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

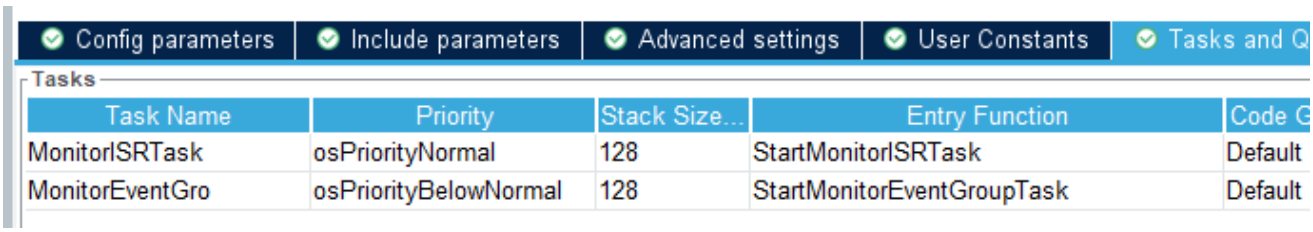
Date: 11/16/2021

# Assignment 8: Event Groups

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

- **An ISR will be triggered by the blue user button and set one of three bits in an event group.**
- **A task will wait on this ISR bit to be set, then will delay 1 second, set a second bit, followed by another 1 second delay and then setting the third bit.**
- **A second task will wait till all three bits are set, clear them, turn the LED2 on, delay a second, then turn it off and wait for the bits to be set again.**

Initial setup will be as all other projects: selecting FreeRTOS from the config menu, selecting a clock source, etc. We will define two tasks, the one to wait for the bit being set in the ISR, and another to wait on the second pair of bits that will toggle the LED.




Additional steps specific to this assignment included setting the *configUSE_TIMERS* option to enabled as well as setting *xTimerPendFunctionCall* and *xEventGroupSetBitFromISR* options to enabled under the Include options tab of the FreeRTOS configuration.


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

Date: 11/16/2021

I didn't dig in enough to find out why, but I was unable to add an event group from the Events tab so in addition to defining the names of bitmasks used in reference to the group, it was declared manually in main.c and initialized in *main()*:

```
37  // Define event group bit names
38  #define FIRST_TASK_BIT (1 << 0)
39  #define SECOND_TASK_BIT (1 << 1)
40  #define ISR_TASK_BIT (1 << 2)
41
56  // Declare event group
57  EventGroupHandle_t eventGroup;
58
59  /* USER CODE END PV */
```

```
122     // Init our event group
123     eventGroup = xEventGroupCreate();
124
```

1. **An ISR will be triggered by the blue user button and set one of three bits in an event group.**

Setup complete, we can define a simple ISR callback using the *xEventGroupSetBitsFromISR()* call to set the ISR_TASK_BIT:

```
75  // Define callback function for user button
76  void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
77  {
78          if (GPIO_Pin == GPIO_PIN_13)
79          {
80                  BaseType_t xHigherPriorityTaskWoken = pdFALSE;
81
82                  // Set the approrpiate bit
83                  xEventGroupSetBitsFromISR(eventGroup, ISR_TASK_BIT, &xHigherPriorityTaskWoken);
84
85                  portYIELD_FROM_ISR(xHigherPriorityTaskWoken);
86          }
87  }
```

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

Date: 11/16/2021

2. **A task will wait on this ISR bit to be set, then will delay 1 second, set a second bit, followed by another 1 second delay and then setting the third bit.**

The task to monitor for that ISR_TASK_BIT can then be configured to check the value of the event group against that bitmask, and once it is made, delay, set the FIRST_TASK_BIT, delay, then the SECOND_TASK_BIT, seen here:

```
516  void StartMonitorISRTask(void const * argument)
517  {
518    /* USER CODE BEGIN 5 */
519
520          // Declare local variable to read-in value of event group
521          EventBits_t readBitsVal;
522
523    /* Infinite loop */
524    for(;;)
525    {
526          // Read value of event bits, do not clear on exit, wait for all bits, use arbitrary delay value
527          readBitsVal = xEventGroupWaitBits(eventGroup, ISR_TASK_BIT, pdFALSE, pdTRUE, pdMS_TO_TICKS (100));
528
529          // If the ISR bit has been set, enter a section to delay, set the next bit,
530          //   delay again, set the final bit
531          if (readBitsVal & ISR_TASK_BIT)
532          {
533            osDelay(1000);
534            xEventGroupSetBits(eventGroup, FIRST_TASK_BIT);
535            osDelay(1000);
536            xEventGroupSetBits(eventGroup, SECOND_TASK_BIT);
537          }
538
539      osDelay(1);
540    }
541    /* USER CODE END 5 */
542  }
```

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

Date: 11/16/2021

3. **A second task will wait till all three bits are set, clear them, turn the LED2 on, delay a second, then turn it off and wait for the bits to be set again.**

In similar fashion, the second task will monitor for all three bits to be set then proceed with a delay/toggle operation. Re-reading the assignment in writing this report, I realized the specification was actually for delay, LED on, delay, LED off. I think the difference is negligible enough to not bother rewriting with the correct HAL API calls.

```
551  void StartMonitorEventGroupTask(void const * argument)
552  {
553    /* USER CODE BEGIN StartMonitorEventGroupTask */
554
555          // Declare local variable to read-in value of event group
556          EventBits_t readBitsVal;
557
558          // Value to monitor against
559          EventBits_t monitorBits = (FIRST_TASK_BIT | SECOND_TASK_BIT | ISR_TASK_BIT);
560
561    /* Infinite loop */
562    for(;;)
563    {
564          // Read value of event bits, clear bits on successful exit, wait for all bits, use arbitrary delay value
565          readBitsVal = xEventGroupWaitBits(eventGroup, monitorBits, pdTRUE, pdTRUE, pdMS_TO_TICKS (100));
566
567          // If all bits are set, enter a section to delay, toggle the LED,
568          //   delay again, then toggle the LED again
569          if (readBitsVal == monitorBits)
570          {
571            osDelay(1000);
572            HAL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin);
573            osDelay(1000);
574            HAL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin);
575          }
576
577          osDelay(1);
578    }
579    /* USER CODE END StartMonitorEventGroupTask */
580  }
```

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

Date: 11/16/2021

**Closing Thoughts**

Yet again, a concept that was well represented and simple to execute but that has potential for lots of critical use in a “real” project for task management and synchronization. It’s obviously a different implementation of an RTOS but I use bit-flags for sequence control frequently in my PLC programs so this felt very natural to pick up.