KETTERING UNIVERSITY
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

CE-426-01

Programming in RTOS Environment II

Darek Konopka

May 20, 2021

Table of Contents

1. Objectives

- ❖ Use signals for inter-thread communication
- ❖ Use mutex to manage access to shared resource

2. Program Source Code

2.1 Traffic Light Control System using Threading

```
/*----------------------------------------------------------------------------

      Designers Guide to the Cortex-M Family
      CMSIS RTOS Signal Example

*----------------------------------------------------------------------------*/
#include "stm32f10x.h"
#include <cmsis_os.h>
#include "Board_LED.h"

void green_thread (void const *argument); // led_Thread1
void yellow_thread (void const *argument); // led_Thread2
void red_thread (void const * argument);

osThreadDef(green_thread, osPriorityNormal, 1, 0);
osThreadDef(yellow_thread, osPriorityNormal, 1, 0);
osThreadDef(red_thread, osPriorityNormal, 1, 0);

osThreadId green_led_ID1;
osThreadId yellow_led_ID2;
osThreadId red_led_ID3;

void delay (void)
{
unsigned int index;
const unsigned int count = 1000000;
      for(index =0;index<count;index++)
      {
            ;
      }
}
/*----------------------------------------------------------------------------
  Flash LED 1 when signaled by the yellow_thread
 *----------------------------------------------------------------------------*/
```

```
void green_thread (void const *argument)
{
	for (;;)
	{

		LED_On(2);
		osSignalSet  (green_led_ID1,0x01);
		osDelay(3000);
		LED_Off(2);
		osSignalSet  (green_led_ID1,0x02);
		osDelay(4000);

	}
}
/*----------------------------------------------------------------------------
  Flash LED two and synchronise the flashing of LED 1 by setting a signal flag
 *----------------------------------------------------------------------------*/
void yellow_thread (void const *argument)
{
	for (;;)
	{
		LED_Off(1);
		osSignalSet  (yellow_led_ID2,0x02);
		osDelay(3000);
		LED_On(1);
		osSignalSet  (yellow_led_ID2,0x01);
		osDelay(1000);
		LED_Off(1);
		osSignalSet  (yellow_led_ID2,0x02);
		osDelay(3000);
	}
}


/*----------------------------------------------------------------------------
  Flash LED 1 when signaled by the led_Thread2
 *----------------------------------------------------------------------------*/
void red_thread (void const *argument)
{
	for (;;)
	{
		LED_Off(0);
		osSignalSet  (red_led_ID3,0x02);
		osDelay(4000);
		LED_On(0);
		osSignalSet  (red_led_ID3,0x01);
		osDelay(3000);

	}
}
```

```
/*----------------------------------------------------------------------------
  Start the threads
 *----------------------------------------------------------------------------*/
int main (void)
{
      osKernelInitialize ();                    // initialize CMSIS-RTOS

      LED_Initialize();
      green_led_ID1 = osThreadCreate(osThread(red_thread), NULL);
      yellow_led_ID2 = osThreadCreate(osThread(green_thread), NULL);
      red_led_ID3 = osThreadCreate(osThread(yellow_thread), NULL);

      osKernelStart ();                         // start thread execution
}
```

2.2 Traffic Light Control System using Mutex for Messaging

```
/*----------------------------------------------------------------------------

    Designers Guide to the Cortex-M Family
    CMSIS RTOS Signal Example

*----------------------------------------------------------------------------*/
#include "stm32f10x.h"
#include <cmsis_os.h>
#include "Board_LED.h"
#include "uart.h"

// Prototype functions
void green_thread (void const *argument); // led_Thread1
void yellow_thread (void const *argument); // led_Thread2
void red_thread (void const * argument);
void SendText(uint8_t *txt);

// Calls this from uart.c
volatile uint8_t          inKey;

osThreadDef(green_thread, osPriorityNormal, 1, 0);
osThreadDef(yellow_thread, osPriorityNormal, 1, 0);
osThreadDef(red_thread, osPriorityNormal, 1, 0);

osThreadId green_led_ID1;
osThreadId yellow_led_ID2;
osThreadId red_led_ID3;

osMutexId uart_mutex;
osMutexDef(uart_mutex);

void delay (void)
```

```
{
unsigned int index;
const unsigned int count = 1000000;
    for(index =0;index<count;index++)
    {
        ;
    }
}

/*----------------------------------------------------------------------------
  Flash LED 1 when signaled by the yellow_thread
 *----------------------------------------------------------------------------*/
void green_thread (void const *argument)
{
    for (;;)
    {
        osMutexWait(uart_mutex, osWaitForever);
        SendText((unsigned  char*)"GREEN LED ON\n");
        osMutexRelease(uart_mutex);
        LED_On(2);
        osSignalSet    (green_led_ID1,0x01);
        osDelay(3000);
        LED_Off(2);
        osSignalSet    (green_led_ID1,0x02);
        osDelay(4000);

    }
}
/*----------------------------------------------------------------------------
  Flash LED two and synchronise the flashing of LED 1 by setting a signal flag
 *----------------------------------------------------------------------------*/
void yellow_thread (void const *argument)
{
    for (;;)
    {

        LED_Off(1);
        osSignalSet    (yellow_led_ID2,0x02);
        osDelay(3000);
        osMutexWait(uart_mutex, osWaitForever);
        SendText((unsigned  char*)"YELLOW LED ON\n");
        osMutexRelease(uart_mutex);
        LED_On(1);
        osSignalSet    (yellow_led_ID2,0x01);
        osDelay(1000);
        LED_Off(1);
        osSignalSet    (yellow_led_ID2,0x02);
        osDelay(3000);

    }
```

```
}


/*----------------------------------------------------------------------------
  Flash LED 1 when signaled by the led_Thread2
 *----------------------------------------------------------------------------*/
void red_thread (void const *argument)
{
    for (;;)
    {

        LED_Off(0);
        osSignalSet    (red_led_ID3,0x02);
        osDelay(4000);
        osMutexWait(uart_mutex, osWaitForever);
        SendText((unsigned  char*)"RED LED ON\n");
        osMutexRelease(uart_mutex);
        LED_On(0);
        osSignalSet    (red_led_ID3,0x01);
        osDelay(3000);


    }
}

/*----------------------------------------------------------------------------
  Start the threads
 *----------------------------------------------------------------------------*/
int main (void)
{
    osKernelInitialize ();                    // initialize CMSIS-RTOS

    LED_Initialize();
    USART1_Init ();
    uart_mutex = osMutexCreate(osMutex(uart_mutex));
    green_led_ID1 = osThreadCreate(osThread(red_thread), NULL);
    yellow_led_ID2 = osThreadCreate(osThread(green_thread), NULL);
    red_led_ID3 = osThreadCreate(osThread(yellow_thread), NULL);

    osKernelStart ();                         // start thread execution
}

//complete this function for sending a string of characters to the UART
void SendText(uint8_t *text) {

       uint8_t i=0;

       // every string end in \0
       while(text[i] != '\0') {
       SendChar(text[i]);
```

```
        i++;
        }
}
```

3. Questions

3.1 Question 1

Before uncommenting the osMutexWait calls, the code will randomly print 1's 2's and \n to the UART terminal. Once you uncomment the osMutexWait functions, then the code will then properly print 10 1's, skip a line, then 10 2's then skip a line, to the UART terminal. These results can be seen in figures 2 and 3 in the Outputs section of this document.

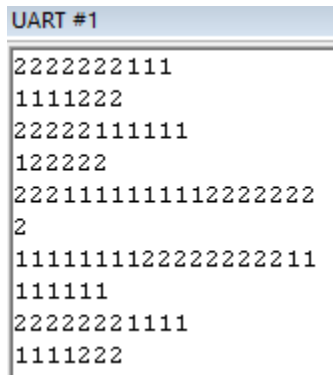

Figure 1 EX 15 with commented osMutexWait

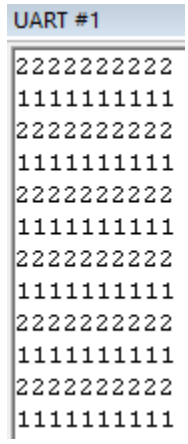

Figure 2 EX 15 with uncommented osMutexWait

3.2 Question 2

The message results are the same whether you have the osMutexWait functions commented out or not. A picture of the output can be found in figure 4 under the outputs. However, once you comment out UART_Init(); then the code will not work at all (the logic analyzer of this situation can be seen in figure 5 under the outputs sections).

4. Outputs

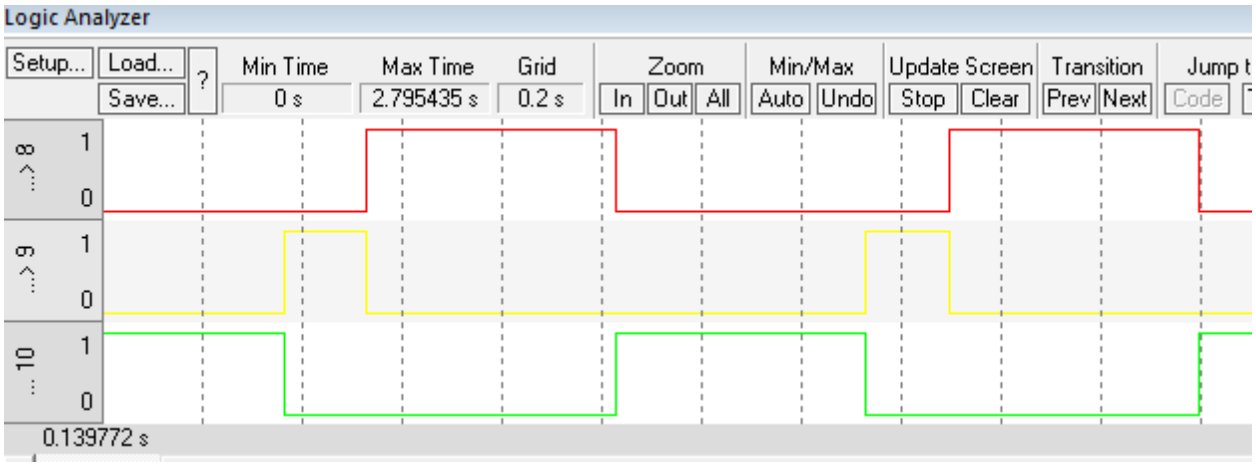

Figure 3 Traffic Light Control using Threads (EX8)

Figure 4 Traffic Light Control Messages

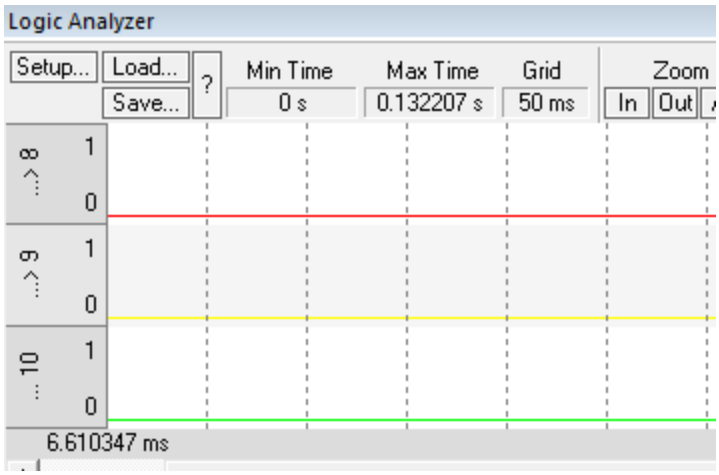

Figure 5 Traffic Light Control Messages when UART_Init is commented out

5. Conclusions

In this assignment we used a Mutex to help the program send messages to the UART terminal while sending simultaneous threads. This lab gives a great example as to why you need a mutex for shared resources when working with inter-thread communication.