ESP32-WROOM-DA

User Manual

Stand-Alone Module with Dual Antennas
Containing Ultra-Low-Power SoC with DualCore CPU
Supporting 2.4 GHz Wi-Fi, Bluetooth®, and Bluetooth LE

ESP32-WROOM-DA

Pre-release v0.5
Espressif Systems
Copyright © 2021

About This Document

This user manual shows how to get started with the ESP32-WROOM-DA module.

Document Updates

Please always refer to the latest version on https://www.espressif.com/en/support/download/documents.

Revision History

For revision history of this document, please refer to the last page.

Documentation Change Notification

Espressif provides email notifications to keep you updated on changes to technical documentation. Please subscribe at www.espressif.com/en/subscribe.

Certification

Download certificates for Espressif products from www.espressif.com/en/certificates.

Contents

1 Overview

1.1 Module Overview

ESP32-WROOM-DA is a powerful Wi-Fi + Bluetooth + Bluetooth LE MCU module, with two complementary PCB antennas in different directions. This module is embedded with ESP32-D0WD-V3 with a rich set of integrated components including SPI flash and 40 MHz crystal oscillator. With two unique antennas design on one single module, ESP32-WROOM-DA can be used to develop IoT applications that need stable connectivity over a broad spectrum, or to deploy Wi-Fi in challenging and hazardous environments, or to overcome communication problems in Wi-Fi-dead spots. This module is an ideal choice for indoor and outdoor devices for smart home, industrial control, consumer electronics, etc.

Table 1: ESP32-WROOM-DA Specifications

Categories	Items	Specifications
Wi-Fi	Protocols	802.11 b/g/n (802.11n up to 150 Mbps)
		A-MPDU and A-MSDU aggregation and 0.4 μs guard interval support
	Frequency range	2412 ~ 2484 MHz
Bluetooth®	Protocols	Protocols v4.2 BR/EDR and Bluetooth® LE specifications
	Radio	Class-1, class-2 and class-3 transmitter
		AFH
	Audio	CVSD and SBC
Hardware	Module interfaces	SD card, UART, SPI, SDIO, I2C, LED PWM, Motor PWM, I2S, IR, pulse counter, GPIO, capacitive touch sensor, ADC, DAC, TWAI® (compatible with ISO 11898-1, i.e. CAN Specification 2.0)
	Integrated crystal	40 MHz crystal
	Integrated SPI flash	8 MB
	Operating voltage/Power supply	3.0 V ~ 3.6 V
	Operating current	Average: 80 mA
	Minimum current delivered by power supply	500 mA
	Recommended operating temperature range	–40 ~ 85 °C
	Moisture sensitivity level (MSL)	Level 3

1.2 Pin Description

The pin diagram below shows the approximate location of pins and the two antennas on the module.

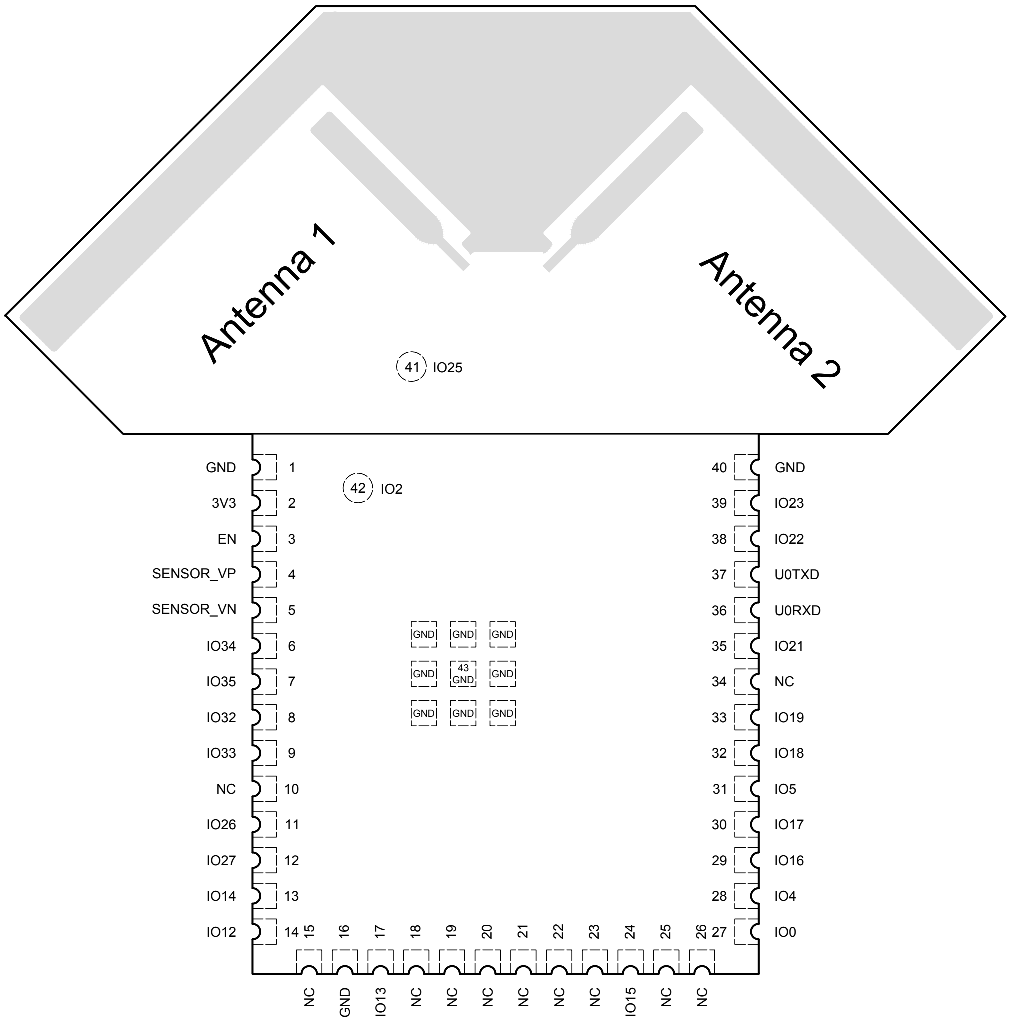


Figure 1: Pin Layout (Top View)

The module has 41 pins and two test points. See pin definitions in Table 2.

Table 2: Pin Definitions

Name	No.	Type	Function[2]
GND	1	P	Ground
3V3	2	P	Power supply
EN	3	I	High: On; enables the chip Low: Off; the chip powers off Note: Do not leave the pin floating.
SENSOR_VP	4	I	GPIO36, ADC1_CH0, RTC_GPIO0
SENSOR_VN	5	I	GPIO39, ADC1_CH3, RTC_GPIO3
IO34	6	I	GPIO34, ADC1_CH6, RTC_GPIO4
IO35	7	I	GPIO35, ADC1_CH7, RTC_GPIO5

Cont'd on next page

Table 2 – cont'd from previous page

Name	No.	Type	Function[2]
IO32	8	I/O	GPIO32, XTAL_32K_P (32.768 kHz crystal oscillator input), ADC1_CH4, TOUCH9, RTC_GPIO9
IO33	9	I/O	GPIO33, XTAL_32K_N (32.768 kHz crystal oscillator output), ADC1_CH5, TOUCH8, RTC_GPIO8
NC	10	—	—
IO26	11	I/O	GPIO26, DAC_2, ADC2_CH9, RTC_GPIO7, EMAC_RXD1
IO27	12	I/O	GPIO27, ADC2_CH7, TOUCH7, RTC_GPIO17, EMAC_RX_DV
IO14	13	I/O	GPIO14, ADC2_CH6, TOUCH6, RTC_GPIO16, MTMS, HSPICLK, HS2_CLK, SD_CLK, EMAC_TXD2
IO12	14	I/O	GPIO12, ADC2_CH5, TOUCH5, RTC_GPIO15, MTDI, HSPIQ, HS2_DATA2, SD_DATA2, EMAC_TXD3
NC	15	—	—
GND	16	P	Ground
IO13	17	I/O	GPIO13, ADC2_CH4, TOUCH4, RTC_GPIO14, MTCK, HSPID, HS2_DATA3, SD_DATA3, EMAC_RX_ER
NC	18	—	—
NC	19	—	—
NC	20	—	—
NC	21	—	—
NC	22	—	—
NC	23	—	—
IO15	24	I/O	GPIO15, ADC2_CH3, TOUCH3, MTDO, HSPICS0, RTC_GPIO13, HS2_CMD, SD_CMD, EMAC_RXD3
NC	25	—	—
NC	26	—	—
IO0	27	I/O	GPIO0, ADC2_CH1, TOUCH1, RTC_GPIO11, CLK_OUT1, EMAC_TX_CLK
IO4	28	I/O	GPIO4, ADC2_CH0, TOUCH0, RTC_GPIO10, HSPIHD, HS2_DATA1, SD_DATA1, EMAC_TX_ER
IO16	29	I/O	GPIO16, HS1_DATA4, U2RXD, EMAC_CLK_OUT
IO17	30	I/O	GPIO17, HS1_DATA5, U2TXD, EMAC_CLK_OUT_180
IO5	31	I/O	GPIO5, VSPICS0, HS1_DATA6, EMAC_RX_CLK
IO18	32	I/O	GPIO18, VSPICLK, HS1_DATA7
IO19	33	I/O	GPIO19, VSPIQ, U0CTS, EMAC_TXD0
NC	34	—	—
IO21	35	I/O	GPIO21, VSPIHD, EMAC_TX_EN
U0RXD	36	I/O	GPIO3, U0RXD, CLK_OUT2
U0TXD	37	I/O	GPIO1, U0TXD, CLK_OUT3, EMAC_RXD2
IO22	38	I/O	GPIO22, VSPIWP, U0RTS, EMAC_TXD1
IO23	39	I/O	GPIO23, VSPID, HS1_STROBE
GND	40	P	Ground
NC [3]	41	—	—

Cont'd on next page

Table 2 – cont'd from previous page

Name	No.	Type	Function[2]
NC [3]	42	—	—
GND	43	P	Ground

[1] For peripheral pin configurations, please refer to *ESP32 Series Datasheet*.

[2] GPIO2 and GPIO25 on the ESP32-D0WD-V3 chip are designed as test points to control RF Switch. The two pins are not led out to the module. To select the working antenna, (Antenna 1 or Antenna 2), configure GPIO2 and GPIO25 as follows:

Table 3: Select Working Antenna

Working Antenna	GPIO2	GPIO25
Antenna 1	High	Low
Antenna 2 (by default)	Low	High

2 Get Started on ESP32-WROOM-DA

2.1 What You Need

To develop applications for module you need:

- 1 x ESP32-WROOM-DA module
- 1 x Espressif RF testing board
- 1 x USB-to-Serial board
- 1 x Micro-USB cable
- 1 x PC running Linux

In this user guide, we take Linux operating system as an example. For more information about the configuration on Windows and macOS, please refer to *ESP-IDF Programming Guide*.

2.2 Hardware Connection

1. Solder the ESP32-WROOM-DA module to the RF testing board as shown in Figure 2.

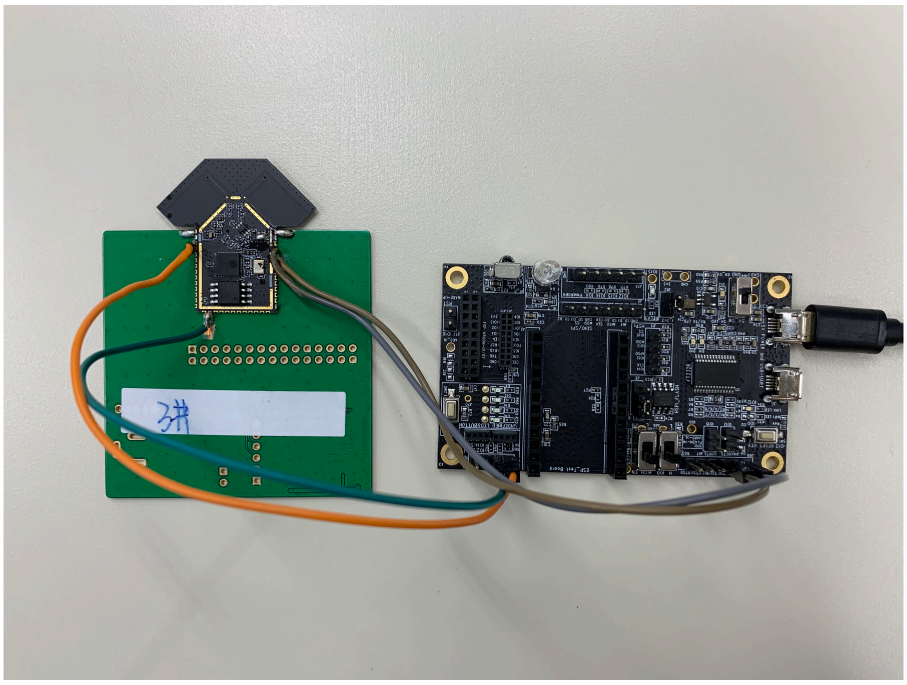


Figure 2: Hardware Connection

2. Connect the RF testing board to the USB-to-Serial board via TXD, RXD, and GND.
3. Connect the USB-to-Serial board to the PC.
4. Connect the RF testing board to the PC or a power adapter to enable 5 V power supply, via the Micro-USB cable.
5. During download, connect IO0 to GND via a jumper. Then, turn "ON" the testing board.
6. Download firmware into flash. For details, see the sections below.

7. After download, remove the jumper on IO0 and GND.
8. Power up the RF testing board again. ESP32-WROOM-DA will switch to working mode. The chip will read programs from flash upon initialization.

Note:
IO0 is internally logic high. If IO0 is set to pull-up, the Boot mode is selected. If this pin is pull-down or left floating, the Download mode is selected. For more information on ESP32-WROOM-DA, please refer to ESP32-WROOM-DA Datasheet.

2.3 Set up Development Environment

The Espressif IoT Development Framework (ESP-IDF for short) is a framework for developing applications based on the Espressif ESP32. Users can develop applications with ESP32 in Windows/Linux/macOS based on ESP-IDF. Here we take Linux operating system as an example.

2.3.1 Install Prerequisites

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

- CentOS 7:

```
sudo yum install git wget flex bison gperf python cmake ninja-build ccache dfu-util
```

- Ubuntu and Debian (one command breaks into two lines):

```
sudo apt-get install git wget flex bison gperf python python-pip python-setuptools cmake
ninja-build ccache libffi-dev libssl-dev dfu-util
```

- Arch:

```
sudo pacman -S --needed gcc git make flex bison gperf python-pip cmake ninja ccache dfu-util
```

Note:
- This guide uses the directory ~/esp on Linux as an installation folder for ESP-IDF.
- Keep in mind that ESP-IDF does not support spaces in paths.

2.3.2 Get ESP-IDF

To build applications for ESP32-WROOM-DA module, you need the software libraries provided by Espressif in ESP-IDF repository.

To get ESP-IDF, create an installation directory (~/esp) to download ESP-IDF to and clone the repository with 'git clone':

```
mkdir -p ~/esp
cd ~/esp
git clone --recursive https://github.com/espressif/esp-idf.git
```

ESP-IDF will be downloaded into ~/esp/esp-idf. Consult ESP-IDF Versions for information about which ESP-IDF version to use in a given situation.

2.3.3 Set up Tools

Aside from the ESP-IDF, you also need to install the tools used by ESP-IDF, such as the compiler, debugger, Python packages, etc. ESP-IDF provides a script named 'install.sh' to help set up the tools in one go.

```
cd ~/esp/esp-idf
./install.sh
```

2.3.4 Set up Environment Variables

The installed tools are not yet added to the PATH environment variable. To make the tools usable from the command line, some environment variables must be set. ESP-IDF provides another script 'export.sh' which does that. In the terminal where you are going to use ESP-IDF, run:

```
. $HOME/esp/esp-idf/export.sh
```

Now everything is ready, you can build your first project on ESP32-WROOM-DA module.

2.4 Create Your First Project

2.4.1 Start a Project

Now you are ready to prepare your application for ESP32-WROOM-DA module. You can start with get-started/hello_world project from examples directory in ESP-IDF.

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

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

There is a range of example projects in the examples directory in ESP-IDF. You can copy any project in the same way as presented above and run it. It is also possible to build examples in-place, without copying them first.

2.4.2 Connect Your Device

Now connect your ESP32-WROOM-DA module to the computer and check under what serial port the module is visible. Serial ports in Linux start with '/dev/tty' in their names. Run the command below two times, first with the board unplugged, then with plugged in. The port which appears the second time is the one you need:

```
ls /dev/tty*
```

Note:
Keep the port name handy as you will need it in the next steps.

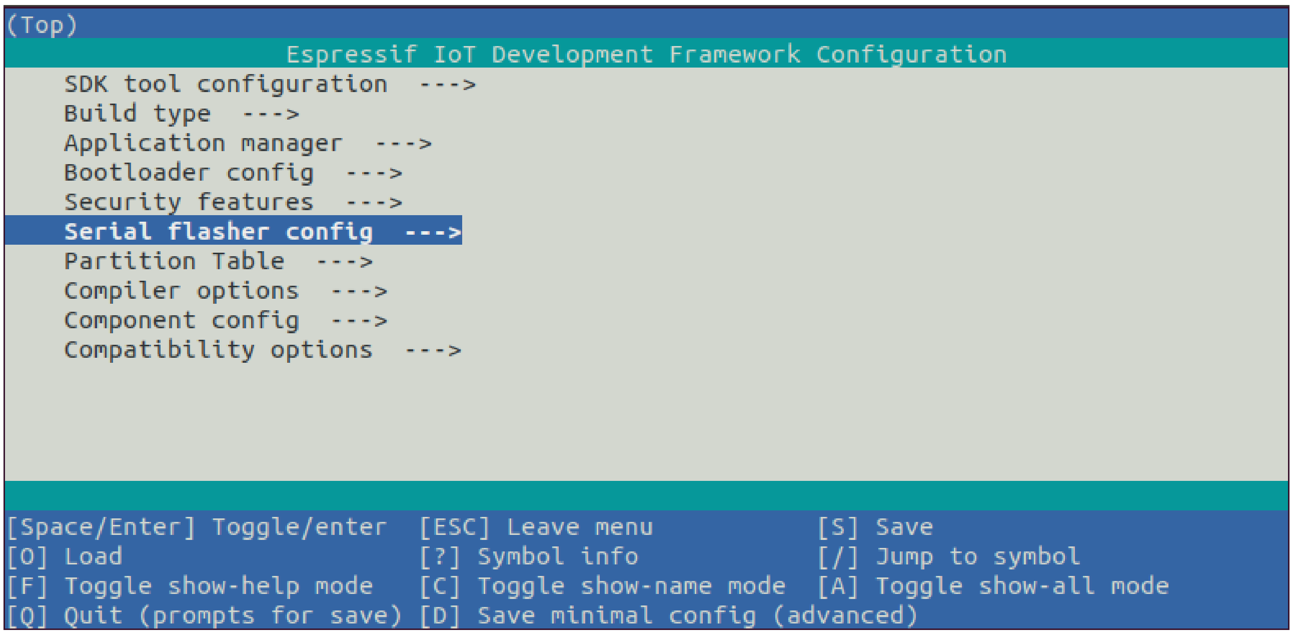
2.4.3 Configure

Navigate to your 'hello_world' directory from Step 2.4.1. Start a Project, set ESP32 chip as the target and run the project configuration utility 'menuconfig'.

```
cd ~/esp/hello_world
idf.py set-target esp32
idf.py menuconfig
```

Setting the target with 'idf.py set-target esp32' should be done once, after opening a new project. If the project contains some existing builds and configuration, they will be cleared and initialized. The target may be saved in environment variable to skip this step at all. See Selecting the Target for additional information.

If the previous steps have been done correctly, the following menu appears:



```
(Top)
                  Espressif IoT Development Framework Configuration
    SDK tool configuration  --->
    Build type  --->
    Application manager  --->
    Bootloader config  --->
    Security features  --->
    Serial flasher config  --->
    Partition Table  --->
    Compiler options  --->
    Component config  --->
    Compatibility options  --->

[Space/Enter] Toggle/enter  [ESC] Leave menu           [S] Save
[O] Load                    [?] Symbol info            [/] Jump to symbol
[F] Toggle show-help mode   [C] Toggle show-name mode  [A] Toggle show-all mode
[Q] Quit (prompts for save) [D] Save minimal config (advanced)
```

Figure 3: Project Configuration - Home Window

The colors of the menu could be different in your terminal. You can change the appearance with the option '--style'. Please run 'idf.py menuconfig --help' for further information.

2.4.4 Build the Project

Build the project by running:

```
idf.py build
```

This command will compile the application and all ESP-IDF components, then it will generate the bootloader, partition table, and application binaries.

```
$ idf.py build
Running cmake in directory /path/to/hello_world/build
Executing "cmake -G Ninja --warn-uninitialized /path/to/hello_world"...
Warn about uninitialized values.
-- Found Git: /usr/bin/git (found version "2.17.0")
-- Building empty aws_iot component due to configuration
-- Component names: ...
-- Component paths: ...

... (more lines of build system output)

[527/527] Generating hello-world.bin
esptool.py v2.3.1

Project build complete. To flash, run this command:
```

```
../../../components/esptool_py/esptool/esptool.py -p (PORT) -b 921600 write_flash --flash_mode dio
--flash_size detect --flash_freq 40m 0x10000 build/hello-world.bin  build 0x1000
build/bootloader/bootloader.bin 0x8000 build/partition_table/partition-table.bin
or run 'idf.py -p PORT flash'
```

If there are no errors, the build will finish by generating the firmware binary .bin file.

2.4.5 Flash onto the Device

Flash the binaries that you just built onto your ESP32-WROOM-DA module by running:

```
idf.py -p PORT [-b BAUD] flash
```

Replace PORT with your module's serial port name from Step: Connect Your Device.

You can also change the flasher baud rate by replacing BAUD with the baud rate you need. The default baud rate is 460800.

For more information on idf.py arguments, see idf.py. If everything goes well, the "hello_world" application starts running after you remove the jumper on IO0 and GND, and re-power up the testing board.

Note:
The option 'flash' automatically builds and flashes the project, so running 'idf.py build' is not necessary.

```
Running esptool.py in directory [...]/esp/hello_world
Executing "python [...]/esp-idf/components/esptool_py/esptool/esptool.py -b 460800 write_flash
@flash_project_args"...
esptool.py -b 460800 write_flash --flash_mode dio --flash_size detect --flash_freq 40m 0x1000
bootloader/bootloader.bin 0x8000 partition_table/partition-table.bin 0x10000 hello-world.bin
esptool.py v2.3.1
Connecting....
Detecting chip type... ESP32
Chip is ESP32
Features: WiFi, BT, Dual Core
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 460800
Changed.
Configuring flash size...
Auto-detected Flash size: 8MB
Flash params set to 0x0220
Compressed 22992 bytes to 13019...
Wrote 22992 bytes (13019 compressed) at 0x00001000 in 0.3 seconds (effective 558.9 kbit/s)...
Hash of data verified.
Compressed 3072 bytes to 82...
Wrote 3072 bytes (82 compressed) at 0x00008000 in 0.0 seconds (effective 5789.3 kbit/s)...
Hash of data verified.
Compressed 136672 bytes to 67544...
Wrote 136672 bytes (67544 compressed) at 0x00010000 in 1.9 seconds (effective 567.5 kbit/s)...
Hash of data verified.
```

```
Leaving...
Hard resetting via RTS pin...
```

2.4.6 Monitor

To check if "hello_world" is indeed running, type 'idf.py -p PORT monitor' Do not forget to replace PORT with your serial port name).

This command launches the IDF Monitor application:

```
$ idf.py -p /dev/ttyUSB0 monitor
Running idf_monitor in directory [...]/esp/hello_world/build
Executing "python [...]/esp-idf/tools/idf_monitor.py -b 115200

[...]/esp/hello_world/build/hello-world.elf"...
--- 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
...
```

After startup and diagnostic logs scroll up, you should see "Hello world!" printed out by the application.

```
...
Hello world!
Restarting in 10 seconds...
This is esp32 chip with 2 CPU cores, WiFi/BT/BLE, silicon revision 3, 8MB flash
Restarting in 9 seconds...
Restarting in 8 seconds...
Restarting in 7 seconds...
```

To exit IDF monitor use the shortcut Ctrl+].

That's all what you need to get started with ESP32-WROOM-DA module! Now you are ready to try some other examples in ESP-IDF, or go right to developing your own applications.

3 U.S. FCC Statement

FCC ID: 2AC7Z-ESPWROOMDA

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions:

- This device may not cause harmful interference.
- This device must accept any interference received, including interference that may cause undesired operation.

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part15 of the FCC Rules.

These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

Caution:

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

This equipment complies with FCC RF radiation exposure limits set forth for an uncontrolled environment. This device and its antenna must not be co-located or operating in conjunction with any other antenna or transmitter. The antennas used for this transmitter must be installed to provide a separation distance of at least 20 cm from all persons and must not be co-located or operating in conjunction with any other antenna or transmitter.

For European markets the relevant directives are mentioned below:

It is the responsibility of the manufacturer of the final product to verify whether or not further standards, recommendations or directives are in force outside these areas. Please note that SAR requirements are specific only for portable devices and not for mobile devices as defined below:

- Portable device: A portable device is defined as a transmitting device designed to be used so that the radiating structure(s) of the device is/are within 20 centimeters of the body of the user.
- Mobile device: A mobile device is defined as a transmitting device designed to be used in other than fixed locations and to generally be used in such a way that a separation distance of at least 20 centimeters is normally maintained between the transmitter's radiating structure(s) and the body of the user or nearby

persons. In this context, the term ”fixed location” means that the device is physically secured at one location and is not able to be easily moved to another location.

OEM Integration Instructions

This device is intended only for OEM integrators under the following conditions The module can be used to installation in another host. The antenna must be installed such that 20 cm is maintained between the antenna and users, and the transmitter module may not be co-located with any other transmit or antenna. The module shall be only used with the integral antenna(s) that has been originally tested and certified with this module. As long as 3 conditions above are met, further transmitter test will not be required. However, the OEM integrator is still responsible for testing their end-product for any additional compliance requirement with this module installed (for example, digital device emission, PC peripheral requirements, etc.)

Notice:

In the event that these conditions cannot be met (for example certain laptop configuration or co-location with another transmitter), then the FCC authorization for this module in combination with the host equipment is no longer considered valid and the FCC ID of the module cannot be used on the final product. In these and circumstance, the OEM integrator will be responsible for re-evaluating. The end product (including the transmitter) and obtaining a separate FCC authorization. The final end product must be labeled in a visible area with the following: **”Contains Transmitter Module FCC ID: 2AC7Z-ESPWROOMDA”** or **”Contains FCC ID: 2AC7Z-ESPWROOMDA”**.

4 Related Documentation and Resources

Related Documentation

- ESP32 Technical Reference Manual – Detailed information on how to use the ESP32 memory and peripherals.
- ESP32 Series Datasheet – Specifications of the ESP32 hardware.
- ESP32 Hardware Design Guidelines – Guidelines on how to integrate the ESP32 into your hardware product.
- ESP32 ECO and Workarounds for Bugs – Correction of ESP32 design errors.
- *Certificates*
  http://espressif.com/en/support/documents/certificates
- *ESP32 Product/Process Change Notifications (PCN)*
  http://espressif.com/en/support/documents/pcns
- *ESP32 Advisories* – Information on security, bugs, compatibility, component reliability.
  http://espressif.com/en/support/documents/advisories
- *Documentation Updates and Update Notification Subscription*
  http://espressif.com/en/support/download/documents

Developer Zone

- ESP-IDF Programming Guide for ESP32 – Extensive documentation for the ESP-IDF development framework.
- *ESP-IDF* and other development frameworks on GitHub.
  http://github.com/espressif
- *ESP32 BBS Forum* – Engineer-to-Engineer (E2E) Community for Espressif products where you can post questions, share knowledge, explore ideas, and help solve problems with fellow engineers.
  http://esp32.com/
- *The ESP Journal* – Best Practices, Articles, and Notes from Espressif folks.
  http://medium.com/the-esp-journal
- See the tabs *SDKs and Demos*, *Apps*, *Tools*, *AT Firmware*.
  http://espressif.com/en/support/download/sdks-demos

Products

- *ESP32 Series SoCs* – Browse through all ESP32 SoCs.
  http://espressif.com/en/products/socs?id=ESP32
- *ESP32 Series Modules* – Browse through all ESP32-based modules.
  http://espressif.com/en/products/modules?id=ESP32
- *ESP32 Series DevKits* – Browse through all ESP32-based devkits.
  http://espressif.com/en/products/devkits?id=ESP32
- *ESP Product Selector* – Find an Espressif hardware product suitable for your needs by comparing or applying filters.
  http://products.espressif.com/#/product-selector?language=en

Contact Us

- See the tabs *Sales Questions*, *Technical Enquiries*, *Circuit Schematic & PCB Design Review*, *Get Samples* (Online stores), *Become Our Supplier*, *Comments & Suggestions*.
  http://espressif.com/en/contact-us/sales-questions

Revision History

Date	Version	Release notes
2021-12-10	v0.5	For certification only

www.espressif.com