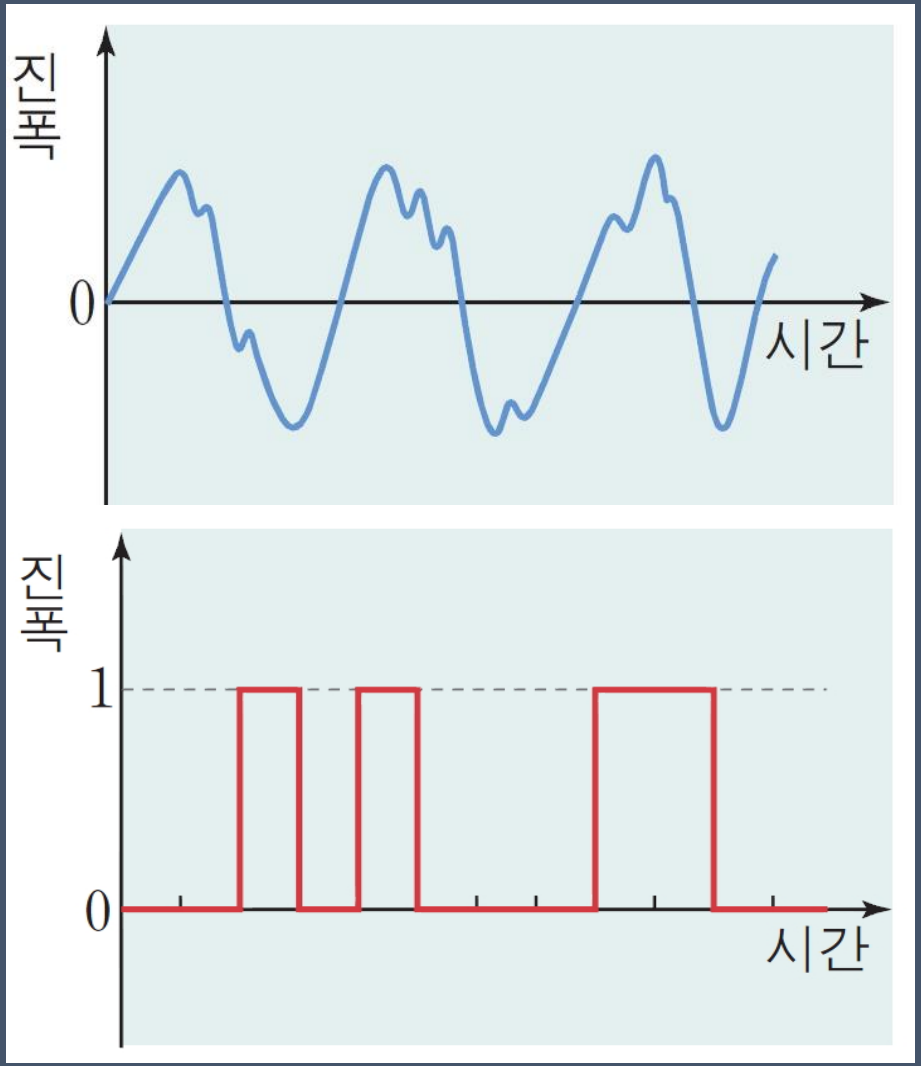
메이커 스페이스
G·캠프

ESP32

온라인 워크숍 초급과정

1. GPIO

3.3Volts의 power를 기반으로 동작하는 회로에서는 0Volts는 '0', 3.3Volts는 '1'값으로 정의



0V	3.3V
Open	Closed
Off	On
Low	High
Clear	Set
0	1
False	True

ESP32 GPIO specification

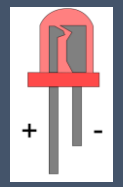
- ESP32 칩에는 40 개의 물리적 GPIO 패드가 있음
- 일부 GPIO 패드는 사용할 수 없거나 칩 패키지에 해당 핀이 없음 (DataSheet 참조).
- 각 패드는 범용 I / O로 사용되거나 내부 주변 신호에 연결 될 수 있음

✓ GPIO 6 ~ 11은 일반적으로 SPI 플래시에 사용
✓ GPIO 34 ~ 39는 입력 모드로만 설정할 수 있으며 소프트웨어 풀업 또는 풀다운 기능이 없음 (중요!!)
✓ GPIO가 RTC저전력 및 아날로그 하위 시스템으로 라우팅 될 때 작동하는 별도의 RTC GPIO 지원도 있음
이 핀 기능은 최대 절전 모드, Ultra Low Power 보조 프로세서 가 실행 중 또는 ADC / DAC / 등과 같은 아날로그 기능을 사용중인 경우에 사용할 수 있음

GPIO Output 실습 - LED On/Off

GPIO Digital Output은 ‘0’, ‘1’ 값을 통해 0, 3.3V 전압을 출력

LED(Light Emitting Diode)는 빛을 내는 반도체
LED는 극을 가지고 있으며 다리가 긴 쪽이 '+' (Anode), 다리가 짧은 쪽이 '-' (Cathode)



Absolute Maximum Ratings: (Ta=25℃) .

ITEMS	Symbol	Absolute Maximum Rating	Unit
Forward Current	I_F	20	mA
Peak Forward Current	I_{FP}	30	mA
Suggestion Using Current	I_{su}	16-18	mA
Reverse Voltage (V_R=5V)	I_R	10	uA
Power Dissipation	P_D	105	mW
Operation Temperature	T_{OPR}	-40 ~ 85	℃
Storage Temperature	T_{STG}	-40 ~ 100	℃
Lead Soldering Temperature	T_{SOL}	Max. 260℃ for 3 Sec. Max. (3mm from the base of the expoxy bulb)	

Absolute Maximum Ratings: (Ta=25℃)

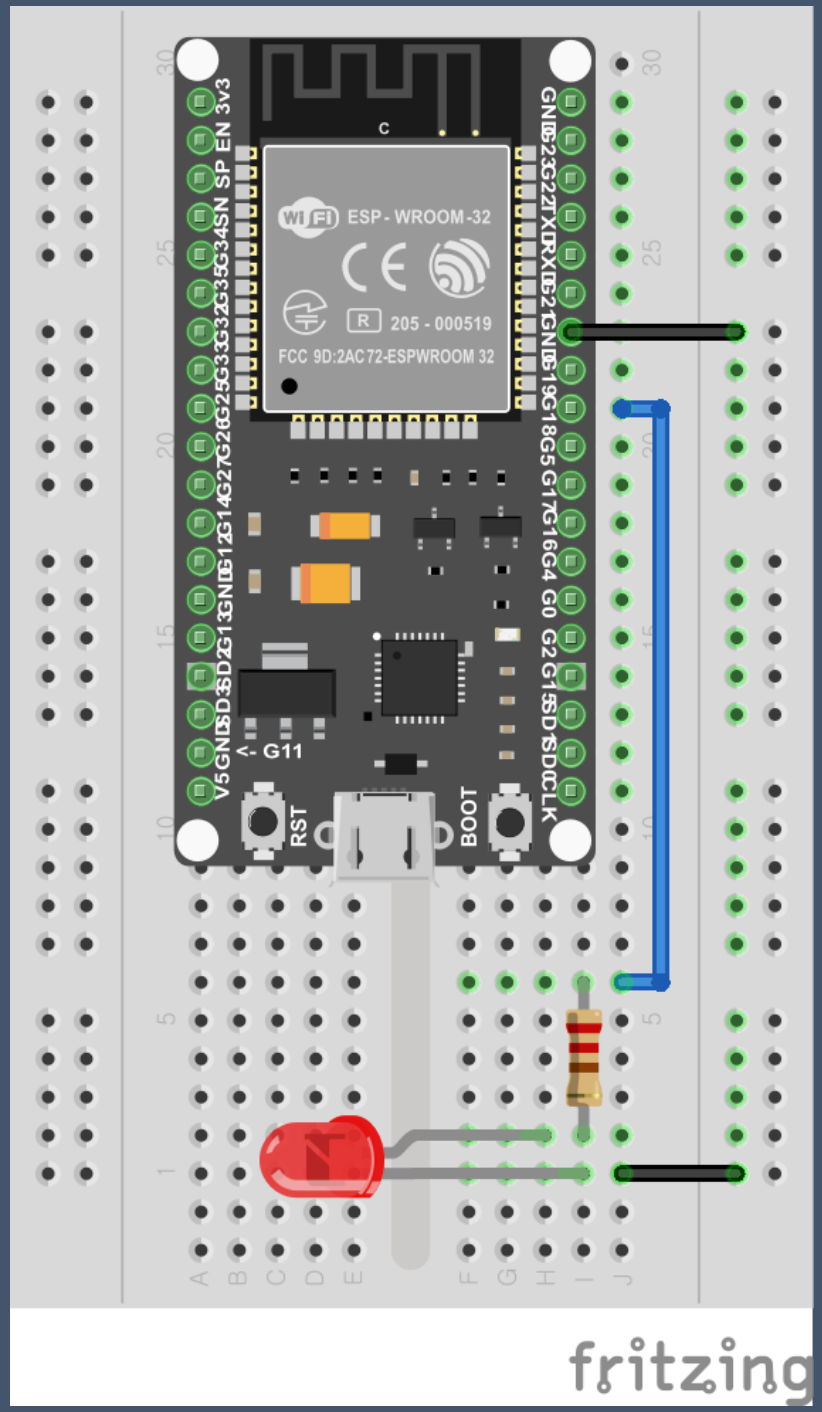
ITEMS	Symbol	Test condition	Min.	Typ.	Max.	Unit
Forward Voltage	V_F	I_F=20mA	1.8	---	2.2	V
Wavelenength (nm) or TC(k)	$\Delta\lambda$	I_F=20mA	620	---	625	nm
*Luminous intensity	I_V	I_F=20mA	150	---	200	mcd
50% Viewing Angle	$2\theta 1/2$	I_F=20mA	40	---	60	deg

LED 저항값 계산하기
LED는 최소 구동전압과 최대 구동전압이 있습니다. 최소 구동전압보다 낮으면 빛이 흐리고 최대 구동전압보다 높으면 LED가 망가지게 됩니다. LED의 스펙은 LED datasheet에 나와 있습니다.

저항값 계산 식
V= I*R (V:전압, I,전류, R:저항)
R=V/I → 저항값 = (공급전압 - LED전압)/LED 소모전류

예) LED의 최소구동전압: 2V
LED의 소모전류: 20mA
디지털출력의 전압: 3.3V

(3.3V - 2V)/0.02A = 65Ω



ESP32 18번 핀 --- 저항 연결 --- LED 연결

gpio_out 예제 빌드 및 flash

esp_err_t gpio_config(*const gpio_config_t *pGPIOConfig*)

GPIO common configuration.

Configure GPIO's Mode,pull-up,PullDown,IntrType

Return

- ESP_OK success
- ESP_ERR_INVALID_ARG Parameter error

Parameters

- `pGPIOConfig` : Pointer to GPIO configure struct

```
/**
 * @brief Configuration parameters of GPIO pad for gpio_config function
 */
typedef struct {
    uint64_t pin_bit_mask;          /*!< GPIO pin: set with bit mask, each bit maps to a GPIO */
    gpio_mode_t mode;               /*!< GPIO mode: set input/output mode                      */
    gpio_pullup_t pull_up_en;       /*!< GPIO pull-up                                          */
    gpio_pulldown_t pull_down_en;   /*!< GPIO pull-down                                        */
    gpio_int_type_t intr_type;      /*!< GPIO interrupt type                                   */
} gpio_config_t;
```

esp_err_t gpio_set_level(gpio_num_t *gpio_num*, uint32_t *level*)

GPIO set output level.

Return

- ESP_OK Success
- ESP_ERR_INVALID_ARG GPIO number error

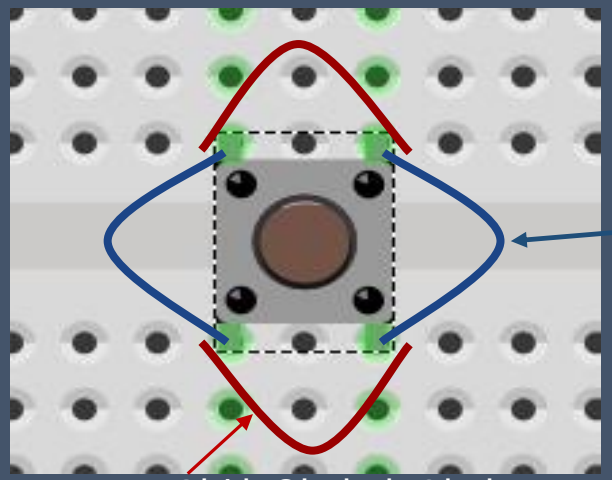
Parameters

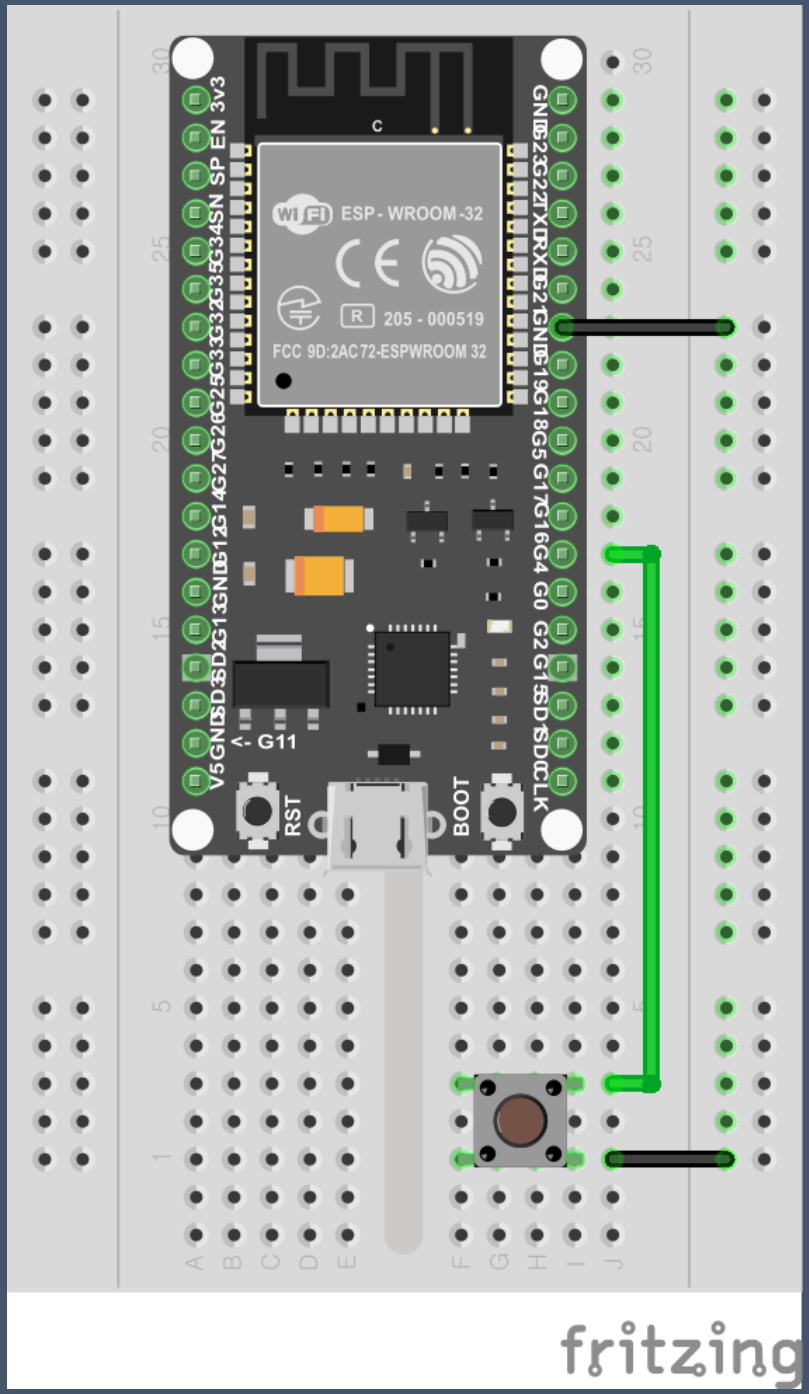
- `gpio_num` : GPIO number. If you want to set the output level of e.g. GPIO16, gpio_num should be GPIO_NUM_16 (16);
- `level` : Output level. 0: low ; 1: high

```
/**
 * @brief  GPIO set output level
 *
 * @param  hw Peripheral GPIO hardware instance address.
 * @param  gpio_num GPIO number. If you want to set the output level of e.g. GPIO16, gpio_nu
 * @param  level Output level. 0: low ; 1: high
 */
static inline void gpio_ll_set_level(gpio_dev_t *hw, gpio_num_t gpio_num, uint32_t level)
{
    if (level) {
        if (gpio_num < 32) {
            hw->out_w1ts = (1 << gpio_num);
        } else {
            hw->out1_w1ts.data = (1 << (gpio_num - 32));
        }
    } else {
        if (gpio_num < 32) {
            hw->out_w1tc = (1 << gpio_num);
        } else {
            hw->out1_w1tc.data = (1 << (gpio_num - 32));
        }
    }
}
```

GPIO Input 실습 - Button

GPIO Digital Input은 0, 3.3V 전압을 ‘0’, ’1’ 값으로 인식
GPIO Output 핀은 Input으로 사용 가능





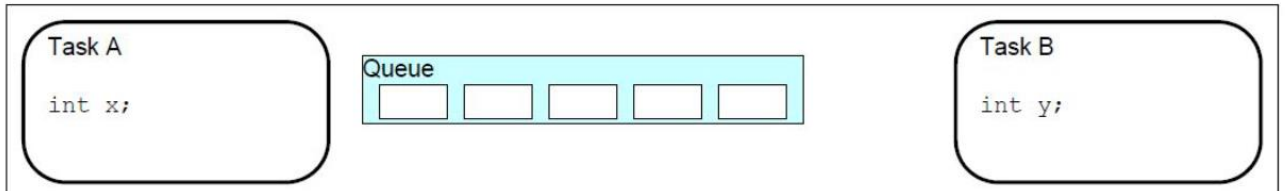
ESP32 4번 핀 --- 스위치 연결

gpio_input 예제 빌드 및 flash

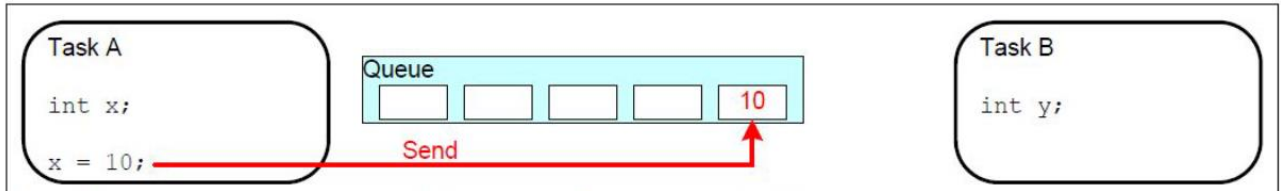
FreeRTOS Elements

```
//create a queue to handle gpio event from isr
    gpio_evt_queue = xQueueCreate(10, sizeof(uint32_t));

//start gpio task
    xTaskCreate(gpio_task_example, "gpio_task_example", 2048, NULL, 10, NULL);

//Interrupt Service routine 에서 특정데이터를 이벤트 Queue로 넘긴다
xQueueSendFromISR(gpio_evt_queue, &gpio_num, NULL);

static void gpio_task_example(void* arg)
{
    uint32_t io_num;
    for(;;) {
        if(xQueueReceive(gpio_evt_queue, &io_num, portMAX_DELAY)) {
            printf("GPIO[%d] intr !!!₩n", io_num);
        }
    }
}
```

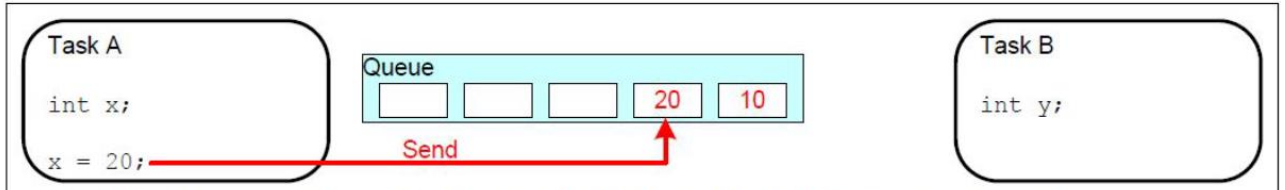
FreeRTOS - Queue Management



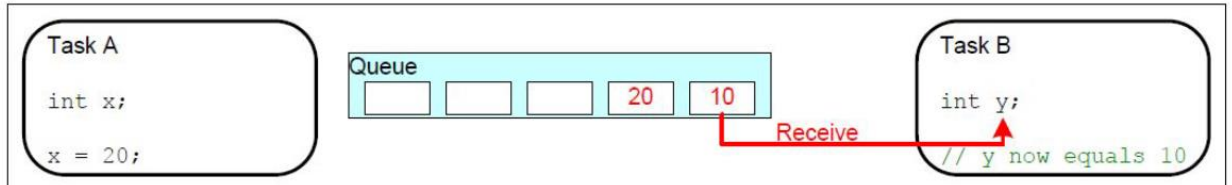
A queue is created to allow Task A and Task B to communicate. The queue can hold a maximum of 5 integers. When the queue is created it does not contain any values so is empty.



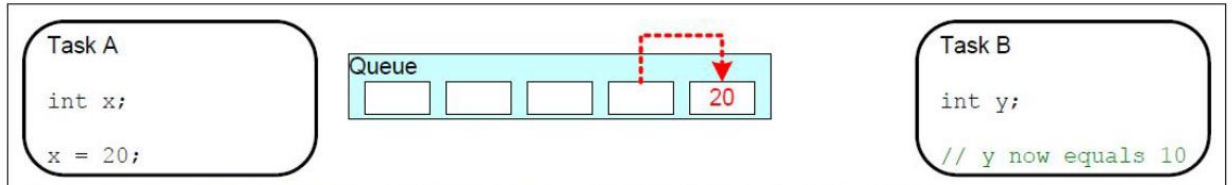
Task A writes (sends) the value of a local variable to the back of the queue. As the queue was previously empty the value written is now the only item in the queue, and is therefore both the value at the back of the queue and the value at the front of the queue.



Task A changes the value of its local variable before writing it to the queue again. The queue now contains copies of both values written to the queue. The first value written remains at the front of the queue, the new value is inserted at the end of the queue. The queue has three empty spaces remaining.



Task B reads (receives) from the queue into a different variable. The value received by Task B is the value from the head of the queue, which is the first value Task A wrote to the queue (10 in this illustration).



Task B has removed one item, leaving only the second value written by Task A remaining in the queue. This is the value Task B would receive next if it read from the queue again. The queue now has four empty spaces remaining.

`esp_err_t gpio_isr_handler_add(gpio_num_t gpio_num, gpio_isr_t isr_handler, void *args)`

Add ISR handler for the corresponding GPIO pin.

Call this function after using gpio_install_isr_service() to install the driver's GPIO ISR handler service.

The pin ISR handlers no longer need to be declared with IRAM_ATTR, unless you pass the ESP_INTR_FLAG_IRAM flag when allocating the ISR in gpio_install_isr_service().

This ISR handler will be called from an ISR. So there is a stack size limit (configurable as "ISR stack size" in menuconfig). This limit is smaller compared to a global GPIO interrupt handler due to the additional level of indirection.

Return

- ESP_OK Success
- ESP_ERR_INVALID_STATE Wrong state, the ISR service has not been initialized.
- ESP_ERR_INVALID_ARG Parameter error

Parameters

- `gpio_num` : GPIO number
- `isr_handler` : ISR handler function for the corresponding GPIO number.
- `args` : parameter for ISR handler.

`esp_err_t gpio_set_intr_type(gpio_num_t gpio_num, gpio_int_type_t intr_type)`

GPIO set interrupt trigger type.

Return

- ESP_OK Success
- ESP_ERR_INVALID_ARG Parameter error

Parameters

- `gpio_num` : GPIO number. If you want to set the trigger type of e.g. of GPIO16, gpio_num should be GPIO_NUM_16 (16);
- `intr_type` : Interrupt type, select from gpio_int_type_t

```
typedef enum {
    GPIO_INTR_DISABLE = 0,     /*!< Disable GPIO interrupt                      */
    GPIO_INTR_POSEDGE = 1,     /*!< GPIO interrupt type : rising edge           */
    GPIO_INTR_NEGEDGE = 2,     /*!< GPIO interrupt type : falling edge          */
    GPIO_INTR_ANYEDGE = 3,     /*!< GPIO interrupt type : both rising and falling edge */
    GPIO_INTR_LOW_LEVEL = 4,   /*!< GPIO interrupt type : input low level trigger */
    GPIO_INTR_HIGH_LEVEL = 5,  /*!< GPIO interrupt type : input high level trigger */
    GPIO_INTR_MAX,
} gpio_int_type_t;
```

`esp_err_t gpio_install_isr_service(int intr_alloc_flags)`

Install the driver's GPIO ISR handler service, which allows per-pin GPIO interrupt handlers.

This function is incompatible with gpio_isr_register() - if that function is used, a single global ISR is registered for all GPIO interrupts. If this function is used, the ISR service provides a global GPIO ISR and individual pin handlers are registered via the gpio_isr_handler_add() function.

Return

- ESP_OK Success
- ESP_ERR_NO_MEM No memory to install this service
- ESP_ERR_INVALID_STATE ISR service already installed.
- ESP_ERR_NOT_FOUND No free interrupt found with the specified flags
- ESP_ERR_INVALID_ARG GPIO error

Parameters

- `intr_alloc_flags` : Flags used to allocate the interrupt. One or multiple (ORred) ESP_INTR_FLAG_* values. See esp_intr_alloc.h for more info.

ESP_INTR_FLAG_* defines. These restrict the choice of interrupts that this routine can choose from. If this value is 0, it will default to allocating a non-shared interrupt of level 1, 2 or 3.

```
//interrupt of falling edge
io_conf.intr_type = GPIO_PIN_INTR_NEGEDGE;

//set as input mode
io_conf.mode = GPIO_MODE_INPUT;
//enable pull-up mode
io_conf.pull_up_en = 1;
io_conf.pull_down_en = 0;
```

esp32 datasheet 46page

Each digital output pin is associated with its configurable drive strength. Column "Drive Strength" in Table IO_MUX lists the default values. The drive strength of the digital output pins can be configured into one of the following four options:

- 0: ~5 mA
- 1: ~10 mA
- 2: ~20 mA
- 3: ~40 mA

"Drive Strength"

The default value is 2. The drive strength of the internal pull-up (wpu) and pull-down (wpd) is ~75 μA.

esp_err_t **gpio_set_drive_capability**(gpio_num_t *gpio_num*, gpio_drive_cap_t *strength*)

Set GPIO pad drive capability.

Return

- ESP_OK Success
- ESP_ERR_INVALID_ARG Parameter error

Parameters

- `gpio_num` : GPIO number, only support output GPIOs
- `strength` : Drive capability of the pad

esp_err_t **gpio_get_drive_capability**(gpio_num_t *gpio_num*, gpio_drive_cap_t **strength*)

Get GPIO pad drive capability.

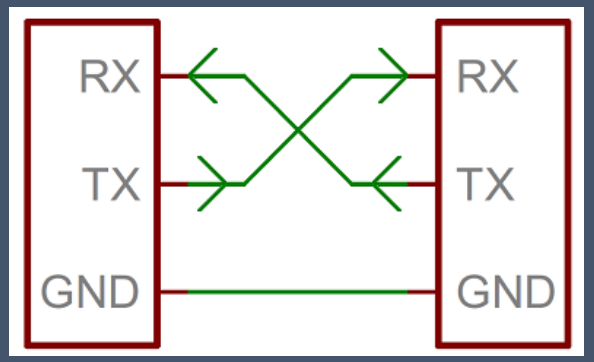
Return

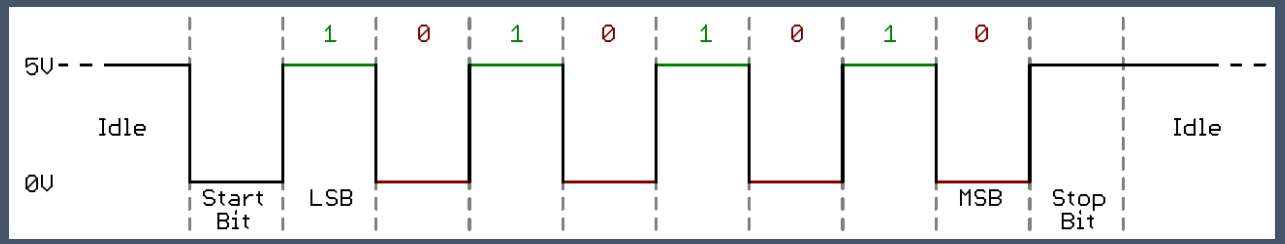
- ESP_OK Success
- ESP_ERR_INVALID_ARG Parameter error

Parameters

- `gpio_num` : GPIO number, only support output GPIOs
- `strength` : Pointer to accept drive capability of the pad

2. UART

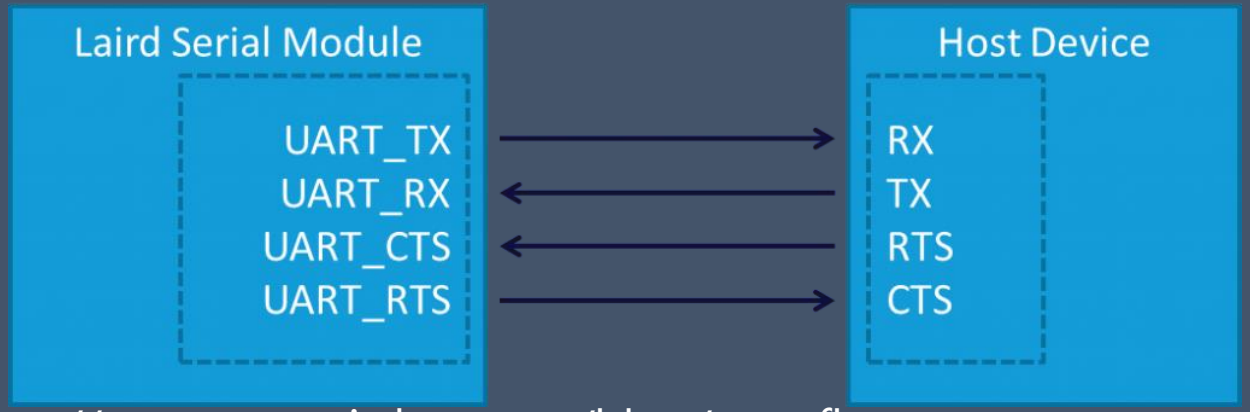




자료 출처 : learn.sparkfun.com (TTL)

Specification	RS232C	RS423	RS422	RS485
동작 모드	Single-Ended	Single-Ended	Differential	Differential
최대 Driver/Receiver 수	1 Driver 1 Receiver	1 Driver 10 Receivers	1 Driver 10 Receivers	32 Drivers 32 Receivers
최대 통달거리	약 15 m	약 1.2 km	약 1.2 km	약 1.2 km
최고 통신속도	20 Kb/s	100 Kb/s	10 Mb/s	10 Mb/s
지원 전송방식	Full Duplex	Full Duplex	Full Duplex	Half Duplex
최대 출력전압	±25V	±6V	-0.25V to +6V	-7V to +12V
최대 입력전압	±15V	±12V	-7V to +7V	-7V to +12V

자료 출처 : https://jusths.tistory.com/41



자료 출처 : http://www.summitdata.com/blog/uart-flow-control-rtscts-necessary-proper-operation-wireless-modules/

ESP32 UART specification

- ESP32 칩에는 프로그래밍과 유연성을 쉽게 하기 위해 동일한 레지스터 세트가 있는 3 개의 UART 컨트롤러 (UART0, UART1 및 UART2)가 있음

- 각 UART 컨트롤러는 전송 속도, 데이터 비트 길이, 비트 순서, 정지 비트 수, 패리티 비트 등과 같은 매개 변수를 사용하여 독립적으로 구성 할 수 있음

- 모든 컨트롤러는 다양한 제조업체의 UART 지원 장치와 호환되며 적외선 데이터 연결 프로토콜도 지원할 수 있음(IrDA)

1. Setting Communication Parameters - Setting baud rate, data bits, stop bits, etc.
2. Setting Communication Pins - Assigning pins for connection to a device.
3. Driver Installation - Allocating ESP32's resources for the UART driver.
4. Running UART Communication - Sending / receiving data
5. Using Interrupts - Triggering interrupts on specific communication events
6. Deleting a Driver - Freeing allocated resources if a UART communication is no longer required

"printf : 당연한 것은 없습니다."

```
379 #if defined(CONFIG_VFS_SUPPORT_IO) && !defined(CONFIG_ESP_CONSOLE_UART_NONE)
380     esp_reent_init(_GLOBAL_REENT);
381     const char* default_uart_dev = "/dev/uart/" STRINGIFY(CONFIG_ESP_CONSOLE_UART_NUM);
382     _GLOBAL_REENT->_stdin  = fopen(default_uart_dev, "r");
383     _GLOBAL_REENT->_stdout = fopen(default_uart_dev, "w");
384     _GLOBAL_REENT->_stderr = fopen(default_uart_dev, "w");
385 #else // defined(CONFIG_VFS_SUPPORT_IO) && !defined(CONFIG_ESP_CONSOLE_UART_NONE)
386     _REENT_SMALL_CHECK_INIT(_GLOBAL_REENT);
387 #endif // defined(CONFIG_VFS_SUPPORT_IO) && !defined(CONFIG_ESP_CONSOLE_UART_NONE)
388
389     esp_timer_init();
390     esp_set_time_from_rtc();
391 #if CONFIG_APPTRACE_ENABLE
392     err = esp_apptrace_init();
393     assert(err == ESP_OK && "Failed to init apptrace module on PRO CPU!");
394 #endif
395 #if CONFIG_SYSVIEW_ENABLE
396     SEGGER_SYSVIEW_Conf();
397 #endif
398 #if CONFIG_ESP_DEBUG_STUBS_ENABLE
399     esp_dbg_stubs_init();
400 #endif
401     err = esp_pthread_init();
402     assert(err == ESP_OK && "Failed to init pthread module!");
403
404     do_global_ctors();
405 #if CONFIG_ESP_INT_WDT
"./components/esp32/cpu_start.c" 587 lines --64%--
```

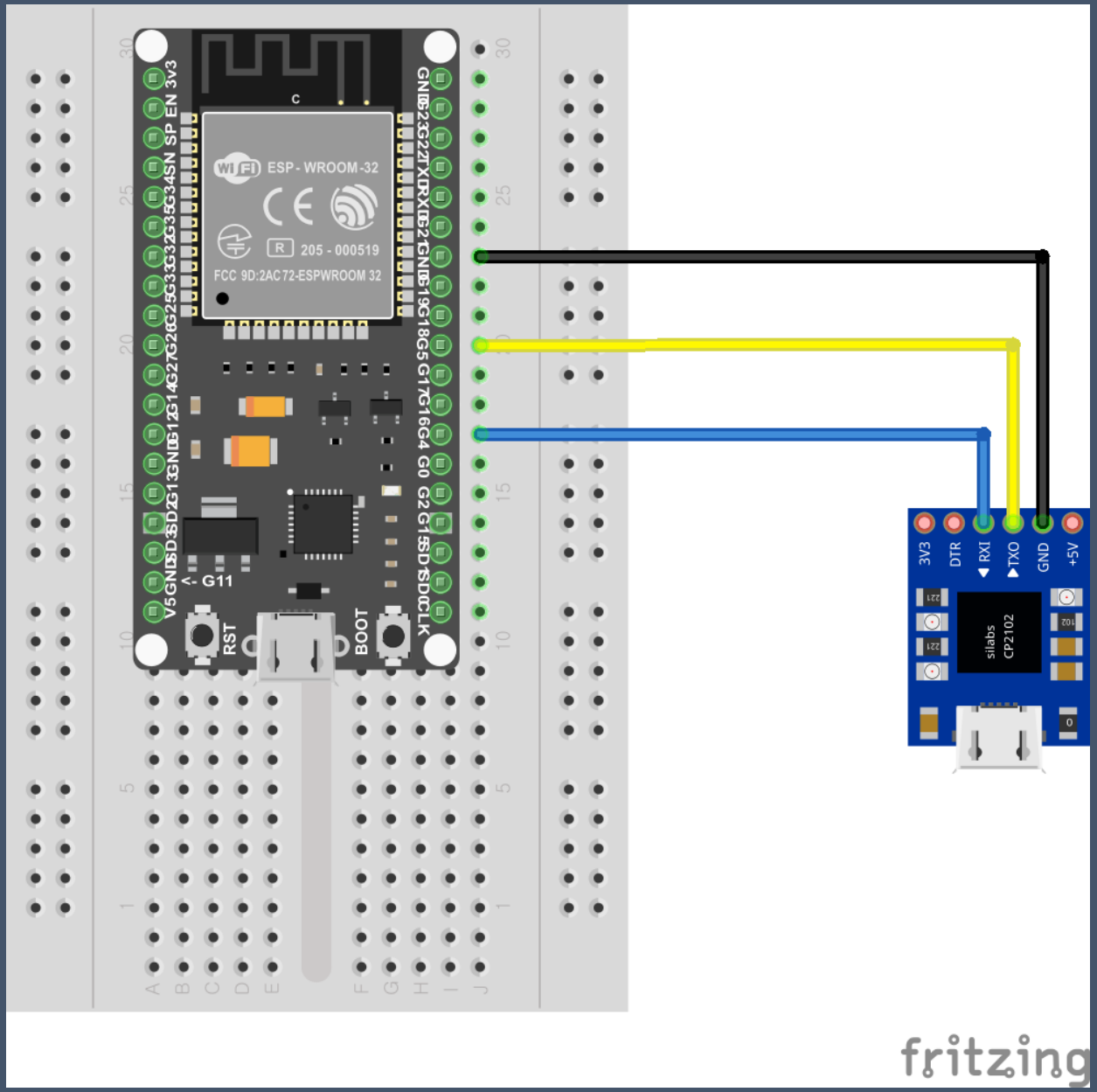
ESP-IDF Command Prompt (cmd.exe) - "C:₩Users₩jbmas₩.espressif₩idf_cmd_init.bat" "C:₩Users₩jbmas₩AppData₩Local₩Programs

```
Adding ESP-IDF tools to PATH...
Not using an unsupported version of tool ninja found in PATH: 1.5.3.
    C:₩Users₩jbmas₩.espressif₩tools₩xtensa-esp32-elf₩esp-2019r2-8.2.0₩xtensa-esp32-elf₩bin
    C:₩Users₩jbmas₩.espressif₩tools₩esp32ulp-elf₩2.28.51.20170517₩esp32ulp-elf-binutils₩bin
    C:₩Users₩jbmas₩.espressif₩tools₩cmake₩3.13.4₩bin
    C:₩Users₩jbmas₩.espressif₩tools₩openocd-esp32₩v0.10.0-esp32-20190313₩openocd-esp32₩bin
    C:₩Users₩jbmas₩.espressif₩tools₩mconf₩v4.6.0.0-idf-20190628₩
    C:₩Users₩jbmas₩.espressif₩tools₩ninja₩1.9.0₩
    C:₩Users₩jbmas₩.espressif₩tools₩idf-exe₩1.0.1₩
    C:₩Users₩jbmas₩.espressif₩tools₩ccache₩3.7₩
    C:₩Users₩jbmas₩.espressif₩python_env₩idf4.0_py3.7_env₩Scripts
    C:₩Users₩jbmas₩work₩esp-idf₩tools

Checking if Python packages are up to date...
Python requirements from C:₩Users₩jbmas₩work₩esp-idf₩requirements.txt are satisfied.

Done! You can now compile ESP-IDF projects.
Go to the project directory and run:

  idf.py build


C:₩Users₩jbmas₩work₩esp-idf>cd ..

C:₩Users₩jbmas₩work>cd esp_test

C:₩Users₩jbmas₩work₩esp_test>cd hello_world

C:₩Users₩jbmas₩work₩esp_test₩hello_world>idf.py menuconfig
```

ESP-IDF Command Prompt (cmd.exe) - "C:₩Users₩jbmas₩.espressif₩idf_cmd_init.bat" "C:₩Users₩jbmas₩AppData₩Local₩Programs₩Python₩P...

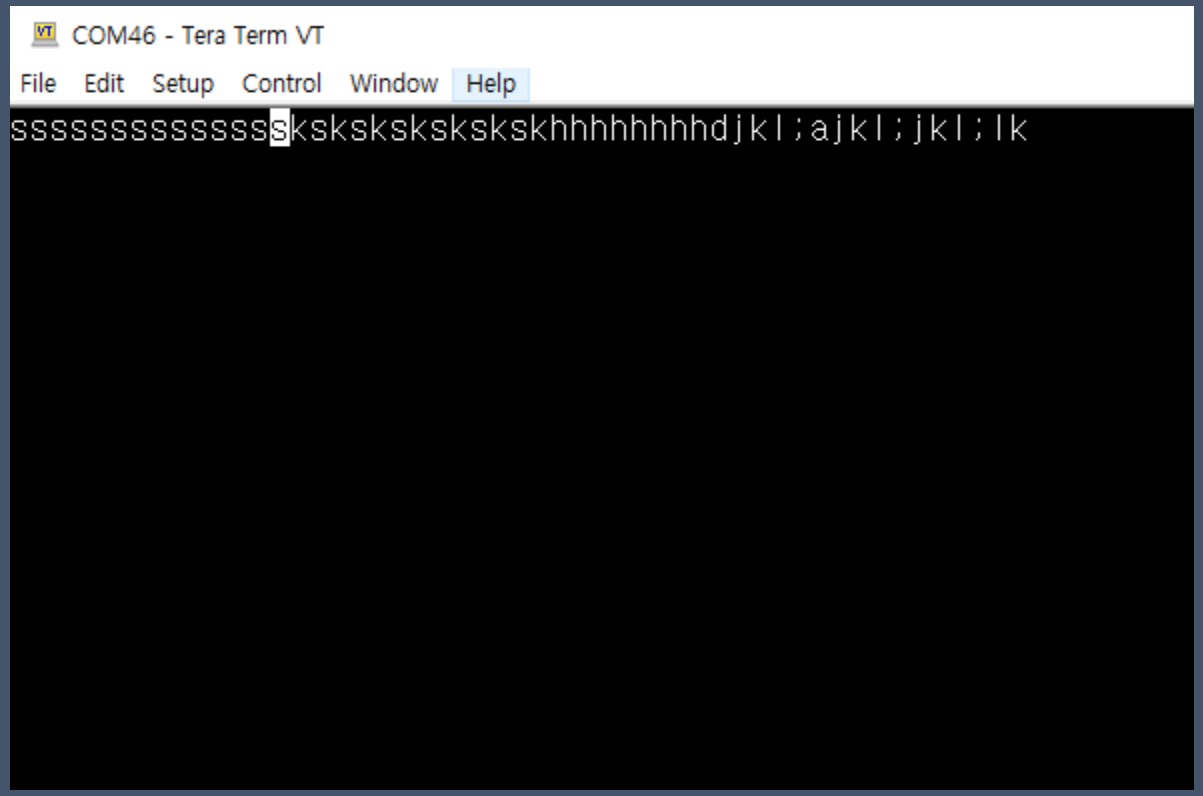
```
C:/Users/jbmas/work/esp_test/hello_world/sdkconfig - Espressif IoT Development Framework Configuration
→ Component config → Common ESP-related
Common ESP-related
Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus ----).  Highlighted letters
are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?>
for Help, </> for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

                    [ ] Enable esp_timer profiling features
                    [*] Enable lookup of error code strings
                    (32) System event queue size
                    (2304) Event loop task stack size
                    (3584) Main task stack size
                    (1024) Inter-Processor Call (IPC) task stack size
                    (3584) High-resolution timer task stack size
                      UART for console output (Default: UART0, TX=GPIO1, RX=GPIO3)  --->
                    (115200) UART console baud rate
                    [*] Interrupt watchdog
                    (300) Interrupt watchdog timeout (ms)
                    [*]   Also watch CPU1 tick interrupt
                    [*] Initialize Task Watchdog Timer on startup
                    [ ]   Invoke panic handler on Task Watchdog timeout
                    (5)   Task Watchdog timeout period (seconds)
                    [*]   Watch CPU0 Idle Task
(+)

                    <Select>    < Exit >    < Help >    < Save >    < Load >
```

ESP32 UART Echo Test



GPIO4 -- USB TTL RX
GPIO5 -- USB TTL TX
GND -- GND

ESP32 UART Echo Test



✓ Echo Test Flow

PC 사용자 입력 – ESP32 수신 – ESP32 송신 – PC 사용자 확인

esp_err_t **uart_driver_install**(uart_port_t *uart_num*, int *rx_buffer_size*, int *tx_buffer_size*, int *queue_size*, QueueHandle_t * *uart_queue*, int *intr_alloc_flags*)

Install UART driver and set the UART to the default configuration.

UART ISR handler will be attached to the same CPU core that this function is running on.

Note

Rx_buffer_size should be greater than UART_FIFO_LEN. Tx_buffer_size should be either zero or greater than UART_FIFO_LEN.

Return

- ESP_OK Success
- ESP_FAIL Parameter error

Parameters

- `uart_num` : UART port number, the max port number is (UART_NUM_MAX -1).
- `rx_buffer_size` : UART RX ring buffer size.
- `tx_buffer_size` : UART TX ring buffer size. If set to zero, driver will not use TX buffer, TX function will block task until all data have been sent out.
- `queue_size` : UART event queue size/depth.
- `uart_queue` : UART event queue handle (out param). On success, a new queue handle is written here to provide access to UART events. If set to NULL, driver will not use an event queue.
- `intr_alloc_flags` : Flags used to allocate the interrupt. One or multiple (ORred) ESP_INTR_FLAG_* values. See esp_intr_alloc.h for more info. Do not set ESP_INTR_FLAG_IRAM here (the driver's ISR handler is not located in IRAM)

```
uart_driver_install(UART_NUM_1, BUF_SIZE * 2, 0, 0, NULL, 0);
```

```
/* Configure parameters of an UART driver,
 * communication pins and install the driver */
uart_config_t uart_config = {
    .baud_rate = 115200,
    .data_bits = UART_DATA_8_BITS,
    .parity    = UART_PARITY_DISABLE,
    .stop_bits = UART_STOP_BITS_1,
    .flow_ctrl = UART_HW_FLOWCTRL_DISABLE,
    .source_clk = UART_SCLK_APB,
};
```

```
/**
 * @brief UART configuration parameters for uart_param_config function
 */
typedef struct {
    int baud_rate;                      /*!< UART baud rate*/
    uart_word_length_t data_bits;       /*!< UART byte size*/
    uart_parity_t parity;               /*!< UART parity mode*/
    uart_stop_bits_t stop_bits;         /*!< UART stop bits*/
    uart_hw_flowcontrol_t flow_ctrl;    /*!< UART HW flow control mode (cts/rt
    uint8_t rx_flow_ctrl_thresh;        /*!< UART HW RTS threshold*/
    union {
        uart_sclk_t source_clk;         /*!< UART source clock selection */
        bool use_ref_tick  __attribute__((deprecated)); /*!< Deprecated method
    };
} uart_config_t;

#ifdef __cplusplus
}
#endif
"~/work/ESP-GIT/esp-idf/components/soc/include/hal/uart_types.h" 146L, 5724C
```

`esp_err_t uart_param_config(uart_port_t uart_num, const uart_config_t *uart_config)`

Set UART configuration parameters.

Return

- ESP_OK Success
- ESP_FAIL Parameter error

Parameters

- `uart_num` : UART port number, the max port number is (UART_NUM_MAX -1).
- `uart_config` : UART parameter settings

`esp_err_t uart_set_pin(uart_port_t uart_num, int tx_io_num, int rx_io_num, int rts_io_num, int cts_io_num)`

Set UART pin number.

Note

Internal signal can be output to multiple GPIO pads. Only one GPIO pad can connect with input signal.

Note

Instead of GPIO number a macro 'UART_PIN_NO_CHANGE' may be provided to keep the currently allocated pin.

Return

- ESP_OK Success
- ESP_FAIL Parameter error

Parameters

- `uart_num` : UART port number, the max port number is (UART_NUM_MAX -1).
- `tx_io_num` : UART TX pin GPIO number.
- `rx_io_num` : UART RX pin GPIO number.
- `rts_io_num` : UART RTS pin GPIO number.
- `cts_io_num` : UART CTS pin GPIO number.

```
uart_param_config(UART_NUM_1, &uart_config);
uart_set_pin(UART_NUM_1, ECHO_TEST_TXD, ECHO_TEST_RXD, ECHO_TEST_RTS, ECHO_TEST_CTS);
```

int **uart_read_bytes**(uart_port_t *uart_num*, uint8_t **buf*, uint32_t *length*, TickType_t *ticks_to_wait*)

UART read bytes from UART buffer.

Return

- (-1) Error
- OTHERS (>=0) The number of bytes read from UART FIFO

Parameters

- `uart_num` : UART port number, the max port number is (UART_NUM_MAX -1).
- `buf` : pointer to the buffer.
- `length` : data length
- `ticks_to_wait` : sTimeout, count in RTOS ticks

int **uart_write_bytes**(uart_port_t *uart_num*, *const* char **src*, size_t *size*)

Send data to the UART port from a given buffer and length,.

If the UART driver's parameter 'tx_buffer_size' is set to zero: This function will not return until all the data have been sent out, or at least pushed into TX FIFO.

Otherwise, if the 'tx_buffer_size' > 0, this function will return after copying all the data to tx ring buffer, UART ISR will then move data from the ring buffer to TX FIFO gradually.

Return

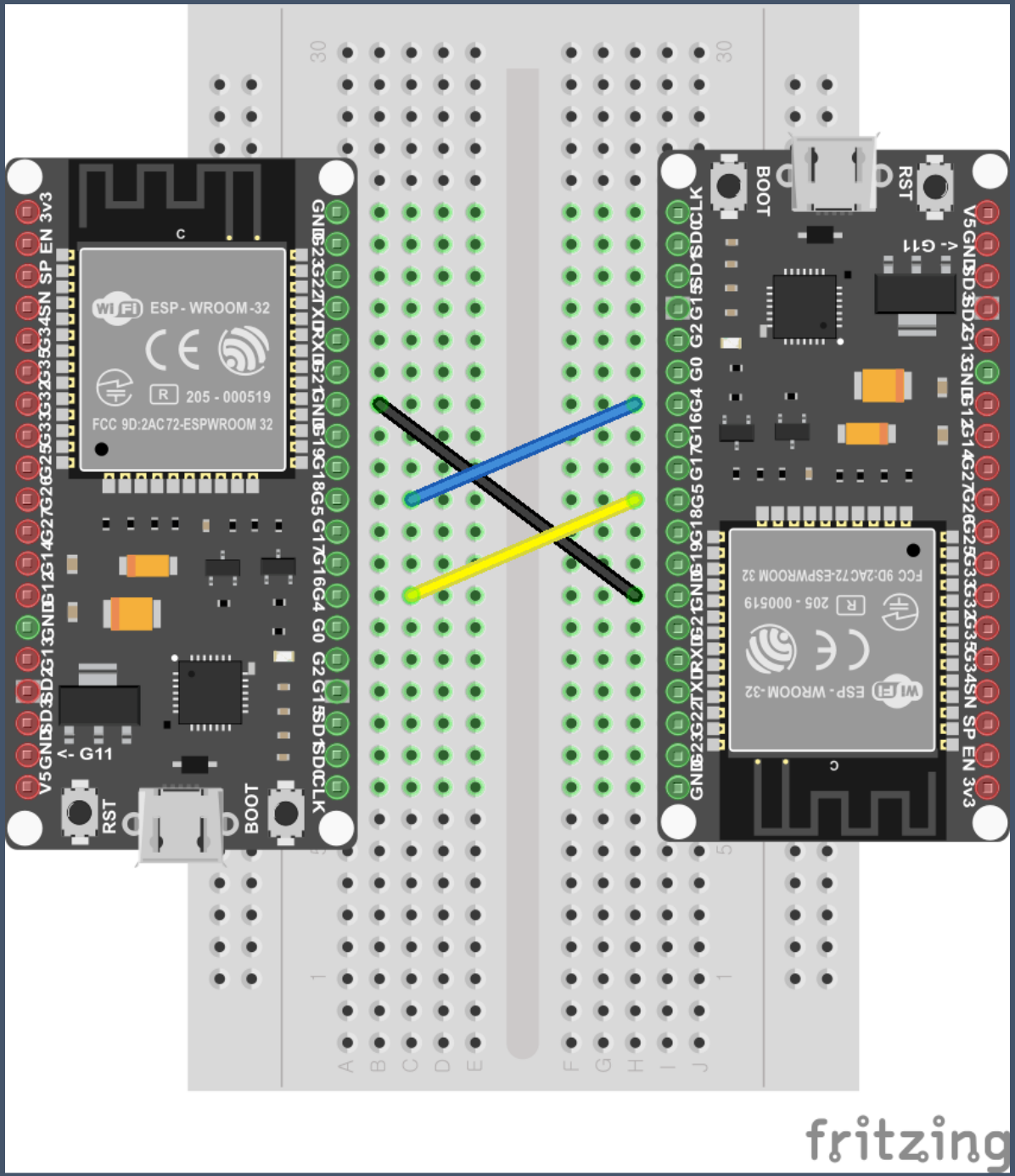
- (-1) Parameter error
- OTHERS (>=0) The number of bytes pushed to the TX FIFO

Parameters

- `uart_num` : UART port number, the max port number is (UART_NUM_MAX -1).
- `src` : data buffer address
- `size` : data length to send

```
// Configure a temporary buffer for the incoming data
uint8_t *data = (uint8_t *) malloc(BUF_SIZE);

while (1) {
    // Read data from the UART
    int len = uart_read_bytes(UART_NUM_1, data, BUF_SIZE, 20 / portTICK_RATE_MS);
    // Write data back to the UART
    uart_write_bytes(UART_NUM_1, (const char *) data, len);
}
```

ESP32 UART Async Task Test



GPIO4 -- GPIO5
GPIO5 -- GPIO4
GND -- GND

ESP32 UART Async Task Test

COM9 - Tera Term VT

File Edit Setup Control Window Help

```
I (700755) RX_TASK: 0x3ffb3fc0   48 65 6c 6c 6f 20 77 6f  72 6c 64                |Hello
world|
I (702315) TX_TASK: Wrote 11 bytes
I (702755) RX_TASK: Read 11 bytes: 'Hello world'
I (702755) RX_TASK: 0x3ffb3fc0   48 65 6c 6c 6f 20 77 6f  72 6c 64                |Hello
world|
I (704315) TX_TASK: Wrote 11 bytes
I (704755) RX_TASK: Read 11 bytes: 'Hello world'
I (704755) RX_TASK: 0x3ffb3fc0   48 65 6c 6c 6f 20 77 6f  72 6c 64                |Hello
world|
I (706315) TX_TASK: Wrote 11 bytes
I (706755) RX_TASK: Read 11 bytes: 'Hello world'
I (706755) RX_TASK: 0x3ffb3fc0   48 65 6c 6c 6f 20 77 6f  72 6c 64                |Hello
world|
I (708315) TX_TASK: Wrote 11 bytes
I (708755) RX_TASK: Read 11 bytes: 'Hello world'
I (708755) RX_TASK: 0x3ffb3fc0   48 65 6c 6c 6f 20 77 6f  72 6c 64                |Hello
world|
I (710315) TX_TASK: Wrote 11 bytes
I (710755) RX_TASK: Read 11 bytes: 'Hello world'
I (710755) RX_TASK: 0x3ffb3fc0   48 65 6c 6c 6f 20 77 6f  72 6c 64                |Hello
world|
I (712315) TX_TASK: Wrote 11 bytes
I (712755) RX_TASK: Read 11 bytes: 'Hello world'
I (712755) RX_TASK: 0x3ffb3fc0   48 65 6c 6c 6f 20 77 6f  72 6c 64                |Hello
world|
I (714315) TX_TASK: Wrote 11 bytes
I (714755) RX_TASK: Read 11 bytes: 'Hello world'
I (714755) RX_TASK: 0x3ffb3fc0   48 65 6c 6c 6f 20 77 6f  72 6c 64                |Hello
world|
I (716315) TX_TASK: Wrote 11 bytes
I (716755) RX_TASK: Read 11 bytes: 'Hello world'
I (716755) RX_TASK: 0x3ffb3fc0   48 65 6c 6c 6f 20 77 6f  72 6c 64                |Hello
world|
```

ESP32 UART Async Task Test

```
static const char *TX_TASK_TAG = "TX_TASK";
esp_log_level_set(TX_TASK_TAG, ESP_LOG_INFO);

ESP_LOGI(logName, "Wrote %d bytes", txBytes);
```

» API Reference » System API » Logging library

Logging library

```
static const char *RX_TASK_TAG = "RX_TASK";
esp_log_level_set(RX_TASK_TAG, ESP_LOG_INFO);

ESP_LOGI(RX_TASK_TAG, "Read %d bytes: '%s'", rxBytes, data);
ESP_LOG_BUFFER_HEXDUMP(RX_TASK_TAG, data, rxBytes, ESP_LOG_INFO);
```

어떤 의미가 있을까요?

ESP32 UART Async Task Test

#include "esp_log.h"

- `ESP_LOGE` - error (lowest)
- `ESP_LOGW` - warning
- `ESP_LOGI` - info
- `ESP_LOGD` - debug
- `ESP_LOGV` - verbose (highest)

» API Reference » System API » Logging library

Logging library

```
esp_log_level_set("*", ESP_LOG_ERROR);        // set all components to ERROR level
esp_log_level_set("wifi", ESP_LOG_WARN);      // enable WARN logs from WiFi stack
esp_log_level_set("dhcpc", ESP_LOG_INFO);     // enable INFO logs from DHCP client
```

void **esp_log_level_set**(*const* char **tag*, esp_log_level_t *level*)

Set log level for given tag.

If logging for given component has already been enabled, changes previous setting.

Note that this function can not raise log level above the level set using CONFIG_LOG_DEFAULT_LEVEL setting in menuconfig.

To raise log level above the default one for a given file, define LOG_LOCAL_LEVEL to one of the ESP_LOG_* values, before including esp_log.h in this file.

Parameters

- `tag` : Tag of the log entries to enable. Must be a non-NULL zero terminated string. Value "*" resets log level for all tags to the given value.
- `level` : Selects log level to enable. Only logs at this and lower verbosity levels will be shown.

3. ESP32 Timer & Timer Interrupt

- ✓ A 16-bit clock prescaler, from 2 to 65536
- ✓ A 64-bit time-base counter
- ✓ Configurable up/down time-base counter: incrementing or decrementing
- ✓ Halt and resume of time-base counter
- ✓ Auto-reload at alarm
- ✓ Software-controlled instant reload
- ✓ Level and edge interrupt generation

ESP32 최초로 만났던 타이머 ??

```
36     for (int i = 10; i >= 0; i--) {
37         printf("Restarting in %d seconds...\n", i);
38         vTaskDelay(1000 / portTICK_PERIOD_MS);
39     }
40     printf("Restarting now.\n");
41     fflush(stdout);
42     esp_restart();
43 }
~
~
~
~
~
"hello_world_main.c" 43L, 1344C
```

???

vTaskDelay

task. h

```
void vTaskDelay( portTickType xTicksToDelay );
```

Delay a task for a given number of ticks. The actual time that the task remains blocked depends on the tick rate. The constant portTICK_RATE_MS can be used to calculate real time from the tick rate - with the resolution of one tick period.

INCLUDE_vTaskDelay must be defined as 1 for this function to be available. See the configuration section for more information.

vTaskDelay() specifies a time at which the task wishes to unblock relative to the time at which vTaskDelay() is called. For example, specifying a block period of 100 ticks will cause the task to unblock 100 ticks after vTaskDelay() is called. vTaskDelay() does not therefore provide a good method of controlling the frequency of a cyclical task as the path taken through the code, as well as other task and interrupt activity, will effect the frequency at which vTaskDelay() gets called and therefore the time at which the task next executes. See vTaskDelayUntil() for an alternative API function designed to facilitate fixed frequency execution. It does this by specifying an absolute time (rather than a relative time) at which the calling task should unblock.

Parameters:

xTicksToDelay The amount of time, in tick periods, that the calling task should block.

Example usage:

```
void vTaskFunction( void * pvParameters )
{
// Block for 500ms.
const portTickType xDelay = 500 / portTICK_RATE_MS;

 for( ;; )
 {
  // Simply toggle the LED every 500ms, blocking between each toggle.
  vToggleLED();
  vTaskDelay( xDelay );
 }
}
```

http://web.ist.utl.pt/~ist11993/FRTOS-API/group___task_ctrl.html

ESP32 64bit Timer 를 다루기 위해서는

- ✓ 타이머 초기화 -타이머가 작동하도록 설정해야 하는 매개 변수와 타이머 구성에 따라 제공되는 특정 기능
- ✓ 타이머 제어 -타이머 값을 읽고, 타이머를 일시 중지 또는 시작하고, 작동 방식을 변경하는 방법
- ✓ 알람 - 알람 설정 및 사용 방법
- ✓ 인터럽트 - 인터럽트 를 활성화하고 사용하는 방법

ESP32 64bit Timer 를 다루기 위해서는

- ✓ 타이머 초기화 -타이머가 작동하도록 설정해야 하는 매개 변수와 타이머 구성에 따라 제공되는 특정 기능
- ✓ 타이머 제어 -타이머 값을 읽고, 타이머를 일시 중지 또는 시작하고, 작동 방식을 변경하는 방법
- ✓ 알람 - 알람 설정 및 사용 방법
- ✓ 인터럽트 - 인터럽트 를 활성화하고 사용하는 방법

✓ 타이머 초기화, 제어

```c
 * Initialize selected timer of the timer group 0
 *
 * timer_idx - the timer number to initialize
 * auto_reload - should the timer auto reload on alarm?
 * timer_interval_sec - the interval of alarm to set
 */
static void example_tg0_timer_init(int timer_idx,
                                   bool auto_reload, double timer_interval_sec)
{
    /* Select and initialize basic parameters of the timer */
    timer_config_t config = {
        .divider = TIMER_DIVIDER,
        .counter_dir = TIMER_COUNT_UP,
        .counter_en = TIMER_PAUSE,
        .alarm_en = TIMER_ALARM_EN,
        .auto_reload = auto_reload,
    }; // default clock source is APB
    timer_init(TIMER_GROUP_0, timer_idx, &config);

    /* Timer's counter will initially start from value below.
       Also, if auto_reload is set, this value will be automatically reload on alarm */
    timer_set_counter_value(TIMER_GROUP_0, timer_idx, 0x00000000ULL);

    /* Configure the alarm value and the interrupt on alarm. */
    timer_set_alarm_value(TIMER_GROUP_0, timer_idx, timer_interval_sec * TIMER_SCALE);
    timer_enable_intr(TIMER_GROUP_0, timer_idx);
    timer_isr_register(TIMER_GROUP_0, timer_idx, timer_group0_isr,
                       (void *) timer_idx, ESP_INTR_FLAG_IRAM, NULL);

    timer_start(TIMER_GROUP_0, timer_idx);
}
```

✓ 타이머 알람, 인터럽트

```c
/*
 * Timer group0 ISR handler
 *
 * Note:
 * We don't call the timer API here because they are not declared with IRAM_ATTR.
 * If we're okay with the timer irq not being serviced while SPI flash cache is disabled,
 * we can allocate this interrupt without the ESP_INTR_FLAG_IRAM flag and use the normal API.
 */
void IRAM_ATTR timer_group0_isr(void *para)
{
    timer_spinlock_take(TIMER_GROUP_0);
    int timer_idx = (int) para;

    /* Retrieve the interrupt status and the counter value
       from the timer that reported the interrupt */
    uint32_t timer_intr = timer_group_get_intr_status_in_isr(TIMER_GROUP_0);
    uint64_t timer_counter_value = timer_group_get_counter_value_in_isr(TIMER_GROUP_0, timer_idx);

    /* Prepare basic event data
       that will be then sent back to the main program task */
    timer_event_t evt;
    evt.timer_group = 0;
    evt.timer_idx = timer_idx;
    evt.timer_counter_value = timer_counter_value;

    /* Clear the interrupt
       and update the alarm time for the timer with without reload */
    if (timer_intr & TIMER_INTR_T0) {
        evt.type = TEST_WITHOUT_RELOAD;
        timer_group_clr_intr_status_in_isr(TIMER_GROUP_0, TIMER_0);
        timer_counter_value += (uint64_t) (TIMER_INTERVAL0_SEC * TIMER_SCALE);
        timer_group_set_alarm_value_in_isr(TIMER_GROUP_0, timer_idx, timer_counter_value);
    } else if (timer_intr & TIMER_INTR_T1) {
        evt.type = TEST_WITH_RELOAD;
        timer_group_clr_intr_status_in_isr(TIMER_GROUP_0, TIMER_1);
    } else {
        evt.type = -1; // not supported even type
    }

    /* After the alarm has been triggered
      we need enable it again, so it is triggered the next time */
    timer_group_enable_alarm_in_isr(TIMER_GROUP_0, timer_idx);

    /* Now just send the event data back to the main program task */
    xQueueSendFromISR(timer_queue, &evt, NULL);
    timer_spinlock_give(TIMER_GROUP_0);
}
```

✓퀴즈 : 1초 마다 알람을 주는 나만의 타이머 만들어 보기

감사합니다.