```c
#include <assert.h>
#include <stdbool.h>
#include <stdint.h>
#include <intrinsics.h>
#include <iolpc2378.h>
#include <bsp.h>
#include <timers.h>
#include <scheduler.h>

static schTCB_t schTasks[TT_SCHED_MAX_TASKS];


void schInit(void) {   // initialise the scheduler

  for (uint8_t i = 0; i < TT_SCHED_MAX_TASKS; i+=1) {
    schTasks[i].task = (pVoidFunc_t)0;
    schTasks[i].delay = 0;
    schTasks[i].period = 0;
    schTasks[i].invocations = 0;
  }
  initTimer(TIMER0, schUpdate, TT_SCHED_TICK_HZ);
}

void schStart(void) {           // start ticking
  startTimer(TIMER0);
  __enable_interrupt();
}

void schUpdate(void) {          // update after a tick -- ISR

  for (uint8_t i = 0; i < TT_SCHED_MAX_TASKS; i+=1) {
    if (schTasks[i].task) {
      if (schTasks[i].delay == 0) {
        schTasks[i].invocations += 1;
        if (schTasks[i].period) {
          schTasks[i].delay = schTasks[i].period;
        }
      } else {
        schTasks[i].delay -= 1;
      }
    }
  }
}

void schDispatch(void) {       // run the next task

  for (uint8_t i = 0; i < TT_SCHED_MAX_TASKS; i+=1) {
    if (schTasks[i].invocations > 0) {
      (*(schTasks[i].task))();
      schTasks[i].invocations -= 1;
      if (schTasks[i].period == 0) {
        schRemoveTask(i);
      }
    }
  }
  schSleep();
}

void schAddTask(               // add a task to the task set
  pVoidFunc_t task,              // the task to add
  uint32_t delay,                // the delay in ms
  uint32_t period) {             // the period
```

```c

  uint8_t i = 0;

  while (i < TT_SCHED_MAX_TASKS && schTasks[i].task != (pVoidFunc_t)0) {
    i += 1;
  }
  assert(i < TT_SCHED_MAX_TASKS);
  schTasks[i].task = task;
  schTasks[i].delay = delay;
  schTasks[i].period = period;
  schTasks[i].invocations = 0;
}

void schRemoveTask(            // remove a set from the task set
  uint8_t id) {                  // identifier of the task to remove

  assert((id < TT_SCHED_MAX_TASKS) && (schTasks[id].task != (pVoidFunc_t)0));

  schTasks[id].task = (pVoidFunc_t)0;
  schTasks[id].delay = 0;
  schTasks[id].period = 0;
  schTasks[id].invocations = 0;
}

void schSleep(void) {          // go to sleep to save power
  PCON |= 1;
}
```