CG2271 Real Time Operating Systems
Lab 3 - Real Time Architectures II

1. Introduction

In the previous lab you explored using two of the simplest but most common real-time architectures: round-robin and round-robin with interrupts.

In this lab we will explore the two somewhat more sophisticated methods; function queue scheduling and timed-loops.

To complete this lab ensure that you have compiled the Arduino libraries for AVR/GNU C++ as detailed in Lab 2.

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.

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 two integers.

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 in Atmel Studio 6 and type in the following code. Remember to set up the include directories and bring in the libarduinocpp library you created last lab. 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 9600. 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.

(Consult http://arduinobasics.blogspot.sg/search/label/Serial%20Monitor to see how to use the Arduino Serial Monitor)

```
#include <avr/io.h>
#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(9600);
}
// 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);
}
// Main has been written to work correctly with Arduino. Do not modify.
int main(void)
{
	init();
	setup();
    while(1)
    {
		loop();
		if(serialEventRun)
			serialEventRun();
    }
}
```

(Yes, there are much better ways to write this program, but this is just an example of how to declare and use function pointers. ☺)

3. Function Queue Scheduling

We will now implement function queue scheduling. In the cg2271lab3.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 *".

Unzip prioq.cpp and prioq.h somewhere on your disk.

Create a new AVR/GNU C++ executable project called cg2271lab3part1a, select to target the Atmega328P, set up all the usual include directories and set up your project to use the libarduinocpp library you compiled during Lab 2.

Add in the prioq.cpp and prioq.h files into the cg2271lab3part1a project, then key in the following code into the cg2271lab3part1a.cpp file:

```
/*
 * cg2271lab3part1a.cpp
 *
 * Created: 12/9/2013 8:39:35 AM
 *  Author: dcstanc
 */


#include <avr/io.h>
#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(9600);

	// Now enqueue 10 numbers in reverse priority
	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);
}

// Main has been set up to work correctly with the Arduino libraries.
// Do not modify.
int main(void)
{
        init();
        setup();

    while(1)
    {
              loop();
              if(serialEventRun)
                    serialEventRun();
    }
}
```

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. Launch a terminal program like puTTY or the Arduino Serial Monitor, open the serial port that your Arduino is connected to, ensuring that you are using 9600 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 (2 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 mean.)

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

Question 3 (4 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 (4 marks)

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

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

We will now use prioq.cpp and prioq.h to build a function queue scheduling system. Create a new project called cg2271lab3part2, and add prioq.h and prioq.cpp to your project. Delete the cg2271lab3part2.cpp file from the project, and add in the skeleton code called "cg2271lab3part2.cpp" in the cg2271lab3.zip file.

Now implement a function queue scheduling system where INT0 has higher priority over INT1. **You are NOT ALLOWED to modify prioq.cpp or prioq.h in any way.**

There is a function called "debounce" in the skeleton code provided. Google for the term "switch debounce" and answer the following questions:

Question 5 (3 marks)

What is switch bouncing? Why is switch bouncing a problem to electronic systems?

Question 6 (6 marks)

Study the function called "debounce". Explain how it solves the bouncing problem.

Question 7 (5 marks)

The enq function accepts items of type (void *) and the deq function returns items of type (void *). Explain how you enqueue and dequeue functions using enq and deq.

Question 8 (7 marks)

Explain, with suitable code snippets from your completed completed cg2271lab3part2 how you have implemented function queue scheduling. In particular explain how you solve the switch bouncing problem.

Question 9 (3 marks)

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

Question 10 (3 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 your observation.

Question 11(5 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.

4. Timed Loops

We will now look at performing periodic tasks using timed loops. Create a new project called cg2271lab3part3, delete the cg2271lab3part3.cpp file, and add in the cg2271lab3part3.cpp skeleton provided in cg2271lab3.zip.

Complete the skeleton using timed-loops only, to flash the LED connected at pin 6 five times a second (goes on and off 5 times a second) and the LED at pin 7 once per second (goes on and off once a second).

You are not allowed to use delay() or any such calls.

Question 12 (5 marks)

Cut and paste important snippets of your completed code to explain how you implemented timed loops.