Nathan Bunnell
Embedded Real-Time Operating Systems
ECE-40290
Student ID: U08895857

Date: 11/9/2021

# Assignment 5: Queues

The following will document completion of the fifth assignment for ECE-40290, with the stated goals of:

- **Create "Queue1" that is length of five to hold up to five uint32_t values.**
- **Create "Task 1" that will run every 2 seconds, increment a count, then send the count to the queue. Use a value of portMAX_DELAY when writing to the queue.**
- **Create "Task 2" that will receive from the queue, then display the count received from the queue. Use a value of portMAX_DELAY when reading from the queue.**
- **Run this on the STM32CubeIDE (with IoT Discovery Board or equivalent) and observe the result.**

## 1. Create "Queue1" that is length of five to hold up to five uint32_t values

To start, we'll generate a default project as before. Under the FreeRTOS configuration window, select the *Task and Queues* tab. Create Queue1 of size 5 * uint32_t as described.

Queues

| Queue Name | Queue Size | Item Size | A |
|---|---|---|---|
| Queue1 | 5 | uint32_t | Dynamic |

## 2. Create "Task 1" that will run every 2 seconds, increment a count, then send the count to the queue. Use a value of portMAX_DELAY when writing to the queue.

Next create Task1 as in previous assignments.

Config parameters | Include parameters | Advanced settings | User Constants | Tasks and Queues

Tasks

| Task Name | Priority | Stack Size (Words) | Entry Function | Code Generation Op |
|---|---|---|---|---|
| Task1 | osPriorityNormal | 128 | StartTask1 | Default |
| Task2 | osPriorityBelowNormal | 128 | StartTask2 | Default |

Within main.c, we can define *StartTask1()* as seen below, where a counter value is declared and initialized to zero in the setup area. Within the for(;;) section, the value is sent to the queue created previously, an error me sage will be printed if necessary, then the counter is incremented before the task delays for two seconds.

Nathan Bunnell
Embedded Real-Time Operating Systems
ECE-40290
Student ID: U08895857

Date: 11/9/2021

```
492  /* USER CODE BEGIN Header_StartTask1 */
493  /**
494    * @brief  Function implementing the Task1 thread.
495    * @param  argument: Not used
496    * @retval None
497    */
498  /* USER CODE END Header_StartTask1 */
499  void StartTask1(void const * argument)
500  {
501    /* USER CODE BEGIN 5 */
502
503            uint32_t counterValue = 0;
504
505            // Result status of queue send to test for success
506            BaseType_t sendResults;
507
508    /* Infinite loop */
509    for(;;)
510    {
511            sendResults = xQueueSend(Queue1Handle, &counterValue, portMAX_DELAY );
512
513            // Test results for success
514            if (sendResults != pdPASS)
515            {
516                    char* errorMsg = "Failed to send value to queue!!!\n";
517                    HAL_UART_Transmit(&huart1, (uint8_t*) errorMsg, strlen(errorMsg), 1000);
518            }
519
520            // Increment counter
521            counterValue++;
522
523            // Two second delay
524            osDelay(2000);
525    }
526    /* USER CODE END 5 */
527  }
528
```

Nathan Bunnell
Embedded Real-Time Operating Systems
ECE-40290
Student ID: U08895857

Date: 11/9/2021

3. **Create "Task 2" that will receive from the queue, then display the count received from the queue. Use a value of portMAX_DELAY when reading from the queue.**

Repeat creation process for Task2. *StartTask2()* will be configured to receive the counter value out of the queue and print it on the UART1 console, or an error message if necessary.

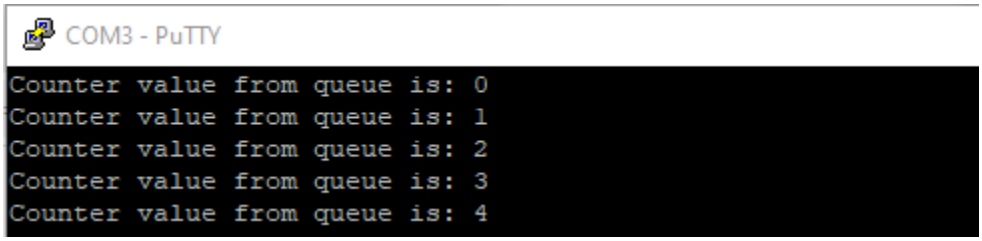
```
529  /* USER CODE BEGIN Header_StartTask2 */
530  /**
531  * @brief Function implementing the Task2 thread.
532  * @param argument: Not used
533  * @retval None
534  */
535  /* USER CODE END Header_StartTask2 */
536  void StartTask2(void const * argument)
537  {
538    /* USER CODE BEGIN StartTask2 */
539            uint32_t receivedValue = 0;
540
541        // Result status of queue receive to test for success
542            BaseType_t receiveResults;
543
544    /* Infinite loop */
545    for(;;)
546    {
547            // Test if queue is empty or not
548            if (uxQueueMessagesWaiting(Queue1Handle) != 0)
549            {
550                    // Receive the value in the queue
551                    receiveResults = xQueueReceive(Queue1Handle, &receivedValue, portMAX_DELAY );
552
553                    // Test results for success
554                    if (receiveResults != pdPASS)
555                    {
556                            char* errorMsg = "Failed to receive value from queue!!!\n";
557                            HAL_UART_Transmit(&huart1, (uint8_t*) errorMsg, strlen(errorMsg), 1000);
558                    }
559                    else
560                    {
561                            char buffer[100];
562                            snprintf(buffer, sizeof(buffer), "Counter value from queue is: %lu\n", receivedValue);
563                        HAL_UART_Transmit(&huart1, (uint8_t*) buffer, strlen(buffer), 1000);
564                    }
565            }
566        osDelay(1);
567    }
568    /* USER CODE END StartTask2 */
569  }
570
```

Nathan Bunnell
Embedded Real-Time Operating Systems
ECE-40290
Student ID: U08895857

Date: 11/9/2021

4. **Run this on the STM32CubeIDE (with IoT Discovery Board or equivalent) and observe the result.**

With the program loaded to the dev board, the console can be opened and the counter value will be seen printing.




**Closing Thoughts**

AS I've noted in other assignments, this example took what could be a frustrating idea to implement, and wraps it up in a nice and easy to use abstraction. I'll be interested to see how this concept is applied in creating the ping-pong feature in the final.