KASL Documentation

Release 1.0.0

K. Isom

Mar 08, 2019

Contents:

CHAPTER 1

Introduction

KASL is a library that builds support for a lot of tools I find myself reaching for repeatedly in various projects. It aims to work among the platforms I commonly use:

- atmelsamd (e.g. the Feather M0 and M4)
- esp8266
- atmega (primary 328p, but nominally includes untested support for 32u4)

CHAPTER 2

Triggers

Including `kasl/trigger.h` makes the `Trigger` class available. This class is constructed with a millisecond delta, and its `ready` method will return true if at least that long has passed since the last call to `ready`. By default, it will be ready immediately; if a second optional true argument is passed to the constructor, it will require waiting `delta` milliseconds before being ready for the first time.

The `ready` method has two forms:

- `bool ready()` calls `ready(millis())`.
- `bool ready(unsigned long now)` allows the same `millis` value to be reused in multiple places to avoid calling the function multiple times. When the allotted time is up, the trigger will reset to the last update time plus the delta.

Finally, there is a `reset` method:

- `void reset()` calls `reset(millis())`.
- `void reset(unsigned long now)` resets the trigger to fire next in `now` + `delta` milliseconds.

CHAPTER 3

Swap functions

The swap functions are defined in `kasl/swap.h`, and provide functions for swapping values in-place. The current swap functions are

- `void swap_u8(uint8_t &a, uint8_t &b)`
- `void swap_ul(unsigned long &a, unsigned long &b)`

CHAPTER 4

Scheduling

This package supports a common interface for running tasks and background threads. This support is included with `kasl/scheduling.h`; three functions are provided:

- `void kasl::scheduleThread` adds a thread to be scheduled. This should be called before `startScheduler`.
- `void kasl::startScheduler` initialises the scheduler.
- `void kasl::runScheduler` runs an interation of the scheduler. This should be placed in the main loop.

4.1 Platform specific notes

- Atmel SAMD boards (`src/scheduling_atmelsamd.cc`):
  - `startScheduler` and `runScheduler` are nops provided to keep a common interface for code that needs to run on multiple architectures.
  - The standard Arduino Scheduler is used.
  - There isn't a hard limit on the number of threads that can be run.
- Other boards (`src/scheduling_default.cc`):
  - There is a limit to the number of tasks that can be scheduled; this is controlled by the `MAX_TASKS` define, with a default of 8. This can be tuned to save memory, e.g. on the AVR boards.
  - Threads are scheduled to run every `SCHEDULER_FREQUENCY` milliseconds; the default is 50.
  - These settings can be changed under `build_flags` in the environment. See `examples/schedutest` for an example.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search