Exercise 5

```
441 int startFlag = 1;
442 void StartAccessSoftware()
443 {
444   /* USER CODE BEGIN 5 */
445   /* Infinite loop */
446   for(;;)
447   {
448       if(startFlag == 1)
449       {
450           startFlag = 0;
451       }
452       else
453       {
454           HAL_GPIO_TogglePin(GPIOD, LD6_Pin);//Toggel Blue LED
455       }
456     for(int i = 0; i < 2000000; i++);//Delay ~0.5 sec
457     startFlag = 1;
458     return;
459   }
460   /* USER CODE END 5 */
461 }
```

```
471 void StartGreenTask(void const * argument)
472 {
473   /* USER CODE BEGIN 5 */
474   /* Infinite loop */
475   for(;;)
476   {
477       HAL_GPIO_WritePin(GPIOD, LD4_Pin, GPIO_PIN_SET);//Green LED ON
478       StartAccessSoftware();//Enter simulation of R/W
479       HAL_GPIO_WritePin(GPIOD, LD4_Pin, GPIO_PIN_RESET);//Green LED OFF
480
481       for(int i = 0; i < 2000000; i++);//Delay ~0.5 sec
482   }
483   /* USER CODE END 5 */
484 }
```

```
493 void StartRedTask(void const * argument)
494 {
495   /* USER CODE BEGIN StartRedTask */
496   /* Infinite loop */
497   for(;;)
498   {
499       HAL_GPIO_WritePin(GPIOD, LD5_Pin, GPIO_PIN_SET);//RED LED ON
500       StartAccessSoftware();//Enter simulation of R/W
501       HAL_GPIO_WritePin(GPIOD, LD5_Pin, GPIO_PIN_RESET);//RED LED OFF
502
503       osDelay(100);//delay 0.1 sec
504   }
505   /* USER CODE END StartRedTask */
506 }
```

Exercise 6)

```
458 void StartAccessFunction(void)
459 {
460   /* USER CODE BEGIN 5 */
461   /* Infinite loop */
462   for(;;)
463   {
464       if(startFlag == 1)
465       {
466           startFlag = 0;
467       }
468       else
469       {
470           HAL_GPIO_TogglePin(LD6_GPIO_Port, LD6_Pin);//Enable Blue LED
471       }
472 
473       for(int i=0; i < 2000000; i++); //~0.5 second delay
474       startFlag = 1;
475       //HAL_GPIO_WritePin(LD6_GPIO_Port, LD6_Pin, GPIO_PIN_RESET);//Disable Blue LED
476       //osDelay(500);// 0.5ms delay
477       return;
478   }
```

```
489 void StartGreenTask(void const * argument)
490 {
491   /* USER CODE BEGIN StartGreenTask */
492   /* Infinite loop */
493   for(;;)
494   {
495       HAL_GPIO_WritePin(LD4_GPIO_Port, LD4_Pin, GPIO_PIN_SET);//Enable Green LED
496       taskENTER_CRITICAL();//disable interrupts
497       StartAccessFunction();
498       taskEXIT_CRITICAL();//enable interrupts
499       for(int i=0; i < 2000000; i++); //~0.5 second delay
500       HAL_GPIO_WritePin(LD4_GPIO_Port, LD4_Pin, GPIO_PIN_RESET);//Off Green LED
501 
502       for(int i=0; i < 2000000; i++); //~0.5 second delay
503   }
504   /* USER CODE END StartGreenTask */
505 }
```

```
514 void StartRedTAsk(void const * argument)
515 {
516   /* USER CODE BEGIN StartRedTAsk */
517   /* Infinite loop */
518   for(;;)
519   {
520 
521       HAL_GPIO_WritePin(LD5_GPIO_Port, LD5_Pin, GPIO_PIN_SET);//Enable Red LED
522       taskENTER_CRITICAL();//disable interrupts
523       StartAccessFunction();
524       taskEXIT_CRITICAL();//enable interrupts
525 
526       HAL_GPIO_WritePin(LD5_GPIO_Port, LD5_Pin, GPIO_PIN_RESET);//Off Red LED
527 
528       for(int i=0; i < 500000; i++); //~0.1 second delay
529   }
530   /* USER CODE END StartRedTAsk */
531 }
```

Document was last saved: Just now

Exercise 7)

P1

```
477 void StartGreenTask(void const * argument)
478 {
479   /* USER CODE BEGIN 5 */
480   /* Infinite loop */
481   for(;;)
482   {
483       HAL_GPIO_WritePin(GPIOD, LD4_Pin, GPIO_PIN_SET);//Green LED ON
484
485       osDelay(200);
486       StartAccessSoftware();//Enter simulation of R/W
487
488       HAL_GPIO_WritePin(GPIOD, LD4_Pin, GPIO_PIN_RESET);//Green LED OFF
489
490       osDelay(200);//delay 0.2 sec
491   }
492   /* USER CODE END 5 */
493 }
```

```
502 void StartRedTask(void const * argument)
503 {
504   /* USER CODE BEGIN StartRedTask */
505   /* Infinite loop */
506   for(;;)
507   {
508       HAL_GPIO_WritePin(GPIOD, LD5_Pin, GPIO_PIN_SET);//RED LED ON
509
510       osDelay(550);
511       StartAccessSoftware();//Enter simulation of R/W
512
513       HAL_GPIO_WritePin(GPIOD, LD5_Pin, GPIO_PIN_RESET);//RED LED OFF
514
515       osDelay(550);//delay 0.55 sec
516   }
517   /* USER CODE END StartRedTask */
518 }
```

```
527 void StartOrangeTask(void const * argument)
528 {
529   /* USER CODE BEGIN StartOrangeTask */
530   /* Infinite loop */
531   for(;;)
532   {
533       HAL_GPIO_TogglePin(GPIOD, LD3_Pin);//Orange LED ON
534       osDelay(50);//delay 0.5 sec
535   }
536   /* USER CODE END StartOrangeTask */
537 }
```

P2

```
477 void StartGreenTask(void const * argument)
478 {
479   /* USER CODE BEGIN 5 */
480   /* Infinite loop */
481   for(;;)
482   {
483       HAL_GPIO_WritePin(GPIOD, LD4_Pin, GPIO_PIN_SET);//Green LED ON
484
485       osDelay(200);
486       taskENTER_CRITICAL();//Disable Interrupts
487       StartAccessSoftware();//Enter simulation of R/W
488       taskEXIT_CRITICAL();//Enable Interrupts
489       HAL_GPIO_WritePin(GPIOD, LD4_Pin, GPIO_PIN_RESET);//Green LED OFF
490
491       osDelay(200);//delay 0.2 sec
492   }
493   /* USER CODE END 5 */
494 }
```

```
503 void StartRedTask(void const * argument)
504 {
505   /* USER CODE BEGIN StartRedTask */
506   /* Infinite loop */
507   for(;;)
508   {
509       HAL_GPIO_WritePin(GPIOD, LD5_Pin, GPIO_PIN_SET);//RED LED ON
510
511       osDelay(550);
512       taskENTER_CRITICAL();//Disable Interrupts
513       StartAccessSoftware();//Enter simulation of R/W
514       taskEXIT_CRITICAL();//Enable Interrupts
515       HAL_GPIO_WritePin(GPIOD, LD5_Pin, GPIO_PIN_RESET);//RED LED OFF
516
517       osDelay(550);//delay 0.55 sec
518   }
519   /* USER CODE END StartRedTask */
520 }
```

Exercise 8)

```
511 void StartRedTask(void const * argument)
512 {
513   /* USER CODE BEGIN StartRedTask */
514   /* Infinite loop */
515   for(;;)
516   {
517       HAL_GPIO_WritePin(GPIOD, LD5_Pin, GPIO_PIN_SET);//RED LED ON
518
519       osSemaphoreWait(CriticalResourceSemaphoreHandle, osWaitForever);
520       //taskENTER_CRITICAL();//Disable Interrupts
521       StartAccessSoftware();//Enter simulation of R/W
522       //taskEXIT_CRITICAL();//Enable Interrupts
523       osSemaphoreRelease(CriticalResourceSemaphoreHandle);
524       osDelay(550);
525       HAL_GPIO_WritePin(GPIOD, LD5_Pin, GPIO_PIN_RESET);//RED LED OFF
526
527       osDelay(550);//delay 0.55 sec
528   }
529   /* USER CODE END StartRedTask */
530 }
```

```
483 void StartGreenTask(void const * argument)
484 {
485   /* USER CODE BEGIN 5 */
486   /* Infinite loop */
487   for(;;)
488   {
489       HAL_GPIO_WritePin(GPIOD, LD4_Pin, GPIO_PIN_SET);//Green LED ON
490       osSemaphoreWait(CriticalResourceSemaphoreHandle, osWaitForever);
491       //taskENTER_CRITICAL();//Disable Interrupts
492       StartAccessSoftware();//Enter simulation of R/W
493
494       osSemaphoreRelease(CriticalResourceSemaphoreHandle);
495       //taskEXIT_CRITICAL();//Enable Interrupts
496       osDelay(200);//delay 0.2 sec
497       HAL_GPIO_WritePin(GPIOD, LD4_Pin, GPIO_PIN_RESET);//Green LED OFF
498
499       osDelay(200);//delay 0.2 sec
500   }
501   /* USER CODE END 5 */
502 }
```