ESP32 Cheatsheet

A comprehensive cheat sheet covering essential information about the ESP32 microcontroller, its pinout, programming, and common functionalities. Perfect for quick reference during development.

ESP32 Pinout and Specifications

Pin Configuration

GPIO	General Purpose Input/Output pins. Configurable for various functions.
ADC1 & ADC2	Analog-to-Digital Converter channels for reading analog signals.
DAC1 & DAC2	Digital-to-Analog Converter channels for generating analog signals.
I2C SDA & SCL	Serial Data (SDA) and Serial Clock (SCL) pins for I2C communication.
SPI MOSI, MISO, CLK, CS	Master Out Slave In (MOSI), Master In Slave Out (MISO), Clock (CLK), Chip Select (CS) pins for SPI communication.
UART TX & RX	Transmit (TX) and Receive (RX) pins for UART communication.
EN	Enable pin. Pull low to disable/reset the ESP32.
RST	Reset pin. Active low reset.

Key Specifications

CPU	Tensilica LX6 Dual-Core Processor
Clock Speed	Up to 240 MHz
Flash Memory	4MB or more (external)
SRAM	520 KB
Wi-Fi	802.11 b/g/n
Bluetooth	Bluetooth v4.2 BR/EDR and BLE
Operating Voltage	3.3V
Power Consumption	Varies, ~80mA typical

Power Pins

VCC	3.3V power supply
GND	Ground
VIN	Voltage Input (e.g., from USB or battery)

Programming with Arduino IDE

Setting up Arduino IDE

1. **Install Arduino IDE:** Download and install the Arduino IDE from the official Arduino website.
2. **Add ESP32 Board Support:**
   - Go to `File > Preferences`.
   - Add the following URL to `Additional Board Manager URLs`:

     ```
     https://dl.espressif.com/dl/package_esp32_index.json
     ```

3. **Install ESP32 Boards:**
   - Go to `Tools > Board > Boards Manager...`
   - Search for `ESP32` and install the `ESP32 by Espressif Systems` package.
4. **Select ESP32 Board:**
   - Go to `Tools > Board` and select your specific ESP32 board (e.g., `ESP32 Dev Module`).

Basic Code Structure

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
}

void loop() {
  // put your main code here, to run repeatedly:
  Serial.println("Hello, ESP32!");
  delay(1000);
}
```

Common Functions

`pinMode(pin, mode)`	Sets the specified pin to either `INPUT`, `OUTPUT`, or `INPUT_PULLUP`.
`digitalWrite(pin, value)`	Writes either `HIGH` or `LOW` to a digital pin.
`digitalRead(pin)`	Reads the value (`HIGH` or `LOW`) from a digital pin.
`analogRead(pin)`	Reads the analog value from the specified analog pin (ADC).
`analogWrite(pin, value)`	Writes an analog value (PWM wave) to a pin (DAC).
`Serial.begin(baudRate)`	Initializes serial communication at the specified baud rate.
`Serial.print(data)`	Prints data to the serial port.

Wi-Fi and Bluetooth Connectivity

Connecting to Wi-Fi

```
#include <WiFi.h>

const char* ssid = "your_SSID";
const char* password = "your_PASSWORD";

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("\nWiFi connected");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
}

void loop() {
  // Your code here
}
```

Using Bluetooth

```
#include <BluetoothSerial.h>

BluetoothSerial SerialBT;

void setup() {
  Serial.begin(115200);
  SerialBT.begin("ESP32_Bluetooth"); //
Bluetooth device name
  Serial.println("The device started, now you
can pair it with bluetooth!");
}

void loop() {
  if (SerialBT.available()) {
    Serial.write(SerialBT.read());
  }
  delay(20);
}
```

Wi-Fi Functions

`WiFi.begin(ssid, password)`	Connects to the specified Wi-Fi network.
`WiFi.status()`	Returns the current Wi-Fi connection status.
`WiFi.localIP()`	Returns the IP address of the ESP32.
`WiFi.macAddress()`	Returns the MAC address of the ESP32.

Bluetooth Functions

`SerialBT.begin(deviceName)`	Initializes Bluetooth serial communication with a given device name.
`SerialBT.available()`	Returns the number of bytes available to read from the Bluetooth serial port.
`SerialBT.read()`	Reads a byte from the Bluetooth serial port.
`SerialBT.write(data)`	Writes data to the Bluetooth serial port.

Deep Sleep and Power Management

Entering Deep Sleep

```
#include <esp_sleep.h>

#define uS_TO_S_FACTOR 1000000  /* Conversion
factor for micro seconds to seconds */
#define TIME_TO_SLEEP  5        /* Time ESP32
will go to sleep (in seconds) */

void setup() {
  Serial.begin(115200);
  Serial.println("Going to sleep now");
  esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP
* uS_TO_S_FACTOR);
  esp_deep_sleep_start();
  Serial.println("This will never be
printed");
}

void loop() {
  // This is not going to be executed
}
```

Wake-up Sources

Timer Wakeup	Wake up after a specified time interval.
External Wakeup (GPIO)	Wake up when a specific GPIO pin changes state (high or low).
Touch Wakeup	Wake up when a touch sensor is activated.

Power Saving Tips

- **Use Deep Sleep:** Utilize deep sleep mode to minimize power consumption during idle periods.
- **Disable Unused Peripherals:** Turn off Wi-Fi and Bluetooth when not in use.
- **Adjust CPU Frequency:** Reduce the CPU clock frequency to save power.
- **Optimize Code:** Ensure efficient code execution to minimize processing time.
- **Use Light Sleep:** For shorter idle periods, use light sleep mode for faster wake-up times.