



Personal

Open source

Business

Explore

Pricing

Blog

Support

This repository

Search

Sign in

Sign up

littlekernel / lk

Watch

144

Star

690

Fork

172

Code

Issues 13

Pull requests 9

Projects 0

Wiki

Pulse

Graphs

Things You Should Know

Travis Geiselbrecht edited this page on 21 Feb · 5 revisions

Safe Actions From IRQ handlers and Timer Callbacks

(Timer callbacks happen from IRQ context)

A number of these functions take a 'reschedule' parameter, which **must** be false when called from IRQ context. If your interrupt handler takes action which will wake a thread and you want to ensure it wakes **now** (provided priority is high enough) rather than when the next quantum expires, return `INT_RESCHEDULE` instead of `INT_NO_RESCHEDULE`. This will cause the kernel to invoke the scheduler before returning to thread context.

The follow actions are IRQ-safe:

- Signal an event with `event_signal()`.
- Reprogram a timer with `timer_set_oneshot()`, `timer_set_periodic()`, or `timer_cancel()`.
- Timer reprogramming is safe even from within that timer's callback.

▼ Pages 7

[Home](#)[Building Outside The LK Tree](#)[Introduction](#)[Kernel APIs And Primitives](#)[LK Code Style](#)[The Build System](#)[Things You Should Know](#)

Clone this wiki locally

<https://github.com/littlekernel/lk/wiki/Things-You-Should-Know>

- Using spinlocks.
- Wake threads on wait queues with `wake_queue_wake_one()` or `wake_queue_wake_all()` , provided you hold the thread lock while doing so (useful for building new synchronization primitives).

IRQ Handlers on Cortex-M Platforms

Things are slightly different on Cortex-M platforms. IRQ handlers there have no return value. They must call `arm_cm_irq_entry()` before doing any work and `arm_cm_irq_exit(bool resched)` before returning. The `resched` parameter to the exit function causes a reschedule on IRQ return if true, similar to returning `INT_RESCHEDULE` on other platforms.

Timekeeping: `lk_time_t`

This unsigned type is used by kernel functions involving timers, timeouts, etc, and is in units of milliseconds. The maximum value (defined as `INFINITE_TIME`) is used to specify a timeout that will never expire. The value 0 passed as a timeout parameter always means ``return immediately if you would have to wait."

`ERR_TIMED_OUT` is the status returned by a function taking a timeout value, if the timeout expires before the requested action can be accomplished.

Destroying Primitives

There are `_destroy()` functions, but they require careful use — it is not safe to destroy a primitive that might be used again from another thread or interrupt context after its destruction.



