The offset mechanism can be used to make the time gaps more frequent (see Figure 84).

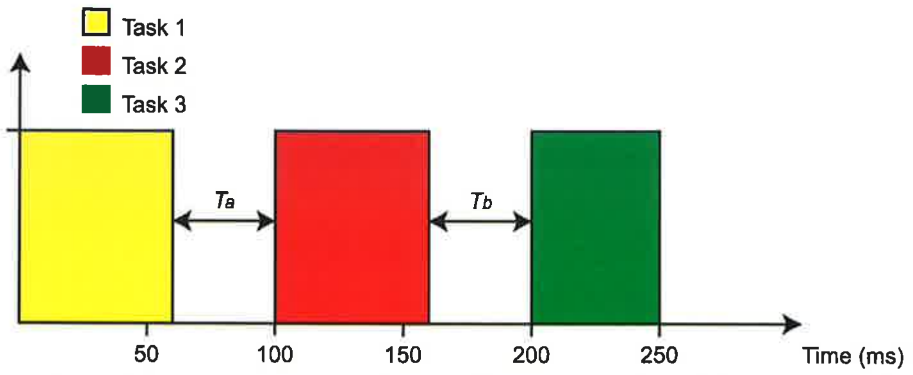

Figure 84. The result of assigning offset to tasks 2 and 3, is that the time available for the execution of other functions occurs more often (T_a).

The same processor handles communication and IEC 61131-3 code. This means that you have to consider how much code you include in each task, when you tune the tasks.

Assume that we have a task running code with an execution time of 500 ms and an interval time of 1000 ms. This means a cyclic load of 50% (load = execution time / interval time). But, this also means that no communication can be performed during the 500 ms execution (since communication has lower priority than the task).

Now, assume that we have divided the code into 4 tasks such that each one corresponds to 125 ms of the execution time. The interval time is still 1000 ms, hence the load is still 50%. But, if we set the offset for the 4 tasks to 0, 250, 500, and 750 ms, the result will be completely different. Now, code will be executed for 125 ms, after which there will be a pause when communication can be performed. Following this, code will be executed for another 125 ms followed by another pause when further communication can be performed. Hence, we still have the same cyclic load, but the possibility for communication has increased considerably.

To conclude, try to tune your tasks using offsets before you change the priority. Actually, the only time you have to change the priority, is when two tasks have so much code that their execution cannot be "contained" within the same time slot, that is, the total execution time exceeds the length of the time slot. It is then necessary to specify which of the two tasks is most important to the system.