Automated Event Control of Vehicles for Drivability Testing

John Kroon

LUND UNIVERSITY

Department of Automatic Control

MSc Thesis
ISRN LUTFD2/TFRT--5989--SE
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Printed in Sweden by Tryckeriet i E-huset
Lund 2015

Abstract

Driving comfort is subjective; every driver has their set of preferences on how a car should behave. As the ability to control and suppress unwanted disturbances has increased with increased technological abilities, the demands on driving comfort in modern cars also rise. To account for these new demands, the testing and evaluation of the driving comfort becomes increasingly important for car manufacturers.

This thesis focuses on how to implement different benchmark tests of an automatic gearbox into the Drivebox, a tool developed at IAV that can act as the accelerator pedal in a car, and the creation of an interface between the Drivebox and the test database Caltet. The final goal with the new implementation of the Drivebox is an easier and more repeatable execution of benchmark tests. In order to verify that the interface and the implemented tests work, the test were first simulated with the help of a computer and then performed in a vehicle. The results from both the simulation and the tests in the vehicle show that it is possible to control and execute the benchmark tests using this new interface with the Drivebox.

Acknowledgements

First I would like to show my gratitude towards all those people that helped me with this thesis. Special thanks goes to my supervisor Felix and IAV for organizing and supporting me during this project. I would also like to thank the fellow students at IAV for great company and helping me whenever I needed. My last and greatest gratitude goes to the person that made everything possible and who was my greatest support throughout the entire thesis, thank you Imke!

Contents

List of Figures

1. Introduction

Car manufacturers are always trying to improve their products. In a highly competitive business all parts matter. One of these parts in the automotive business is the driving experience. New technological inventions allow a more precise and increased possibility of suppressing disturbances. In order to reach the highest comfort level a lot of testing is required since more parameters can be measured and adjusted. This also enhances the need for repeatability of the tests so the progress of the new settings can be measured.

1.1 Goal of the Thesis

IAV wants to make benchmark tests for the comfort of a gear shift in cars with an automatic gearbox more reliable and repeatable (Matthies, von Rüden, Kahlbau, Wurm, & Petzke, 2014). This requires that the tests are performed in a specified way and that they are repeatedly executed in the same way. This thesis aims at improving a tool, the Drivebox (DB), which can act as the accelerator pedal (AP) in a car (Gust, 2012) (Pogede, 2014). The aim of the thesis is that the DB can perform the benchmark tests and be controlled from Caltet, a program that the test driver uses to supervise the test.

This thesis covers the implementation of the benchmark tests in the DB and a new interface between the DB and Caltet. At the moment the DB is controlled via a touch screen and can be used to run parts of the tests. To reduce the number of tools that the test driver must control and communicate with during the tests the test driver should only have to interact with Caltet. To achieve this all communication with the DB is supposed to run in the background of Caltet and should not be directly visible to the test driver. The screen was used during the initial testing as it helped forming the concept for how the tests are specified and

executed. This setup is therefore mentioned in the thesis as background information.

1.2 Benchmark Tests

For evaluating the gear shifts a series of different benchmark tests are performed in the car. The tests have alternating external conditions, different types of gear shifts, various driving style modes and alternating AP values. For example, the test should be performed on even ground and be a power upshift in sports mode with 30% AP from 1st to 2nd gear. The benchmark tests are used during the entire calibration process, as they form the basis reference used to verify the improvement of the calibration. It is therefore important that the tests are performed in the same way each time. Currently the tests are performed by a test driver that drives a set sequence and evaluates the shifts by rating them on a subjective scale from 1 to 10, where 10 is optimal. To increase the accuracy in the evaluation of the test, the tests need to be more automated. This allows the test driver to just supervise the test and evaluate the gear shifts. A step in this direction is to relieve the driver of having to control the AP during the tests and instead let the AP be controlled by the DB. There are several reasons for this:

- Increase accuracy of the pedal movement
- Less tasks distracting the driver
- Increase repeatability of tests
- May decrease test time

The drawbacks for using the DB are:

- The tests requires more time to be specified
- The driving might be unnatural
- The DB cannot perform all tests

The DB cannot control the brake of the car, and this would be necessary to be able to perform all tests with the DB. To engage the kick down mode, which is used for fast accelerations, an extra input signal is usually needed. Since the DB only sends a signal of the pedal position it cannot perform these tests either. Despite these short comings the DB can still perform the majority of the tests.

1.3 Problem Formulations

The DB can currently perform smaller tasks which are manually controlled by the test driver. In order to perform the benchmark tests the DB needs to be able to perform test sequences. To make the DB more user friendly it should be able to receive commands from a computer and inform the computer of its current state. Caltet is always run during the tests and should, therefore, be modified so control of the DB is possible. The interface between Caltet and the DB should use a CAN bus for transmitting the data. This leads to the following problem formulations:

- Create a way of executing benchmark tests in the DB
- Create an interface between Caltet and DB
- Create control options in Caltet for the DB
- Verify tests in a simulated environment and in a car

As the scope of the thesis can be wide these limitations are set:

- Caltet should be able to:
    - Start/stop a test sequence
    - Tell DB which test to run
    - Send CAN configuration to DB
    - Change PID parameters in DB
    - Request DB to learn pedal mapping
    - Show status in DB
    - Modify tests
- The DB should be able to:
    - Perform accelerate in the shift , coast upshift, power up- and downshift events
    - Receive and interpret orders from Caltet
    - Notify Caltet of errors

There are more features and tests that could be implemented but as the thesis work runs for constrained time focus have been kept on the most essential parts. This should provide a good foundation for future improvements.

1.4 Thesis Outline

In Chapter 2 background information regarding the automatic gearbox, the gear shift and the DB are provided. This is then followed by the method in Chapter 3, containing an explanation of how the tests were performed, how they were setup

and the implementation of hardware and software. The results from the performed tests are then presented in Chapter 4. Chapter 5 discusses the relevance and the importance of the results from Chapter 4 and the limitations of the tests and the implementation of the interface between Caltet and the DB. Finally, in Chapter 6, the thesis is concluded and the future of the DB discussed.

2. Background

It is hard to define exactly what an acceptable level of comfort is since different drivers have a subjective feeling of the different levels of comfort. The comfort level in cars is also increasing, which changes the level of acceptance as well, but as a general rule the driver is supposed to feel as little as possible of the gear shift. There is one exception to this rule though, and that occurs if the car is driven in a sporty driving style. The driver has then decided to abandon comfort for performance. The driver might even be disappointed if the shifts are not noticeable enough as it can be seen as reducing the joy of driving.

The human body is sensitive to the change of acceleration (Strandemar, 2005), also known as jerk, the amount of jerk during the shift determines whether the shift is comfortable or not. In order to minimize the jerk the acceleration should be kept at a constant level throughout the gear shift. This means that the gearbox should be able to transmit power during the gearshift to maintain a constant acceleration (Genta & Morello, 2009). In a manual car the power transmission is interrupted during the shift, because the clutch disengages the entire gearbox from the engine. This makes it impossible to transfer any torque during the shift. In the automatic gearbox a combination of clutches and brakes change the gear ratio. If the engagement and disengagement of the gears overlap the gearbox is able to transmit power during the entire gearshift and thereby create a smoother shift. The overlap is achieved by allowing the clutches to slip for a short period during the gear shift. Both gears are then engaged during the overlap, and torque can therefore be transmitted during the shift. It is therefore very hard to achieve a shift with a manual gearbox that is as smooth as a shift with a well calibrated automatic gearbox.

2.1 Automatic Gearbox

Modern automatic gearboxes typically consist of a number of planetary gears, a torque converter, brakes and clutches. By combining the different planetary gears in different ways automatic gearboxes can have up to 9 different gears. The planetary gears are assembled as trains, where each planetary gear helps with changing the gear ratio (Naunheimer, Bertsche, Ryborz, & Novak, 2011). An illustration of a planetary gear can be seen in *Figure 2-1*. It consists of a sun gear (1), planetary gears (2), a spider (3) and a ring gear (4). The sun gear is fixed on an axle that can rotate but is fixed in space. Around the sun gear smaller gears rotate, they are called planetary gears. These gears are connected together with the spider. The spider has at least three planetary gears connected to it, and it rotates around the sun axis. The planetary gears can rotate around their own axis on the spider. The last part is the ring gear which encapsulates both the sun gear and the planetary gears. To create a transmission either the sun gear, the spider or the ring gear have to be fixed while the other two are allowed to rotate, one of the rotating axels acts as the input axis and the other one as the output axis of the gear. By altering the fixed axis new gear ratios can be achieved. In a car all combinations of the gear ratios cannot be utilized since the orientation of the gear is fixed.

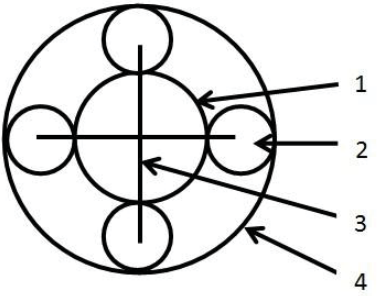

Figure 2-1 Illustration of a planetary gear and components.

The torque converter, seen in *Figure 2-2*, consists of an impeller (1), turbine (2) and a stator (3). The torque converter is hydraulically coupled, which means that the power transmission from input shaft to output shaft is done with a fluid. The impeller is connected to the motor/input shaft and the turbine to the gearbox/output shaft of the torque converter. The impeller drives a fluid around inside the torque converter, the fluid hits the blades of the turbine and the force from the fluid is transferred through the gearbox to the output shaft of the gearbox. Since the input shaft and the output shaft of the torque converter are not directly

connected, the engine can maintain idle speed while the first gear is engaged and the car is standing still. Torque is still transferred through the gearbox even if the AP is set to zero. To keep the turbine still a braking force needs to be applied on the output shaft. When this occurs the torque converter is stalling, as the turbine is not allowed to move. The constant torque transfer is also the reason why cars with an automatic gearbox have a hard time standing still when idling. The stator's role is to redirect the fluid between the impeller and the turbine allowing the torque to be converted. The stator is mounted on to the housing of the gearbox with a one way bearing, allowing it to spin but only in one direction. Most modern torque converters also have a lockup clutch installed. The lockup clutch can connect the impeller to the turbine, the motor is then directly connected to the gearbox. This increases the efficiency of the torque converter since the losses from transforming the torque is reduced (Naunheimer, Bertsche, Ryborz, & Novak, 2011). The torque converter can also be utilized as a damper in the shifting or when torsional vibrations occur. If the lock up clutch is controlled right the impeller can be released when dampening is needed. This allows for some slip in the torque converter which creates a smoother shift and increases the comfort level (Genta & Morello, 2009).

The brakes and clutches change the parts of the planetary gears that are locked. The new configuration of the planetary gears in the gearbox achieves a new gear ratio and the gearbox has thereby changed gear (Naunheimer, Bertsche, Ryborz, & Novak, 2011).

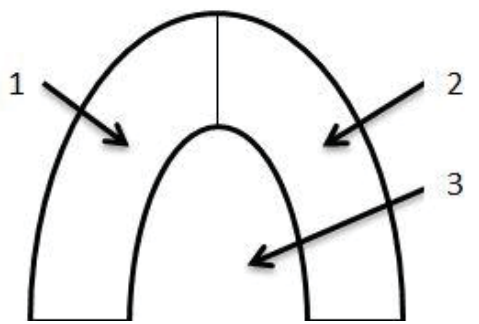

Figure 2-2 Illustration of the upper half of a torque converter.

Dual Clutch and Continuously Variable Transmission

The automatic gearbox described above is the most common automatic transmission but not the only one. Besides the automatic transmission the two

most common transmissions are the Dual Clutch and the Continuously Variable Transmission (CVT). The dual clutch is a split gearbox where the even gears are connected to one shaft and the odd gears are connected to another shaft. Both shafts have their own clutch connecting them to the engine. The shaft that is not currently transmitting any torque can here preselect the next gear before it is engaged. To switch into the next gear the car then only needs to engage the other clutch, if the TCU has selected the right gear. During the switch the clutches overlap in the handover which allows power to be transmitted during the entire shift (Naunheimer, Bertsche, Ryborz, & Novak, 2011).

The CVT's main part is the variator, see *Figure 2-3*. The variator usually consists of tapered discs and a chain. One set of discs are connected to the input shaft and the other set to the output shaft. By varying the axial distance between the discs the diameter for the chain is changed on each shaft which changes the gear ratio between the shafts. Since the distance between the discs can be varied without any steps the CVT has an infinite number of gears. For taking off it is also common for CVT to utilize a torque converter, like the automatic gearbox (Naunheimer, Bertsche, Ryborz, & Novak, 2011).

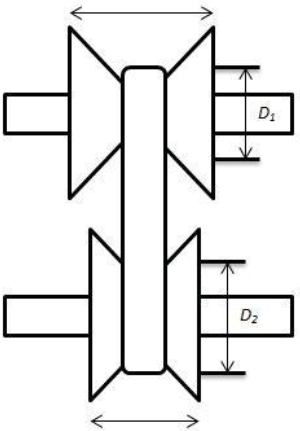

Figure 2-3 Illustration of a variator in the CVT.

Transmission Control Unit

The TCU is the brain of the automatic gearbox. It decides when shifts should occur and which gear should be engaged. When it has selected which gear to engage it sends signals to the brakes and clutches inside the gearbox to move to new positions. The new position of the clutches and brakes creates a new gear ratio between the input and output shaft, and a new gear is then engaged. The

TCU selects the new gear by comparing the speed of the car with the position of the AP in a shift map. This makes the AP's position the most important input to the TCU (Genta & Morello, 2009).

Shift maps are defined by mapping the AP position against the speed of the car for each gear shift. This creates a range for each gear, defining when the gear should be used or not. The shift map contains an up shift line and a down shift line for each forward gear, *Figure 2-4*. When the shift maps are created the downshift lines always have to be at a lower speed than their respective upshift line, otherwise an area with an undefined gear will occur. The shift to and from neutral and reverse are selected manually and therefore not considered in the shift map. The shift map can be optimized after a number of criteria e.g. fuel consumption, comfort and performance. The influence of different criteria depends on the selected driving style; common modes are e.g. Sport, Comfort, Winter etc. but some modern gearboxes can also adapt and change the shift map to the driver's preferences automatically. The shift map is adapted by measuring parameters that imply whether the driver is satisfied or not with the behavior of the car, see Shifting Strategy in Chapter 2.

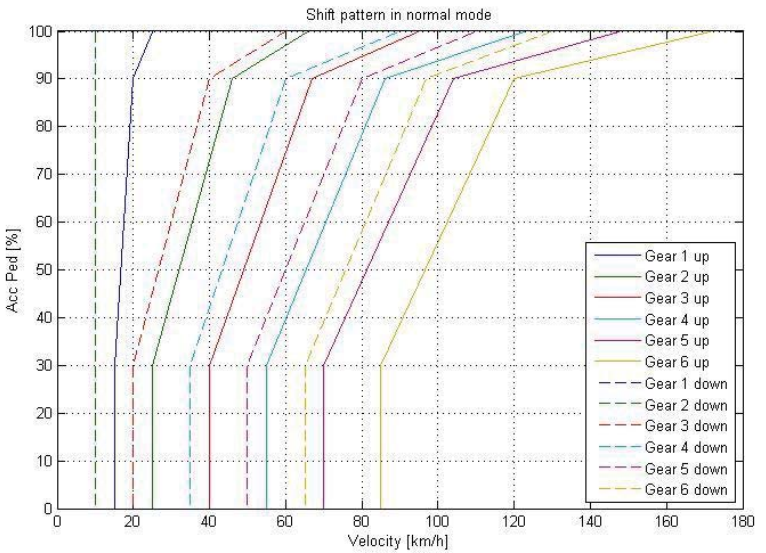

Figure 2-4 Example of a shift map with six gears.

2.2 Shifting Gears

There are several aspects for the TCU to consider when shifting gears in an automatic gearbox:

- The type of shift
- Which gear to engage
- When should the gear be engaged
- The driver's reaction

A good way of realizing the complexity of the shift is to compare it to the manual shift. In a manual car the driver usually eases off a bit on the AP when a shift is about to occur, so that the rotational speed of the engine does not increase significantly when it is separated from the gearbox. The clutch is pressed down around the same time or slightly after the AP is released. When the clutch has disengaged the gearbox from the engine the new gear is selected. To finish the shift the clutch is released and the engine and gearbox are connected again. When the clutch is released the AP is pushed down to match the speed of the engine with the speed of the car. A period of slipping then occurs in the clutch before the gear is fully engaged. If the speed difference of the shafts is to big when the clutch is released the car will accelerate or decelerate quickly, creating an uncomfortable shift.

Since the shifts are controlled by the TCU in a car with an automatic gearbox instead of the driver, the driver of an automatic car is usually not aware of when a gear shift will occur. Because of this, the acceleration pedal is usually kept at the same position during a shift but it can also be moved in an undesirable way. To minimize the jerk the TCU can therefore send requests to the ECU on how the engine should behave. It is done so that the speed of the engaging clutches and brakes can be matched with the release of the disengaging clutches and brakes. The control of the engine speed is usually achieved by changing the fuel injection or the ignition timing. This overlapping with a correct amount of slip and matching of the speed of the shafts is tricky and should work well at different rotational speeds and types of gear shifts.

When shifting manually the driver is aware of the surrounding traffic. This may cause the driver to reevaluate and cancel a planned shift. The TCU must therefore also be able to handle these types of events and still perform a comfortable shift. As cars get equipped with more sensors it makes it easier to predict these unexpected events. An example of an unexpected event is when a slower vehicle in front of the car forces it to decelerate. The driver then reduces

the speed and the TCU initiates a coast downshift. During the shift an opportunity to overtake the other vehicle occurs and the AP is therefore rapidly pushed down. The TCU must now decide whether to go through with the down shift or go back to the previous gear. If the TCU decides to cancel the gear shift the response from the car might be slow as it will have to accelerate from a high gear. A down shift on the other hand could result in a very high engine speed. Regardless of the choice the TCU should still be able to perform a pleasant experience for the driver.

Types of Shifts

When driving forward there are four kinds of different shifts that can occur in the automatic gearbox:

- Power upshift
- Power downshift
- Coast upshift
- Coast downshift

During the power shifts a positive torque is applied from the engine through the gearbox. The power upshift occurs when the car is accelerating, as the speed increases the shift-up line is crossed in the shift map, which causes the TCU to shift into the next gear. The power downshift can occur in two different ways. First, if the car is driven at a relatively slow speed for the gear and the accelerator pedal is set to a high value the down shift line will be passed in the shift map and the TCU shifts down to provide a faster acceleration. The second case when the power downshift occurs is when the car is decelerating even though the motor provides a positive torque. This usually happens when driving up a steep hill. The speed is then reduced and the down shift line is crossed. The TCU then shifts down so that a higher torque can be applied to the wheels from the motor.

During the coast shifts the motor is providing a negative torque to the gearbox. The coast upshift can occur in two ways. First, when an acceleration is aborted, the car's momentum will make it accelerate after the AP has been released or set to a low value. If the speed is high enough as the AP is released the up-shift line is crossed in the shift map, causing an up shift to occur. The second way of creating a coast upshift occurs when driving down a steep hill with a low AP value. If the hill is steep enough the speed of the car will increase. When the speed increases and the upshift line in the shift map is crossed an upshift will occur.

Lastly, the coast downshift occurs when the car is decelerating and the accelerator pedal set to a low or idle value.

A Gear Shift

At lot of research have been spent on how to achieve a comfortable shift in the automatic gearbox. One of the areas where most effort has been spent is on optimizing the control of the clutches and brakes during the shifts. To optimize the shift with regards to the comfort the clutches should engage/disengage with a torque handover. In a sequential gear shift only two clutches are active at the same time. If the engaging clutch is too late there will be a power drop in the shift and if the disengaging clutch is too late a negative torque will be generated. During the shift there are three different phases: the oil filling phases, the torque phase and the speed phase. During the oil phase the engaging clutch is filled with oil and the pressure is built up so the clutch can start to transfer torque. The power transfer from the new gear is non-existing during this phase. During the torque phase the clutches start the handover of the power transmission while still running in the lower gear ratio. The speed phase is where the higher gear ratio is engaged. This is the most important phase since the inertia from the car has to be matched with the torque of the engine. To make the transition less notable the torque produced in the engine is adapted by changing the airflow and the spark timing (Guo, Wang, Su, Li, Xu, & Cui, 2014).

Shifting Strategy

The modern gearbox uses a lot of strategies to determine how it should behave in order to please the driver. Normal driving can cause problems because of external factors, a slope for example. When driving up a slope the force needed to maintain the same speed is increased, which causes the car to decelerate if the AP is held at the same position. To compensate for this the driver pushes the AP down and the torque from the engine is increased. When the AP is pushed down the TCU also decides to shift down to give a faster acceleration. The driver lets the acceleration pedal up after the sought speed is achieved. This can then cause an upshift in TCU as the accelerator pedal is kept at a lower value with a higher speed. When the new gear is engaged the new gear ratio cannot maintain the same speed as the lower gear could, so the car decelerates. This makes the driver push the acceleration pedal down again and a downshift occurs. If this becomes a cycle it can be very frustrating for the driver. To compensate for this the shift lines are

temporarily moved when a slope is detected, a higher speed is then required for an upshift (Genta & Morello, 2009).

By observing how the accelerator pedal is moved it is possible to determine how the driver drives and they expect the car to behave. A driver that has a more sporty driving style usually makes big shifts in the AP whereas a driver that wants to save fuel and drive more comfortable keeps the pedal at lower consistent values (Genta & Morello, 2009). If the driver moves the AP to accelerate and the car does not respond fast enough the AP will be pushed down further. This indicates that the driver would prefer a faster response from the car. The interpreted signal from the AP is then changed so the pedal gives a higher output for a small movement. The mapping of the AP is therefore usually not linear, instead it is adapted to the driving style.

Most modern cars have a kick down that can be engaged by pushing the AP over a small resistance before the AP is fully pushed down. When this mode is engaged the driver expects a fast acceleration from the car. In order to achieve this the TCU shifts down so the engine can provide a higher torque on the wheels, hence the name kick down.

2.3 The Drivebox

The DB has been developed at IAV for precise control of the AP signal in a car with a drive-by-wire system. It can choose to send the analog signal from the actual pedal through or replace it with a signal of its own (Pogede, 2014) (Gust, 2012). This section contains a technical description of the DB and it also describes the initial implementation in the DB and how it was used before the thesis. Changes done to the software are discussed in the Method chapter.

Hardware

The DB can be seen in *Figure 2-5*. The front of the DB is dominated by the touch screen. On the top side a power switch and a USB-connector are located and on the bottom from left to right a CAN-bus connector, a power connection, and the output connector for the pedal can be found. The processing unit in the DB is a LPC2478 microcontroller and the following interfaces are used with the microcontroller (IAV Gmbh, 2014):

- MMC/SD-Card
- 2 CAN buses
- LCD Display

- AD-Converter
- DA-Converter
- JTAG Programmer
- USB

The SD-Card is used as a hard drive where CAN configurations, AP mappings and test sequences are stored. Each time the DB is started it reads the name of the last used configuration and then loads the CAN configuration and the AP mapping for this configuration. Originally only one of the CAN buses was implemented in the software. This bus is dedicated to reading messages from the car's control units, referred to as car bus from now on. Being able to read the car bus makes it possible for the DB to update the states of the parameters of the car. This is vital as the shift events are triggered by the state of the car. The DB has an LCD-display with touch function allowing the user to interact with it and choose which test to run. The AD/DA converter is used for regulating the analog output of the DB and to read the position of the AP. The JTAG programmer interface is necessary for uploading software on to the microcontroller. The USB interface can be used to read and write to the SD card from an external computer.

Figure 2-5 The DB seen from the front, top and bottom side.

Microcontroller

The processor in the DB is a microcontroller using a Real Time Operation System (RTOS), called Power Pac created by IAR Systems. Since the DB only has one processor the tasks cannot be executed in parallel, instead it has to jump between the different tasks after a section of the code has been executed. The RTOS therefore creates a system that emulates the behavior of a multicore CPU.

After the initialization of the hardware and RTOS three tasks are started, see *Figure 2-6*. The main task reads the configure file for the CAN signals on the car bus and the pedal map for the AP. It then creates the Graphic User Interface (GUI) and waits for a button event. The user can select a number of different operation modes by clicking on the different buttons on the screen. The touch task waits for an input on the screen, it then decides which button was pressed and signals a button event to the main task. The main task can then retrieve which button was pushed and then execute the action requested by the user. If the DB wants to control the AP an "OP_on" event is signaled by the main thread. The "OP_on" event causes the output task to wake up and it then checks if the brake is released. If the brake is released, the operation mode is turned on. In this mode the DB replaces the AP signal with a signal of its own. The DB takes over the control of the AP by switching two relays and then sending the requested voltage to AD/DA converter which outputs the current to the ECU.

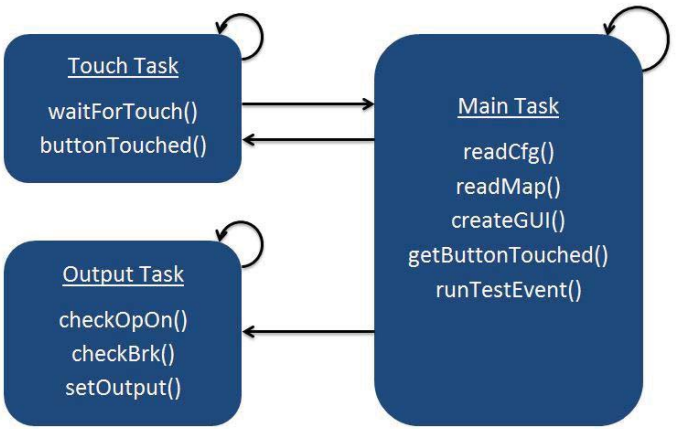

Figure 2-6 Description of the three tasks in the DB while using the screen.

The signal of the AP varies between different car brands and car models. This requires that the DB can adapt to different pedal mappings. The DB can automatically read the mapping of a pedal by sending a signal to the ECU and then reading the position of the pedal via the car bus. To get the entire mapping the DB sends a predefined voltage to the ECU. It is usually in the 20-50% of the AP range of most cars. The DB then lowers the output until it reads back 0% on the car bus. When the 0% level is found it is saved, and then the DB increases the output and saves the value for each percent until a 100%. The mapping of the pedal is then complete and the mapping is stored on the SD-card so it can be used

without having to remap the pedal. The DB does not keep a register over the pedal mappings used in different cars, it only remembers the pedal map last used. (Gust, 2012).

Safety Features

Since unexpected events can occur during a test it is important that the test can be aborted while it is running. To ensure this the DB continuously checks the value of the brake pedal. If the pedal is being used the DB leaves the operation mode and returns the control of the AP to the test driver. The relays, that allow the DB to exchange the real value from the AP against a generated signal inside the DB, are set to let the signal from the pedal through as default setting. They will return to the default setting if there is a power failure, so if the power to the DB is cut the AP can be used as normal.

CAN Bus

The Controller Area Network (CAN) bus is a serial bus system used throughout large parts of the modern car industry. The bus is used by the control units to exchange information about their sensors and the control unit's current state. One example of this is the AP value. The raw value from the AP is read in the ECU but other control units like the TCU, for example, needs to know the value of the AP to make a correct gear shift. The ECU therefore sends this information on the CAN bus so it can be read by the TCU. Transceivers and receivers are referred to as nodes on the CAN bus. All nodes are allowed to transmit data on the bus and the bus also allows multiple nodes to read the data at the same time.

To avoid collisions in the data traffic the CAN bus uses Carrier Sense Multiple Access Collision Detection (CSMA/CD) (Fijalkowski, 2011). This means that the nodes listen to the bus before they start transmitting. If the bus is silent the nodes are allowed to try and send. The ID of a message lets the other nodes decide if they want the read the message or not, and it is also used as a way of prioritizing different messages. Nodes does not have different priorities to send but the messages being sent on the bus do. A low ID means that the message has a high priority (Robert Bosch Gmbh, 1991).

The DB can utilize two different CAN buses for communication, when the screen is used the DB only uses one bus and when it is disabled both buses are used. The first bus or the car bus is directly connected to the control units in the car and the second bus or the control bus is used for communicating between Caltet and the DB. The bus connected to the car contains many signals and since

not all of them might be known to the user of the DB no transmission can be allowed on this bus. This is done for security reasons since a message might appear to be unused but it may cause unexpected behavior of the car. To communicate with the DB the control bus is used instead. On this bus all signals are known and separated from the car bus. Transmitting on this bus can therefore be done without risking an unexpected behavior in the car.

Interaction with Surroundings

Since the DB needs to communicate with Caltet, it needs to be able to read the messages from Caltet. To read the messages on the control bus two options exists for how this could be implemented: the DB could poll the bus or use interrupts. Polling means that the program repeatedly checks if a new message has arrived until the sought message is received. An interrupt on the other hand is caused by the external device. The hardware detects the interrupt, halts the current execution and saves the place of execution. The CPU then jumps to an Interrupt Service Routine (ISR) where the interrupt is handled. The ISR should be written as short as possible as the current execution has been interrupted. Typically information is just stored or updated in the ISR and then the CPU continues to execute at the same place where it was previously halted. The new information changes the state of the parameters stored in the DB which affects the execution of the program later on in the code. If polling is used and the system contains many external inputs the CPU needs to use a considerable amount of time checking the receive buffers and status of external inputs if no new changes has occurred checking the buffer will create a lot of unnecessary computation. In this case it is more efficient to use interrupts that update the state when new information is available (Årzén, 2012). When the system is less complex it is more a question of preference.

PID Controller

In the DB a PID controller is implemented which is used to regulate the speed of the vehicle and rotational speed of the engine in the car. It is primarily used when a shift event has been run and the DB is waiting for a grade confirmation or when a shift event requires an initial speed, e.g. a power downshift. A PID controller consists of three parts a Proportional, Integral and Derivative part, see Figure 2-7.

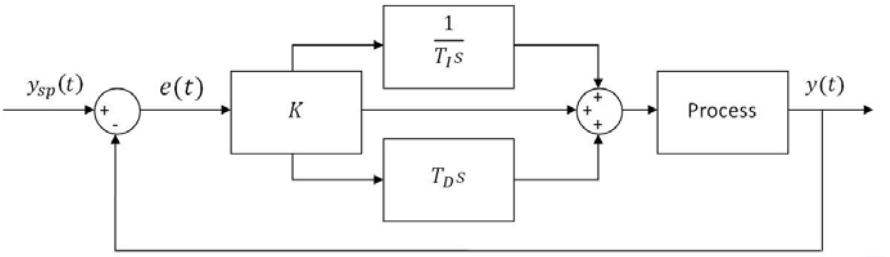

Figure 2-7 Block diagram of a PID Controller.

The PID controller can be described by (Årzén, 2012)

where $e(t) = y_{sp}(t) - y(t)$. y_{sp} is the desired value and $y(t)$ is current value of the process and the difference between them is then the control error, (t). K, T_I and T_D represents the proportional, integral and derivative gain. PID regulators use the error to determine how to regulate the output. The proportional term depends on the current error in the controller. The integral term uses the sum of previous errors and the derivate term depends on the extrapolation of future error. If only the proportional part of the controller is used a steady state error will typically occur. The regulator will come close to the desired value but will not be to keep the error at zero. To compensate for this problem the integral term is used. If a steady state error occurs the integral part will push the error towards zero as the errors accumulate. The integral part can cause a problem if the error is large. It will then saturate the output of the controller regardless of what the proportional part is. This will create an overshoot in the system as the integral part will continue to grow until the error switches sign. It might require a considerable amount of time until it the accumulated error recedes again. This phenomenon is known as integral windup, in the DB this has been compensated for by deducting the added error for the integral part when it saturates the controller. The derivative part tries to predict what is going to happen in the future and thereby improve the damping of an oscillatory system. The derivative part is sensitive to measurement noise and is often combined with a low-pass filter to negate this. The derivative part can magnify the noise and make the system unstable. It can be very hard to calibrate and it is therefore often not used (Årzén, 2012).

2.4 Software

To help visualize, interact, program and simulate a car during the development of the DB some software was utilized. This section gives a short overview of the different programs that have been used in the thesis and what they have been used for.

Velodyn

Velodyn is a Simulink based program developed at IAV for simulating the physical aspects of a car with an automatic gearbox (Matthies, von Rüden, Kahlbau, Wurm, & Petzke, 2014). Velodyn was used to simulate the network of the car in the test bench setup. It is used with Simulink's vehicle toolbox which enables the creation of a CAN interface to Velodyn. This creates a simulated car environment where the execution of the tests and the performance of the DB can be validated before it is put into an actual car. For the thesis a model of the car was provided by IAV. In *Figure 2-8* Velodyn and the model used during the simulations can be seen.

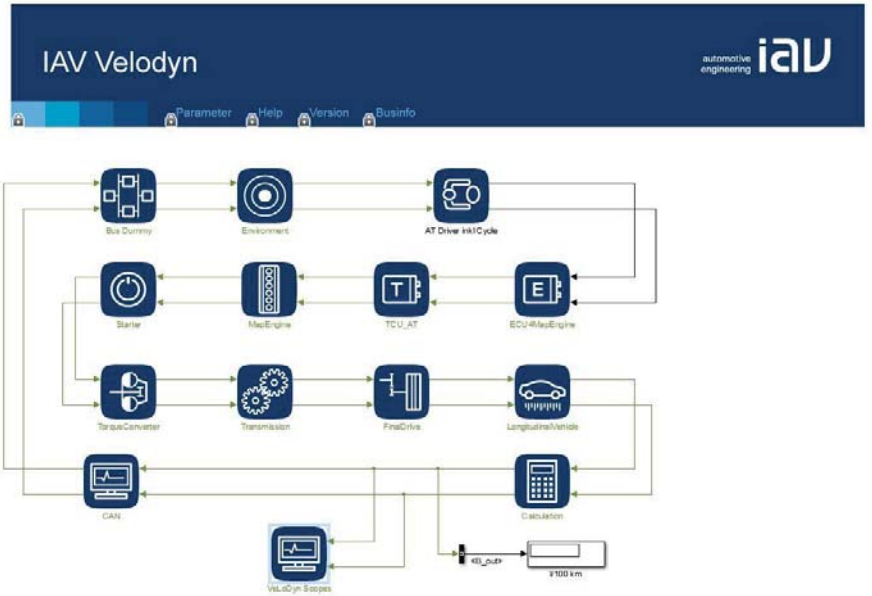

Figure 2-8 Illustration of Velodyn.

CANape

CANape is a program developed by Vector Infomatik. It is developed for calibrating parameters in the control units in the car via the CAN bus and the XCP protocol (Vector Infomatik, 2015). In this thesis it was used to read the messages on the car bus both in the simulation and the car and visualize them during the tests. Caltet also utilizes CANape to record the tests, as it has no direct connection to the car bus.

Caltet

Caltet is a Matlab based program developed by IAV for evaluating tests. It records the test driver experienced grade of the performance of each step in the test. With this data and the data recorded by CANape it creates a basis for analyzing how the gearbox should be tuned to increase the comfort in the gearshifts. As Caltet is used by the test driver to grade the shifts and to keep track of which shift event to run the end goal is that it can send the shift event to the DB. The DB then runs the shift event without requiring any extra inputs from the driver. This would help reduce the tasks the test driver has to perform and the number of instruments to survey. In *Figure 2-9* Caltet can be seen with the testing menu open.

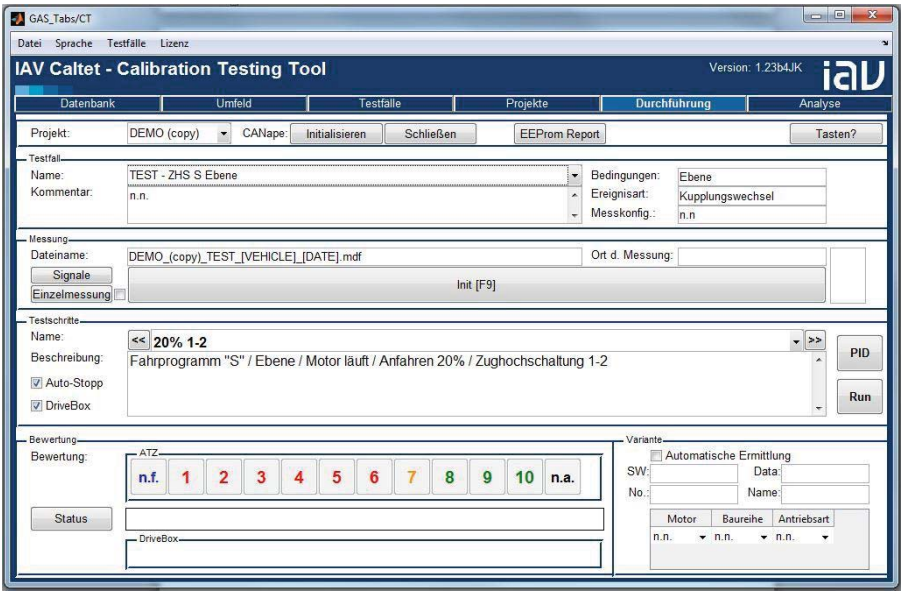

Figure 2-9 Illustration of Caltet.

IAR Embedded Workbench

IAR Embedded Workbench is an integrated development environment used to program microcontrollers. IAR Embedded Workbench was used to create the new software in the DB for configuring the control bus, implementation of the new test cases and the interface between Caltet and the DB. It supports C based programming and allows the programmer to debug the code.

3. Method

At first the tests and shift events needed to be specified so they could be implemented into the DB. When it was verified that the DB can perform the different test sequences the implementation of the interface between Caltet and the DB began. Once the interface was specified and implemented test could be performed to increase the performance of the execution of the tests. When the tests had been verified in a simulated environment the tests were tried in a real car for improvement and final validation.

3.1 Tests and Shift Events

Tests are created for evaluating the current settings in one specific type of gear shift, usually for all gears in the gearbox at set external conditions and with varying AP values. A shift event in the DB defines a single gear shift. The DB does not check any external conditions before the shift event is executed. The test driver is responsible for ensuring that these conditions are met. A test is built up by several shift events. Typically, one test consists of one type of shift event for all forward gears of the gearbox with alternating values for the AP.

Shift Events

A shift event starts when the TCU decides to change gear and ends when the new gear is fully engaged and the previous gear is disengaged or when the gear shift is aborted. This means that the shift events have to be defined so the car executes the desired shift. In order to achieve the sought shift with the DB, a shift event usually consists of these seven steps:

- Set speed to reach start gear
- When start gear is reached maintain speed to stabilize
- When stable, change AP value

- Maintain new value for the AP until gear shift occurs
- Gear shift
- Target gear reached, shifting event finished
- Maintain similar speed until shift is evaluated

The structure of how a shift event is executed is based on this pattern. All shift events start in the same way: the desired speed and gear, if defined, is set. The type of shift event is then selected and executed. Sometimes the tests are not always defined with the vehicle speed. In Caltet they can instead be specified with a rotational speed of the engine. After the shift event is finished the DB regulates the speed of the vehicle to the last read value until all gears have been evaluated.

Specifying a Shift Event

The shift events should be specified so that just the desired shift occurs, regardless of whether the entire test is executed or just one shift event. When specifying a shift event great care must be given to the shift map, as it defines when shifts occur. From the shift map the speed or the rotational speed and the AP position needed to evoke a shift event can be read. In order to specify a shift event the DB also needs to know which type of shift event should be executed. Lastly, the DB needs to know between which gears the shift event occurs. This means that six parameters are needed to specify a shift event. For the accelerate-in-the-shift tests one special seventh parameter is needed, the delay until the AP is changed. The parameters are:

EventType: Tells which type of shift event will be executed
AccPed: Tells which value the AP should have during the shift event.
GearCurr: Tells which gear the shift event will start in.
GearTar: Tells the target gear for the shift event.
vVeh: Tells what speed in km/h the car should have before shift event.
EngSpeed: Tells at what engine speed in rpm the AP should be released.
TilTime: Tells how many milliseconds the DB should wait until new AP is set.

As most parameters represent a numerical value they are self explanatory but an explanation to the different values of the EventType parameter have been provided in *Table 1*. The EventType value that the DB reads is a number and can be seen in the EventType column. The user of Caltet is not expected to know all these numbers, an abbriviation for each EventType is therefore shown in Caltet

instead. The abbriviation can be seen in the second column and in the last column the full name of the event is shown.

Table 1 Table of the different values for the EventType parameter.

EventType Value	Abbriviation English/German	Type of shift event
1	POU/ZHS	Power upshift
2	POD/ZRS	Power downshift
3	CSU/SHS	Coast upshift
4	CSD/SRS	Coast downshift
5	CDAS/SRSGS	Coast downshift, accelerate in shift
6	CSTPO/KSTZHS	Const speed 10s accelerate until next gear
7	CUAS/SHSGS	Coast upshift, accelerate in shift
8	TO/TO	Maintain const AP until timeout
9	MAN	Regulate on rpm and wait for manual shift

When the DB was used with the touch screen the tests were initially saved on the SD-card. As the screen was removed the shift events kept the same structure but instead of reading the information from the memory the shift events are sent with a message via the control bus from Caltet. The entire test is also no longer run in sequence, with an exception for the power upshift (an explanation for this follows later in the text). Each shift event is instead executed when it is received by the DB. This is done to create a more flexible structure. The test driver might want to skip parts of the test that has been done earlier or just redo one step that went to fast or was hard to grade.

Power Upshift

In the power upshift the DB outputs the AP value as soon as the brake has been released, if the right gear is engaged. If another gear is engaged than the one selected for the start of the test, the speed of the vehicle is first regulated to the value specified in "AccPed" until it is stable. When this speed has been reached the right gear should also be engaged. The test then runs until the highest gear has been engaged with the set AP value. If the AP value is too low to reach the higher gears a timeout will be signaled if no new gear has been engaged within 20s. The grade of the shift can be saved under the running test so the test driver has the opportunity to grade the shift immediately. The implementation also allows the driver to select which gear to start in, which means that each test can be run from any gear to last the gear.

The power upshift is the only test where all shift events are executed without interruption. The DB does not wait for the test driver to grade each shift between the shifts. The reason for the special execution comes from the speed regulation after each shift event. After a shift event has been performed the DB will normally remain in control of the AP until the shift has been graded or if the entire test is finished or cancelled. When a power upshift event has been completed a quick release of the AP will occur as the PID controller tries to regulate the speed. As the car is still accelerating it might cross another shift line causing a coast upshift. The coast upshift then occurs close to the power upshift which can make it hard to distinguish and evaluate the right shift.

It is impossible to avoid this problem if a high AP value is set since the car obtains a high momentum before the shift occurs. An example of this shown in *Figure 3-1* the car accelerates with 80% AP value until 40km/h were a power up-shift occurs. If the speed set after the event is set to 40km/h the car's momentum will make it accelerate to a higher velocity. To compensate for this the PID controller sets the AP value to 0%, as the red arrow shows, but then a coast up-shift will follow immediately after the power upshift giving the test driver no time to distinguish when the first shift ends and the second shift begins. In order to give some delay between the gear shifts a higher speed reference can be set after the shift event is finished, shown by the black arrow. As none of these options were suitable it was decided that it would be better to execute all steps at once keeping the AP value at 80% until the last gear is reached, shown by the blue arrow.

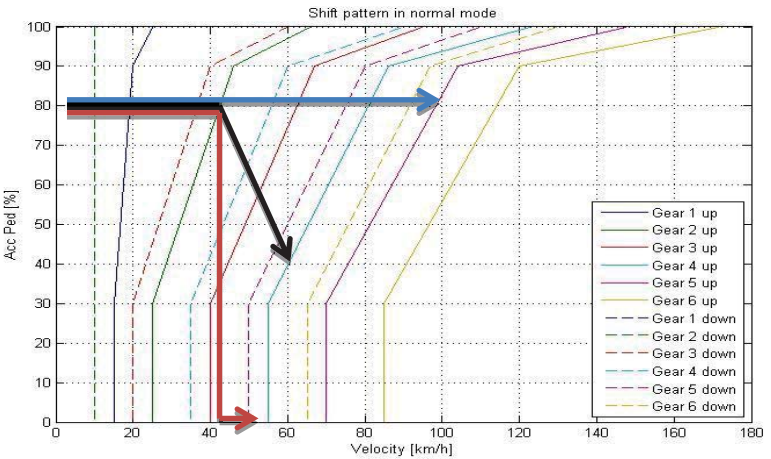

Figure 3-1 Example of three different ways of implementing a power up-shift.

Power Downshift

In the power downshift the car is first regulated to a speed that is slightly above the down shift line. When the speed has been stabilized and the right gear is engaged the AP value is changed. If the value is set above 30% it is interpreted as a down shift on a plain road and if it is set to a value below 30% the test is assumed to be carried out driving up a hill. The two cases are handled slightly differently. In the first case, when the acceleration occurs on a plain road, the down shift should occur almost immediately after the pedal has been pressed down. If the down shift is not initiated within 500ms after the pedal is pressed down it is considered a failed test. The DB then sends a timeout signal to Caltet that informs the driver that the shift will never occur. When driving up the slope it will take a longer time before the shift occurs. The timeout has therefore been extended and will occur after 20s if the target gear is not reached. Otherwise the shift events are identical.

Coast Up/Down Shift

For the coast upshift the DB accelerates with a set AP value until the rotational speed of the motor in the car reaches the specified value. If a lower gear than the starting gear is engaged when the coast upshift is started the car will first regulate towards the set vehicle speed until the right gear is engaged. When the engine speed has been reached the DB starts to decrease the value of the AP slowly. This is done to mimic the behavior of a human driver. If the AP is quickly released it indicates that the driver want to brake. The car then wants to maximize the motor braking and the TCU therefore stops an upshift from occurring. If the pedal is released slowly the TCU realizes that this is the speed that the driver wants to drive with and allows an upshift to occur. If no upshift occurs within 3s after the AP is released a timeout is sent to Caltet as no coast upshift is considered possible.

The coast downshift cannot really be performed with the DB since the DB cannot control the brake. The implementation of the coast downshift therefore only consists of a speed regulation to engage the start gear and when this has been achieved the control of the car is returned to the test driver. The down shifts are performed with varying deceleration which requires the use of the brake, therefore the use of the DB is impossible for this shift event.

Acceleration in the Shift

The implementations of these tests are similar to the previously mentioned coast shifts with the exception of an extra acceleration in the shift. It is hard for the test

driver to execute the acceleration in the shift event repeatedly with the same delay for each shift, because the driver has to realize when the shift starts and then estimate the delay until the acceleration. The DB on the other hand can see when the shift target is changed and within the microcontroller is an accurate clock which makes it possible to hit different or the same delay after the shift has started. After the specified delay the DB will set the AP to the specified value for 500ms and then regulate the speed of the car until the shift has been evaluated.

Timeout, Constant Speed and Manual

In the timeout event the AP is held at a set value for the entire test. The car accelerates from a standstill until a timeout occurs or the highest gear is engaged. Each time a gearshift occurs the clock for the timeout is reset, if no new gearshift has happened within 20s a timeout occurs. For the constant speed shift the car is driven at a set speed for ten seconds before an acceleration occurs. The shift event ends when the next upshift has been performed. In the manual shift event the car is driven at a set engine speed. When the engine speed is right a message is sent to Caltet and the color field in Caltet turns yellow indicating that the driver should shift gear. The driver then selects whether an up or downshift should occur. The shift event is finished when the specified target gear has been engaged.

3.2 Test Bench

The test bench is created to see how the DB reacts in a simulated car environment before it is tried in an actual car. This is done because it creates a safe environment and provides an easy way of testing and developing the implemented code.

Setup with Screen

The communication within the test bench is setup as in *Figure 3-2* when the screen is utilized. In the test bench Velodyn simulates how the car network behaves by sending the current state of the car via the car bus to the DB and CANape. In the DB the state of the car determines how the AP should be moved. When the DB has determined what the position of the AP should be it sends an analog signal to the A/D converter. The A/D converter reads the analog value and sends a CAN message containing the position of the AP back to the Velodyn model. Velodyn simulates the impact of the new AP position and sends an updated state of the car on the car bus. The DB behavior can therefore be observed as it

would be located in an actual car. CANape is used as a visual feedback where the state of the car can be read, which makes it easier to understand how the DB behaves during the simulation.

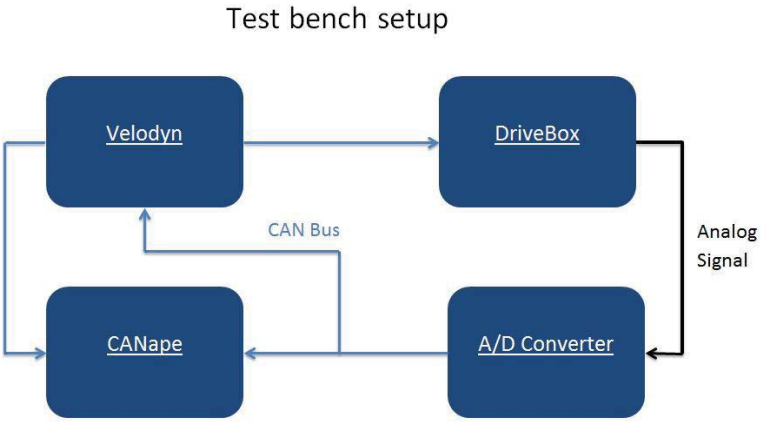

Figure 3-2 Layout of the communication in the test bench with screen enabled.

In the beginning the analog signal was disconnected to provide a good way of determining how the DB reacts to different states of the car, as the state of the car can be controlled from the Velodyn model. When the DB responded correctly and the program had the right graphic user interface (GUI) the DB was fully connected to the loop. After the DB was fully connected to the test bench the evaluation of how it performed the shift events where possible. When the shift events where executed at an acceptable level the next step for the DB was to disconnect the screen and execute the test with the help of an interface between Caltet and the DB.

Setup without Screen

For evaluating the DB when the screen is disabled the test bench needed to be modified somewhat. It is, however, very similar to the setup with the screen, see *Figure 3-3*. When the screen is removed the interaction between user and DB works in a different way but the communication between the DB and the car remains the same. As there is no way of giving any direct inputs to the DB, all communication with the user must go via Caltet. This is achieved by utilizing the control bus. To activate the control bus in the DB some of the output pins on the microcontroller that were allocated for the screen are used. This makes it impossible to use both screen and both buses at the same time. Caltet can already

control CANape which makes it possible to start and stop the measurements with Caltet.

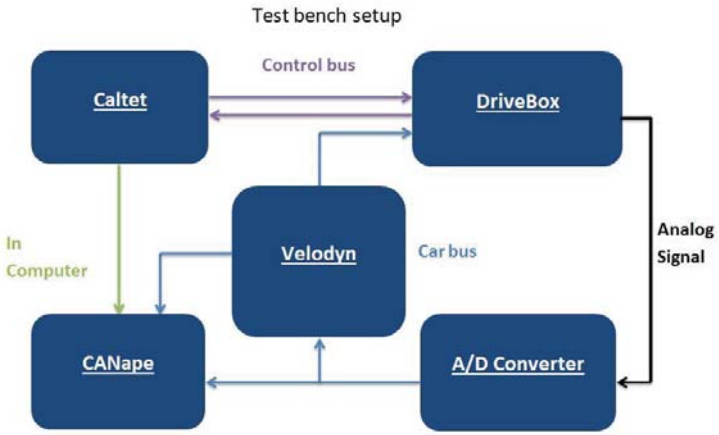

Figure 3-3 Layout of the communication in the test bench with screen disabled.

3.3 Drivebox

Before the start of the thesis the DB was already used for other types of tests but it did not contain any implementation of the shift events for the benchmark tests and it used the touch screen to interact with the user. The initial shift events were therefore controlled with the touch screen using a similar way of interacting with the DB as the other tests. When the shift events had been verified the next goal was to execute the test without using the screen. The implementation of the shift events is still based on the first experience gathered when the screen was used and the execution of the tests is therefore also similar to it.

When the DB is turned on the hardware is initiated and the control bus, pedal map and the PID controller are configured. *Figure 3-4* shows a description of how the DB is initiated, the reception of a message on the control bus and the work flow for the main task in the DB. After the initialization of the DB or when a task is finished the main task waits for a message event to occur. When a message is received the processor jumps to the ISR where the message's ID and data field are read. After the information has been stored a message event is signaled. If the main task is waiting for a message it is woken. The main task reads the ID and data of the message and selects the right mode or action. When the task is finished

or an error or a timeout occurs while executing the task the DB sends a message back to Caltet containing the information of what has happened.

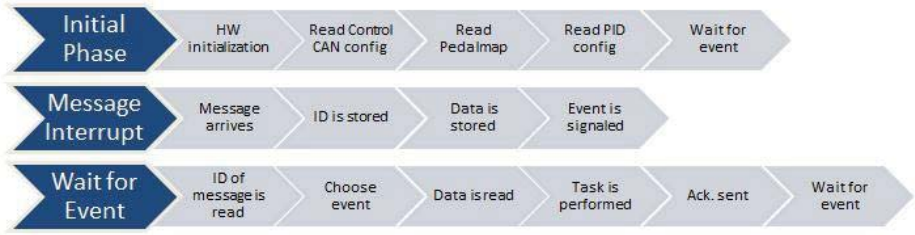

Figure 3-4 Work flow of the DB.

The DB utilizes polling to read the messages on the car bus. When information about the car's state is needed the DB reads the ID of the current message in the receive buffer. If it corresponds to the ID of the sought message the data is read, otherwise, the buffer is released and a new message is received which is checked in the same way. If no message is read within 500ms the car bus is considered to be offline. The DB then enters a never ending while loop and it has to be restarted and initialized before it can be used again.

The messages received on the control bus are instead read with the help of interrupts. When a message is received the ID and data of the message are saved and a message event is signaled. This allows the DB to put the main task into a sleep mode when it is waiting for orders from Caltet. During a test the DB is checking if the message event has been flagged so that a cancel signal from Caltet can be read.

The Different Modes

After a message event has been received in the DB it can cause the DB to enter five different modes: test, USB, PID, CAN and learn pedal map. In addition to these modes the DB can also return the values of the PID and send an echo for the handshake. The different modes are selected by the received message's ID and data. The DB first looks at the ID of the received message. Some modes have a unique message type: test, PID and CAN. Their mode is selected directly from read the ID of the message. The learn pedal map mode and USB mode are selected with an order message where the data inside the message tells which mode it should enter, see *Figure 3-5*. The echo for the handshake and the current PID settings are also demanded by reading the Order message data field.

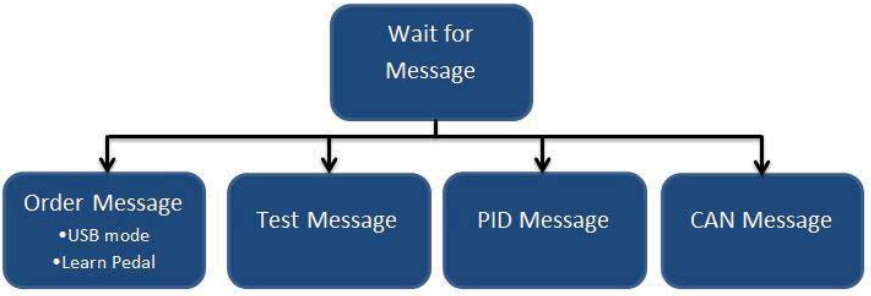

Figure 3-5 Illustration of the different messages received by the DB.

When a test message is received the data is read and split into the EventType, Gear curr, Gear tar, AccPed, vVeh, EngSpeed and TilTime parameters, which are the same parameters used to define a test. After this the shift event is setup, the DB then makes the car drive at the right speed or engine speed and if specified the right gear. After the right start conditions have been reached the type of shift event decides how the DB will control the car. There are nine different types of shift events implemented into the DB:

- Power upshift
- Power downshift
- Coast upshift
- Coast downshift
- Coast downshift, accelerate in shift
- Constant speed for 10s, new AP value until next gear
- Coast upshift, accelerate in shift
- Drive with constant AP value until timeout
- Manual shift at constant engine speed

During the execution of each shift event the DB checks if any messages have been sent from Caltet. If the DB receives a cancel signal the shift event is cancelled other messages are ignored and deleted, as Caltet should not send any other messages at this point. When a step is finished the DB sends a "step_finished" message to Caltet and waits for a new message from Caltet confirming to the DB that the step has been graded. When a confirmation is received the DB then returns and waits for a new event. After the confirmation of the grade has been sent and more shift events exists in the test. Caltet automatically sends the next shift event to the DB or if no more shift events exists a cancel signal is sent

instead. If a timeout should occur under the execution of the shift event the DB informs Caltet of the timeout instead of sending the "step_finished".

In the USB mode the SD-card of the DB can be accessed with a USB cable. This mode existed in the old version of the DB and has not been altered. It is used to change the files that are saved on the SD-card. Previously the different configurations for different vehicles and tests where saved here. When Caltet uses the interface it sends the configuration for the car bus and the shift events via the control bus which makes it unnecessary to save these in the DB. The need for the user to access this mode does not exist anymore. The USB mode is only used to change the definition of the messages of the interface between Caltet and the DB. It is therefore not something that the user should be able to access, only the programmer of the DB and Caltet needs to be able to access this mode if messages in the interface should be added or altered. There is therefore no option implemented in Caltet to reach this mode. Instead a command must be written on the control bus with another program telling the DB to go to this mode.

The PID mode is activated when a PID message is read by the DB. The DB then reads the different factors for the proportional, integral and derivative parts of the PID controller from the message. The new parameters replace the old and when the values have been updated they are saved on the SD-card so that they can be used if the DB is restarted without having to ask the user for new inputs. This is something the user needs to pay attention to each time the DB is used in a new vehicle, as a sports car behaves quite differently from a big van with the same PID values.

In the CAN mode the different signals read on the car bus are initialized. The DB reads all signals in one sequence it is therefore important that the messages are sent in the right order. If they are sent in the wrong order the DB will mistake them for another signal. Caltet therefore reads the information from the user and sorts it in the right order before it is sent. When all seven required messages have been received, the DB sends a "Signal_Updated" back to Caltet confirming that the DB is configured.

3.4 Interface

When the screen of the DB is disabled another way of communicating with the DB is necessary. An interface between the DB and Caltet has therefore been specified. It provides a structured way for the communication between the two programs. The interface is needed so each message is correctly interpreted in both

programs. It is declared as a list of the different CAN messages with defined signals.

Messages

The interface consists of five messages: Order, Test, PID, Box and CAN. The layout of signals in the messages can be seen in *Figure 3-6*. As the order and box message only contain one 8 bit signal the different information being sent with those messages is shown instead. The Order message contains information about how the user wants the DB to act. It is used to put the DB in a special mode like the USB or Learn Pedal mode and to communicate with the DB under tests. In the initiation of the DB it also requests an echo from the DB to confirm that the DB is connected to the control bus. The order message can also be used to request information from the DB about the current PID settings in the DB. The DB then sends the PID settings back to Caltet with the PID message. The PID message is used for transmitting both new and current parameters of the PID controller. It contains three signals Kp, Ki and Kd, one for each parameter of the PID controller.

The Test message is built up by signals that create the same structure as the shift event. Which makes it compatible with the previous implementation of a test. The difference here is that instead of reading the entire test at once from a file in the memory each shift event is sent from Caltet to the DB. When the DB receives a Test message it reads the data and splits it up in the different signals and saves these values. After the values have been saved the DB executes the requested shift event. The Box message is used by the DB to send information to Caltet. It contains information about where the DB is executing and if an error or timeout should occur under a test it informs Caltet about the problem.

The CAN message is used to configure the car bus. As the DB is used in several cars it needs to be able adjust the CAN configuration to each car. Caltet reads the .dbc file associated with the chosen project and searches through the file for the signals that the user has selected. When a signal has been found the information is put in a CAN message and transferred to the DB. The order of the messages is important as the DB expects them in a predefined order. In total seven different signals need to be specified in Caltet for the DB to work properly. Not all signals are used in the benchmark tests but they were used in the implementation without a screen. The signals were kept to make it easier to implement the old test cases in the new program.

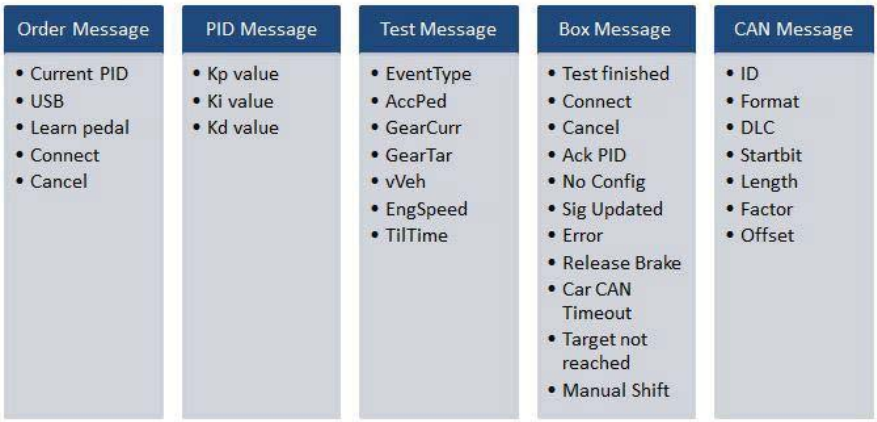

Figure 3-6 Messages in the interface.

Initialization between Caltet and DB

Before any messages can be sent to the DB Caltet needs to create a CAN channel and configure the messages for the interface. When the channel has been setup and started Caltet tries to interact with the DB, see *Figure 3-7*. Where the dotted lines represent the messages being sent between the different programs. The horizontal lines between the boxes mean that the program is waiting for a message before the execution continues. The first interaction between the DB and Caltet is a handshake to make sure that the DB is present on the control bus. The handshake is implemented as an echo. Caltet sends data to the DB with an Order message and waits for a response from the DB. The DB receives the message and then sends the same data back to Caltet in a Box message. If the DB is not connected or turned off there will be a passive error occurring on the bus, assuming that the DB is the only instrument connected to the bus. When a passive error occurs the hardware detects it and sends error frames in the CAN messages. The messages received in Caltet are checked for error frames and if error frames inside the messages are seen the user is asked to reconnect the DB or restart it. If the DB is correctly connected and the power is turned on the echo will be registered by Caltet and the initialization will continue. After the presence of the DB has been established Caltet sends the configuration for the car bus to the DB. In order to configure the DB seven signals have to be sent:

- Engine Speed
- Target Gear
- Current Gear

- Torque
- Brake
- Acceleration Pedal
- Vehicle Speed

If all the received signals ID's are higher than zero an acknowledgement is sent back to Caltet. If a message is missing the DB will send a "No_config" message after two seconds. If the DB is located in a new car it needs to configure the mapping of the accelerator pedal for the car. A prompt asks the user at each initiation to decide whether this is necessary or not. Before the DB is fully initialized it also reads the last saved PID settings and load these parameters. After this the initialization of the DB is done. The DB then waits for a message from Caltet telling it which task to perform.

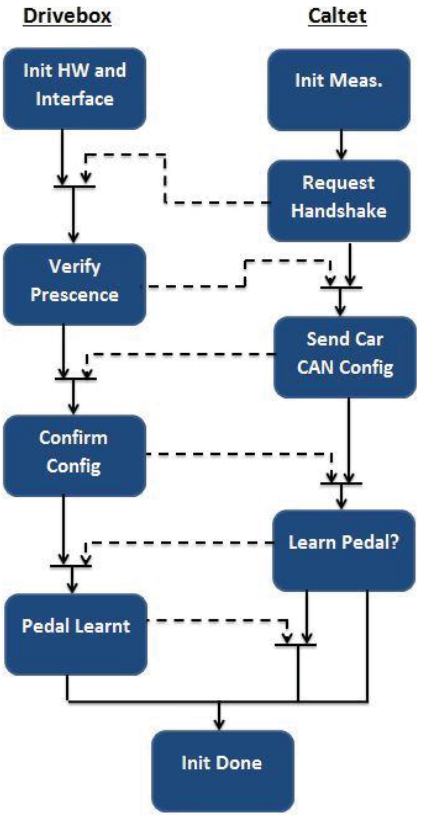

Figure 3-7 Interaction between the DB and Caltet during the initiation.

Request/Send PID Parameters

The interaction between the DB and Caltet when new PID parameters are set is illustrated in *Figure 3-8*. First, the PID button in Caltet is pressed, an order message telling the DB to send the PID parameters back to Caltet is then sent to the DB. The DB sends the parameters and goes back to wait for a new message event. Caltet receives the parameters and show them to the user. If the user presses the ok button the new parameters are sent with a PID message to the DB. The DB saves and updates the parameters and thereafter sends an acknowledgement to Caltet that it has received the new PID parameters.

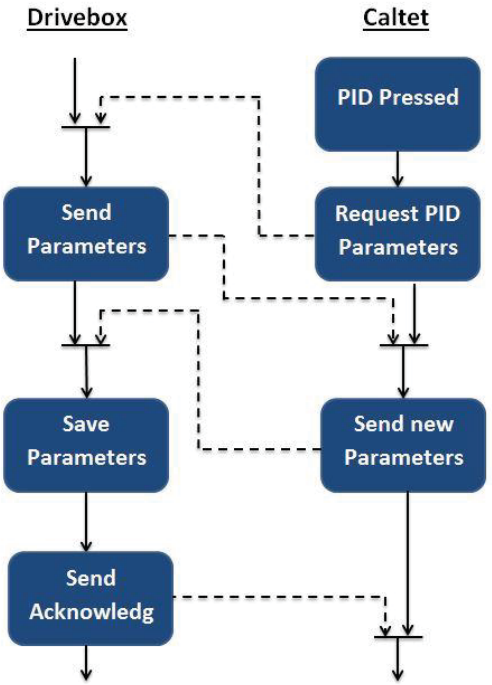

Figure 3-8 Interaction between the DB and Caltet for the PID button.

Run and Cancel Test

When the Run button is pressed the selected shift event is sent from Caltet to the DB, as seen in *Figure 3-9*. The DB is waiting for an order from Caltet. The order is first read and interpreted then the execution of the shift event starts. When the shift event has finished or timed out the DB sends an acknowledgement or a timeout signal back to Caltet. The DB then regulates the speed and waits for the

next shift event. Once the shift event has been graded in Caltet the next shift event is sent to the DB. To indicate that a test has finished a cancel signal is sent to the DB. It then returns the control of the AP to the driver and goes back to the wait for new message state.

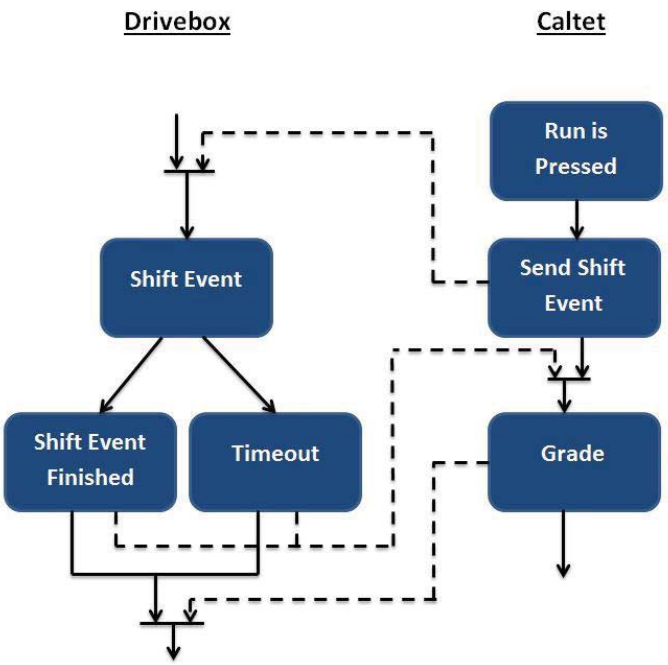

Figure 3-9 Interaction between the DB and Caltet for the run button.

During the execution of the shift event the driver must be able to cancel the running shift event. The run button switches label and becomes a cancel button after it has been pressed. When the cancel button is pressed a message is sent to the DB that breaks the current execution, see *Figure 3-10*. The cancel button can be pressed at any time when the DB is in control of the AP signal and it should always break the control of the DB.

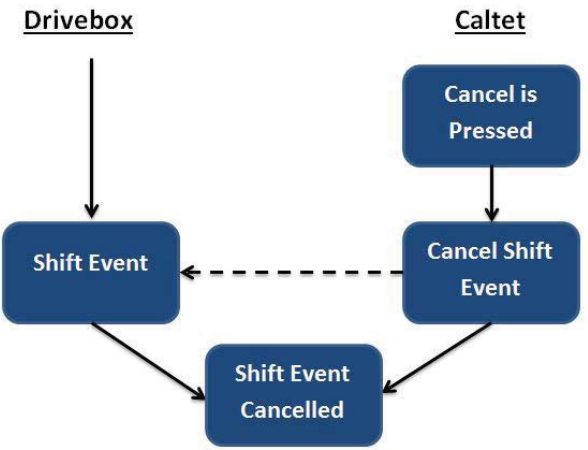

Figure 3-10 Interaction between the DB and Caltet for the cancel button.

3.5 Caltet

With the new interface several changes to Caltet are needed. The user must still be aware of what is going on with the DB and be able to send commands to it. The interface also needs to be implemented so that the communication between the DB and Caltet works.

GUI

In order to control the DB from Caltet some changes to the GUI in Caltet are needed. The user must be able to be informed of the state of the DB, select whether the DB should be used or not, change the PID parameters and start/stop the execution of a shift event. As the DB is used when the tests are run most changes have been implemented in the Testing menu but the new way of specifying the shift events also requires that the Testcases menu is slightly altered. The new fields that contain the parameters specifying a shift event can now be displayed for each shift event in the Teststep menu in the Testcases panel, *Figure 3-11*. The change of the GUI is indicated with the red box.

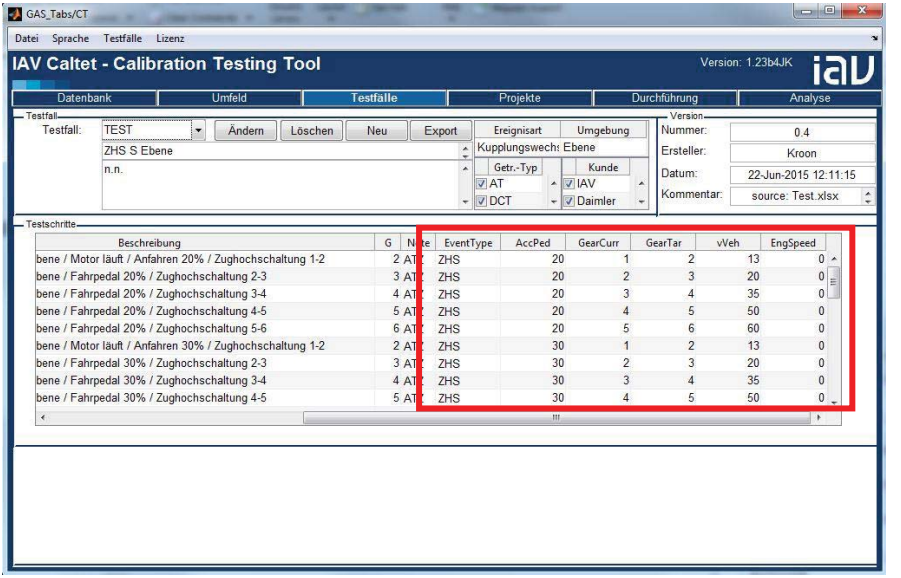

Figure 3-11 Illustration of the new values for the shift event in Caltet.

To inform the driver of the state of the DB a new textbox has been implemented in the bottom of the Evaluate panel of the Testing menu. In this textbox errors can be displayed while the test is executing and requests to grade the shift event will occur after a shift event is completed. On the right hand side of the test steps panel a PID and a Run button have been added. The PID button opens a menu where the current values of the PID regulator are shown. To change the values of the PID controller new values can be written in the fields and when the user pushes ok the values are sent to the DB. The Run button is used to start the execution of the shift event, after being pushed it changes into a cancel button. During a shift event it can then be used to abort the execution of the shift event. The colored field in the measurement panel also shifts color for different states. If the test is being run the field is green, when a shift event has occurred and a request for a grade appears the field turns blue. If an error or timeout occurs during the test the field turns red and when the measurement is halted the field turns orange. To select whether the DB should be used or not a checkbox has been created on the left hand side of the test steps panel. All changes done to the Testing menu can be seen in *Figure 3-12*. Changes made to the GUI are indicated with the red boxes.

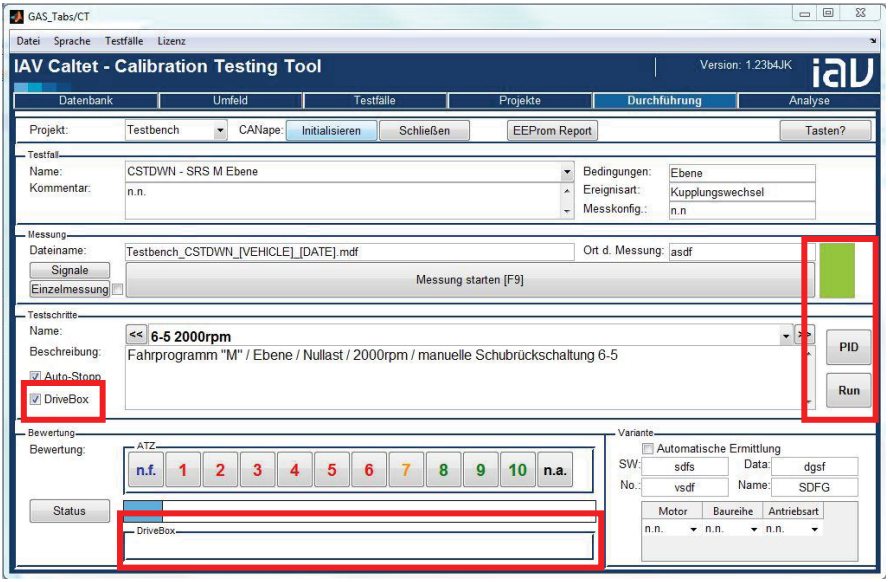

Figure 3-12 Illustration of the new GUI for the Testing menu in Caltet.

When a test is being run Caltet can enter three different states run, grade and timeout. In all states the AP is controlled by the DB. In order to enter the Run state the measurement in CANape must be running, so that data can be recorded when the test is executing. In the run state Caltet first sends a teststep to the DB and then waits for a message from the DB. If it receives a "Step_Finished" message Caltet enters the grade mode and waits for the user to evaluate the shift. While waiting the speed of the car is kept about the same speed it had when the shift finished. After the step has been graded Caltet sends the next step and reenters the run state again. When the last step of a test is finished the measurement in CANape is halted and no new shift event is sent to the DB. Instead a cancel signal is sent so the DB returns the control of the car to the test driver. If a timeout or an error should occur Caltet enters the timeout state. Here the DB maintains the last read speed on car bus and an error message is shown in the DB dialog window. The DB then waits for a message from Caltet.

Before the DB can be used it needs to be initialized, see Initialization. The initialization can be triggered in two ways: with the Init button in the measurement panel or with the initialize button in the project panel, see *Figure 3-12*.

Implementation of Interface

To communicate with the DB Caltet also needs to create a way of interacting via the control bus. The Vehicle Network Toolbox in Matlab supports the creation of a CAN interface. Before any interaction between Caltet and the DB occur a CAN channel and messages are created. The structure of the messages is specified in the control_CAN.dbc file, this file is required in the Caltet folder in order to make the interface work properly. To use the channel new functions for communication were implemented. Caltet needs to be able to both send and receive messages but not for all messages. The order and test messages are just sent from Caltet to the DB and therefore only needs a send function. Box messages are only received by Caltet and therefore only needs a receive message function. The PID message can be both sent from and received by Caltet which makes it the only message with both a send and receive function. To receive a message Caltet uses polling. A problem with the CAN channel in Matlab is that it records its own sent message and stores it in the receive buffer. This is the reason why the DB has its own message. If Caltet is looking for a message that it cannot send the problem with reading its own messages is averted otherwise the sought message might be erased or mistaken for the message just sent from Caltet. Each function just reads the expected message e.g. the receive box messages function only reads messages of the type box message.

3.6 Test in Car

To validate that the DB, Caltet and the interface between the programs work in a car, it has to be tested. One of IAV's test vehicles was used for this purpose. Being equipped with a dual clutch gear, it allows the program to be used in a real application where the interaction between the user and the program differs from the simulated environment. The brake is utilized more in the real process then in the simulation, as it is has to be used to keep the vehicle still and sometimes to avoid physical objects. The setup of the communication in the test environment is similar to the previous tests, see *Figure 3-13*. Velodyn and the AD converter are no longer need to simulate the behavior of the car as they have been replaced with the real car.

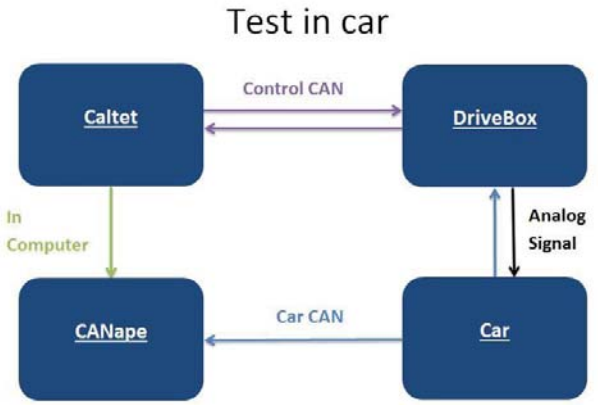

Figure 3-13 Layout of communication when the DB is tested in a car.

4. Results

Due to a lack of time all tests could not be verified in the car. The most important tests where therefore selected: the power upshift, power downshift, coast upshift and accelerate in the shift. This chapter contains the result from both the simulation and the tests performed in the car.

A shift can be identified by looking at the upper graph of each test. When the target gear is changed it means that the TCU has initiated the gear shift. While the current gear and the target gear are separated from each other the shift is occurring and once the current gear is set to the same level as the target gear the shift is finished. The model used in Velodyn is simplified and does not support a simulation of the engine speed but it simulates the speed of the transmission. For the coast upshift the rotational speed of the transmission was used instead of the engine speed. The transmission speed has a direct relationship to the speed of the vehicle since it is connected to the wheels. The engine speed is therefore not shown in the diagrams from the simulation, since it will have the same shape as the speed of the vehicle when it is normalized.

4.1 Power Upshift

In the power upshift the brake is first released and the acceleration pedal value is set. The pedal is kept at a constant level until the last gear is reached. When the last gear has been engaged the DB waits for the driver to fill out the results of the test. During this waiting period the PID controller regulates the speed of the car. The test driver is relative quick with entering the grades for the simulation test, see *Figure 4-1*, but at in the end of the diagram the AP is pushed down a little bit before the test is entirely graded. Which indicates that the PID controller starts to regulate the speed. When the test is run in the real process, see *Figure 4-2*, it behaves as the simulated process until an external event forces the driver to end the test. The test was run until the 5th gear, at 37s the TCU wants to shift into the

6th gear but as the test is aborted and the car decelerates the upshift is cancelled, the TCU signals a cancelled gearshift by setting the gear target to 14.

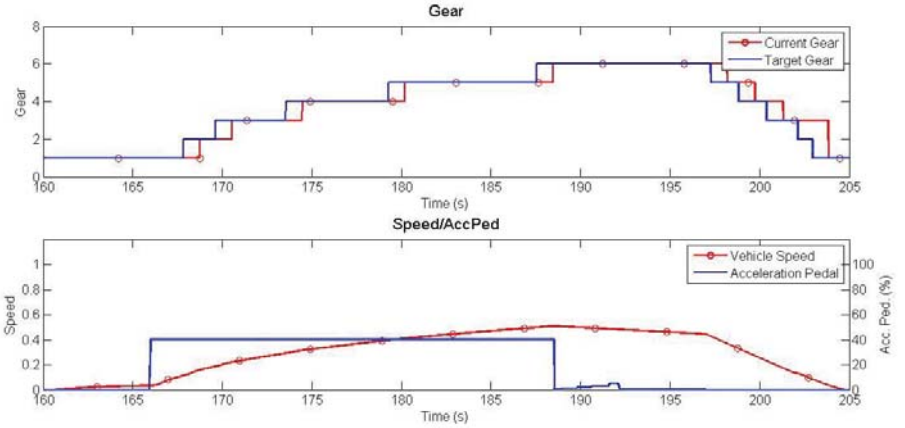

Figure 4-1 The power upshift in testbench with 40% AP value.

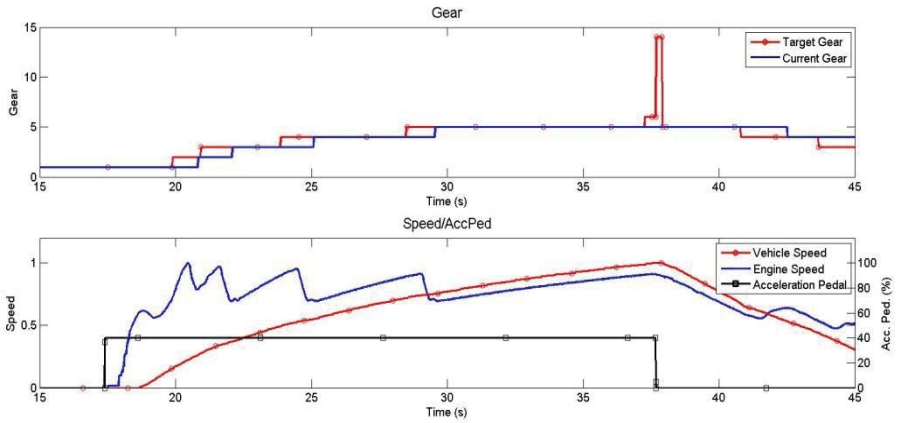

Figure 4-2 The power upshift test in car with 40% AP value.

4.2 Power Downshift

In the power downshift the right gear is first engaged and the speed is regulated to a low speed for that gear. The AP is then pushed down to a set value and held in this position until 2s after the shift has finished. In *Figure 4-3* the power downshift in the simulation can be seen and the same event but with 80% AP is shown in

Figure 4-4 for the test in the car. The need for a big difference in the AP value between the simulation and the car stems from different shift maps.

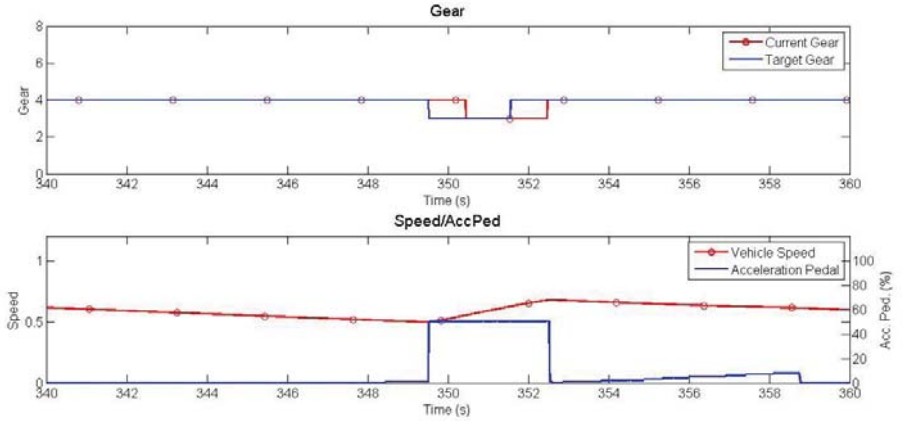

Figure 4-3 The power downshift from 4th to 3rd gear with 50% AP in testbench.

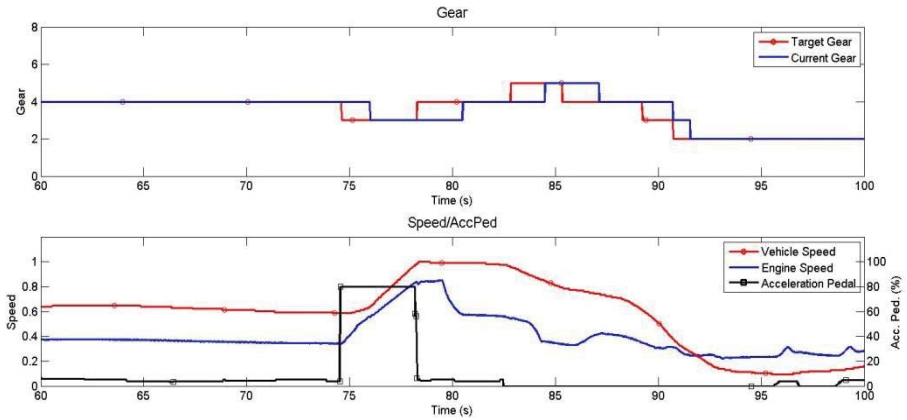

Figure 4-4 The power downshift from 4th to 3rd gear with 80% AP in car.

4.3 Coast Upshift

In *Figure 4-5* and *Figure 4-6* the coast upshift event from the 3rd to the 4th gear is shown. The 3rd gear has been engaged and the car accelerates to a set engine speed. When the rotational speed has been reached the AP is slowly released. After a short while the car shifts up to the 4th gear.

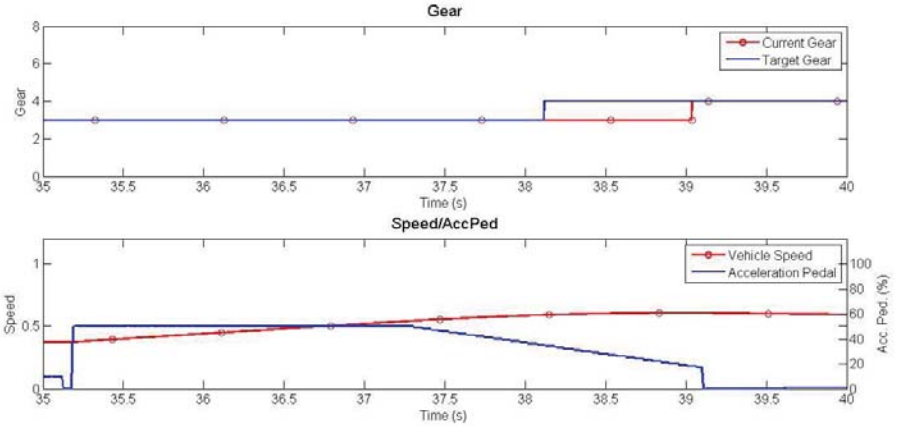

Figure 4-5 The coast upshift from 3rd to 4th gear in testbench.

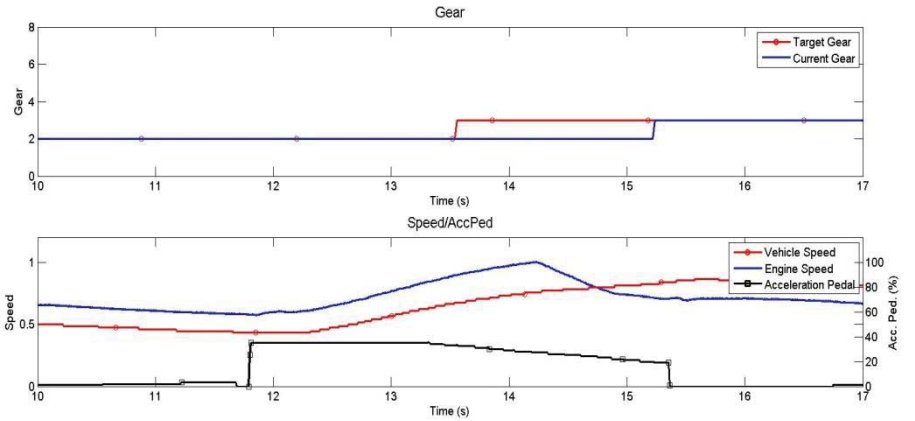

Figure 4-6 The coast upshift from 3st to 4th gear in car.

4.4 Accelerate in Shift

In the accelerate-in-the shift event, the most important factor is the delay after the shift has been initiated until the acceleration occurs. In the test performed in the simulation the delay until the acceleration is specified to 600ms and can be seen in *Figure 4-7*. From the measurement it can be noted that the actual delay until the acceleration in the execution is ~630ms. After the delay the DB should then keep the AP value at 60% for 500ms. 600ms proved to be a relative long time for the shift in the real process so the delay was changed to 300ms. In *Figure 4-8* it can be

noted that the delay until the acceleration is slightly longer than the specified value.

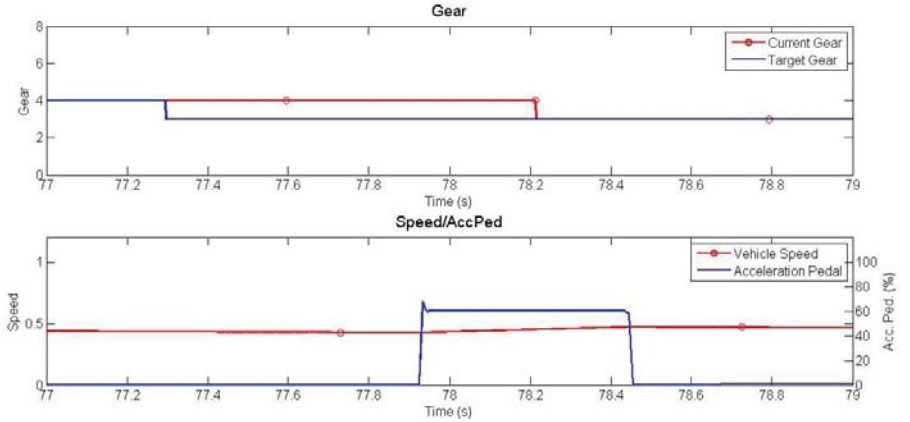

Figure 4-7 The acceleration in the shift in a coast downshift from 4th to 3rd gear in testbench.

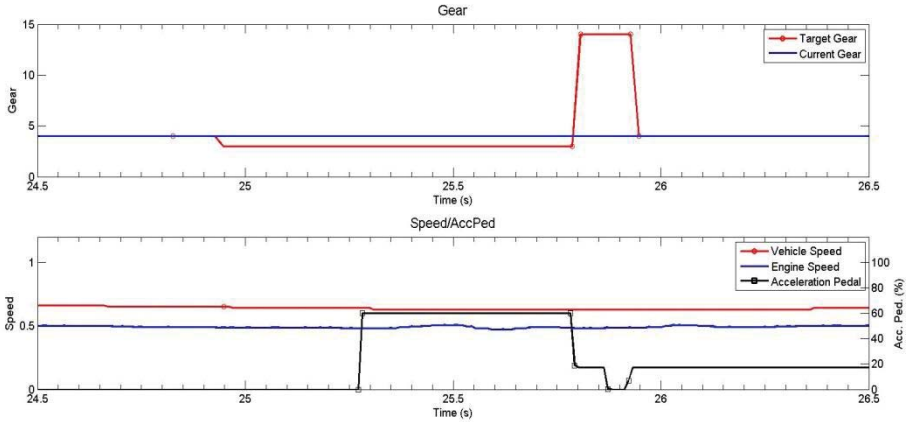

Figure 4-8 The acceleration in the shift in a coast downshift from 4th to 3rd gearshift in car.

5. Discussion

The results from the tests in Chapter 4 shows that the DB can perform the right sequences for all tests. This shows that an automation of the benchmark tests is possible.

5.1 Limitations in the Testing

In those tests where the PID controller is used initially, a significant amount of time might be needed on achieving the acceptable control parameters. If the constraints on the speed regulation are too narrow a considerable amount of time during the test can be spent on achieving the right speed before the shift event occurs. This can be seen in the down shift events where it takes more than 15 seconds before the shift event is triggered, see *Figure 4-3* and *Figure 4-4*.

The execution of the coast upshift event shows a satisfying result. It can, however, be rather hard to specify as a too slow removal of the AP causes a power upshift and a too fast removal makes the TCU want to maximize the motor braking and therefore not allow an upshift. In Figure 4-5 and Figure 4-6 the car is still accelerating and the AP is held at a rather high value when the shift occurs. It can therefore be hard to judge if it is a power upshift or a coast upshift event that has been performed.

The time critical regions of the tests can only guarantee a minimum wait time. For most tests this is not an issue but in the accelerate in the shift event this may cause the acceleration to occur in the wrong phase. The problem stems from how the messages on the car bus are read. The nodes on the car bus sends out their information with a set interval if no other node requests a message. When DB wants to read a message it polls the car bus until a message with the right ID is read. During this polling no task switches are allowed which can cause the following scenario. The main task is executing a time critical section. Since the output task has a higher priority than the main task, the output task will always be

able to interrupt the execution of the main task. This means that when the worst case execution time for the main task is the execution time for the section plus the cycle time for the message containing the brake signal. If a message is being checked within the time critical region this cycle time needs to be considered as well. If the cycle time for receiving a message is long or many messages are received this might cause the output task to check the status of the brake several times.

The best setup for reading the messages on the car bus would therefore be to use interrupts with a hardware filter. The filter let selected messages through to the CAN controller to cause an interrupt in the microcontroller. When a message is received it should update the status of the signals in that message. No time would then be wasted on polling messages until the desired message was read. The DB would just need to check a parameter instead of waiting for a new message to be sent from the car. Another solution to make the execution more accurate is by prohibiting any switching of tasks in time critical regions. This can be dangerous if the delay is long as there is no way of breaking the control of the AP with the brake pedal.

5.2 Improvements

In the DB the reading of the pedalmap in the car is the same one as in the previous implementation of the DB. The reading is quite naive and contains some assumptions about how the AP is constructed. The AP have two signals that it outputs to the ECU for fail safe reasons. These signals does not have to be linear or with a fix relation to each other. This is something that the DB assumes is true for every pedal. When the mapping is read it only reads the value from one signal and then doubles it to create the second signal. This works for most cars but can cause problems in the future if a pedal is mapped in a different way.

The model in the simulation does not behave like a car in all situations. The TCU in the model is not as smart as the one in the car. This creates a problem with the test cases as they do not always behave the same way in the car as they did in the simulation. The simulation strictly reads the pedal mapping and cannot recognize any intentions of the driver. The TCU in the car is adapted to and optimizes the driving in many ways, so it does not always shift gear at once when a shift line is crossed. One example of this is that it does not allow an upshift instantly if the AP is released. The TCU then realizes that the driver wants to

reduce the speed. So it maximizes the motor braking by keeping the lower gear engaged.

The biggest concern with the current implementation is that it requires more inputs in the specification of the shift events than the previously. This means that all tests that are to be run by the DB have to be updated so they are compatible with the DB. The next step in the development would be to make Caltet more compatible with the old specification. When the shift events are specified in a set way it should be possible to interpret the information from these events and extract information for the shift events for the DB.

At the moment only the PID parameters can be changed in the DB. In the future a setup menu where different delays and timeouts could be set would be a good feature. This would enable the test driver to adapt the behavior of the DB in a better way.

6. Conclusion

In this thesis a way of executing the benchmark tests with the DB has been presented. The DB can be controlled via an interface from Caltet and the solution has been verified with simulations and in a car. The communication between Caltet and DB runs in the background without any direct control of the DB from the user. The greatest problem lies in specifying the shift events. If the shift events are badly prepared a considerable time can be spent on adjusting the shift event so the right gear shift occurs. This is mostly a problem for the coast upshift, as the car needs to accelerate to the right speed and then release the AP slowly.

From the results in Chapter 4 it is clear that the DB can be controlled from Caltet and perform the benchmark test. It is hard to say to what extend it will be used but there are some test cases where the DB perform really well. The accelerate-in-the-shift event and the power upshift are good examples of this. The accelerate-in-the-shift is difficult for a human to perform repeatedly because of the high timing demands and the need of detecting when a shift starts. The test's repeatability can be highly improved by the DB, making it worth spending time on specifying the tests cases. The power upshift is the easiest test to specify, which makes it more likely to be used. For a fully automated test tool more different shift events need to be implemented but this thesis will hopefully form a good foundation for future work on this tool.

At the moment a new version of the DB is being developed at IAV. It aims at being smaller than the current version and operates without a screen. For it to work properly an interface between the computer and the DB is then needed. This thesis will hopefully help form the foundation for the interface of the new version of the DB.

Bibliography

Fijalkowski, B. (2011). *Automotive Mechatronics: Operational and Practical Issues.* Cracow: Springer Science+Business Media B.V.

Genta, G., & Morello, L. (2009). *The Automotive Chassis Volume 1: Components Design.* Springer Science+Business Media B.V.

Guo, W., Wang, S. H., Su, C. G., Li, W. Y., Xu, X. Y., & Cui, L. Y. (2014). Method for Precise Controlling of The AT Shift Control System. *International Journal of Automotive Technology, Vol. 15, No. 4* , pp. 683-698.

Gust, P. (2012). *Programmierung und Validierung eines elektronischen Fahrpedalvorgabemoduls zur Unterstützung des Applikationsprozesses.* Berlin.

IAV Gmbh. (2014). *Spezifikation: Fahrpedalvorgabe-System.* Berlin.

Matthies, F., von Rüden, K., Kahlbau, S., Wurm, A., & Petzke, W. (2014). *Transmission Calibration Efficiency increase through a Tool Based Process.* Berlin.

Naunheimer, H., Bertsche, B., Ryborz, J., & Novak, W. (2011). *Automotive Transmissions Fundamentals, Selection, Design and Application Second Edition.* Berlin: Springer-Verlag.

Pogede, N. (2014). *Implementierung weiterer CAN-Protokolle in ein Modul zur Fahrpedalwertvorgabe für Versuchsfahrzeuge.* Berlin.

Robert Bosch Gmbh. (1991). *CAN Specification.* Stuttgart.

Strandemar, K. (2005). On Objective Measures for Ride Comfort Evalution. Stockholm, Sweden.

Vector Infomatik. (2015). *Product Information CANape.*

Årzén, K.-E. (2012). *Real-Time Control Systems.* Lund: KFS AB.

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER´S THESIS

Date of issue
September 2015

Document Number
ISRN LUTFD2/TFRT--5989--SE

Author(s)
John Kroon

Supervisor
Felix Matthies, IAV
Karl-Erik Årzén, Dept. of Automatic Control, Lund University, Sweden
Anton Cervin, Dept. of Automatic Control, Lund University, Sweden (examiner)

Sponsoring organization

Title and subtitle

Automated Event Control of Vehicles for Drivability Testing

Abstract

Driving comfort is subjective; every driver has their set of preferences on how a car should behave. As the ability to control and suppress unwanted disturbances has increased with increased technological abilities, the demands on driving comfort in modern cars also rise. To account for these new demands, the testing and evaluation of the driving comfort becomes increasingly important for car manufacturers.

This thesis focuses on how to implement different benchmark tests of an automatic gearbox into the Drivebox, a tool developed at IAV that can act as the accelerator pedal in a car, and the creation of an interface between the Drivebox and the test database Caltet. The final goal with the new implementation of the Drivebox is an easier and more repeatable execution of benchmark tests. In order to verify that the interface and the implemented tests work, the test were first simulated with the help of a computer and then performed in a vehicle. The results from both the simulation and the tests in the vehicle show that it is possible to control and execute the benchmark tests using this new interface with the Drivebox.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
1-62

Recipient's notes

Security classification

http://www.control.lth.se/publications/