Q1: What is RTC-GPIO?

There is RTC GPIO support on the ESP32. The GPIOs routed to the RTC low-power subsystem can be used when the ESP32 is in deep sleep. These RTC GPIOs can be used to wake up the ESP32 from deep sleep when the Ultra Low Power (ULP) co-processor is running.

Q2: What is * (GPIO) on silk label?

GPIOs 34 to 39 are GPIs – input only pins. These pins don't have internal pull-up or pull-down resistors. They can't be used as outputs, so use these pins only as inputs:

- GPIO 34,GPIO 35, GPIO 36, GPIO 39

Q3: What is capacitive touch pin?

The ESP32 has 10 internal capacitive touch sensors. These can sense variations in anything that holds an electrical charge, like the human skin. So they can detect variations induced when touching the GPIOs with a finger. These pins can be easily integrated into capacitive pads and replace mechanical buttons. The capacitive touch pins can also be used to wake up the ESP32 from deep sleep.

Q4: ADC and related channel in ESP

The ESP32 has 18 x 12 bits ADC input channels (while the ESP8266 only has 1x 10 bits ADC). These are the GPIOs that can be used as ADC and respective channels.

NOTE: ESP32 DEV board indicates these by decimal point.

For example: A1.6 indicates ADC1 Channel 6.

ADC1_CH0 (GPIO 36)

ADC1_CH1 (GPIO 37)

ADC1_CH2 (GPIO 38)

ADC1_CH3 (GPIO 39)

ADC1_CH4 (GPIO 32)

ADC1_CH5 (GPIO 33)

ADC1_CH6 (GPIO 34)

ADC1_CH7 (GPIO 35)

ADC2_CH0 (GPIO 4)

ADC2_CH1 (GPIO 0)

ADC2_CH2 (GPIO 2)

ADC2_CH3 (GPIO 15)

ADC2_CH4 (GPIO 13)

ADC2_CH5 (GPIO 12)

ADC2_CH6 (GPIO 14)

ADC2_CH7 (GPIO 27)

ADC2_CH8 (GPIO 25)

ADC2_CH9 (GPIO 26)

The ADC input channels have a 12-bit resolution. This means that you can get analog readings ranging from 0 to 4095, in which 0 corresponds to 0V and 4095 to 3.3V. You can also set the resolution of your channels on the code and the ADC range.

> **NOTE**: he ESP32 ADC pins don't have a linear behavior in range 0.0 to 0.1 and 3.2 to 3.3. You'll probably won't be able to distinguish between 0 and 0.1V, or between 3.2 and 3.3V.

Q5: DAC Digital to Analog Converter in ESP32

There are 2 x 8 bits DAC channels on the ESP32 to convert digital signals into analog voltage signal outputs. These are the DAC channels:

DAC1 (GPIO25)

DAC2 (GPIO26)

Q6: PWM CHANNEL in ESP32

The ESP32 LED PWM controller has 16 independent channels that can be configured to generate PWM signals with different properties. All pins that can act as outputs can be used as PWM pins (GPIOs 34 to 39 can't generate PWM).

To set a PWM signal, you need to define these parameters in the code:

Signal's frequency;

Duty cycle;

PWM channel;

GPIO where you want to output the signal.

Q7: ESP32 I2C CHANNEL

The ESP32 has two I2C channels and any pin can be set as SDA or SCL. When using the ESP32 with the Arduino IDE, the default I2C pins are:

- GPIO 21 (SDA)
- GPIO 22 (SCL)

If you want to use other pins when using the wire library, you just need to call:

CODE: **Wire.begin(SDA, SCL);**

Q8: SPI CHANNEL in ESP32

ESP32 have HSPI,VSPI,and SPI.

By default, the pin mapping for SPI is:

SPI	MOSI	MISO	CLK	CS
VSPI	GPIO 23	GPIO 19	GPIO 18	GPIO 5
HSPI	GPIO 13	GPIO 12	GPIO 14	GPIO 15

Q9: What is HSPI VSPI in esp32?

ESP32 integrates four SPI peripherals.

- SPI0 and SPI1 are used internally to access the ESP32's attached flash memory and thus are currently not open to users. They share one signal bus via an arbiter.
- SPI2 and SPI3 are general purpose SPI controllers, sometimes referred to as HSPI and VSPI, respectively. They are open to users. SPI2 and SPI3 have independent signal buses with the same respective names. Each bus has three CS lines to drive up to three SPI slaves.

Q10: Interrupt facility in ESP32?

All GPIOs can be configured as interrupts.

Q11: STRAPPING PINS in ESP32

The ESP32 chip has the following strapping pins:

GPIO 0

GPIO 2

GPIO 4

GPIO 5 (must be HIGH during boot)

GPIO 12 (must be LOW during boot)

GPIO 15 (must be HIGH during boot).

These are used to put the ESP32 into bootloader or flashing mode. On most development boards with built-in USB/Serial, you don't need to worry about the state of these pins. However, if you have peripherals connected to those pins, you may have trouble trying to upload new code, flashing the ESP32 with new firmware, or resetting the board. If you have some peripherals connected to the strapping pins and you are getting trouble uploading code or flashing the ESP32, it may be because those peripherals are preventing the ESP32 from entering the right mode.

Q12: EN – ENABLE pin in ESP32

Enable (EN) is the 3.3V regulator's enable pin. It's pulled up, so connect to ground to disable the 3.3V regulator. This means that you can use this pin connected to a pushbutton to restart your ESP32, for example.

Q13: Are ESP32 pins 5V tolerant?

The ESP32's operating voltage range is 2.2 to 3.6V. Under normal operation the ESP32 Thing will power the chip at 3.3V. The I/O pins are not 5V-tolerant!.

Q14: GPIO Current drawn in ESP32?

The absolute maximum current drawn per GPIO is 40mA according to the "Recommended Operating Conditions" section in the ESP32 datasheet.

Q15: Wi-Fi standard and Data rate?

2.4 GHz up to 150 Mbits/s.

Q16: Bluetooth support?

BLE (Bluetooth Low Energy) and legacy Bluetooth

Q17: Architecture

32 Bit, CLK support max 240 MHz. The ESP32 chip has a Tensilica Xtensa LX6 microprocessor in both dual-core and single-core variation. All chips in the series are dual-core, except the ESP32-S0WD, Performance = 600 DMIPS (Dhrystone Million Instructions Per Second).

Q18: Memory segmentation and Provision?

Internal memory for the ESP32 is as follows – (Note :Variation as per verssion)

ROM: 448 KiB (**for booting/core functions**),

SRAM: 520 KiB (for data/instructions),

RTC fast SRAM: 8 KiB (for data storage/main CPU during boot from sleep mode),

RTC slow SRAM: 8 KiB (for co-processor access during sleep mode),

eFuse: 1 KiBit (256 bits used for the system (MAC address and chip configuration) and 768 bits reserved for customer applications).

KiB = Kilo byte.

Q19: ESP32 Timer

The ESP32 has two timer groups, each one with two general purpose hardware timers. All the timers are based on **64 bits counters** and **16 bit prescalers** .

The timer counters can be configured to count up or down and support automatic reload and software reload. They can also generate alarms when they reach a specific value, defined by the software. The value of the counter can be read by the software program.