English | 中文

To RTOS

To RTOS is a lightweight real-time kernel designed for Cortex-M. The objective is to demonstrate priority scheduling, time slices, sleep timers, and the most basic IPC framework using as little code and resources as possible.

1. Features (Current)

- Fixed priorities + Bitmap fast lookup
- Time slice rotation (round-robin) with the same priority
- Thread lifecycle: initialization / startup / sleep / yield / deletion / restart / exit
- Software timers per thread (for sleep and timeouts)
- Ordered timer linked list
- Lightweight formatted output `t_printf`
- IPC support: semaphores, mutexes (normal and recursive), message queues
- Platform-specific code is independent (assembly context switching + stack initialization)
- Extremely low resource consumption: Under the -O3 optimization, when comparing with the map file, V1.00 only adds approximately 1.00 KB of FLASH and 0.17 KB of RAM compared to the basic system.

2. Directory Layout

```
include/        Public kernel headers (e.g. ToRTOS.h, tdef.h)
libcpu/CM4F/     Cortex-M4 port (context switch asm, stack init, ffs/fls)
src/           Core modules (scheduler, thread, timer, list, service, board,
ipc)
readme/         Documentation (*.md)
bsp/            Board support packet
```

Key headers:

- include/ToRTOS.h: Public API
- include/tdef.h: Types, structs, macros
- Board config: `bsp/.../Core/Inc/ToRTOS_Config.h` (see readme/ToRTOS_CONFIG.md)

3. Configuration Overview

Edit `ToRTOS_Config.h` then full rebuild.
Essential macros:

- `TO_THREAD_PRIORITY_MAX`
- `TO_TICK`
- `TO_IDLE_STACK_SIZE`
- `TO_USING_IPC` (+ per IPC: SEMAPHORE / MUTEX / RECURSIVE_MUTEX / QUEUE)
- `TO_DEBUG`

  See details: readme/ToRTOS_CONFIG.md

4. Quick Start (Pseudo Flow)

```
#include "ToRTOS.h"

#define STACK_SZ 512
static t_thread_t t1;
static t_uint8_t t1_stack[STACK_SZ];

static void thread1_entry(void *arg)
{
    while (1)
    {
        t_printf("t1 running, tick=%d\n", (int)t_tick_get());
        t_mdelay(1000);
    }
}

int main(void)
{
    /* Hardware init (clock, UART putc override, SysTick -> t_tick_increase) */

    t_tortos_init();                /* core init: scheduler, timers, idle,
banner */

    t_thread_create_static(thread1_entry, t1_stack, STACK_SZ, 5, NULL, 10, &t1);
    t_thread_startup(&t1);

    t_sched_start();               /* never returns */
    while (1);
}
```

SysTick handler must call:

```
void SysTick_Handler(void)
{
    t_tick_increase();
}
```

Override character output (example):

```
void t_putc(char c)
{
    /* UART transmit or ITM_SendChar(c); */
}
```

5. Core APIs (Snapshot)

From include/ToRTOS.h:

- Thread: `t_thread_create_static`, `t_thread_create` (dynamic), `t_thread_startup`, `t_thread_sleep`, `t_thread_exit`, `t_thread_delete`, `t_thread_restart`
- Scheduler control: `t_sched_start`
- Timing: `t_mdelay`, `t_tick_get`
- IPC: Semaphore (`T_SEMA_*`), Mutex (`T_MUTEX_*`), Recursive Mutex (`T_MUTEX_RECURSIVE_*`), Queue (`T_QUEUE_*`)
- Debug / log: `t_printf`, `T_DEBUG_LOG`

Return codes: `t_status_t` (see include/tdef.h)

6. Timing Model

- Global tick: incremented in SysTick via `t_tick_increase()`
- Time slice reload per thread: `init_tick`
- Sleep: per-thread timer inserted in ordered list; expiration callback readies thread
- Signed time comparisons handle wraparound

7. Porting (Summary)

More related to transplantation (see extended guide readme/ToRTOS_TRANS.md):

- Context switch assembly: `t_first_switch_task`, `t_normal_switch_task`, PendSV handler
- Stack frame layout in `t_stack_init`
- IRQ mask: `t_irq_disable / t_irq_enable`
- Tick source calling `t_tick_increase`
- Optional `__t_ffs` / `__t_fls` (can fall back to builtin or loop)

8. Coding Style Principles

- Intrusive circular doubly linked lists for all queues
- Critical sections: IRQ disable only (short)
- Static allocation by default, optional dynamic allocation
- Minimal inline assembly isolation

9. License

```
Copyright (c) 2026 ToRTOS
SPDX-License-Identifier: MIT
```

10. Contributing

1. Fork / branch
2. Keep modules small & isolated
3. Add brief Doxygen comments
4. Submit PR with test description

11. Minimal Troubleshooting

Symptom	Likely Cause	Remedy
No context switch	SysTick missing or `__t_ffs` / `__t_fls` wrong	Check handler + bitmap
Sleep never wakes	`t_tick_increase` not invoked	Verify SysTick_Handler
Output garbled	Concurrent `t_printf`	Accept or wrap with lock
HardFault on start	Stack not 8-byte aligned	Inspect `t_stack_init`

Happy hacking with ToRTOS!