StaRT_Config.h 使用文档

该文件集中配置内核功能、规模参数与调试开关。修改后需全量重新编译。

当前文件内容：

```
#define START_THREAD_PRIORITY_MAX      32
#define START_USING_CPU_FFS            1
#define START_TIMER_SKIP_LIST_LEVEL    1
#define START_TICK                     1000
#define S_PRINTF_BUF_SIZE              128
#define START_IDLE_STACK_SIZE          256
#define START_USING_MUTEX              1
#define START_USING_SEMAPHORE          1
#define START_USING_MESSAGEQUEUE       1
#define START_DEBUG                    1
#define START_USING_IPC                1
```

---

1. 优先级与调度

START_THREAD_PRIORITY_MAX

- 最大可用优先级数量（0 ~ N-1）
- 影响：位图宽度/就绪表大小。增大将增加 RAM 占用（`s_thread_priority_table`）。

- 建议：32，根据任务数量规划。

START_USING_CPU_FFS

- 1：使用内置 __s_ffs（位扫描）优化最高优先级查找
- 0：可退回软件查找（需自行实现简易循环）
- 若架构无 CLZ/汇编支持，可保持 1 并提供 C 函数。

---

2. 定时器与 Tick

START_TIMER_SKIP_LIST_LEVEL

- 目前实现仅使用 Level=1（有序链表）
- 未来可扩展跳表加速插入。

START_TICK

- 每秒 Tick 数（Hz）
- 用于：时间片、sleep、信号量超时
- 取值注意：过大增加中断负载，过小降低时间分辨率（典型 1000）

---

3. 打印

S_PRINTF_BUF_SIZE

- s_printf 内部缓冲区大小
- 格式化字符串长度超过此值将被截断
- 增大意味着消耗更多栈（在 s_printf 调用栈帧中）

---

4. 空闲线程

START_IDLE_STACK_SIZE

- Idle 线程栈大小
- Idle 中仅执行清理与可选低功耗，通常较小即可（128~512）

---

5. IPC 功能开关

START_USING_IPC

- 总控开关：为 0 时所有 IPC 模块（信号量/互斥量/消息队列）编译剔除

START_USING_SEMAPHORE

- 信号量支持（依赖 START_USING_IPC=1）
- 关闭：相关结构与 API 不编译，节省代码空间

START_USING_MUTEX

- 互斥量结构预留；当前逻辑未完成但可用于条件编译模板

START_USING_MESSAGEQUEUE

- 消息队列结构预留；当前 API 未实现

---

6. 调试

START_DEBUG

- 1：启用 `S_DEBUG_LOG` 输出
- 0：调试宏为空，不产生代码
- 输出级别宏（信息/警告/错误）由调用处传入 level

> 目前支持三档，若需进一步按等级过滤，可扩展：`#define START_DEBUG_LEVEL n` 并在宏内判断 `(level)<=START_DEBUG_LEVEL`

---

7. 典型裁剪配置示例

最小精简（仅线程 + 睡眠）

```
#define START_THREAD_PRIORITY_MAX   32
#define START_USING_CPU_FFS         1
#define START_TIMER_SKIP_LIST_LEVEL 1
#define START_TICK                  1000
```

```
#define S_PRINTF_BUF_SIZE           64
#define START_IDLE_STACK_SIZE       128
#define START_USING_IPC             0
#define START_DEBUG                 0
```

完整开发调试

```
#define START_THREAD_PRIORITY_MAX   32
#define START_USING_CPU_FFS         1
#define START_TIMER_SKIP_LIST_LEVEL 1
#define START_TICK                  1000
#define S_PRINTF_BUF_SIZE           256
#define START_IDLE_STACK_SIZE       512
#define START_USING_IPC             1
#define START_USING_SEMAPHORE       1
#define START_USING_MUTEX           1
#define START_USING_MESSAGEQUEUE    1
#define START_DEBUG                 1
```

---

8. 依赖关系

宏	依赖
START_USING_SEMAPHORE	START_USING_IPC
START_USING_MUTEX	START_USING_IPC
START_USING_MESSAGEQUEUE	START_USING_IPC
START_USING_CPU_FFS	提供 __s_ffs 实现
START_TICK	SysTick 配置

9. 修改注意

- 修改 `START_THREAD_PRIORITY_MAX` 后需清理/重建工程（防止旧对象文件里数组尺寸不匹配）。
- 降低 START_TICK 需要调整与毫秒换算：`s_tick_from_ms` = (ms * START_TICK)/1000。
- 禁用 IPC 后，无法再调用 `s_sem_xxx` 等 IPC 函数。

10. 运行时验证建议

配置项	验证
START_THREAD_PRIORITY_MAX	创建多个不同优先级线程并确认调度顺序
START_TICK	使用延时 100ms 测试 tick 频率精度（串口时间戳）
S_PRINTF_BUF_SIZE	输出长字符串确认截断行为
START_DEBUG	确认调试日志有/无输出
START_USING_SEMAPHORE	测试信号量阻塞 & 释放

START_IDLE_STACK_SIZE	压测 idle（增加打印/栈水位检查）

---

11. 扩展建议

可新增配置：

- START_DEBUG_LEVEL （过滤日志等级）
- START_ENABLE_ASSERT （启用断言）
- START_STACK_PATTERN (栈填充用于溢出检测)
- START_TICKLESS_ENABLE (未来 Tickless 支持)

---

12. 常见配置错误

现象	排查
创建线程失败无输出	priority>=MAX 或 tick=0
信号量 API 未定义	START_USING_SEMAPHORE 未开启
调度不工作	START_TICK 未配置 SysTick，或 __s_ffs 实现错误
日志花屏/交叉	多线程同时调用 s_printf，无锁属正常竞争
