ESP-WROOM-S2 Hardware Evaluation Platform

Version 1.0

About This Guide

This document introduces ESP-WROOM-S2 hardware platform developed by Espressif.

The structure is as below:

Chapter	Title	Content
Chapter 1	Hardware Evaluation Plarform	Hardware evaluation platform details.
Chapter 2	Preparing the Hardware	STM32 development board connection.
Chapter 3	Downloading Firmware to ESP-WROOM-S2	Download firmware to HSPI Flash via S2 Module Serial Interface.
Chapter 4	Host MCU Configuration and Programming	Operation on Host side.
Chapter 5	Host - ESP Interaction	Host and ESP8266 interaction process and scripting tool.

Release Notes

Date	Version	Release notes
2016.08	V1.0	Initial release.

Table of Contents

1. Hardware Evaluation Platform

Hardware evaluation platform is composed of 3 parts: ESP-WROOM-S2 module + ESP-Launcher backplane + STM32 development board (Type: ALIENTEK ELITE STM32F103).

ESP-Launcher details:

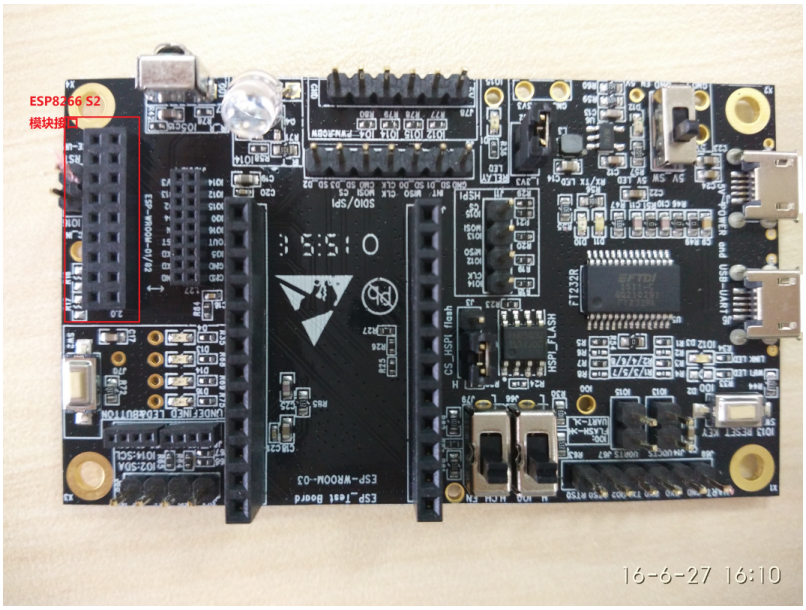

Figure 1-1: ESP-Launcher Backplane

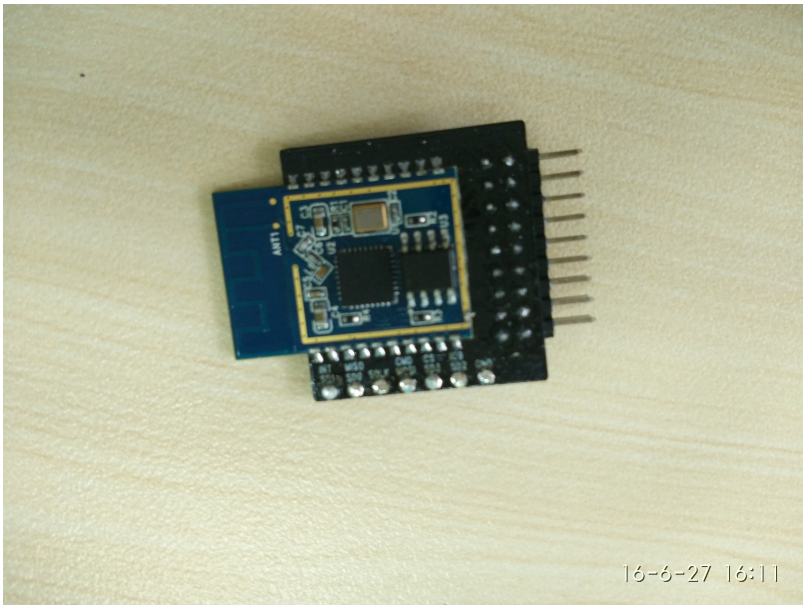

Figure 1-2: ESP-WROOM-S2

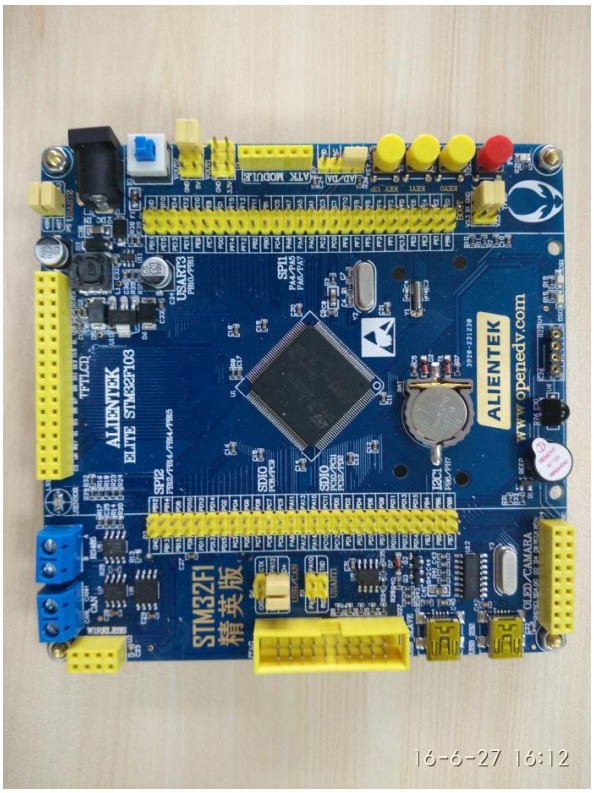

Figure 1-3: The Front View of the STM32 Development Board

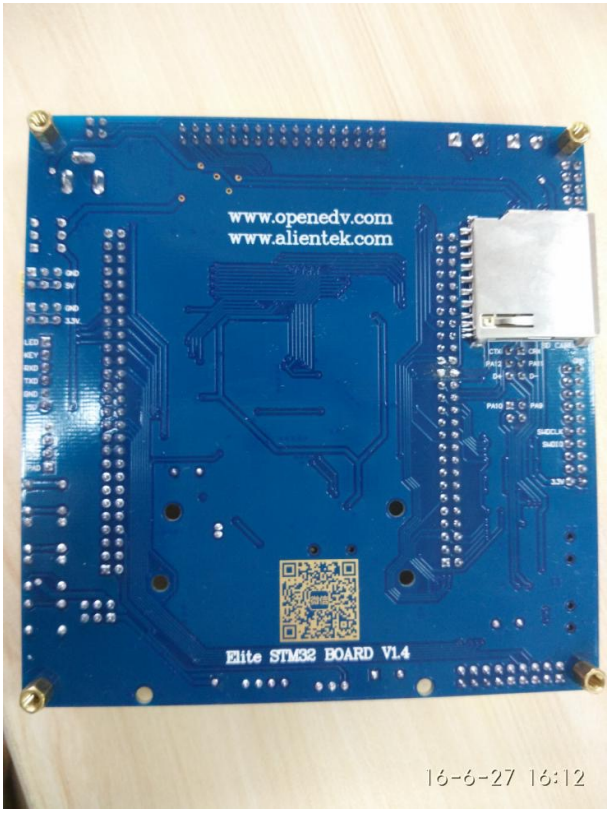

Figure 1-4: The Back View of the STM32 Development Board

2. Preparing the Hardware

1. Remove resistor R29 on the ESP8266 backplane.
2. Lead the SDIO communication lines out of ESP-WROOM-S2 and connect it to the SDIO interface lines on STM32 development board (or any other MCU development board with SDIO host, hereinafter referred to as MCU).In practice, users should follow MCU development board interfacing guidelines when making connections. STM32F103ZET6 SDIO pin functions are as follows:
    - Pin PC8 ————> SD0 function.
    - Pin PC9 ————> SD1 function.
    - Pin PC10 ———-> SD2 function.
    - Pin PC11 ———-> SD3 function.
    - Pin PC12 ———-> SCLK function.
    - Pin PD2 ————> CMD function.

> ⚠ ***Notice:***
>
> *Lead the SDIO interface pins with short jumper wires of approximately equal length and keep them away from other components operating at high frequencies! Pull-up or termination resistors matching the host MCU board may be used.*

3. Connect ESP-WROOM-S2 to the interface on ESP8266 backplane. The location of the interface's location is marked on the upper left side of Figure 1-1 for your reference.

> ⚠ ***Notice:***
>
> *Please connect the pins in the correct order.*

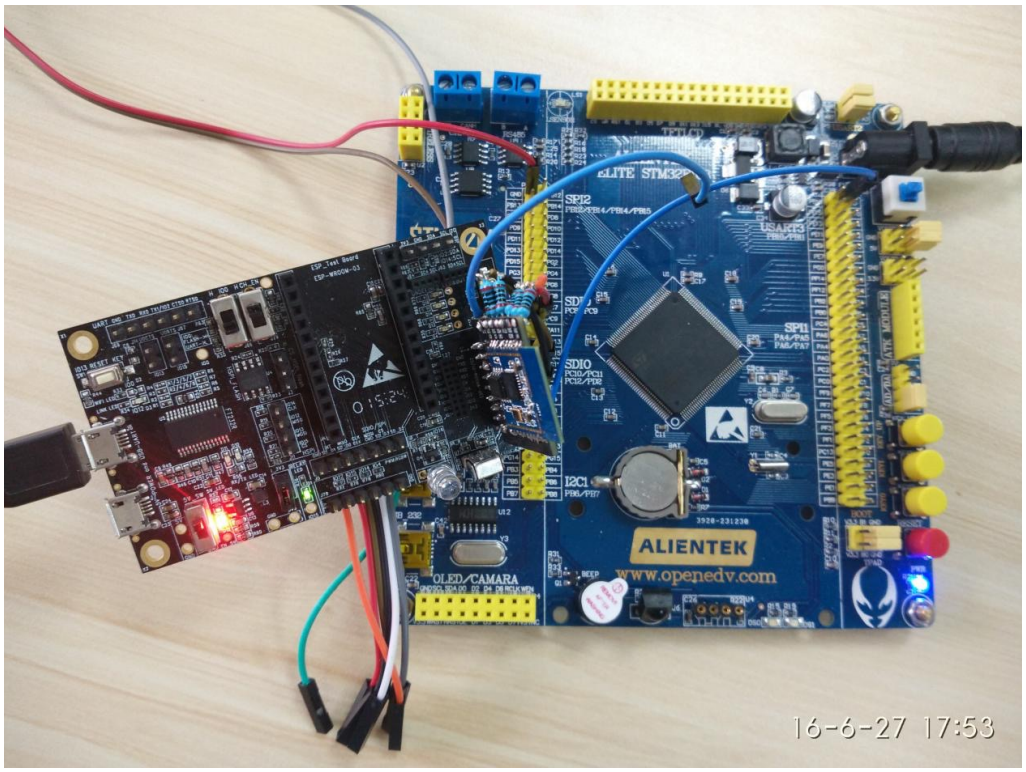

Figure 2-1: ALIENTEK ELITE STM32F103 Elite with ESP-WROOM-S2 Connected

3. Downloading Firmware to ESP-WROOM-S2

Download firmware to HSPI FLASH via ESP-WROOM-S2 Serial Interface.

1. Put IO0 to low level and CH_EN to high level.
2. Connect IO15 to GND with Dupont Lines. Power on S2 Module so that it can enter the download state.
3. Disconnect IO15 and GND.
4. Open ESP8266 DOWNLOAD TOOL and enter HSPIDownload interface. Choose DIO for SPI MODE, open COM port and the binaries, then click START to start download.
5. After download, restart MCU development board and ESP-WROOM-S2.

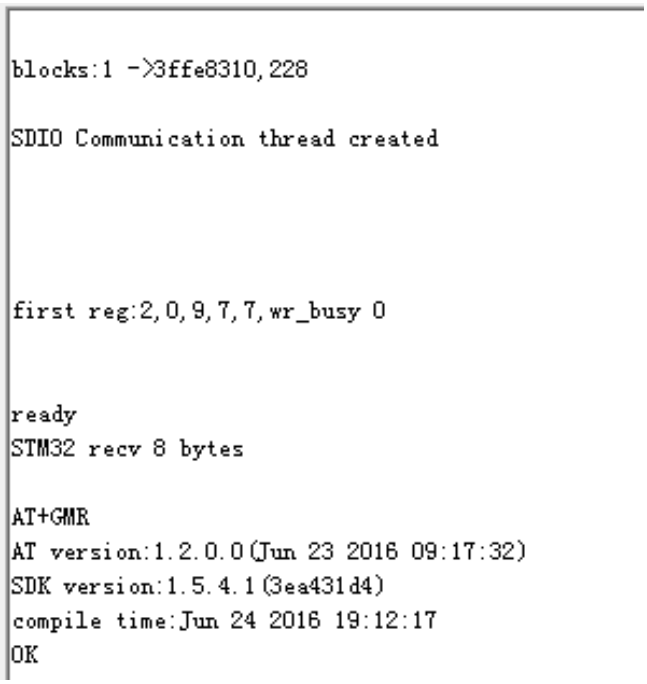

Figure 3-1: Log after Restart

> ⚠ ***Notice:***
>
> *Due to pin conflicts, AT instructions based on SDIO can not use UART flow control.*

HSPIDownload interface (subject to official release):

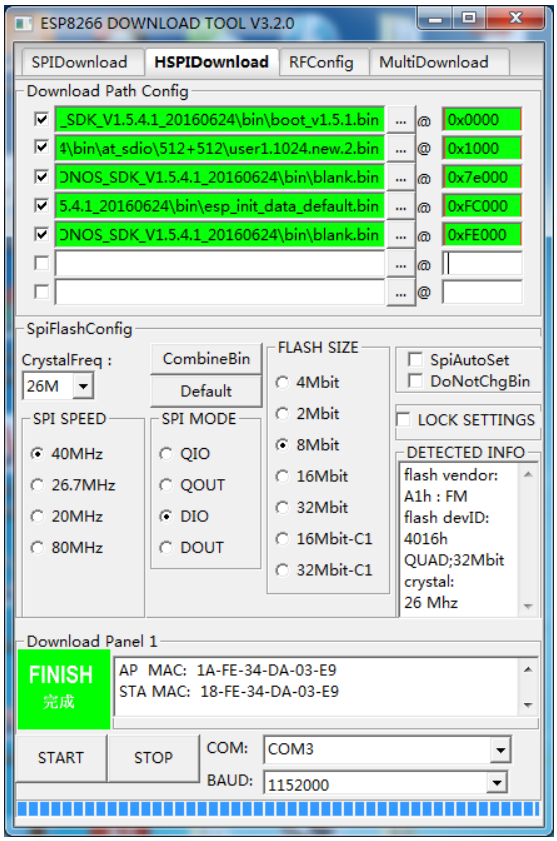

Figure 3-2: HSPIDownload Interface

4. Host MCU Configuration and Programming

ESP-WROOM-S2 will start only after it receives the boot file sent from MCU. Software can realise this function in the host MCU by downloading ***boot.bin*** to the MCU. Specific code information can be found in official demo.

In the demo, ***boot.bin*** has been converted to arrays and integrated into the STM32 main code. After the development board starts, STM32 development board will transfer ***boot.bin*** to ESP-WROOM-S2 via SDIO interface. Then ESP-WROOM-S2 can start using the boot binary. At this stage, send AT commands to the STM32 via serial interface. STM32 will then receive the commands and forward them to the ESP8266 via SDIO interface.

Note that the STM32 can be run on its own, with the ESP-WROOM-S2 acting as a companion for handling internet connectivity tasks and thus offloading the STM32. The STM32 can also be configured to program the ESP-WROOM-S2 on the fly, without the requirement of connecting/disconnecting jumpers manually.

> ⚠ ***Notice:***
>
> *The document specifies host MCU configuration and programming for interfacing with the ESP-WROOM-S2 module. The programming and configuration procedure of the host MCU development board itself is not covered here. Please consult the development board user manual for relevant information.*

5. Host - ESP Interaction

5.1. Recommended Interaction Sequence

The host MCU must interact in a specific sequence with the ESP-WROOM-S2 to enable it to boot properly and execute the firmware stored in the primary HSPI flash memory of the module. Here are the recommended steps of this sequence:

1. User program running on the MCU initialises SDIO host mode on the host MCU.
2. Host MCU detects Slave. If Slave is detected, host MCU will load ESP8266 ***boot.bin*** to ESP8266 RAM.
3. After ***boot.bin*** is loaded, host MCU will send instructions and make ESP8266 jump to boot.bin.
4. After receiving boot jump instruction, ESP8266 will jump to boot binary and read program from HSPI Flash and execute it.
5. When the program is running, if ESP8266 receives data sent via SDIO interface, it will generate a corresponding interrupt and post data to the application layer.
6. ESP8266 can send data to host MCU by loading data to SDIO buffer and then interrupting the host using the IRQ signal.
7. Host MCU generates interrupt IRQ and then reads interrupt information from ESP8266. Data will be read from ESP8266 if there is any.

5.2. Integrating *boot.bin* in Host MCU Program

Expressif supplies ***bin_to_hex_flash.py***, a scripting tool that can convert ***boot.bin*** into array format for integrating directly into host MCU source code. To run the script, simply place it under the same directory with ***boot_v1.5.1.bin*** and execute. The code is listed below:

```
def bin_to_hex(bin_data):
    hex_list = []
    for c in bin_data:
        hex_list.append(ord(c))
    return hex_list
    pass



def modify_flash_map(flash_map, hex_list):
    hex_list[3] = (hex_list[3] & 0x0F) + (int(flash_map, base=10) <<
4)
```

```
        return hex_list


def hex_to_array(hex_data):
    data = "unsigned char bin_array[] = {\r\n"
    i = 0
    for _hex in hex_data:
        i += 1
        data += "0x%02x, " % _hex
        if i % 16 == 0:
            data += "\r\n    "
    data = data[:-2]
    data += "};\r\n\r\n"
    data += "unsigned int bin_array_len = %d;\r\n" % len(hex_data)
    return data
    pass


def main():
    #f_name = raw_input("Please input file name:\r\n")
    f_name = "boot_v1.5.1.bin"
    flash_map = raw_input("Please select flash map:\r\n"
                          "0. 4M_256_256;\r\n"
                          "2. 8M_512_512;\r\n"
                          "3. 16M_512_512;\r\n"
                          "4. 32M_512_512;\r\n"
                          "5. 16M_1024_1024;\r\n"
                          "6. 32M_1024_1024;\r\n")
    with open(f_name, "rb") as f:
        bin_data = f.read()

    hex_data = bin_to_hex(bin_data)
    hex_data = modify_flash_map(flash_map, hex_data)
    file_content = hex_to_array(hex_data)
```

```
        with open(f_name + ".flash.bin", "wb+") as f:
            f.write("".join([chr(h) for h in hex_data]))

        with open(f_name + ".flash.c", "wb+") as f:
            f.write(file_content)
        pass


if __name__ == '__main__':
    main()
```

Espressif IOT Team

www.espressif.com