Embedded Control Systems

Exam 2016-06-02

14:00 – 18.00: M Building

Course code: SSY190

Teachers: Knut Åkesson

The teacher will visit examination halls twice to answer questions. This will be done approximately one hour after the examination started and one hour before it ends.

The exam comprises 30 credits. For the grades 3, 4, and 5, is respectively required 15, 20 and 25 credits.

Solutions and answers should be complete, written in English and be unambiguously and well motivated. In the case of ambiguously formulated exam questions, the suggested solution with possible assumptions must be motivated. The examiner retains the right to accept or decline the rationality of assumptions and motivations.

Exam results will be reported in Ladok. *The results* are open for review 2016-06-23 and 2016-08-31, between 12:30-13:30 at the department.

No aids are allowed on the written exam except:

- Standard pocket calculator (no hand computer). Erased memory.
- The report: "Computer Control: An Overview" Björn Wittenmark, Karl-Johan Åström, and Karl-Erik Årzén. International Federation of Automatic Control (IFAC) Professional Briefs. No comments are allowed in the article.
- The report: "C for Java Programmers", J. Maassen. No comments are allowed in the article.
- Dictionary from/to your native language to/from English

1

a) Describe how an acasual modeling language like Modelica is different from a casual based modeling language like Simulink.

(1p)

b) Explain how the priority inversion problems can occur, give a concrete example.

(2p)

c) Explain how semaphores works and how they can be used to implement mutual exclusion for multi-threaded programs.

(2p)

d) Explain what an interrupt is and how it can be used to implement the scheduler in a real-time operating systems.

(1p)

e) Demonstrate how Zeno-behavior might occur in modeling of hybrid systems and what problems it might cause during simulation.

(1p)

2

Consider the C-programs below.

a)
```
int main() {
    int i;
    int a[] = {0, 1, 2, 3, 4};
    int k = 5;
    foo(a, k);
    for(i = 0; i < k; i++)
        printf("%d, ", a[i]);
    return 0;
}

foo(int *b, int n) {
    int i;
    int j;
    for(i = n - 1; i > 0; i--) {
        for (j = 0; j < i; j++) {
            *(b+j) = *b + i;
        }
    }
}
```

Determine what is printed when the program is executed. Explain your reasoning.

(2p)

b)

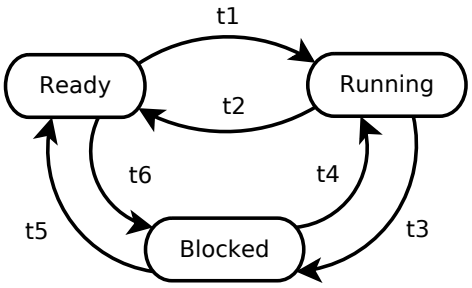
```
int *foo1()
{
  int x = 5;
  return &x;
}

int *foo2()
{
  int *x;
  *x = 10;
  return x;
}

int *foo3()
{
  int *x;
  x = (int *)malloc(sizeof(int));
  *x = 10;
  return x;
}
```

Discuss for each of the three functions `foo1`, `foo2`, and `foo3` if the way the functions are implemented are likely to cause problems with pointers.

(3p)

3

Each process in a real-time operating system can be in the following three states.



Which of the following transitions (`t1,...,t6`) exist for a task in a real-time operating system? Explain what can cause each transition to occur.

(3p)

4

Consider two tasks to be executed periodically on a single processor, where task 1 has period $p_1 = 4$ and and task 2 has period $p_2 = 10$. Assume task 1 has execution time $e_1 = 1$, and task 2 has execution time $e_2 = 7$. The deadline for each task is equal to the period of the same task.

a) Sketch a rate-monotonic schedule (for 20 time units, the least common multiple of 4 and 10).

(1p)

b) Now suppose task 1 and 2 contend for a mutex lock, assuming that the lock is acquired at the beginning of each execution and released at the end of each execution. Also, suppose that acquiring or releasing locks takes zero time and the priority inheritance protocol is used. Is the rate-monotonic schedule feasible, i.e. are the deadlines always met?

(2p)

c) Assume still that tasks 1 and 2 content for a mutex lock, as in part (b). Suppose that task 2 is running an anytime algorithm, which is an algorithm that can be terminated early and still deliver useful results. Find the maximum value for the execution time e_2 of task 2 such that the rate-monotonic schedule is feasible.

(2p)

d) For the original problem, where $e_1 = 1$ and $e_2 = 7$, and there is no mutex lock, sketch an EDF schedule for 20 time units. For tie-breaking among task executions with the same deadline, assume the execution of task 1 has higher priority than the execution of task 2. Is the schedule feasible?

(1p)

5

We have three processes that we want to execute on a single processor. T is the period of the task, D is the deadline of the task, C is the worst-case computation time.

Task	T	D	C
P_1	3	2	1
P_2	5	5	2
P_3	2	1	0.5

a) Assign priorities to the processes (P_1, P_2 and P_3) according to the Rate Monotonic Scheduling (RMS) principle and check using response-time analysis whether the tasks will meet their deadlines.

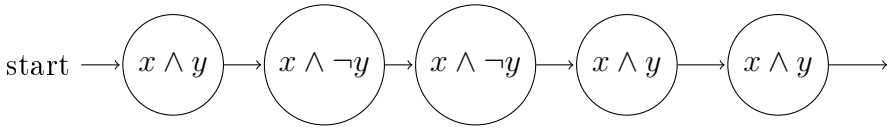
(2p)

b) Is the task set schedulable using the Earliest Deadline First (EDF) scheduling algorithm.

(1p)

6

Determine for each of the following LTL formula if they are satisified by the model below. Notation: G – Globally, F – Eventually, X – Next, U – Until. Motivate your answer!



a) $x \vee \neg y$

(1p)

b) $X\neg y$

(1p)

c) GF y.

(1p)

d) x U $\neg y$.

(1p)

e) X(y U $\neg x$)

(1p)

f) y U (XX y)

(1p)

Good Luck!

Formula sheet

Liu-Layland schedulability test

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq n(2^{1/n} - 1)$$

Response-time analysis

$$R_i = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

1/ See book.

2a/ pos 0 pos 1 pos 2 pos 3 pos 4

Printed: 10, 11, 10, 8, 4

Inside foo:

Initially a[i] = i for i = 0, ... 4

i goes from 4 to 0

j goes from 0 to i

Vector position j = Vector position 0 + i

- First a[4] = 4 a[3] = a[2] = a[1] = a[0] = 4
- Then a[3] = 8
  a[2] = 10
  a[1] = 11
  a[0] = 10

2b/ foo 1: Returns an address to a variable created on the stack. The value at the memory address might be overwritten when another function is called. Unsafe.

foo 2: X is a pointer to an int. No memory allocated for the value of an integer. Value 10 might be stored at arbitrary memory location. Unsafe.

foo 3: Memory properly allocated inside function. Safe behavior, but memory has to be explicitly deallocated.

3/ t4, and t6 does not exist.

t1: The process is scheduled for execution (highest priority process in the ready queue)

t2: The process is preempted (some other process with higher priority will be scheduled for execution instead).

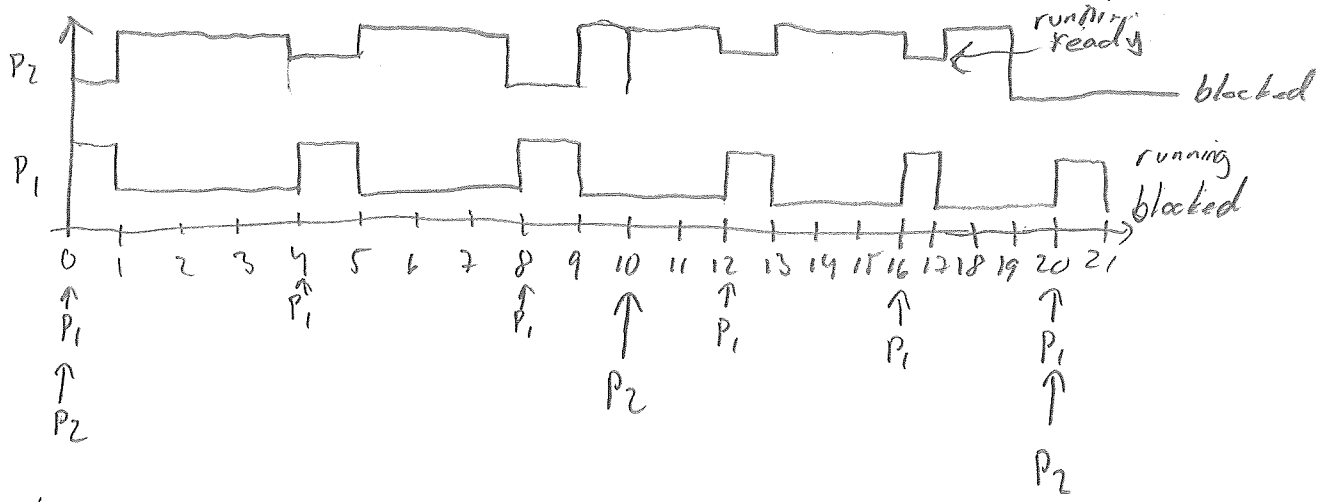
t3: The process might block on a semaphore (if value on semaphore becomes < 0). Or it might wait for some timer interrupt to be generated (this might be implemented by using a semaphore).

t5: The process might become unblocked as a consequence of some other process increased the value of a semaphore that this process was waiting for.

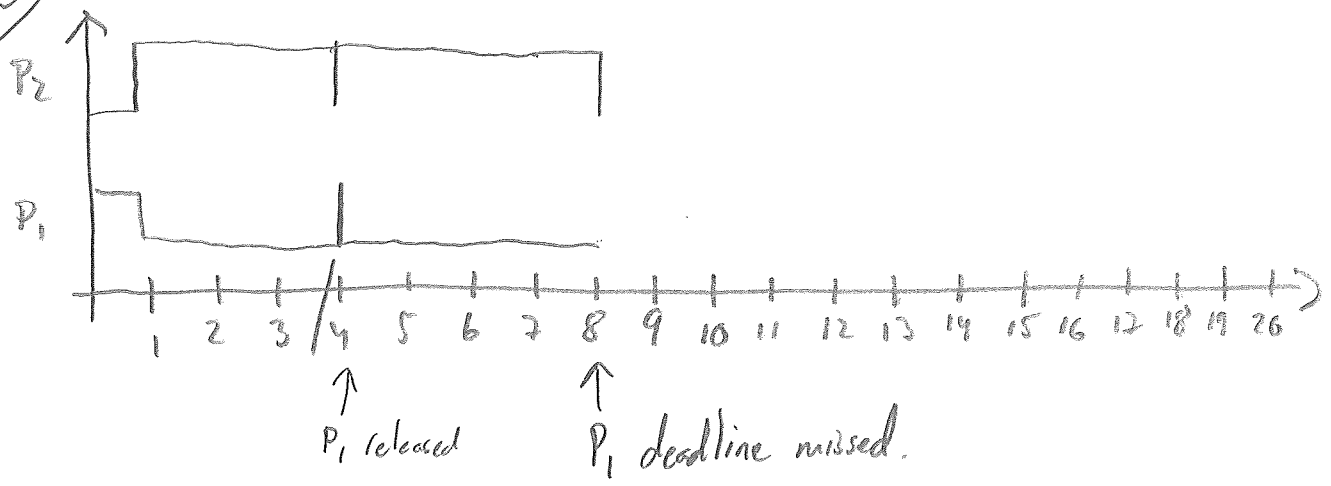
4/

a/

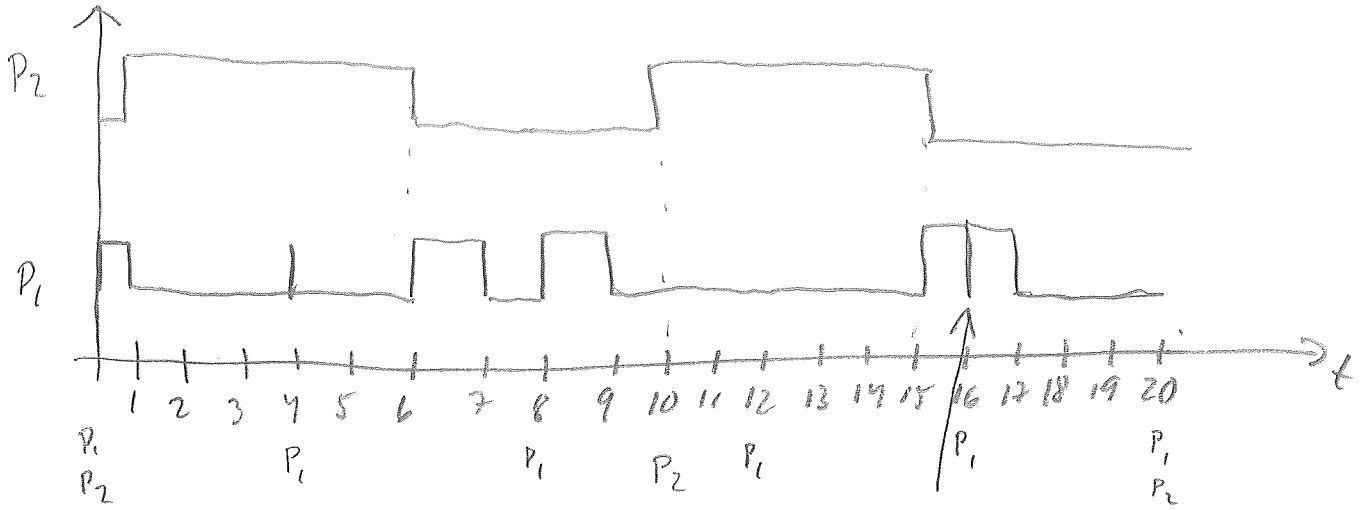
	T	C	D	
P_1	4	1	4	highest prio
P_2	10	7	16	lowest prio



b/

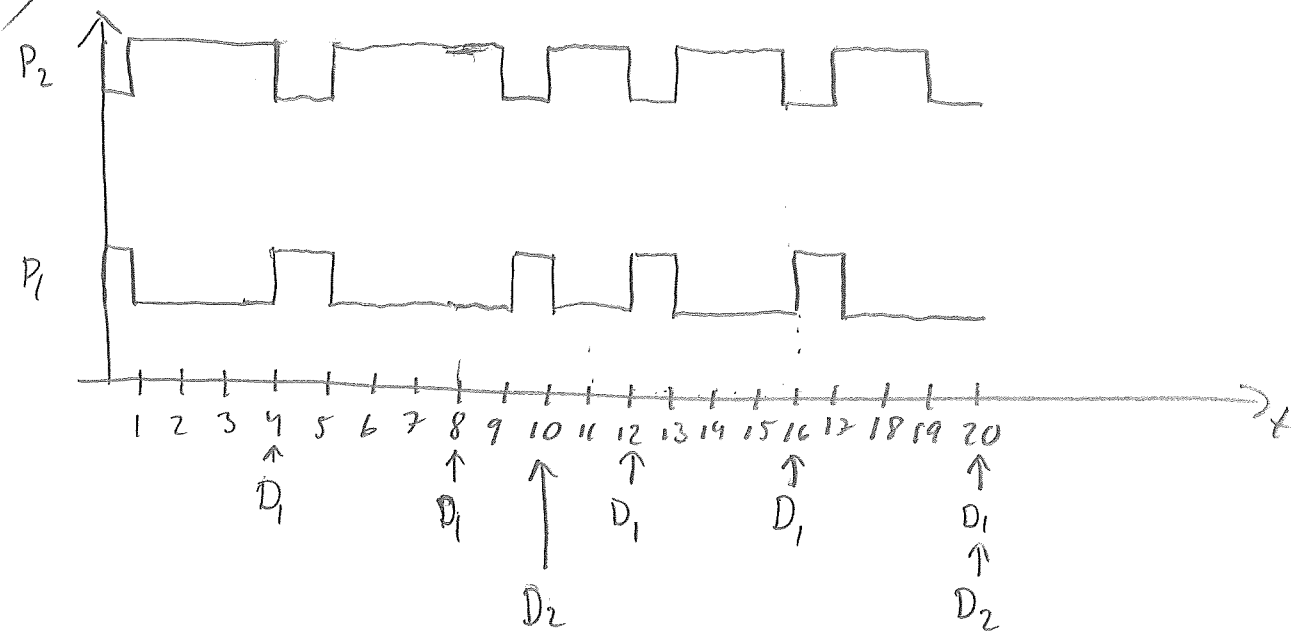


4c/ Test $C_2 = 5$



Critical. With $C_2 = 5$ P_1 will finish right before the deadline.

4d/ $C_2 = 7$



Yes, the schedule is feasible.

5/ a/

Highest prio : P_3

Medium prio : P_1

Low prio : P_2

$$R_i = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

$R_3 = 0,5$ (Deadline for R_3 is fulfilled)

$$R_1^0 = 1 + \left\lceil \frac{0,5}{3} \right\rceil \cdot 0,5 = 1,5$$

$$R_1^1 = 1 + \left\lceil \frac{1,5}{3} \right\rceil \cdot 0,5 = 1,5 \Rightarrow R_1 = 1,5$$ (Deadline for R_1 ok)

$$R_2^0 = 2 + \left\lceil \frac{2}{3} \right\rceil \cdot 1 + \left\lceil \frac{2}{2} \right\rceil \cdot 0,5 = 3,5$$

$$R_2^1 = 2 + \left\lceil \frac{3,5}{3} \right\rceil \cdot 1 + \left\lceil \frac{3,5}{2} \right\rceil \cdot 0,5 = 5$$

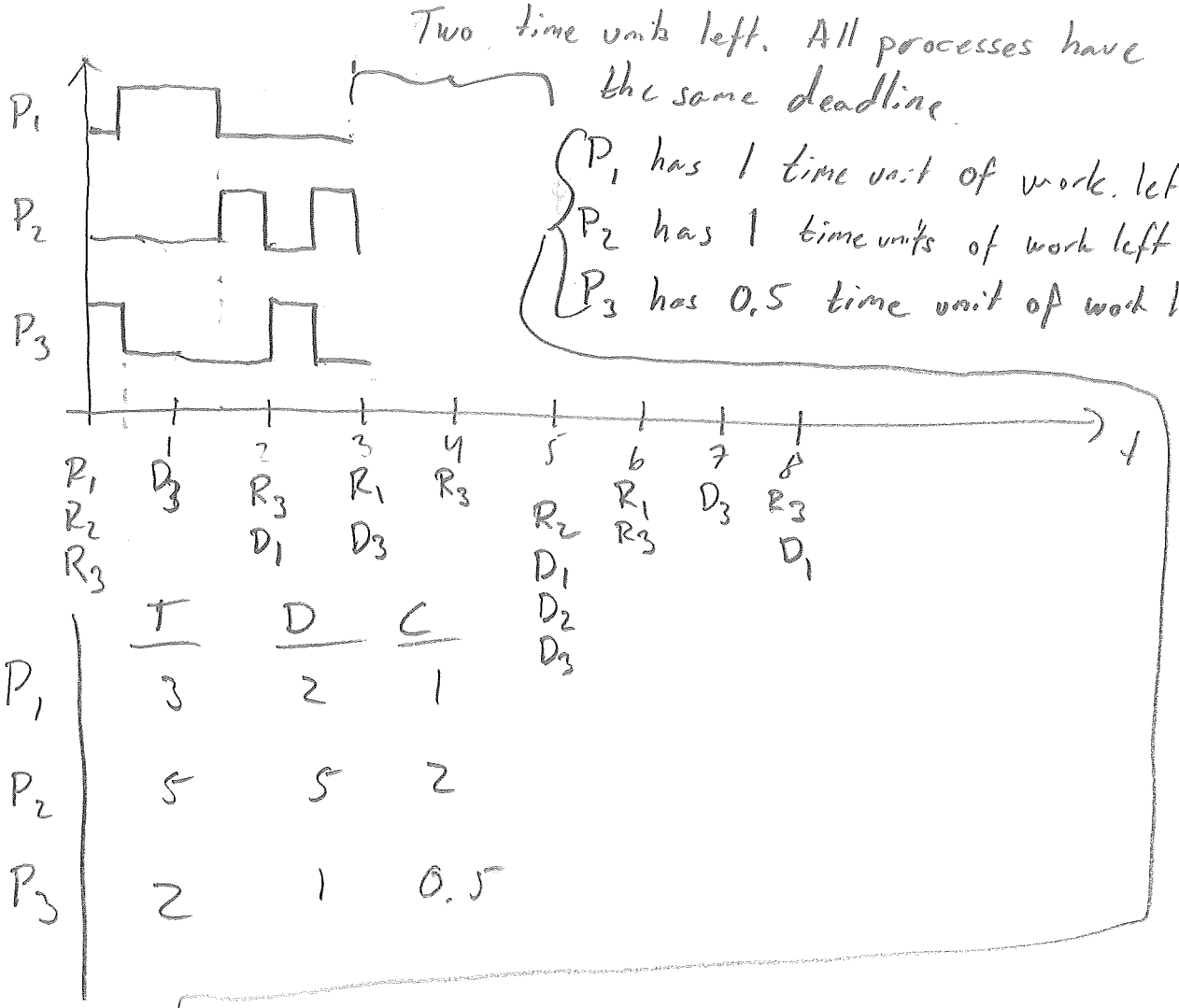
$$R_2^2 = 2 + \left\lceil \frac{5}{3} \right\rceil \cdot 1 + \left\lceil \frac{5}{2} \right\rceil \cdot 0,5 = 5,5$$

$$R_2^3 = 2 + \left\lceil \frac{5,5}{3} \right\rceil \cdot 1 + \left\lceil \frac{5,5}{2} \right\rceil \cdot 0,5 = 5,5$$ (Deadline not fulfilled)

$R_2 > D_2$.

5b) Since D≠T the utilization test for EDF cannot be used.

Draw schedule



	T	D	C
P_1	3	2	1
P_2	5	5	2
P_3	2	1	0.5

↳ No matter how the processes are scheduled will we be able to meet all three deadlines, thus the task set is not schedulable under EDF.

6/ Comment: After last arrow $x \wedge y$ are always true.

a/ Only applies to the initial state: True

b/ True ($\neg x$ holds in the second state)

c/ For every state, y will eventually be true. Thus true

d/ True: x is true until $\neg y$ becomes true

e/ False: $\neg y$ holds but x is true

f/ True: In the first state y is true, and in the second state XXy is true.