Verónica Gaspes
School of IDE

Embedded Systems Programming

Written Exam

August 16th, 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: 167380.

- **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!

1. (a) **(2 pts.)** Write functions

```
void set(unsigned int *port, unsigned int mask)
void clear(unsigned int *port, unsigned int mask)
```

   to set resp. clear some bits in a port. The bits that have to be set or made clear in the port are given by the bits in the second argument (`mask`) that are set.

   (b) **(1 pts.)** Show how to use the function `set` to set the bits in positions 0, 2 and 4 of the port `PORT`.

2. Consider the following fragments we discussed in one of the lectures. The first two functions use busy waiting to detect and read input values from a sonar and a radio device. We do not show the code for the functions that implement ordinary algorithms (`control` and `decode`) and for generating output to the servo device.

```
int sonar_read(){
   while(SONAR_STATUS & READY == 0);
   return SONAR_DATA;
}
void radio_read(struct Packet *pkt){
  while(RADIO_STATUS & READY == 0);
  pkt->v1 = RADIO_DATA1;
  ...
  pkt->vn = RADIO_DATAn;
}
main(){
   struct Params params;
   struct Packet packet;
   int dist, signal;
   while(1){
     dist = sonar_read();
     control(dist, &signal, &params);
     servo_write(signal);

     radio_read(&packet);
     decode(&packet,&params);
   }
}
```

   (a) **(1 pts.)** Discuss some of the problems with this style of programming.

   (b) **(2 pts.)** Show some way of modifying the program so that one of the input sources does not shadow the other one.

(c) **(2 pts.)** Explain how you would need to modify the program in case the decoding algorithm takes too much time compared to the frequency at which sonar echoes are produced.

3. In laboration 2 we programmed with a kernel (tinythreads) that supports threads. Using the kernel function

```
void spawn(void (* function)(int), int argument)
```

a program can start a new thread to execute a call to a `function` with an integer `argument`. The different threads are interleaved automatically by the kernel that calls `yield()` at regular intervals (we call this time slicing). With small enough intervals the program seems to be doing several things at the same time (concurrently).

Imagine now a C program that uses this kernel and a function `f` that might be called from several concurrent threads. The function `f` both reads from and writes to the following types of variables:

(a) its arguments,

(b) global variables declared at top level of the program,

(c) local variables declared within `f`,

(2 pts.) Indicate **for each of these cases** whether access to the variables in question may constitute a critical section, that has to be protected by some mutual exclusion mechanism.

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.

(4 pts.) Program a class for reactive counters that can be incremented, reset, read and report how many times it has been reset. In other words, your class should implement the methods

```
// reset the counter to 0
int reset(Counter *self, int x)

// increment the counter by 1
int inc(Counter *self, int x)

// return the value of the counter
```

```
int read(Counter *self, int x)

// return the number of times the counter has been reset
int nrResets(Counter *self, int x)
```

5. Assume there is a Tinytimber class `PIEZO` for reactive objects implementing device drivers for piezo elements. These reactive objects provide the methods

   ```
   int on(PIEZO * self, int x)
   int off(PIEZO * self, int x)
   ```

   (4 pts.) Program a class for reactive objects that use a piezo element to produce a sound. Your class should allow for setting the frequency and the duration of the sound as well as playing the sound. A positive duration indicates how long the sound should be played. A duration of 0 indicates that it should be played forever.

6. In Android an app is organized using Activities, Services, ContentProviders, Notifications and other components.
   (2 pts.) Explain what Activities and Services are used for and why you might need to use threads in both Activities and Services.