

Control over CAN and Flexray  
**Embedded Control Systems**

Sai Krishna Kalluri & Snorri Stefansson

April 2, 2017

# Contents

<b>Introduction</b>	<b>2</b>
<b>1 Part 1</b>	<b>3</b>
1.1 Introduction . . . . .	3
1.2 Response Time analysis . . . . .	3
1.2.1 Response time analysis per processing unit . . . . .	3
1.2.2 Response time analysis for the CAN bus messages . . . . .	3
1.3 System Model Derivation and Control Parameter Design . . . . .	3
1.3.1 Controller Structure . . . . .	4
1.3.2 Controller Design . . . . .	4
1.4 Design decision . . . . .	5
1.5 Results . . . . .	5
<b>2 Part 2</b>	<b>6</b>
2.1 Introduction . . . . .	6
2.2 Response Time analysis . . . . .	7
2.2.1 Response time analysis per processing unit . . . . .	7
2.2.2 Response time analysis for the CAN bus messages . . . . .	8
2.3 Optimization for sensor-to-actuator delay . . . . .	9
2.4 System model . . . . .	9
2.5 Design decision . . . . .	9
2.6 Results . . . . .	9
2.7 Conclusions . . . . .	9
<b>3 Part 3</b>	<b>11</b>
3.1 Introduction . . . . .	11
3.2 Answer all the questions . . . . .	11
3.2.1 Theoretical analysis versus actual implementation . . . . .	11
3.3 Design decision . . . . .	11
3.4 Results . . . . .	11
3.5 Conclusions . . . . .	11
3.6 Results . . . . .	11
3.7 Conclusion . . . . .	11

# Introduction

# Chapter 1

## Part 1

### 1.1 Introduction

### 1.2 Response Time analysis

#### 1.2.1 Response time analysis per processing unit

Table 1.1: By running the Matlab script ResponsetimeAnylnsis\_FPP.m with the different parameters given for PU1 and PU2 these response times are obtained for each of the tasks. These files are then delivered as PU1.m PU2.m

PU1	$T_1$	$T_2$	$T_3$	$T_4 (T_s)$
Matlab (ms)	0.1	2.1	4.1	7.2

PU2	$T_5$	$T_6$	$T_7$	$T_8$
Matlab (ms)	6	3	9	5

#### 1.2.2 Response time analysis for the CAN bus messages

Table 1.2: Response times for the CAN bus messages

CAN	$m_2$	$m_1$	$m_3$	$m_8$
Matlab (ms)	2	3	4	4

### 1.3 System Model Derivation and Control Parameter Design

The objectives for the controller is to properly control the given system with a set of design parameters. These parameters, shown in Table

There are namely a number of steps taken until all parameters can be considered to satisfy the performance constraints of the controller. These basic steps can be seen in the following list.

1. Step 1: Derive the system model. Determine the A, B and C matrices in relation to all constants by hand calculations. We are already provided with the matrices in assignment description, hence no need to derive. Insert this into Matlab.

2. Step 2: Derive the values for sampling period and sensor-to-actuator delay based on our system design.
3. Step 3: Choosing desired poles according to the given requirements. Verifying the controllability of the system for the choosen values. Computing the controller Feed-Forward and Feed-Back gains  $F$  and  $K$ . Inserting these calculations into MATLAB. More about this step can be found in Section
4. Step4: Design  $K$  and  $F$  values, compute current input activation( $u$ ) and outputs( $x$ ) from equations, insert these calculations to MATLAB and simulate the system.
5. Step5: Apply multi-objective genetic algorithm on the above system, with pole positions as parameters, and settling-time and maximum input voltage as our objectives. Obtain pareto optimal points satisfying design constraints.

Table 1.3: Constants and design parameters referenced to in calculations

Symbol	Description	Value	Unit
$\theta$	Rotor position	-	rad
$i_m$	Motor current	-	A
$J$	Inertia	$3.2284 \cdot 10^{-6}$	$Kgm^2$
$b$	Friction coefficient	$3.5077 \cdot 10^{-6}$	Nms/rad
$R$	Armature Resistance	4	Ohm
$L$	Inductance of Motor Winding	$2.75 \cdot 10^{-6}$	H
$K_m$	Motor constant	0.0274	Nm/A
$K$	Feed-BackGain	-	-
$F$	Feed-Forward Gain	-	-

### 1.3.1 Controller Structure

The system can be described with the second order differential equation shown in Equation 1.1 as given in the assignment description. Equation 1.2 and Equation 1.3 describe the statespace representation of our system. From these equations the Feedback- and Feedforward gain can be determined in relations to the constants in the system.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \theta \\ \dot{\theta} \\ i \end{bmatrix} = \begin{bmatrix} \theta \\ \omega \\ i \end{bmatrix} \quad \text{and} \quad \dot{\mathbf{x}} = \begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \\ \dot{i} \end{bmatrix} \quad (1.1)$$

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u \quad \text{and} \quad y = \mathbf{C}\mathbf{x} \quad \text{where} \quad \text{input: } u = V \quad , \quad \text{output: } y = i \quad (1.2)$$

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & \frac{-b}{J} & \frac{K_m}{J} \\ 0 & \frac{-K_m}{L} & \frac{-R}{L} \end{bmatrix} \begin{bmatrix} \theta \\ \omega \\ i \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{L} \end{bmatrix} V \quad \text{and} \quad y = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \omega \\ i \end{bmatrix} \quad (1.3)$$

### 1.3.2 Controller Design

In this subsection we discuss the method for designing the controller design parameters FeedbackGain- $K$  and FeedForwardGain- $F$ . In this problem we have the scenario where  $D_c < h$ . Since our controller operates in discrete sample time, we start with converting our continuous system as described in equation 1.4 into discrete domain. Therefore on applying ZOH sampling with period  $h_c$  and constant sensor-actuator Delay  $D_c$  we achieve the equation 1.5 for discrete sample time system. From 1.5 we can notice that the next output not only depends on current input but also on previous input. Hence, we simplify the system and get it into standard form representing equation 1.6 by invoking equation1.7. After applying above simplification input

$u[k]$  can be represented in terms of controller gains using equation 1.8. and matrices  $\Phi, C$  are converted to corresponding augmented matrices  $\Phi_{aug}, C_{aug}$ , whereas  $\Gamma_1, \Gamma_2$  are converted to single augmented matrix  $\Gamma_{aug}$ .

$$\dot{x} = Ax + Bu \quad \text{and} \quad y = Cx \quad (1.4)$$

$$x[k+1] = \phi x[k] + \Gamma_1(D_c)u[k-1] + \Gamma_0(D_c)u[k] \quad \text{and} \quad y[k] = Cx[k] \quad (1.5)$$

$$x[k+1] = \phi z[k] + \Gamma_{aug}u[k] \quad \text{and} \quad y[k] = C_{aug}z[k] \quad (1.6)$$

$$z[k] = \begin{bmatrix} x[k] \\ u[k-1] \end{bmatrix} \quad (1.7)$$

$$u[k] = Kz[k] + Fr \quad (1.8)$$

$$K = -[0 \ 0 \ \dots \ 1] \gamma_{aug}^{-1} H(\phi_{aug}) \quad (1.9)$$

$$F = \frac{1}{C_{aug}(I - \phi_{aug} - \Gamma_{aug}K)^{-1}\Gamma_{aug}} \quad (1.10)$$

Before deriving the controller gains it is important to verify if the system is controllable under given configuration. This can be verified by calculating  $\det(\gamma_{aug})$ . If the determinant is not equal to 0, then system is controllable. After verifying the controllability of the system  $K$  can be derived by applying Ackermanns formula described in equation 1.9 and  $F$  can be derived from equation 1.10. However, one can notice the matrix  $H(\Phi_{aug})$  depends on the desired poles which in turn depends on the design requirements. The desired poles alter the frequency spectrum of the transfer function of the system and play a significant role in controlling the behaviour of output parameters of the system. Therefore various design requirements such as Overshoot, settling time, boundary ranges of the parameters in the system depends on the desired poles and thus in turn influence controller gains  $K$  and  $F$ . Following design, work flow has been developed in MATLAB to derive pareto optimal points (pole locations) satisfying design constraints using multi-objective genetic algorithm and one of the optimal point is selected for simulation.

## 1.4 Design decision

## 1.5 Results

Firstly: Response time analysis

Secondly: Control system input and output

# Chapter 2

## Part 2

### 2.1 Introduction

Analyzing and confirming broad analysis done on paper and in Matlab allows us to confirm the hypothesis made about the behavior of the CAN bus and its specific tasks from Part 1. This analyses is done in Inchron, which allows us to explore the real-time behavior of this embedded system in full detail. Some exploration on how the system should work has already been done in Part 1, [Add Table reference to periods and priorities](#), this is used here by feeding the settings of the embedded system to the tool. The settings included is the hierarchy of the Processing Units (PUs) and their tasks which each have different periods, execution time and priority. The tree view, Figure 2.1, shows the details of the hierarchy from the Inchron's perspective.

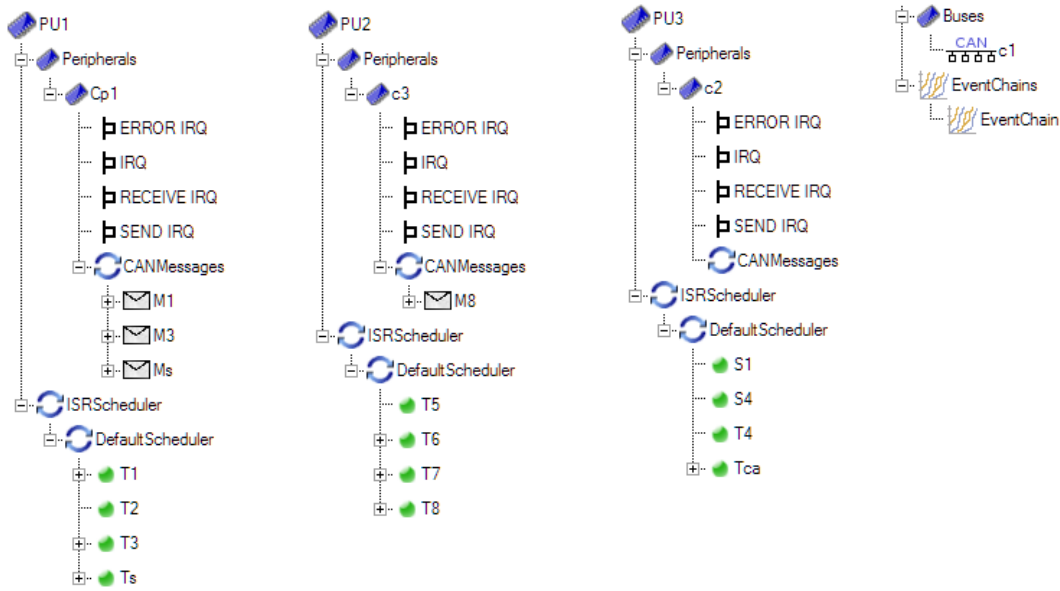


Figure 2.1: This shows the coarse grain hierarchy of the system ported into Inchron for verification and simulation of the real time components of our embedded system.

## 2.2 Response Time analysis

### 2.2.1 Response time analysis per processing unit

For this analysis all details have to be imported to Inchron as stated earlier. Now paying special attention to the messages which transfer the packets between controllers which make a full system.

When validating and simulating the model with the settings mentioned in Part 1 and Figure 2.1 and 2.5 we can observe the response times for each task when inserting the Worst Case Execution Time into the tool. Next each processing unit and their Worst Case Response Time (WCRT) will be investigated and then compared to the Matlab model (Response Time analysis). The resulting figures were obtained after several tries with various settings until the correct configuration of the tool was found, e.g. setting the schedule as preemptive was not set as it was not found in the first try.

#### PU1

For PU1 a fixed preemptive priority scheme with four tasks T1,T2,T3 and Ts. Now when validating the model, Figure 2.2 is obtained showing the results and they are compared to the Matlab response times in Table 2.1.

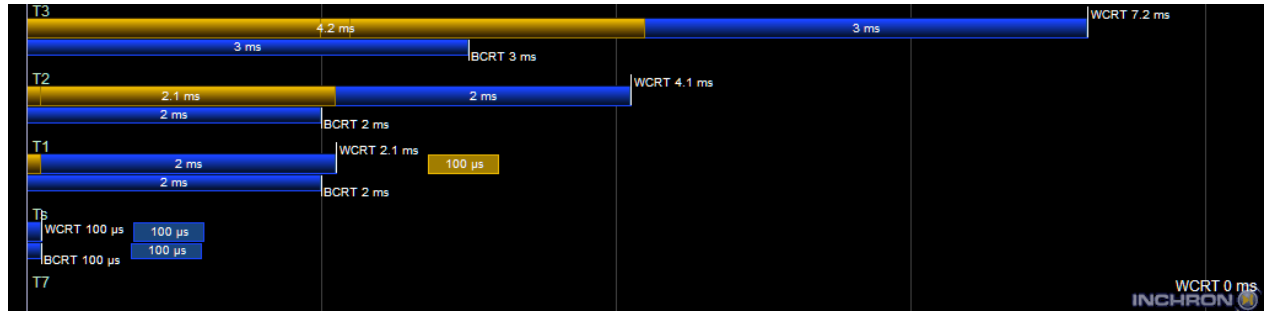


Figure 2.2: Showing the WCRT analysis for PU1. The Results can be read from each horizontal bar.

#### PU2

Now PU2, has the same sheme as PU1, fixed preemptive priority, with four tasks T5,T6,T7 and T8. Now when validating the model, Figure 2.3 is obtained showing the results and they are compared to the Matlab response times in Table 2.1.

#### PU3

Although PU3 has a time division multiplexing (TDM) its tasks will still be analyzed here, shown in Figure 2.4. It is important to state that T4 is not a real-time task as it is sending a message *out to the blue* but  $T_{ca}$  is an important task, actuating on the sensor value and completing the sensor to actuator delay.





Figure 2.3: Showing the WCRT analysis for PU2. The Results can be read from each horizontal bar.

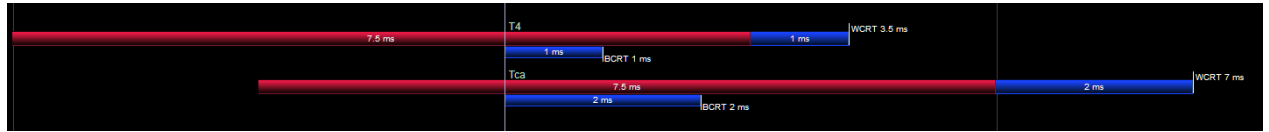


Figure 2.4: Showing the WCRT analysis for PU3 (TDM). The Results can be read from each horizontal bar

Table 2.1: By running the Matlab script `ResponsetimeAnylsis.FPP.m` with the different parameters given for PU1 and PU2 these response times are obtained. These files are then delivered as PU1.m and PU2.m

PU1	$T_1$	$T_2$	$T_3$	$T_4 (T_s)$
Matlab (ms)	0.1	2.1	4.1	7.2
Inchron (ms)	0.1	2.1	4.1	7.2

PU2	$T_5$	$T_6$	$T_7$	$T_8$
Matlab (ms)	6	3	9	5
Inchron (ms)	6	3	9	5

### 2.2.2 Response time analysis for the CAN bus messages

PU1 and PU2 are the only units within the system that are sending messages, shown for clarity in Figure 2.5, but PU3 contains the computing and actuating task which will receive the  $m_s$  message and mark the end of the sensor to actuator delay.

Table 2.2: CAN messages of the system compared from the Inchron tool suite and Matlab. Showing identical results

CAN	$m_2$	$m_1$	$m_3$	$m_8$
Matlab ms	2	3	4	4
Inchron	2	3	4	4

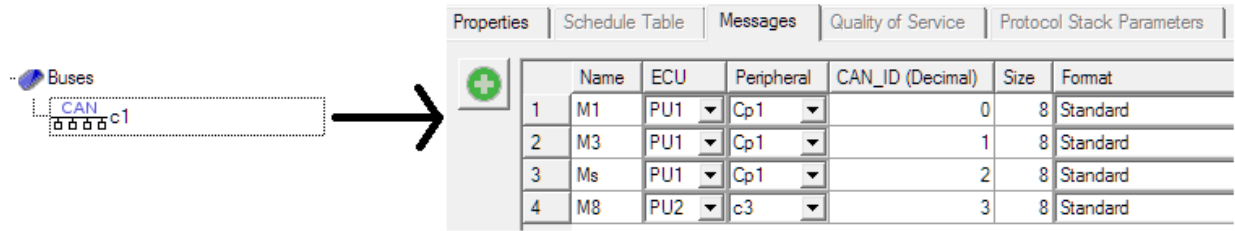


Figure 2.5: All can messages in the system, indicating PU source, individual CAN id and message size in bytes

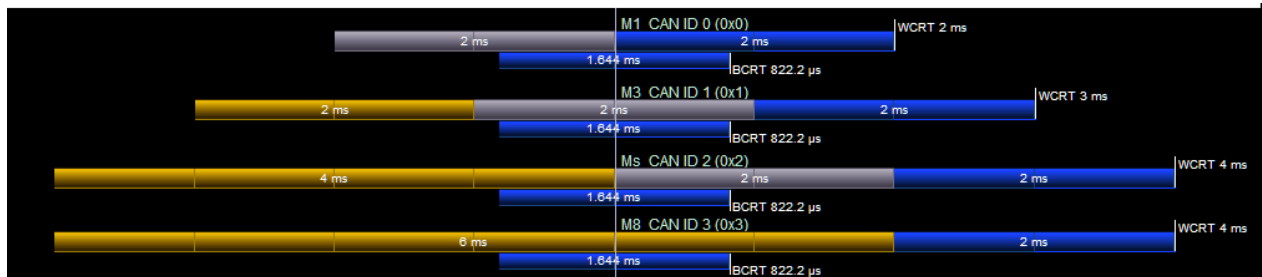


Figure 2.6: Showing the WCRT analysis for the CAN messages. The Results can be read from each horizontal bar and is compared in Table 2.2.

## 2.3 Optimization for sensor-to-actuator delay

Now optimizing the sensor-to-actuator delay requires running the CrhonOpt tool within Inchron and setting up objectives for the real time requirements. That will be set to target: event-chain, which is set to be smaller or equal to 5ms.

The initial priorities were selected according to period and execution time, shown in Figure 2.7 on the left, but when running the simulation to enhance the sensor to actuator delay, the tool changed the priorities in PU2, shown in Figure 2.7 on the right. This showed to be an improvement although this did not change the sensor to actuator delay.

## 2.4 System model

## 2.5 Design decision

## 2.6 Results

Firstly: Response time analysis

Secondly: Plots from chronVIEW (before and after optimization)

Last: Control system input and output

## 2.7 Conclusions

```

%% period, deadline, wcet and priority

%PU1
T = [5, 10, .1, 1;
     10, 1, 2, 2;
     15, 1, 2, 3;
     20, 5, 3, 4;
     ];

%PU2
T = [15, 10, 1, 3;
     10, 1, 3, 2;
     20, 1, 3, 4;
     10, 5, 2, 1;
     ];

Task Inchron Priorities
% Ts 4
% T1 3
% T2 2
% T3 1

Task Inchron Priorities
% T5 2
% T6 3
% T7 1
% T8 4

```

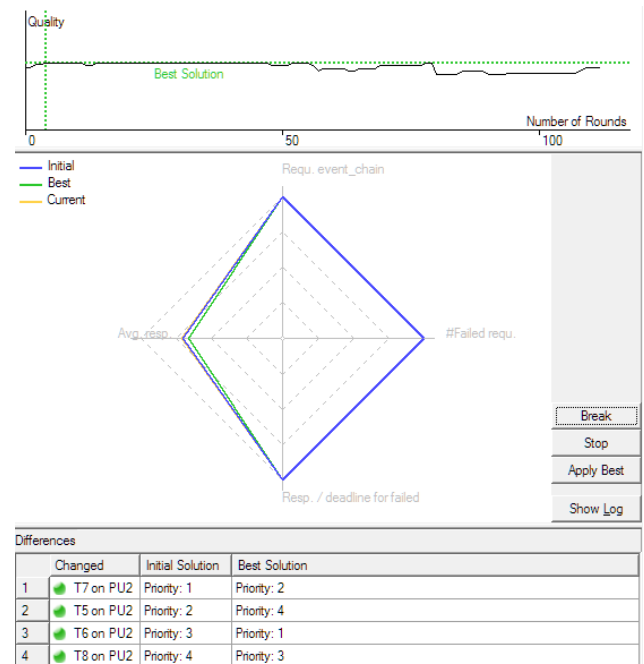


Figure 2.7: On the left, initial priorities used in the Matlab script and on the right the ChronOPT optimization of these parameters.

# Chapter 3

## Part 3

### 3.1 Introduction

### 3.2 Answer all the questions

#### 3.2.1 Theoretical analysis versus actual implementation

### 3.3 Design decision

### 3.4 Results

Firstly: Solution to the design problem. (Include the parameters you have chosen)  
Secondly: from chronVIEW for your design

### 3.5 Conclusions

### 3.6 Results

### 3.7 Conclusion