10.4 Utilization for multiple tasks

10.5 Jitter

A task's ***jitter*** is the delay between the time that a task was *ready* to start executing and the time that it *actually* starts executing on the microcontroller. Jitter can be determined by drawing a timing diagram that shows ready and executing tasks for the tasks' hyperperiod, as earlier. Jitter is caused by two or more tasks being ready; the scheduler executes one of those tasks, causing jitter in the other tasks. Suppose Task1 has period 100 ms and WCET of 20 ms, Task2 has period of 100 ms and WCET of 10 ms, and the scheduler schedules Task1 first. Then each time Task2 is ready to execute, Task2 experience 20 ms of jitter (waiting for Task1 to end). Task1 has no jitter.

If Task2's period was 50 ms, then every other time Task2 is ready, Task2 will execute immediately without waiting for Task1 (which only executes every 100 ms). Task2's worst-case jitter is 20 ms (at 0 ms, 100 ms, 200 ms, etc.), and average jitter is (20 + 0) / 2 or 10 ms.

An example where jitter matters is a system where Task1 blinks an LED repeatedly, and Task2 computes something. Jitter for Task1 might cause the LED to blink in an inconsistent manner, whereas jitter for Task2 might be more acceptable. Another example is a system where Task1 monitors an input for a change, and Task2 computes something. Jitter for Task1 might cause an input event be missed, or introduce extra latency.

Avoiding timer overruns is thus not the only goal when scheduling tasks to execute on a microcontroller. Other goals, such as reducing jitter, are also important. Attention should thus be paid to creating good scheduling approaches.

Participation Activity 10.5.1: 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	What is Task1's worst-case jitter? ✔ 0 Because Task1 never waits for Task2, Task1 has no jitter.	0 ms
2	What is Task2's worst-case jitter? ✔ 50 Task1 executes first every 500 ms, taking 50 ms, so Task2 is always delayed by 50 ms.	50 ms

10.6 Scheduling