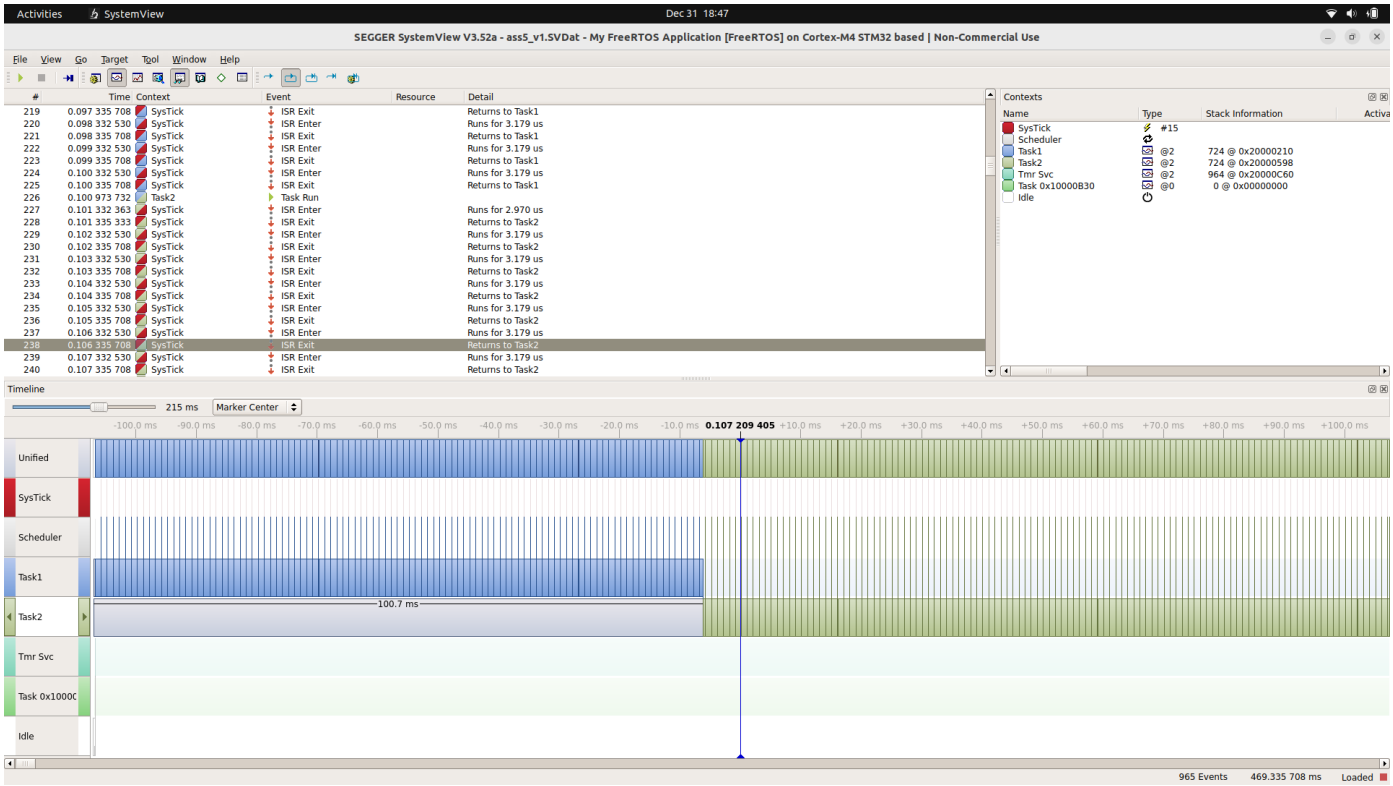
Assignment 5

Scenario 1




Task 1 is created with priority 2 and task 2 was created with the same priority, and taskYIELD() is called. Code shown below:

```
void Task1(void *temp)
{
	for(;;)
	{
		HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_12);
		HAL_Delay(100);
		taskYIELD();
	}
}

void Task2(void *temp)
{
	for(;;)
	{
		HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_13);
		HAL_Delay(500);
		taskYIELD();
	}
}
```

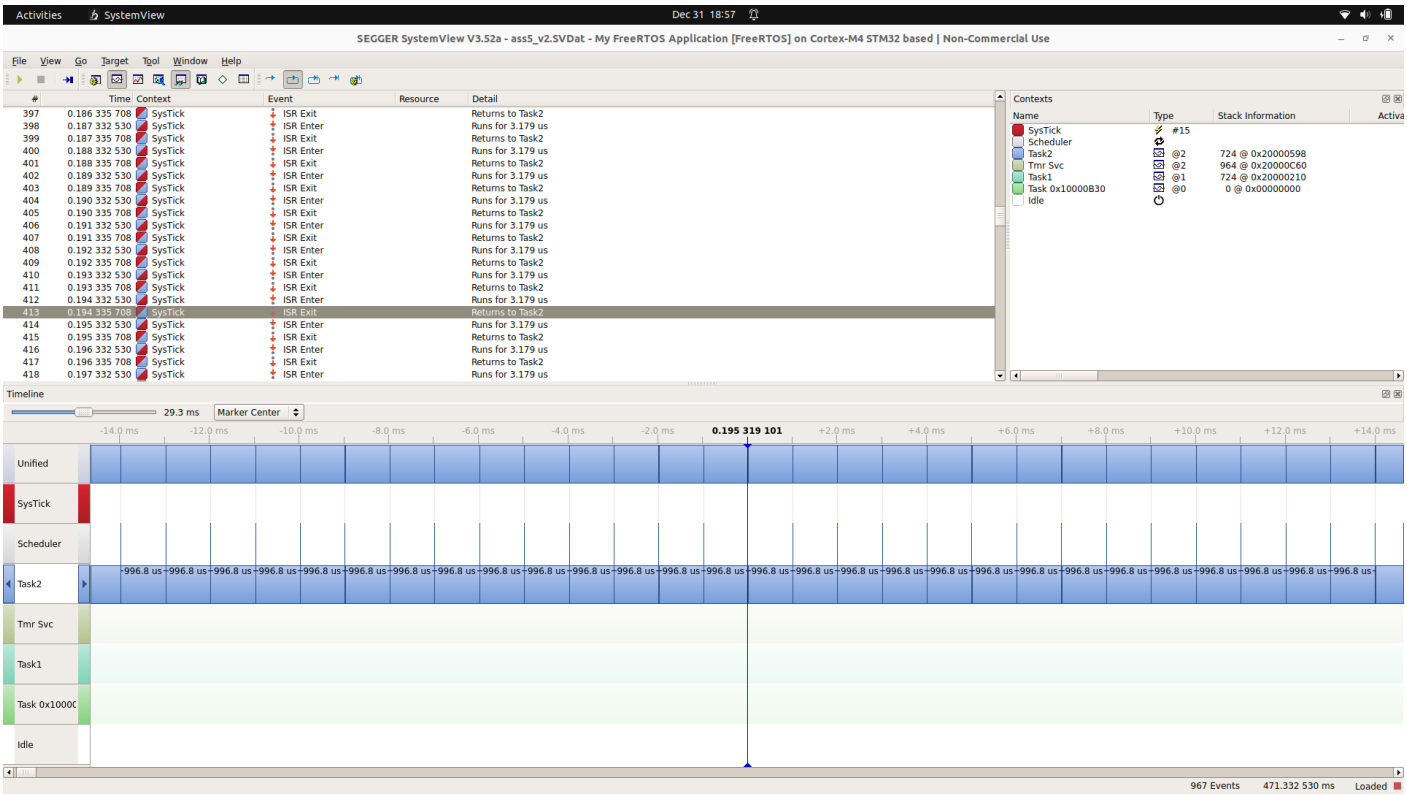
The taskYIELD() is a macro prototype, which functions as the following:

When a task calls taskYIELD(), the scheduler will select another Ready state task of equal priority to enter the Running state in its place. If there are no other Ready state tasks of equal priority then the task that called taskYIELD() will itself be transitioned straight back into the Running state.
The scheduler will only ever select a task of equal priority to the task that called taskYIELD() because, if there were any tasks of higher priority that were in the Ready state, the task that called taskYIELD() would not have been executing in the first place.

Thus, we can see in the trace that, after the execution of task 1 when the taskYIELD was called the scheduler, gives the CPU to the other task with same priority i.e. task2.

Scenario 2

In scenario 2, we just change the priority of task1 from 2 to 1.




In the above scenario, we can see that due to change in priority, task 2 gets the higher priority. Thus, we can see that only task 2 keeps on executing continously, even after taskYIELD being called, there is no Task with priority higher than task 2's priority i.e. 2 Thus, the same task will be keep on executing.