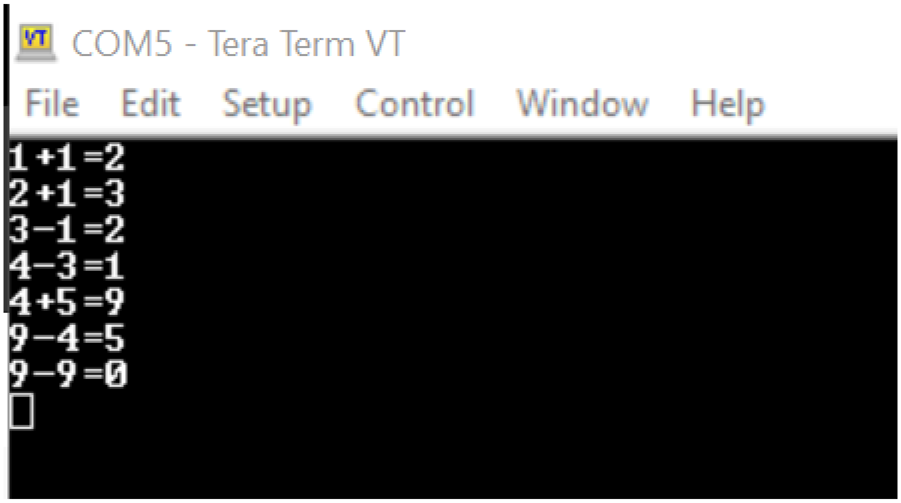
The American University in Cairo
The Department of Computer Science and Engineering

Ismael Elsharkawi 900173030

Embedded Systems Lab
Lab 5 Report

Link to Youtube video: https://youtu.be/e7YF2VyqdIM



```
1+1=2
2+1=3
3-1=2
4-3=1
4+5=9
9-4=5
9-9=0
```

main.c

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under Ultimate Liberty license
  * SLA0044, the "License"; You may not use this file except in compliance with
  * the License. You may obtain a copy of the License at:
  *                             www.st.com/SLA0044
  *
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "cmsis_os.h"
#include "queue.h"
```

```
#include "semphr.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
UART_HandleTypeDef huart1;
SemaphoreHandle_t event_signal;

//QueueHandle_t queueRQ=0;
//QueueHandle_t queueSQ=0;
uint8_t my_char;
/* Definitions for defaultTask */
osThreadId_t defaultTaskHandle;
const osThreadAttr_t defaultTask_attributes = {
  .name = "defaultTask",
  .priority = (osPriority_t) osPriorityNormal,
  .stack_size = 128 * 4
};
/* Definitions for Task1_Evaluate */
osThreadId_t Task1_EvaluateHandle;
const osThreadAttr_t Task1_Evaluate_attributes = {
  .name = "Task1_Evaluate",
  .priority = (osPriority_t) osPriorityNormal,
  .stack_size = 128 * 4
};
/* Definitions for Task2_UARTSend */
osThreadId_t Task2_UARTSendHandle;
```

```
const osThreadAttr_t Task2_UARTSend_attributes = {
  .name = "Task2_UARTSend",
  .priority = (osPriority_t) osPriorityNormal,
  .stack_size = 128 * 4
};
/* Definitions for Task3_UARTRecei */
osThreadId_t Task3_UARTReceiHandle;
const osThreadAttr_t Task3_UARTRecei_attributes = {
  .name = "Task3_UARTRecei",
  .priority = (osPriority_t) osPriorityNormal,
  .stack_size = 128 * 4
};
/* Definitions for queueRQ */
osMessageQueueId_t queueRQHandle;
const osMessageQueueAttr_t queueRQ_attributes = {
  .name = "queueRQ"
};
/* Definitions for queueSQ */
osMessageQueueId_t queueSQHandle;
const osMessageQueueAttr_t queueSQ_attributes = {
  .name = "queueSQ"
};
/* Definitions for usart_lock */
osMutexId_t usart_lockHandle;
const osMutexAttr_t usart_lock_attributes = {
  .name = "usart_lock"
};
/* Definitions for event_signal */
osSemaphoreId_t event_signalHandle;
const osSemaphoreAttr_t event_signal_attributes = {
  .name = "event_signal"
};
/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART1_UART_Init(void);
void StartDefaultTask(void *argument);
void StartTask02(void *argument);
void StartTask03(void *argument);
void StartTask04(void *argument);
```

```
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

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

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART1_UART_Init();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* Init scheduler */
  osKernelInitialize();
  /* Create the mutex(es) */
  /* creation of usart_lock */
  usart_lockHandle = osMutexNew(&usart_lock_attributes);

  /* USER CODE BEGIN RTOS_MUTEX */
  /* add mutexes, ... */
  /* USER CODE END RTOS_MUTEX */

  /* Create the semaphores(s) */
  /* creation of event_signal */
  event_signalHandle = osSemaphoreNew(1, 1, &event_signal_attributes);

  /* USER CODE BEGIN RTOS_SEMAPHORES */
  /* add semaphores, ... */
  /* USER CODE END RTOS_SEMAPHORES */

  /* USER CODE BEGIN RTOS_TIMERS */
  /* start timers, add new ones, ... */
  /* USER CODE END RTOS_TIMERS */

  /* Create the queue(s) */
  /* creation of queueRQ */
  queueRQHandle = osMessageQueueNew (4, sizeof(uint16_t), &queueRQ_attributes);

  /* creation of queueSQ */
  queueSQHandle = osMessageQueueNew (1, sizeof(uint16_t), &queueSQ_attributes);

  /* USER CODE BEGIN RTOS_QUEUES */
  /* add queues, ... */
  /* USER CODE END RTOS_QUEUES */

  /* Create the thread(s) */
  /* creation of defaultTask */
  defaultTaskHandle = osThreadNew(StartDefaultTask, NULL, &defaultTask_attributes);

  /* creation of Task1_Evaluate */
  Task1_EvaluateHandle = osThreadNew(StartTask02, NULL, &Task1_Evaluate_attributes);

  /* creation of Task2_UARTSend */
  Task2_UARTSendHandle = osThreadNew(StartTask03, NULL,
&Task2_UARTSend_attributes);

  /* creation of Task3_UARTRecei */
  Task3_UARTReceiHandle = osThreadNew(StartTask04, NULL,
&Task3_UARTRecei_attributes);

  /* USER CODE BEGIN RTOS_THREADS */
  /* add threads, ... */
  /* USER CODE END RTOS_THREADS */

  /* USER CODE BEGIN RTOS_EVENTS */
  /* add events, ... */
  /* USER CODE END RTOS_EVENTS */

  /* Start scheduler */
//          HAL_UART_Receive(&huart1, &my_char, 1, HAL_MAX_DELAY);
//          xQueueSendToBackFromISR(queueRQ, &my_char, NULL);
//          uint8_t my_char_receive;
//          xQueueReceiveFromISR(queueRQ, &my_char_receive, NULL);
//          HAL_UART_Transmit(&huart1, &my_char_receive, 4, HAL_MAX_DELAY);
vSemaphoreCreateBinary( event_signal ); // Create the semaphore
            //xSemaphoreTake(event_signal, 0); // Take semaphore after creating it.
__HAL_UART_ENABLE_IT(&huart1,UART_IT_RXNE);
  osKernelStart();

  /* We should never get here as control is now taken by the scheduler */
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  RCC_PeriphCLKInitTypeDef PeriphClkInit = {0};
  /** Configure LSE Drive Capability
  */
  HAL_PWR_EnableBkUpAccess();
  __HAL_RCC_LSEDRIVE_CONFIG(RCC_LSEDRIVE_LOW);
  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType =
RCC_OSCILLATORTYPE_LSE|RCC_OSCILLATORTYPE_MSI;
  RCC_OscInitStruct.LSEState = RCC_LSE_ON;
  RCC_OscInitStruct.MSIState = RCC_MSI_ON;
  RCC_OscInitStruct.MSICalibrationValue = 0;
  RCC_OscInitStruct.MSIClockRange = RCC_MSIRANGE_6;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_MSI;
  RCC_OscInitStruct.PLL.PLLM = 1;
  RCC_OscInitStruct.PLL.PLLN = 16;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV7;
  RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
  RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_1) != HAL_OK)
  {
    Error_Handler();
  }
  PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_USART1;
  PeriphClkInit.Usart1ClockSelection = RCC_USART1CLKSOURCE_PCLK2;
  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
  {
    Error_Handler();
  }
  /** Configure the main internal regulator output voltage
  */
  if (HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1) !=
HAL_OK)
  {
    Error_Handler();
  }
  /** Enable MSI Auto calibration
  */
  HAL_RCCEx_EnableMSIPLLMode();
}

/**
  * @brief USART1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART1_UART_Init(void)
{

  /* USER CODE BEGIN USART1_Init 0 */

  /* USER CODE END USART1_Init 0 */

  /* USER CODE BEGIN USART1_Init 1 */

  /* USER CODE END USART1_Init 1 */
  huart1.Instance = USART1;
  huart1.Init.BaudRate = 115200;
  huart1.Init.WordLength = UART_WORDLENGTH_8B;
  huart1.Init.StopBits = UART_STOPBITS_1;
  huart1.Init.Parity = UART_PARITY_NONE;
  huart1.Init.Mode = UART_MODE_TX_RX;
  huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart1.Init.OverSampling = UART_OVERSAMPLING_16;
  huart1.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
  huart1.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
  if (HAL_UART_Init(&huart1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART1_Init 2 */

  /* USER CODE END USART1_Init 2 */
```

```
}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();

}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/* USER CODE BEGIN Header_StartDefaultTask */
/**
  * @brief  Function implementing the defaultTask thread.
  * @param  argument: Not used
  * @retval None
  */
/* USER CODE END Header_StartDefaultTask */
void StartDefaultTask(void *argument)
{
  /* USER CODE BEGIN 5 */
  /* Infinite loop */
  for(;;)
  {
    osDelay(1);
  }
  /* USER CODE END 5 */
}

/* USER CODE BEGIN Header_StartTask02 */
/**
* @brief Function implementing the Task1_Evaluate thread.
* @param argument: Not used
* @retval None
*/
```

```
/* USER CODE END Header_StartTask02 */
void StartTask02(void *argument)
{
//  /* USER CODE BEGIN StartTask02 */
        UBaseType_t temp;
        int first_num;
        int sec_num;
        int res;
        uint8_t my_char_receive;
//  /* Infinite loop */
  for(;;)
  {

                if(uxQueueSpacesAvailable(queueRQHandle)==0){
                        xQueueReceive(queueRQHandle, &my_char_receive, NULL);
                        first_num = my_char_receive - '0';
                        xQueueReceive(queueRQHandle, &my_char_receive, NULL);
                        if(my_char_receive== '+'){
                                xQueueReceive(queueRQHandle, &my_char_receive, NULL);
                                sec_num = my_char_receive - '0';
                                res = first_num + sec_num;
                        }else{
                                if(my_char_receive== '-'){
                                        xQueueReceive(queueRQHandle, &my_char_receive,
NULL);

                                        sec_num = my_char_receive - '0';
                                        res = first_num - sec_num;
                                }
                        }
                        xQueueReceive(queueRQHandle, &my_char_receive, NULL);
                        my_char_receive = res+ '0';
                        xQueueSend(queueSQHandle, &my_char_receive, NULL);
//                      taskENTER_CRITICAL();
//                      HAL_UART_Transmit(&huart1, &my_char_receive, 1,
HAL_MAX_DELAY);
//                      taskEXIT_CRITICAL();
                }
//              uint8_t my_char_receive;
//              xQueueReceive(queueRQ, &my_char_receive, NULL);
//              taskENTER_CRITICAL();
//
//              HAL_UART_Transmit(&huart1, &my_char_receive, 1, HAL_MAX_DELAY);
//
//              taskEXIT_CRITICAL();
```

```
    osDelay(1);
  }
//  /* USER CODE END StartTask02 */
}

/* USER CODE BEGIN Header_StartTask03 */
/**
* @brief Function implementing the Task2_UARTSend thread.
* @param argument: Not used
* @retval None
*/
/* USER CODE END Header_StartTask03 */
void StartTask03(void *argument)
{
  /* USER CODE BEGIN StartTask03 */
            uint8_t my_char_receive;
            uint8_t new_line;
            uint8_t carriage_return;
            carriage_return = 13;
            new_line = 10;

  /* Infinite loop */
  for(;;)
  {
                        if(uxQueueSpacesAvailable(queueSQHandle)==0){
                                    xQueueReceive(queueSQHandle, &my_char_receive, NULL);
                                    taskENTER_CRITICAL();
                                    HAL_UART_Transmit(&huart1, &my_char_receive, 1,
HAL_MAX_DELAY);
                                    HAL_UART_Transmit(&huart1, &carriage_return, 1, HAL_MAX_DELAY);
                                    HAL_UART_Transmit(&huart1, &new_line, 1, HAL_MAX_DELAY);
                                    taskEXIT_CRITICAL();
                        }
    osDelay(1);
  }
  /* USER CODE END StartTask03 */
}

/* USER CODE BEGIN Header_StartTask04 */
/**
* @brief Function implementing the Task3_UARTRecei thread.
* @param argument: Not used
* @retval None
*/
```

```
/* USER CODE END Header_StartTask04 */
void StartTask04(void *argument)
{
  /* USER CODE BEGIN StartTask04 */
        int temp;
        uint8_t my_char;
  /* Infinite loop */
  for(;;)
  {
                if(xSemaphoreTake(event_signal, 9999999)) {
                        temp=taskENTER_CRITICAL_FROM_ISR();
                        HAL_UART_Receive(&huart1, &my_char, 1, HAL_MAX_DELAY);
                        taskEXIT_CRITICAL_FROM_ISR(temp);
                        xQueueSendFromISR(queueRQHandle, &my_char, NULL);
                        //xSemaphoreGive(event_signal);
                }
    osDelay(1);
  }
  /* USER CODE END StartTask04 */
}

 /**
  * @brief  Period elapsed callback in non blocking mode
  * @note   This function is called  when TIM1 interrupt took place, inside
  * HAL_TIM_IRQHandler(). It makes a direct call to HAL_IncTick() to increment
  * a global variable "uwTick" used as application time base.
  * @param  htim : TIM handle
  * @retval None
  */
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
  /* USER CODE BEGIN Callback 0 */

  /* USER CODE END Callback 0 */
  if (htim->Instance == TIM1) {
    HAL_IncTick();
  }
  /* USER CODE BEGIN Callback 1 */

  /* USER CODE END Callback 1 */
}

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

stm3214xx_it.c

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file    stm32l4xx_it.c
  * @brief   Interrupt Service Routines.
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under Ultimate Liberty license
  * SLA0044, the "License"; You may not use this file except in compliance with
  * the License. You may obtain a copy of the License at:
  *                             www.st.com/SLA0044
  *
  ******************************************************************************
  */
/* USER CODE END Header */

/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "FreeRTOS.h"
#include "task.h"
#include "queue.h"
#include "cmsis_os2.h"
#include "semphr.h"

#include "stm32l4xx_it.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN TD */

/* USER CODE END TD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */

/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */

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

/* USER CODE END 0 */

/* External variables --------------------------------------------------------*/
extern UART_HandleTypeDef huart1;
extern TIM_HandleTypeDef htim1;
extern  QueueHandle_t queueRQHandle;
extern  QueueHandle_t queueSQHandle;
extern SemaphoreHandle_t event_signal;
/* USER CODE BEGIN EV */

/* USER CODE END EV */

/******************************************************************************/
/*           Cortex-M4 Processor Interruption and Exception Handlers          */
/******************************************************************************/
/**
  * @brief This function handles Non maskable interrupt.
  */
void NMI_Handler(void)
{
  /* USER CODE BEGIN NonMaskableInt_IRQn 0 */

  /* USER CODE END NonMaskableInt_IRQn 0 */
  /* USER CODE BEGIN NonMaskableInt_IRQn 1 */
  while (1)
  {
  }
  /* USER CODE END NonMaskableInt_IRQn 1 */
}

/**
  * @brief This function handles Hard fault interrupt.
  */
void HardFault_Handler(void)
{
  /* USER CODE BEGIN HardFault_IRQn 0 */

  /* USER CODE END HardFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_HardFault_IRQn 0 */
    /* USER CODE END W1_HardFault_IRQn 0 */
  }
}

/**
  * @brief This function handles Memory management fault.
  */
void MemManage_Handler(void)
{
  /* USER CODE BEGIN MemoryManagement_IRQn 0 */

  /* USER CODE END MemoryManagement_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_MemoryManagement_IRQn 0 */
    /* USER CODE END W1_MemoryManagement_IRQn 0 */
  }
}

/**
  * @brief This function handles Prefetch fault, memory access fault.
  */
void BusFault_Handler(void)
{
  /* USER CODE BEGIN BusFault_IRQn 0 */

  /* USER CODE END BusFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_BusFault_IRQn 0 */
    /* USER CODE END W1_BusFault_IRQn 0 */
  }
}

/**
  * @brief This function handles Undefined instruction or illegal state.
  */
void UsageFault_Handler(void)
{
  /* USER CODE BEGIN UsageFault_IRQn 0 */
  /* USER CODE END UsageFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_UsageFault_IRQn 0 */
    /* USER CODE END W1_UsageFault_IRQn 0 */
  }
}

/**
  * @brief This function handles Debug monitor.
  */
void DebugMon_Handler(void)
{
  /* USER CODE BEGIN DebugMonitor_IRQn 0 */

  /* USER CODE END DebugMonitor_IRQn 0 */
  /* USER CODE BEGIN DebugMonitor_IRQn 1 */

  /* USER CODE END DebugMonitor_IRQn 1 */
}

/******************************************************************************/
/* STM32L4xx Peripheral Interrupt Handlers                                    */
/* Add here the Interrupt Handlers for the used peripherals.                  */
/* For the available peripheral interrupt handler names,                      */
/* please refer to the startup file (startup_stm32l4xx.s).                    */
/******************************************************************************/

/**
  * @brief This function handles TIM1 update interrupt and TIM16 global interrupt.
  */
void TIM1_UP_TIM16_IRQHandler(void)
{
  /* USER CODE BEGIN TIM1_UP_TIM16_IRQn 0 */

  /* USER CODE END TIM1_UP_TIM16_IRQn 0 */
  HAL_TIM_IRQHandler(&htim1);
  /* USER CODE BEGIN TIM1_UP_TIM16_IRQn 1 */

  /* USER CODE END TIM1_UP_TIM16_IRQn 1 */
}

/**
  * @brief This function handles USART1 global interrupt.
  */
void USART1_IRQHandler(void)
{
  /* USER CODE BEGIN USART1_IRQn 0 */
        int temp;
        uint8_t my_char;
        xSemaphoreGiveFromISR(event_signal, NULL);
        //taskENTER_CRITICAL();
//      temp=taskENTER_CRITICAL_FROM_ISR();
//      HAL_UART_Receive(&huart1, &my_char, 1, HAL_MAX_DELAY);
//      taskEXIT_CRITICAL_FROM_ISR(temp);
//      //taskEXIT_CRITICAL();
//      xQueueSendFromISR(queueRQHandle, &my_char, NULL);


////////////////////////        uint8_t my_char_receive;
////////////////////////        xQueueReceiveFromISR(queueRQHandle, &my_char_receive, NULL);
////////////////////////        temp=taskENTER_CRITICAL_FROM_ISR();
////////////////////////        HAL_UART_Transmit(&huart1, &my_char_receive, 1,
HAL_MAX_DELAY);
////////////////////////        taskEXIT_CRITICAL_FROM_ISR(temp);

  /* USER CODE END USART1_IRQn 0 */
  HAL_UART_IRQHandler(&huart1);
  /* USER CODE BEGIN USART1_IRQn 1 */

  /* USER CODE END USART1_IRQn 1 */
}

/* USER CODE BEGIN 1 */

/* USER CODE END 1 */
/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```