50.046 Cloud Computing and IoT

Embedded devices

2023 Term 7

Assoc. Prof. JIANG Wenchao

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Definition

- What is an embedded system?
  - Purposely-built system (hardware / software) serving some **specific / dedicated** application
  - Implications:
    - Minimize cost / size / energy consumption
    - Resource constrained

Outline

- Processor types
  - MCU vs MPU
- Computer architecture
  - Based on instruction set and memory
- ESP32 as an example
  - Processor
  - Memory
  - I/O peripheral

Processor Types

MCU vs. MPU

What is a Microcontroller (MCU)?

- Single chip
- Processor + Memory + I/O
- Application-specific
- Low-power consumption

- Examples
  - Digital thermometer
  - Automatic coffee maker
  - Washing machine

What is a Microprocessor (MPU)?

- General-purpose
- Requires external components
- Higher computational power
- Versatile but power-consuming

- Examples
  - Laptops
  - Servers
  - High-end drones

Key Differences - Memory

- MCU: Onboard RAM and flash
- MPU: External memory

- Example:
  - MCU: Arduino, ESP32
  - MPU: Raspberry Pi, Smartphone chips

Key Differences - Power Consumption

- MCU: Low-power, battery-operable
- MPU: Higher power consumption

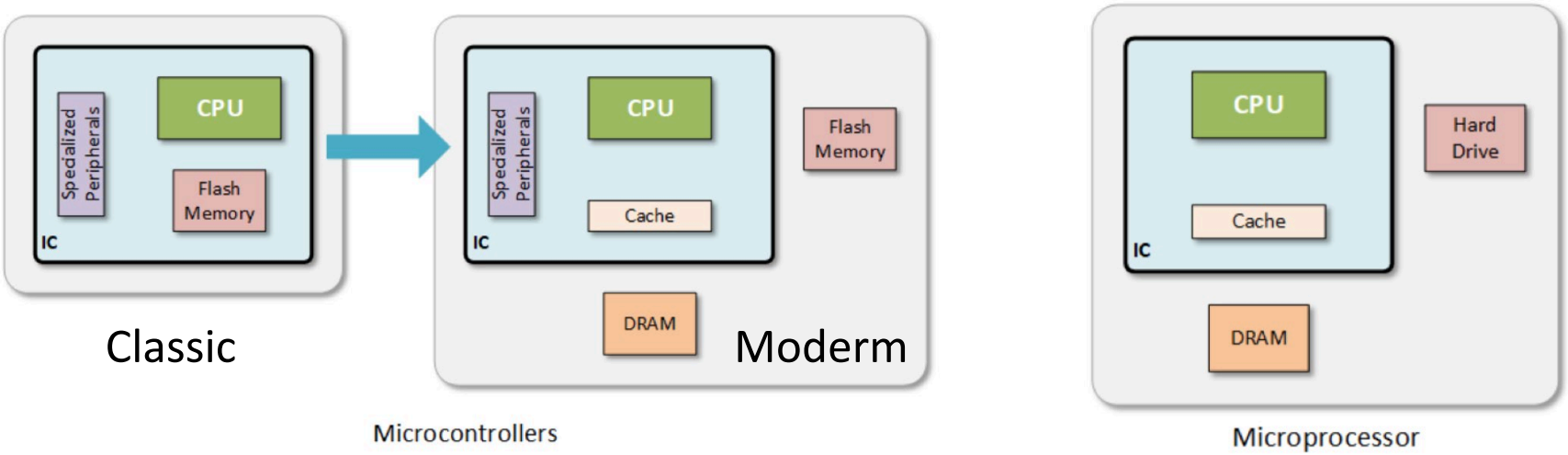
- Example
  - MCU: Fitness trackers
  - MPU: Gaming consoles

Key Differences - I/O Capabilities

- MCU: Built-in I/O capabilities
- MPU: Requires external I/O controllers

- Example:
  - MCU: ESP32 with WiFi
  - MPU: Desktop PC with external network card

Evolution of MCU



A controller / a computer
Purposely built
On-chip flash (usually)
Specialized peripheral
Optimize for latency (real-time)
SW: bare-metal or RTOS

A processor
General purpose
On-chip cache
No peripheral
Optimize for throughput
SW: Full OS

https://semiengineering.com/mpu-vs-mcu/

Computer Architecture

von Neumann vs. Harvard

RISC vs. CISC

Computer architecture in a nutshell

- Separation of **Processor** and **memory** distinguishes programmable computer
- Processor fetches instructions from memory
- **Registers** help out: program counter (PC), instruction register (IR), general-purpose registers, etc.

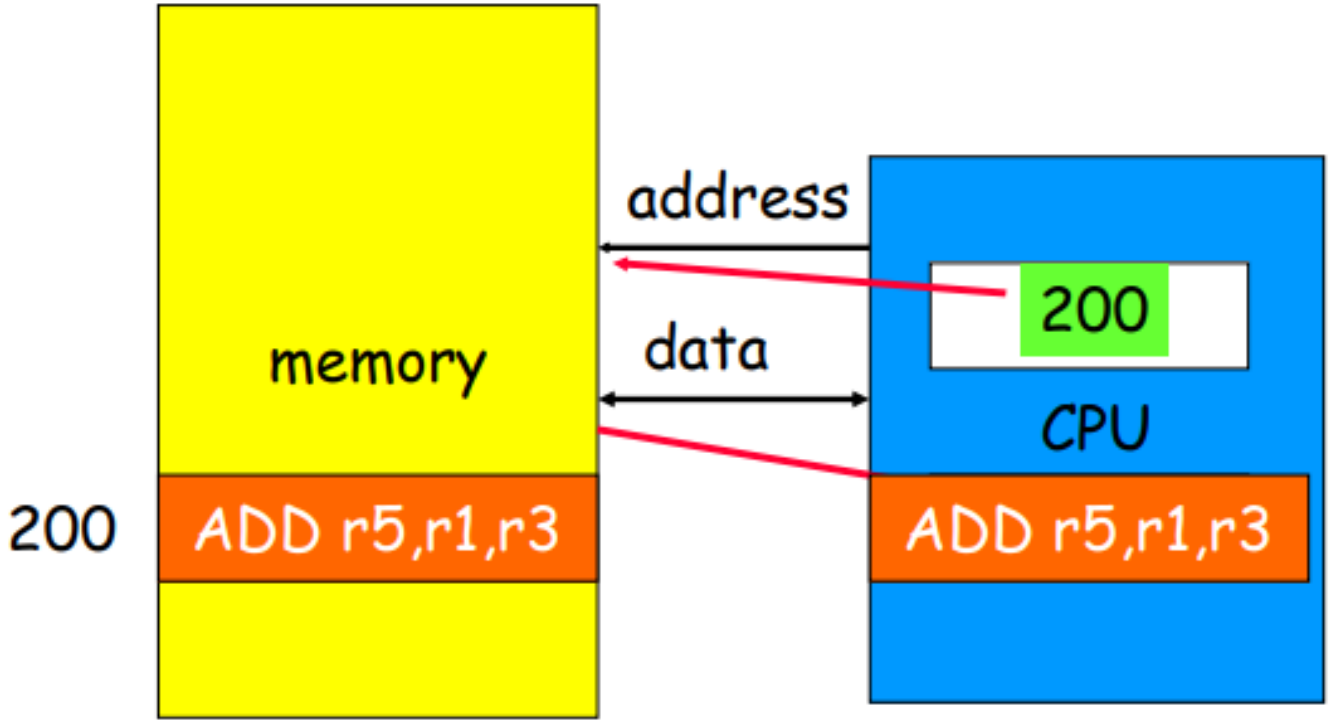
Computer architecture categories

- If instruction and data share memory and buses
  - von Neumann vs. Harvard

Bus is a communication system that transfers data between components inside a computer
- Data buses
- Address buses

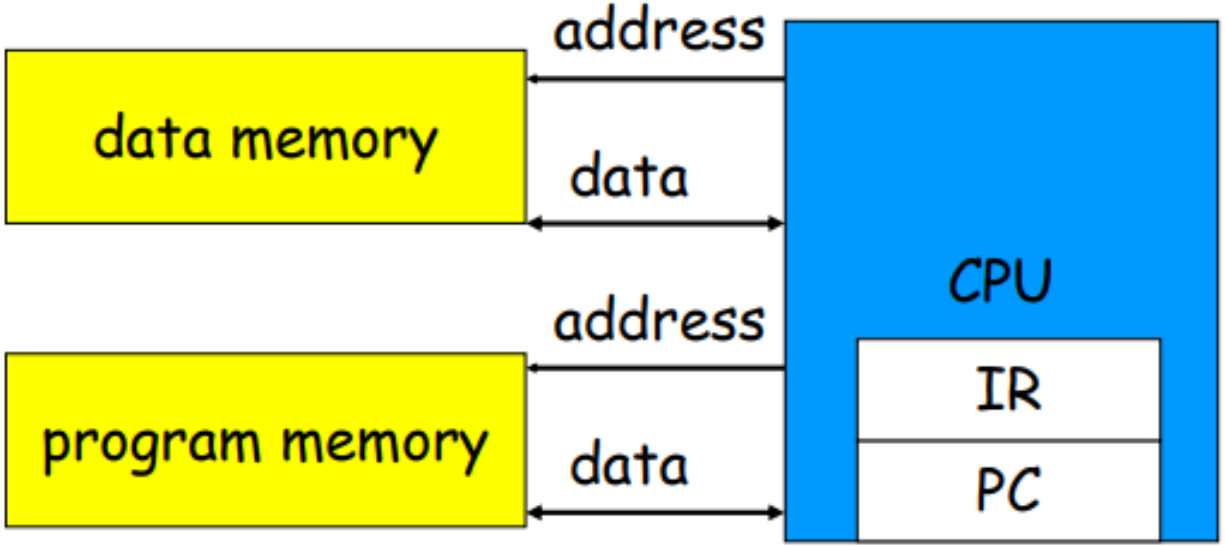
von Neumann architecture

- Same memory holds data, instructions.
- A single set of address/data buses between processor and memory



Harvard architecture

- Named after Harvard Mark I, an IBM computer in WWII
- Separate memories for data and instructions.
- Two sets of address/data buses between processor and memory
  - Greater memory bandwidth

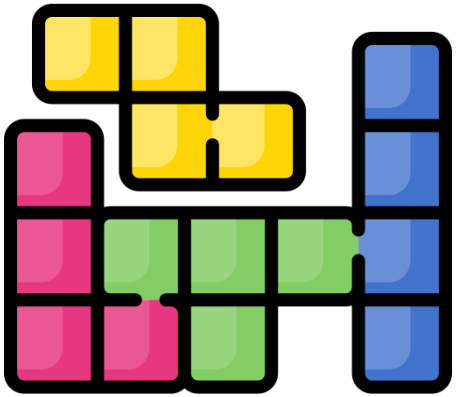


Computer architecture categories

- If instruction and data share memory and buses
  - von Neumann vs. Harvard
- Based on processor instruction sets architecture (ISA)
  - RISC vs. CISC

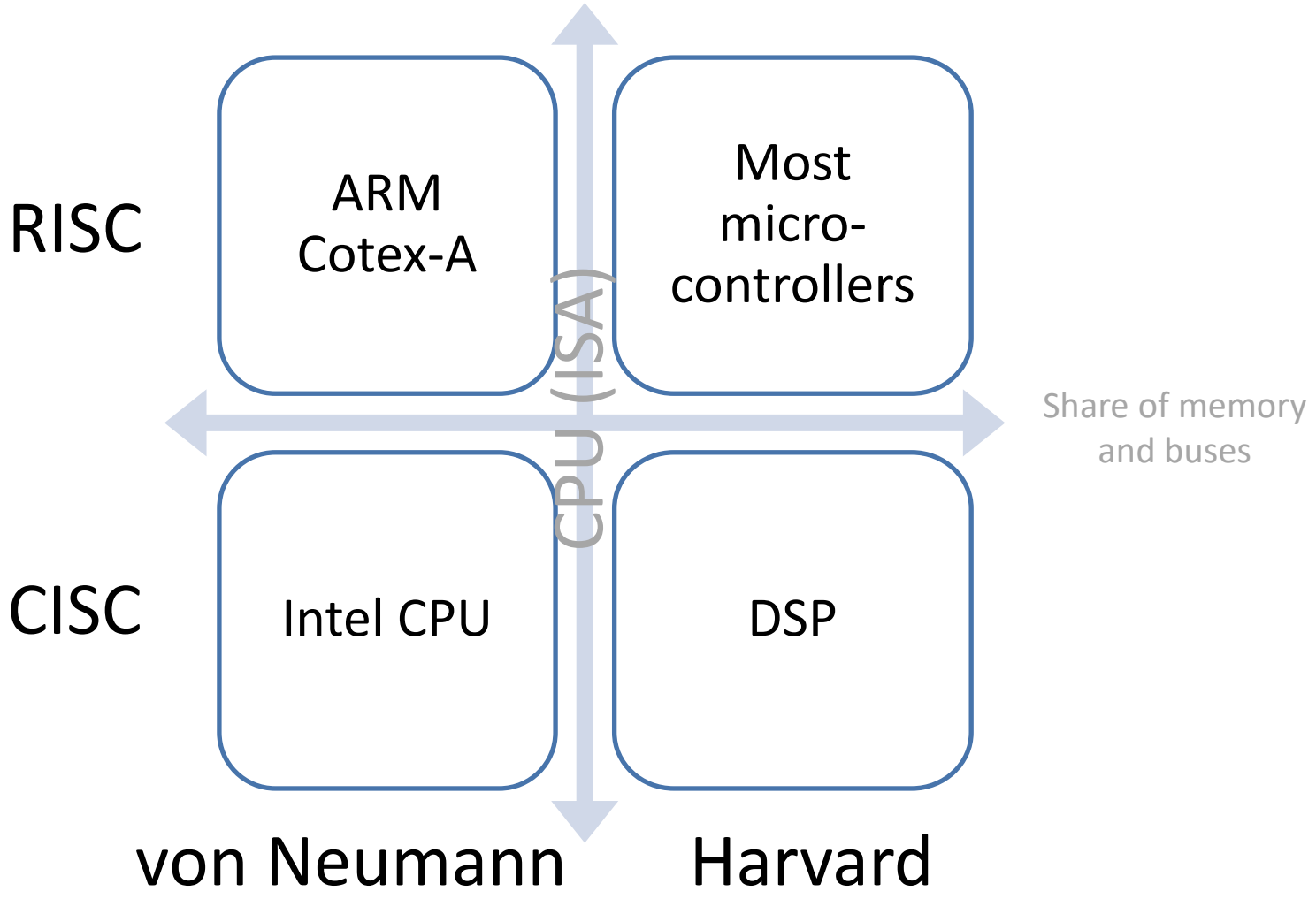
RISC vs CISC

- Reduced Instruction Set Computer (RISC)
  - **Compact, uniform instructions** -> facilitate pipelining
  - More lines of code -> large memory footprint
  - Allow effective compiler optimization
- Complex Instruction Set Computer (CISC)
  - Many addressing modes and long instructions
  - High code density
  - Often require manual optimization of assembly code for embedded systems



Like Tetris!

Microprocessors

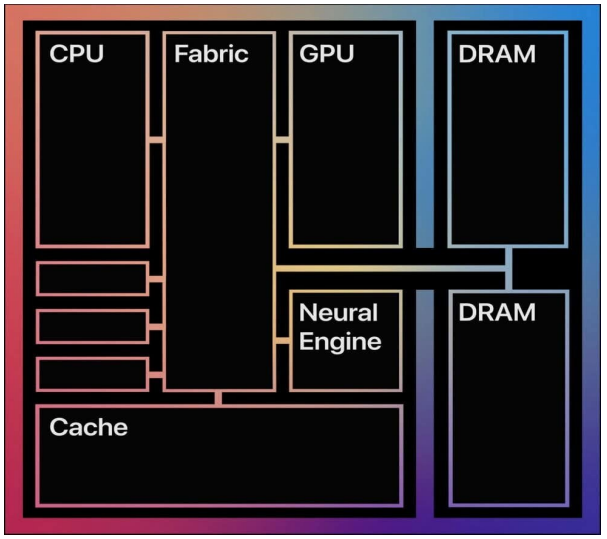


Case Study: Apple's M1 chip

- ISA is **ARMv8.5-A**
- The high-performance cores have an unusually large 192 KB of L1 **instruction cache** and 128 KB of L1 **data cache** and share a 12 MB L2 cache
- The M1 uses a 128-bit LPDDR4X SDRAM in a unified memory configuration **shared by all the components** of the processor.
- Integrated memory, GPU, NPUs, decoders, …

Question:

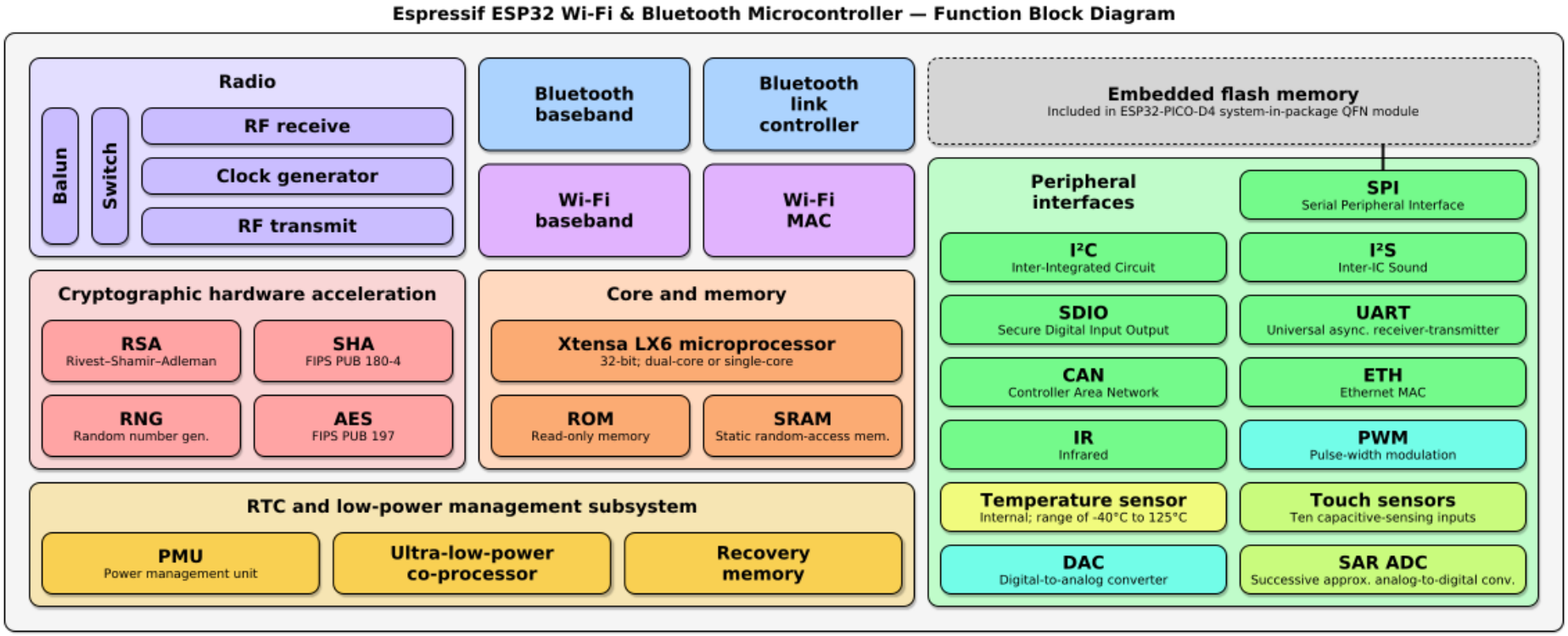
What architectural designs are adopted?



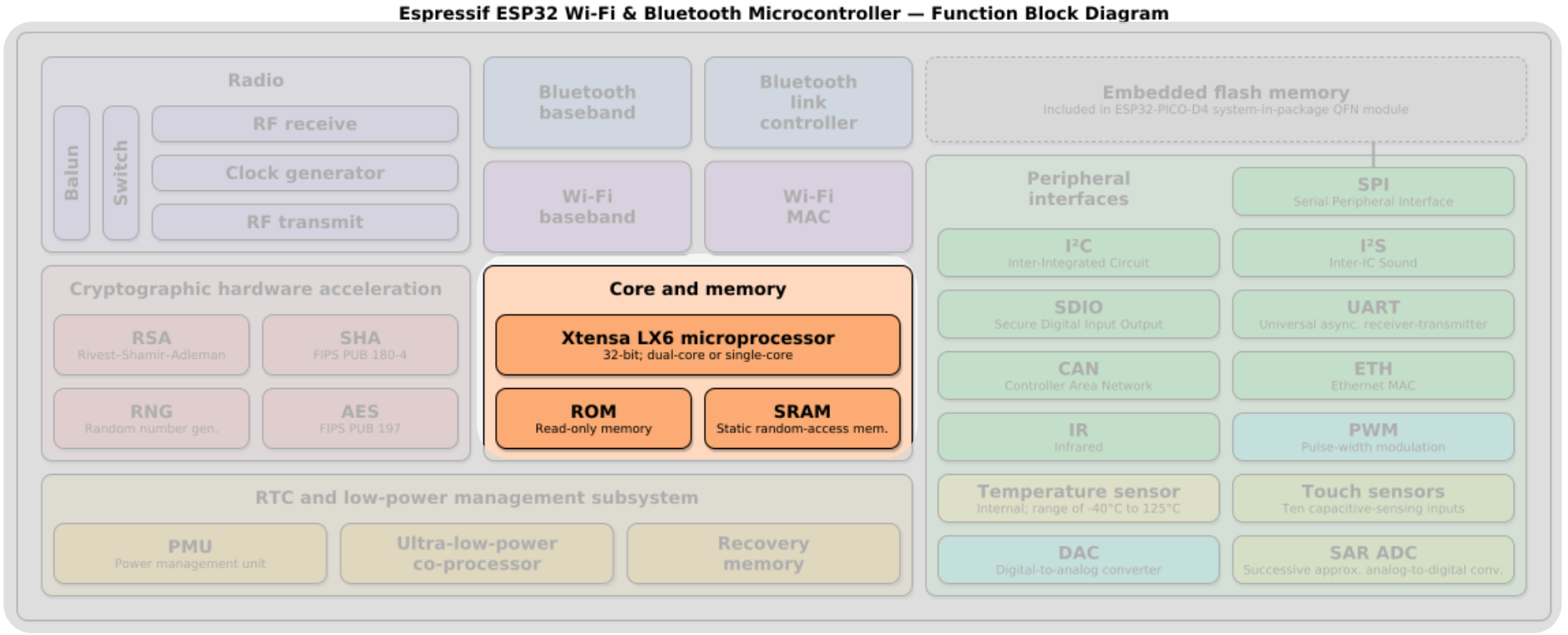
Apple M1 - Wikipedia

How about ESP32?

ESP32 MCU function block diagram



ESP32 MCU function block diagram

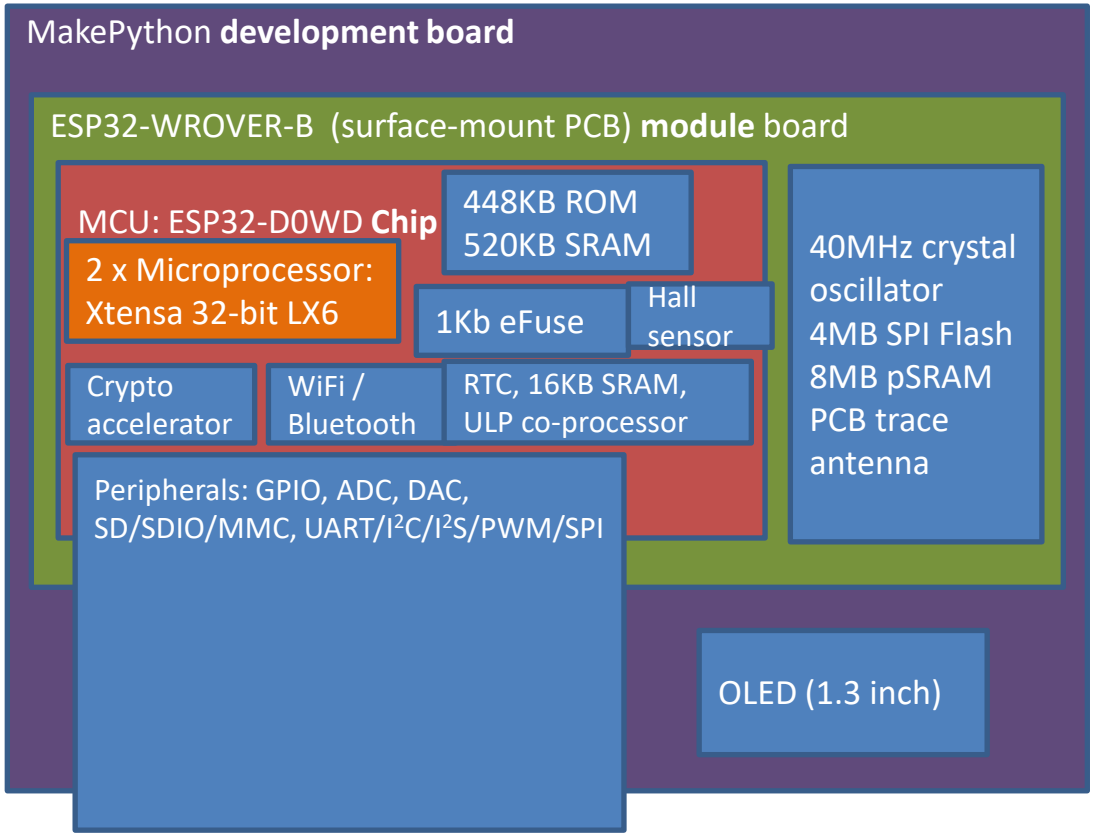


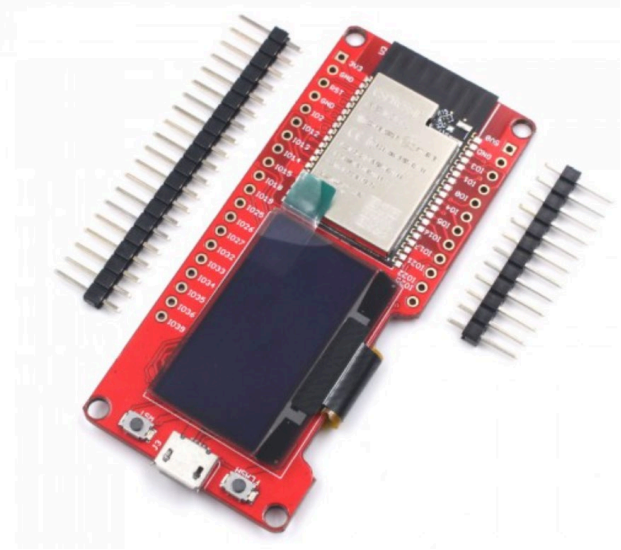
ESP32 processor

- Xtensa 32-bit LX6
  - IP core from Tensilica (part of Cadence design system)
  - RISC Instruction Set Architecture (ISA)
  - Harvard architecture
  - 4 GB (32-bit) address space for both data bus and instruction bus
  - 240 MHz
- Two LX6 cores in ESP32
  - Protocol core + application core

Integration levels

- System-on-a-Chip (SoC) *vs.* System-on-Module (SoM) *vs.* development board

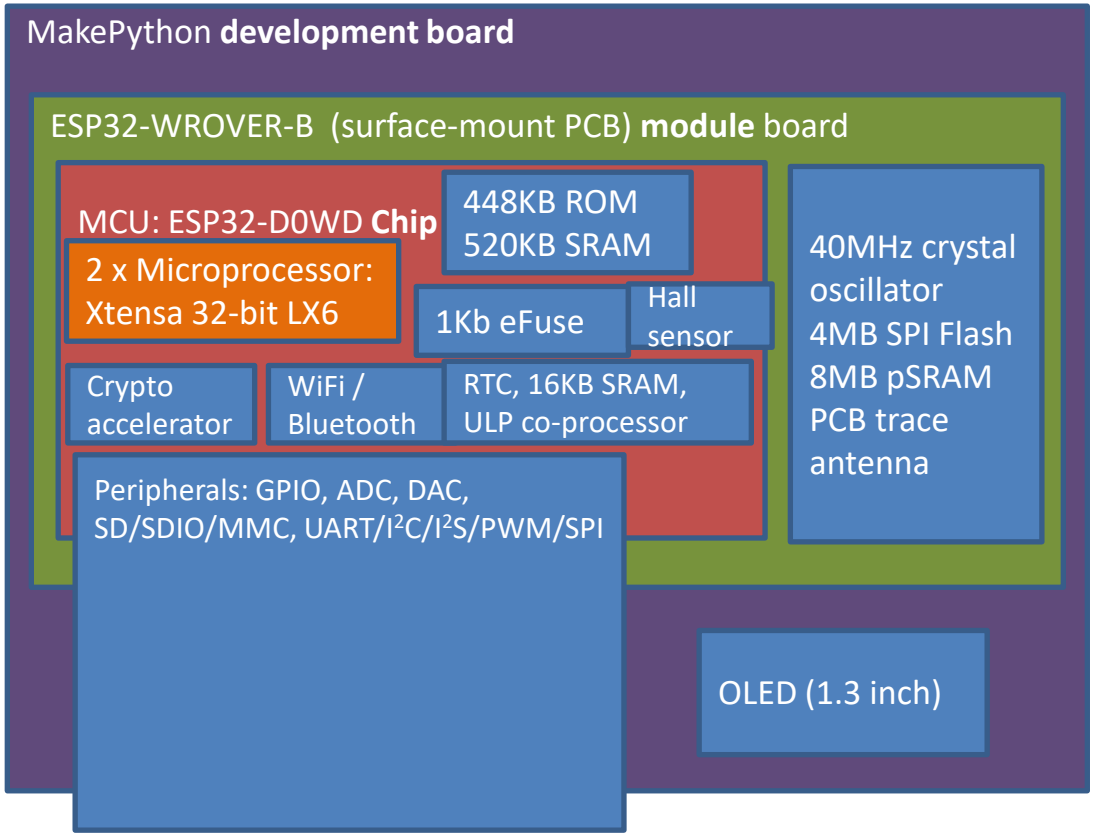




MakePython ESP32 Development Board

Integration levels

- System-on-a-Chip (SoC) *vs.* System-on-Module (SoM) *vs.* development board



Level 0: Xtensa 32-bit LX microprocessors (IP core from Cadence Tensilica)

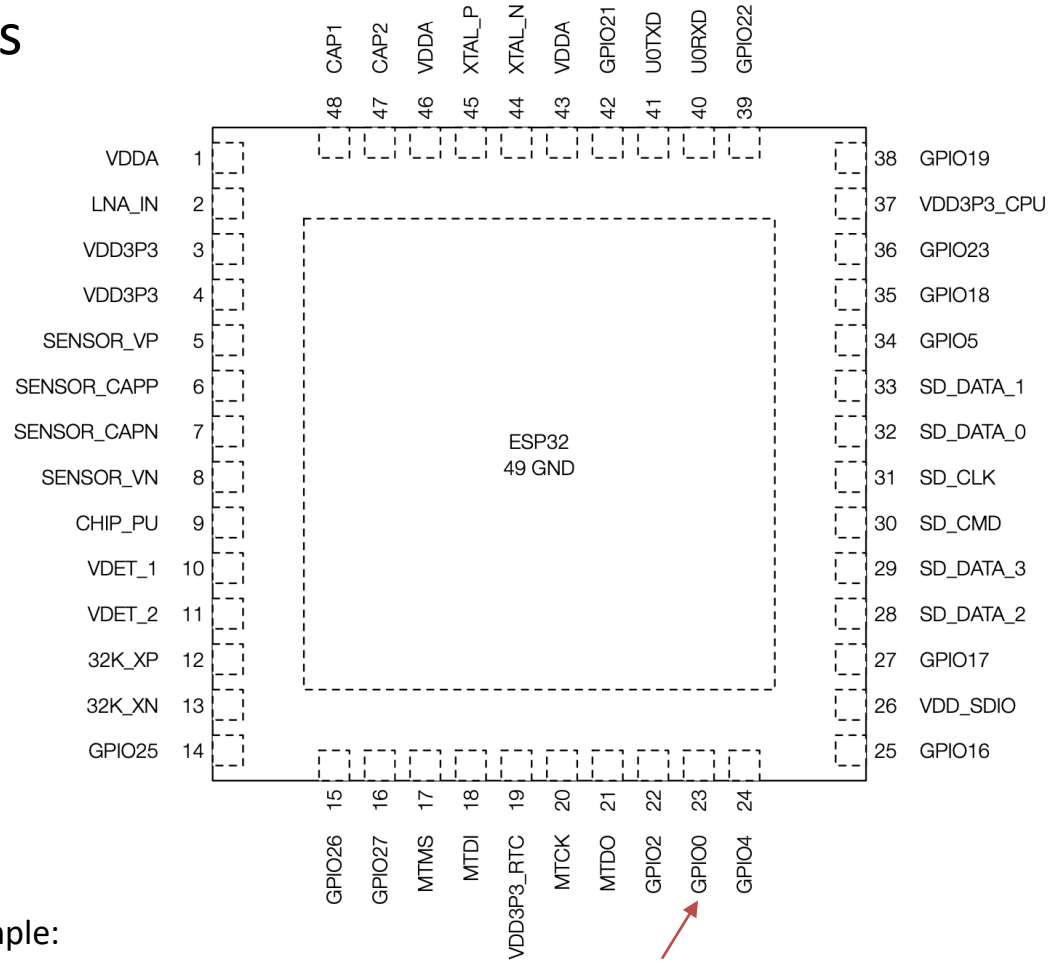
Level 1: ESP32-D0WD chip (by Espressif, TSMC 40nm technology, 5mm x 5mm, ~1 USD)

Level 2: WROVER-B module (by Espressif, 31.4mm x 18mm, ~5 USD)

Level 3: MakePython ESP32 development board (by Makerfabs, 70mm x 32.6mm, , ~15 USD)

ESP32 SoC pin definition
Src: esp32 datasheet

Pin layout: 49 pins



Pin description example:

GPIO0	23	I/O	GPIO0,	ADC2_CH1,	RTC_GPIO11,	TOUCH1,	EMAC_TX_CLK,	CLK_OUT1,

Pins can be reused

ESP32-WROVER module pin definition
Src: esp32-wrover datasheet

Pin layout: 39 pins

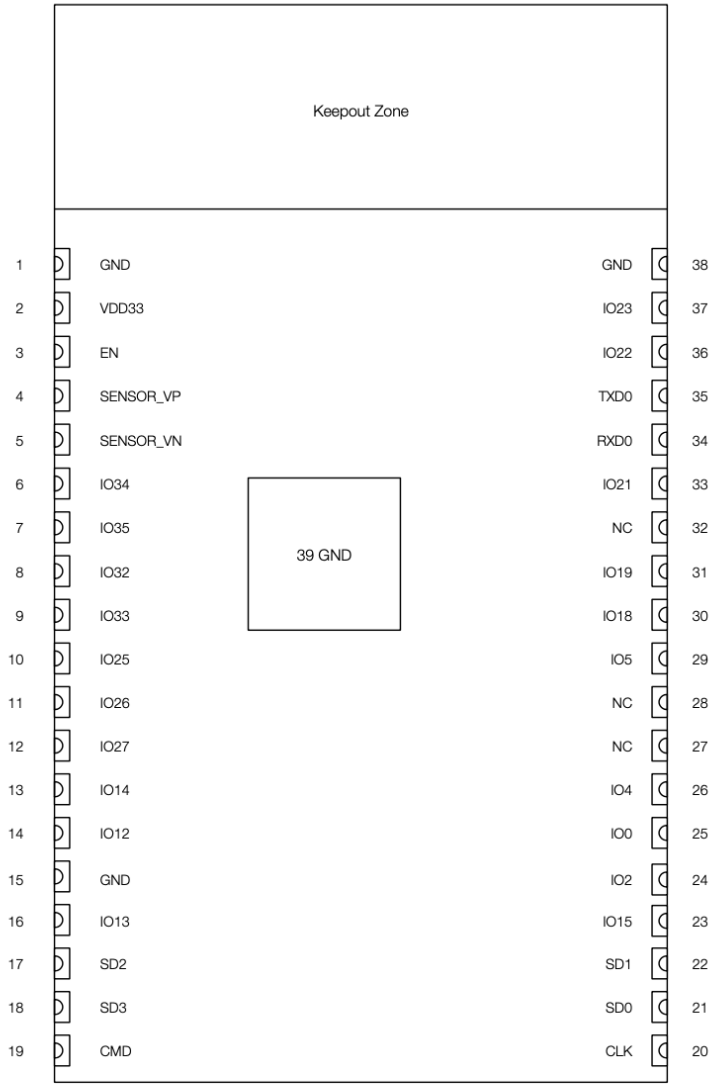
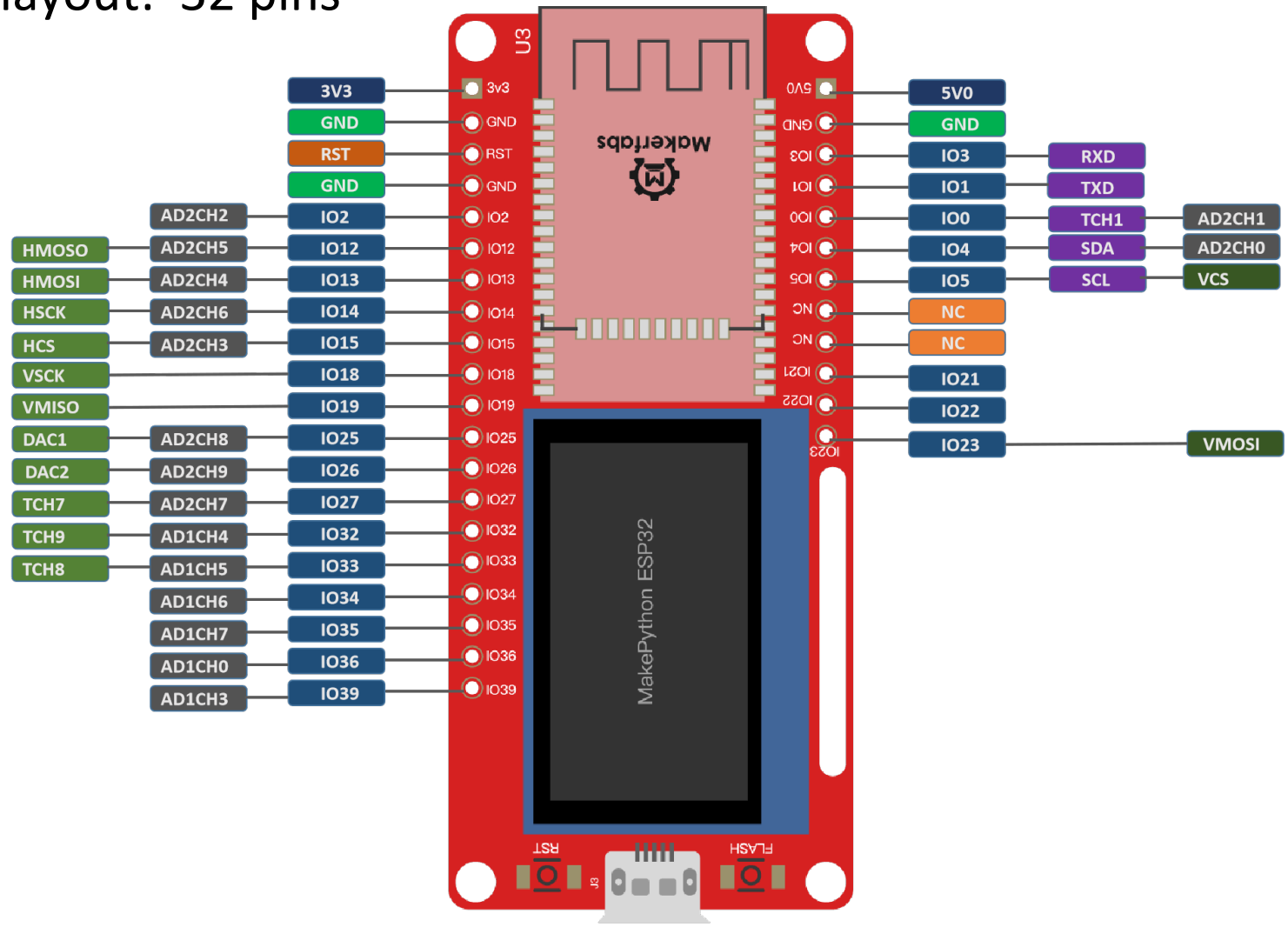


Figure 1: Pin Layout of ESP32-WROVER (Top View)

Makepython Pinout

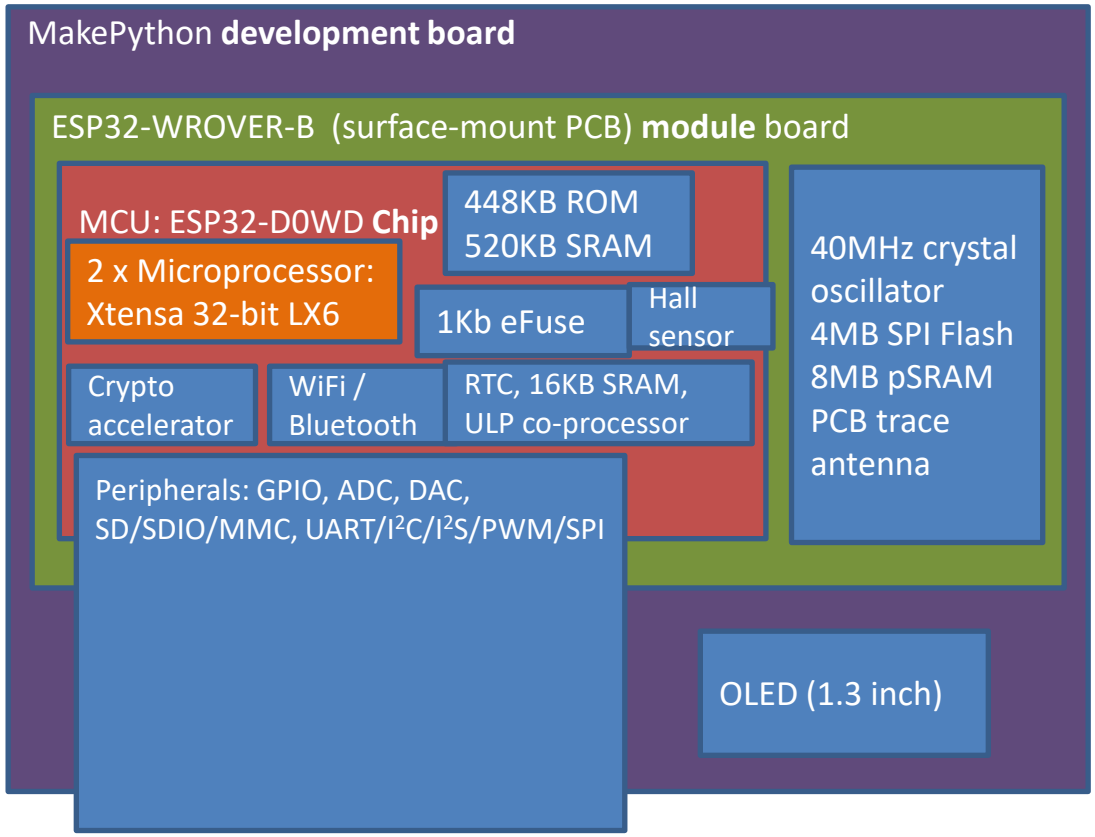
Pin layout: 32 pins



MCU: Evolution

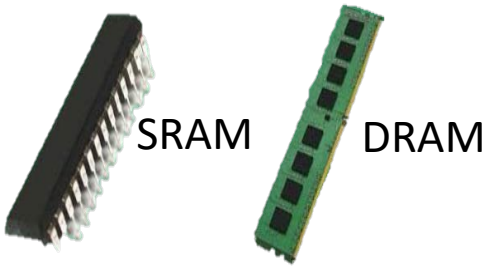
Buses	Example MCU Chip	Example boards	Spec
4 bit	TMS 1000 (Texas Instruments)	Dated 1974…	1KB ROM, 256b RAM
8 bit	AVR ATmega328P (Microchip)	Arduino Uno Rev3	20MHz 32 KB Flash, 2KB SRAM
16 bit	MSP430G2553 (Texas Instruments)	TI MSP-EXP430G2ET kit	16MHz 16KB Flash, 512B SRAM
32 bit	Kinetis KL26Z (NXP): ARM® Cortex-M0+, ARMv6-M	Micro:Bit	48MHz 128KB Flash, 16KB SRAM
	STM32F103C8T6 (STMicroelectronics): ARM® Cortex-M3 core, ARMv7-M	STM32duino (blue pill)	72MHz 128KB Flash, 20KB SRAM
	ESP32-D0WD (Espressif): Xtensa LX6	ESP32 development board	240MHz, 448KB ROM Off-chip Flash, 520KB SRAM

How about the memories/storage?



What's the difference between cache, main memory, and flash?

Memory: Types

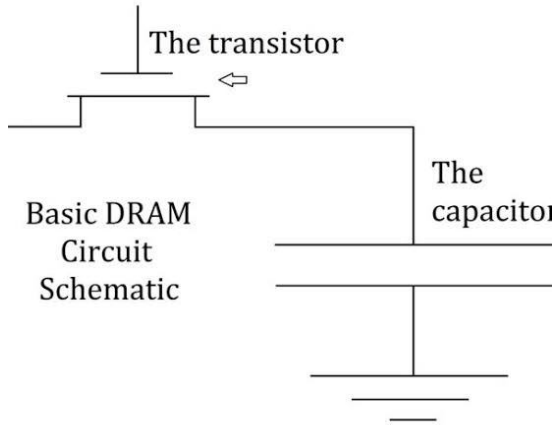


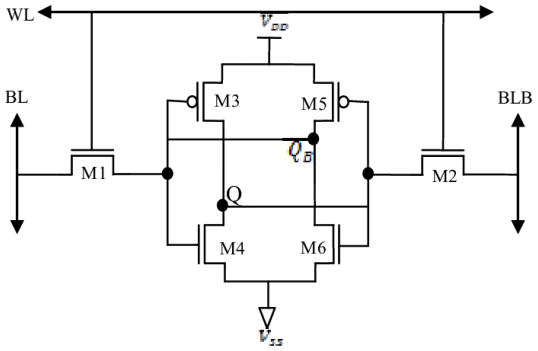
- Random access memory (RAM)
  - **Static RAM (SRAM)**: cache memory in CPUs, certain high-speed registers.
  - **Dynamic RAM (DRAM)**: main memory in computers, smartphones, tablets, etc.
    - **Pseudostatic RAM (PSRAM)**: IoT, mobile devices, MP3, game consoles.
  - Non-volatile RAM (NVRAM)
    - Ferroelectric RAM (FeRAM): smart card and RFID.

SRAM vs DRAM

- Basic Operation:
  - **DRAM**: periodic refreshing the memory to maintain the data it holds, where “dynamic” comes from.
  - **SRAM**: no reflesh cycle. Hold data as long as power applies.

- Structure:
  - **DRAM**: uses a **single transistor** and **a capacitor** to store each bit of data. Over time, the charge in the capacitor leaks, which necessitates the refresh cycles.
    - PSRAM is DRAM with in-built reflash circuit. Perform like SRAM.
  - **SRAM**: uses **six transistors** to store each bit of data, which makes it faster but also more expensive in terms of silicon area.





SRAM Schematic Diagram

SRAM vs DRAM

- Speed
  - **DRAM**: Generally slower than SRAM because of the refresh cycles required.
  - **SRAM**: Faster access times than DRAM.
- Power consumption
  - **DRAM**: More power during active use because of the refresh cycles, but has lower leakage (standby) power due to the capacitive storage.
  - **SRAM**: Generally consumes more power in standby mode since it requires power to hold data, but it can consume less power during active use as there are no refresh cycles.
- Density: DRAM higher than SRAM

Memory: Types

- Read-only memory (ROM)
  - Traditionally considered non-volative and “read-only”
  - Programmable read-only memory (PROM)
    - EEPROM (also E^2PROM): electrically erasable programmable read-only memory (byte-level erase block)
      - Flash memory (high speed / density, but larger erase block)

Memory Type	Volatility	Speed	Cost	Typical Use	Retains Data When Power Off?
DRAM	Volatile	Fast	Low	General-purpose storage	No
SRAM	Volatile	Very Fast	High	High-speed cache	No
PSRAM	Volatile	Moderate	Moderate	Expanded storage	No
NVRAM	Non-Volatile	Slow	Moderate	Settings, states	Yes
ROM	Non-Volatile	Slow	Low	Firmware	Yes
EEPROM	Non-Volatile	Moderate	Moderate	Config data, (erasable)	Yes

ESP32 chip Memory

- **Embedded Memory**
  – 448 KB Internal ROM (ROM0 384K + ROM1 64K)
  – 520 KB Internal SRAM (RAM0 192K + RAM1 128K + RAM2 200K)
  – 8 KB RTC FAST Memory (SRAM)
  – 8 KB RTC SLOW Memory (SRAM)
  – No on-chip Flash memory

- **External Memory (through QSPI)**
  – Supports up to 16 MB off-Chip Flash
  – Supports up to 8 MB off-Chip SRAM

Speed difference between on & off chip memory: an example

- ESP32:
  - Internal SRAM (520KB) is 32bit @ 240MHz max, so **960MByte/second**
  - Quad SPI connected PSRAM and Flash memory (they share the same QSPI) is 4-bit @ 80MHz, so **40MByte/second**

Question:

How to address speed difference?

ESP32 memory

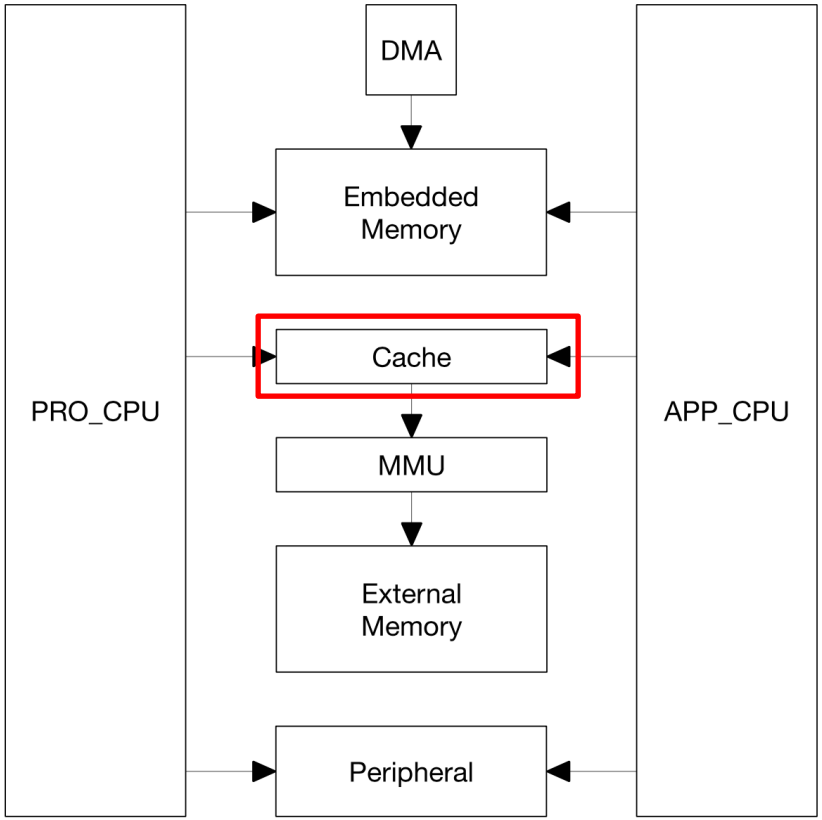


Figure 1: System Structure

Source: ESP32 Technical Reference Manual

Memory Address Mapping

Table 4: Embedded Memory Address Mapping

Bus Type	Boundary Address		Size	Target	Comment
	Low Address	High Address			
Data	0x3FF8_0000	0x3FF8_1FFF	8 KB	RTC FAST Memory	PRO_CPU Only
	0x3FF8_2000	0x3FF8_FFFF	56 KB	Reserved	-
Data	0x3FF9_0000	0x3FF9_FFFF	64 KB	Internal ROM 1	-
	0x3FFA_0000	0x3FFA_DFFF	56 KB	Reserved	-
Data	0x3FFA_E000	0x3FFD_FFFF	200 KB	Internal SRAM 2	DMA
Data	0x3FFE_0000	0x3FFF_FFFF	128 KB	Internal SRAM 1	DMA
Bus Type	Boundary Address		Size	Target	Comment
	Low Address	High Address			
Instruction	0x4000_0000	0x4000_7FFF	32 KB	Internal ROM 0	Remap
Instruction	0x4000_8000	0x4005_FFFF	352 KB	Internal ROM 0	-
	0x4006_0000	0x4006_FFFF	64 KB	Reserved	-
Instruction	0x4007_0000	0x4007_FFFF	64 KB	Internal SRAM 0	Cache
Instruction	0x4008_0000	0x4009_FFFF	128 KB	Internal SRAM 0	-
Instruction	0x400A_0000	0x400A_FFFF	64 KB	Internal SRAM 1	-
Instruction	0x400B_0000	0x400B_7FFF	32 KB	Internal SRAM 1	Remap
Instruction	0x400B_8000	0x400B_FFFF	32 KB	Internal SRAM 1	-
Instruction	0x400C_0000	0x400C_1FFF	8 KB	RTC FAST Memory	PRO_CPU Only
Bus Type	Boundary Address		Size	Target	Comment
	Low Address	High Address			
Data Instruction	0x5000_0000	0x5000_1FFF	8 KB	RTC SLOW Memory	-

Source: ESP32 Technical Reference Manual V4.3

Another view (including peripheral)

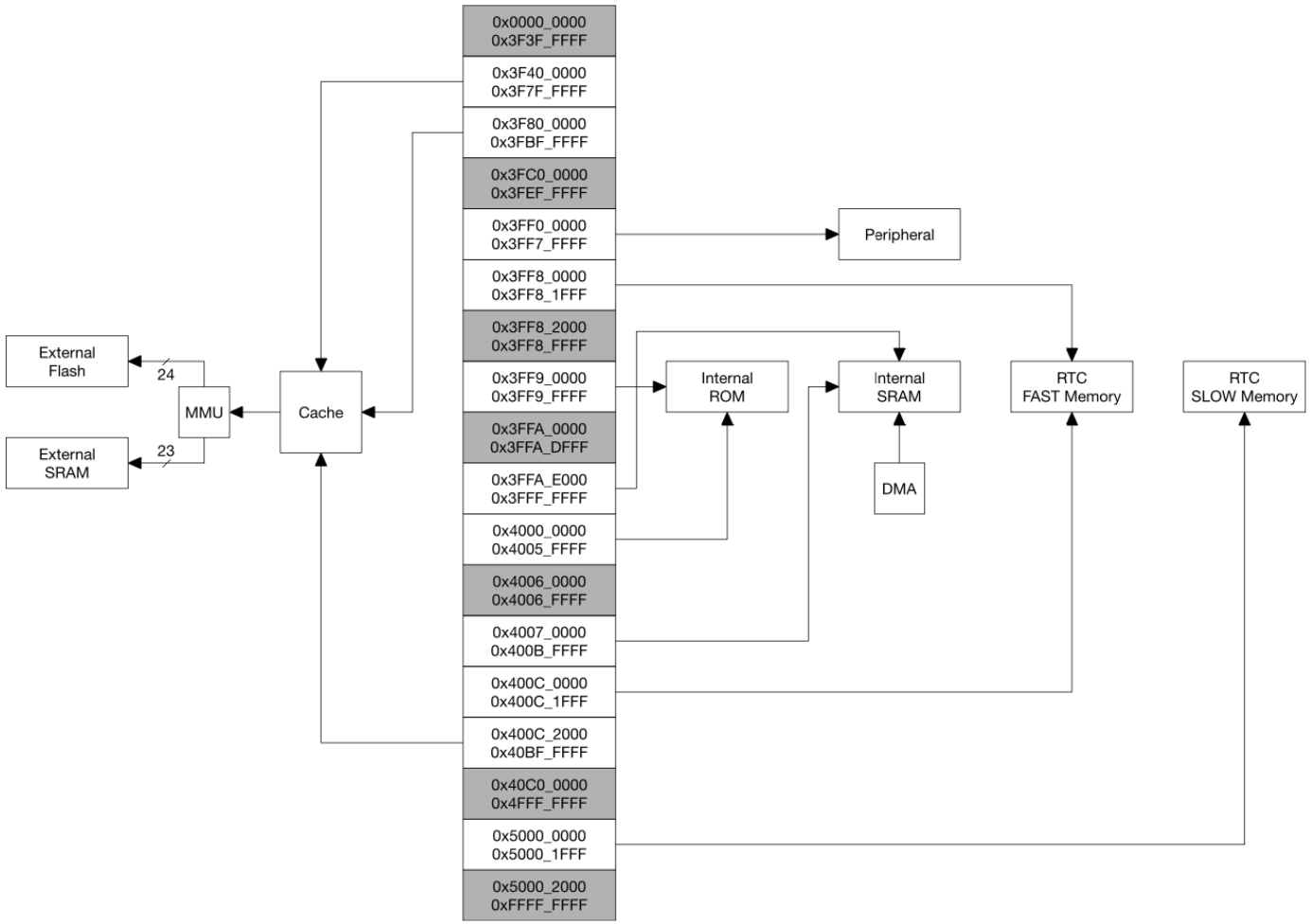


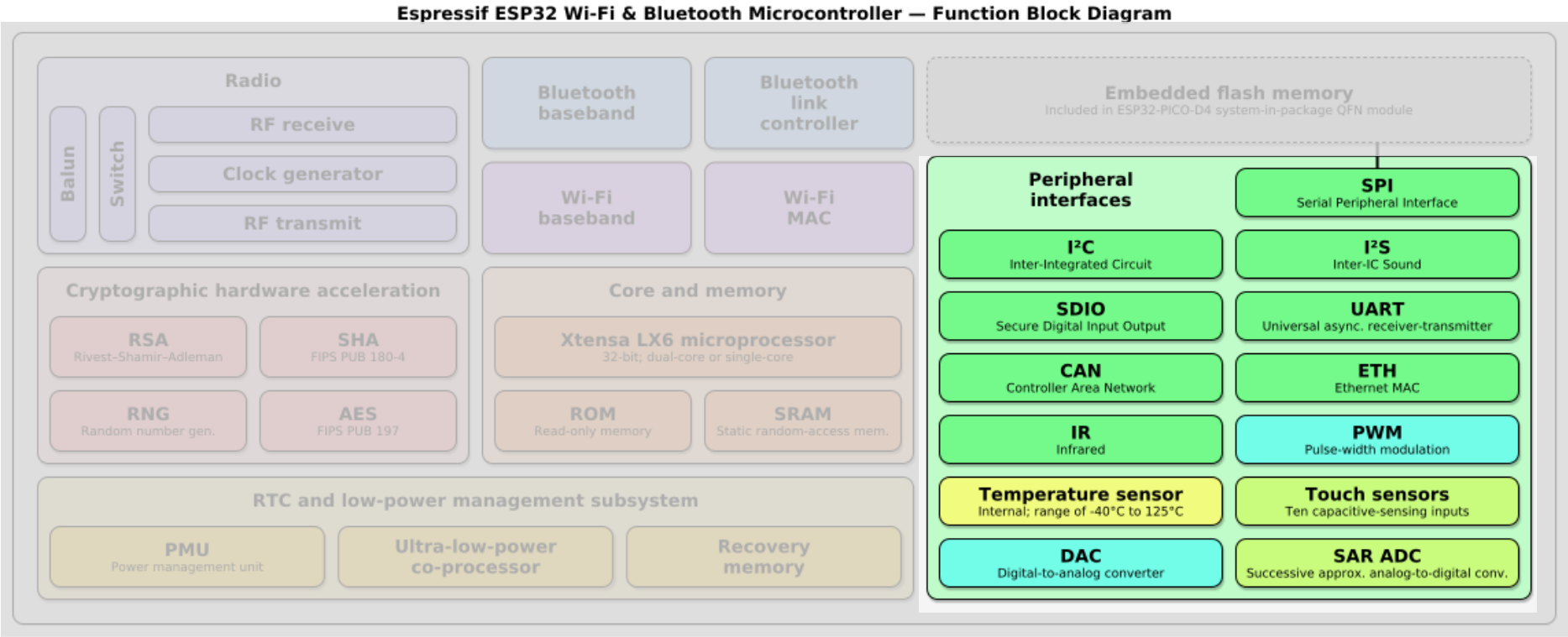
Figure 2: System Address Mapping

Source: ESP32 Technical Reference Manual

I/O Peripheral

Overview

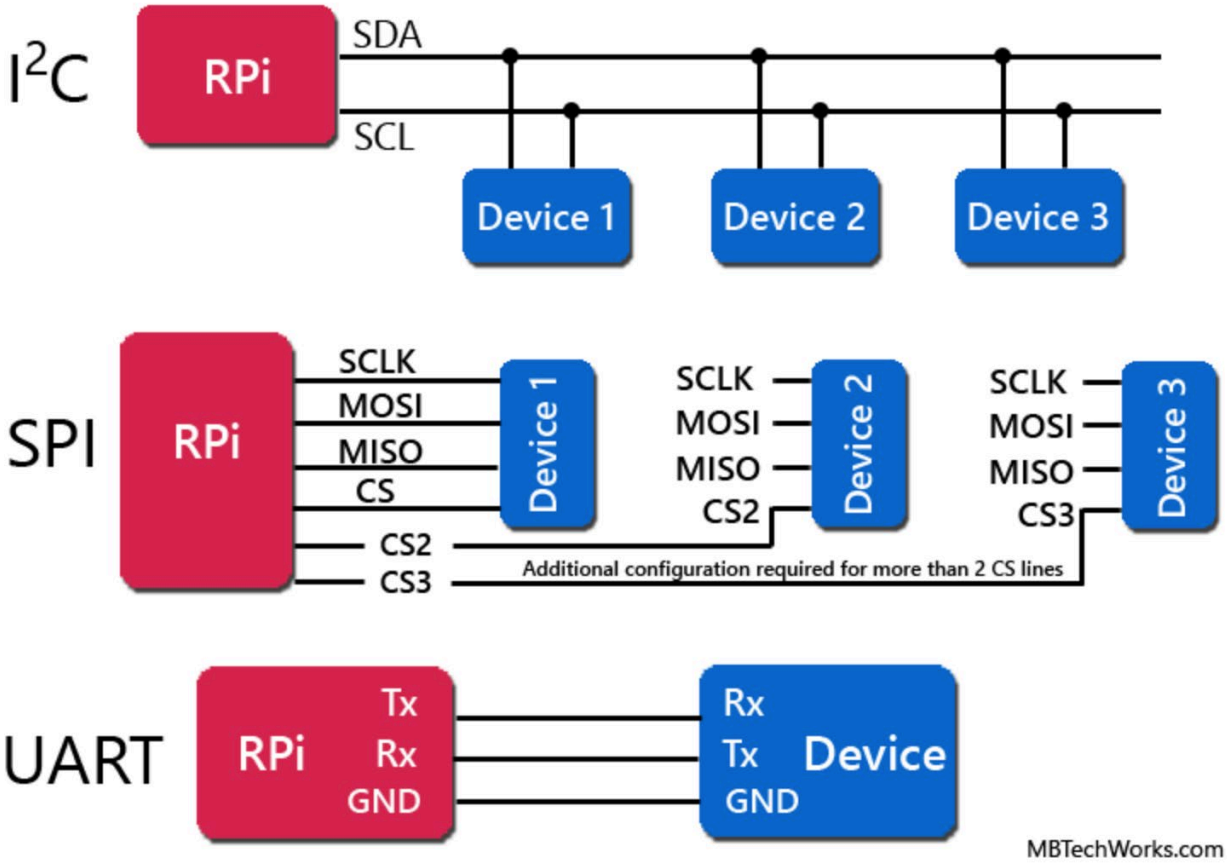
ESP32 MCU function block diagram



I/O Peripherals

- UART (Universal Asynchronous Receiver-Transmitter)
  - 3x in ESP32
- I^2C (inter-integrated circuit)
  - 2x in ESP32
- SPI (Serial Peripheral Interface)
  - 4x in ESP32

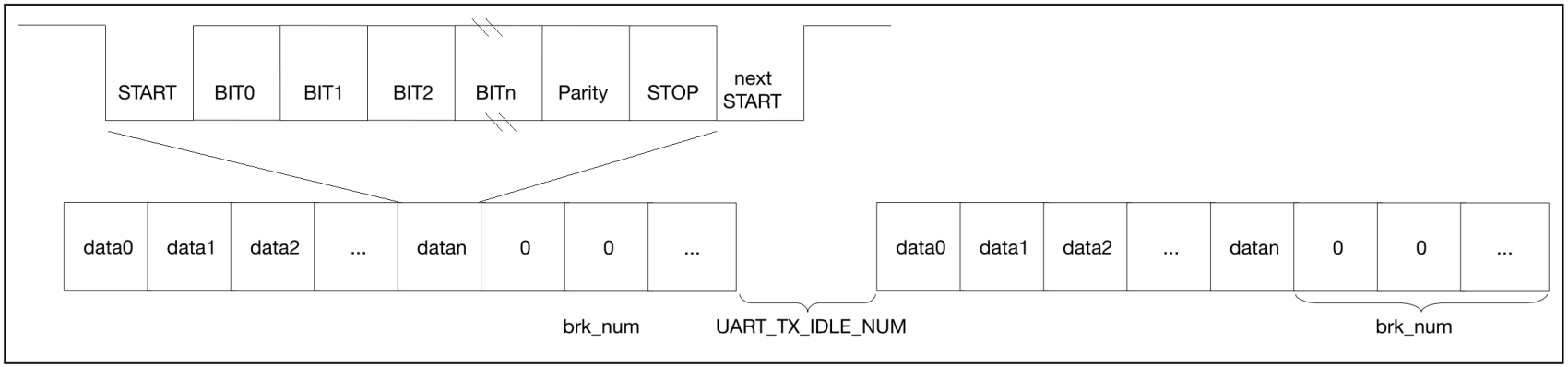
Comparison (src: MBTechWorks.com)



I2C, SPI, UART Connection Diagram

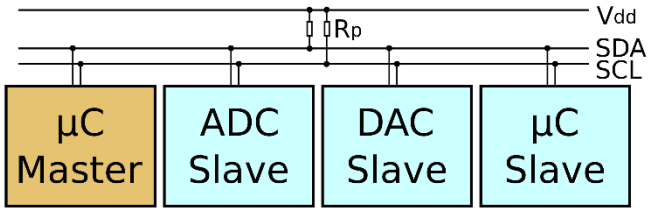
UART

- Universal Asynchronous Receiver-Transmitter
  - character-oriented data link
  - Asynchronous --- no clock wire needed
  - Need to agree on baud rate (e.g., 115200bps) and other settings (data length, STOP bits, parity bits)



I²C (inter-integrated circuit)

- I²C: 100kbps, 400kbps, 5Mbps (ultra fast-mode)
  - History: Philips Semiconductor (now NXP), 1982
  - Synchronous, multi-master, multi-slave, packet switched, serial comm bus
  - Design objectives:
    - simplicity, low manufacturing cost
  - 2 wires:
    - SDA (Serial Data Line), SCL (Serial Clock Line)
    - ACK for successful transmission
    - Open-drain design, pull-up resisters needed
  - Applications: low-speed DAC/ADC, small OLED/LCD display, hardware configure / monitor / diagnostic, access RTC and NVRAM



I²C addressing

- 7-bit addressing

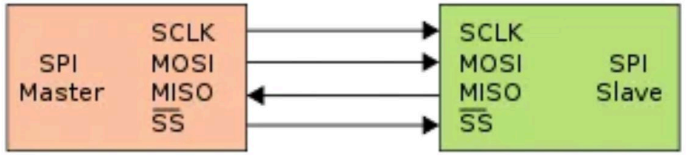
Field:	S	I²C address field							R/W'	A	I²C message sequences...	P
Type	Start	Byte 1								ACK	Byte X etc... Rest of the read or write message goes here	Stop
Bit position in byte X		7	6	5	4	3	2	1	0			
7-bit address pos		7	6	5	4	3	2	1				
Note		MSB						LSB	1 = Read 0 = Write			

- 10-bit addressing

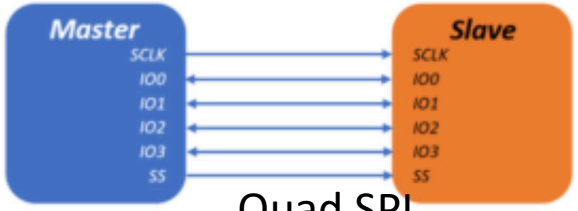
Field:	S	10-bit mode indicator					Upper addr		R/W'	A	Lower address field								I²C message sequences	P
Type	Start	Byte 1								ACK	Byte 2								Byte X etc. Rest of the read or write message goes here	Stop
Bit position in byte X		7	6	5	4	3	2	1	0		7	6	5	4	3	2	1	0		
Bit value		1	1	1	1	0	X	X	X		X	X	X	X	X	X	X	X		
10-bit address pos							10	9			8	7	6	5	4	3	2	1		
Note		Indicates 10-bit mode					MSB		1 = Read 0 = Write									LSB		

SPI (Serial Peripheral Interface)

- Synchronous serial communication
  - History: Motorola, 1980
  - Use: Flash memory, LCD



  - Full duplex, master-slave (single master)
    - Dual SPI, Quad SPI (these are half-duplex)
  - Four-wire serial bus: SCLK, MOSI, MISO, SS (Slave select)
    - No ACK for successful data transfer
  - Clock polarity & phase (mode #)
  - No top speed limit => 10Mbps or more



Quad SPI

Summary

- Inside an embedded system
  - Processor, memory, I/O
- MCU vs. MPU
  - MCU: integrated processor, memory, I/O
  - MPU: separated
- Inside the processor
  - Instruction set architecture (ISA): RISC vs. CISC
  - Instruction and data
    - Share memory and buses (von Neumann)
    - Separated memory and buses (Harvard)

Summary

- Memory
  - RAM vs ROM
  - RAM: SRAM vs. DRAM vs. PSRAM vs. NVRAM
  - ROM: traditionally considered “read-only” but now with EEPROM and Flash
- I/O peripheral
  - A glance at UART, I2C, SPI
  - More to go next Monday

Resources

- ESP32 Technical Reference Manual - Espressif Systems

https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf

- ESP-IDF (Espressif IoT Development Framework) https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/index.html

- ESP32 Microcontroller Hardware and Software

https://www.youtube.com/watch?v=xU-Kxhd0jEw

- ESP32 Teardown

https://www.youtube.com/watch?v=ZQXpfKDbXKs

- https://medium.com/the-esp-journal/esp32-memory-analysis-case-study-eacc75fe5431

- https://semiengineering.com/mpu-vs-mcu/