ESP32-C3-MINI-1

User Manual

Pre-release v0.1

Espressif Systems

About This Document

This user manual shows how to get started with the ESP32-C3-MINI-1 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-C3-MINI-1 is a general-purpose Wi-Fi and Bluetooth LE module. The rich set of peripherals and a small size make this module an ideal choice for smart homes, industrial automation, health care, consumer electronics, etc.

Table 1: ESP32-C3-MINI-1 Specifications

Categories	Parameters	Specifications
Wi-Fi	Protocols	802.11 b/g/n (up to 150 Mbps)
	Frequency range	2412 ~ 2462 MHz
Bluetooth®	Protocols	Bluetooth® LE: Bluetooth 5 and Bluetooth mesh
	Radio	Class-1, class-2 and class-3 transmitter
Hardware	Module interfaces	GPIO, SPI, UART, I2C, I2S, remote control peripheral, LED PWM controller, general DMA controller, TWAI® controller (compatible with ISO 11898-1), temperature sensor, SAR ADC
	Integrated crystal	40 MHz crystal
	Operating voltage/Power supply	3.0 V ~ 3.6 V
	Operating current	Average: 80 mA
	Minimum current delivered by power supply	500 mA
	Ambient temperature	–40 °C ~ +105 °C
	Moisture sensitivity level (MSL)	Level 3

1.2 Pin Description

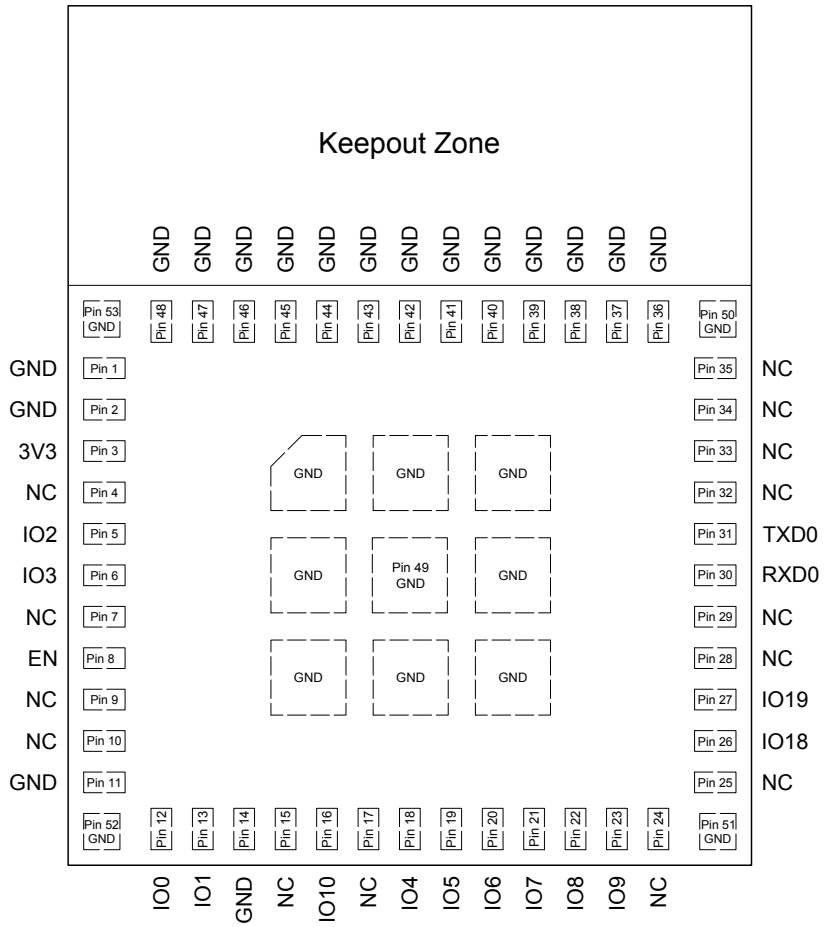

Figure 1: Pin Layout (Top View)

The module has 53 pins. See pin definitions in Table 2.

For peripheral pin configurations, please refer to *ESP32-C3 Family Datasheet*.

Table 2: Pin Definitions

Name	No.	Type	Function
GND	1, 2, 11, 14, 36-53	P	Ground
3V3	3	P	Power supply
NC	4	—	NC
IO2	5	I/O/T	GPIO2, ADC1_CH2, FSPIQ
IO3	6	I/O/T	GPIO3, ADC1_CH3
NC	7	—	NC
EN	8	I	High: on, enables the chip. Low: off, the chip powers off. Note: Do not leave the EN pin floating.
NC	9	—	NC
NC	10	—	NC

Cont'd on next page

Table 2 – cont'd from previous page

Name	No.	Type	Function
IO0	12	I/O/T	GPIO0, ADC1_CH0, XTAL_32K_P
IO1	13	I/O/T	GPIO1, ADC1_CH1, XTAL_32K_N
NC	15	—	NC
IO10	16	I/O/T	GPIO10, FSPICS0
NC	17	—	NC
IO4	18	I/O/T	GPIO4, ADC1_CH4, FSPIHD, MTMS
IO5	19	I/O/T	GPIO5, ADC2_CH0, FSPIWP, MTDI
IO6	20	I/O/T	GPIO6, FSPICLK, MTCK
IO7	21	I/O/T	GPIO7, FSPID, MTDO
IO8	22	I/O/T	GPIO8
IO9	23	I/O/T	GPIO9
NC	24	—	NC
NC	25	—	NC
IO18	26	I/O/T	GPIO18
IO19	27	I/O/T	GPIO19
NC	28	—	NC
NC	29	—	NC
RXD0	30	I/O/T	GPIO20, U0RXD,
TXD0	31	I/O/T	GPIO21, U0TXD
NC	32	—	NC
NC	33	—	NC
NC	34	—	NC
NC	35	—	NC

2 Get Started on ESP32-C3-MINI-1

2.1 What You Need

To develop applications for ESP32-C3-MINI-1 module you need:

- 1 x ESP32-C3-MINI-1 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-C3-MINI-1 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-C3-MINI-1 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-C3-MINI-1, please refer to ESP32-C3-MINI-1 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 chips. Users can develop applications with ESP chips 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-C3-MINI-1 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-C3-MINI-1 module.

2.4 Create Your First Project

2.4.1 Start a Project

Now you are ready to prepare your application for ESP32-C3-MINI-1 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-C3-MINI-1 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-C3 as the target and run the project configuration utility 'menuconfig'.

```
cd ~/esp/hello_world
idf.py set-target esp32c3
idf.py menuconfig
```

Setting the target with 'idf.py set-target esp32c3' 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:

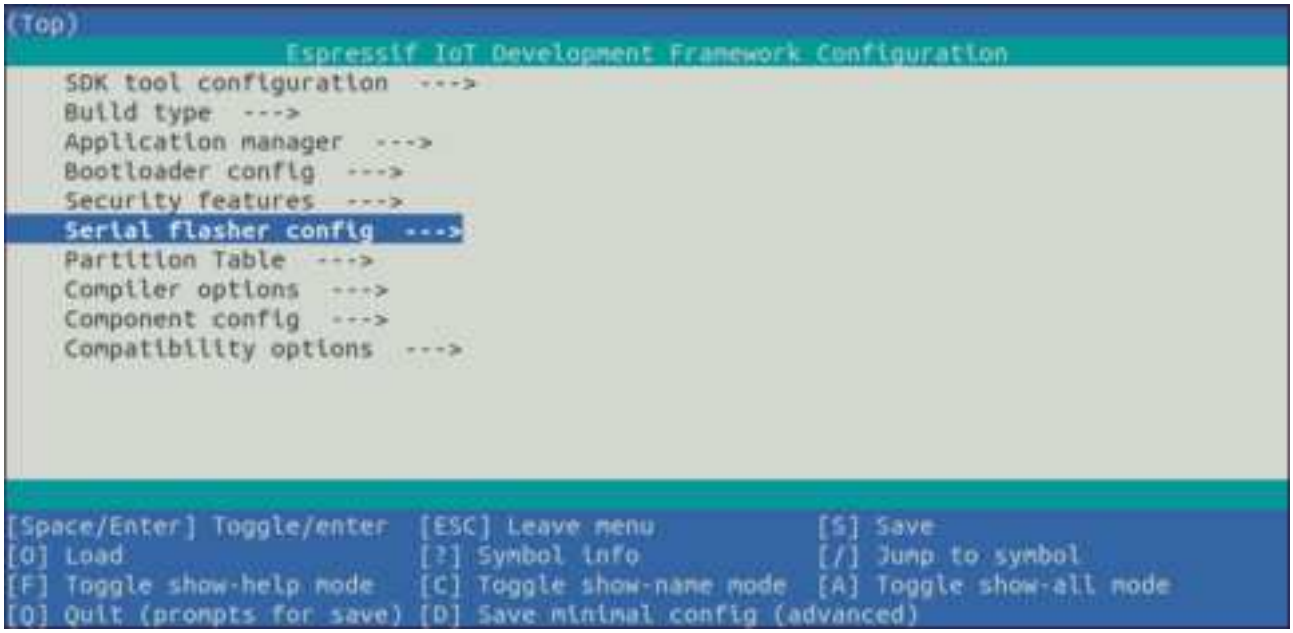

Figure 3: Project Configuration - Home Window

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-C3-MINI-1 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.

```
        ...
        esptool.py --chip esp32c3 -p /dev/ttyUSB0 -b 460800 --before=default_reset --after
            =hard_reset write_flash --flash_mode dio --flash_freq 80m --flash_size 2MB 0x
            8000 partition_table/partition-table.bin 0x0 bootloader/bootloader.bin 0x10000
             hello-world.bin
        esptool.py v3.0
        Serial port /dev/ttyUSB0
        Connecting....
        Chip is ESP32-C3
        Features: Wi-Fi
        Crystal is 40MHz
        MAC: 7c:df:a1:40:02:a4
        Uploading stub...
        Running stub...
        Stub running...
        Changing baud rate to 460800
        Changed.
        Configuring flash size...
        Compressed 3072 bytes to 103...
        Writing at 0x00008000... (100 %)
        Wrote 3072 bytes (103 compressed) at 0x00008000 in 0.0 seconds (effective 4238.1
            kbit/s)...
        Hash of data verified.
```

```
        Compressed 18960 bytes to 11311...
        Writing at 0x00000000... (100 %)
        Wrote 18960 bytes (11311 compressed) at 0x00000000 in 0.3 seconds (effective 584.9
             kbit/s)...
        Hash of data verified.
        Compressed 145520 bytes to 71984...
        Writing at 0x00010000... (20 %)
        Writing at 0x00014000... (40 %)
        Writing at 0x00018000... (60 %)
        Writing at 0x0001c000... (80 %)
        Writing at 0x00020000... (100 %)
        Wrote 145520 bytes (71984 compressed) at 0x00010000 in 2.3 seconds (effective
            504.4 kbit/s)...
        Hash of data verified.

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

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 esp32c3 chip with 1 CPU core, WiFi/BLE, 4MB 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 ESP32-C3-MINI-1 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

Please familiarize yourself with the following documents:

- *ESP32-C3 Family Datasheet*

  This is an introduction to the specifications of the ESP32-C3 hardware, including overview, pin definitions, functional description, peripheral interface, electrical characteristics, etc.

- *ESP-IDF Programming Guide*

  Extensive documentation for the ESP-IDF development framework, ranging from hardware guides to API reference.

- *ESP32-C3 Technical Reference Manual*

  Detailed information on how to use the ESP32-C3 memory and peripherals.

- *Espressif Products Ordering Information*

3.2 Important Resources

Here are the important ESP32-C3-related resources.

- ESP32 BBS

  Engineer-to-Engineer (E2E) Community for Espressif products where you can post questions, share knowledge, explore ideas, and help solve problems with fellow engineers.

Revision History

Date	Version	Release notes
2021-02-01	V0.1	Preliminary release

www.espressif.com

Disclaimer and Copyright Notice