

Exam — Concurrent and Real-Time Programming — English
2004-12-13, kl 14.00-19.00

Instructions

You are allowed to use the Java quick reference. To pass the exam, grade 3, you have to solve most of the first 8 problems (which are more of a theoretical type), and you have to develop an acceptable solution to problem 8 (which means programming). To get a higher grade, you also have to provide a sufficient solution to problem 9 (which means design). To get the highest grade you also have to have a sufficiently high total score (typically 30p). Results will be published on the internal part of the course web page (id=eda040, pwd=ht04).

Problems

1. The class `MutexSem` implements the interface `Semaphore` meaning that the methods `take` and `give` are implemented, which can be used for mutual exclusion. Another class that implements the same interface is `CountingSem`, which implements a counting semaphore in its basic form. Mention one (out of two explained in the course) reasons for making a difference between these two types of objects. (1p)

2. What is the reason for using priorities in a real-time system? (Max two sentences.) (1p)

3. We have discussed a concept called *static (cyclic) scheduling*.

a) Explain what static scheduling means. (1p)

b) Give one advantage and one disadvantage to using static scheduling instead of other scheduling strategies. (1p)

4. A real-time system is implemented using four threads denoted A, B, and C with the characteristics found in the table below (C = maximum execution time/period, T = period):

<i>Tråd</i>	<i>C (ms)</i>	<i>T (ms)</i>
A	3	20
B	4	15
C	3	6

The deadline for the threads are equal to the period of the threads. We also assume that the threads is scheduled according to the principle RMS (Rate Monotonic Scheduling) and that the threads are completely independent of each other, i.e. they do not communicate with each other and can thus not block each other (they can of course still preempt each other). Furthermore, we ignore the cost of context switches, etc.

a) What will the worst-case execution time (WCET) be for each of the three threads? (3p)

b) The system above is not schedulable using RMS. Suggest an alternative scheduling strategy that makes the system schedulable. (1p)

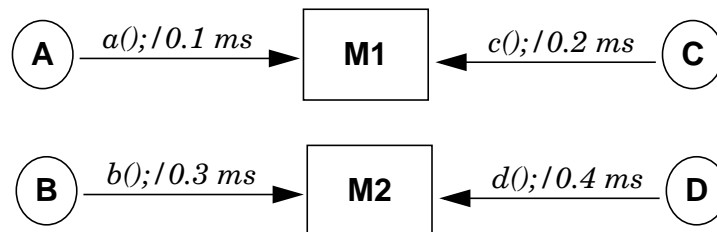
c) Show that the scheduling strategy you suggested in the previous question makes the system schedulable. (1p)

5. In order to perform scheduling analysis we must, among other things, have knowledge of the worst-case execution times for the various threads in the system. There are two main approaches to determining the worst-case execution time for a thread.

a) What are the two main approaches to determining the worst-case execution time for a thread? (1p)

b) Give a shortcoming of each of the two approaches in the previous subquestion. (1p)

6. A real-time system is implemented using four threads denoted A, B, C, and D. Thread A has the highest priority, B the next highest priority, and D have the lowest priority. Thread A communicates with thread C via a monitor called M1. Each time A and C respectively is executed, they call a monitor operation (once and only once) according to the following figure. Thread B and D communicate with each other in the same way via the monitor M2. The system is scheduled according to strict priority-based scheduling with *dynamic priority inheritance* (basic priority inheritance protocol).



What will the maximum blocking time for each of the four threads A, B, C, and D be? I.e., we are asking for the longest time each of the four threads can be blocked by *lower priority threads*. The maximum execution time for each of the operations is shown in the figure. (4p)

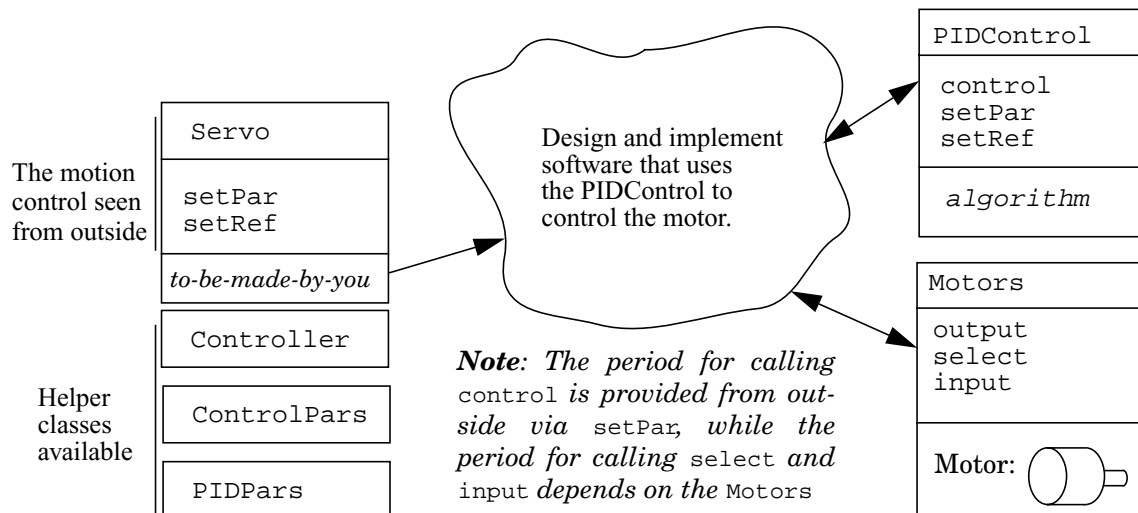
7. Show by writing a program how you by implementing a *monitor* (using synchronized methods but not using semaphores or messages) can implement rendez-vous between two threads. Assume that one of the threads calls a monitor operation `accept`, and the other thread calls the monitor method `entry`.

Hint: You may assume that only two threads are involved and that the data transfer is done directly in one of the methods `accept` or `entry` (you select one of the, and you do *not* need to make a generalized solution, such as a base-class for the rendez-vous and a sub-class for the specific data transfer, so you can simply implement everything is one single class). (3p)

8. Motor control

In a robot control system a set of joint angles are read by (as implemented within the available class `Motors`) writing the joint number (e.g. 1 for the first joint of the robot) to an output port, and a certain time thereafter the joint angle is read from an input port. Here we assume the angle is expressed in radians as a floating point number. The angle (of the motor shaft compared to some defined zero angle) is by the hardware defined within one motor turn, but due to a gearbox reduction the motor has to turn many times to cover the working range of the robot, and we therefore have to count the number of turns in a software revolution counter. That is done by (within the available method `input` below) adding or deducting 2π when the obtained motor angle over one sample period changes more than π radians (i.e. more than half a motor turn), which is the case when the measurement within one motor turn flips over to a new revolution. The implication for the software we are to develop is that the readings (call of `input` below) has to be done frequent enough (otherwise starting on a new motor shaft revolution will be confused with an ordinary turning and the robot will use an angle that is a multiple of 2π wrong).

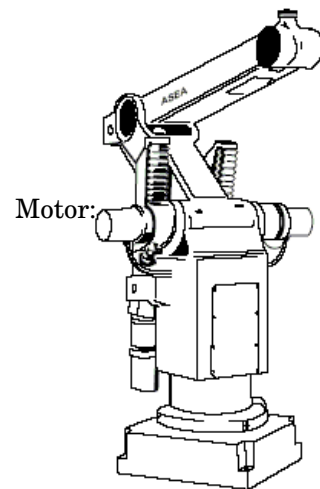
Apart from the importance of sufficiently frequent readings, for efficiency reasons we do not want to do the readings too frequent, and the time from reading the angle until the control of the motor should be as short as possible for performance reasons. These three criterias (clarified by the specifications below) are to be considered when the software for the motor control (not the algorithms) should be developed in this problem.



The main real-time programming problem (that should be clearly solved by your solution) is that we need a *rapid sequence of calls for fulfilling the needs for a short response time* for the feedback control, but this should be done periodically with a *period that is defined by the parameters that are set from outside* (and that can be slower than what is required for correct reading of the motor angle input). On the same time, the *readings of the motor angle must be frequent enough not to loose count on the motor turns* (that would make the robot go to the wrong position).

Note that both the revolution counting nor the feedback control algorithms are available and should not be written by you.

A robot with five motors (one marked) is shown to the right.



To focus on the real-time aspects only (and not on control or robotics), we assume that the robot has three degrees of freedom; x, y and z that are directly controlled by the motors 0, 1 and 2 (respectively). For the feedback control we need to be able to set the period for calling the control algorithms for each joint/motor, which motivates the use of separate threads for the control of each motor. Given that there are several such motor controllers, we from now on study the control/handling of one motor only. This simplified problem was depicted in the figures above.

Communication with available objects and hardware devices is done by using the following classes. Note that you do not need to know anything about automatic control or the control algorithm used in order to solve this problem. The available classes include:

```
public final class Motors { // Static interface to motor hardware and IO.
    public static final int MOTOR_X = 1;    // Use this device in this problem.
    public static final int MOTOR_Y = 2;    // Another motor, for the y dir.
    public static final int MOTOR_Z = 3;    // And yet another one, for z.
    public static final int MOTOR_NULL = 0; // To set before each new reading.

    /**
     * Select what motor angle to read. If MOTOR_NULL was selected previously,
     * a valid motor device such as MOTOR_X will trigger a new reading.
     * @param device the number of the motor to read, according to constants.
     */
    public final void select(int device) { // ...
    }

    /**
     * Read the motor angle, blocking caller until data available. Revolution
     * counting is done inside this method, if called frequently enough.
     * @param device the number of the motor to read, according to constants.
     */
    public final double input(int device) { // ...
    }

    /**
     * Send motor control signal as output to actuate control output.
     * @param device the number of the motor to read, according to constants.
     * @param value the desired motor torque, from Controller.control().
     */
    public final void output(int device, double value) { // ...
    }

    // Instead of constructor, to create new object with hardware access:
    public static native Motors createMotors(String unit) {
        // System dependent way to find hardware ports...
    }
}
```

The following class comprises the generic regulator/control:

```
public abstract class Controller {

    protected ControlPars pars; // Currently used parameters.
    protected double ref;      // Currently used reference

    public void setPars(ControlPars pars) { // Sets new parameters.
    }

    public void setRef(double ref) { // Sets a new reference to be reached.
    }
}
```

```

/**
 * Perform the control, using the currently set reference and parameters.
 */
public double control(double feedback) {} // To be defined by subclass.
}

```

The class `PIDControl` then implements the method `control` in a special way, but that is not part of this problem.

```

public abstract class ControlPars extends RTEvent {
    long period;
    public long getPeriod() {return period;}
    public void setPeriod(long period) {this.period = period;}
    // Etc., etc., for other parameters, which are not important here!
}

```

A sub-class `PIDPars` that inherits from `ControlPars` is then used in `PIDControl`, but also this part can be neglected here since we do not care about the control algorithms. Handling of the references (setpoints) and parameters is not time critical. The time critical part is the control and output after calling the `select` and `input` methods, as shown in the following example:

Example: Assume that instance 1 of the class `Controller` is of the subtype `PIDControl`, and that object is `pid1`, and that the instance of `Motors` is `motor`, then the control is done by

```

motors.select(Motors.MOTOR_NULL); // Reset after any previous reading.
motors.select(Motors.MOTOR_X); // Trigger reading by selecting motor.
double feedback = motor.input(Motors.MOTOR_X); // Blocks until ready.
double torque = pid1.control(feedback);
motor.output(Motors.MOTOR_X, torque);

```

Here we ignore any need for dividing calls between different threads as well as mutual exclusion (needed for `select` with the following `input`, but `output` can be called by any thread at any time without adding any synchronization).

Classes and values for parameters and references should be taken care of within the class `Servo` that should be written by you, such that sampling periods and references are passed to the controller object (`pid1` in the example above). As mentioned, there are no specific real-time demands for this part of the system. Your task is to design and implement classes/objects/threads according to the following specifications:

1. The overall control (not part of this problem) commands motions by using an instance of the **class `servo` with methods according to the figure above**. This class **should be implemented, together with additional threads and classes as needed**.
2. Within the software (that you are to develop) an instance of (provided as an argument to the `Servo` constructor) the class **`Motors` is used to handle the hardware interface**.
3. When controlling several motors, the control will be configured (by certain periods and offsets in time) such that only one control thread at a time need to execute. That is, here we can study the control of one single motor. It is suggested to use `Motors.MOTOR_X`.
4. A motor angle is read by selecting the actual motor address, after resetting the address (see `Motors.select`), and thereafter **reading the position via the blocking call of `input` (see `Motors.input`) that returns 8ms after selection of motor number**.
5. After selection of motor and reading of motor angle, one has to wait until the requested value is obtained. thus, a **reading of an input value cannot be interrupted**.
6. **Control of a motor** starts with **requesting** (calling `select`) **the angle** and the obtained angle (item 4) should then used as an **argument to `PIDControl.control` no later than 12ms after calling `select`**. the control method then returns the control signal, which in turn comprises the argument to `Motors.output`. This control sequence is the most time critical part of the system.

7. After control of the motor, alternatively concurrently with a lower priority, any new **parameters and references** are to be passed to the control object. the parameters **defines what period that applies to the calls of `PIDControl.control`**.
8. To ensure that revolution counting works, **reading of the motor angle must be done at least every 100ms**.
9. To ensure that the reading are actually carried out on time, we use an RTOS that provides strict priorities¹.
10. Times and sleeping is supported by the actual RTOS with a granularity of 1ms.
11. To minimize delays and to maximize performance, it is desirable to perform the control (calling `PIDControl.control`) in the same context as the reading of the input value is done in (that is, by the same thread that calls `Motors.input`).

You do not need to supervise that `Motors.input` actually returns within the prescribed time (we assume the hardware works or that it is checked by other means). Describe in particular how (e.g. by a sequence diagram or similar) the sequence of actions for input, control and output takes place (item 6), if an additional sensor reading was just requested (item 8).

(10p alt.² 8p)

1. When the software during initial development and testing is run in a ordinary JVM using a desktop OS (such as Windows or Linux, not realtime and without strict priorities) there is a simulated motor with internal logic that awaits the reading such that the needed frequent reading are ensured, which provides concurrently correctness. In this problem, however, we assume use of an RTOS and strict priorities.

2. A proper solution gives **8 p if the last requirement is not fulfilled**.

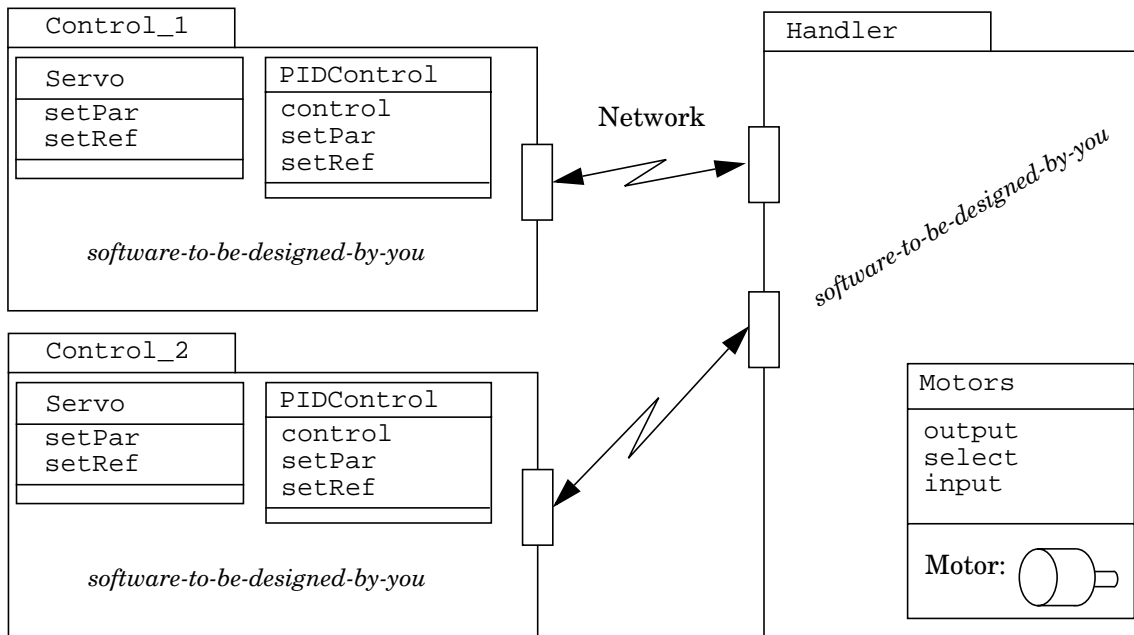
9. Safe motor control

Assume that a robot should work in the same workspace as manual work takes place. Then we must (apart from the reliable angle measurement according to previous problem) ensure that the computation of the control signals to the robot motors are performed correctly and that the values are actuated in the right time, or else personnel injuries may occur. Since the hardware can fail, we will in this problem make use of redundant/double computations of the control signals, which than are transferred via TCP/IP, or some more predictable network protocol and supporting hardware (see for instance www.tttech.com).

Based on then motor control according to the previous problem, you should now design a modified system where overall control (that is not part of this problem but that uses the objects of type `Servo`) and the motor control are carried out in two different computers. These computers are shown in the figure below as the `Control_1` and `Control_2` computers/components, which each contains an instance of `Servo` (problem 8). Implementation of this class as well as other parts of the software in `Control_*` must, however, be changed such that:

- Reading of angles are requested from the computer `Handler` with the frequency that should be used for the feedback control (that we also assume is the same in both `Control_1` and `Control_2`).
- A computed control signal is send via the network to `Handler` for actuation with the motor. (You may name your own network routines and you may assume that complete objects can be sent directly, but all writing to the sockets are assumed to be possible without blocking and all reading will be blocking until the requested data is available).

Handling of the hardware interface `Motors` is done in `Handler`, which according to the following figure communicates with the control using sockets.



Design of `Handler` should also be carried out. An important part in `Handler` is the comparison of the two control signals from `Control_1` and `Control_2`, and call of `Motors.output` if the signals are determined to be correct. Alternatively, if not correct, the available method `ControlSystem.emergency()` should be called. Values are considered to be correct if:

1. The two values are received at times that differ no more than 6ms.
2. These tow values have time-stamps (from the sender) that differ no more than 8ms.
3. The values differ no more than $0.01\%^1$

1. With the same implementation and the same type of computer, the values should be identical if the same input data is used, but obtained time values may be part of the algorithm and we plan to (like in the space shuttle) to have the control software developed by two independent groups (numerics may differ).

You should design but not implement this new safe structure, but to explain some key parts some source code can be useful. One such part is how the asynchronous control signals are taken care of in Handler, considering correctness in time and value according to the items above. It is expected by you to show the implementation of just this specific class (handling the control signals but not the measured input values), but otherwise no code is required.

Carry out the design, with focus on what threads that are needed and how these threads communicate/synchronize/block, using appropriate figures and explanations.

(8p)

