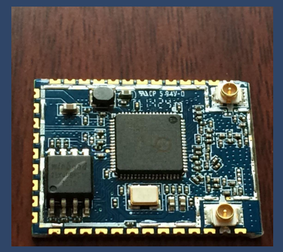
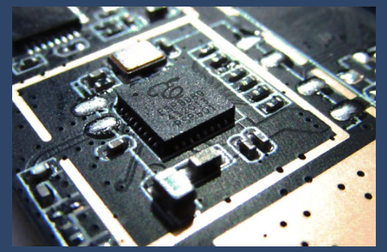
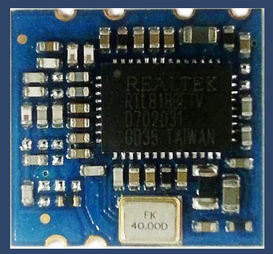
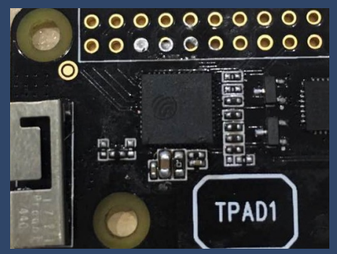
Something about ESP chips

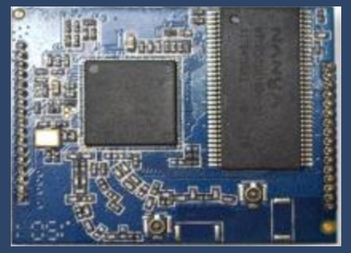
Find More Chip

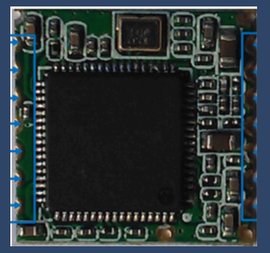












我们的优势有哪些?

ESP8XXX - Chip

ESP8089	ESP8266EX(ESP8266)	ESP8285	ESP8266SIP
			-
Android/Linux	IoT/MCU	IoT/MCU	IoT/MCU
UART/SDIO	UART/SPI...		
No WiFi stack	WiFi/TCP/IP		
SDIO download firmware	UART download		
No flash	SPI flash		
2.4G WiFi/ HT20			

Strapping:

- Strapping pads

U0TXD	MTDO(GPIO15)	GPIO2	GPIO0	
1	0	0	0	RESERVED
1	0	0	1	UART download
1	0	1	0	RESERVED
1	0	1	1	SPI boot
1	1	X	X	SDIO
0	X	X	X	TEST MODE

- Boot log:

```
ets Jan  8 2013,rst cause:2, boot mode:(1,6)
```

想一想

1. 芯片或者模块无法下载程序怎么办?
2. 为什么接上串口看不到打印?
3. 为什么接上串口，输出都是乱码?
4. 为什么我的程序速度那么慢?
5. 为什么 *throughput* 总是不能提高?
6. 为什么加了 *log* 之后，*TCP* 变得那么慢?

IO MUX

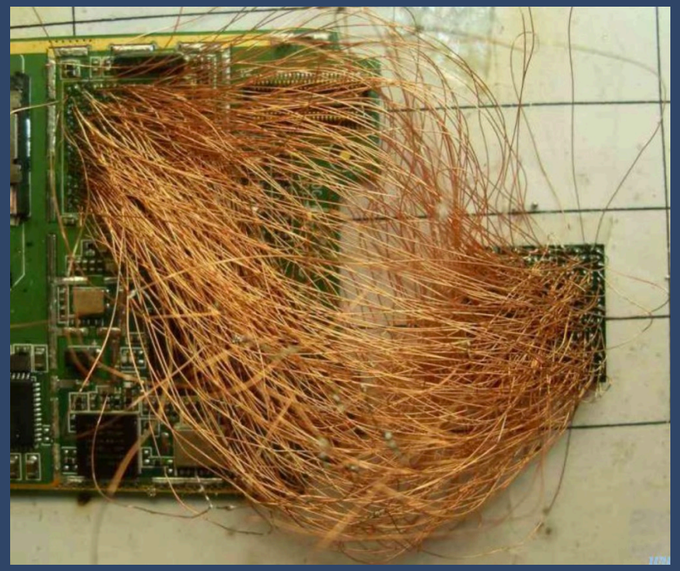
- `PIN_FUNC_SELECT(MUX_REG, FUNC_IDX)`
- pin 脚固定，容易产生冲突

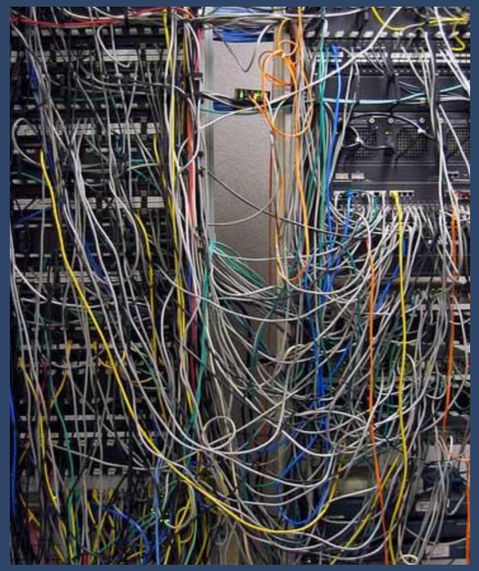
Function1	Type	Function2	Type	Function3	Type	Function4	Type	Function5	Type	At Reset	After Reset	Sleep
MTDI	I	I2SI_DATA	I/O/T	HSPIQ MISO	I/O/T	GPIO12	I/O/T	U0DTR	O	oe=0, wpu	wpu	oe=0
MTCK	I	I2SI_BCK	I/O/T	HSPID MOSI	I/O/T	GPIO13	I/O/T	U0CTS	I	oe=0, wpu	wpu	oe=0
MTMS	I	I2SI_WS	I/O/T	HSPICLK	I/O/T	GPIO14	I/O/T	U0DSR	I	oe=0, wpu	wpu	oe=0
MTDO	O/T	I2SO_BCK	I/O/T	HSPICS	I/O/T	GPIO15	I/O/T	U0RTS	O	oe=0, wpu	wpu	oe=0
U0RXD	I	I2SO_DATA	I/O/T		O	GPIO3	I/O/T	CLK_XTAL	O	oe=0, wpu	wpu	oe=0
U0TXD	O	SPICS1	I/O/T		O	GPIO1	I/O/T	CLK_RTC	O	oe=0, wpu	wpu	oe=0
SD_CLK	I	SPICLK	I/O/T		O	GPIO6	I/O/T	U1CTS	I	oe=0		oe=0
SD_DATA0	I/O/T	SPIQ	I/O/T		O	GPIO7	I/O/T	U1TXD	O	oe=0		oe=0
SD_DATA1	I/O/T	SPID	I/O/T		O	GPIO8	I/O/T	U1RXD	I	oe=0		oe=0
SD_DATA2	I/O/T	SPIHD	I/O/T		O	GPIO9	I/O/T	HSPIHD	I/O/T	oe=0		oe=0
SD_DATA3	I/O/T	SPIWP	I/O/T		O	GPIO10	I/O/T	HSPIWP	I/O/T	oe=0		oe=0
SD_CMD	I/O/T	SPICS0	I/O/T		O	GPIO11	I/O/T	U1RTS	O	oe=0		oe=0
GPIO0	I/O/T	SPICS2	I/O/T		O		I/O/T	CLK_OUT	O	oe=0, wpu	wpu	oe=0
GPIO2	I/O/T	I2SO_WS	I/O/T	U1TXD	O		I/O/T	U0TXD	O	oe=0, wpu	wpu	oe=0
GPIO4	I/O/T	CLK_XTAL	O							oe=0		oe=0
GPIO5	I/O/T	CLK_RTC	O							oe=0		oe=0
XPD_DCDC	O	RTC_GPIO0	I/O/T	EXT_WAKEUP	I	DEEPSLEEP	O	BT_XTAL_EN	I	oe=1,wpd	oe=1,wpd	oe=1

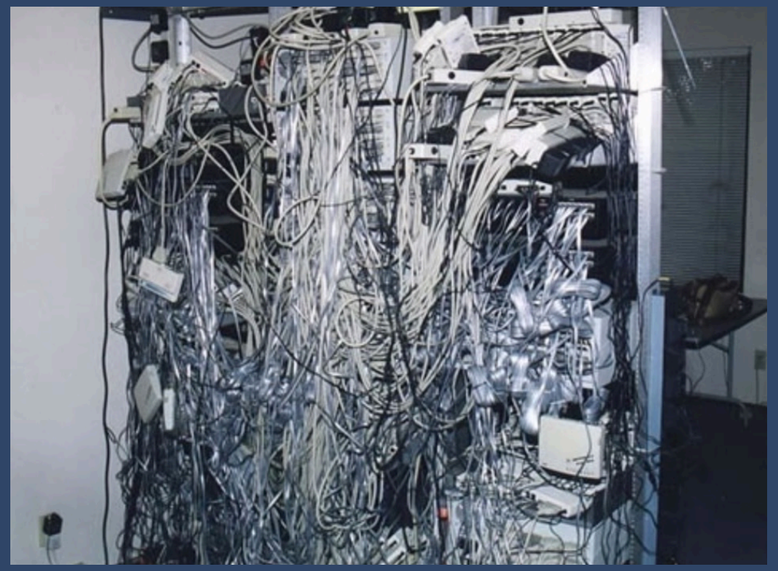
QFN32 封装

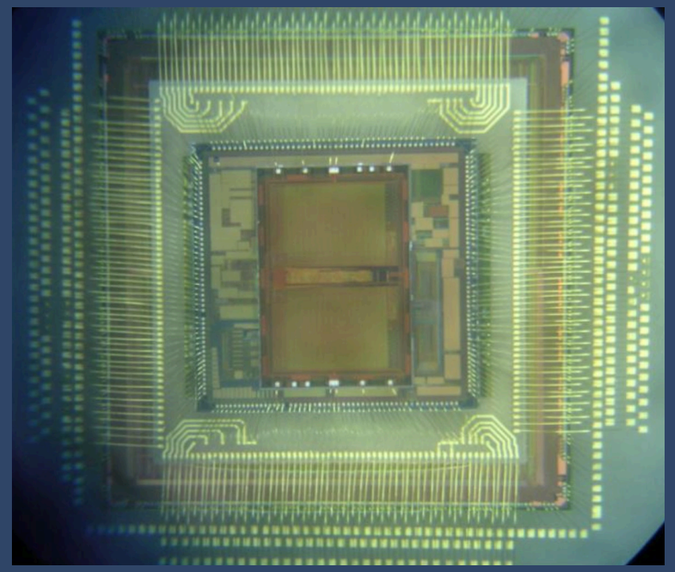
1. QFN 是一种无引脚封装，呈正方形或矩形;
2. 封装底部中央位置有一个大面积裸露焊盘用来导热;
3. 围绕大焊盘的封装外围四周有实现电气连结的导电焊盘。

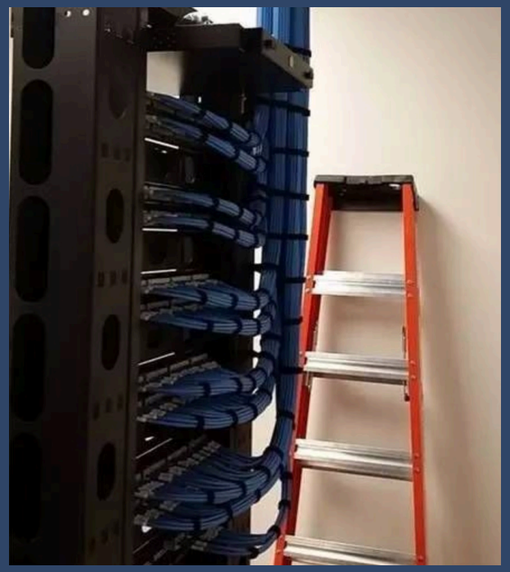
想一想：为什么需要封装？良好的封装和接口能带来什么好处？

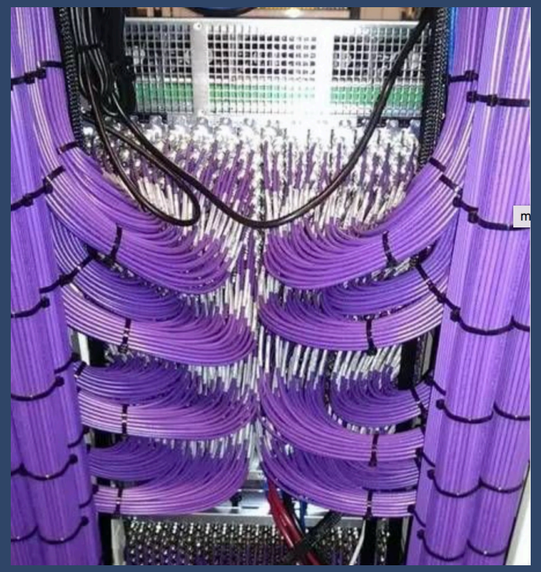












Flash Map

Offset	File	Usage
0x0	boot.bin	2nd bootloader
0x1000	user1.bin	user app
0x11000	user2.bin	user app(for OTA)
0x1fc000	init_data.bin	RF init params
0x1fd000~0x1fffff	blank.bin	Clear system params

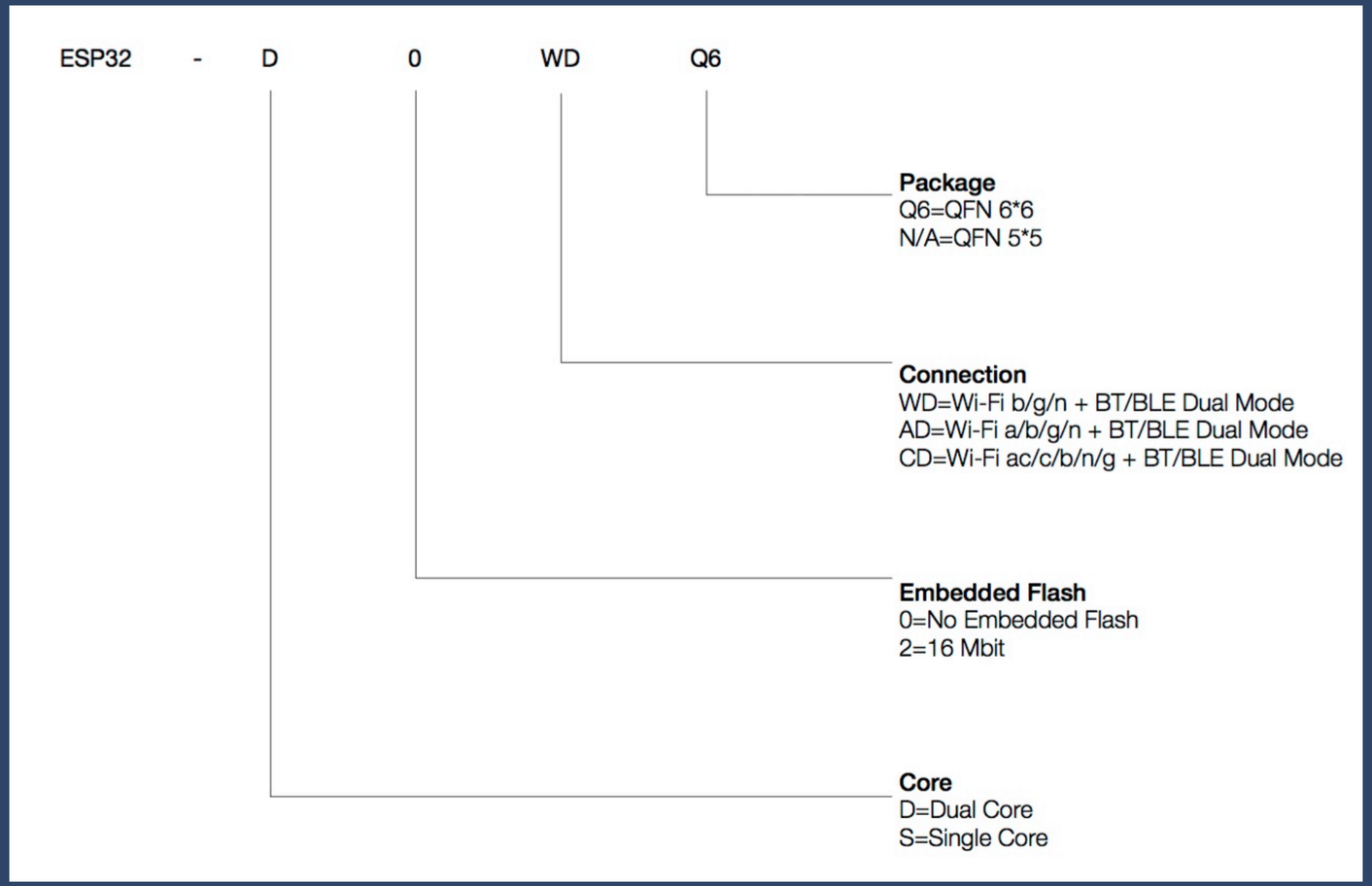
想一想

1. *What is 2nd bootloader?*
2. *What is 1st bootloader?*
3. *Why the minimum size if 0x1000?*
4. *What is OTA?*
5. *What can we do for RF params?*

ESP32xx series - Chip

ESP8689	ESP32D0WDQ6	ESP32S0WDQ5	ESP32D2WDQ5	ESP32SIP
		--	--	--
Android/Linux	IoT	--	--	--
No WiFi stack	WiFi/TCP/IP	--	--	--
WiFi + BT	WiFi + BT	--	--	--
UART/SDIO	UART/I2C/I2S/SPI/ SDIO/LEDC...	--	--	--
No flash	work with flash	--	--	--
2.4G WiFi/ HT20/ HT40	--	--	--	--

Part number



Order code

Ordering code	Core	Embedded flash	Connection	Package
ESP32-D0WDQ6	Dual core	No embedded flash	Wi-Fi b/g/n + BT/BLE Dual Mode	QFN 6*6
ESP32-D0WD	Dual core	No embedded flash	Wi-Fi b/g/n + BT/BLE Dual Mode	QFN 5*5
ESP32-D2WD	Dual core	16-Mbit embedded flash	Wi-Fi b/g/n + BT/BLE Dual Mode	QFN 5*5
ESP32-S0WD	Single core	No embedded flash	Wi-Fi b/g/n + BT/BLE Dual Mode	QFN 5*5

Strapping pin

Boot strap

GPIO0	GPIO2	GPIO4	MTDO	GPIO5	
1	X	X	X	X	SPI boot
0	0	X	X	X	Download boot(UART0+UART1+SDIO_Slave)
0	1	0	X	X	HSPI boot
0	1	1	0	1	SDIO slave download mode

SDIO mode

MTDO	GPIO5	采样	输出
0	0	下降沿	下降沿
0	1	下降沿	上升沿
1	0	上升沿	下降沿
1	1	上升沿	上升沿

VSDIO_SEL

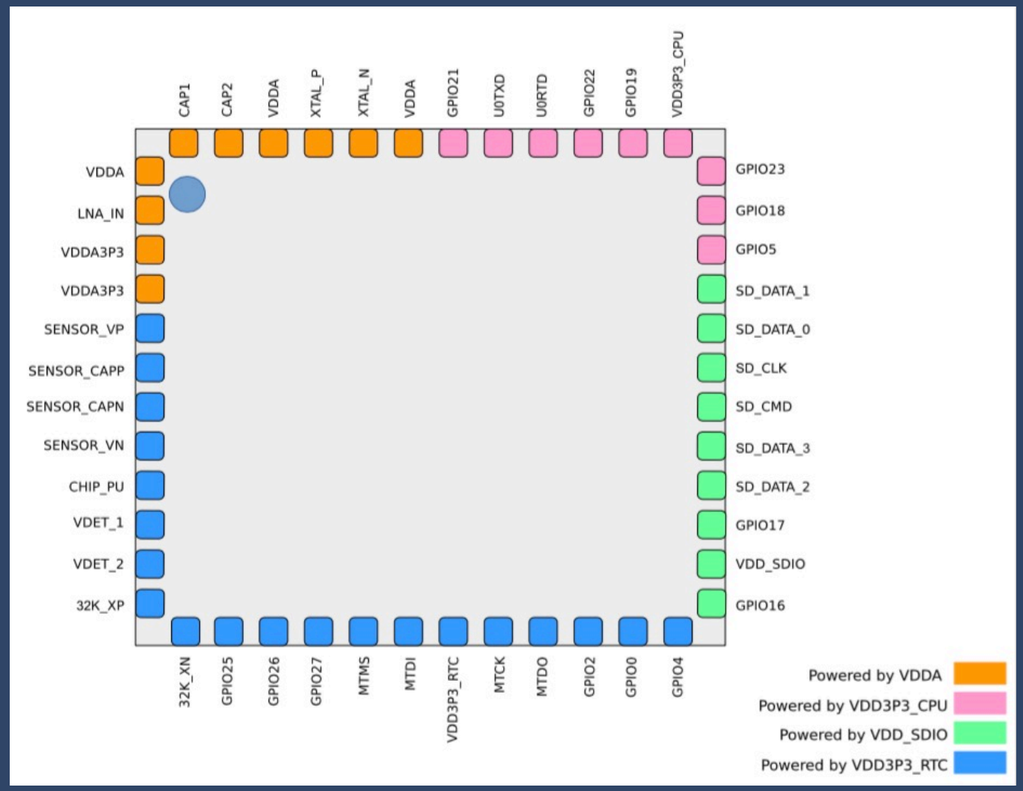
MTDI	VSDIO
0	3.3V
1	1.8V

Boot log

```
ets Jun  8 2016 00:22:57
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
```

QFN48 6*6 mm 封装

48 pin(12 * 4)



IO MUX

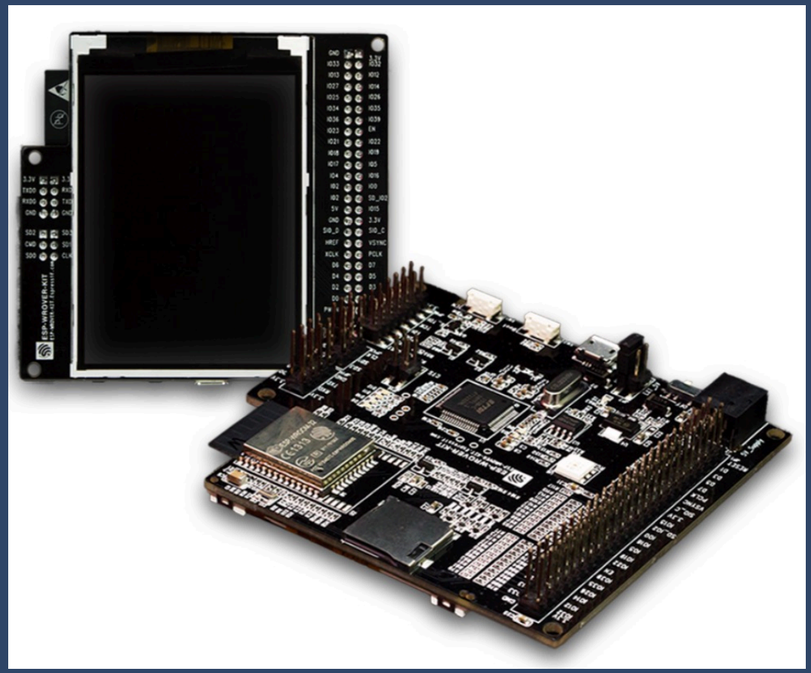
Pin No.	Power Supply Pin	Analog Pin	Digital Pin	Power Domain	Analog Function1	Analog Function2	Analog Function3	RTC Function1	RTC Function2	Function1	Type	Function2	Type	Function3	Type	Function4	Type	Function5	Type	Function6	Type	Drive Strength (2'd2: 20 mA)	At Reset	After Reset
1	VDDA			VANA in																				
2		LNA_IN		VANA in																				
3	VDD3P3			VANA in																				
4	VDD3P3			VANA in																				
5		SENSOR_VP		VRTC	ADC_H	ADC1_CH0		RTC_GPIO0		GPIO36	I			GPIO36	I									ie=0
6		SENSOR_CAPP		VRTC	ADC_H	ADC1_CH1		RTC_GPIO1		GPIO37	I			GPIO37	I									ie=0
7		SENSOR_CAPN		VRTC	ADC_H	ADC1_CH2		RTC_GPIO2		GPIO38	I			GPIO38	I									ie=0
8		SENSOR_VN		VRTC	ADC_H	ADC1_CH3		RTC_GPIO3		GPIO39	I			GPIO39	I									ie=0
9		CHIP_PU		VRTC																				
10		VDET_1		VRTC		ADC1_CH6		RTC_GPIO4		GPIO34	I			GPIO34	I									ie=0
11		VDET_2		VRTC		ADC1_CH7		RTC_GPIO5		GPIO35	I			GPIO35	I									ie=0
12		32K_XP		VRTC	XTAL_32K_P	ADC1_CH4	TOUCH9	RTC_GPIO9		GPIO32	I/O/T			GPIO32	I/O/T							2'd2		ie=0
13		32K_XN		VRTC	XTAL_32K_N	ADC1_CH5	TOUCH8	RTC_GPIO8		GPIO33	I/O/T			GPIO33	I/O/T							2'd2		ie=0
14			GPIO25	VRTC	DAC_1	ADC2_CH8		RTC_GPIO6		GPIO25	I/O/T			GPIO25	I/O/T					EMAC_RXD0	I	2'd2		ie=0
15			GPIO26	VRTC	DAC_2	ADC2_CH9		RTC_GPIO7		GPIO26	I/O/T			GPIO26	I/O/T					EMAC_RXD1	I	2'd2		ie=0
16			GPIO27	VRTC		ADC2_CH7	TOUCH7	RTC_GPIO17		GPIO27	I/O/T			GPIO27	I/O/T					EMAC_RX_DV	I	2'd2		ie=1
17			MTMS	VRTC		ADC2_CH6	TOUCH6	RTC_GPIO16		MTMS	I0	HSPICLK	I/O/T	GPIO14	I/O/T	HS2_CLK	O	SD_CLK	I0	EMAC_TXD2	O	2'd2	wpu, ie=1	wpu, ie=1
18			MTDI	VRTC		ADC2_CH5	TOUCH5	RTC_GPIO15		MTDI	I1	HSPIQ	I/O/T	GPIO12	I/O/T	HS2_DATA2	I1/O/T	SD_DATA2	I1/O/T	EMAC_TXD3	O	2'd2	wpd, ie=1	wpd, ie=1
19	VDD3P3_RTC			VRTC supply in																				
20			MTCK	VRTC		ADC2_CH4	TOUCH4	RTC_GPIO14		MTCK	I1	HSPID	I/O/T	GPIO13	I/O/T	HS2_DATA3	I1/O/T	SD_DATA3	I1/O/T	EMAC_RX_ER	I	2'd2	wpu, ie=1	wpu, ie=1
21			MTDO	VRTC		ADC2_CH3	TOUCH3	RTC_GPIO13	I2C_SDA	MTDO	O/T	HSPICS0	I/O/T	GPIO15	I/O/T	HS2_CMD	I1/O/T	SD_CMD	I1/O/T	EMAC_RXD3	I	2'd2	wpu, ie=1	wpu, ie=1
22			GPIO2	VRTC		ADC2_CH2	TOUCH2	RTC_GPIO12	I2C_SCL	GPIO2	I/O/T	HSPIWP	I/O/T	GPIO2	I/O/T	HS2_DATA0	I1/O/T	SD_DATA0	I1/O/T			2'd2	wpd, ie=1	wpd, ie=1
23			GPIO0	VRTC		ADC2_CH1	TOUCH1	RTC_GPIO11	I2C_SDA	GPIO0	I/O/T	CLK_OUT1	O	GPIO0	I/O/T					EMAC_TX_CLK	I	2'd2	wpu, ie=1	wpu, ie=1
24			GPIO4	VRTC		ADC2_CH0	TOUCH0	RTC_GPIO10	I2C_SCL	GPIO4	I/O/T	HSPIHD	I/O/T	GPIO4	I/O/T	HS2_DATA1	I1/O/T	SD_DATA1	I1/O/T	EMAC_TX_ER	O	2'd2	wpd, ie=1	wpd, ie=1
25			GPIO16	VSDIO						GPIO16	I/O/T			GPIO16	I/O/T	HS1_DATA4	I1/O/T	U2RXD	I1	EMAC_CLK_OUT	O	2'd2		ie=1
26	VDD_SDIO			VSDIO supply out/in																				
27			GPIO17	VSDIO						GPIO17	I/O/T			GPIO17	I/O/T	HS1_DATA5	I1/O/T	U2TXD	O	EMAC_CLK_OUT_180	O	2'd2		ie=1
28			SD_DATA_2	VSDIO						SD_DATA2	I1/O/T	SPIHD	I/O/T	GPIO9	I/O/T	HS1_DATA2	I1/O/T	U1RXD	I1			2'd2	wpu, ie=1	wpu, ie=1
29			SD_DATA_3	VSDIO						SD_DATA3	I0/O/T	SPIWP	I/O/T	GPIO10	I/O/T	HS1_DATA3	I1/O/T	U1TXD	O			2'd2	wpu, ie=1	wpu, ie=1
30			SD_CMD	VSDIO						SD_CMD	I1/O/T	SPICS0	I/O/T	GPIO11	I/O/T	HS1_CMD	I1/O/T	U1RTS	O			2'd2	wpu, ie=1	wpu, ie=1
31			SD_CLK	VSDIO						SD_CLK	I0	SPICLK	I/O/T	GPIO6	I/O/T	HS1_CLK	O	U1CTS	I1			2'd2	wpu, ie=1	wpu, ie=1
32			SD_DATA_0	VSDIO						SD_DATA0	I1/O/T	SPIQ	I/O/T	GPIO7	I/O/T	HS1_DATA0	I1/O/T	U2RTS	O			2'd2	wpu, ie=1	wpu, ie=1
33			SD_DATA_1	VSDIO						SD_DATA1	I1/O/T	SPID	I/O/T	GPIO8	I/O/T	HS1_DATA1	I1/O/T	U2CTS	I1			2'd2	wpu, ie=1	wpu, ie=1
34			GPIO5	VIO						GPIO5	I/O/T	VSPICS0	I/O/T	GPIO5	I/O/T	HS1_DATA6	I1/O/T			EMAC_RX_CLK	I	2'd2	wpu, ie=1	wpu, ie=1
35			GPIO18	VIO						GPIO18	I/O/T	VSPICLK	I/O/T	GPIO18	I/O/T	HS1_DATA7	I1/O/T					2'd2		ie=1
36			GPIO23	VIO						GPIO23	I/O/T	VSPID	I/O/T	GPIO23	I/O/T	HS1_STROBE	I0					2'd2		ie=1
37	VDD3P3_CPU			VIO supply in																				
38			GPIO19	VIO						GPIO19	I/O/T	VSPIQ	I/O/T	GPIO19	I/O/T	U0CTS	I1			EMAC_TXD0	O	2'd2		ie=1
39			GPIO22	VIO						GPIO22	I/O/T	VSPIWP	I/O/T	GPIO22	I/O/T	U0RTS	O			EMAC_TXD1	O	2'd2		ie=1
40			U0RXD	VIO						U0RXD	I1	CLK_OUT2	O	GPIO3	I/O/T							2'd2	wpu, ie=1	wpu, ie=1
41			U0TXD	VIO						U0TXD	O	CLK_OUT3	O	GPIO1	I/O/T					EMAC_RXD2	I	2'd2	wpu, ie=1	wpu, ie=1
42			GPIO21	VIO						GPIO21	I/O/T	VSPIHD	I/O/T	GPIO21	I/O/T					EMAC_TX_EN	O	2'd2		ie=1
43	VDDA			VANA in																				
44		XTAL_N		VANA																				
45		XTAL_P		VANA																				
46	VDDA			VANA																				
47		CAP2		VANA																				
48		CAP1		VANA																				
Total Number	8	14	26																					

Note:
Please see Table: Notes on ESP32 Pin Lists. (请参考表：管脚清单说明。)

ESP32 - Development Board

ESP-WROVER-Kit

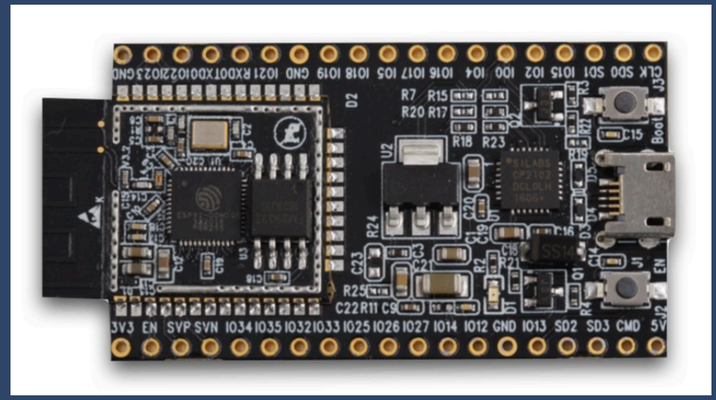
- 基于 ESP32 的开发板
- 支持 LCD 显示、MicroSD 卡、模组 I/O 扩展等功能
- 板载 FT2232HL 芯片，用户可通过 USB 线连接 PC 与开发板，使用 JTAG 功能对 ESP32 芯片进行调试，进而缩短开发周期，降低开发成本



ESP32 - Development Board

ESP-DevKit

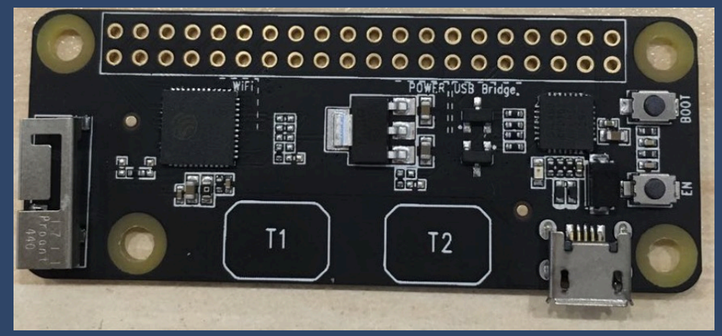
- ESP32 的核心开发板
- 尺寸 27.9 mm × 48.2 mm
- 主要 I/O 都被引出至两侧排针，用户可以根需求配置连接至对应的外设



ESP32 - Development Board

ESP32-PICO-KIT

采用 ESP32-PICO-D4(D2) 芯片，flash 及晶振全部集成至芯片内部。

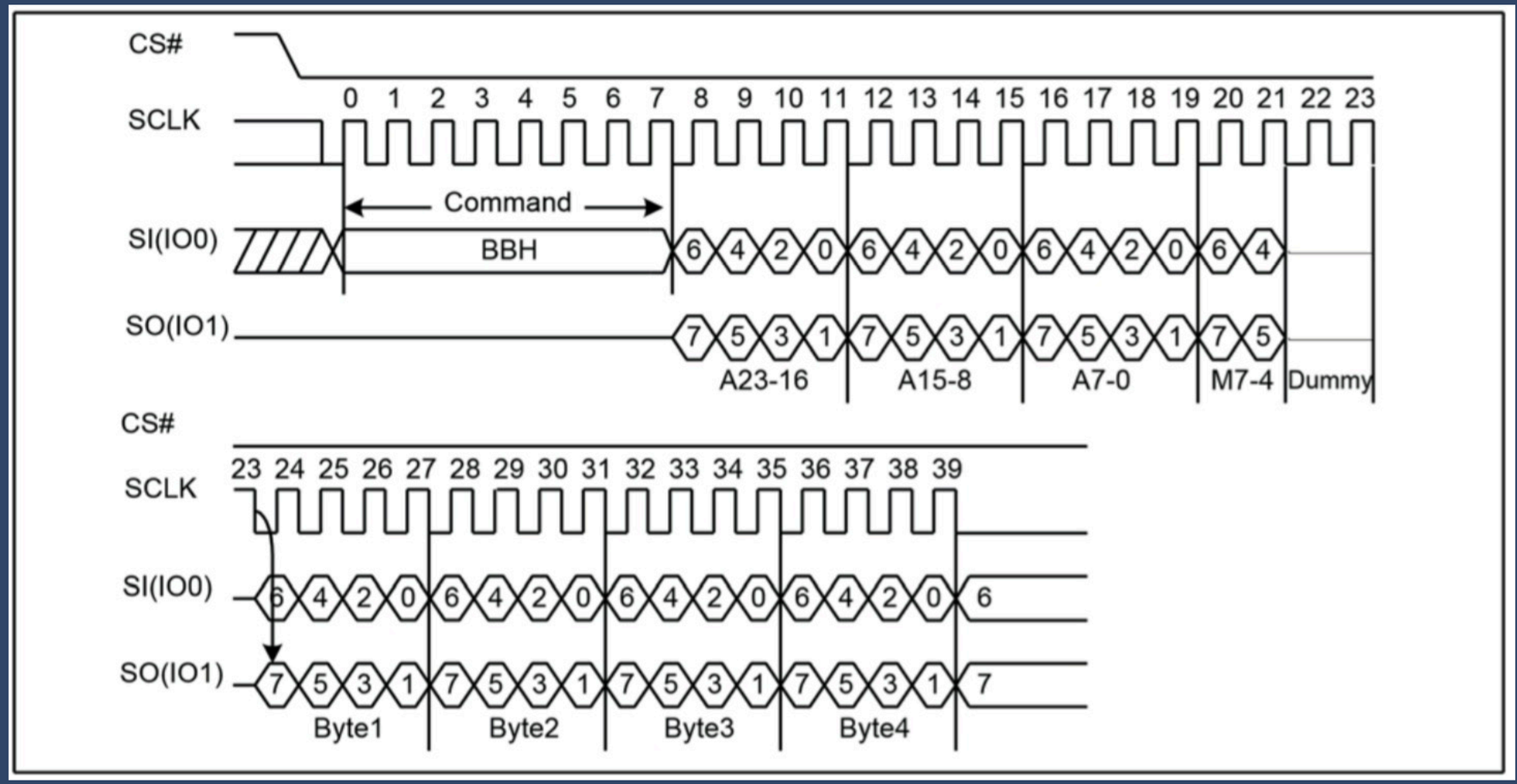


Flash

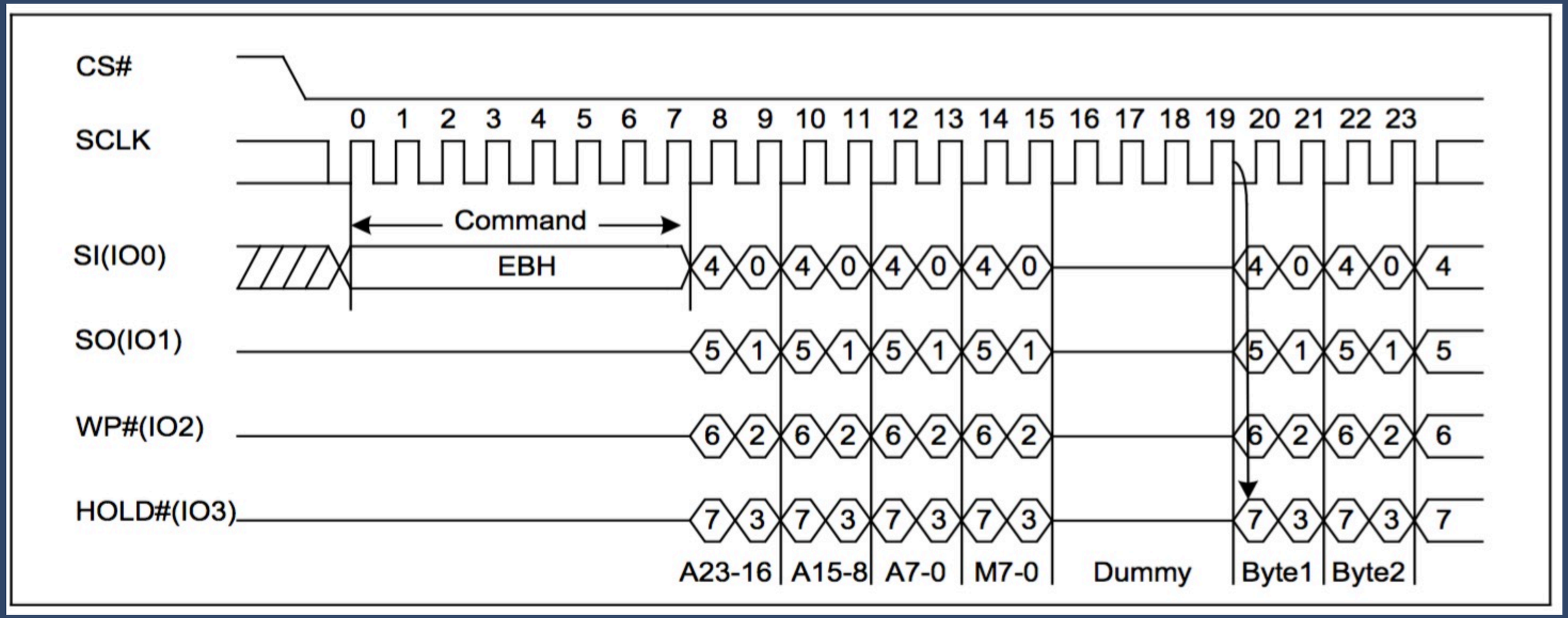
Why do we have to use flash?

- Instruction/data cache
- Save instructions
- Save system/user data
- Cheap
- fast enough for Wi-Fi

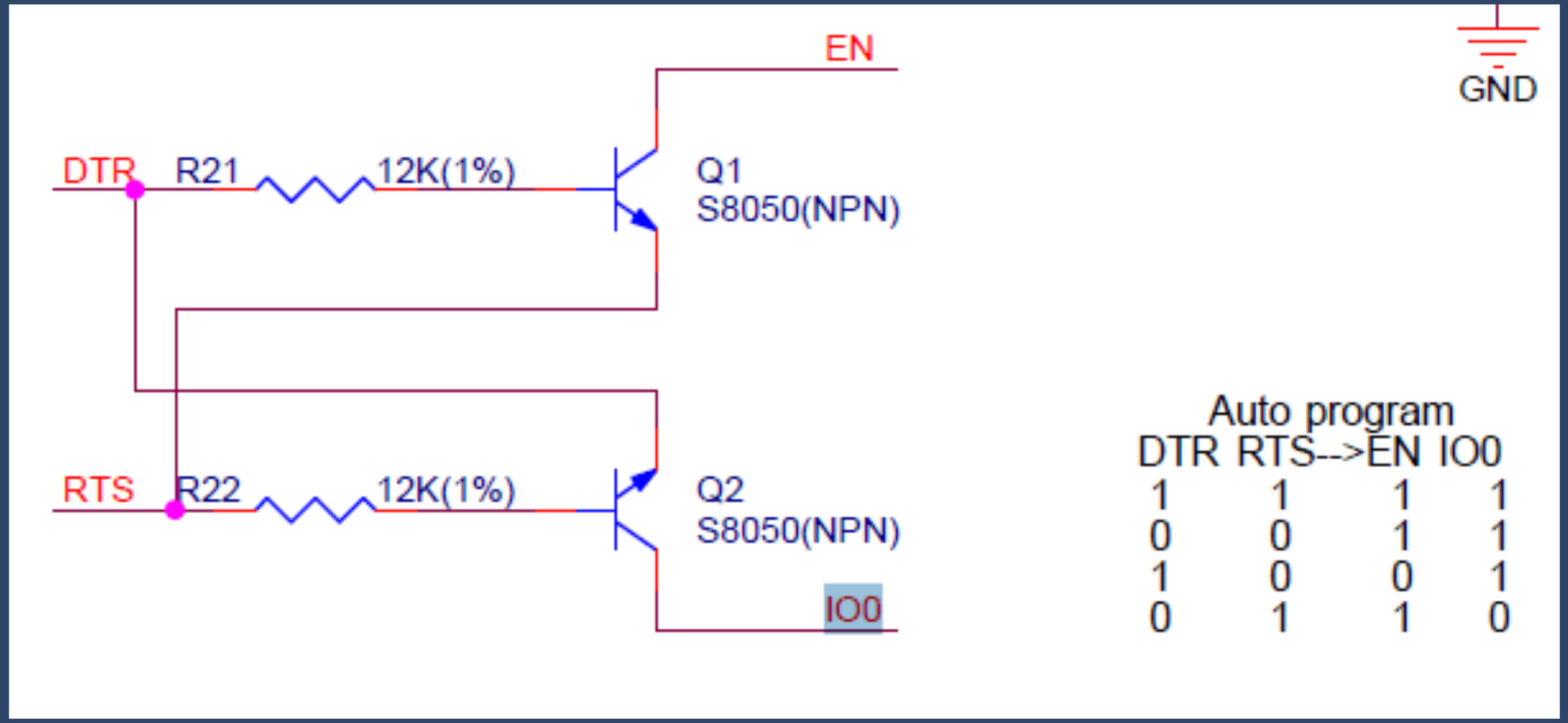
Flash - Dual mode



Flash - Quad mode



Flash - Auto-reboot and reburn



Auto program

DTR	RTS-->EN	IO0	
1	1	1	1
0	0	1	1
1	0	0	1
0	1	1	0

ESPTOOL - Installation

```
pip install esptool
```

ESPTOOL - Function

```
esptool.py -h
usage: esptool [-h] [--chip {auto,esp8266,esp32}] [--port PORT] [--baud BAUD]
               [--before {default_reset,no_reset}]
               [--after {hard_reset,soft_reset,no_reset}] [--no-stub]
               {load_ram,dump_mem,read_mem,write_mem,write_flash,run,image_info,make_image,elf2image,read_mac,chip_id,flash_id,read_flash_status,write_flash_status,read_flash,verify_flash,erase_flash,erase_region,version}
               ...
esptool.py v2.0.1 - ESP8266 ROM Bootloader Utility
positional arguments:
{load_ram,dump_mem,read_mem,write_mem,write_flash,run,image_info,make_image,elf2image,read_mac,chip_id,flash_id,read_flash_status,write_flash_status,read_flash,verify_flash,erase_flash,erase_region,version}
                      Run esptool {command} -h for additional help
    load_ram          Download an image to RAM and execute
    dump_mem          Dump arbitrary memory to disk
    read_mem          Read arbitrary memory location
    write_mem         Read-modify-write to arbitrary memory location
    write_flash       Write a binary blob to flash
    run               Run application code in flash
    image_info        Dump headers from an application image
    make_image        Create an application image from binary files
    elf2image         Create an application image from ELF file
    read_mac          Read MAC address from OTP ROM
    chip_id           Read Chip ID from OTP ROM
    flash_id          Read SPI flash manufacturer and device ID
    read_flash_status Read SPI flash status register
    write_flash_status Write SPI flash status register
    read_flash        Read SPI flash content
    verify_flash      Verify a binary blob against flash
    erase_flash       Perform Chip Erase on SPI flash
    erase_region      Erase a region of the flash
    version           Print esptool version
optional arguments:
  -h, --help          show this help message and exit
  --chip {auto,esp8266,esp32}, -c {auto,esp8266,esp32}
                      Target chip type
  --port PORT, -p PORT  Serial port device
  --baud BAUD, -b BAUD  Serial port baud rate used when flashing/reading
  --before {default_reset,no_reset}
                      What to do before connecting to the chip
  --after {hard_reset,soft_reset,no_reset}, -a {hard_reset,soft_reset,no_reset}
                      What to do after esptool.py is finished
  --no-stub           Disable launching the flasher stub, only talk to ROM
                      bootloader. Some features will not be available.
```

ESPTOOL - Burn flash

```
esptool.py write_flash -h
usage: esptool write_flash [-h] [--flash_freq {keep,40m,26m,20m,80m}]
                           [--flash_mode {keep,qio,qout,dio,dout}]
                           [--flash_size FLASH_SIZE]
                           [--spi-connection SPI_CONNECTION] [--no-progress]
                           [--verify] [--compress | --no-compress]
                           <address> <filename> [<address> <filename> ...]
positional arguments:
  <address> <filename>  Address followed by binary filename, separated by
                        space
optional arguments:
  -h, --help            show this help message and exit
  --flash_freq {keep,40m,26m,20m,80m}, -ff {keep,40m,26m,20m,80m}
                        SPI Flash frequency
  --flash_mode {keep,qio,qout,dio,dout}, -fm {keep,qio,qout,dio,dout}
                        SPI Flash mode
  --flash_size FLASH_SIZE, -fs FLASH_SIZE
                        SPI Flash size in MegaBytes (1MB, 2MB, 4MB, 8MB, 16M)
                        plus ESP8266-only (256KB, 512KB, 2MB-c1, 4MB-c1)
  --spi-connection SPI_CONNECTION, -sc SPI_CONNECTION
                        ESP32-only argument. Override default SPI Flash
                        connection. Value can be SPI, HSPI or a comma-
                        separated list of 5 I/O numbers to use for SPI flash
                        (CLK,Q,D,HD,CS).
  --no-progress, -p     Suppress progress output
  --verify              Verify just-written data on flash (mostly superfluous,
                        data is read back during flashing)
  --compress, -z        Compress data in transfer (default unless --no-stub is
                        specified)
  --no-compress, -u     Disable data compression during transfer (default if
                        --no-stub is specified)
```

eFuse - Introduction

ESP32 中有多个 eFuse，其中存储着系统参数。作为一种非易失性存储单位，eFuse 的 bit 一旦被烧写为 1，不能恢复为 0。eFuse 控制器按照软件操作完成对 eFuse 中各个系统参数中的各个 bit 的烧写。这些系统参数有些可以通过 eFuse 控制器被软件读取，有些直接由硬件模块使用。

Block	Component	Bit
BLOCK0	system parameters	256 bit
BLOCK1	flash encryption key	256 bit
BLOCK2	secure boot key	256 bit
BLOCK3	customer AES KEY	256 bit

eFuse - ESP eFuse

```
espefuse.py -h
usage: espefuse [-h] [--port PORT] [--before {default_reset,no_reset,esp32r1}]
                [--do-not-confirm]
{dump,summary,burn_efuse,read_protect_efuse,write_protect_efuse,burn_key,set_flash_voltage}
                ...
espefuse.py v2.0.1 - ESP32 efuse get/set tool
positional arguments:
{dump,summary,burn_efuse,read_protect_efuse,write_protect_efuse,burn_key,set_flash_voltage}
                      Run espefuse.py {command} -h for additional help
    dump              Dump raw hex values of all efuses
    summary           Print human-readable summary of efuse values
    burn_efuse        Burn the efuse with the specified name
    read_protect_efuse  Disable readback for the efuse with the specified name
    write_protect_efuse
                      Disable writing to the efuse with the specified name
    burn_key          Burn a 256-bit AES key to EFUSE BLK1,BLK2 or BLK3
                      (flash_encryption, secure_boot).
    set_flash_voltage   Permanently set the internal flash voltage regulator
                      to either 1.8V, 3.3V or OFF. This means GPIO12 can be
                      high or low at reset without changing the flash
                      voltage.
optional arguments:
  -h, --help          show this help message and exit
  --port PORT, -p PORT  Serial port device
  --before {default_reset,no_reset,esp32r1}
                      What to do before connecting to the chip
  --do-not-confirm    Do not pause for confirmation before permanently
                      writing efuses. Use with caution.
```

eFuse - Read eFuse Value

```
espefuse.py -p /dev/cu.SLAB_USBtoUART summary
espefuse.py v2.0.1
Connecting....
Security fuses:
FLASH_CRYPT_CNT         Flash encryption mode counter                    = 0 R/W (0x0)
FLASH_CRYPT_CONFIG      Flash encryption config (key tweak bits)         = 0 R/W (0x0)
CONSOLE_DEBUG_DISABLE   Disable ROM BASIC interpreter fallback           = 1 R/W (0x1)
ABS_DONE_0              secure boot enabled for bootloader               = 0 R/W (0x0)
ABS_DONE_1              secure boot abstract 1 locked                    = 0 R/W (0x0)
JTAG_DISABLE            Disable JTAG                                     = 0 R/W (0x0)
DISABLE_DL_ENCRYPT      Disable flash encryption in UART bootloader      = 0 R/W (0x0)
DISABLE_DL_DECRYPT      Disable flash decryption in UART bootloader      = 0 R/W (0x0)
DISABLE_DL_CACHE        Disable flash cache in UART bootloader           = 0 R/W (0x0)
BLK1                    Flash encryption key
  = 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 R/W
BLK2                    Secure boot key
  = 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 R/W
BLK3                    Variable Block 3
  = 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 R/W
Efuse fuses:
WR_DIS                  Efuse write disable mask                         = 0 R/W (0x0)
RD_DIS                  Efuse read disablemask                           = 0 R/W (0x0)
CODING_SCHEME           Efuse variable block length scheme               = 0 R/W (0x0)
KEY_STATUS              Usage of efuse block 3 (reserved)                = 0 R/W (0x0)
Config fuses:
XPD_SDIO_FORCE          Ignore MTDI pin (GPIO12) for VDD_SDIO on reset   = 0 R/W (0x0)
XPD_SDIO_REG            If XPD_SDIO_FORCE, enable VDD_SDIO reg on reset  = 0 R/W (0x0)
XPD_SDIO_TIEH           If XPD_SDIO_FORCE & XPD_SDIO_REG, 1=3.3V 0=1.8V  = 0 R/W (0x0)
SPI_PAD_CONFIG_CLK      Override SD_CLK pad (GPIO6/SPICLK)               = 0 R/W (0x0)
SPI_PAD_CONFIG_Q        Override SD_DATA_0 pad (GPIO7/SPIQ)              = 0 R/W (0x0)
SPI_PAD_CONFIG_D        Override SD_DATA_1 pad (GPIO8/SPID)              = 0 R/W (0x0)
SPI_PAD_CONFIG_HD       Override SD_DATA_2 pad (GPIO9/SPIHD)             = 0 R/W (0x0)
SPI_PAD_CONFIG_CS0      Override SD_CMD pad (GPIO11/SPICS0)              = 0 R/W (0x0)
DISABLE_SDIO_HOST       Disable SDIO host                                = 0 R/W (0x0)
Identity fuses:
MAC                     MAC Address                                      = 30:ae:a4:0e:ab:e0 R/W
CHIP_VERSION            Chip version                                     = 8 R/W (0x8)
CHIP_PACKAGE            Chip package identifier                          = 0 R/W (0x0)
Flash voltage (VDD_SDIO) determined by GPIO12 on reset (High for 1.8V, Low/NC for 3.3V).
```

TALK IS CHEAP, SHOW ME THE CODE !

LIFE IS SHORT, USE PYTHON!

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one - and preferably only one - obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!

优美胜于丑陋（Python 以编写优美的代码为目标）
明了胜于晦涩（优美的代码应当是明了的，命名规范，风格相似）
简洁胜于复杂（优美的代码应当是简洁的，不要有复杂的内部实现）
复杂胜于凌乱（如果复杂不可避免，那代码间也不能有难懂的关系，要保持接口简洁）
扁平胜于嵌套（优美的代码应当是扁平的，不能有太多的嵌套）
间隔胜于紧凑（优美的代码有适当的间隔，不要奢望一行代码解决问题）
可读性很重要（优美的代码是可读的）
即便假借特例的实用性之名，也不可违背这些规则（这些规则至高无上）
不要包容所有错误，除非你确定需要这样做（精准地捕获异常，不写 except:pass 风格的代码）
当存在多种可能，不要尝试去猜测
而是尽量找一种，最好是唯一一种明显的解决方案（如果不确定，就用穷举法）
虽然这并不容易，因为你不是 Python 之父（这里的 Dutch 是指 Guido ）
做也许好过不做，但不假思索就动手还不如不做（动手之前要细思量）
如果你无法向人描述你的方案，那肯定不是一个好方案；反之亦然（方案测评标准）
命名空间是一种绝妙的理念，我们应当多加利用（倡导与号召）

Q&A

Thank you!