MPCLI User Guide

V1.6
2024/07/26

Realtek Semiconductor Corp.
No.2, Innovation Road II, Hsinchu Science Park, Hsinchu 300, Taiwan
Tel.: +886-3-578-0211. Fax: +886-3-577-6047
www.realtek.com

USING THIS DOCUMENT

This document is intended for the software engineer's reference and provides detailed programming information. Though every effort has been made to ensure that this document is current and accurate, more information may have become available subsequent to the production of this guide.

ELECTROSTATIC DISCHARGE (ESD) WARNING

This product can be damaged by Electrostatic Discharge (ESD). When handling, care must be taken. Damage due to inappropriate handling is not covered by warranty.

Do not open the protective conductive packaging until you have read the following, and are at an approved anti-static workstation.

- Use an approved anti-static mat to cover your work surface
- Use a conductive wrist strap attached to a good earth ground
- Always discharge yourself by touching a grounded bare metal surface or approved anti-static mat before picking up an ESD-sensitive electronic component
- If working on a prototyping board, use a soldering iron or station that is marked as ESD-safe
- Always disconnect the microcontroller from the prototyping board when it is being worked on

Revision History

Date	Version	Comments	Author	Reviewer
2021/08/01	V1.0	Draft	yoyo_yan	
2021/08/24	V1.1	Update	yoyo_yan	
2022/06/06	V1.2	Update	yoyo_yan	
2022/11/08	V1.3	Update	yoyo_yan	
2023/11/15	V1.4	Update	yoyo_yan	
2024/05/15	V1.5	Update	yoyo_yan	
2024/07/26	V1.6	Update	yoyo_yan	

Contents

Table List

Figure List

1 Overview

MPCLI is a command-line download tool. This document describes the basic functions and command formats of the MPCLI. The options it supports are listed in the following table.

Table 1.1 options

option		**function**
h	help	help menu
V	version	show version
c	com	com port, e.g. –c com9
b	baud	set baud, e.g. –b 115200
a	auto	programming binary files from json file automatically
f	json	json file, e.g., -f [json file path]
P	packetimage	packed image, e.g., -P [packet image path]
e	pageerase	erase sector, need to be used with –A and –S together
p	program	program, need to be used with –A and –F together
v	verify	verify, need to be used with –A and –S together
s	savebin	save bin file, need to be used with -A ,-F and -S together
w	window_dump_data	used with –A and –S together, flash data in Command prompt window.
m	modify	modify bytes in sequence, following new values(in hex, separated by colon, maximum 32 bytes), need to be used with -A
A	address	address in Hex, e.g. –A 0x801000
S	size	size, e.g. –S 4096
F	filepath	bin file's path name, e.g. -F D:\\Some\\Path\\To\\name.bin
T	set_xtal_calibration	Set the 40MHz XTAL Internal Cap Calibration value.
	set_tx_power	modify TX Power value, unit:dBm,-20,0,2,3,4,7.5
x	set_mac	set BT address, e.g., -x 11:22:33:DD:EE:FF
n	productid	product ID，4bytes，need to be used with -x and -k together
k	secretkey	secret key, 32bytes, need to be used with -x and -n together
u	efuse_json	programming eFuse json file (for App encryption)
U	efuse_otp	write eFuse as OTP
I	get_euid_mac	read EUID and MAC
B	get_back_mac	keep original MAC in Flash if download image file.
	get_back_xtal_calibration	keep original 40MHz XTAL Calibrationin Flash if download image file.
M	mandatory	Mandatory option. Selection of program mode, default –M 1
D	debugpassword	debug password in Hex.
E	chiperase	chip erase
C	erase	chip erase but remain 4k(0x800000 - 0x801000)
r	reboot	reboot
	spi_pin_config_json	SPI_PIN config json file path

2 Download Mode

MPCLI supports two ways to download files: the one is by configuring a JSON file, and the other is packed Image.

2.1 JSON file mode

Users need to customize the JSON file, such as port number, baud, files' name, address and so on.The zip of MPCLI release contains a configuration file -- mptoolconfig.json, as is shown in Figure 2.1. Users need to fill in the correct information, fill the port number in the "port" field, set the baud rate in "baud" and its default value is 1Mbps, its maximum is 2Mbps. Write the image bin storage path at the "relativepath" filed, can use the default path "bin". The "file" field used to configure information about each file to burn, "name" indicates the image name, "address" indicates the location where the image is written to flash. "Enable" used to select whether to download this image, and can be set to either "1" or "0", when JSON file does not contain "Enable", it is equivalent to setting "Enable":"1". Users can delete or add items, like {"id": 8,"address": "0x00920000","name": "null"} or {"id": 8,"address": "0x00920000","name": "null","enable":"1"}. It's important to note that each new item also be separated by commas, otherwise MPCLI will fail to parse the JSON.

```
{
    "mptoolconfig": {"port": "COM6","baud": "1000000",
        "appimage": {"relativepath": "bin",
            "file": [{"id": 0,"address": "0x00801000","name": "null","enable":"1"},
                     {"id": 1,"address": "0x00802000","name": "null","enable":"1"},
                     {"id": 2,"address": "0x00803000","name": "null","enable":"1"},
                     {"id": 3,"address": "0x0080E000","name": "null","enable":"1"},
                     {"id": 4,"address": "0x00840000","name": "null","enable":"0"},
                     {"id": 5,"address": "0x00840000","name": "null","enable":"0"},
                     {"id": 6,"address": "0x008F0000","name": "null","enable":"0"},
                     {"id": 7,"address": "0x00920000","name": "null","enable":"0"}]
        }
    }
}
```

Figure 2.1 JSON file format

The format of the command in JSON burning mode: mpcli.exe -f mptoolconfig.json –a [option].

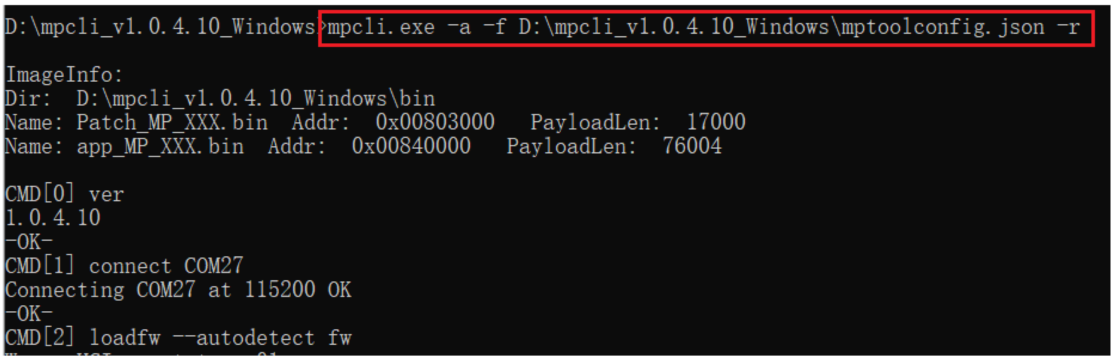
-f: specify the path and name of the JSON file.

-a: mandatory. Indicates that this command will burn all compliance image configured in JSON.

[option]: users can add options according to needs. Like –r option, when download all image, execute IC Reboot. See Chapter 3 for more details on functional options.

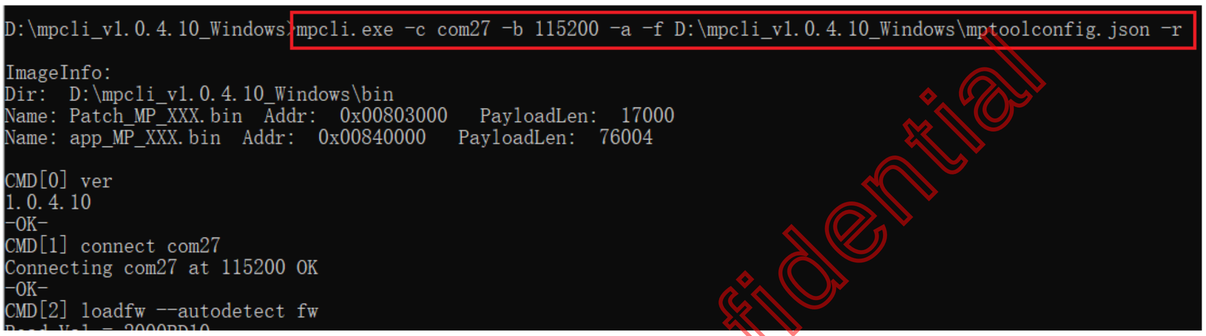
Additional instructions, users need to ensure that the configurations in the JSON file are correct; otherwise, parsing errors may occur. By the way, users can configure the "port" and "baud" in JSON file, like Figure 2.2; and can also configure them by options, like Figure 2.3, this behavior will ignore "port" or "baud" in the JSON file.

Steps: (1) make sure configured port number or baud correctly; (2) fill in JSON file; (3) copy image files to the destination folder as configured in JSON file; (4) get command-line, and download.



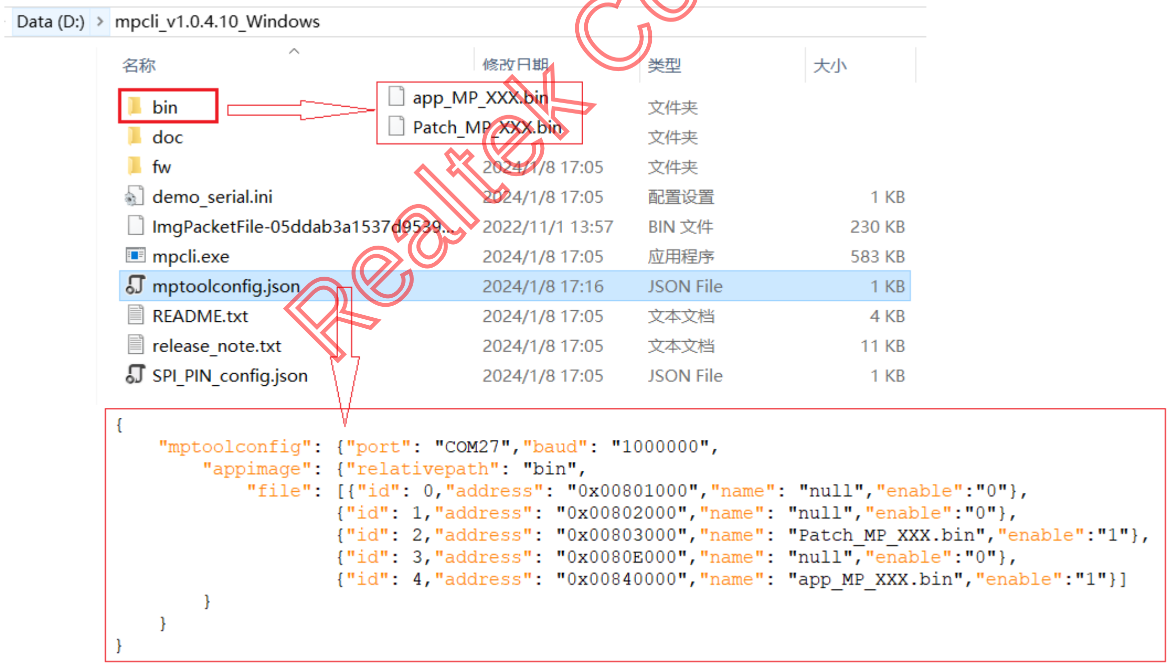
```
D:\mpcli_v1.0.4.10_Windows>mpcli.exe -a -f D:\mpcli_v1.0.4.10_Windows\mptoolconfig.json -r

ImageInfo:
Dir:  D:\mpcli_v1.0.4.10_Windows\bin
Name: Patch_MP_XXX.bin  Addr:  0x00803000   PayloadLen:  17000
Name: app_MP_XXX.bin  Addr:  0x00840000   PayloadLen:  76004

CMD[0] ver
1.0.4.10
-OK-
CMD[1] connect COM27
Connecting COM27 at 115200 OK
-OK-
CMD[2] loadfw --autodetect fw
```

Figure 2.2 JSON file mode command (a)



```
D:\mpcli_v1.0.4.10_Windows>mpcli.exe -c com27 -b 115200 -a -f D:\mpcli_v1.0.4.10_Windows\mptoolconfig.json -r

ImageInfo:
Dir:  D:\mpcli_v1.0.4.10_Windows\bin
Name: Patch_MP_XXX.bin  Addr:  0x00803000   PayloadLen:  17000
Name: app_MP_XXX.bin  Addr:  0x00840000   PayloadLen:  76004

CMD[0] ver
1.0.4.10
-OK-
CMD[1] connect com27
Connecting com27 at 115200 OK
-OK-
CMD[2] loadfw --autodetect fw
```

Figure 2.3 JSON file mode command (b)



Data (D:) > mpcli_v1.0.4.10_Windows

名称	修改日期	类型	大小
bin		文件夹	
doc		文件夹	
fw	2024/1/8 17:05	文件夹	
demo_serial.ini	2024/1/8 17:05	配置设置	1 KB
ImgPacketFile-05ddab3a1537d9539...	2022/11/1 13:57	BIN 文件	230 KB
mpcli.exe	2024/1/8 17:05	应用程序	583 KB
mptoolconfig.json	2024/1/8 17:16	JSON File	1 KB
README.txt	2024/1/8 17:05	文本文档	4 KB
release_note.txt	2024/1/8 17:05	文本文档	11 KB
SPI_PIN_config.json	2024/1/8 17:05	JSON File	1 KB

app_MP_XXX.bin
Patch_MP_XXX.bin

```
{
    "mptoolconfig": {"port": "COM27","baud": "1000000",
        "appimage": {"relativepath": "bin",
            "file": [{"id": 0,"address": "0x00801000","name": "null","enable":"0"},
                    {"id": 1,"address": "0x00802000","name": "null","enable":"0"},
                    {"id": 2,"address": "0x00803000","name": "Patch_MP_XXX.bin","enable":"1"},
                    {"id": 3,"address": "0x0080E000","name": "null","enable":"0"},
                    {"id": 4,"address": "0x00840000","name": "app_MP_XXX.bin","enable":"1"}]
        }
    }
}
```

Figure 2.4 JSON File Demo

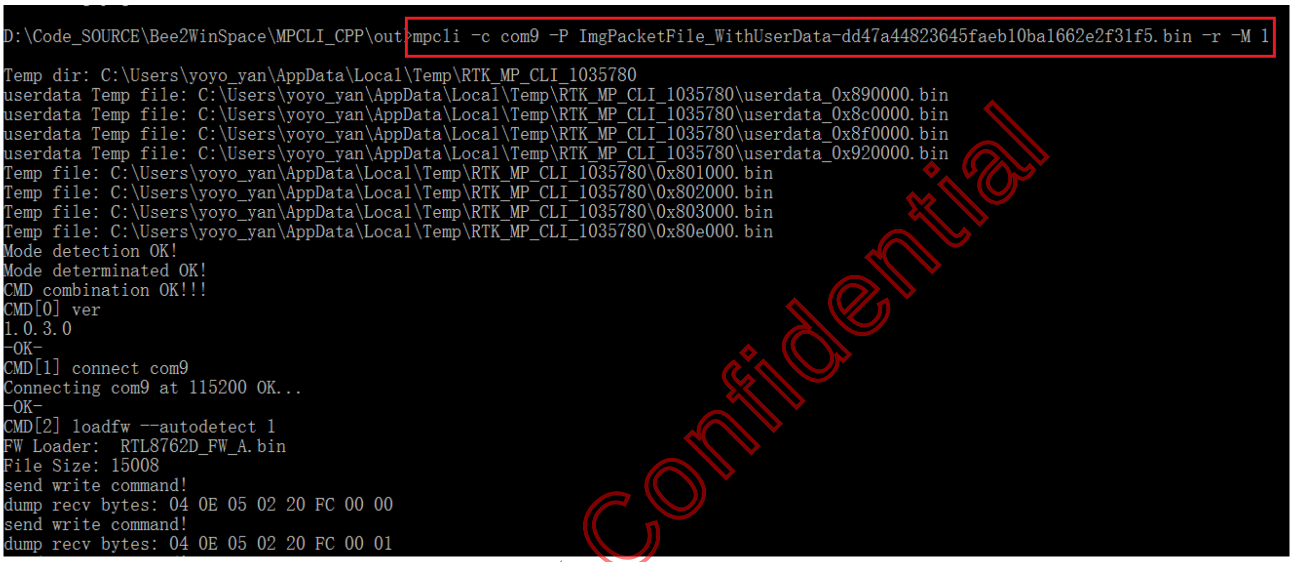
2.2 Packed Image mode

Packed image is generated by Pack Tool. For the specific use of Pack Tool, please refer to the user guide of Pack Tool.

This way is more convenient for users, just configure port number by “-c”, and default baud is 1Mbps, configure it by “-b”; use “-P” to specify the packed image path and name, it could be an absolute path or a relative path. The following Figure 2.5 shows details.

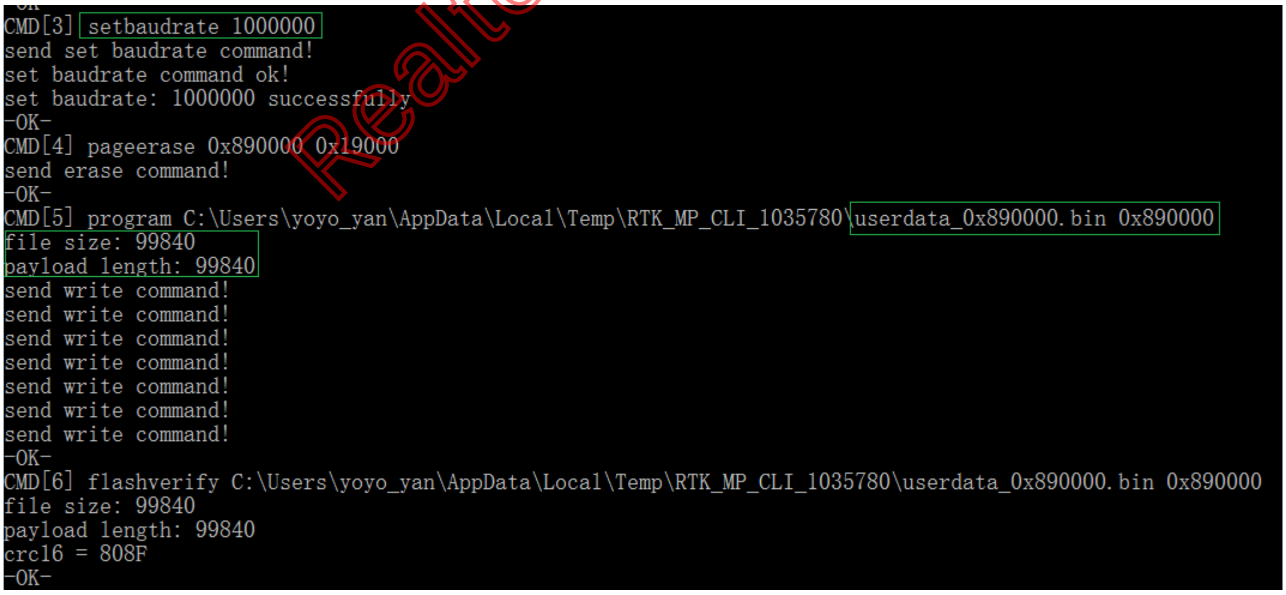
The tool execution process is as follows: a. parse packed image, and save it locally; b. burn individual image in turn. When MPCLI starts running, image details’ info are printed, such as burn address, users can check whether the selected Packed Image is correct.

Now MPCLI supports to download packed image with user data, command as shown in Figure 2.6.



```
D:\Code_SOURCE\Bee2WinSpace\MPCLI_CPP\out>mpcli -c com9 -P ImgPacketFile_WithUserData-dd47a44823645faeb10ba1662e2f31f5.bin -r -M 1

Temp dir: C:\Users\yoyo_yan\AppData\Local\Temp\RTK_MP_CLI_1035780
userdata Temp file: C:\Users\yoyo_yan\AppData\Local\Temp\RTK_MP_CLI_1035780\userdata_0x890000.bin
userdata Temp file: C:\Users\yoyo_yan\AppData\Local\Temp\RTK_MP_CLI_1035780\userdata_0x8c0000.bin
userdata Temp file: C:\Users\yoyo_yan\AppData\Local\Temp\RTK_MP_CLI_1035780\userdata_0x8f0000.bin
userdata Temp file: C:\Users\yoyo_yan\AppData\Local\Temp\RTK_MP_CLI_1035780\userdata_0x920000.bin
Temp file: C:\Users\yoyo_yan\AppData\Local\Temp\RTK_MP_CLI_1035780\0x801000.bin
Temp file: C:\Users\yoyo_yan\AppData\Local\Temp\RTK_MP_CLI_1035780\0x802000.bin
Temp file: C:\Users\yoyo_yan\AppData\Local\Temp\RTK_MP_CLI_1035780\0x803000.bin
Temp file: C:\Users\yoyo_yan\AppData\Local\Temp\RTK_MP_CLI_1035780\0x80e000.bin
Mode detection OK!
Mode determinated OK!
CMD combination OK!!!
CMD[0] ver
1.0.3.0
-OK-
CMD[1] connect com9
Connecting com9 at 115200 OK...
-OK-
CMD[2] loadfw --autodetect 1
FW Loader:  RTL8762D_FW_A.bin
File Size: 15008
send write command!
dump recv bytes: 04 0E 05 02 20 FC 00 00
send write command!
dump recv bytes: 04 0E 05 02 20 FC 00 01
```

Figure 2.5 Packed Image mode Command



```
CMD[3] setbaudrate 1000000
send set baudrate command!
set baudrate command ok!
set baudrate: 1000000 successfully
-OK-
CMD[4] pageerase 0x890000 0x19000
send erase command!
-OK-
CMD[5] program C:\Users\yoyo_yan\AppData\Local\Temp\RTK_MP_CLI_1035780\userdata_0x890000.bin 0x890000
file size: 99840
payload length: 99840
send write command!
send write command!
send write command!
send write command!
send write command!
send write command!
send write command!
-OK-
CMD[6] flashverify C:\Users\yoyo_yan\AppData\Local\Temp\RTK_MP_CLI_1035780\userdata_0x890000.bin 0x890000
file size: 99840
payload length: 99840
crc16 = 808F
-OK-
```

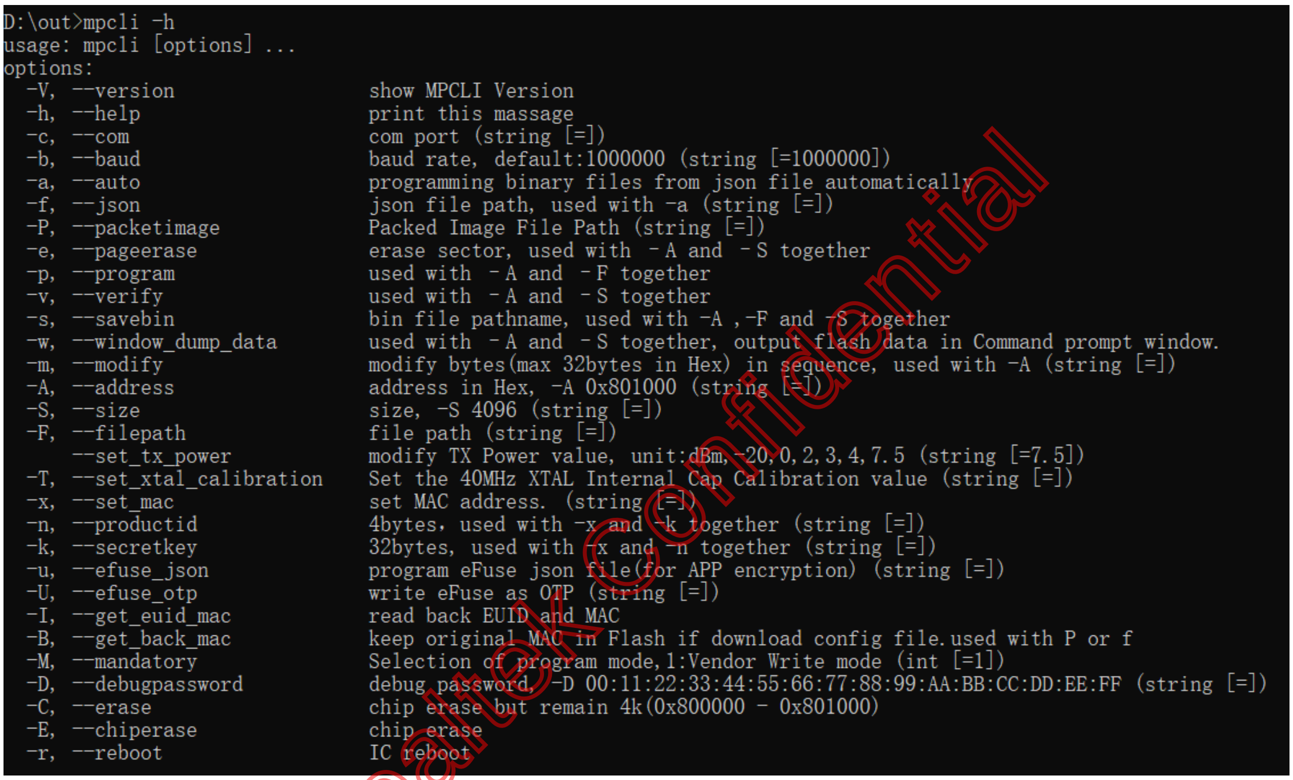
Figure 2.6 support packed image with user data

3 Options Function

Used MPCLI format is as mpcli.exe –option [value]. But some options do not accept value. The following describes the options supported by MPCLI in details.

3.1 Help Menu

Command: mpcli.exe –h, as shown below.



```
D:\out>mpcli -h
usage: mpcli [options] ...
options:
  -V, --version                 show MPCLI Version
  -h, --help                    print this massage
  -c, --com                     com port (string [=])
  -b, --baud                    baud rate, default:1000000 (string [=1000000])
  -a, --auto                    programming binary files from json file automatically
  -f, --json                    json file path, used with -a (string [=])
  -P, --packetimage             Packed Image File Path (string [=])
  -e, --pageerase               erase sector, used with -A and -S together
  -p, --program                 used with -A and -F together
  -v, --verify                  used with -A and -S together
  -s, --savebin                 bin file pathname, used with -A ,-F and -S together
  -w, --window_dump_data        used with -A and -S together, output flash data in Command prompt window.
  -m, --modify                  modify bytes(max 32bytes in Hex) in sequence, used with -A (string [=])
  -A, --address                 address in Hex, -A 0x801000 (string [=])
  -S, --size                    size, -S 4096 (string [=])
  -F, --filepath                file path (string [=])
      --set_tx_power            modify TX Power value, unit:dBm,-20,0,2,3,4,7.5 (string [=7.5])
  -T, --set_xtal_calibration    Set the 40MHz XTAL Internal Cap Calibration value (string [=])
  -x, --set_mac                 set MAC address. (string [=])
  -n, --productid               4bytes, used with -x and -k together (string [=])
  -k, --secretkey               32bytes, used with -x and -n together (string [=])
  -u, --efuse_json              program eFuse json file(for APP encryption) (string [=])
  -U, --efuse_otp               write eFuse as OTP (string [=])
  -I, --get_euid_mac            read back EUID and MAC
  -B, --get_back_mac            keep original MAC in Flash if download config file.used with P or f
  -M, --mandatory               Selection of program mode,1:Vendor Write mode (int [=1])
  -D, --debugpassword           debug password, -D 00:11:22:33:44:55:66:77:88:99:AA:BB:CC:DD:EE:FF (string [=])
  -C, --erase                   chip erase but remain 4k(0x800000 - 0x801000)
  -E, --chiperase               chip erase
  -r, --reboot                  IC reboot
```

Figure 3.1 Help Menu

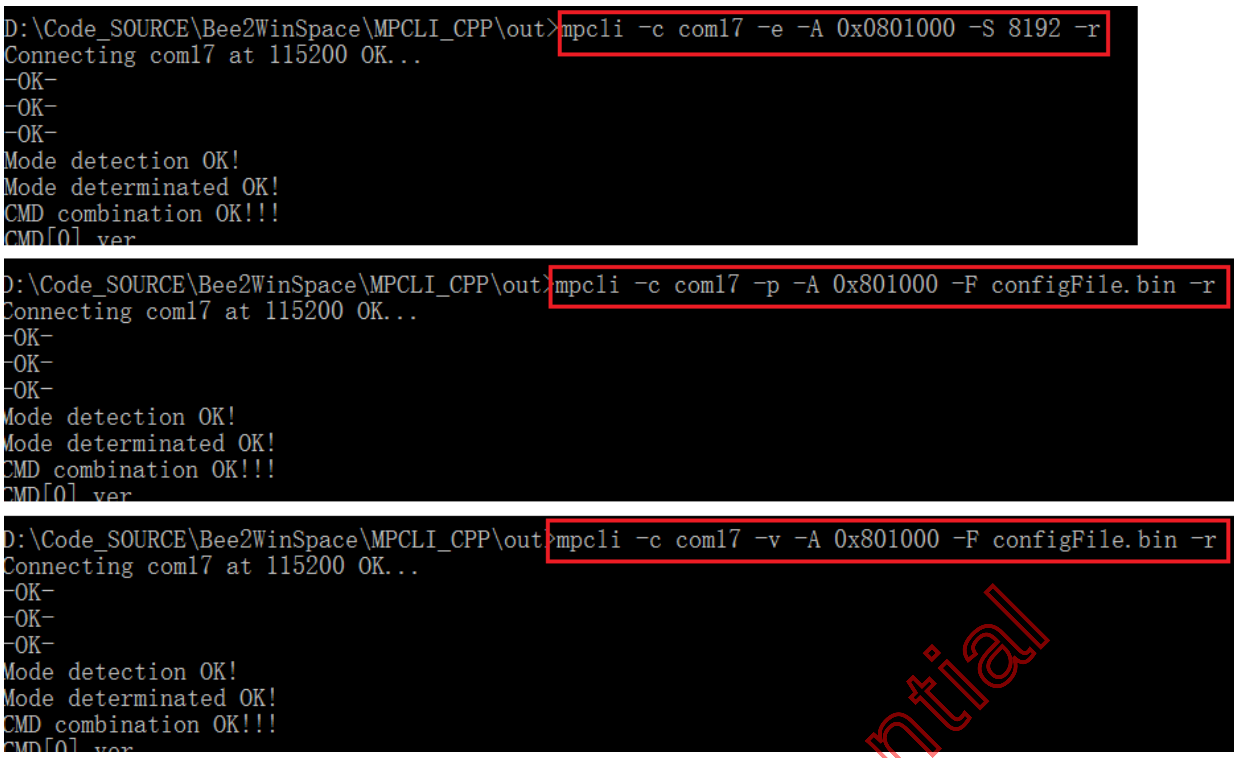
3.2 Burn Single File

Burn a separate binary file, as shown in Figure 3.2

Firstly, erase sector. –e means that the command performs erase. –A means erase address; -S means erase size aligned to 4K.

Secondly, program. –p means that this command performs program. –A means program address; -F indicates the path of single bin file to be burned.

Lastly, verify. –v means that this command performs verity. –A verify address; -F indicates the path of single bin file to be burned.

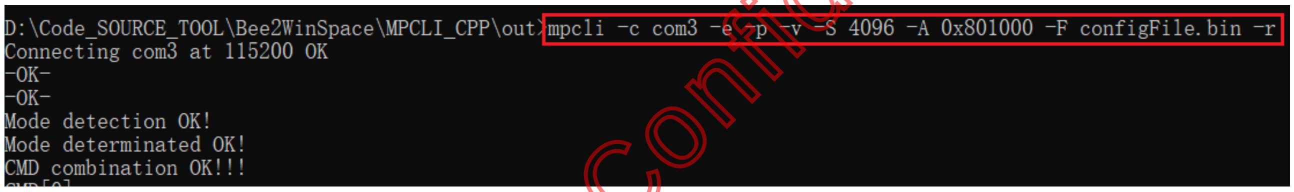


```
D:\Code_SOURCE\Bee2WinSpace\MPCLI_CPP\out>mpcli -c com17 -e -A 0x0801000 -S 8192 -r
Connecting com17 at 115200 OK...
-OK-
-OK-
-OK-
Mode detection OK!
Mode determinated OK!
CMD combination OK!!!
CMD[0] ver
```

```
D:\Code_SOURCE\Bee2WinSpace\MPCLI_CPP\out>mpcli -c com17 -p -A 0x801000 -F configFile.bin -r
Connecting com17 at 115200 OK...
-OK-
-OK-
-OK-
Mode detection OK!
Mode determinated OK!
CMD combination OK!!!
CMD[0] ver
```

```
D:\Code_SOURCE\Bee2WinSpace\MPCLI_CPP\out>mpcli -c com17 -v -A 0x801000 -F configFile.bin -r
Connecting com17 at 115200 OK...
-OK-
-OK-
-OK-
Mode detection OK!
Mode determinated OK!
CMD combination OK!!!
CMD[0] ver
```

Figure 3.2 Burn Single File Command (1)



```
D:\Code_SOURCE_TOOL\Bee2WinSpace\MPCLI_CPP\out>mpcli -c com3 -e -p -v -S 4096 -A 0x801000 -F configFile.bin -r
Connecting com3 at 115200 OK
-OK-
-OK-
Mode detection OK!
Mode determinated OK!
CMD combination OK!!!
CMD[0]
```

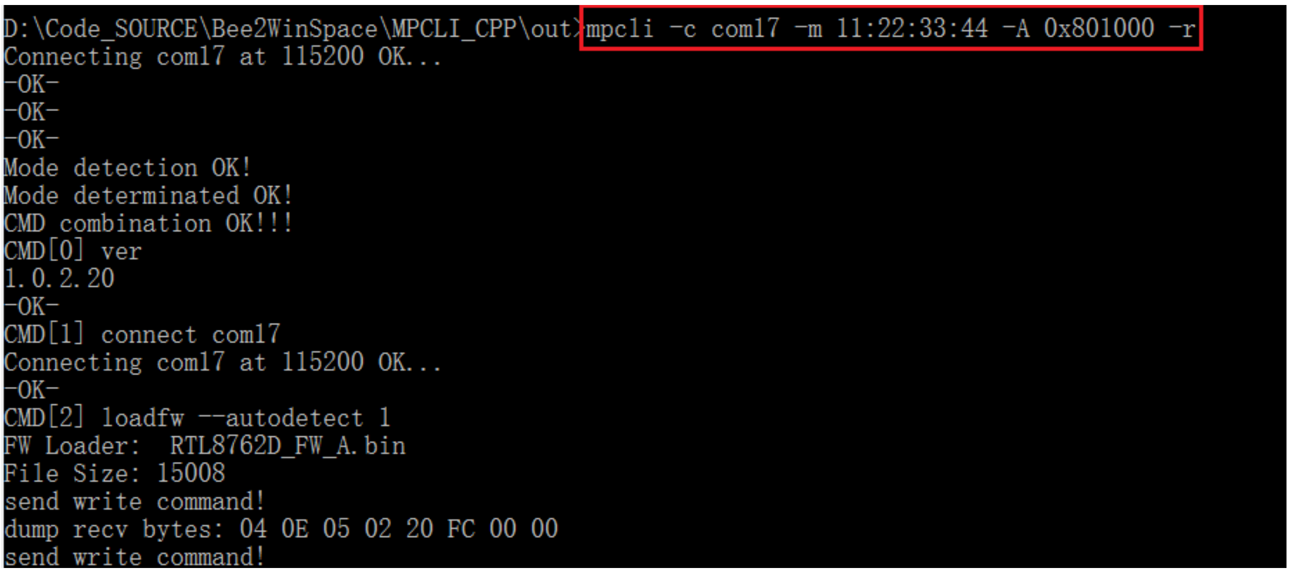
Figure 3.3 Burn Single File Command (2)

3.3 Modify Bytes in specified address

From Version1.0.1.13, MPCLI supports –m [value], means that modify sequential bytes starting at the specified address. [value] is a string of hex separated by colons.

As shown in Figure 3.4 changes 4 bytes starting at a specified address 0x801000 to 11 22 33 44.

The maximum length that can be changed at time is 32 bytes.



```
D:\Code_SOURCE\Bee2WinSpace\MPCLI_CPP\out>mpcli -c com17 -m 11:22:33:44 -A 0x801000 -r
Connecting com17 at 115200 OK...
-OK-
-OK-
-OK-
Mode detection OK!
Mode determinated OK!
CMD combination OK!!!
CMD[0] ver
1.0.2.20
-OK-
CMD[1] connect com17
Connecting com17 at 115200 OK...
-OK-
CMD[2] loadfw --autodetect 1
FW Loader:  RTL8762D_FW_A.bin
File Size: 15008
send write command!
dump recv bytes: 04 0E 05 02 20 FC 00 00
send write command!
```

Figure 3.4 Modify Bytes in specified address Command

3.4 Save Flash data to a bin file

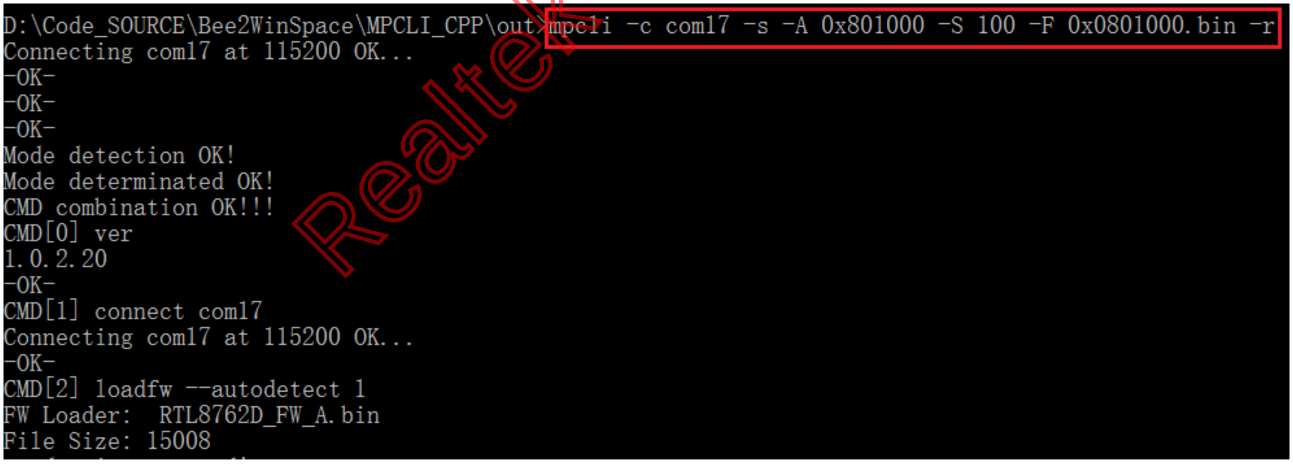
As shown in Figure 3.5,

-s indicates that this command is to save bin file.

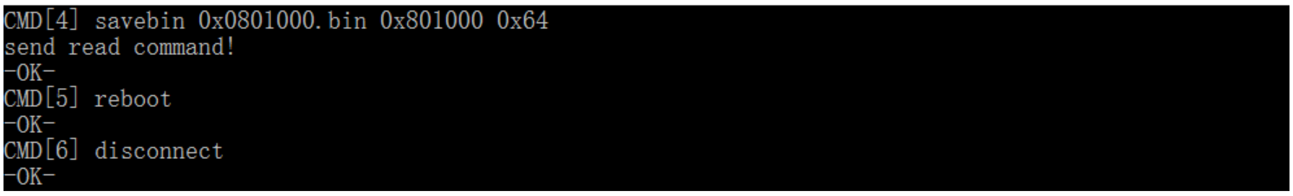
–A [value], [value] indicates the address of bytes to be saved.

-S [value], [value] indicates the number of bytes to be saved.

-F [value], [value] indicates the path and file name to save.



```
D:\Code_SOURCE\Bee2WinSpace\MPCLI_CPP\out>mpcli -c com17 -s -A 0x801000 -S 100 -F 0x0801000.bin -r
Connecting com17 at 115200 OK...
-OK-
-OK-
-OK-
Mode detection OK!
Mode determinated OK!
CMD combination OK!!!
CMD[0] ver
1.0.2.20
-OK-
CMD[1] connect com17
Connecting com17 at 115200 OK...
-OK-
CMD[2] loadfw --autodetect 1
FW Loader:  RTL8762D_FW_A.bin
File Size: 15008
```



```
CMD[4] savebin 0x0801000.bin 0x801000 0x64
send read command!
-OK-
CMD[5] reboot
-OK-
CMD[6] disconnect
-OK-
```

Figure 3.5 Save bin file Command

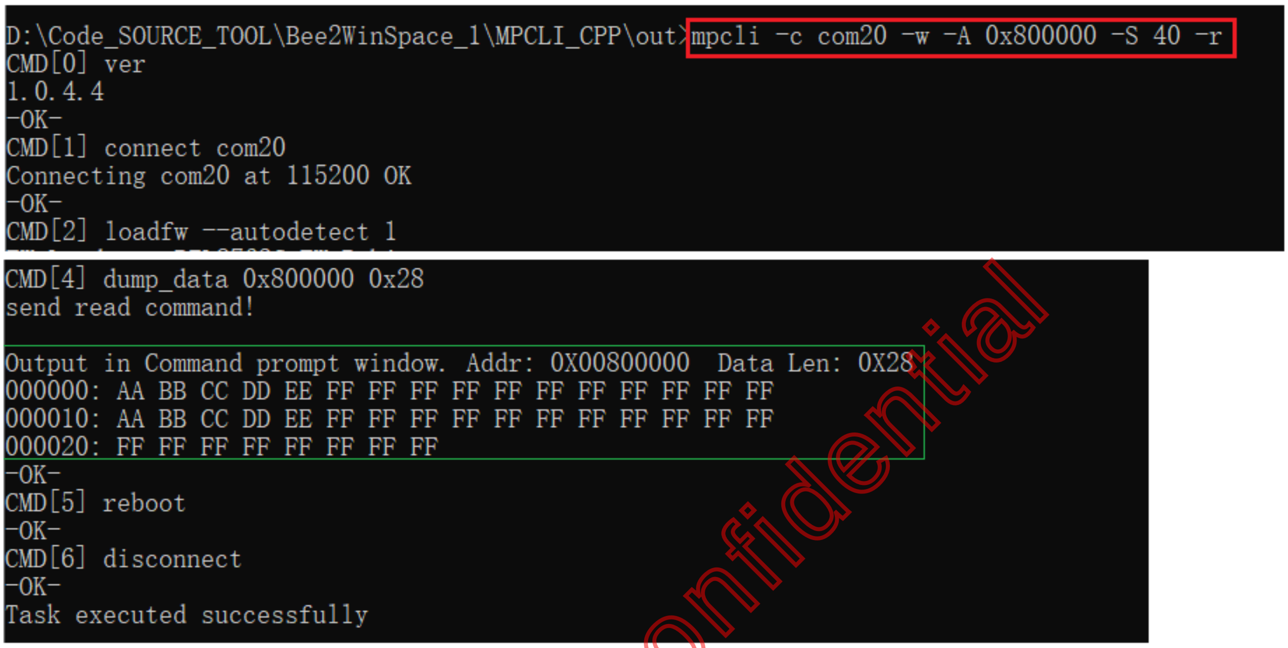
3.5 Dump Flash data in Command prompt window

As shown in Figure 3.6,

-w indicates that this command is to dump flash data in command prompt window.

-A [value], [value] indicates the address of bytes to be saved.

-S [value], [value] indicates the number of bytes to be saved.



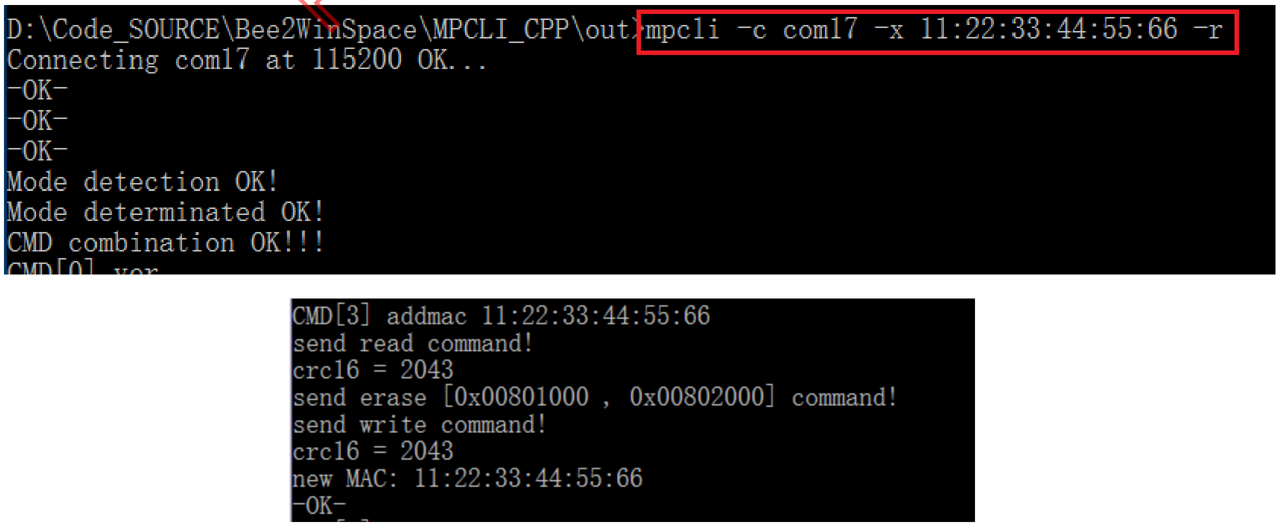
```
D:\Code_SOURCE_TOOL\Bee2WinSpace_1\MPCLI_CPP\out>mpcli -c com20 -w -A 0x800000 -S 40 -r
CMD[0] ver
1.0.4.4
-OK-
CMD[1] connect com20
Connecting com20 at 115200 OK
-OK-
CMD[2] loadfw --autodetect 1
```

```
CMD[4] dump_data 0x800000 0x28
send read command!

Output in Command prompt window. Addr: 0X00800000  Data Len: 0X28
000000: AA BB CC DD EE FF FF FF FF FF FF FF FF FF FF FF
000010: AA BB CC DD EE FF FF FF FF FF FF FF FF FF FF FF
000020: FF FF FF FF FF FF FF FF
-OK-
CMD[5] reboot
-OK-
CMD[6] disconnect
-OK-
Task executed successfully
```

Figure 3.6 dump Flash data in command prompt window

3.6 Modify MAC

MCPLI supports modify MAC address by –x [value], [value] is a string of hex separated by colons, as shown in Figure 3.7.



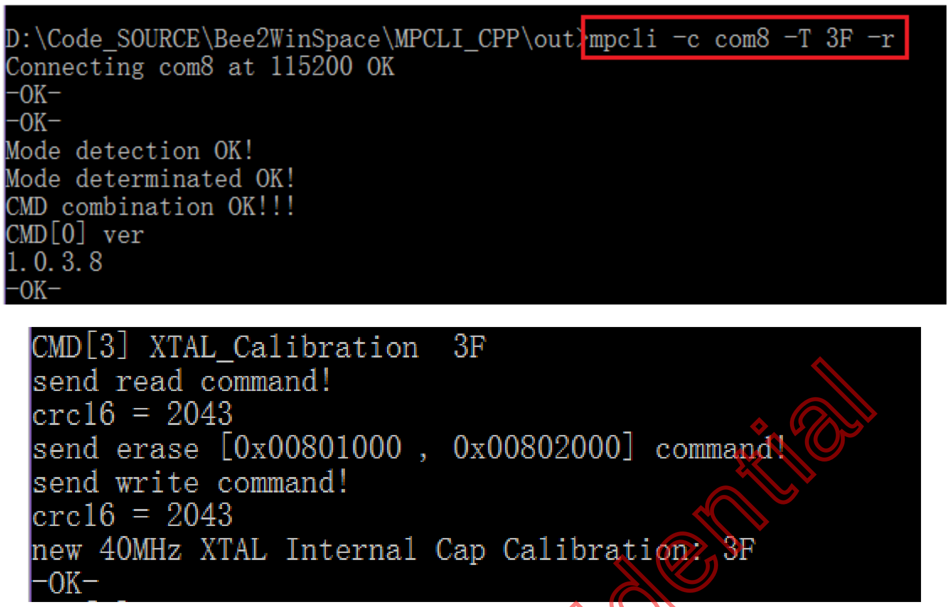
```
D:\Code_SOURCE\Bee2WinSpace\MPCLI_CPP\out>mpcli -c com17 -x 11:22:33:44:55:66 -r
Connecting com17 at 115200 OK...
-OK-
-OK-
-OK-
Mode detection OK!
Mode determinated OK!
CMD combination OK!!!
```

```
CMD[3] addmac 11:22:33:44:55:66
send read command!
crc16 = 2043
send erase [0x00801000 , 0x00802000] command!
send write command!
crc16 = 2043
new MAC: 11:22:33:44:55:66
-OK-
```

Figure 3.7 Modify MAC Command

3.7 Modify XTAL Internal Cap Calibration

MCPLI supports modify XTAL Internal Cap Calibration by –T [value], [value] is a byte of hex, size range is 00-7F, and its character length is 2. As shown in Figure 3.8.

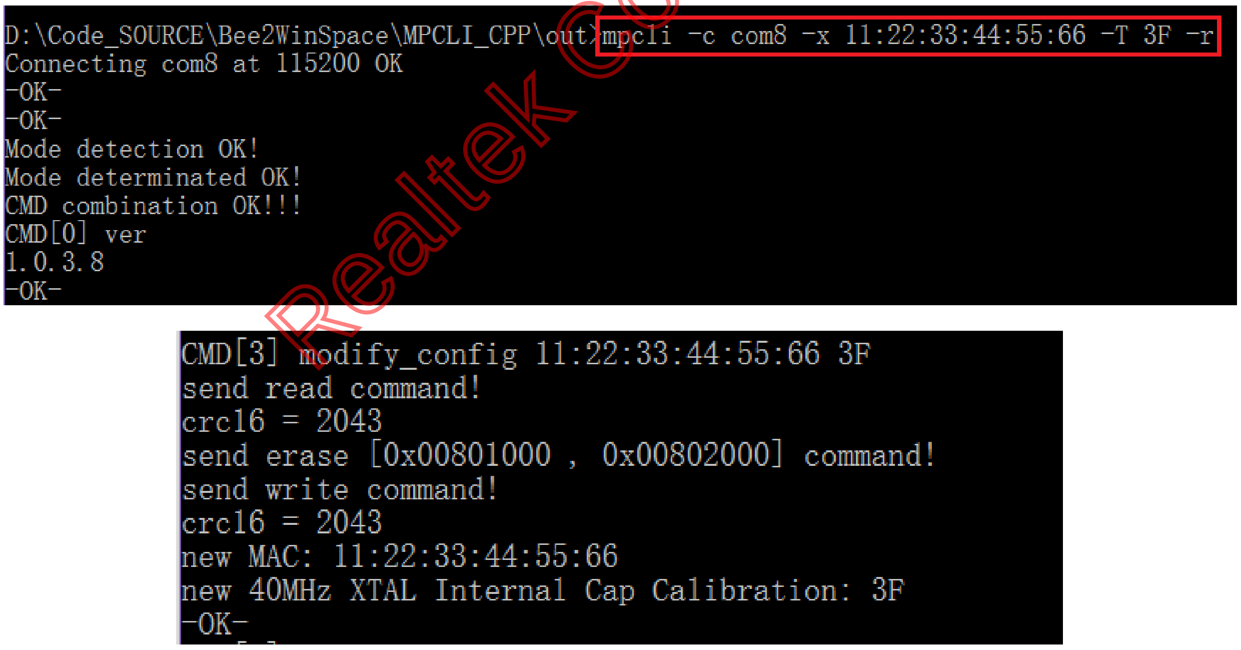


```
D:\Code_SOURCE\Bee2WinSpace\MPCLI_CPP\out>mpcli -c com8 -T 3F -r
Connecting com8 at 115200 OK
-OK-
-OK-
Mode detection OK!
Mode determinated OK!
CMD combination OK!!!
CMD[0] ver
1.0.3.8
-OK-
```

```
CMD[3] XTAL_Calibration  3F
send read command!
crc16 = 2043
send erase [0x00801000 , 0x00802000] command!
send write command!
crc16 = 2043
new 40MHz XTAL Internal Cap Calibration: 3F
-OK-
```

Figure 3.8 Modify XTAL Internal Cap Calibration Command

Modify both MAC and XTAL Internal Cap Calibration Command in the same command as shown in Figure 3.9.



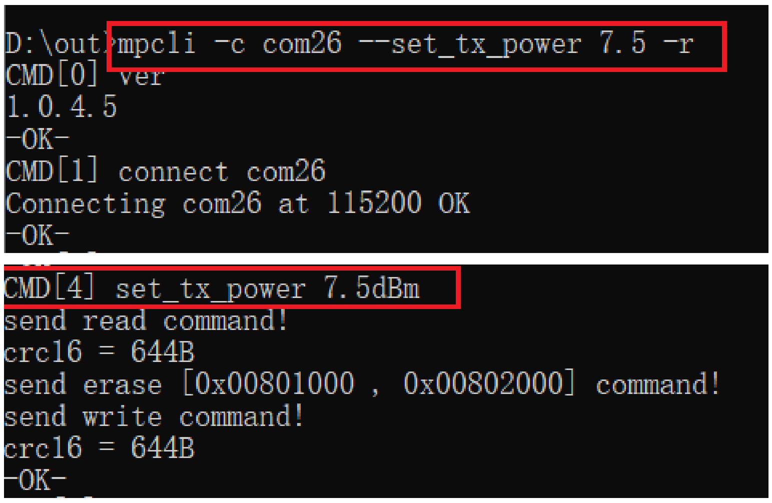
```
D:\Code_SOURCE\Bee2WinSpace\MPCLI_CPP\out>mpcli -c com8 -x 11:22:33:44:55:66 -T 3F -r
Connecting com8 at 115200 OK
-OK-
-OK-
Mode detection OK!
Mode determinated OK!
CMD combination OK!!!
CMD[0] ver
1.0.3.8
-OK-
```

```
CMD[3] modify_config 11:22:33:44:55:66 3F
send read command!
crc16 = 2043
send erase [0x00801000 , 0x00802000] command!
send write command!
crc16 = 2043
new MAC: 11:22:33:44:55:66
new 40MHz XTAL Internal Cap Calibration: 3F
-OK-
```

Figure 3.9 Modify MAC and XTAL Internal Cap Calibration Command

3.8 Modify TX Power

MCPLI supports modify TX Power by –set_tx_power [value], [value] is a dBm string. As shown in Figure 3.10.



```
D:\out>mpcli -c com26 --set_tx_power 7.5 -r
CMD[0] ver
1.0.4.5
-OK-
CMD[1] connect com26
Connecting com26 at 115200 OK
-OK-
```

```
CMD[4] set_tx_power 7.5dBm
send read command!
crc16 = 644B
send erase [0x00801000 , 0x00802000] command!
send write command!
crc16 = 644B
-OK-
```

Figure 3.10 Modify TX Power

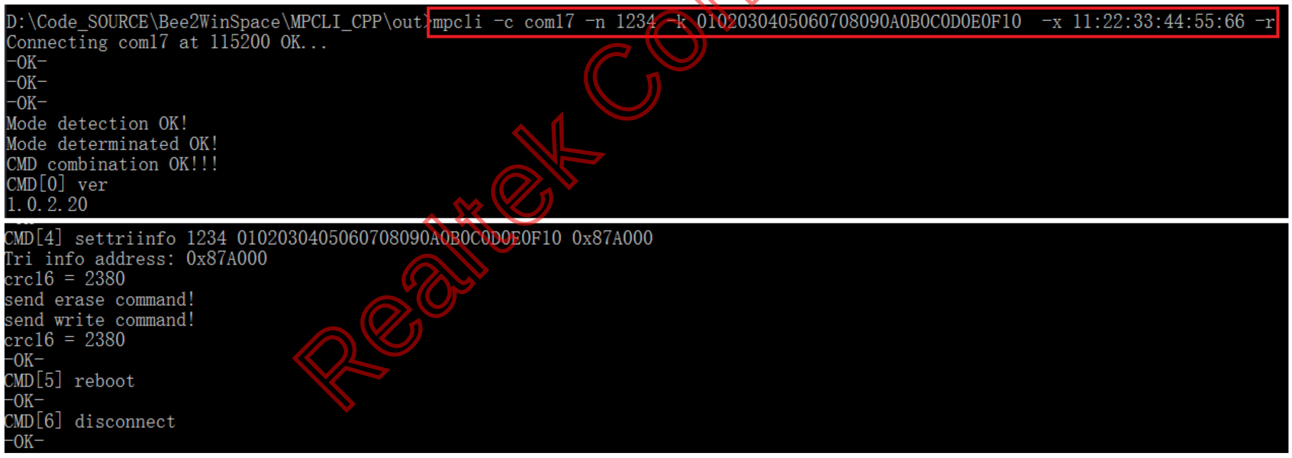
3.9 Write tri information

MPCLI supports write tri information shows in Figure 3.11.

-n [value], product ID, 4 bytes.

-k [value], secret key, 32 bytes.

-x [values],MAC address.



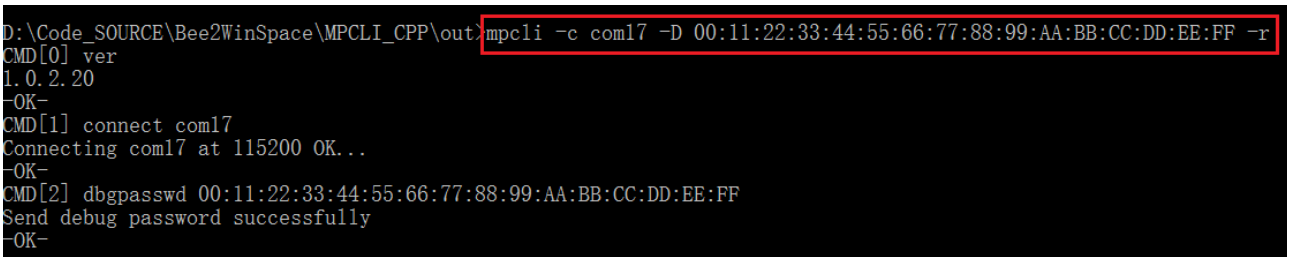
```
D:\Code_SOURCE\Bee2WinSpace\MPCLI_CPP\out>mpcli -c com17 -n 1234 -k 0102030405060708090A0B0C0D0E0F10  -x 11:22:33:44:55:66 -r
Connecting com17 at 115200 OK...
-OK-
-OK-
-OK-
Mode detection OK!
Mode determinated OK!
CMD combination OK!!!
CMD[0] ver
1.0.2.20
```

```
CMD[4] settriinfo 1234 0102030405060708090A0B0C0D0E0F10 0x87A000
Tri info address: 0x87A000
crc16 = 2380
send erase command!
send write command!
crc16 = 2380
-OK-
CMD[5] reboot
-OK-
CMD[6] disconnect
-OK-
```

Figure 3.11 write tri information Command

3.10 Debug Password

This is a custom function. The Flash of the terminal product will be locked after delivery. In case of any problem, it needs to be returned to the factory for Rework. In order to locate the problem while Rework, need to unlock the feature. Therefore, MPCLI is required to send 16-bytes Debug Password through and HCI CMD to unlock the lower layer. In this case, MPCLI is only responsible for sending Password and does not take responsibility for subsequent unlock operations. As shown in Figure 3.12, -D [value], a string of 16-bytes hex separated by colons. This function does not require Firmware Loader.



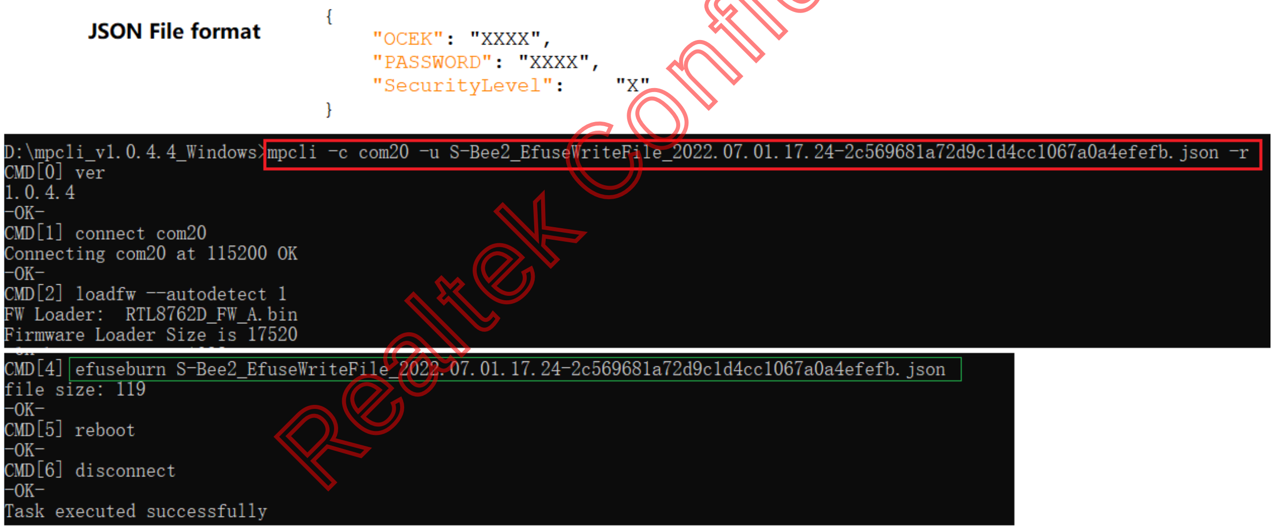
```
D:\Code_SOURCE\Bee2WinSpace\MPCLI_CPP\out>mpcli -c com17 -D 00:11:22:33:44:55:66:77:88:99:AA:BB:CC:DD:EE:FF -r
CMD[0] ver
1.0.2.20
-OK-
CMD[1] connect com17
Connecting com17 at 115200 OK...
-OK-
CMD[2] dbgpasswd 00:11:22:33:44:55:66:77:88:99:AA:BB:CC:DD:EE:FF
Send debug password successfully
-OK-
```

Figure 3.12 Debug Password Command

3.11 Write eFuse JSON File

MPCLI supports write Efuse JSON File as shows in below.

FW Loader will make a blank check before writing key. It will check whether the existed key is valid or not. If the existed key is valid, it will report pass and not write any byte. If the existed key is invalid, it will report failure and not write any byte.



JSON File format

```
{
    "OCEK": "XXXX",
    "PASSWORD": "XXXX",
    "SecurityLevel":    "X"
}
```

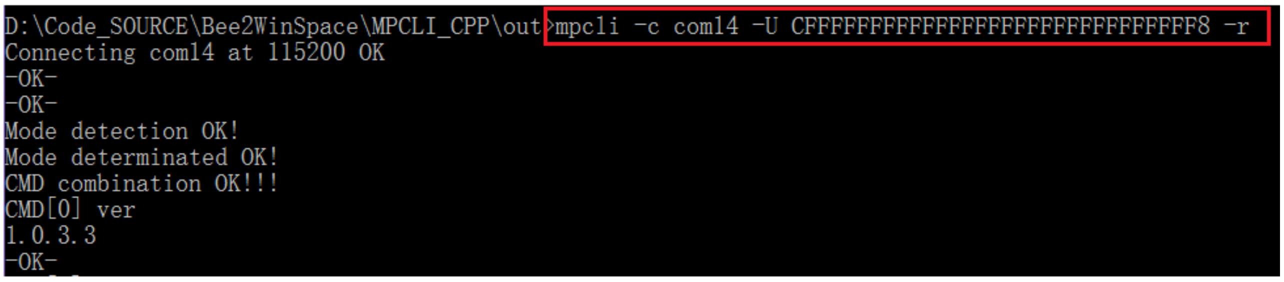
```
D:\mpcli_v1.0.4.4_Windows>mpcli -c com20 -u S-Bee2_EfuseWriteFile_2022.07.01.17.24-2c569681a72d9c1d4cc1067a0a4efefb.json -r
CMD[0] ver
1.0.4.4
-OK-
CMD[1] connect com20
Connecting com20 at 115200 OK
-OK-
CMD[2] loadfw --autodetect 1
FW Loader:  RTL8762D_FW_A.bin
Firmware Loader Size is 17520
CMD[4] efuseburn S-Bee2_EfuseWriteFile_2022.07.01.17.24-2c569681a72d9c1d4cc1067a0a4efefb.json
file size: 119
-OK-
CMD[5] reboot
-OK-
CMD[6] disconnect
-OK-
Task executed successfully
```

Figure 3.13 Write Efuse JSON File

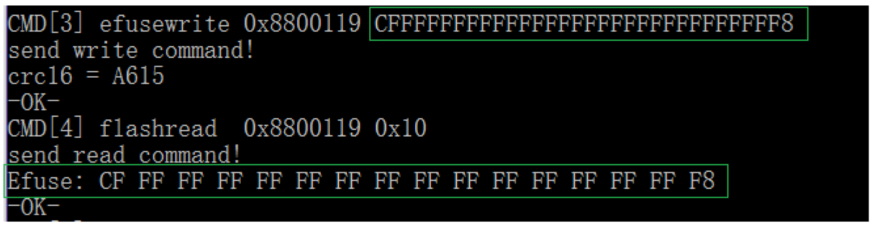
3.12 Write eFuse as OTP

Form MPCLI V1.0.3.3, MPCLI supports to write Efuse, as shown in Figure 3.14. –U [value], is a string of hex, and its characters length is 32 bytes, Figure 3.15 shows as an error. After burning Efuse, it's automatically read back and printed in the command window. When both –U and –P or –f –a are selected, -U will be ignored. This is, writing Efuse and burning image cannot be implemented on the same command line.

Only used with RTL8762C.

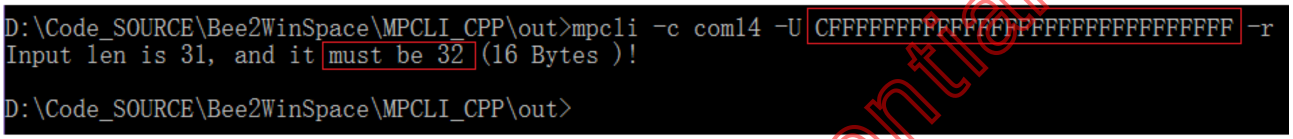


```
D:\Code_SOURCE\Bee2WinSpace\MPCLI_CPP\out>mpcli -c com14 -U CFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF8 -r
Connecting com14 at 115200 OK
-OK-
-OK-
Mode detection OK!
Mode determinated OK!
CMD combination OK!!!
CMD[0] ver
1.0.3.3
-OK-
```



```
CMD[3] efusewrite 0x8800119 CFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF8
send write command!
crc16 = A615
-OK-
CMD[4] flashread  0x8800119 0x10
send read command!
Efuse: CF FF FF FF FF FF FF FF FF FF FF FF FF FF FF F8
-OK-
```

Figure 3.14 Write Efuse Command



```
D:\Code_SOURCE\Bee2WinSpace\MPCLI_CPP\out>mpcli -c com14 -U CFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF -r
Input len is 31, and it must be 32 (16 Bytes )!

D:\Code_SOURCE\Bee2WinSpace\MPCLI_CPP\out>
```

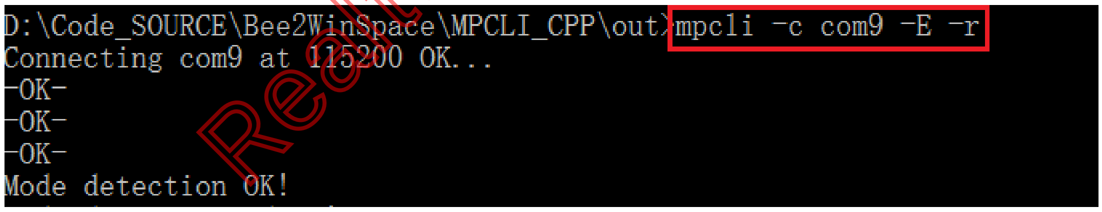
Figure 3.15 Write Efuse Error

3.13 Chip Erase

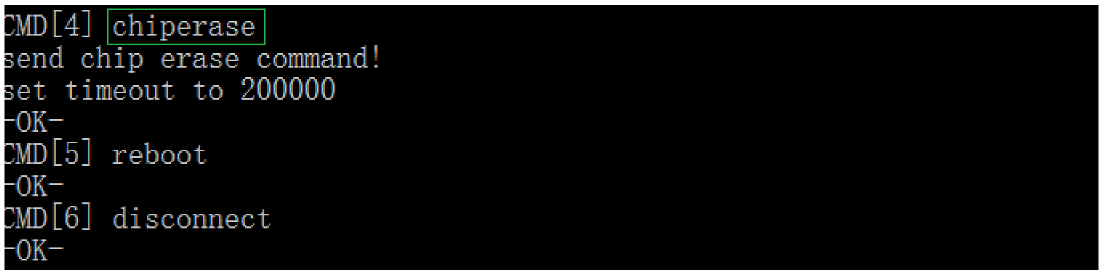
MPCLI supports write tri information shows in Figure 3.16 and Figure 3.17.

-E, chip erase

-C, chip erase exclude 0x800000-0x801000.

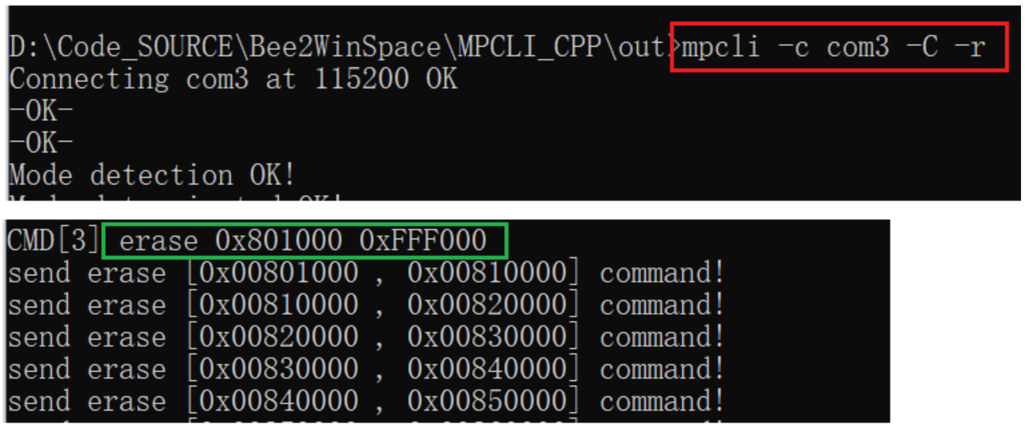


```
D:\Code_SOURCE\Bee2WinSpace\MPCLI_CPP\out>mpcli -c com9 -E -r
Connecting com9 at 115200 OK...
-OK-
-OK-
-OK-
Mode detection OK!
```



```
CMD[4] chiperase
send chip erase command!
set timeout to 200000
-OK-
CMD[5] reboot
-OK-
CMD[6] disconnect
-OK-
```

Figure 3.16 chip erase –E Command



```
D:\Code_SOURCE\Bee2WinSpace\MPCLI_CPP\out>mpcli -c com3 -C -r
Connecting com3 at 115200 OK
-OK-
-OK-
Mode detection OK!
```

```
CMD[3] erase 0x801000 0xFFF000
send erase [0x00801000 , 0x00810000] command!
send erase [0x00810000 , 0x00820000] command!
send erase [0x00820000 , 0x00830000] command!
send erase [0x00830000 , 0x00840000] command!
send erase [0x00840000 , 0x00850000] command!
```

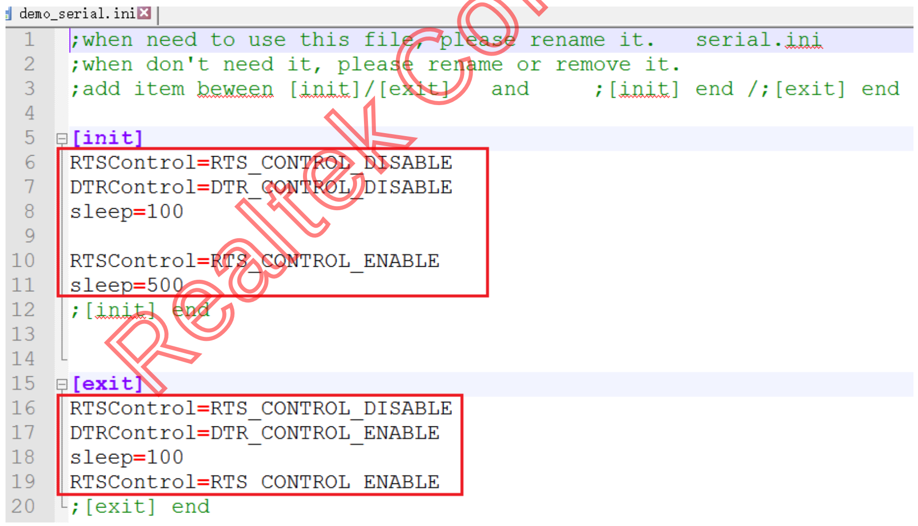
Figure 3.17 chip erase –C Command

3.14 Control serial

serial.ini in the same directory as mpcli.exe, mpcli parses the ini file when the serial port is opened or closed, and then executes the configuration items one by one. Currently, only supports RTS/DTR and SLEEP.

MPCLI.ZIP places demo_serial.ini, if necessary, rename to serial.ini, else can delete or rename to other name; configuration is shown in Figure 3.18.

Note: add or delete items in the red box.



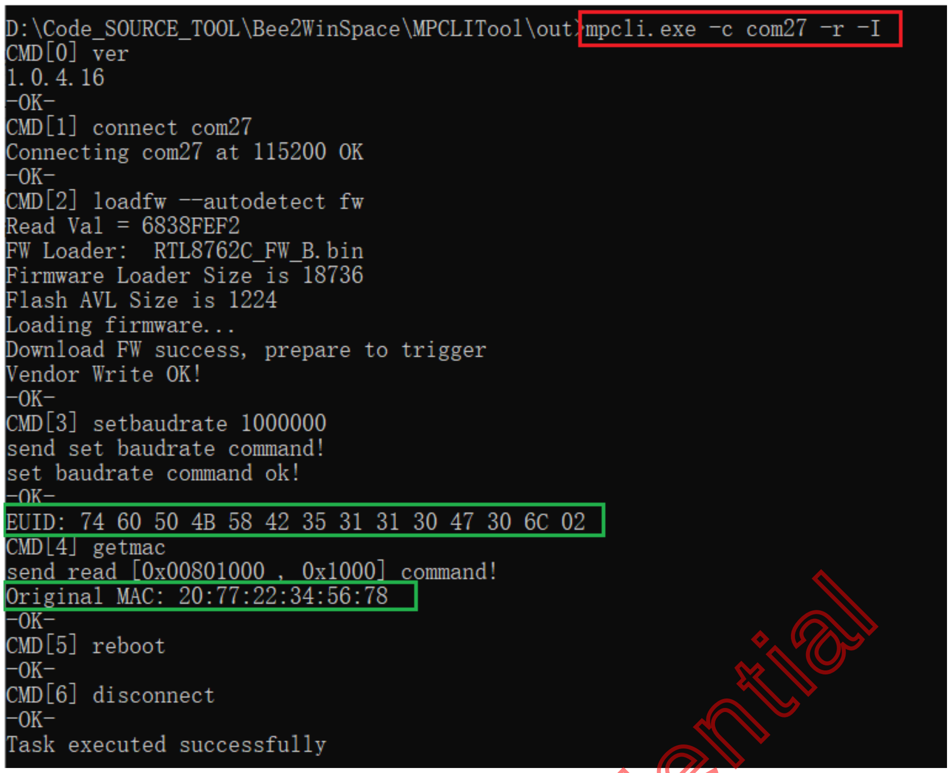
```
;when need to use this file, please rename it.   serial.ini
;when don't need it, please rename or remove it.
;add item beween [init]/[exit]   and     ;[init] end /;[exit] end

[init]
RTSControl=RTS_CONTROL_DISABLE
DTRControl=DTR_CONTROL_DISABLE
sleep=100

RTSControl=RTS_CONTROL_ENABLE
sleep=500
;[init] end


[exit]
RTSControl=RTS_CONTROL_DISABLE
DTRControl=DTR_CONTROL_ENABLE
sleep=100
RTSControl=RTS_CONTROL_ENABLE
;[exit] end
```

Figure 3.18 Control serial

3.15 Read MAC and EUID

MPCLI supports reading the IC MAC, and printing it in the command prompt window.

The -I option has been changed to only dump MAC, while during the normal execution of MPCLI, the EUID is automatically printed.



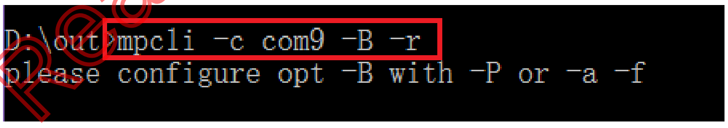
```
D:\Code_SOURCE_TOOL\Bee2WinSpace\MPCLITool\out>mpcli.exe -c com27 -r -I
CMD[0] ver
1.0.4.16
-OK-
CMD[1] connect com27
Connecting com27 at 115200 OK
-OK-
CMD[2] loadfw --autodetect fw
Read Val = 6838FEF2
FW Loader:  RTL8762C_FW_B.bin
Firmware Loader Size is 18736
Flash AVL Size is 1224
Loading firmware...
Download FW success, prepare to trigger
Vendor Write OK!
-OK-
CMD[3] setbaudrate 1000000
send set baudrate command!
set baudrate command ok!
-OK-
EUID: 74 60 50 4B 58 42 35 31 31 30 47 30 6C 02
CMD[4] getmac
send read [0x00801000 , 0x1000] command!
Original MAC: 20:77:22:34:56:78
-OK-
CMD[5] reboot
-OK-
CMD[6] disconnect
-OK-
Task executed successfully
```

Figure 3.19 Read MAC and EUID

3.16 Read back MAC

This function is designed to burn image without changing the existing MAC in Config region Flash, used by –B both with –P or –f –a. The implementation is as follows: before burning new image, get original MAC, and after burning image, write back this MAC to Flash. The details are attached below:

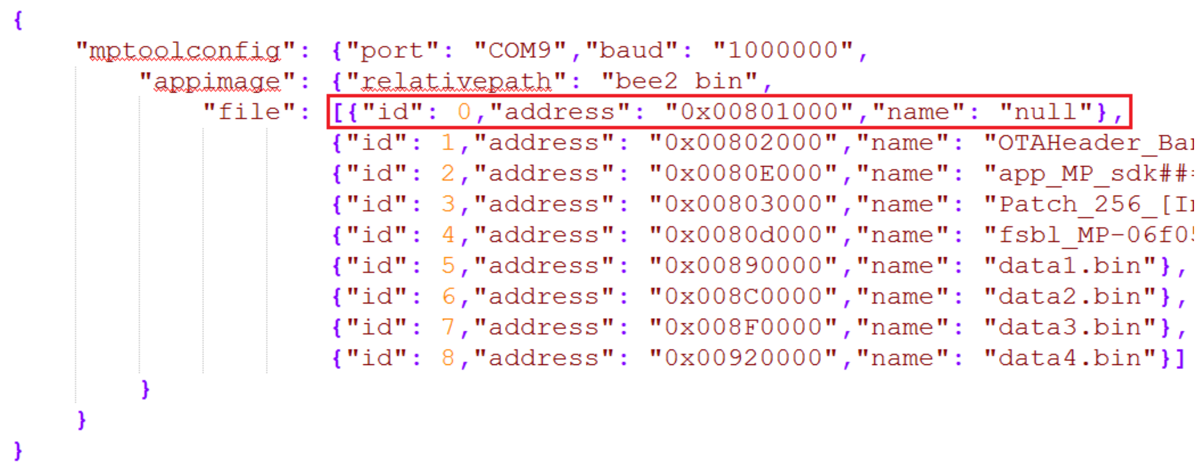
1. Figure 3.20 shows error log .



```
D:\out>mpcli -c com9 -B -r
please configure opt -B with -P or -a -f
```

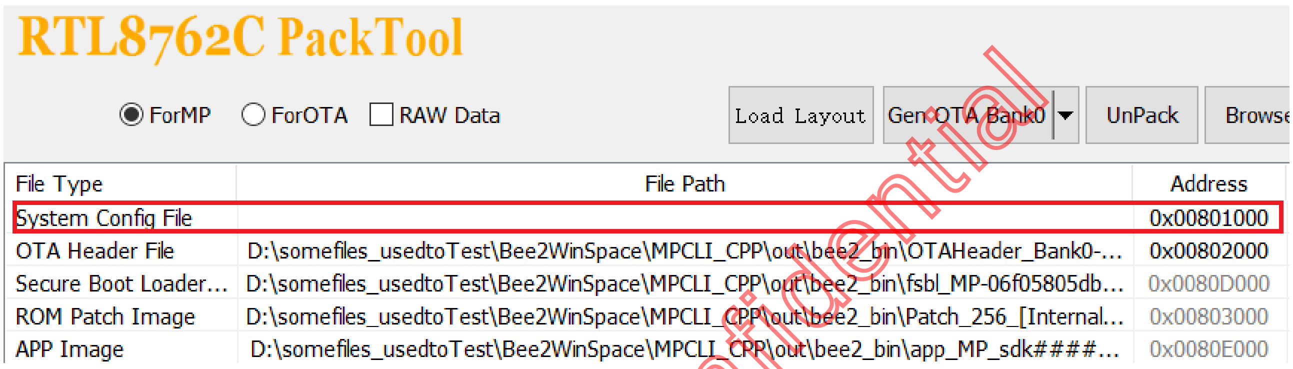
Figure 3.20 Keep original MAC error

2. When select –P or –a –f at the same command, but the config-file not in the packed image or not configure in JSON file, as shown in Figure 3.21 and Figure 3.22, MPCLI prints error message shown in Figure 3.23.



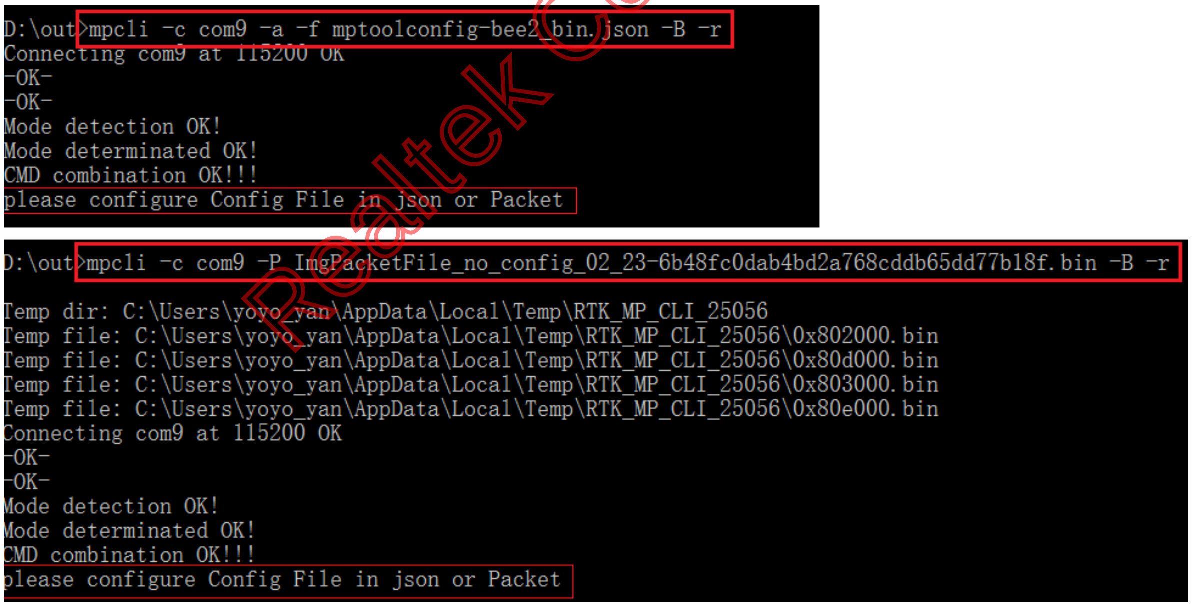
```
{
    "mptoolconfig": {"port": "COM9","baud": "1000000",
        "appimage": {"relativepath": "bee2 bin",
            "file": [{"id": 0,"address": "0x00801000","name": "null"},
                    {"id": 1,"address": "0x00802000","name": "OTAHeader_Ba
                    {"id": 2,"address": "0x0080E000","name": "app_MP_sdk##
                    {"id": 3,"address": "0x00803000","name": "Patch_256_[I
                    {"id": 4,"address": "0x0080d000","name": "fsbl_MP-06f0
                    {"id": 5,"address": "0x00890000","name": "data1.bin"},
                    {"id": 6,"address": "0x008C0000","name": "data2.bin"},
                    {"id": 7,"address": "0x008F0000","name": "data3.bin"},
                    {"id": 8,"address": "0x00920000","name": "data4.bin"}]
        }
    }
}
```

Figure 3.21 Keep Original MAC error 2_a



File Type	File Path	Address
System Config File		0x00801000
OTA Header File	D:\somefiles_usedtoTest\Bee2WinSpace\MPCLI_CPP\out\bee2_bin\OTAHeader_Bank0-...	0x00802000
Secure Boot Loader...	D:\somefiles_usedtoTest\Bee2WinSpace\MPCLI_CPP\out\bee2_bin\fsbl_MP-06f05805db...	0x0080D000
ROM Patch Image	D:\somefiles_usedtoTest\Bee2WinSpace\MPCLI_CPP\out\bee2_bin\Patch_256_[Internal...	0x00803000
APP Image	D:\somefiles_usedtoTest\Bee2WinSpace\MPCLI_CPP\out\bee2_bin\app_MP_sdk####...	0x0080E000

Figure 3.22 Keep Original MAC error 2_b

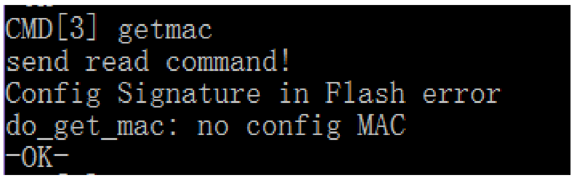


```
D:\out>mpcli -c com9 -a -f mptoolconfig-bee2_bin.json -B -r
Connecting com9 at 115200 OK
-OK-
-OK-
Mode detection OK!
Mode determinated OK!
CMD combination OK!!!
please configure Config File in json or Packet
```

```
D:\out>mpcli -c com9 -P ImgPacketFile_no_config_02_23-6b48fc0dab4bd2a768cddb65dd77b18f.bin -B -r

Temp dir: C:\Users\yoyo_yan\AppData\Local\Temp\RTK_MP_CLI_25056
Temp file: C:\Users\yoyo_yan\AppData\Local\Temp\RTK_MP_CLI_25056\0x802000.bin
Temp file: C:\Users\yoyo_yan\AppData\Local\Temp\RTK_MP_CLI_25056\0x80d000.bin
Temp file: C:\Users\yoyo_yan\AppData\Local\Temp\RTK_MP_CLI_25056\0x803000.bin
Temp file: C:\Users\yoyo_yan\AppData\Local\Temp\RTK_MP_CLI_25056\0x80e000.bin
Connecting com9 at 115200 OK
-OK-
-OK-
Mode detection OK!
Mode determinated OK!
CMD combination OK!!!
please configure Config File in json or Packet
```

Figure 3.23 Keep Original MAC error 2_c

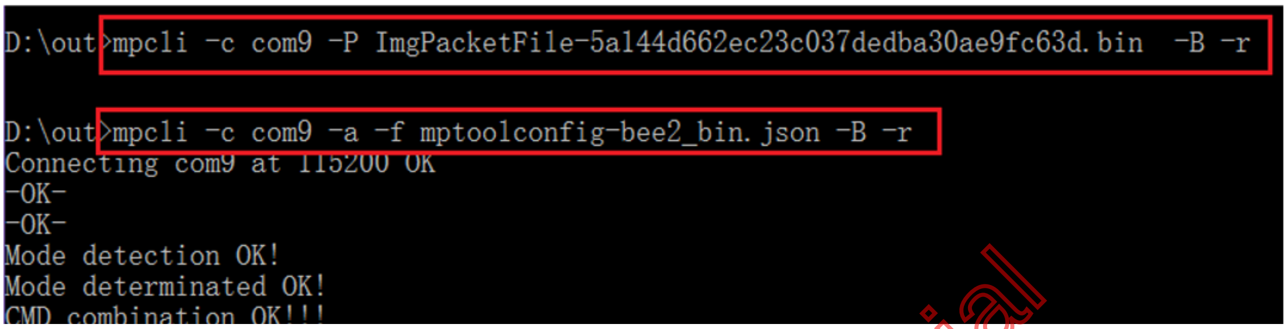
3. If the burned board is empty or the Config Signature is incorrect, and –B is selected at the same time, MCPLI displays a message as shown in Figure 3.24, and continue performing.



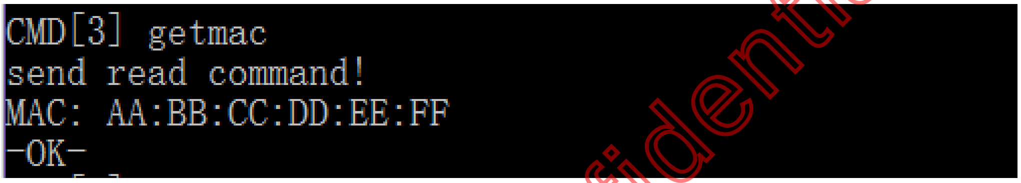
```
CMD[3] getmac
send read command!
Config Signature in Flash error
do_get_mac: no config MAC
-OK-
```

Figure 3.24 Keep Original MAC error 3

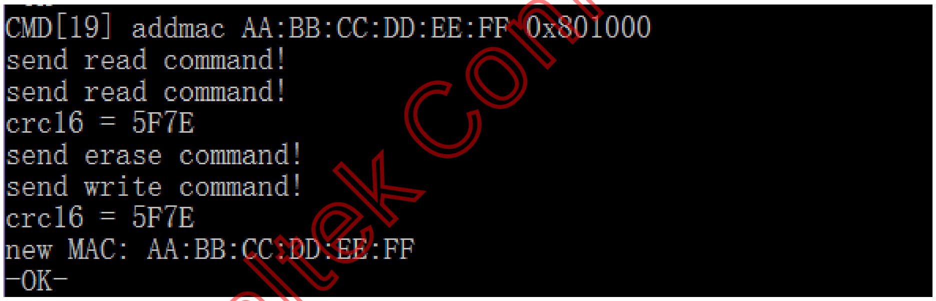
4. Example of Correct execution is shown in Figure 3.25.



```
D:\out>mpcli -c com9 -P ImgPacketFile-5a144d662ec23c037dedba30ae9fc63d.bin  -B -r

D:\out>mpcli -c com9 -a -f mptoolconfig-bee2_bin.json -B -r
Connecting com9 at 115200 OK
-OK-
-OK-
Mode detection OK!
Mode determinated OK!
CMD combination OK!!!
```



```
CMD[3] getmac
send read command!
MAC: AA:BB:CC:DD:EE:FF
-OK-
```



```
CMD[19] addmac AA:BB:CC:DD:EE:FF 0x801000
send read command!
send read command!
crc16 = 5F7E
send erase command!
send write command!
crc16 = 5F7E
new MAC: AA:BB:CC:DD:EE:FF
-OK-
```

Figure 3.25 Keep Original MAC Correct

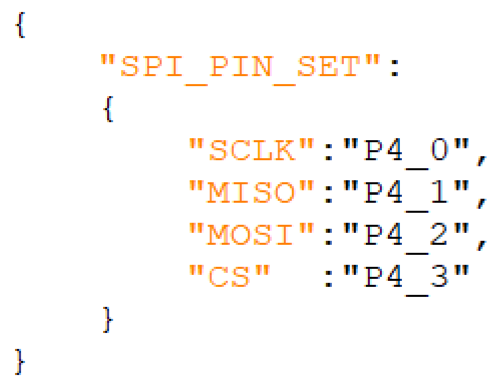
3.17 Read back 40MHz XTAL Internal Cap Calibration

This function is designed to burn image without changing the existing 40MHz XTAL Internal Cap Calibration in Config region Flash, used by --get_back_xtal_calibration both with –P or –f –a. The implementation is as above Read back Mac. Before burning new image, get original 40MHz XTAL Internal Cap Calibration, and after burning image, write back this 40MHz XTAL Internal Cap Calibration to Flash.

3.18 Config SPI Flash PIN

This function is designed to config SPI Flash PIN, write image to SPI Flash, used by --spi_pin_config_json [spi_pin_config.json] both with –P or –f –a.

The implementation is as follows: before burning new image, parse spi_pin_config.json, As shown in Figure 3.26



```
{
        "SPI_PIN_SET":
        {
                "SCLK":"P4_0",
                "MISO":"P4_1",
                "MOSI":"P4_2",
                "CS"  :"P4_3"
        }
}
```

Figure 3.26 Config SPI PIN json file

Command:

-->mpcli.exe –c comX –P packedimage.bin --spi_pin_config_json SPI_PIN_config.json –r

--> mpcli.exe –c comX –a –f fileconfig.json --spi_pin_config_json SPI_PIN_config.json –r

3.19 Other Options

When burn image, and modify MAC at the same command:

mpcli.exe –c com1 –b 115200 –P ImgPacketFile.bin –x 11:22:33:44:55:66 -r

mpcli.exe –a –f mptoolconfig.json –x 11:22:33:44:55:66 –r

When burn image, and modify XTAL Internal Calibration value at the same command:

mpcli.exe –c com1 –b 115200 –P ImgPacketFile.bin –T 3F -r

mpcli.exe –a –f mptoolconfig.json –T 3F -r

mpcli.exe –c com1 –b 115200 –P ImgPacketFile.bin –x 11:22:33:44:55:66 –T 3F -r

mpcli.exe –a –f mptoolconfig.json –x 11:22:33:44:55:66 –T 3F -r

After burning image, read back EUID and MAC at the same command:

mpcli.exe –c com1 –b 115200 –P ImgPacketFile.bin –I-r

mpcli.exe –a –f mptoolconfig.json –I –r

4 Attentions

1. Burning address requires 4 bytes alignment.
2. Before burning image, the actual erased Flash size is scaled up to 4k alignment based on the size of the file to be burned.
3. The MPCLI tool can automatically load firmware in the FW folder based on the IC type. Currently, the firmware of the supported IC types RTL8762C, RTL8762D, and RTL8762E is stored in the FW directory of the released version. You cannot modify or delete the contents in this directory.
4. Run -C command (reserve the 0x800000-0x801000 4K) to run the RTL8762E full erase command. If you run the -E command, the full erase command will fail due to timeout

5 Error Code

Table 5.1 Error Code

Error Code	Indication
0	Success
1	Parameter error
2	Port number error
3	Open file error
4	ic type error
5	No ready error
6	hci event error
7	mp event error
8	crc check error
9	Timeout error
10	Flash read error
11	Flash verify error
12	Configuration signature error
13	Packed image parse error
14	make temp dir error
15	alloc error
16	save error
17	delete error
18	Dump data error