OPL1000

ULTRA-LOW POWER 2.4GHZ WI-FI + BLUETOOTH SMART SOC

OTA WIFI Demo User Guide

OPULINKS

http://www.opulinks.com/

OPL1000-Demo-ota-wifi-guide-R01 | Version V04

REVISION HISTORY

Date	Version	Contents Updated
2018-08-23	0.1	● Initial Release
2019-01-10	0.2	● http_server.exe support AP SSID and password input
2019-07-10	0.3	● modify fw_binary folder to fw_pack folder
2020-01-02	0.4	● update http_server.exe use guide

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE

1. INTRODUCTION

1.1. Application Scope

This document describes (1) the procedure to use SDK API to realize OTA update through WIFI on OPL 1000. (2) the procedure utilizes the http_server tool provided to construct a simple HTTP server to realize OTA feature.

1.2. Abbreviations

Abbr.	Explanation
AP	Wireless Access Point
APP	APPlication
APS	Application Sub-system, refers to M3 MCU in this document
Blewifi	BLE config WIFI
DevKit	Development Kit
OTA	Over-the-Air Technology
TCP	Transmission Control Protocol

1.3. References

[1] DEVKIT Quick Start Guide OPL1000-DEVKIT-getting-start-guide.pdf

[2] Download Tool User Guide OPL1000-patch-download-tool-user-guide.pdf

[3] SDK Application Development Guide OPL1000-SDK-Development-guide.pdf

2. REALIZATION OF OTA WIFI PROGRAM

2.1. Working Principle

The directory for OTA WIFI sample program is under SDK\APS_PATCH\examples\system\ota_wifi

The operation procedure is as follow:.

1 Start WIFI, configure OPL1000 as Station mode.
2 Scan for available AP.
3 If the designated AP SSID for connection is in the list of available AP, attempt to connect.
4 After AP has been successfully connected, OPL1000 assumes http client terminal to establish connection with the http server. The address and port for http server is defined by the macro-defined HTP_SERVER_IP and HTP_SERVER_PORT
5 After establishing connection with the http server, use the http get command to obtain OTA Image file from the designated the http server.
6 Call ota related API to store the OTA Image data obtained in Flash based on the specified format.

The designated AP SSID and Password are defined in http_ota_example.h file, as shown in Figure 1.

Figure 1: Setting AP SSID and Password

```
#define  WIFI_SSID   "Opulinks-TEST-AP"
#define  WIFI_PASSWORD   "1234abcd"
```

The designated HTTP Server IP and port number are defined in http_ota_example.c file, as shown in Figure 2.

Figure 2: Setting HTTP server IP and Port

```
#define  HTP_SERVER_IP   "192.168.1.102"
#define  HTP_SERVER_PORT   "8000"
```

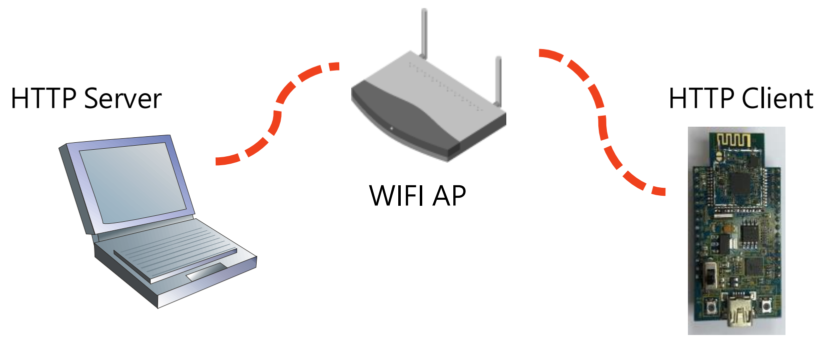
The user can select any available AP and change the AP name and Password to AP SSID and Password in http_ota_example.h file.

In this demonstration, the PC is used as the Http Server. The IP for the PC is the dynamic IP address assigned after PC is connected to AP. The user can use ipconfig command to check the IP address for the PC and fill into the macro-definition of HTP_SERVER_IP shown in Figure 2.

The data transmission network topology established between HTTP server and OPL1000 is shown in Figure 3.

OPL1000 plays the role of the Station to connect to WIFI AP and the PC also connects to WIFI AP as Station. This is done such that PC and OPL1000 are connected to the same AP and are in the same LAN segment. When the PC executes http_server.exe program, it serves as the http server, allowing http_server.exe program to connect to the AP you have selected.

Figure 3：Sample of OPL1000 http client network connection diagram



Firstly, execute http_server.exe program from the PC (path of Demo\ OTA_WIFI) and the execution interface is shown in Figure 4.

The specific use of http_server.exe program is as follows.

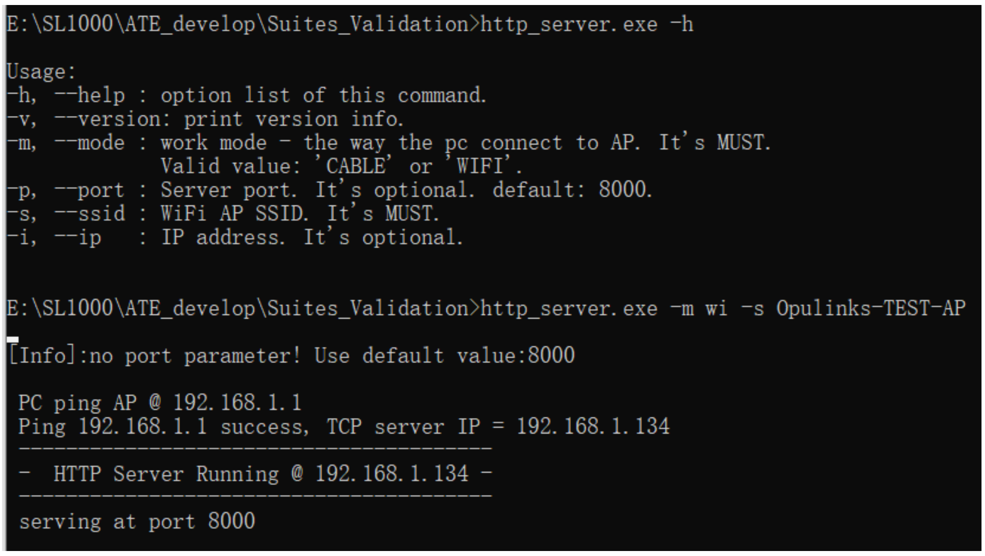
Enter the directory where http_server.exe program is located and execute command:

http_server.exe -m <mode> -s <AP SSID>

Execute command：http_server.exe -m wi -s Opulinks-TEST-AP

Message showing the IP address assigned by WIFI AP to the PC will pop up. The user enters this IP address into the macro-definition of HTP_SERVER_IP and edit. The OTA image firmware generated is then downloaded into OPL1000. Connect OPL1000 to WIFI AP. After successful connection, connect to the HTTP Server, and obtain the OTA Image firmware required for update from the HTTP server. Save the firmware in Flash. When the PC is next powered on, the newly downloaded firmware will be used.

Figure 4: Execution interface for the HTTP Server



```
E:\SL1000\ATE_develop\Suites_Validation>http_server.exe -h

Usage:
-h, --help : option list of this command.
-v, --version: print version info.
-m, --mode : work mode - the way the pc connect to AP.  It's MUST.
             Valid value: 'CABLE' or 'WIFI'.
-p, --port : Server port.  It's optional.  default: 8000.
-s, --ssid : WiFi AP SSID.  It's MUST.
-i, --ip   : IP address.  It's optional.


E:\SL1000\ATE_develop\Suites_Validation>http_server.exe -m wi -s Opulinks-TEST-AP

[Info]:no port parameter! Use default value:8000

 PC ping AP @ 192.168.1.1
 Ping 192.168.1.1 success, TCP server IP = 192.168.1.134
 ----------------------------------------
 -  HTTP Server Running @ 192.168.1.134 -
 ----------------------------------------
 serving at port 8000
```

2.2. API

The description for API used in the OTA wifi example is shown in Table 1

Table 1: Description for call API

API Interface	API Description
wifi_register_event_handler	Register the internal WIFI event handler
wifi_event_loop_init	Initialize the event handler loop function, and is defined as wifi_event_handler_cb in this example
wifi_init	Initialize the stack used for WIFI and wifi initialize completed event handler
wifi_set_config	Configure the working mode for OPL1000 WIFI
wifi_start	Start WIFI
osThreadCreate	Establish user application process, the process entrance is user_wifi_app_entry
wifi_event_handler_cb	Define the corresponding handler operation when WIFI related event information ID is received
wifi_do_scan	Scan for available wireless access point
wifi_connection	If the designated AP is in the list of available AP, connect to it

API Interface	API Description
lwip_net_start	Start lwip network protocol stack
lwip_network_init	Initialize Tcpip protocol stack and network port
lwip_net_ready	Wait for connection and obtain the dynamically allocated IP address from AP
ota_download_by_http	Download OTA firmware through http
httpclient_connect	http client terminal connects to the http server, called by ota_download_by_http
ota_http_retrieve_get	OPL1000 uses http get request to obtain OTA firmware data from server, called by ota_download_by_http
httpclient_close	Close connection with http client terminal
httpclient_send_request	http client terminal sends get request and obtain specific data from the server. Called by ota_http_retrieve_get
ota_http_retrieve_offset	Store the data in designated offset position in cache, called by ota_http_retrieve_get
ota_data_write	Write the data in cache to Flash, called by ota_http_retrieve_ge

3. OTA WIFI AUTHENTICATE FUNCTION

3.1. Example of Editing OTA WIFI

The example of editing ota wifi includes three steps:

Step1: Firstly, check the WIFI AP that the user needs to connect to and fill in its SSID and the visitor password into WIFI_SSID and WIFI_PASSWORD macro-defined in tcp_client.h file. If the AP is open with no password, leave the WIFI_PASSWORD definition blank.

```
#define WIFI_SSID              "Opulinks-TEST-AP"
#define WIFI_PASSWORD          "1234abcd"
```

Step2: Connect PC to WIFI AP by executing http_server.exe program and start the http server. The execution interface in Figure 4 shows that the IP for the http server is 192.168.1.134.

```
#define  HTP_SERVER_IP    "192.168.1.134"
#define  HTP_SERVER_PORT    "8000"
```

Step3: Use Keil C to edit ota wifi project and the path for the project file is

SDK\APS_PATCH\examples\system\ota_wifi\opl1000_app_m3.uvprojx

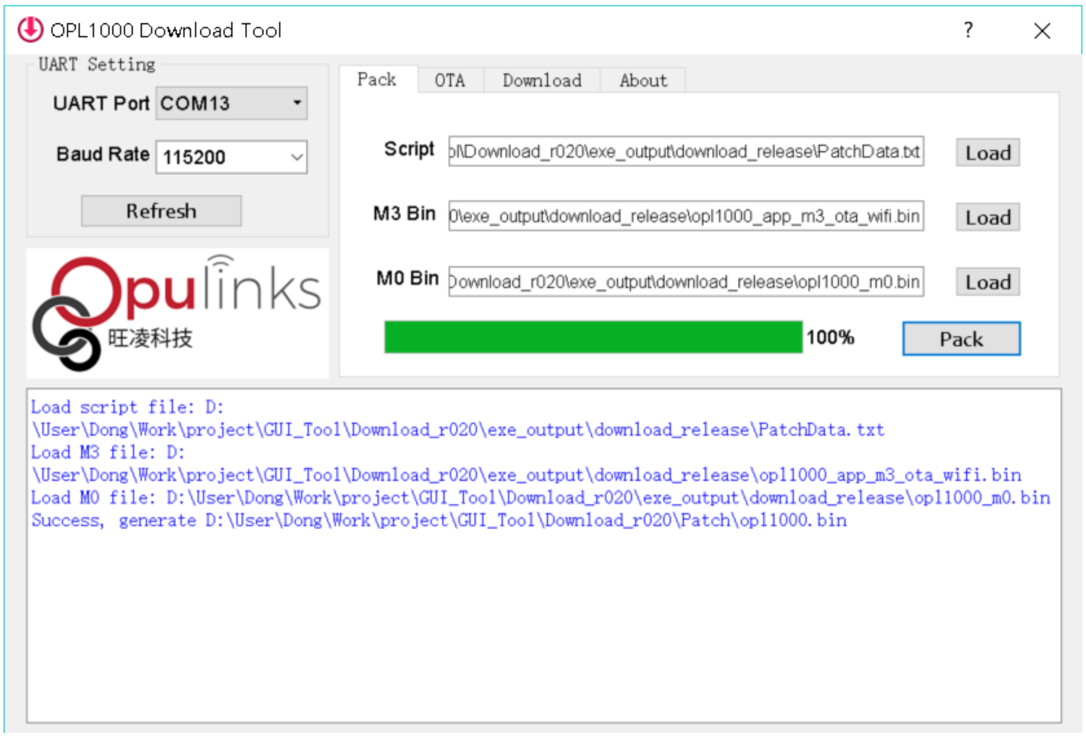
Please refer to reference [3] SKD Application Development Guide for the tool setting and editing procedure in Keil C

3.2. Generate and Download OTA Firmware

After the successful edit, opl1000_app_m3.bin file will be produced under SDK\APS_PATCH\examples\system\ota_wifi\Output\Objects directory. There are three steps required to convert opl1000_app_m3.bin to firmware that supports OTA function:

Step 1 : Copy opl1000_app_m3.bin file to FW_Pack directory and use the download tool to combine it with m0 bin file, producing opl1000.bin file as shown in Figure 5.

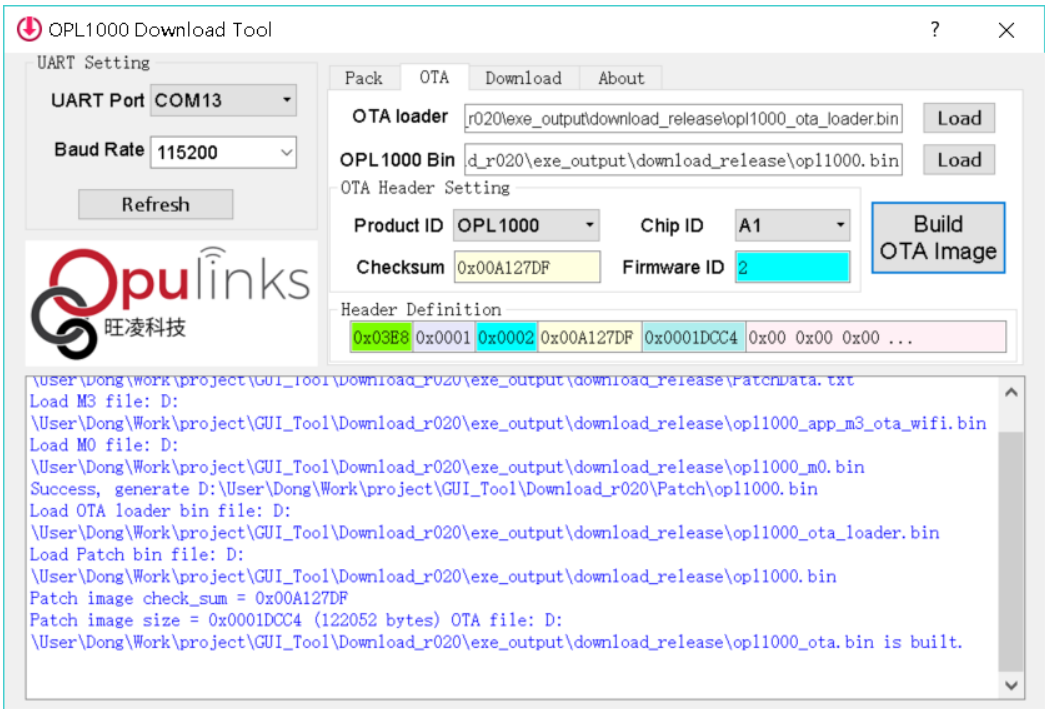
Figure 5: Combines OTA WIFI M3 file and M0 file to produce OPl1000.bin


OPL1000 Download Tool
UART Setting
UART Port COM13
Baud Rate 115200
Refresh
Pack
OTA
Download
About
Script pl\Download_r020\exe_output\download_release\PatchData.txt
Load
M3 Bin 0\exe_output\download_release\opl1000_app_m3_ota_wifi.bin
Load
M0 Bin Download_r020\exe_output\download_release\opl1000_m0.bin
Load
100%
Pack

```
Load script file: D:
\User\Dong\Work\project\GUI_Tool\Download_r020\exe_output\download_release\PatchData.txt
Load M3 file: D:
\User\Dong\Work\project\GUI_Tool\Download_r020\exe_output\download_release\opl1000_app_m3_ota_wifi.bin
Load M0 file: D:\User\Dong\Work\project\GUI_Tool\Download_r020\exe_output\download_release\opl1000_m0.bin
Success, generate D:\User\Dong\Work\project\GUI_Tool\Download_r020\Patch\opl1000.bin
```

Step 2 : The opl1000.bin file generated in Step 1 is a firmware without OTA loader. We need to produce Header message based on it and combine with OTA loader to produce the OTA Image file. As shown in Figure 6, OTA loader file (under FW_Pack directory) and the recently produced opl1000.bin file are loaded in the OTA tab. Click "Build OTA Image" and, opl1000_ota.bin file is generated.

Figure 6: Produce OTA Image file



UART Setting
UART Port COM13
Baud Rate 115200

OTA loader r020\exe_output\download_release\opl1000_ota_loader.bin
OPL1000 Bin d_r020\exe_output\download_release\opl1000.bin

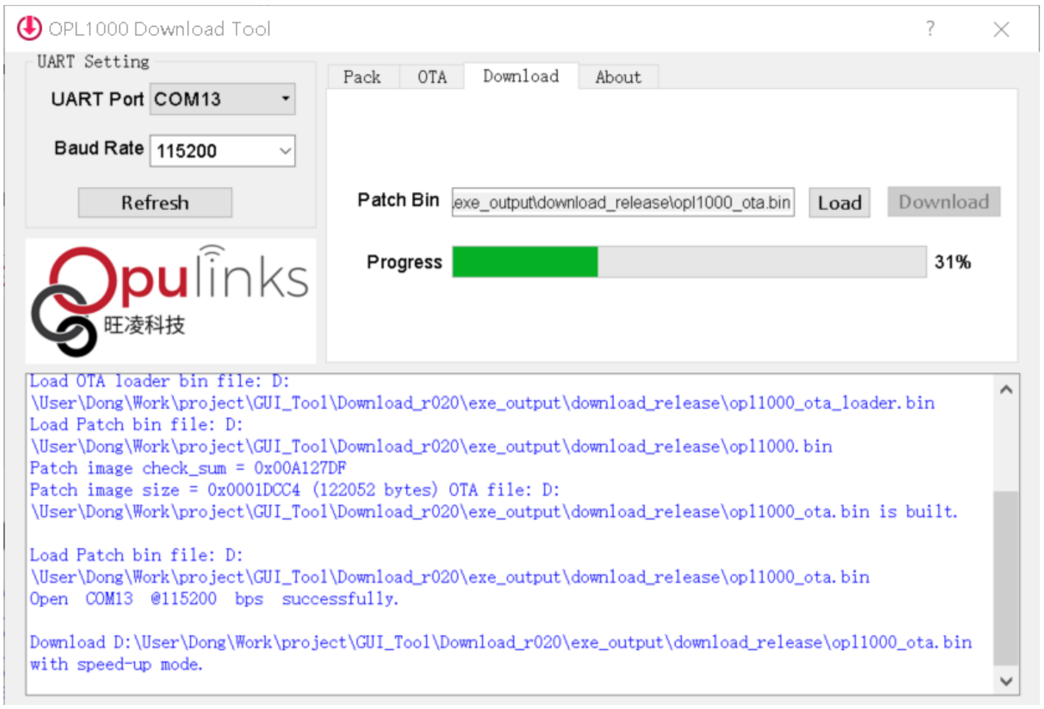
OTA Header Setting
Product ID OPL1000 Chip ID A1
Checksum 0x00A127DF Firmware ID 2

Header Definition
0x03E8 0x0001 0x0002 0x00A127DF 0x0001DCC4 0x00 0x00 0x00 ...

```
\User\Dong\Work\project\GUI_Tool\Download_r020\exe_output\download_release\PatchData.txt
Load M3 file: D:
\User\Dong\Work\project\GUI_Tool\Download_r020\exe_output\download_release\opl1000_app_m3_ota_wifi.bin
Load M0 file: D:
\User\Dong\Work\project\GUI_Tool\Download_r020\exe_output\download_release\opl1000_m0.bin
Success, generate D:\User\Dong\Work\project\GUI_Tool\Download_r020\Patch\opl1000.bin
Load OTA loader bin file: D:
\User\Dong\Work\project\GUI_Tool\Download_r020\exe_output\download_release\opl1000_ota_loader.bin
Load Patch bin file: D:
\User\Dong\Work\project\GUI_Tool\Download_r020\exe_output\download_release\opl1000.bin
Patch image check_sum = 0x00A127DF
Patch image size = 0x0001DCC4 (122052 bytes) OTA file: D:
\User\Dong\Work\project\GUI_Tool\Download_r020\exe_output\download_release\opl1000_ota.bin is built.
```

Step 3： Load the opl1000_ota.bin file produced in Step 2 in the download tab. Please note that the user needs to click "reset" to reset OPL1000 within 5 seconds after clicking "download". This is shown in Figure 7.

Figure 7: Download opl1000_ota.bin file



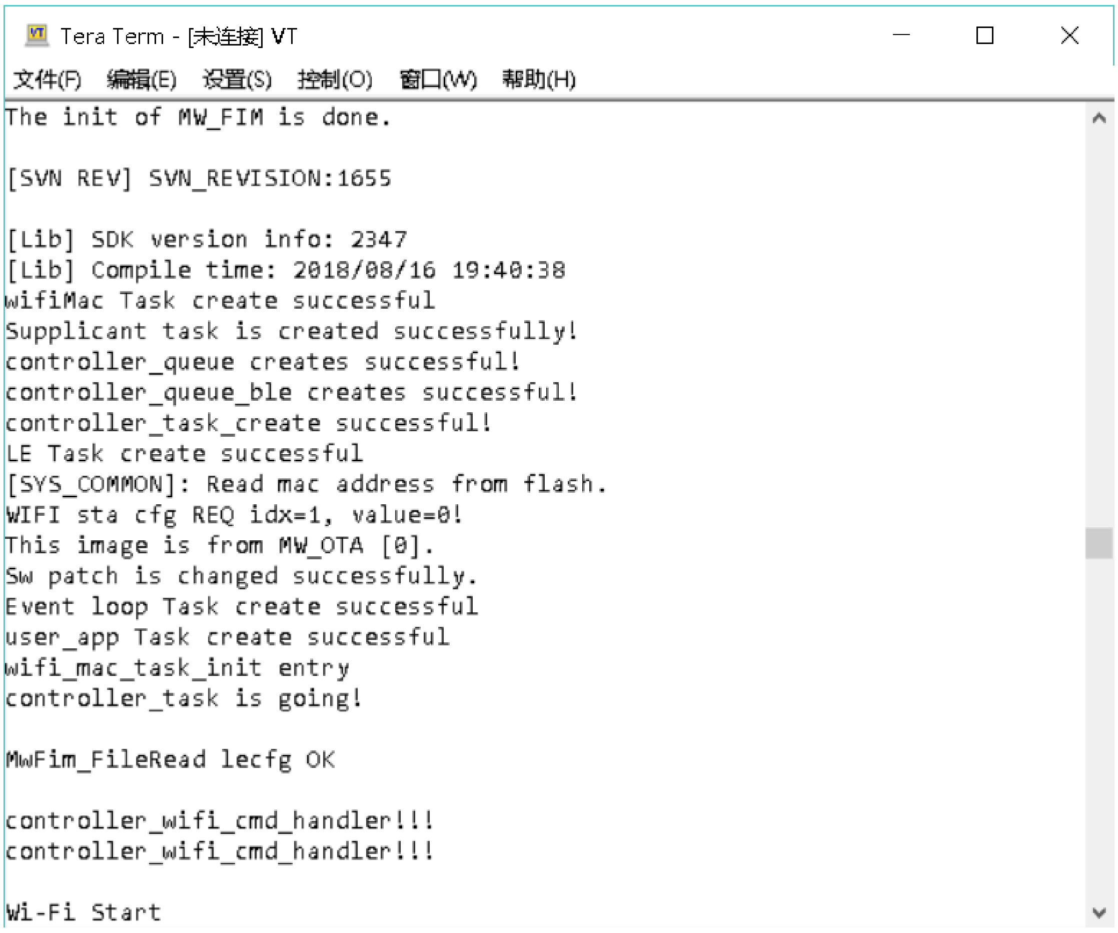
UART Setting
UART Port COM13
Baud Rate 115200

Patch Bin exe_output\download_release\opl1000_ota.bin
Progress 31%

```
Load OTA loader bin file: D:
\User\Dong\Work\project\GUI_Tool\Download_r020\exe_output\download_release\opl1000_ota_loader.bin
Load Patch bin file: D:
\User\Dong\Work\project\GUI_Tool\Download_r020\exe_output\download_release\opl1000.bin
Patch image check_sum = 0x00A127DF
Patch image size = 0x0001DCC4 (122052 bytes) OTA file: D:
\User\Dong\Work\project\GUI_Tool\Download_r020\exe_output\download_release\opl1000_ota.bin is built.

Load Patch bin file: D:
\User\Dong\Work\project\GUI_Tool\Download_r020\exe_output\download_release\opl1000_ota.bin
Open  COM13  @115200  bps  successfully.

Download D:\User\Dong\Work\project\GUI_Tool\Download_r020\exe_output\download_release\opl1000_ota.bin
with speed-up mode.
```

After download has been completed, the phrase "This image is from MW_OTA[0]" will appear on Debug UART, indicating the success in generating and downloading OTA firmware. This is shown in Figure 8.

Figure 8: The output log message from execution of OTA Image



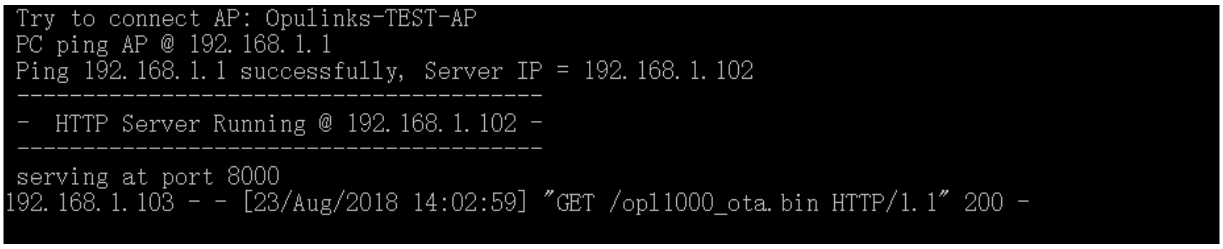
```
The init of MW_FIM is done.

[SVN REV] SVN_REVISION:1655

[Lib] SDK version info: 2347
[Lib] Compile time: 2018/08/16 19:40:38
wifiMac Task create successful
Supplicant task is created successfully!
controller_queue creates successful!
controller_queue_ble creates successful!
controller_task_create successful!
LE Task create successful
[SYS_COMMON]: Read mac address from flash.
WIFI sta cfg REQ idx=1, value=0!
This image is from MW_OTA [0].
Sw patch is changed successfully.
Event loop Task create successful
user_app Task create successful
wifi_mac_task_init entry
controller_task is going!

MwFim_FileRead lecfg OK

controller_wifi_cmd_handler!!!
controller_wifi_cmd_handler!!!

Wi-Fi Start
```

Please refer to reference [2] Download Tool User Guide for the use of downloaded tool. Please refer to reference [1] DEVKIT Quick Start Guide for the use of DEVKIT.

3.3. Use the HTTP Server from the PC

After the http server service is started from the PC, wait for the Get request from OPL1000. Provide response after receiving the request and the feedback message from the Client terminal will be printed on the Server terminal. This is shown in Figure 9.

Figure 9: PC HTTP server

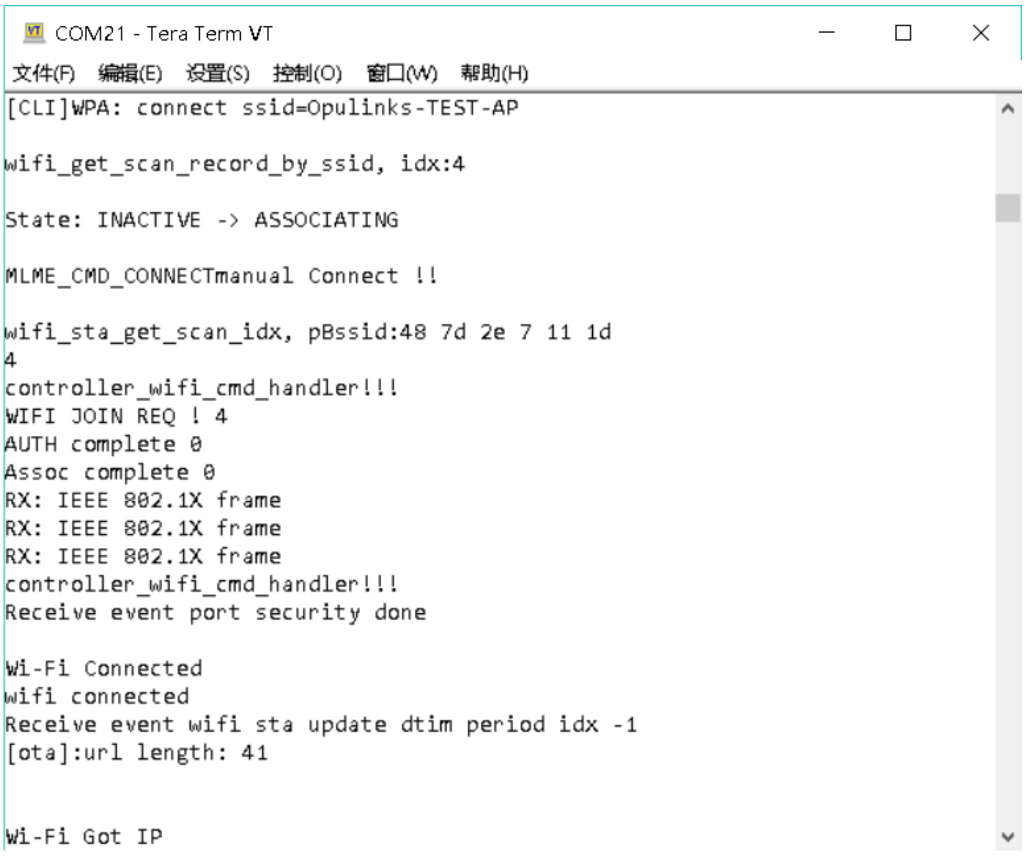


```
 Try to connect AP: Opulinks-TEST-AP
 PC ping AP @ 192.168.1.1
 Ping 192.168.1.1 successfully, Server IP = 192.168.1.102
 ----------------------------------------
 -  HTTP Server Running @ 192.168.1.102 -
 ----------------------------------------
 serving at port 8000
192.168.1.103 - - [23/Aug/2018 14:02:59] "GET /opl1000_ota.bin HTTP/1.1" 200 -
```

3.4. Execute from OPL1000 http Client Terminal

The execution status of OPL1000 http client terminal can be observed from the printed messages on Debug UART. Figure 10 shows that OPL1000 has successfully connected to WIFI AP.

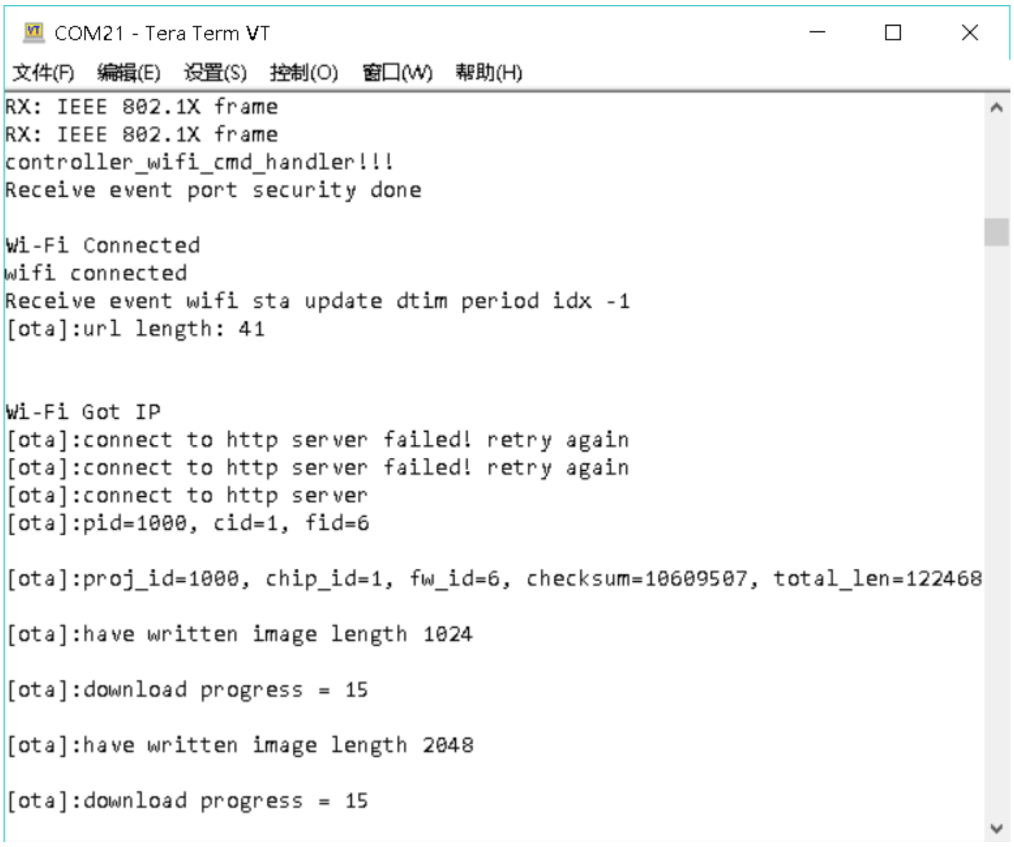
Figure 10: OPL1000 has successfully connected to WIFI AP



```
[CLI]WPA: connect ssid=Opulinks-TEST-AP

wifi_get_scan_record_by_ssid, idx:4

State: INACTIVE -> ASSOCIATING

MLME_CMD_CONNECTmanual Connect !!

wifi_sta_get_scan_idx, pBssid:48 7d 2e 7 11 1d
4
controller_wifi_cmd_handler!!!
WIFI JOIN REQ ! 4
AUTH complete 0
Assoc complete 0
RX: IEEE 802.1X frame
RX: IEEE 802.1X frame
RX: IEEE 802.1X frame
controller_wifi_cmd_handler!!!
Receive event port security done

Wi-Fi Connected
wifi connected
Receive event wifi sta update dtim period idx -1
[ota]:url length: 41


Wi-Fi Got IP
```

After OPL1000 is connected to AP, attempt to connect to the Http server. Figure 11 shows that OPL1000 has successfully connected to the http server and obtained the OTA image header message stored in Server.

Figure 11: OPL1000 has successfully connected to the http server

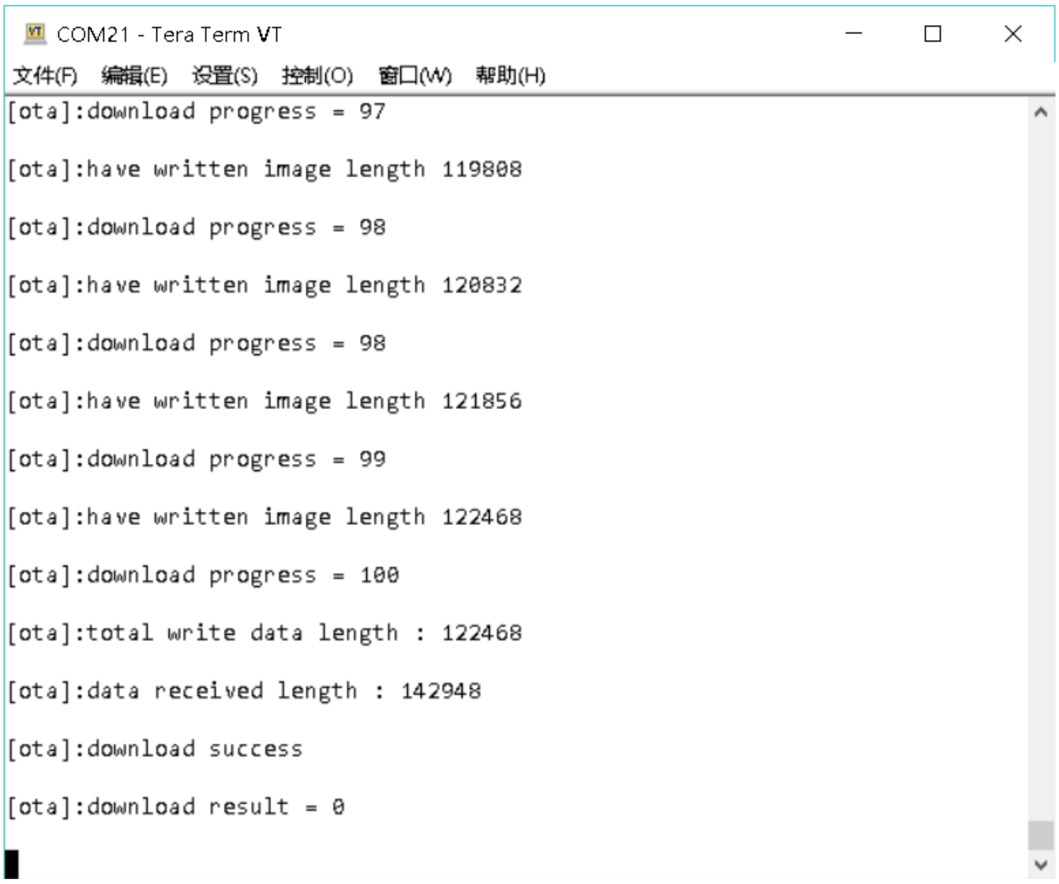


```
RX: IEEE 802.1X frame
RX: IEEE 802.1X frame
controller_wifi_cmd_handler!!!
Receive event port security done

Wi-Fi Connected
wifi connected
Receive event wifi sta update dtim period idx -1
[ota]:url length: 41


Wi-Fi Got IP
[ota]:connect to http server failed! retry again
[ota]:connect to http server failed! retry again
[ota]:connect to http server
[ota]:pid=1000, cid=1, fid=6

[ota]:proj_id=1000, chip_id=1, fw_id=6, checksum=10609507, total_len=122468

[ota]:have written image length 1024

[ota]:download progress = 15

[ota]:have written image length 2048

[ota]:download progress = 15
```

Next, Opl1000 obtain OTA image file from the http server segment by segment. If the whole process runs as normal, the download success message will be printed.

Figure 12 shows that OTA image has been successfully downloaded.

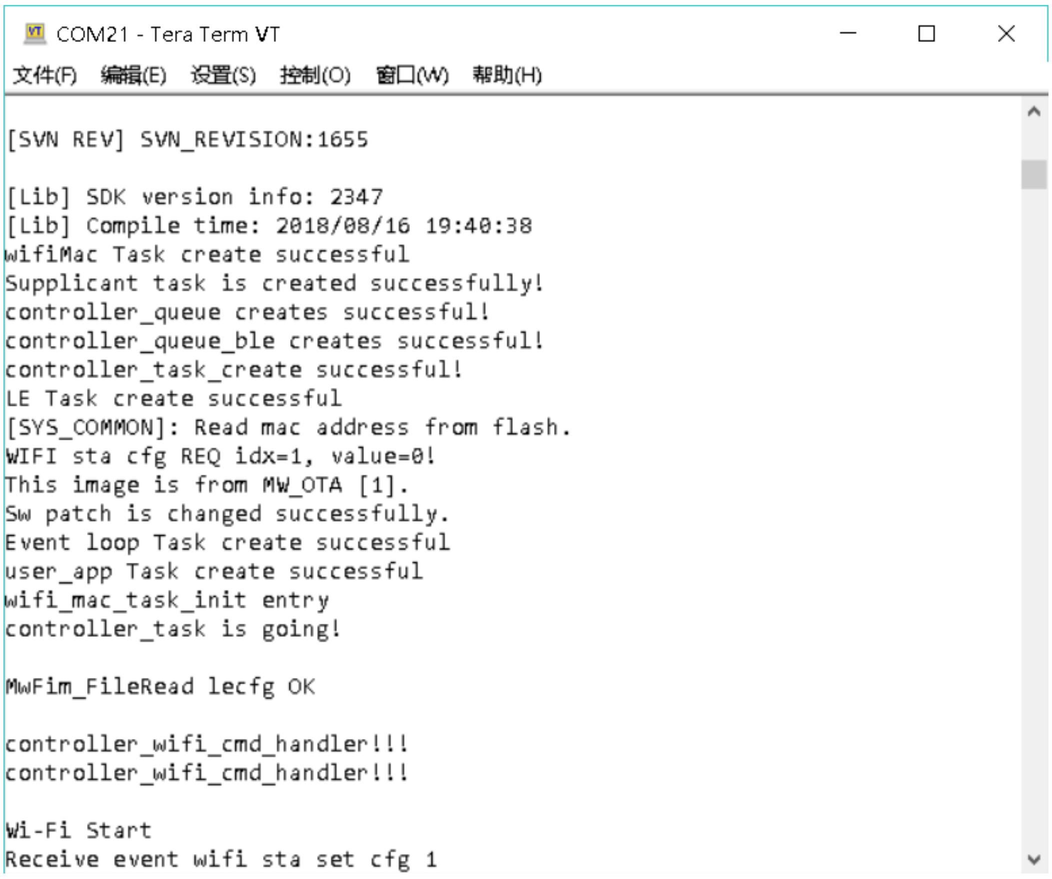
Figure 12: OPL1000 has successfully downloaded OTA image



```
[ota]:download progress = 97

[ota]:have written image length 119808

[ota]:download progress = 98

[ota]:have written image length 120832

[ota]:download progress = 98

[ota]:have written image length 121856

[ota]:download progress = 99

[ota]:have written image length 122468

[ota]:download progress = 100

[ota]:total write data length : 122468

[ota]:data received length : 142948

[ota]:download success

[ota]:download result = 0
```

Reboot OPL1000 and the downloaded OTA image file will be loaded and executed. In "Figure 8: The output log message from execution of OTA Image", the original OTA firmware is saved in MW_OTA[0] domain while the new firmware is saved in MW_OTA[1] domain. 错误!书签自引用无效。 shows the message "This image is from MW_OTA[1]" suggested the newly downloaded firmware has been executed.

Figure 13: The output message from execution of updated OTA image



```
[SVN REV] SVN_REVISION:1655

[Lib] SDK version info: 2347
[Lib] Compile time: 2018/08/16 19:40:38
wifiMac Task create successful
Supplicant task is created successfully!
controller_queue creates successful!
controller_queue_ble creates successful!
controller_task_create successful!
LE Task create successful
[SYS_COMMON]: Read mac address from flash.
WIFI sta cfg REQ idx=1, value=0!
This image is from MW_OTA [1].
Sw patch is changed successfully.
Event loop Task create successful
user_app Task create successful
wifi_mac_task_init entry
controller_task is going!

MwFim_FileRead lecfg OK

controller_wifi_cmd_handler!!!
controller_wifi_cmd_handler!!!

Wi-Fi Start
Receive event wifi sta set cfg 1
```

CONTACT

sales@Opulinks.com