OPL1000

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

TCP Client Demo User Guide

OPULINKS

http://www.opulinks.com/

OPL1000-Demo-TCP-client-guide-R01 | Version V04

REVISION HISTORY

Date	Version	Contents Updated
2018–05–31	0.1	● Initial Release
2018–07–27	0.2	● Add processing: TCP server response ACK message to client
2018–12–14	0.3	● Add skip DTIM, timeout, high power, low power definition
2019–07–19	0.4	● Update dtim description to sync with example.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Application Scope

This document describes the procedure to use SDK API to configure POL1000 as TCP Client. This allows for connection of the TCP Client with TCP server on the same network segment and perform data transmission between the two.

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
DTIM	Delivery Traffic Indication Message
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

[4] Power Consumption Measurement Guide OPL1000-Power-Consumption-Measurement-Guide.pdf

2. TCP_CLIENT PROGRAM DESIGN

2.1. Working Principle

The directory for TCP_Client sample program is under

SDK\APS_PATCH\examples\ examples\protocols\tcp_client

Its 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 successful connect, establish Socket and connect with the designated TCP Server and port number.
5 If the connection is successful, send a string to the TCP Server.

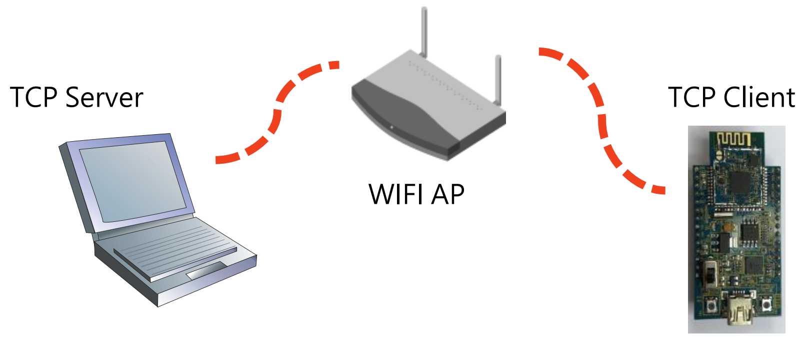
The designated AP SSID and TCP Server for connection, port number, DTIM are defined in the tc_client.h file as shown below.

```
#define WIFI_SSID              "Opulinks-TEST-AP"
#define WIFI_PASSWORD          "1234abcd"
#define TCP_SERVER_ADDR         "192.168.43.80"
#define TCP_SERVER_PORT         8181
#define  DTIM_SKIP_COUNT       29
#define  TCP_RECV_TIMEOUT     180
#define  __RF_LP_MODE__
```

The TCP data transmission network topology established between TCP server and OPL1000 is shown in Figure 1.

OPL1000plays 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.

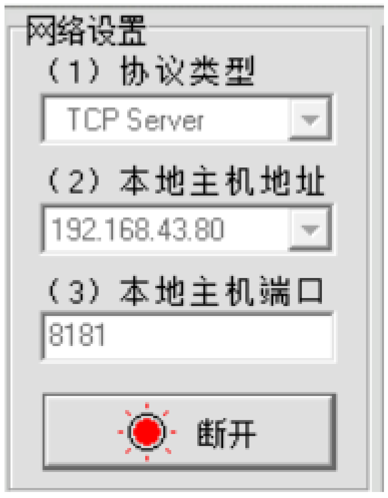
Figure 1： Sample of OPL1000 TCP client network connection diagram



Execute Network Debugging Assistant program, NetAssist.exe (path Demo\TCP_Client). In the network setup dialogue, choose TCP Server as the protocol type and fill in the IP address WIFI AP assigned to the PC. The IP address in this example is 192.168.43.80. Any number can be chosen as the local port number, but it is best not to use a known and commonly used port number such as 8080. The port number defined in this example is 8181.

The IP address and port parameter for TCP Server have been macro-defined in the TCP_Client example as TCP_SERVER_ADDR and TCP_SERVER_PORT.

Figure 2: Parameter configuration for TCP Server in Network Debugging Assistant



网络设置

（1）协议类型: TCP Server

（2）本地主机地址: 192.168.43.80

（3）本地主机端口: 8181

2.2. Call API

The description for API used in the TCP_Client 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
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
socket	Establish Socket handler
connect	Connect to the designated TCP Server and port
write	Write/transmit data to TCP Server

3. TCP CLIENT AUTHENTICATE FUNCTION

3.1. Example of Editing TCP_client

The example of editing a TCP client project includes three steps:

Step1: Firstly, check the WIFI AP that the user needs to connect to and fill in its SSID and the visiting 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 the PC to WIFI AP and use ipconfig command to check the IP address assigned to the PC by WIFI AP. Fill in the IP address of PC into the macro-defined TCP_SERVER_ADDR and TCP_SERVER_PORT.

```
#define TCP_SERVER_ADDR     "192.168.43.80"
#define TCP_SERVER_PORT     8181
```

Step3: Add DTIM parameter and setting DTIM interval, and allow AP to write the multicast flow based on that interval; 29 (wake up time is 30*0.1s=3s), 29 represents the values skipped by OPL1000. Only the 30th value will be received and the 0.1s represents the beacon interval time for AP.

```
#define DTIM_SKIP_COUNT   29
```

Step4:Add timeout parameter :

```
#define TCP_RECV_TIMEOUT   180
```

Step5: Define the current RF state. If it is in high power state, no additional definition is required as the default state is high power state. If it is in low power state, the following parameter should be added.

```
#define __RF_LP_MODE__
```

Step6: Use Keil C to edit TCP client project and the path for the project file is

SDK\APS_PATCH\examples\protocols\tcp_client\opl1000_app_m3.uvprojx

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

3.2. Firmware Download

After a successful edit, opl1000_app_m3.bin file will be produced under the SDK\APS_PATCH\examples\protocols\tcp_client\Output\Objects directory. Copy the file to FW_binary directory and use the download tool to combine it with the m0 bin file. This is followed by downloading the resultant file into DEVKIT. 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. Execute Network Debugging Assistant from the PC

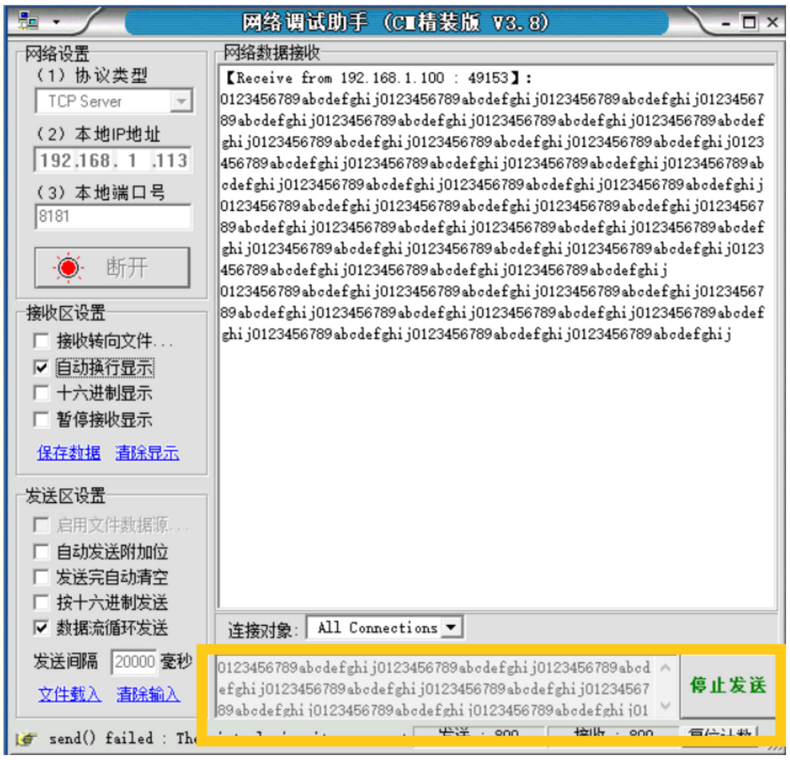
Start the Network Debugging Assistant procedure from the PC. Select TCP_server protocol and fill in the IP address and port number. Please note that the IP address and port number should be the same as TCP_SERVER_ADDR and TCP_SERVER_PORT as defined in the tcp_client.h file. Click on the "connect" button and start TCP Server. Reset OPL1000 DEVKIT after ensuring that WIFI AP is working normally . It will automatically try to connect to WIFI AP. After it is successful, it will attempt data communication with TCP Server.

Construct TCP Server based on the aforementioned environment. After connecting to the wireless router, use wifi module as TCP Client and set the Server to send a data package to the Client every two minutes. The size of the package is 200 bytes. The Client COM port will print the data and return it to TCP Server and measure average current over 10 hours. Please refer to reference [4] Power Consumption Measurement Guide. DTiM interval is set as 3s.

The loading method for the two hundred bytes is to click on the "Load File" button to load the file. The file is the measurement data for the 200 bytes. That is the data for writing 200 bytes in a loop every 120s.

Figure 3 showcases the result of the successful communication. The data reception window will print strings continuously. The circled portion in the lower part of the figure is the message written by OPL1000.

Figure 3: Example using OPL1000 as TCP Client for communication



(1) 协议类型: TCP Server
(2) 本地IP地址: 192.168. 1 .113
(3) 本地端口号: 8181

☐ 接收转向文件...
☑ 自动换行显示
☐ 十六进制显示
☐ 暂停接收显示

☐ 启用文件数据源...
☐ 自动发送附加位
☐ 发送完自动清空
☐ 按十六进制发送
☑ 数据流循环发送
发送间隔 20000 毫秒

```
【Receive from 192.168.1.100 : 49153】:
0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij01234567
89abcdefghij0123456789abcdefghij0123456789abcdefghij0123456789abcdef
ghij0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij0123
456789abcdefghij0123456789abcdefghij0123456789abcdefghij0123456789ab
cdefghij0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij
0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij01234567
89abcdefghij0123456789abcdefghij0123456789abcdefghij0123456789abcdef
ghij0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij0123
456789abcdefghij0123456789abcdefghij0123456789abcdefghij
0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij01234567
89abcdefghij0123456789abcdefghij0123456789abcdefghij0123456789abcdef
ghij0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij
```

连接对象: All Connections

```
0123456789abcdefghij0123456789abcdefghij0123456789abcd
efghij0123456789abcdefghij0123456789abcdefghij01234567
89abcdefghij0123456789abcdefghij0123456789abcdefghij01
```

```
send() failed : The
```

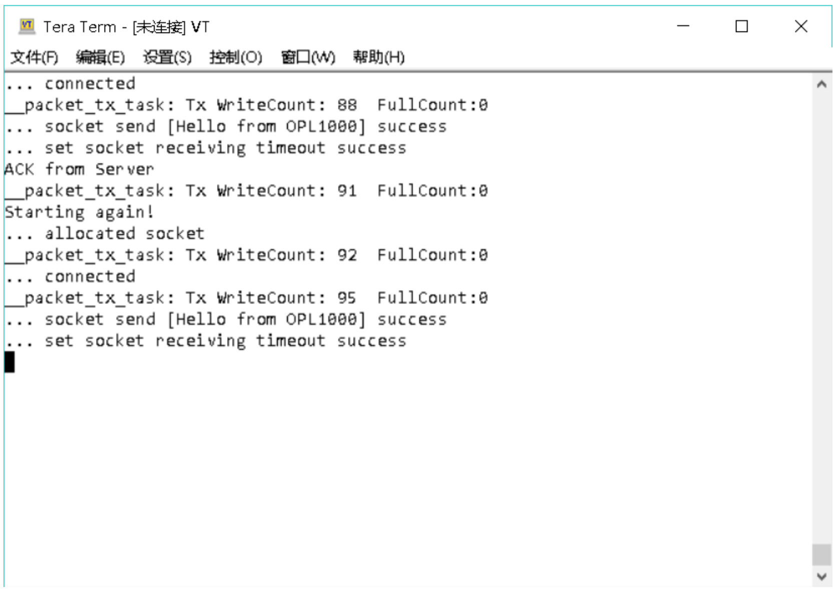
发送 : 800 接收 : 800

After receiving the 200 bytes data from OPL1000 Client terminal, enter “ACK from Server” in the send window for Network Debugging Assistant software and press the “Send” button. The Client terminal will receive the corresponding string.

Figure 4 showcases OPL1000 printing the information

received from the port. It can be seen that OPL1000 has received the “ACK from Server” response message from the Server terminal.

Figure 4: Message received by OPL1000 Client terminal



```
... connected
__packet_tx_task: Tx WriteCount: 88  FullCount:0
... socket send [Hello from OPL1000] success
... set socket receiving timeout success
ACK from Server
__packet_tx_task: Tx WriteCount: 91  FullCount:0
Starting again!
... allocated socket
__packet_tx_task: Tx WriteCount: 92  FullCount:0
... connected
__packet_tx_task: Tx WriteCount: 95  FullCount:0
... socket send [Hello from OPL1000] success
... set socket receiving timeout success
```

In order to ensure the right execution, please use Ping command from the PC to verify if the OPL1000 connection is normal before executing "Network Debugging Assistant routine". If it is successful, it means that they are in the same network segment.

```
C: \>ping 192.168.43.150

正在 Ping 192.168.43.150 具有 32 字节的数据:
来自 192.168.43.150 的回复: 字节=32 时间=133ms TTL=255
来自 192.168.43.150 的回复: 字节=32 时间=44ms TTL=255
来自 192.168.43.150 的回复: 字节=32 时间=162ms TTL=255
来自 192.168.43.150 的回复: 字节=32 时间=174ms TTL=255

192.168.43.150 的 Ping 统计信息:
    数据包: 已发送 = 4，已接收 = 4，丢失 = 0 (0% 丢失)，
往返行程的估计时间(以毫秒为单位):
    最短 = 44ms，最长 = 174ms，平均 = 128ms
```

CONTACT

sales@Opulinks.com