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

Date: 11/9/2021

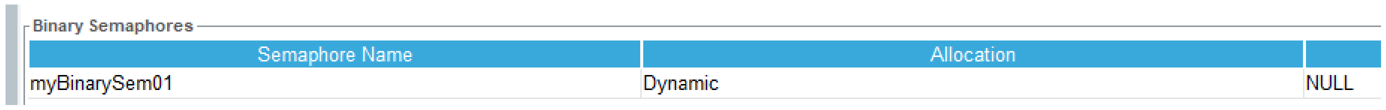
Assignment 6: Interrupts

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

- **Create a binary semaphore.**
- **Create a task that will wait on the semaphore, then it should toggle LED2 on your STM board.**
- **The ISR should release the semaphore whenever it receives an interrupt. On the STM board, the ISR will receive an interrupt by pressing the Blue button. Every time you press the Blue button on the STM board, you should see the LED2 toggle on or off.**

1. Create a binary semaphore.

After creating a new, default FreeRTOS project, create a semaphore object under the *Semaphore* tab.

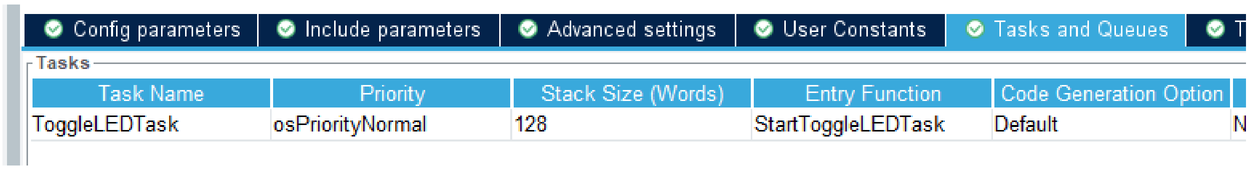


Binary Semaphores

Semaphore Name	Allocation	
myBinarySem01	Dynamic	NULL

2. Create a task that will wait on the semaphore, then it should toggle LED2 on your STM board.

Under the *Tasks and Queues* tab, create the toggle task as in previous assignments.



Tasks

Task Name	Priority	Stack Size (Words)	Entry Function	Code Generation Option	
ToggleLEDTask	osPriorityNormal	128	StartToggleLEDTask	Default	N

Within main.c, *StartToggleLEDTask()* can be defined to attempt to take the semaphore, and, if successful, toggle the LED2 GPIO pin. I also added a simple console message to show functionality within the scope of this report.

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

Date: 11/9/2021

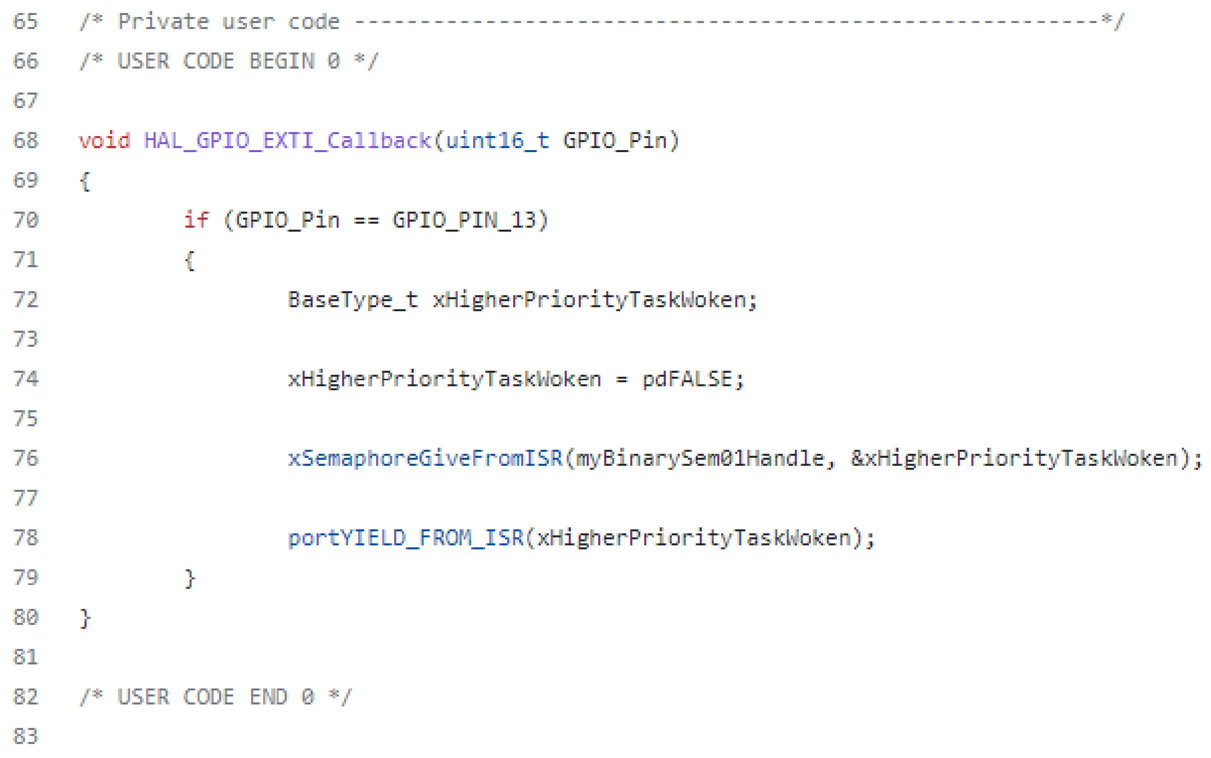
```c
/* USER CODE BEGIN Header_StartToggleLEDTask */
/**
  * @brief  Function implementing the ToggleLEDTask thread.
  * @param  argument: Not used
  * @retval None
  */
/* USER CODE END Header_StartToggleLEDTask */
void StartToggleLEDTask(void const * argument)
{
  /* USER CODE BEGIN 5 */

        // Used to test results of taking the semaphore
        BaseType_t takeResults;

  /* Infinite loop */
  for(;;)
  {
          // Attempt to take the semaphore
          takeResults = xSemaphoreTake(myBinarySem01Handle, portMAX_DELAY);

          // Test results, toggle LED on success
          if (takeResults == pdPASS)
          {
                  HAL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin);

                  // Add in printf as a proof-of-function
                  char* toggleMsg = "Toggle...\n";
                  HAL_UART_Transmit(&huart1, (uint8_t*) toggleMsg, strlen(toggleMsg), 1000);
          }

      osDelay(1);
  }
  /* USER CODE END 5 */
}
```

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

Date: 11/9/2021

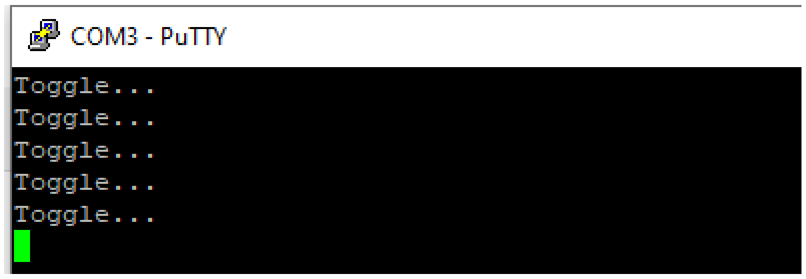
3. **The ISR should release the semaphore whenever it receives an interrupt. On the STM board, the ISR will receive an interrupt by pressing the Blue button. Every time you press the Blue button on the STM board, you should see the LED2 toggle on or off.**

Finally, we'll define the *HAL_GPIO_EXTI_Callback()* function to give the semaphore object in the event that Blue user button on GPIO13 is pressed.



```
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
        if (GPIO_Pin == GPIO_PIN_13)
        {
                BaseType_t xHigherPriorityTaskWoken;

                xHigherPriorityTaskWoken = pdFALSE;

                xSemaphoreGiveFromISR(myBinarySem01Handle, &xHigherPriorityTaskWoken);

                portYIELD_FROM_ISR(xHigherPriorityTaskWoken);
        }
}

/* USER CODE END 0 */
```

All of this code flashed to the target board, we can press the button and see the LED blink, or open the console and see the toggle message print:



COM3 - PuTTY

```
Toggle...
Toggle...
Toggle...
Toggle...
Toggle...
```

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

Date: 11/9/2021

Closing Thoughts

Many years ago (high school, I think), I stumbled onto the Wikipedia page for semaphores while browsing random CS topics. I recall reading an example that likened the software concept to the signals used historically at intersections such as railways and not really getting the relationship. Years later, looking through the lens of this class, its almost laughable how simple the concept turned out to understand in theory and practice. AS Ive noted previously, I really like how the course and readings break out all of these concepts in easy to understand chunks that are in turn easy to assemble together into a functional program.