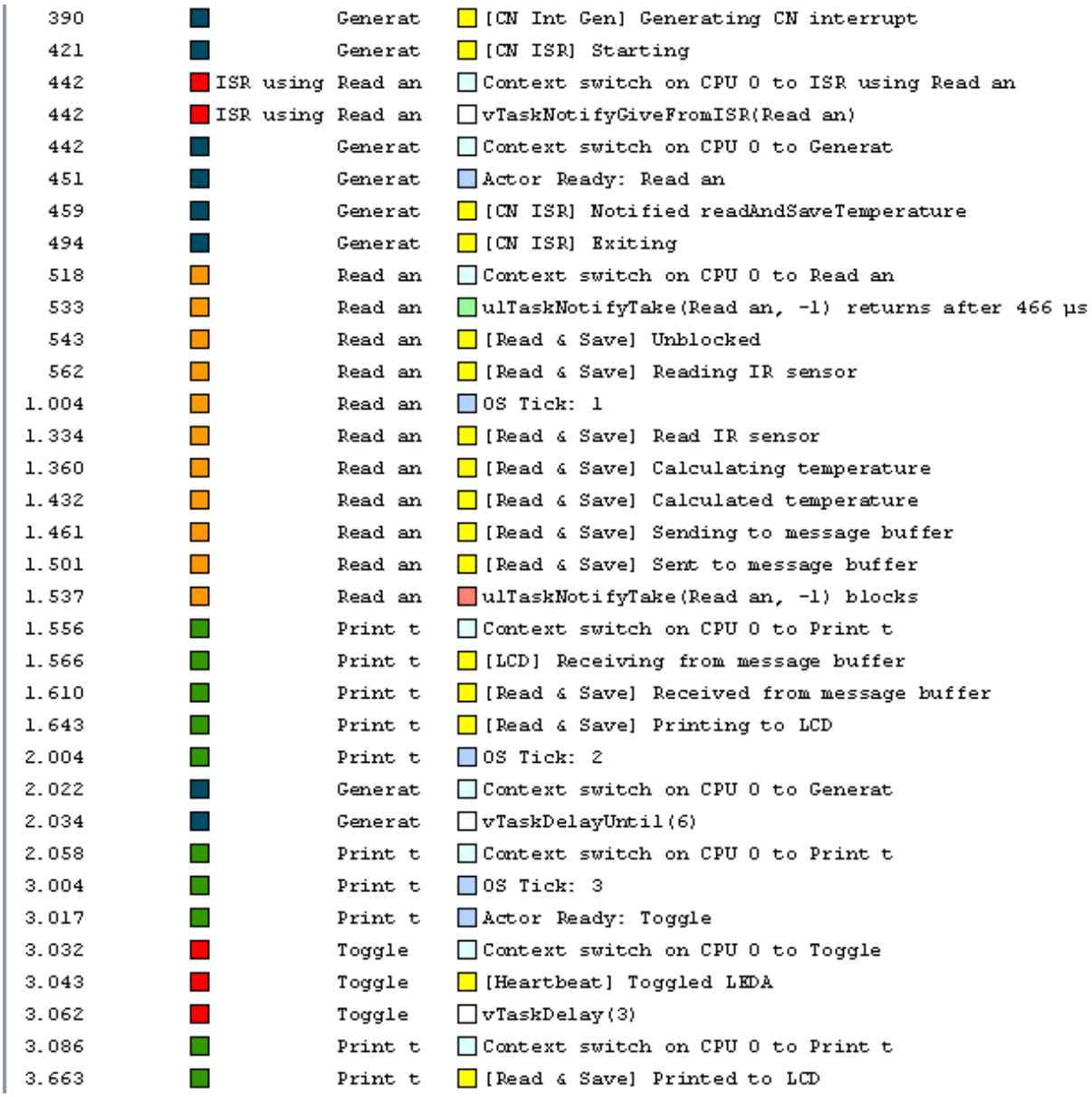
Parker Piedmont

ECE 443

10/4/2021

Project 4 Report

Program Execution



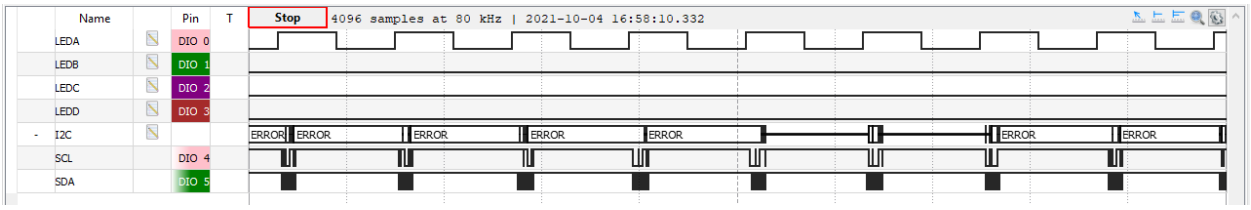
The screenshot above shows a trace of my program's execution, from generating a CN interrupt to printing to the LCD. This trace shows that my program successfully performed each operation in the project and performed them in the correct order. The screenshot shows that the readAndSaveTemperature task was unblocked immediately after the CN ISR exited (timestamp 0.543), verifying that my task notification worked correctly. The screenshot also shows that task preemption worked properly in my project, as the printToLCD task was preempted by the toggleLEDA task (timestamp 3.032). Additionally, the kernel successfully switched from the printToLCD task to the generateCNInt task, of the same priority, while the printToLCD task was busy waiting for the write to the LCD to finish (timestamp 2.022).

The screenshot above also indicates the performance of my project. The latency between generating the CN interrupt and completing the write to the LCD was around 3.3ms, well below the 6ms period at which the CN interrupt was generated. My program spent only 73µs in the CN ISR, about 25% faster than my project 3 CN ISR, which used a semaphore rather than a task notification. Reading from the IR sensor was surprisingly fast, consuming only about 800µs. Writing to the LCD was by far the slowest operation, taking almost exactly 2ms to complete.

Heartbeat

0	Toggle	Context switch on CPU 0 to Toggle
3	Toggle	[Heartbeat] Toggled LEDA
26	Toggle	vTaskDelay(3)
3.032	Toggle	Context switch on CPU 0 to Toggle
3.043	Toggle	[Heartbeat] Toggled LEDA
3.044	IDLE	End Of Trace

The screenshots above show two consecutive heartbeats that occurred 3.04ms apart. This is slightly higher than the 3ms delay I used, most likely due to the time it took the heartbeat task to toggle LEDA and print a trace message. Nevertheless, the screenshot above and the screenshot below show that my heartbeat task was not delayed by any other operations.



(The SMBus was not receiving errors – the window was simply zoomed out too far for Waveforms to represent the data.)

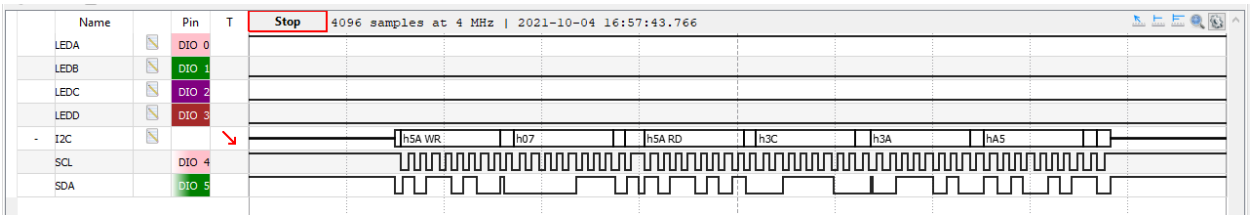
CN Interrupt Generator

```
   379          Generat   Context switch on CPU 0 to Generat
   390          Generat   [CN Int Gen] Generating CN interrupt
   421          Generat   [CN ISR] Starting

 6.095          Generat   Context switch on CPU 0 to Generat
 6.106          Generat   [CN Int Gen] Generating CN interrupt
 6.129          Generat   [CN ISR] Starting
```

The screenshots above show that the generateCNInt task executed about every 5.7ms. I am not sure why this task executed slightly early, since scheduling errors usually cause tasks to execute late. Perhaps this was the closest the scheduler could get to 6ms without going over.

SMBus



The screenshot above shows an example of reading from the IR sensor using SMBus. The waveform matches that in the IR sensor datasheet through the read control byte. My program successfully read three bytes from the IR sensor, including the PEC.