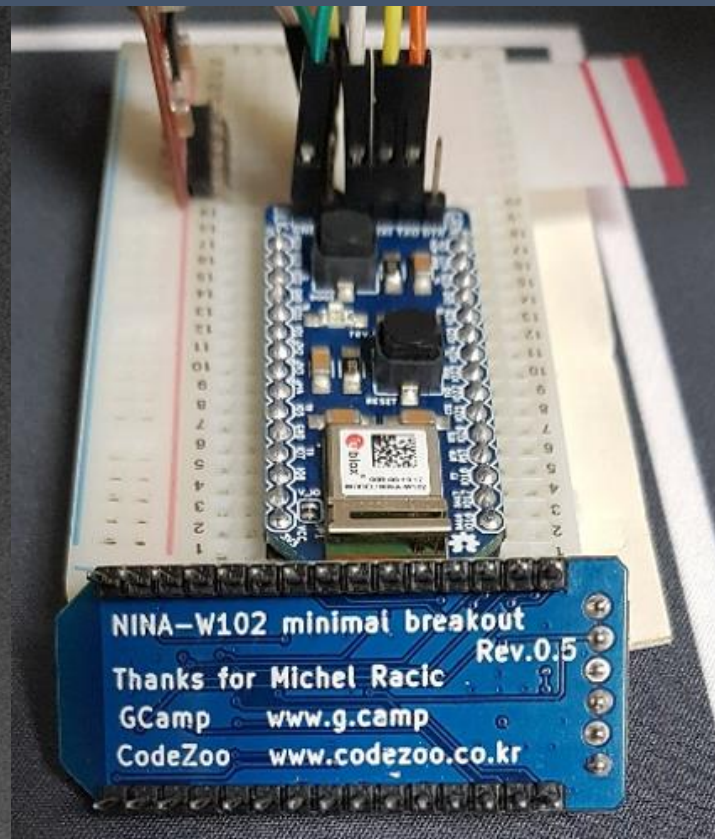
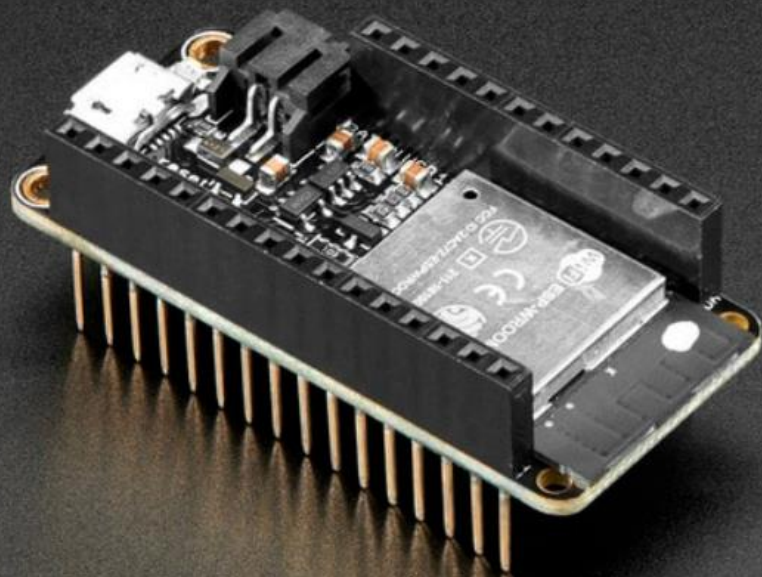


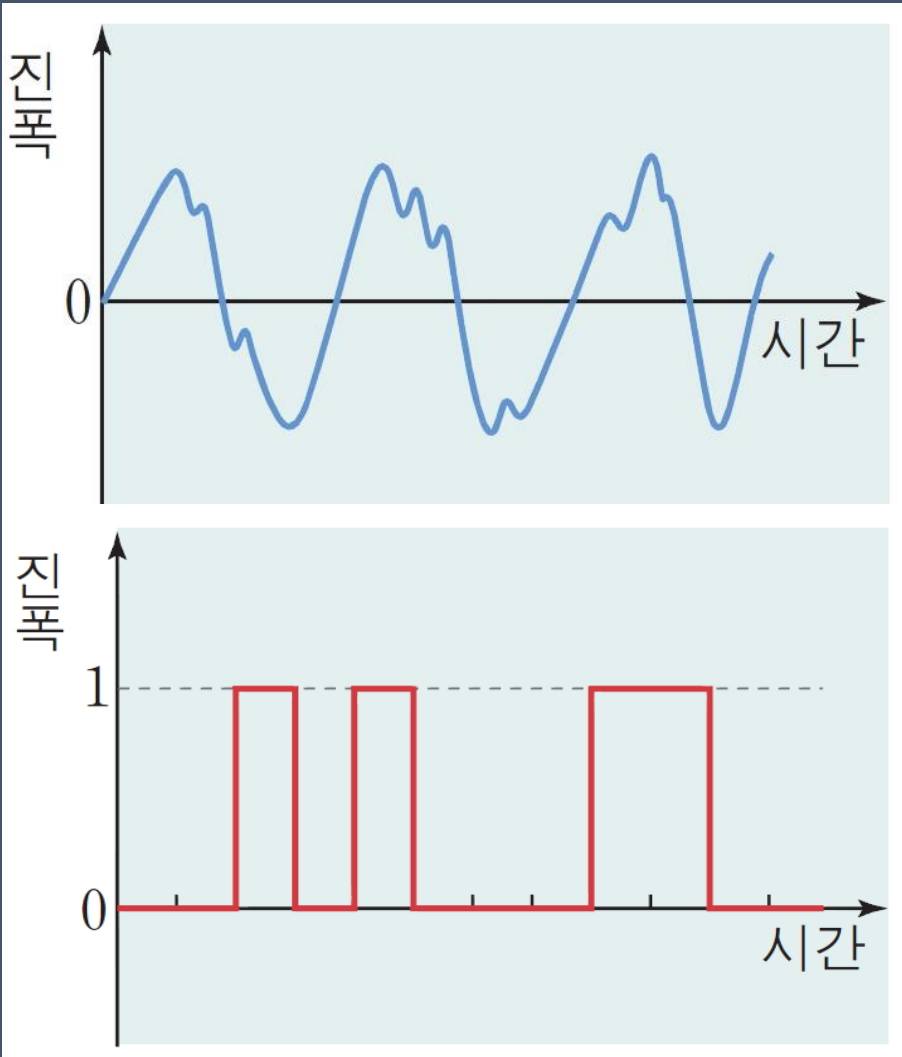
ESP32

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

0. 두번째 Hello World (NINA-W102)



1. hello_world\ sdkconfig 변경

```
46 # CONFIG_ESPTOOLPY_FLASHSIZE_1MB is not set
47 CONFIG_ESPTOOLPY_FLASHSIZE_2MB=y
48 # CONFIG_ESPTOOLPY_FLASHSIZE_4MB is not set
49 # CONFIG_ESPTOOLPY_FLASHSIZE_8MB is not set
50 # CONFIG_ESPTOOLPY_FLASHSIZE_16MB is not set
```

0. 두번째 Hello World

2. 소스코드 빌드

```
Generated C:/Users/jbmas/work/gcamp/get-started/hello_world/build/hello-world.bin
```

Project build complete. To flash, run this command:

```
C:\Users\jbmas\espressif\python_env\idf4.0_py3.7_env\Scripts\python.exe ..\..\..\wesp-idf\components\wesp-tool_py\wesp-tool\wesp-tool.py -p (PORT) -b 460800 --before default_reset --after hard_reset write_flash --flash_mode dio --flash_size detect --flash_freq 40m 0x1000 build\bootloader\bootloader.bin 0x8000 build\partition_table\partition-table.bin 0x10000 build\hello-world.bin  
or run 'idf.py -p (PORT) flash'
```

```
C:\Users\jbmas\work\gcamp\get-started\hello_world>
```


0. 두번째 Hello World

3. 문제상황 발생

```
(212) cpu_start: App cpu up.
I (237) heap_init: Initializing. RAM available for dynamic allocation:
I (243) heap_init: At 3FFAE6E0 len 00001920 (6 KiB): DRAM
I (249) heap_init: At 3FFB3100 len 0002CF00 (179 KiB): DRAM
I (256) heap_init: At 3FFE0440 len 00003AE0 (14 KiB): D/IRAM
I (262) heap_init: At 3FFE4350 len 0001BCB0 (111 KiB): D/IRAM
I (268) heap_init: At 40089A84 len 0001657C (89 KiB): IRAM
I (275) cpu_start: Pro cpu start user code
I (293) spi_flash: detected chip: generic
E (293) spi_flash: failed to get chip size
assertion "flash ret == ESP_OK" failed: file C:/Users/jbmas/work/esp-idf/components/esp_flash/spi_flash.c, line 400, function: spi_flash_get_chip_size
abort() was called at PC 0x400d56df on core 0

ELF file SHA256: b85b298addffa8b5

Backtrace: 0x4008491d:0x3ffe3b30 0x40084c9d:0x3ffe3b50 0x400d56df:0x3ffe3b70 0x400d56df:0x3ffe3ba0 0x40081243:0x3ffe3bf0 0x40079497:0x3ffe3c30 |<-CORRUPTED

Rebooting...
ets Jun  8 2016 00:22:57

rst:0xc (SW_CPU_RESET),boot:0x33 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0018,len:4
load:0x3fff001c,len:6892
load:0x40078000,len:14076
```

spi_flash에서 chip size 에러 발생 후
시스템 무한 재부팅되는 현상

0. 두번째 Hello World

4. 솔루션

- 1) ESP IDF SDK 3.x version 사용
- 2) CONFIG_SPI_FLASH_USE_LEGACY_IMPL 설정 활성화
: 이유> SDK 4.x versio으로 올라가면서 SPI Flash 드라이버의 대규모 변경으로 인해 기존 SPI Flash 하드웨어와 호환성 이슈 발생.

```
485 # CONFIG_SPI_FLASH_DANGEROUS_WRITE_FAILS is not set
486 # CONFIG_SPI_FLASH_DANGEROUS_WRITE_ALLOWED is not set
487 CONFIG_SPI_FLASH_USE_LEGACY_IMPL=y
488 CONFIG_SPI_FLASH_SUPPORT_ISSI_CHIP=y
489 CONFIG_SPI_FLASH_SUPPORT_GD_CHIP=y
490 CONFIG_SPIFFS_MAX_PARTITIONS=3
```

0. 두번째 Hello World

```
I (276) cpu_start: Pro cpu start user code
I (294) cpu_start: Starting scheduler on PRO CPU.
I (0) cpu_start: Starting scheduler on APP CPU.
Hello world!
This is ESP32 chip with 2 CPU cores, WiFi/BT/BLE, silicon revision 1, 2MB external flash
Restarting in 10 seconds...
Restarting in 9 seconds...
Restarting in 8 seconds...
Restarting in 7 seconds...
Restarting in 6 seconds...
Restarting in 5 seconds...
Restarting in 4 seconds...
Restarting in 3 seconds...
Restarting in 2 seconds...
Restarting in 1 seconds...
_
```

Clear !!!

0. 두번째 Hello World

5. 솔루션 결론 (2)

1) ESP IDF SDK 3.x version 사용

2) CONFIG_SPI_FLASH_USE_LEGACY_IMPL 설정 활성화

: 이유> SDK 4.x versio으로 올라가면서 SPI Flash 드라이버의 대규모 변경으로 인해 기존 SPI Flash 하드웨어와 호환성 이슈 발생.

CMakeLists.txt 에 사용하는 idf_component_register 와 같은 설정명령들이 SDK3.X 대에서는 지원이 안됨. 최신 리소스를 사용하려면 설정파일을 수정해야 하는데 시간과 분량이 적지 않음

ESP32 GPIO specification

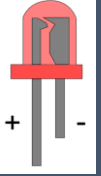
- ESP32 칩에는 40 개의 물리적 **GPIO** 패드가 있음
 - 일부 **GPIO** 패드는 사용할 수 없거나 칩 패키지에 해당 핀이 없음 (**DataSheet** 참조).
 - 각 패드는 범용 **I/O**로 사용되거나 내부 주변 신호에 연결 될 수 있음
-
- ✓ **GPIO 6 ~ 11**은 일반적으로 **SPI** 플래시에 사용
 - ✓ **GPIO 34 ~ 39**는 입력 모드로만 설정할 수 있으며 소프트웨어 풀업 또는 풀다운 기능이 없음 (중요!!)
 - ✓ **GPIO**가 **RTC**저전력 및 아날로그 하위 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 _{BU}	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 epoxy 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
Wavelength (nm) or TC(k)	Δ λ	I _F =20mA	620	---	625	nm
*Luminous intensity	I _v	I _F =20mA	150	---	200	mcd
50% Viewing Angle	2 θ 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

$V = I \cdot R$ (V:전압, I:전류, R:저항)

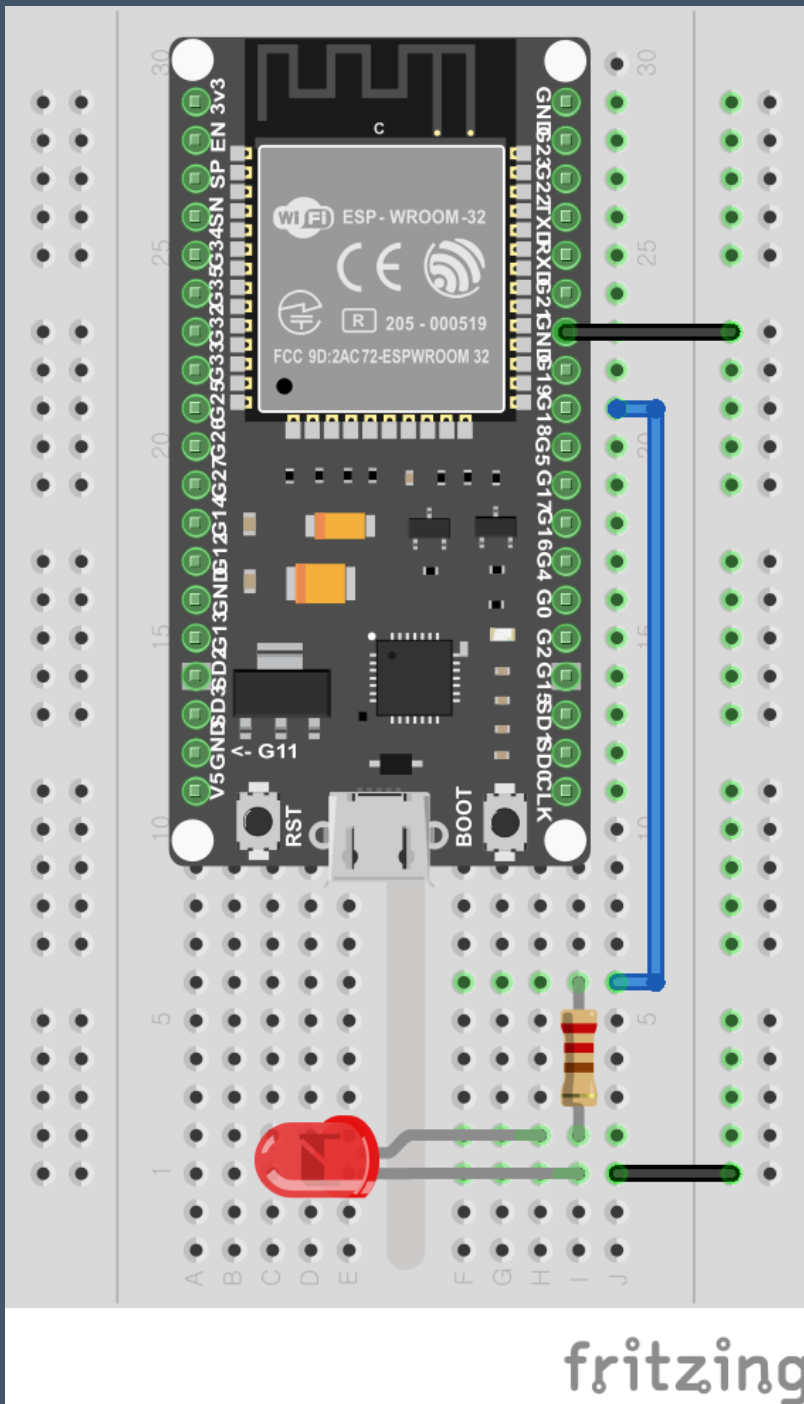
$R = V/I \rightarrow \text{저항값} = (\text{공급전압} - \text{LED전압})/\text{LED 소모전류}$

예) LED의 최소구동전압: 2V

LED의 소모전류: 20mA

디지털출력의 전압: 3.3V

$$(3.3V - 2V)/0.02A = 65\Omega$$



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

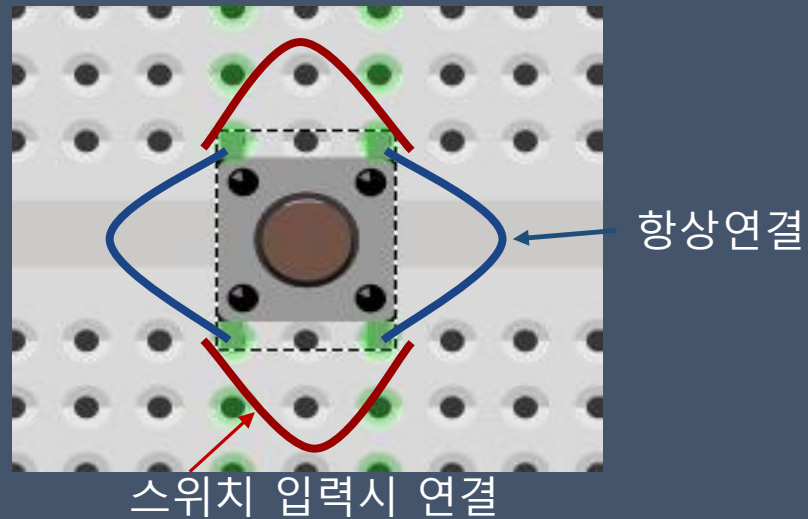
Parameters

- `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_num
 * @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



FreeRTOS Elements

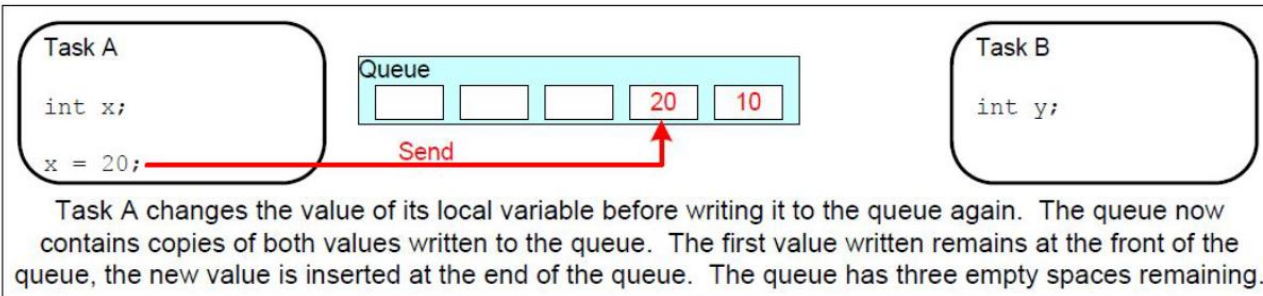
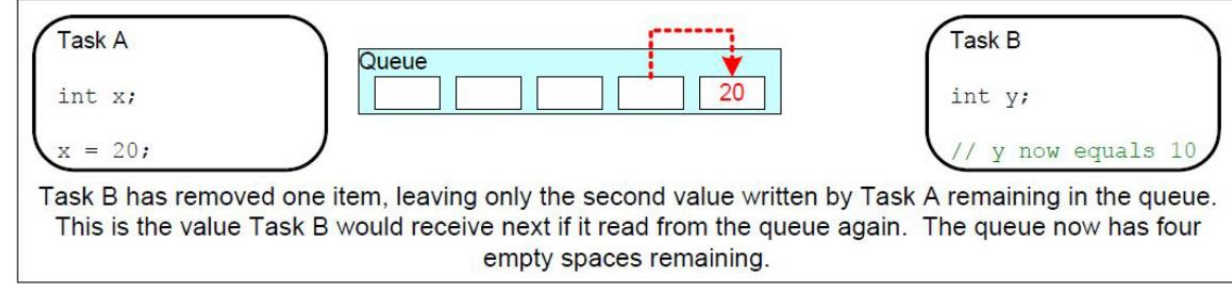
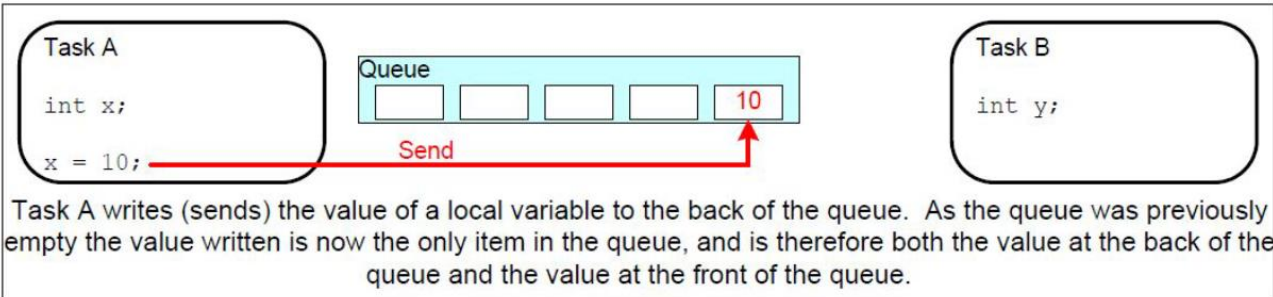
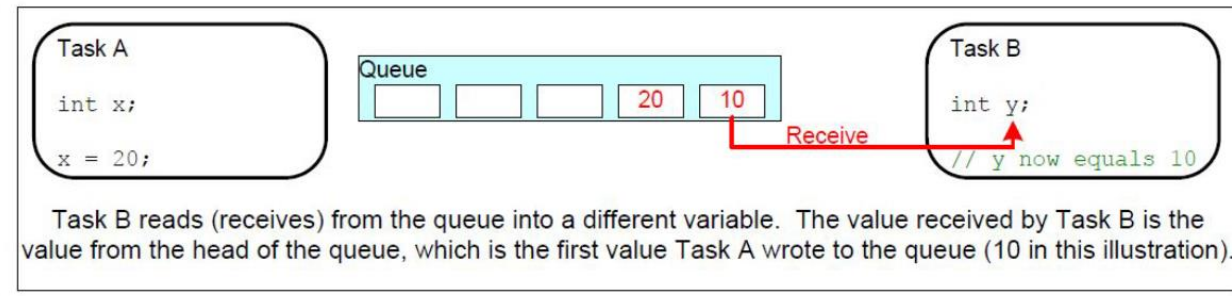
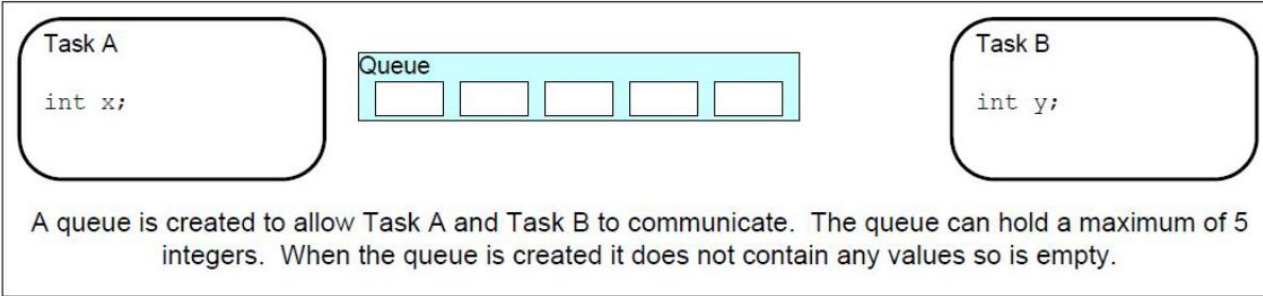
```
//create a queue to handle gpio event from isr
    gpio_evt_queue = xQueueCreate(10, sizeof(uint32_t));

//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



```
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 μ A.

```
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.

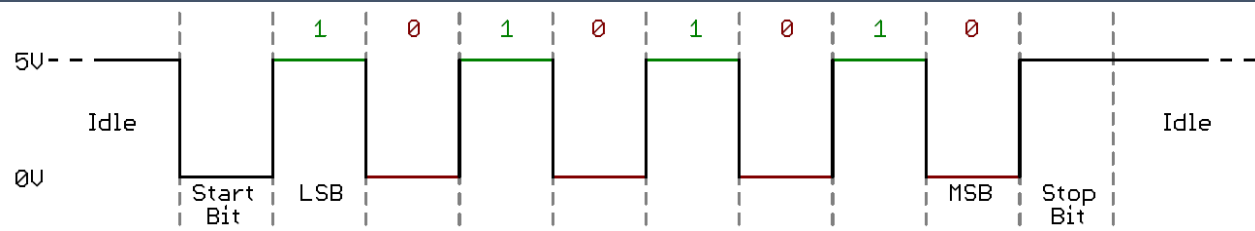
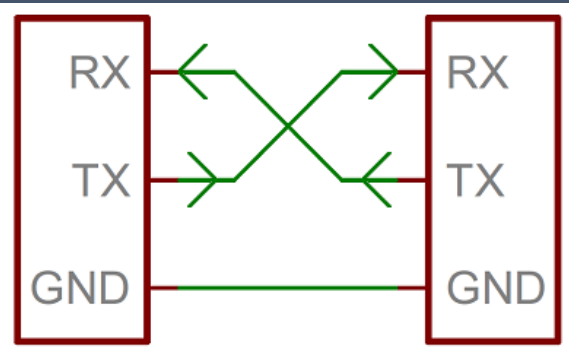
Return

- 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

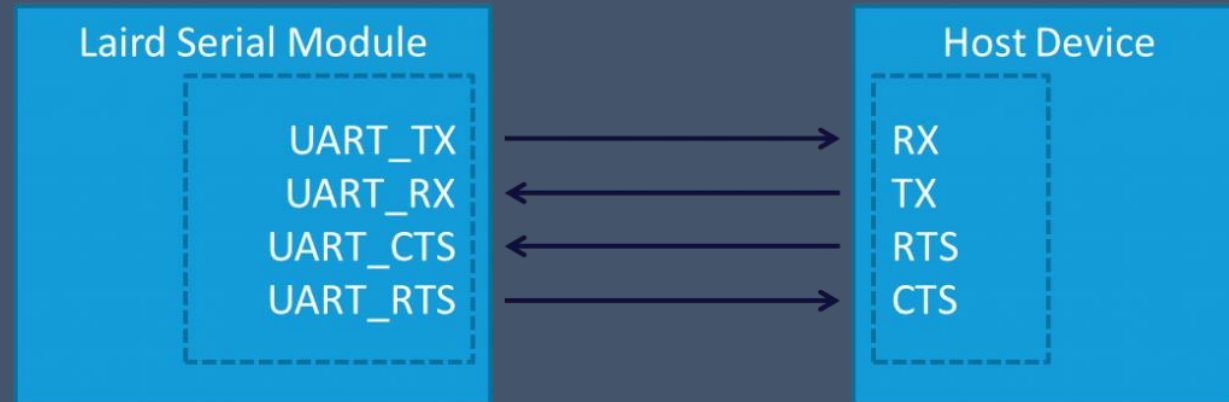


자료 출처 : learn.sparkfun.com (TTL)



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-rtscs-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)가 잡힐 때까지 차단된
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     이렇게 감시할 fd 들은 어디에 담을까?
389     esp_timer_init();
390     esp_set_time_from_rtc();
391 #if CONFIG_APPTRACE_ENABLE특성에 따라 두 번째, 세 번째, 네 번째 인자에 담긴다.
392     err = esp_apprace_init();
393     assert(err == ESP_OK && "Failed to init apptrace module on PRO CPU!");
394 #endif
395 #if CONFIG_SYSVIEW_ENABLEwritefds 집합 : 쓰기 연산을 끝낼 수 있는지 확인할 fd 들
396     SEGGER_SYSVIEW_Conf();
397 #endif
398 #if CONFIG_ESP_DEBUG_STUBS_ENABLEreadfds 집합 : 예외가 발생했는지 확인할 fd 들
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
406     // WDT 초기화
407     esp_wdt_init();
408     // WDT 타이머 설정
409     esp_wdt_set_timeout(ESP_WDT_TIMEOUT_SECONDS);
410     // WDT 타이머 시작
411     esp_wdt_start();
412 #endif
413     return 0;
414 }
415
416 #endif // CONFIG_ESP_CONSOLE_UART_NONE
417
418 #endif // CONFIG_VFS_SUPPORT_IO
419
420 #endif // CONFIG_ESP_CONSOLE_UART_NONE
421
422 #endif // CONFIG_VFS_SUPPORT_IO
423
424 #endif // CONFIG_ESP_CONSOLE_UART_NONE
425
426 #endif // CONFIG_VFS_SUPPORT_IO
427
428 #endif // CONFIG_ESP_CONSOLE_UART_NONE
429
430 #endif // CONFIG_VFS_SUPPORT_IO
431
432 #endif // CONFIG_ESP_CONSOLE_UART_NONE
433
434 #endif // CONFIG_VFS_SUPPORT_IO
435
436 #endif // CONFIG_ESP_CONSOLE_UART_NONE
437
438 #endif // CONFIG_VFS_SUPPORT_IO
439
440 #endif // CONFIG_ESP_CONSOLE_UART_NONE
441
442 #endif // CONFIG_VFS_SUPPORT_IO
443
444 #endif // CONFIG_ESP_CONSOLE_UART_NONE
445
446 #endif // CONFIG_VFS_SUPPORT_IO
447
448 #endif // CONFIG_ESP_CONSOLE_UART_NONE
449
450 #endif // CONFIG_VFS_SUPPORT_IO
451
452 #endif // CONFIG_ESP_CONSOLE_UART_NONE
453
454 #endif // CONFIG_VFS_SUPPORT_IO
455
456 #endif // CONFIG_ESP_CONSOLE_UART_NONE
457
458 #endif // CONFIG_VFS_SUPPORT_IO
459
460 #endif // CONFIG_ESP_CONSOLE_UART_NONE
461
462 #endif // CONFIG_VFS_SUPPORT_IO
463
464 #endif // CONFIG_ESP_CONSOLE_UART_NONE
465
466 #endif // CONFIG_VFS_SUPPORT_IO
467
468 #endif // CONFIG_ESP_CONSOLE_UART_NONE
469
470 #endif // CONFIG_VFS_SUPPORT_IO
471
472 #endif // CONFIG_ESP_CONSOLE_UART_NONE
473
474 #endif // CONFIG_VFS_SUPPORT_IO
475
476 #endif // CONFIG_ESP_CONSOLE_UART_NONE
477
478 #endif // CONFIG_VFS_SUPPORT_IO
479
480 #endif // CONFIG_ESP_CONSOLE_UART_NONE
481
482 #endif // CONFIG_VFS_SUPPORT_IO
483
484 #endif // CONFIG_ESP_CONSOLE_UART_NONE
485
486 #endif // CONFIG_VFS_SUPPORT_IO
487
488 #endif // CONFIG_ESP_CONSOLE_UART_NONE
489
490 #endif // CONFIG_VFS_SUPPORT_IO
491
492 #endif // CONFIG_ESP_CONSOLE_UART_NONE
493
494 #endif // CONFIG_VFS_SUPPORT_IO
495
496 #endif // CONFIG_ESP_CONSOLE_UART_NONE
497
498 #endif // CONFIG_VFS_SUPPORT_IO
499
500 #endif // CONFIG_ESP_CONSOLE_UART_NONE
501
502 #endif // CONFIG_VFS_SUPPORT_IO
503
504 #endif // CONFIG_ESP_CONSOLE_UART_NONE
505
506 #endif // CONFIG_VFS_SUPPORT_IO
507
508 #endif // CONFIG_ESP_CONSOLE_UART_NONE
509
510 #endif // CONFIG_VFS_SUPPORT_IO
511
512 #endif // CONFIG_ESP_CONSOLE_UART_NONE
513
514 #endif // CONFIG_VFS_SUPPORT_IO
515
516 #endif // CONFIG_ESP_CONSOLE_UART_NONE
517
518 #endif // CONFIG_VFS_SUPPORT_IO
519
520 #endif // CONFIG_ESP_CONSOLE_UART_NONE
521
522 #endif // CONFIG_VFS_SUPPORT_IO
523
524 #endif // CONFIG_ESP_CONSOLE_UART_NONE
525
526 #endif // CONFIG_VFS_SUPPORT_IO
527
528 #endif // CONFIG_ESP_CONSOLE_UART_NONE
529
530 #endif // CONFIG_VFS_SUPPORT_IO
531
532 #endif // CONFIG_ESP_CONSOLE_UART_NONE
533
534 #endif // CONFIG_VFS_SUPPORT_IO
535
536 #endif // CONFIG_ESP_CONSOLE_UART_NONE
537
538 #endif // CONFIG_VFS_SUPPORT_IO
539
540 #endif // CONFIG_ESP_CONSOLE_UART_NONE
541
542 #endif // CONFIG_VFS_SUPPORT_IO
543
544 #endif // CONFIG_ESP_CONSOLE_UART_NONE
545
546 #endif // CONFIG_VFS_SUPPORT_IO
547
548 #endif // CONFIG_ESP_CONSOLE_UART_NONE
549
550 #endif // CONFIG_VFS_SUPPORT_IO
551
552 #endif // CONFIG_ESP_CONSOLE_UART_NONE
553
554 #endif // CONFIG_VFS_SUPPORT_IO
555
556 #endif // CONFIG_ESP_CONSOLE_UART_NONE
557
558 #endif // CONFIG_VFS_SUPPORT_IO
559
560 #endif // CONFIG_ESP_CONSOLE_UART_NONE
561
562 #endif // CONFIG_VFS_SUPPORT_IO
563
564 #endif // CONFIG_ESP_CONSOLE_UART_NONE
565
566 #endif // CONFIG_VFS_SUPPORT_IO
567
568 #endif // CONFIG_ESP_CONSOLE_UART_NONE
569
570 #endif // CONFIG_VFS_SUPPORT_IO
571
572 #endif // CONFIG_ESP_CONSOLE_UART_NONE
573
574 #endif // CONFIG_VFS_SUPPORT_IO
575
576 #endif // CONFIG_ESP_CONSOLE_UART_NONE
577
578 #endif // CONFIG_VFS_SUPPORT_IO
579
580 #endif // CONFIG_ESP_CONSOLE_UART_NONE
581
582 #endif // CONFIG_VFS_SUPPORT_IO
583
584 #endif // CONFIG_ESP_CONSOLE_UART_NONE
585
586 #endif // CONFIG_VFS_SUPPORT_IO
587
588 #endif // CONFIG_ESP_CONSOLE_UART_NONE
589
590 #endif // CONFIG_VFS_SUPPORT_IO
591
592 #endif // CONFIG_ESP_CONSOLE_UART_NONE
593
594 #endif // CONFIG_VFS_SUPPORT_IO
595
596 #endif // CONFIG_ESP_CONSOLE_UART_NONE
597
598 #endif // CONFIG_VFS_SUPPORT_IO
599
600 #endif // CONFIG_ESP_CONSOLE_UART_NONE
601
602 #endif // CONFIG_VFS_SUPPORT_IO
603
604 #endif // CONFIG_ESP_CONSOLE_UART_NONE
605
606 #endif // CONFIG_VFS_SUPPORT_IO
607
608 #endif // CONFIG_ESP_CONSOLE_UART_NONE
609
610 #endif // CONFIG_VFS_SUPPORT_IO
611
612 #endif // CONFIG_ESP_CONSOLE_UART_NONE
613
614 #endif // CONFIG_VFS_SUPPORT_IO
615
616 #endif // CONFIG_ESP_CONSOLE_UART_NONE
617
618 #endif // CONFIG_VFS_SUPPORT_IO
619
620 #endif // CONFIG_ESP_CONSOLE_UART_NONE
621
622 #endif // CONFIG_VFS_SUPPORT_IO
623
624 #endif // CONFIG_ESP_CONSOLE_UART_NONE
625
626 #endif // CONFIG_VFS_SUPPORT_IO
627
628 #endif // CONFIG_ESP_CONSOLE_UART_NONE
629
630 #endif // CONFIG_VFS_SUPPORT_IO
631
632 #endif // CONFIG_ESP_CONSOLE_UART_NONE
633
634 #endif // CONFIG_VFS_SUPPORT_IO
635
636 #endif // CONFIG_ESP_CONSOLE_UART_NONE
637
638 #endif // CONFIG_VFS_SUPPORT_IO
639
640 #endif // CONFIG_ESP_CONSOLE_UART_NONE
641
642 #endif // CONFIG_VFS_SUPPORT_IO
643
644 #endif // CONFIG_ESP_CONSOLE_UART_NONE
645
646 #endif // CONFIG_VFS_SUPPORT_IO
647
648 #endif // CONFIG_ESP_CONSOLE_UART_NONE
649
650 #endif // CONFIG_VFS_SUPPORT_IO
651
652 #endif // CONFIG_ESP_CONSOLE_UART_NONE
653
654 #endif // CONFIG_VFS_SUPPORT_IO
655
656 #endif // CONFIG_ESP_CONSOLE_UART_NONE
657
658 #endif // CONFIG_VFS_SUPPORT_IO
659
660 #endif // CONFIG_ESP_CONSOLE_UART_NONE
661
662 #endif // CONFIG_VFS_SUPPORT_IO
663
664 #endif // CONFIG_ESP_CONSOLE_UART_NONE
665
666 #endif // CONFIG_VFS_SUPPORT_IO
667
668 #endif // CONFIG_ESP_CONSOLE_UART_NONE
669
670 #endif // CONFIG_VFS_SUPPORT_IO
671
672 #endif // CONFIG_ESP_CONSOLE_UART_NONE
673
674 #endif // CONFIG_VFS_SUPPORT_IO
675
676 #endif // CONFIG_ESP_CONSOLE_UART_NONE
677
678 #endif // CONFIG_VFS_SUPPORT_IO
679
680 #endif // CONFIG_ESP_CONSOLE_UART_NONE
681
682 #endif // CONFIG_VFS_SUPPORT_IO
683
684 #endif // CONFIG_ESP_CONSOLE_UART_NONE
685
686 #endif // CONFIG_VFS_SUPPORT_IO
687
688 #endif // CONFIG_ESP_CONSOLE_UART_NONE
689
690 #endif // CONFIG_VFS_SUPPORT_IO
691
692 #endif // CONFIG_ESP_CONSOLE_UART_NONE
693
694 #endif // CONFIG_VFS_SUPPORT_IO
695
696 #endif // CONFIG_ESP_CONSOLE_UART_NONE
697
698 #endif // CONFIG_VFS_SUPPORT_IO
699
700 #endif // CONFIG_ESP_CONSOLE_UART_NONE
701
702 #endif // CONFIG_VFS_SUPPORT_IO
703
704 #endif // CONFIG_ESP_CONSOLE_UART_NONE
705
706 #endif // CONFIG_VFS_SUPPORT_IO
707
708 #endif // CONFIG_ESP_CONSOLE_UART_NONE
709
710 #endif // CONFIG_VFS_SUPPORT_IO
711
712 #endif // CONFIG_ESP_CONSOLE_UART_NONE
713
714 #endif // CONFIG_VFS_SUPPORT_IO
715
716 #endif // CONFIG_ESP_CONSOLE_UART_NONE
717
718 #endif // CONFIG_VFS_SUPPORT_IO
719
720 #endif // CONFIG_ESP_CONSOLE_UART_NONE
721
722 #endif // CONFIG_VFS_SUPPORT_IO
723
724 #endif // CONFIG_ESP_CONSOLE_UART_NONE
725
726 #endif // CONFIG_VFS_SUPPORT_IO
727
728 #endif // CONFIG_ESP_CONSOLE_UART_NONE
729
730 #endif // CONFIG_VFS_SUPPORT_IO
731
732 #endif // CONFIG_ESP_CONSOLE_UART_NONE
733
734 #endif // CONFIG_VFS_SUPPORT_IO
735
736 #endif // CONFIG_ESP_CONSOLE_UART_NONE
737
738 #endif // CONFIG_VFS_SUPPORT_IO
739
740 #endif // CONFIG_ESP_CONSOLE_UART_NONE
741
742 #endif // CONFIG_VFS_SUPPORT_IO
743
744 #endif // CONFIG_ESP_CONSOLE_UART_NONE
745
746 #endif // CONFIG_VFS_SUPPORT_IO
747
748 #endif // CONFIG_ESP_CONSOLE_UART_NONE
749
750 #endif // CONFIG_VFS_SUPPORT_IO
751
752 #endif // CONFIG_ESP_CONSOLE_UART_NONE
753
754 #endif // CONFIG_VFS_SUPPORT_IO
755
756 #endif // CONFIG_ESP_CONSOLE_UART_NONE
757
758 #endif // CONFIG_VFS_SUPPORT_IO
759
760 #endif // CONFIG_ESP_CONSOLE_UART_NONE
761
762 #endif // CONFIG_VFS_SUPPORT_IO
763
764 #endif // CONFIG_ESP_CONSOLE_UART_NONE
765
766 #endif // CONFIG_VFS_SUPPORT_IO
767
768 #endif // CONFIG_ESP_CONSOLE_UART_NONE
769
770 #endif // CONFIG_VFS_SUPPORT_IO
771
772 #endif // CONFIG_ESP_CONSOLE_UART_NONE
773
774 #endif // CONFIG_VFS_SUPPORT_IO
775
776 #endif // CONFIG_ESP_CONSOLE_UART_NONE
777
778 #endif // CONFIG_VFS_SUPPORT_IO
779
780 #endif // CONFIG_ESP_CONSOLE_UART_NONE
781
782 #endif // CONFIG_VFS_SUPPORT_IO
783
784 #endif // CONFIG_ESP_CONSOLE_UART_NONE
785
786 #endif // CONFIG_VFS_SUPPORT_IO
787
788 #endif // CONFIG_ESP_CONSOLE_UART_NONE
789
790 #endif // CONFIG_VFS_SUPPORT_IO
791
792 #endif // CONFIG_ESP_CONSOLE_UART_NONE
793
794 #endif // CONFIG_VFS_SUPPORT_IO
795
796 #endif // CONFIG_ESP_CONSOLE_UART_NONE
797
798 #endif // CONFIG_VFS_SUPPORT_IO
799
800 #endif // CONFIG_ESP_CONSOLE_UART_NONE
801
802 #endif // CONFIG_VFS_SUPPORT_IO
803
804 #endif // CONFIG_ESP_CONSOLE_UART_NONE
805
806 #endif // CONFIG_VFS_SUPPORT_IO
807
808 #endif // CONFIG_ESP_CONSOLE_UART_NONE
809
810 #endif // CONFIG_VFS_SUPPORT_IO
811
812 #endif // CONFIG_ESP_CONSOLE_UART_NONE
813
814 #endif // CONFIG_VFS_SUPPORT_IO
815
816 #endif // CONFIG_ESP_CONSOLE_UART_NONE
817
818 #endif // CONFIG_VFS_SUPPORT_IO
819
820 #endif // CONFIG_ESP_CONSOLE_UART_NONE
821
822 #endif // CONFIG_VFS_SUPPORT_IO
823
824 #endif // CONFIG_ESP_CONSOLE_UART_NONE
825
826 #endif // CONFIG_VFS_SUPPORT_IO
827
828 #endif // CONFIG_ESP_CONSOLE_UART_NONE
829
830 #endif // CONFIG_VFS_SUPPORT_IO
831
832 #endif // CONFIG_ESP_CONSOLE_UART_NONE
833
834 #endif // CONFIG_VFS_SUPPORT_IO
835
836 #endif // CONFIG_ESP_CONSOLE_UART_NONE
837
838 #endif // CONFIG_VFS_SUPPORT_IO
839
840 #endif // CONFIG_ESP_CONSOLE_UART_NONE
841
842 #endif // CONFIG_VFS_SUPPORT_IO
84
```

```
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

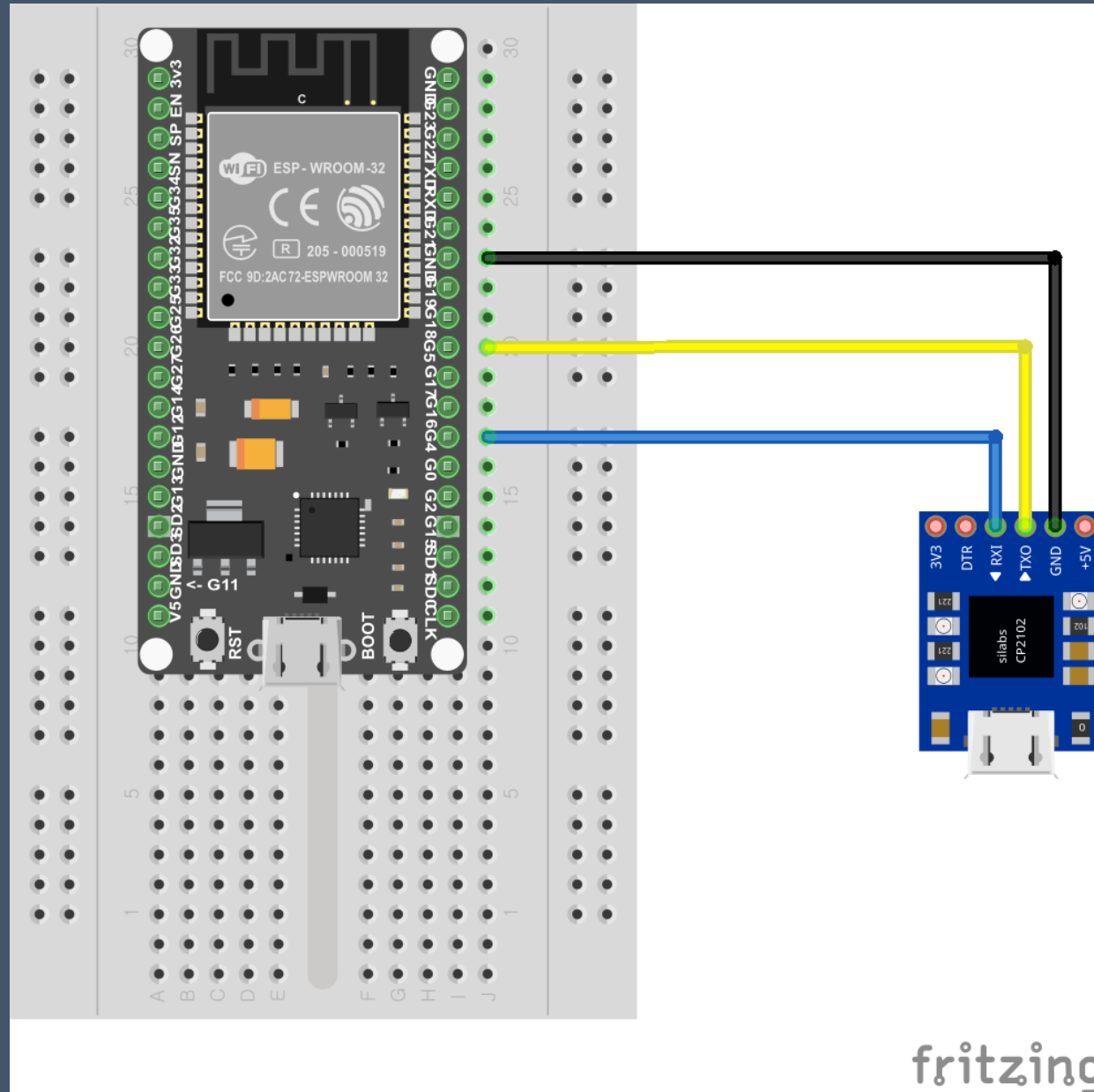
C:\Users\jbmas\work\esp_test>cd hello_world

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

```
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

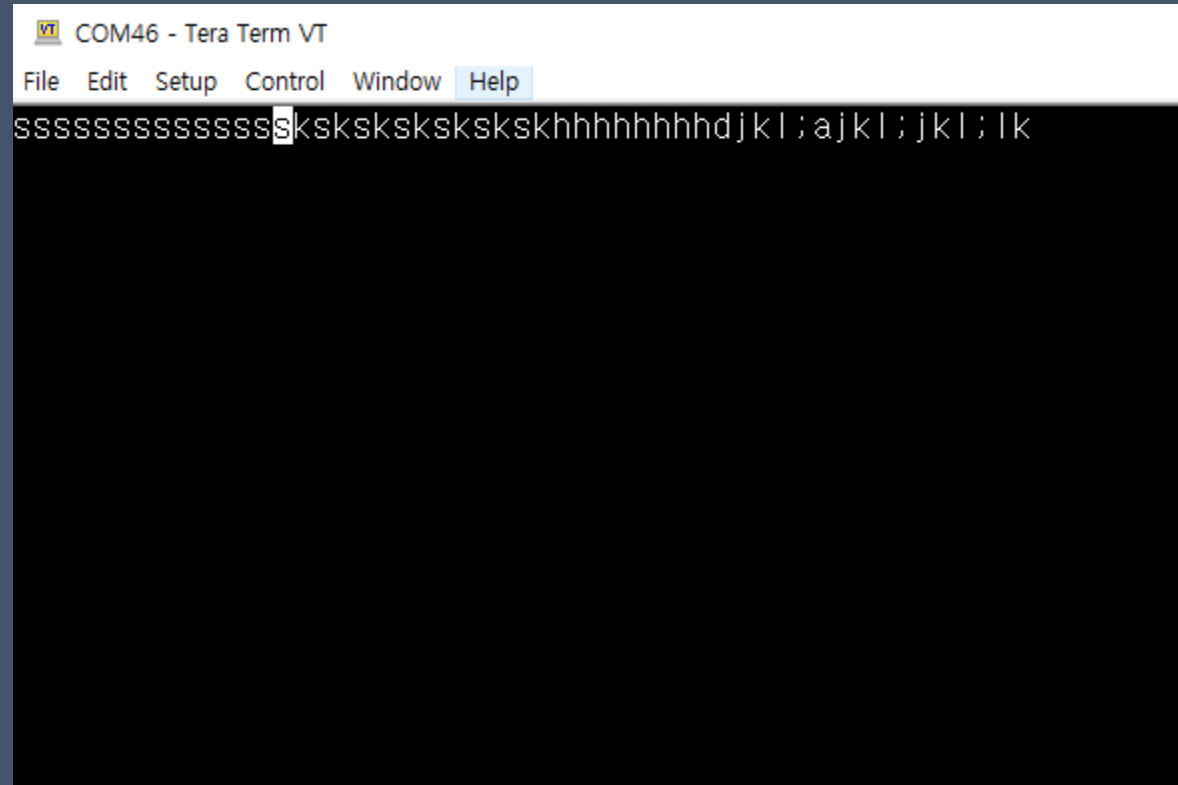
[?] [ ] Enable esp_timer profiling features
[?] [*] Enable lookup of error code strings
[?] (32) System event queue size
[?] (2304) Event loop task stack size
[?] (3584) Main task stack size
[?] (1024) Inter-Processor Call (IPC) task stack size
[?] (3584) High-resolution timer task stack size
[?] UART 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



```
COM46 - Tera Term VT
File Edit Setup Control Window Help
ssssssssssssssskskskskskskskhhhhhhhdjkl;ajkl;jkl;l
ssssssssssssssskskskskskskskhhhhhhhdjkl;ajkl;jkl;l
```

✓ 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)

```
/* 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,
};
```

```
128 /**
129  * @brief UART configuration parameters for uart_param_config function
130  */
131 typedef struct {
132     int baud_rate; /*!< UART baud rate*/
133     uart_word_length_t data_bits; /*!< UART byte size*/
134     uart_parity_t parity; /*!< UART parity mode*/
135     uart_stop_bits_t stop_bits; /*!< UART stop bits*/
136     uart_hw_flowcontrol_t flow_ctrl; /*!< UART HW flow control mode (cts/rt
137     uint8_t rx_flow_ctrl_thresh; /*!< UART HW RTS threshold*/
138     union {
139         uart_sclk_t source_clk; /*!< UART source clock selection*/
140         bool use_ref_tick __attribute__((deprecated)); /*!< Deprecated method
141     };
142 } uart_config_t;
143
144 #ifdef __cplusplus
145 #include <sys/select.h>
146 #endif
"~/work/ESP-GIT/esp-idf/components/soc/include/hal/uart_types.h" 146L, 5724C
```

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

```
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

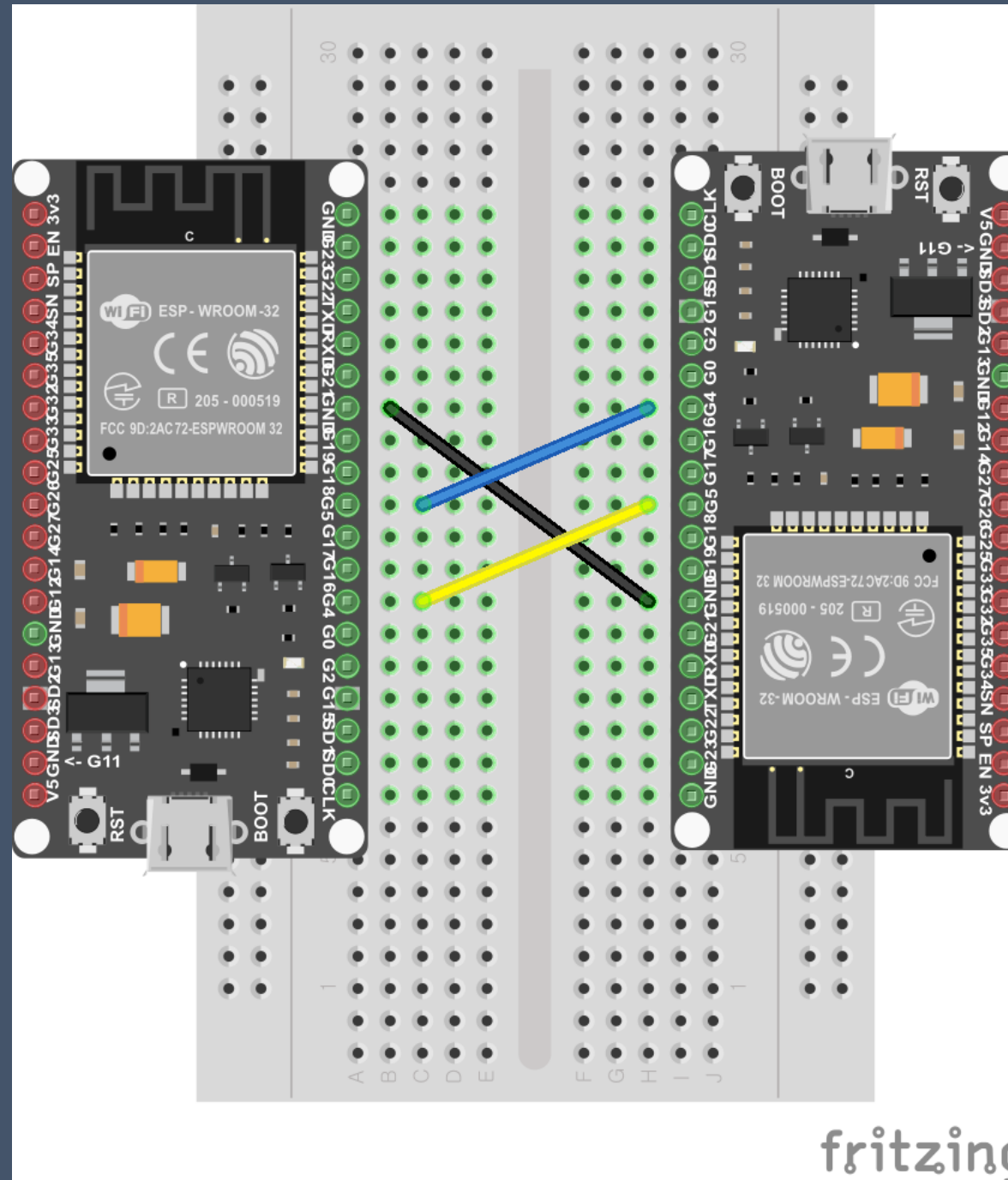
Parameters

- `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);
```

```
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);
```

🏠 » [API Reference](#) » [System API](#) » [Logging library](#)

Logging library

어떤 의미가 있을까요?

Logging library

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)

```
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.

ESP32 UART uart fifo interrupt

ESP32 SDK에서 제공하는 예제의 경우, 추상화 레벨이 높아서 실제 펌웨어 엔지니어들이 즐겨쓰는 코드 형태로 제작하였습니다.

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 }
```

```
#include <sys/select.h>
```

```
"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 );
    }
}
```

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

```
97  * Initialize selected timer of the timer group 0
98  *
99  * timer_idx - the timer number to initialize
100 * auto_reload - should the timer auto reload on alarm?
101 * timer_interval_sec - the interval of alarm to set
102 */
103 static void example_tg0_timer_init(int timer_idx,
104                                     bool auto_reload, double timer_interval_sec)
105 {
106     /* Select and initialize basic parameters of the timer */
107     timer_config_t config = {
108         .divider = TIMER_DIVIDER,
109         .counter_dir = TIMER_COUNT_UP,
110         .counter_en = TIMER_PAUSE,
111         .alarm_en = TIMER_ALARM_EN,
112         .auto_reload = auto_reload,
113     }; // default clock source is APB
114     timer_init(TIMER_GROUP_0, timer_idx, &config);
115
116     /* Timer's counter will initially start from value below.
117        Also, if auto_reload is set, this value will be automatically reload on alarm */
118     timer_set_counter_value(TIMER_GROUP_0, timer_idx, 0x00000000ULL);
119
120     /* Configure the alarm value and the interrupt on alarm. */
121     timer_set_alarm_value(TIMER_GROUP_0, timer_idx, timer_interval_sec * TIMER_SCALE);
122     timer_enable_intr(TIMER_GROUP_0, timer_idx);
123     timer_isr_register(TIMER_GROUP_0, timer_idx, timer_group0_isr,
124                       (void *) timer_idx, ESP_INTR_FLAG_IRAM, NULL);
125
126     timer_start(TIMER_GROUP_0, timer_idx);
127 }
```

✓ 타이머 알람, 인터럽트

```
48 /*
49 * Timer group0 ISR handler
50 * _MONITOR_1 - Thi... Arduino_Core_STM32...
51 * Note:
52 * We don't call the timer API here because they are not declared with IRAM_ATTR.
53 * If we're okay with the timer irq not being serviced while SPI flash cache is disabled,
54 * we can allocate this interrupt without the ESP_INTR_FLAG_IRAM flag and use the normal API.
55 */
56 void IRAM_ATTR timer_group0_isr(void *para)
57 {
58     timer_spinlock_take(TIMER_GROUP_0);
59     int timer_idx = (int) para;
60     struct timeval {
61         /* Retrieve the interrupt status and the counter value
62          * from the timer that reported the interrupt */
63         uint32_t timer_intr = timer_group_get_intr_status_in_isr(TIMER_GROUP_0);
64         uint64_t timer_counter_value = timer_group_get_counter_value_in_isr(TIMER_GROUP_0, timer_idx);
65
66         /* Prepare basic event data
67          * that will be then sent back to the main program task */
68         timer_event_t evt;
69         evt.timer_group = 0; out 이 NULL 일 때 : 무한정 기다린다. fd 중 하나가 준비되거나 신호가 잡힐 때까지 차단된
70         evt.timer_idx = timer_idx;
71         evt.timer_counter_value = timer_counter_value;
72         timeout 이 0 일 때 : 전혀 기다리지 않는다. 차단 없이 fd 상태만 확인할 경우 쓰인다.
73         /* Clear the interrupt 이 0 이 아닐 때 : 지정된 sec 나 usec 만큼 기다린다. fd 중 하나가 준비되거나 시간이
74          * and update the alarm time for the timer with without reload */
75         if (timer_intr & TIMER_INTR_T0) {
76             evt.type = TEST_WITHOUT_RELOAD;
77             timer_group_clr_intr_status_in_isr(TIMER_GROUP_0, TIMER_0);
78             timer_counter_value += (uint64_t) (TIMER_INTERVAL0_SEC * TIMER_SCALE);
79             timer_group_set_alarm_value_in_isr(TIMER_GROUP_0, timer_idx, timer_counter_value);
80         } else if (timer_intr & TIMER_INTR_T1) {
81             evt.type = TEST_WITH_RELOAD;
82             timer_group_clr_intr_status_in_isr(TIMER_GROUP_0, TIMER_1);
83         } else {
84             evt.type = -1; // not supported even type
85         }
86         reafds 집합 : 자료 읽기가 준비되었는지 확인할 fd 들
87         wrirefds 집합 : 쓰기 연산을 할 수 있는지 확인할 fd 들
88         exceptfds 집합 : 예외가 발생했는지 확인할 fd 들
89
90         /* After the alarm has been triggered
91          * we need enable it again, so it is triggered the next time */
92         timer_group_enable_alarm_in_isr(TIMER_GROUP_0, timer_idx);
93
94         /* Now just send the event data back to the main program task */
95         xQueueSendFromISR(timer_queue, &evt, NULL);
96         timer_spinlock_give(TIMER_GROUP_0);
97     }
```


✓ 퀴즈 : 1초 마다 알람을 주는 나만의 타이머 만들어 보기

감사합니다.