

WS63V100 HTTP

开发指南

文档版本 02

发布日期 2024-10-12

前言

概述

本文档主要介绍 HTTP Client 功能开发实现示例。

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
WS63	V100

读者对象





本文档主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
 危险	表示如不避免则将会导致死亡或严重伤害的具有高等级风险的危

符号	说明
	害。
 警告	表示如不可避免则可能导致死亡或严重伤害的具有中等级风险的危害。
 注意	表示如不可避免则可能导致轻微或中度伤害的具有低等级风险的危害。
 须知	用于传递设备或环境安全警示信息。如不可避免则可能会导致设备损坏、数据丢失、设备性能降低或其它不可预知的结果。 “须知”不涉及人身伤害。
 说明	对正文中重点信息的补充说明。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害信息。

修改记录

文档版本	发布日期	修改说明
02	2024-10-12	更新 “1.1 概述” 小节内容。
01	2024-04-10	第一次正式版本发布。
00B01	2024-03-15	第一次临时版本发布。

目 录

前言i

1 开发指南.....1

1.1 概述1

1.2 代码示例1

1 开发指南

1.1 概述

1.2 代码示例

1.1 概述

HTTP 示例通过 lwIP 提供的 API 完成对指定 IP 的建立链接，获取网页的功能。

说明

HTTP 实现不新增额外的 API，仅依赖 lwIP 提供的 API 接口。相关接口说明请参见《WS63V100lwIP 开发指南》。

1.2 代码示例

以下为 HTTP Client 主动获取用户传入 IP 地址首页的代码示例：

```
#include <stdio.h>
#include "lwip/sockets.h"
#include "lwip/netdb.h"

#define HTTPC_DEMO_RECV_BUFSIZE 64
#define SOCK_TARGET_PORT 80
static const char *g_request = "GET / HTTP/1.1\r\n\
    Content-Type: application/x-www-form-urlencoded;charset=UTF-8\r\n\
    Host: baidu.com\r\n\
    Connection: close\r\n\
    \r\n";
```

```
unsigned int http_client_get(int argc, char* argv[])
{
    if ((argc != 1) || (argv == NULL)) {
        return 1;
    }
    struct sockaddr_in addr = {0};
    int s, r;
    char recv_buf[HTTPC_DEMO_RECV_BUFSIZE];
    addr.sin_family = AF_INET;
    addr.sin_port = PP_HTONS(SOCK_TARGET_PORT);
    addr.sin_addr.s_addr = inet_addr(argv[0]);
    s = socket(AF_INET, SOCK_STREAM, 0);
    if (s < 0) {
        return 1;
    }
    printf("... allocated socket");
    if (connect(s, (struct sockaddr*)&addr, sizeof(addr)) != 0) {
        printf("... socket connect failed errno=%d", errno);
        lwip_close(s);
        return 1;
    }
    printf("... connected");
    if (lwip_write(s, g_request, strlen(g_request)) < 0) {
        lwip_close(s);
        return 1;
    }
    printf("... socket send success");
    struct timeval receiving_timeout;
    /* 5S Timeout */
    receiving_timeout.tv_sec = 5;
    receiving_timeout.tv_usec = 0;
    if (setsockopt(s, SOL_SOCKET, SO_RCVTIMEO, &receiving_timeout,
sizeof(receiving_timeout)) < 0) {
        printf("... failed to set socket receiving timeout");
        lwip_close(s);
        return 1;
    }
    printf("... set socket receiving timeout success");
    /* Read HTTP response */
    do {
        (void)memset_s(recv_buf, sizeof(recv_buf), 0, sizeof(recv_buf));
        r = lwip_read(s, recv_buf, sizeof(recv_buf) - 1);
        for (int i = 0; i < r; i++) {
```

```
        putchar(recv_buf[i]);
    }
} while (r > 0);
printf("... done reading from socket. Last read return=%d errno=%d\r\n", r, errno);
lwip_close(s);
return 0;
}
```