ESP32 SERIES

Getting Started Guide

for SDK based on FreeRTOS

Version 0.1

About This Guide

This guide describes the software, firmware environment and development recommendations required to build an application around the ESP32 development board with build-in ESP32 chip.

> Note :
>
> This guide is only for SDK Verison 1.0.0. Refer to the “ESP32 System Description” for more details on the ESP32 development board.

This document is structured as follows:

Chapter	Title	Subject
Chapter 1	Introduction	Introduces the ESP32 and software and hardware required. Explains the software package with details on its architecture and contents.
Chapter 2	Getting Started	Guides users how to build and upload an application firmware provided by Espressif.

Style Identifiers

This is a code input style.

```
Code Input
```

This is a code output style.

```
Code output
```

Release Notes

Date	Version	Release notes
2015.12	V0.1	Confidential for internal use only.

Table of Contents

1. Introduction

1.1. ESP32

ESP32 provides a Wi-Fi plus Bluetooth 4.2 combo solution in the 2.4GHz band by using 40nm technology. It delivers highly integrated Wi-Fi SoC solution to meet the continuous demands for efficient power usage, compact design and reliable performance.

You can use ESP32 series hardware and software provided by Espressif to develop and program some useful Wi-Fi, Bluetooth and wearable electronics applications and products in Internet of Things(IoT) industry.

1.2. Requirements

Before getting started to run and develop firmware applications on your ESP32, there are some hardware and software requirements.

1.2.1. Hardware

- An ESP32 development board or ESP-WROOM-03(build-in ESP32).
- A USB TTL Serial cable or a Micro-USB cable.
- Debian GNU/Linux OS on an x86 machine.

Notes :

1. We assume to use the Debian or Ubuntu Linux OS. You may make some modifications for the other Linux OS.
2. For more information about ESP-WROOM-03, refer to "**ESP-WROOM-03 Specification**".

ESP32 Development Board

ESP32 development board (hereafter referred to as a board) is shown as Figure 1-1 and Table 1-1.

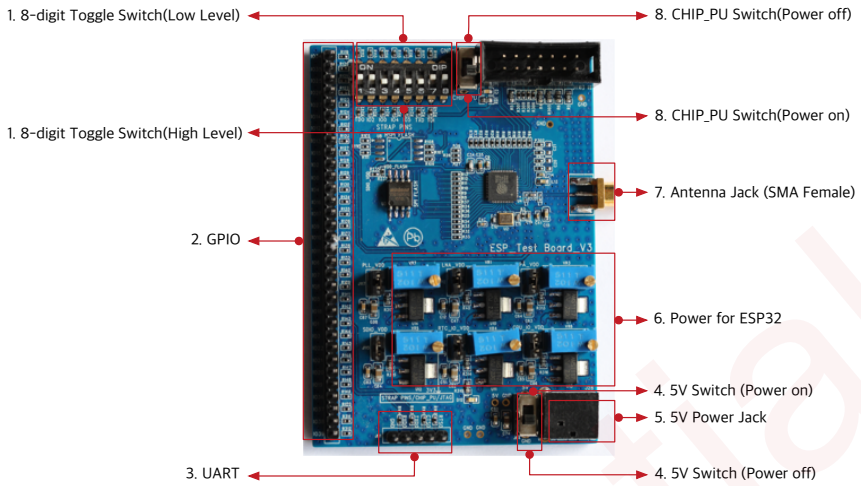

Figure 1-1. ESP32 Development Board

Table 1-1. ESP32 Development Board

No.	Name	Description
1	8-digit Toggle Switch	Set the board to the SPI and the uploading mode with the 8-digit Toggle Switch. More information refer to **Table 1-2**.
2	GPIO	-
3	UART	Upload the firmware with UART.
4	5V Switch	GND: Power off 5V: Power on
5	5V Power Jack(DC-044)	Connect the board with a 5V/1A(Max) Dc power cable.
6	Power for ESP32	-
7	Antenna Jack	Connect the antenna to the board with the SMA Female Jack.
8	CHIP_PU Switch	GND: Power off CHIP_PU: Power on

Table 1-2. 8-digit Toggle Switch

Switch No.	1	2	3	4	5	6
Pin	MTDO	GPIO2	GPIO0	GPIO4	GPIO5	U0TXD
SPI Mode	x	x	High Level	x	High Level	High Level
Uploading Mode	x	Low Level	Low Level	x	x	x

1.2.2. Software

Development Platform

ESP32 RTOS SDK is a development platform provided for the users to develop their project.

- For the latest version, go to **Download ESP32 RTOS SDK in Github**.
- For the package components, refer to Figure 1-2.

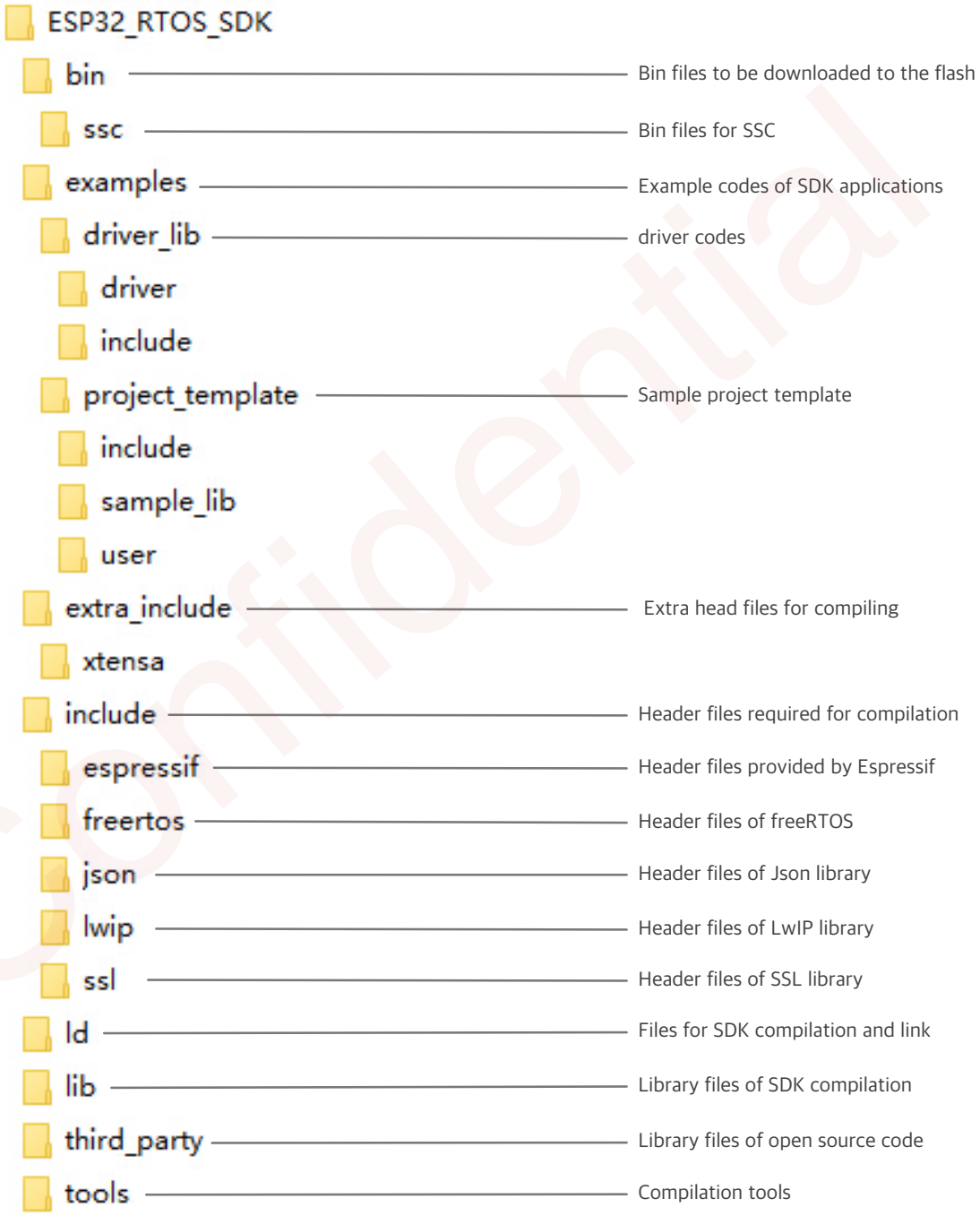

Figure 1-2. Package Components

Serial Terminal Tool

We recommend using **Minicom** as a serial terminal tool to show the communication information and debug code.

Uploading Tool

We recommend using **Python 2.6** or **2.7** and **esptool_ESP32.py** to upload firmware to the flash.

- For the latest version of **Python**, go to **Download Python**.
- **esptool_ESP32.py** file is in the **ESP32 RTOS SDK > tools** folder, refer to **Figure 1-2. Package Components**.
- For the details of instructions, refer to **2.5.2 Uploading Files**.

2. Getting Started

2.1. Getting Started to Use SDK

2.1.1. Process Overview

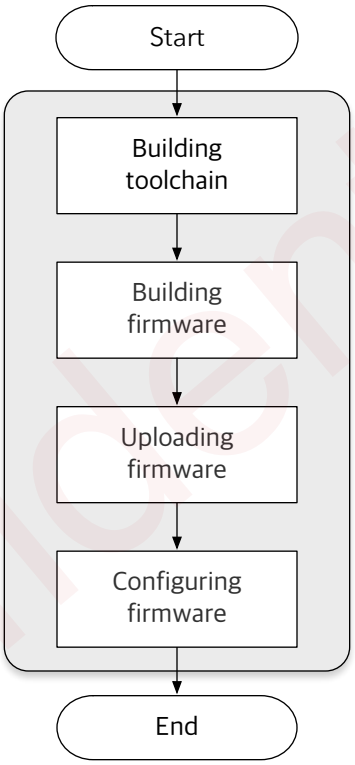

Figure 2-1. Process Overview

2.1.2. Building Toolchain

We suggest to choose **Crosstool-ng** as the compiler toolchain. Follow the instructions below to install **Crosstool-ng**.

> Note :
>
> Your PC needs to connect to the Internet.

1. Update the system.

```
sudo apt-get update
```

2. **Install the required toolchain packages.**

```
sudo apt-get install git autoconf build-essential gperf bison flex
texinfo libtool libncurses5-dev wget gawk libc6 python-serial
libexpat-dev
```

```
2 upgraded, 37 newly installed, 0 to remove and 395 not upgraded.
Need to get 10.4 MB of archives.
After this operation, 48.7 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

```
y
```

```
Setting up texinfo (5.2.0.dfsg.1-2) ...
Setting up libwww-perl (6.05-2) ...
Setting up liblwp-protocol-https-perl (6.04-2ubuntu0.1) ...
Setting up libxml-parser-perl (2.41-1build3) ...
Setting up libxml-sax-expat-perl (0.40-2) ...
update-perl-sax-parsers: Registering Perl SAX parser XML::SAX::Expat
with priority 50...
update-perl-sax-parsers: Updating overall Perl SAX parser modules
info file...
Replacing config file /etc/perl/XML/SAX/ParserDetails.ini with new
version
```

3. **Create a directory to store the toolchain.**

```
sudo mkdir /opt/Espressif
```

4. **Make the current user as the owner.**

```
sudo chown $USER /opt/Espressif/
```

5. **Download the latest toolchain installation file.**

```
cd /opt/Espressif/
git clone -b esp108-1.21.0 git://github.com/jcmvbkbc/crosstool-NG.git
```

```
Cloning into 'crosstool-NG'...
remote: Counting objects: 28443, done.
remote: Total 28443 (delta 0), reused 0 (delta 0), pack-reused 28443
Receiving objects: 100% (28443/28443), 15.83 MiB | 71.00 KiB/s, done.
Resolving deltas: 100% (16473/16473), done.
Checking connectivity... done.
```

6. Install toolchain.

```
cd crosstool-NG/
./bootstrap && ./configure --prefix=`pwd` && make && make install
```

```
For auto-completion, do not forget to install 'ct-ng.comp' into
your bash completion directory (usually /etc/bash_completion.d)
username:/opt/Espressif/crosstool-NG$
```

```
./ct-ng xtensa-esp108-elf
```

```
************************************************************
WARNING! This sample may enable experimental features.
         Please be sure to review the configuration prior
         to building and using your toolchain!
Now, you have been warned!
************************************************************
Now configured for "xtensa-esp108-elf"
username:/opt/Espressif/crosstool-NG$
```

```
./ct-ng build
```

```
[INFO ]  Cleaning-up the toolchain's directory
[INFO ]    Stripping all toolchain executables
[INFO ]  Cleaning-up the toolchain's directory: done in 1.85s (at
24:28)
[INFO ]  Build completed at 20151210.175153
[INFO ]  (elapsed: 24:27.54)
[INFO ]  Finishing installation (may take a few seconds)...
```

Note :

It takes about dozens of minutes to build toolchain.

7. Set the PATH variable to point to the newly compiled toolchain.

```
export PATH=/opt/Espressif/crosstool-NG/builds/xtensa-esp108-elf/bin:
$PATH
```

⚠ Notice :

Make sure you set the correct path, or it will occur a compile error.

> Note :
>
> You need to do **Step 6** every time you open a new shell, or you can put it inside your **.bashrc** file.

END

2.1.3. Building Firmware

1. Create a directory to store a new project.

```
mkdir ~/Workspace
cd ~/Workspace
```

2. Download the latest ESP32 RTOS SDK.

```
git clone https://github.com/espressif/ESP32_RTOS_SDK.git
```

> Notes :
>
> 1. For the package components of the SDK, refer to "**1.3 Package Components**".
> 2. You can build customized project according the ESP32_RTOS_SDK/examples/project_template.

3. Create a directory to store the.bin files compiled.

```
mkdir -p ~/Workspace/ESP32_BIN
```

4. Set the PATHs variable to point to the SDK and BIN files.

```
export SDK_PATH=~/Workspace/ESP32_RTOS_SDK
export BIN_PATH=~/Workspace/ESP32_BIN
```

> ⚠ Warning :
>
> Make sure you set the correct paths, or it will occur a compile error.

5. Start to compile files.

```
cd ~/Workspace/project_template
make clean
make
```

> Note :
>
> You need to do Step 3 every time you open a new shell, or you can put it inside your **.bashrc** file.

END

Note :

If your project is successfully compiled, the **irom1.bin**, **irom0_flash.bin**, and **user.ota** files will be generated in **/Workspace/ESP32_BIN** directory. Current contents of **/Workspace/ESP32_BIN** directory is shown as Table2-1.

Table 2-1. ESP32_BIN Directory Files

File Name	Description
blank.bin	• Provided with the SDK. • Used to initialize the system parameters.
boot.bin	• Provided with the SDK. • Main program.
irom1.bin	• Generated from compiling. • Main program.
irom0_flash.bin	• Generated from compiling. • Main program.
User.ota	• Generated from compiling. • Main program. • Don't need to be uploaded, and will be updated by OTA.

2.1.4. Uploading Firmware

1. **Power on the board with 5V/1A power.**
2. **Connect the development board to a PC.**
3. **Toggle 8-digit Toggle Switch to the Download Boot as Table 1-2.**
4. **Install Minicom.**

```
sudo apt-get install minicom
```

5. **Run Minicom.**

```
sudo minicom -s
```

6. **Configure the items as Figure 2-2.**

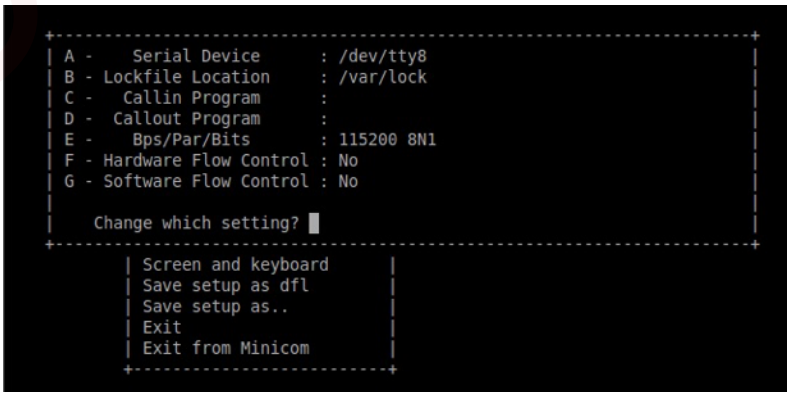

Figure 2-2. Configuration items of Minicom

The configuration items are shown in Table 2-2.

Table 2-2. Configuration Items of Minicom

Item	Value
Bps/Par/Bits	115200 8N1
Hardware Flow Control	No
Software Flow Control	No

7. Upload files.

```
>python esptool_ESP32.py -p dev/tty8 -b 115200 write_flash -ff 40m -
fm qio -fs 2m
0x0 ~/Workspace/ESP32_BIN/boot.bin
0x04000 ~/Workspace/ESP32_BIN/irom1.bin
0x40000 ~/Workspace/ESP32_BIN/bin/irom0_flash.bin
0xFC000 ~/Workspace/ESP32_BIN/blank.bin
0x1FC000 ~/Workspace/ESP32_BIN/esp_init_data_default.bin
```

For the configuration instructions, refer to **Table 2-3**. The system will show the information as below.

```
Connecting...
Erasing flash...
Wrote 3072 bytes at 0x00000000 in 0.3 seconds (73.8 kbit/s)...
Erasing flash...
Wrote 395264 bytes at 0x04000000 in 43.2 seconds (73.2 kbit/s)...
Erasing flash...
Wrote 1024 bytes at 0x40000000 in 0.1 seconds (74.5 kbit/s)...
Erasing flash...
Wrote 4096 bytes at 0xfc000000 in 0.4 seconds (73.5 kbit/s)...
Erasing flash...
Wrote 4096 bytes at 0x1fc00000 in 0.5 seconds (73.8 kbit/s)...
Leaving...
```

Table 2-3. Configuration Items of uploading files

Item	Value
COM Port	You can choose an available COM port for the uploading. We use dev/tty in this example.
Baud Rate	Set the baud rate of the uploading. The default value is 115200 bps.

Item	Value
SPI Speed	Set the SPI speed of the uploading. The default value is 40 MHz.
SPI Mode	Set the SPI speed of the uploading. The default value is QIO.
Flash Size	The size of your flash, We use 2Mbit flash in this example. The flash size of 1MB, 2MB, 4MB, 8MB, 16MB and 32MB are supported.
Start Address	Set the start addresses of the bin files. The start address of blank.bin and esp_init_data_default.bin may be different when you use a different flash size. For the start Address of different size of flash, refer to Table 2-4.

Table 2-4. Configuration Items of uploading files

File Name	Start Address		Notes
	Flash Size: 1 MB	Flash Size: 2 MB	
boot.bin	0x00000		-
irom1.bin	0x04000		-
irom0_flash.bin	0x40000		-
user.ota①	Doesn't need to be uploaded.		-
blank.bin	0xFE000	0x1FE000	As shown in the figure above, the file will start from the second section(4K) from bottom of the flash.②
esp_init_data_default.bin③	0xFC000	0x1FC000	As shown in the figure above, the file will start from the fourth section(4K) from bottom of the flash.②

Notes:

1. **User.ota** file doesn't need to be uploaded, and will be used in OTA upgrading.
2. The flash size of 1MB, 2MB, 4MB, 8MB, 16MB and 32MB are supported.
3. **esp_init_data_default.bin** isn't provided in the version 1.0.0.

END

Result

The flash layout is shown as Figure 2-3.

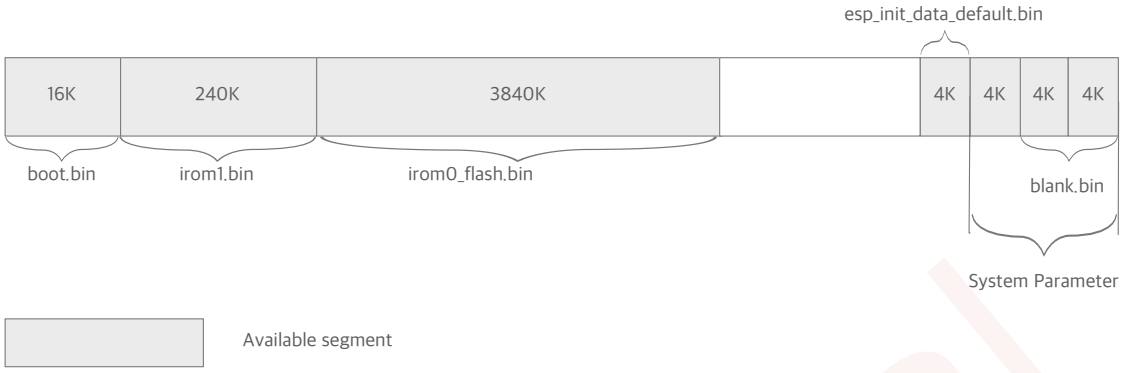

Figure 2-3. Flash Layout

For the flash usage, refer to Table 2-5.

Table 2-5. Flash Usage

Segment Name	Start Address	Length
dport0_0_seg	org = 0x3FF00000	len = 0x10
dram0_0_seg	org = 0x3FFD8000	len = 0x24000
iram1_0_seg	org = 0x40040000	len = 0x20000
irom0_0_seg	org = 0x40080010	len = 0x37FFF0
irom0_1_seg	org = 0x3FE04010	len = 0x3BFF0

2.1.5. Running Application

1. **Power off and switch the hardware to the running mode.**
2. **Power on the hardware and run the application.**

END

2.2. SSC Command Reference

Here lists some common Wi-Fi commands for you to test the board.

2.2.1. op

Description

op commands are used to set and query the Wi-Fi mode of the system.

Example

```
op -Q
```

```
op -S -o wmode
```

Parameter

Table 2-6. op Command Parameter

Parameter	Description
-Q	Query Wi-Fi mode.
-S	Set Wi-Fi mode.
wmode	There are 3 Wi-Fi modes: • mode = 1: STA mode • mode = 2: AP mode • mode = 3: STA+AP mode

2.2.2. sta

Description

sta commands are used to scan the STA network interface, connect or disconnect AP, and query the connecting status of STA network interface.

Example

```
sta -S [-s ssid] [-b bssid] [-n channel] [-h]
sta -Q
sta -C [-s ssid] [-p password]
sta -D
```

Parameter

Table 2-7. sta Command Parameter

Parameter	Description
-S scan	Scan Access Points.
-s ssid	Scan or connect Access Points with the ssid.
-b bssid	Scan the Access Points with the bssid.
-n channel	Scan the channel.
-h	Show scan results with hidden ssid Access Points.
-Q	Show STA connect stutus.
-D	Disconnected with current Access Points.

2.2.3. ap

Description

ap commands are used to set the parameter of AP network interface.

Example

```
ap -S [-s ssid] [-p password] [-t encrypt] [-n channel] [-h] [-m max_sta]
ap -Q
ap -L
```

Parameter

Table 2-8. ap Command Parameter

Parameter	Description
-S	Set AP mode.
-s ssid	Set AP ssid.
-p password	Set AP password.
-t encrypt	Set AP encrypt mode.
-h	Hide ssid.
-m max_sta	Set AP max connections.
-Q	Show AP parameters.
-L	Show MAC Address and IP Address of the connected station.

2.2.4. mac

Description

mac commands are used to query the MAC address of the network interface.

Example

```
mac -Q [-o mode]
```

Parameter

Table 2-9. mac Command Parameter

Parameter	Description
-Q	Show MAC address.
-o mode	• mode = 1: MAC address in STA mode. • mode = 2: MAC address in AP mode.

2.2.5. dhcp

Description

dhcp commands are used to enable or disable dhcp server/client.

Example

```
dchp -S [-o mode]
dhcp -E [-o mode]
dhcp -Q [-o mode]
```

Parameter

Table 2-10. dhcp Command Parameter

Parameter	Description
-S	Start DHCP (Client/Server).
-E	End DHCP (Client/Server).
-Q	show DHCP status.
-o mode	• mode = 1 : DHCP client of STA interface. • mode = 2 : DHCP server of AP interface. • mode = 3 : both.

2.2.6. ip

Description

ip command are used to set and query the IP address of the network interface.

Example

```
ip -Q [-o mode]
ip -S [-i ip] [-o mode] [-m mask] [-g gateway]
```

Parameter

Table 2-11. ip Command Parameter

Parameter	Description
-Q	Show IP address.
-o mode	• mode = 1 : IP address of interface STA. • mode = 2 : IP address of interface AP. • mode = 3 : both
-S	Set IP address.
-i ip	IP address.
-m mask	Subnet address mask.
-g gateway	Default gateway.

2.2.7. reboot

Description

reboot command is used to reboot the board.

Example

```
reboot
```

2.2.8. ram

ram command is used to query the size of the remaining heap in the system.

Example

```
ram
```

Espressif System

IOT Team

http://bbs.espressif.com