CG2271 Real Time Operating system AY 2017

Lab 3 – Real-Time Software Architectures (8% of your final grade)

Lab Lessons: Wednesday 4 and 11 Oct
IVLE Submission of answer book: Friday 13 Oct 2359
Demo: Wednesday 25 Oct

1. Introduction

In this lab, you will explore interrupts and function queue scheduling. **This lab is worth 30 marks for the answer book, 5 marks for the demo and 5 marks for Q&A for a total of 40 marks.**

For this lab we will continue to use the same circuit from Lab 2, with two push-buttons connected to pins 2 and 3, and two LEDs connected to pins 6 and 7. The potentiometer and touch sensors are not required for this lab.

2. Function Pointers

Before we begin, we will explore the use of function pointers. Conventionally you specify a function to call at the time you write your program, and once your program is compiled, you cannot change which function is called. For example:

```
int f1(int x, int y)
{
       return x+y;
}

…
int main()
{
    int z=f1(3, 4);
       …
}
```

When this program is executed, f1 is called. There is no way to change this once the program is compiled.

A function pointer is a variable that points to a function. In the example above, we can create a function pointer using the following syntax:

```
int (*funcptr)(int, int);
```

This creates a variable called “funcptr” that points to a function that accepts two integers and returns one integer.

In the example below we have two functions fun1 and fun2, and we generate a random number between 0 and 1, and call fun1 50% of the time and fun2 50% of the time. You can create a new project and type in the following code. After uploading you can use a serial terminal to view the results from the program from the serial port that your Arduino is connected to. Remember to set the baud rate to 115200. You will see that the program calls fun1 (returning a value of 2+3=5) and fun2 (returning a value of 2*3=6) randomly.

```
#include <Arduino.h>
#include <stdlib.h>
#include <limits.h>

int fun1(int x, int y)
{
	return x+y;
}

int fun2(int x, int y)
{
	return x*y;
}

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

// Declare the function pointer
int (*funcptr)(int, int);

void loop()
{
	float turn=(float) rand() / INT_MAX;
	int result;

	if(turn>0.5)
		funcptr=fun1;
	else
		funcptr=fun2;


	// Invoke the function
	result=funcptr(2,3);

	Serial.print("Computation result:");
	Serial.println(result);

	//200ms pause
	delay(200);
}
```

3. Priority Queue

NOTE: You can read the tutorial about priority queue (IVLE: Lab 3_PQ.pdf) first before you start this section.

We will now implement function queue scheduling. To support function queue scheduling, we need a priority queue. In the Lab3.zip file you will find two files prioq.cpp and prioq.h. Both of these have been written as "generic" queue functions in that they accept arguments of type "void *". You need to read and understand the function definitions in the header file and the code in the .cpp file.

Unzip prioq.cpp and prioq.h somewhere on your disk. Add in the prioq.cpp and prioq.h files into your project workspace in eclipse, then key in the following code into the cg2271lab3part1.cpp file:

```
#include <Arduino.h>
#include "prioq.h"

#define QLEN 10

TPrioQueue *queue;

void setup()
{
	// Set up the queue.
	queue=makeQueue();

	// Initialize the serial port
	Serial.begin(115200);

	// Enqueue 10 numbers
	for(int i=0; i<QLEN; i++)
		enq(queue, (void *)i, QLEN-i-1);
}



void loop()
{
	int val;

	// If we still have an item to dequeue
	if(qlen(queue)>0)
	{
		// Dequeue it
		val=(int) deq(queue);

		// And print it on the serial port.
		Serial.println(val);
	}


	//500ms pause
	delay(500);
}
```

This code demonstrates how to use the prioq functions. Setup enqueues 10 numbers from 0 to 9, while loop dequeues one number at a time and prints it onto the serial port. Open the serial port that your Arduino is connected to, ensuring that you are using 115200 bps. Note: You may have to reset the Arduino. Locate the small pushbutton on the Arduino board labelled RESET and press it. Then answer the following questions:

Question 1 (1 marks)

Describe what you see being output to the terminal program.

Question 2 (3 marks)

Do the numbers appear in the same order that they were enqueued? Why or why not? (Hint: Look at the comments in the prioq.h file to see what the parameters to enq function mean.)

Hint for Questions 3 and 4: Look up "typecasting in C".

Question 3 (2 marks)

Within the setup function we have this call to enq:

```
enq(queue, (void *)i, QLEN-i-1);
```

Why is there a need for the (void *) highlighted in this statement? What does the (void *) mean?

Question 4 (2 marks)

Similarly there is an (int) statement in the deq (highlighted). What does this (int) mean? Why is it necessary?

```
val=(int) deq(queue);
```

4. Function Queue Scheduling

We will now use prioq.cpp and prioq.h to build a function queue scheduling system. Comment out cg2271lab3part1.cpp. Add in the skeleton code called "cg2271lab3part2.cpp" from the Lab3.zip file.

Question 5 (12 marks)

Complete the skeleton program "cg2271lab3part2.cpp" to implement the following functionality:

- Pressing (or releasing depending on how you implement it) the push button at INT0 pin flashes (on and off) the LED at **digital pin 6** five times at 5 Hz (on for 0.1 sec and off for 0.1 sec). The LED flashing function is implemented in 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 7** five times at 1 Hz. The LED flashing function is implemented in function `int1task()`
- `int1task()` function has higher priority over `int0task()`

The functionality should be implemented using a function queue scheduling system**. You are NOT ALLOWED to modify prioq.cpp or prioq.h in any way.**

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 enqueue the corresponding functions in the priority queue with appropriate priority.

The main loop should dequeue functions from the priority queue and execute them.

Cut and paste your code in your answer book.

Question 6 (2 marks)

In the priority queue implementation, the enq function accepts items of type (void *) and the deq function returns items of type (void *). Explain how you can enqueue and dequeue functions in the priority queue using enq and deq.

Question 7 (2 marks)

Press the button connected to INT1 then immediately press the button connected to INT0. Describe the sequence in which the LEDs flash. Does the LED connected to pin 6 or pin 7 flash first? Explain you observation.

Question 8 (2 marks)

Press the button connected to INT0 ONCE, and immediately press the button connected to INT1 ONCE. Describe which LED flashes first. The one connected to pin 6, or to pin 7? Explain your observation.

Question 9 (4 marks)

Alternately press the button connected to INT1, then INT0, five times. That is:

Press INT1 button
Press INT0 button
Press INT1 button
Press INT0 button
…

Describe the sequence of LED flashes. Does the LED connected to pin 6 almost always flash before the LED connected to pin 7? Explain your observations.

6. Demo Session

Submit the Lab 3 answer book to IVLE Files → Lab3-Submissions. Please ensure that you submit your file as GXX.docx or GXX.pdf (XX is your group number). You should make one submission per team.

You will demonstrate Question 1 and Question 5 to your lab TA at the start of your lab session for Lab 4. Please be punctual for your demonstration. Your demonstration and Q&A is worth a total of 10 marks.