I²C Communication (SDA and SCL Pins)

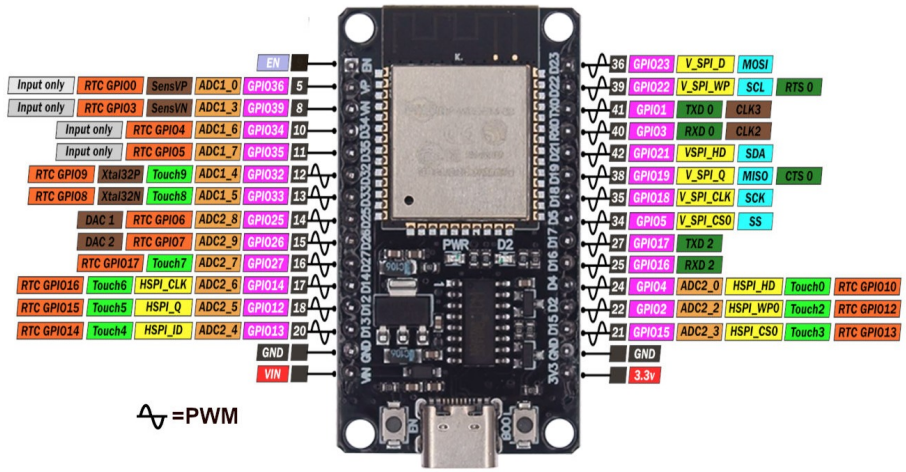

Figure 1: **ESP32 I²C Communication Pins**

The ESP32 supports **I²C (Inter-Integrated Circuit)** communication protocol, which is widely used to connect sensors, displays, and other peripherals with just two wires: **SDA (Serial Data)** and **SCL (Serial Clock)**.

Default I²C Pins on ESP32

- **SDA (GPIO21)** – Serial Data Line, used for data transfer.
- **SCL (GPIO22)** – Serial Clock Line, provides timing signal.

Characteristics

- Supports multiple devices using unique **7-bit or 10-bit addresses**.
- Speed modes supported: **Standard (100 kHz)**, **Fast (400 kHz)**, and **High-speed (up to 3.4 MHz)**.
- Requires **pull-up resistors** (typically 4.7k ohm) on SDA and SCL lines.
- Both SDA and SCL lines are **open-drain**, meaning devices only pull the line low, and resistors pull it high.

Applications

- Connecting sensors (e.g., BMP280, MPU6050, BME680).

- Interfacing with OLED/LCD displays.
- Communicating with RTC modules (DS3231, DS1307).
- Expanding GPIOs using I^2C expanders.

Example Usage (Arduino IDE)

The default I^2C pins are **GPIO21 (SDA)** and **GPIO22 (SCL)**. However, ESP32 allows changing these pins to any available GPIO.

```
#include <Wire.h>

void setup() {
  Wire.begin(21, 22); // SDA = 21, SCL = 22
  Serial.begin(115200);
  Serial.println("I2C Initialized!");
}

void loop() {
  // Communication with I2C devices
}
```

SPI Communication (MISO, MOSI, SCK, CS)

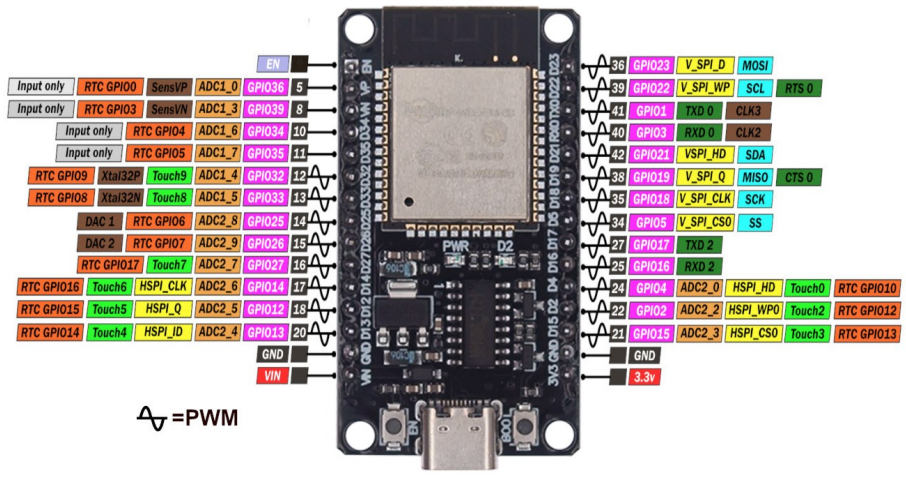

Figure 2: **ESP32 SPI Communication Pins**

The ESP32 supports the **SPI (Serial Peripheral Interface)** protocol, which is widely used for fast communication with peripherals such as displays, memory cards, and sensors.

Default SPI Pins on ESP32

The ESP32 has multiple SPI controllers (HSPI, VSPI, and FSPI depending on the board), and pins are **configurable**. Default commonly used pins are:

- **MISO (GPIO19)** – Master In Slave Out, data from slave to master.
- **MOSI (GPIO23)** – Master Out Slave In, data from master to slave.
- **SCK (GPIO18)** – Serial Clock, generated by the master.
- **CS / SS (GPIO5)** – Chip Select (or Slave Select), selects which slave to communicate with.

Characteristics

- Full-duplex communication (data can be sent and received simultaneously).
- Faster than I^2C, typical clock up to **80 MHz** (depending on the device).
- Supports multiple devices using separate CS pins for each slave.
- More pins required than I^2C (at least 4).

Applications

- Interfacing with **OLED/TFT displays** (e.g., ST7735, ILI9341).
- Reading/writing to **SD cards**.
- Using **external flash memory**.
- Communicating with high-speed sensors.

Example Usage (Arduino IDE)

Using default VSPI pins: MOSI = 23, MISO = 19, SCK = 18, CS = 5.

```
#include <SPI.h>

void setup() {
  SPI.begin(18, 19, 23, 5); // SCK, MISO, MOSI, CS
  Serial.begin(115200);
  Serial.println("SPI Initialized!");
}

void loop() {
  // SPI communication with devices
}
```

ESP32 SPI Buses: VSPI and HSPI

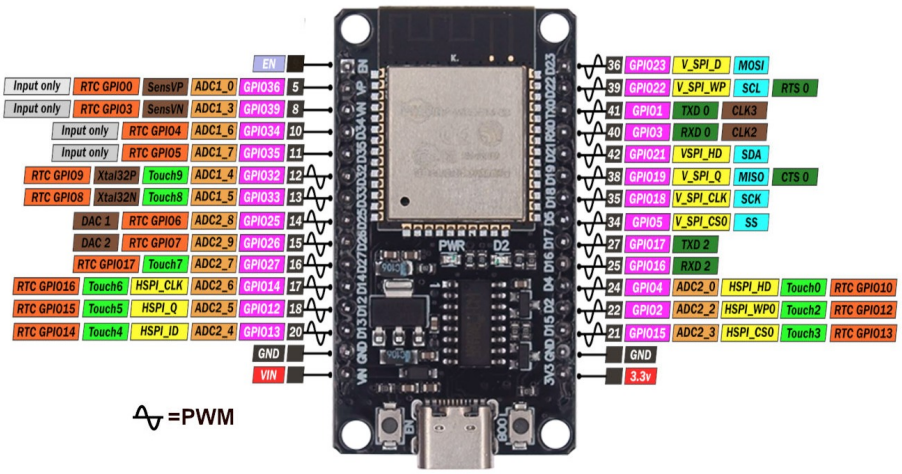

Figure 3: **ESP32 SPI Communication Pins**

The ESP32 features multiple SPI (Serial Peripheral Interface) controllers, including **VSPI** and **HSPI**. These are general-purpose SPI buses that can communicate with high-speed peripherals such as SD cards, displays, and sensors.

VSPI Bus

- **MOSI / D (Master Out Slave In / Data):** Sends data from ESP32 (master) to the peripheral (slave).

- **MISO / Q (Master In Slave Out / Data):** Receives data from the peripheral to ESP32.
- **SCK / CLK (Clock):** Provides the timing signal for synchronization.
- **CS / CS0 (Chip Select):** Selects which peripheral is active on the SPI bus.
- **HD (Hold / Data Hold):** Optional signal used in certain flash memories for pausing operations.
- **WP (Write Protect):** Optional signal used to protect flash memory from write operations.

HSPI Bus

- **MOSI / D:** Sends data from ESP32 to the slave device.
- **MISO / Q:** Receives data from the slave device.
- **SCK / CLK:** Serial clock for synchronization.
- **CS / CS0:** Selects the active slave.
- **HD (Hold):** Optional hold signal.
- **WP (Write Protect):** Optional write protection signal.

Key Points

- Both VSPI and HSPI are **hardware SPI controllers**, allowing simultaneous operation with different peripherals.
- Pin mapping for VSPI and HSPI is flexible and can be remapped to almost any GPIO.
- Useful for applications requiring multiple SPI devices without timing conflicts.
- Supports full-duplex communication and high-speed data transfer (up to 80 MHz typical).

Example Usage (Arduino IDE)

```
// Initialize VSPI and HSPI
#include <SPI.h>

SPIClass vspi(VSPI);
SPIClass hspi(HSPI);

void setup() {
  Serial.begin(115200);

  // VSPI default pins: MOSI=23, MISO=19, SCK=18, CS=5
  vspi.begin();

  // HSPI default pins: MOSI=13, MISO=12, SCK=14, CS=15
  hspi.begin(14, 12, 13, 15);

  Serial.println("VSPI and HSPI initialized!");
}

void loop() {
  // Communicate with peripherals using VSPI or HSPI
}
```