

2. Customising ESP-AT Firmware



In the previous tutorial, we saw how to build and flash ESP-AT, but we didn't look at what configuration options were available. In this tutorial we will learn how to configure the ESP-AT firmware in order to use the UART-0 interface for both info logging and AT commands.

For this tutorial you'll need:

Introduction #

All the ESP* modules come with the default AT firmware installed, but this firmware sends logs to the UART-0 interface (on our board, this is connected to the USB port throught the USB/UART bridge), and expects the commands to be sent to the UART-1 interface:



To use the UART-0 interface for both logs and commands, thus not requiring any additional USB/UART bridge, we must alter the configuration of the ESP-AT firmware. Recompiling the AT firmware is useful for other purposes too, for instance the SPP protocol of classic BT is not enabled by default, and can be enabled with a few configuration changes.

Updating Configuration with menuconfig #

First let's check to see if the UART-0 interface is set as default console output port, so first head to the location you downloaded the ESP-AT source and open the menuconfig tool: cd ~/esp-at

./build.py menuconfig

Go to Component config > Common ESP-related and you should see UART for console output set to: Default: UART0, TX=GPI01, RX=GPI03





Note that this is the console **output** port, it is not the **command** port, the configuration for the command port is not in the menuconfig, but it is in a configuration file:

gedit components/customized_partitions/raw_data/factory_param/factory_param_data.csv &

This file lists all the modules and the default GPIOs and parameters for the AT firmware. Since we're using the ESP32-WROOM-32*, we'll need to update line 2:

| Open | | | <pre>*factory_param_data.csv ~/esp/esp-at/components/customized_partitions/raw_data/factory_param</pre> | Save | | | • 😣 |
|--|------|-------|---|----------|--------|--------|----------|
| 1 platfo | ۰rm, | modul | e_name,description,magic_flag,version,module_id,tx_max_pov | wer,uart | t_port | t,sta | rt_chann |
| 2 PLATEC | RM_ | ESP32 | ,WROOM-32,,0xfcfc,2,1,78,0,1,13,CN,115200,1,3,-1,-1,-1,-1 | | | | |
| 3 PLATEC | RM_ | ESP32 | WROVER-32,,0xfcfc,2,2,78,1,1,13,CN,115200,22,19,15,14,-1, | ,-1 | | | |
| 4 PLATFO | RM_ | ESP32 | ,PICO-D4,,0xfcfc,2,3,78,1,1,13,CN,115200,22,19,15,14,-1,-1 | 1 | | | |
| 5 PLATFO | RM_ | ESP32 | ,SOLO-1,,0xfcfc,2,4,78,1,1,13,CN,115200,17,16,15,14,-1,-1 | | | | |
| 6 PLATFO | RM_ | ESP32 | ,MINI-1, "ESP32-U4WDH chip inside", 0xfcfc, 2, 5, 78, 1, 1, 13, CN, | ,115200, | ,22,19 | 9,15,3 | 14,-1,-1 |
| 7 PLATEC | RM_ | ESP32 | ,ESP32-D2WD, "2MB flash, No OTA", 0xfcfc, 2, 6, 78, 1, 1, 13, CN, 11 | 15200,22 | 2,19,1 | 15,14 | ,-1,-1 |
| 8 PLATEC | RM | ESP82 | 56, WROOM-02, TX:15 RX:13, 0xfcfc, 2, 0, 78, 0, 1, 13, CN, 115200, 15, | ,13,3,1, | ,-1,-1 | L | |
| 9PLATFORM_ESP8266,WR00M-5V2L,5V UART level,0xfcfc,2,1,78,0,1,13,CN,115200,15,13,3,1,5,-1 | | | | | | | |
| 0 PLATFO | RM_ | ESP82 | 56,ESP8266_1MB,No OTA,0xfcfc,2,2,78,0,1,13,CN,115200,15,13 | 3,3,1,-1 | 1,-1 | | |
| 1 PLATFO | RM | ESP82 | 56, WROOM-02-N, TX:1 RX:3, 0xfcfc, 2, 3, 78, 0, 1, 13, CN, 115200, 1, 3 | 3,-1,-1, | -1,-1 | L | |
| 2 PLATEC | RM | ESP82 | 56,WROOM-S2,,0xfcfc,2,4,78,0,1,13,CN,115200,15,13,3,1,-1,. | -1 | | | |
| 3 PLATEC | RM | ESP32 | S2,WROOM,, 0xfcfc, 2, 0, 78, 1, 1, 13, CN, 115200, 17, 21, 20, 19, -1, -1 | 1 | | | |
| 4 PLATEC | RM | ESP32 | S2,WROVER,,0xfcfc,2,1,78,1,1,13,CN,115200,17,21,20,19,-1,. | -1 | | | |
| 5 PLATEC | RM_ | ESP32 | 52, SOLO,, 0xfcfc, 2, 2, 78, 1, 1, 13, CN, 115200, 17, 21, 20, 19, -1, -1 | | | | |
| 6 PLATFO | RM_ | ESP32 | <pre>S2,MINI,,0xfcfc,2,3,78,1,1,13,CN,115200,17,21,20,19,-1,-1</pre> | | | | |

With this row we replace the UART (command) port to UART-0, the TX pin to 1 and RX to 3, these are the GPIOs linked to the USB/UART bridge on the ESP32-DEVKIT-C:

PLATFORM_ESP32,WROOM-32,,0xfcfc,2,1,78,0,1,13,CN,115200,1,3,-1,-1,-1

Build & Flash

We are now ready to build and flash the modified firmware: ./build.py flash

As in the last tutorial, if you get a SerialException:

you'll need to add the current user to the group dialout and restart the system: sudo adduser \$USER dialout sudo reboot

AT command test

To test the firmware we have just flashed, open up a serial terminal -I am going to use GTKTerm, which can be found and installed from the software center.

ESP-AT expects CR LF as new line so we must set *Configuration->CR LF auto*. Then open *Configuration->Port* and set the serial port configuration to: Baud Rate: 115200

Parity: none Bits: 8 Stopbits: 1 Flow control: none

| | | Configuratio | on | | 8 | | |
|--------------------------|---|--------------|----|---------------|--------|--|--|
| erial port | | | | | | | |
| Port: | | Baud Rate | 2 | Par | ity: | | |
| /dev/ttyUSB0 | • | 115200 | • | none | • | | |
| Bits: | | Stopbits: | | Flow control: | | | |
| 8 | - | 1 | - | none | - | | |
| 8 Advanced Configural | | 1 Drns | • | none | | | |
| , | | | | ОК | Cancel | | |



If everything was configured correctly, we should see the log output (since the UART-0 interface is still the log port), and once the module has initialised we should see *ready* outputted. You can now give the command at, which is a test command, and input the CR LF – this is achieved in GTKTerm by pressing both *ENTER* and *CTRL-J*:

| GTKTerm - /dev/ttyUSB0 115200-8-N-1 | - | | 8 |
|--|------|-----|-----|
| File Edit Log Configuration Control signals View Help | | | |
| I (106) boot: 5 ota_1 OTA app 00 11 00280000 | 001 | 800 | 00 |
| I (114) boot: End of partition table I (118) boot comm: chip revision: 1, min, application chip revisio | on : | | |
| I (125) esp image: segment 0: paddr=0x00100020 vaddr=0x3f400020 s. | ize= | 0x2 | b3 |
| a8 (177064) map | | | |
| I (198) esp_image: segment 1: paddr=0x0012b3d0 vaddr=0x3ffbdb60 s. | ize= | | 39 |
| d0 (14800) load | | | |
| I (204) esp_image: segment 2: paddr=0x0012eda8 vaddr=0x40080000 s: | ize= | | 004 |
| 00 (1024) load T (205) oon imagaa cogmont 2, naddr-0y0012flb0 yaddr-0y40000400 c | | | |
| 1 (205) esp_image: segment 5: pauli=0x00121100 vauli=0x40000400 s. | IZe= | | 100 |
| I (215) esp image: segment 4: paddr=0x00130018 vaddr=0x400d0018 s | ize= | 0x1 | 1f |
| 008 (1175560) map | | | |
| I (643) esp_image: segment 5: paddr=0x0024f028 vaddr=0x40081260 s. | ize= | 0x1 | .b8 |
| a0 (112800) load | | | |
| I (690) esp_image: segment 6: paddr=0x0026a8d0 vaddr=0x400c0000 s. | ize= | 0x0 | 00 |
| 64 (100) load | | | |
| I (708) boot: Loaded app from partition at offset 0x100000 | | | |
| I (708) DOOT: DISADIING RNG early entropy source | | | |
| ready | | | |
| at | | | |
| | | | |
| OK | | | |
| | | | |
| | | | |

/dev/ttyUSB0 115200-8-N-1

DTR RTS CTS CD DSR RI

The firmware should reply OK and thus our firmware has been correctly configured.

Conclusion #

We configured and compiled the ESP-AT firmware to change the command output from the default UART-1 interface to the UART-0 interface, which is connected to the usb port in the ESP32-DEVKIT-C and we sent our first command and checked that everything works correctly. We are now ready to use the AT firmware for both WiFi and BLE application.