

ESP32 DevKitC

ARQUITECTURA INTERNA DEL NODO IoT



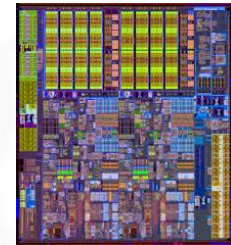
ESP32 DevKitC



ESP32-WROOM-32



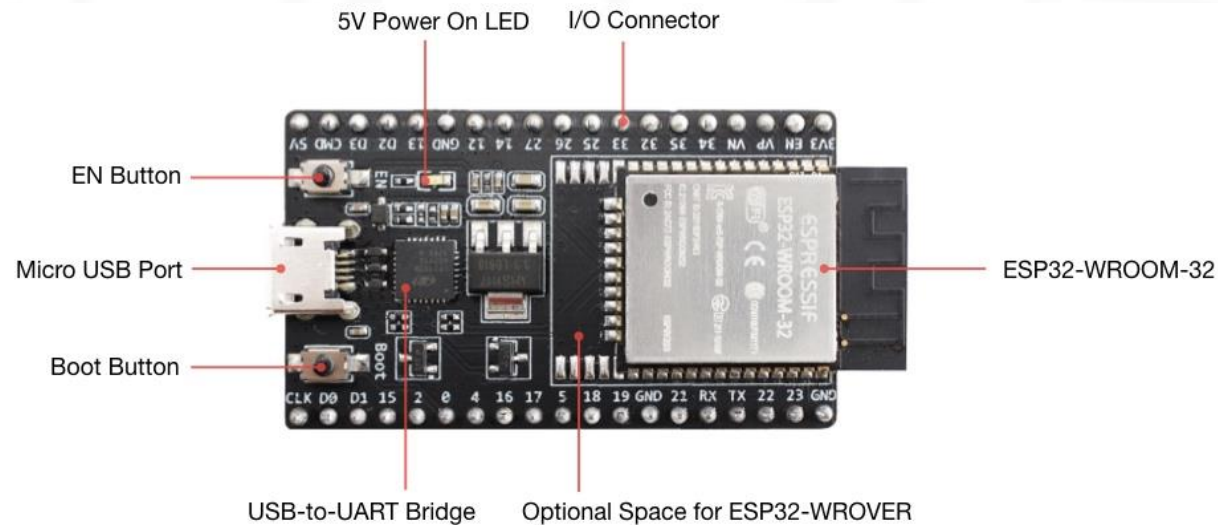
ESP32-D0WD-V3



*Xtensa dual-core
32-bit LX6*

- ❑ Bluetooth (BLE) y Wifi
- ❑ 4 MB Flash, soporte para tarjeta SD

- ❑ UART, SPI, SDIO, I²C, LED PWM, Motor PWM, I²S, IR, pulse counter, GPIO, capacitive touch sensor, ADC, DAC
- ❑ 2 cores con frecuencias entre 80MHz y 240MHz



ESP32 DevKitC

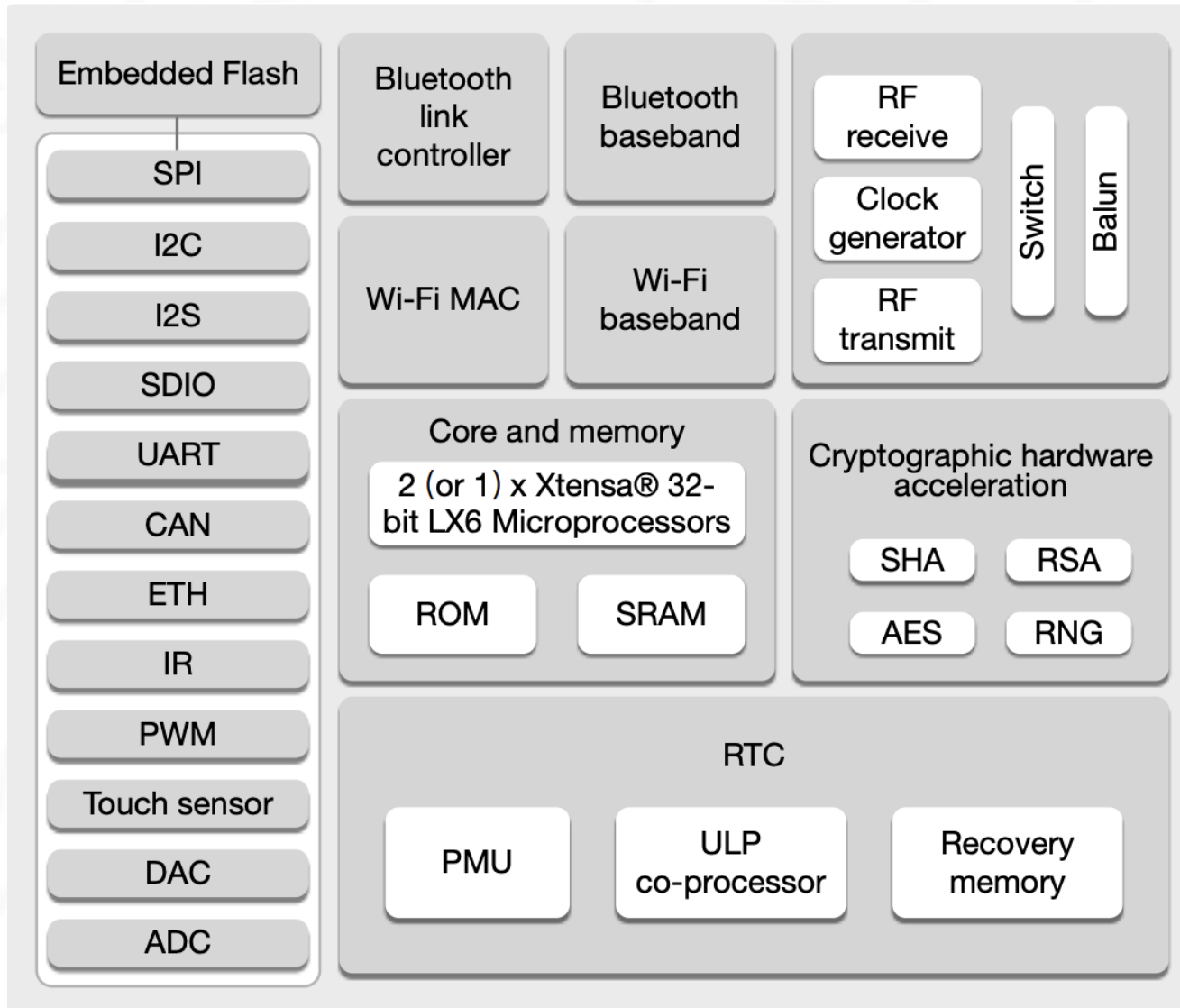
<https://www.espressif.com/en/products/devkits/esp32-devkitc>

<https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-devkitc.html>

<https://products.espressif.com/#/product-comparison>



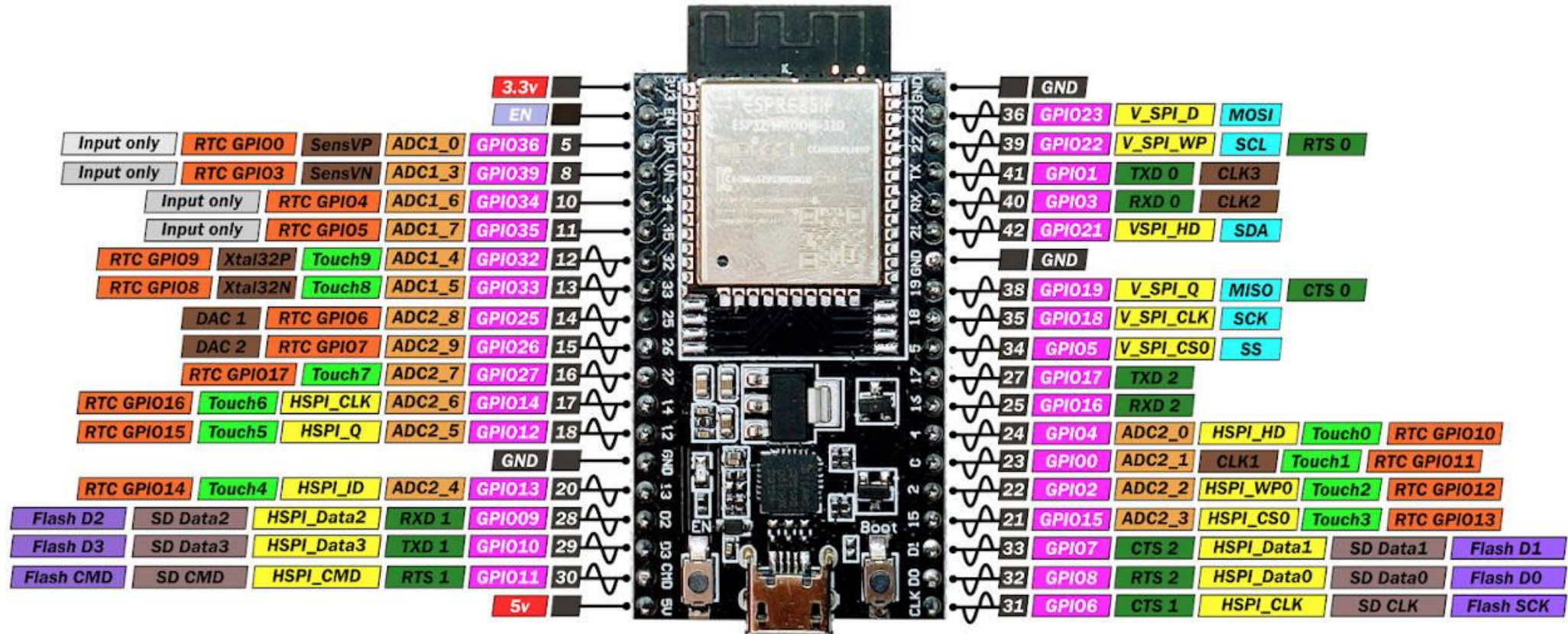
- ❑ Diseñado para bajo consumo
 - Varios modos de consumo, *power scaling*
 - Es posible *despertar* al ESP32 periódicamente o ante determinados eventos
- ❑ Integra la funcionalidad para WiFi y BLE
- ❑ CPU Xtensa 32-bit LX6 microprocessor
- ❑ 34 GPIOs programables, 18 canales de ADC de 12 bits, SPI, UART, I2C
 - Touch sensors y Hall sensors
- ❑ Secure boot, aceleración HW para cifrado (AES, RSA...)

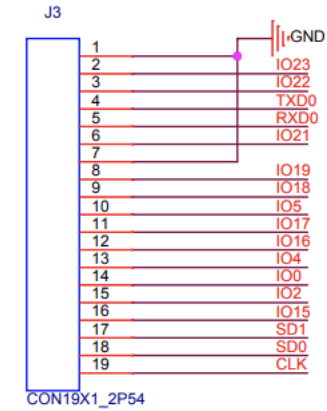
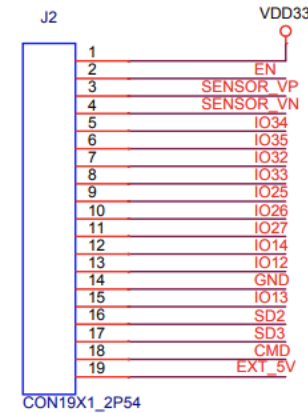
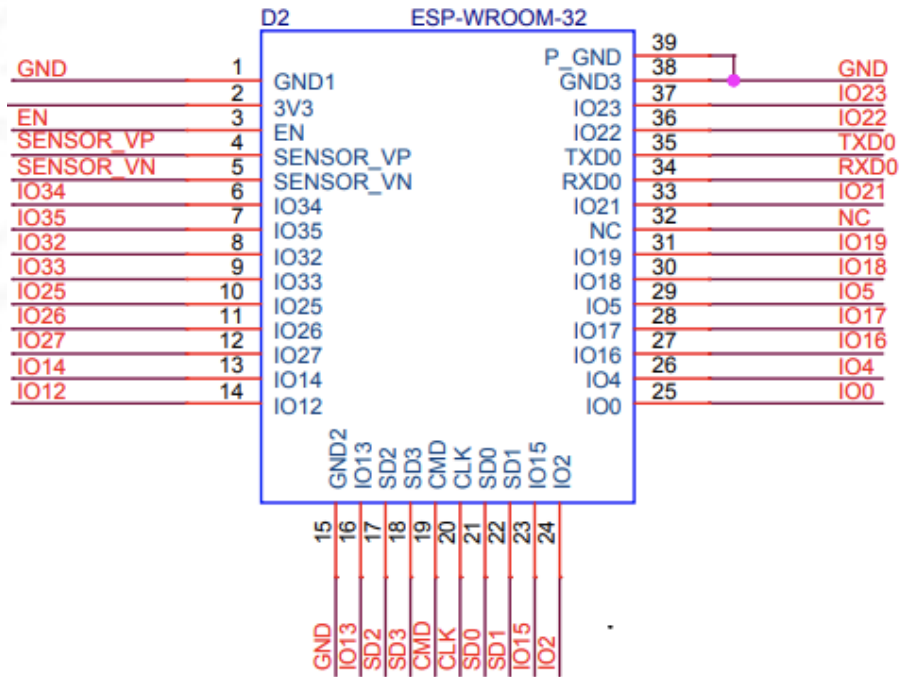


https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

ESP32 DevKitC V4

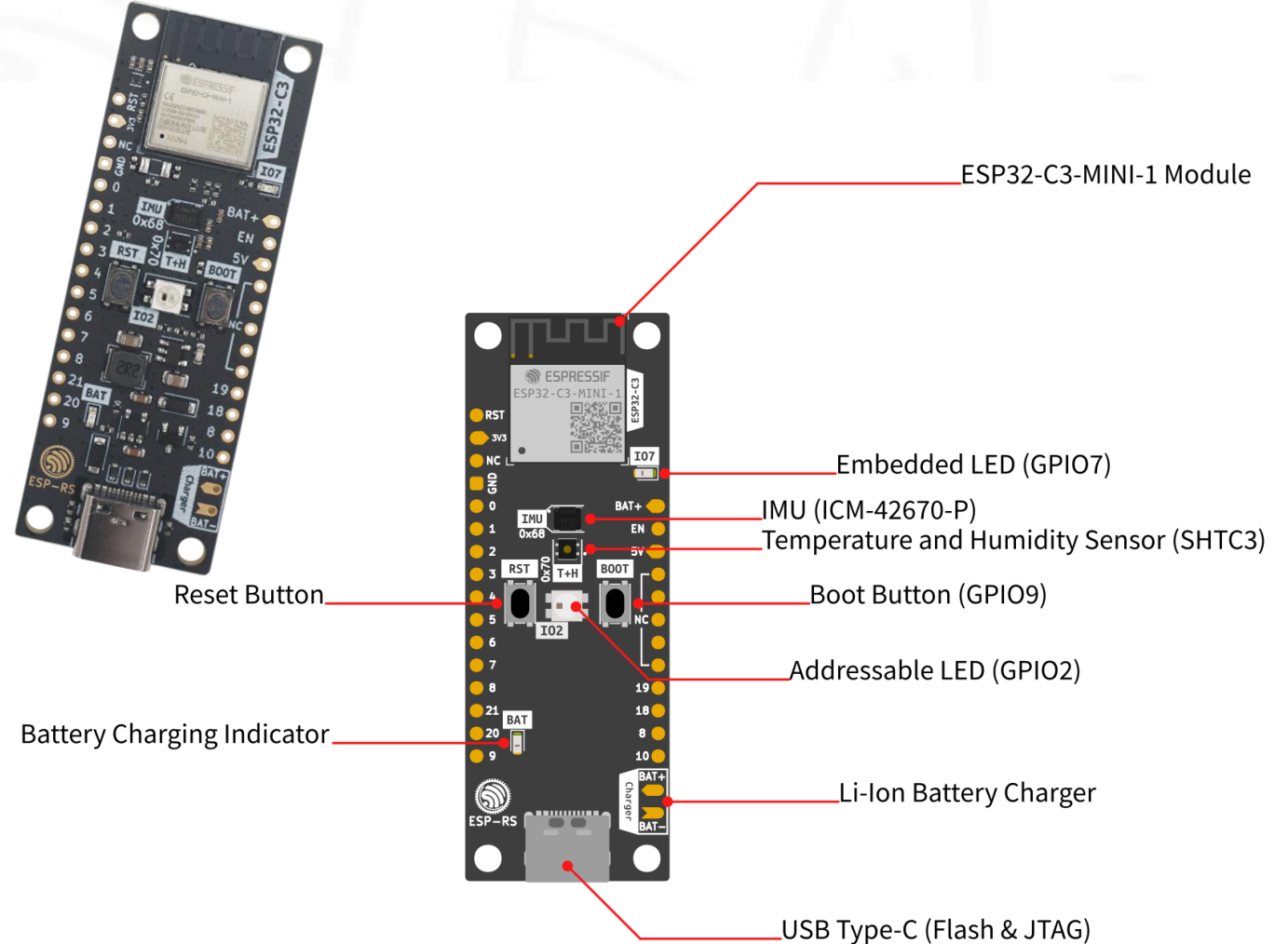
PINOUT





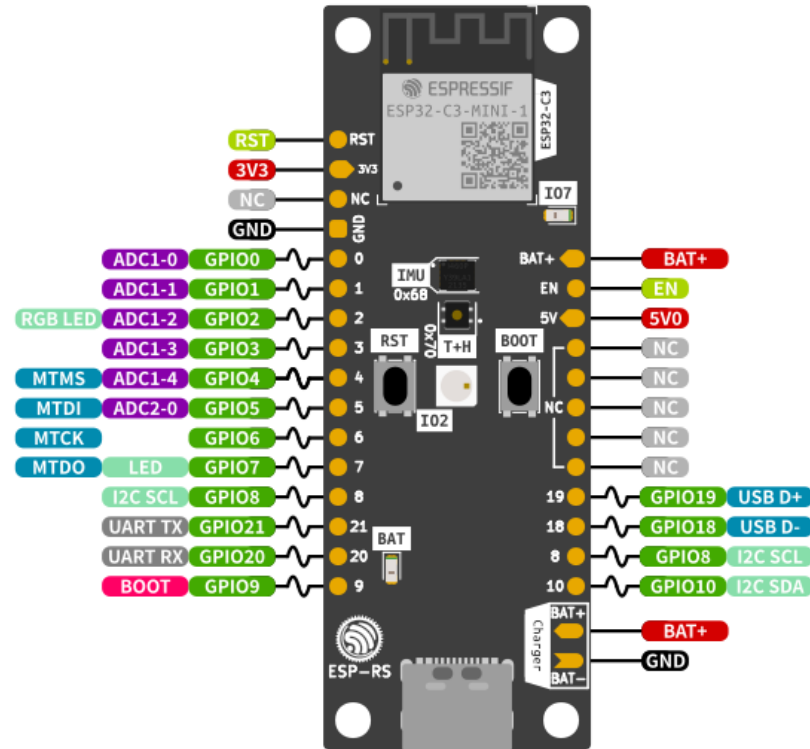
https://dl.espressif.com/dl/schematics/esp32_devkitc_v4-sch.pdf

- ❑ WiFi y BLE
- ❑ Core RISC-V, hasta 160MHz
- ❑ 383 KB ROM 400 KB SRAM
- ❑ 4 MB flash
- ❑ GPIO (22 pines programables)
- ❑ Sensor de temperatura en SoC
- ❑ Temp + IMU por I2C



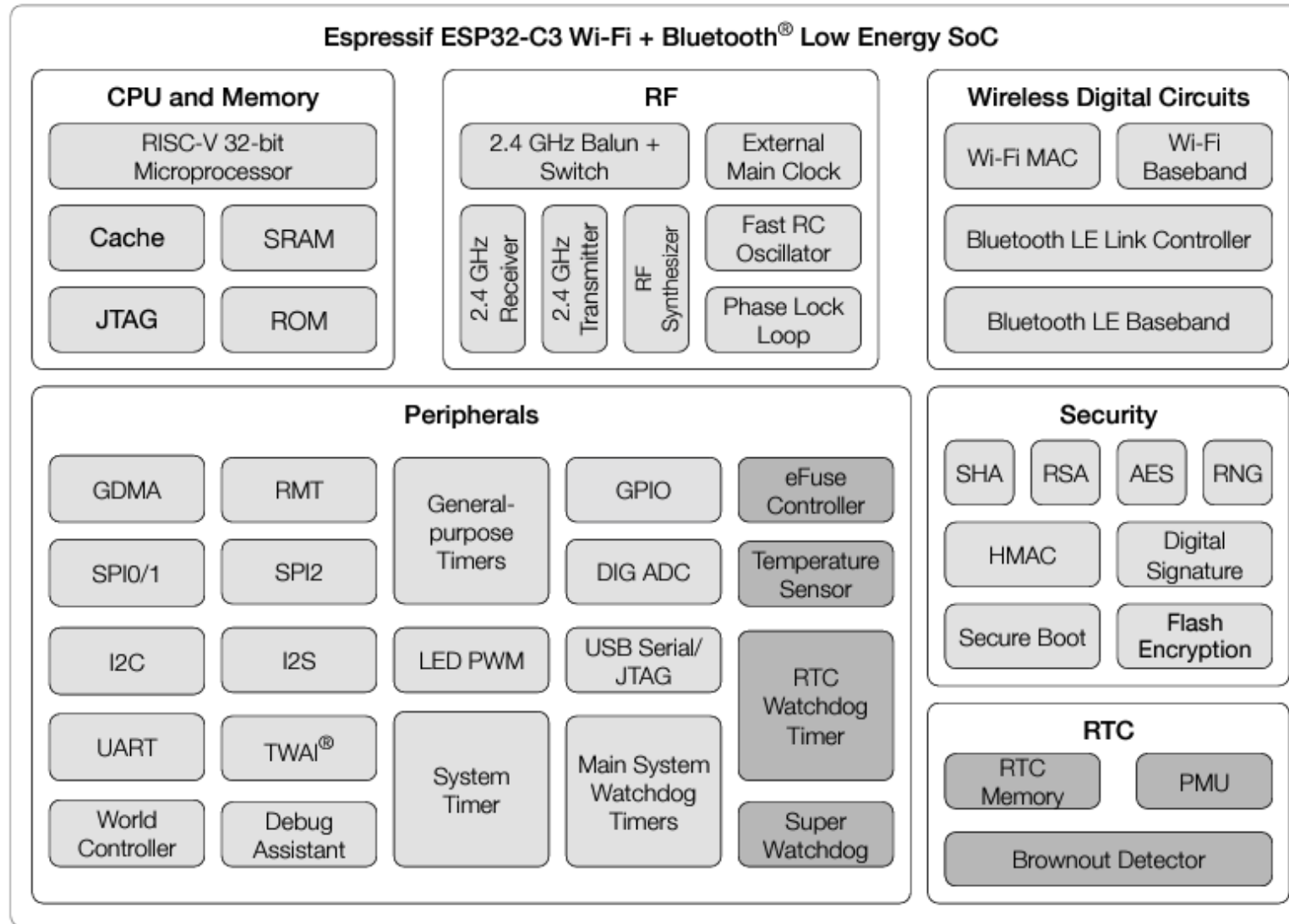
Rust Board ESP32-C3

Rust Board ESP32-C3



- PWM Capable Pin
- Miscellaneous/Secondary Functions
- General Purpose Input and Output
- Board Related Functions
- Strapping Pin Functions
- JTAG for Debugging and/or USB
- Analog-to-Digital Converter
- Serial for Debug/Programming
- Ground Plane
- Power Rails

ESP32-C3 Diagrama de bloques funcionales

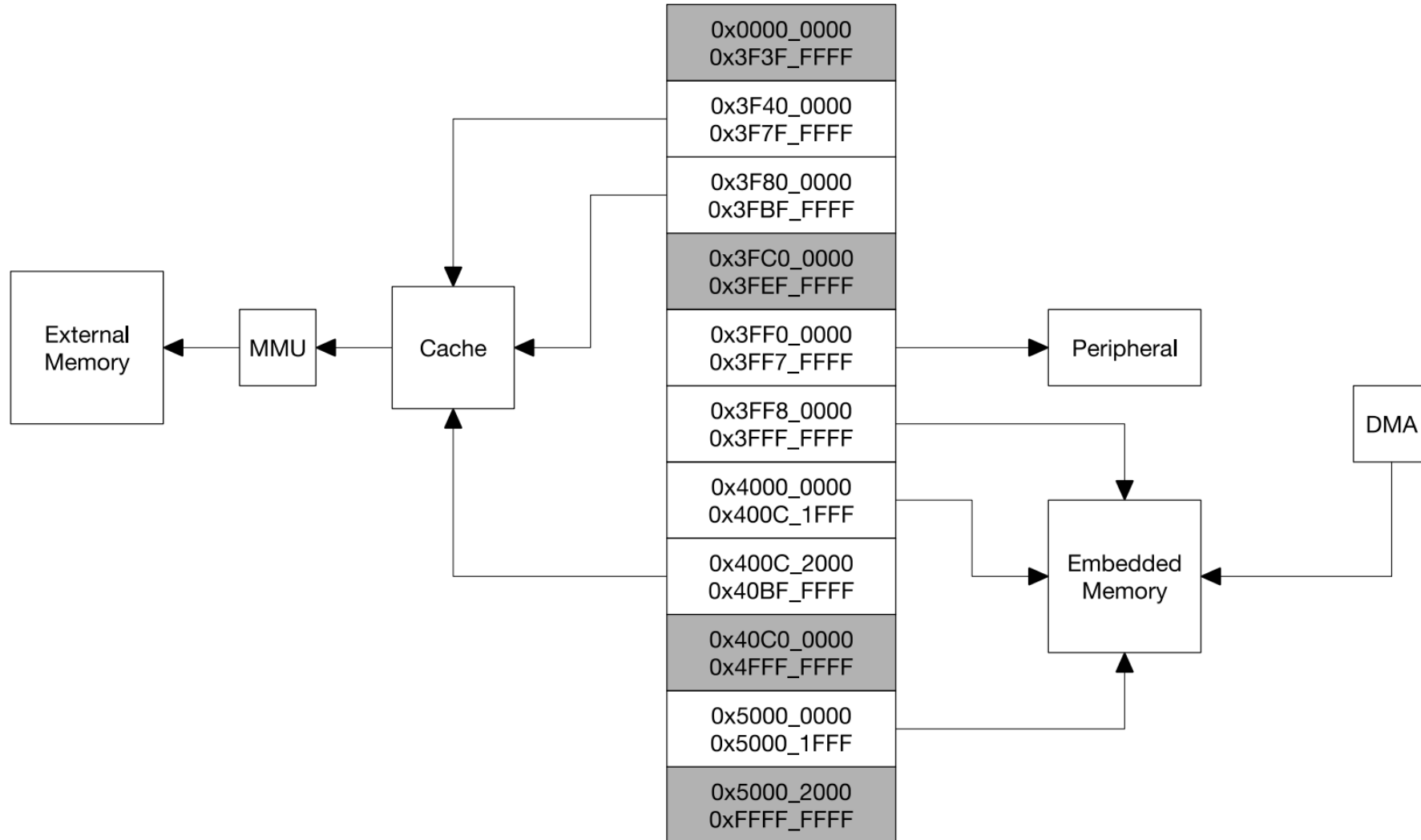


Power consumption

-  Normal
-  Low power consumption components capable of working in Deep-sleep mode

- ❑ https://www.espressif.com/sites/default/files/documentation/esp32-c3-mini-1_datasheet_en.pdf
- ❑ https://www.espressif.com/sites/default/files/documentation/esp32-c3_technical_reference_manual_en.pdf
- ❑ https://www.espressif.com/sites/default/files/documentation/esp32-c3_datasheet_en.pdf
- ❑ <https://docs.espressif.com/projects/esp-hardware-design-guidelines/en/latest/esp32c3/index.html>

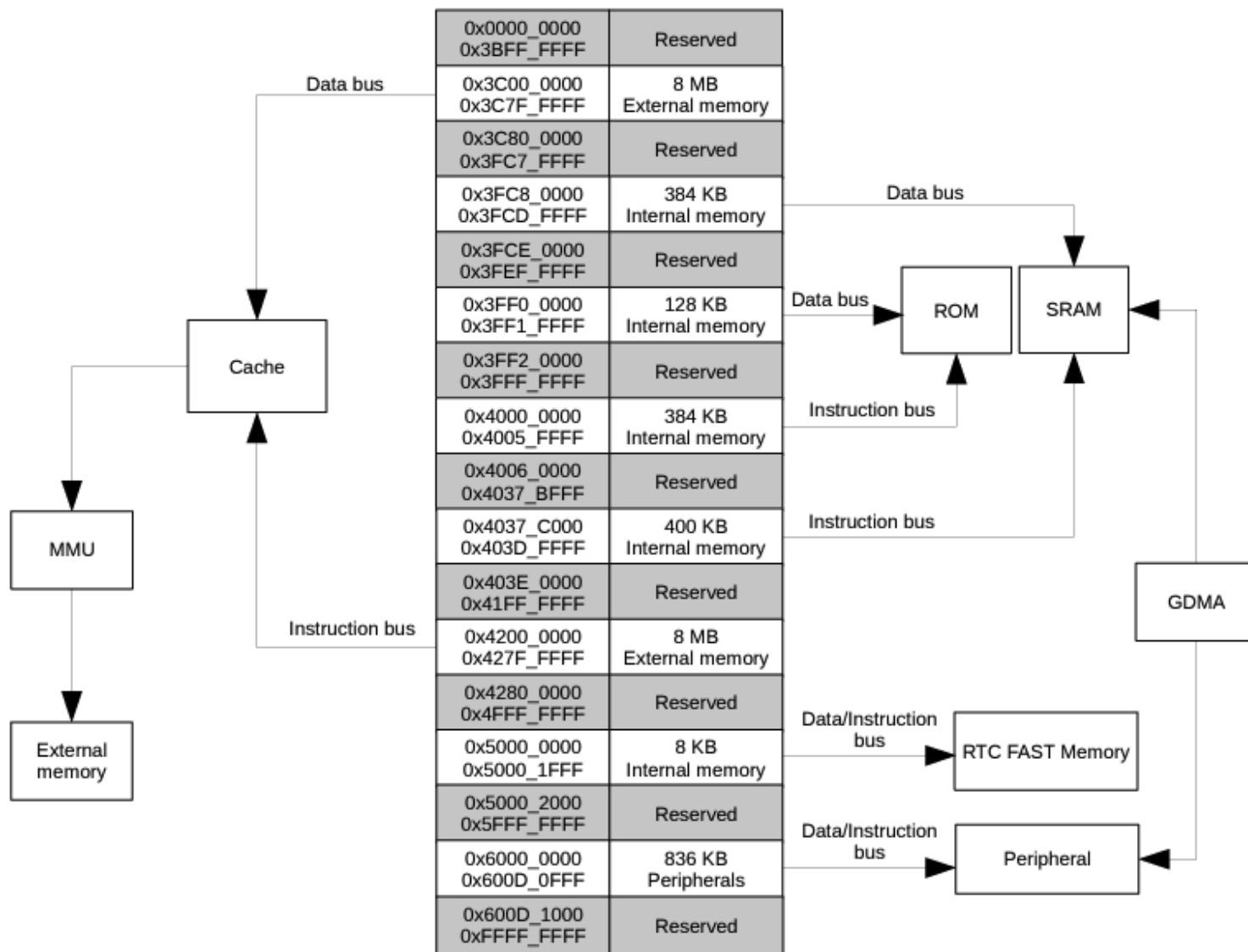
- ❑ Ambos SoCs / módulos incorporan varias memorias físicas:
 - ROM, SRAM, Flash....
- ❑ Las memorias, junto con el resto de dispositivos, se proyectan en el **Mapa de Memoria**
 - La posición 0x000000 de la memoria ROM puede estar en la dirección 0x123400 del mapa
 - El controlador de GPIO puede estar en la dirección 0xFF00000
 -



SoC ESP32-D0WD-V3. Mapa de memoria

Category	Target	Start Address	End Address	Size
Embedded Memory	Internal ROM 0	0x4000_0000	0x4005_FFFF	384 KB
	Internal ROM 1	0x3FF9_0000	0x3FF9_FFFF	64 KB
	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
	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
Peripheral	RTC	0x3FF4_8000	0x3FF4_8FFF	4 KB
	IO MUX	0x3FF4_9000	0x3FF4_9FFF	4 KB
	SDIO Slave	0x3FF4_B000	0x3FF4_BFFF	4 KB
	UDMA1	0x3FF4_C000	0x3FF4_CFFF	4 KB

Mapa de memoria de ESP32-C3



- ❑ Internal ROM (384 KB): sólo lectura
 - Dividida en Internal ROM0 e internal ROM1
- ❑ Internal SRAM (400KB)
 - Internal SRAM 0. 16KB. R/W. Accesible por bus instrucciones (4 bytes)
 - Internal SRAM 1. 384 KB. R/W. Accesible por bus instrucciones (4 bytes) o datos (1,2,4b)
- ❑ RTC Fast Memoria. 8KB. R/W.
- ❑ Memoria externa
 - 4MB Quad SPI Flash
 - Accesible via cache (16KB, 8-way)
 - ESP32-C3 puede acceder hasta 16MB de flash externa: hasta 8MB de instrucciones y 8MB de datos

- ❑ ESP-IDF es el framework oficial de Espressif para el uso de la familia ESP32
 - <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/>
- ❑ Basado en FreeRTOS, sistema operativo de tiempo real ampliamente utilizado
 - <https://www.freertos.org/>
- ❑ ESP-IDF es un *porting* de FreeRTOS
 - Con bastantes complementos: módulos para WiFi, BLE, Eventos...

❑ Documentación

- <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/memory-types.html>
- <https://docs.espressif.com/projects/esp-idf/en/latest/esp32c3/api-guides/memory-types.html>

❑ DRAM (Data RAM)

- Secciones .data y .bss se llevan Internal SRAM

❑ IRAM (Instruction RAM)

- Para albergar código crítico (por latencia): IRSs...

❑ IROM (Code Executed from flash)

- Emplazamiento por defecto del código

❑ DROM (Data stored in flash)

- Para datos de sólo lectura

- ❑ Espacio de memoria único para todas las tareas
- ❑ Cada tarea tiene su propia pila
 - Por defecto, el espacio de pila de cada tarea se coge del *heap*
 - OJO: tamaño fijo de pila. ¡Desbordamientos frecuentes!
- ❑ ¿Qué pasa con *malloc()* o similar?
 - Toda memoria interna SRAM que no se use para instrucciones, se usará para datos
 - Hay varios *heaps* --> *heap_caps_malloc()*
- ❑ Memory Capabilities
 - MALLOC_CAP_8BIT, MALLOC_CAP_32BIT, MALLOC_CAP_DMA, MALLOC_CAP_DEFAULT...
 - *malloc()* → *heap_caps_malloc(MALLOC_CAP_DEFAULT)*
 - Siempre se libera con *free()*
- ❑ API completa para controlar el *heap*
 - Información, *hooks*, detección de corrección...