EK057

User Manual

Prerelease version 0.2
Espressif Systems
Copyright © 2020

About This Document

This user manual shows how to get started with EK057 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 customers updated on changes to technical documentation. Please subscribe at www.espressif.com/en/subscribe. Note that you need to update your subscription to receive notifications of new products you are not currently subscribed to.

Certification

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

Disclaimer and Copyright Notice

Information in this document, including URL references, is subject to change without notice. THIS DOCUMENT IS PROVIDED AS IS WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE.
All liability, including liability for infringement of any proprietary rights, relating to use of information in this document is disclaimed. No licenses express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein. The Wi-Fi Alliance Member logo is a trademark of the Wi-Fi Alliance. The Bluetooth logo is a registered trademark of Bluetooth SIG.
All trade names, trademarks and registered trademarks mentioned in this document are property of their respective owners, and are hereby acknowledged.
Copyright © 2020 Espressif Systems (Shanghai) Co., Ltd. All rights reserved.

Contents

1 Overview 1
1.1 Module Overview 1
1.2 Pin Description 1

2 Get Started on EK057 3
2.1 What You Need 3
2.2 Hardware Connection 3
2.3 Set up Development Environment 4
2.3.1 Install Prerequisites 4
2.3.2 Get ESP-IDF 4
2.3.3 Set up Tools 5
2.3.4 Set up Environment Variables 5
2.4 Create Your First Project 5
2.4.1 Start a Project 5
2.4.2 Connect Your Device 5
2.4.3 Configure 6
2.4.4 Build the Project 6
2.4.5 Flash onto the Device 7
2.4.6 Monitor 8

3 Learning Resources 10
3.1 Must-Read Documents 10
3.2 Must-Have Resources 10

Revision History 12

1. Overview

1.1 Module Overview

EK057 is a powerful, generic Wi-Fi+Bluetooth®+Bluetooth® LE MCU module that targets a wide variety of applications, ranging from low-power sensor networks to the most demanding tasks, such as voice encoding, music streaming and MP3 decoding.

Table 1: EK057 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	UART, SPI, I2C, I2S, GPIO, ADC
	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 °C ~ +85 °C
	Moisture sensitivity level (MSL)	Level 3

1.2 Pin Description

The module has 14 pins and 7 testing points. See pin definitions in Table 2.

Table 2: Pin Definitions

Name	No.	Type	Function
IO32	A1	I/O	GPIO32, XTAL_32K_P (32.768 kHz crystal oscillator input), ADC1_CH4, TOUCH9, RTC_GPIO9
IO16	A2	I/O	GPIO16, HS1_DATA4, U2RXD, EMAC_CLK_OUT
IO17	A3	I/O	GPIO17, HS1_DATA5, U2TXD, EMAC_CLK_OUT_180
IO5	A4	I/O	GPIO5, VSPICS0, HS1_DATA6, EMAC_RX_CLK
3V3	A5	P	Power supply
GND	A6	P	Ground

Name	No.	Type	Function
GND	A7	P	Ground
GND	A8	P	Ground
GND	A9	P	Ground
IO18	A10	I/O	GPIO18, VSPICLK, HS1_DATA7
IO23	A11	I/O	GPIO23, VSPID, HS1_STROBE
IO19	A12	I/O	GPIO19, VSPIQ, U0CTS, EMAC_TXD0
IO33	A13	I/O	GPIO33, XTAL_32K_N (32.768 kHz crystal oscillator output), ADC1_CH5, TOUCH8, RTC_GPIO8
EN	A14	I	High: On; enables the chip Low: Off; the chip powers off Note: Do not leave the pin floating.
IO14	TP22	I/O	GPIO14, ADC2_CH6, TOUCH6, RTC_GPIO16, MTMS, HSPICLK, HS2_CLK, SD_CLK, EMAC_TXD2
IO15	TP21	I/O	GPIO15, ADC2_CH3, TOUCH3, MTDO, HSPICS0, RTC_GPIO13, HS2_CMD, SD_CMD, EMAC_RXD3
IO13	TP18	I/O	GPIO13, ADC2_CH4, TOUCH4, RTC_GPIO14, MTCK, HSPID, HS2_DATA3, SD_DATA3, EMAC_RX_ER
IO12	TP17	I/O	GPIO12, ADC2_CH5, TOUCH5, RTC_GPIO15, MTDI, HSPIQ, HS2_DATA2, SD_DATA2, EMAC_TXD3
IO0	TP19	I/O	GPIO0, ADC2_CH1, TOUCH1, RTC_GPIO11, CLK_OUT1, EMAC_TX_CLK
RXD	TP16	I/O	GPIO3, U0RXD, CLK_OUT2
TXD	TP20	I/O	GPIO1, U0TXD, CLK_OUT3, EMAC_RXD2

2. Get Started on EK057

2.1 What You Need

To develop applications for EK057 module you need:

- 1 x EK057 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 EK057 module to the RF testing board as shown in Figure 1.

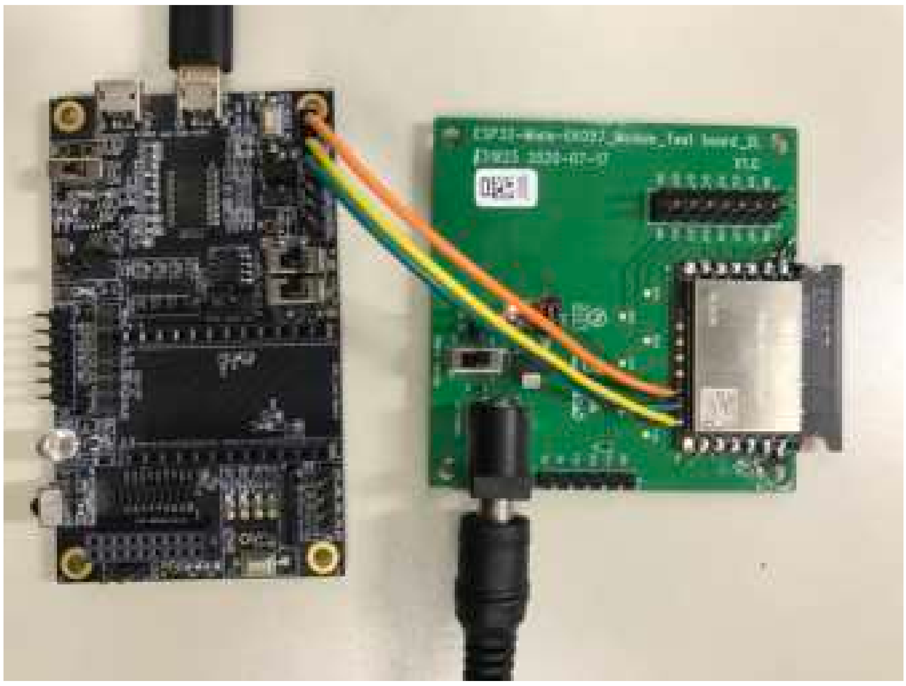


Figure 1: Hardware Connection

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. EK057 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 EK057, please refer to EK057 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 EK057 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 EK057 module.

2.4 Create Your First Project

2.4.1 Start a Project

Now you are ready to prepare your application for EK057 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 EK057 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.

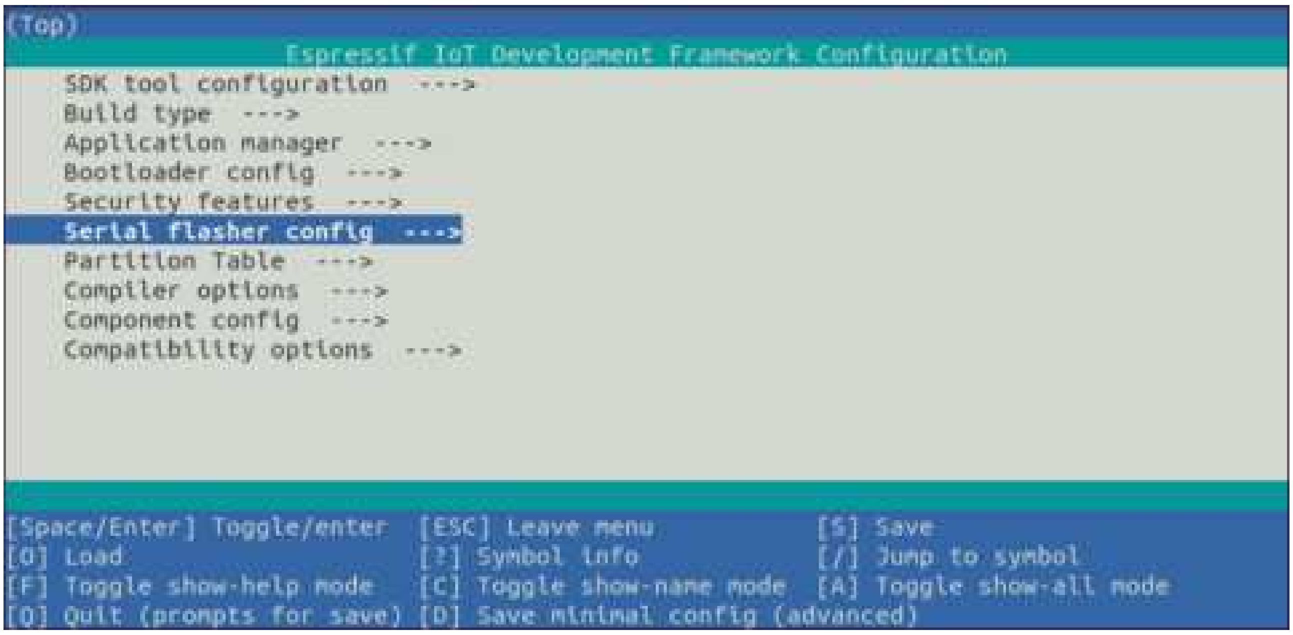
2.4.3 Configure

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 2: 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 EK057 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.

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 ESP32D0WDQ6 (revision 1)
Features: WiFi, BT, Dual Core
Uploading stub ...
Running stub ...
Stub running ...
Changing baud rate to 460800
Changed.
```

```
Configuring flash size ...
Auto-detected Flash size: 4MB
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...
```

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.

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 1, 2MB external 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 EK057 module! Now you are ready to try some other examples in ESP-IDF, or go right to developing your own applications.

3. Learning Resources

3.1 Must-Read Documents

The following link provides documents related to ESP32.

- *ESP32 Datasheet*
  This document provides an introduction to the specifications of the ESP32 hardware, including overview, pin definitions, functional description, peripheral interface, electrical characteristics, etc.
- *ESP32 ECO V3 User Guide*
  This document describes differences between V3 and previous ESP32 silicon wafer revisions.
- *ECO and Workarounds for Bugs in ESP32*
  This document details hardware errata and workarounds in the ESP32.
- *ESP-IDF Programming Guide*
  It hosts extensive documentation for ESP-IDF ranging from hardware guides to API reference.
- *ESP32 Technical Reference Manual*
  The manual provides detailed information on how to use the ESP32 memory and peripherals.
- ESP32 Hardware Resources
  The zip files include the schematics, PCB layout, Gerber and BOM list of ESP32 modules and development boards.
- *ESP32 Hardware Design Guidelines*
  The guidelines outline recommended design practices when developing standalone or add-on systems based on the ESP32 series of products, including the ESP32 chip, the ESP32 modules and development boards.
- *ESP32 AT Instruction Set and Examples*
  This document introduces the ESP32 AT commands, explains how to use them, and provides examples of several common AT commands.
- *Espressif Products Ordering Information*

3.2 Must-Have Resources

Here are the ESP32-related must-have resources.

- ESP32 BBS
  This is an Engineer-to-Engineer (E2E) Community for ESP32 where you can post questions, share knowledge, explore ideas, and help solve problems with fellow engineers.
- ESP32 GitHub
  ESP32 development projects are freely distributed under Espressif's MIT license on GitHub. It is established to help developers get started with ESP32 and foster innovation and the growth of general knowledge about the hardware and software surrounding ESP32 devices.

- ESP32 Tools
  This is a webpage where users can download ESP32 Flash Download Tools and the zip file ”ESP32 Certification and Test”.
- ESP-IDF
  This webpage links users to the official IoT development framework for ESP32.
- ESP32 Resources
  This webpage provides the links to all available ESP32 documents, SDK and tools.

Revision History

Date	Version	Release notes
2020-11-06	V0.2	Preliminary release v0.2