Understanding the NanoRK Scheduler

Osman, Hamid (10424075)
Li, Guanpeng (39225099)
Ton, Hason (65889065)
Wong, Cary (19096072)

October 11, 2012

1. **Introduction:** The NanoRK scheduler schedules tasks based on a static predefined configuration of runtimes associated with tasks. The scheduler is a single method that is inlined for compiler efficiency: the method body is copied wherever the method is called. The scheduler uses guards to determine whether certain methods should be called based on the configuration of the calling method. A free queue and a ready queue represents tasks that have been removed from the ready queue and tasks that are ready to run respectively. NanoRK also provides a standard interface where stub methods are overridden by classes in the hardware abstraction layer (hal). In the case of the scheduler, timing functions are hardware specific and overridden. The subsequent sections will cover the structure of the scheduler from top down.

2. **Initialization and Time Adjustments:** Initialization consists of managing the interrupts and aligning the systems timer. If the system support watchdogs, *nrk_watchdog_reset()* is called from the scheduler and implemented by the processor to signal that the system is responsive. The nrk system time is updated by adding the previous timer value to the current system time and adjusting it based on the number of nanoseconds. The scheduler also accounts for over shooting with the modulus operation that follows.

3. **Current Task Management:** The scheduler manages the current task (*nrk_cur_task_TCB*), a pointer to the head of the ready queue. The current task goes through a series of conditionals to determine what state it should transition to. For example, a task may not be finished executing, suspended and waiting for a resource (i.e *EVENT_SUSPENDED*), in which the state is updated to suspended and the task is moved from the ready to free queue. If *NRK_STATS_TRACKER* is defined, calls to multiple functions are made to collect statistics. If a task is currently running, the remaining cpu time must be decremented. If it is in violation or the remaining cpu time has been exhausted, the task is suspended and removed from the ready queue.

4. **General Task Management:** The scheduler must decrement the next wake up time and period of all tasks and determine whether they are ready to be placed in the ready queue. The scheduler iterates through the set of tasks and determines which suspended tasks can potentially preempt the current running task, and at what time. (Figure **??**) Tasks are added back in priority order into the ready queue, external of the scheduler, in task.c. The tasks CPU remaining time, next wake up, period, etc. are re-initialized.

5. **Idle Time:** NRK has a deep sleep mode in which the processor could be powered down to save energy if there are no tasks to run. The scheduler contains logic to go into deep sleep and wake up from deep sleep in time for a context switch.

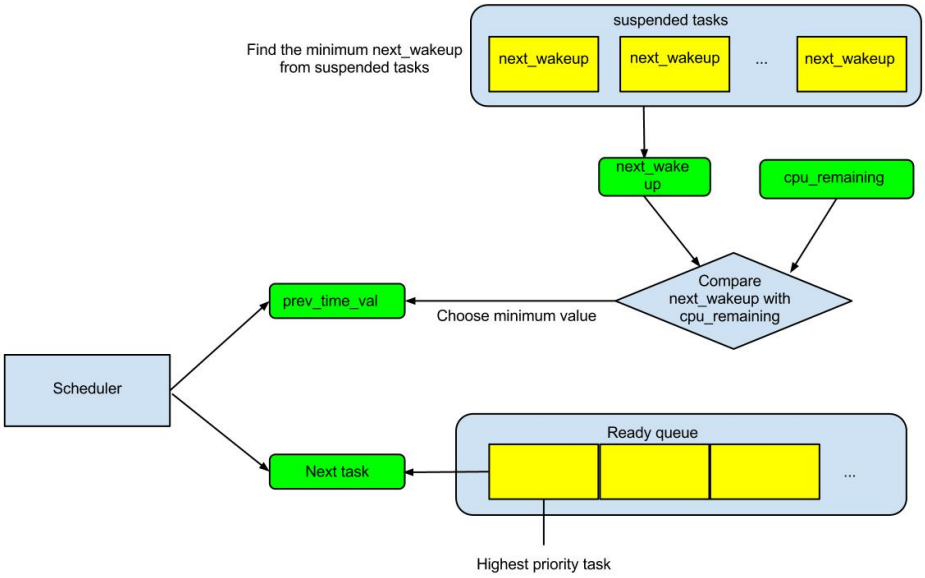

Figure 1: Scheduler flowchart