What is a Microcontroller (MCU)?

A Microcontroller Unit (MCU) is a compact integrated circuit designed to perform specific control tasks in embedded systems. It's often described as a small computer on a chip because it includes a processor (CPU), memory, and input/output peripherals—all integrated into one unit.

Key Components of a Microcontroller:

1. **CPU (Central Processing Unit)** ○ The processing core that executes instructions.
   - Responsible for controlling operations and performing calculations.
2. **Memory**
   - **Flash**: Non-volatile memory used to store program code. ○ **SRAM**: Volatile memory used to store temporary data during execution.
   - **EEPROM**: Non-volatile memory used to store data that must persist across resets.
3. **Clock System** ○ Generates timing signals that synchronize operations across the MCU.
   - Sources can be internal (e.g., RC oscillator) or external (e.g., crystal oscillator).
4. **Input/Output (I/O) Ports** ○ Interface with the external world.
   - Used to connect devices such as LEDs, buttons, sensors, and motors.
5. **Timers and Counters**
   - Used for measuring time intervals, generating delays, or counting external events.
6. **Analog Interfaces** ○ **ADC**: Converts analog input signals into digital values. ○ **DAC**: Converts digital data into analog output.
7. **Communication Interfaces** ○ Allow the MCU to communicate with other devices:

    - **UART**: Serial communication
    - **SPI**: Fast synchronous interface
    - **I2C**: Two-wire communication for multiple devices

8. **Interrupt System**
    - Enables the MCU to react immediately to important events (e.g., button press), improving efficiency compared to continuous polling.

Common MCU Applications:

- Home automation
- Robotics and drones
- Wearable devices (e.g., smartwatches)
- Automotive systems
- Industrial automation and control

MCU Architecture Overview

The architecture of a microcontroller refers to how its internal components are arranged and connected. It defines how the MCU processes data and communicates with internal and external modules.

Core Architectural Blocks:

1. **CPU** ○ Includes:
    - **ALU**: Performs arithmetic and logical operations.
    - **Control Unit**: Manages instruction decoding and data flow.
    - **Registers**: Provide fast access to frequently used data.
2. **Memory Units** ○ Flash (program storage) ○ SRAM (runtime variables) ○ EEPROM (non-volatile configuration data)

3. **System Clock** ○ Determines how fast instructions are executed.
4. **Bus System** ○ Includes:
    - **Data Bus**: Transfers data
    - **Address Bus**: Selects memory locations
    - **Control Bus**: Carries control signals
5. **I/O Ports** ○ Digital or analog interfaces used to interact with the environment.
6. **Timers/Counters** ○ Measure time, generate PWM signals, or count pulses.
7. **Peripheral Modules** ○ ADC, DAC, PWM modules for sensor input and actuator control.
8. **Communication Modules** ○ UART, SPI, I2C for external communication.
9. **Interrupt Controller** ○ Allows the CPU to pause current tasks and respond to high-priority events.

MCU Architecture Types:

Architecture Type	**Description**
Harvard	Separate memory for instructions and data.
Von Neumann	Shared memory for both instructions and data.
Modified Harvard	Combines advantages of both; commonly used today.

Clock System in MCUs

The clock system provides the timing signals that determine how fast instructions and operations are carried out.

Main Components:

Component	Description
Clock Source	Generates the base clock signal.
Clock Generator	Adjusts the frequency (via dividers, multipliers).
Clock Distribution Network	Routes the clock to CPU, buses, and peripherals.
Prescalers	Reduce clock speed for power savings or slower peripherals.
Multiplexers	Allow switching between clock sources.

Clock Source Types:

Source Type	Description
Internal RC Oscillator	Built-in, moderate accuracy, cost-effective.
External Crystal	High accuracy and stability.
PLL	Multiplies frequency to higher values.
LSE (Low-Speed External)	Typically 32.768 kHz for RTC.
HSI/HSE	High-speed internal or external oscillators.

Why Clock Configuration Matters:

- Affects **performance**, **power consumption**, and **timing requirements** of peripherals.
- Example: STM32 MCUs often use an 8 MHz internal oscillator and a PLL to achieve 72 MHz system clock.

Memory Mapping in MCUs

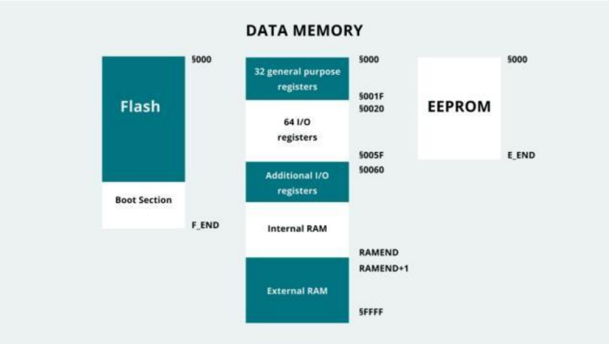
Memory mapping defines how different memory sections (Flash, RAM, registers, peripherals) are organized within the address space of the MCU.

Why It Matters:

- Helps locate code, variables, and peripherals.
- Enables **memory-mapped I/O** (peripherals accessed like variables).
- Critical for understanding stack behavior, DMA, and interrupt vectors.

Typical Memory Segments:

Segment	**Purpose**
Flash (PMEM)	Stores code, non-volatile
SRAM (DMEM)	Stores variables, volatile
EEPROM / ROM	Stores constant/configuration data
Peripheral Area	Registers to control hardware modules
Interrupt Table	Jump addresses to ISR functions, often at 0x00000000



Memory Access Example:

```
#define GPIOA_ODR (*(volatile unsigned int*)0x48000014)

GPIOA_ODR |= (1 << 5);  // Turn on LED connected to pin 5
```

MCU Bus Interfaces

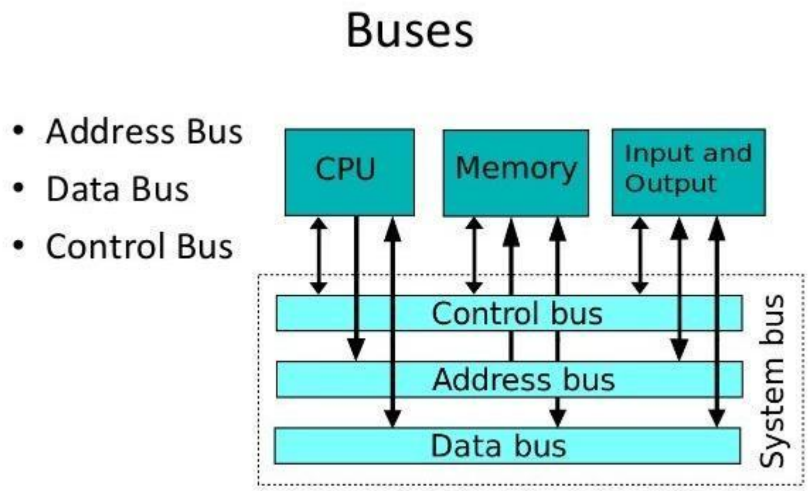
A bus is a group of lines that carry data, addresses, and control signals inside the MCU.

Common Bus Types:

Bus Type	**Function**	**Direction**
Data Bus	Transfers actual data	Bidirectional

Address Bus Specifies memory locations Unidirectional Control

Bus Carries read/write signals Unidirectional



Buses

- Address Bus
- Data Bus
- Control Bus

Internal Bus Architectures:

1. **Harvard**: Separate buses for data and instructions → faster but more complex.
2. **Von Neumann**: Single shared bus → simpler but slower.

AMBA Bus Standards (used in ARM MCUs):

Bus Type	Use Case	Speed
AHB	High-speed access (CPU, RAM, DMA)	Fast
APB	Low-speed peripherals (GPIO, UART)	Slower
AXI	Advanced high-speed systems	Very Fast

- **Bus bridges** allow communication between AHB and APB.
- AMBA makes system design modular and scalable.

Reading MCU Datasheets

A datasheet is your technical manual. Learn how to use it effectively to understand the microcontroller's features and limitations.

Key Sections to Focus On:

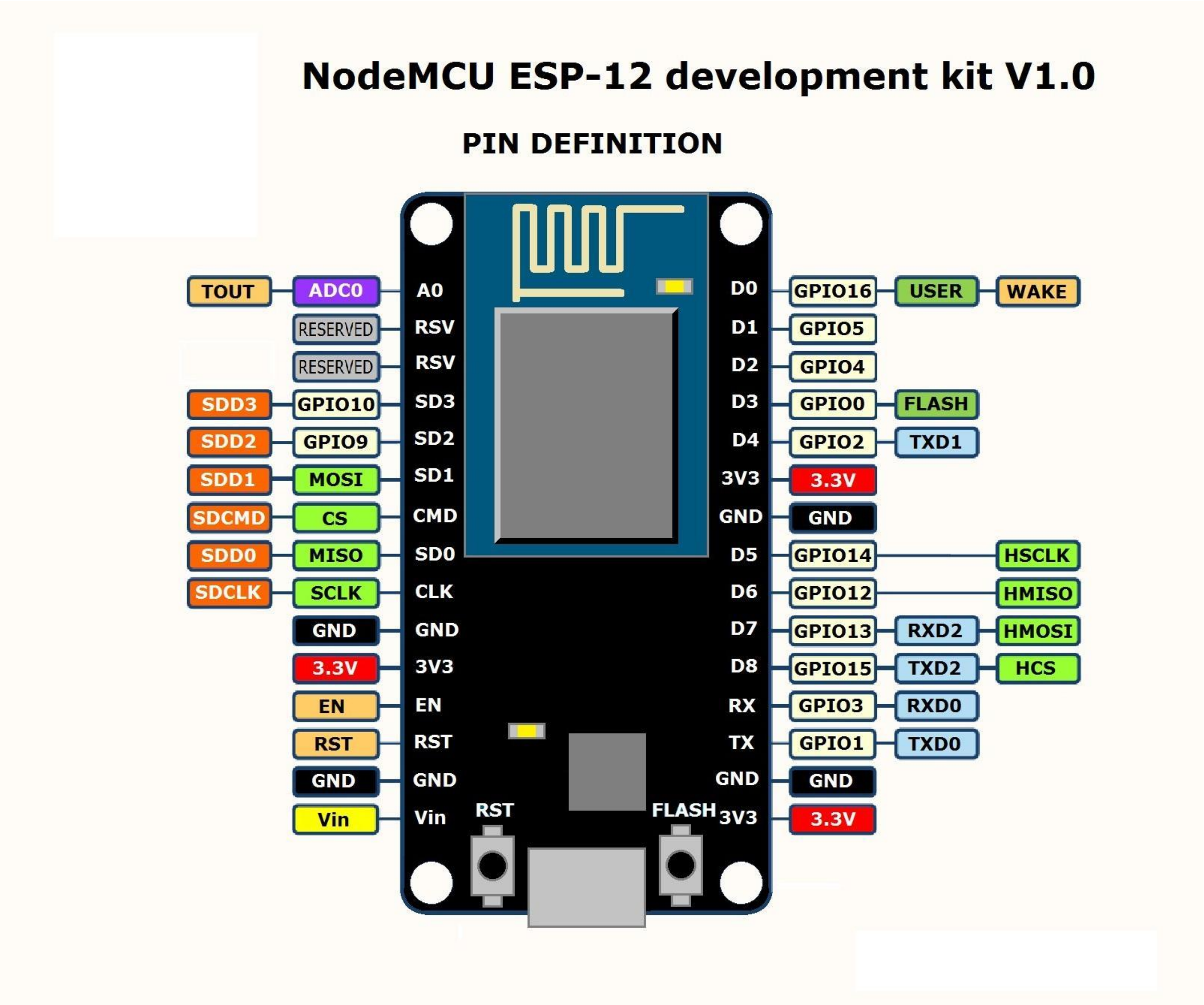
Section	Details
Features Summary	CPU speed, memory size, peripherals, etc.
Pin Configuration	Names, functions, and layout of MCU pins
Block Diagram	Visual overview of internal architecture
Memory Map	Address ranges of Flash, RAM, peripherals
Electrical Specs	Voltage ranges, power consumption
Clock Info	Clock sources, PLLs, and frequency options
Section	**Details**
GPIO Details	Input/output modes, pull-up/down, alternate functions
Communication	UART, SPI, I2C pin mappings and registers
Interrupts	NVIC, priorities, vector addresses

Power Modes	Sleep, stop, standby modes
Programming Options	Bootloader, debug interface (SWD, JTAG)

Tips:

- Always check the electrical section before powering the MCU.
- Use the reference manual for register-level configurations.
- Cross-reference with errata sheets to avoid known hardware bugs.

Example:



NodeMCU ESP-12 development kit V1.0

PIN DEFINITION