



Hi3861V100 / Hi3861LV100 HTTP

开发指南

文档版本 01

发布日期 2020-04-30

版权所有 © 上海海思技术有限公司2020。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HISILICON、海思和其他海思商标均为海思技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受海思公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，海思公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

上海海思技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编： 518129

网址： <https://www.hisilicon.com/cn/>

客户服务邮箱： support@hisilicon.com



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
Hi3861	V100
Hi3861L	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	表示如不可避免则将会导致死亡或严重伤害的具有高等级风险的危害。
 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。



符号	说明
 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
01	2020-04-30	第一次正式版本发布。
00B01	2020-01-15	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《Hi3861V100 / Hi3861LV100 lwIP 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 "hi_stdlib.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;
}
```