Assignment 2, Embedded Systems II Mälardalen University

Emil Broberg
School of Innovation, Design and Engineering
Mälardalen University, Västerås, Sweden
Email: ebg20007@student.mdu.se

I. Question 1

In this assignment

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

Figure 1. Task set

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

Figure 2. Task set

$R^0 = C_i$ iterate until $R^{n+1} = R^n$, or $R^{n+1} > D_i$. If $R^{n+1} > D_i$, the task set is not schedulable.

A. *Priorities*

The priority of each task is determined by the size of its deadline. The proirity is inversely proportional to the deadline, i.e. the smaller the deadline, the higher the priority. The following list shows the priorities of the tasks in the task set where the highest priority is 7 and the lowest proirity is 1:

- $P(A) = 4$
- $P(B) = 1 \leftarrow$ Lowest
- $P(C) = 2$
- $P(D) = 5$
- $P(E) = 3$
- $P(F) = 6$
- $P(G) = 7 \leftarrow$ Highest

The priority ceiling of each semaphore is the highest priority of the tasks that use the semaphore. The following list shows the priority ceilings of the semaphores:

- $ceil(S_1) = max(P(A), P(D), P(G)) = 7$

- $ceil(S_2) = max(P(B), P(C)) = 2$
- $ceil(S_3) = max(P(A), P(B)) = 4$
- $ceil(S_4) = P(B) = 1$

Using the Priority Ceiling Protocol (PCP), the priority of a task is the highest priority of the semaphores it is waiting for.

B. *Blocking time*

In this example we need to consider blocking time such that we can use the formula:

$$R_i^{n+1} = C_i + B_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i^n}{T_i} \right\rceil C_j$$

where B_i is the blocking time of task i. Since we are going to find the response times of task A and G we need to fint B_A and B_G.

$B_A =?$
$lp(A) =$ all tasks that has lower priority than A.

- $lp(A) = \{B, C, E\}$
- B uses S_2, S_3 and S_4
  - $P(A) > ceil(S_2) = 4 > 2$? Yes $\rightarrow$ B with S_2 cannot block A.
  - $P(A) > ceil(S_3) = 4 > 4$? No $\rightarrow$ B with S_3 can block A.
  - $P(A) > ceil(S_4) = 4 > 1$? Yes $\rightarrow$ B with S_4 cannot block A.
- C uses S_2
  - $P(A) > ceil(S_2) = 4 > 2$? No $\rightarrow$ C with S_2 can block A.
- E uses no semaphore
  - $P(A) > P(E) = 4 > 3$? Yes $\rightarrow$ E cannot block A.
- $B_A = CS(B, S_3) = 5$

$B_G =?$
$lp(G) =$ all tasks that has lower priority than G.
$lp(G) = \{A, B, C, D, E, F\}$

- A uses S_1 and S_3
  - $P(G) > ceil(S_1) = 7 > 7$? No $\rightarrow$ A with S_1 can block G.
  - $P(G) > ceil(S_3) = 7 > 4$? Yes $\rightarrow$ A with S_2 cannot block G.
- B uses S_2, S_3 and S_4
  - $P(G) > ceil(S_2) = 7 > 2$? Yes $\rightarrow$ B with S_2 cannot block G.
  - $P(G) > ceil(S_3) = 7 > 4$? Yes $\rightarrow$ B with S_3 cannot block G.
  - $P(G) > ceil(S_4) = 7 > 1$? Yes $\rightarrow$ B with S_4 cannot block G.
- C uses S_2
  - $P(G) > ceil(S_2) = 7 > 2$? Yes $\rightarrow$ C with S_2 cannot block G.
- D uses S_1
  - $P(G) > ceil(S_1) = 7 > 7$? No $\rightarrow$ D with S_1 can block G.
- E uses no semaphore
  - $P(G) > P(E) = 7 > 3$? Yes $\rightarrow$ E cannot block G.
- F uses no semaphore
  - $P(G) > P(F) = 7 > 6$? Yes $\rightarrow$ F cannot block G.
- $B_G = max(CS(A, S_1), CS(D, S_1)) = 2$

Conclusively, $B_A = 5$ and $B_G = 2$.

C. Response Time Analysis (RTA)

Doing the RTA of the tasks A and G we get the following results:
$R_A^{n+1} = C_A + B_A + \sum_{\forall j \in hp(A)} \left\lceil \frac{R_A^n}{T_j} \right\rceil C_j$
$j \in hp(A) = \{D, F, G\}$
$R_A^0 = C_A + B_A = 8$
$R_A^1 = C_A + B_A + \sum \left\lceil \frac{R_A^0}{T_j} \right\rceil C_j = 3 + 5 + \left\lceil \frac{8}{57} \right\rceil 5 + \left\lceil \frac{8}{7} \right\rceil 1 + \left\lceil \frac{8}{30} \right\rceil 2 = 3 + 5 + [1] * 5 + [2] * 1 + [1] * 2 = 17$
$R_A^2 = C_A + B_A + \sum \left\lceil \frac{R_A^1}{T_j} \right\rceil C_j = 3 + 5 + \left\lceil \frac{17}{57} \right\rceil 5 + \left\lceil \frac{17}{7} \right\rceil 1 + \left\lceil \frac{17}{30} \right\rceil 2 = 3 + 5[1] * 5 + [3] * 1 + [1] * 2 = 18$
$R_A^3 = C_A + B_A + \sum \left\lceil \frac{R_A^2}{T_j} \right\rceil C_j = 3 + 7 + \left\lceil \frac{18}{57} \right\rceil 5 + \left\lceil \frac{18}{7} \right\rceil 1 + \left\lceil \frac{18}{30} \right\rceil 2 = 3 + 5[1] * 5[3] * 1 + [1] * 2 = 18$
$R_A^3 = R_A^2 = 18$

$R_A = 18$

$R_G^{n+1} = C_G + B_G + \sum_{\forall j \in hp(G)} \left\lceil \frac{R_G^n}{T_j} \right\rceil C_j$
$j \in hp(G) = \{$No tasks with higher priority than $G\}$
Worst case scenario, $R_G = C_G + B_G = 2 + 2 = 4$

$R_G = 4$

II. Question 2

A. Assignment A

Message	Maximum bit size	Transmission time (ms)
Sense A	75	1.000
Sense B	95	1.267
Sense C	95	1.267
Act A	65	0.867
Act B	75	1.000
Act C	95	1.267

Figure 3. Maximum bit size and transmission time for each message

B. Assignment B

Node & CAN	Load (percent)
Node A	55.33
Node B	55.33
Node C	50.33
Node A	50.00
CAN	54.00

Figure 4. Load (utilization) of on each node and the CAN bus. Displayed in percetage.

C. Assignment C

Node/CAN	Task/Message	Priority
Node A	SenseA	2
	ActA	2
	P1A	1
	P2A	3
	P3A	4
Node B	SenseB	2
	ActB	2
	P1B	1
	P2B	3
	P3B	4
Node C	SenseC	2
	ActC	2
	P1C	1
	P2C	3
	P3C	4
Node D	CalcA	1
	CalcB	1
	CalcC	2
CAN	SenseA	1
	ActA	1
	SenseB	1
	ActB	1
	SenseC	2
	ActC	2

Figure 5. The priorities of the tasks and messages. Priority is set according to Rate Monotonic and 1 is the highest priority possible.

D. Assignment D

Node	Task	Local Response-Time (ms)
Node A	SenseA	2.000
	ActA	6.867
	P1A	1.000
	P2A	7.000
	P3A	13.000
Node B	SenseB	2.000
	ActB	8.267
	P1B	1.000
	P2B	9.000
	P3B	13.000
Node C	SenseC	5.000
	ActC	29.800
	P1C	1.000
	P2C	5.000
	P3C	14.000
Node D	CalcA	4.000
	CalcB	5.267
	CalcC	20.400

Figure 6. The local response-time for each task on each node.

Busy periods on CAN	Periods
Trans1	1.867
Trans2	2.267
Trans2	10.800

Figure 7. The busy periods on the CAN bus.

Transaction	Transaction Time
Trans1	6.867
Trans2	8.267
Trans2	29.800

Figure 8. The transaction time for each transaction.

APPENDIX

The code written in FpsCalc:

```
! ======================================================================
! ============================ Emil Broberg ============================
! ======================================================================

! Global variables declarations
scalar
        LoadNode_A, LoadNode_B, LoadNode_C, LoadNode_D, LoadNode_CAN;
scalar
        MBS_S_senseA, MBS_S_senseB, MBS_S_senseC, MBS_S_actA, MBS_S_actB, MBS_S_actC;
scalar
        Trans_t_S_senseA, Trans_t_S_senseB, Trans_t_S_senseC, Trans_t_S_actA,
```

```
        Trans_t_S_actB , Trans_t_S_actC;
indexed
        Jafter1 , Jafter2 , Jafter3 , Jafter4 , TransactionsR;
tasks
        Trans1 , Trans2 , Trans3;

! ==========================================================================

system Node_A {
        declarations {
                indexed T, D, C, W, J, R, LoadArray;
                priority P;
                scalar B;

                ! Tasks in Node A
                tasks Sense_A, Act_A, P1A, P2A, P3A;
        }

        initialise {
                ! Period for each task
                T[Sense_A] = 10;
                T[Act_A] = 10;
                T[P1A] = 5;
                T[P2A] = 15;
                T[P3A] = 50;

                ! Release Jitter for each task
                J[Sense_A] = 0;
                J[Act_A] = 0;
                J[P1A] = 0;
                J[P2A] = 2;
                J[P3A] = 5;

                ! Execution time for each task
                C[Sense_A] = 1;
                C[Act_A] = 1;
                C[P1A] = 1;
                C[P2A] = 2;
                C[P3A] = 1;

                ! Priority according to Rate Monotonic, 1 = Highest priority
                P[Sense_A] = 2;
                P[Act_A] = 2;
                P[P1A] = 1;
                P[P2A] = 3;
                P[P3A] = 4;

                ! Deadline for each task
                D[Sense_A] = 15;
                D[Act_A] = 15;
                D[P1A] = 2;
                D[P2A] = 10;
                D[P3A] = 25;

                ! Blocking is zero since we use no semaphores and RM-scheduling
                B = 0;
```

```
        }

        formulas {
                ! Act_A inherits Jitter from previously executed tasks
                J[Act_A] = Jafter4[Trans1];

                ! Calculate window of interference
                W[i] = C[i] + B + sigma(hp, ceiling((W[i]+J[j])/T[j]) * C[j]);
                ! Calculate the response-time
                R[i] = W[i] + J[i];

                ! Jitter to be inherited by Node_CAN
                Jafter1[Trans1] = R[Sense_A];

                ! Save R in global array TransactionsR
                TransactionsR[Trans1] = R[Act_A];

                ! Load (utilization)
                LoadArray[i] = C[i] / T[i];
                ! Convert to percent to display as result
                LoadNode_A = 100 * (LoadArray[Sense_A] + LoadArray[Act_A] +
        LoadArray[P1A] + LoadArray[P2A] + LoadArray[P3A]);
        }
}

! ====================================================================

system Node_B {
        declarations {
                indexed T, D, C, W, J, R, LoadArray;
                priority P;
                scalar B;

                ! Tasks in Node B
                tasks Sense_B, Act_B, P1B, P2B, P3B;
        }

        initialise {
                ! Period for each task
                T[Sense_B] = 10;
                T[Act_B] = 10;
                T[P1B] = 5;
                T[P2B] = 15;
                T[P3B] = 50;

                ! Release Jitter for each task
                J[Sense_B] = 0;
                J[Act_B] = 0;
                J[P1B] = 0;
                J[P2B] = 2;
                J[P3B] = 5;

                ! Execution time for each task
                C[Sense_B] = 1;
                C[Act_B] = 1;
                C[P1B] = 1;
```

```
            C[P2B] = 2;
            C[P3B] = 1;

            ! Priority according to Rate Monotonic, 1 = Highest priority
            P[Sense_B] = 2;
            P[Act_B] = 2;
            P[P1B] = 1;
            P[P2B] = 3;
            P[P3B] = 4;

            ! Deadline for each task
            D[Sense_B] = 20;
            D[Act_B] = 20;
            D[P1B] = 2;
            D[P2B] = 10;
            D[P3B] = 25;

            ! Blocking is zero since we use no semaphores and RM-scheduling
            B = 0;
        }

        formulas {
            ! Act_B inherits Jitter from previously executed tasks
            J[Act_B] = Jafter4[Trans2];

            ! Calculate window of interference
            W[i] = C[i] + B + sigma(hp, ceiling((W[i]+J[j])/T[j]) * C[j]);
            ! Calculate the response-time
            R[i] = W[i] + J[i];

            ! Jitter to be inherited by Node_CAN
            Jafter1[Trans2] = R[Sense_B];

            ! Save R in global array TransactionsR
            TransactionsR[Trans2] = R[Act_B];

            ! Load (utilization)
            LoadArray[i] = C[i] / T[i];
            ! Convert to percent to display as result
            LoadNode_B = 100 * (LoadArray[Sense_B] + LoadArray[Act_B] + LoadArray[P1B]
        + LoadArray[P2B] + LoadArray[P3B]);
        }
}

! ==================================================================

system Node_C {
        declarations {
            indexed T, D, C, W, J, R, LoadArray;
            priority P;
            scalar B;

            ! Tasks in Node C
            tasks Sense_C, Act_C, P1C, P2C, P3C;
        }
```

```
initialise {
        ! Period for each task
        T[Sense_C] = 20;
        T[Act_C] = 20;
        T[P1C] = 5;
        T[P2C] = 15;
        T[P3C] = 50;

        ! Release Jitter for each task
        J[Sense_C] = 0;
        J[Act_C] = 0;
        J[P1C] = 0;
        J[P2C] = 2;
        J[P3C] = 5;

        ! Execution time for each task
        C[Sense_C] = 2;
        C[Act_C] = 1;
        C[P1C] = 1;
        C[P2C] = 2;
        C[P3C] = 1;

        ! Priority according to Rate Monotonic, 1 = Highest priority
        P[Sense_C] = 3;
        P[Act_C] = 3;
        P[P1C] = 1;
        P[P2C] = 2;
        P[P3C] = 4;

        ! Deadline for each task
        D[Sense_C] = 40;
        D[Act_C] = 40;
        D[P1C] = 2;
        D[P2C] = 10;
        D[P3C] = 25;

        ! Blocking is zero since we use no semaphores and RM-scheduling
        B = 0;
}

formulas {
        ! Act_C inherits Jitter from previously executed tasks
        J[Act_C] = Jafter4[Trans3];

        ! Calculate window of interference
        W[i] = C[i] + B + sigma(hp, ceiling((W[i]+J[j])/T[j]) * C[j]);
        ! Calculate the response-time
        R[i] = W[i] + J[i];

        ! Jitter to be inherited by Node_CAN
        Jafter1[Trans3] = R[Sense_C];

        ! Save R in global array TransactionsR
        TransactionsR[Trans3] = R[Act_C];

        ! Load Utilization
```

```
            LoadArray[i] = C[i] / T[i];
            ! Convert to percent to display as result
            LoadNode_C = 100 * (LoadArray[Sense_C] + LoadArray[Act_C] + LoadArray[P1C]
        + LoadArray[P2C] + LoadArray[P3C]);
        }
}

! ======================================================================

system Node_D{
        declarations {
                indexed T, J, C, W, R, LoadArray;
                priority P;
                scalar B;

                ! Tasks in Node D
                tasks CalcA, CalcB, CalcC;
        }

        initialise {
                ! Period for each task
                T[CalcA] = 10;
                T[CalcB] = 10;
                T[CalcC] = 20;

                ! Release Jitter for each task
                J[CalcA] = 0;
                J[CalcB] = 0;
                J[CalcC] = 0;

                ! Execution time for each task
                C[CalcA] = 1;
                C[CalcB] = 2;
                C[CalcC] = 4;

                ! Priority according to Rate Monotonic, 1 = Highest priority
                P[CalcA] = 1;
                P[CalcB] = 1;
                P[CalcC] = 2;

                ! Blocking is zero since we use no semaphores and RM-scheduling
                B = 0;
        }

        formulas {
                ! Jitter inherited from previous tasks
                J[CalcA] = Jafter2[Trans1];
                J[CalcB] = Jafter2[Trans2];
                J[CalcC] = Jafter2[Trans3];

                ! Calculate window of interference
                W[i] = C[i] + B + sigma(hp, ceiling((W[i]+J[j])/T[j]) * C[j]);
                ! Calculate the response-time
                R[i] = W[i] + J[i];

                ! Store response-time as jitter to be inherited in the next
```

```
        step of transaction
                Jafter3[Trans1] = R[CalcA];
                Jafter3[Trans2] = R[CalcB];
                Jafter3[Trans3] = R[CalcC];

                ! Load (utilization)
                LoadArray[i] = C[i] / T[i];
                ! Convert to percent to display as result
                LoadNode_D = 100 * (LoadArray[CalcA] + LoadArray[CalcB] + LoadArray[CalcC]);
        }
}

! =======================================================================

system Node_CAN {
        declarations {
                indexed S, R, W, T, J, C, MaxBitSize, LoadArray;
                scalar bps, tau, B;
                priority P;

                ! "Tasks" in CANBUS
                tasks S_senseA, S_actA, S_senseB, S_actB, S_senseC, S_actC;

        }

        initialise {
                ! Data size for all messages (bytes)
        S[S_senseA] = 2;
        S[S_actA] = 1;
        S[S_senseB] = 4;
        S[S_actB] = 2;
        S[S_senseC] = 4;
        S[S_actC] = 4;

                ! Period of the messages
                T[S_senseA] = 10;
                T[S_actA] = 10;
                T[S_senseB] = 10;
                T[S_actB] = 10;
                T[S_senseC] = 20;
                T[S_actC] = 20;

                ! Priority of messages, 1 = Highest priority
                P[S_senseA] = 1;
                P[S_actA] = 1;
                P[S_senseB] = 1;
                P[S_actB] = 1;
                P[S_senseC] = 2;
                P[S_actC] = 2;

                ! Blocking is zero according to assignment description (zero blocking
        from lower priority messages)
                B = 0;

                ! CAN transmission speed (bits per second)
                bps = 75000; ! 75kps
```

```
}

    formulas {
            ! tau is how how many milliseconds it takes to send one bit
            tau = 1/(bps/1000);

            ! Jitter to be inherited by CAN-BUS from node A, B and C
            J[S_senseA] = Jafter1[Trans1];
            J[S_senseB] = Jafter1[Trans2];
            J[S_senseC] = Jafter1[Trans3];

            ! Jitter inherited from node D
            J[S_actA] = Jafter3[Trans1];
            J[S_actB] = Jafter3[Trans2];
            J[S_actC] = Jafter3[Trans3];

            ! MaxBitSize, including stuff-bits (fromula from lecture 6 p.91)
            MaxBitSize[i] = 47 + S[i] * 8 + floor((34 + S[i] * 8 - 1) / 4);

            ! Assign each bit sizes to global variales (to be able to print in the end)
            MBS_S_senseA = MaxBitSize[S_senseA];
            MBS_S_senseB = MaxBitSize[S_senseB];
            MBS_S_senseC = MaxBitSize[S_senseC];
            MBS_S_actA = MaxBitSize[S_actA];
            MBS_S_actB = MaxBitSize[S_actB];
            MBS_S_actC = MaxBitSize[S_actC];

            ! Transmission time for each message
            C[i] = MaxBitSize[i] * tau;

            ! Assign each transmission time to global
    variales (to be able to print in the end)
            Trans_t_S_senseA = C[S_senseA];
            Trans_t_S_senseB = C[S_senseB];
            Trans_t_S_senseC = C[S_senseC];
            Trans_t_S_actA = C[S_actA];
            Trans_t_S_actB = C[S_actB];
            Trans_t_S_actC = C[S_actC];

            ! Calculate window of interference, C[i] not included because of
    non-preemptive transmission
            W[i] = B + sigma(hp, ceiling((W[i]+J[j]+tau)/T[j]) * C[j]);
            ! Calculate the response-time, C[i] is added here for each transmission
            R[i] = C[i] + W[i] + J[i];

            ! Jitter after CAN to be inherited by Node D
            Jafter2[Trans1] = R[S_senseA];
            Jafter2[Trans2] = R[S_senseB];
            Jafter2[Trans3] = R[S_senseC];

            ! Jitter after CAN to be inherited by Node A, B and C
            Jafter4[Trans1] = R[S_actA];
            Jafter4[Trans2] = R[S_actB];
            Jafter4[Trans3] = R[S_actC];

            ! Load (utilization)
```

```
                LoadArray[i] = C[i] / T[i];
                ! Convert to percent to display as result
                LoadNode_CAN = 100 * (LoadArray[S_senseA] + LoadArray[S_senseB] + LoadArray[S_
        + LoadArray[S_actA] + LoadArray[S_actB] + LoadArray[S_actC]);
        }
}

! ===========================================================================

system global {
  ! Declare a variable R with the same structure as the global TransactionsR
  declarations {
        scalar
                Maxbitsize_senseA, Maxbitsize_senseB, Maxbitsize_senseC, Maxbitsize_actA,
        Maxbitsize_actB, Maxbitsize_actC;
        scalar
                Transmissiontime_in_ms_senseA, Transmissiontime_in_ms_senseB,
        Transmissiontime_in_ms_senseC, Transmissiontime_in_ms_actA,
        Transmissiontime_in_ms_actB, Transmissiontime_in_ms_actC;
        scalar
                LoadOnNodeA, LoadOnNodeB, LoadOnNodeC, LoadOnNodeD, LoadOnCAN;
        indexed
                MSB;
        indexed
                LocalResponseTimeNodeD;
        indexed
                CANbusy_period;
    indexed
                R_in_ms;
    tasks
                Trans1, Trans2, Trans3;
  }

        ! Initialise global variables
        initialise {
    TransactionsR[i] = 0;
    Jafter1[i] = 0;
    Jafter2[i] = 0;
        Jafter3[i] = 0;
        Jafter4[i] = 0;
  }

  formulas {
    ! This copying is necessary to print the final values

        ! Print max bit sizes for each message
        Maxbitsize_senseA = MBS_S_senseA;
        Maxbitsize_senseB = MBS_S_senseB;
        Maxbitsize_senseC = MBS_S_senseC;
        Maxbitsize_actA = MBS_S_actA;
        Maxbitsize_actB = MBS_S_actB;
        Maxbitsize_actC = MBS_S_actC;

        ! Print transmissiontime for each CAN message
        Transmissiontime_in_ms_senseA = Trans_t_S_senseA;
        Transmissiontime_in_ms_senseB = Trans_t_S_senseB;
```

```
        Transmissiontime_in_ms_senseC = Trans_t_S_senseC;
        Transmissiontime_in_ms_actA = Trans_t_S_actA;
        Transmissiontime_in_ms_actB = Trans_t_S_actB;
        Transmissiontime_in_ms_actC = Trans_t_S_actC;

        ! Print the utilizations
        LoadOnNodeA = LoadNode_A;
        LoadOnNodeB = LoadNode_B;
        LoadOnNodeC = LoadNode_C;
        LoadOnNodeD = LoadNode_D;
        LoadOnCAN = LoadNode_CAN;

        ! Print response-time for each transaction
        R_in_ms[i] = TransactionsR[i];

        ! Print
        LocalResponseTimeNodeD[i] = Jafter3[i] - Jafter2[i];

        CANbusy_period[i] = (Jafter2[i] - Jafter1[i]) + (Jafter4[i] - Jafter3[i]);
    }
}
```

The results printed from the code above:

```
System 'Node_A'
------------------

J[Sense_A] = 0.000000
J[Act_A] = 4.866667
J[P1A] = 0.000000
J[P2A] = 2.000000
J[P3A] = 5.000000

W[Sense_A] = 2.000000
W[Act_A] = 2.000000
W[P1A] = 1.000000
W[P2A] = 5.000000
W[P3A] = 8.000000

R[Sense_A] = 2.000000
R[Act_A] = 6.866667
R[P1A] = 1.000000
R[P2A] = 7.000000
R[P3A] = 13.000000

Jafter1[Trans1] = 2.000000
Jafter1[Trans2] = 2.000000
Jafter1[Trans3] = 5.000000

TransactionsR[Trans1] = 6.866667
TransactionsR[Trans2] = 8.266667
TransactionsR[Trans3] = 29.800000

LoadArray[Sense_A] = 0.100000
LoadArray[Act_A] = 0.100000
LoadArray[P1A] = 0.200000
LoadArray[P2A] = 0.133333
```

```
LoadArray[P3A] = 0.020000

LoadNode_A = 55.333333


System 'Node_B'
------------------

J[Sense_B] = 0.000000
J[Act_B] = 6.266667
J[P1B] = 0.000000
J[P2B] = 2.000000
J[P3B] = 5.000000

W[Sense_B] = 2.000000
W[Act_B] = 2.000000
W[P1B] = 1.000000
W[P2B] = 7.000000
W[P3B] = 8.000000

R[Sense_B] = 2.000000
R[Act_B] = 8.266667
R[P1B] = 1.000000
R[P2B] = 9.000000
R[P3B] = 13.000000

Jafter1[Trans1] = 2.000000
Jafter1[Trans2] = 2.000000
Jafter1[Trans3] = 5.000000

TransactionsR[Trans1] = 6.866667
TransactionsR[Trans2] = 8.266667
TransactionsR[Trans3] = 29.800000

LoadArray[Sense_B] = 0.100000
LoadArray[Act_B] = 0.100000
LoadArray[P1B] = 0.200000
LoadArray[P2B] = 0.133333
LoadArray[P3B] = 0.020000

LoadNode_B = 55.333333


System 'Node_C'
------------------

J[Sense_C] = 0.000000
J[Act_C] = 25.800000
J[P1C] = 0.000000
J[P2C] = 2.000000
J[P3C] = 5.000000

W[Sense_C] = 5.000000
W[Act_C] = 4.000000
W[P1C] = 1.000000
W[P2C] = 3.000000
```

W[P3C] = 9.000000

R[Sense_C] = 5.000000
R[Act_C] = 29.800000
R[P1C] = 1.000000
R[P2C] = 5.000000
R[P3C] = 14.000000

Jafter1[Trans1] = 2.000000
Jafter1[Trans2] = 2.000000
Jafter1[Trans3] = 5.000000

TransactionsR[Trans1] = 6.866667
TransactionsR[Trans2] = 8.266667
TransactionsR[Trans3] = 29.800000

LoadArray[Sense_C] = 0.100000
LoadArray[Act_C] = 0.050000
LoadArray[P1C] = 0.200000
LoadArray[P2C] = 0.133333
LoadArray[P3C] = 0.020000

LoadNode_C = 50.333333

System 'Node_D'

J[CalcA] = 3.000000
J[CalcB] = 3.266667
J[CalcC] = 10.400000

J[CalcA] = 3.000000
J[CalcB] = 3.266667
J[CalcC] = 10.400000

J[CalcA] = 3.000000
J[CalcB] = 3.266667
J[CalcC] = 10.400000

W[CalcA] = 1.000000
W[CalcB] = 2.000000
W[CalcC] = 10.000000

R[CalcA] = 4.000000
R[CalcB] = 5.266667
R[CalcC] = 20.400000

Jafter3[Trans1] = 4.000000
Jafter3[Trans2] = 5.266667
Jafter3[Trans3] = 20.400000

Jafter3[Trans1] = 4.000000
Jafter3[Trans2] = 5.266667
Jafter3[Trans3] = 20.400000

```
Jafter3[Trans1] = 4.000000
Jafter3[Trans2] = 5.266667
Jafter3[Trans3] = 20.400000

LoadArray[CalcA] = 0.100000
LoadArray[CalcB] = 0.200000
LoadArray[CalcC] = 0.200000

LoadNode_D = 50.000000


System 'Node_CAN'
-------------------

tau = 0.013333

J[S_senseA] = 2.000000
J[S_actA] = 4.000000
J[S_senseB] = 2.000000
J[S_actB] = 5.266667
J[S_senseC] = 5.000000
J[S_actC] = 20.400000

J[S_senseA] = 2.000000
J[S_actA] = 4.000000
J[S_senseB] = 2.000000
J[S_actB] = 5.266667
J[S_senseC] = 5.000000
J[S_actC] = 20.400000

J[S_senseA] = 2.000000
J[S_actA] = 4.000000
J[S_senseB] = 2.000000
J[S_actB] = 5.266667
J[S_senseC] = 5.000000
J[S_actC] = 20.400000

J[S_senseA] = 2.000000
J[S_actA] = 4.000000
J[S_senseB] = 2.000000
J[S_actB] = 5.266667
J[S_senseC] = 5.000000
J[S_actC] = 20.400000

J[S_senseA] = 2.000000
J[S_actA] = 4.000000
J[S_senseB] = 2.000000
J[S_actB] = 5.266667
J[S_senseC] = 5.000000
J[S_actC] = 20.400000

J[S_senseA] = 2.000000
J[S_actA] = 4.000000
J[S_senseB] = 2.000000
J[S_actB] = 5.266667
J[S_senseC] = 5.000000
```

J[S_actC] = 20.400000

MaxBitSize[S_senseA] = 75.000000
MaxBitSize[S_actA] = 65.000000
MaxBitSize[S_senseB] = 95.000000
MaxBitSize[S_actB] = 75.000000
MaxBitSize[S_senseC] = 95.000000
MaxBitSize[S_actC] = 95.000000

MBS_S_senseA = 75.000000

MBS_S_senseB = 95.000000

MBS_S_senseC = 95.000000

MBS_S_actA = 65.000000

MBS_S_actB = 75.000000

MBS_S_actC = 95.000000

C[S_senseA] = 1.000000
C[S_actA] = 0.866667
C[S_senseB] = 1.266667
C[S_actB] = 1.000000
C[S_senseC] = 1.266667
C[S_actC] = 1.266667

Trans_t_S_senseA = 1.000000

Trans_t_S_senseB = 1.266667

Trans_t_S_senseC = 1.266667

Trans_t_S_actA = 0.866667

Trans_t_S_actB = 1.000000

Trans_t_S_actC = 1.266667

W[S_senseA] = 0.000000
W[S_actA] = 0.000000
W[S_senseB] = 0.000000
W[S_actB] = 0.000000
W[S_senseC] = 4.133333
W[S_actC] = 4.133333

R[S_senseA] = 3.000000
R[S_actA] = 4.866667
R[S_senseB] = 3.266667
R[S_actB] = 6.266667
R[S_senseC] = 10.400000
R[S_actC] = 25.800000

Jafter2[Trans1] = 3.000000
Jafter2[Trans2] = 3.266667

```
Jafter2[Trans3] = 10.400000

Jafter2[Trans1] = 3.000000
Jafter2[Trans2] = 3.266667
Jafter2[Trans3] = 10.400000

Jafter2[Trans1] = 3.000000
Jafter2[Trans2] = 3.266667
Jafter2[Trans3] = 10.400000

Jafter4[Trans1] = 4.866667
Jafter4[Trans2] = 6.266667
Jafter4[Trans3] = 25.800000

Jafter4[Trans1] = 4.866667
Jafter4[Trans2] = 6.266667
Jafter4[Trans3] = 25.800000

Jafter4[Trans1] = 4.866667
Jafter4[Trans2] = 6.266667
Jafter4[Trans3] = 25.800000

LoadArray[S_senseA] = 0.100000
LoadArray[S_actA] = 0.086667
LoadArray[S_senseB] = 0.126667
LoadArray[S_actB] = 0.100000
LoadArray[S_senseC] = 0.063333
LoadArray[S_actC] = 0.063333

LoadNode_CAN = 54.000000


System 'global'
-------------------

Maxbitsize_senseA = 75.000000

Maxbitsize_senseB = 95.000000

Maxbitsize_senseC = 95.000000

Maxbitsize_actA = 65.000000

Maxbitsize_actB = 75.000000

Maxbitsize_actC = 95.000000

Transmissiontime_in_ms_senseA = 1.000000

Transmissiontime_in_ms_senseB = 1.266667

Transmissiontime_in_ms_senseC = 1.266667

Transmissiontime_in_ms_actA = 0.866667

Transmissiontime_in_ms_actB = 1.000000
```

```
Transmissiontime_in_ms_actC = 1.266667

LoadOnNodeA = 55.333333

LoadOnNodeB = 55.333333

LoadOnNodeC = 50.333333

LoadOnNodeD = 50.000000

LoadOnCAN = 54.000000

R_in_ms[Trans1] = 6.866667
R_in_ms[Trans2] = 8.266667
R_in_ms[Trans3] = 29.800000

LocalResponseTimeNodeD[Trans1] = 1.000000
LocalResponseTimeNodeD[Trans2] = 2.000000
LocalResponseTimeNodeD[Trans3] = 10.000000

CANbusy_period[Trans1] = 1.866667
CANbusy_period[Trans2] = 2.266667
CANbusy_period[Trans3] = 10.800000
```