T-MICRO32

User Guide

Version 2.0

Copyright © 2019

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 T-MICRO32. Through a simple example, this document illustrates how to use ESP-IDF (Xinyuan IoT Development Framework), including the menu based configuration wizard, compiling the ESP-IDF and firmware download to the ESP32 module.

Release Notes

Date	Version	Release notes
2019.04	V2.0	First release.

Table of Contents

1. Introduction 1
   1.1. T-MICRO32 1
   1.2. ESP-IDF 1
   1.3. Preparation 1
2. Get 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. Connect 5
5. Configure 6
6. Build and Flash 7
   6.1. Build and Flash 7
   6.2. Monitor 8
7. SSC Command Reference 9
   7.1. op 9
   7.2. sta 9
   7.3. ap 10
   7.4. mac 10
   7.5. dhcp 11
   7.6. ip 11
   7.7. reboot 12
   7.8. ram 12

Appendix – Notices 13
   I. Federal Communications Commission (FCC) Declaration of Conformity 13

1. Introduction

1.1. T-MICRO32

T-MICRO32 is a powerful, generic Wi-Fi+BT+BLE 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. **T-MICRO32** integrates a U.FL connector.

1.1.1 CE Part

(1)Frequency Range:2.4G Wi-Fi: 2412~2472MHz/2422~2462MHz
BLE:2402~2480MHz Bluetooth: 2402~2480MHz

(2)Transmit Power: 2.4G Wi-Fi: 18.68dBm(802.11b),9.39dBm(802.11g)
9.94dBm(802.11n20),9.87dBm(802.11n40)
BLE:4.99dBm Bluetooth: 6.90dBm

1.1.2 FCC Part

(1)Frequency Range:2.4G Wi-Fi: 2412~2462MHz
BLE: 2402~2480MHz Bluetooth: 2402~2480MHz

(2)Transmit Power:2.4G Wi-Fi: 22.38dBm(802.11b),14.74dBm(802.11g)
15.23dBm(802.11n20),14.68dBm(802.11n40)
BLE: 5.56dBm Bluetooth: 9.17dBm

The manufacturer is Shenzhen Xin Yuan Electronic Technology Co, Ltd.

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. Preparation

To develop applications for ESP32 you need:

- PC loaded with either Windows, Linux or Mac operating system
- Toolchain to build the Application for ESP32
- ESP-IDF that essentially contains API for ESP32 and scripts to operate the Toolchain
- A text editor to write programs (Projects) in C, e.g. Eclipse
- The ESP32 board itself and a USB cable to connect it to the PC

2. Get Started

2.1. Standard Setup of Toolchain for Linux

The quickest way to start development with ESP32 is by installing a prebuilt toolchain. Pick up your OS below and follow provided instructions.

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:***
>
> - *While cloning submodules on Windows platform, the* `git clone` *command may print some output starting* `': not a valid identifier...`. *This is a known issue but the git clone still succeeds without any problems.*
> - *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. To do so, follow instructions specific to ***:ref:`Windows <add-idf_path-to-profile-windows>`***, ***:ref:`Linux and MacOS <add-idf_path-to-profile-linux-macos>`*** in section ***:doc:`add-idf_path-to-profile`***.

3. Start a Project

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

Copy ***:example:`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 ***:idf:`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. Connect

You are almost there. To be able to proceed further, connect ESP32 board to PC, check under what serial port the board is visible and verify if serial communication works. If you are not sure how to do it, check instructions in section ***:doc:`establish-serial-connection`***. Note the port number, as it will be required in the next step.

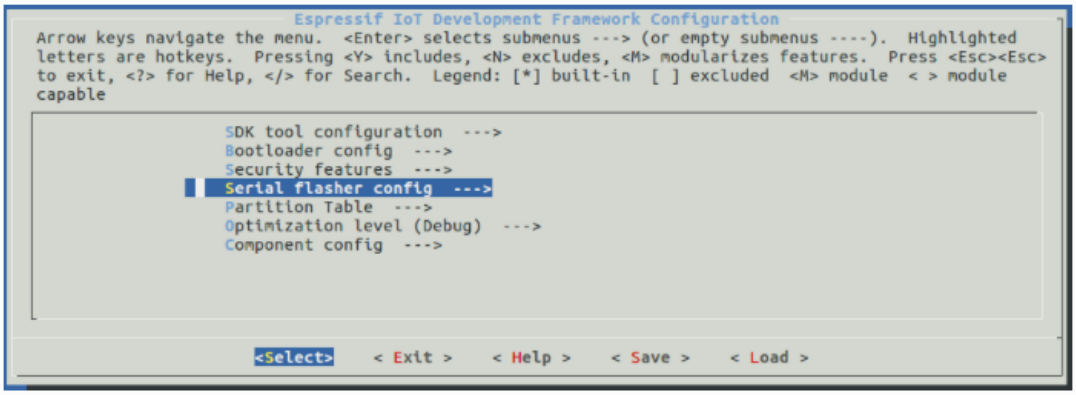
5. Configure

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.

> 📖 ***Notes:***
>
> - *On Windows, serial ports have names like COM1. On MacOS, they start with **/dev/cu.**. On Linux, they start with **/dev/tty.** (See **:doc:`establish-serial-connection`** for full details.)*
> - *If you are Arch Linux user, navigate to **SDK tool configuration** and change the name of **Python 2 interpreter** from **python** to **python2**.*
> - *Most ESP32 development boards have a 40 MHz crystal installed. However, some boards use a 26 MHz crystal. If your board uses a 26MHz crystal, or you get garbage output from serial port after code upload, adjust the **:ref:`CONFIG_ESP32_XTAL_FREQ_SEL`** option in **menuconfig**.*

6. Build and Flash

6.1. 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.

If you'd like to use the Eclipse IDE instead of running make, check out the ***:doc:`Eclipse guide <eclipse-setup>`***.

6.2. Monitor

To see if “hello_world” application is indeed running, type `make monitor`. This command is launching ***:doc:`IDF Monitor <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`. Check section ***:doc:`IDF Monitor <idf-monitor>`*** for handy shortcuts and more details on using this application.

That's all what you need to get started with ESP32!

Now you are ready to try some other ***:idf:`examples`***, or go right to developing your own applications.

7. SSC Command Reference

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

7.1. 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

7.2. 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.

7.3. 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.

7.4. 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.

7.5. 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.

7.6. 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.

7.7. reboot

Description

reboot command is used to reboot the board.

Example

```
reboot
```

7.8. ram

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

Example

```
ram
```

FCC Label: The FCC ID is on the front of the device. It is easily visible.

The device FCC ID is 2ASYE-T-MICRO32.

A label with the following statements must be attached to the host end product:

This device contains FCC ID: 2ASYE-T-MICRO32.

The manual provides guidance to the host manufacturer will be included in the documentation that will be provided to the OEM.

The module is limited to installation in mobile or fixed applications.

The separate approval is required for all other operating configurations, including portable configurations and different antenna configurations.

The OEM integrators are responsible for ensuring that the end-user has no manual or instructions to remove or install module.

The module is limited to OEM installation ONLY.

Module grantee (the party responsible for the module grant) shall provide guidance to the host manufacturer for ensuring compliance with the Part 15 Subpart B requirements.

The host manufacturer is responsible for additional testing to verify compliance as a composite system. When testing the host device for compliance with the Part 15 Subpart B requirements, the host manufacturer is required to show compliance with the Part 15 Subpart B while the transmitter module(s) are installed and operating. The modules should be transmitting and the evaluation should confirm that the module's intentional emissions are compliant (i.e. fundamental and out of band emissions) with the Radio essential requirements. The host manufacturer must verify that there are no additional unintentional emissions other than what is permitted in the Part 15 Subpart B or emissions are complaint with the Radio aspects.

CAUTION:
Any changes or modifications not expressly approved by the grantee of this device could void the user's authority to operate the equipment.

FCC RF Exposure Requirements

This device complies with FCC RF radiation exposure limits set forth for an uncontrolled enviroment.

The antenna(s) used for this transmitter must not be co-located or operating in conjunction with any other antenna or transmitter and must be installed to provide a separation distance of at least 20cm from all persons.

FCC Regulations

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 operation.

This device has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 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.

Company Address

401 4F Bldg.building B. No.5, 1st Rd Bantianshangxue Technology Park ,Bantian St, Longgang Dist.,Shenzhen Guangdong China