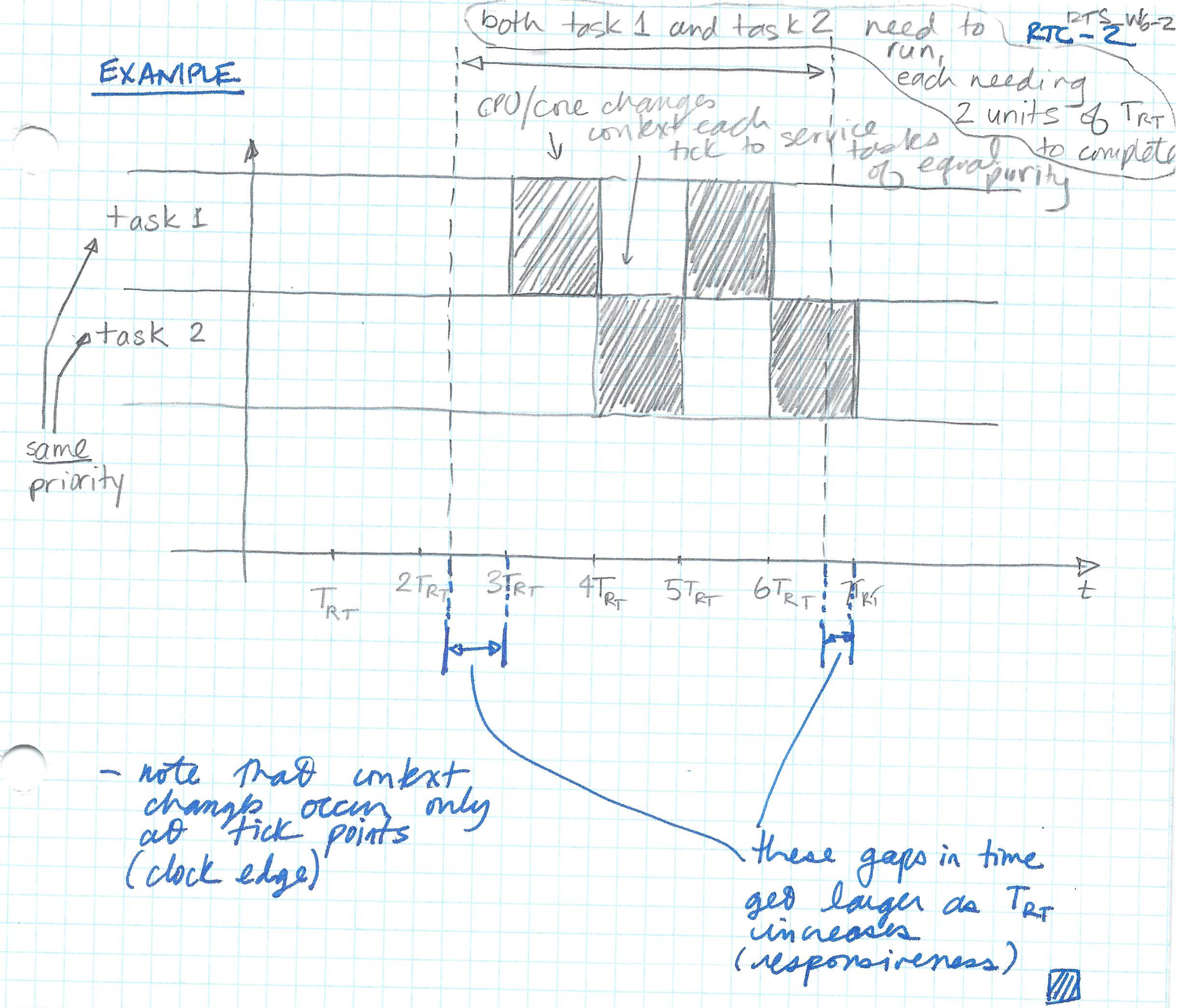
TICK

- most real-time systems are based on clock period known as a "tick" which is usually quite large compared with the CPU clock period...
- the tick is also known as a time slice or quantum
- there is a trade off here — if the tick is too short, then the scheduler may be too busy and the system will consume more power, however, if the tick is too long, the system will be less responsive to events or the need for context switches
- the tick is often around 10ms (a freq. of 100Hz).

PREEMPTIVE MULTITASKING

- all of the OS architectures we deal with in EMBT are multitasking, including, Linux, Windows, or Free RTOS.
- while in "conventional" or non-pre-emptive OS operation, tasks are explicitly programmed to yield to other tasks, in a real-time or pre-emptive OS, tasks themselves are often coded as infinite loops (non-exiting) and it is the scheduler which invokes context switches (using an interrupt mechanism).

EXAMPLE


both task 1 and task 2 need to run, each needing 2 units of T_{RT} to complete
CPU/core changes context each tick to service tasks of equal purity
task 1
task 2
same priority
T_{RT}
$2T_{RT}$
$3T_{RT}$
$4T_{RT}$
$5T_{RT}$
$6T_{RT}$
T_{RT}
t

- note that context changes occur only at tick points (clock edge)

these gaps in time get larger as T_{RT} increases (responsiveness) ▨

ESTIMATING PERFORMANCE (THEORY)

- we can estimate the expected performance of real time systems using an analytical approach

- we can calculate execution times associated with tasks defined as the number of CPU clock cycles need to run a machine language implementation of a task (N.B., your choice of programming language and techniques will have a significant impact), often denoted by T_{ET}.

- we can also estimate latency, since we can determine an approximate number of memory reads/writes from our program, as well as estimate the number of likely preemptions for the task

more importance

TASK	EXECUTION TIME	LATENCY		PREEMPTION DELAY	
		min	max	min	max
S_0	$T_{ET,0}$ 1	0.5 $T_{L,0}^{min}$	2 $T_{L,0}^{max}$	0 $T_{PD,0}^{min}$	1 $T_{PD,0}^{max}$
S_1	$T_{ET,1}$ 2	0.5 $T_{L,1}^{min}$	3 $T_{L,1}^{max}$	2 $T_{PD,1}^{min}$	4 $T_{PD,1}^{max}$
S_2	$T_{ET,2}$ 5	0.5 $T_{L,2}^{min}$	1 $T_{L,2}^{max}$	4 $T_{PD,2}^{min}$	6 $T_{PD,2}^{max}$

example times in milliseconds

Q: How many times can each task occur (considered on its own) in a 100 ms period of time?

SOLN

$$T_0^{min} \leq T_0 \leq T_0^{max}$$

$$\Leftrightarrow \quad T_{ET,0} + T_{L,0}^{min} + T_{PD,0}^{min} \leq T_0 \leq T_{ET,0} + T_{L,0}^{max} + T_{PD,0}^{max}$$

$$\Leftrightarrow \quad 1 + 0.5 + 0 \leq T_0 \leq 1 + 2 + 1$$

$$\boxed{1.5 \leq T_0 \leq 4}$$

$\therefore$ S_0 happens between 25 and 66 times in 100ms

⋮

so for example, if S_0 corresponds to a sampling operation, we can sample at a maximum rate of $\frac{1}{1.5}$ kHz $\doteq$ 0.667 kHz $\cong$ $\boxed{667 \text{ Hz}}$