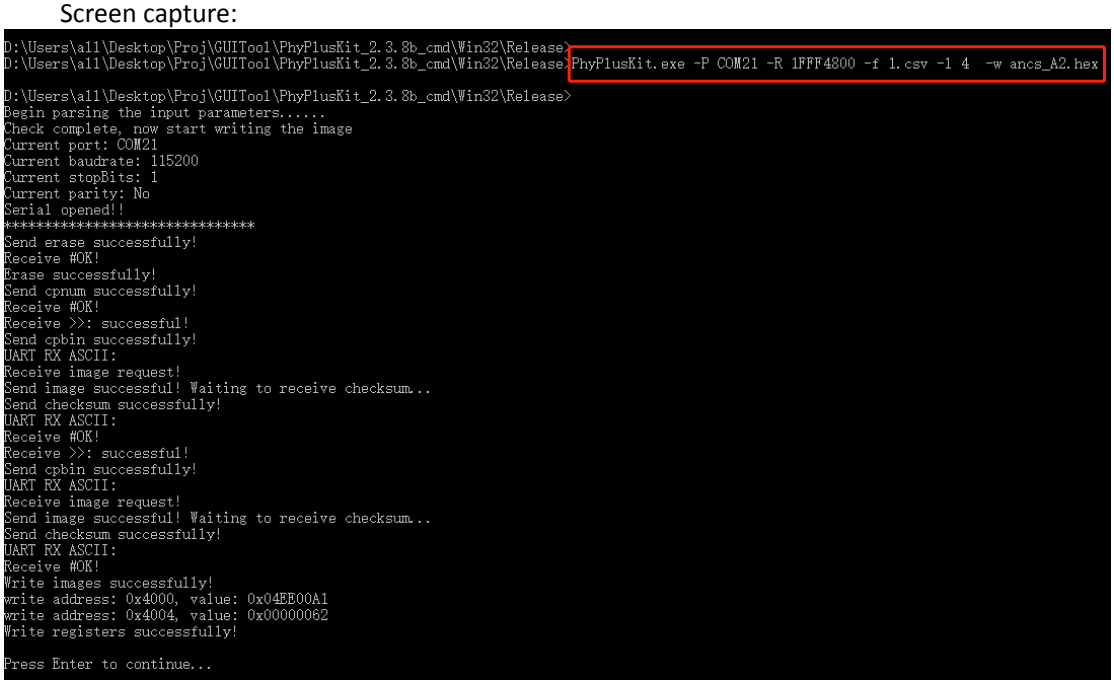
1. PhyPlusKit.exe programming tool, firmware programming, configuration file .csv, etc. need to be in the same directory
2. It is written in the form of appending when the log file is saved. The programming process will be saved in a log file. The default log file name is zPhyPlusKit.log, which is in the same path as the programming tool.

3.6.1. Program only

1. command: PhyPlusKit.exe -P COM21 -R 1FFF4800 -f 1.csv -l 4 -w ancs_A2.hex (Program hex file)

Description: Specify the Uart port as COM3, the chip belongs to PHY6212, set the run address to 1FFF4800, and write 1.csv (specified number of lines, the 4th line) and ancs_A1.hex (in the same directory as the main program) into the chip (The chip needs to be connected to the computer through the serial port), and the erasing will be performed automatically before writing. (Note: TM is pulled high)

Screen capture:



2. Command: PhyPlusKit.exe -P COM21 -R 1FFF4800 -f 1.csv -l 4 -w ancs_A2.hexf (Program hexf file)

Description: Specify the Uart port as COM3, the chip belongs to PHY6212, set the run address to 1FFF4800, write 1.csv (specified number of lines, the 4th line) and ancs_A1.hexf (in the same directory as the main program) into the chip (The chip needs to be connected to the computer through the serial port), and the erasing will be performed automatically before writing.

Screen capture:

```
D:\Users\a11\Desktop\Proj\GUITool\PhyPlusKit_2.3.8b_cmd\Win32\Release>
D:\Users\a11\Desktop\Proj\GUITool\PhyPlusKit_2.3.8b_cmd\Win32\Release>
D:\Users\a11\Desktop\Proj\GUITool\PhyPlusKit_2.3.8b_cmd\Win32\Release>PhyPlusKit.exe -P COM21 -R 1FFF4800 -f 1.csv -1 4  -w ancs_A2.hexf

D:\Users\a11\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. Merge only

Command: PhyPlusKit.exe -c -p wrist_115a.hex -r E:\test\test\bin\Debug\test.bin -a 70000 -m NO -e chip

Description: Execute the merge command, the mode is No OTA, the app file is wrist_115a.hex, the resource file is E:\test\test\bin\Debug\test.bin, the writing start address is 0x70000, and the encryption method is chip id encryption

Screen capture:

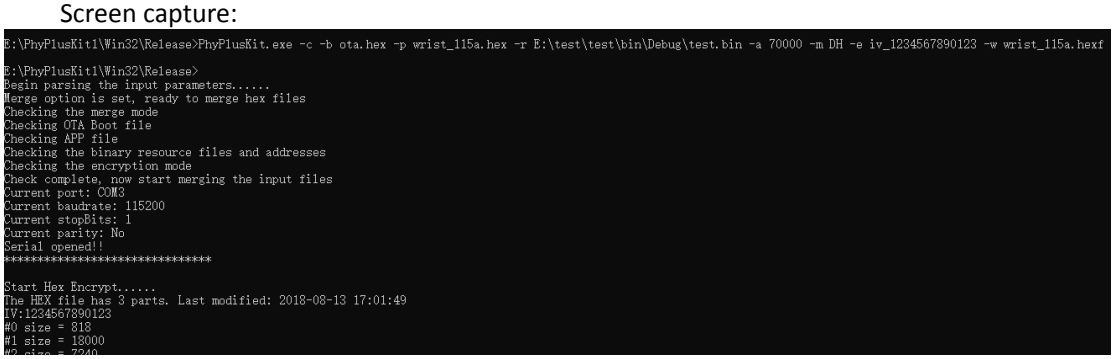
```
E:\PhyPlusKit1\Win32\Release>
E:\PhyPlusKit1\Win32\Release>
E:\PhyPlusKit1\Win32\Release>PhyPlusKit.exe -c -p wrist_115a.hex -r E:\test\test\bin\Debug\test.bin -a 70000 -m NO -e chip

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!!
********************************
12345678901234561234567890123456789012345612345678901234123456 78
CHECK CHIP ID[VALID]!
CHIP ID: 12345678901234561234567890123456789012345612345678901234123456 78
```

3.6.3. Merge then program

Command: 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

Description: Execute the merge command, the mode is Dual Has FCT, the boot file is ota.hex, the app file is wrist_115a.hex, the resource file is E:\test\test\bin\Debug\test.bin, and its writing start address is 0x70000, the encryption method is iv value encryption and the iv is 1234567890123.

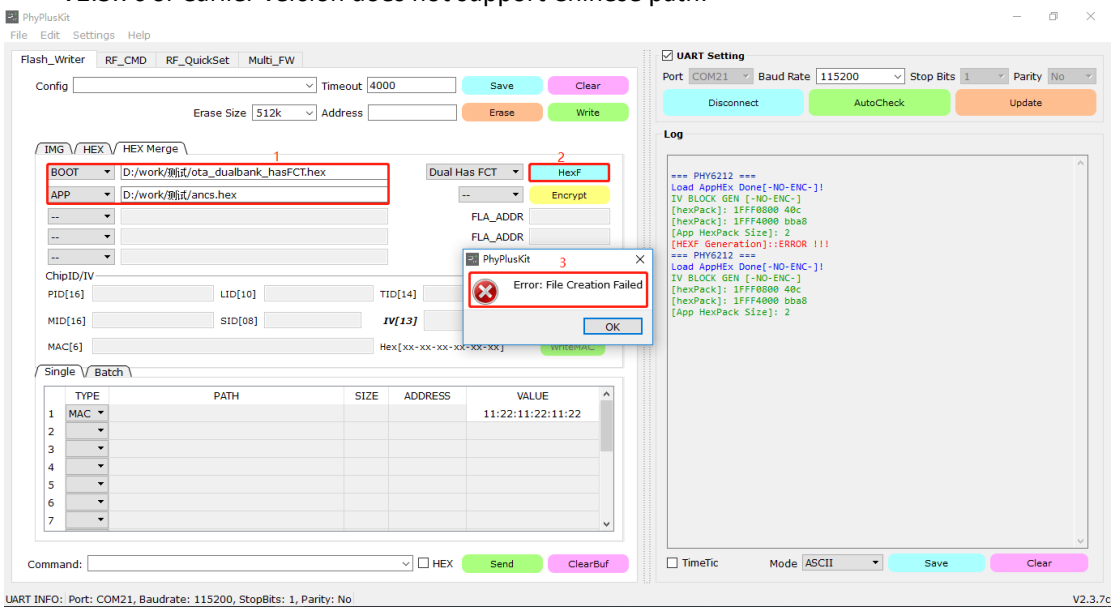
Screen capture:



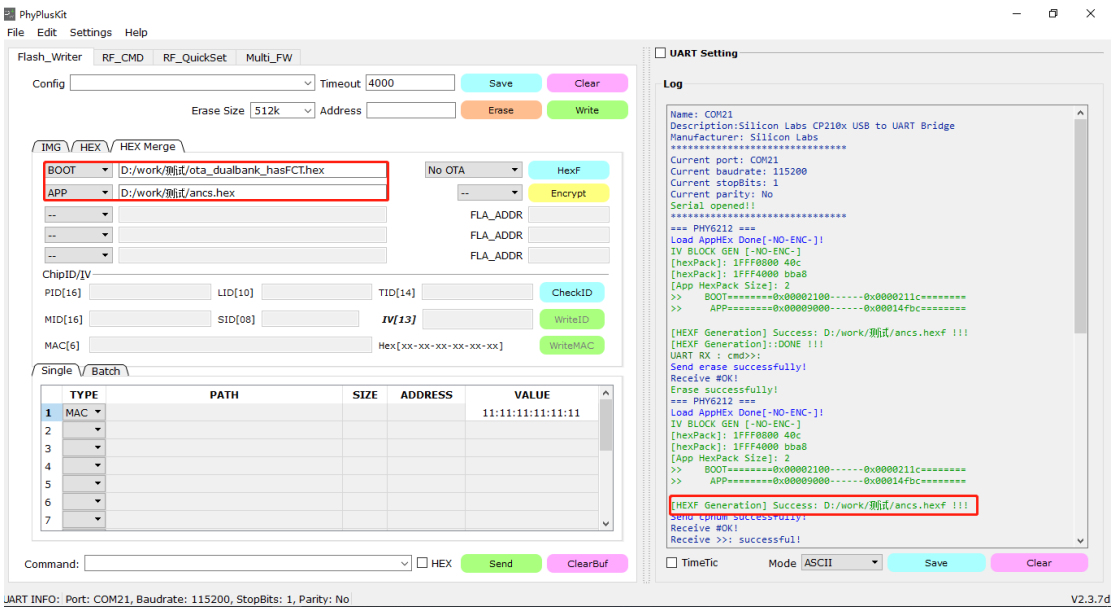
3.7. Additional setting of Flash programming

3.7.1. BOOT and APP support Chinese path

V2.3.7c or earlier version does not support Chinese path:

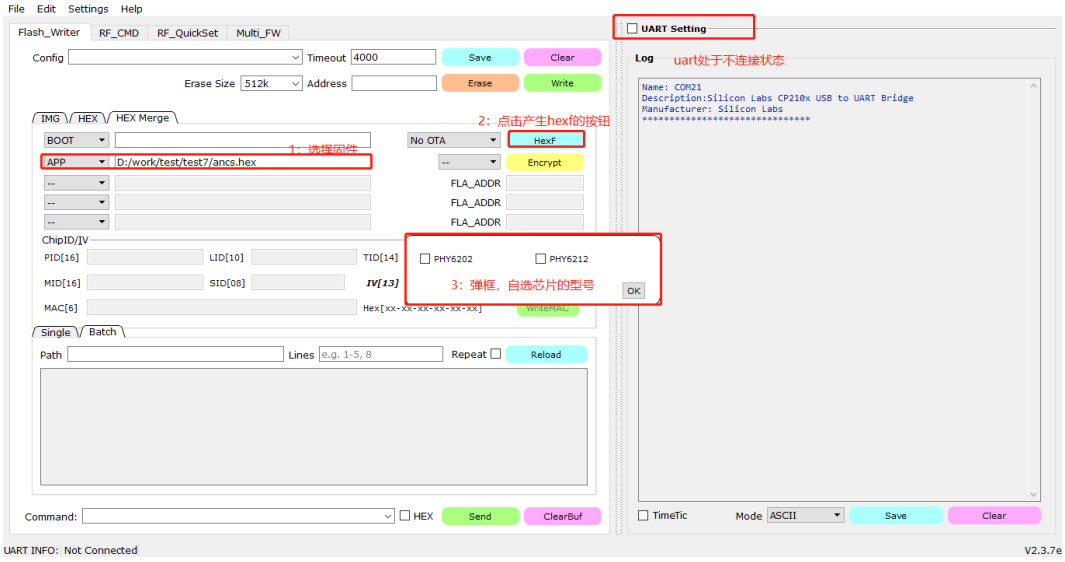


As shown in the figure, if there are Chinese characters in the file paths of BOOT and APP, an error will be reported: "File Creation Failed", and the next v2.3.7d and later versions will support Chinese paths, which will not affect the generation and programming of hexf files.

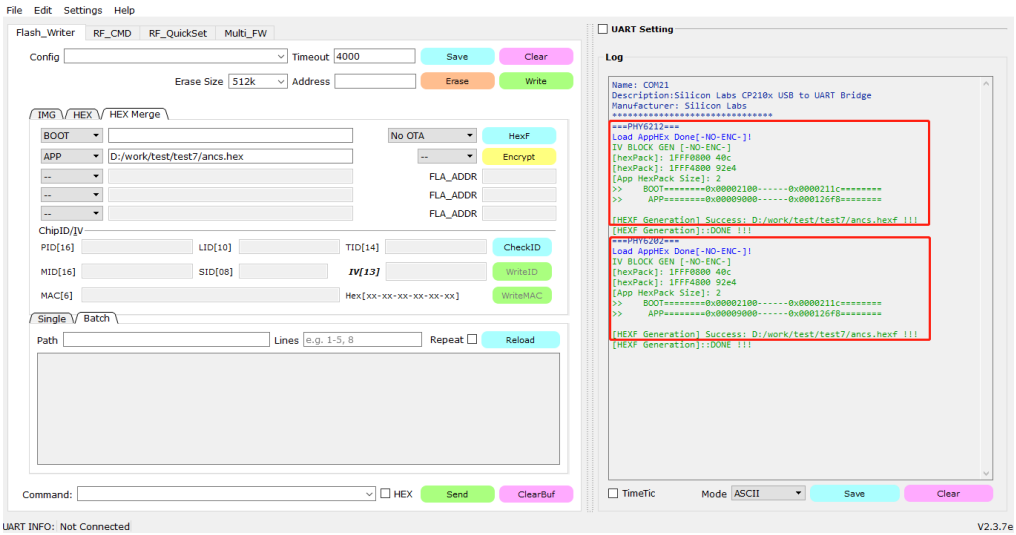


3.7.2. Program preference when device not connect

Starting from V2.3.7e, when the Uart is not connected, the default setting of the chip model is changed to the optional setting, which is shown in the following figure:

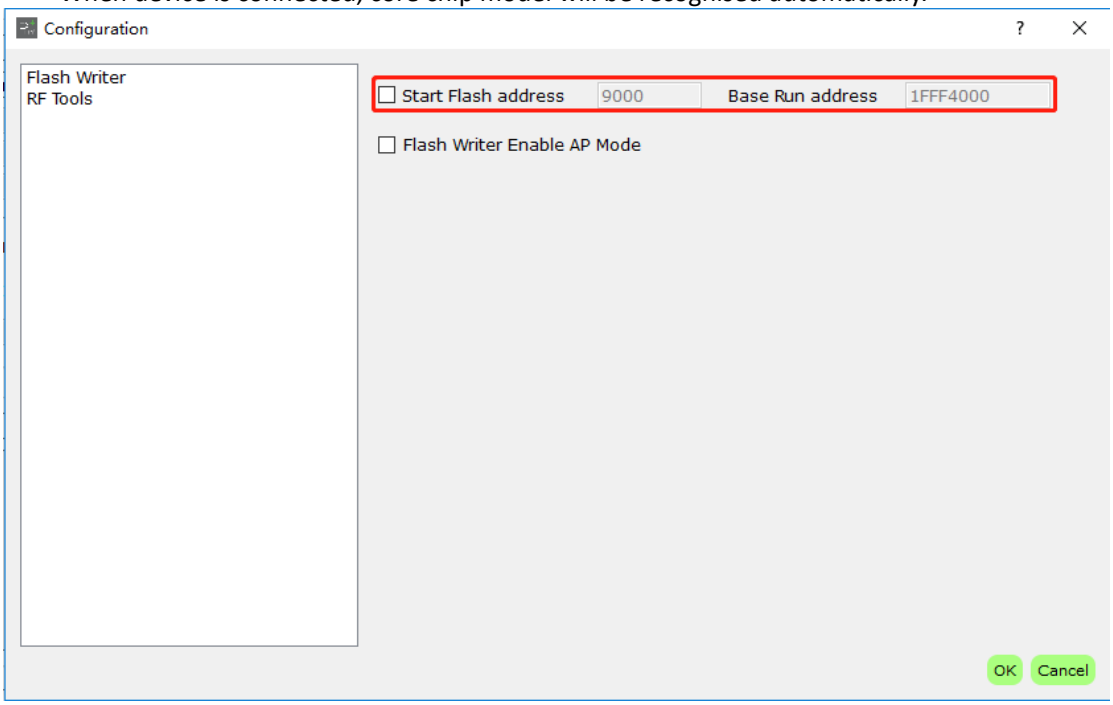


A. Device is not connected through UART
B. Load application .hex file
C. Click HexF button, dialog box pop up
D. Select the corresponding chip model and click the OK button to generate the hexf file of the loaded application firmware.



3.7.3. Programming preference when device is connected

When device is connected, core chip model will be recognised automatically.

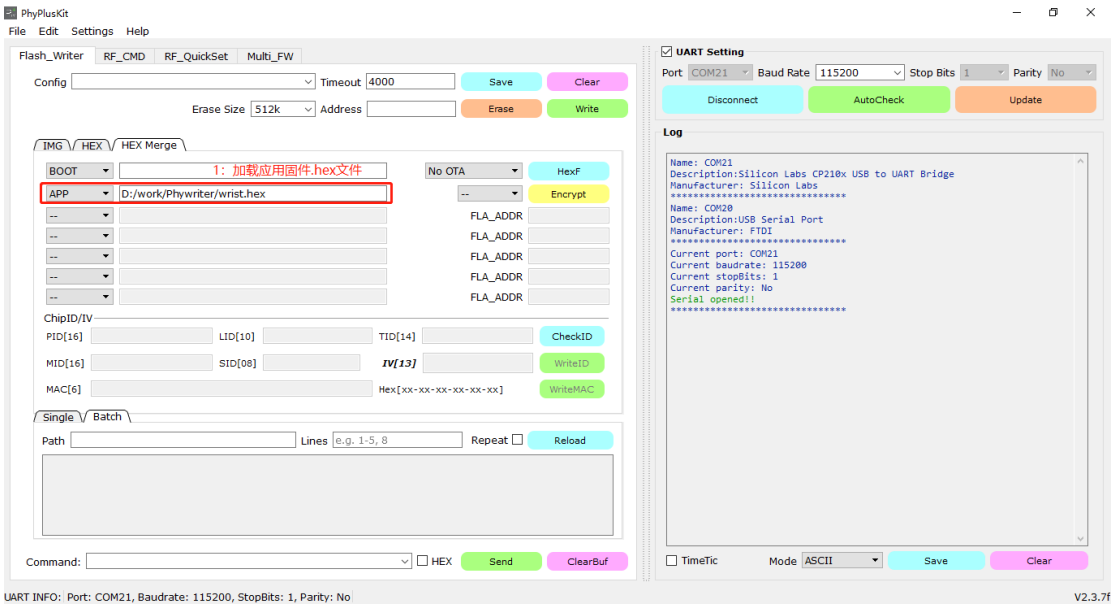


1. setting->configuration
   Start Flash address and Base Run address front setting checkbox, default setting is not ticked：Start Flash address：9000，Base Run address：1FFF4000；if ticked, parameter can be adjusted accordingly.
2. Configuration of programming flash:
   PHY6202 –> Base Run address:1FFF4000
   PHY6212 –> Base Run address:1FFF4800
   Add the input box of RUN_ADDR on the HEX tab page, you can modify the flash configuration, you do not need to open the configuration to modify the settings, which is convenient for users to operate.

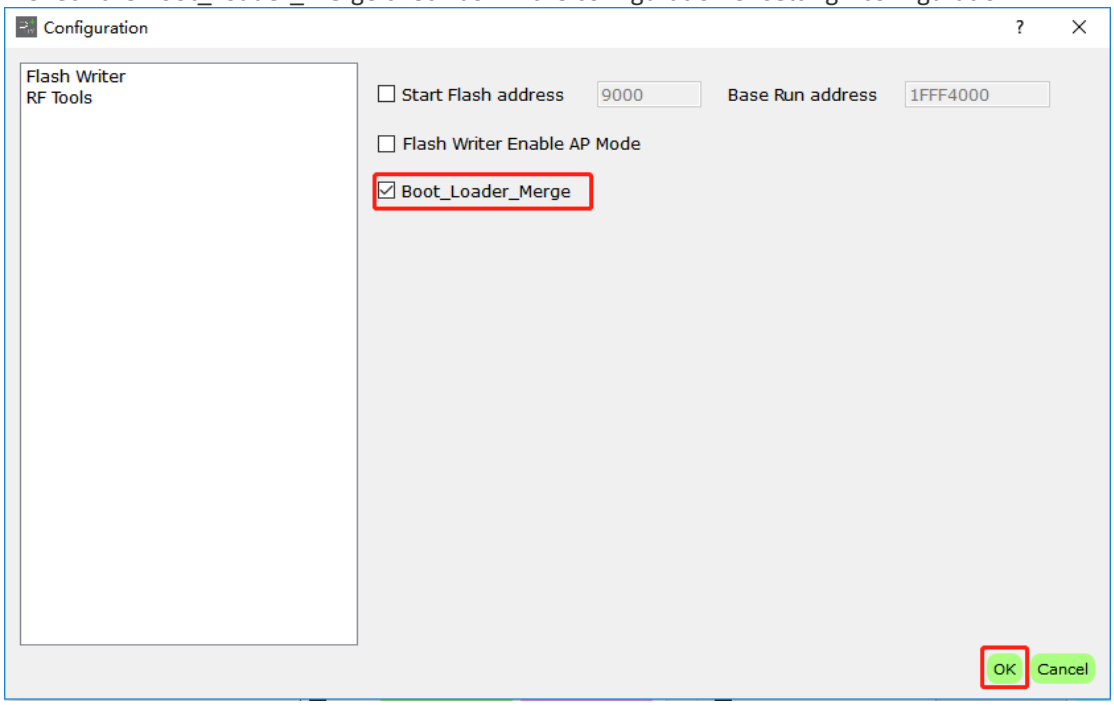
3.7.4. Merge 1M flash file for off-line programming

A. Merge the .hexf file generated by the .hex application firmware with the burning boot file *.hexf through the HEXF button on the HEXMerge page. This function starts from v2.3.7f version.

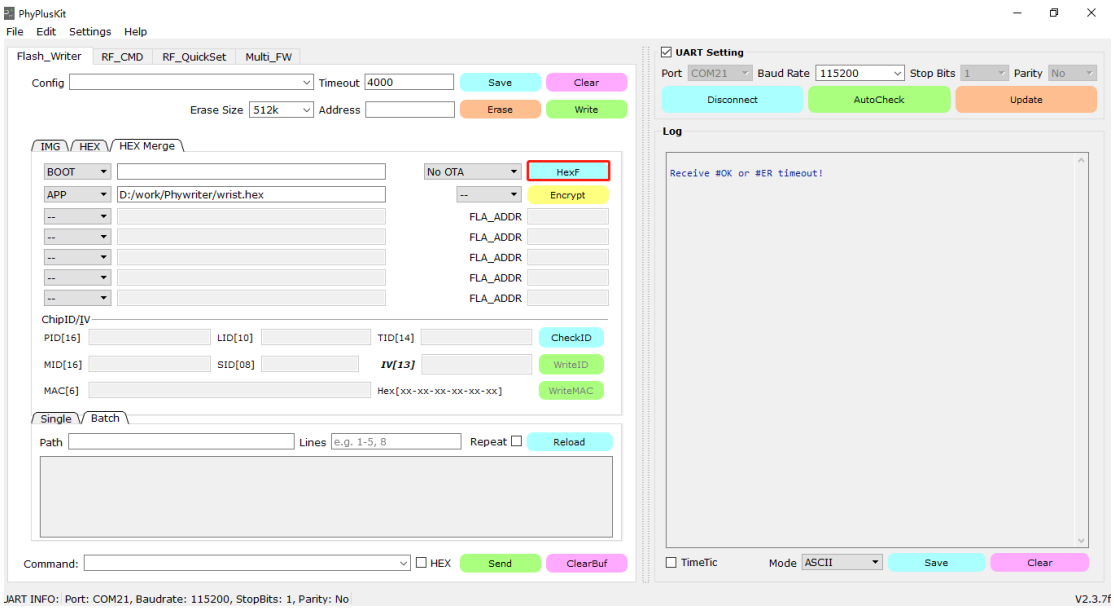
1. Load the application firmware .hex file to be burned



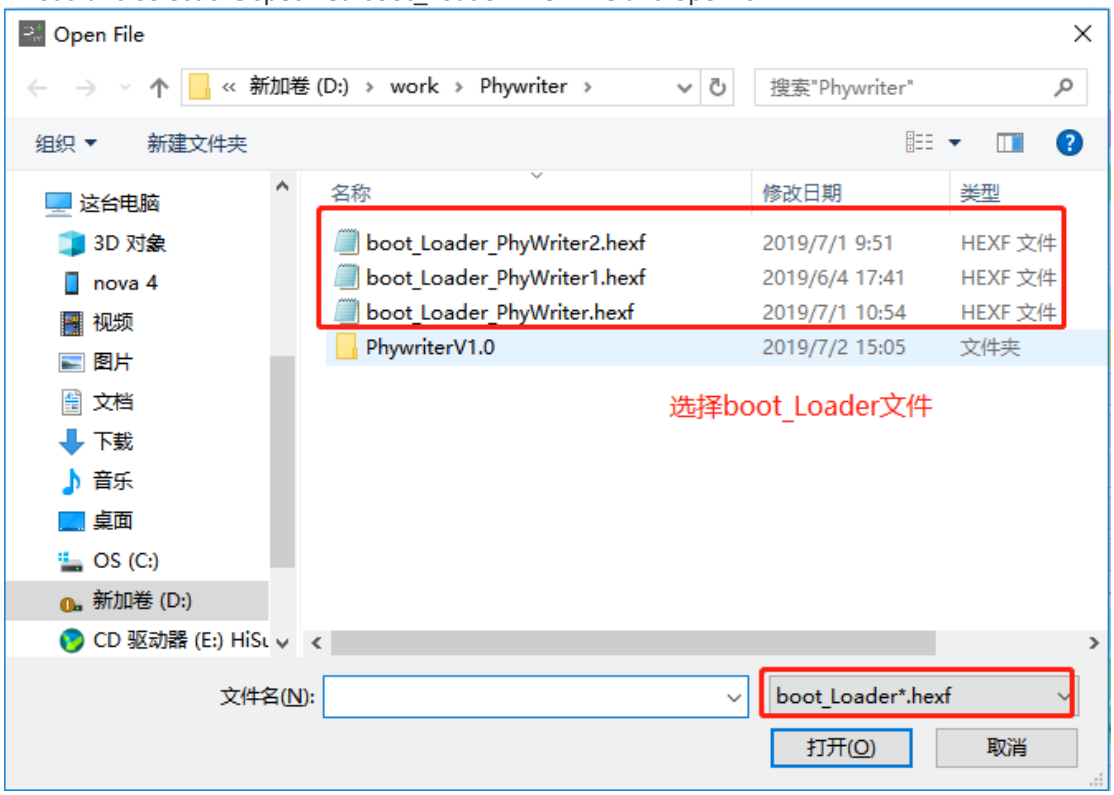
2.Check the Boot_Loader_Merge check box in the configuration of setting->configuration



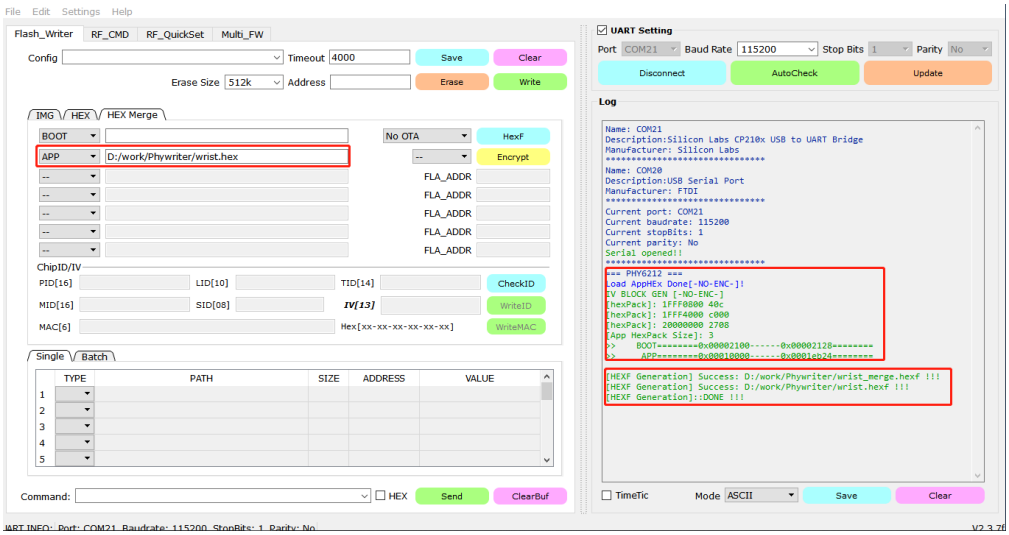
3.点击HexF按钮



4.Load and select the specified boot_Loader*.hexf file and open it



5.Generate *merge.hexf files (with bootloader) and .hexf (without bootloader) files. (Intercept the initial positions of the two *merge.hexf files for illustration, the two hexf files were successfully merged)




```
:02000004FFFFFC
:1021000002000000000A00000601D00000040FF1F52
:0C21100068BD00000C0400000008FF1F68
:02000004FFFFFC
:10A000003868FF1FD540FF1F000000000000005F
:10A010000000000000000000000000000000000040
:10A020000000000000000000000000000000000030
:10A030000000000000000000000000000000000020
:10A040000000000000000000000000000000000010
:10A050000000000000000000000000000000000000
:10A060000000000000000000000000000000000FF0
:10A070000000000000000000000000000000000EE0
:10A080000000000000000000000000000000000DD0
:10A090000000000000000000000000000000000CC0
:10A0A0000000000000000000000000000000000BB0
:10A0B0000000000000000000000000000000000AA0
:10A0C0000348854600F01CF80048004729 56FF1F4A
:10A0D0003868FF1F0648804706480268064B1A4248
:10A0E00001D106480047FEE7FEE7FEE7FEE7FEE790
:10A0F000DD43FF1FA8F0004001000000C140FF1F2A
:10A10000064C0125064E05E0E36807CC2B430C3CCA
:10A1100098471034B442F7D3FFF7D6FFCC5CFF1F4B
:10A12000FC5CFF1F30B58C180278401C13071B0F16
:10A1300001D10378401C120906D10278401C03E0CB
```

boot文件初始位置

```
:10C150000000000000000000000000000000000000DF
:10C160000000000000000000000001FF01FF01FF01FFCF
:04C17000910000003A
:020000040000FA
:1021000003000000000010000C000000040FF1FAD
:1021100008C001000C0400000008FF1F1CC40100DF
:08212000082700000000002068
:020000040001F9
:1000000068400020D540FF1FDD40FF1FDF40FF1F7D
:10001000000000000000000000000000000000000E0
:10002000000000000000000000000E140FF1F91
:10003000000000000000000E340FF1FE540FF1F3C
:10004000000000000000000000000000000000000B0
:10005000000000000000000000000000000000000A0
:1000600000000000000000000000000000000000090
:1000700000000000000000000000000000000000080
:1000800000000000000000000000000000000000070
:1000900000000000000000000000000000000000060
:1000A0000000000000000000000000000000000050
:1000B0000000000000000000000000000000000040
:1000C0000348854600F04CFD00480047C5D2FF1F9D
:1000D000684000200448804704480047FEE7FEE7E8
:1000E000FEE7FEE7FEE7FEE7D570FF1FC140FF1FFA
```

boot文件的末尾及应用固件的初始位置

Note: The merge function of two hexfs can be triggered only if Boot_Loader_Merge is checked in the configuration of setting->configuration. (Re-opening PhyPlusKit.exe requires re-checking the settings)

B. Combine two .hexf files through the merge button on the HEX page (version V2.4.1a)
Note: The format of the firmware after M0 must be .hexf. For details of the merge process between .hex file and .hexf, see content 1. 1. Error message:

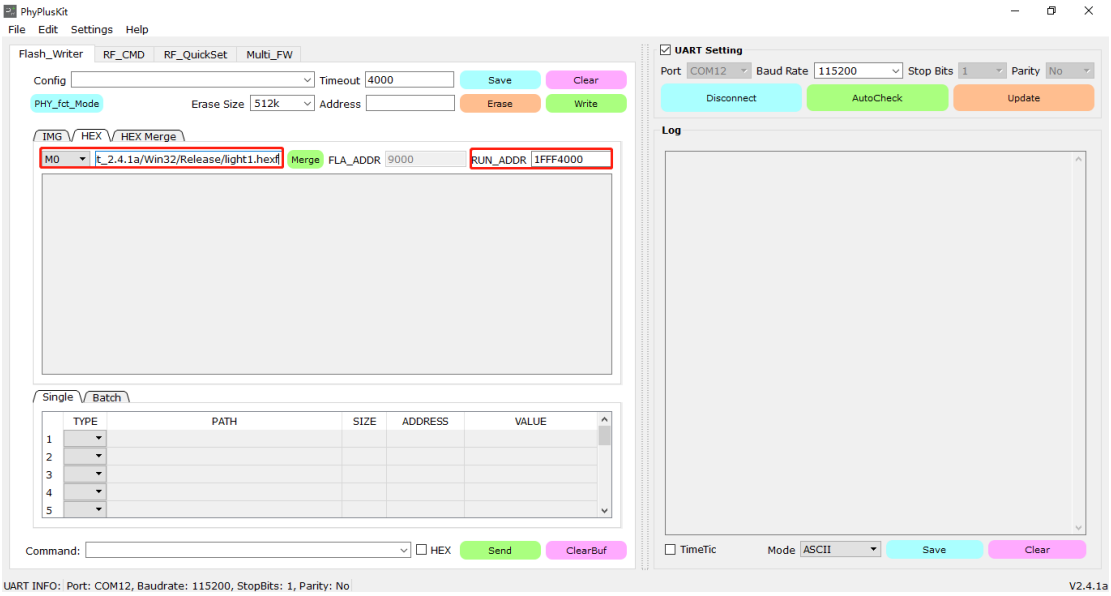
1） The HEX page does not load the firmware, click the merge button, Hint: The merge file is empty.

2） The HEX page loads the firmware in .hex format, click the merge button, Hint: The merge file is not valid.

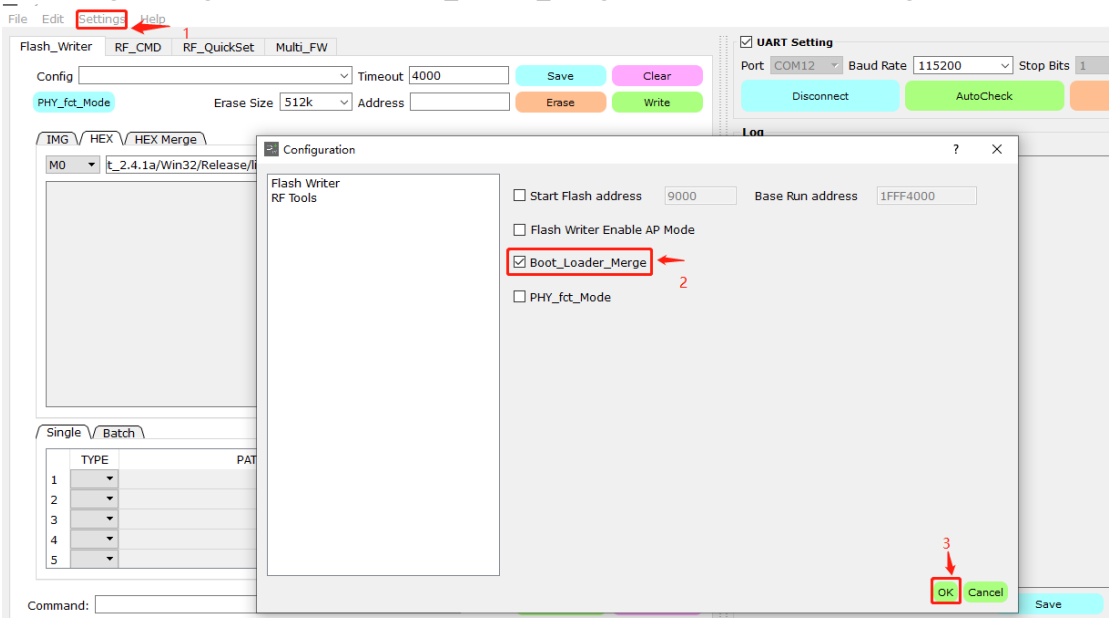
3） The HEX page loads the firmware in .hexf format, without adding a composite file, click the merge button, Hint: [HEXF Merge] Fail:+loaded file path

2.Two .hexf file synthesis steps

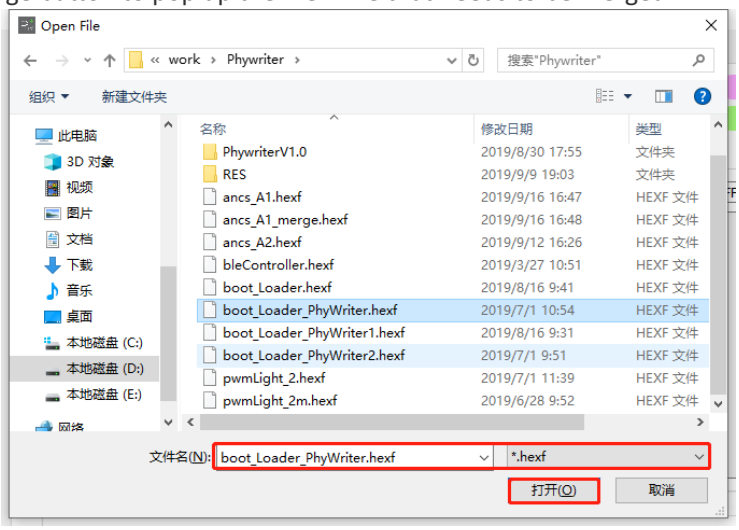
1） Load the application firmware .hexf file on the HEX page, and correspondingly modify the RUN_ADDR of the firmware running, PHY6202--1FFF4000, PHY6212–1FFF4800.



2) setting—configuration Set the Boot_Loader_Merge check box, after checking, click OK



3) Click the merge button to pop up the .hexf file that needs to be merged



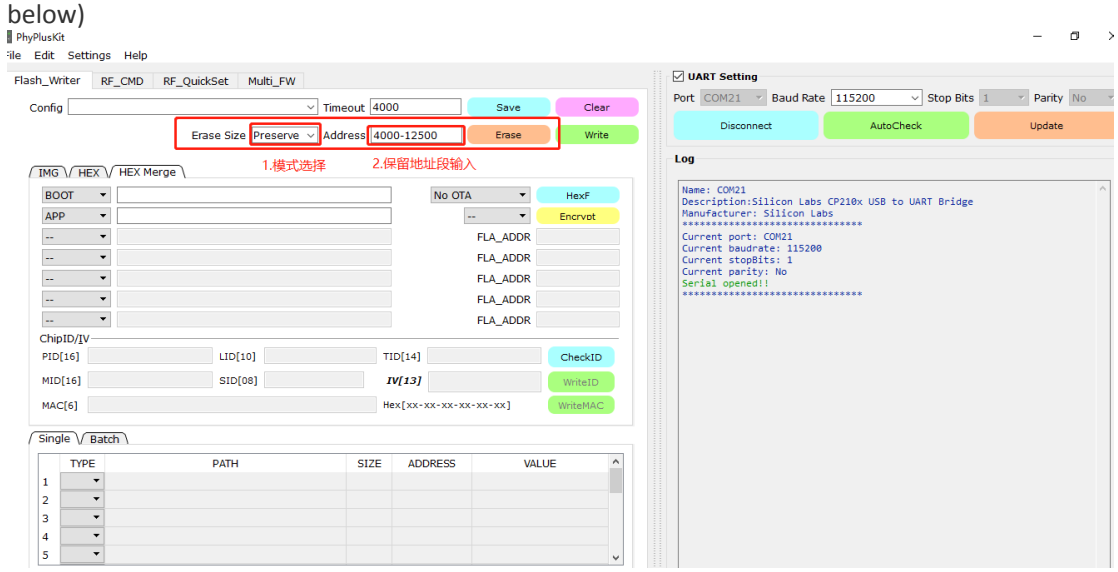
4） After clicking to open, the interface LOG prompts: [HEXF Merge] Success: + loaded firmware path _merge.hexf

3.7.5. Preserve mode address segment retention and erasure of multiple address segments

Since V2.3.8a, the Preserve mode is added to the Erase erase function, which can realize the flash area reservation of a certain address segment and the direct erase function of multiple specified address segments. The specific performance is as follows:

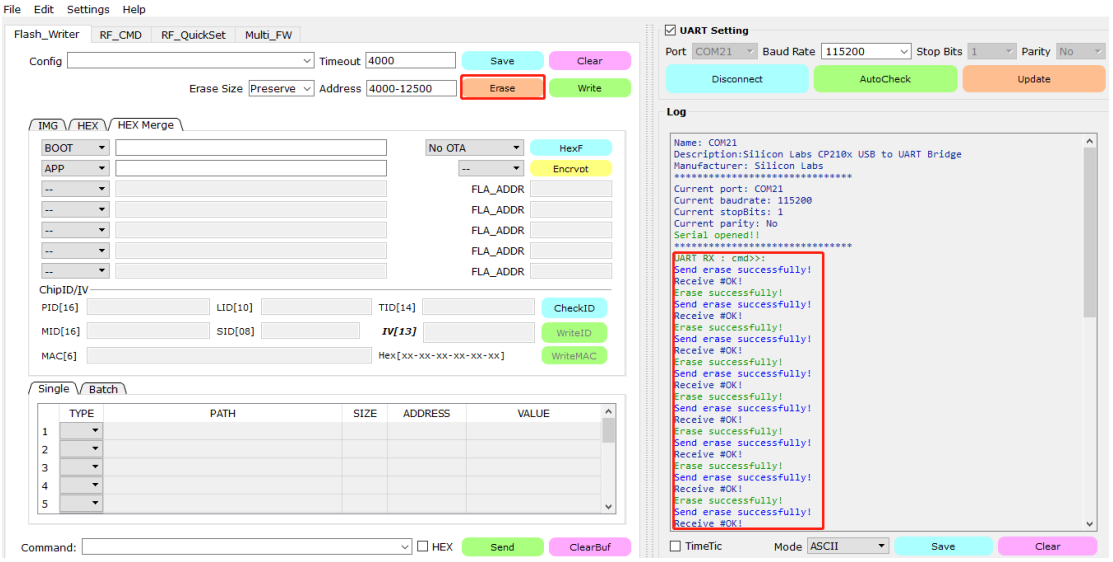
1. Preserve a address segment （Preserve）

A · Preserve mode selection and reserved address segment input (eg: 4000-12500 as below)



B. Pull high TM pin, reset the EVK

C. Click Erase button, erase all flash memory except the segment 4000-12500



2.Erase multiple address segments in preserve mode (ie: 2000~5000,8000~12500)

Remarks: "," in the middle of multiple address segments must be input in English characters. Only the flash content of the input multiple address segments will be erased

3.7.6. PHY_fct_Mode feature

Starting from version V2.3.8b, PhyPlusKit supports FCT mode of PRBMD00 products. When the chip enters FCT mode, it cannot read and write registers in the programming mode, which can ensure the security of the program. The specific operation process is as follows:

There are two paths to specifically support the FCT mode:

1. The PHY_fct_Mode button on the main interface is convenient for users to click to enter FCT mode at any time.

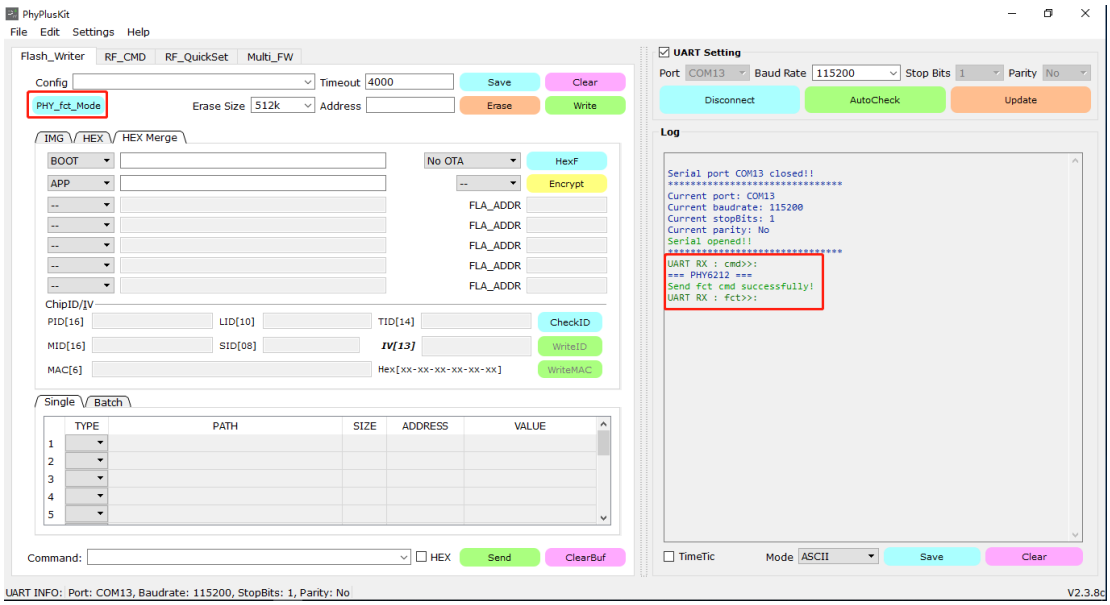
   Steps:

   a. Pull high TM pin，reset device/EVK，LOG returns UART RX : cmd>>:

   b. Click PHY_fct_Mode button, LOG display：

      a) === PHY6212 ===
      b) Send fct cmd successfully!

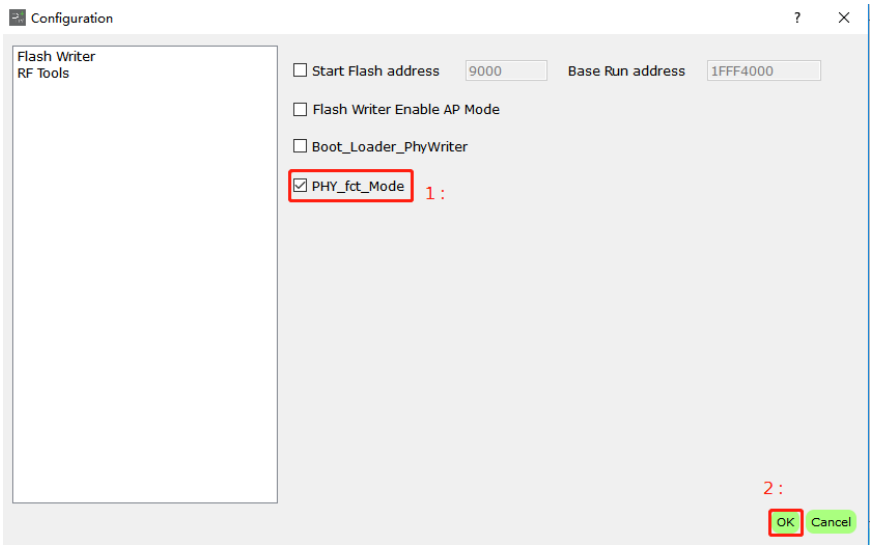
   c. TM is in a high state, reset the development board, log returns to UART RX : fct>>: then enter FCT mode



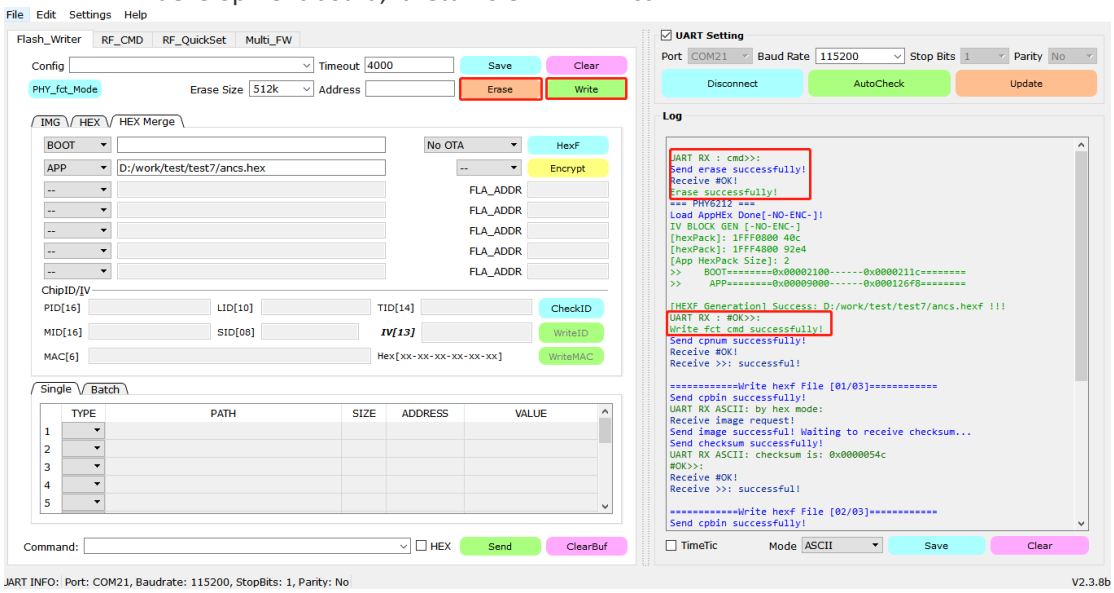
2. Configure FCT mode through setting—configuration:

   Steps

   a. Click setting—configuration, enter the settings of the configuration file.



b. Check the PHY_fct_Mode check box, click OK, this setting will always be saved
c. Pull hight TM pin, reset device or EVK, it returns：UART RX : cmd>>:
d. Click Erase button, erase process
e. Click the Write button to write the firmware, the chip will enter the programming mode, and the log information will be printed and displayed.：
   a) UART RX : #OK>>:
   b) Write fct cmd successfully!
f. At this point, the chip has entered the FCT mode, which can be verified by the reset development board, it returns UART RX : fct>>:



3. Operation to exit FCT mode:
   a) Erase Size is 512K
   b) Click Erase button to erase flash
   c) Reset EVK/device
   d) Quit from fct mode, and it returns UART RX : cmd>>:

Note: PHY_fct_Mode mode currently only supports PRBMD00 chip products.
Method 1: It is convenient for users to quickly enter the FCT mode, click the button to trigger and enter the FCT mode

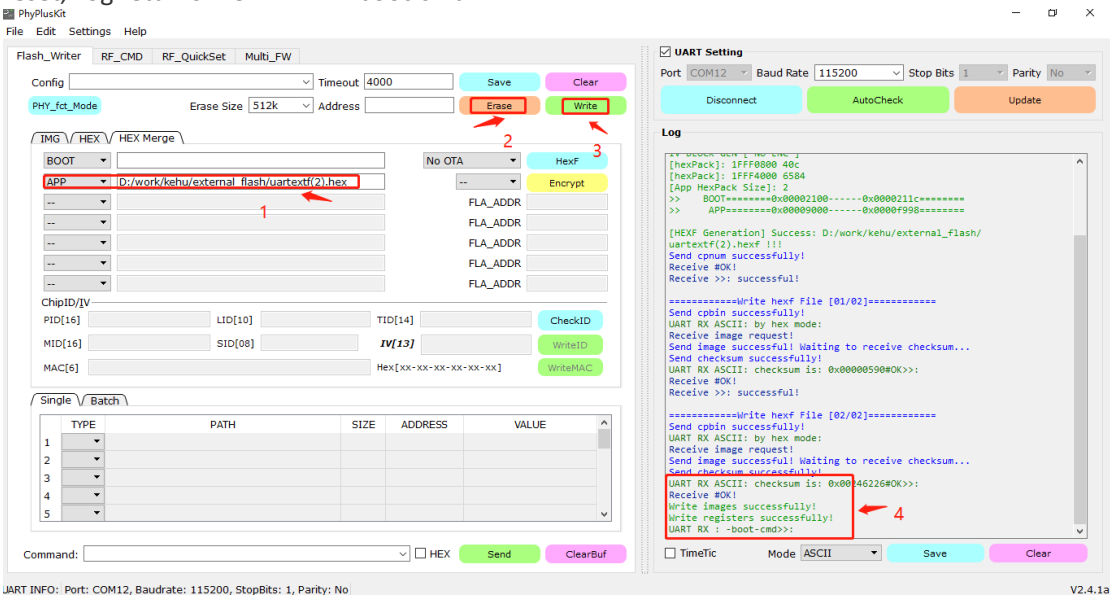
Mode 2: It is convenient for the user to use the FCT mode many times, and it can keep triggering all the time. Note: Every time you reopen the software, you need to reconfigure the check to keep triggering the FCT mode

3.7.7. Support external flash programming function

The main update function of V2.4.1a is to support the programming of external flash. (mainly for PHY6202 products)

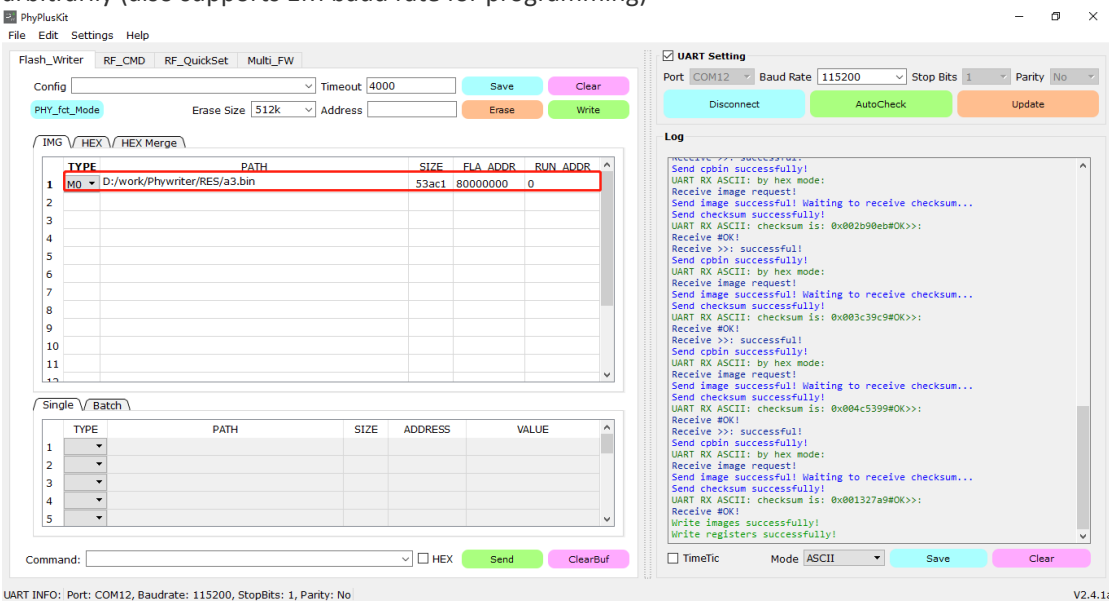
1. Write the boot file that supports external flash -- uartextf.hex file

Pull high TM pin, reset, Erase and then Write; after programming success, pull low TM pin, reset, Log returns：UART RX : -boot-cmd>>:



2.External flash programming

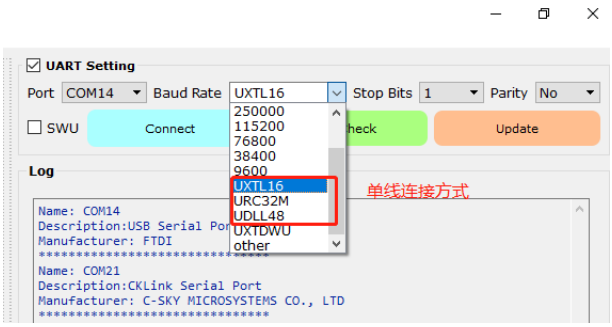
IMG page: Double-click to load the *.bin file in Path; size is the size of the loaded file, which is automatically generated; FLA_ADDR: The flash address of the external flash is based on 80000000, and the value can be increased by itself; RUN_ADDR: The value can be filled in arbitrarily (also supports 1M baud rate for programming)



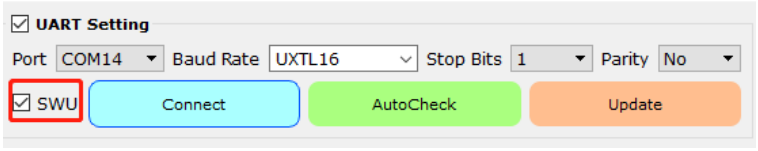
3.7.8. Support single-wire programming function

One of the main update functions of the V2.4.2c version is to support the single-wire programming function, and the P10 port is used to complete the transmission and reception of data respectively.

One-line programming, TM = 0, need to send uart sequence connection through the tool and enter programming mode, there are three main connection methods for single-line programming: "UXTL16" "URC32M" “UDLL48”. Illustrated as below:



Note: The host computer is compatible with both single-line and dual-line programming modes. The single-line and double-line distinction and data processing are performed by checking the SWU control. When it is checked, it is the single-line processing mode, and the default dual-line programming is not checked. choose.

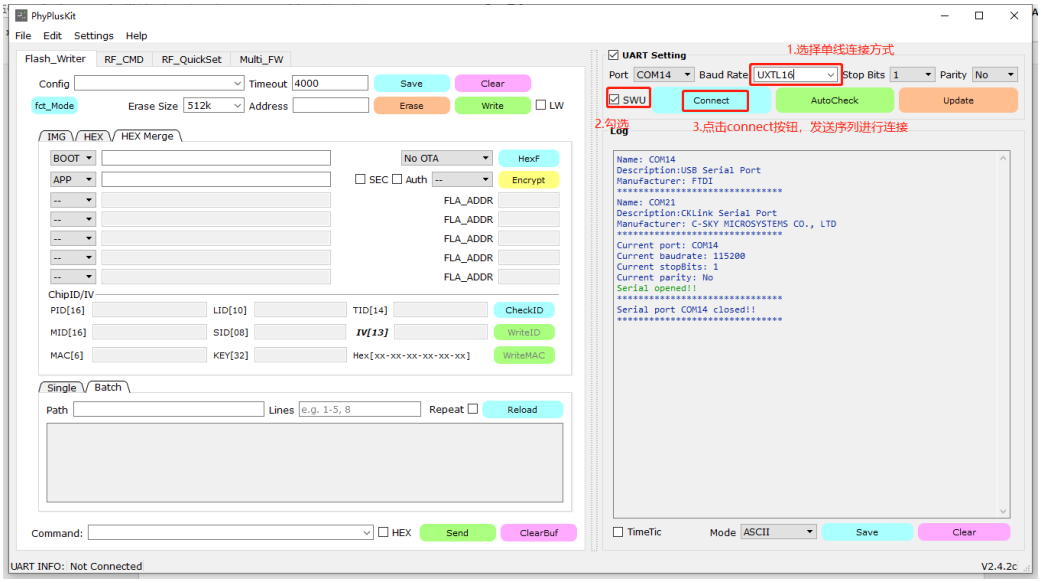


Taking "UXTTL16" as an example, the operation flow of single-wire programming is introduced in detail.

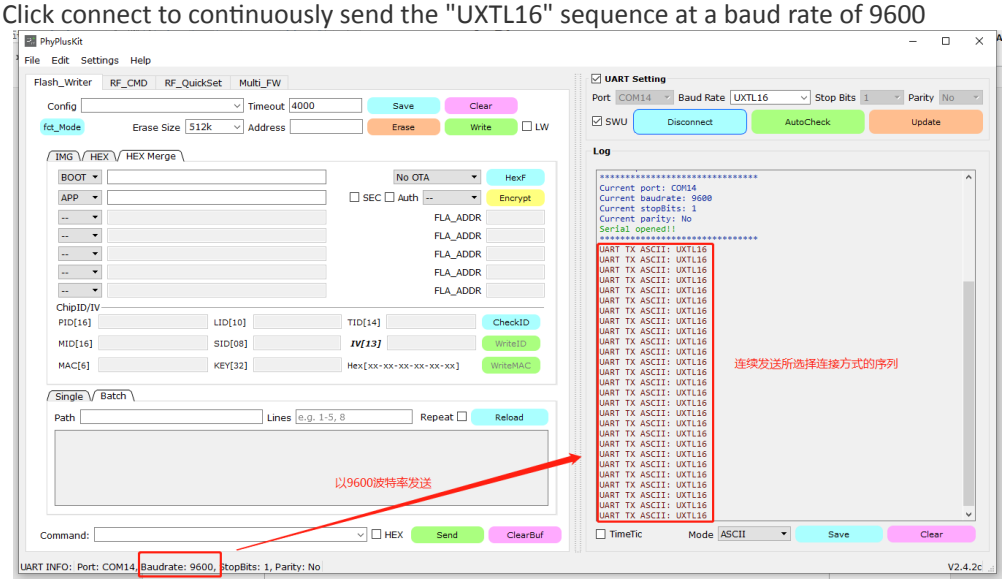
The host computer continuously sends the command at 9600 baud rate, and captures through reset or power-on again. After capturing cmd>>:, the connection is successful and automatically switches to 115200 baud rate.

The detailed process and schematic diagram are as follows:

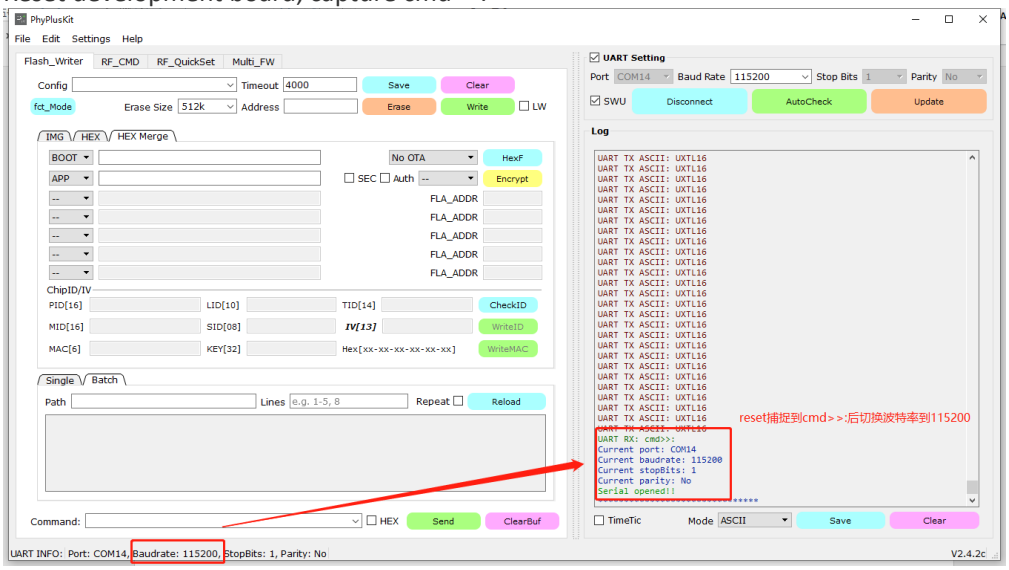
1. TM pull low （TM=0）
2. Select the single-line connection method, check the SWU control, and click the connect button to connect



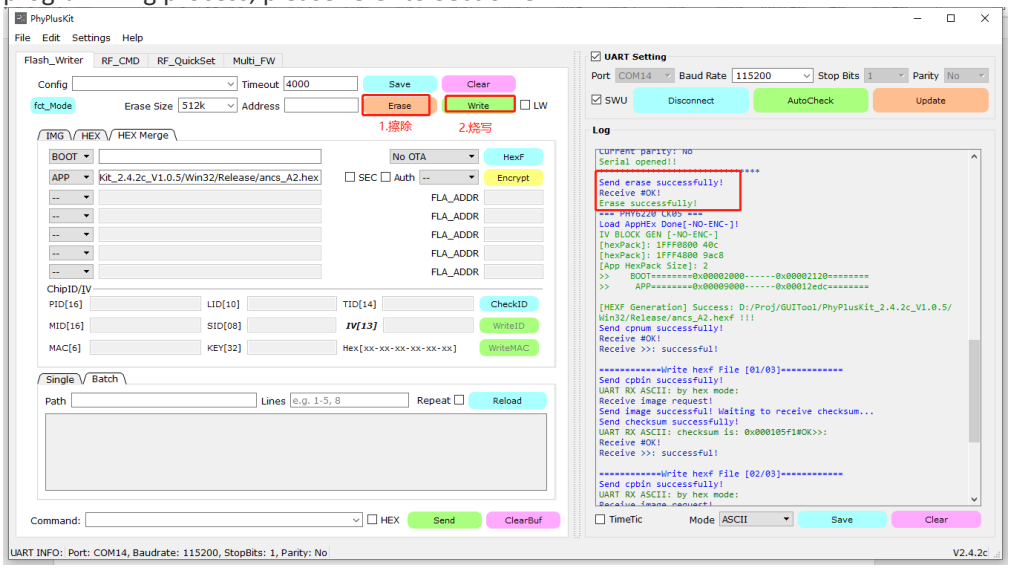
3. Click connect to continuously send the "UXTL16" sequence at a baud rate of 9600



4. Reset development board, capture cmd>>:

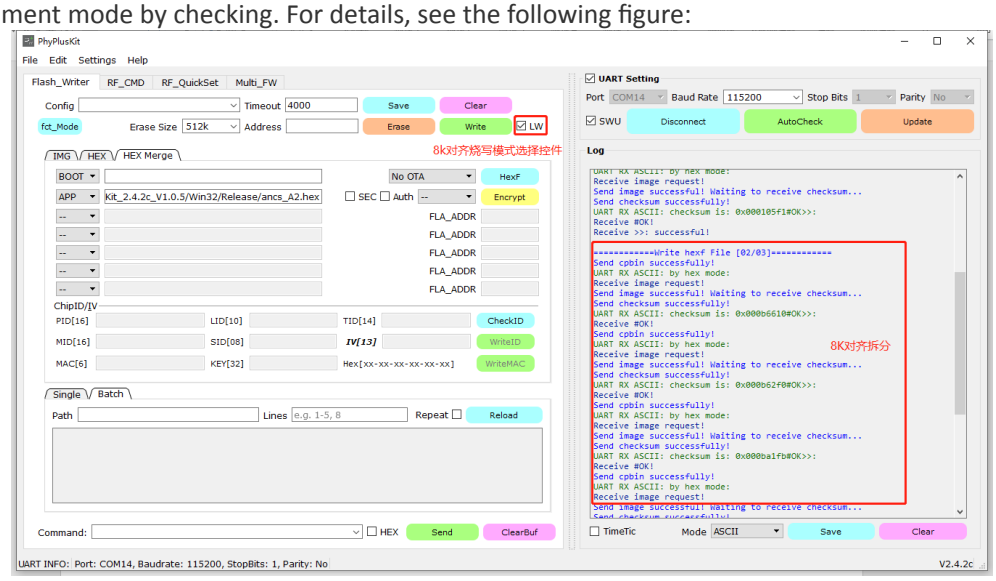


5. Select the APP firmware to be programmed, erase first and then program. For details of the programming process, please refer to Section 3.2



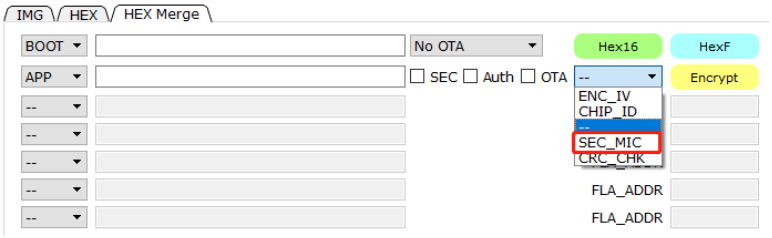
The above is the whole process of single-line programming, which mainly includes two steps: connection and programming.

Considering the different flash programming speeds, the kit tool also supports 8k alignment programming mode, which can be selected through the LW control and can be programmed in 8k alignment mode by checking. For details, see the following figure:

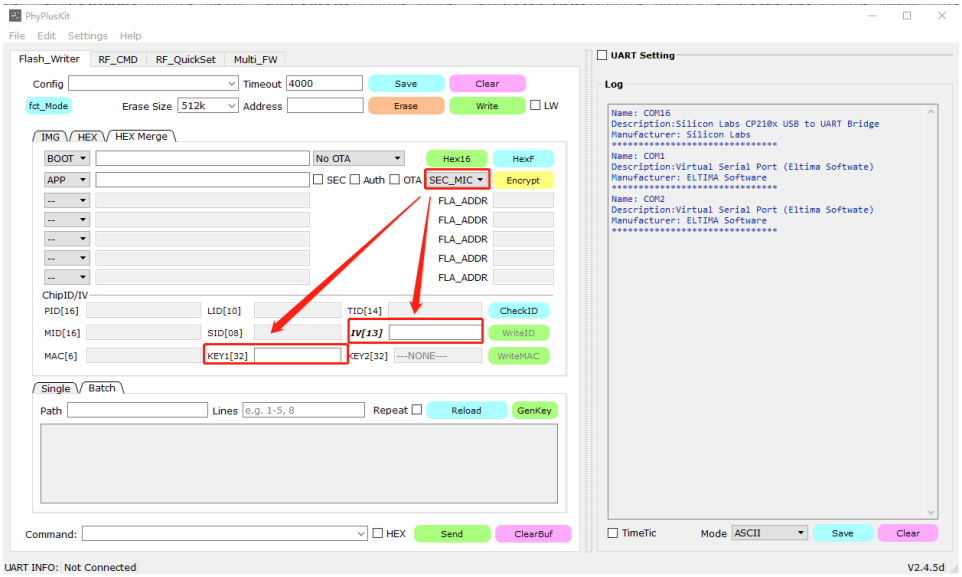


3.7.9. Security boot

After the V2.4.5a version, another function updated by PhyPlusKit is the security boot function, which supports the encrypted boot module function, encrypts the image partition data of the application firmware according to the aes_ccm algorithm, and adopts the secure boot mode. This function module is mainly supported in the SEC_MIC mode selected in the following figure. Select the corresponding SEC_MIC form to use the security boot function. The specific selection is as follows:

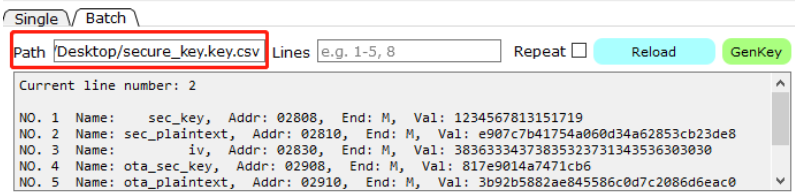


When the SEC_MIC control mode is selected, the IV and Key edit boxes are in editable typing state.

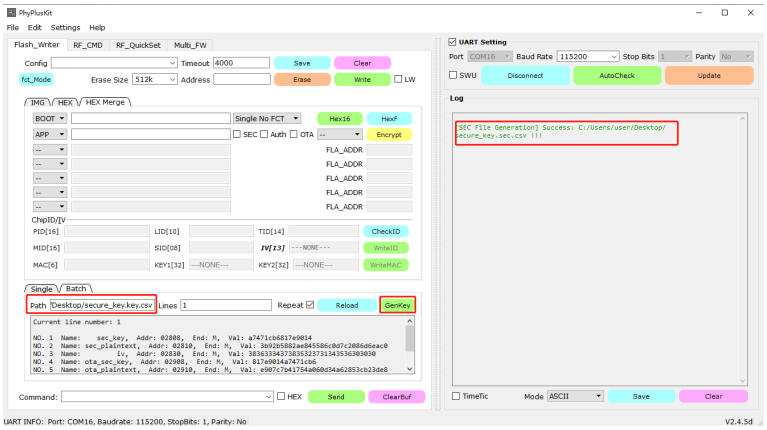


First, the customer is required to provide a series of flash key and efuse key combinations to generate g_sec_key for encryption, that is, the corresponding key to the KEY1[32] position in the figure. The specific operations are as follows:

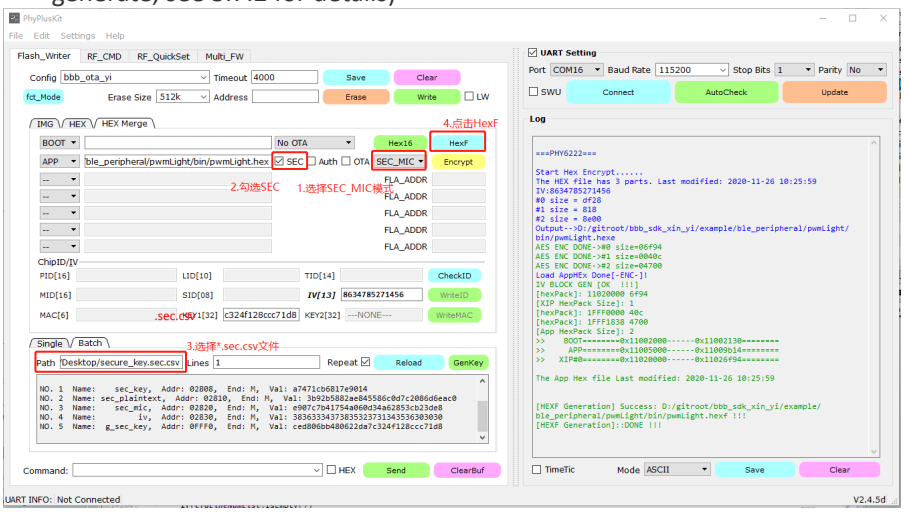
1. The generation method of security boot encrypted g_sec_key is as follows:
   a) Double-click to load the *.key.csv file on the Batch page (be careful to import the .key.csv file type, otherwise an error will be reported)
   b) Correspondingly display the provided flash key and other content, as shown below:



b. Click the GenKey button, and the *.sec.csv file processed by the currently displayed flash key will be generated correspondingly, and the data of the corresponding line (*.sec.csv file) can be generated according to the Lines value filled in.



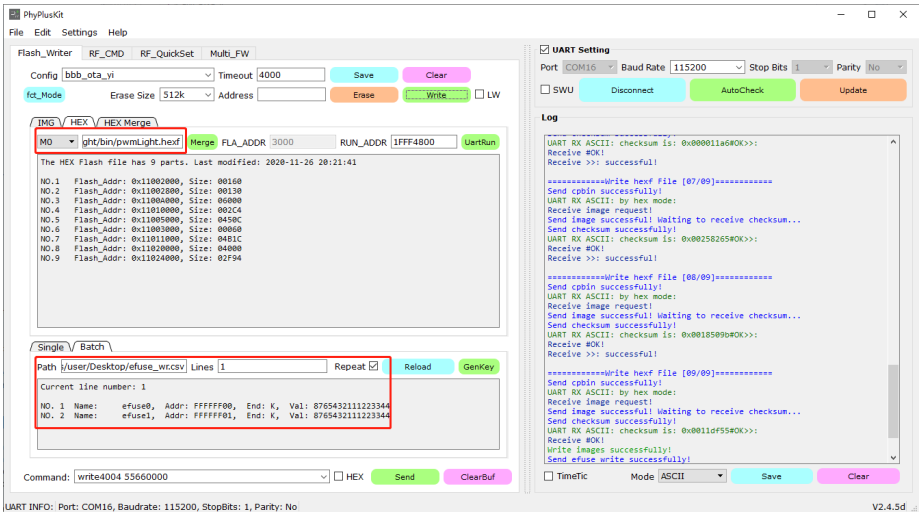
2. Security boot hexf merge process
   a. Select SEC_MIC mode
   b. SEC control selection (AUTH control is only used for PHY6220 series products, only used to encrypt and decrypt image info in bootloader)
   c. In the Batch page, double-click to select the *.sec.csv file generated by GenKey in 1)
   d. Click the HexF button (Note: when the uart is not connected, it will be generated according to the corresponding chip model; after connecting the uart, it must be in the burning mode, that is, cmd>>: mode, and click the HexF button to correctly generate, see 3.7.2 for details)



The values in KEY1[32] and IV[13] are keyed by parsing the *.sec.csv file and do not need to be entered manually.

3. PhyPlusKit programming process

In addition to the .hexf ciphertext generated by the above configuration, Security boot programming also needs to program the efuse key corresponding to the encrypted g_sec_key key. The specific operations are as follows:



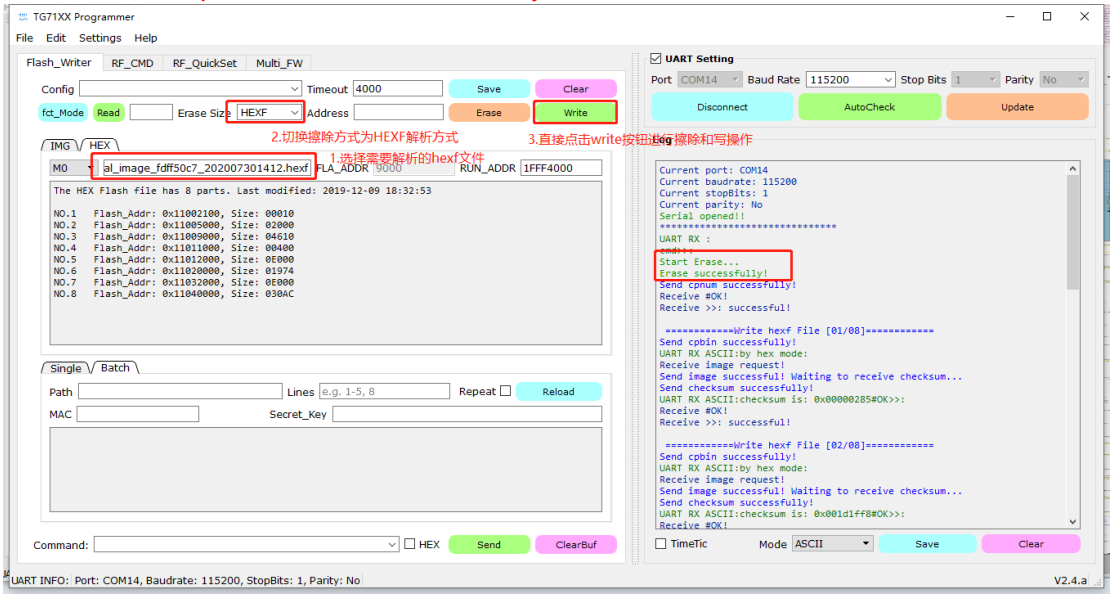
After the programming is successful, power on again to complete the safe boot steps

The offline programmer needs to provide the hexf file generated in the above step 2) and the triple csv file of the corresponding efuse key.

3.7.10. Retain Erase Mode for HEXF Parsing

Since V2.4.5a, the HEXF erasing mode is added to the Erase erasing function. By analyzing the content of the hexf file that needs to be written to the flash, it can selectively erase the address segment of the flash that needs to be written, and the remaining locations are not erased. Operation, the specific operation steps are as follows:

1. Select the corresponding hexf file to be parsed. Note that it must be a hexf file, and the flash address can be correctly parsed. If there is no selection, there will be a corresponding prompt. If the format is wrong, the erasing method will not take effect.
2. Select the erasure method of HEXF parsing
3. Click the Write button to erase and write operations. Note that there is no need to manually click the erase button here, just click the Write button.



The specific operation process is shown in the figure above. There will be prompts for erasing operations such as Start erase... in the log area.