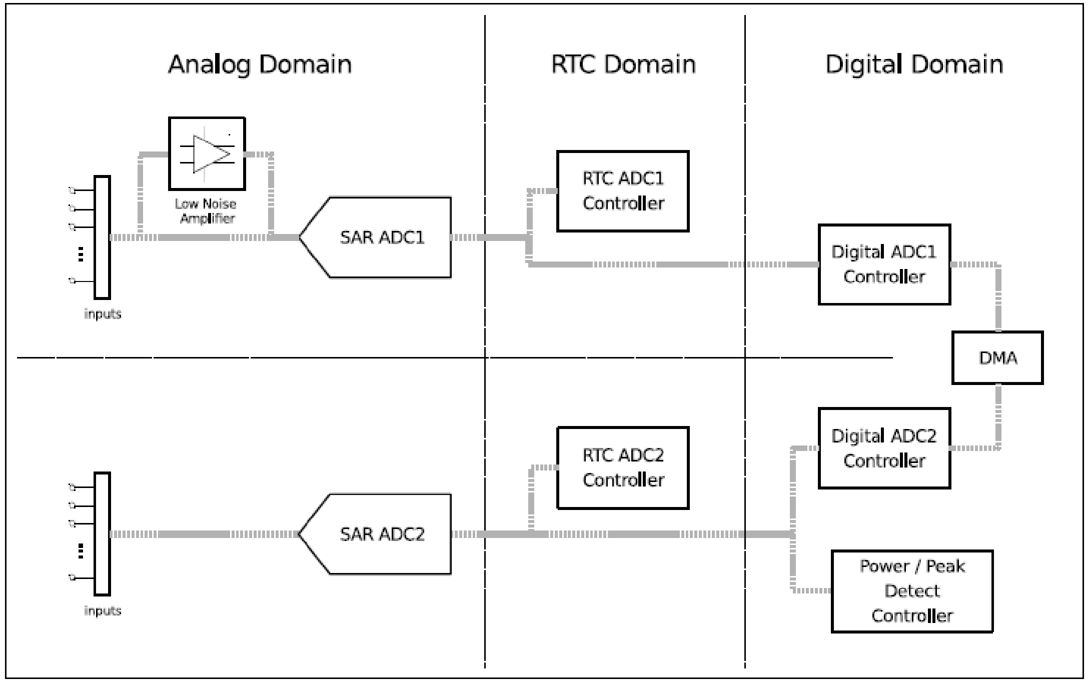
Prática 5

ADC

ADC

- ESP32 integrates two 12-bit SAR ADCs.
- It is also possible to measure internal signals, such as vdd33.



https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/peripherals/adc.html

ADC

- Two SAR ADCs, with simultaneous sampling and conversion
- Up to 18 analog input pads
- 12-bit, 11-bit, 10-bit, 9-bit configurable resolution
- DMA support (available on one controller)
- Multiple channel-scanning modes (available on two controllers)
- Operation during deep sleep (available on one controller)

ADC

- The ESP32 integrates 2 SAR (Successive Approximation Register) ADCs, supporting a total of 18 measurement channels (analog enabled pins).

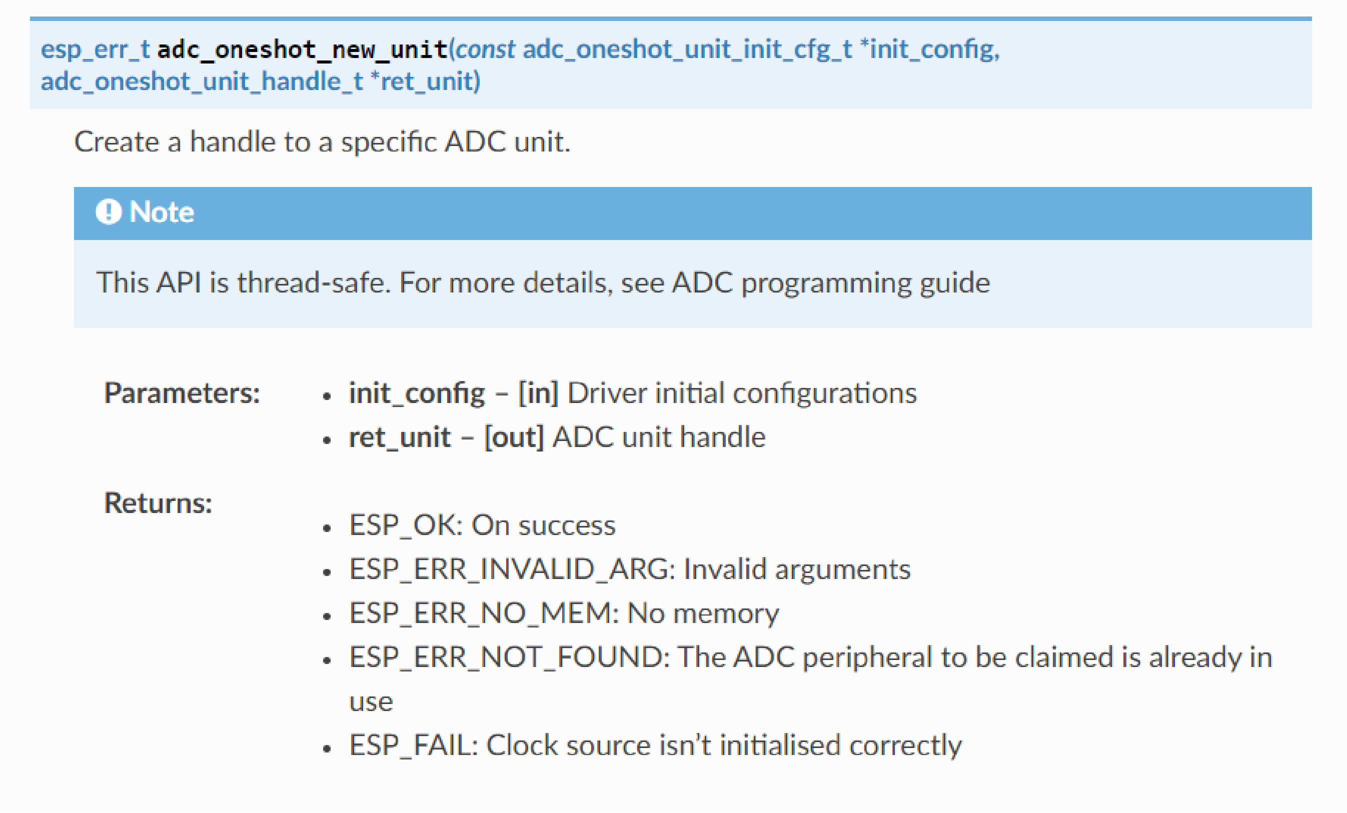
- These channels are supported:

- ADC1:
- 8 channels: GPIO32 - GPIO39

- ADC2:
- 10 channels: GPIO0, GPIO2, GPIO4, GPIO12 - GPIO15, GOIO25 - GPIO27

```c
/-------------ADC1 Init---------------//
adc_oneshot_unit_handle_t adc1_handle;
adc_oneshot_unit_init_cfg_t init_config1 = {
    .unit_id = ADC_UNIT_1,
};
adc_oneshot_new_unit(&init_config1, &adc1_handle);

//-------------ADC1 Config---------------//
adc_oneshot_chan_cfg_t config = {
    .bitwidth = ADC_BITWIDTH_DEFAULT,
    .atten = EXAMPLE_ADC_ATTEN,
};
adc_oneshot_config_channel(adc1_handle, EXAMPLE_ADC1_CHAN0, &config);

//-------------ADC1 Calibration Init---------------//
adc_cali_line_fitting_config_t cali_config = {
        .unit_id = ADC_UNIT_1,
        .atten = EXAMPLE_ADC_ATTEN,
        .bitwidth = ADC_BITWIDTH_DEFAULT,
    };

adc_cali_create_scheme_line_fitting(&cali_config, &handle);
```

adc_oneshot_new_unit



```
esp_err_t adc_oneshot_new_unit(const adc_oneshot_unit_init_cfg_t *init_config, adc_oneshot_unit_handle_t *ret_unit)
```

Create a handle to a specific ADC unit.

Note

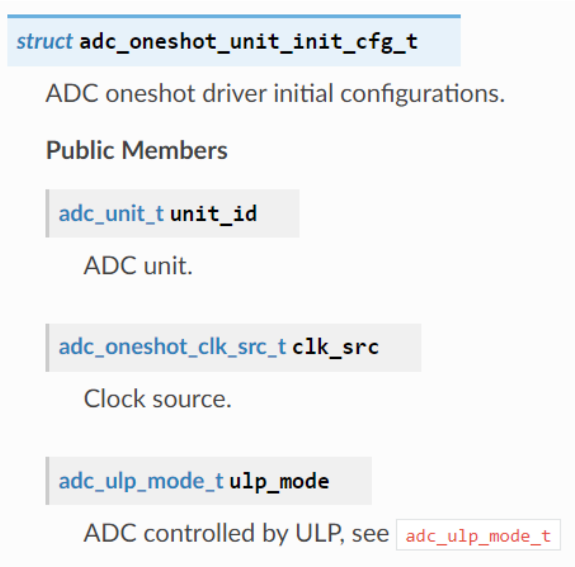
This API is thread-safe. For more details, see ADC programming guide

Parameters:

- **init_config** – **[in]** Driver initial configurations
- **ret_unit** – **[out]** ADC unit handle

Returns:

- ESP_OK: On success
- ESP_ERR_INVALID_ARG: Invalid arguments
- ESP_ERR_NO_MEM: No memory
- ESP_ERR_NOT_FOUND: The ADC peripheral to be claimed is already in use
- ESP_FAIL: Clock source isn't initialised correctly



```
struct adc_oneshot_unit_init_cfg_t
```

ADC oneshot driver initial configurations.

Public Members

```
adc_unit_t unit_id
```

ADC unit.

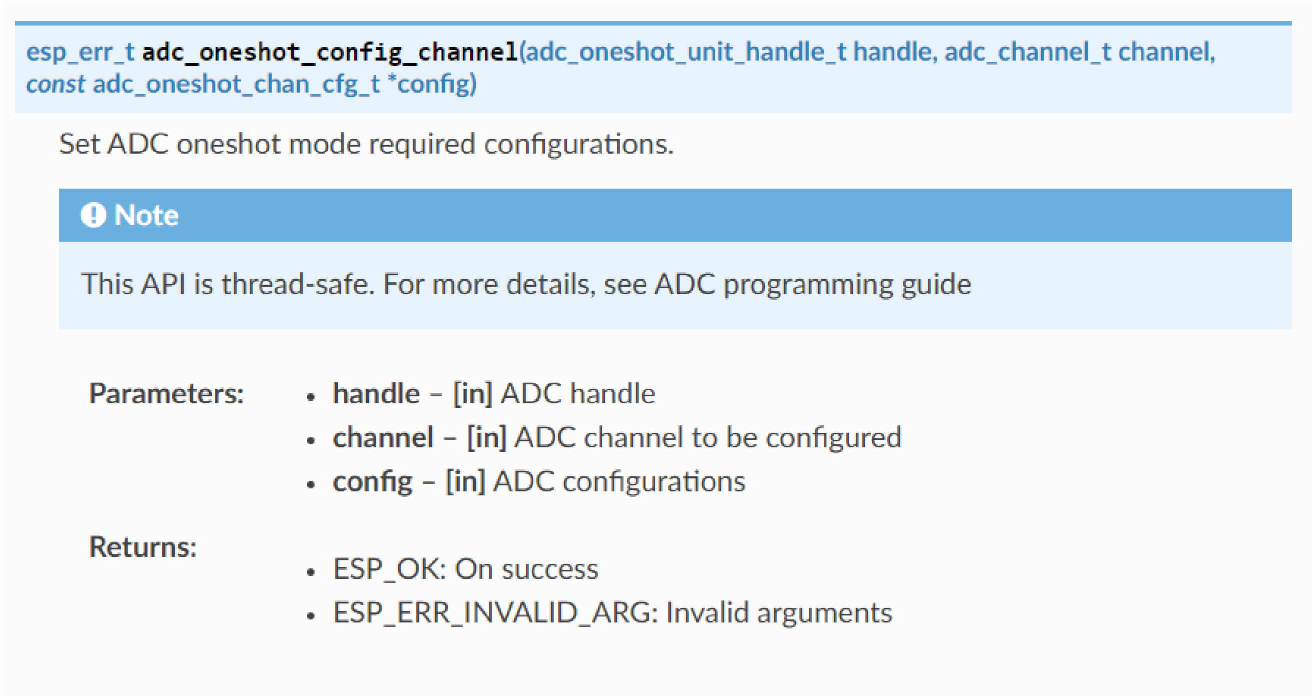
```
adc_oneshot_clk_src_t clk_src
```

Clock source.

```
adc_ulp_mode_t ulp_mode
```

ADC controlled by ULP, see `adc_ulp_mode_t`

adc_oneshot_config_channel



esp_err_t **adc_oneshot_config_channel**(adc_oneshot_unit_handle_t handle, adc_channel_t channel, *const* adc_oneshot_chan_cfg_t *config)

Set ADC oneshot mode required configurations.

Note

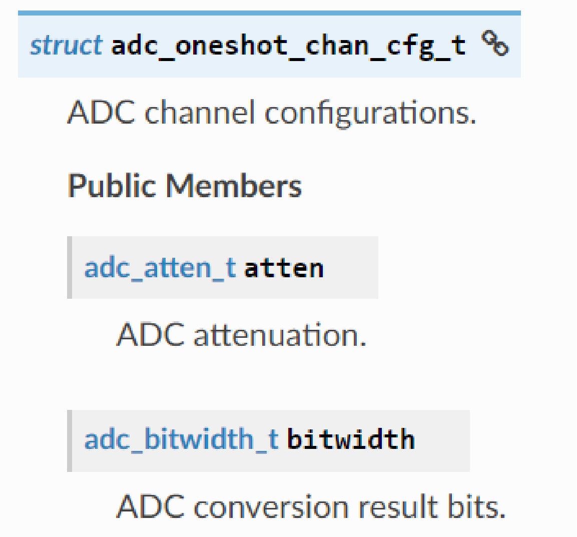
This API is thread-safe. For more details, see ADC programming guide

Parameters:

- **handle** – **[in]** ADC handle
- **channel** – **[in]** ADC channel to be configured
- **config** – **[in]** ADC configurations

Returns:

- ESP_OK: On success
- ESP_ERR_INVALID_ARG: Invalid arguments



struct **adc_oneshot_chan_cfg_t**

ADC channel configurations.

Public Members

adc_atten_t **atten**

ADC attenuation.

adc_bitwidth_t **bitwidth**

ADC conversion result bits.

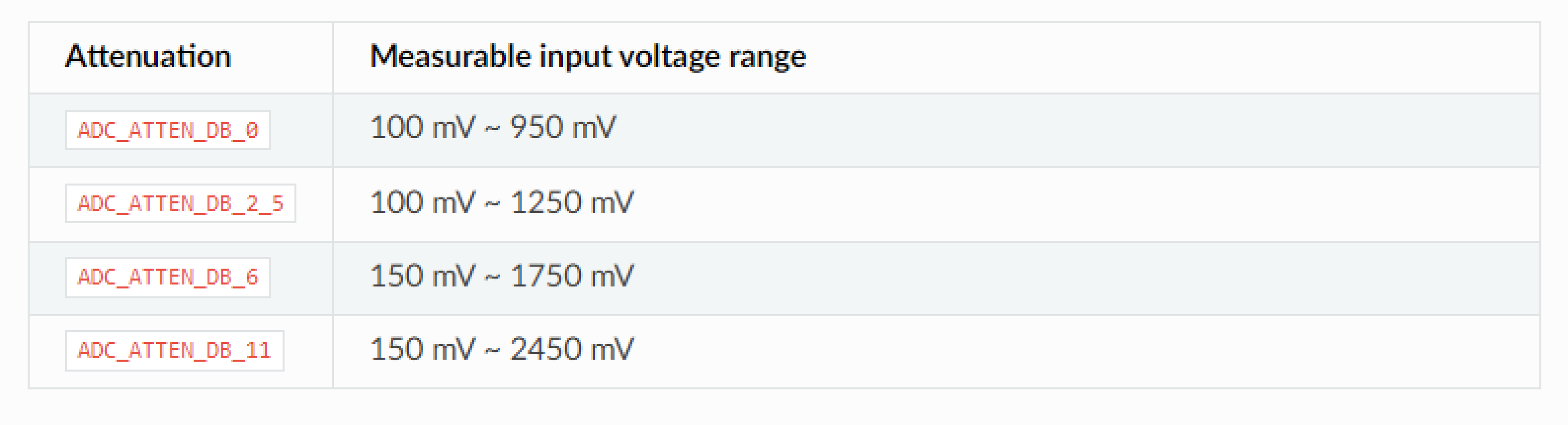
ADC Atenuation

The ESP32 ADCs can measure analog voltages from 0 V to Vref.

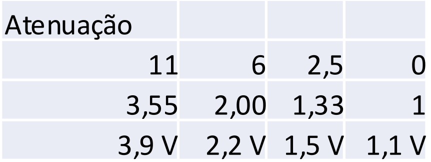
Among different chips, the Vref varies, the median is 1.1 V.

In order to convert voltages larger than Vref, input voltages can be attenuated before being input to the ADCs.

There are 4 available attenuation options, the higher the attenuation is, the higher the measurable input voltage could be.



Attenuation	Measurable input voltage range
`ADC_ATTEN_DB_0`	100 mV ~ 950 mV
`ADC_ATTEN_DB_2_5`	100 mV ~ 1250 mV
`ADC_ATTEN_DB_6`	150 mV ~ 1750 mV
`ADC_ATTEN_DB_11`	150 mV ~ 2450 mV

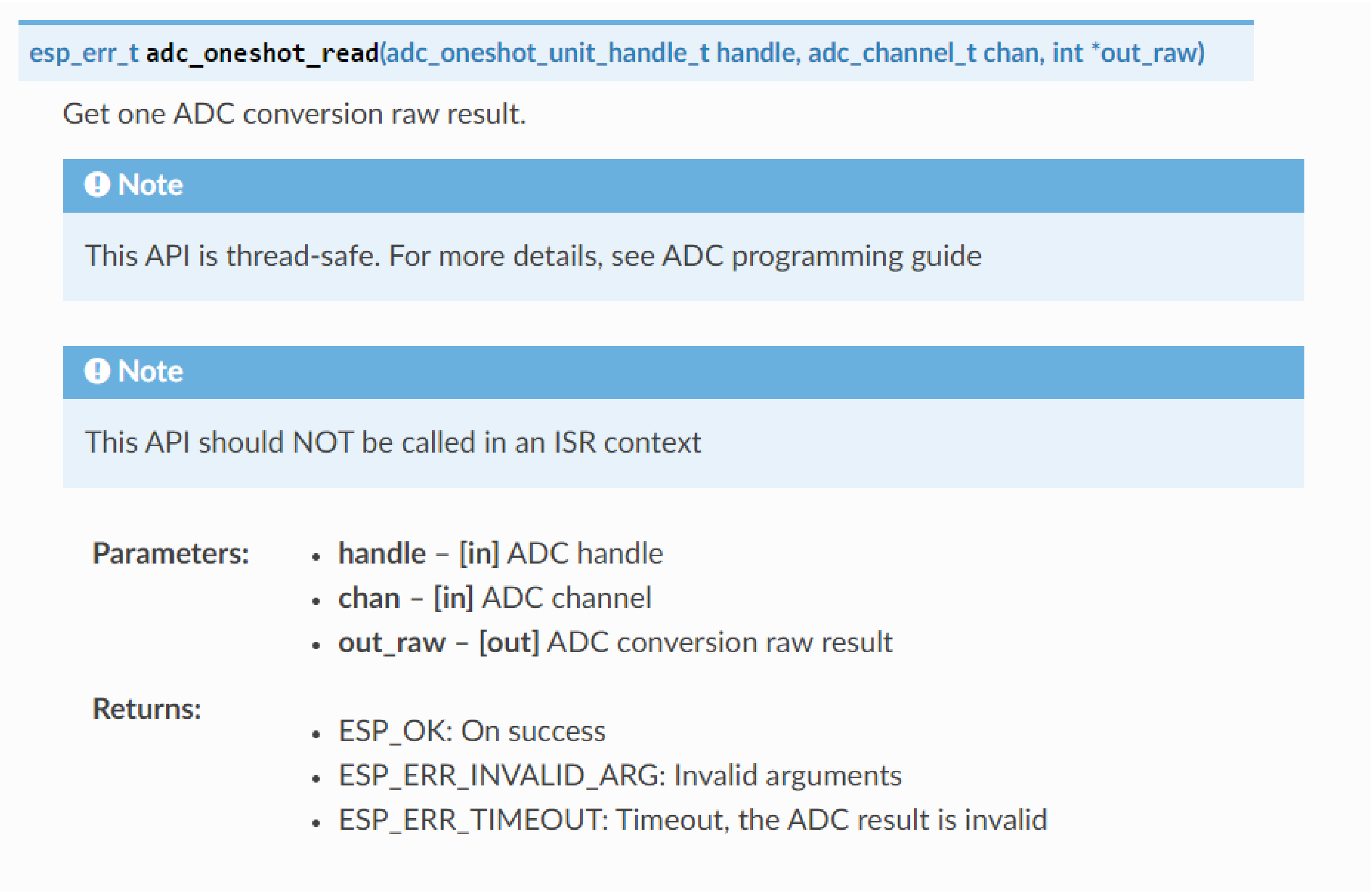


Atenuação			
11	6	2,5	0
3,55	2,00	1,33	1
3,9 V	2,2 V	1,5 V	1,1 V

Exemplo

```
while (1) {


        adc_oneshot_read(adc1_handle, EXAMPLE_ADC1_CHAN0, &adc_raw[0][0]);

        if (do_calibration1) {
        adc_cali_raw_to_voltage(adc1_cali_handle, adc_raw[0][0], &voltage[0][0]);
        }

        adc_oneshot_read(adc2_handle, EXAMPLE_ADC2_CHAN0, &adc_raw[1][0]);
        if (do_calibration2) {
        adc_cali_raw_to_voltage(adc2_cali_handle, adc_raw[1][0], &voltage[1][0]);
        }


        vTaskDelay(pdMS_TO_TICKS(1000));
}
```

adc_oneshot_read



esp_err_t **adc_oneshot_read**(adc_oneshot_unit_handle_t handle, adc_channel_t chan, int *out_raw)

Get one ADC conversion raw result.

Note

This API is thread-safe. For more details, see ADC programming guide

Note

This API should NOT be called in an ISR context

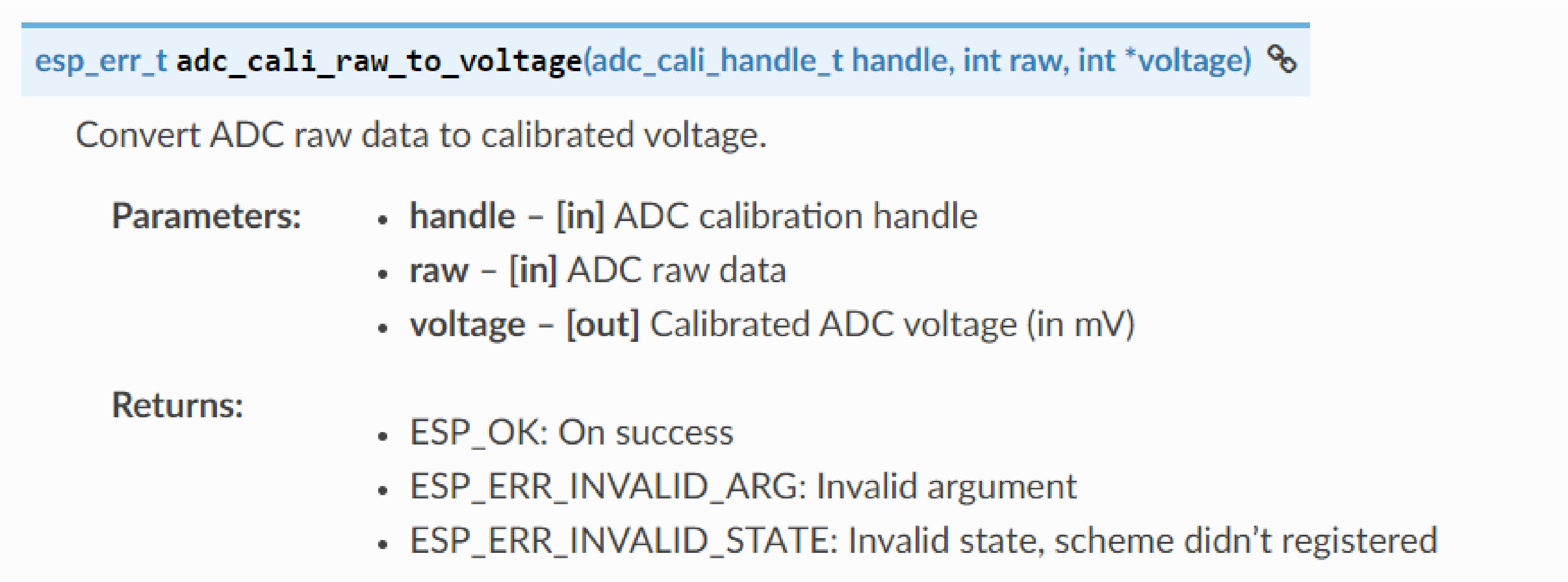
Parameters:

- **handle** – **[in]** ADC handle
- **chan** – **[in]** ADC channel
- **out_raw** – **[out]** ADC conversion raw result

Returns:

- ESP_OK: On success
- ESP_ERR_INVALID_ARG: Invalid arguments
- ESP_ERR_TIMEOUT: Timeout, the ADC result is invalid

adc_cali_raw_to_voltage



esp_err_t **adc_cali_raw_to_voltage**(adc_cali_handle_t handle, int raw, int *voltage)

Convert ADC raw data to calibrated voltage.

Parameters:

- **handle** – **[in]** ADC calibration handle
- **raw** – **[in]** ADC raw data
- **voltage** – **[out]** Calibrated ADC voltage (in mV)

Returns:

- ESP_OK: On success
- ESP_ERR_INVALID_ARG: Invalid argument
- ESP_ERR_INVALID_STATE: Invalid state, scheme didn’t registered

Referências

- https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf