Part 2
IoT Architecture

Fabrice MULLER

Fabrice.Muller@univ-cotedazur.fr

2021 - 2022

Part 2 - IoT Architecture

Lab 1 : ESP32 Architecture

- Main characteristics
- Real-Time Clock

Lab 2 : ESP32 Memory

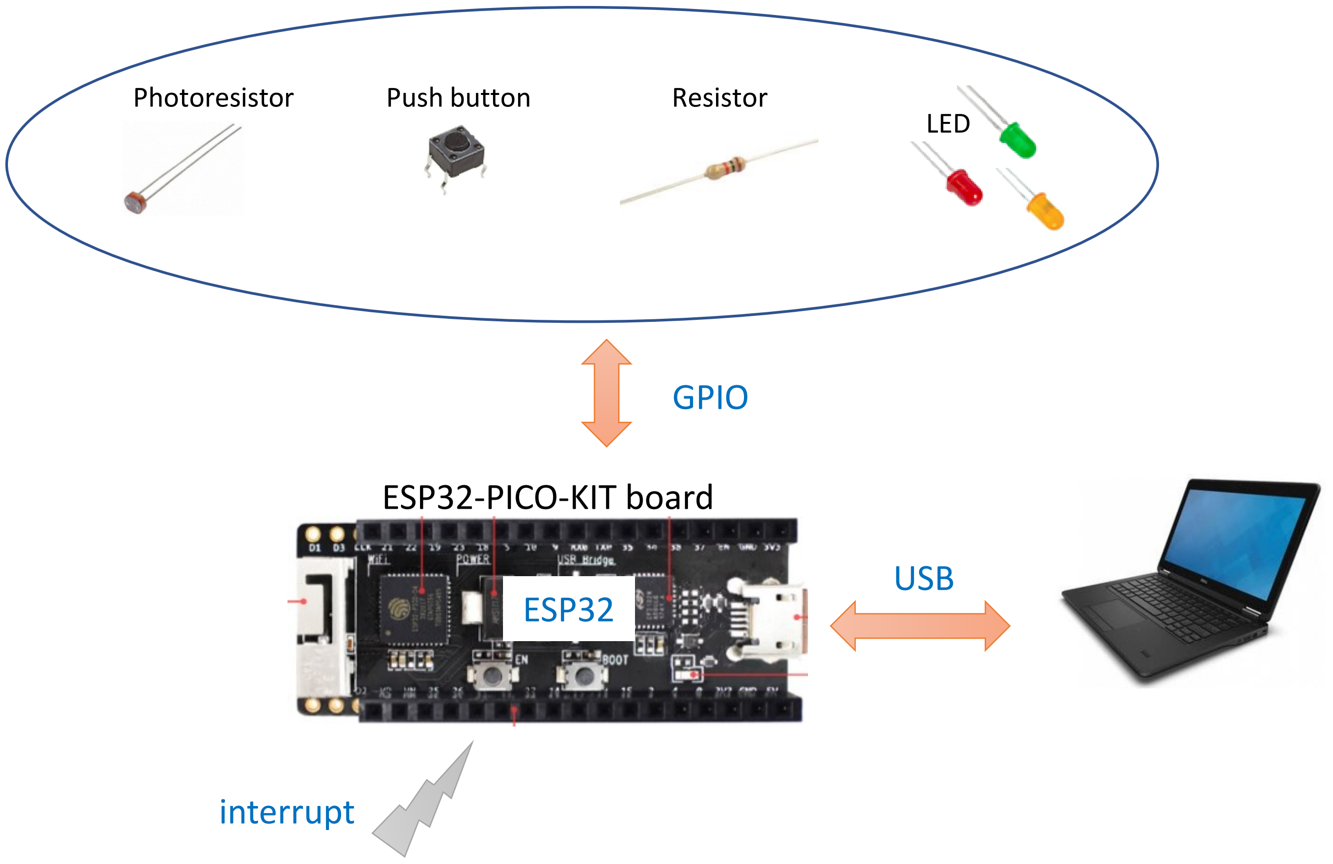
- IRAM, DRAM
- Non-Volatile SRAM (NVS)
- SPI flash file system (SPIFFS)

Lab 3 : GPIO

- Configuration I/O (push button, Led)
- Advanced GPIO configurations

Lab 4 : Interrupt

- GPIO interrupt
- Push button example



ESP32

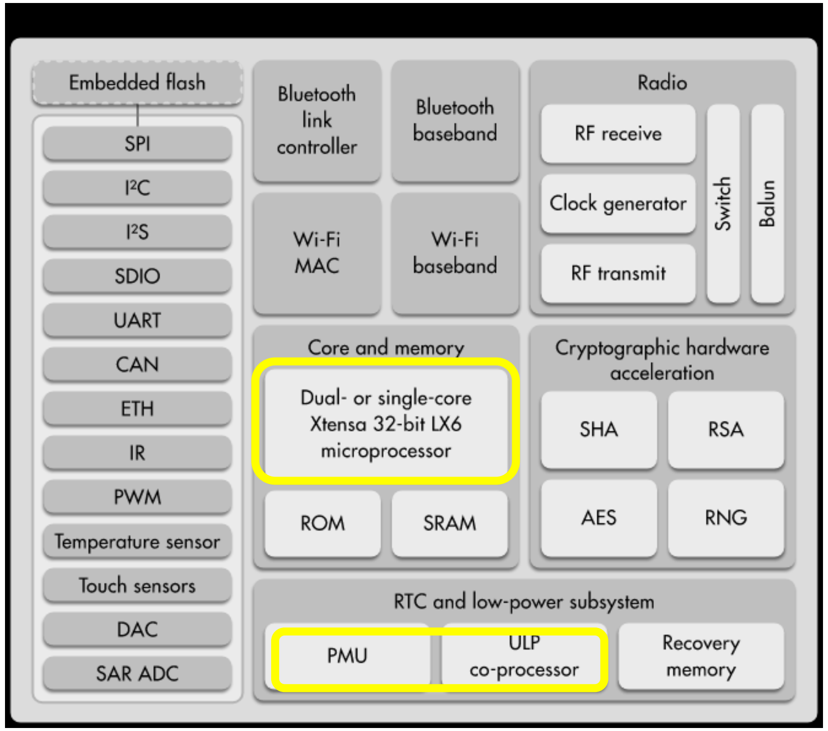
Processor

Real-Time Clock

ESP32 processors

- Processors
  - Tensilica Xtensa 32-bit LX6 microprocessor
  - 1 or 2 Cores
    - PRO_CPU
    - APP_CPU
  - up to 240 MHz for clock frequency
  - up to 600 DMIPS (Dhrystone MIPS)
  - Ultra Low Power (ULP) Coprocessor
  - Phasor measurement unit (PMU)

ESP32



System Time - Real-Time Clock (RTC)

- Choose accuracy requirements for system time
  - 1 system time source
  - 2 system time sources simultaneously (default)
- RTC timer source
  - Allows keeping the system time during any resets and sleep modes
  - Depends on an RTC Clock Source (Component config/ESP32-specific/CONFIG_ESP32_RTC_CLK_SRC)
  - affects accuracy only in sleep modes (6.6667 us resolution)
- High-resolution timer source
  - Not available during any reset and sleep modes
  - uses the APB_CLK clock source (typically 80 MHz)
- Programming
  - Using *settimeofday()/gettimeofday()* POSIX functions
  - Using *<time.h>* library with standard C library functions
- More information: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/system_time.html#rtc-clock-source

Memories

IRAM/DRAM

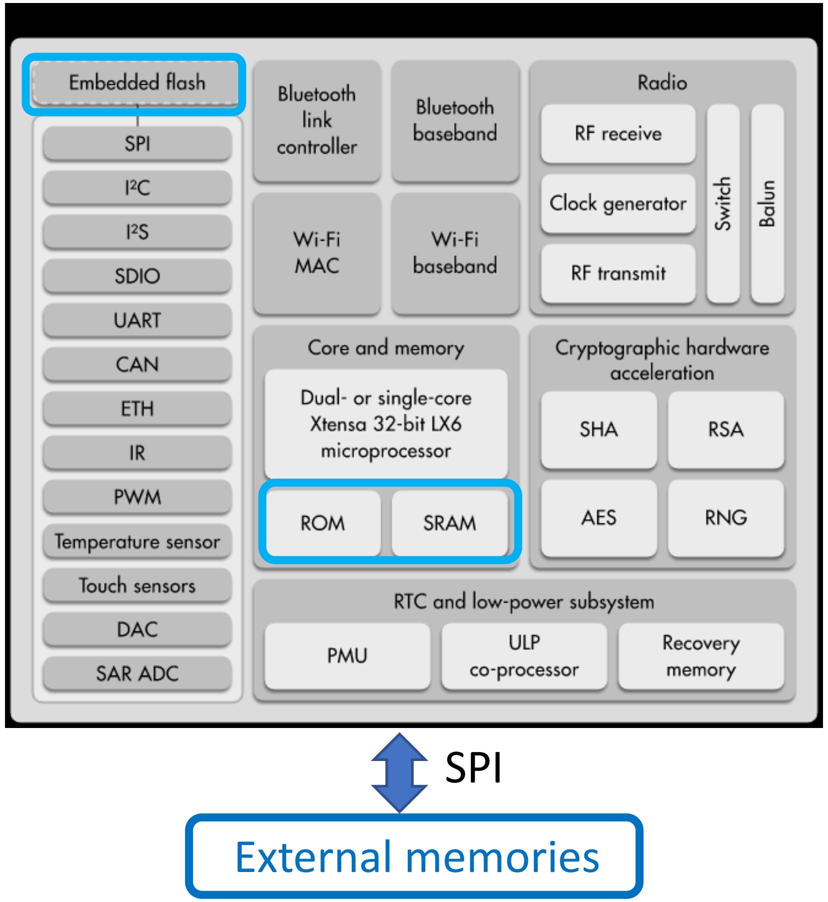
Non-Volatile Flash (NVS)

SPI Flash File System (SPIFFS)

Memories

- Internal memories
  - ROM: 448 KiB
  - SRAM: 520 KiB
  - RTC fast SRAM: 8 KiB
  - RTC slow SRAM: 8 KiB
  - eFuse: 1 Kibit
  - Embedded flash: 0/2/4 MiB
- External memories
  - Up to 16 MiB of external flash
  - Up to 8 MiB of SRAM memory
  - SPI (Serial Peripheral Interface) communication

ESP32



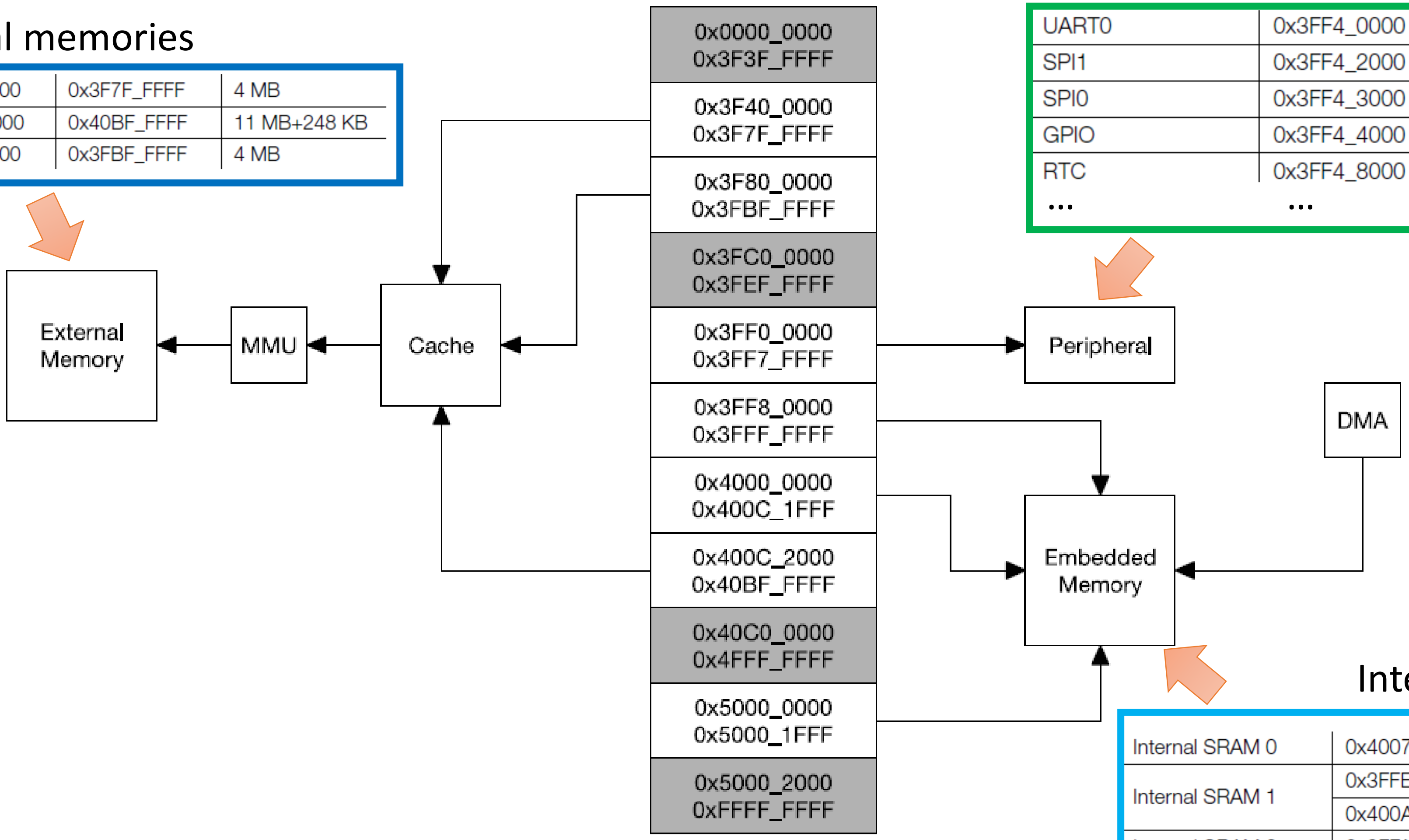
Memory Map

External memories

External Flash	0x3F40_0000	0x3F7F_FFFF	4 MB
	0x400C_2000	0x40BF_FFFF	11 MB+248 KB
External RAM	0x3F80_0000	0x3FBF_FFFF	4 MB

Peripheral input/output

UART0	0x3FF4_0000	0x3FF4_0FFF	4 KB
SPI1	0x3FF4_2000	0x3FF4_2FFF	4 KB
SPI0	0x3FF4_3000	0x3FF4_3FFF	4 KB
GPIO	0x3FF4_4000	0x3FF4_4FFF	4 KB
RTC	0x3FF4_8000	0x3FF4_8FFF	4 KB
...	...	...	...

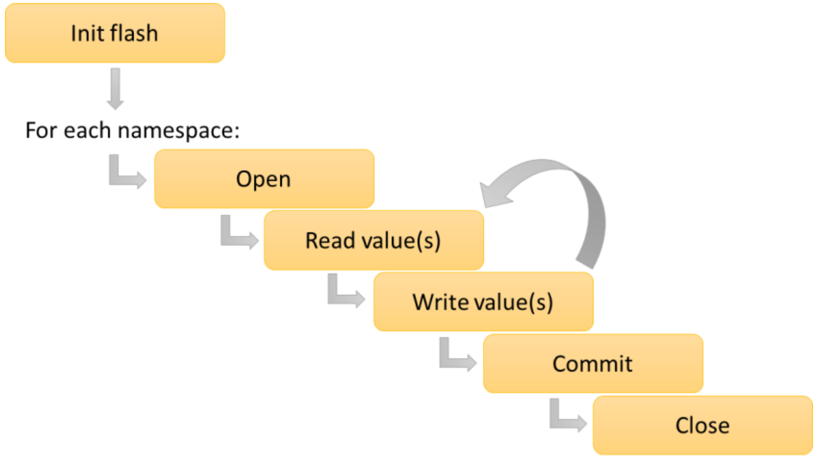


Internal memories

Internal SRAM 0	0x4007_0000	0x4009_FFFF	192 KB
Internal SRAM 1	0x3FFE_0000	0x3FFF_FFFF	128 KB
	0x400A_0000	0x400B_FFFF	
Internal SRAM 2	0x3FFA_E000	0x3FFD_FFFF	200 KB

Non-Volatile Storage (NVS)

- Designed to store key-value pairs in flash
- Using a portion of main flash memory (from spi_flash_{read|write|erase})
- Using partitions with type:data and subtype:nvs
- Sequence
  1. nvs_flash_init()
  2. nvs_open() / nvs_open_from_partition()
  3. nvs_get_*(), nvs_set_*()
  4. nvs_commit()
  5. nvs_close()



- More information: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/storage/nvs_flash.html

* = u8, u16, u32, u64, i16, i32, i64, str (4000 bytes)

SPI Flash File System (SPIFFS)

- SPIFFS is a file system intended for SPI Flash memories
- SPI = Serial Peripheral Interface communication
- Store a full directory into flash
- Flat structure
  - Does not support directory structure but the file name is like a path
  - SPIFFS is mounted under */spiffs*
  - File path: */spiffs/tmp/myfile.txt*
  - Create a file called */tmp/myfile.txt*
- Not a real-time stack
  - One write operation might take much longer than another
- No detection or handling bad blocks

More details: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/storage/spiffs.html

SPI Flash File System (SPIFFS)

- Size is configured in the partition table
  - *my_storage,data,spiffs,0x110000,1M,*
- Using a tool to convert a directory to a bin file.
  - *spiffsgen.py 0x100000 spiffs_dir spiffs_dir.bin*
- Using a tool to load the file into flash.
  - *esptool.py --chip esp32 --port /dev/ttyUSB0 --baud 115200 write_flash -z 0x110000 spiffs_dir.bin*
- How to handle file system?
  - Using common C file functions to create read/update/delete files

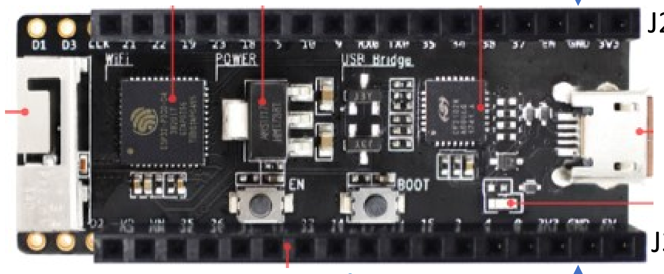
GPIO

Input/Output

Interrupt

General Purpose Input/Output (GPIO)

- Can be configure as input or output
- GPIO pins
    - GPIO1 (1), GPIO3-11 (9), GPIO16-19 (4), GPIO21-23 (3), GPIO25-27 (3), GPIO36-39 (4)
    - GPIO6-11 are usually used for SPI flash.
    - GPIO36-39 can only be set as input mode and do not have software pullup or pulldown functions.
    - Pins GPIO16, GPIO17, SD_CMD, SD_CLK, SD_DATA_0 and SD_DATA_1 are used for connecting the embedded flash
- RTC_GPIO pins
    - RTC_GPIO0-17
    - can be used to trigger ESP32 from sleep mode
- Not all pins can be used at the same time!
- More information: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/gpio.html



GPIO / RTC_GPIO pins

J3 Header

GPIO

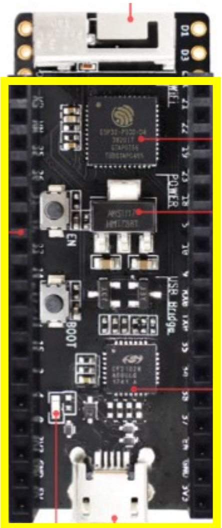
No.	Name	Type	Function
1	FLASH_CS (FCS)	I/O	GPIO16, HS1_DATA4 (See 1) , U2RXD, EMAC_CLK_OUT
2	FLASH_SD0 (FSD0)	I/O	GPIO17, HS1_DATA5 (See 1) , U2TXD, EMAC_CLK_OUT_180
3	FLASH_SD2 (FSD2)	I/O	GPIO11, SD_CMD, SPICS0, HS1_CMD (See 1) , U1RTS
4	SENSOR_VP (FSVP)	I IN	GPIO36, ADC1_CH0, RTC_GPIO0
5	SENSOR_VN (FSVN)	I IN	GPIO39, ADC1_CH3, RTC_GPIO3
6	IO25	I/O	GPIO25, DAC_1, ADC2_CH8, RTC_GPIO6, EMAC_RXD0
7	IO26	I/O	GPIO26, DAC_2, ADC2_CH9, RTC_GPIO7, EMAC_RXD1
8	IO32	I/O	32K_XP (See 2a) , ADC1_CH4, TOUCH9, RTC_GPIO9
9	IO33	I/O	32K_XN (See 2b) , ADC1_CH5, TOUCH8, RTC_GPIO8
10	IO27	I/O	GPIO27, ADC2_CH7, TOUCH7, RTC_GPIO17, EMAC_RX_DV
11	IO14	I/O	ADC2_CH6, TOUCH6, RTC_GPIO16, MTMS, HSPICLK, HS2_CLK, SD_CLK, EMAC_TXD2
12	IO12	I/O	ADC2_CH5, TOUCH5, RTC_GPIO15, MTDI (See 4) , HSPIQ, HS2_DATA2, SD_DATA2, EMAC_TXD3
13	IO13	I/O	ADC2_CH4, TOUCH4, RTC_GPIO14, MTCK, HSPID, HS2_DATA3, SD_DATA3, EMAC_RX_ER
14	IO15	I/O	ADC2_CH3, TOUCH3, RTC_GPIO13, MTDO, HSPICS0 HS2_CMD, SD_CMD, EMAC_RXD3
15	IO2	I/O	ADC2_CH2, TOUCH2, RTC_GPIO12, HSPIWP, HS2_DATA0, SD_DATA0
16	IO4	I/O	ADC2_CH0, TOUCH0, RTC_GPIO10, HSPIHD, HS2_DATA1, SD_DATA1, EMAC_TX_ER
17	IO0	I/O	ADC2_CH1, TOUCH1, RTC_GPIO11, CLK_OUT1 EMAC_TX_CLK
18	VDD33 (3V3)	P	3.3V power supply
19	GND	P	Ground
20	EXT_5V (5V)	P	5V power supply

RTC_GPIO

RTC_GPIO

3. *This pin is connected to the pin of the USB bridge chip on the board.*
4. *The operating voltage of ESP32-PICO-KIT's embedded SPI flash is 3.3V. Therefore, the strapping pin MTDI should hold bit zero during the module power-on reset. If connected, please make sure that this pin is not held up on reset.*

ESP32-PICO-D4



J3 Header

USB

J2 Header

J2 Header

GPIO

No.	Name	Type	Function
1	FLASH_SD1 (FSD1)	I/O	GPIO8, SD_DATA1, SPID, HS1_DATA1 (See 1) , U2CTS
2	FLASH_SD3 (FSD3)	I/O	GPIO7, SD_DATA0, SPIQ, HS1_DATA0 (See 1) , U2RTS
3	FLASH_CLK (FCLK)	I/O	GPIO6, SD_CLK, SPICLK, HS1_CLK (See 1) , U1CTS
4	IO21	I/O	GPIO21, VSPIHD, EMAC_TX_EN
5	IO22	I/O	GPIO22, VSPIWP, U0RTS, EMAC_TXD1
6	IO19	I/O	GPIO19, VSPIQ, U0CTS, EMAC_TXD0
7	IO23	I/O	GPIO23, VSPID, HS1_STROBE
8	IO18	I/O	GPIO18, VSPICLK, HS1_DATA7
9	IO5	I/O	GPIO5, VSPICS0, HS1_DATA6, EMAC_RX_CLK
10	IO10	I/O	GPIO10, SD_DATA3, SPIWP, HS1_DATA3, U1TXD
11	IO9	I/O	GPIO9, SD_DATA2, SPIHD, HS1_DATA2, U1RXD
12	RXD0	I/O	GPIO3, U0RXD (See 3) , CLK_OUT2
13	TXD0	I/O	GPIO1, U0TXD (See 3) , CLK_OUT3, EMAC_RXD2
14	IO35	I IN	ADC1_CH7, RTC_GPIO5
15	IO34	I IN	ADC1_CH6, RTC_GPIO4
16	IO38	I IN	GPIO38, ADC1_CH2, RTC_GPIO2
17	IO37	I IN	GPIO37, ADC1_CH1, RTC_GPIO1
18	EN	I	CHIP_PU
19	GND	P	Ground
20	VDD33 (3V3)	P	3.3V power supply

RTC_GPIO

1. *This pin is connected to the flash pin of ESP32-PICO-D4.*
2. *32.768 kHz crystal oscillator: a) input b) output*

Interrupt

- Polling mode
  - CPU steadily checks whether the devices (GPIO …) need attention
  - CPU waste countless processor cycles by repeatedly checking devices!
- Interrupt mode
  - the devices notice the CPU that it requires its attention.
  - CPU is simply disturbed once any device interrupts it.
- Benefits
  - No wasting CPU cycles
  - Minimum latency (depends on priority)
  - Save energy of the battery

Interrupt – GPIO example

- IRQ
  - Interrupt ReQuest
  - Manage various hardware operations
  - Assign different IRQ addresses to different hardware devices
- ISR
  - Interrupt Sub Routine
  - Interrupt handler
- Programming
  - Handler function
  - Initializer function for handler
    - Pass the handler function
    - How to trig handler ? (input, level)
  - Enable/disable handler