Verónica Gaspes
School of IDE

Embedded Systems Programming

Written Exam

January 13th, 2012, from 09.00 to 13.00

- **Allowed tools:** An English dictionary (a paper such, not an electronic one).
- **Grading criteria:** You can get at most 20 points.

  To pass you need at least 50% of the points.

  For the highest grade you need more than 90% of the points.
- **Responsible:** Verónica Gaspes. Telephone number 7380.

- **Read carefully!** Some exercises might include explanations, hints and/or some code. What you have to do in each exercise is marked with the points that you can get for solving it (as **(X pts.)**).
- **Write clearly!**
- **Motivate your answers!**

Good Luck!

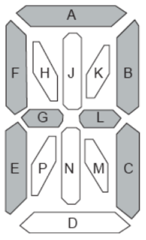

Figure 1: Digit segments in the LCD

Register Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LCDDRx	K	–	–	A	K	-	-	A
LCDDRx+5	J	F	H	B	J	F	H	B
LCDDRx+10	L	E	G	C	L	E	G	C
LCDDRx+15	M	P	N	D	M	P	N	D

Figure 2: LCD data registers

1. In the first three laborations of this course we used a demonstration platform that includes an LCD. The LCD is controlled via the registers `LCDDR0` to `LCDDR19`, each of them one byte wide. Figure 1 shows what items can be turned on/off in each digit while Figure 2 shows what bits of the registers are used to control each of the items. In this exercise we are interested in the fact that four registers are used to control 2 positions.

   (a) **(2 pts.)** Write a code fragment that shows character A (the pattern shown with dark items in Figure 1) in the position controlled by the higher nibbles of the registers `LCDDR0`, `LCDDR5`, `LCDDR10` and `LCDDR15`.

   (b) **(1 pts.)** Explain how you managed to make sure that only the segments for the A appear in the right position and how you managed to make sure that the rest of the LCD is not modified by your code fragment.

2. (a) **(1 pts.)** What does *busy-waiting* mean?

   (b) **(1 pts.)** Explain what you used busy-waiting for in laboration 1.

   (c) **(2 pts.)** In laboration 1 you programmed the Butterfly board to print prime numbers on the LCD, to blink regularly with a segment of the LCD and to toggle an LCD segment using the joystick. In the last exercise you had to put all these parts together. Explain what challenges you faced and how you dealt with them given that we were not using any support for threads.

3. Consider the embedded system illustrated in Figure 3 taken from the course book. In this simple embedded system T takes readings from a set of ther-

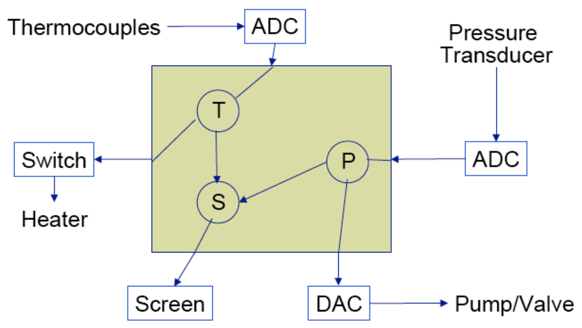

Figure 3: A simple embedded system.

mocouples (via an ADC) and makes appropriate changes to a heater (via a digitally controlled switch). P is doing a similar thing but for pressure. P and T communicate data to S which presents measurement data to an operator via a screen. The overall purpose of the system is to keep the temperature and pressure of some chemical process within defined limits. Someone has already written functions for some of the parts of the program:

```
int readTemperature(){
   while(! TEMP_READY );
   return TEMP_VALUE;
}
int readPressure(){
   while(! PRESSURE_READY );
   return PRESSURE_VALUE;
}
void writeToSwitch(int v){SWITCH_VAL = v;}

void writeToPump(int v){PUMP_VAL = v;}

void temperatureConvert(int temp, int* switchValue){
  // A function that calculates the value to write to
  // the switch given a temperature measurement.
  // You can assume it is already implemented!
}
void pressureConvert(int press, int* pumpValue){
  // A function that calculates the value to write to
  // the pump given a pressure measurement.
  // You can assume it is already implemented!
}
```

You can print values on the screen using `printf`.

(a) **(3 pts.)** Write a program for the complete embedded system using the kernel `Tinythreads` that we used in laboration 2. The kernel provides an operation

```
void spawn(void (*code)(int), int arg);
```

to start a thread that executes `code(arg)`. The kernel then does time-slicing, alternating between the threads giving them equally long time slices.

(b) **(1 pts.)** What can you say about your solution when the temperature values are available at relatively long intervals?

4. Using TinyTimber you can organize programs with *reactive objects* while programming in C. As a programmer you have to follow some conventions and TinyTimber guarantees that the methods of a reactive object are executed strictly sequentially, thus protecting the local state of the object from critical section problems. Here is a reactive object implementing a counter:

```
typedef struct {
  Object super;
  int value;
} Counter;

#define initCounter(){initObject(),0}

int inc(Counter * self, int x){
  self -> value++;
}

int reset(Counter * self, int x){
  self -> value = 0;
}

int readValue(Counter * self, int x){
  return self -> value;
}
```

Imagine a program where several objects share the same counter, for example

```
#include "TinyTimber.h"
#include "counter.h"
#include "o1.h"
#include "o2.h"
#include "lcd.h"

Counter c  = initCounter();
Lcd lcd    = initLcd();
O1 client1 = initO1(&c);
O1 client2 = initO1(&c);
O2 reader  = initO2(&c, &lcd);

INTERRUPT(...)
INTERRUPT(...)
STARTUP(CONFLCD;ASYNC(&c,reset,0);...)
```

```
typedef struct {
  Object super;
  Counter *cnt;
} O1;
...

typedef struct {
  Object super;
  Counter *cnt;
  Lcd *lcd;
} O2;
...
```

(a) **(1 pts.)** In the class `O1` a method needs to increment the counter. Say how this should be done and motivate your answer.

(b) **(1 pts.)** In the class `O2` a method needs to read the value of the counter. Show how this should be done and motivate your answer.

(c) **(1 pts.)** What are the problems of using `inc(self->cnt,0)` in a method of `O1`?

5. In this exercise you will write a reactive object for controlling a moving object. The device drivers for controlling the movement by giving the object a speed (speed 0 when the object is at rest) and for reading a distance using a sensor are already programmed as reactive objects:

```
typedef struct {
  Object super;
} Mover;

#define initMover(){initObject()}

int setSpeed(Mover * self, int speed){
  // sets the speed of the device
}
```

```
typedef struct {
  Object super;
} DistanceSensor;

#define initDistanceSensor(){initObject()}

int getDistance(DistanceSensor * self, int x){
  // reads a value from the
  // register associated to the sensor
  // and converts it into centimeters.
  // It returns this value
}
```

(4 pts.) Program a reactive object that controls the moving object so that it moves forward with a speed of `SPEED`. If an obstacle is detected closer to `MINDIST`cm the object should stop. If the obstacle is removed the object should continue moving forward. For the given speed and distance to obstacle it is enough to check the value in the sensor every `POLLING_TIME` milliseconds.

Note: You do not need to define the device drivers `Mover` and `DistanceSensor`! Just use the interfaces we provided when programming your controller.

6. **(2 pts.)** In the last laboration of the course you implemented an Android app. Explain how you organized the app. You need to explain what components you used, how they exchange data and why you organized it as you did.