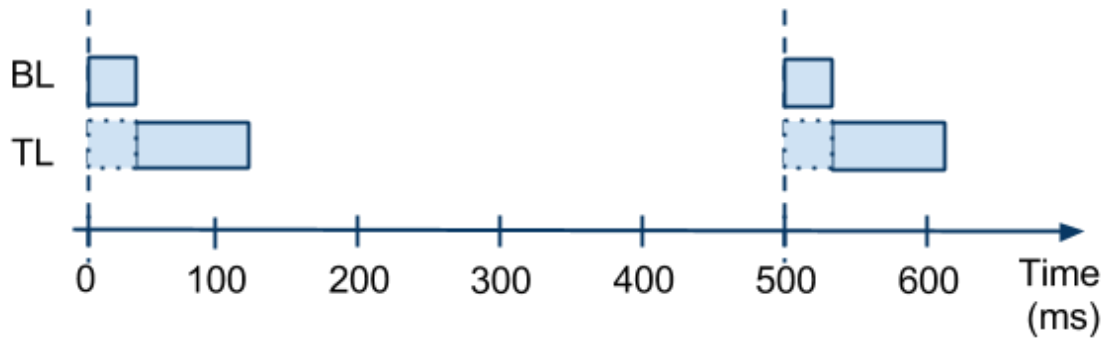
10.3 WCET

10.4 Utilization for multiple tasks

Microcontroller utilization analysis is even more commonly done when multiple tasks are implemented on a single microcontroller. Consider an LedShow system with two tasks, BlinkLed and ThreeLeds, each with a 500 ms period. Suppose microcontroller M executes 100 instr/sec, or 0.010 sec/instr. Suppose BlinkLed's worst-case instructions per tick is 3 instr, meaning a WCET on microcontroller M of 3 instr * 0.010 sec/instr = 30 ms, and ThreeLeds' worst-case instructions per tick is 9 instr, for a WCET of 9 instr * 0.010 sec/instr = 90 ms. The time window of interest is 500 ms, because the two tasks tick once every 500 ms. Thus, the microcontroller utilization will be (30 ms + 90 ms) / 500 ms or 24%.

The timing diagram below is a ***microcontroller usage diagram*** showing how the two tasks execute on microcontroller M using our earlier scheduler and assuming each task tick executes for the task's WCET. In the timing diagram, a dotted block represents a ready task that isn't executing, and a solid block represents a task that is executing. At time 0, both tasks are ready to execute, per the convention that tasks should tick at system startup. The earlier-introduced simple task scheduler may execute BL first, which takes 30 ms, followed by TL, which takes 90 ms. At time 120 ms, the microcontroller becomes idle (it is executing no task), until time 500 ms when the pattern of execution repeats.

Figure 10.4.1: LedShow microcontroller usage diagram.

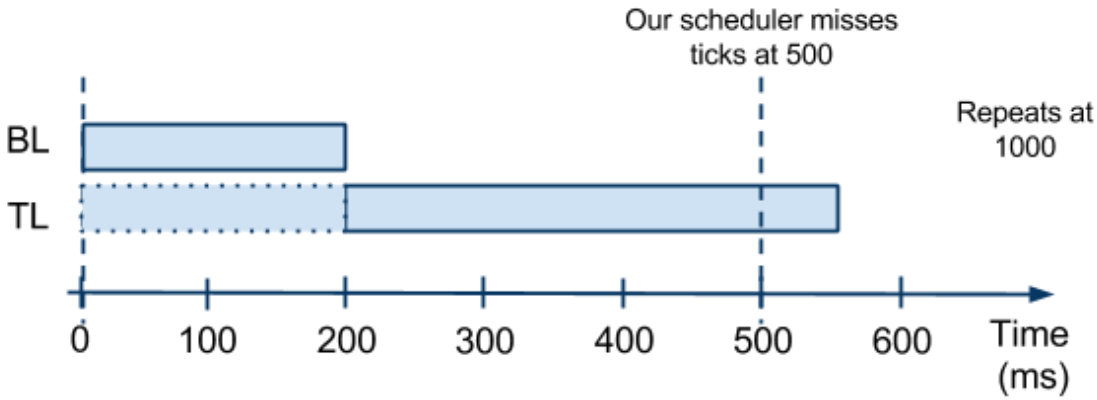


Above, microcontroller usage matches the utilization analysis, namely the microcontroller is in use 24% of the time. However, usage does not *always* match utilization analysis. One obvious such case is when utilization is computed to be greater than 100%, as now discussed.

A computed utilization greater than 100% would mean that the microcontroller cannot execute all the tasks within the given period. For example, suppose that BL's WCET was instead 200 ms, and TL's WCET was 350 ms. Then utilization would be (200 ms + 350 ms) / 500 ms = 550 ms / 500 ms or 110%. The timing diagram below illustrates microcontroller usage for those tasks. BL executes for 200 ms, then TL for 350 ms. The microcontroller

timer ticks every 500 ms. At time 500 ms, a timer tick occurs, but the scheduler code misses the tick, so neither BL nor TL will tick at 500 ms. At 550 ms, the scheduler code clears timerFlag and waits for a timer tick. For our scheduler, the next tick that will be detected will be at 1000 ms. Although utilization was computed to be 110%, actual microcontroller usage using our scheduler is only slightly over 50% (and task ticks are missed).

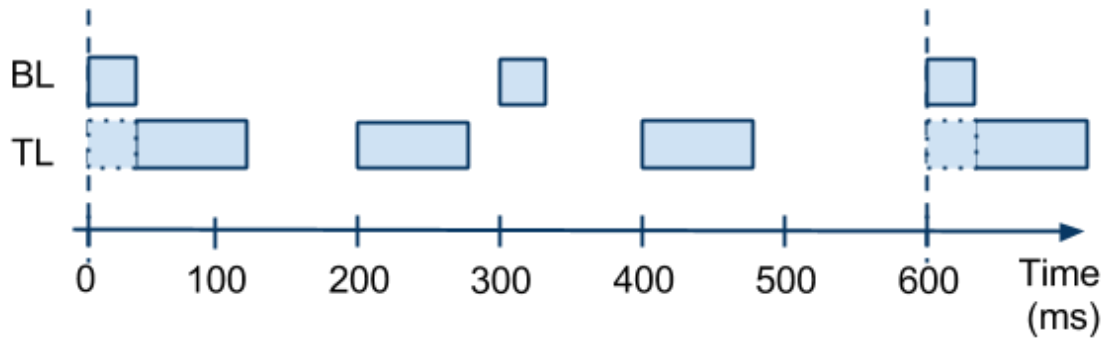
Figure 10.4.2: LedShow microcontroller usage for utilization exceeding 100 percent.



Possible remedies to the above clearly undesirable situation are the same remedies as for a single synchSM, such as increasing the period (of one or more synchSMs), or decreasing the execution time per tick via more efficient action code, splitting states in two, reducing functionality, etc. For multiple tasks, the remedy of "reducing functionality" may include eliminating an entire task from the system.

When two or more tasks have different periods, the time window of interest for computing utilization or determining actual microcontroller usage is the ***hyperperiod*** of the tasks, which is the least-common-multiple (LCM) of the tasks' periods. For example, suppose BL's period is 300 ms and TL's period is 200 ms. LCM(300 ms, 200 ms) is 600 ms. Thus, every 600 ms, the pattern of execution should repeat, as shown in the timing diagram below, with BL's WCET being 30 ms, and TL's being 90 ms.

Figure 10.4.3: LedShow microcontroller usage diagram for hyperperiod.



During the hyperperiod of 600 ms, BL will execute 600 ms / 300 ms or 2 times, while TL will execute 600 ms / 200 ms or 3 times, as shown above. The utilization during the hyperperiod is (2*30ms + 3*90ms) / 600 ms is 330 ms / 600 ms or 55%.

Summarizing, utilization for tasks T1...Tn executing on a microcontroller M is determined as follows:

- Determine M's sec/instr rate -- call this R
- Analyze each task Ti to determine its worst case number of instructions per tick, then multiply that number by R to determine Ti.WCET
- Determine the hyperperiod H as LCM(T1.period, T2.period, ..., Tn.period)
- Utilization = ((H/T1.period)*T1.WCET + (H/T2.period)*T2.WCET + ... + (H/Tn.period)*Tn.WCET) / H. Note that H/Ti.period is simply the number of times that Ti executes during hyperperiod H.

Participation Activity 10.4.1: Utilization and microcontroller usage for multiple tasks.

Given a microcontroller that executes Task1 with WCET 50 ms and period 500 ms, and Task2 with WCET 100 ms and period 500 ms. Assume the scheduler schedules Task1 first. Draw the microcontroller usage diagram before answering the following.

#	Question	Your answer
1	Task1 executes starting at time 0 ms. At what time does that task end? ✔ 50 Task1's WCET is 50 ms. So if the task starts at 0 ms, we assume the task ends at 50	50 ms

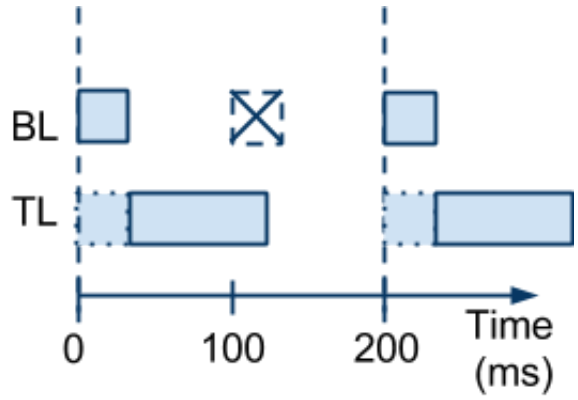
	ms.	
2	At what time does Task2 first start executing? ✔ 50 Task2 was ready to execute at time 0 ms, but had to wait for Task1 to execute. Task1 required 50 ms to execute, so Task2 begins at 50 ms.	50 ms
3	When does Task2 first end? ✔ 150 Task2 started at 50 ms. WCET is 100, so Task2 finishes at 50+100 or 150 ms.	150 ms
4	After Task1 and Task2 first execute and end, at what time does a next task execute? ✔ 500 After Task1 executes from 0-50 ms and Task2 from 50-150 ms, the microcontroller is idle until the period of 500 ms elapses. At 500 ms, the timer ticks, and the scheduler code executes Task1 again, then executes Task2 at 550 ms.	500 ms
5	Assume Task1's WCET was 600 ms and everything else was the same. Using our earlier-defined scheduler, what task will begin executing at time 600 ms? Valid answers are Task1, Task2, or none. Draw the new microcontroller usage diagram, and look at the scheduler code (especially its for loop) before answering. ✔ Task2 At 0 ms, both tasks are ready, and the scheduler starts executing Task1. The timer tick at 500 ms is ignored because Task1 is still	Task2

executing. At 600 ms, the scheduler's for loop moves on to execute Task2.

The relationship of utilization analysis and timer overruns for our scheduler code is as follows.

- Utilization > 100%: Timer overrun *will* occur (assuming WCET)
- Utilization < 100%
  - Single task: Timer overrun shouldn't occur
  - Multiple tasks: Timer overrun *may* occur. For example, suppose task BL's period and WCET are 100 ms and 30 ms respectively, and TL's are 200 ms and 90 ms. Computed utilization is (30+90)/200 or 60%. However, the timing diagram below shows that the tick at 100 ms is missed because of TL's execution, even though utilization is well below 100%.

Figure 10.4.4: LedShow microcontroller usage diagram showing timer overrun.



Such timer overrun can be detected through further analysis, namely by noting that at every hyperperiod (starting with 0), all tasks will be ready and thus all will execute one after the other, representing the worst-case execution situation within a hyperperiod. Thus:

Checking if *the sum of WCETs* for all tasks *exceeds the smallest period of any task* will indicate whether a timer overrun will occur in that worst-case situation.

The key here is that utilization is a computed value that does *not* take into account the details of the scheduler. While utilization is useful initially, a more thorough analysis with

scheduler awareness reveals a more accurate picture.

Participation Activity 10.4.2: Utilization and timer overrun.

#	Question	Your answer
1	For a microcontroller running one task, utilization < 100% indicates timer overrun should not occur. ✔ Such utilization means each task's tick takes less than the task's period. The timer ticks, the scheduler executes the task, and the task ends, and then later the next timer tick occurs.	**True** False
2	For a microcontroller running two tasks, utilization < 100% indicates timer overrun should not occur. ✔ A fast-ticking task may yield timer overrun if the other task has a long WCET, causing the scheduler to be busy executing that task when a timer tick occurs.	True **False**
3	Task1's period is 50 ms, and Task2's is 500 ms. Task1's WCET is 10 ms. Task2's is 100 ms. Timer overrun will occur, even though computed utilization is only 110/500 or 22%. ✔ At 0 ms, Task1 executes for 10 ms, then Task2 for 110 ms. The tick at 50 ms will thus cause timer overrun.	**True** False

10.5 Jitter