Assignment 2: Response Time Analysis

Question 1:
Assume a real-time system with seven periodic tasks:

Task	Period (T)	Deadline (D)	Exec. time (C)
A	1000	20	3
B	100	100	10
C	50	50	20
D	57	10	5
E	33	33	1
F	7	7	1
G	30	5	2

There are also four semaphores used to protect shared resources, accessed by the tasks as follows (The critical sections are NOT nested):

Task	Semaphore	Length of critical section
A	S_1	2
	S_3	2
B	S_2	7
	S_3	5
	S_4	2
D	S_1	2
C	S_2	1
G	S_1	1

For example, we can see above that task B uses three semaphores S_2, S_3 , S_4, and the length of the critical sections of task B is 7 when using S_2, 5 for S_3 and 2 for S_4.

Assume *Deadline Monotonic (DM)* priority assignment for the tasks, and *Priority Ceiling Protocol (PCP)* for semaphore access.

Calculate worst-case response times of the tasks **A** and **G**. Be sure to show the steps in your calculations, For example, explicitly show how you arrive at the blocking times.

Question 2
Holistic Response-Time Analysis

Consider the system depicted in Figure 1, consisting of 3 identical sensor-actuator nodes called A, B and C and the control-node D. The nodes are connected with a CAN-bus running at 75kbps.

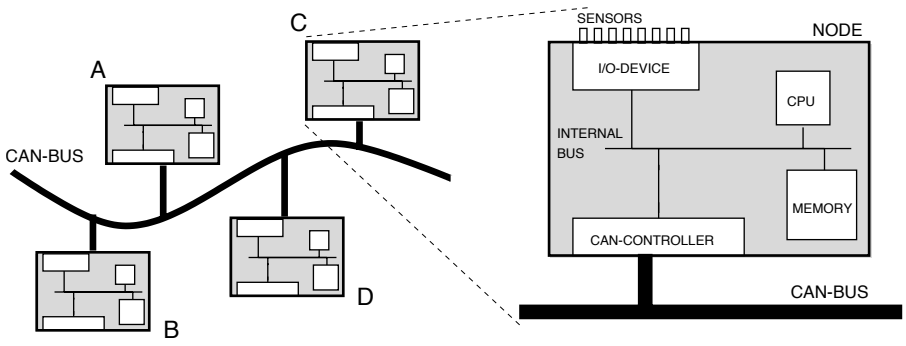

Figure 1: *A CAN-based architecture with four micro-controllers*

The system contains three distributed transactions. The transactions all have the same structure: Read a sensor at node X, send the sensor value to the controller node, calculate the actuator-value at node D, send the actuator value to node X, and finally actuate with the received value at node X (where X∈{A,B,C}). Figure 2 shows the complete graph of task and messages, and their precedence constraints (as well as their grouping into three transactions).

The transactions are denoted Trans.1, Trans.2, and Trans.3. They all origin (and ends) in node A, B, and C respectively. The parameters for each of the periodic transactions are (transactions have no release-jitter, C_{name} is the execution time of task *name* in a transaction X and S_{name} is the size of the payload for CAN-message *name* in transaction X, X∈{A,B,C}):

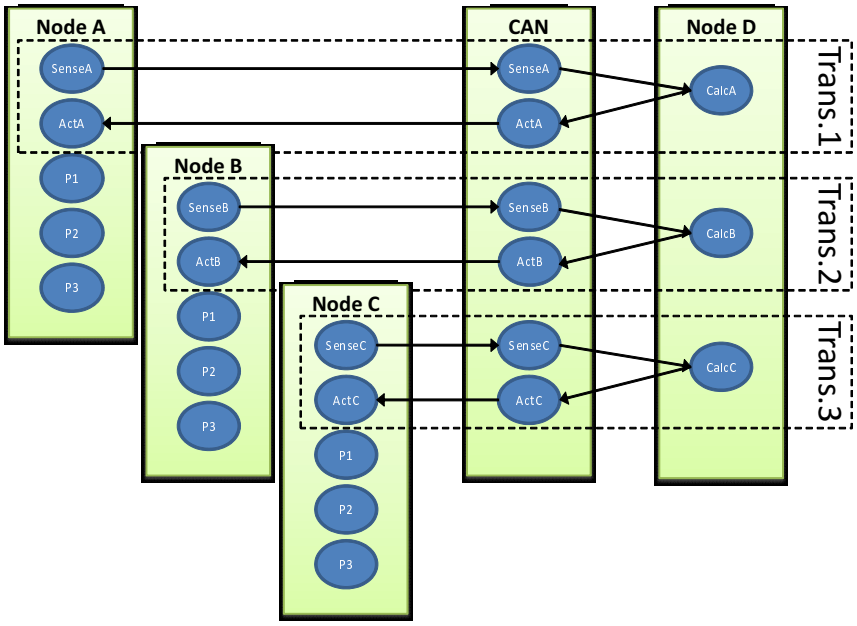

Figure 2: *All the tasks and messages in the system*

Trans	T ms	D ms	C_{Sense} ms	S_{Sense} bytes	C_{Calc} ms	S_{Act} bytes	C_{Act} ms
Trans.1	10	15	1	2	1	1	1
Trans.2	10	20	1	4	2	2	1
Trans.3	20	40	2	4	4	4	1

In addition to the tasks from the transactions, each of the sensor-actuator nodes has a set of local tasks according to figure 2 with the following parameters:

Task	T	D	C	J
P1	5	2	1	0
P2	15	10	2	2
P3	50	25	1	5

For the controller-node there are no other tasks than the tree control tasks. For the CAN-bus we assume that there may be low-priority background frames of unknown size. For simplicity we assume that tasks have zero blocking.

Preparations

Download the tool FpsCalc from Canvas (under Files->Others). (Windows-version is verified to work; *NIX-version may work if you are in luck. . . .) Also download the man-ual and the two example files.

Familiarize yourself with FpsCalc by reading the manual and experimenting with the example files. The example "holistic_example.fps" is very important to understand to be able to solve the following assignments. (Teachers will be very restrictive to answer questions regarding technical issues that are documented in the manual – you are expected figure out how to use FpsCalc using the manuals and examples.)

Rules

All assignments (A–E) should be solved using FpsCalc. The solution handed in should be a short report with a section for each assignment below. The section should report the **bold face** numbers from each assignment in short and readable way (e.g. nice looking tables).

In addition your handed in solution should have an appendix with the final FpsCalc input-file you use (i.e. a file that include the solutions to assignments A, B, C and E). This FpsCalc file should be nicely formatted and commented and use the conventional notation (e.g. C_i for execution-/transmission-time).

Is a second appendix you should inculde to complete output of running FpsCalc on your handed in input-file. Note: Only *one* FpsCalc input-file shold be handed in.

Assignment A

Calculate the maximum **size (in bits)** and **transmission time** for each of the CAN messages.

Assignment B

Calculate the load (**utilization**) for each of the nodes and the CAN-bus. HINT: You'd have to declare an array for each task in a node and assign the result from a "sigma(all, . . .)" operation to each to the elements in that array.

Assignment C

Assign **priorities** according to rate-monotonic to each of the tasks and messages. HINT: Where there is local precedence order between tasks; priorities have to be assigned in ascending order (i.e. from high to low in the order of precedence). HINT2: You do not need to use FpsCalc to *assign* priorities (although it is possible to do so); but you need to provide the FpsCalc-file with the correct order.

Assignment D

Assign jitter to the tasks and messages according to the jitter-inheritance method for holistic scheduling. Now, calculate the **busy-period windows** and local **response-times**. These response-times will include the final response-times for the transactions: Also, make FpsCalc separately report the **transactions' response-times**.