

Assignment : 07 - Producer Consumer Task
Subject : CMPE - 244
Submitted by : Tirth Pandya
Date : 10/20/2020

(1) Consumer 'High', Producer 'Low'

- Consumer task has high priority and waits on the queue till the queue element is received.
- Producer task has low priority and it sleeps for 1000ms after sending the element to the queue.

[illegible]

- Observation :**

Having high priority, consumer task gets executed first but it is waiting on the queue to receive an element, therefore co-operative context-switch happens and producer task sends the data to the queue. As soon as the element is sent to

the queue, co-operative context-switch takes place and consumer task gets executed and after that only the producer task's contents [printf statement] gets executed.

(2) Consumer 'Low', Producer 'High'

- Producer task has high priority. It sends data to the queue and sleeps for 1000ms after sending the data.
- Consumer task has low priority and it waits on the queue receive to receive the data.

[illegible]

- **Observation :**

The producer task having high priority, sends the data to the queue and sleeps for 1000ms. Then the preemptive context switching takes place and the consumer task having low priority gets executed which is waiting on the queue to receive data. Once the producer task wakes up, the context-switching takes place for the producer task.

(3) Equal priority for both the tasks

- Both the producer and consumer tasks have equal(low) priority. The producer task goes to sleeps for 1000ms after sending the data to the queue and the consumer task waits on the queue till the data is received.

[illegible]

- **Observation :**

The producer task sends the data on the queue and goes to sleep. The consumer task gets executed as soon as the producer task sleeps and the preemptive context switching takes place for the producer task once it wakes up from the sleep.

Question

What is the purpose of the block time during xQueueReceive()?

Answer

When a task is waiting on the queue to receive an element, the block time enables the task to check the queue status periodically and the task sleeps in between the intervals. This releases the CPU and reduces the CPU usage for the receive task.

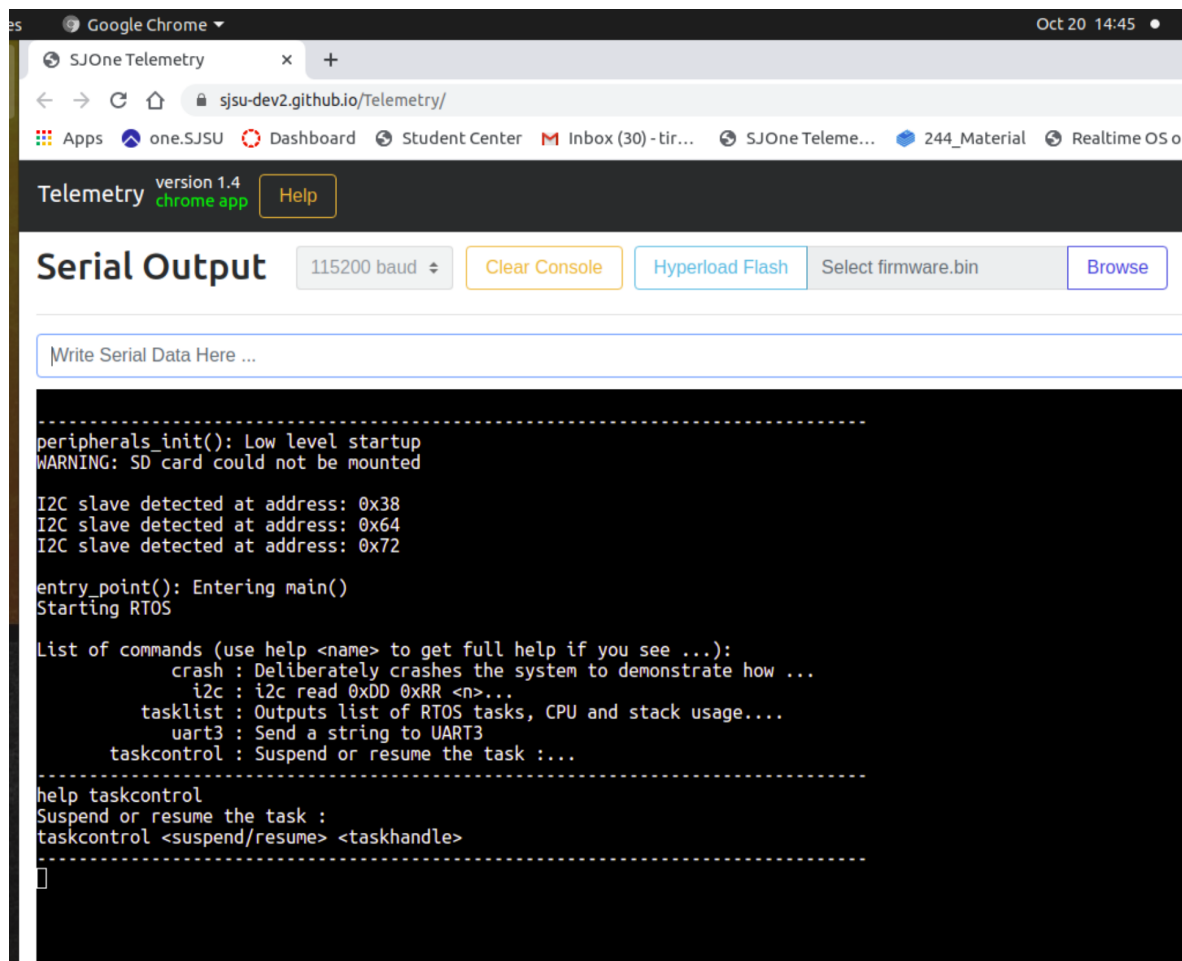
Question

What if you use ZERO block time during xQueueReceive()?

Answer

If there is zero block time for a queue receive task, then the receive task will poll the CPU until the data is received and the receive task will never go to sleep and exhaust the CPU cycles.

Extra Credit Assignment : CLI command handler



The screenshot shows a web browser window with the URL `sjsu-dev2.github.io/Telemetry/`. The page has a dark header with "Telemetry version 1.4 chrome app" and a "Help" button. Below the header is a "Serial Output" section with a baud rate dropdown set to "115200 baud", a "Clear Console" button, a "Hyperload Flash" button, a "Select firmware.bin" button, and a "Browse" button. A text input field labeled "Write Serial Data Here ..." is present. The main area displays a black terminal window with white text showing the following output:

```
-----
peripherals_init(): Low level startup
WARNING: SD card could not be mounted

I2C slave detected at address: 0x38
I2C slave detected at address: 0x64
I2C slave detected at address: 0x72

entry_point(): Entering main()
Starting RTOS

List of commands (use help <name> to get full help if you see ...):
  crash : Deliberately crashes the system to demonstrate how ...
  i2c : i2c read 0xDD 0xRR <n>...
  tasklist : Outputs list of RTOS tasks, CPU and stack usage....
  uart3 : Send a string to UART3
  taskcontrol : Suspend or resume the task :...
-----
help taskcontrol
Suspend or resume the task :
taskcontrol <suspend/resume> <taskhandle>
-----
█
```

Google Chrome Oct 20 14:14

SJOne Telemetry x +

sjsu-dev2.github.io/Telemetry/

Apps one.SJSU Dashboard Student Center Inbox (30) - tir... SJOne Teleme... 244_Material Realtime OS o... P

Telemetry version 1.4 chrome app Help

Serial Output

115200 baud Clear Console Hyperload Flash Select firmware.bin Browse

Write Serial Data Here ...

```
tasklist : Outputs list of RTOS tasks, CPU and stack usage...
uart3 : Send a string to UART3
taskcontrol : Suspend or resume the task :...
-----
L-----L-----LED : 0
LED : 1
LED : 0
LED : 1
LED : 0
LED : 1
LED : 0
LED : 1
LED : 0
LED : 1
LED : 0
LED : 1
LED : 0
LED : 1
LED : 0
LED : 1
taskcontrol suspend led0
-----
LED : 1
LED : 1
LED : 1
LED : 1
LED : 1
LED : 1
LED : 1
LED : 1
LED : 1
LED : 1
LED : 1
LED : 1
taskcontrol resume led0
-----
LED : 0
LED : 1
LED : 0
LED : 1
LED : 0
```

EOF