TX

```
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_system.h"
#include "esp_log.h"
#include "driver/uart.h"
#include "string.h"
#include "driver/gpio.h"

static const int RX_BUF_SIZE = 1024;
int count = 0;

#define TXD_PIN (GPIO_NUM_10)
#define RXD_PIN (GPIO_NUM_9)

void init(void) {
    const 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,
    };
    // We won't use a buffer for sending data.
    uart_driver_install(UART_NUM_1, RX_BUF_SIZE * 2, 0, 0, NULL, 0);
    uart_param_config(UART_NUM_1, &uart_config);
    uart_set_pin(UART_NUM_1, TXD_PIN, RXD_PIN, UART_PIN_NO_CHANGE, UART_PIN_NO_CHANGE);
}

int sendData(const char* logName, const char* data) {
    const int len = strlen(data);
    const int txBytes = uart_write_bytes(UART_NUM_1, data, len);
    return txBytes;
}

static void tx_task(void *arg) {
    char str[80];
    static const char *TX_TASK_TAG = "TX_TASK";

    while (1) {
        sprintf(str, "count: %d\n", count);
        printf("%s", str);
        sendData(TX_TASK_TAG, str);
        vTaskDelay(2000 / portTICK_PERIOD_MS);   // Read slow
        count++;
        if (count > 1000) count = 0;
    }
}

void app_main(void) {
    init();
    xTaskCreate(tx_task, "uart_tx_task", 1024*2, NULL, configMAX_PRIORITIES-1, NULL);
}
```

RX

```
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_system.h"
#include "esp_log.h"
#include "driver/uart.h"
#include "string.h"
#include "driver/gpio.h"

static const int RX_BUF_SIZE = 1024;

#define TXD_PIN (GPIO_NUM_17)
#define RXD_PIN (GPIO_NUM_16)

void init(void) {
    const 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_DEFAULT,
    };
    // We won't use a buffer for sending data.
    uart_driver_install(UART_NUM_2, RX_BUF_SIZE * 2, 0, 0, NULL, 0);
    uart_param_config(UART_NUM_2, &uart_config);
    uart_set_pin(UART_NUM_2, TXD_PIN, RXD_PIN, UART_PIN_NO_CHANGE, UART_PIN_NO_CHANGE);
}

static void rx_task(void *arg) {
    uint8_t* data = (uint8_t*) malloc(RX_BUF_SIZE+1);
    while (1) {
        const int rxBytes = uart_read_bytes(UART_NUM_2, data, RX_BUF_SIZE, 20 / portTICK_PERIOD_MS);   // Read fast
        if (rxBytes > 0) {
            data[rxBytes] = 0;
        }
        printf(">>>> %s: \n", data);
    }
    free(data);
}

void app_main(void) {
    init();
    xTaskCreate(rx_task, "uart_rx_task", 1024*2, NULL, configMAX_PRIORITIES, NULL);
}
```