ESP32-S3 Command Set

The command set of ESP32-S3 is mainly implemented through its supported firmware, such as Arduino IDE, Espressif IDF (IoT Development Framework) or AT command set. Here are some commonly used commands and libraries:

1. Arduino IDE

- Basic settings

cpp
```
void setup() {
pinMode(LED_BUILTIN, OUTPUT); // Set the built-in LED to output mode
}

void loop() {
digitalWrite(LED_BUILTIN, HIGH); // Turn on the LED
delay(1000); // Delay 1 second
digitalWrite(LED_BUILTIN, LOW); // Turn off the LED
delay(1000); // Delay 1 second
}
```

- WiFi connection

cpp
```
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(1000);
Serial.println("Connecting to WiFi...");
}

Serial.println("Connected to WiFi");
}

void loop() {
// WiFi connected logic
}
```

2. Espressif IDF (IoT Development Framework)

- Basic settings

c

```
void app_main(void) {
gpio_pad_select_gpio(GPIO_NUM_2);
gpio_set_direction(GPIO_NUM_2, GPIO_MODE_OUTPUT);

while (1) {
gpio_set_level(GPIO_NUM_2, 1);
vTaskDelay(1000 / portTICK_PERIOD_MS);
gpio_set_level(GPIO_NUM_2, 0);
vTaskDelay(1000 / portTICK_PERIOD_MS);
}
}
```

- WiFi connection

c

```
void app_main(void) {
esp_log_level_set("wifi", ESP_LOG_NONE); // disable wifi driver logging

wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
esp_wifi_init(&cfg);
esp_wifi_set_mode(WIFI_MODE_STA);

wifi_config_t sta_config = {
.sta = {
.ssid = "your_SSID",
.password = "your_PASSWORD",
.bssid_set = false
}
};

esp_wifi_set_config(ESP_IF_WIFI_STA, &sta_config);
esp_wifi_start();
esp_wifi_connect();

// Add your connection logic here
}
```

3. AT Command Set

- WiFi Connection

```
AT+CWMODE=1 // Set to Station Mode
AT+CWJAP="your_SSID","your_PASSWORD" // Connect to WiFi Network
AT+CIFSR // Get Local IP Address
```

These commands and code examples cover the basic operation, WiFi connection and pin assignment of ESP32-S3. The specific pin functions and command sets may vary depending on the development board. It is recommended to refer to the user manual of the specific development board and the official documentation of Espressif.

4. MicroPython

MicroPython is a stripped-down version of Python designed for microcontrollers. ESP32-S3 supports running MicroPython.

- Basic setup

```
python
from machine import Pin
from time import sleep

led = Pin(2, Pin.OUT)

while True:
led.value(1)
sleep(1)
led.value(0)
sleep(1)
```

- WiFi connection

```
python
import network

ssid = 'your_SSID'
password = 'your_PASSWORD'

wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect(ssid, password)
```

```
while not wlan.isconnected():
pass

print('network config:', wlan.ifconfig())
```

5. ESP-NOW

ESP-NOW is a low-power wireless communication protocol for ESP32 chips for fast data transmission.

- Initialize ESP-NOW

```
cpp
include <esp_now.h>
include <WiFi.h>

void setup() {
Serial.begin(115200);
WiFi.mode(WIFI_STA);
if (esp_now_init() != ESP_OK) {
Serial.println("Error initializing ESP-NOW");
return;
}
}

void loop() {
// Add ESP-NOW communication logic here
}
```

6. Bluetooth Low Energy (BLE)

ESP32-S3 supports BLE and can communicate as a BLE device.

- Basic setup

```
cpp
include <BLEDevice.h>
include <BLEServer.h>
include <BLEUtils.h>
include <BLE2902.h>

BLEServer pServer = NULL;
BLECharacteristic pCharacteristic = NULL;
bool deviceConnected = false;
```

```
void setup() {
Serial.begin(115200);
BLEDevice::init("ESP32_S3_BLE");
pServer = BLEDevice::createServer();
pServer->setCallbacks(new MyServerCallbacks());

BLEService pService = pServer->createService(SERVICE_UUID);
pCharacteristic = pService->createCharacteristic(
CHARACTERISTIC_UUID,
BLECharacteristic::PROPERTY_READ |
BLECharacteristic::PROPERTY_WRITE
);
pCharacteristic->setValue("Hello World");
pService->start();
BLEAdvertising pAdvertising = BLEDevice::getAdvertising();
pAdvertising->start();
}

void loop() {
// BLE communication logic here
}
```

7. FreeRTOS

ESP32-S3 has a built-in FreeRTOS real-time operating system, allowing users to create multi-tasking applications.

- Basic settings

```
c
void blinkTask(void pvParameter) {
gpio_pad_select_gpio(GPIO_NUM_2);
gpio_set_direction(GPIO_NUM_2, GPIO_MODE_OUTPUT);

while(1) {
gpio_set_level(GPIO_NUM_2, 1);
vTaskDelay(1000 / portTICK_PERIOD_MS);
gpio_set_level(GPIO_NUM_2, 0);
vTaskDelay(1000 / portTICK_PERIOD_MS);
}
}

void app_main() {
```

```
xTaskCreate(&blinkTask, "blinkTask", 1024, NULL, 5, NULL);
}
```

8. AT command set (continued)

- Get version information

```
AT+GMR
```

- List available WiFi networks

```
AT+CWLAP
```

- Establish TCP connection

```
AT+CIPSTART="TCP","example.com",80
```

- Send data

```
AT+CIPSEND=length
```

9. HTTP client

ESP32-S3 can use HTTP protocol for network communication. Here is an example of HTTP client using Arduino IDE:

- HTTP GET request

```
cpp
include <WiFi.h>
include <HTTPClient.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(1000);
Serial.println("Connecting to WiFi...");
}

Serial.println("Connected to WiFi");
}

void loop() {
if (WiFi.status() == WL_CONNECTED) {
HTTPClient http;
http.begin("http://example.com");
int httpCode = http.GET();

if (httpCode > 0) {
String payload = http.getString();
Serial.println(payload);
}

http.end();
}
delay(10000);
}
```

10. MQTT Client

ESP32-S3 can act as an MQTT client to communicate with the MQTT server.

- MQTT Publish/Subscribe

cpp

```
#include <WiFi.h>
#include <PubSubClient.h>

const char* ssid = "your_SSID";
const char* password = "your_PASSWORD";
const char* mqtt_server = "broker.hivemq.com";

WiFiClient espClient;
PubSubClient client(espClient);

void setup() {
Serial.begin(115200);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(1000);
Serial.println("Connecting to WiFi...");
}

client.setServer(mqtt_server, 1883);
client.setCallback(callback);
}

void callback(char* topic, byte* payload, unsigned int length) {
Serial.print("Message arrived [");
Serial.print(topic);
Serial.print("] ");
for (unsigned int i = 0; i < length; i++) {
Serial.print((char)payload[i]);
}
Serial.println();
}

void reconnect() {
while (!client.connected()) {
Serial.print("Attempting MQTT connection...");
if (client.connect("ESP32Client")) {
Serial.println("connected");
client.subscribe("test/topic");
} else {
Serial.print("failed, rc=");
Serial.print(client.state());
delay(5000);
}
}

}

void loop() {
if (!client.connected()) {
reconnect();
}
client.loop();
client.publish("test/topic", "Hello world");
delay(10000);
}
```