Implementing control algorithms for platooning based on V2V communication

JOAKIM KJELLBERG

Master’s Degree Project
Stockholm, Sweden April 2011

XR-EE-RT 2011:008
Abstract

Platooning allows vehicles to drive in close longitudinal convoys to mitigate traffic congestion. With fully autonomous systems, the drivers would become passengers in their own cars and they would have more time to focus on other tasks than driving. With vehicle-to-vehicle communication small inter-vehicle distances can be achieved and energy can be saved.

In 2011, the two organizations High Tech Automotive Systems and TNO are hosting a competition in platooning called Grand Cooperative Driving Challenge (GCDC), where eleven teams will compete in a set of predefined traffic scenarios. KTH and Scania are collaborating in a project called Scoop to participate in GCDC with a Scania truck. The team consists of 12 members, where 7 are master’s students.

The task in this thesis is to develop a framework for implementing control strategies within the Scoop project. To control the speed of the truck, the controllers will ask for an acceleration which will then be sent to either the engine or the brakes. The controllers are designed by another master’s student, but which controller to use at a certain point in time is decided in this thesis.

To limit the scenarios that might occur, the system is only able to use up to three vehicles ahead as reference: the two vehicles directly ahead, and the platoon leader. This limitation was made because a certain scenario requires a certain controller configuration.

Before the entire Cooperative Driving System (CDS) – developed by Scoop – was finished, the tests had to be carried out with a standalone software developed in this thesis. It simulates a platoon of three vehicles ahead of the real truck and packages all the information in the same way that the CDS will do before sending it to the control unit.

Results show that the framework behaves as intended. It manages to join and split a platoon and also select the correct controller configuration depending on the current conditions.
Sammanfattning

Att köra i fordonstag innebär att man skapar longitudinella konvojer för att undvika trafikstockning. Med helt autonoma system skulle förarna bli passagerare i sina egna bilar och de skulle få mer tid till att fokusera på andra uppgifter än just bilkörning. Med trådlös kommunikation mellan varje fordon kan man hålla ner avständen i tåget och på så vis spara energi.

Under år 2011 står de två organisationerna High Tech Automotive Systems och TNO värderar för en tävling i fordonstagkörning som kallas Grand Cooperative Driving Challenge (GCDC), där elva lag kommer att tävla i ett antal fördefinierade trafikscenarion. KTH och Scania samarbetar i ett projekt vid namn Scoop för att delta i GCDC med en Scania lastbil. Laget består av 12 medlemmar, varav 7 stycken är mastersstudenter.

Uppgiften i det här examensarbetet är att utveckla ett ramverk för implementation av reglerstrategier inom ramarna för Scoop-projektet. För att styra lastbilens fart kommer regulatorerna att begära en acceleration som sedan skickas aningen till motorn eller till bromsarna. Regulatorerna är framtagna av en annan mastersstudent, men vilken regulator som ska användas vid ett visst tillfälle bestäms i det här arbetet.

För att begränsa de scenarion som kan uppstå, så kan systemet bara använda upp till tre framförvarande fordon som referens: de två direkt framför, samt ledarfordonet. Denna begränsning gjordes pga. att ett visst scenario kräver en viss regulatorkonfiguration.

Innan hela Cooperative Driving System (CDS) – utvecklat av Scoop – var färdigställt, så utfördes testerna med en fristående mjukvara utvecklad i det här examensarbetet. Den simulerar ett tåg bestående av tre fordon framför den riktiga lastbilen och paketerar all information på samma sätt som CDS kommer att göra innan den skickas till styrenheten.

Resultat visar att ramverket beter sig enligt förväntningarna. Den klarar av både att ansluta till ett fordonstag och att koppla ifrån det, samt att välja rätt regulatorkonfiguration efter rådande omständigheter.
Acknowledgments

I would like to thank all members of the Scoop project: Elin Stålklinga, Dennis Sundman, Liliana Garcia, Mattias Björk, Mohammadreza Khaksari, Muhammad Altamash Ahmed Khan, Rickard Lyberger, Sagar Behere and Simon Pettersson. Without you it would have been impossible to realize this project.

And a special thanks to my mentors Henrik Pettersson at Scania and Jonas Mårtensson at KTH, for all the support and feedback that you have given me during the project.

I love LaTeX.
# Contents

1 **Introduction**  
1.1 Platooning ........................................ 1  
1.1.1 Platooning strategies ............................ 2  
1.2 Grand Cooperative Driving Challenge ............. 3  
1.2.1 Rules ............................................. 3  
1.2.2 Technology ...................................... 4  
1.2.3 Vehicle requirements ............................ 5  
1.3 The Scoop project .................................. 5  
1.4 Problem formulation ................................ 6  

2 **System Architecture and Vehicle Models**  
2.1 System architecture .................................. 7  
2.2 Controller Area Network ............................ 7  
2.3 Sensor data .......................................... 9  
2.4 CAN messages ....................................... 9  
2.4.1 Examples of CAN message design ............... 11  
2.5 Model of the truck .................................. 11  
2.6 Modeling the vehicles ahead ....................... 12  
2.7 Traffic light model .................................. 13  

3 **Platooning Coordination and Vehicle Control**  
3.1 Requirements and limitations ....................... 14  
3.2 Controlling the truck ............................... 15  
3.2.1 Tuning the brake request ....................... 16  
3.2.2 Preparing for controller implementation ........ 16  
3.3 Different controller configurations ............... 17  
3.3.1 Control strategies ............................... 18  
3.4 Platooning logic .................................... 18  
3.5 Traffic lights ....................................... 19  

4 **Controller Implementation**  
4.1 Software ............................................. 23  
4.2 Controller layout .................................... 23  
4.3 On/Off .............................................. 24  
4.4 Changing controller configuration ............... 26  
4.5 Stopping at a traffic light......................... 27
4.6 Platooning logic ........................................... 27
4.7 Error handling ........................................... 28
  4.7.1 Loss of critical components ......................... 28

5 Testing and Results ....................................... 32
  5.1 Platoon simulation for standalone testing ............ 32
  5.2 Deactivate controller on driver demand ............... 33
  5.3 Go to speed limit at green light ......................... 33
  5.4 Go from green light and catch up with a platoon .... 34
  5.5 Follow three vehicles .................................. 35
    5.5.1 In simulations .................................... 35
    5.5.2 In reality .......................................... 35
  5.6 Stop at a red light .................................... 35

6 Discussion and Conclusions ............................... 40
  6.1 Conclusions ........................................... 40
  6.2 Discussion ............................................. 41
  6.3 Future work ............................................ 41

7 Bibliography ................................................. 43

A CAN Specifications for ECU and WSU .................. A
  A.1 Platooning_state ....................................... A
  A.2 ECU_status_and_speed_intent ......................... B
  A.3 Platoon_and_system_information ...................... C
  A.4 Platooning_input ...................................... D
  A.5 Road_side_unit_information ............................ E
  A.6 VehicleAhead .......................................... F
  A.7 Vehicle1_information1 ................................. G
  A.8 Vehicle1_information2 ................................ H
  A.9 Vehicle2_information1 through Vehicle10_information2 . I
1 Introduction

Platooning and intelligent transportation systems are popular areas of research today. KTH and Scania are working together to enter the Grand Cooperative Driving Challenge held in the Netherlands in 2011. This report describes one very important part in this collaborative project – namely to implement control algorithms that will control the truck.

1.1 Platooning

Platooning allows vehicles to drive in close longitudinal convoys which will improve the traffic throughput. Arranging the vehicles in such a manner would reduce the fleet’s energy consumption significantly. However, the energy reduction for a single vehicle varies a lot with the inter-vehicle spacing (around 10–20 % for a truck at 70 km/h with 5–15 m spacing)[1, 2]. Traveling in platoons like this could also help the drivers to free up time, since they no longer focus on driving. An illustrative example of that is shown in Figure 1.1.

Maintaining a small inter-vehicle distance, without compromising the safety, can be achieved by using e.g. radar or wireless communication[3, p. 5]. The latter case is known as Vehicle-to-Vehicle (V2V) communication and allows the vehicles to share their current speed, acceleration, position and much more. This information can then be used to control the distance to the vehicle ahead.

Figure 1.1: Example of how platooning helps to free up time since the vehicles drive autonomously. Illustration from www.sartre-project.eu, February 2011
Infrastructure, such as traffic lights, can also be equipped with wireless transceivers. This enables Vehicle-to-Infrastructure (V2I) communication and could be used to send traffic light information. It could also inform the driver about the current traffic situation or even about the weather. For example, the project INTERSAFE uses V2I communication to reduce the number of fatal collisions at intersections[4, p. 1].

1.1.1 Platooning strategies

In platooning, from a control theory point-of-view, it is important to keep the vehicles from oscillating. Below are some short examples of how other projects have chosen to solve the control part of platooning.

String stability is when changes in vehicle speed are not amplified to the following vehicles. In [5], adaptive cruise control (ACC) is combined with V2V communication and significant improvement of the string stability is achieved. This is done by using previously recorded data from three manually driven cars and designing a model predictive control algorithm for a fourth, simulated vehicle. The setup is shown in Figure 1.2. The model predictive control had two primary objectives: prevent the distance to the vehicle ahead from becoming too close or too distant and to suppress the relative speed of all vehicles ahead to zero.

In [6], however, a platoon with three vehicles is modeled as one dynamic system. The input to the cars is acceleration, and one key issue is that the designed linear quadratic controller is used in all three vehicles. Nonetheless, it is shown that by getting information about every vehicle’s state, the platoon becomes string stable. It is also much more robust to disturbances compared to if the controller would only have access to the state of the vehicle ahead.

The SARTRE project aims to develop systems that enable platooning where the lead vehicle is driven by a professional driver on unmodified public roads. Since the follower vehicles in the platoon have both longitudinal and lateral automatic control, the drivers will have time to work or relax[7, p. 12].
Interestingly, many platooning strategies out there are based on the fact that all vehicles in a platoon are controlled using the same controller. Furthermore, all the vehicles are assumed to be the same, i.e. they have the same dynamics. None of these two assumptions can be made in this thesis, where both controllers and dynamics will vary considerably for the participating vehicles.

1.2 Grand Cooperative Driving Challenge

High Tech Automotive Systems[8] and TNO[9] have initiated what they call Grand Cooperative Driving Challenge[10] (GCDC). It is a competition arranged in the Netherlands that will be held in May, 2011.

Ten teams from six different countries will compete in having the most effective cooperative vehicle-infrastructure system – in a set of predefined traffic scenarios. GCDC is a unique challenge that aims to accelerate the implementation of cooperative driving systems and contribute to mitigate traffic problems worldwide[11].

1.2.1 Rules

In the competition, the participants are divided into team A and team B, with an equal number of vehicles. The two teams are lined up on a highway with team A on the left lane and team B on the right. The vehicles’ positions in the line-up are random, but the number of vehicles in each lane is the same. Winners of a heat is the team that crosses the finish line first. However, to make scoring individual, there will be several heats with random line-ups as described above. Extra points can be awarded if the platoon is kept short and/or the length variation is minimized.

The scenario is divided in two, where the first part takes place in an urban setting starting at a red light. Each platoon has to maximize the throughput at the traffic light by accelerating in a coordinated way when the light turns green. When the platoons coming from the traffic light reaches a certain point, two other platoons further upstream are signaled to start. The tailing platoons will now be catching up with the platoons ahead and should join them smoothly. The complete urban scenario is illustrated by Figure 1.3.

The second part is a highway scenario, which starts when the urban scenario ends. Now that the two platoons are complete, a manually driven car (hosted by GCDC) accelerates with a predefined acceleration. This vehicle acts as a speed reference for both platoons. After some time, the GCDC vehicle will introduce disturbances to the platoons; disturbances that are comparable to those found in regular highway situations. It is then up to the participants to maintain a coherent platoon, without compromising the safety[12, pp. 17–21].
Chapter 1. Introduction 1.2. Grand Cooperative Driving Challenge

Figure 1.3: The complete urban GCDC scenario from start to finish. Illustrations from [12]

1.2.2 Technology

It is required that the participating vehicle is able to control its longitudinal motion, while taking the surrounding vehicles into consideration. Moreover, the steering wheel must be manually controlled by a licensed driver. The control system has to be implemented in such a way that the driver is always able to manually override the controller by pushing the brake or throttle pedal – or using the mandatory emergency button. And the system may not activate again until it is explicitly instructed to do so.

The vehicle must be equipped with actuators and sensors that enable fully automatic acceleration and braking, which entails that a manual transmission is not allowed. Since the requirement on motion data accuracy is so strict (see Table 1.1), an RTK (Real-Time Kinetics) GPS will be required. However, the use of a radar or similar is not mandatory, but allowed.

When it comes to communications architecture, both V2V and V2I is based upon IEEE 802.11p. During a GCDC outdoor safety test it will be verified that the communication range is at least 200 m under normal conditions[12, pp. 22–25].
Table 1.1: Required motion data accuracy in GCDC

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position accuracy</td>
<td>$\leq 1 \text{ m}$</td>
</tr>
<tr>
<td>Velocity accuracy</td>
<td>$\leq 0.5 \text{ m/s}$</td>
</tr>
<tr>
<td>Acceleration accuracy</td>
<td>$\leq 0.2 \text{ m/s}^2$</td>
</tr>
</tbody>
</table>

1.2.3 Vehicle requirements

For GCDC it is stated that a participating vehicle must be able to generate acceleration and deceleration within certain limits[12, p. 23]. These limits are shown in Table 1.2.

Table 1.2: Requirements on a vehicle’s acceleration in GCDC

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum acceleration, at least</td>
<td>1.5 m/s$^2$</td>
</tr>
<tr>
<td>Maximum acceleration</td>
<td>2.0 m/s$^2$</td>
</tr>
<tr>
<td>Maximum deceleration, at least</td>
<td>−4.0 m/s$^2$</td>
</tr>
<tr>
<td>Maximum deceleration</td>
<td>−4.5 m/s$^2$</td>
</tr>
</tbody>
</table>

1.3 The Scoop project

Scania and KTH are collaborating in a project called Scoop, Stockholm Cooperative Driving[13], to enter the GCDC competition. The project consists of 7 master’s students who are responsible for developing the system, plus 2 PhD students, 1 researcher and 2 Scania engineers that acts as both developers and mentors.

The system to be developed has been named Cooperative Driving System (CDS) and is described in section 2.1. Responsible for the system architecture is Mattias Björk[14] who has to take into account the demands on safety, availability and resource constraints. This is done in close cooperation with Liliana Garcia[15], who is responsible for gathering the project’s main requirements and defining the integration tests.

The two main components in the CDS is the Wireless Safety Unit (WSU) and the Electrical Control Unit (ECU). The communication protocol will be placed in the WSU, and responsible for that is Farzad Khaksari[16]. His task is to establish communication between the vehicles, while following the specifications set by GCDC. The received data is forwarded to an estimator, developed by Muhammad Altamash Ahmed Khan[17], that filters the information and uses several sources of information to create a state estimate for every vehicle in the platoon. This data is then sent to the ECU.

Inside of the ECU is where the automatic control of the truck takes place. This thesis handles the development of the control strategy framework (the colored box in Figure 2.1), which prepares the implementation of the control strategy. This strategy is
designed by Elin Stålklingsa[18] and will make the truck behave according to the rest of the platoon.

Simplified, the problem that this project is facing is illustrated by Figure 1.4. A leader has formed a platoon which other vehicles can request to join – thus becoming followers. All platooning vehicles are equipped with wireless communication and share their information (e.g. speed, position and heading) with anyone who is interested.

![Figure 1.4: A possible platooning scenario that the project needs to handle. Through V2V communication, our vehicle knows a lot about the other platoon members, e.g. the relative distances, \(d_i\), and the absolute velocities, \(v_i\)](image)

**1.4 Problem formulation**

The problem considered in this thesis is to develop a framework for implementing control strategies within the Scoop project. As mentioned in the previous section, the controllers to be implemented have been designed by E. Stålklingsa. Several controllers are needed because of the different possible scenarios that will occur, such as being a platoon leader or follower. Therefore, one objective will be to decide which control strategy to use when.

The software will be developed in Simulink and downloaded to an ECU. Vehicle and environment information will be received from the estimator in the WSU and will be used to make decisions and to control the truck.

In this project many limitations will be made; one example is that the platoon will never turn, just go in a straight line (more on this in section 3.1). Furthermore, because of the limited time for this thesis it will not be possible to implement an extensive amount of error handling. The biggest potential errors will of course be taken care of, but this thesis is mainly about putting it all together and making it work. And – hopefully – winning the competition.
2 System Architecture and Vehicle Models

The architecture of the Cooperative Driving System (in Figure 2.1) includes a data bus which is used to send information between the WSU and the ECU. The ideal case is that there is information about every vehicle in the platoon, but it must not be forgotten that data may be lost and unpredicted errors might occur.

Except for data transfer, this chapter also describes the models developed which enable simulation of the software and the control strategies.

2.1 System architecture

The Cooperative Driving System is shown in Figure 2.1, and as described in section 1.3 the two main parts in the CDS are the WSU and the ECU. The estimator receives information from every vehicle in the platoon, but also from truck’s data bus and the local GPS.

This thesis covers the colored box in Figure 2.1, i.e. the control strategy framework which contains the control strategy. A very important assumption made in this thesis is that the estimator’s filtered values are reliable, even though some wirelessly transmitted messages are lost.

2.2 Controller Area Network

There is a standardized way of sending information internally in a vehicle and it is called Controller Area Network (CAN). CAN was introduced by Robert Bosch in the late 80’s and the technology was adopted by Scania only a few years later. In 1991 Bosch published the CAN 2.0 specification which has a 29-bit message identifier and an additional 64 bits for storing data. Although CAN was mainly intended for the automotive industry, the first applications actually came from a Finnish elevator manufacturer.

An advantage with CAN, compared to the previous data bus systems, is that it allows microcontrollers and devices to communicate with each other without the need of a central bus master. Another benefit is that the message with the highest priority is granted access to the bus without any delays. This priority is included in the message identifier[19].
Today, the most commonly used CAN specifications for trucks and buses are based on the standard J1939\(^1\), which is easy to use but might be inflexible for some applications\[^{21}\]. However, in this standard it is stated that every signal within a data message must have the two extra states \textit{Error} and \textit{Not Available}, which enables an application to always be able to report the true signal status.

Since both the Scania truck and the hardware used in the Cooperative Driving System utilize CAN buses for data transfer, the platoon information has to be packaged according to the J1939 standard.

\[\text{\textit{A set of standards designed by SAE\[^{20}\]}}\]

\(^1\) A set of standards designed by SAE\[^{20}\]

Joakim Kjellberg
April 20, 2011
2.3 Sensor data

The most important signals that will be available for every vehicle in GCDC are:

- **Position**
  The vehicle’s absolute position – acquired simply by reading the GPS position. This variable is probably the most important one when it comes to following a vehicle.

- **Position accuracy**
  An estimation of the position accuracy, measured in meters. Mostly given by the GPS, but could perhaps be extracted from the position estimates.

- **Speed**
  The speed is given both by the GPS and the vehicle’s tachometer. Sensor fusion may be applied to use both signals to get closer to the true value.

- **Longitudinal acceleration**
  The absolute acceleration of the vehicle might be hard to measure with sensors (since accelerometers are known to be quite noisy), but it could be estimated by deriving the speed with respect to time.

- **Heading**
  The heading is the direction in which the vehicle is pointing, but it might be hard to estimate it at standstill unless a compass is used. This value is not very critical in the GCDC scenarios described earlier, but it is highly important if one wants to implement autonomous steering.

- **Yaw rate**
  Just as the acceleration, the yaw rate (change in heading) is hard to measure accurately, but one could use a gyro and combine it with derivation of the heading with respect to time.

The raw data of these signals will be processed in the estimator (described in section 1.3 and shown in Figure 2.1) before they reach the ECU. But in order for them to reach the controller, they first have to be packaged into CAN messages, which is described below.

2.4 CAN messages

When sending the information from the estimator in the WSU to the ECU, it has to be packaged into CAN messages. To decide how many bits a signal should consist of and which resolution it needs, one has to look at what the information will be used for and which values that are relevant.
The estimated values are assumed to be of good quality, but this might not always be the case. Therefore, it is always important to include the values Error and Not Available in every signal. These two status flags can be used to warn the controller when data was not estimated correctly. However, if that is not enough, the signal Position accuracy will be included for every vehicle. This signal contains information about how ”wrong” a vehicle’s estimated position could be (expressed in meters). There is no extra effort in acquiring this value, since it has to be sent out by every platooning vehicle in GCDC. This value allows the time headway to the vehicle ahead to be increased according to the estimated position accuracies.

The information about a single vehicle can fit into two different CAN messages. Consequently, to be able to support up to 10 vehicles in a single platoon, a total of 20 messages will be needed just for vehicle information (see Appendix A.7–A.9). If some ‘vehicle identifier’ signal was to be added in the information, only two unique messages would be required and many more vehicles could be supported. However, this will not be used – and the only explanation for this is: simplicity. It is easier for the ECU to sort out the information if it is delivered in the way illustrated by Figure 2.2, i.e. the information for the platoon leader is always sent in the two messages Vehicle_1_information_1 and Vehicle_1_information_2, and so on. To denote where our vehicle is in the platoon, the signal Our_order_in_the_platoon is used (see Appendix A.3).

In addition to the regular vehicle information messages, one extra important message has been added – this message is called Vehicle_ahead (see Appendix A.6). The purpose of this additional message is to always contain information about the vehicle ahead (if any), even if the vehicle ahead is not part of the platoon. In the case that the vehicle ahead does not send out any wireless information, the radar can still be used to fill this message with vital information. The usage of this message is shown in Figure 2.2 as well.

![Figure 2.2: Illustration of how which vehicle’s information is packaged into which CAN messages. The message Vehicle_ahead always contains information about the vehicle directly ahead of us – regardless of if that vehicle belongs to our platoon or not](image-url)
2.4.1 Examples of CAN message design

Below are just a few examples of signals received by the ECU. All signals and messages can be found in Appendix A.

The position of a vehicle is very important, therefore the resolution has to be high while the maximal representable range is still long enough. To decrease the number of data bits, the position is chosen to be relative rather than absolute. Also, relative position will most likely be easier to use when controlling the truck. The result is shown in top part of Table 2.1. As seen, the position of a vehicle relative to the truck may not be more than 3212 m, which is a very long distance and should not affect the control strategy.

When it comes to velocity, it is reasonable to stick to what seems to be the standard structure: with a length of 16 bits and resolution of 0.0025 m/s, the maximal representable speed is 160 m/s. This is absolutely enough, since the speed in GCDC is limited to about 22 m/s. The result is shown in the middle part of Table 2.1.

For a vehicle’s acceleration, it would be the most intuitive to use the absolute acceleration rather than the relative. A resolution of 0.1 m/s^2 is enough since Table 1.1 says 0.2 m/s^2 is required, and 8 bits results in ±12.5 m/s^2. The result is shown in the lower part of Table 2.1.

<table>
<thead>
<tr>
<th>Length</th>
<th>Signal</th>
<th>State</th>
<th>Resolution</th>
<th>Offset</th>
<th>Data range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 bits</td>
<td>Relative_position</td>
<td>0.1</td>
<td>-0.1</td>
<td>-3212</td>
<td>-3212 – 3212</td>
<td>m</td>
</tr>
<tr>
<td></td>
<td>Error</td>
<td>0xFE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Not Available</td>
<td>0xFF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16 bits</td>
<td>Absolute_velocity</td>
<td>0.0025</td>
<td>0</td>
<td>0</td>
<td>0 – 160.6375</td>
<td>m/s</td>
</tr>
<tr>
<td></td>
<td>Error</td>
<td>0xFE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Not Available</td>
<td>0xFF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8 bits</td>
<td>Absolute_acceleration</td>
<td>0.1</td>
<td>-0.1</td>
<td>-12.5</td>
<td>-12.5 – 12.5</td>
<td>m/s^2</td>
</tr>
<tr>
<td></td>
<td>Error</td>
<td>0xFE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Not Available</td>
<td>0xFF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For all the signals in Table 2.1, Not Available means that there is no vehicle there, while Error means that there is a vehicle there, but the signal value could not be estimated.

2.5 Model of the truck

One of the commonly used truck models at Scania that is implemented in Simulink is based on physical relationships between the forces acting on the truck, using Newton’s

second law as

$$m \ddot{v} = F_{\text{drive}} - F_{\text{air}} - F_{\text{roll}} - F_{\text{slope}} - F_{\text{environment}}$$  \hspace{1cm} (2.1)

where $F_{\text{environment}}$ is all the longitudinal forces acting on the truck: aerodynamic drag ($F_{\text{air}}$), rolling resistance ($F_{\text{roll}}$) and gravity due to the road slope angle ($F_{\text{slope}}$). The force driving the truck forward is $F_{\text{drive}}$. The constant $m$ is the vehicle’s mass and $v$ is the truck’s velocity [22, pp. 8–9].

This model has been extended from time to time and now contains e.g. the engine’s torque curves as well as its cruise controller. Even gear ratios and gearchange logic that controls on engine rpm is included in the model.

Although this model is very sophisticated, the ability to request a deceleration directly from an external source has not been implemented. Therefore, the brakes have to be modeled, and the torque generated by the brakes is given by

$$T_{\text{brakes}} = F_{\text{brakes}} \cdot r_{\text{wheel}} = (-F_{\text{environment}} - ma) \cdot r_{\text{wheel}}$$  \hspace{1cm} (2.2)

where $a$ is the requested acceleration and $a < 0$ is required in order for braking to occur. The constant $r_{\text{wheel}}$ is of course the wheel radius.

To be able to request deceleration as well as forward speed, $F_{\text{drive}}$ in (2.1) is modified to be

$$F_{\text{drive}} = F_{\text{driveline}} - F_{\text{brakes}}$$  \hspace{1cm} (2.3)

where $F_{\text{driveline}}$ is the force generated by the engine and is acting on the wheel axle.

2.6 Modeling the vehicles ahead

In the simulation environment, it is possible to simulate up to three additional vehicles in the platoon. They are modeled exactly the same way as the truck described in the previous section. However, the only controller used for these vehicles is the engine’s already existing speed controller. By cascade connection, the leader vehicle’s current speed becomes the speed reference for the second vehicle and so on. This results in equal speed for the three vehicles, although the inter-vehicle spacing may vary.

To mimic the task of the estimator, the vehicle information is packaged in the same way as the CAN messages are received by the controller. The data is also labeled with the correct message identifier according to Figure 2.2.
2.7 Traffic light model

To be able to see how the controller software behaves when the truck approaches a traffic light, a model of a traffic light is created. Not only does it have to switch between the three different colors red, yellow and green, but it should also output the time left to each color. This information will be available in GCDC through the road side unit[12, p. 38] and the values will be received by the ECU in the \texttt{Road\_side\_unit\_information} message (see Appendix A.5). An illustrative example is shown in Figure 2.3. Of course the time left to each color is decreased every second – not updated just once in every new state.

\begin{verbatim}
Time: 0 s
Current color: Red
Next color: Yellow, 20 s
Second next color: Green, 23 s

20 s
Current color: Yellow
Next color: Green, 3 s
Second next color: Yellow, 23 s

23 s
Current color: Green
Next color: Yellow, 20 s
Second next color: Red, 3 s

43 s
Current color: Yellow
Next color: Red, 3 s
Second next color: Yellow, 23 s
\end{verbatim}

Figure 2.3: Illustration of how a traffic light might behave. The developed model also outputs information about the upcoming colors and the time left.
3 Platooning Coordination and Vehicle Control

The objective is to control the vehicle within a platoon while the software should be able to handle a variety of scenarios that can happen along the way. However, the truck needs to be reconfigured in order for the autonomous control to work.

3.1 Requirements and limitations

The scenarios mentioned in section 1.2 can be broken down into requirements and limitations that affects this thesis. The former specifies what the software has to be able to accomplish. The latter, however, specifies some scenarios that will not have to be accounted for. Below are two lists of the main requirements and limitations.

Requirements
1. Start from a green light
2. Join and split a platoon
3. Adapt the speed to the vehicle(s) ahead
4. Never go faster than the road’s speed limit
5. Turn off automatic control when
   - throttle pedal is pressed
   - brake pedal is pressed
   - the system is manually turned off

Limitations
1. The platoon will only move in a straight line – steering is manually controlled by the driver
2. No vehicles will enter or exit a platoon sideways
3. The speed limit will never decrease, only increase
Apart from starting on a green light, requirement 1 will be extended to stopping at a red light, although this may never occur in the GCDC competition. The reason is that the truck will have to make a 'stop' decision if it is standing at a red light on system startup, so the extra effort required to brake for a red light is not big.

### 3.2 Controlling the truck

In a Scania truck there are several systems with different functionalities. All the systems are connected to a controller area network, allowing them to communicate with each other. When connected to the network there are several ways to control the truck. One way is to let the software send throttle and brake pedal signals, another is to request a torque from the engine. In this thesis, however, it was chosen to use the already existing speed controller in the Engine Management System (EMS) which accepts a speed request via the CAN message *Cruise Control Management Speed Request* (CCMSR).

Since the speed controller is not able to slow down the truck, the Electronic Braking System (EBS) will be used. The EBS can apply the brakes to slow down the vehicle with a certain deceleration, which is requested through the message *External Brake Request* (XBR).

To avoid collisions on the CAN bus when the new software is connected to the network, the interface between the driver and the cruise control has to be turned off. An illustration of how the new software communicates with the EMS and the EBS is shown in Figure 3.1.

![Figure 3.1: Illustration of how the developed software sends the two messages CCMSR and XBR in order to control the truck](image-url)
3.2.1 Tuning the brake request

The electronic braking system’s deceleration request is not a closed loop and the main reason is that it is hard to accurately measure the obtained deceleration. Since it is not a closed loop, the acquired deceleration is not exactly what is being requested. By multiplying the required acceleration with a nonlinear gain, the acquired acceleration will at least be closer to what is requested by the application.

By performing a series of brake requests between $-1$ and $-4.5 \text{ m/s}^2$, and comparing them to the truck’s change in speed, the plot in Figure 3.2 can be produced (the origin has been added manually). The plot shows which input is required to obtain the desired acceleration. In the same figure, a second order polynomial has been fitted to the data points – this curve is the nonlinear gain needed for the truck to brake as intended.

![Figure 3.2: A plot showing which acceleration that actually has to be requested when a certain acceleration is desired. The solid line is a curve fitted to the data points](image_url)

Important to remember is that this is still an open loop system, but at least it will be easier for the controllers to close the loop if the real deceleration is closer to what is desired.

3.2.2 Preparing for controller implementation

The controllers designed for this project will request an acceleration – both positive and negative. However, the already existing speed controller in the engine is only able to deliver an increase of the speed, not a decrease. For that, the brakes have to be used.
As described above the brake system accepts a negative acceleration as input, which means that the request made by a controller can be sent directly to the brakes. However, since the engine’s speed controller only accepts a speed as reference, the requested positive acceleration has to be integrated. The solution is illustrated by Figure 3.3, where the desired acceleration is sent to the correct system depending on its sign.

Figure 3.3: If the desired acceleration, \( a \), is positive it is integrated and sent as a reference, \( v_{\text{ref}} \), to the engine’s speed controller, while negative acceleration values are sent directly to the brake system as \( a_{\text{brake}} \) (\( Ts \) is the integrator’s sampling time)

### 3.3 Different controller configurations

Different scenarios call for different controller configurations; the truck should behave differently depending on which position it has in the platoon. Together with E. Ståklinga it has been decided to limit the amount of scenarios by looking only at one, two or three vehicles ahead at the most (where one vehicle is always the platoon leader). This is illustrated in Figure 3.4. The result is four different controller configurations depending on the position, \( i \), where \( i = 1, i = 2, i = 3 \) or \( i \geq 4 \).

Figure 3.4: This vehicle has position \( i \) in the platoon. To simplify, only look at the vehicles \( i - 1, i - 2 \) and 1 when controlling the truck

Included in Figure 3.4 is a potential vehicle ahead of the platoon. This vehicle has to be taken into consideration, because despite that it is not part of the platoon, it still has to be avoided. It might even be part of another platoon that has to be joined.

Another possible scenario is that the potential vehicle lacks wireless communication and is therefore not joinable. The solution is to have another controller configuration which takes the distance to the vehicle into consideration and uses its speed as reference. This
scenario is crucial to handle, since the GCDC leader vehicle mentioned in section 1.2.1 will not be joinable.

### 3.3.1 Control strategies

As mentioned before the control strategies are not designed in this thesis, but by E. Stålklinga. Her approach has been to compare an offline\(^1\) LQI (linear quadratic integral) controller with an online\(^2\) MPC (model predictive control) to see if there is a significant difference between the two. One of the biggest advantages with MPC is that distinct constraints can be applied to the controller, e.g. keeping the acceleration lower than \(2.0 \text{ m/s}^2\).

Towards the end, however, she chose to design an offline LQR (linear quadratic regulator) controller with the three states: own acceleration, relative speed and error in relative distance. For best behavior, different weight matrices has been designed for five different speed intervals. Moreover, the weights on relative speed and error in relative distance are high, while the weight on acceleration is low.

The LQR controller was designed to look at only one vehicle ahead \((i - 1)\), but was extended by adding relative speed for vehicle \(i - 2\). However, the system became non-minimal, which required the weights to be chosen carefully in order to stabilize the system.

Since the controllers to be implemented had not yet been fully developed when the testing phase began a temporary controller was designed by Henrik Pettersson (the results are shown in chapter 5). This controller uses the speeds of several vehicles ahead to create a weighted speed reference as input signal, where the closest vehicle’s speed weighs the least. Its task is to keep a distance of 15 m or 1 s (whichever is the furthest) but never get closer than 0.9 s or 10 m. The new desired speed is then calculated as

\[
v_{\text{ref}} = v_{\text{weighted}} + w \cdot \frac{(d_{\text{rel}} - d_{1s})}{d_{1s}} \tag{3.1}
\]

where \(v_{\text{weighted}}\) is the weighted speeds, \(d_{1s}\) is the distance traveled in 1 s (or 15 m) and \(d_{\text{rel}}\) is the relative distance to the vehicle ahead. The constant \(w\) weights how much the error in distance affects the desired speed, \(w = 2\) was used during the testing.

### 3.4 Platooning logic

The decision making of ‘joining’ and ‘splitting’ a platoon is referred to as **platooning logic**. This logic is responsible for figuring out when to be platoon leader and when to

---

\(^1\)Pre-optimized values are used for the feedback gain  
\(^2\)A new optimal feedback gain is calculated continuously
be a follower. A platoon may consist of only one vehicle, this vehicle is then a leader without followers (the top scenario in Figure 3.5).

Every vehicle has unique vehicle ID and platoon ID and both are being broadcasted periodically. In GCDC, a vehicle shows it is a leader by setting its platoon ID equal to its vehicle ID\cite[p. 32]{12}. Whenever a vehicle is a follower, it has its platoon ID set to the platoon leader’s platoon ID (illustrated in the lower half of Figure 3.5).

![Figure 3.5](image.png)

Figure 3.5: Every vehicle has a unique vehicle ID (vID), while its platoon ID (pID) is always set to the leader’s vehicle ID

An initial approach to platooning logic is to keep it as simple as possible. If a vehicle ahead is "close enough" a join request should be sent. Similarly, when the vehicle is "too far away" the platoon should be split. However, if the vehicle ahead is still close but its platoon ID has changed, then it is just a matter of sending a join request to the new platoon leader.

To send a join request, the vehicle should actually flag that it wants to join, rather than sending a request per se. In GCDC, this is done by changing the \texttt{PlatooningState} from ‘stable’ to ‘transition’ and setting the platoon ID to the leader’s ID\cite[pp. 34, 43]{12}. If an answer is not received within 3 seconds it is considered a timeout\cite[p. 36]{12}.

A flowchart describing the platooning logic is shown in Figure 3.6 on page 20.

### 3.5 Traffic lights

There are two key issues when it comes to traffic lights: coming to a halt at the upcoming traffic light when it is red and to go when the light turns green. With the road side unit (RSU) information at hand, it is possible to know which color the traffic light will change to – and when.
3.5. Traffic lights

Figure 3.6: Flowchart for the platooning logic
When approaching a traffic light (see Figure 3.7), there are a few things that have to be calculated. Firstly, one has to know which color the traffic light will have when the vehicle arrives at the traffic light’s position. Secondly, if it is going to turn red, will it be possible to pass it with the current speed? The time left to red, $t_{\text{red}}$, is easily found by looking at the signals in the message `Road\_side\_unit\_information` (see Appendix A.5). The criterion for having to stop at a red light is now given by

$$t_{\text{red}} \leq \frac{d - d_{\text{offset}}}{v}, \quad \text{need to stop}$$

$$t_{\text{red}} > \frac{d - d_{\text{offset}}}{v}, \quad \text{will make it past}$$

(3.2)

where $d$ is the distance left to the traffic light and $v$ is the vehicle’s current speed. Also, the constant $d_{\text{offset}}$ has been added to create a safety margin to the traffic light.

When it is not possible to pass the green light it has to be calculated where to start braking with a "comfortable" deceleration, $a_{\text{comfort}}$, in order to stop at the traffic light’s location. Toricelli’s equation\[23, p. 70\] calculates the final speed, $v_{\text{final}}$, from an initial speed, $v_{\text{initial}}$, as

$$v_{\text{final}}^2 = v_{\text{initial}}^2 + 2ad$$

(3.3)

where $d$ is the total distance that the acceleration $a$ is applied. By setting $v_{\text{final}} = 0$ and $a = a_{\text{comfort}}$, the distance for when to apply the brakes, $d_{\text{threshold}}$, is given by

$$d_{\text{threshold}} = -\frac{v^2}{2a_{\text{comfort}}}$$

(3.4)

where $a_{\text{comfort}} < 0$ when braking. Of course $v_{\text{initial}} = v$ since this equation is calculated every sample.

Continuously calculating the deceleration is done by solving (3.3) for $a$ as

$$a = -\frac{v^2}{2 \cdot (d - d_{\text{offset}})}$$

(3.5)

Obviously, $v_{\text{final}} = 0$ since the truck should come to a stop. Also, $v_{\text{initial}} = v$ as before and $d_{\text{offset}}$ is needed to stop at the intended position.
Sometimes it is not desired to stop 0.0 m from the traffic light and it might also be nice to have some margin. Therefore the distance, \( d \), has to be modified as \( d_{\text{new}} = d - d_{\text{offset}} \) in equations (3.2) and (3.5). The variable \( d_{\text{offset}} \) is also illustrated in Figure 3.7.

The relative distance to the traffic light is stored in the Road_side_unit_information message (just like the colors and times). Since the amount of data bits is limited, it is important to find out at which distance the traffic light has to be visible to the algorithm. By inserting the truck’s maximum speed 90 km/h and an acceleration of \(-1.25 \text{ m/s}^2\) into (3.4), \( d_{\text{threshold}} \) becomes 250 m. With a resolution of 1 m, exactly one byte is required for the traffic light distance. The resulting CAN signal is shown in Table 3.1, which can be compared to the signals in Table 2.1.

<table>
<thead>
<tr>
<th>Len.</th>
<th>Signal</th>
<th>State</th>
<th>Res.</th>
<th>Offset</th>
<th>Data range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 bits Traffic_light_relative_position</td>
<td>1</td>
<td>0</td>
<td>0 – 250</td>
<td>m</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error</td>
<td>0xFE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not Available</td>
<td>0xFF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.1: The structure of the CAN signal that holds information about the distance to an upcoming traffic light
4 Controller Implementation

This chapter describes the implementation of everything that has been mentioned in the previous chapters. It aims to describe how the software is constructed, including the main error handling features and small, but important, fixes that are important for the controller to work and function safely.

4.1 Software

Implementation of the control algorithms is done by using Matlab Simulink. It is a very powerful software, especially when it comes to simulations. But the reasons for choosing this software are many. Firstly, Scania has a wide variety of custom libraries for solving different tasks. One of them is reading and writing CAN messages; it is just a matter of adding a block to the workspace and all messages are coded and decoded perfectly. Secondly, the control strategies developed by E. Stålklinga are created in Simulink, so they will not have to be modified much to be compatible with the rest of the code.

Simulink also has an add-on called Real-Time Workshop that enables C and C++ code generation. This is used to build the developed software and convert it to machine code so that it can be run on the ECU.

4.2 Controller layout

The two primary parts of the software is the configuration selector and the controller, seen in Figure 4.1. The configuration selector makes decisions about which controller that should be used, if it is time to brake for a red light or if the automatic control should be deactivated for some reason. These decisions are based both on information from the red CAN bus and from the internal bus. The former receives data about the truck, while the latter receives estimated information about the truck, the platoon and the cooperative driving system.

The box labeled ‘controller’ in Figure 4.1 contains all the different controllers and makes use of the platoon information to control the vehicle. All controllers control the truck’s acceleration and the active controller’s values are sent to the red CAN bus in the messages CCMSR and XBR (see section 3.2). Some information is also sent back to the WSU, using the internal bus.
A more detailed illustration of the controller block is shown in Figure 4.2, where the six controllers output an acceleration reference. However, stopping at a traffic light will override the other controllers – more on this in section 4.5.

Another part of the software is the platooning logic (also shown in Figure 4.1), which only uses information from the internal CAN bus to make its decisions. The data containing the current platooning state and platoon ID is sent back to the WSU. Since this logic is not dependent on the controller state, it has been implemented in such a way that it can be active even when the vehicle is operated in manual mode. This permits testing of the platooning logic together with other vehicles without driving in automatic mode.

### 4.3 On/Off

The terms "On" and "Off" in a scope like this is interpreted as controlling the truck autonomously or not. This implies that the driver has full control of the truck when the controller is in 'Off' mode.

When the driver has decided to turn on the autonomous control, the signal `Automatic_mode` (see Appendix A.3) will be set to 'On' – and the controller should just be turned on. However, to avoid any unexpected output signals caused by integrators, a reset signal is also triggered to make sure that all controllers are reset to zero.

Turning off the system is of course done with the off button, but there are times when the automatic control has to be turned off due to errors and/or hazardous situations. Examples of such situations are listed in Table 4.1.
4.3. On/Off

It might seem strange to turn off the automatic control completely when information is missing, but it is important to remember that the automatic control is not vital for the vehicle to function, the driver can still control it with the throttle and brake pedals.

To make future debugging much easier, the 'Off' state has been extended to these five different states:

- Off due to driver decision
- Off due to information error
- Off due to WSU status
- Off due to CAN timeout
- Off due to radar error (only reachable if the radar is currently in use)

This makes it possible to see why the ECU has stopped sending commands to the truck. The entire statechart can be seen in Figure 4.3 on page 30, where two "Idle" states have been added to ensure that the on/off button goes to 'Off' before it goes to 'On' again. A description on how to interpret the statechart is shown in Figure 4.4 on page 31.
Table 4.1: Example of situations when the automatic control should be turned off

<table>
<thead>
<tr>
<th>Situation</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Road speed missing</td>
<td>The maximum speed limit on the road is unavailable</td>
</tr>
<tr>
<td>Traffic light error</td>
<td>Important information (position, color, etc.) about an upcoming traffic light is missing</td>
</tr>
<tr>
<td>Vehicle ahead error</td>
<td>Vital information about the vehicle ahead is missing</td>
</tr>
<tr>
<td>Radar error</td>
<td>The radar is in use by the controller, but it is showing an error</td>
</tr>
<tr>
<td>CAN timeout</td>
<td>A CAN message from the WSU has not been received for 200 ms</td>
</tr>
<tr>
<td>WSU subsystem status</td>
<td>GPS, estimator and wireless communications are all malfunctioning</td>
</tr>
</tbody>
</table>

4.4 Changing controller configuration

The criteria for choosing between the different controller configurations that were described in section 3.3 are implemented simply by using if-statements, as illustrated by Table 4.2, where the numbers in "vehicle 1" and "vehicle 2" etc. denotes a vehicle’s order in the platoon. And i is this vehicle’s order, while the numbers in the left most column is just a numbering of the configurations.

Table 4.2: The different controller configurations and their criteria (strongly connected to Figure 3.4)

<table>
<thead>
<tr>
<th>Configuration / Action</th>
<th>Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Drive at maximum allowed speed</td>
</tr>
<tr>
<td></td>
<td>(i == 1) AND vehicle ahead is NOT very close</td>
</tr>
<tr>
<td>2</td>
<td>Control only on the vehicle ahead</td>
</tr>
<tr>
<td></td>
<td>(i == 2) OR ((i == 1) AND vehicle ahead is very close)</td>
</tr>
<tr>
<td>3</td>
<td>Control on vehicle 1 and 2</td>
</tr>
<tr>
<td></td>
<td>(i == 3)</td>
</tr>
<tr>
<td>4</td>
<td>Control on vehicle (i - 1), (i - 2) and 1</td>
</tr>
<tr>
<td></td>
<td>(i \geq 4)</td>
</tr>
<tr>
<td>5</td>
<td>Brake for an upcoming red light</td>
</tr>
<tr>
<td></td>
<td>Need to stop at a red light</td>
</tr>
</tbody>
</table>

The condition 'vehicle ahead is very close' in Table 4.2 comes from the scenario mentioned in section 3.3, where the 'potential vehicle' ahead of the platoon might lack wireless communication and is therefore not joinable.

A real problem with a setup like this, with multiple controllers where only one can be used at a time, is windup. The windup is caused by integrators when an inactive controller does not have the ability to control the system. The controller will try to apply more and more energy to get what it wants.

To solve the windup problem, a 'set' signal is introduced. The set signal is sent out whenever the controller configuration changes and can be compared to the reset signal.
mentioned in section 4.3. The set signal’s task is to set the newly activated controller’s control error to zero. This results in a smoother transition when a new configuration is chosen.

The set signal is also triggered when the braking for a red light has ended. The reason is that this braking overrides all other controllers and has not been considered a change in controller configuration. More on this in the next section.

### 4.5 Stopping at a traffic light

Just because equation (3.2) in section 3.5 says ”need to stop”, it does not mean that it is time to activate the brakes. An additional requirement is that $d_{\text{threshold}}$ in equation (3.4) fulfills

$$d - d_{\text{offset}} \leq d_{\text{threshold}}$$

(4.1)

to avoid braking with a tiny deceleration from a far off distance.

When applying the brakes, it is important that the algorithm does not brake harder than $a_{\text{min}} = -4.5 \text{ m/s}^2$. This is solved by using the $\max$ function when calculating the acceleration, $a$, like so

$$a = \max \left\{ \frac{v^2}{2(d - d_{\text{offset}})}, a_{\text{min}} \right\}$$

(4.2)

where negative values of acceleration result in braking. This solution is of course very good in a simulated environment, but when used in reality the estimated speed value might be quite inaccurate for low speeds. At least this is true for the vehicle’s tachometer. Therefore an extra state has been added which applies full brakes ($a_{\text{min}}$) when

$$d \leq d_{\text{offset}}$$

(4.3)

e.g. when the truck is between the desired stopping point and the traffic light’s position. The brakes are not released until the light turns green or the control system is turned off. However, passing the traffic light (even if red) will tell the system that there is no longer a traffic light ahead – resulting in the brakes to be released.

When the conditions above are fulfilled and the vehicle has started braking, no change in controller configuration occurs. For softest possible experience, the one which brakes the hardest (active controller or brake algorithm) will affect the brakes. But as mentioned earlier, the set signal is triggered when the traffic light algorithm stops braking.

### 4.6 Platooning logic

For simplicity, the platooning logic developed as a flowchart in section 3.4 is converted to a statechart and implemented in the Simulink tool Stateflow. It consists only of
three states: 'leader', 'follower' and 'transition'. Since a platoon may consist of only one vehicle, the 'leader' state is set to be the initial state.

When deciding to signal a join request (i.e. entering the 'transition' state) it is made sure that the vehicle is actually catching up to the vehicle ahead. Furthermore, it makes sure the distance between the vehicles is smaller than the so called 'join distance'. To avoid infinite switching, the 'split distance' (used for splitting the platoon and becoming a leader) is chosen be be somewhat greater than the 'join distance'.

Whenever the control system is activated and the reset signal is triggered, the platooning logic is affected too: no matter the current state, the logic is returned to the 'leader' state.

4.7 Error handling

When working with advanced systems, numerous errors can occur. For the system to operate safely it is therefore important to handle the most critical potential errors. When an error occurs in the controller software that cannot be treated, the system will be turned off, according to Table 4.1.

One possible error is that the throttle and brake pedal messages are no longer sent on the CAN bus, which makes it impossible to know if the driver is pressing any of the pedals. Therefore, the automatic control is turned off if any of these messages are not received on time.

If the CAN component in the WSU stops sending messages to the ECU, it will result in packet loss. Losing the key message Platoon_and_system_information will trigger a status flag in the ECU, causing the automatic control to be turned off.

Not all errors must cause the system to shut down. Most of them can actually be treated by taking safety precautions, as described below.

4.7.1 Loss of critical components

There are three key components in the WSU that are critical for the entire system to function safely – these are: estimator, wireless communications and GPS. If all of them starts to malfunction, the controller is turned off. However, as long as the estimator is working and reports valid values it is safe to move on, but safety measures have to be taken. This is done by increasing the time headway to the vehicle ahead.

Although the estimator is a vital component, it is still possible to control the truck automatically if the estimator stops working. If the software detects that the estimator is malfunctioning, it will use the truck’s own hardware to provide a backup controller with the required information. This is done by reading the tachometer to acquire the speed and the radar to acquire information about the vehicle ahead, i.e. distance, speed.
and acceleration. The truck’s own acceleration is acquired by filtering the speed’s discrete derivative through a simple low-pass filter. When using these components to control the truck, it is of course important to check that they are sending valid data – if they are not, the controller is turned off and the driver is in charge.
Figure 4.3: Statechart for the on/off logic. The two "Idle" states are present to make sure the system does not reactivate itself (a description of how to interpret this statechart is shown on the next page).
State_name

entry:
  stateActive = 1;
  counter = counter + 1;

exit:
  stateActive = 0;

[myCondition == true]

Figure 4.4: A description of how to interpret the statechart in Figure 4.3 on page 30
5 Testing and Results

This chapter shows some of the tests that have been performed during the project and the results reflect the requirements mentioned in section 3.1. Before the tests could be executed, a software for standalone testing had to be developed.

All test cases presented in this chapter were performed using a real Scania truck, although one simulated result has been included as well – for comparison.

5.1 Platoon simulation for standalone testing

Testing the software on the real truck in the absence of the rest of the Cooperative Driving System required that another software had to be developed. Its task is to receive information from the user – such as controller activation, speed limit, inter-vehicle distances and more. The information is used to create a virtual platoon ahead of the truck, and the data for the three vehicles is re-packaged to imitate what the real CDS will send.

Two different versions of this testing software was developed: a simpler and an extended one. To create the three vehicles ahead, the simpler version represents their speeds as stepwise constant signals. Both speeds and inter-vehicle distances are the same for all the vehicles. When the truck sends a join request to the virtual platoon, the testing software immediately accepts it and tells the truck that it is now the fourth vehicle in a platoon, instead of the first one.

The extended version, however, uses three separate (but equal) second order systems to describe the vehicles’ speeds. To simulate real conditions, the rise time of the system is set to be the maximum allowed acceleration in GCDC and the system also has some overshoot. The speed reference for the leader vehicle is decided by user input, while the actual speed of the leader is fed back to become a reference for the next vehicle, and so on. This creates an initially oscillating behavior in the vehicles’ speeds – and since there is no active control running, the inter-vehicle distances vary.

Another feature in the extended version is that the number of virtual vehicles ahead of the truck may vary from one to three, depending on the user’s decision. This is of course considered when the testing software accepts a join request from the truck.

The model of a traffic light described in section 2.7 is also implemented in both versions of the testing software, which enables testing of starting and stopping on the signal from
5.2 Deactivate controller on driver demand

One safety requirement set by GCDC is that the automatic control should be turned off whenever the driver presses either the throttle pedal or brake pedal. Figure 5.1 shows how the brake pedal is pressed by only a few percent and causes the ECU status to go from ‘on’ to ‘off’ and, of course, make the truck decelerate from the speed previously set by the control system. As desired, the system does not reactivate itself when the pedal is released.

![Deactivate controller when brake pedal is pressed](image)

Figure 5.1: A test to see that pressing the brake pedal deactivates the automatic control

5.3 Go to speed limit at green light

One of the most basic scenarios in GCDC is to stand still at a red traffic light and go when it turns green. In Figure 5.2 the truck has been standing at a traffic light for about 20 seconds when the light switches from red to yellow to green. As soon as it stops being

---

Joakim Kjellberg 33 Electrical Engineering
April 20, 2011 Systems and Control
red, the system tells the truck to accelerate with \(1.0 \text{ m/s}^2\) to the maximum speed limit – which it does, but with an acceleration of \(0.64 \text{ m/s}^2\).

![Graph showing speed profile](image)

**Figure 5.2:** The truck starts going towards the maximum speed limit with an average acceleration of \(0.64 \text{ m/s}^2\) when the light is no longer red

### 5.4 Go from green light and catch up with a platoon

When going from a green light there might be vehicles ahead. In this test case, the truck drives on green light and catches up with a platoon of three vehicles driving at around 20 km/h. The result is shown in Figure 5.3 (note that only the vehicle directly ahead is shown, not the two leading vehicles). When the vehicle ahead is 190 m away, the platooning role goes from 'leader' to 'follower' and the controller configuration goes from 1 (leader) to 4 (fourth in line) (see Table 4.2), causing the truck to follow the three vehicles.

Once the truck catches up with the platoon ahead, the distance to the closest vehicle is 13–21 m, even when the platoon leader increases the speed from 20 to 30 km/h. As a result, the truck tries to keep up with the leader and accelerates even before the vehicle ahead of us does.
5.5 Follow three vehicles

The following of three vehicles has been performed both in the simulation environment and in reality. The biggest difference between the two is that the test tool (simpler version) used for the real test has virtual vehicles ahead represented by a constant speed, as described in section 5.1 – while the simulation uses vehicle models as described in section 2.6.

5.5.1 In simulations

Figure 5.4 shows the simulated version of this test case. There are three vehicles ahead, but the speed is only plotted for the closest one. The leader vehicle follows the stepwise constant speed reference showed in the upper plot and the other vehicles uses only a speed controller to follow the leader. The distance to the closest vehicle is within 12–16 m.

Due to an error in the simulation environment, they do not apply brakes when the reference speed changes from 50 to 37 km/h – they only wait for the rolling resistance to slow them down.

5.5.2 In reality

Figure 5.5 shows the version of this test case where a real truck was used. The automatic control was activated when traveling at about 20 km/h to give the truck a change to catch up with the platoon.

The sudden braking at $t = 70$ s is an acceleration of $-6.74 \text{ m/s}^2$. This is because the lack of a lower bound for the requested acceleration in the controller – this limit was added in a later version.

5.6 Stop at a red light

Stopping at a red light should result in two things: coming to a halt and splitting the platoon. Figure 5.6 shows exactly this – where the test case was performed with a real truck. When the truck approaches the traffic light, the system requests it to slow down by $-1.20 \text{ m/s}^2$. However, the acquired deceleration is $-1.26 \text{ m/s}^2$ so the truck stops 9 m away from the traffic light instead of the desired 3 m.

When the vehicle ahead reaches a relative distance of 210 m (at $t = 29$ s), the platooning logic makes the decision to become platoon leader and the controller configuration changes from 4 to 1 (once again, see Table 4.2).
Figure 5.3: A test to go from a green light with a real truck to catch up with a platoon of three virtual vehicles. As soon as the truck has passed the traffic light, the signal color does not affect the control (exactly as intended)
### Figure 5.4: A simulation of how the truck follows a platoon of three vehicles where the leading vehicle’s speed reference is stepwise constant
Figure 5.5: A real truck follows three virtual vehicles that are only represented by a stepwise constant speed (may be compared with Figure 5.4)
Figure 5.6: Stopping at a red light while the other three vehicles keep going causes the truck to split the platoon.
6 Discussion and Conclusions

In this chapter, conclusions are drawn based on the results shown in the previous chapter. It is also discussed what went well and what could have been made differently. Lastly, examples for future work are given.

6.1 Conclusions

The focus in this thesis has been to develop a framework for implementing control strategies within the Scoop project. The developed software has been implemented in such a way that is able to choose which strategy to use based on the current circumstances.

The results verify that the system shuts down as soon as the brake pedal is pressed and the system stays deactivated when the pedal is released – which also applies to the throttle pedal. Having the possibility to abort the control in an easy way makes it safe to operate the system.

When the truck is standing at a red traffic light it applies full brakes, as intended in the software implementation. And when the light switches from red, a ramp with slope $1.0 \text{ m/s}^2$ is sent as a reference to the engine’s speed controller. Since the ramp starts at 0 km/h the requested torque is lower than the idle torque for a period of 4 s – which explains the delay between desired speed and actual speed in Figure 5.2. However, the outcome is quite surprising, since it results in the truck leaving the traffic light exactly when it turns green. The acquired acceleration of $0.64 \text{ m/s}^2$ is due to that the test was carried out on snow, which caused the anti-slip controller to take charge of the torque demand. That the tires are slipping especially shows up at $t = 120 \text{ s}$ in Figure 5.2, where the speed makes a sudden leap.

Catching up with vehicles ahead causes the system to signal a join request and becoming a member of the platoon – which is intended. Results show that when the distance to the vehicle ahead becomes too far (e.g. when stopping at a red light) the truck forms its own platoon by becoming a leader and splitting the platoon ahead. Stopping at a red light is not a closed loop, which results in the truck braking too hard and stopping too early. However, since equation (3.5) is calculated continuously it will result in the truck braking harder and harder if it is not slowing down as planned.

In the test case illustrated by Figure 5.3 the vehicle’s speed is quite shaky. The reason for this behavior is that the controller used at that time had not been perfected. So,
from a control perspective it might not look very good – although it still manages to follow the speed change from 20 to 30 km/h. From a platooning point of view, however, it does have the desired behavior: it approaches a platoon from the rear, going from leader to follower and stays in the platoon.

### 6.2 Discussion

One of the performed tests that is not included in this report is using the radar to control the truck when the estimator is malfunctioning. It actually works in the sense that it is able to follow a virtual vehicle ahead of the truck, but it was not shown because it would be quite redundant due to the use of virtual vehicles once again. However, it would have been interesting to see the results if the test had been executed with a real vehicle in front of the truck, but there was not enough time for that. One major issue is that the radar only reports proper values while moving, not at standstill, which would of course have to be accounted for in the software.

Speaking of the radar and continuing the platooning without the estimator, it is not really possible to do so in reality without modifying the signals a bit. The problem arises from the fact that the software will turn off the automatic control if information about vehicle ahead, speed limit or traffic light is missing. Since all this information is sent by the estimator, it will not reach the ECU if the estimator is broken. This makes the error handling a bit contradictive. During the testing phase however, these three signals were sent to the ECU to keep it running, but the software only used the truck's own information (from radar and tachometer) to follow the virtual vehicle ahead.

If I had the chance to do this project allover again I would definitely design a simulation environment that the software could just be plugged in to and that could also be used during the tests with the real truck. Much of the strange behavior that occurred during the testing phase was due to bugs in the test tool. If the two environments would have been exactly the same, the errors could have been discovered on the desktop rather than in the truck.

Another thing that could (and should) be done differently is the way the vehicle information messages are handled. Instead of using different messages for every vehicle, a unique identifier should be inserted so that only one set of messages have to be used. Why I chose do to what I did (see Figure 2.2) was due to the inflexibility in Simulink and that it would most likely require some kind of data storage for the vehicle information.

### 6.3 Future work

Below is a list with suggestions for potential future work:

---

Joakim Kjellberg
April 20, 2011

41 Electrical Engineering

Systems and Control

• Values for 'join distance' and 'split distance' (see section 4.6) should be calibrated more thoroughly. For the test cases in this report they were set to be 190 and 210 m, respectively. But the question is when a vehicle should be considered to be part of a platoon or not – it might have to do with the inter-vehicle distances within the platoon to join. Also, the parameter 'vehicle ahead is very close' (used in section 4.3) needs to be calibrated. During testing, this value was set to 100 m.

• Error handling for vehicles further ahead in the platoon than just the vehicle directly ahead is needed. For example, if controller configuration 4 is active (see Table 4.2) and information about vehicle $i-2$ is erroneous, configuration 1 should be activated instead, until all information is available once again.

• For better performance when braking for a red light, the acceleration request designed in section 3.5 should be complemented with an active controller. Although the truck manages to stop before the traffic light, it does not always stop at the desired position – especially when the non-linear deceleration gain is not calibrated properly.
7 Bibliography


Appendix A

CAN Specifications for ECU and WSU

A.1 Platooning_state

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Tmin</th>
<th>Tmax</th>
<th>PGN</th>
<th>Default Priority</th>
<th>DPext</th>
<th>DP</th>
<th>Group Extension</th>
<th>Source</th>
<th>Destination</th>
<th>Standard/proprietary</th>
</tr>
</thead>
<tbody>
<tr>
<td>18FF0100</td>
<td>0.1</td>
<td>0.1</td>
<td>FF01</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>01</td>
<td>00</td>
<td>00</td>
<td>Prop</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S e n t</th>
<th>R e c i v e r</th>
<th>B y t e</th>
<th>L e n g t h</th>
<th>Explanation</th>
<th>S t a t e</th>
<th>Resolution</th>
<th>Offset</th>
<th>Operating range</th>
<th>Data range</th>
<th>Unit</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 4</td>
<td>Current_platoon_ID</td>
<td>1 1 -</td>
<td>1 - 11</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error</td>
<td>0xE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not Available</td>
<td>0xF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 5 4</td>
<td>Current_platooning_state</td>
<td>1 0 -</td>
<td>0 - 1</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stable</td>
<td>0x0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transition</td>
<td>0x1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error</td>
<td>0xE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not Available</td>
<td>0xF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 1 4</td>
<td>Current_platooning_role</td>
<td>1 0 -</td>
<td>0 - 1</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Follower</td>
<td>0x0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Leader</td>
<td>0x1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error</td>
<td>0xE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not Available</td>
<td>0xF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## A.2 ECU_status_and_speed_intent

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Tmin</th>
<th>Tmax</th>
<th>PGN</th>
<th>Default</th>
<th>Priority</th>
<th>DPext</th>
<th>DP</th>
<th>Group</th>
<th>Extension</th>
<th>Source</th>
<th>Destination</th>
<th>Standard/proprietary</th>
</tr>
</thead>
<tbody>
<tr>
<td>18FF0000</td>
<td>0.1</td>
<td>0.1</td>
<td>FF00</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>00</td>
<td>00</td>
<td>Prop</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Speed_intent
- **Explanation**: Speed_intent
- **Resolution**: 0.0025
- **Offset**: 0
- **Operating range**: 0 - 160.6375 m/s
- **Unit**: m/s
- **Note**: 0xFExx

### Software_version
- **Explanation**: Software_version
- **Resolution**: 0.1
- **Offset**: 0
- **Data range**: 0 - 250
- **Unit**: —
- **Note**: 0xFE

### Time_left_to_speed_intent
- **Explanation**: Time_left_to_speed_intent
- **Resolution**: 0.1
- **Offset**: 0
- **Data range**: 0 - 25 s
- **Unit**: s
- **Note**: 0xFE

### Controller_state
- **Explanation**: Controller_state
- **Resolution**: 1
- **Offset**: 0
- **Data range**: 0 - 10
- **Unit**: —
- **Note**: 0xFE

### ECU_status
- **Explanation**: ECU_status
- **Resolution**: 1
- **Offset**: 0
- **Data range**: 0 - 10
- **Unit**: —
- **Note**: 0xF

### Distance_violation_status
- **Explanation**: Distance_violation_status
- **Resolution**: 1
- **Offset**: 0
- **Data range**: 0 - 3
- **Unit**: —
- **Note**: 0xF

### Need_to_stop_at_a_red_light
- **Explanation**: Need_to_stop_at_a_red_light
- **Resolution**: 1
- **Offset**: 0
- **Data range**: 0 - 1
- **Unit**: —
- **Note**: 0xF

---

Joakim Kjellberg  
April 20, 2011  
B Electrical Engineering  
Systems and Control
## A.3 Platoon_and_system_information

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Tmin</th>
<th>Tmax</th>
<th>PGN</th>
<th>Default</th>
<th>Priority</th>
<th>Source</th>
<th>Destination</th>
<th>Standard/proprietary</th>
</tr>
</thead>
<tbody>
<tr>
<td>18FF0001</td>
<td>0.1</td>
<td>0.1</td>
<td>FF00</td>
<td>6</td>
<td>1</td>
<td>00</td>
<td>01</td>
<td>Prop</td>
</tr>
</tbody>
</table>

### Explanation

<table>
<thead>
<tr>
<th>Byte[16]</th>
<th>Explanation</th>
<th>Resolution</th>
<th>Offset</th>
<th>Operating range</th>
<th>Data range</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>WSU_subsystem_status</td>
<td>1</td>
<td>0</td>
<td>0 - 250</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>Software_version</td>
<td>1</td>
<td>0</td>
<td>0 - 250</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>Vehicles_in_platoon</td>
<td>1</td>
<td>1</td>
<td>1 - 10</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>Our_order_in_the_platoon</td>
<td>1</td>
<td>1</td>
<td>1 - 10</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>Automatic_mode</td>
<td>1</td>
<td>0</td>
<td>0 - 1</td>
<td>-</td>
</tr>
</tbody>
</table>

### RS-485 Bit Length

<table>
<thead>
<tr>
<th>Date</th>
<th>System</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>25/04/2011</td>
<td>RS-485</td>
<td></td>
</tr>
<tr>
<td>18/05/2011</td>
<td>RS-485</td>
<td></td>
</tr>
</tbody>
</table>

---

Joakim Kjellberg  
April 20, 2011  
C Electrical Engineering  
Systems and Control
### A.4 Platooning_input

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Tmin</th>
<th>Tmax</th>
<th>PGN</th>
<th>Default Priority</th>
<th>DPext</th>
<th>DP</th>
<th>Group Extension</th>
<th>Source</th>
<th>Destination</th>
<th>Standard/proprietary</th>
<th>System</th>
<th>Extension</th>
<th>Source</th>
<th>Destination</th>
<th>Standard/proprietary</th>
</tr>
</thead>
<tbody>
<tr>
<td>18FF0201</td>
<td>0.1</td>
<td>0.1</td>
<td>FF02</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>02</td>
<td>01</td>
<td>01</td>
<td>Prop</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S e r i e s</th>
<th>R e s e r v e d</th>
<th>B y t e</th>
<th>L e n g t h</th>
<th>Explanation</th>
<th>S t a t e</th>
<th>R e s o l u t i o n</th>
<th>O f f s e t</th>
<th>O p e r a t i n g r a n g e</th>
<th>D a t a r a n g e</th>
<th>U n i t</th>
<th>N o t e</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>Join_request_answer</td>
<td>1</td>
<td>0</td>
<td>0 - 1</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Denied 0x0
- Accepted 0x1
- Error 0x2
- Not Available 0x3
### Appendix A. CAN Specifications for ECU and WSU

#### A.5 Road_side_unit_information

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Tmin</th>
<th>Tmax</th>
<th>PGN</th>
<th>Default Priority</th>
<th>DPext</th>
<th>DP</th>
<th>Group Extension</th>
<th>Source</th>
<th>Destination</th>
<th>Standard/proprietary</th>
</tr>
</thead>
<tbody>
<tr>
<td>18FF0101</td>
<td>0.1</td>
<td>0.1</td>
<td>FF01</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>01</td>
<td>01</td>
<td></td>
<td>Prop</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Byte</th>
<th>Byte</th>
<th>Length</th>
<th>Explanation</th>
<th>Resolution</th>
<th>Offset</th>
<th>Operating range</th>
<th>Data range</th>
<th>Unit</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>8</td>
<td>Road_speed_limit</td>
<td>0.15</td>
<td>0</td>
<td>0 - 37.5</td>
<td>m/s</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Error</td>
<td>0xFE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Not Available</td>
<td>0xFF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>8</td>
<td>Traffic_light_relative_position</td>
<td>1</td>
<td>0</td>
<td>0 - 250</td>
<td>m</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Error</td>
<td>0xFE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Not Available</td>
<td>0xFF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>8</td>
<td>Time_left_to_next_color</td>
<td>0.5</td>
<td>0</td>
<td>0 - 125</td>
<td>s</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Error</td>
<td>0xFE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Not Available</td>
<td>0xFF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>8</td>
<td>Time_left_to_second_next_color</td>
<td>0.5</td>
<td>0</td>
<td>0 - 125</td>
<td>s</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Error</td>
<td>0xFE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Not Available</td>
<td>0xFF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>4</td>
<td>Traffic_light_current_color</td>
<td>1</td>
<td>0</td>
<td>0 - 3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Red light</td>
<td>0x0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Yellow light</td>
<td>0x1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Green light</td>
<td>0x2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Error</td>
<td>0xE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Not Available</td>
<td>0xF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>4</td>
<td>Traffic_light_next_color</td>
<td>1</td>
<td>0</td>
<td>0 - 3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Red light</td>
<td>0x0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Yellow light</td>
<td>0x1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Green light</td>
<td>0x2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Error</td>
<td>0xE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Not Available</td>
<td>0xF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>4</td>
<td>Traffic_light_second_next_color</td>
<td>1</td>
<td>0</td>
<td>0 - 3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Red light</td>
<td>0x0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Yellow light</td>
<td>0x1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Green light</td>
<td>0x2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Error</td>
<td>0xE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Not Available</td>
<td>0xF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Joakim Kjellberg
April 20, 2011
E Electrical Engineering
Systems and Control
### A.6 Vehicle_ahead

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Tmin</th>
<th>Tmax</th>
<th>PGN</th>
<th>Default</th>
<th>Priority</th>
<th>DPext</th>
<th>DP</th>
<th>Group</th>
<th>Extension</th>
<th>Source</th>
<th>Destination</th>
<th>Standard/proprietary</th>
</tr>
</thead>
<tbody>
<tr>
<td>18FF1001</td>
<td>0.1</td>
<td>0.1</td>
<td>FF10</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>10</td>
<td>01</td>
<td>Prop</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Byte</th>
<th>Length</th>
<th>Explanation</th>
<th>Resolution</th>
<th>Offset</th>
<th>Operating range</th>
<th>Data range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Relative_position</td>
<td>0.1</td>
<td>-3212</td>
<td>-3212 - 3212</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Absolute_velocity</td>
<td>0.0025</td>
<td>0</td>
<td>0 - 160.6375</td>
<td>m/s</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>Absolute_acceleration</td>
<td>0.1</td>
<td>-12.5</td>
<td>-12.5 - 12.5</td>
<td>m/s²</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>Vehicle_ID</td>
<td>1</td>
<td>1</td>
<td>1 - 11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>Platoon_ID</td>
<td>1</td>
<td>1</td>
<td>1 - 11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>Position_accuracy</td>
<td>0.1</td>
<td>0</td>
<td>0 - 25</td>
<td>m</td>
<td></td>
</tr>
</tbody>
</table>

Joakim Kjellberg  
April 20, 2011  
F Electrical Engineering  
Systems and Control
## A.7 Vehicle_1_information_1

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Tmin</th>
<th>Tmax</th>
<th>PGN</th>
<th>Default Priority</th>
<th>DPext</th>
<th>DP</th>
<th>Group Extension</th>
<th>Source</th>
<th>Destination</th>
<th>Standard/proprietary</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF1101</td>
<td>0.1</td>
<td>0.1</td>
<td>FF11</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>11</td>
<td>00</td>
<td>Prop</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sensor Byte</th>
<th>Bit Length</th>
<th>Explanation</th>
<th>Resolution</th>
<th>Offset</th>
<th>Data range</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 16</td>
<td></td>
<td>Relative_position</td>
<td>0.1</td>
<td>-3212</td>
<td>-3212 - 3212</td>
<td>m</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Error</td>
<td></td>
<td>0xExxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 1 16</td>
<td></td>
<td>Information_timestamp</td>
<td>0.01</td>
<td>0</td>
<td>0 - 642.55</td>
<td>s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Error</td>
<td></td>
<td>0xExxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 1 16</td>
<td></td>
<td>Absolute_velocity</td>
<td>0.0025</td>
<td>0</td>
<td>0 - 160.6375</td>
<td>m/s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Error</td>
<td></td>
<td>0xExxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 1 8</td>
<td></td>
<td>Absolute_acceleration</td>
<td>0.1</td>
<td>-12.5</td>
<td>-12.5 - 12.5</td>
<td>m/s²</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Error</td>
<td></td>
<td>0xFE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Joakim Kjellberg  
April 20, 2011  
G Electrical Engineering  
Systems and Control
## A.8 Vehicle information

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Tmin</th>
<th>Tmax</th>
<th>PGN</th>
<th>Default (Priority)</th>
<th>DPext</th>
<th>DP</th>
<th>Group Extension</th>
<th>Source</th>
<th>Destination</th>
<th>Standard/proprietary</th>
</tr>
</thead>
<tbody>
<tr>
<td>18FF1201</td>
<td>0.1</td>
<td>0.1</td>
<td>FF12</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>12</td>
<td>01</td>
<td>Prop</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sector</th>
<th>Byte</th>
<th>Length</th>
<th>Explanation</th>
<th>Resolution</th>
<th>Offset</th>
<th>Operating range</th>
<th>Data range</th>
<th>Unit</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 4</td>
<td>Vehicle_ID</td>
<td>1 1</td>
<td>1 - 11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Error: 0xE
Not Available: 0xF

<table>
<thead>
<tr>
<th>Sector</th>
<th>Byte</th>
<th>Length</th>
<th>Explanation</th>
<th>Resolution</th>
<th>Offset</th>
<th>Operating range</th>
<th>Data range</th>
<th>Unit</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 5 4</td>
<td>Platoon_ID</td>
<td>1 1</td>
<td>1 - 11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Error: 0xE
Not Available: 0xF

<table>
<thead>
<tr>
<th>Sector</th>
<th>Byte</th>
<th>Length</th>
<th>Position_accuracy</th>
<th>Resolution</th>
<th>Offset</th>
<th>Operating range</th>
<th>Data range</th>
<th>Unit</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 1 8</td>
<td>0.1</td>
<td>0</td>
<td>0 - 25</td>
<td>0</td>
<td></td>
<td>m</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Error: 0xFE
Not Available: 0xFF
A.9 Vehicle_2_information_1 through Vehicle_10_information_2

All messages called Vehicle_X_information_1 are the same as Vehicle_1_information_1, while all messages called Vehicle_X_information_2 are the same as Vehicle_1_information_2, except for the message identifiers. Each message ID is shown in the table below:

<table>
<thead>
<tr>
<th>Message</th>
<th>Identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vehicle_2_information_1</td>
<td>0x18FF2101</td>
</tr>
<tr>
<td>Vehicle_2_information_2</td>
<td>0x18FF2201</td>
</tr>
<tr>
<td>Vehicle_3_information_1</td>
<td>0x18FF3101</td>
</tr>
<tr>
<td>Vehicle_3_information_2</td>
<td>0x18FF3201</td>
</tr>
<tr>
<td>Vehicle_4_information_1</td>
<td>0x18FF4101</td>
</tr>
<tr>
<td>Vehicle_4_information_2</td>
<td>0x18FF4201</td>
</tr>
<tr>
<td>Vehicle_5_information_1</td>
<td>0x18FF5101</td>
</tr>
<tr>
<td>Vehicle_5_information_2</td>
<td>0x18FF5201</td>
</tr>
<tr>
<td>Vehicle_6_information_1</td>
<td>0x18FF6101</td>
</tr>
<tr>
<td>Vehicle_6_information_2</td>
<td>0x18FF6201</td>
</tr>
<tr>
<td>Vehicle_7_information_1</td>
<td>0x18FF7101</td>
</tr>
<tr>
<td>Vehicle_7_information_2</td>
<td>0x18FF7201</td>
</tr>
<tr>
<td>Vehicle_8_information_1</td>
<td>0x18FF8101</td>
</tr>
<tr>
<td>Vehicle_8_information_2</td>
<td>0x18FF8201</td>
</tr>
<tr>
<td>Vehicle_9_information_1</td>
<td>0x18FF9101</td>
</tr>
<tr>
<td>Vehicle_9_information_2</td>
<td>0x18FF9201</td>
</tr>
<tr>
<td>Vehicle_10_information_1</td>
<td>0x18FFA101</td>
</tr>
<tr>
<td>Vehicle_10_information_2</td>
<td>0x18FFA201</td>
</tr>
</tbody>
</table>