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

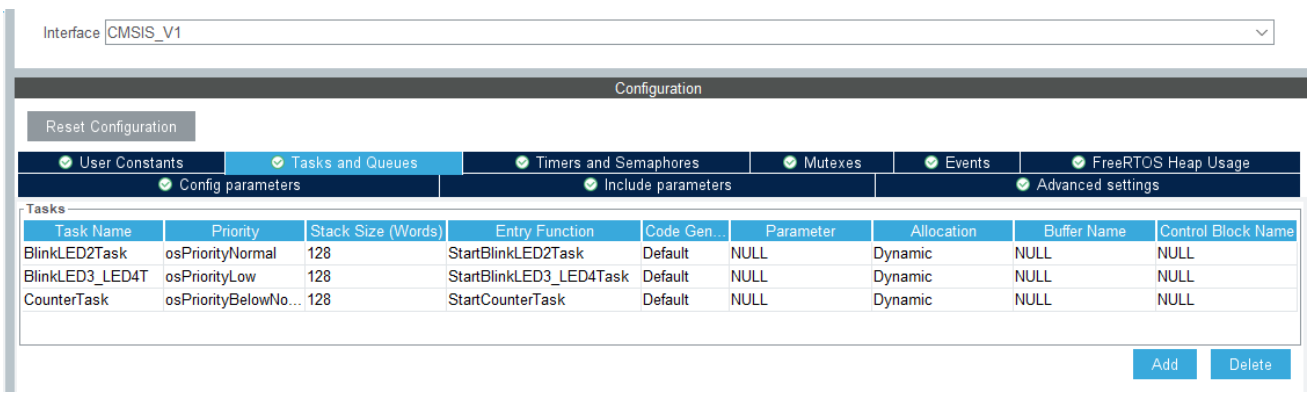
Date: 10/25/2021

Assignment 3: Tasks

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

- **Create "Task 1" that blinks the LED2 (Green LED) every 250 msec**
- **Create "Task 2" that blinks LED3_WIFI_LED4_BLE every 1000 msec**
- **Create "Task 3" that displays a counter (e.g. "count = 1", "count = 2", etc.) on the STM32 Virtual Console**

After generating a new project file with the default FreeRTOS config we've used in previous examples, we can navigate to the Task and Queues tab under the configuration interface and define the three tasks listed above, using appropriately descriptive names for each. Note the varied priority levels between each to prevent creating a locked-out condition between one task or another of equal priority.



Task Name	Priority	Stack Size (Words)	Entry Function	Code Gen...	Parameter	Allocation	Buffer Name	Control Block Name
BlinkLED2Task	osPriorityNormal	128	StartBlinkLED2Task	Default	NULL	Dynamic	NULL	NULL
BlinkLED3_LED4T	osPriorityLow	128	StartBlinkLED3_LED4Task	Default	NULL	Dynamic	NULL	NULL
CounterTask	osPriorityBelowNo...	128	StartCounterTask	Default	NULL	Dynamic	NULL	NULL

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

Date: 10/25/2021

1. **Create "Task 1" that blinks the LED2 (Green LED) every 250 msec**

Tasks defined in the configuration ad the project rebuilt, we can define code as needed for each one. BlinkLED2Task is self-explanatory:

```
/* USER CODE BEGIN Header_StartBlinkLED2Task */
/**
  * @brief  Function implementing the BlinkLED2Task thread.
  * @param  argument: Not used
  * @retval None
  */
/* USER CODE END Header_StartBlinkLED2Task */
void StartBlinkLED2Task(void const * argument)
{
  /* USER CODE BEGIN 5 */
  /* Infinite loop */
  for(;;)
  {
      // Create "Task1" that blinks LED2 (Green LED) every 250 msec
      HAL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin);
      osDelay(250);
  }
  /* USER CODE END 5 */
}
```

2. **Create "Task 2" that blinks LED3_WIFI_LED4_BLE every 1000 msec**

And again, for BlinkLED3_LED4Task:

```
/* USER CODE BEGIN Header_StartBlinkLED3_LED4Task */
/**
* @brief Function implementing the BlinkLED3_LED4T thread.
* @param argument: Not used
* @retval None
*/
/* USER CODE END Header_StartBlinkLED3_LED4Task */
void StartBlinkLED3_LED4Task(void const * argument)
{
  /* USER CODE BEGIN StartBlinkLED3_LED4Task */
  /* Infinite loop */
  for(;;)
  {
      // Create "Task2" that blinks LED3_WIFI__LED4_BLE every 1000 msec
      HAL_GPIO_TogglePin(LED3_WIFI__LED4_BLE_GPIO_Port, LED3_WIFI__LED4_BLE_Pin);
      osDelay(1000);
  }
  /* USER CODE END StartBlinkLED3_LED4Task */
}
```

3. **Create "Task 3" that displays a counter (e.g. "count = 1", "count = 2", etc.) on the STM32 Virtual Console**

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

Date: 10/25/2021

CounterTask is similarly straightforward:

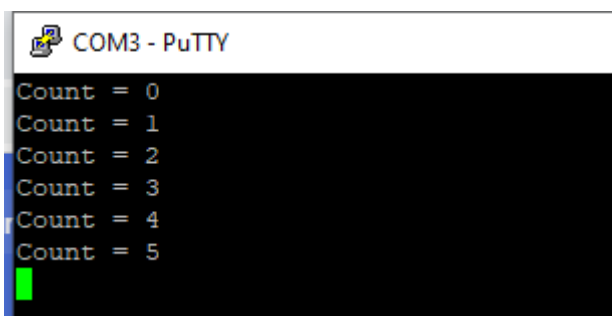
```c
/* USER CODE BEGIN Header_StartCounterTask */
/**
* @brief Function implementing the CounterTask thread.
* @param argument: Not used
* @retval None
*/
/* USER CODE END Header_StartCounterTask */
void StartCounterTask(void const * argument)
{
  /* USER CODE BEGIN StartCounterTask */

  unsigned int counter = 0;

  /* Infinite loop */
  for(;;)
  {
      // Create "Task 3" that displays a counter on the virtual console
      char buffer[100];
      snprintf(buffer, sizeof(buffer), "Count = %d\n", counter++);
      HAL_UART_Transmit(&huart1, (uint8_t*) buffer, strlen(buffer), 1000);

      osDelay(1000);
  }
  /* USER CODE END StartCounterTask */
}
```

All of that complete and downloaded to the board, we observe the LEDs ticking away at their proscribed intervals as well as the counter incrementing on the console:



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

Date: 10/25/2021

Closing Thoughts

Again, a fairly straightforward assignment to complete. A couple of years back, I picked up the Kindle version of *Beginning STM32: Developing with FreeRTOS, libopencm3 and GCC* and read the first few chapters during a flight back east for the holidays. At the time, I found a lot of the content to be both abstract and difficult to follow as well as kind of boring. As we continue to progress through the course and the reading in the *Mastering FreeRTOS* book, I see how wrong I was to dismiss the concept so quickly.