CmpE 244: Lab Watchdog-app

Akhil Cherukuri 10/27/2020
Akash Vachhani
Salvo Nicosia

Code:

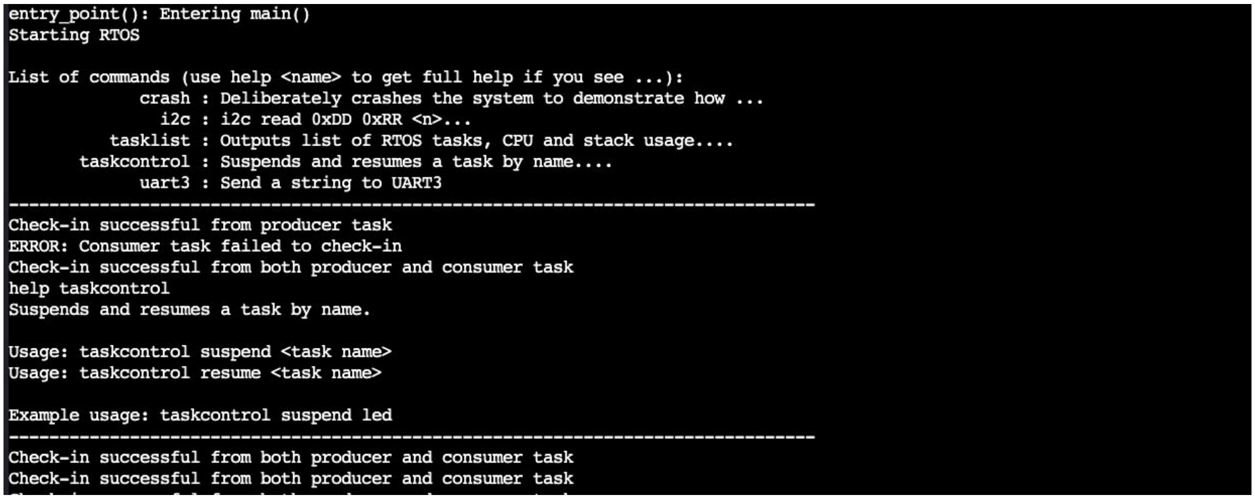
- Handlers
    - sensor_queue
        - QueueHandle_t type
    - checkin
        - EventGroupHandle_t type
    - mutex
        - SemaphoreHandle_t type
- Tasks
    - producer_task
        - Void type
        - 512 bytes
    - consumer_task
        - Void type
        - 512 bytes
    - watchdog_task
        - Void type
        - 512 bytes
- (Void) producer_task (void *p)
    - Variables
        - average_sensor_value
            - uint16_t
        - sensor_values_sum
            - uint16_t
        - sensor_values_count
            - Size_t
    - Logic
        - While(1)
            - If (count < 100)
                - Sensor_values_sum += acceleration__get_data().x
                    - Get the x axis data and save to sensor_values_sum
                - Increment count for sensor_values_count
                    - So we know how much data we have
                - vTaskDelay(1)
            - Else
                - Calculate average_sensor_value
                    - Sensor_values_sum / sensor_values_count
                - xQueueSend()
                    - Send average_sensor_value

- xEventGroupSetBits(checkin, bit_1)
  - Set eventgroupbit for producer task to true
  - Saying the event happened
- Reset sum and count values to 0
- vTaskDelay(100)

- (Void) consumer_task (void *p)
  - Variables
    - Average_sensor_value
      - int16_t
    - time_elapsed
      - uint32_t
    - *filename
      - Const char
      - Set to "accelerometer_x_axis_data.csv"
  - Logic
    - xQueueReceive()
      - Receive average_sensor_value
    - If (sys_uptime_ms - time_elapsed is more than 1 second)
      - Take mutex
        - Write value to filename
          - Using write_file_using_fatfs_pi
      - Give mutex back
        - Mutex is to safeguard writing to SD card
      - Update time_elapsed value to current sys_time__get_uptime_ms()
    - Call xEventGroupSetBits to set bit_2
      - Allows checkin for consumer task
- (Void) watchdog_task (void *p)
  - Variables
    - Wait_for_bit_1_bit_2
      - EventBits_t type
    - *filename
      - Const Char type
      - Set to "accelerometer_x_axis_data.csv"
    - Bits_set
    - EventBits_t type
  - Logic
    - If (bit 1 and 2 are set)
      - Printf
        - "Check-in successful from both producer and consumer task”
    - Else if bit 1 is not set
      - Printf
        - “Check-in successful from producer task”
        - "ERROR: Consumer task failed to check-in”
      - Take Mutex

        - Write error message to file on SD card
    - Give Mutex back
- Else if bit 2 not set
    - Printf
        - “Check-in successful from consumer task”
        - "ERROR: Producer task failed to check-in”
    - Take Mutex
        - Write error message to file on SD card
    - Give Mutex back
- Else
    - Printf
        - "ERROR: Producer and Consumer task failed to check-in within the 200ms threshold”
        - "ERROR: Possibly due to consumer task blocked waiting on receiving as result of producer task failing to check-in”
    - Take mutex
        - Write error messages to file on SD card
    - Give mutex back
- vTaskDelay(1000)

Screenshots and Observation:

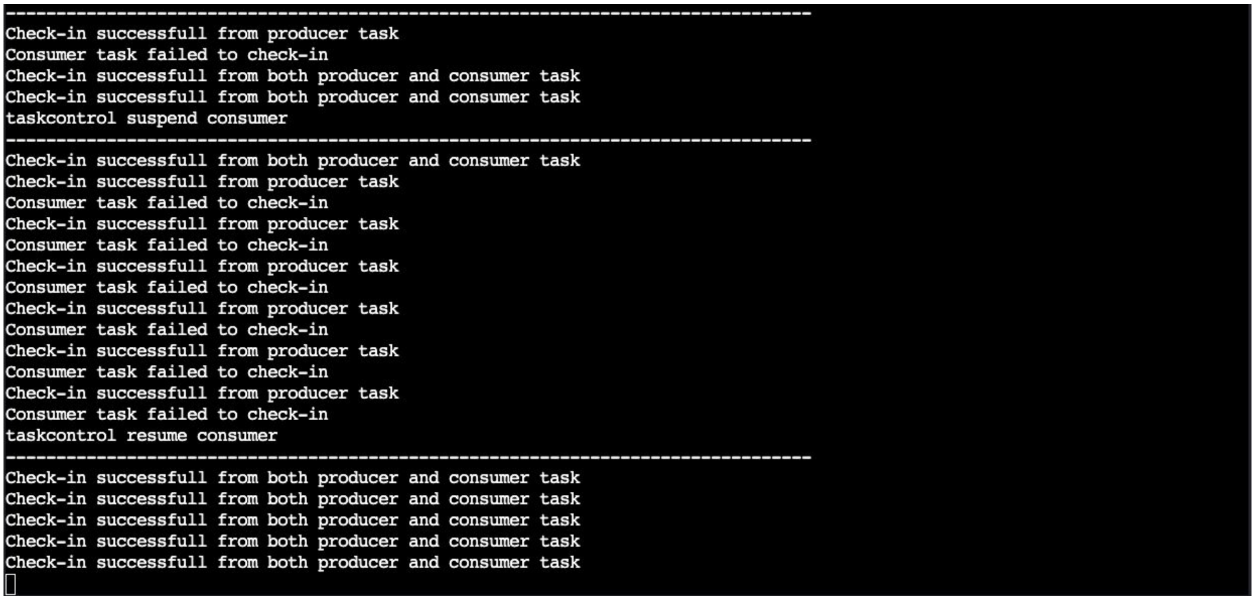


```
entry_point(): Entering main()
Starting RTOS

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....
       taskcontrol : Suspends and resumes a task by name....
             uart3 : Send a string to UART3
--------------------------------------------------------------------------------
Check-in successful from producer task
ERROR: Consumer task failed to check-in
Check-in successful from both producer and consumer task
help taskcontrol
Suspends and resumes a task by name.

Usage: taskcontrol suspend <task name>
Usage: taskcontrol resume <task name>

Example usage: taskcontrol suspend led
--------------------------------------------------------------------------------
Check-in successful from both producer and consumer task
Check-in successful from both producer and consumer task
```

Figure 1: help taskcontrol cli

Observations:

- “help taskcontrol” command
    - taskcontrol suspend <task name>
    - taskcontrol resume <task name>
- Figure 1 shows how we are able to successfully print out help menu for the taskcontrol command

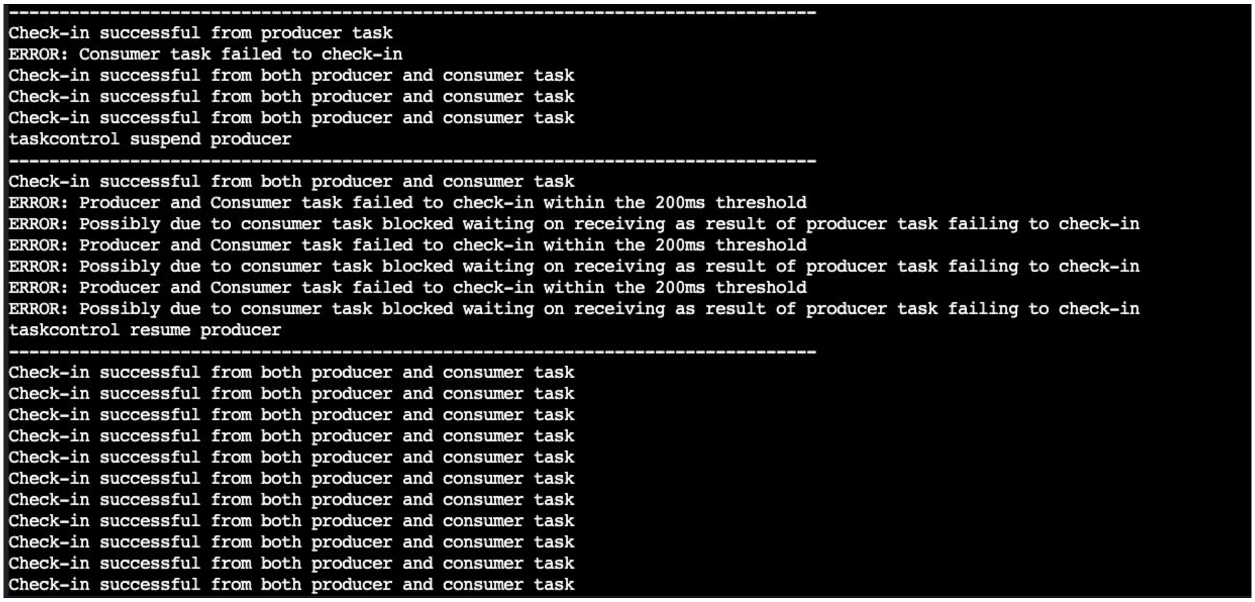


```
--------------------------------------------------------------------------------
Check-in successfull from producer task
Consumer task failed to check-in
Check-in successfull from both producer and consumer task
Check-in successfull from both producer and consumer task
taskcontrol suspend consumer
--------------------------------------------------------------------------------
Check-in successfull from both producer and consumer task
Check-in successfull from producer task
Consumer task failed to check-in
Check-in successfull from producer task
Consumer task failed to check-in
Check-in successfull from producer task
Consumer task failed to check-in
Check-in successfull from producer task
Consumer task failed to check-in
Check-in successfull from producer task
Consumer task failed to check-in
Check-in successfull from producer task
Consumer task failed to check-in
taskcontrol resume consumer
--------------------------------------------------------------------------------
Check-in successfull from both producer and consumer task
Check-in successfull from both producer and consumer task
Check-in successfull from both producer and consumer task
Check-in successfull from both producer and consumer task
Check-in successfull from both producer and consumer task
```

Figure 2: taskcontrol suspend consumer and taskcontrol resume consumer

Observations:

- Figure 2 highlights how we suspend the consumer task and then resume it moments later
- As shown, once we suspend our task, we see that the watchdog_task prints that the consumer task failed to check-in
- When the task is resumed, we can see the watchdog_task print that both tasks checked in



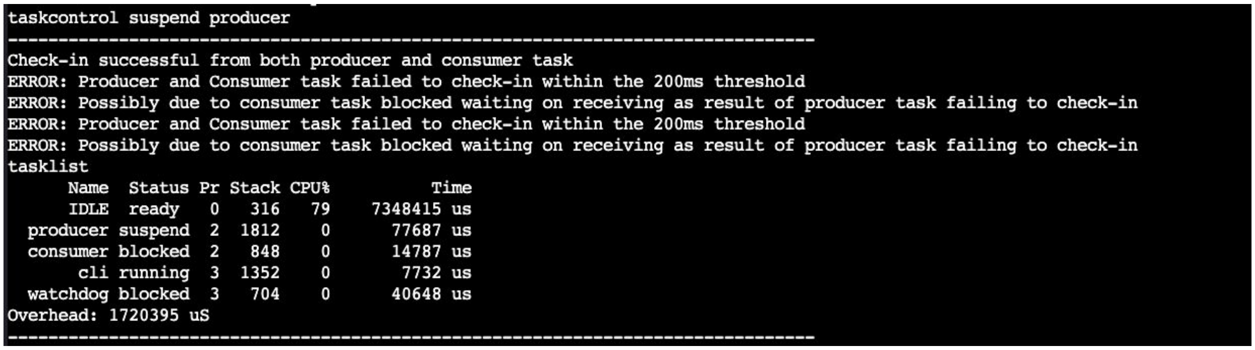
```
--------------------------------------------------------------------------------
Check-in successful from producer task
ERROR: Consumer task failed to check-in
Check-in successful from both producer and consumer task
Check-in successful from both producer and consumer task
Check-in successful from both producer and consumer task
taskcontrol suspend producer
--------------------------------------------------------------------------------
Check-in successful from both producer and consumer task
ERROR: Producer and Consumer task failed to check-in within the 200ms threshold
ERROR: Possibly due to consumer task blocked waiting on receiving as result of producer task failing to check-in
ERROR: Producer and Consumer task failed to check-in within the 200ms threshold
ERROR: Possibly due to consumer task blocked waiting on receiving as result of producer task failing to check-in
ERROR: Producer and Consumer task failed to check-in within the 200ms threshold
ERROR: Possibly due to consumer task blocked waiting on receiving as result of producer task failing to check-in
taskcontrol resume producer
--------------------------------------------------------------------------------
Check-in successful from both producer and consumer task
Check-in successful from both producer and consumer task
Check-in successful from both producer and consumer task
Check-in successful from both producer and consumer task
Check-in successful from both producer and consumer task
Check-in successful from both producer and consumer task
Check-in successful from both producer and consumer task
Check-in successful from both producer and consumer task
Check-in successful from both producer and consumer task
Check-in successful from both producer and consumer task
Check-in successful from both producer and consumer task
```

Figure 3: taskcontrol suspend producer and taskcontrol resume producer

Observations:

- Figure 2 highlights that the producer task was unable to checkin

    - Due to this, the consumer task freezes
        - portMAX_DELAY means wait until we the data is ready
        - Because producer is suspended, no data is sent
        - Both producer and consumer event bits remain unset
    - When the producer task is resumed
        - Functionality is resumed and we see that both task can checkin

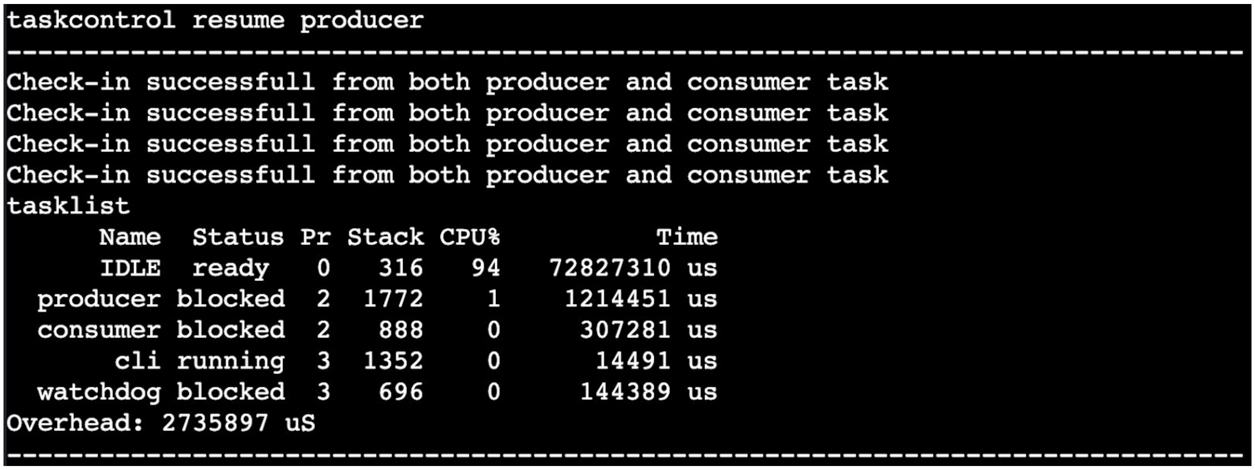


```
taskcontrol suspend producer
---------------------------------------------------------------------------
Check-in successful from both producer and consumer task
ERROR: Producer and Consumer task failed to check-in within the 200ms threshold
ERROR: Possibly due to consumer task blocked waiting on receiving as result of producer task failing to check-in
ERROR: Producer and Consumer task failed to check-in within the 200ms threshold
ERROR: Possibly due to consumer task blocked waiting on receiving as result of producer task failing to check-in
tasklist
      Name  Status Pr Stack CPU%          Time
      IDLE  ready   0   316   79    7348415 us
  producer suspend  2  1812    0      77687 us
  consumer blocked  2   848    0      14787 us
       cli running  3  1352    0       7732 us
  watchdog blocked  3   704    0      40648 us
Overhead: 1720395 uS
---------------------------------------------------------------------------
```

Figure 4: tasklist when taskcontrol suspend producer

Observations:

- Tasklist output in figure 4
    - Highlights consumer task is blocked and producer task is suspended
- CPU remains 79% IDLE
    - Consumer is waiting for data and is blocked until the data is available from the queue



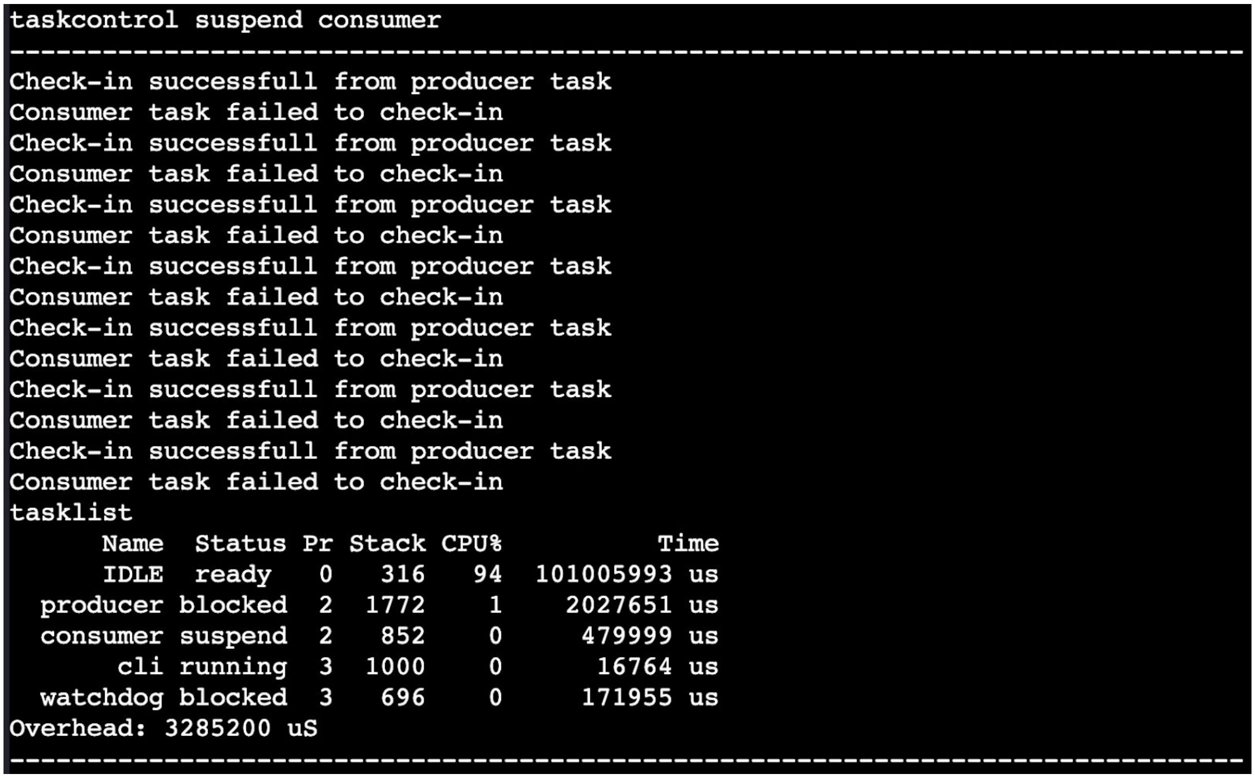
```
taskcontrol resume producer
--------------------------------------------------------------------------------
Check-in successfull from both producer and consumer task
Check-in successfull from both producer and consumer task
Check-in successfull from both producer and consumer task
Check-in successfull from both producer and consumer task
tasklist
      Name  Status Pr Stack CPU%          Time
      IDLE  ready   0   316   94   72827310 us
  producer blocked  2  1772    1    1214451 us
  consumer blocked  2   888    0     307281 us
       cli running  3  1352    0      14491 us
  watchdog blocked  3   696    0     144389 us
Overhead: 2735897 uS
--------------------------------------------------------------------------------
```

Figure 5: tasklist when taskcontrol resume producer

Observations:

- tasklist output in figure 5
    - Highlights producer task has resumed
- CPU usage is 94% idle

- Producer task is using 1% after being resumed

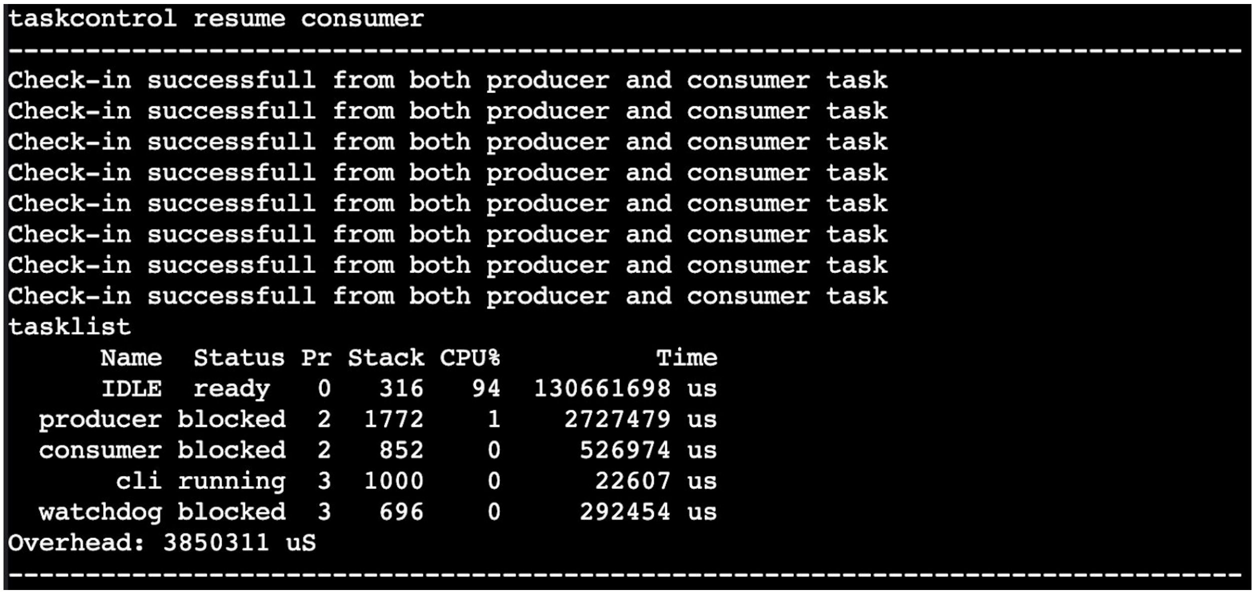


```
taskcontrol suspend consumer
--------------------------------------------------------------------------------
Check-in successfull from producer task
Consumer task failed to check-in
Check-in successfull from producer task
Consumer task failed to check-in
Check-in successfull from producer task
Consumer task failed to check-in
Check-in successfull from producer task
Consumer task failed to check-in
Check-in successfull from producer task
Consumer task failed to check-in
Check-in successfull from producer task
Consumer task failed to check-in
Check-in successfull from producer task
Consumer task failed to check-in
tasklist
      Name  Status Pr Stack CPU%          Time
      IDLE  ready   0   316   94  101005993 us
  producer blocked  2  1772    1    2027651 us
  consumer suspend  2   852    0     479999 us
       cli running  3  1000    0      16764 us
  watchdog blocked  3   696    0     171955 us
Overhead: 3285200 uS
--------------------------------------------------------------------------------
```

Figure 6: tasklist when taskcontrol suspend consumer

Observations:

- tasklist output in figure 6
  - Consumer task suspended as shown on output
- CPU usage is 94% idle
- Producer task is using 1%

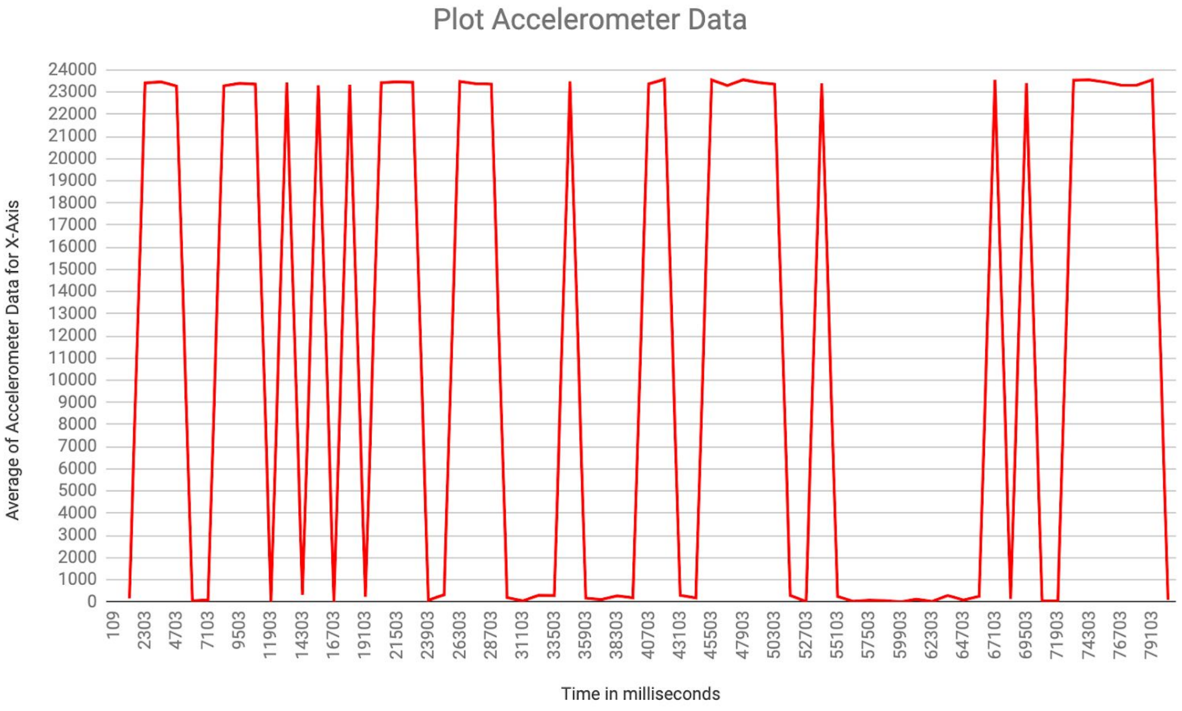


```
taskcontrol resume consumer
--------------------------------------------------------------------------------
Check-in successfull from both producer and consumer task
Check-in successfull from both producer and consumer task
Check-in successfull from both producer and consumer task
Check-in successfull from both producer and consumer task
Check-in successfull from both producer and consumer task
Check-in successfull from both producer and consumer task
Check-in successfull from both producer and consumer task
Check-in successfull from both producer and consumer task
tasklist
      Name  Status Pr Stack CPU%          Time
      IDLE  ready   0   316   94  130661698 us
  producer blocked  2  1772    1    2727479 us
  consumer blocked  2   852    0     526974 us
       cli running  3  1000    0      22607 us
  watchdog blocked  3   696    0     292454 us
Overhead: 3850311 uS
--------------------------------------------------------------------------------
```

Figure 7: tasklist when taskcontrol resume consumer

Observations:

- tasklist output in figure 7
    - Consumer task resumed as shown on output
- CPU usage is 94% idle
- Producer task is using 1%



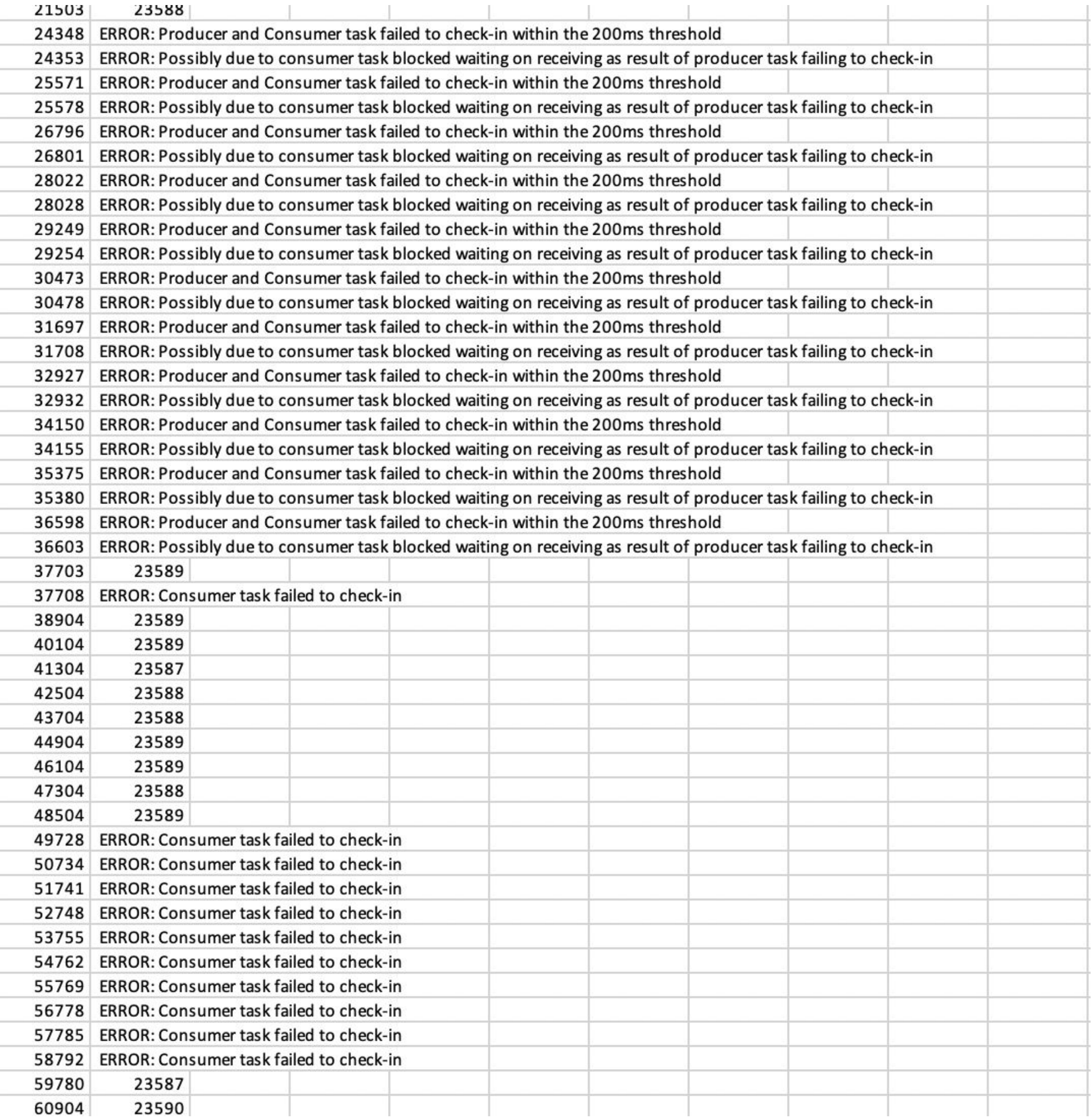
Graph: Average of Accelerometer Data for X-Axis

[Rocking SJ-2 Board Back and Forth along X-Axis]

Data in Excel Sheet:

109	ERROR: Consumer task failed to check-in
1101	146
2303	23407
3503	23459
4703	23267
5903	49
7103	85
8303	23274
9503	23391
10703	23350
11903	12
13103	23428
14303	316
15503	23307
16703	7
17903	23328
19103	233
20303	23416
21503	23465
22703	23441
23903	73
25103	317
26303	23479
27503	23371
28703	23354
29903	190
31103	44
32303	290
33503	281
34703	23479
35903	165

37103	103
38303	269
39503	179
40703	23364
41903	23578
43103	293
44303	172
45503	23549
46703	23287
47903	23556
49103	23431
50303	23348
51503	283
52703	17
53903	23395
55103	234
56303	28
57503	73
58703	54
59903	2
61103	121
62303	20



21503	23588
24348	ERROR: Producer and Consumer task failed to check-in within the 200ms threshold
24353	ERROR: Possibly due to consumer task blocked waiting on receiving as result of producer task failing to check-in
25571	ERROR: Producer and Consumer task failed to check-in within the 200ms threshold
25578	ERROR: Possibly due to consumer task blocked waiting on receiving as result of producer task failing to check-in
26796	ERROR: Producer and Consumer task failed to check-in within the 200ms threshold
26801	ERROR: Possibly due to consumer task blocked waiting on receiving as result of producer task failing to check-in
28022	ERROR: Producer and Consumer task failed to check-in within the 200ms threshold
28028	ERROR: Possibly due to consumer task blocked waiting on receiving as result of producer task failing to check-in
29249	ERROR: Producer and Consumer task failed to check-in within the 200ms threshold
29254	ERROR: Possibly due to consumer task blocked waiting on receiving as result of producer task failing to check-in
30473	ERROR: Producer and Consumer task failed to check-in within the 200ms threshold
30478	ERROR: Possibly due to consumer task blocked waiting on receiving as result of producer task failing to check-in
31697	ERROR: Producer and Consumer task failed to check-in within the 200ms threshold
31708	ERROR: Possibly due to consumer task blocked waiting on receiving as result of producer task failing to check-in
32927	ERROR: Producer and Consumer task failed to check-in within the 200ms threshold
32932	ERROR: Possibly due to consumer task blocked waiting on receiving as result of producer task failing to check-in
34150	ERROR: Producer and Consumer task failed to check-in within the 200ms threshold
34155	ERROR: Possibly due to consumer task blocked waiting on receiving as result of producer task failing to check-in
35375	ERROR: Producer and Consumer task failed to check-in within the 200ms threshold
35380	ERROR: Possibly due to consumer task blocked waiting on receiving as result of producer task failing to check-in
36598	ERROR: Producer and Consumer task failed to check-in within the 200ms threshold
36603	ERROR: Possibly due to consumer task blocked waiting on receiving as result of producer task failing to check-in
37703	23589
37708	ERROR: Consumer task failed to check-in
38904	23589
40104	23589
41304	23587
42504	23588
43704	23588
44904	23589
46104	23589
47304	23588
48504	23589
49728	ERROR: Consumer task failed to check-in
50734	ERROR: Consumer task failed to check-in
51741	ERROR: Consumer task failed to check-in
52748	ERROR: Consumer task failed to check-in
53755	ERROR: Consumer task failed to check-in
54762	ERROR: Consumer task failed to check-in
55769	ERROR: Consumer task failed to check-in
56778	ERROR: Consumer task failed to check-in
57785	ERROR: Consumer task failed to check-in
58792	ERROR: Consumer task failed to check-in
59780	23587
60904	23590

Figure 8: Error messages in data file when tasks are suspended and resumed

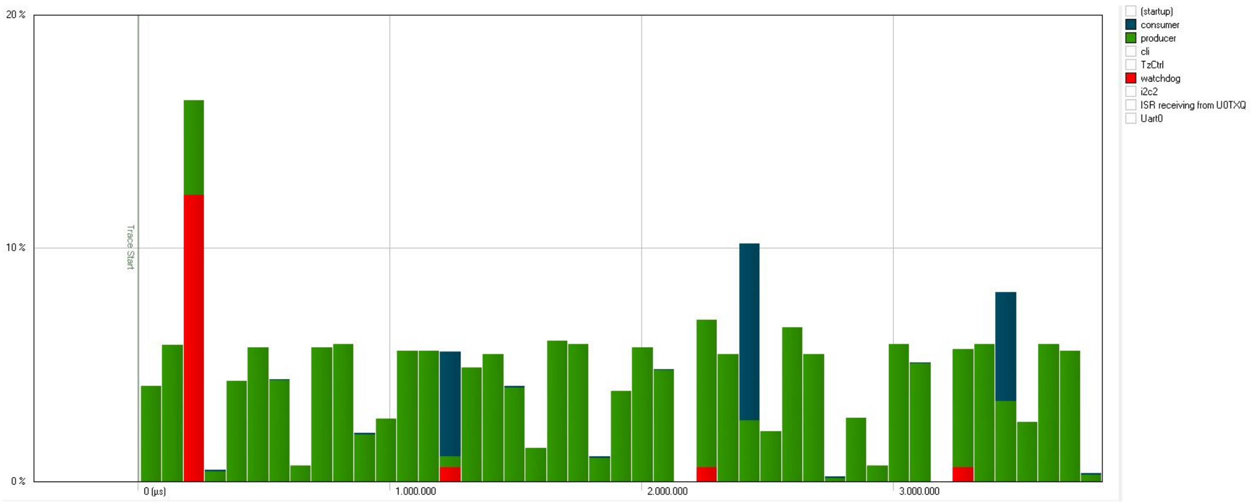


Figure 9: FreeRTOS Trace (Our results)

Observations:

- Isolated the data to showcase watchdog, producer and consumer task
- CPU Usage:
    - When SD card was written
        - Consumer utilized 3.8%
        - Producer utilized 2.39%
        - Watchdog utilized 0.5%
    - When SD Card was not written to
        - Consumer utilized 0.16%
        - Producer utilized 5.62%
        - Watchdog utilized 0.98%
- When the SD card was written, more CPU time was utilized to write data to the card
    - This was the expected behavior
        - Watchdog has highest priority but waits for data to be ready
            - The data is event bits being set
        - Consumer needs enough time to finish writing to the SD card
        - Producer continues to collect data in a round robin schedule since both consumer and producer have equal priority
- When SD card was not written to, producer utilized a majority of the CPU usage
    - This was the expected behavior
        - Data is only written to SD card once every 1 second by the consumer
        - Producer during this time uses more CPU usage to produce the accelerometer data
        - Watch continues to wait for data to be set, if they aren't than data is written to SD card
            - This usage is shown in the 2nd bar where the consumer task was not running, due to the round robin scheduler
                - Watchdog usage jumped to 12.31% due to the size of the error messages

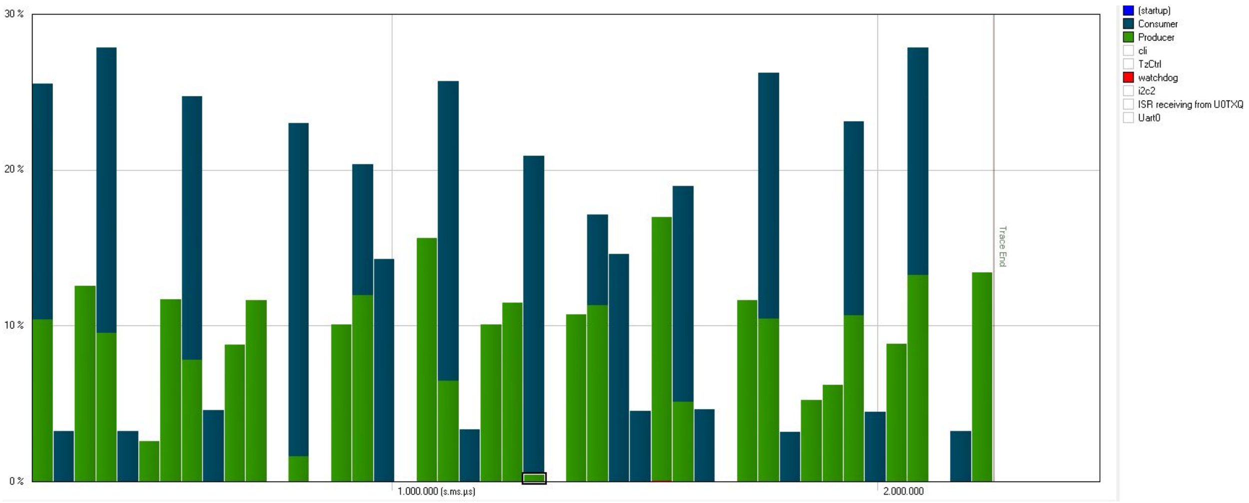


Figure 10: FreeRTOS Trace (Preet's Provided)

Observations:

- Isolated the data to showcase watchdog, producer and consumer task
- CPU Usage:
  - Producer Utilizes about the same amount as our producer task
  - Consumer task is utilizing 3x the CPU usage than our consumer task
  - Watchdog utilizes about the same/a little less than our watchdog task
    - We believe Preet's watchdog task is not logging information to the SD card which is why we do not see those initial spikes when a task is not running

Note: Trace file collected can be found in the l5_application folder of the merge request