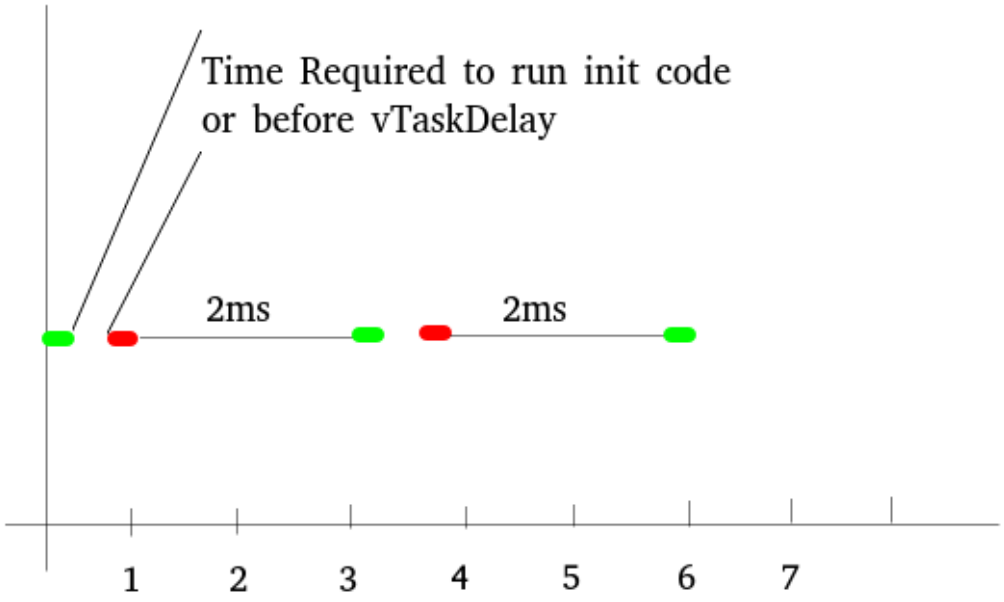
Difference between vTaskDelay & vTaskDelayUntil

In case of vTaskDelay:

Consider a Task with the following functionality.

```
void Task(void *)
{
	//init code//
	vTaskDelay(pdMS_TO_TICKS(2));
	for(;;) {
		//func//
	}
}
```

vTaskDelay is used to block a task, it takes the delay value and provides the delay after the running of init code or the code before the taskdelay call. For example, the trace diagram will be seen as shown below.



This means that, while using vTaskDelay the delay is applied after the code before the delay was called, this results in the creation of unperiodic task. Which means that, the task which was supposed to run every 2ms, runs for more than 2ms, causing aperiodicity, Thus, in critical task applications, it will be difficult for the periodic execution of a specific task functionality.

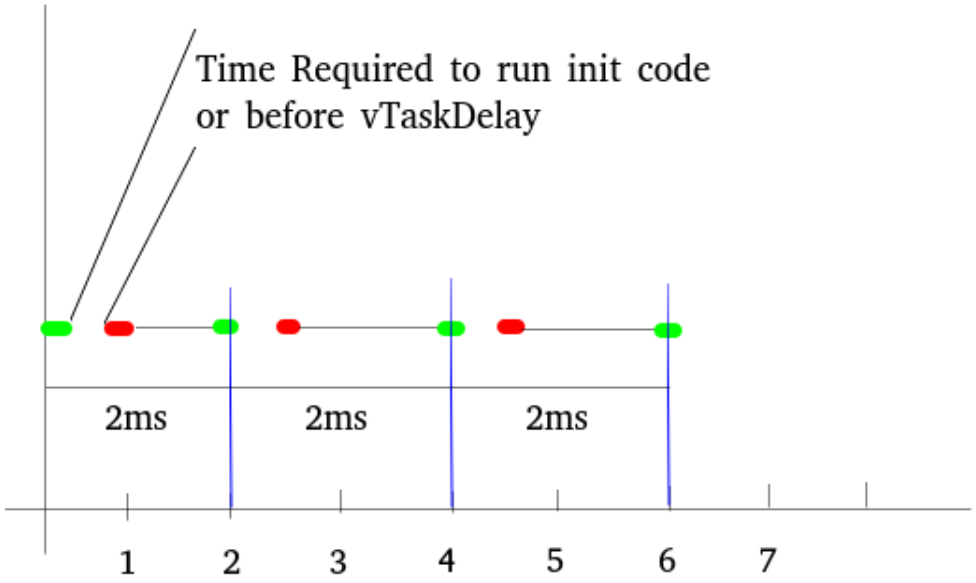
In case of vTaskDelayUntil:

Consider a Task with the following functionality.

```
void Task(void *)
{
	//init code//
	woken = xTaskGetTickCount();
	for(;;) {
		//func//
		vTaskDelayUntil(&woken, 2);
	}
}
```

vTaskDelayUntil is used to create periodic tasks, it takes two parameters, firstly the time at which the task last left the Blocked state, secondly the delay which will provide the periodicity to the task.

What the vTaskDelayUntil does that, it stores the last woken value, and whatever time, it takes to execute the code before it, it will execute the task only for the delay that it was given. The trace for a vTaskDelay until will look like as shown below.



This diagram shows the execution of a periodic task, which occurs every 2ms. This is possible because of vTaskDelayUntil. It just takes the last woken tick and according to that it makes the changes in the execution so that the task is finished in the specified delay interval.