CG2271 Real Time Operating system AY 2017/18

Lab 5 –Synchronization and Communication (10% of your final grade)

Lab Lessons: Wednesday 1 Nov & 8 Nov
IVLE Submission of answer book: Friday 10 Nov midnight
Demo: Wednesday 15 Nov

Introduction

In this lab, you will explore synchronization and communication in FreeRTOS. **This lab is worth 30 marks for the answer book and 10 marks for the demo for a total of 40 marks.**

In this lab, you will continue to use the same circuit from Lab 2: two push-buttons connected to pins 2 and 3, two LEDs connected to pins 6 and 7, the potentiometer and the touch sensor connected to analog inputs A0 and A1.

Follow the instructions in Lab 4 to set up FreeRTOS. Additionally, you need to enable the following semaphore options in the configuration file FreeRTOS.h of FreeRTOS and recompile FreeRTOS (with the application) for this lab:

```
#ifndef configUSE_MUTEXES
      #define configUSE_MUTEXES 1
#endif

#ifndef configUSE_COUNTING_SEMAPHORES
      #define configUSE_COUNTING_SEMAPHORES 1
#endif
```

1. Why do we need Synchronization?

```
#include <Arduino.h>
#include <avr/io.h>
#include <FreeRTOS.h>
#include <task.h>

#define STACK_SIZE    200

void task1and2(void *p) {
      while (1) {
            int taskNum = (int) p;
            Serial.print("Task ");
            Serial.println(taskNum);
            vTaskDelay(1);
      }
}

void setup() {
      Serial.begin(115200);
}

void loop() {
      /* create two tasks one with higher priority than the other */
      xTaskCreate(task1and2, "Task1", STACK_SIZE, (void * ) 1, 1, NULL);
      xTaskCreate(task1and2, "Task2", STACK_SIZE, (void * ) 2, 2, NULL);
      /* start scheduler */
      vTaskStartScheduler();
}
```

Question 1 [4 marks]

(A) Task 1 and Task 2 share exactly the same code. That is, both the tasks are created using the same function "`task1and2( )`". Explain why it is possible to create two different tasks using the same function. **[1 mark]**

(B) By inspecting the code, describe what you EXPECT this program to write to the serial port. **[1 mark]**

(C) Launch the Serial Monitor in the Arduino IDE (or use any other terminal program that can monitor the serial port) to monitor the program's output. Observe that the program generates garbled output. Explain why the output is garbled. (Hint: The serial port communication takes time.)

[2 marks]

2. Synchronization with Mutex Semaphores

Now you will use FreeRTOS mutex semaphores to implement the functionality of the code fragment in Question 1 correctly (i.e., Task1 and Task2 will continuously output "Task 1" and "Task 2", respectively, instead of garbled output).

Question 2 [6 marks]

Read Section 8.3 of Chapter 8 "Interprocess Communication Primitives in FreeRTOS" from the book "Real-Time Embedded Systems: Open-Source Operating Systems Perspective". The chapter is available under IVLE Files → Reading

Also read http://www.freertos.org/Embedded-RTOS-Binary-Semaphores.html

http://www.freertos.org/Real-time-embedded-RTOS-Counting-Semaphores.html

http://www.freertos.org/Real-time-embedded-RTOS-mutexes.html

(A) Explain the difference between binary semaphore and mutex semaphore. **[2 marks]**

(B) Create a properly synchronized version of Question 1 using semaphore. Cut and paste your code in the answer sheet. **[4 marks]**

3. Synchronization with Message Queues

Now you will use FreeRTOS message queues (instead of mutex semaphores) to implement the functionality of the code fragment in Question 1 correctly (i.e., Task1 and Task2 will continuously output "Task 1" and "Task 2", respectively, instead of gabled output).

Question 3 [6 marks]

Read Section 8.2 of Chapter 8 "Interprocess Communication Primitives in FreeRTOS" from the book "Real-Time Embedded Systems: Open-Source Operating Systems Perspective". The chapter is available under IVLE Files → Reading

Also read http://www.freertos.org/Embedded-RTOS-Queues.html

To implement the correct functionality using message queues, you will remove the serial printing functions from `task1and2( )` and instead create a dedicated task `serialPrint()` to print on the serial port. The modified `task1and2( )` will simply send the value to be printed through a message queue to the `serialPrint( )` task, wait for 1ms, and then repeat itself. The `serialPrint( )` task will receive the value from the message queue and print it on the serial port. The templates are given in the next page.

```
void serialPrint(void *p) {
      while (1) {
            /*receive taskNum from message queue */
            Serial.print("Task ");
            Serial.println(taskNum);
      }
}

void task1and2(void *p) {
      while (1) {
            int taskNum = (int) p;
            /* send taskNum to message queue */
            vTaskDelay(1);
      }
}
```

(A) Fill in the message queue related code (highlighted in yellow) such that the program works properly. Cut and paste your code in the answer booklet. You have to choose `xTicksToWait` value carefully for sender and receiver tasks for the program to work correctly. **[4 marks]**

(B) You will observe that after running the program for a while, "Task2" is printed more frequently on the screen. Explain why this happens. [Hint: Think about what happens when a slot becomes available in a full message queue.] **[2 marks]**

4. Synchronization between ISR and Tasks

Question 4 (7 marks)

You will implement the following functionality that was earlier implemented in Lab 3 using Function Queue Scheduling. Now you have to implement the same functionality using **binary semaphores to synchronize between the ISR and the corresponding task** as well as rely on **FreeRTOS priority-driven scheduling.**

- Pressing (or releasing depending on how you implement it) the push button at INT0 pin flashes (on and off) the LED at digital pin 7 five times at 4 Hz. The LED flashing function is implemented in the **task (and not function)** `int0task()`
- Pressing (or releasing depending on how you implement it) the push button at INT1 pin flashes (on and off) the LED at digital pin 6 five times at 2 Hz. The LED flashing function is implemented in the **task (and not function)** `int1task()`
- `int0task()` task has higher priority over `int1task()`

You need to write two interrupt service routines corresponding to the two push buttons. Please do not forget to handle switch de-bouncing. The interrupt service routines should wake up the corresponding tasks through binary semaphores.

(A) Why do you need a special primitive `xSemaphoreGiveFromISR ( )` to perform V( ) operation on a semaphore from within an interrupt handler? **[1 mark]**

(B) Explain the need for `*pxHigherPriorityTaskWoken` parameter in the `xSemaphoreGiveFromISR ( )` function and how the parameter is used. **[1 mark]**

(C) Read the sample code using `xSemaphoreGiveFromISR( )` from http://www.freertos.org/a00124.html
Now implement the functionality of two LEDs blinking on button press using binary semaphore synchronization and priority in FreeRTOS. Please note that `portYIELD_FROM_ISR( )` routine is platform specific. You need to replace it with `taskYIELD()` routine. Cut and paste your code in the answer booklet. **[5 marks]**

5. Producer-Consumer Problem with Counting Semaphores

Question 5 (7 marks)

In this problem, you will implement producer-consumer problem with a **circular buffer of 4 entries** protected by mutex and counting semaphores as shown in slide 49 of the lecture slides on Synchronization.

The producer task is woken up by the ISR every time INT0 button is pressed (use binary semaphore as you have done in Question 4). Once the producer task is woken, it reads the potentiometer value and puts it in a circular buffer. The consumer task is a periodic task with a period of 5 sec; each task instance reads a value from the circular buffer and prints the value on the serial port. Both the producer and the consumer task have the same priority.

(A) Cut and paste your code in the answer sheet. **[5 marks]**

(B) Press the button roughly once every 5 sec or so and turn the potentiometer towards the left by a little every time you press the button. What do you observe about the values printed on the serial port? Explain your observation. **[1 mark]**

(C) Now press the button once every 2 sec or so and turn the potentiometer towards the left by a little every time you press the button. What do you observe about the values printed on the serial port? Explain your observation. **[1 mark]**

6. Demo Session

Submit the Lab 5 answer book to IVLE Files → Lab5-Submissions → Your lab group. Please ensure that you submit to the correct Lab group. You should make one submission per team.

You will demonstrate Questions 3, 4, 5 to your lab TA on 15 Nov. Please be punctual for your demonstration. Your demonstration is worth 10 marks.