EE-156 Lab 3

Emmett Roberts & Griffin Faecher

10/6/2025

Using FreeRTOS and UART to Blink LEDS

Professor Joel Grodstein

Tufts University

Part One:

The global queue is only 32 bytes in size so those 25 instances don't all get written into the queue. It fills up. The xQueueSend() function adds the entire string to the global queue and if it doesn't fit waits until it can fit. Then it goes to writer 2 but since the queue is full it also taskdelays. Everything is delayed until the next tick interrupt. Then cycle repeats, daemon reads one char, writer 1 puts one more char on the queue (filling it back up), writer 2 auto delays because full queue.\

The way it switches the round robin order is by using a while(1) loop at the end of writer 1 task so that it takes the whole rest of the tick up until the interrupt. So after the daemon runs the first next tick, writer 2 task gets called first.

Writer 2 task never finishes because once the writer 2 task puts all its characters in the queue, it will set the boolean g_writer2_done boolean to true and therefore, the daemon will stop printing characters since the printing happens in this loop. While (!g_writer1_done || !g_writer2_done)

Part Two:

The characters switch every tick from writer 1 to writer 2 because the queue starts as empty and since xQueueReceive() is called in daemon, the daemon stops and goes to next task, and as soon as task writer 1 goes into serial write lab 3 and puts the first character into the queue, daemon immediately gets priority. Since daemon has the highest priority, as soon as the queue is not empty, the xQueueReceive() will resume and print out that one character in the queue. Now the queue is empty again so the xQueueReceive() blocks again and writer 2 task writes its first character into the queue, daemon resumes and writes that character. And it keeps alternating until they are both done.

Part Three:

The problem is that task one is never ending from an interrupt/tick, it always gets called when xQueueReceive() has an empty queue and blocks, then it grabs a mutex making the tick unable to interrupt it, and adds the entire string to the queue. Queue receive can't take back control after one character; it has to let task1 write the whole string because it has a mutex, but as soon as the mutex is released, daemon takes back control and writes the whole string. It will only switch from writer 1 to writer 2 when it finishes and runs a while one loop, making the round robing switch to writer 2. Another issue is that it doesn't finish printing the queue because the daemon while loop depends on bools that switch as soon as tasks are done. Since the writers add strings at a time and daemon prints characters at a time, it won't be able to write the last characters since the

To fix this we implemented a vTaskDelay after the task writer 1 or 2 writes one string, we put one in the for loop, after the start string and end string. This ensures that the next writer task will run after the daemon queue loop becomes empty. To fix the characters not being printed out you can add a condition to the while loop, keep writing characters until the writer task has put everything in the queue && the queue is empty.