QuecPython Socket 应用指导

LTE Standard 模块系列

版本：1.0.0

日期：2020-11-12

状态：临时文件

上海移远通信技术股份有限公司始终以为客户提供最及时、最全面的服务为宗旨。如需任何帮助，请随时联系我司上海总部，联系方式如下：

上海移远通信技术股份有限公司
上海市闵行区田林路 1016 号科技绿洲 3 期（B 区）5 号楼　　邮编：200233
电话：+86 21 51086236　　邮箱：info@quectel.com

或联系我司当地办事处，详情请登录：http://www.quectel.com/cn/support/sales.htm。

如需技术支持或反馈我司技术文档中的问题，可随时登陆如下网址：
http://www.quectel.com/cn/support/technical.htm 或发送邮件至：support@quectel.com。

前言

上海移远通信技术股份有限公司提供该文档内容用以支持其客户的产品设计。客户须按照文档中提供的规范、参数来设计其产品。因未能遵守有关操作或设计规范而造成的损害，上海移远通信技术股份有限公司不承担任何责任。在未声明前，上海移远通信技术股份有限公司有权对该文档进行更新。

免责声明

上海移远通信技术股份有限公司尽力确保开发中功能的完整性、准确性、及时性或效用，但不排除上述功能错误或遗漏的可能。除非其他有效协议另有规定，否则上海移远通信技术股份有限公司对开发中功能的使用不做任何暗示或明示的保证。在适用法律允许的最大范围内，上海移远通信技术股份有限公司不对任何因使用开发中功能而遭受的损失或损害承担责任，无论此类损失或损害是否可以预见。

保密义务

除非上海移远通信技术股份有限公司特别授权，否则我司所提供文档和信息的接收方须对接收的文档和信息保密，不得将其用于除本项目的实施与开展以外的任何其他目的。未经上海移远通信技术股份有限公司书面同意，不得获取、使用或向第三方泄露我司所提供的文档和信息。对于任何违反保密义务、未经授权使用或以其他非法形式恶意使用所述文档和信息的违法侵权行为，上海移远通信技术股份有限公司有权追究法律责任。

版权申明

本文档版权属于上海移远通信技术股份有限公司，任何人未经我司允许而复制转载该文档将承担法律责任。

文档历史

修订记录

版本	日期	作者	变更表述
-	2020-11-12	Rivern/ Kingka	文档创建
1.0.0	2020-11-12	Rivern/ Kingka	临时版本

目录

1 引言

本文档主要以移远通信 EC100Y-CN 模块为例介绍如何使用 QuecPython 类库 API，实现基本的 Socket 通讯功能。

本文档适用以下移远通信模块：

- EC100Y-CN
- EC600S-CN

2 Socket 概述

2.1. Socket 简介

所谓 Socket（套接字），就是对网络中不同主机上的应用进程之间进行双向通信的端点的抽象。一个 Socket 就是网络上进程通信的一端，提供了应用层进程利用网络协议交换数据的机制。从所处的地位来讲，Socket 上联应用进程，下联网络协议栈，是应用程序通过网络协议进行通信的接口，是应用程序与网络协议根进行交互的接口。

Socket 可以看成是两个网络应用程序进行通信时，各自通信连接中的端点，这是一个逻辑上的概念。它是网络环境中进程间通信的 API（应用程序编程接口），也是可以被命名和寻址的通信端点，使用中的每一个 Socket 都有其类型和一个与之相连进程。通信时其中一个网络应用程序将要传输的一段信息写入它所在主机的 Socket 中，该 Socket 通过与网络接口卡（NIC）相连的传输介质将这段信息送到另外一台主机的 Socket 中，使对方能够接收到这段信息。Socket 是由 IP 地址和端口结合的，提供向应用层进程传送数据包的机制。

2.2. Socket 的应用

Socket 可以使一个应用从网络中读取和写入数据，不同计算机上的两个应用可以通过连接发送和接受字节流，注意，当发送消息时，需要知道对方的 IP 和端口。在日常生活中有很多应用场景，当你浏览网页时，浏览器进程怎么与 web 服务器进程通信；当你用 QQ 聊天时，QQ 进程怎么与服务器或好友所在的 QQ 进程通信，这些都是通过 socket 来实现的。

2.3. QuecPython Socket API 详解

Socket 起源于 Unix，而 Unix/Linux 基本哲学之一就是“一切皆文件”，都可以用“打开（open）→读写（write/read）→关闭（close）”模式来操作。在实现过程中服务端可以看作是 web 服务器，客户端可以看作是要访问 web 服务器的浏览器，访问过程就可以和打开→读写→关闭一一对应。

QuecPython 类库中通过 usocket 实现 Socket 功能，usocket 模块提供对 BSD 套接字接口的访问。该模块实现相应 CPython 模块的子集，更多信息请参阅 CPython 文档：socket。其中 usocket 对 Socket 功能的具体流程及实现的相关 API 介绍如下。

2.3.1. usocket.socket

该函数用于服务端或客户端创建一个 Socket 对象。

- **函数原型**

```
sock=usocket.socket(usocket.AF_INET, usocket.SOCK_STREAM)
```

- **参数**

usocket.AF_INET：
网络协议，IPv4

usocket.SOCK_STREAM：
对应 TCP 的流式 Socket。更多常量定义详见 Quectel QuecPython 类库 API 说明。

- **返回值**

无。

2.3.2. usock.getaddrinfo

该函数用于将主机域名（host）和端口（port）转换为用于创建套接字的 5 元组序列，元组结构如下：(family, type, proto, canonname, sockaddr)

- **函数原型**

```
usocket.getaddrinfo(host, port)
```

- **参数**

host：
主机域名。

port：
端口。

- **返回值**

无。

2.3.3. sock.bind

该函数用于将 Socket 对象和服务端 IP：端口绑定。由于 TCP 口是动态的，客户端无需进行绑定。使用该函数之前，Socket 必须未进行绑定。

- **函数原型**

```
sock.bind(address)
```

- **参数**

address：
由地址端口号组成的列表或者元组。

- **返回值**

无。

2.3.4. sock.listen

该函数允许服务端接受 Socket 连接，可指定最大连接数。

- **函数原型**

```
sock.listen(backlog)
```

- **参数**

backlog：
接受的最大 Socket 连接数，至少为 0。

- **返回值**

无。

2.3.5. sock.accept

该函数用于服务端接受客户端连接请求。

- **函数原型**

```
(conn, addres) = sock.accept()
```

- **参数**

无。

- **返回值**

返回元组，包含新的 Socket 和客户端地址，形式为：(conn, address)。

conn：
新的 Socket 对象，用来发送和接收数据。

address：
连接到服务器的客户端地址。

2.3.6. sock.connect

客户端使用该函数连接到指定地址的服务器。

- **函数原型**

```
sock.connect(address)
```

- **参数**

address：
连接到客户端的服务端地址。

- **返回值**

无。

2.3.7. sock.recv

该函数用于接受客户端或服务端发送的数据。

- **函数原型**

```
recv_data = sock.recv(bufsize)
```

- **参数**

bufsize：
一次接收的最大数据量。

- **返回值**

返回值是一个字节对象，表示接收到的数据。

2.3.8. sock.send

该函数用于发送数据到服务端或客户端。

- **函数原型**

```
sock.send(send_data.encode("utf8"))
```

- **参数**

send_data：
表示要发送的数据。

- **返回值**

返回实际发送的字节数。

备注

TCP 协议的 Socket 是基于字节流的，通过 Socket 发送数据之前，需先使用 encode("utf8")对数据进行编码，其中"utf8"为编码方式。

2.3.9. sock.close

该函数用于关闭 Socket 通信。

- **函数原型**

```
sock.close()
```

- **参数**

无。

- **返回值**

无。

2.3.10. sock.read

该函数用于从 Socket 中读取 *size* 字节数据，如果没有指定 *size*，则会从套接字读取所有可读数据，直到读取到数据结束。

- **函数原型**

```
socket.read([ size ])
```

- **参数**

[size]：
要读取的字节数。

- **返回值**

字节对象。

2.3.11. sock.readinto

该函数用于将字节读取到缓冲区中。

- **函数原型**

```
sock.readinto(buf [ , nbytes ])
```

- **参数**

buf：
存放读取字节的缓冲区。

nbytes：
读取的字节数。

- **返回值**

实际读取的字节数。

2.3.12. sock.readline

该函数用于按行读取数据，遇到换行符结束。

- **函数原型**

```
sock.readline()
```

- **参数**

无。

- **返回值**

返回读取的数据行。

2.3.13. sock.write

该函数用于向缓存区写入数据。

- **函数原型**

```
sock.write(buf)
```

- **参数**

buf：
写入缓冲区的数据。

- **返回值**

返回实际写入的字节数。

2.3.14. sock.sendall

该函数用于将所有数据都发送到 Socket。

- **函数原型**

```
sock.sendall(bytes)
```

- **参数**

bytes：
缓存 buffer，存放 bytes 型数据。

- **返回值**

无。

2.3.15. sock.sendto

该函数用于将数据发送到 Socket。该 Socket 不应连接到远程 Socket，因为目标 Socket 是由 *address* 指定的。

- **函数原型**

```
sock.sendto(bytes, address)
```

- **参数**

bytes：
缓存 buffer，存放 bytes 型数据。

address：
包含地址和端口号的元组或列表。

- **返回值**

无。

2.3.16. sock.recvfrom

该函数用于从 Socket 接收数据。返回一个元组，包含字节对象和地址。

- **函数原型**

```
socket.recvfrom(bufsize)
```

- **参数**

bufsize：
接收的缓存数据。

- **返回值**

返回一个元组，包含字节对象和地址，形式为：(bytes, address)。

bytes：
接收数据的字节对象。

address：
发送数据的 Socket 的地址。

2.3.17. sock.setsockopt

该函数用于设置 socket 选项的值。

- **函数原型**

```
socket.setsockopt(level, optname, value)
```

- **参数**

level：
socket 选项级别。

optname：
socket 选项。

value：
既可以是一个整数，也可以是一个表示缓冲区的 bytes 类对象。

- **返回值**

无。

2.3.18. sock.setblocking

该函数用于设置 Socket 为阻塞模式或者非阻塞模式。如果 *flag* 为 false，则将 Socket 设置为非阻塞模式，否则设置为阻塞模式。

- **函数原型**

```
socket.setblocking(flag)
```

- **参数**

flag：
Ture　　阻塞模式
False　　非阻塞模式

- **返回值**

无。

2.3.19. sock.settimeout

该函数用于设置 Socket 的超时时间，单位：秒。

- **函数原型**

```
socket.settimeout(value)
```

- **参数**

value：
可以是秒的非负浮点数，也可以是 None。如果将其设置为一个非零值，OSError 在该操作完成之前已超过超时时间值，则随后的 Socket 操作将引发异常；如果将其设置为零，则将 Socket 置于非阻塞模式。如果未指定该值，则 Socket 将处于阻塞模式。

- **返回值**

无。

2.3.20. Socket.makefile

该函数用于生成一个文件与 socket 对象关联，之后即可像读取普通文件一样使用 socket。（普通文件的操作有 open 和 write 等。）

- **函数原型**

```
socket.makefile(mode='rb')
```

- **参数**

mode：
二进制模式（rb 和 wb）。

- **返回值**

返回与套接字关联的文件对象。

3 Socket 功能实现

为了使用户更清楚的了解 Socket 功能，本章节提供了一个在 QuecPython 上创建 Socket 的实例，即模拟浏览器访问 web 服务器获取网页内容。首先在 Xshell 中，连接模块主串口，进入交互界面，然后按如下步骤实现 Socket 功能：

1. 导入 usocket 模块，创建一个 Socket 实例：

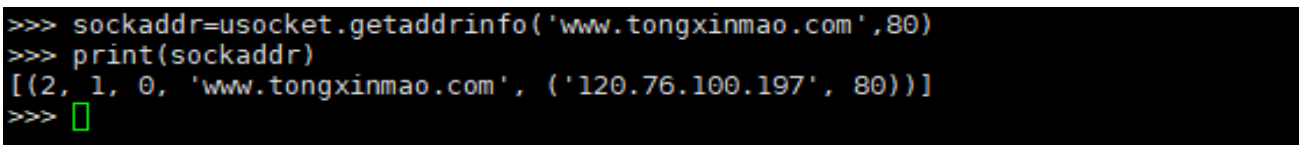
```
import usocket
sock = usocket.socket(usocket.AF_INET, usocket.SOCK_STREAM)
```

2. 解析域名：

```
sockaddr=usocket.getaddrinfo('www.tongxinmao.com',80)[0][-1]
```

将主机域名（host）和端口（port）转换为用于创建 Socket 的 5 元组序列，元组结构如下：

(family, type, proto, canonname, sockaddr)



3. 建立与服务端的连接：

```
sock.connect(sockaddr)
```

```
>>> #
>>> sock.connect(sockaddr)
>>>
```

4. 向服务端发送消息：

```
ret=sock.send('GET /News HTTP/1.1\r\nHost: www.tongxinmao.com\r\nAccept-Encoding: deflate\r\nConnection: keep-alive\r\n\r\n')
print('send %d bytes' % ret)
```

```
>>> ret=sock.send('GET /News HTTP/1.1\r\nHost: www.tongxinmao.com\r\nAccept-Encoding: deflate\r\nConnection: keep-alive\r\n\r\n')
>>> print('send %d bytes' % ret)
send 98 bytes
>>>
```

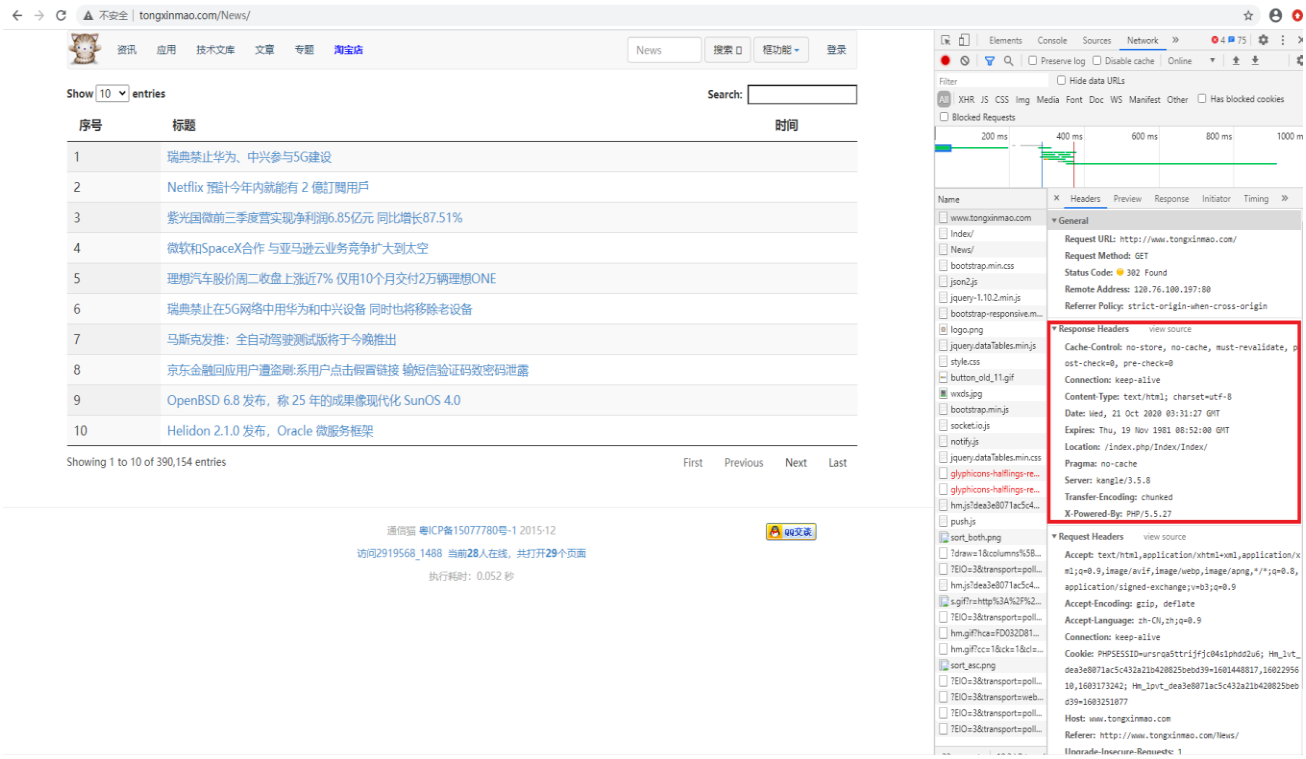
5. 接收服务端的消息：

```
data=sock.recv(1024)
print('recv %s bytes:' % len(data))
print(data.decode())
```

```
>>> ret=sock.send('GET /News HTTP/1.1\r\nHost: www.tongxinmao.com\r\nAccept-Encoding: deflate\r\nConnection: keep-alive\r\n\r\n')
>>> print('send %d bytes' % ret)
send 98 bytes
>>>
>>> data=sock.recv(1024)
>>> print('recv %s bytes:' % len(data))
recv 1024 bytes:
>>> print(data.decode())
HTTP/1.1 200 OK
Server: kangle/3.5.8
Date: Wed, 21 Oct 2020 03:42:33 GMT
Set-Cookie: PHPSESSID=bu0l6f311c0ml153k8g8nsq531; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Content-Type: text/html; charset=utf-8
X-Powered-By: SBT
X-Server: SBT
Server: SBT
Transfer-Encoding: chunked
Connection: keep-alive

1ee0
<!DOCTYPE html>
<html lang="zh">
<head>
<meta http-equiv="Content-Type" content="text/html;charset=utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<meta name="author" content="tcml23@126.com">
<meta name ="keywords" content="通信猫">
<meta name="description" content="通信猫 ">
<title>通信猫--业界资讯</title>
<link rel="stylesheet" href="//tongxinmao.com/assets/css/bootstrap.min.css">
<link href="//tongxinmao.com/assets/css/bootstrap-responsive.min.css" rel="stylesheet">
<link href="//tongxinmao.com/assets/css/style.css" rel="stylesheet">
<script src='//tongx
>>>
```

服务端消息接收完成后，可在浏览器上发起请求，验证返回消息是否与 Socket 接收的消息一致，如下所示：



6. 关闭连接：

```
sock.close()
```

```
>>> sock.close()
>>>
```

以上部分代码可见于移远通信提供的 SDK 工具包中，路径为 *moudles/socket/example_socket.py*，也可通过 example 模块来执行该脚本文件。

usocket 服务端功能实现代码如下图：

```
#导入 usocket 模块
import usocket

#创建一个 socket 实例
sock = usocket.socket(usocket.AF_INET, usocket.SOCK_STREAM)
#设置端口复用
sock.setsockopt(usocket.SOL_SOCKET, usocket.SO_REUSEADDR, 1)

sock.bind(('127.0.0.1', 6000))

sock.listen(50)
```

```
while True:
    newSock, addr = sock.accept()
    newSock.send('hello world')
    recv_data = newSock.recv(256)
    print(recv_data.decode())
    newSock.close()
    break
```

usocket-API 说明文档，详见 qpy.quectel.com/wiki/#/zh-cn/api/。

4 附录

表 1：术语缩写

术语	英文全称	中文全称
API	Application Programming Interface	应用程序编程接口
BSD	Berkeley Socket	Berkeley 套接字
HTTP	Hypertext Transfer Protocol	超文本传输协议
IPv4	Internet Protocol version 4	第 4 版互联网协议
NIC	Network Interface Controller	网络接口控制器
SDK	Software Development Kit	软件开发工具包
TCP	Transmission Control Protocol	传输控制协议