Real-Time Control Systems

Exam 2010-03-10

8:30 – 12:30: Halls at "Maskin"

Course code: SSY190

Teacher: Knut Åkesson, 0701-749525

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 announced on the department notice board on the latest 2010-03-24 at 12:30. *The results* are open for review 2010-03-25, 12:30-13:30 at the department.

Allowed aids:

- An A4 sheet with handwritten notes on one page. You should hand-in your notes together with your solutions.
- A pocket calculator with erased memory.
- Dictionary (paper and electronic) between English and the students native language.

1

a) What is priority inversion? Explain by giving a simple example. (1p)

b) Explain shortly how the priority inheritance protocol works. (1p)

c) Explain the windup problem and describe one anti-windup approach. (2p)

d) Describe the main differences between a semaphore and a monitor. (1p)

e) Describe how TTCAN (Time-Triggered CAN) can be used to decrease the latency and jitter for real-time control applications. (1p)

f) Compare Ethernet and EtherCAT, explain what make EtherCAT more suitable for distributed control systems than standard Ethernet. (2p)

2

A PD-controller is given by the following transfer function

$$F(s) = K_p + \frac{K_d s}{1 + T_f s}, \ T_f > 0.$$

a) Discretize the controller using the Euler Forward method and determine conditions for when the discretized controller will be stable.

(2p)

b) Discretize the controller using the Euler backward method and determine conditions for when the discretized controller will be stable.

(1p)

c) Let $R(s)$ be the set-point signal and $Y(s)$ be the actual output value from the process. In many situations the preferred solution is to compute the control signal for the PD-controller as

$$K_p(R(s) - Y(s)) + \frac{K_d s}{1 + T_f s}(\gamma R(s) - Y(s)),$$

where $0 \leq \gamma \leq 1$. The textbook controller has $\gamma = 1$, but in many pratical situations it is preferable to use a smaller γ. Explain the rationable behind using γ smaller than 1?

(2p)

3

We have five tasks that 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 time.

Task	T	D	C
P_1	1	0.1	0.1
P_2	1	1	0.2
P_3	2	0.5	0.4
P_4	2	2	0.4
P_5	10	10	1

a) Priorities are set using the Deadline Monotonic principle, thus the priorities fulfill the following relationship: $P_1 > P_3 > P_2 > P_4 > P_5$ (P_1 has the highest priority and P_5 the lowest, note that P_3 has higher priority than P_2.). Determine if all deadlines will be met. (assume the following: all tasks are release at time 0, no interprocess communication, tasks may not suspend themselves, ideal real-time kernel).

(3p)

b) Assume that task P_3 is a controller. The controller is designed in continous time and then discretized. In continous time the loop transfer function has a crossover frequency $\omega_c = 0.5$ rad$/s$. Since P_3 has a relative deadline of 0.5 seconds, the control output will be set at most 0.5 seconds after the release time – assuming all deadlines can be met. However, this delay might casue instability problems for the closed loop system. Determine how much the phase-margin (approximately) has to be increased to handle compensate for the delay caused by the implementation.

(2p)

4

We have two tasks that execute on a single processor using Earliest Deadline First scheduling. T is the period of the task, D is the deadline of the task, C is the worst-case computation time time.

Task	T	D	C
P_1	2	2	1
P_2	3	3	2

Assume that all tasks are released simultaneously at time 0. When two tasks have the same deadline then the task with the earliest release has priority. Since the utilization is above 100% all deadlines cannot be met. At what time and for what task will the first deadline miss occur?

(2p)

5

Three nodes are sending frames on a CAN network. All three nodes are sending the frames at the same time. Describe the arbitration phase, i.e., what bits are the nodes sending and what bits do they read back from the CAN-bus. The arbitration data for the three frames are given below.

Frame	Message priority in binary
A	1001 1000 0111
B	1000 0110 1010
C	1010 0101 1010

(1p)

6

A system consists of three processes, P_1, P_2, and P_3. Each process executes the following sequences of semaphore operations. The system has five semaphores A, B, C, D, and E.

P_1	P_2	P_3
wait(C)	wait(D)	wait(B)
wait(B)	wait(E)	wait(E)
signal(C)	signal(E)	wait(A)
wait(A)	signal(D)	signal(B)
wait(D)	wait(C)	signal(E)
signal(B)	signal(C)	signal(A)
signal(A)	wait(D)	
signal(D)	signal(D)	

a) Assume all semaphores are initialized to 1, determine if the system may deadlock or not. If it might deadlock then describe the deadlock situation.

(2p)

b) Assume that sempahores A, C, D, and E are initialized to 1 while sempahore B is initialized to 2. Determine if the system may deadlock or not. If it might deadlock then describe the deadlock situation.

(2p)

7

In many embedded application the processor do not have a floating-point unit thus the calculations for implementing a control algorithm for example has to be implemented using fixed-point arithmetics. You would like to implement the following controller

$$\begin{aligned} x(k+1) &= 0.90x(k) + 0.23y(k) \\ u(k) &= -3.3x(k) - 0.85y(k) \end{aligned}$$

using fixed-point arithmetic. All variables (y, u and x) should be represented using 8-bit signed integers, while all intermediate results may use 16 bits.

a) Convert the coefficient above to fixed-point numbers such that the you get the best possible resolution (given the restrictions above)

(1p)

b) Write pseduo C-code for implementing the controller calculations using fixed-point arithmetics. Declare the size of each variable. Make sure that x and u does not overflow. You can use the following methods for reading and writing data.

```
int8_t readInput();
void writeOutput(int8_t u);
```

(2p)

8

A semaphore can be defined either as only allowing positive values or it might be allowed to take both positive and negative values. The former approach is presented in the course book, while the latter was used in the lectures.

The code below shows the logical semantics of the semaphore primitives Wait and Signal as presented in the course book.

```
Wait(sem); <--->

IF sem^.counter = 0 THEN
  insert Running in the waiting queue of the semaphore;
ELSE
  sem^.counter = sem^.counter - 1;


Signal(sem); <--->

IF waiting queue is not empty THEN
  move first process in waiting queue to ReadyQueue;
ELSE
  sem^.counter = sem^.counter + 1;
```

Modify the implementation above such that both positive and negative values of the counter will be allowed. The behavior of an application using the Wait and Signal methods should not change.

(2p)

1)

a) See example 4.5 in the book.

b) See book.

c) See book

d) Monitors are used for solving the mutual exclusion problem while semaphores ~~can~~ be used to implement different synchronization problems.

Conditions are associated with monitors, this is not the case for semaphores.

For monitors, typically all tasks waiting for the monitor is woken up when a condition is fulfilled. For semaphores only one thread is woken up.

e) TTCAN has a predefined schedule, so different processes may have preallocated slots where they are guaranteed to be the single sender. Thus it it possible to estimate excatly when a process can send data on the bus

f) Ethernet - one frame for each message

EtherCAT - one frame for multiple messages (telegrams)

EtherCAT is also point-to-point protocol, thus no medium access protocol is necessary.

8/

$$F(s) = K_p + \frac{K_d \cdot s}{1 + T_f s} \quad , \quad T_f > 0$$

Euler forward $s \leftrightarrow \frac{z-1}{h}$, Euler backward $s \leftrightarrow \frac{z-1}{zh}$

a/

$$H(z) = K_p + \frac{K_d \cdot \frac{z-1}{h}}{1 + T_f \cdot \frac{z-1}{h}} = \frac{K_p + K_p T_f \left(\frac{z-1}{h}\right) + K_d \left(\frac{z-1}{h}\right)}{1 + T_f \cdot \frac{z-1}{h}} =$$

$$= \frac{\frac{h \cdot K_p}{T_f} + \left(K_p + \frac{K_d}{T_f}\right)(z-1)}{z + \frac{h - T_f}{T_f}}$$

The controller is stable if $\left| \frac{h - T_f}{T_f} \right| < 1$

b/

$$H(z) = K_p + \frac{K_d \cdot \frac{z-1}{zh}}{1 + T_f \frac{z-1}{zh}} = \frac{zh K_p + K_d (z-1)}{zh + T_f (z-1)} = \frac{-K_d + (K_d + h K_p) z}{(h + T_f) z - T_f}$$

The controller is stable if $\left| \frac{T_f}{h + T_f} \right| < 1$

Since $h, T_f > 0$ ... this will always be fulfilled.

c/ See set-point weighting in the book

3/ Step 1

$$R_1^0 = 0,1$$

$$R_2^0 = 0,2 + 0,1 + 0,4 = 0,7$$

$$R_3^0 = 0,5$$

$$R_4^0 = 0,4 + 0,4 + 0,2 + 0,1 = 1,1$$

$$R_5^0 = 1,1 + 1 = 2,1$$

Step 2

$$R_1^1 = 0,1$$

$$R_2^1 = 0,2 + \left\lceil \frac{0,7}{2} \right\rceil \cdot 0,4 + \left\lceil \frac{0,7}{1} \right\rceil \cdot 0,1 = 0,7$$

$$R_3^1 = 0,4 + \left\lceil \frac{0,5}{1} \right\rceil \cdot 0,1 = 0,5$$

$$R_4^1 = 0,4 + \left\lceil \frac{1,1}{2} \right\rceil \cdot 0,4 + \left\lceil \frac{1,1}{1} \right\rceil \cdot 0,2 + \left\lceil \frac{1,1}{1} \right\rceil \cdot 0,1 = 0,4 + 0,4 + 0,4 + 0,2 = 1,4$$

$$R_5^1 = 1 + \left\lceil \frac{2,1}{2} \right\rceil \cdot 0,4 + \left\lceil \frac{2,1}{2} \right\rceil \cdot 0,4 + \left\lceil \frac{2,1}{1} \right\rceil \cdot 0,2 + \left\lceil \frac{2,1}{1} \right\rceil \cdot 0,1 =$$

$$= 1 + 0,8 + 0,8 + 0,6 + 0,3 = 3,5$$

Step 3

$$R_4^2 = 0,4 + \left\lceil \frac{1,4}{2} \right\rceil \cdot 0,4 + \left\lceil \frac{1,4}{1} \right\rceil \cdot 0,2 + \left\lceil \frac{1,4}{1} \right\rceil \cdot 0,1 = 1,4$$

$$R_5^2 = 1 + \left\lceil \frac{3.5}{2} \right\rceil \cdot 0.4 + \left\lceil \frac{3,5}{2} \right\rceil \cdot 0.4 + \left\lceil \frac{3.5}{1} \right\rceil \cdot 0,2 + \left\lceil \frac{3.5}{1} \right\rceil \cdot 0,1 =$$

$$= 1 + 0,8 + 0,8 + 0,8 + 0.4 = 3.8$$

Step 4 $R^3 = 1 + \lceil 3.8 \rceil \cdot 0.4 + \lceil 3.8 \rceil \cdot 0.4 + \lceil 3.8 \rceil \cdot 0.2 + \lceil 3.8 \rceil \cdot 0.1 = 3.8$

3b) $\omega_c = 0{,}5$ rad/s

$D = 0.5$ sekunder

$\tilde{G}(s) = e^{-0.5s}$

$\arg G(j\omega) = -0.5 j\omega$

$\arg G(j\omega_c) = -0.25$ rad $\hat{=} \underline{\underline{14{,}3°}}$

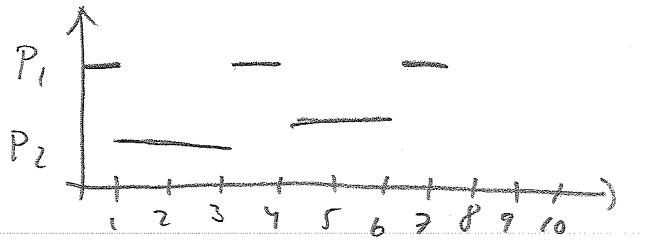
4/

$$U = \frac{1}{2} + \frac{2}{3} > 1$$

(release time, deadline)

P_1: (0, 2), (2, 4), (4, 6), (6, 8)

P_2: (0, 3), (3, 6), (6, 9)



P_1 miss deadline

5)

	1000	0110	1010
A	1001	1000	0111
B	1000	0110	1010
C	10~~1~~0	0101	1010

A stops sending (at 1001)

C stops sending (at 1010)

0 is dominant

b/

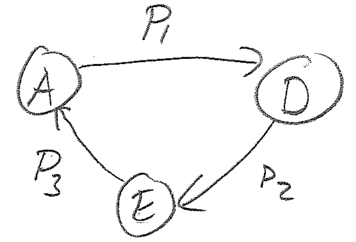
P_1	P_2	P_3
wait(C) (2)	wait(D)	wait(B)
wait(B)	wait(E)	wait(E)
signal(C)	signal(E)	wait(A)
wait(A)	signal(D)	signal(B)
wait(D)	wait(C) } (1)	signal(E)
signal(B)	signal(C) } (1)	signal(A)
signal(A)	wait(D) } (1)	
signal(D)	signal(D) } (1)	

A circular chain is necessary for deadlock situations.

(1). Note that we have no hold-and-wait situation so a deadlock is impossible here. (se P_2)

(2) C is used by both P_1 and P_2, but no deadlock can occur in P_2 while calling wait(C), thus there might be no deadlock in P_1 either (when calling wait(C)).

Note that the following chain exits



There might be a deadlock, however. note that if B is initialized to 1 then either P_1 can book A and D <u>or</u> P_3 can book E and A. Thus it is not possible to create the chain unless B is initialized to 2.

However, if B is initialzed to 2 then the following sequence will result in a deadlock: P_1: wait(C), P_1: wait(B), P_1: signal(C), P_1: wait(A), P_2: wait(D), P_3: wait(B), P_3: wait(E) ⟹ deadlock.

7/ 8 bits (signed) ⟹ we can represent numbers in the range $[-128, +127]$.

Largest coefficient is 3.3 ⟹ $\log_2\left(\frac{128}{3.3}\right) = 5.28$ ⟹ we can scale all coefficients with a factor 2^5.

$A = \text{round}(0.90 \cdot 2^5) = 29$

$B = \text{round}(0.23 \cdot 2^5) = 7$

$C = \text{round}(-3.3 \cdot 2^5) = -104$

$D = \text{round}(-0.85 \cdot 2^5) = -27$

b/
```
#define A 29
#define B 7
#define C -104
#define D -27

int8_t y, x, u;
int16_t x16 = 0, u16 = 0;

y = readInput();

u16 = u16 + ((int16_t)D * (int16_t)y) >> N;

if (u16 > 127) u = 127
else if (u16 < -128) u = -128
else u = u16
writeOutput(u)
x16 = ((int16_t)A * (int16_t)x + (int16_t)B * (int16_t)y) >> N
if (x16 > 127) x = 127
else if x16 < -128 x = -128
else x = x16;
u16 = ((int16_t)C * (int16_t)x) >> N
```

8/

Wait(sem) ⟺

sem^.counter = sem^.counter - 1;
IF sem^.counter < 0 THEN
  insert Running in the waiting queue of the semaphore;

Signal(sem) ⟺

IF sem^.counter < 0 THEN
  move first process in the waiting queue to Ready Queue
sem^.counter = sem^.counter + 1;