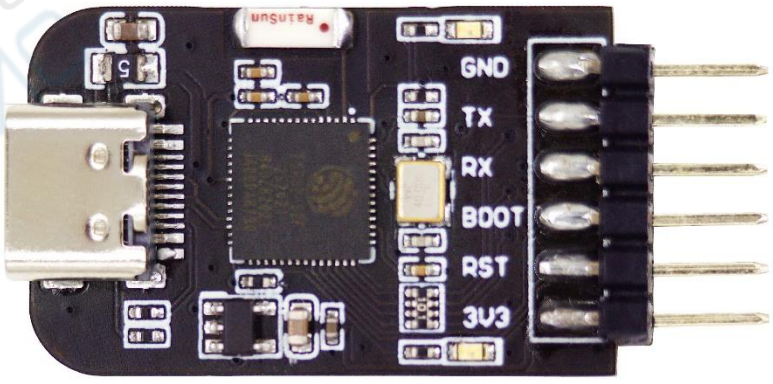
ESPLink v1.2 (v1.0)

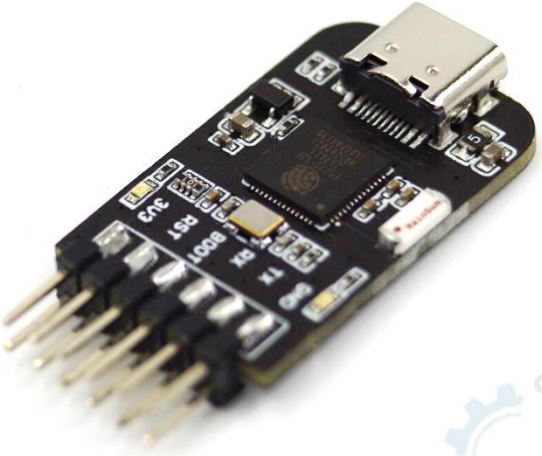
中文

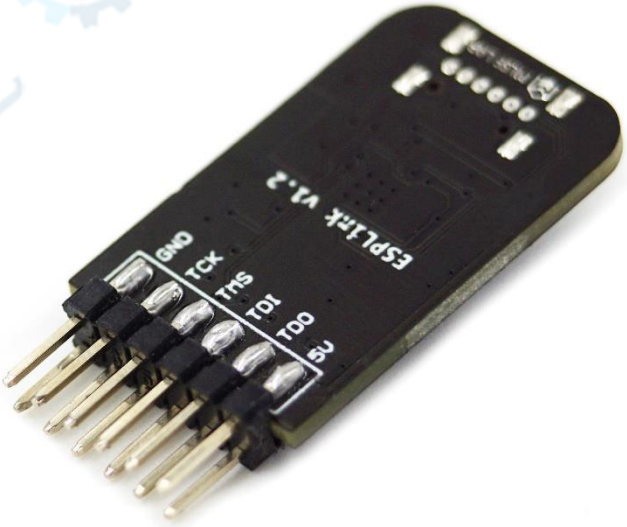
- ESPLink Introduce
- Features
- Product Link
- Reference

ESPLink Introduce

ESPLink is a debug tool build for Expressif's ESP chips made by MuseLab based on Espressif's esp-usb-bridge, all Espressif's ESP chips are supported, currently supports ESP8266/ESP32/ESP32-S2/ESP32-C3/ESP32-S3 (also can update firmware to support Espressif's latest chips), with usb-to-serial (compatible with traditional use), drag-n-drop program, and jtag debug, made it very convenient for development and test.







Features

USB-to-Serial

compatible with the traditional use, can used to program with esptool.py or monitor the serial output message. examples for reference:

```
$idf.py -p /dev/ttyACM0 flash monitor
$esptool.py --chip esp32c3 \
           -p /dev/ttyACM0 \
           -b 115200 \
           --before=default_reset \
           --after=hard_reset \
           --no-stub \
           write_flash \
           --flash_mode dio \
           --flash_freq 80m \
           --flash_size 2MB \
           0x0     esp32c3/bootloader.bin \
           0x8000  esp32c3/partition-table.bin \
           0x10000 esp32c3/blink_100.bin
```

Drag-and-Drop Program

the ESPLink support drag-n-drop program, after power on the board, a virtual USB Disk named `ESPLink` will appear, just drag the flash image into the ESPLink, wait for some seconds, then the ESPLink will automatically complete the program work. it's very convenient since it's no need to rely on external tools and work well on any platform (Win/Linux/Mac .etc), some typical usage scenarios: quickly verification, compile on cloud server and program on any PC, firmware upgrade when used on commercial products .etc
note: the flash image is in uf2 format, can generated with espressif's official tool `idf.py`

```
$idf.py uf2
$cp build/uf2.bin /media/pi/ESPLink/
```

JTAG Debug

ESPLink support jtag interface to debug the ESP32 series chip, it's useful for product developer to fix the bug when system crash, here are the instructions for reference (target chip is ESP32-S2 here as an example)

Openocd Install

```
$git clone https://github.com/espressif/openocd-esp32.git
$cd openocd-esp32
$./bootstrap
$./configure
$make -j
$sudo make install
```

Attach to ESP32-S2

```
pi@raspberrypi:~/oss/openocd-esp32 $ sudo ./src/openocd -s /home/pi/oss/openocd-
esp32/tcl -f tcl/interface/esp_usb_bridge.cfg -f tcl/target/esp32s2.cfg
Open On-Chip Debugger  v0.11.0-esp32-20220411-5-g03cd2031 (2022-04-25-09:45)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
Info : only one transport option; autoselect 'jtag'
Info : esp_usb_jtag: VID set to 0x303a and PID to 0x1002
Info : esp_usb_jtag: capabilities descriptor set to 0x30a
adapter speed: 4000 kHz
Warn : Transport "jtag" was already selected
Info : Listening on port 6666 for tcl connections
Info : Listening on port 4444 for telnet connections
Info : esp_usb_jtag: serial (84F703D20134)
Info : esp_usb_jtag: Device found. Base speed 750KHz, div range 1 to 1
Info : clock speed 750 kHz
Info : JTAG tap: esp32s2.cpu tap/device found: 0x120034e5 (mfg: 0x272 (Tensilica),
part: 0x2003, ver: 0x1)
Info : esp32s2: Debug controller was reset.
Info : esp32s2: Core was reset.
Info : starting gdb server for esp32s2 on 3333
Info : Listening on port 3333 for gdb connections
```

Debug

when attach success, open another ternimal to debug, you can use gdb or telnet, explained separately as follows

Debug with Gdb

```
$xtensa-esp32s2-elf-gdb -ex 'target remote 127.0.0.1:3333' ./build/blink.elf
(gdb) info reg
(gdb) s
(gdb) continue
```

Debug with telnet

```
$telnet localhost 4444
>reset
>halt
>reg
>step
>reg pc
>resume
```

Pin Description

Power

GND,3V3,5V

JTAG

TCK,TMS,TDI,TDO

UART

TX,RX
USB-Serial TX and RX

BOOT

BOOT
set low, and reset the board, then the board will enter to bootloader for flash program, note that BOOT and RST is controled by ESPLink USB-Serial DTR and RTS, compatible with other USB-Serial chip like CP2102 and CH340.