ESP32-WROVER
User Guide

Version 1.0

About This Guide

This document is intended to help users set up the basic software development environment for developing applications using hardware based on the ESP32-WROVER. Through a simple example, this document illustrates how to use ESP-IDF (Espressif IoT Development Framework), including the menu based configuration wizard, compiling the ESP-IDF and firmware download to the ESP32 module.

The document is structured as follows.

Release Notes

Date	Version	Release notes
2017.05	V1.0	First release.

Table of Contents

1. Introduction 1
    1.1. ESP32 1
    1.2. ESP-IDF 1
    1.3. Preparing the Hardware 1
2. Getting Started 2
    2.1. Standard Setup of Toolchain for Linux 2
        2.1.1. Install Prerequisites 2
        2.1.2. Toolchain Setup 2
    2.2. Get ESP-IDF 3
    2.3. Set up Path to ESP-IDF 3
3. Start a Project 4
4. Configuration 5
5. Build and Flash 6
    5.1. Build and Flash 6
    5.2. Monitor 7
6. SSC Command Reference 8
    6.1. op 8
    6.2. sta 8
    6.3. ap 9
    6.4. mac 9
    6.5. dhcp 10
    6.6. ip 10
    6.7. reboot 11
    6.8. ram 11

1. Introduction

1.1. ESP32

ESP32 integrates Wi-Fi (2.4 GHz band) and Bluetooth 4.2 solutions on a single chip, along with dual high performance cores and many other versatile peripherals. Powered by 40 nm technology, ESP32 provides a robust, highly integrated platform to meet the continuous demands for efficient power usage, compact design, security, high performance, and reliability.

Espressif provides the basic hardware and software resources that empowers application developers to build their ideas around the ESP32 series hardware. The software development framework provided by Espressif is intended for rapidly developing Internet-of-Things (IoT) applications, with Wi-Fi, Bluetooth, flexible power management and other advanced system features.

The RF frequency range is 2.412-2.462 GHz(Wi-Fi)/2.402-2.480GHz (Bluetooth/BLE). The maximum RF transmit power is 23 dBm.

The manufacturer is Espressif Systems (Shanghai) Pte., Ltd, which is shown at the bottom of the product.

1.2. ESP-IDF

The Espressif IoT Development Framework (ESP-IDF for short) is a framework for developing applications based on the Espressif ESP32. Users can develop applications in Windows/Linux/MacOS based on ESP-IDF. It is recommended to use Linux distribution. ***Lubuntu*** 16.04 has been used as an example in this document for illustration purposes.

1.3. Preparing the Hardware

- 1 x ESP32-WROVER
- 1 x USB-to-TTL serial cable or a Micro-USB cable.

2. Getting Started

2.1. Standard Setup of Toolchain for Linux

Depending on your experience and preferences, you may follow standard installation process or customize your environment. Instructions immediately below are for standard installation. To set up the system your own way go to section ***Customized Setup of Toolchain***.

2.1.1. Install Prerequisites

To compile with ESP-IDF you need to get the following packages:

- CentOS 7:

```
sudo yum install git wget make ncurses-devel flex bison gperf python pyserial
```

- Ubuntu and Debian:

```
sudo apt-get install git wget make libncurses-dev flex bison gperf python python-serial
```

- Arch:

```
sudo pacman -S --needed gcc git make ncurses flex bison gperf python2-pyserial
```

2.1.2. Toolchain Setup

ESP32 toolchain for Linux is available for download from Espressif website:

- for 64-bit Linux:

https://dl.espressif.com/dl/xtensa-esp32-elf-linux64-1.22.0-61-gab8375a-5.2.0.tar.gz

- for 32-bit Linux:

https://dl.espressif.com/dl/xtensa-esp32-elf-linux32-1.22.0-61-gab8375a-5.2.0.tar.gz

Download this file, then extract it in ***~/esp*** directory

```
mkdir -p ~/esp

cd ~/esp

tar -xzf ~/Downloads/xtensa-esp32-elf-linux64-1.22.0-61-gab8375a-5.2.0.tar.gz
```

The toolchain will be extracted into ***~/esp/xtensa-esp32-elf/*** directory.

To use it, you will need to update your `***PATH***` environment variable in ***~/.bash_profile*** file. To make ***xtensa-esp32-elf*** available for all terminal sessions, add the following line to your ***~/.bash_profile*** file:

```
export PATH=$PATH:$HOME/esp/xtensa-esp32-elf/bin
```

Alternatively, you may create an alias for the above command. This way you can get the toolchain only when you need it. To do this, add different line to your ***~/.bash_profile*** file:

```
alias get_esp32="export PATH=$PATH:$HOME/esp/xtensa-esp32-elf/bin"
```

Then when you need the toolchain you can type ***get_esp32*** on the command line and the toolchain will be added to your ***PATH***.

2.2. Get ESP-IDF

Once you have the toolchain (that contains programs to compile and build the application) installed, you also need ESP32 specific API / libraries. They are provided by Espressif in ESP-IDF repository. To get it, open terminal, navigate to the directory you want to put ESP-IDF, and clone it using git clone command:

```
cd ~/esp

git clone --recursive https://github.com/espressif/esp-idf.git
```

ESP-IDF will be downloaded into ***~/esp/esp-idf***.

> 📖 ***Note:***
> *Do not miss the* `--recursive` *option. If you have already cloned ESP-IDF without this option, run another command to get all the submodules:*
>
> ```
> cd ~/esp/esp-idf
> git submodule update --init
> ```

2.3. Set up Path to ESP-IDF

The toolchain programs access ESP-IDF using ***IDF_PATH*** environment variable. This variable should be set up on your PC, otherwise projects will not build. Setting may be done manually, each time PC is restarted. Another option is to set up it permanently by defining ***IDF_PATH*** in user profile.

Set up ***IDF_PATH*** by adding the following line to ***~/.bash file***:

```
export IDF_PATH=~/esp/esp-idf
```

Log off and log in back to make this change effective.

If you do not like to have ***IDF_PATH*** set up permanently, you should enter it manually in terminal window on each restart or logout.

Run the following command to check if ***IDF_PATH*** is set:

```
printenv IDF_PATH
```

The path previously entered in ***~/.bash*** file (or set manually) should be printed out.

3. Start a Project

Now you are ready to prepare your application for ESP32. To start off quickly, we will use ***get-started/hello_world*** project from ***examples*** directory in IDF.

Copy ***get-started/hello_world*** to ***~/esp*** directory:

```
cd ~/esp

cp -r $IDF_PATH/examples/get-started/hello_world .
```

You can also find a range of example projects under the ***examples*** directory in ESP-IDF. These example project directories can be copied in the same way as presented above, to begin your own projects.

> 📖 ***Note:***
>
> *The ESP-IDF build system does not support spaces in paths to ESP-IDF or to projects.*

4. Configuration

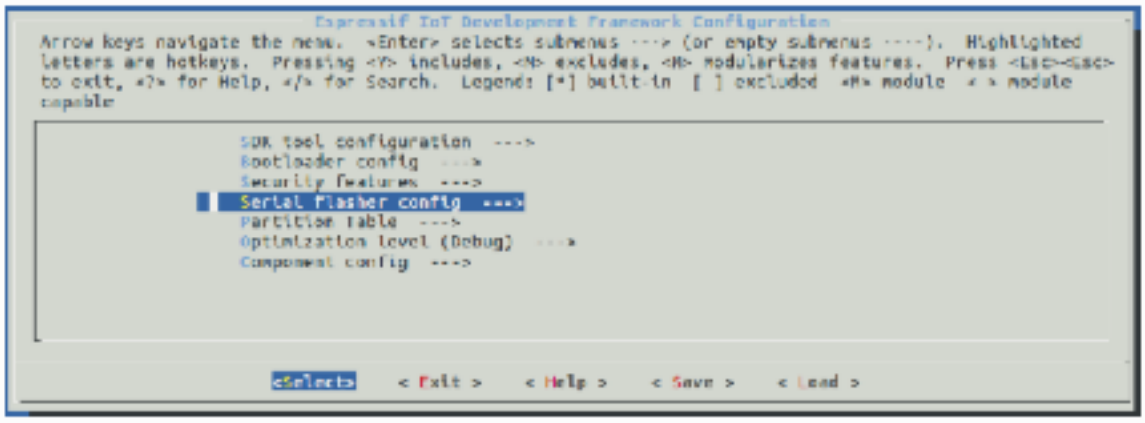
You are almost there. To be able to proceed further, connect ESP32-WROVER to PC, check under what serial port the board is visible and verify if serial communication works.

Being in terminal window, go to directory of ***hello_world*** application by typing `cd ~/esp/hello_world`. Then start project configuration utility ***menuconfig***:

```
cd ~/esp/hello_world

make menuconfig
```

If previous steps have been done correctly, the following menu will be displayed:



Project configuration - Home window

In the menu, navigate to ***Serial flasher config*** > ***Default serial port*** to configure the serial port, where project will be loaded to. Confirm selection by pressing enter, save configuration by selecting **< *Save* >** and then exit application by selecting **< *Exit* >**.

Here are couple of tips on navigation and use of ***menuconfig***:

- Use up & down arrow keys to navigate the menu.
- Use Enter key to go into a submenu, Escape key to go out or to exit.
- Type ***?*** to see a help screen. Enter key exits the help screen.
- Use Space key, or ***Y*** and ***N*** keys to enable (Yes) and disable (No) configuration items with checkboxes "***[*]***".
- Pressing ***?*** while highlighting a configuration item displays help about that item.
- Type **/** to search the configuration items.

5. Build and Flash

5.1. Build and Flash

Now you can build and flash the application. Run:

```
make flash
```

This will compile the application and all the ESP-IDF components, generate bootloader, partition table, and application binaries, and flash these binaries to your ESP32 board.

```
esptool.py v2.0-beta2

Flashing binaries to serial port /dev/ttyUSB0 (app at offset 0x10000)...

esptool.py v2.0-beta2

Connecting........___

Uploading stub...

Running stub...

Stub running...

Changing baud rate to 921600

Changed.

Attaching SPI flash...

Configuring flash size...

Auto-detected Flash size: 4MB

Flash params set to 0x0220

Compressed 11616 bytes to 6695...

Wrote 11616 bytes (6695 compressed) at 0x00001000 in 0.1 seconds (effective 920.5 kbit/s)...

Hash of data verified.

Compressed 408096 bytes to 171625...

Wrote 408096 bytes (171625 compressed) at 0x00010000 in 3.9 seconds (effective 847.3 kbit/
s)...

Hash of data verified.

Compressed 3072 bytes to 82...

Wrote 3072 bytes (82 compressed) at 0x00008000 in 0.0 seconds (effective 8297.4 kbit/s)...

Hash of data verified.


Leaving...

Hard resetting...
```

If there are no issues, at the end of build process, you should see messages describing progress of loading process. Finally, the end module will be reset and “hello_world” application will start.

5.2. Monitor

To see if “hello_world” application is indeed running, type `make monitor`. This command is launching IDF Monitor application:

```
$ make monitor

MONITOR

--- idf_monitor on /dev/ttyUSB0 115200 ---

--- Quit: Ctrl+] | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---

ets Jun  8 2016 00:22:57


rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)

ets Jun  8 2016 00:22:57

...
```

Several lines below, after start up and diagnostic log, you should see “Hello world!” printed out by the application.

```
...

Hello world!

Restarting in 10 seconds...

I (211) cpu_start: Starting scheduler on APP CPU.

Restarting in 9 seconds...

Restarting in 8 seconds...

Restarting in 7 seconds...
```

To exit monitor use shortcut ***Ctrl+]***. To execute `make flash` and `make monitor` in one shoot type `make flash monitor`.

6. SSC Command Reference

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

6.1. 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 6-1. 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

6.2. 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 6-2. sta Command Parameter

Parameter	Description
-S scan	Scan Access Points.

Parameter	Description
-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.

6.3. 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 6-3. 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.

6.4. mac

Description

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

Example

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

Parameter

Table 6-4. 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.

6.5. 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 6-5. 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.

6.6. 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 6-6. 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.

6.7. reboot

Description

reboot command is used to reboot the board.

Example

```
reboot
```

6.8. ram

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

Example

```
ram
```

FCC Warning:

Any changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operaion.

This equipment complies with FCC radiation exposure limits set forth for an uncontrolled environment. This equipment should be installed and operated with minimum distance 20cm between the radiator & your body.

This transmitter must not be co-located or operating in conjunction with any other antenna or transmitter.

FCC Label Instructions:

If using a permanently affixed label, the modular transmitter must be labeled with its own FCC identification number, and, if the FCC identification number is not visible when the module is installed inside another device, then the outside of the device into which the module is installed must also display a label referring to the enclosed module. This exterior label can use wording such as the following:

"Contains Transmitter Module FCC ID: 2AC7Z-ESP32WROVER, or

"Contains FCC ID: 2AC7Z-ESP32WROVER

Any similar wording that expresses the same meaning may be used. The Grantee may either provide such a label, an example of which must be included in the application for equipment authorization, or, must provide adequate instructions along with the module which explain this requirement.

Espressif IOT Team

www.espressif.com