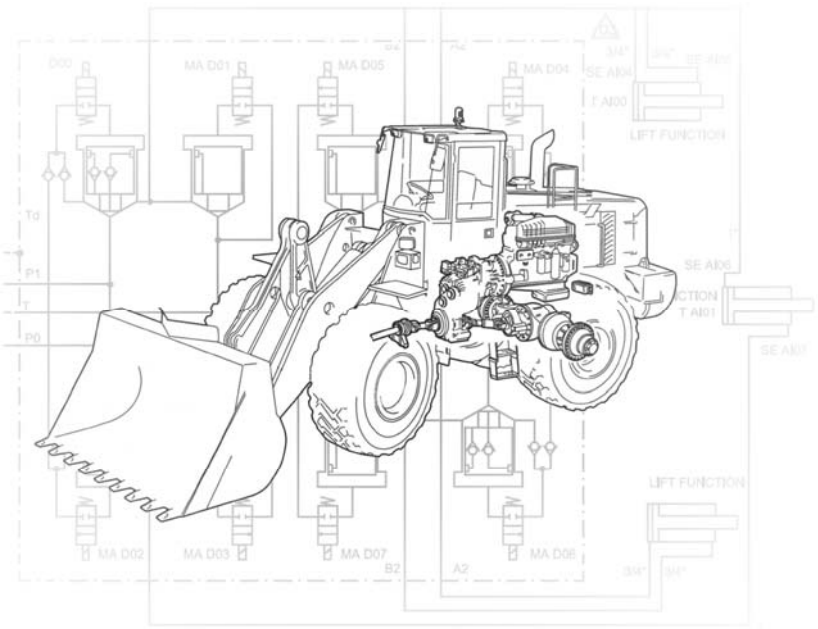
Master Thesis

Implementation of a Pump Control System for a Wheel Loader Application

Daniel Gunnarsson & Emanuel Strid

February 2007

LiU-IEI-TEK-A--07/0041--SE



Linköpings universitet
INSTITUTE OF TECHNOLOGY

Division of Fluid and Mechanical Engineering Systems
Department of Management and Engineering
Linköping University, SE-581 83 Linköping

Implementation Of A Pump Control System For A Wheel Loader Application

Daniel Gunnarsson & Emanuel Strid

February 2007

LiU-IEI-TEK-A--07/0041--SE

Division of Fluid and Mechanical Engineering Systems
Department of Management and Engineering
Linköping University, SE-581 83 Linköping

Avdelning, institution
Division, Department
Institutionen för ekonomisk och industriell utveckling
Fluid och mekanisk systemteknik
Department of Management and Engineering
Fluid and Mechanical Engineering Systems

Datum 2007-02-06
Date 02/06/2007

Språk
Language

☐ Svenska/Swedish
☒ Engelska/English

☐ _____

Rapporttyp
Report category

☐ Licentiatavhandling
☒ Examensarbete
☐ C-uppsats
☐ D-uppsats
☐ Övrig rapport

☐ _____

ISBN

ISRN
LIU-IEI-TEK-A--07/0041--SE
Serietitel och serienummer **ISSN**
Title of series, numbering _____

URL för elektronisk version
-

Titel Title	Implementation av styrsystem för pumpstyrning i en hjullastare Implementation of a pump control system for a wheel loader application
Författare Author	Daniel Gunnarsson & Emanuel Strid

Sammanfattning
Abstract

A lot of today's new developments strive for energy efficiency. This includes the hydraulic side of industry. The Division of Fluid and Mechanical Engineering Systems of Linköpings University in collaboration with Volvo Construction Equipment in Eskilstuna has developed a new hydraulic concept when it comes to the control of cylinder loads in a wheel loader. The concept differs from today's application, where the cylinder load is controlled via a valve, in the way that the load is solely controlled by a pump. To control this system, an electrical feed back of operators demanded signal is needed. These signals have to be correctly interpreted so that valve and the pumps perform the requested operation. The new system is going to need a unit that can perform these operations in a way that corresponds to the operating level of today's hydraulically controlled system.

The study aims to develop a software platform that solves this. This platform shall, besides performing the operators' demands, monitor the system. The monitoring of the system is a crucial part because of security issues, but also when analyzing the systems functionality. The implementation of this software will be done in a real-time computer with the ability to collect data, interpret it and then control the connected units of the system. Further work that is to be done is an energy consumption study of today's hydraulic system, and on the basis of this study, theoretically evaluate the new system.

The study has resulted in a great insight of an industrial mechanic machine, this in a level that includes an entire system. The wide range of this task has brought analysis and development of both hydraul mechanical-, electrical- and software related systems. With an understanding of these, both separate and in interaction with each other, a platform has been designed that shall facilitate the forthcoming development of energy efficient hydraulics, both at VCE and LiTH.

Nyckelord:

Keywords: Pump control, Wheel loader, differential mode, CompactRIO, energy recuperation, valvistor

Abstract

A lot of today's new developments strive for energy efficiency. This includes the hydraulic side of industry. The Division of Fluid and Mechanical Engineering Systems of Linköpings University in collaboration with Volvo Construction Equipment in Eskilstuna has developed a new hydraulic concept when it comes to the control of cylinder loads in a wheel loader. The concept differs from today's application, where the cylinder load is controlled via a valve, in the way that the load is solely controlled by a pump. To control this system, an electrical feed back of operators demanded signal is needed. These signals have to be correctly interpreted so that the valve and the pumps perform the requested operation. The new system is going to need a unit that can perform these operations in a way that corresponds to the operating level of today's hydraulically controlled system.

The study aims to develop a software platform that solves this. This platform shall, besides performing the operators' demands, monitor the system. The monitoring of the system is a crucial part because of security issues, but also when analyzing the systems functionality. The implementation of this software will be done in a real-time computer with the ability to collect data, interpret it and then control the connected units of the system. Further work that is to be done is an energy consumption study of today's hydraulic system, and on the basis of this study, theoretically evaluate the new system.

The study has resulted in a great insight of an industrial mechanic machine, this in a level that includes an entire system. The wide range of this task has brought analysis and development of both hydro mechanical-, electrical- and software related systems. With an understanding of these, both separate and in interaction with each other, a platform has been designed that shall facilitate the forthcoming development of energy efficient hydraulics, both at VCE and LiTH.

Sammanfattning

Mycket av dagens utveckling i världen strävar efter energieffektivisering. Detta gäller även inom hydraulikbranschen. I samarbete med Volvo Construction Equipment i Eskilstuna och Linköpings Tekniska Högskola har ett helt nytt hydrauliskt koncept för att styra cylinderlaster på en hjullastare tagits fram. Konceptet bygger på, till skillnad från i dagens system att styra cylinderlaster med hjälp av en ventil, istället styra dessa enbart med pump. Detta koncept medför att elektrisk återkoppling av operatörens styrsignaler måste ske och att dessa signaler således måste tolkas för att styra ut ventiler och pumpar så att önskad rörelse uppfylls. Det nya systemet kommer att behöva en styrenhet som beräknar och utför dessa operationer, på ett sätt som medför en körbarhet likt dagens hydrauliskt styrda system.

Detta examensarbete syftar till att mjukvarumässigt ta fram en utvecklingsplattform som kan lösa denna mät och styrbarhet. Plattformen skall, förutom att utföra operatörens önskningar, kunna övervaka hydraulsystemet. Detta dels för att ett säkerhetssystem skall kunna aktiveras om något fel inträffar, men också för att kunna analysera systemets funktionalitet genom att aktivt spara informationsdata till fil. Implementation av denna mjukvara kommer att ske till en realtidsdator som har kapacitet att både läsa av information, tolka, reglera och styra ut funktioner i systemet. Vidare skall en energianalys av dagens arbetshydraulik utföras, denna skall sedan teoretiskt jämföras med det nya konceptet.

Arbetet har resulterat i en god inblick på helsystemsnivå av en industriell mekanisk maskin. Uppgiftens bredd har medfört analys och utveckling av både hydraulmekaniska-, elektriska- och mjukvarutekniska system. Med förståelse i hur dessa fungerar, både enskilt och i interaktion med varandra, har en plattform tagits fram som ska underlätta den fortsatta utvecklingen av energieffektiv hydraulik både på VCE och LiTH.

Acknowledgements

The work done in this thesis has been carried out at the Division of Fluid and Mechanical Engineering Systems at Linköpings University in collaboration with Volvo Construction Equipment in Eskilstuna.

We would like to thank…

Project supervisor Kim Heybroek and examiner Jonas Larsson for your participation, all the useful help, interesting discussions, valuable inputs and the great feedback you have given us during this time.

Supervisor Johan Lillemets, at Volvo Construction Equipment in Eskilstuna, for the great assistance with tests and practical arrangements.

Opponent Johan Larsson for a well performed review with interesting questions at issue.

Professor Karl Erik Rydberg for all your shared knowledge regarding hydraulics and fluid power systems during our time at LiTH.

Sören, Tosse and Mankan in the engineering workshop at LiTH, great thanks for your practical help surrounding the study.

Mulle Meck in the engineering workshop at Volco CE for providing tools and great assistance

Pär Degerman for all the help during the development of the electrical components. Thanks to your great knowledge the measure- and control unit ended up both safe and with great precision.

Finally, great thanks go out to Björn, Z, Rita, Rösth and Lasse at the division for their rewarding help and discussions.

Nomenclature

A_A	effective area, cylinder chamber - A	m^2
A_B	effective area, cylinder chamber - B	m^2
b_{0P} ... b_{6P}	design parameters	-
C_q	flow coefficient	-
D_p	displacement	$cc = cm^3$
F_f	friction force	N
F_{f_cyl}	friction force - cylinder	N
F_{f_joints}	friction force - joints	N
F_{fk}	friction force, piston	N
F_{fs}	friction force, piston shaft	N
F_{load}	cylinder acting force due to the load	N
n_p	pump revolutions per second	rev/s
p	pressure	Pa
p_{cylt}	cylinder pressure	Pa
p_L	load pressure	Pa
p_L	external feed pressure	Pa
p_p	pump pressure	Pa
P	power	W
P_{ref}	current power take out	W
P_{loss}	power, loss	W
$P_{\max}$	power, maximum	W
$P_{\min}$	power, minimum	W
T	torque	Nm
T_r	requested torque	Nm
U_{ref}	reference voltage	V
U_{joy_lift}	joystick signal – lift	V
U_{joy_tilt}	joystick signal – tilt	V
q	Volume flow	m^3/s
q_{cylt}	Volume flow to cylinder	m^3/s
Q	energy	J
x	cylinder position	m
$\dot{x}$	cylinder velocity	m/s
v_{cyl}	piston speed	m/s
δ_p	pump damping factor	-
ε	relative displacement	-
Δp	pressure difference	Pa
η_{hm}	hydraul mechanical efficiency	-
η_{vol}	volumetric efficiency	-
μ	dynamic viscosity	kg/ms
ν	kinematic viscosity	m^2/s
ρ	density	kg/m^3

Shortenings

SPID	*Servo Pressure Induced Differential lowering*
FPGA	*Field Programmable Gate Array*
GUI	*Graphical User Interface*
LS	*Load Sensing*
CAN	*Controller Area Network*
PWM	*Pulse Modulated Width*
ECU	*Electrical Control Unit*
MAC	*Measure, Analyze and Control*
EPR	*Electrical Pressure Reducer*
HPR	*Hydraulic Pressure Reducer*
DLL	*Dynamic Link Library*
RPS	*Revolutions Per Second*
RPM	*Revolutions Per Minute*

Contents

CHAPTER 1 INTRODUCTION **3**

1.1 BACKGROUND 3
1.2 PROBLEM DESCRIPTION 3
1.3 AIMS AND LIMITATIONS 3

CHAPTER 2 CONCEPTUAL STUDY **5**

2.1 OVERVIEW L60E 5
2.2 TODAY'S HYDRAULIC SYSTEM 6
2.3 THE NEW HYDRAULIC SYSTEM 7
2.3.1 Semi-differential lowering mode *9*
2.3.2 Valve Package *10*
2.3.3 Open circuit system velocity control *12*
2.3.4 Pumps - Parker P1/PD IDEC Pump *13*
2.4 MEASURE, ANALYZE AND CONTROL UNIT 19

CHAPTER 3 HARDWARE AND SIGNALS **21**

3.1 ELECTRICAL OPERATOR SIGNALS 21
3.2 INSTALLATION OF THE TRANSDUCERS 21
3.2.1 Pressure transducers *21*
3.2.2 Position transducers *22*
3.3 CAN 23
3.3.1 Overview of CAN *23*
3.3.2 Connection to the L60's CAN *24*
3.4 VALVE PACKAGE CONTROL 24
3.5 MAC 26
3.5.1 Connections and connectors *26*
3.5.2 Voltage Regulators *27*
3.5.3 Router *28*
3.5.4 CompactRIO *28*

CHAPTER 4 SOFTWARE SYSTEM **31**

4.1 LABVIEW 31
4.1.1 FPGA *31*
4.1.2 Target *32*
4.1.3 Host *34*
4.1.4 System Communication *34*
4.1.5 Measurement and Automation Explorer *34*
4.2 IMPLEMENTED CONTROLLERS 35
4.2.1 Valve Controller *35*
4.2.2 Pump Controllers *37*

CHAPTER 5 EXPERIMENTS **43**

5.1 CYLINDER AND FRAMEWORK FRICTION 43
5.1.1 Objectives *44*
5.1.2 Method *44*
5.1.3 Determining Ff *45*
5.1.4 Test performance *46*
5.1.5 Analysis and results *46*
5.2 LOSSES DUE TO CONNECTORS 47
5.3 FUNCTIONALITY TEST - LOWERING OF LOAD 47
5.4 ENERGY ANALYZE OF TODAY'S WORKING HYDRAULICS 49
5.4.1 Objectives *49*
5.4.2 The short duty cycle *49*
5.4.3 Method *50*
5.4.4 Test performance *51*
5.4.5 Preparation of saved data - datagen *51*
5.4.6 Power calculations – calc_power *52*

5.4.7 Total energy consumption – calc_energy ... *53*
5.4.8 Analysis and results ... *54*
5.5 VALVE TESTS ... 57
5.5.1 Objectives ... *57*
5.5.2 Method ... *57*
5.5.3 Test performance ... *57*
5.5.4 Analysis and results - Simulation ... *57*
5.5.5 Analysis and results – EPR Test stand ... *58*
5.5.6 Analysis and results – HPR Test stand ... *59*
5.6 P1 PUMP TEST ... 60
5.6.1 Objective ... *60*
5.6.2 Method ... *60*
5.6.3 Test performance ... *60*
5.6.4 Analysis and results ... *60*

CHAPTER 6 RESULTS AND DISCUSSION **63**

6.1 THEORETICAL ENERGY CONSUMPTION REFERENCE ... 63
6.2 LABVIEW ... 65
6.3 PERFORMANCE OF MAC ... 65
6.4 PERFORMANCE OF COMPACTRIO ... 65
6.5 FUTURE ... 67
6.5.1 Verifying tests/experiments ... *67*
6.5.2 New developments & improvements ... *67*

CHAPTER 7 BIBLIOGRAPHY **71**

A – CONNECTION DIAGRAM **73**

B – CODE CONVERSION MANUAL **74**

C – IMPLEMENTED CONTROLLERS **75**

C.1– VALVE CONTROLLER ... 75
C.2 – PUMP LOGICS ... 79
C.3 – MINIMUM PRESSURE ... 81
C.4 – MAXIMUM POWER TAKE OUT ... 82

D – MATLAB M-SCRIPT **86**

D.1 – NONLINEAR LEAST MEAN SQUARE ... 86
D.2 – RESULTING EXCEL DOCUMENT (LIFT) ... 87
D.3 – DATA GENERATION ... 88
D.4 – POWER CALCULATION ... 89
D.5 – CLEAN UP FUNCTION ... 92
D.6 – ENERGY CALCULATIONS ... 93

E – TEST RESULTS, LOWERING FUNCTIONALITY **95**

F – PUMP TESTS **97**

F.1 – DEFINITIONS ... 97
F.2 – EXTERNAL PRESSURE FEED TESTS ... 98

G – ELECTRICAL POWER CONSUMPTION **100**

H – ORDERED COMPONENTS **101**

I – FILE DESCRIPTIONS **102**

J – LABVIEW FILES **104**

K – HYDRAULIC SCHEME VOLVO L60:E **105**

Chapter 1 Introduction

1.1 Background

Stricter regulations when it comes to emissions from vehicles that use a combustion engine are becoming a common notion. Previous study's has shown that the efficiency in a wheel loader can be increased by introducing a new solution for the working hydraulics. One solution that looks promising is the concept of a pump controlled application. The purpose of this study is to develop software, implement and evaluate this in such a solution.

1.2 Problem description

The hydraulic system that has been developed in a previous study has some major changes to it and brings a new set of problems to the table. The actuators in the new system are controlled via two new pumps. These pumps have the ability to work as motors when lowering a load and can therefore generate a torque. An entirely new valve package has been developed to direct the flow and to handle functions not available in today's system. Both the pumps and the valve have to be controlled to give the wheel loader the same functionalities as it has in today's system. To meet this standard, the system has to be electrically controlled, in comparison to the hydraul mechanical solution of today. To be able to utilize electrical control, active controllers have to be implemented.

1.3 Aims and limitations

The study starts of with a general overview of the major changes in today's hydraulic system in a wheel loader vs. the new. It is followed by a review of the new hydraulic components and thereafter how these are controlled to accomplish an acceptable level of operating the entire wheel loader. A study is done on the energy consumption of today's hydraulic system. This will be compared to a theoretically calculated consumption of the new system, and in a final solution also to the real system.

The aim of this study is to design an advanced development platform for an energy efficient hydraulic system in a wheel loader. This includes building a control unit with belonging software; perform tests on today's hydraulic system and to write a manual to render possible further development.

Chapter 2 Conceptual Study

2.1 Overview L60E

A wheel loader is a machine mainly used in the mining and construction industry, where it uses a loading unit on the end of framework to shift material. The loading unit is often in the form of a bucket which is used to freight dirt and gravel. This unit can easily be replaced with other tools, such as a fork lift.

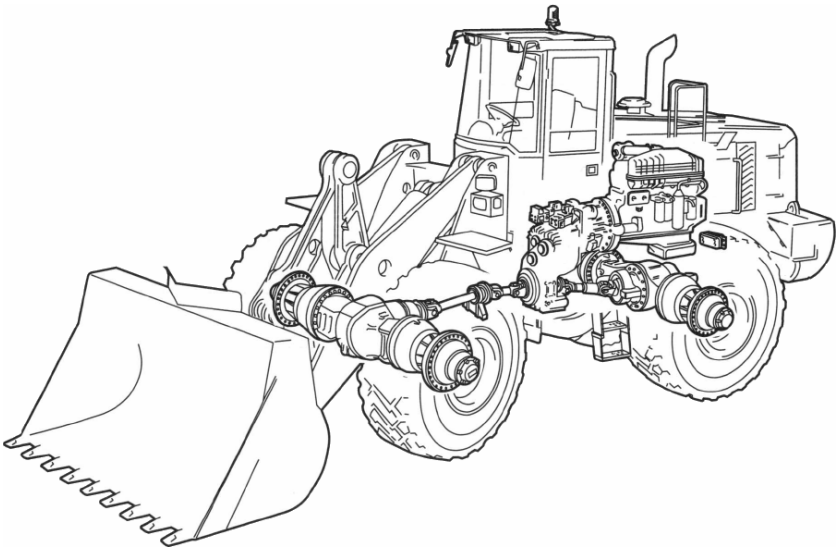


Figure 1: A Volvo wheel loader

Both the loading, steering and propulsion system of the wheel loader consumes its power from the same source in the system. This power source is in the form a diesel combustion engine that is located at the back end of the wheel loader. This means that the power has to be shared when simultaneously using the different systems. The power to the propulsion is delivered via a transmission and a flywheel. The power to the loading unit is delivered via the so called working hydraulics.

2.2 Today's hydraulic system

The hydraulics in the L60E can be divided into three parts; auxiliary-, steering- and working hydraulics. The auxiliary hydraulics is powered with the P3 pump and its job is to make sure that brakes and the cooling system maintain their working pressure. The working hydraulics is powered from P1. Strokes of the working cylinders are made by the operator using directly operated LS sliding spool valves.

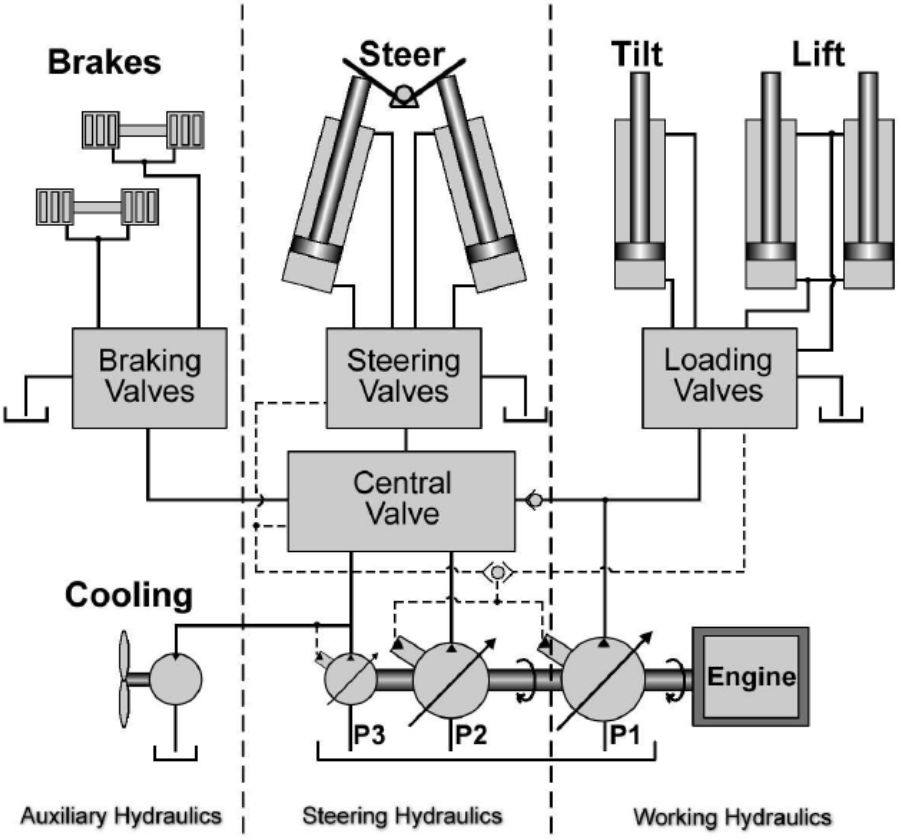


Figure 2: Simplified hydraulic configuration of a wheel loader

The P1 pump is an 85cc LS pump that makes sure that the system keeps a pressure level of 20 bar over the highest load pressure. When a stroke of a working cylinder is requested, some of the built up P1 pressure is lost over valves and hydraulic connectors, hence a 20 bar overpressure is needed.

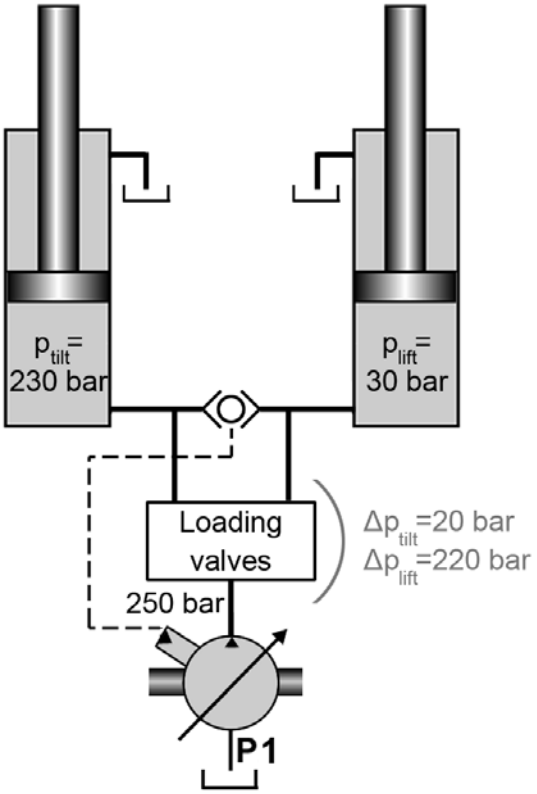


Figure 3: Lifting scenario in today's hydraulic system

$$p_p = p_{A\_tilt} \quad or \quad p_{A\_lift} + p_{diff} \tag{2.3.1}$$

$$\Delta p_{tilt} = p_p - p_{A\_tilt} \tag{2.3.2}$$

$$\Delta p_{lift} = p_p - p_{A\_lift} \tag{2.3.3}$$

Figure 3 illustrates a lifting scenario where the pressure in the A-chambers of the tilt- respectively lift differs when lowering. The pump will set the system pressure to 20 bar over the highest load pressure, i.e. in this case 250 bar. The pressure losses between the pump and respectively working cylinder will therefore be 20 bar and 220 bar. Thus the lift operation will be performed with a very low grade of efficiency.

2.3 The new hydraulic system

With focus on more energy efficient hydraulic control of cylinder loads, a new concept has been developed. The new system is based on controlling the load directly with the pump and not with a valve. To do this, two electrical joysticks will be installed so that an open circuit velocity control can be achieved. The controlling of the cylinder directions will be realized with a valve that to a great extent is on/off controlled. This is put to practise in order to minimize pressure drops between the pump and the loads, and hence minimizing the energy consumption. The concept also includes a semi differential operating mode, that when lowering transforms the potential energy to a torque on the hydraulic pump shaft. This torque can be used to power other hydraulic components, but also by the propulsion unit. To render this possible, the 85cc pump will be replaced with two 75 cc Parker P1's that have the

ability to operate with negative displacements, i.e. work as a motor. The pumps will be working separately, one controlling the tilt- and one the lift cylinders. The loading valve package will be replaced with two united valvistor controlled packages, also here one for each working function. For complete hydraulic scheme, see Appendix K.

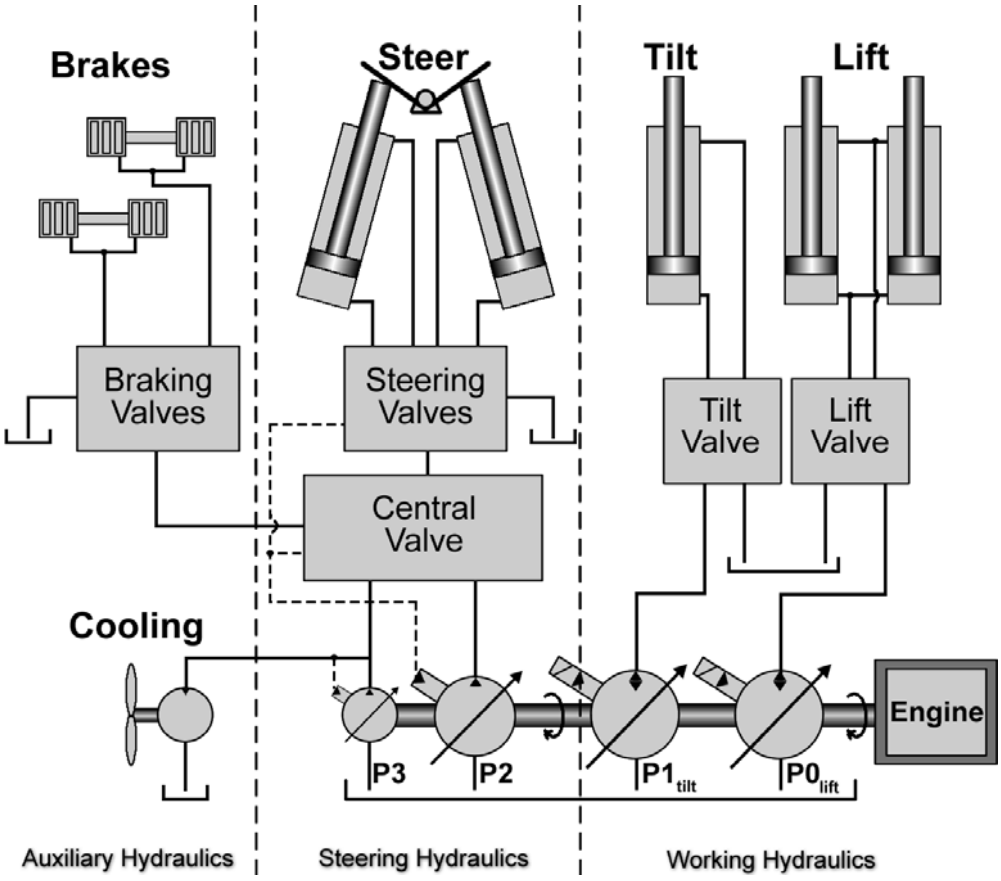


Figure 4: Simplified new hydraulic configuration

In order to make use of this system, a measure, analyze and control (MAC) unit will be installed. The MAC unit will need access to information from different parts in the wheel loader. This information will partly be picked up from external and internal transducers, but also from the built in CAN-bus system. In the phase of development and in order to enable effective tests, a number of extra transducers will be installed in the wheel loader. In order to minimize costs, but most important, to make the new system as robust as possible, the number of transducers used by the control unit will be minimized. After the development phase, most of these extra installed transducers are no longer necessary.

Concisely, the changed components and added functions can be divided in

- Semi-differential lowering mode
- Valve package
- Pumps

2.3.1 Semi-differential lowering mode

To understand the semi-differential mode, an understanding of the differential mode is needed.

A differential mode means that the cylinder chambers and the pump are connected with each other. When lowering the load in this mode, flow will pass from A chamber over to the B chamber, and the excess flow will pass through the pump. This gives two advantages in comparison to normal lowering. The first is that the lowering speed is increased threefold. The other one is that the flow that passes through the pump, which now acts like a motor, can be regenerated. This means that the energy that is generated can be used by other functions in the system. In 2006 Heybroek, K. [1] a downside of the differential mode is reviewed, and that is a flow-pressure transformation. This means that the decrease in flow out from a volume for a certain piston speed corresponds to an aggregation in pressure with the same ratio.

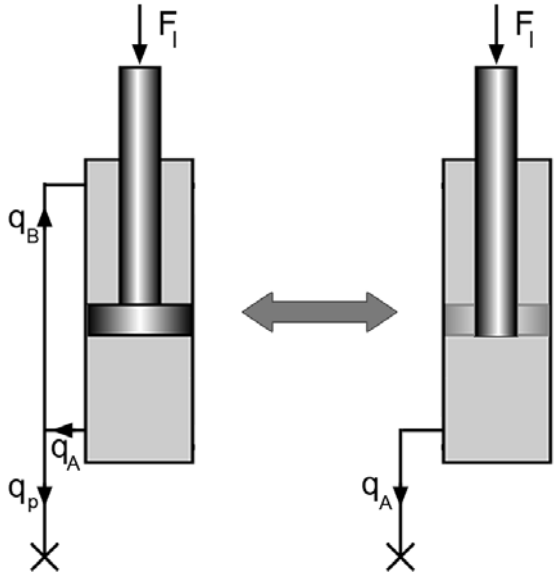


Figure 5: Illustration of the change in actuator properties for differential mode

When the two chambers are connected, the pressure in both chambers is the same. The only thing that differs between these two chambers now is the area. The actuator properties can be simplified as seen in *Figure (5).* What this shows is that the load is held up by the piston shaft area alone.

To come to terms with the pressure increase, the semi-differential mode is presented. This mode means that the valvistor that directs flow in to the piston chamber is proportionally controlled and therefore works like a variable orifice. By using this variable orifice, the pressure that is led in to the B chamber can be controlled. Reducing the pressure that is led in to the B chamber leads to that the pressure in the entire system can be reduced.

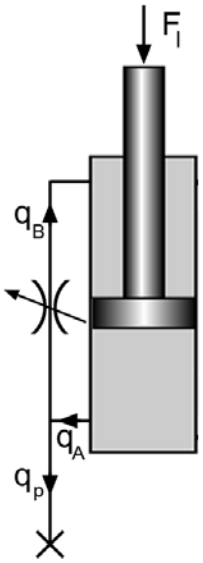


Figure 6: Illustration of semi-differential mode

2.3.2 Valve Package

The valve package consists of two parts with four valvistors in each section, and all the valvistor elements can be individually controlled. Its main task is to direct the flow that is given by the pump.

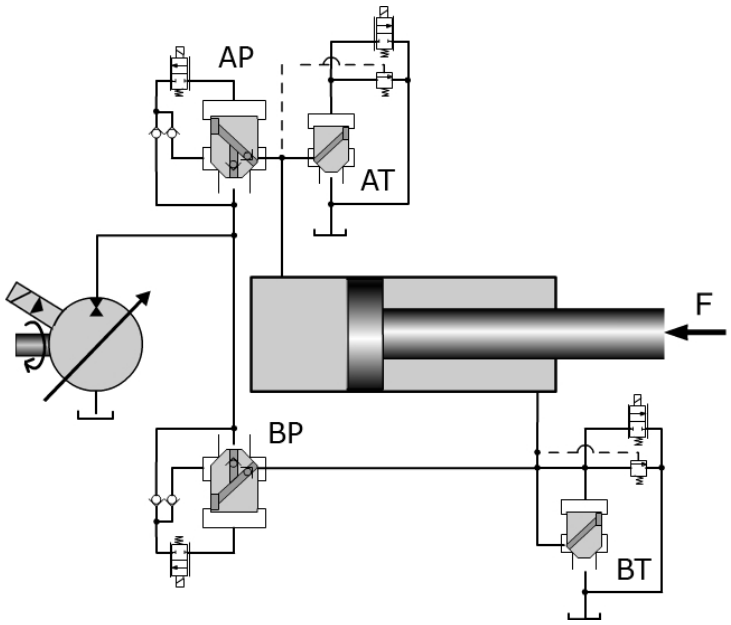


Figure 7: Schematic connection diagram of one section of the valve package

Valvistor

A valvistor is a seat valve that is originally proportionally controlled, but will in this application to a great extent be used as an on/off valve. In this concept however, two of the valvistors will be proportionally controlled (BP in *figure 7*).

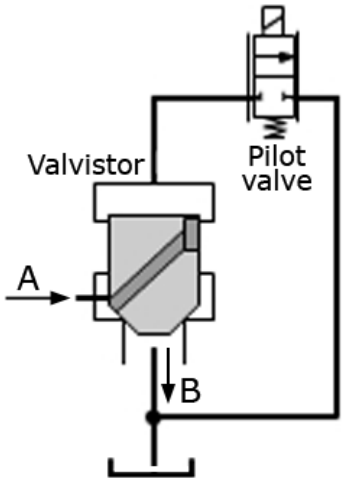


Figure 8: Schematic of the connections between valvistor and pilot valve

When the pilot valve is closed the pressure in A is transferred, via a channel in the valvistor, to the space above it, giving the same pressure as in port A. Due to a difference in the area ratio of the top and bottom of the valvistor, the resulting force pushes down the valvistor in its seat. By opening the pilot valve, the pressure that resides in space above the valvistor is reduced, therefore reducing the downward force. The resulting force that is due to the pressure in A, pushes the valvistor upwards and opens a flow path from A to B until the slot orifice area equals the pilot orifice area.

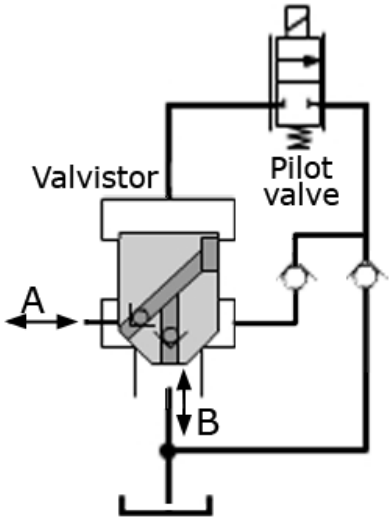


Figure 9: Schematic of a valvistor that can direct flow in both directions

To handle a semi-differential lowering, main flow has to be controllable from A to B and vice versa. The valvistor seen in *figure 9* works in the same manner as described in section above. The difference however, is that check valves are installed inside the valvistor. This enables the valvistor to direct flow in both directions.

Pilot Valve

The pilot valve is used to control the position of the valvistor. By applying a PWM signal, an electromagnet forces the spool to move and therefore opening the valve. The valve that is used in this application is called Vickers EPV10 and directs flow

from port two to port one. The valve has to be fed with a PWM signal with the frequency 150Hz and amplitude 24V. This is done to minimize the friction in the pilot valve. If the amplitude of the signal is 24V the valve uses about 0.7A.

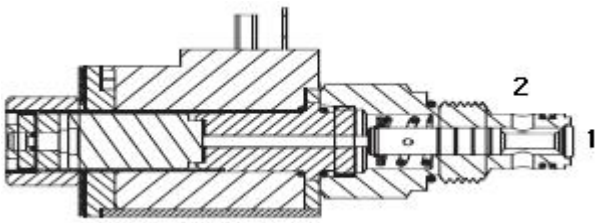


Figure 10: A cross-section of the pilot valve

The pilot valves are proportionally controlled by PWM signals generated by the MAC-unit.

2.3.3 Open circuit system velocity control

The two levers that the operator affects to initiate a motion of the lift- and tilt cylinders are open circuit system velocity controlled. A certain stroke is equivalent to a certain requested cylinder velocity for each respectively functions. If the operator pushes the lever to its max he requests a predefined maximum velocity. The maximum velocity of the cylinders is in this system is set to 0.1 m/s.

The advantage of velocity steering is that a certain stroke always corresponds to a certain velocity irrespectively of the rpm in the engine, presupposed that $n_{\max}$ is not reached.

Today's system is controlled by the operator with directly operated sliding spool valves that adjusts the flow to the cylinder chambers. The new system will have an electric feed back steering and to fulfil this two electrical joysticks are installed.

In order to achieve a velocity steering of the cylinders it is the flow that has to be controlled. To reach the same piston velocity of a cylinder when rising and lowering, different flows into the A- respectively B-chamber of the cylinders is needed.

The flow that is needed to achieve a certain piston velocity in a cylinder is defined as

$$q_{cyl} = A_{cyl} \cdot v_{cyl} \tag{2.3.1}$$

$$q_p = \varepsilon_p \cdot D_p \cdot n_p \cdot \eta_{volp} \tag{2.3.2}$$

To realize this flow, the relative displacement of the pump has to be calculated, equations *(2.3.1)* and *(2.3.2)* where $q_{cyl} = q_p$ gives

$$\varepsilon_p = \frac{A_{cyl} \cdot v_{cyl}}{Dp \cdot n_p \cdot \eta_{volp}} \tag{2.3.3}$$

For negative strokes, a differential drive mode is used. This means that the flow that leaves the A-chamber both stream into the B-chamber and to the P1 pump that for negative flow will work as a motor. The quotient of this dividing of the flow can be calculated by studying the areas of the A- and B-chamber of the actual cylinder.

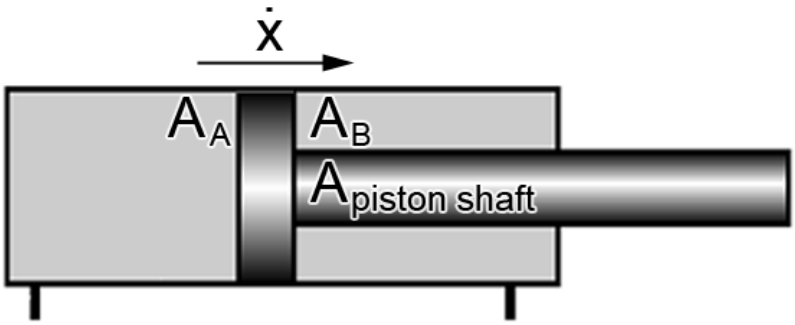


Figure 11: Area relations in a cylinder

$$A_A = A_B + A_{piston\_shaft} \tag{2.3.4}$$

The area in the A-chamber is equal to the sum of the area of the B-chamber and piston shaft. Thus the flow delivered to the P1 pump when performing negative strokes will be proportional to the area of the piston shaft.

$$q_{A\_out} = q_{B\_in} + q_{motor\_in} \tag{2.3.5}$$

(2.3.1), (2.3.4) and (2.3.5) gives

$$q_{motor\_in} = A_{piston\_shaft} \cdot v_{cyl} \tag{2.3.6}$$

Therefore the following will be valid for positive strokes

$$\varepsilon_p = \frac{A_A \cdot v_{cyl}}{Dp \cdot n_p \cdot \eta_{volp}} \tag{2.3.7}$$

And for negative strokes

$$\varepsilon_m = \frac{A_{piston\_shaft} \cdot v_{cyl}}{Dp \cdot n_p} \cdot \eta_{volm} \tag{2.3.8}$$

A negative stroke means that $v_{cyl} < 0$, which will give rise to a negative relative displacement to the P1 pump.

2.3.4 Pumps - Parker P1/PD IDEC Pump

The pumps used in the new hydraulic system are named P1/PD IDEC and are manufactured by Parker. The pump is an axial piston pump with a maximum displacement of 75 cc. What's special about it is that the relative displacement is

electrically controlled from -1 to 1. This means that it can also operate with negative displacement, and thereby work as a motor. Instead of consuming energy, it produces it to the hydraulic axle. This enables other functions further on to use this energy i.e. the propulsion system.

The P1 has got a number of interesting and useful qualities.

- A built in pressure transducer that measures the pump pressure at the feed port.
- A transducer that measures the true/real relative displacement
- A transducer that measures the rpm of the axel connected to the pump
- A maximum pressure controller that supervise the relative displacement and regulates it if the pressure exceeds a defined maximum pressure.
- An internal software program with a graphical user interface called GUI. From this program it is possible to control the pump, supervise the signals from the transducers, change the settings of the built in controller, log to file etc.

Hydraulic connectors

The P1 has got a number of connectable ports. The port named A in *figure 12* is the main inlet port, and B is the outlet. Further, the S port in *figure 13* is the port for connecting an external feed pressure, and X on the housing is ported to an extra pressure transducer.

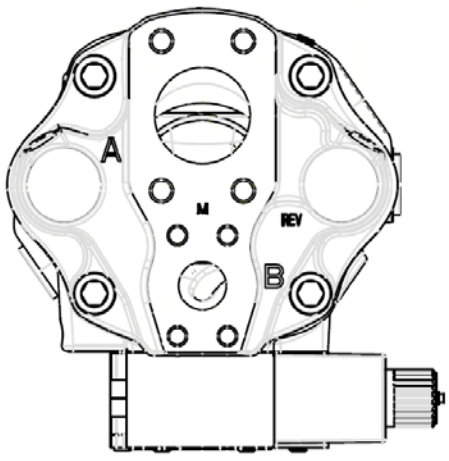


Figure 12: Front drawing of P1/PD [2]

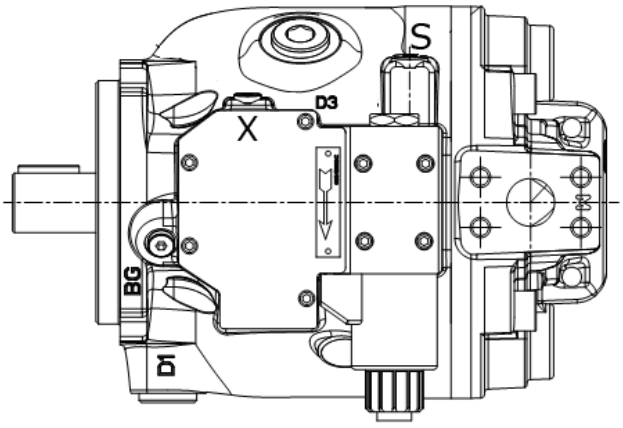


Figure 13: Drawing of P1/PD from below [2]

Pump setup and powering

The communication and power supply take place through CON1 that consist of a 30 pin connector. CON 2 is only connected if steering and setups is wanted to be controlled using the pumps internal software system, P1/PD GUI.

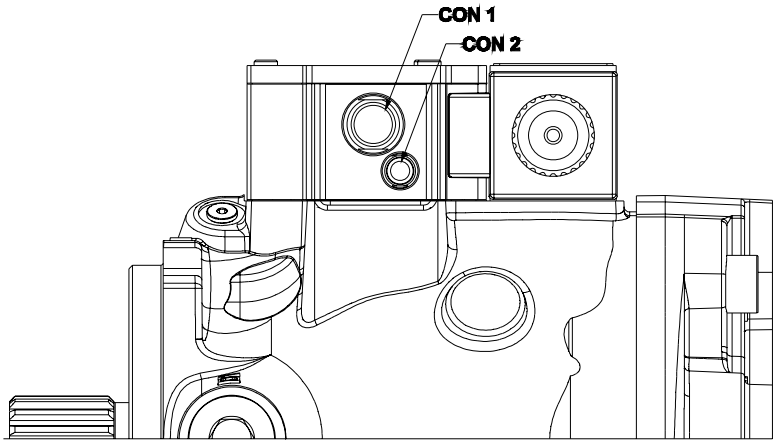


Figure 14: A View of the pump showing the connection locations of the 30 pin connector, and 3 pin RS-232 connector [2]

Pin	Designation	Signal	Connected to
1	EXT PWR (+24V)	(+12 ~ 36 V)	External power supply
2	EXT PWR (+24V)	(+12 ~ 36 V)	External power supply
3	EXT PWR GND	(0 V)	External power supply
4	EXT PWR GND	(0 V)	External power supply
8	ANALOG OUT	(0 ~ ± 5 V)	CB68-LP Data Acquisition Device
10	PUMP ENABLE INPUT	(+12 ~ 36 V)	External power supply
18	PRESSURE COMMAND VOLT	(0 to +5 V)	CB68-LP Data Acquisition Device
20	DISPLACEMENT/LOAD SENSE CMD VOLT	(0 ~ ± 5 V)	CB68-LP Data Acquisition Device
24	CAM POSITION	(0 ~ 3.3 V)	CB68-LP Data Acquisition Device
25	PUMP OUTLET PRESSURE	(0 ~ 3.3 V)	CB68-LP Data Acquisition Device

Table 1: Connector pins of interest in CON 1 [2]

Power usage

The hydraulics used in today's system is operated by using an 85 cc load sensing (LS) pump. The maximum power take out from a hydraulic pump can be approximated using the following rule of thumb [10]

$$P_{LS\_\max} = 1.5 \cdot D_p \cdot 10^6 \qquad [\text{kW}] \tag{2.3.9}$$

Which for an 85 cc pump results in

$$P_{\max} = 128 \text{ kW} \tag{2.3.10}$$

The new hydraulic system will accordingly to the intro of *chapter 2.3* be designed so that two Parker P1's will be controlling the lift respectively tilt cylinders individually. These two pumps can together request a much larger power take out than the previous solution, and must thereby be limited in the maximum power take out. In order to get a view of how much power that is needed at each point, we first must consider the hydraul-mechanical efficiency of the pump.

Internal maximum load pressure controller
In order to protect the hydraulic components and maintain a high security when operating the wheel loader, a pressure controller has to be used. The internal software system of the P1 pump includes a maximum load pressure controller. This controller activates if the pre set maximum load pressure is exceeded. This means that it takes control, and decreases the relative displacement so that the pressure reduces. This value of the maximum pressure is set either through the GUI or by pin 18 in CON 1. The default value in the GUI is set to 250 bar. Tests have been made to analyze the performance of the controller and they have shown that it works as wanted, and shows no strange characteristics what so ever.

Hydraul mechanical efficiency - η_{hmp}
An axial piston pump has got a hydraul mechanical efficiency that depends on a number of factors. It among other things includes the following relations.

- System pressure - p_H An increasing system pressure give raise to a decrease in efficiency. This is due to the friction in the pump between bearings etc.
- External feed - p_L Despite that an external feed pressure results in an improved response of the relative displacement, it leads according to the discussion above to a decreasing hydraul-mechanical efficiency.
- Viscosity - ν The viscosity strongly affects the hydraul-mechanical efficiency. A high value of the viscosity strongly increases the bearing forces and thereby also the losses.
- Rel. displacement - ε In most cases a higher relative displacement results in a better efficiency. If a small relative displacement is requested, the total losses become a greater part of the total supplied power.
- Rps – n A very low rps increases the losses since thereby a greater torque is needed to deliver the requested flow. Likewise the efficiency of the pump decreases for extremely high values of the rps. Somewhere in between this, there is an optimal rps for a given pump model.

In 1983 Rydberg, K E. [3] introduced the following model of the hydraulic-mechanical efficiency.

For a pump

$$\eta_{hmp} = \frac{1}{1 + \frac{b_{0P}}{\varepsilon} + b_{1P} + (\frac{b_{2P}}{\varepsilon} + b_{3P})\frac{p_L}{\Delta p} + b_{4P}\frac{\left|p_H + \delta_p \cdot p_L\right|}{\varepsilon \cdot \Delta p} + b_{5P}\frac{2\pi \cdot \mu \cdot n}{\varepsilon \cdot \Delta p} + b_{6P}\frac{(\varepsilon \cdot n)^2}{\Delta p}} \quad (2.3.11)$$

and for a motor

$$\eta_{hmm} = 1 - \frac{b_{0P}}{\varepsilon} - b_{1P} - (\frac{b_{2P}}{\varepsilon} + b_{3P})\frac{p_L}{\Delta p} - b_{4P}\frac{\left|p_H + \delta_p \cdot p_L\right|}{\varepsilon \cdot \Delta p} - b_{5P}\frac{2\pi \cdot \mu \cdot n}{\varepsilon \cdot \Delta p} - b_{6P}\frac{(\varepsilon \cdot n)^2}{\Delta p} \quad (2.3.12)$$

When the thesis does not include any test results from a motor of a Parker P1 type, the b_0 to b_6 can't be set for the motor. For that reason, η_{hmm} is set equal to η_{hmp}. This is because the hydraul-mechanical efficiency is not, in a great extent, dependent on whether a pump or motor is considered. A reason for this is that the same friction forces acts at the bearings independently of the flow direction.

[1] presents the following values

$$b_{0P} = 8.24 \cdot 10^{-3}$$
$$b_{1P} = 8.67 \cdot 10^{-3}$$
$$b_{2P} = 1.61 \cdot 10^{-2}$$
$$b_{3P} = 1.25 \cdot 10^{-1}$$
$$b_{4P} = 5.59 \cdot 10^{-2}$$
$$b_{5P} = 6.09 \cdot 10^{-2}$$
$$b_{6P} = 5.67 \cdot 10^{-8}$$

These values has in [1] been calculated through tests on a Sauer SPV-22 pump with a displacement of 70 cc. This pump is both an axial pump and has a displacement similar to the Parker P1. The values can therefore with adequate precision be used as the model for the Parker pumps, this since similar test results for the P1's don't exist and is complex and time demanding to generate.

When the operator don't request any flow (relative displacement), the pumps still consumes power for rotation of the pump shaft. This phenomenon is called idling consumption. The model presented above is not functional for relative displacements at zero and must therefore be further analyzed.

Idling power consumption

To get an insight in the size of the idling consumption for different pressure and rps, *equation 2.3.11* is inserted in a torque model and plotted for epsilon close to zero. In this model a viscosity of 10 cSt is assumed. This is a value that often is aimed at for in similar out door machineries because it results in an optimal performance of the pump. With varying pressure and rps, an insight of the idling consumption can be calculated.

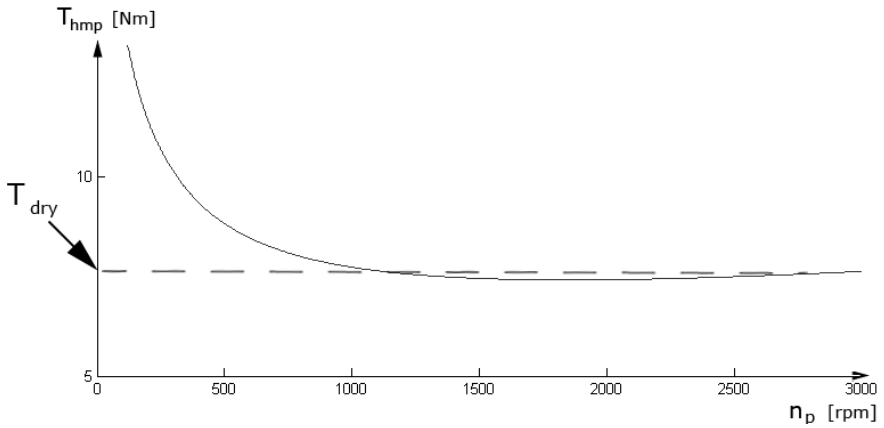


Figure 15: Idling torque

By watching a tangent at the end values that corresponds to a L60's rpm interval (780 – 2800 rpm) in the *figure 15* above, a torque independent of the static friction can be calculated.

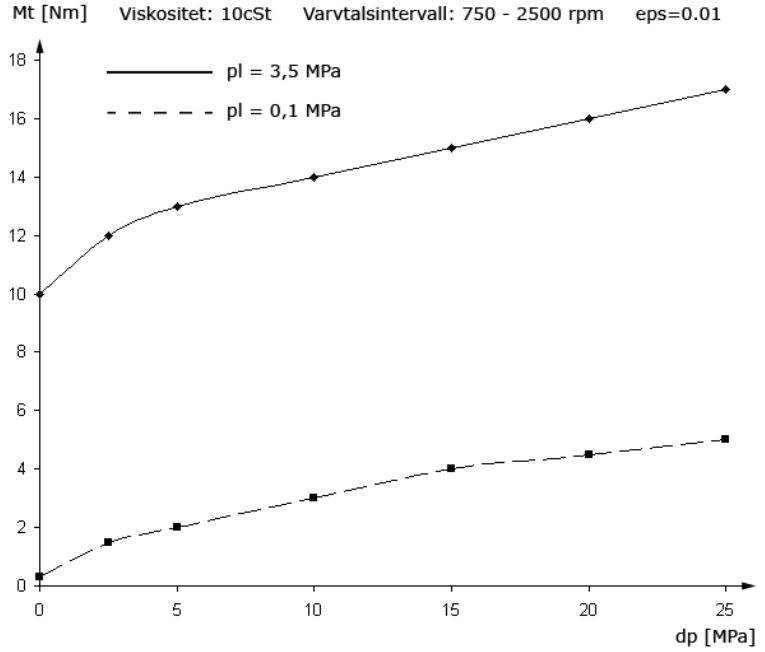


Figure 16: Idling torque for varying loads and external load pressures

Figure 16 shows the resulting plot of the idling torque and how it varies for different pressure levels Δp. Like mentioned earlier, the idling torque is greater for a higher value of the external feed, which is more energy consuming, but give rise to a better response of the pump. Interpolation of the results gives the following function for the idling torque, this when using an external feed of 35 bar.

$$T_{idling} = -8 \cdot 10^{-5} \cdot \Delta p^4 + 0.0046 \cdot \Delta p^3 - 0.0938 \cdot \Delta p^2 + 0.963 \cdot \Delta p + 10{,}033 \qquad (2.3.13)$$

Torque calculations

When the operator demands a motion, he request a torque from the pump that controls that function, this is defined as

$$T_r = \frac{\varepsilon_p \cdot D_p}{2\pi} \cdot \Delta p \cdot \frac{1}{\eta_{hmp}} \tag{2.3.14}$$

The requested power is expressed as

$$P_{shaft} = n_p \cdot T_{tot} \tag{2.3.15}$$

Where the total torque term consist of the following

$$T_{tot} = T_{P1\_lift} + T_{P1\_lift\_idling} + T_{P1\_tilt} + T_{P1\_tilt\_idling} \tag{2.3.16}$$

When the operator signal is zero the $T_{P1\_tilt\_idling}$ and $T_{P1\_lift\_idling}$ will be equivalent to the torque that is needed to rotate the P1 pumps without delivering any flow. When the operator signal is changed from zero, these idling torques will "disappear" and become a part of T_{P1_lift} and T_{P1_tilt}. This has to be realized in a logical function and therefore equation *(2.3.16)* won't be true without considering this.

Power limitation
Equations *(2.3.14), (2.3.15)* and *(2.3.16)* give for both the P1´s, after rewriting and considering the discussion above, that

$$P_{shaft} = \frac{n_p \cdot D_p}{2\pi} \cdot \left(\frac{\varepsilon_{p_i lift} \cdot \Delta p_{tilt}}{\eta_{hmp\_tilt}} + \frac{\varepsilon_{p_i tilt} \cdot \Delta p_{lift}}{\eta_{hmp\_lift}} \right) + n_p \cdot (T_{P1\_tilt\_idling} + T_{P1\_lift\_idling}) \tag{2.3.17}$$

Where $\Delta p = p_p$, this because the pressure at the tank side is about zero. We now set $P_{shaft} \leq P_{\max}$ and receive the following limitation of the P1´s relative displacement.

$$P_{\max} \geq \frac{n_p \cdot D_p}{2\pi} \cdot \left(\frac{\varepsilon_{p_i lift} \cdot \Delta p_{tilt}}{\eta_{hmp\_tilt}} + \frac{\varepsilon_{p_i tilt} \cdot \Delta p_{lift}}{\eta_{hmp\_lift}} \right) + n_p \cdot (T_{P1\_tilt\_idling} + T_{P1\_lift\_idling})$$

(2.3.18)

On the basis of this equation, we can now limit the relative displacement of the P1 pumps so that they never request a greater power that the diesel engine can deliver.

2.4 Measure, analyze and control unit

To achieve an active measure and control of the wheel loader a number of transducers will be installed. A MAC unit will be built, that can read both transducer signals and J1939 CAN-buss messages. This will be accomplished by installing the real time computer CompactRio. To operate the CompactRio, National Instruments professional software system LabView 8.2 will be used. The control algorithms will be coded in Visual C++ and implemented as dynamic link library (dll) files in LabView. This is to increase the clarity of the controllers, and to render possible the usage of the written codes for other programs than LabView.

Chapter 3 HARDWARE AND SIGNALS

The measuring and control systems main task is to, if possible, fulfil the operators' demands. To make this achievable, the system has to have access to a number of information sources on the wheel loader.

3.1 Electrical operator signals

In order to control the system, two electrical joysticks are installed. They come from Parker and are of model IQAN LSL-A02. These joysticks are identical apart from a "kick down" button on one of them. Connection of the three pins used is shown in the table below

Pin	Wire color	Signal
2	Red	Vref +
4	Orange	Boom level
1	Black	Ground

Table 2: Connection table for LSL-A02

The joysticks are fed with 5 Volt and return a boom level reference voltage between 0.5 and 4.5 Volt. It is also recommended to install a dead band in the software so that very small strokes won't be read. In this application the dead band is set between 2.3 and 2.7 Volt.

3.2 Installation of the transducers

In the development process, a number of transducers have to be installed. This is done to acquire enough information to evaluate the system. These transducers will not be needed in a final solution, but are crucial in the development procedure.

3.2.1 Pressure transducers

The pressure transducers that are installed on the wheel loader all comes from Parker. The model name is, IQAN-SP500 and this transducer can measure pressures up to 500 bar. The electrical outlet that is used to connect the transducers are called AMP JPT and are numbered from one to three, where

Pin	Wire color	Signal
1	White	Vref +
2	Green	Transducer signal
3	Brown	Ground

Table 3: Connection table for AMP JPT

According to datasheet [1] the transducers are fed with 5 Volt and return a voltage between 0.5 and 4.5 Volt. In order to maximize their performance they all have been calibrated at VCE:s technical laboratory in Eskilstuna. The results of these have given the offset and sensitivity of the transducers. *Table 4* contains position, type, unique id-number and calibration settings for each of the transducers.

Position	Type	Id-number	Offset mV	Sensitivity mV/MPa
A-chamber lift cylinder	IQAN-SP500	300860619	510,21	80,33
B-chamber lift cylinder	IQAN-SP500	300940619	506,29	80,24
A-chamber tilt cylinder	IQAN-SP500	300700619	502,21	80,55
B-chamber tilt cylinder	IQAN-SP500	300870619	503,79	80,29
Pump feed	IQAN-SP500	300690619	501,93	80,32

Table 4: Information about the pressure transducers

The actual rescaling of the pressure signals (the calibration) takes place in a subsystem in the developed software and is called “Scaling and calibration”.

3.2.2 Position transducers

Two position transducers are installed, one on the tilt cylinder and the other on one of the lift cylinders. They are fed with 10 V and contain a rolled wire that depending on how far the wire is pulled out sends back a reference voltage. This voltage can then be translated to a certain position. Calibration of the position transducers is done by checking the reference voltage for minimal respectively maximum stroke of a cylinder. By comparing this length against the difference in reference voltage, a linear relation between the two is obtained.

The position x in meter as a function of reference voltage U_{ref} in Volt.

$$x = k \cdot U_{ref} + m \tag{3.2.1}$$

The calibration has given the following equations for tilt- respectively lift cylinders

$$x_{tilt} = 0.1434 \cdot U_{ref\_tilt} - 0.0357 \tag{3.2.2}$$

$$x_{lift} = 0.1656 \cdot U_{ref\_lift} - 0.1059 \tag{3.2.3}$$

These transducers prove to be very precise and almost free from noise. This conveys good possibilities to derive the measured position and thereby observe both the speed and the acceleration of respectively cylinder.

3.3 CAN

3.3.1 Overview of CAN

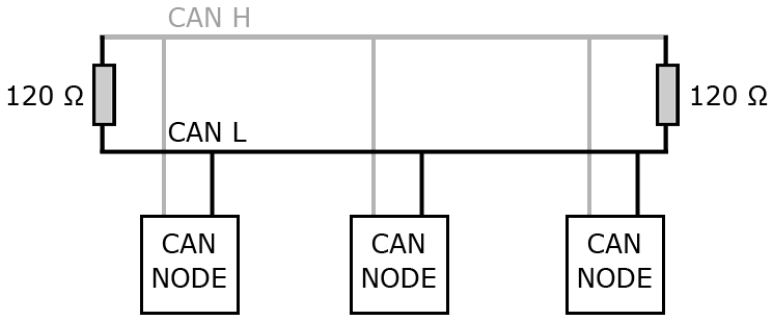


Figure 17: Schematic of a CAN-bus

CAN stand for Controller Area Network and is commonly used in the vehicle industry. The CAN system is built up accordingly to *figure 17* above. It consists of several CAN nodes which measures and controls different parts of a system. It is a serial bus system which uses data frames (bits) to transfer information from one node to another. The data frames are called messages and are continuously sent from the nodes on the bus to communicate with each other. CAN was designed to be robust in the handling of communication where there is a lot of electromagnetic interference, it is therefore ideal for communication in a vehicle. The resistors that are seen mounted in *figure 17* are used to prevent reflections of signals on the bus. They also serve a secondary purpose, and that is to make sure that when no signal is sent over the bus, the signal is zero. This prevents noise from being interpreted as message.

A message consists of the arbitration id, the number of data bytes in the frame and finally the data itself. The standard within CAN specifies two ID formats. First there is the standard format which has an 11 bit identifier and the extended format, which has an identifier consisting of 29 bits. The latter protocol is called J1939.

Figure 18: CAN Message

Because of the fact that the nodes continuously are able to send messages over the bus, the chance of several nodes sending a message at the same time is big. This combined with that the bus only being able to handle one message at a time, collisions of messages will occur if not handled. To handle the collisions, the messages have different id:s and the message with the highest priority is sent through, as seen in the *figure 19*. The messages with the lower priority can retry directly afterward.

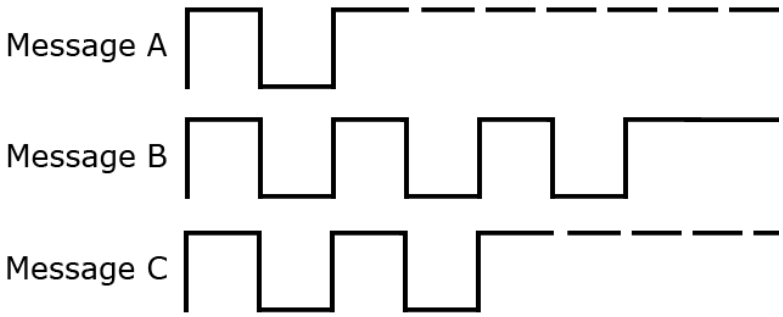


Figure 19: Message ID:s seen in bit form (logic high and low, where low is dominant)

In the illustration above, three messages are sent at the same time. B has the highest priority and is therefore allowed to send itself over the bus. The other ones will have to retry after B has been sent.

A message consists of information from several sources in the system. The information from the different sources is called channels, and it is these bundled together that forms a message.

3.3.2 Connection to the L60's CAN

The L60: s CAN system sends messages over the bus using the J1939 protocol, and it does so at a speed of 250 kbit/s. Several messages are available on the bus, but there is one message in particular that is interesting, and that is the engine speed. To be able to measure this signal an uplink to the wheel loaders machine control unit has to be done. The unit is called the V-ECU and is located behind the driver seat. To connect to the V-ECU, a 16 pin connector has to be used. On this contact only two pins will be read, and that is pin 13 and pin 8 which corresponds to can high respectively can low.

3.4 Valve package control

To be able to control the pilot valves, a PWM signal will be used. PWM stands for Pulse Modulated Width and has the appearance of a square wave as seen in the *figure 20*. The difference between a square wave and a PWM is the time that the signal is logic high. In a PWM, this is called the duty cycle and this can be varied in order to give different outputs. In a square wave this value is static. By using a logic high of 10 V, the output signal can be proportionally controlled in the range 0-10V. This in turn gives the ability of controlling the pilot valves proportionally.

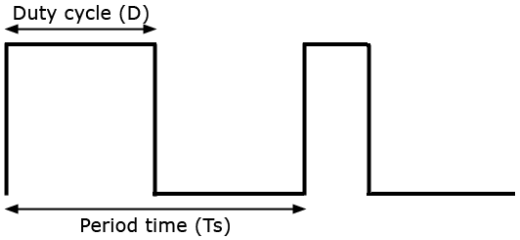


Figure 20: A Pulse Modulated Width signal

According to [4], the average voltage of a PWM wave is given by:

$$\bar{y} = \frac{1}{T}\int_0^T f(t)\cdot dt \qquad (3.4.1)$$

Where $f(t)$ is a square wave which is high

$$y_{\max} \; for \; 0 < t < D\cdot T \qquad \text{(D=Duty Cycle, see } \textit{figure 20}\text{)} \qquad (3.4.2)$$

and low

$$y_{\min} \; for \; D\cdot T < t < T \qquad (3.4.3)$$

The equations above gives:

$$\bar{y} = \frac{1}{T}\cdot\left(\int_0^{D\cdot T} y_{\max}\cdot dt + \int_{D\cdot t}^{T} y_{\min}\cdot dt\right) \Rightarrow D\cdot y_{\max} + (1-D)\cdot y_{\min} \qquad (3.4.4)$$

If $y_{\min}$ is zero the average volt for the PWM is given by

$$\bar{y} = D\cdot y_{\max} \qquad (3.4.5)$$

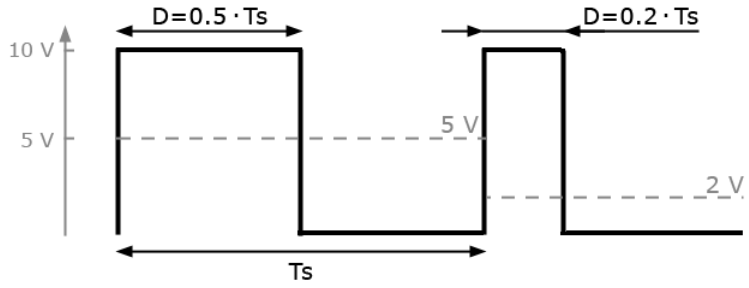


Figure 21: Voltage dependency to the Duty Cycle

3.5 *MAC*

Too be able to feed, monitor and control the components mentioned before, an electrical circuit unit has to be installed. This unit is named MAC, which stands for Measure, Analyze and Control. Appendix A shows the complete electrical diagram of the MAC.

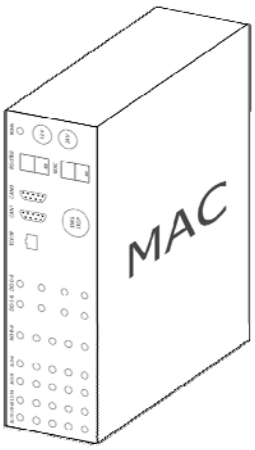


Figure 22: The Measure, Analyze and Control unit

Because the MAC has to feed and control components in the system, it is very important that the MAC itself is fed with enough power. The MAC is fed from a cigarette socket located in the wheel loader cabin. This socket can deliver 24 V and up 10 A. Too prevent a power overload, an 8 A fuse is connected between the socket and the MAC. The power from this 24 V socket provides power to the transducers mentioned in chapter *3.1, 3.2.1* and *3.2.2*. See Appendix G for power consumption calculations.

3.5.1 Connections and connectors

The table below shows the connectors on the front side of the MAC.

Type	Quantity	Connector	Info
Power	2	Socket 2-way	Built in 8 Ampere fuse
Analog in	20	Chassis pin 3 poles	
Analog out	4	Chassis pin 3 poles	
Digital out	8	Chassis pin 3 poles	
CAN	2	Dsub 9 poles	Only CAN 0 is connected
Data communication	1	Reverse SMA	Antenna for wireless communication
Data communication	1	TCP/IP	Communication with cable

Table 5: Connectors used in the system

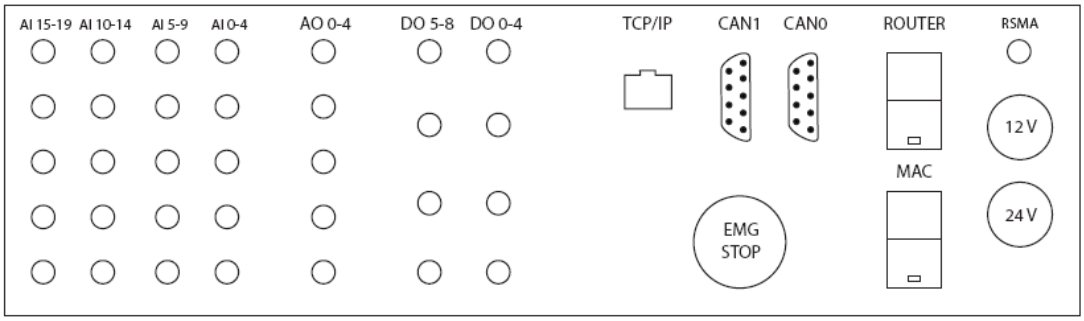


Figure 23: MAC-front

The following table describes how the connections of the control/transducer- cables connected to the MAC. It also shows the voltage that lies over the connection.

ID	Connected unit	Measured/controlled function	U-out [V]	U-in [V]
AI 0	Position transducer	Lift cylinder	10	0,5 – 9,5
AI 1	Position transducer	Tilt cylinder	10	0,5 – 9,5
AI 2	Position transducer	Left steering cylinder	10	0,5 – 9,5
AI 3	Position transducer	Right steering cylinder	10	0,5 – 9,5
AI 4	Pressure transducer	Lift A-chamber	5	0,5 – 4,5
AI 5	Pressure transducer	Lift B-chamber	5	0,5 – 4,5
AI 6	Pressure transducer	Tilt A-chamber	5	0,5 – 4,5
AI 7	Pressure transducer	Tilt B-chamber	5	0,5 – 4,5
AI 8	Pressure transducer	Pump Tilt	5	0,5 – 4,5
AI 9	Pressure transducer	Pump Lift	5	0,5 – 4,5
AI 10	Operator joystick	Tilt	5	0,5 – 4,5
AI 11	Operator joystick	Lift	5	0,5 – 4,5
AI 19	Emergency Stop Button	Emergency Stop		0 – 5
AO 0	Pump P1- lift	Relative displacement lift	-5 – +5	
AO 1	Pump P1- tilt	Relative displacement tilt	-5 – +5	
DO 0	Pilot valve	Tilt AP	0 –24	
DO 1	Pilot valve	Tilt AT	0 –24	
DO 2	Pilot valve	Tilt BP	0 –24	
DO 3	Pilot valve	Tilt BT	0 –24	
DO 4	Pilot valve	Lift AP	0 –24	
DO 5	Pilot valve	Lift AT	0 –24	
DO 6	Pilot valve	Lift BP	0 –24	
DO 7	Pilot valve	Lift BT	0 –24	

Table 6: MAC connectors

3.5.2 Voltage Regulators

The transducers that are used in the system have to be fed with power to give an output signal. The MAC draws its power from the cigarette port in the wheel loader. To be able to use the same power source, a conversion of the voltage has to be done. The setup as seen in *figure 24* below does just this. Depending on the L78XX, conversion to different voltage can be done, see data sheet [2]. The output of the circuit does not depend on the input; therefore variations of the input voltage will not have any effect on the output. Because the pressure transducers have to be supplied with 5V and the position transducers with 10V, two setups of the circuit have to be

built. One converts 24V to 10V using a L7810 and the other one converts 10V to 5V using a L7805.

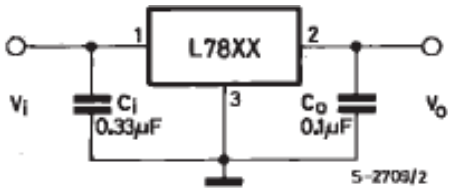


Figure 24: Voltage regulator circuit

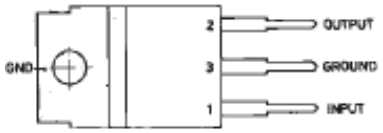


Figure 25: Illustration of the L78XX

3.5.3 Router

By using a wireless router, communication with the MAC can be performed both by using the wireless option, but also by plugging in a TCP/IP cable. The router is fed separately from the MAC by a cigarette socket in the roof of the cabin, which gives an output of 12 V. The reason for why the router is fed separately from the MAC is that the 8 A that is available in the feeding of the MAC is not always enough. This is because the wireless router alone consumes about 3 A. See appendix G.

3.5.4 CompactRIO

The CompactRIO is a computer used in real-time applications. It can be used for both controlling and acquiring data from outside systems. The version used in the new system has four slots which enable the RIO to use four different modules. This results in four different ways of communicating with the outside system. The CompactRIO is a two part computer, where one part, the target, is the computer that holds the code that handles the control loops and data acquisition. The other part is called the FPGA (Field Programmable Gate Array). The FPGA, which in short is a virtual circuit board, uses different modules as an extension of itself to send acquired data to the target and also send data specified by the target.

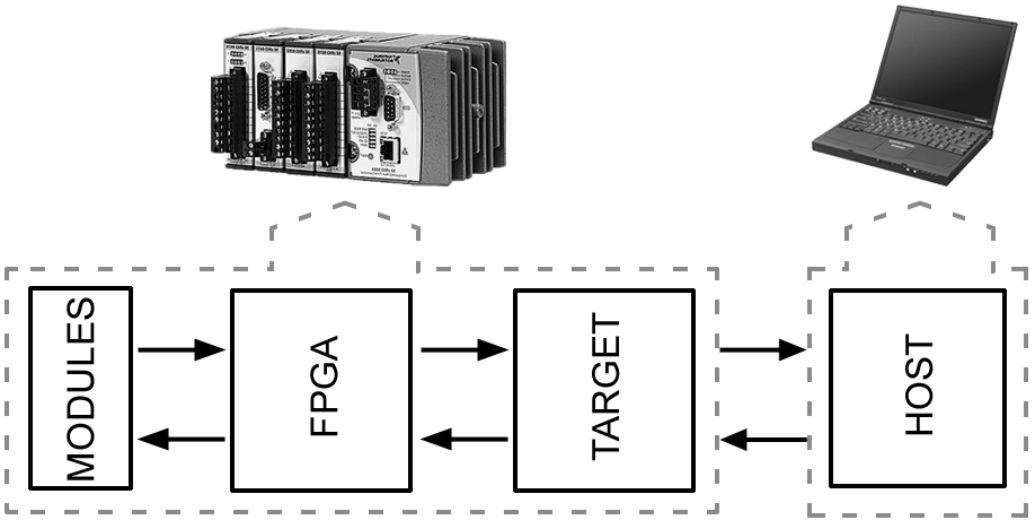


Figure 26: CompactRIO – Parts & Communication

Analog In

To handle analog signals into the system we use a module called 9205, see data sheet [3]. With the 9205 there are three different ways of measuring a signal.

Differential

Each channel has its own ground therefore almost eliminating interference noise from other channels. The downside of using this mode is the loss of channels. This is therefore discarded as an option.

Reference Single-Ended

Each channel is connected to the same ground which enables the use of more channels. This however results in noisier signal. When using this mode, a filtering of the signal in the software is recommended. This option is what is used in new system.

Non Referenced Single Ended

This is a combination between Differential and Single Ended. Tests where made on this mode and showed no improvement on the signal and was therefore discarded.
For each channel there is the option to choose which range the channel should span over. By using this, a better accuracy of the signal is obtained.

Because of the high impedance in the module, there is no way of discovering if a transducer loss has occurred. If there is no signal in one of the channel wires, a capacitor is built up between it and its neighboring channels. This results in a faulty signal being sent to the system. To come to terms with this, resistors of 1 kΩ is mounted between the signal wire and the ground wire as seen in *figure 27* below. This ensures that in case of a transducer loss, the signal will be zero.

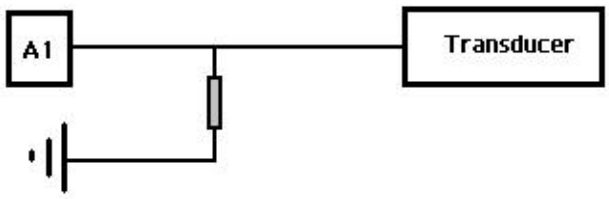


Figure 27: Circuit to enable a signal to be zero

Analog Out

The module that is used to send analog signals is called 9263, see data sheet [4], and has a range of ±10V. Signals can be sent from this module in two different ways.

Single reference

The first one is where all the channels are connected via a common ground. Due to a noisier signal, this one is not used.

Differential

The connection that is used in today's application is called differential and it means that each one of the channels has its own ground. This mode decreases the noise intensity of the output signal.

Digital Out

To be able to create the PWM signal, a module with a very high frequency output has to be used. The pilot valve also requires a PWM with amplitude of 24 V. The module that meets these requirements is called 9474, see data sheet [5]. This module has eight channels out and can be fed externally with 5 to 30 V, thus being able to deliver a signal with the amplitude in this span. It has the ability to change the output signal with a frequency up to 1 MHz, and is therefore suitable for this application.

CAN

Because of the fact that the controllers in this system use the engine speed in some functions, this value has to be obtained. This is solved by using the high speed CAN module 9853, see data sheet [6]. This can handle the J1939 protocol on which signals are sent over the wheel loader bus. The signal on the wheel loaders CAN-bus is sent at baud rate of 250 kbit/s, which means that this parameter has to be changed in the option menu of the module.

Another very usable option in this module and is the ability to use a filter. The filter is used to discard messages before ever entering the software system. In this system only one message is of importance and therefore other messages don't have to be sent to the software for further processing. By sorting out only one message on the bus, a much faster frame to channel conversion can be done. The frame to channel conversion is described in *chapter 4.1.2*.

Chapter 4 Software System

4.1 LabView

To create the functionality that is desired in *chapter 2*, a software system has to be created. The system has to make sure that all the functions that are described in *chapter 3* become functional and meets the requirements that are set. The system and its sub systems are all implemented in LabView 8.2. The system consists of three parts. Two of these are run on the CompactRIO and the third one is run on the host computer. The system is in three parts to optimize the loop rates and for better handling of data logging. Communication between the different parts of the system is done by shared variables. More information about shared variables can be found in *chapter 4.1.4.*

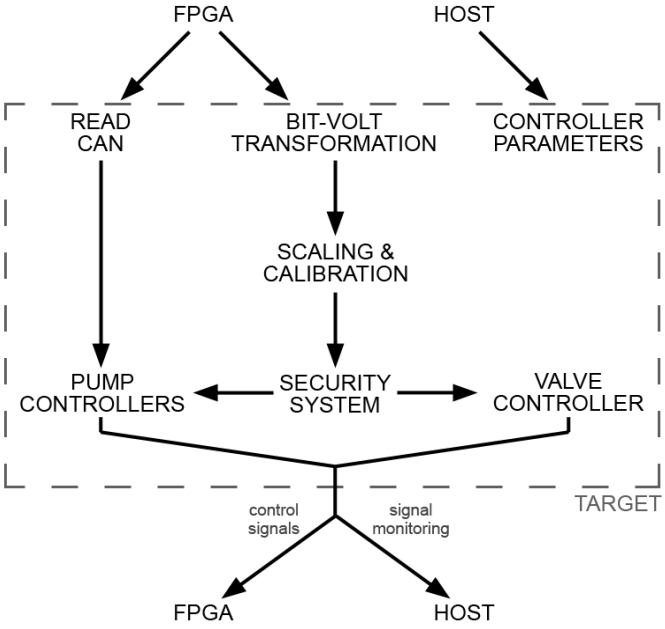


Figure 28: Signal flow through the target

Figure 28 shows how the signals flow through out the three parts of the software system.

4.1.1 FPGA

In the FPGA software system, the module and the channels that will be used in the application are specified. This is done because often all the channels of a module is not needed, and would if specified put extra load on the FPGA

The average output of a PWM can be very close to zero; this means that a very short duty cycle will have to be used. This will result in an extremely short time that the signal is high. To be able to handle these kinds of quick changes of the signal, the place where the PWM is generated has to run in a high frequency. Because the FPGA runs in 40 MHz it is the ideal place for generating this kind of output signal. The

specification of the signal is generated here in a software environment, but it is the module that is mentioned in *chapter 3.6.3* that creates the actual signal.

4.1.2 Target

In this part of the system, all of the controllers are implemented. It also handles what to do with the data acquired by the FPGA. The different signal handlings are described below.

Analog In

The FPGA sends data to the target in bit form with 16 bit precision. To handle these signals a conversion to a more understandable unit has to be made, in this case, voltage. The alteration is done by a simple equation.

$$Voltage = \left(\frac{Signal\ in\ bit\ \ form}{2^{16}} \right) \cdot \left(Measurement\ range \right) \qquad (4.1.1)$$

The measurement range is the input range specified in the option tab of the 9205 module.

The pressure transducer signals all pass through a low pass filter. The filter is used to decrease the noise intensity of the signal and therefore give a much more reliable value for the controllers. This results in stable controllers, and hence this more stable system.

Another conversion has to be made in order to translate the voltage signal in to a signal corresponding to the transducer that is used, e.g. pressure transducer signal will be converted to bar.

Analog out

For the FPGA to handle demanded analog outputs, a similar conversion of the signal as in the analog in has to be made. Here instead, the signal is converted to bit form with 16 bit precision. It is done by the following equation.

$$Signal\ in\ bit\ \ form = \left(\frac{Voltage}{Output\ \ range} \right) \cdot \left(2^{16} \right) \qquad (4.1.2)$$

Digital out

Since the actual PWM signal is generated by the software running on the FPGA, the software on the target simply specifies the period time and the duty cycle of the wave. But for the FPGA to understand the specification, the period time and duty cycle has to be converted into, for the FPGA, an understandable unit. This unit is called ticks. One tick is equal 25 nano seconds which means that the FPGA goes through 40 million of these ticks every second. This means that the period time has to be specified in how many ticks in a second this corresponds to. The duty cycle then uses a percentage of the period to define how many ticks of the period it will be logical high. In [6] it is described that the conversion is done by the following equations:

$$PWM\,period\,(Ticks) = \frac{40000000}{PWM\ \ period\,(Hz)} \cdot \cdot \qquad (4.1.3)$$

$$PWM\ duty\,cycle\,(Ticks) = \frac{PWM\ \ period\,(Ticks) \cdot PWM\ duty\,cycle\,(\%)}{100} \qquad (4.1.4)$$

CAN read

The nodes in LabView can only access the frames, the raw data, from the CAN bus. In this application however it is vital to access the CAN data as channels. The channels are the equivalent of signals which is represented in physical units such as voltage or rps. LabView provides two virtual interfaces, see [6], these two are connected through a virtual CAN bus and it is by using these interfaces a frame to channel conversion can be done.

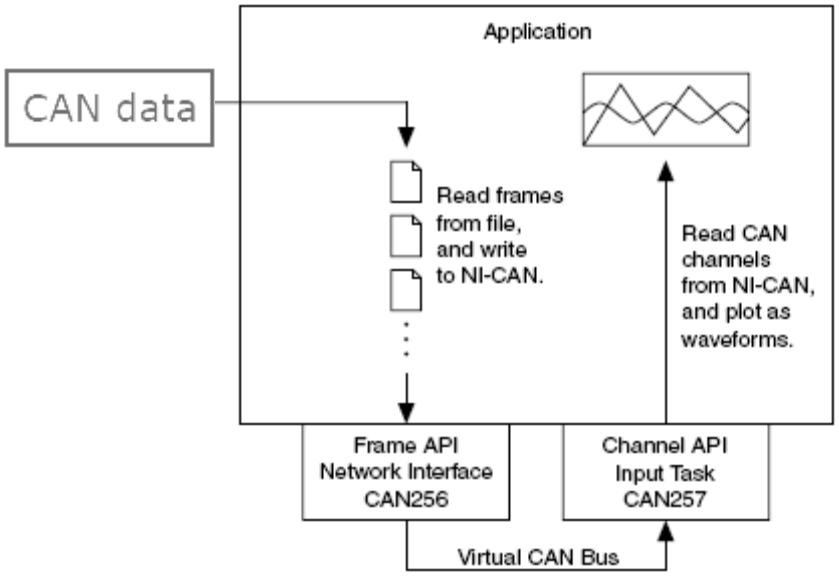


Figure 29: A frame to channel conversion using a virtual bus

To translate the incoming frames to the correct physical unit it has to know the scaling factor of every incoming frame. This is specified by using a can data-base file, see *chapter 4.1.5*

Security system

In case something goes wrong, there has to be a security system that handles this and shuts down the system in a correct manner. Because of the fact that most of the controllers use transducer information to perform calculations, it could be devastating if a transducer loss occurred without the system doing anything about it. To handle this, the system always checks if all transducers are ok. If by any chance they are not, the system tells the valves to close and the pump to set the displacement to zero. By doing this, a faulty signal will not cause the system to go haywire and thereby causing damage to the surrounding area.

The system pressure is limited in the software to the pumps and they are controlled internally to never go over this.

An emergency stop is also mounted on the MAC. This button essentially does the same as in the case of a transducer loss. This button exists for a very quick and correct shutdown of the system in case of an emergency.

4.1.3 Host

The host is the last part of the system and is mainly used to monitor the system and to trim controller parameters. It can also be used for logging data during a run of the system. Logging of data can be done directly on the target, but due to the lack of memory in the CompactRIO, a longer logging session is not recommended. By using the host to log the data, the session is only limited by the free space on the hard drive on the host. The use of the logging function is described in the LabView code.

4.1.4 System Communication

The CompactRIO is indeed a computer, but to let this handle the user interface is just a waste of its capacity. It will result in slow iteration duration and therefore a slow system. To optimize the system, it is divided in to three parts. By doing this, the communication between the systems has to be configured. This is done by introducing shared variables.

Two forms of shared variables will be used. The first form is called a network-published shared variable and is used to transfer information from the target to the host and vice versa. These variables are mainly used to send parameters to the controllers that run on the target, but it is also used to display vital information to the user of the system. The information is sent to the host computer where visualization of the data is possible without slowing down the rest of the system. Another advantage of doing this is mentioned earlier and that is that it enables a very easy way of logging a large amount of data.

The problem with network-published variables is that they are not optimal to use in a time critical loop. Because of this another shared variable will be used, and this one is called a single-process shared variable. This variable is used to send data between loops. In this system the target uses two loops, where one is the control loop and the other one is the loop that handles the data transfer with the host computer. The latter loop is of a lower priority which means that the control loop will execute when it needs to. If there is time, the lower priority loop will then be executed. The single-process variable is used in the control loop and sends data to the data handling loop which uses the network-published variables to send the data on to the host.

4.1.5 Measurement and Automation Explorer

The measurement and automation explorer or MAX as it is also called, is a very versatile program with several application areas when it comes to configuring remote systems such as the CompactRIO. To enable communication between LabView and the CompactRIO an IP address has to be specified in MAX. It is recommended to use

a static IP address on the CompactRIO, this to avoid error in the communication due to that the CompactRIO: s IP address has changed.

Another important thing to notice before using the CompactRIO is to make sure that the software on the CompactRIO is compatible with the one running on the host computer. If they are not compatible, communication will most certainly result in an error. The adding and removal of software on the CompactRIO is also done in MAX.

As mentioned in *chapter 4.1.2*, a CAN data base file (dbc file) has to be used to translate CAN frames to a physical unit. This is also done in MAX, see [6], here a dbc file can be loaded, and from this a selection of different messages is made possible. The dbc file specifies which bits in a message that belongs to each of the channels. It also holds all the information necessary for converting frames to the right unit.

4.2 Implemented Controllers

All of the controllers in this system are implemented in C code, and to read this code in LabView, it has to be converted into a dynamic link library (dll) file. To convert the code, a couple of adjustment in the program has to be done, and these are described in Appendix B.

4.2.1 Valve Controller

Filename: valve_controller.dll **(Appendix C.1)**

The valve controllers' main task is to direct the flow in the right the direction by controlling the valvistors in the valve package. The package is also utilized to hold the load when no operating signal is given. The controller uses signals from the pump controller, described later in this chapter, and two pressure signals to decide in which way to direct the flow. The controllers other task is to make sure that the functionality that is described in *chapter 2.3.1* is fulfilled.

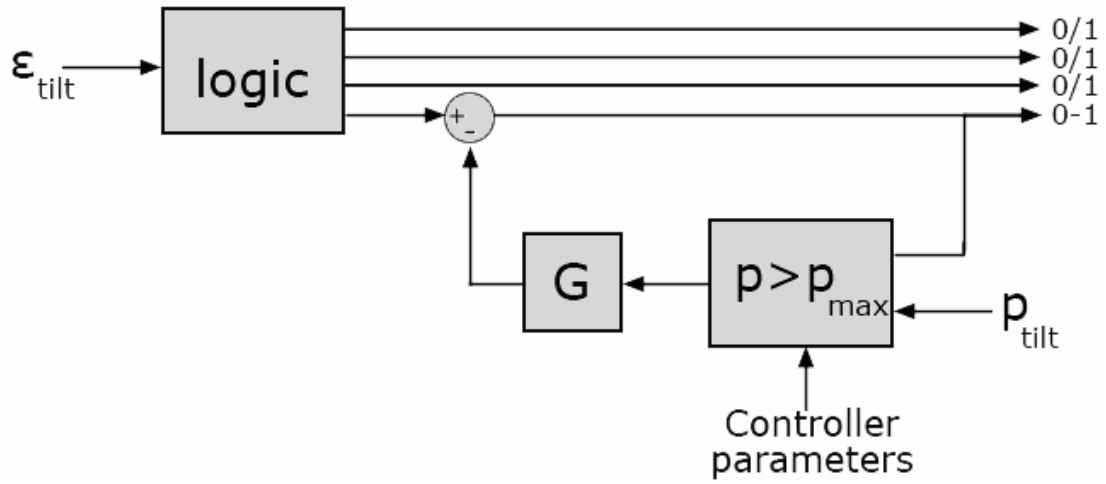


Figure 30: Valve controller

The controller that handles the semi-differential lowering is a simple P-controller. This part of the valve controller is initialized when the pressure in chamber A, in any of the cylinders, exceeds the limit pre set by the operator. The controller will then send a signal to the valve to ensure that the pressure in the chamber is maintained.

A sudden change in the control signal could be the source of initializing oscillations in the system. To avoid this, a type of rate limiter has been implemented. This limiter gives the control signal a slope and thereby eliminates these sudden changes in the proportional signals. It is implemented to give a greater slope the bigger the control error is.

	Description	Name	Unit
Signals In	Relative displacement – tilt	eps_tilt	-
	Relative displacement – lift	eps_lift	-
	Pressure – chamber A – tilt	PA_tilt	bar
	Pressure – chamber A – lift	PA_lift	bar
	Emergency Stop	Em_stop	Boolean
	Loop period	Period	ms
Signals Out	Controlled Valvistors	AP_tilt	-
	See *figure 31* for definitions	AT_tilt	-
		BP_tilt	-
		BT_tilt	-
		AP_lift	-
		AT_lift	-
		BP_lift	-
		BT_lift	-
Parameters	Controller gain tilt	Gain_tilt	-
	Controller gain lift	Gain_lift	-
	Limit for using semi-diff.	Lowlimit_tilt	bar
	Slope in control signal	Highlimit_tilt	bar
	Limit for using semi-diff.	Lowlimit_lift	bar
	Slope in control signal	Highlimit_lift	bar
	Time for closing valves	Zero_tilt	s
	Time for closing valves	Zero_lift	s
	Time to initialize press down	Pdtime_tilt	s
	Time to initialize press down	Pdtime_lift	s
	Press down pressure	Pa_press_tilt	bar
	Press down pressure	Pa_press_lift	bar

Table 7: Signals and parameters of valve_controller.dll

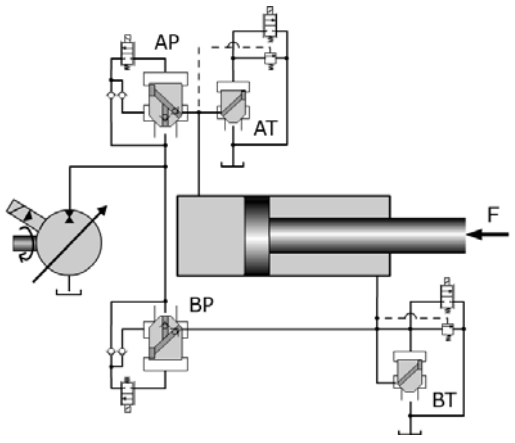


Figure 31: Valvistors name and position in the system

4.2.2 Pump Controllers

The pump controllers are divided in three dll files. This is because of different reasons; one is that it is easier to isolate them if an error occurs. The first file is named and contain as follows

4.2.2.1

Filename: pump_logics.dll **(Appendix C.2)**

- Relative displacement scaling factor
  A scaling factor of the relative displacement that can be useful when only small strokes is desired e.g. at experiments or security tests.
- Dead band of the joysticks
  This functions neglects very small strokes of the joysticks
- Velocity control
  Declared separate below
- Maximum flow control
  Declared separate below
- Handling of transducer losses
  The error message U_error indicates whether any pressure transducer has lost its connection. If so happens, this will be indicated on the host computer and the relative displacement of the function it is connected to, will be set to zero.
- Maximum flow take out status
  Declared separate below

	Description	Name	Unit
Signals In	Joystick signal – Tilt	U_joy_tilt	V
	Joystick signal – Lift	U_joy_lift	V
	Dead band in the transducers	U_dead_band	V
	Pump rpm	n_p	rev/s
	Volumetric efficiency of pump - tilt	etha_volp_tilt	-
	Volumetric efficiency of pump – lift	etha_volp_lift	-
	Scaling factor – tilt	eps_scale_tilt	-
	Scaling factor - lift	eps_scale_lift	-
Signals Out	Computed relative displacement – tilt	eps_t_op	-
	Computed relative displacement – lift	eps_l_op	-
	Error message – Transducer loss	U_error	Integer
	Indicates if any pump woks at full Dp	pump_limit_reached	Boolean
Parameters	A-chamber area in tilt cylinder	A_t_a	m^2
	B-chamber area in tilt cylinder	A_t_b	m^2
	A-chamber area in lift cylinder	A_l_a	m^2
	B-chamber area in lift cylinder	A_l_b	m^2
	Minimum voltage from transducer	U_min	V
	Maximum voltage from transducer	U_max	V
	Maximum cylinder speed - raise	v_raise_max	m/s
	Maximum cylinder speed - lower	v_lower_max	m^2
	Pump displacement	Dp	m^3/rev

Table 8: Signals and parameters of pump_logics.dll

Dead band
The operator signals are measured using two electrical joysticks according to *chapter 3.1*. The operator signals are read by MAC and sent to the CompactRIOs' target. A dead band has been implemented in order to neglect small strokes of the tilt- and lift joysticks. The voltage range of which the dead band is active is between 2.3 and 2.7 Volt. The size of the dead band span can easily be changed in the host.

Velocity control
Since the operator signal in to the controller is a voltage and not the desired speed, further conversion must be done. By measuring the voltage for different strokes, a function can be designed where a specific voltage corresponds to a certain stroke factor for respectively function. This factor is in its turn directly proportional to the requested velocity

$$x_{vref} = \frac{U_{ref} - U_{\min}}{U_{\max} - U_{\min}} \tag{4.2.1}$$

$$v_{cyl} = x_{vref} \cdot v_{\max} \tag{4.2.2}$$

Maximum flow control
The same file also includes some maximum flow logics that, in current drive case, controls if the pump can deliver the requested speed of the cylinders. Thus the control of the flow will, if it has to, put restriction on the speed of the cylinders. In order to fulfil a speed control the following equations are observed.

For positive cylinder strokes

$$\varepsilon_p = \frac{A_A \cdot v_{cyl}}{Dp \cdot n_p \cdot \eta_{volp}} \tag{4.2.3}$$

And for negative cylinder strokes because of the area difference in the A and B chamber.

$$\varepsilon_m = \frac{(A_A - A_{piston\_shaft}) \cdot v_{cyl}}{Dp \cdot n_p} \cdot \eta_{volm} \tag{4.2.4}$$

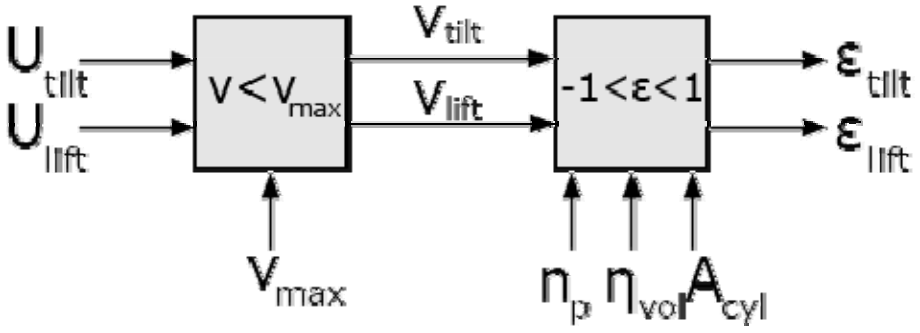


Figure 32: Velocity and maximum flow control

The regulator controls which relative displacement the requested speed corresponds to. As long as this value is in between -1 and +1, no limitation takes place. If the requested velocity on the other hand is equivalent to a relative displacement outside of -1 to +1 the displacement will be scaled down to -1 or +1. This can be interpreted as that the maximum velocity is reduced.

The volumetric efficiency η_{vol} of the pump is sent as a signal in to the *pump_logics* controller. This value can either be set to a constant or by implementing a function out side of the controller. The default value has been set to 0.95.

4.2.2.2
Filename: pmin.dll (Appendix C.3)

	Description	Name	Unit
Signals In	Pump pressure on feed port – tilt	pp_t	Pa
	Pump pressure on feed port – tilt	pp_l	Pa
	Relative displacement in – tilt	eps_t_op	-
	Relative displacement in - lift	eps_l_op	-
	Control gain – lower	G_l	-
	Control gain - raise	G_h	-
	Minimum pump pressure	pmin	Pa
	Control difference in pressure	pmax	Pa
	Standby factor before deactivating the controller	delta	-
Signals Out	Relative displacement out – tilt	Eps_t_op_reg	-
	Relative displacement out - lift	Eps_l_op_reg	-
Parameters	Gain – tilt controller	Deps	-
	Gain – tilt regained pressure	Deps1	-
	Gain – lift controller	Deps2	-
	Gain – lift regained pressure	Deps3	-

Table 9: Signals and parameters of pmin.dll

Minimum load pressure controller
When lowering a load there is the issue that the pump can draw oil out of the system faster than the load is able to. This results in a pressure drop in the system which in turn can lead to cavitation. Because of that this phenomenon is corrosive on parts in the system, and a controller that attends to this problem has to be implemented.

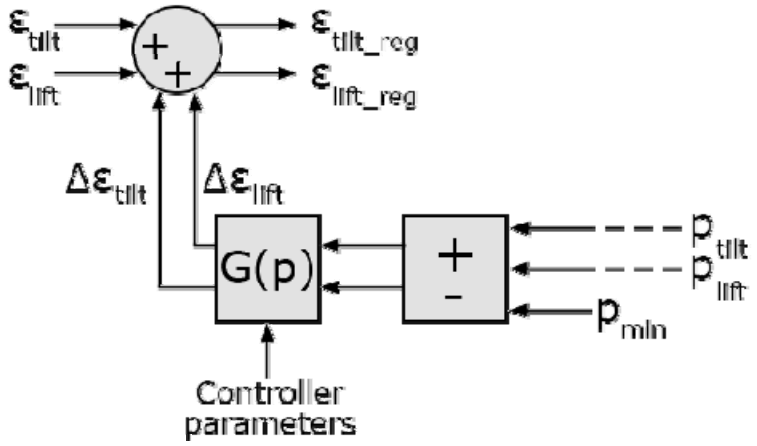


Figure 33: Simplified illustration of the min-pressure controller

The controller is a Proportional controller which looks at the pump pressure at each respective pump. If the pressure drops below a certain level, the controller activates and increases the displacement to increase the pressure to an acceptable level.

If the pressure drops below $p_{\min}$ the following action will take place on the relative displacement

$$\varepsilon_{out} = \varepsilon_{ref} + \Delta\varepsilon \tag{4.2.5}$$

where

$$\Delta\varepsilon = -G \cdot p_{\max} \cdot \frac{(p_p - p_{\min})}{(P_{\max} - p_{\min})} \tag{4.2.6}$$

4.2.2.3
Filename: P_controller.dll (Appendix C.4)

- Hydraul-mechanical efficiency modulation
  Declared separate below
- Maximum power controlling of the pumps
  Declared separate below
- Security system
  Contains a guardian function that checks for loss of CAN-communication. A loss of the CAN-communication will be indicated on the host computer and the relative displacement for both pumps will be set to zero. The error will also be sent to the valve controller so that all the valves close as well.
- Verifier of the boundaries in the relative displacement
  Makes sure that the outgoing relative displacements are within the boundaries of -1 to +1, if not the closest approved value will be set.

	Description	Name	Unit
Signals In	Maximum total power take out	P_max	W
	Pump rotational speed	np	rps
	Pump pressure – tilt	pp_tilt	Pa
	Pump pressure – lift	pp_lift	Pa
	Emergency stop	Em_stop	Boolean
	Relative displacement in – tilt	eps_tilt	-
	Relative displacement in – lift	eps_lift	-
Signals Out	Idling energy take out - tilt	T_eps0_tilt	Ws
	Idling energy take out – lift	T_eps0_lift	Ws
	Hydraul mechanical efficiency - tilt	etha_hm_tilt	-
	Hydraul mechanical efficiency - lift	etha_hm_lift	-
	Relative displacement out – tilt	eps_tilt	-
	Relative displacement out – lift	eps_lift	-
	Computed actual power take out	P_ref	Ws
	CAN communication failure message	error_msg	Boolean
Parameters	Constant for computing etha_hm	b0P	-
	Constant for computing etha_hm	b1P	-
	Constant for computing etha_hm	b2P	-
	Constant for computing etha_hm	b3P	-
	Constant for computing etha_hm	b4P	-
	Constant for computing etha_hm	b5P	-
	Constant for computing etha_hm	b6P	-
	Kinematic viscosity	my	Ns/m^3
	Damping of the pumps - δ_p	delta_p	-
	Constant for computing T_eps0	k_eps0	-
	Constant for computing T_eps0	m_eps0	-
	External feed pressure	pL	Pa
	Pump displacement	Dp	m^3/rev
	Pressure difference	dp_lift	Pa
	Pressure difference	dp_tilt	Pa

Table 10: Signals and parameters of P_controller.dll

Hydraulic mechanical efficiency modulation

A controller has been developed, that in every time step computes the hydraul-mechanical efficiency. This derives from the discussion and equations in *chapter 2.3.4*.

Maximum power controlling of the pumps

The hydraulic system of today has got a function that always gives priority to the tilt cylinder before the lift cylinder. The maximum power controller is built up in a similar way. If the operator requests a relative displacements that in all exceeds $P_{\max}$ the $\varepsilon_{p_l lift}$ will be cut back first. If this is not enough the $\varepsilon_{p_l tilt}$ will also be reduced. The controller sees to that $P_{\max}$ is maintained as long as the operator demands this or more.

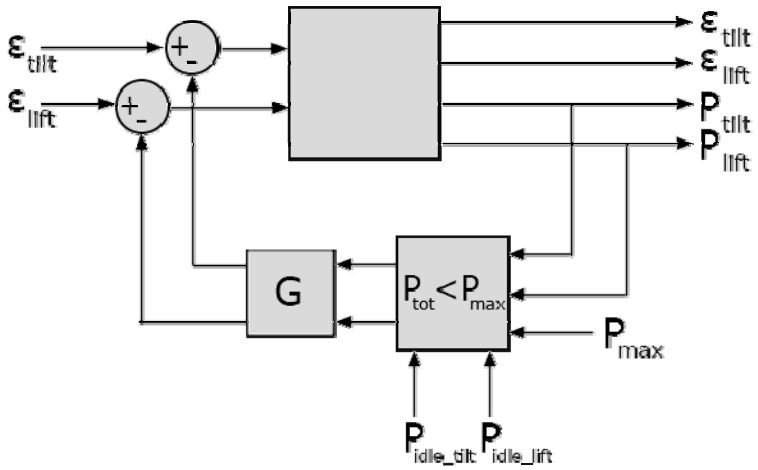


Figure 34: Simplified illustration of the maximum power controller

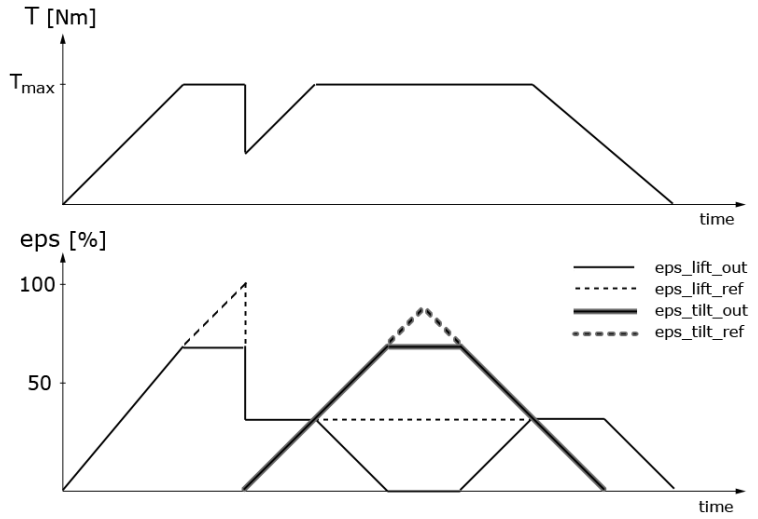


Figure 35: Maximum torque and function priority

In this thesis, $P_{\max}$ has been simplified to correspond to a constant value. This is because of the complex nature in modelling the torques acting in the wheel loader. $P_{\max}$ is set to the maximum power take out of the hydraulic pump installed in the L60E today. This value was estimated in *chapter 2.3.4*.

The controller operates according to the following equation

$$P_{\max} \geq \frac{n_p \cdot D_p}{2\pi} \cdot \left(\frac{\varepsilon_{p_i lift} \cdot \Delta p_{tilt}}{\eta_{hmp\_tilt}} + \frac{\varepsilon_{p_i tilt} \cdot \Delta p_{lift}}{\eta_{hmp\_lift}} \right) + n_p \cdot (M_{P1\_tilt\_dry} + M_{P1\_lift\_dry} + M_{Rex1} + M_{Rex2}) \quad (4.2.7)$$

where the hydraulic mechanical efficiency

$$\eta_{hmp} = \frac{1}{1 + \frac{b_{0P}}{\varepsilon} + b_{1P} + (\frac{b_{2P}}{\varepsilon} + b_{3P}) \frac{p_L}{\Delta p} + b_{4P} \frac{|p_H + \delta_p \cdot p_L|}{\varepsilon \cdot \Delta p} + b_{5P} \frac{2\pi \cdot \mu \cdot n}{\varepsilon \cdot \Delta p} + b_{6P} \frac{(\varepsilon \cdot n)^2}{\Delta p}} \quad (4.2.8)$$

Chapter 5 Experiments

A series of tests to check, analyze and verify both previous and newly developed systems has been performed in order to fulfil the requirements in *chapter 1*. All of the tests have been achieved using the developed software platform MAC, described in *chapter 3.5*. The performed tests can be summarized in to the following categories.

- Modelling the cylinder and framing joint friction
- Study the hydraulic pressure losses due to connectors
- Examine the functionality of today's hydraulic lowering of the lift cylinders
- Perform an energy consumption calculation of today's hydraulic system, using the short duty cycle as reference
- Explore whether a valvistor can operate as a pressure reducer or not, this fed back both electrically and hydraulically.

5.1 Cylinder and framework friction

To describe the friction forces in a hydraulic cylinder with a formula is a complex challenge that demands a pretty deep analysis. One of the reasons to this is that the friction depends on so many factors and that these often prove to have nonlinear characteristics. The pressure in the cylinder chambers, speed of the piston, temperature and type of hydraulic oil are some examples that strongly affect the resulting friction force. Another important notice is that the friction force acts both between the piston and the cylinder wall F_{fk} and on the surface between the piston shaft and cylinder wall F_{fs}.

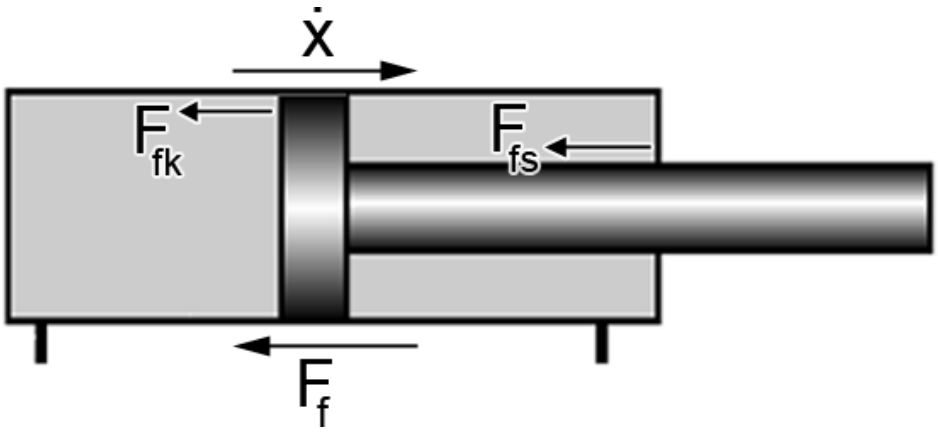


Figure 36: Acting friction forces

$$F_{f\_cyl} = F_{fk} + F_{fs} \tag{5.1.1}$$

The friction force can be divided in to two parts, static friction and kinetic friction. The first mentioned give rise to a dead band and mostly affects the total friction force for low speeds of the piston. In opposite to the static friction, the kinetic friction grows for increasing speeds.

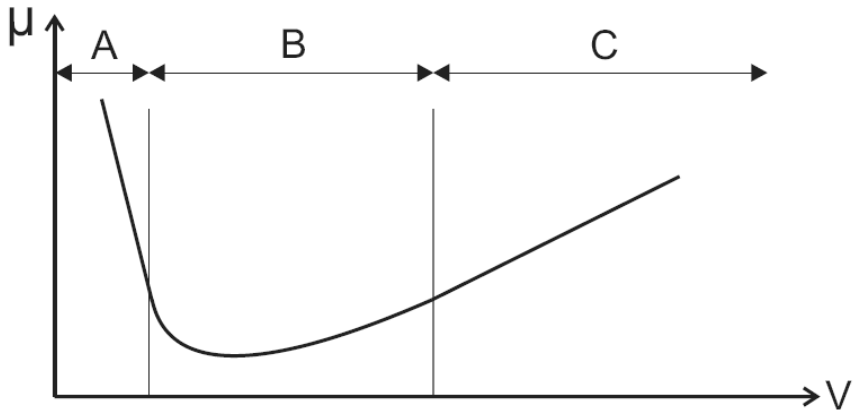


Figure 37: Striebeck curve; A is static friction, B transition - and C kinetic friction [7]

5.1.1 Objectives

The purpose with this task is to create a detailed model that describes the cylinder friction in the lift- respectively tilt cylinders.

5.1.2 Method

When the cylinder friction model will be used in a simulation model of a wheel loader, model L60E, it ought to contain parameters that is accessible there. Inquiries, mostly in 2006 master thesis *Smart cylinder* [7], leads with this discussion to that a parametrisized friction force model can be produced that depends on piston speed and pressures in A- respectively B chamber.

The model looks as follows

$$F_f = x_1 e^{x_2 \cdot |v|} + x_3 (p_1 - p_2) + x_4 p_2 + x_5 v \tag{5.1.2}$$

The model is nonlinear and consists of five constants. In order to compute these, a nonlinear problem has to be solved. This also assumes that it is possible, by using experiments, to receive measured data of the friction force, cylinder speed and pressures.

5.1.3 Determining Ff

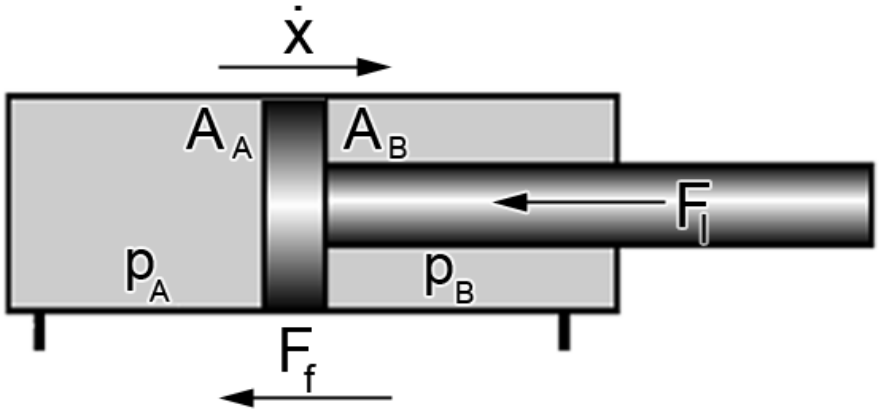


Figure 38: Acting forces when raising a piston with a load

F_f is set by the following stationary relationships

For $\dot{x} > 0$

$$F_l + F_{f\_cyl} = p_A A_A - p_B A_B \tag{5.1.3}$$

For $\dot{x} < 0$

$$F_l - F_{f\_cyl} = p_A A_A - p_B A_B \tag{5.1.4}$$

F_l is due to geometrical reasons in the wheel loader depending on the cylinder strokes x.

$$F_l = F_{load}(x) + F_{f\_joints} \tag{5.1.5}$$

$$F_f = F_{f\_cyl} + F_{f\_joints} \tag{5.1.6}$$

F_l can be divided in to two parts where the first one depends on the load in the bucket and the cylinder stroke x. The second term is the friction force that acts in the joints of the frameworks. Stationary it is valid that F_l is proportional to x for both positive and negative strokes. Thus an experiment where pressure and cylinder position get measured for constant $\dot{x}$ in positive and negative cylinder strokes gives us F_f because $F_{load}(x)$ than can be eliminated.

$(5.1.3) - (5.1.6) \Rightarrow$

$$F_f = \frac{(p_{A\_1} A_A - p_{B\_1} A_B) - (p_{A\_2} A_A - p_{B\_2} A_B)}{2} \tag{5.1.7}$$

This results in a way to perform tests to obtain F_f for varying loads and speeds. These can in turn be used to determine the constants x(1:5).

5.1.4 Test performance

The tests where done by raising and lowering the load at a constant speed. This will be done with all other functions completely still. During this time p_A, p_B and x is logged from the current cylinder using the MAC. The test is repeated for different speeds and loads in order to generate different pressure levels. With these tests done, the F_f for respectively speed and pressure level can be calculated.

5.1.5 Analysis and results

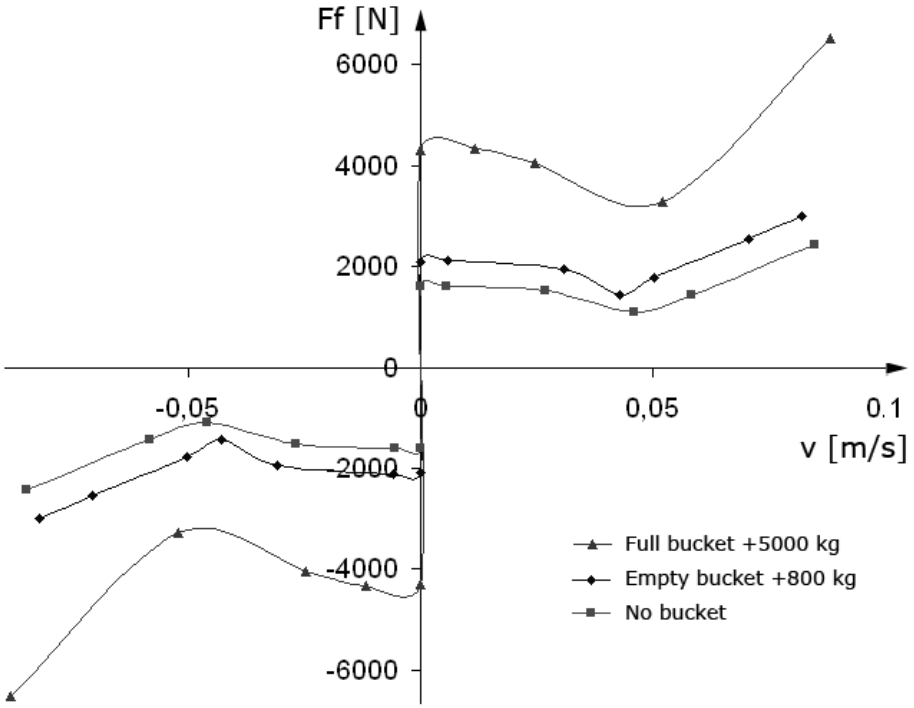


Figure 39: Friction forces of the lift cylinders as a function of v for varying loads

The plot above shows the result from a test made on the lift cylinders using three different loads. This data can be found in appendix D.2. With collected points for different loads and speeds it is possible to make a determination of the parameters in *equation 5.1.2*. This is done by using a nonlinear least square method. The calculations are written in m-code in Matlab and are shown in appendix D.1. The function that is used is called *fminunc* and can be found in the Matlab add-on "optimization toolbox". *fminunc* attempts to find a minimum of a scalar function of several variables, starting at an initial estimate. This is generally referred to as an unconstrained nonlinear optimization.

By defining f, and minimize this using calculated Ff with corresponding pressures and speeds, the parameters x(1:5) can be determined.

$$f = \sum (x_1 e^{x_2 \cdot |v|} + x_3 (p_1 - p_2) + x_4 p_2 + x_5 v - F_f)^2 \qquad (5.1.8)$$

After executing the script using the settings showed in appendix D.1, the following x values is generated for respectively cylinder.

$$x_{lift} = \left[1191{,}2,\ 20{,}815,\ 23{,}919,\ 21{,}516,\ -58221\right] \quad (5.1.9)$$

$$x_{tilt} = \left[674{,}39,\ 20{,}615,\ 30{,}753,\ 81{,}541,\ -17957\right] \quad (5.1.10)$$

5.2 Losses due to connectors

In order to get an insight over the hydraulic losses due to hydraulic connectors in the system, a pressure loss test was carried out. This was realized by performing a movement of one function, tilt or lift, when all other functions kept still. During this time the outlet pressure of the pump is measured and compared with the pressure in the cylinders inlet chamber. The test was done without heavy loads or movements so that the pump always where able to keep a system pressure of 20 bar above the maximum load pressure. Thus the difference between the measured pressures was 20 bar, that is the result of the pressure loss over the valve, plus any additional losses in the system. The test was carried out about 20 times with varying pressure levels and flows. The objective of the test was to use its results to design a model of the pressure losses for varying loads and flow. Unfortunately no dependencies could be found from the tests, but an average pressure drop of 2.35 bar appeared. Reasons for this are partly because the pressure differences are so small that noise and uncertainty from the transducers strongly affects them. Another reason might be that the 20 bar that the pump is set to keep over the highest load pressure is not all that precise and spreads for varying test cases.
The conclusion, despite everything, is that the pressure losses in the system where measurable and pretty constant around 2.35 bar.

5.3 Functionality test - Lowering of load

The valve in the current system contains a function that means, when lowering the lift cylinders; the A-chamber is filled with oil from the B-chamber. This can be described as a kind of differential lowering. In order to determine the amount of energy that is being consumed for different test cases, an analysis of when this function is active or idle has to be done. A case where the function described above aught to work, is when a positive stroke on the tilt cylinder is done, and a lowering of the lift cylinder is activated during this time. In *figure 40* it is shown that the pressure from the tilt function, ”Servo pressure tilt in”, is at this time led past check valve (8) to the spring side of the right load carriage valve (2). Thus the load carriage valve will be closed, and the flow into the B-chamber will instead occur through the anti cavitation valve (6), that receives its oil from the A-chambers out flow. We call this function SPID, Servo Pressure Induced Differential lowering. This function does not however exist for the tilt function and hence this, all the tilt movements are carried out with pump delivering all the flow.

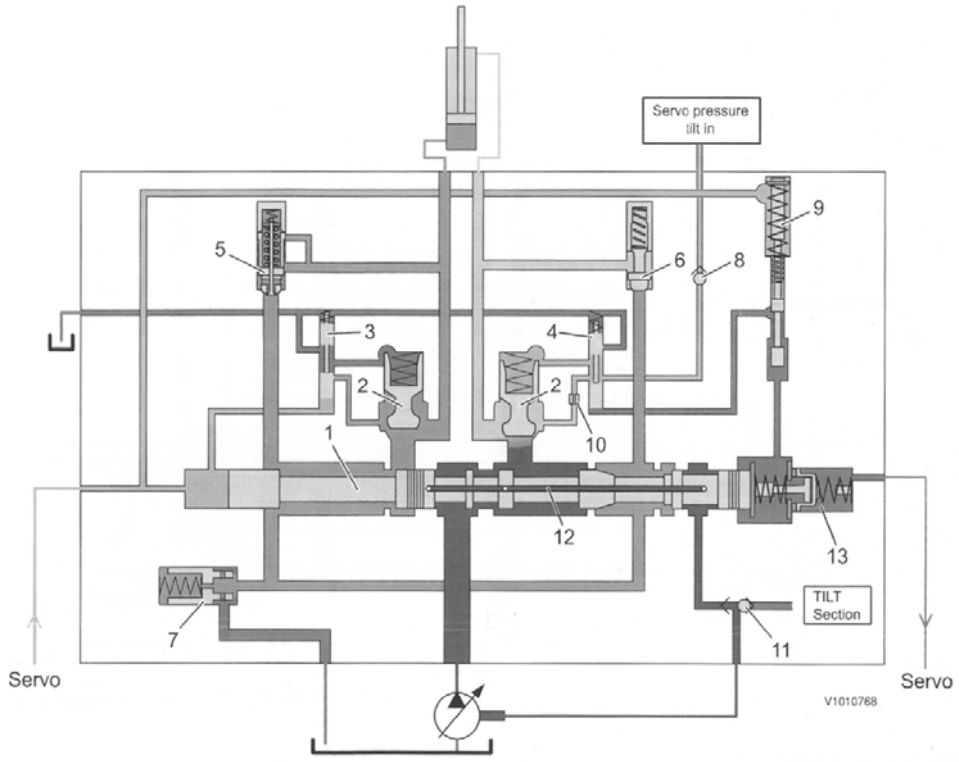


Figure 40: Hydraulic scheme for the lift side of the load valve [8]

To verify the functionality of the SPID, a series of tests where done on the wheel loader. To do this, a flow transducer had to be temporarily installed, and then a number of tilt- and lift movements where performed. The result that is seen in Appendix E shows that the function tested only works a few times in the cases where it should be activated.

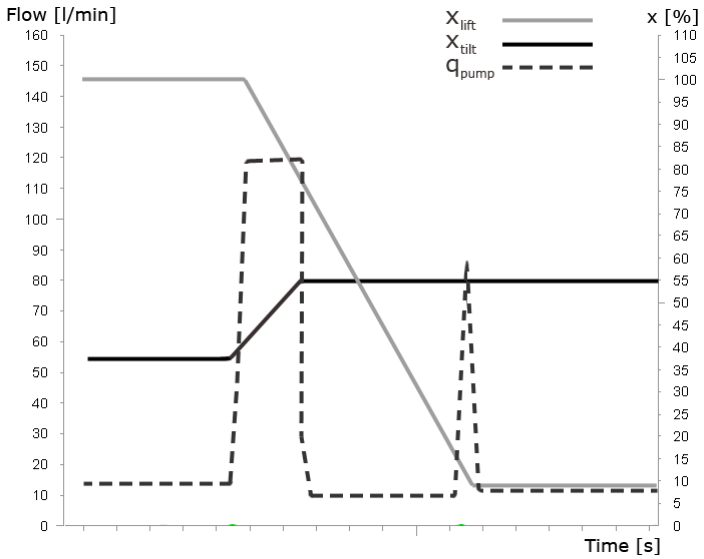


Figure 41: Illustration of a lift cylinder lowering test

Figure 41 illustrates a test where lowering of the lift cylinder x_{lift} is initiated right after a positive stroke is done on the tilt cylinder (increase of x_{tilt}). The dashed line

shows the variation in pump flow during this test. As seen in the test, the load carriage valve (2) is closed because of the operation done on the tilt cylinder. By doing this, no flow is delivered from the B-chamber during a certain time. In time, the pressure that holds (2) down is reduced, allowing it to open during the last part of lowering the lift. Flow is now being delivered from the pump.

The test shows that the SPID function works as it should in the case presented above. However, in some cases illustrated in Appendix E, the function does not always work. Reasons for why the function doesn't work in cases where it should work can not be deciphered from these tests. Thus the conclusion is that there is no way of knowing if the function will work during e.g. a short duty cycle (*chapter 5.4.2)*.

5.4 Energy analyze of today's working hydraulics

5.4.1 Objectives

The energy analyse is divided in some main goals.

- Get an insight in how much energy that is consumed in a short duty cycle
- Be able to measure and calculate how much of this energy that is losses due to hydraulic, and to get an insight of what these losses depends on.
- With knowledge of above, calculate a theoretical energy consumption for the new hydraulic system.

5.4.2 The short duty cycle

"The wheel loader is a complex system where drives, such as working hydraulics, transmission, brakes and steering, constantly interact with each other. To evaluate the energy efficiency of the working hydraulics in a wheel loader, a typical short duty cycle ha been studied, see *figure 42*. Performing this cycle is often the main purpose of these machines; therefore, the cost effectiveness of the cycle is economically significant to the entrepreneur. The cycle last about 30 seconds and includes the machine going into a pile of granular material, lifting a full bucket, reversing out of the pile, forwarding to empty the bucket in a truck, and going back out while lowering the empty bucket." (Source: [1])

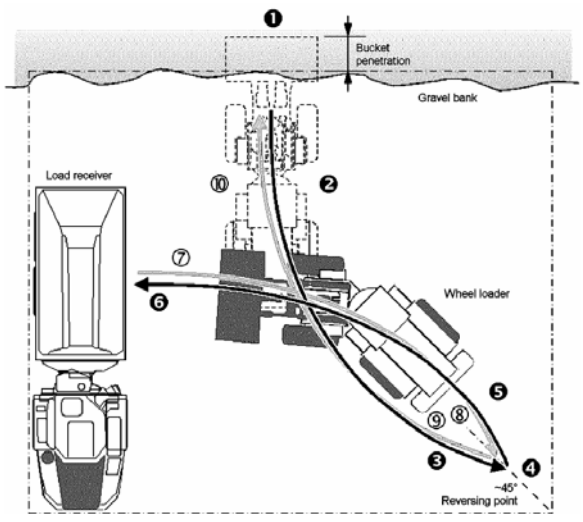


Figure 42: Wheel loader movement during duty cycle [1]

5.4.3 Method

In order to make the test comparable to a real situation, the short duty cycle was used. During this cycle, position, speed and pressure of the cylinders and pump feed port will be measured. With these, it is then possible to calculate the consumption of energy per cycle, using the following equation.

$$P = q \cdot \Delta p \tag{5.4.1}$$

Depending on which pressure difference that is compared, it is possible to calculate the usage from different parts in the system. In the following model, the hydraulic system is assumed to be free of leakage between the pump and the cylinder.

$$q_{tot} = q_{cyl\_lift} + q_{cyl\_tilt} = A_{cyl\_lift} \cdot \dot{x}_{cyl\_lift} + A_{cyl\_tilt} \cdot \dot{x}_{cyl\_tilt} \tag{5.4.2}$$

The flow will therefore be calculated from the speed of the cylinder, or cylinders that are in motion. The area for respectively cylinder will nevertheless be different for positive and negative cylinder strokes.

$$\dot{x}_{lift} > 0 \quad \rightarrow \quad A_{cyl\_lift} = A_{lift\_A} \tag{5.4.3}$$

$$\dot{x}_{lift} < 0 \quad \rightarrow \quad A_{cyl\_lift} = A_{lift\_B} \tag{5.4.4}$$

And corresponding for the tilt cylinder

$$\dot{x}_{tilt} > 0 \quad \rightarrow \quad A_{cyl\_tiltt} = A_{tilt\_A} \tag{5.4.5}$$

$$\dot{x}_{tilt} < 0 \quad \rightarrow \quad A_{cyl\_tilt} = A_{tilt\_B} \tag{5.4.6}$$

With knowledge of the flow that the pump delivers, the power for different pressure differences can be defined. The total power that is consumed by the working hydraulic is defined as

$$P_{tot} = p_p \cdot (q_{cyl\_tilt} + q_{cyl\_lift}) \tag{5.4.7}$$

Notice that in cases where the tilt in function described in *chapter 5.3* is active, which leads to that the flow, q_{cyl_lift} will be neglected because the flow in these cases is taken differentially from the B-chamber of the tilt cylinder.

Now the difficulties in the calculation of the energy consumption begins, this because it is not always certain that the SPID function actually is activated when it aught to.

The power that is lost between pump and respectively cylinder is defined as

$$P_{loss} = q_{cyl\_lift} \cdot (p_p - p_{cyl\_lift}) + q_{cyl\_tilt} \cdot (p_p - p_{cyl\_tilt}) \tag{5.4.8}$$

where p_{cyl} corresponds to the pressure in A- respectively B-chamber for $\dot{x} > 0$ and $\dot{x} < 0$. This term can further on be split up in to two different types of losses, connector losses and valve related pressure losses. The discussion in *chapter 2.2* leads to that a dividing of P_{loss} is not all that trivial.

5.4.4 Test performance

The test was performed by an experienced operator in VCE:s test area in Eskilstuna, where the short duty cycle is driven repeatedly for a number of times. During this time, the MAC logged the following data:

- p_p – Pump pressure
- p_{cyl} – Cylinder pressure in A- and B-chamber in lift- respectively tilt cylinder
- x – Position of the lift and tilt cylinders

Rain on the day the test was performed lead to that the loaded gravel where heavier than usual. This resulted in that the hydraulic often where working on maximum. Further on it can be mentioned that the test where performed with a horizontal function turned on, but a hydraulic oscillation attenuator turned off.

5.4.5 Preparation of saved data - datagen

To both in a functional but also in a re-useful way analyse and calculate values of the energy consumption, all the operations has been written in Matlab m-files. These are designed so that they are easy to both understand and reuse if new tests will be done in the future.

During the time the test was performed, data was logged using the MAC and the filename where set to *lcycle*. As always during pressure measurements, noise from the transducers will occur. Before starting to analyze the measurement files, it is important to minimize this noise. Filtering and dividing the relevant data is made in the function *datagen*. The code for it is attached in Appendix D.3 and has the following help prompt in Matlab

```
% ----------------------------
Write in following form:
[t, pla, plb, pta, ptb, xl, xt, vl, vt, pp] = datagen(data);
DATAGEN sends back typed measured data filtered using a second order
Butterworth-filter.
% ----------------------------
```

The function *datagen* differentiates the matrix data's components and defines these in the order that they have been stored by the save function in LabView.

The function also makes sure, by using a second order Butterworth low pass filter, to reduce the noise of the data. The filter parameters are created in *datagen* by the function butter, and these parameters are not the same when filtering data of pressures and position, this because they prove to have different noise frequencies.

Finally the filtered cylinder positions are differentiated and speed vectors is defined for lift- respectively tilt cylinders. In the end all value vectors are sent back to the Matlab Workspace.

5.4.6 Power calculations – calc_power

With access to the generated data vectors from the function *datagen*, the power usage and losses during a short duty cycle can now be set using *equation 5.4.7* and *5.4.8*. The program *calc_power*, see Appendix D.4, computes, using for-loops among others, the following effects and sends these back to the Matlab Workspace.

```
PLIFTtot   Power usage for the lift cylinders
PLIFTdp    Power losses between pump and lift cylinders

PTILTtot   Power usage for the tilt cylinder
PTILTdp    Power losses between pump and tilt cylinder

Pltot(N)   Power usage for positive cylinder strokes
Ptot(N)    Power usage for negative cylinder strokes
```

Finally a plot is generated that illustrates the total power usage during a short duty cycle. It is divided in positive and negative cylinder strokes.

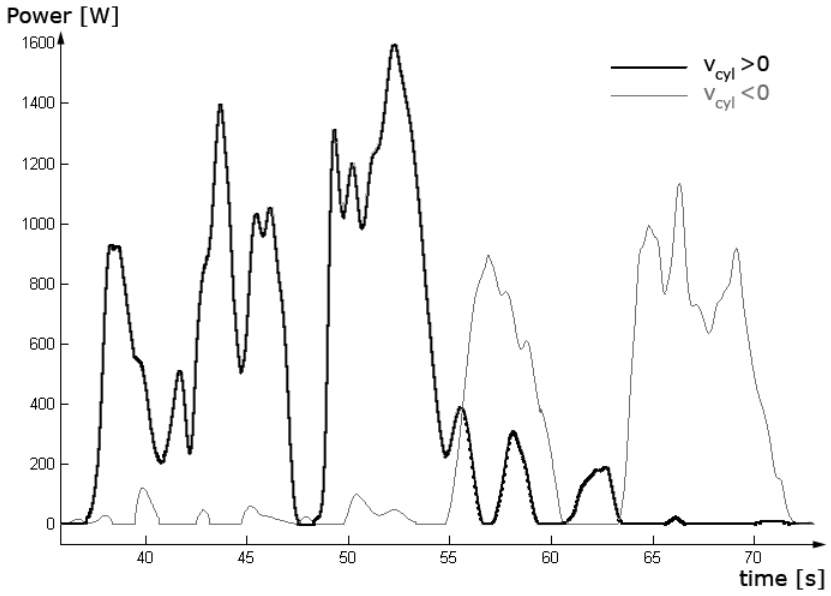


Figure 43: Power consumption during a short duty cycle

In *figure 43* some areas is shown where the position transducers has read movements of the cylinders that is not made by the operator. These are mostly caused by movements from other functions in the system that generates oscillations, which in turn gives rise to movements throughout the system. Therefore these movements are not initiated by any delivered flow from the pump and thus won't be a consumption of power. Because of this, it is sought to remove these non power consuming movements that are generated. A function called *setzero* has been created in Matlab that does this and how to use it is declared in the following help script

```
% ------------------------------------------------------------------
% Write
% [data1 data2 data3] = setzero(data1, data2, data3, t1, t2);
% Where (data1 data2 data3) =(Ptot,   P,   P2  ) if error when v > 0
%                           =(Pltot,  Pl,  Pl2 ) if error when v < 0
% and   [data1 data2 data3] = [PtotN  PN  P2N ]  if error when v > 0
%                           = [PltotN PlN Pl2N]  if error when v < 0
% "This function is internally used by setzero_lcycle"
% ------------------------------------------------------------------
```

Consequently you are demanded to manually find these areas in the short duty cycle and deliver their time spans to the *setzero* clean up function. The reason for this is that it is impossible to mathematically find these areas. The interpretation whether the movement is generated by the operator or if it is a case of self-induced oscillations is impossible to determine in a script. Data over the calls to *setzero* that has been used in the coming calculations has been defined in the m-file *setzero_lcycle* and can be found in appendix D.5. To use it just type *setzero_lcycle* after using the *calc_power*.

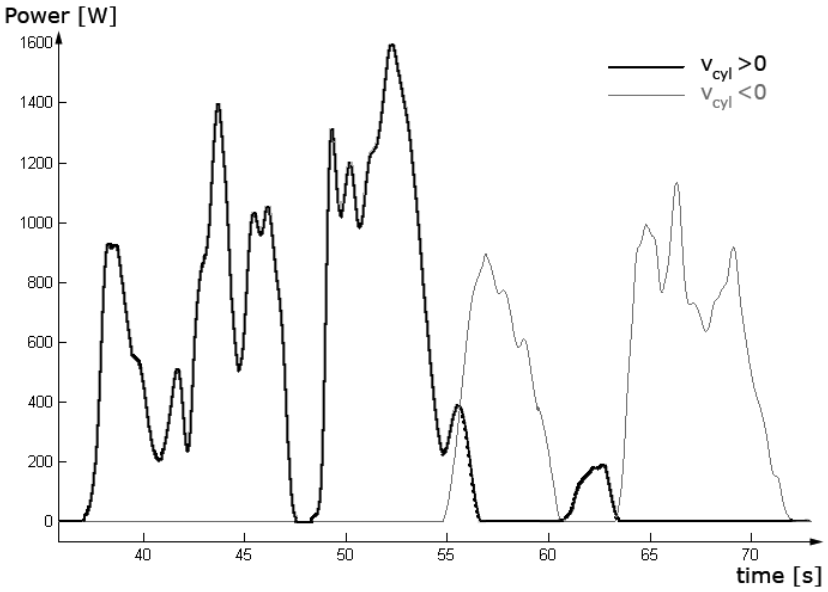


Figure 44: Filtered power consumption

5.4.7 Total energy consumption – calc_energy

The energy calculations are then done in the file *calc_energy*, see appendix D.6, according to the following relations.

$$Q_{tot} = \int_{t_0}^{t_{tot}} P_{tot} \cdot dt \tag{5.4.9}$$

$$Q_{no\ losses} = \int_{t_0}^{t_{tot}} (P_{tot} - P_{loss}) \cdot dt \tag{5.4.10}$$

This script performs calculations on the generated vectors created by the *setzero_lcycle* file.
It also plots the three following figures:

- Figure(1) – Energy consumption for all cylinder strokes where $v > 0$
- Figure(2) – Energy consumption for all cylinder strokes where $v < 0$
- Figure(3) – Total energy consumption

5.4.8 Analysis and results

In this stage of the analysis, the energy calculations are done with the assumption that the SPID function does not work. Calculations that shows the difference between the SPID never working and always working is done later in the chapter. It is very hard to tell if it is the former or the latter that is the correct one. The truth should be somewhere in between.

Energy consumption $v > 0$

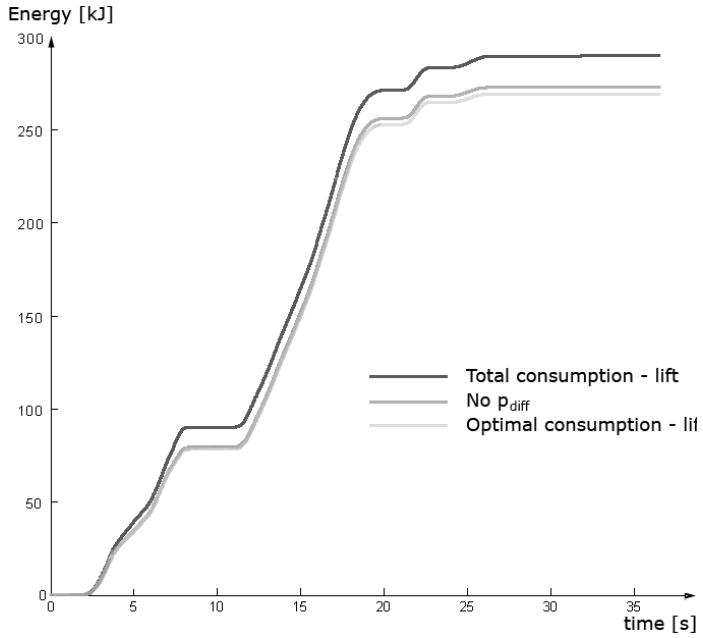


Figure 45: Energy consumptions of today's system,
for positive strokes during a short duty cycle

Figure 45 shows the energy consumption during a short duty cycle where only positive cylinder strokes for tilt and lift are considered. It shows, during this short duty cycle, a hydraulic energy loss of 7.24% where,

- 80% of the losses are due to p_{diff}
- 20% is due to the pressure losses in the hydraulic connectors.

Energy consumption $v < 0$

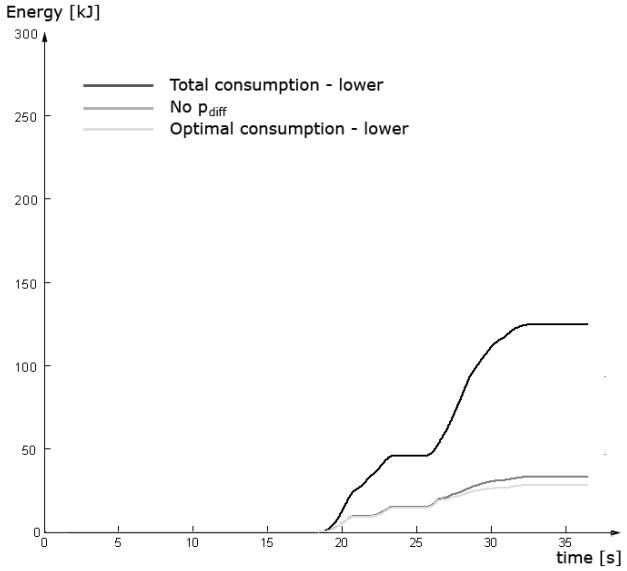


Figure 46: Energy consumptions of today's system, for negative strokes during a short duty cycle

Figure 46 shows in a corresponding way the total energy consumption for negative cylinder strokes. The hydraulic energy losses when looking at this part of the duty cycle adds up to 72.9% where,

- 94% of the losses are due to p_{diff}
- 6% is due to the pressure losses in the hydraulic connectors.

An explanation to the high energy loss can partly be explained by the fact that the SPID function does not work, but also because of the very large pressure drop between the pump and the load.

Total energy consumption

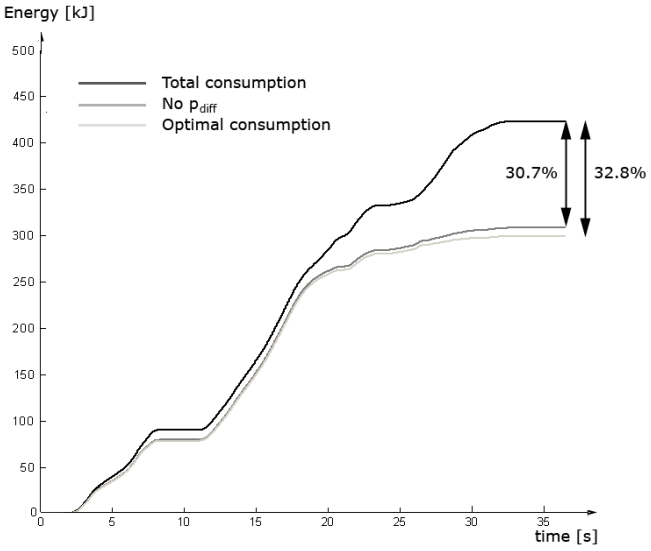


Figure 47: Total energy consumptions of today's system, during a short duty cycle

Figure 47 shows lift- and tilt strokes added together. Now a 32.8% loss of supplied energy between pump and cylinder is shown, where 2.1% is due to losses in the hydraulic connectors, and this is a term that is hard to make more efficient. The 30.7% due to p_{diff} on the other hand is a loss that won't exist in the new hydraulic system. In other words, there is a theoretical energy saving factor of 30.7 %, just because of p_{diff}, in the new system.

This is however not a final value on the energy savings in the new system. More pumps are added which consumes energy, but the new semi-differential lowering mode will on the other hand lead to a recuperation of the energy.

Benefits of using the clean up function

When computing the energy consumption without using the clean up package *setzero_lcycle*, the result came up 2.5% higher than when using it. This shows that the function actually where of great use, and therefore leads to a more reliable result of the total energy consumption.

Energy consumption with/without working SPID function

A calculation of the energy consumption during a short duty cycle, if its assumed that the SPID function always works and get activated when its aught to, is illustrated in *figure 48* and *49* below.

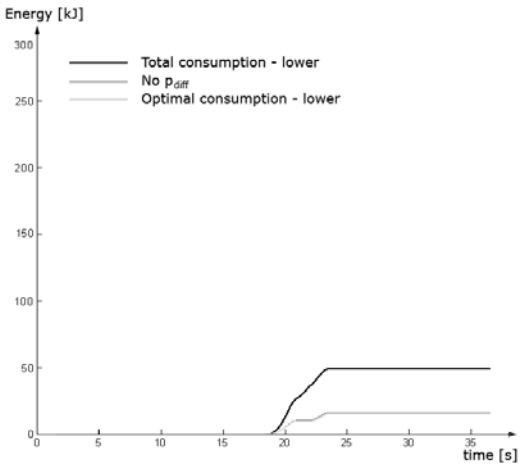


Figure 48: Energy consumption when lowering with working SPID function.

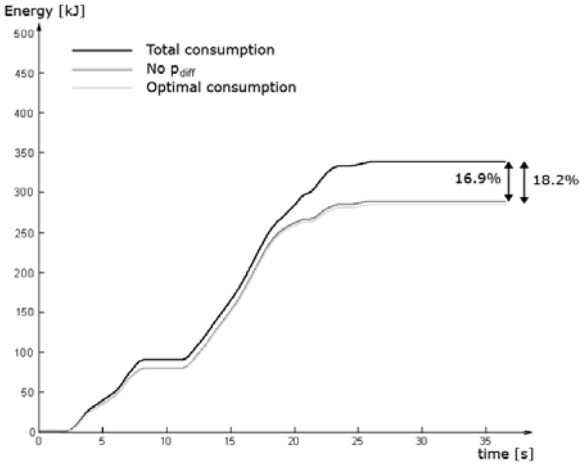


Figure 49: Total energy consumption with working SPID function.

In a correctly driven short duty cycle, a tilt-in occurs two times, one when the bucket is filled with gravel and one when the gravel is emptied in the load receiver. After or during the second tilt-in, a lowering of the lift cylinders follows. If the SPID function works it should during this second lowering be activated. This will reduce the energy consumption when lowering the lift cylinders, as seen in *figure 48*. This leads to a reduction of the total energy consumption of 20% and the losses also decreases to almost half, compare *figure 47* and *figure 49*.

5.5 Valve tests

When lowering a load in the differential mode, a pressure increase in the system will occur. To handle this increase in pressure, another mode will be used, and this mode is called semi-differential. In this mode the valvistor that directs the flow between chambers A and B in a cylinder will be proportionally controlled. By doing this, the pressure in the system will decrease and it will be possible to complete a lowering without damaging the system with a too high pressure.

5.5.1 Objectives

The main objective of this test is to get an insight of how well the valvistor and its controller are able to reduce the pressure in the system.

5.5.2 Method

Because of the fact that there was no wheel loader to perform tests on, a test rig was built in a laboratory environment. Tests on the controller were initially performed on a simple simulation model of the system. The model was based on the following equations:

$$p_A = \frac{A^2 \cdot {A_B}^2 \cdot F_l \cdot K - 2 \cdot A^2 \cdot A_A \cdot A_B \cdot F_l \cdot K + A^2 \cdot {A_B}^2 \cdot F_l \cdot K^2 - {A_B}^3 \cdot {q_p}^2}{A^2 \cdot (A_A - A_B)^3 \cdot K^2} \quad (5.5.1)$$

$$p_B = \frac{A^2 \cdot {A_B}^2 \cdot F_l \cdot K - 2 \cdot A^2 \cdot A_A \cdot A_B \cdot F_l \cdot K + A^2 \cdot {A_B}^2 \cdot F_l \cdot K^2 - A_A \cdot {A_B}^3 \cdot {q_p}^2}{A^2 \cdot (A_A - A_B)^3 \cdot K^2} \quad (5.5.2)$$

Where:

$$K = C_q \cdot \sqrt{\frac{2}{\rho}} \quad (5.5.3)$$

A is the controlled variable orifice that corresponds to the pilot valve.
A_A is the area of the piston and A_B is the area of the piston shaft.
F_l is the load on the cylinder.

5.5.3 Test performance

By successively increasing the system pressure using a LS-pump, a test close to the wheel loader application could be performed.

5.5.4 Analysis and results - Simulation

By using the *equations 5.5.1* and *5.5.2*, a static version of a cylinder and a valve could be created and thereafter be used in a simulation environment. The test that was performed was to show the functionality of the pressure reducer in the valve controller.

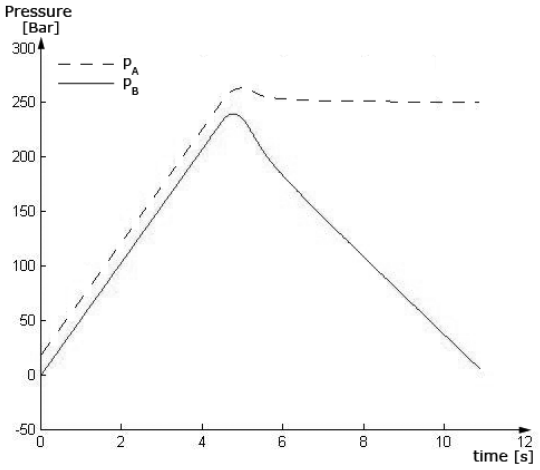


Figure 50: Shows wanted pressure characteristic in a semi-differential lowering

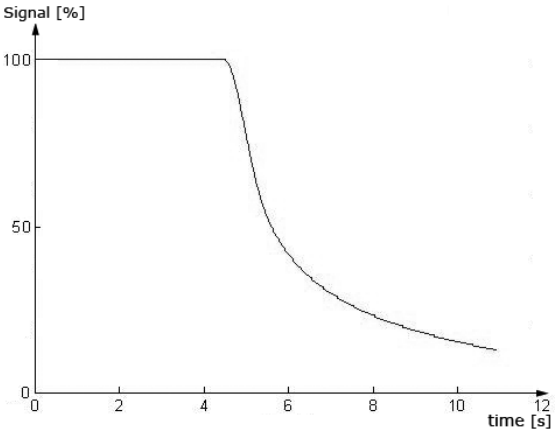


Figure 51: Shows the control signal during the cycle shown in *figure 50*.

Results of the simulation tests are shown in *figure 50* and *51*. *Figure 50* shows that when the pressure in the A chamber exceeds 250 bar the valve starts closing and thereby reducing the pressure in the B chamber. This decrease in pressure will lead to that the total system pressure is reduced. *Figure 51* shows how the control signal to the valve changes to maintain 250 bar in the A chamber. These results are the ones that are wanted in the real wheel loader application.

5.5.5 Analysis and results – EPR Test stand

To test the performance of the valvistor, a test rig that is seen in the picture below was constructed in a laboratory. To minimize sources of errors the valvistor was initially controlled by manually decreasing a variable orifice.

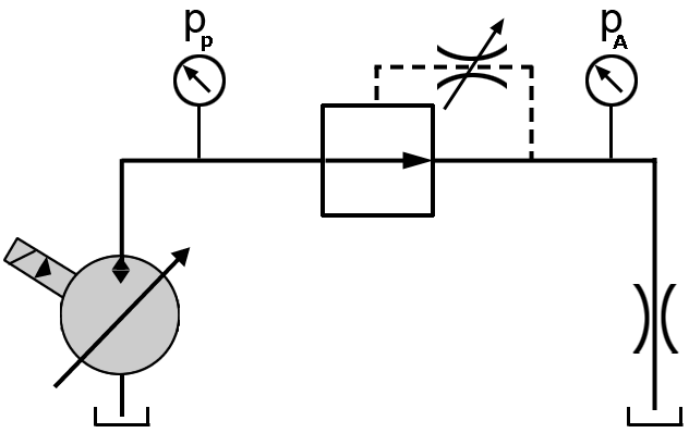


Figure 52: Schematic of the test stand for valvistor testing

When this test showed that the valvistor was able to reduce the pressure, tests with the valvistor and its electrical pilot valve could be performed. Initially the valvistor was manually controlled in LabView by increasing and decreasing the voltage fed to the pilot valve.

When these tests showed a good result, a test with the controller that was developed in the simulation environment could be used. This solution is called EPR, and stands for Electrical Pressure Reducer.

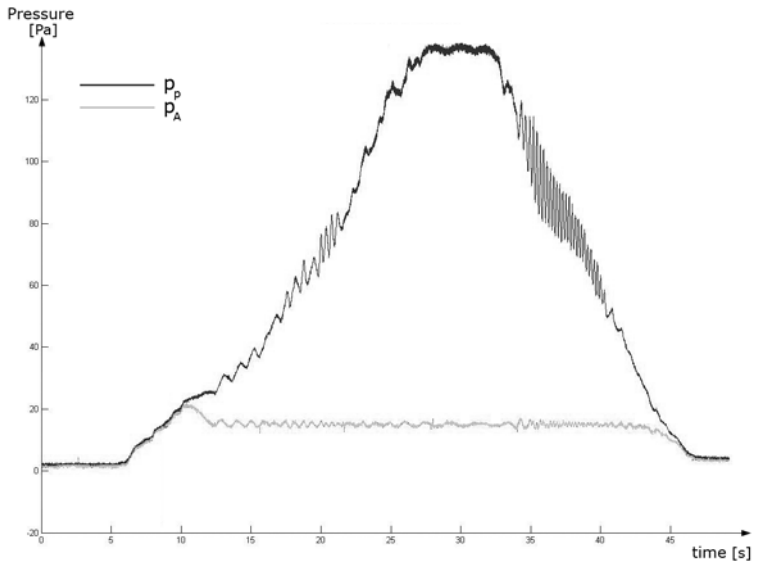


Figure 53: Results from the EPR test

The figure above shows a test where the pressure reducing controller is set to be active when PA exceeds 17 bar. Pp shows that the pump pressure was increased successively over time. This was done to see if the valvistor and the controller were able to maintain a stable pressure in PA despite the pressure drop.

5.5.6 Analysis and results – HPR Test stand

Although the EPR shows very good results, it is not seen as the ultimate solution in a final product. Another way to solve the pressure reducing is to use a HPR. HPR stands for Hydraulic Pressure Reducer and this solution uses a pressure reducing valve instead of the pilot valve. This solution is a solely hydraulic.

A similar test that was done on the EPR solution was also done on the HPR.

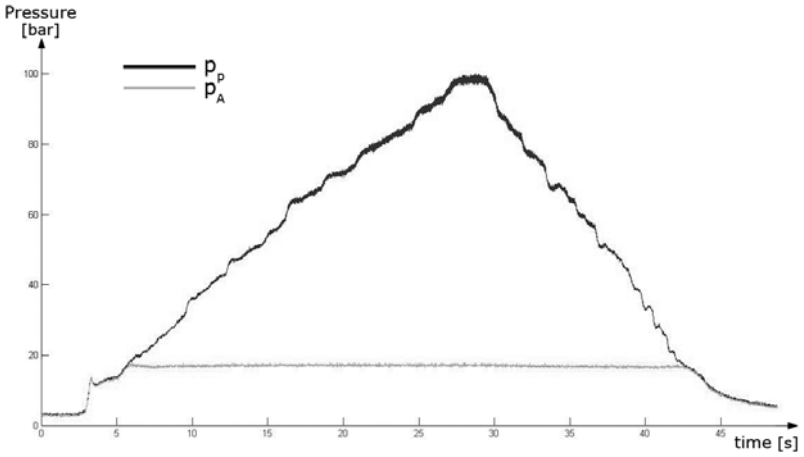


Figure 54: Results from the HPR test

The conclusion of this test is that this solution works very well. This solution is however not used in this platform. The reasons for this are that components for tests with a much higher pressure lacked and that simulation of the HPR installed in the final solution showed a very unstable behaviour. This might be a result of the characteristics of the chosen HPR-valve

5.6 P1 pump test

The P1 pumps use its own built up pressure to control the displacement. In the case of low pressure in the system and a quick response from the pumps are required, an external feeding of pressure to the pumps are needed.

5.6.1 Objective

The objective of this test is to show how much faster response time is acquired when using an externally fed pump. The raise time is calculated as the time it takes from which a step is initiated, until the true relative displacement has reached 63 % of its end value. This value is also of interest (see Appendix F.1 for definitions).
The test is also performed to investigate if there is any leak flow over the cylinder that controls the displacement.

5.6.2 Method

As it is the response time that is examined, a step response is the ideal test. This will also show whether any leak flow exists.

5.6.3 Test performance

The tests has been carried out using the following step pattern (%) of the relative displacement: 5-10 -- 5-25 -- 5-50
This has given us three step responses to compare.

5.6.4 Analysis and results

The response times seems to be trustworthy and acts the same as test that has been carried out in the spring of 2006, that is to say that the response time for increasing relative displacements is longer than when decreasing it. When decreasing the displacement the pump, the response time is independent of the external feed pressure.

The raise times on the other hand aren't very reliable. It seems that the external feed don't affect the raise times at all.

During the tests, the leakage flow from the pump that delivers the external feed to the P1 pump was logged. The data obtained from this test couldn't give any good read offs. The reason for this is that it was such a small flow, and because of that, noise and saturations therefore affected the signal greatly. What could be read was that the maximal leakage flow occurred when using a 150 bar external feed pressure and that the leakage flow was about 0.35 l/min.

Without external feed pressure		**(Appendix F.2):**
Response time:		Positive stroke: **~47ms**
		Negative stroke: **~16 ms**
Raise time	does not occur	(5-10% step in epsilon)
	does not occur	(5-25%)
	not definable	(5-50%)
35 bar external feed pressure		**(Appendix F.2):**
Response time:		Positive stroke: **~24 ms**
		Negative stroke: **~16 ms**
Raise time	63 ms	(5-10% step in epsilon)
	43.5 ms	(5-25%)
	78.4 ms	(5-50%)
70 bar external feed pressure		**(Appendix F.2):**
Response time:		Positive stroke: **~34 ms**
		Negative stroke: **~16 ms**
Raise time	57 ms	(5-10% step in epsilon)
	66.0 ms	(5-25%)
	109.7 ms	(5-50%)
150 bar external feed pressure		**(Appendix F.2):**
Response time:		Positive stroke: **~16 ms**
		Negative stroke: **~16 ms**
Raise time	58.8 ms	(5-10% step in epsilon)
	63.0 ms	(5-25%)
	87.6 ms	(5-50%)

Chapter 6 Results And Discussion

6.1 Theoretical energy consumption reference

Chapter 2.2 and *2.3* shows great differences between today's hydraulic system and the new system. Here can e.g. discern that the new system does not have great pressure losses over the valve package, this because of the new on/off controlled valve package. There will however be a pressure loss when using the semi-differential mode, but since not all of the energy can or will be reused in the system this is a minor problem. The hydraulic losses due to connectors in the new system will on the other hand be close to the same in the new system.

Further on, all the lowering occasions will be done without any flow delivered from the pumps. This will instead be taken from the A-chamber and from here delivered to the B-chamber and the pump for respectively function.

When lowering, the hydraulic oil feeds the P1, which in turn generates a torque to other functions in the wheel loader. Without performing any tests on the pump, it is very hard to describe the efficiency of this recovered energy. And because of the fact that no tests have been done, and that it at this time is impossible to perform any, the efficiency of the recovered energy is set to 75%. This is a speculated value because of the reasons mentioned above and the fact that the P1 is a very unusual type of pump/motor hybrid, which makes it hard to compare to any other hydraulic motors.

In the new system it should be taken in to consideration that two pumps are mounted for the working hydraulics. Theses are each at 75cc compared to the 85cc pump of today's system. Today's pump does not always work at its maximum capacity, but e.g. during a short duty cycle often works at or close to its maximum, this in order to maintain a system pressure of 20 bar over the maximum load pressure. The new pumps will neither they always work at their maximum capacity during a short duty cycle, but just as much so that they deliver the requested flow.

From *figure 44* the total time when a lift and a tilt function are active can be estimated to 45% for both lift and tilt. Further on, the today's 85 cc pump approximately work at 70% of its maximum capacity during this time. The P1's on the other hand is approximated to work at 90% of its maximum when either performing the lift- or the tilt process that it is set to operate. Finally *equation 3.4.17* gives an estimate of the power consumption towards the maximum power take out when idling the P1's of 5%. This figure will accordingly work on the pumps during the 55% of time when they are not active. This idling time will be much longer for the P1's that for the old 85cc pump.
The discussion above leads to the following rough estimation of the energy consumption between the old 85cc pump and the two new 75cc pumps.

$$\frac{Q_{p_1 tot}}{Q_{p_{today} tot}} = \frac{\int_{t_0}^{t_{tot}} 2 \cdot P_{P_1} \cdot dt}{\int_{t_0}^{t_{tot}} P_{p_{todayd}} \cdot dt} = \frac{2 \cdot 75 \cdot (0.45 \cdot 0.9 + 0.55 \cdot 0.05)}{85 \cdot (0.9 \cdot 0.65 + 0.1 \cdot 0.05)} \approx 1.20 \qquad (6.1.1)$$

This shows that the new pumps will consume about 20% more energy during a short duty cycle compared to the old pump. With this included and that all the lowering states occurs in a semi-differential mode with 75% efficiency, and finally that the SPID function is removed in the new system, the following plots has been calculated.

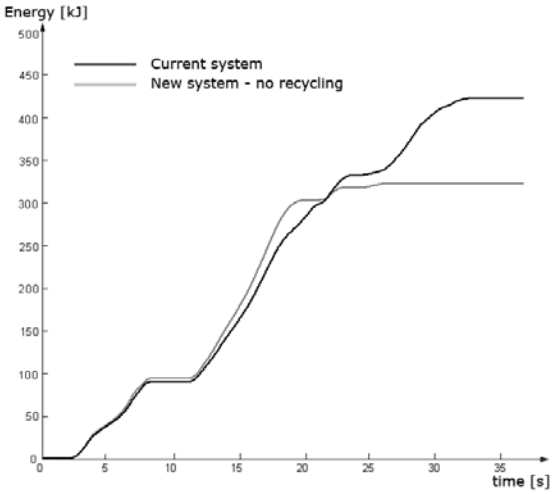


Figure 55: Energy comparison without any recuperation.

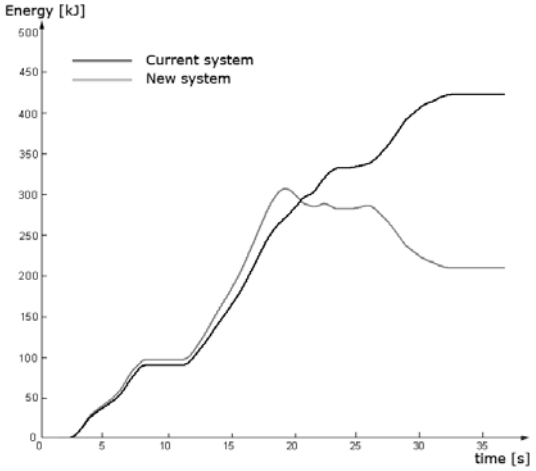


Figure 56: Energy comparison with 75% effective lowering recuperation

Figure 56 shows that the new hydraulic system only consumes about 50% energy compared to today's system. This corresponds to a saving just over 200 kilojoules per cycle. However, on must take into consideration that the recuperated energy in practice is hard to fully make use of. This is partly because of that other functions not always is consuming any energy when the recuperated energy is generated. In these cases the recovered energy will only be used for the rotation of the idling pumps. An accumulator could theoretically be charged in these situations. The accumulator could later be discharged e.g. when raising a load. Implementing an accumulator is however difficult. This because it is impossible to interpret what the operator is interested in doing before the operator signal actually is sent. Therefore the time control of charging and discharging an accumulator is complex. Add to this that an accumulator has got a low grade of efficiency when using incorrect charge/discharge times.

Figure 55 shows the energy consumption for the systems, if it is assumed that no energy is recovered. What can be said is that it is the negative cylinder strokes is what saves the idea of the new hydraulic system, this because of the two pumps that decreases the efficiency. Clear is that negative cylinder strokes always take place without supplying any energy to the hydraulics.

6.2 LabView

Because of its graphical interface, programming a quite complex application in LabView can be done by a person with low programming experience. When building a small program, the system overview is phenomenal. One can follow the wires of each signal with ease. But as the system grows, so does the wiring. When it comes to the point where every line is crossing each other, grasping what the system does is very close to impossible. When it comes to programming more and more complex system, the LabView knowledge level has to be fairly high.

When comparing LabView to other programs where a real-time application can be built, such as Matlab/Dspace and XPCtarget, we found that the learning threshold for programming in LabView is much higher.

6.3 Performance of MAC

The MAC has proven itself in an excellent way when it comes to feeding transducers and transferring signal to and from the CompactRIO. Thanks to its connectors on the front side of the MAC, connecting new transducers or changing channels on old ones becomes very easy. The fact that it contains a wireless router makes it especially useful when it comes to development, this because it enables the developer to monitor the system from a distance. This comes to show during a short duty cycle test when two people don't have to crowd in the small cabin of a wheel loader.

6.4 Performance of CompactRIO

Why is the CompactRIO used in this application? Because of its small enclosure and extreme ruggedness, it was perfectly suited for usage in such a harsh environment as in a wheel loader. The CompactRIO: s versatile way of being able to communicate with surrounding system also made it a perfect candidate.

As the development of the pump controlled system started to escalate, problems arose. An initial problem of using the CompactRIO is the compilation time of the FPGA. If a small change is done in the FPGA, the entire system has to be compiled again. This can take a very long time, depending on the size of the program that is downloaded to the FPGA. As a reference, the compilation time of our system takes about 45 minutes. If changes are often to be made, this will be a very time consuming and annoying process.

A very critical problem with the CompactRIO is its poor ability to have a high sample rate. This puts a lot of restrictions when comes to a good real time capability. Tests that where made during this study, shows that the maximum possible sample rate is limited to about 100 Hz. The sample rate in this system is of course affected by the way the system is implemented in LabView. The system can surely be optimized to give a higher loop rate, but our experience in LabView puts a limit to how much that can be done.

There is however a way to increase the sample rate drastically and that is to do all of the implementation in the FPGA of the CompactRIO. The FPGA has the ability to run in a theoretical sample rate of up to 40 MHz. As the development of a new

system often requires a lot of changes in the program, this will result in a lot of compilations of the FPGA. As mentioned before, this is a very time consuming process. Another problem with implementing a large system on the FPGA is the lack of space that exists here. A better way of more utilizing the space on the FPGA is to use more of the ground components available in LabView. This again requires quite an extended knowledge of LabView and its components.

The CompactRIO sometimes show a very strange behavior, and that is its ability to without any warning lose its connection with the host. This forces everything to shut down. One would think that to start the program again its only one push of the button away. Sometimes it works like charm to just restart the program. But sometimes it requires a reboot of the CompactRIO. This is done by either doing a system reboot in MAX or by turning off the power to the MAC.

National Instruments is the company behind the CompactRIO, and they have performed several tests on the loop rate performance of their own products.

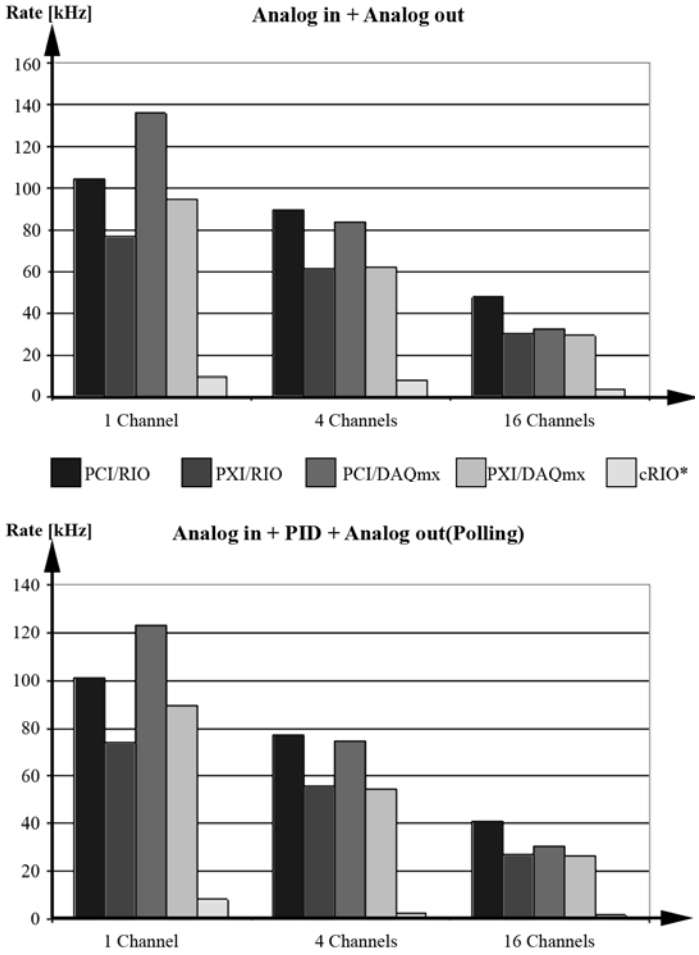


Figure 55: Comparison of loop rates for different NI products [9]

Figure 57 above shows two different cases where measuring and controlling is done. As seen in the pictures, the performance of the CompactRIO is far from impressive when it is compared to its adjacent products. A reason why the difference in loop rate is so great is that the CompactRIO is the only product that is constructed as a computer. The other products are measuring boards that are connected to a PC. These

can use the clock in the computers processor as a sample rate. A computer works at a speed that measures up to GHz and it is quite obvious that the sample rate of these boards will be greater.

6.5 Future

6.5.1 Verifying tests/experiments

When the new hydraulic system is completed, a validation phase will open. This will prove the functionality of both the developed software and hardware, and further also show whether any changes has to be made. After tuning in the new software with the wheel loader, an analysis of the energy consumption is of interest to investigate. The true consumption can then later on be compared with the theoretical.

6.5.2 New developments & improvements

Filter optimization

The low pass filters implemented in LabView has not yet been optimized to the level of noise for the different types of measured signals. This can easily be done by studying the frequencies of the noise and implementing the right cut-off frequency for each filter respectively.

CAN communication

Because of the complex nature in converting frames to channels, it has been a great challenge in making this work. Unfortunately we were not able to get this to fully work on the wheel loader and therefore the problem still exists. The problem in hand is that the flow of messages halts about one second every five seconds. In discussion with people at VCE we came to the conclusion that the problem could be due to the great amount of information that revaluate on the CAN-bus. Hardware filtering of uninteresting messages has been tried without improving the stability of the read CAN signals.

To analyze the reading of the CAN-bus a virtual was developed. Here Dspace was used to simulate the wheel loaders CAN-bus system. Here we implemented some messages which we where able to read without any time lags or other malfunctions. This might be, as discussed above, because of that the virtual bus did not contain as much messages as in the wheel loaders CAN-bus.

Increasing the sample rate

A way to increase the sample rate is to use a more efficient way of dividing the system and to add a faster communication protocol than the shared variables. The shared variables are mainly to be used for supervisory monitoring of system and are not constructed for high speed streaming of data. A way to handle this is to implement the communication in the form of TCP/IP.

Optimize the controllers

The implemented controllers are all verified individually. When connecting these together, in order to operate the working hydraulics of a wheel loader, some tuning must take place. Parameters for the controllers can easily be tuned through the host computer while operating in real time.

Modelling the current power take out
A very interesting and more efficient way of setting the maximum power take out is to compute it in real time. If the P1 pumps somehow could know how much power that is available, they could control their total requested torque by adjusting the relative displaccement. This is in order to maximize the speed and efficiency of the working hydraulics. One way to implement this variable maximum power take out is to supervise the diesel engine, and how much power it generates. With knowledge of this, acceleration-, power train-, idling, cooling- and steering consumptions etc. can be deducted, and in that way see what's left to operate the working hydraulic. The information needed to compute these consumptions can partly be found on the CAN-bus. To achieve this, new messages may have to be developed and therefore more transducers be installed, but it is hard to guess since no work has been done in this area and in this thesis.

Energy recuperation
How much of the potential energy that can be regained and used by other functions is very difficult to predict without performing further tests of the entire new system. That it is possible to recover energy when lowering is in any case obvious, and the true energy consumption in a short duty cycle should for that reason lie somewhere between what's shown in *figure 55* and *figure 56*. Thus a 20 – 50% improvement of the energy consumption for the short duty cycle can be achieved.

Minimize the number of external transducers
Today, the controllers are dependent of two pressure transducers installed at the A- and B-chamber of the lift and tilt cylinders. It might be possible to remove these and only using the internal pressure transducers in the two P1 pumps. This would result in a more robust and safe system. A reason for this is that the risk of broken wires or failure of these transducers would be migrated.

Hydraulically controlled pressure reducers - HPR
If the pressure reducing function could be implemented hydraulically, the system would be both faster and more robust. Today there is however no complete way of solving this. A HPR has been tested and it worked very well for low pressure levels. However, it showed unstable behaviour when the system pressure was increased. A new HPR-valve with more suitable characteristics might solve this problem.

Implementing the software system in the ECU:s
As a step to a final solution, the controllers could be implemented in the ECU of the wheel loader. Due to the fact that all of the controllers are written in C-code, it is therefore possible to implement these without a great amount of effort.

Press down mode
In the developed system there is no function implemented that enables the press down mode. This is however possible to implement and has been done at LiTH in previous pump controlled system studies.

Tilt in over the load force equilibrium position
If the operator tilts in, over the edge of were the acting load force changes from acting on the A-chamber to the B-chamber of the cylinder, then a scenario emerge

that has not yet been considered. If this happens in the new system the pressure will drop in the A-chamber and rapidly increase in the B-chamber. This might result in that the bucket will fall to its end position. A way of solving this is to implement a tilt-in mode. One way of solving this problem could be by using position transducers on the tilt cylinder. Another could be to monitor the pressure in the A chamber of respectively cylinder. By observing a migration in pressures here it might be possible to discern if the cylinder has passed the equilibrium position. This must however be done in a combination with monitoring lift and tilt since both have an effect on the equilibrium position. It is however interesting to find an alternative way, this because the final system aught to rely on a minimal amount of transducers.

Chapter 7 Bibliography

Written

[1] K. Heybroek. *Open circuit solution for pump controlled actuators*, Pages 21-25, 29. Master thesis, LiTH, 2006

[2] Parker Hannifin, *P1/PD IDEC Pump User Guide*.

[3] K-E. Rydberg. *On Performance Optimization and Digital Control of Hydrostatic Drives for Vehicle Applications.* PhD thesis, Linköpings university, 1983. Dissertation No. 99.

[4] Wikipedia Hompage the free encyclopedia. Key word: ???, November 2006. www.wikipedia.org

[5] National Instruments Homepage. Key word: LabView FPGA PWM, *Pulse Width Modulation (PWM) IP Core for LabView FPGA.* www.ni.com

[6] National Instruments Homepage. Key word: NI-CAN installation CD, *NI-CAN Hardware and Software Manual.* www.ni.com

[7] J. Wiklund and J Åkesson. *Smart Cylinder*. Pages 19-24.Master thesis, LiTH, 2006.

[8] Volvo Construction Equipment. *Verkstadshandbok L60E/L70E/L90E 3/3.* February 2005

[9] National Instruments Homepage. Key word: Performance CompactRIO. *Benchmarking Single-Point Performance on National Instruments Real-Time Hardware*. Modified design www.ni.com

Data sheets

[1] *www.iqan.com* model: IQAN-SP500

[2] *www.elfa.se* Art. No: 73-095-60

[3] National Instruments Homepage. Key word: 9205, *NI 9205: Analog Input Module.* www.ni.com

[4] National Instruments Homepage. Key word: 9263, *NI 9263: Analog Output Module.* www.ni.com

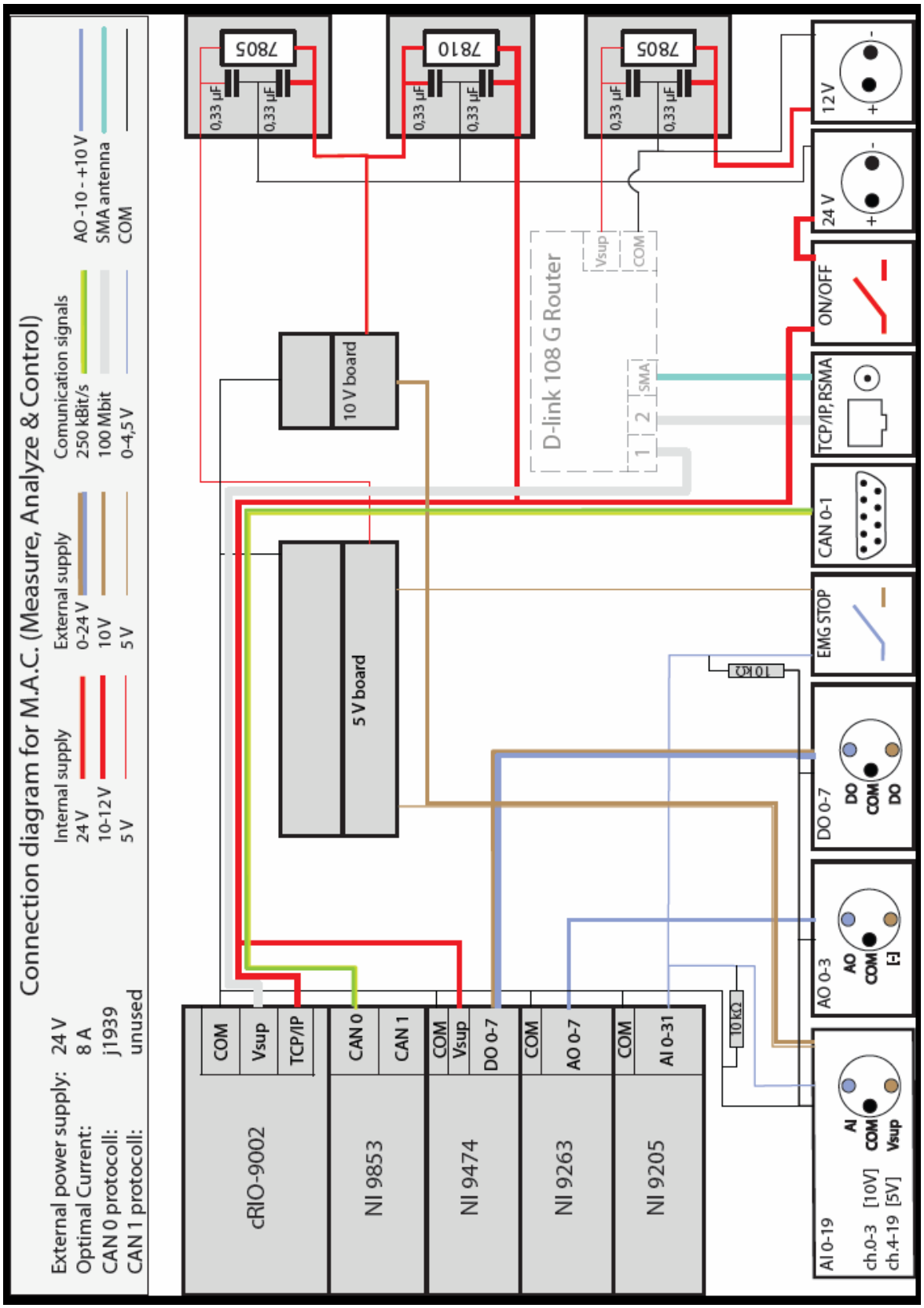
[5] National Instruments Homepage. Key word: 9474, *NI cRIO-9474: Sourcing Digital Output Module.* www.ni.com

[6] National Instruments Homepage. Key word: 9853, *NI 9853 for CompactRIO: High-Speed Controller Area Network (CAN) Module.* www.ni.com

Oral

[10] K-E. Rydberg. Professor at Linköpings University, Division of Fluid and Mechanical Engineering Systems

A – Connection diagram



B – Code Conversion Manual

As the code is written in Visual C++, the conversion of code to dll is described in this program.

- Select new project, Win 32 Dynamic-Link Library and name the project.
- Select new files and add a source file to the project. This is the file where all the code is written.
- To be able to use the dll:s in LabView the following files has to be included in the project folder: *extcode.h, fundtypes.h, platdefines.h.*
- Right click on the project in the file view and select settings. Go to the tab C++ and select code generation under category. Under "Use run-time library" select debug multithreaded dll. Also under "Struct member alignment" select 1 Byte.
- When building a dll to use in LabView, the source file has to have the right format. This format is a c-file and not a cpp-file as is generated when creating a source file. This is however easily corrected. Go to the explorer and rename the file as a .c-file. In the project tree simply replace the cpp-file with the newly created c-file.
- Press build to build the application as a dll.
- See Appendix C on how to start writing the code.

To use dll:s in LabView can be quite troublesome, especially if they are to be used together with the CompactRIO.

A QUICK MANUAL TO USING DLL-FILES TOGETHER WITH COMPACTRIO

- When dll is obtained use ftp and put the dll in a folder on the CompactRIO. The folder lies in the root of the RIO, C:\ that is.
- The filed placed on the CompactRIO has to be placed in a folder with the same name on the host computer (where the application is being built). The folder has to lie in the root, C:\. It is to this file that the specification of the dll in LabView is done.
- This is done so that when the program is transferred to the RIO, the filepath to the dll is the same.

C – Implemented Controllers

C.1– Valve Controller

```
// Include extcode to handle booleans in LabView
#include "extcode.h"
// Include math for fabs function
#include<math.h>
_declspec(dllexport) void valve_controller (double eps_lift,
                                            double eps_tilt,
                                            double period,
                                            double PA_lift,
                                            double PA_tilt,
                                            LVBoolean *Em_stop,
                                            double PA_press_lift,
                                            double PA_press_tilt,
                                            double zero_lift,
                                            double zero_tilt,
                                            double pdtime_lift,
                                            double pdtime_tilt,
                                            double deadband_lift,
                                            double deadband_tilt,
                                            double Gain_lift,
                                            double Gain_tilt,
                                            double lowlimit_lift,
                                            double highlimit_lift,
                                            double lowlimit_tilt,
                                            double highlimit_tilt,
                                            double *AP_lift,
                                            double *AT_lift,
                                            double *BP_lift,
                                            double *BT_lift,
                                            double *AP_tilt,
                                            double *AT_tilt,
                                            double *BP_tilt,
                                            double *BT_tilt);
// Internal states
double zerotimer_lift=0.0;
double zerotimer_tilt=0.0;
double pdtimer_lift=0.0;
double pdtimer_tilt=0.0;
double lift_mode;
double tilt_mode;
double BP_lift_max=0.000055;
double BP_tilt_max=0.000055;
double BP_lift_temp=0.000055;
double BP_tilt_temp=0.000055;
```

```
_declspec(dllexport) void valve_controller (double eps_lift,
                                            double eps_tilt,
                                            double period,
                                            double PA_lift,
                                            double PA_tilt,
                                            LVBoolean *Em_stop,
                                            double PA_press_lift,
                                            double PA_press_tilt,
                                            double zero_lift,
                                            double zero_tilt,
                                            double pdtime_lift,
                                            double pdtime_tilt,
                                            double deadband_lift,
                                            double deadband_tilt,
                                            double Gain_lift,
                                            double Gain_tilt,
                                            double lowlimit_lift,
                                            double highlimit_lift,
                                            double lowlimit_tilt,
                                            double highlimit_tilt,
                                            double *AP_lift,
                                            double *AT_lift,
                                            double *BP_lift,
                                            double *BT_lift,
                                            double *AP_tilt,
                                            double *AT_tilt,
                                            double *BP_tilt,
                                            double *BT_tilt)
{
//---------------------------------------------/
//--------------------LIFT--------------------/
//---------------------------------------------/
          {
          if (eps_lift==0)
                    zerotimer_lift +=period/1000;
          else
                    {zerotimer_lift=0.0;}
          if (PA_lift<PA_press_lift)
                    {pdtimer_lift +=period/1000;}
          else
                    {pdtimer_lift=0;}
          if (*Em_stop==1)                          //STOP
                    {lift_mode=0;}
          else if (eps_lift>0)                      // Lift
                    {lift_mode=1;}
          else if (eps_lift<0)
          {
          if (pdtimer_lift>pdtime_lift)    //Press Down
                    {lift_mode=2;}
          else                             //Lower semi-diff
```

```
                    {lift_mode=3;}
          }
          else if (zerotimer_lift>zero_lift)
                    {lift_mode=0;}
          if (lift_mode==0)
                    {*AP_lift=0; *AT_lift=0; *BP_lift=0;
                    *BT_lift=0;}
          else if (lift_mode==1)
                    {*AP_lift=1; *AT_lift=0; *BP_lift=0;
                    *BT_lift=1;}
          else if (lift_mode==2)
                    {*AP_lift=0; *AT_lift=1; *BP_lift=1;
                    *BT_lift=0;}
          else if (lift_mode==3)
                    {*AP_lift=1;
                     *AT_lift=0;
                     *BT_lift=0;
                     *BP_lift=BP_lift_temp/BP_lift_max;
          if (PA_lift>lowlimit_lift)
                    {
                    BP_lift_temp=(BP_lift_temp-((PA_lift-
                    lowlimit_lift)/(highlimit_lift-
                    lowlimit_lift)*highlimit_lift)*Gain_lift);
                    }
          else if (PA_tilt<lowlimit_lift &&
                    BP_lift_temp<BP_lift_max)
                    BP_lift_temp=(BP_lift_temp-((PA_lift-
                    lowlimit_lift)/(highlimit_lift-
                    lowlimit_lift)*highlimit_lift)*Gain_lift);
          if (BP_lift_temp<0)
                    BP_lift_temp=0;
                    }
          }
//-----------------------------------------/
//---------------------TILT--------------------/
//-----------------------------------------/
          {
          if (eps_tilt==0)
                    zerotimer_tilt +=period/1000;
          else
                    {zerotimer_tilt=0.0;}
          if (PA_tilt<PA_press_tilt)
                    {pdtimer_tilt +=period/1000;}
          else
                    {pdtimer_tilt=0;}
          if (*Em_stop==1)                          //STOP
                    {tilt_mode=0;}
          else if (eps_tilt>0)                      // tilt
                    {tilt_mode=1;}
```

```
            else if (eps_tilt<0)
            {
            if (pdtimer_tilt>pdtime_tilt)    //Press Down
                      {tilt_mode=2;}
            else                    //Lower semi-diff
                      {tilt_mode=3;}
            }
            else if (zerotimer_tilt>zero_tilt)
                      {tilt_mode=0;}
            if (tilt_mode==0)
                      {*AP_tilt=0; *AT_tilt=0; *BP_tilt=0;
                      *BT_tilt=0;}
            else if (tilt_mode==1)
                      {*AP_tilt=1; *AT_tilt=0; *BP_tilt=0;
                      *BT_tilt=1;}
            else if (tilt_mode==2)
                      {*AP_tilt=0; *AT_tilt=1; *BP_tilt=1;
                      *BT_tilt=0;}
            else if (tilt_mode==3)
                      {*AP_tilt=1;
                       *AT_tilt=0;
                       *BT_tilt=0;
                       *BP_tilt=BP_tilt_temp/BP_tilt_max;
            if (PA_tilt>lowlimit_lift)
                      {
                      BP_tilt_temp=(BP_tilt_temp-((PA_tilt-
                      lowlimit_lift)/(highlimit_lift-
                      lowlimit_lift)*highlimit_lift)*Gain_tilt);
                      }
            else if (PA_tilt<lowlimit_lift &&
                      BP_tilt_temp<BP_tilt_max)
                      BP_tilt_temp=(BP_tilt_temp-((PA_tilt-
                      lowlimit_lift)/(highlimit_lift-
                      lowlimit_lift)*highlimit_lift)*Gain_tilt);
            if (BP_tilt_temp<0)
                      BP_tilt_temp=0;
                      }
            }
}
```

C.2 – Pump Logics

```
// Include extcode to handle booleans in LabView
#include "extcode.h"
// Include math for fabs function
#include<math.h>
//#include<cmath>
_declspec(dllexport) void pump_logics      (double U_joy_tilt,
                                            double U_joy_lift,
                                            double U_dead_band,
                                            double n_p,
                                            double etha_volp_tilt,
                                            double etha_volp_lift,
                                            double eps_scale_tilt,
                                            double eps_scale_lift,
                                            double *eps_t_op,
                                            double *eps_l_op,
                                            double *U_error,
                                    double *pump_limit_reached);
// Internal states
double A_t_a=0.017671458676443;  // A-chamber area in tilt cylinder
double A_t_b=0.012644910430699;  // B-chamber area in tilt cylinder
double A_l_a=0.009503317777109;  // A-chamber area in lift cylinder
double A_l_b=0.005654866776462;  // B-chamber area in lift cylinder
double U_min=0.5;                // Minimum voltage from transducer
double U_max=4.5;                // Maximum voltage from transducer
double v_raise_max = 0.1;  // Maximum speed when raising a cylinder
double v_lower_max = 0.1;  // Maximum speed when lowering a cylinder
double n_p=0;                    // Pump speed
double eps_t_op = 0;             // Relative displacement [-]
double eps_l_op = 0;             // Relative displacement [-]
double Dp = 60e-6;               // displacement [m^3/rev]

_declspec(dllexport) void pump_logics      (double U_joy_tilt,
                                            double U_joy_lift,
                                            double U_dead_band,
                                            double n_p,
                                            double etha_volp_tilt,
                                            double etha_volp_lift,
                                            double eps_scale_tilt,
                                            double eps_scale_lift,
                                            double *eps_t_op,
                                            double *eps_l_op,
                                            double *U_error,
                                    double *pump_limit_reached)
{
*pump_limit_reached=0;           // 0 equals no error
*U_error = 0;
{
          if (U_joy_tilt < U_min)
                    {*U_error = 1;}
                    // 1 equals loss of Joystick - tilt
          else if (U_joy_lift < U_min)
                    {*U_error = 2;}
                    // 2 equals loss of Joystick - lift
}
// Interpret the voltage and translate it to a valid eps
{
          if (U_joy_lift> 2.5 + U_dead_band)
                    {*eps_l_op =
                    eps_scale_lift*2*A_l_a*v_raise_max*(U_joy_lift-
```

```
                    U_min)/(Dp*n_p*etha_volp_lift*(U_max-U_min));
                    // Two cylinders
                    {
          if  (*eps_l_op > 1)
                    {*eps_l_op = 1;
                    *pump_limit_reached=1;}
          else
                    {*eps_l_op=*eps_l_op;}
                    }
                    }
          else if (U_joy_lift < 2.5 - U_dead_band)
                    {*eps_l_op = -eps_scale_lift*2*(A_l_a-
                    A_l_b)*v_lower_max*(U_joy_lift-
                    U_min)/(Dp*n_p*etha_volp_lift*(U_max-U_min));
                    {
          if (*eps_l_op < -1)
                    {*eps_l_op = -1;
                    *pump_limit_reached=1;}
          else
                    {*eps_l_op=*eps_l_op;}
                    }
                    }
          else
                    {*eps_l_op = 0;}
          }
          {
          if (U_joy_tilt > 2.5 + U_dead_band)
                    {*eps_t_op =
                    eps_scale_tilt*A_t_a*v_raise_max*(U_joy_tilt-
                    U_min)/(Dp*n_p*etha_volp_tilt*(U_max-U_min));
                    // Two cylinders
          {
          if  (*eps_t_op > 1)
                    {*eps_t_op = 1;
                    *pump_limit_reached=1;}
          else
                    {*eps_t_op=*eps_t_op;}
          }
          }
          else if (U_joy_tilt < 2.5 - U_dead_band)
                    {*eps_t_op = -eps_scale_tilt*(A_t_a-
                    A_t_b)*v_lower_max*(U_joy_tilt-
                    U_min)/(Dp*n_p*etha_volp_tilt*(U_max-U_min));
          {
          if (*eps_t_op < -1)
                    {*eps_t_op = -1;
                    *pump_limit_reached=1;}
          else
                    {*eps_t_op=*eps_t_op;}
                              }
                    }
          else
                    {*eps_t_op = 0;}
          }
}
```

C.3 – Minimum Pressure

```
// Include extcode to handle booleans in LabView
#include "extcode.h"
// Include math for fabs function
#include<math.h>
_declspec(dllexport) void pmin   (double pp_t,
                                  double pp_l,
                                  double eps_t_op,
                                  double eps_l_op,
                                  double *eps_t_op_preg,
                                  double *eps_l_op_preg,
                                  double G_l,
                                  double G_h,
                                  double pmin,
                                  double pmax,
                                  double delta);
                                  double deps;
                                  double deps1;
                                  double deps2;
                                  double deps3;

_declspec(dllexport) void pmin  (double pp_t,
                                  double pp_l,
                                  double eps_t_op,
                                  double eps_l_op,
                                  double *eps_t_op_preg,
                                  double *eps_l_op_preg,
                                  double G_l,
                                  double G_h,
                                  double pmin,
                                  double pmax,
                                  double delta)
{
// Define the change factor of eps.
// deps  = eps factor for tilt
// deps1 = eps factor for tilt when pressure is regained
// deps2 = eps factor for lift
// deps3 = eps factor for lift when pressure is regained
deps=(-((pp_t-pmin)/(pmax-pmin)*pmax)*G_l);
deps1=(-((pp_t-pmin)/(pmax-pmin)*pmax)*G_h);
deps2=(-((pp_l-pmin)/(pmax-pmin)*pmax)*G_l);
deps3=(-((pp_l-pmin)/(pmax-pmin)*pmax)*G_h);
           {
           if (pp_t<pmin)
                      {*eps_t_op_preg=eps_t_op+deps;}
           else if (pp_t>=pmin && (*eps_t_op_preg-eps_t_op>delta))
                      {*eps_t_op_preg=eps_t_op+deps1;}
           else
                      {*eps_t_op_preg=eps_t_op;}
           }
           {
           if (pp_l<pmin)
                      {*eps_l_op_preg=eps_l_op+deps2;}
           else if (pp_l>=pmin && (*eps_l_op_preg-eps_l_op>delta))
                      {*eps_l_op_preg=eps_l_op+deps3;}
           else
           {*eps_l_op_preg=eps_l_op;}
           }
}
```

C.4 – Maximum Power Take Out

```
// Include extcode to handle booleans in LabView
#include "extcode.h"
// Include math for fabs function
#include<math.h>
//#include<cmath>

_declspec(dllexport) void P_controller      (double P_max,
                                             double n_p,
                                             double pp_tilt,
                                             double pp_lift,
                                             double Em_stop,
                                             double *T_eps0_tilt,
                                             double *T_eps0_lift,
                                             double *etha_hm_tilt,
                                             double *etha_hm_lift,
                                             double *eps_tilt,
                                             double *eps_lift,
                                             double *P_ref,
                                             double *error_msg);

// Internal states
double b0P = 8.24e-3;
double b1P = 8.67e-3;
double b2P = 0.0161;
double b3P = 0.125;
double b4P = 0.0559;
double b5P = 0.0609;
double b6P = 5.67e-8;
double my  = 0.0087;
// Calculated for 10 cSt and a density of 870 kg/m^3
double delta_p = 24.0;
double pi  = 3.141592653589793;
double k_eps0 = 0.35;
double m_eps0 = 5.0;
double pL = 3.5;
double P_tilt;
double P_lift;
double pL_tilt;
double pL_lift;
double Dp = 0.00006;

_declspec(dllexport) void P_controller      (double P_max,
                                             double n_p,
                                             double pp_tilt,
                                             double pp_lift,
                                             double Em_stop,
                                             double *T_eps0_tilt,
                                             double *T_eps0_lift,
                                             double *etha_hm_tilt,
                                             double *etha_hm_lift,
                                             double *eps_tilt,
                                             double *eps_lift,
                                             double *P_ref,
                                             double *error_msg)
{

// Internal variables
           double dp_tilt = pp_tilt - pL*1e5;
           double dp_lift = pp_lift - pL*1e5;
           double pH_tilt = pp_tilt;
```

```
            double pH_lift = pp_lift;

// External feed pressure equals the largest of pL amd pp
            {
                        if (pp_tilt > pL*1e5)
                                    {pL_tilt=pp_tilt;}
                        else
                                    {pL_tilt=pL*1e5;}
            }
            {
                        if (pp_lift > pL*1e5)
                                    {pL_lift=pp_lift;}
                        else
                                    {pL_lift=pL*1e5;}
            }
// Define the hydraul mechanical efficiency
            {
                        if (*eps_tilt > 0.0)
                                    {*etha_hm_tilt =
                                    1/(1+b0P/(*eps_tilt)+b1P+(b2P/(*eps_
                                    tilt)+b3P)*pL/dp_tilt+b4P*fabs(pH_ti
                                    lt+delta_p*pL)/(*eps_tilt*dp_tilt)+b
                                    5P*2*pi*my*n_p/(*eps_tilt*dp_tilt)+b
                                    6P**eps_tilt**eps_tilt*n_p*n_p/dp_ti
                                    lt);
                                    *T_eps0_tilt = 0;}
                        else if (*eps_tilt == 0.0)
                                    {*etha_hm_tilt=0.0;
                                    *T_eps0_tilt =
                                    k_eps0*dp_tilt/1e6+m_eps0;}
                        else if (*eps_tilt < 0.0)
                                    {*etha_hm_tilt = 1/(1+b0P/-
                                    *eps_tilt+b1P+(b2P/-
                                    *eps_tilt+b3P)*pL/dp_tilt+b4P*fabs(p
                                    H_tilt+delta_p*pL)/(-
                                    *eps_tilt*dp_tilt)+b5P*2*pi*my*n_p/(
                                    -*eps_tilt*dp_tilt)+b6P*-*eps_tilt*-
                                    *eps_tilt*n_p*n_p/dp_tilt);
                                    *T_eps0_tilt = 0;}
            }
            {
                        if (*eps_lift > 0.0)
                                    {*etha_hm_lift =
                                    1/(1+b0P/(*eps_lift)+b1P+(b2P/(*eps_
                                    lift)+b3P)*pL/dp_lift+b4P*fabs(pH_li
                                    ft+delta_p*pL)/(*eps_lift*dp_lift)+b
                                    5P*2*pi*my*n_p/(*eps_lift*dp_lift)+b
                                    6P**eps_lift**eps_lift*n_p*n_p/dp_li
                                    ft);
                                    *T_eps0_lift = 0.0;}
                        else if (*eps_lift == 0.0)
                                    {*etha_hm_lift=0.0;
                                    *T_eps0_lift =
                                    k_eps0*dp_lift/1e6+m_eps0;}
                        else if (*eps_lift < 0.0)
                                    {*etha_hm_lift = 1/(1+b0P/-
                                    *eps_lift+b1P+(b2P/-
                                    *eps_lift+b3P)*pL/dp_lift+b4P*fabs(p
                                    H_lift+delta_p*pL)/(-
                                    *eps_lift*dp_lift)+b5P*2*pi*my*n_p/(
```

```
                                -*eps_lift*dp_lift)+b6P*-*eps_lift*-
                                *eps_lift*n_p*n_p/dp_lift);
                                *T_eps0_lift = 0;}
          }
// Make sure the computed values are within correct boundaries
          {
                    if (*etha_hm_tilt > 1 )
                                {*etha_hm_tilt = 1;}
                    else if (*etha_hm_tilt < 0)
                                {*etha_hm_tilt = 0;}
          }

          {
                    if (*etha_hm_lift > 1 )
                                {*etha_hm_lift = 1;}
                    else if (*etha_hm_lift < 0)
                                {*etha_hm_lift = 0;}
          }
// Set initial error to 0 (no error)
          *error_msg = 0;
// Define requested torque
          {
                    if (*eps_tilt > 0 )
                                {P_tilt=*eps_tilt*pp_tilt*n_p*Dp/(*e
                                tha_hm_tilt);}
                    else if (*eps_tilt < 0 )
                                {P_tilt=*eps_tilt*pp_tilt*n_p*Dp*(*e
                                tha_hm_tilt);}
                    else
                                {P_tilt = 0;}
          }
          {
                    if (*eps_lift > 0 )
                                {P_lift=*eps_lift*pp_lift*n_p*Dp/(*e
                                tha_hm_lift);}
                    else if (*eps_lift < 0)
                                {P_lift=*eps_lift*pp_lift*n_p*Dp*(*e
                                tha_hm_lift);}
                    else
                                {P_lift = 0;}
          }
// Check if limitations of eps due to Pmax is needed
          {
          if (P_tilt + P_lift <= P_max-
          (*T_eps0_tilt+*T_eps0_lift)*n_p)
                    {*eps_tilt = *eps_tilt;
                    *eps_lift = *eps_lift;}
          else if (P_tilt > P_max-(*T_eps0_tilt+*T_eps0_lift)*n_p)
// Higher priority of the tilt
                    {*eps_tilt=(P_max-
                    n_p*(*T_eps0_tilt+*T_eps0_lift))*(*etha_hm_tilt
                    )/(pp_tilt*n_p*Dp);
                    *eps_lift=0;}
          else if (P_tilt <= P_max-(*T_eps0_tilt+*T_eps0_lift)*n_p)
// Lift-cyl get whats left of the flow
                    {*eps_tilt=*eps_tilt;
                    *eps_lift=((P_max-
                    n_p*(*T_eps0_tilt+*T_eps0_lift))/(n_p*Dp)-
                    *eps_tilt*pp_tilt/(*etha_hm_tilt))*(*etha_hm_li
                    ft)/pp_lift;}
```

```
    }
           {
           if (*eps_tilt == 0 && *eps_lift == 0)
                      {*P_ref=n_p*(*T_eps0_tilt+*T_eps0_lift);}
           else if (*eps_tilt == 0)
                      {*P_ref=(*eps_lift*pp_lift/(*etha_hm_lift))*n_p
                      *Dp+n_p*(*T_eps0_tilt+*T_eps0_lift);}
           else if (*eps_lift == 0)
                      {*P_ref=(*eps_tilt*pp_tilt/(*etha_hm_tilt))*n_p
                      *Dp+n_p*(*T_eps0_tilt+*T_eps0_lift);}
           else
                      {*P_ref=(*eps_tilt*pp_tilt/(*etha_hm_tilt)+*eps
                      _lift*pp_lift/(*etha_hm_lift))*n_p*Dp+n_p*(*T_e
                      ps0_tilt+*T_eps0_lift);}
           }
// Security and boundary verification
    {
// If Emerg. stop is activated, set the relative displacement to 0.
           if (Em_stop == 1)
                      {*eps_lift = 0;
                      *eps_tilt = 0;}
// If CANmsg is wrong, halt to avoid divide by zero
           else if (n_p < (500/60))
                      {*eps_lift = 0;
                      *eps_tilt = 0;
                      *error_msg=5;}
           else
                      {*eps_lift = *eps_lift;
                      *eps_tilt = *eps_tilt;}
    }
// If calculated rel. disp. is > 1 set it to 1
    {
           if (*eps_lift > 1)
                       {*eps_lift = 1;}
           else if (*eps_lift < -1)
                      {*eps_lift = -1;}
           else
                      {*eps_lift = *eps_lift;}
    }
// If calculated rel. disp. is > 1 set it to 1
    {
           if (*eps_tilt > 1)
                      {*eps_tilt = 1;}
           else if (*eps_tilt < -1)
                      {*eps_tilt = -1;}
           else
                      {*eps_tilt = *eps_tilt;}
    }
}
```

D – Matlab M-script

D.1 – Nonlinear Least Mean Square

```
function f = fitfun3(x,data)

f=sum((x(1)*exp(x(2)*abs(data(:,3)))+x(3)*(data(:,1)-
data(:,2))+x(4)*data(:,2)+x(5)*data(:,3)-data(:,4)).^2);

% _____________________________________________________________
% Use:  The program computes the constants x(1:5) so that f get
minimized.
% Type: Nonlinear unconstrained least square method
%
% Command Window commands:
% options.MaxIter=2000
% options.MaxFunEvals=2000
% options = optimset('GradObj','off')
% x = fminunc(@(x) fitfun3(x,data),[zeros(1,5)], options)
% _____________________________________________________________


% Result tilt
% x =

%   6.7439e+002
%   2.0615e+001
%   3.0753e+001
%   8.1541e+001
%  -1.7957e+004

% Result lift
% x =

%   1.1912e+003
%   2.0815e+001
%   2.3919e+001
%   2.1516e+001
%  -5.8221e+004

% Plot the result of the calculated lift constants
x = fminunc(@(x) fitfun3(x,data),zeros(5,1), options)
w=0:0.0001:0.1;
plot(w, x(1)*exp(x(2)*abs(w))+x(3)*(50-4)+x(4)*4+x(5)*w)
hold
plot(data(1:6,3),data(1:6,4),'o')
title('Test');
ylabel('Ff [N]');
xlabel('v [m/s]');
legend('F(v,p1,p2)', 'measured')
```

D.2 – Resulting Excel Document (Lift)

FRICTION _ LIFT

Constants:	dla	dlb	dta	dtb	Ala	Alb	Ata	Atb
	0,11	0,07	0,15	0,08	9,50E-03	3,85E-03	1,77E-02	5,03E-03

Punkt	Testfil	t1	t2	v1 [m/s]	v2 [m/s]	pa1 [bar]	pb1 [bar]	pa2 [bar]	pb2 [bar]	pa1*Ala-pb1*Alb	pa2*Ala-pb2*Alb	Ff [N]	v [m/s]
	1	2049	2505	0,006832	0,004901929	50,153	3,232	44,682	0,798	0,46	0,42	2131,276	0,005867
	2	1461	2085	0,030831	3,10E-02	51,976	6,275	45,898	1,407	0,47	0,43	1951,345	0,030916
	3	2437	2810	0,042919	4,30E-02	50,153	3,841	45,898	0,798	0,46	0,43	1436,289	0,042943
	4	2644	3165	0,051640	4,92E-02	48,937	0,19	54,407	4,449	0,46	0,50	1779,630	0,050434
	5	513	2221	0,072011	6,92E-02	45,29	2,015	51,368	3,841	0,42	0,47	2536,695	0,070604
	6	845	1280	0,084581	7,94E-02	53,8	4,449	46,506	2,015	0,49	0,43	2997,504	0,082009
	7	1430	2588	0,005218	5,95E-03	27,054	2,015	23,407	1,407	0,25	0,22	1615,937	0,005586
	8	433	1564	0,026588	2,73E-02	28,9	5,1	26,4	6,9	0,25	0,22	1506,299	0,026968
	9	882	1381	0,045982	4,61E-02	27,7	4,4	25,8	5,7	0,25	0,22	1099,828	0,046021
	10	1920	2369	0,059018	5,75E-02	29,5	4,4	26,4	4,4	0,26	0,23	1444,029	0,058246
	11	836	1135	0,087273	8,24E-02	31,3	2,6	27,7	6,3	0,29	0,24	2435,465	0,084838
	12	681	976	0,013502	1,01E-02	152,3	5,1	141,9	2,0	1,43	1,34	4324,347	0,011802
	13	1876	2529	0,024565	2,47E-02	147,4	5,7	137,7	2,6	1,38	1,30	4036,114	0,024629
	14	929	1456	0,053777	5,03E-02	143,8	3,2	135,9	0,8	1,35	1,29	3286,404	0,052058
	15	1763	2149	0,086496	9,03E-02	151,1	2,6	137,1	2,0	1,43	1,29	6526,109	0,088411
Discarded measure points													
	3	330	928	2,62E-02	2,57E-02	53,8	5,1	47,1	2,0	0,49	0,44	2592	0,025944
	12	2115	2414	3,51E-02	3,55E-02	30,7	5,1	27,1	5,7	0,27	0,24	1850	0,035306
	15	155	524	8,76E-02	9,34E-02	29,5	3,8	28,9	8,7	0,27	0,24	1226	0,090504

Result lift		Full bucket		Empty bucket		No bucket	
x1	1.1912e+003	-6526,109	-0,088410758	-2997,504	-0,082009	-2435,465	-0,08483841
x2	2.0815e+001	-3286,404	-0,052058073	-2536,695	-0,070604	-1444,029	-0,05824583
x3	2.3919e+001	-4036,114	-0,024628658	-1779,630	-0,050434	-1099,828	-0,04602077
x4	2.1516e+001	-4324,347	-0,011802064	-1436,289	-0,042943	-1506,299	-0,02696755
x5	-5.8221e+004	-4300	0	-1951,345	-0,030916	-1615,937	-0,00558555
		4300	0	-2131,276	-0,005867	-1600	0
		4324,347	0,011802064	-2100	0	1600	0
		4036,114	0,024628658	2100	0	1615,937	0,005585551
		3286,404	0,052058073	2131,276	0,0059671	1506,299	0,026967554
		6526,109	0,088410758	1951,345	0,0309163	1099,828	0,046020768
				1436,289	0,0429432	1444,029	0,058245834
				1779,630	0,0504344	2435,465	0,084838411
				2536,695	0,0705041		
				2997,504	0,082009		

D.3 – Data Generation

```
function [t, pla, plb, pta, ptb, xl, xt, vl, vt, pp] = datagen(data)

% ----------------------------
% Write in following form:
% [t, pla, plb, pta, ptb, xl, xt, vl, vt, pp] = datagen(data);
% DATAGEN sends back typed measured data filtered using a second
order
% Butterworth-filter.
% ----------------------------

ts=0.02;                            % Time step

% Filterdesign
[B,A] = butter(2,0.05,'low');       % Pressure transeducer
[B2,A2] = butter(2,0.02,'low');     % Position transeducer

% Seprate "data":s komponents
t=[0:ts:(length(data(:,1))-1)*ts]';
opla=data(:,1);
oplb=data(:,2);
opta=data(:,3);
optb=data(:,4);
oxl=data(:,5);
oxt=data(:,6);
opp=data(:,7);

% Filter all measured date
pla=filtfilt(B,A,opla);
plb=filtfilt(B,A,oplb);
pta=filtfilt(B,A,opta);
ptb=filtfilt(B,A,optb);
xl=filtfilt(B2,A2,oxl);
xt=filtfilt(B2,A2,oxt);
pp=filtfilt(B,A,opp);

% Derive the filtered position and define velocity vectors
vl=diff(xl)./0.02;
vt=diff(xt)./0.02;
vl=[vl ; vl(length(vl))];
vt=[vt ; vt(length(vt))];
```

D.4 – Power Calculation

```
% ----------------------------
% Command: calc_energy
% calc_energy computes the total- and optimal energy comsumption.
% ----------------------------
load lcycle                     % Loads data file
% Separates one duty cycle from the data file
lcycle2=lcycle(round(5/0.02):round(78/0.02),1:7);
% Generates och lowpass filter the measured data
[t, pla, plb, pta, ptb, xl, xt, vl, vt, pp] = datagen(lcycle2);
%_____________________________________________
%_________           Declaration of constants_____
dla = 0.110;                    % Cylinder diameter lift        [m]
dlb = 0.070;                    % Piston shaft diameter lift    [m]
dta = 0.150;                    % Cylinder diameter tilt        [m]
dtb = 0.080;                    % Piston shaft diameter tilt    [m]
Ala = 2*pi*dla^2/4;             % Area A-chamber lift         [m^2]
Alb = 2*Ala-2*pi*dlb^2/4;       % Area B-chamber lift       [m^2]
Ata = pi*dta^2/4;               % Area A-chamber tilt         [m^2]
Atb = Ata-pi*dtb^2/4;           % Area B-chamber tilt         [m^2]
dt=0.02;                        % Time step                   [s]
dploss=2.35;                    % Pressure losses due to hooses and
orifices
%_____________________________________________
%_________           Power calculations vcyl > 0________
%_____________________________________________
R=length(t);                    % Temporary help variable
Ptot=zeros(R,1);                % Total Power usage vcyl > 0
P=zeros(R,1);                   % Both due to hooses, orifices and
pdiffmax
P2=zeros(R,1);                  % Due to pdiffmax
Ptott1=zeros(R,1);              % Temporary part of Ptot
Ptott2=zeros(R,1);              % Temporary part of Ptot
Pt1=zeros(R,1);                 % Temporary part of Pt
Pt2=zeros(R,1);                 % Temporary part of Pt
P2t1=zeros(R,1);                % Temporary part of Pt
P2t2=zeros(R,1);                % Temporary part of Pt
for N = 1:(R-1)
    if ((vl(N) > 0.0005) && (vt(N) > 0.0005))
         Ptott1(N) = (Ala*vl(N)*(pp(N)))*1e5;
         Ptott2(N) = (Ata*vt(N)*(pp(N)))*1e5;
         Pt1(N) = (Ala*vl(N)*(pp(N)-pla(N)))*1e5;
         Pt2(N) = (Ata*vt(N)*(pp(N)-pta(N)))*1e5;
         P2t1(N) = (Ala*vl(N)*(pp(N)-pla(N)-dploss))*1e5;
         P2t2(N) = (Ata*vt(N)*(pp(N)-pta(N)-dploss))*1e5;
    elseif  ((vl(N) > 0.0005) && (vt(N) <= 0.0005))
         Ptott1(N) = (Ala*vl(N)*(pp(N)))*1e5;
         Ptott2(N) = 0;
```

```
        Pt1(N) = (Ala*vl(N)*(pp(N)-pla(N)))*1e5;
        Pt2(N) = 0;
        P2t1(N) = (Ala*vl(N)*(pp(N)-pla(N)-dploss))*1e5;
        P2t2(N) = 0;
    elseif ((vt(N) > 0.0005) && (vl(N) <= 0.0005))
        Ptott1(N) = 0;
        Ptott2(N) = (Ata*vt(N)*(pp(N)))*1e5;
        Pt1(N) = 0;
        Pt2(N) = (Ata*vt(N)*(pp(N)-pta(N)))*1e5;
        P2t1(N) = 0;
        P2t2(N) = (Ata*vt(N)*(pp(N)-pta(N)-dploss))*1e5;
    else
        Ptott1(N) = 0;
        Ptott2(N) = 0;
        Pt1(N) = 0;
        Pt2(N) = 0;
        P2t1(N) = 0;
        P2t2(N) = 0;
    end
Ptott1(N) = clearneg(Ptott1(N));
Ptott2(N) = clearneg(Ptott2(N));
Pt1(N) = clearneg(Pt1(N));
Pt2(N) = clearneg(Pt2(N));
P2t1(N) = clearneg(P2t1(N));
P2t2(N) = clearneg(P2t2(N));
Ptot(N) = abs(Ptott1(N) + Ptott2(N));
P(N) = abs(Pt1(N) + Pt2(N));
P2(N) = abs(P2t1(N) + P2t2(N));
end
%_____________________________________________
%__________          Power calculations vcyl < 0_______
%_____________________________________________
Pltot=zeros(R,1);       % Total power usage vcyl < 0
Pl=zeros(R,1);          % Power due to hooses, orifices and pdiffmax
Pl2=zeros(R,1);         % Power due to dpdiff
Pltott1=zeros(R,1);     % Temporary part of Ptot
Pltott2=zeros(R,1);     % Temporary part of Ptot
Plt1=zeros(R,1);        % Temporary part of Pt
Plt2=zeros(R,1);        % Temporary part of Pt
Pl2t1=zeros(R,1);       % Temporary part of Pt
Pl2t2=zeros(R,1);       % Temporary part of Pt
for N = 1:(R-1)
    if ((vl(N) < -0.0005) && (vt(N) < -0.0005))
        Pltott1(N) = (Alb*vl(N)*(pp(N)))*1e5;
        Pltott2(N) = (Atb*vt(N)*(pp(N)))*1e5;
        Plt1(N) = (Alb*vl(N)*(pp(N)-plb(N)))*1e5;
        Plt2(N) = (Atb*vt(N)*(pp(N)-ptb(N)))*1e5;
        Pl2t1(N) = (Alb*vl(N)*(pp(N)-plb(N)-dploss))*1e5;
        Pl2t2(N) = (Atb*vt(N)*(pp(N)-ptb(N)-dploss))*1e5;
    elseif  ((vl(N) < -0.0005) && (vt(N) >= -0.0005))
```

```
          Pltott1(N) = (Alb*vl(N)*(pp(N)))*1e5;
          Pltott2(N) = 0;
          Plt1(N) = (Alb*vl(N)*(pp(N)-plb(N)))*1e5;
          Plt2(N) = 0;
          Pl2t1(N) = (Alb*vl(N)*(pp(N)-plb(N)-dploss))*1e5;
          Pl2t2(N) = 0;
     elseif ((vt(N) < -0.0005) && (vl(N) >= -0.0005))
          Pltott1(N) = 0;
          Pltott2(N) = (Atb*vt(N)*(pp(N)))*1e5;
          Plt1(N) = 0;
          Plt2(N) = (Atb*vt(N)*(pp(N)-ptb(N)))*1e5;
          Pl2t1(N) = 0;
          Pl2t2(N) = (Atb*vt(N)*(pp(N)-ptb(N)-dploss))*1e5;
     else
          Pltott1(N) = 0;
          Pltott2(N) = 0;
          Plt1(N) = 0;
          Plt2(N) = 0;
          Pl2t1(N) = 0;
          Pl2t2(N) = 0;
     end
Pltott1(N) = clearpos(Pltott1(N));
Pltott2(N) = clearpos(Pltott2(N));
Plt1(N) = clearpos(Plt1(N));
Plt2(N) = clearpos(Plt2(N));
Pl2t1(N) = clearpos(Pl2t1(N));
Pl2t2(N) = clearpos(Pl2t2(N));
Pltot(N) = abs(Pltott1(N) + Pltott2(N));
Pl(N) = abs(Plt1(N) + Plt2(N));
Pl2(N) = abs(Pl2t1(N) + Pl2t2(N));
end
PLIFTtot=Ptott1+abs(Pltott1);
PLIFTdp=P2t1+abs(Pl2t1);
PTILTtot=Ptott2+abs(Pltott2);
PTILTdp=P2t2+abs(Pl2t2);
Figure(1);
plot(t(round(length(t)/2):length(t)),
Ptot(round(length(t)/2):length(t))/20,'k');
hold on;
plot(t(round(length(t)/2):length(t)),
Pltot(round(length(t)/2):length(t))/20,'r');
plot(t(round(length(t)/2):length(t)),
vl(round(length(t)/2):length(t)).*1100-100);
plot(t(round(length(t)/2):length(t)),
vt(round(length(t)/2):length(t)).*1100-100,'g');
hold off;
title('Use setzeroUSER to clear "bad" areas');
legend('Ptot(v>0)','Ptot(v<0)','vl','vt');
ylabel('Power [W]');
xlabel('t [s]');
```

D.5 – Clean *Up Function*

setzero

```
% ------------------------------------------------------------------
% Write
% [data1 data2 data3] = setzero(data1, data2, data3, t1, t2);
% Where (data1 data2 data3) =(Ptot,   P,   P2  ) if error when v > 0
%                           =(Pltot,  Pl,  Pl2 ) if error when v < 0
% and   [data1 data2 data3] = [PtotN  PN  P2N ]  if error when v > 0
%                           = [PltotN PlN Pl2N]  if error when v < 0
% “This function is internally used by setzero_lcycle”
% ------------------------------------------------------------------

function [data1 data2 data3] = setzero(data1, data2, data3, t1, t2);

dt=0.02;
t1=round(t1/dt);
t2=round(t2/dt);
for I = t1:t2
    data1(I)=0;
    data2(I)=0;
    data3(I)=0;
end
```

setzero_lcycle

```
% Use after calc_power and before calc_energy
[PtotN   PN   P2N ] = setzero(Ptot,P,P2,27.5,30);
[PtotN   PN   P2N ] = setzero(PtotN,PN,P2N,8.4,9.8);
[PtotN   PN   P2N ] = setzero(PtotN,PN,P2N,57,59.5);
[PtotN   PN   P2N ] = setzero(PtotN,PN,P2N,65,72.7);
[PltotN   PlN   Pl2N ] = setzero(Pltot,Pl,Pl2,0.01,18);
[PltotN   PlN   Pl2N ] = setzero(PltotN,PlN,Pl2N,35,54);

Figure(2);
plot(t(1:length(t)), PtotN/20,'k');
hold on;
plot(t(1:length(t)), PltotN/20,'r');
plot(t(1:length(t)), vl.*1100-100);
plot(t(1:length(t)), vt.*1100-100,'g');
hold off;
title('FIXED!');
legend('Ptot(v>0)','Ptot(v<0)','vl','vt');
```

D.6 – Energy Calculations

```
% ------------------------------------------------------------
% Command: calc_energy
% calc_energy computes the total- and optimal energy consumption.
% ------------------------------------------------------------

R=length(t);       % Total number of time steps R=t/ts
QtotN=zeros(R,1);  % Total energy consumption vcyl > 0
Q1N=zeros(R,1);    % Optimal energy consumption (no losses) vcyl > 0
Q2N=zeros(R,1);    % Energy consumption without losses in hooses and
                     orifices vcyl > 0
QltotN=zeros(R,1); % Total energy consumption vcyl < 0
Ql1N=zeros(R,1);   % Optimal energy consumption (no losses) vcyl < 0
Ql2N=zeros(R,1);   % Energy consumption without losses in hooses and
                     orifices vcyl < 0
for N = 2:R
    QtotN(N)=QtotN(N-1)+PtotN(N)*dt;
    Q1N(N)=Q1N(N-1)+(PN(N))*dt;
    Q2N(N)=Q2N(N-1)+(P2N(N))*dt;
    QltotN(N)=QltotN(N-1)+PltotN(N)*dt;
    Ql1N(N)=Ql1N(N-1)+(PlN(N))*dt;
    Ql2N(N)=Ql2N(N-1)+(Pl2N(N))*dt;
end

Figure(1);
plot(t(1:round(length(t)/2)), QtotN(1:round(length(t)/2)));
hold on;
plot(t(1:round(length(t)/2)), (QtotN(1:round(length(t)/2))-
Q2N(1:round(length(t)/2))), 'k');
plot(t(1:round(length(t)/2)), (QtotN(1:round(length(t)/2))-
Q1N(1:round(length(t)/2))), 'r');
hold off;
title('Power vcyl > 0 - One chort loading cycle');
ylabel('Energy [Ws]');
xlabel('t[s]');
legend('Qtot','Q(no pdiffmax)','Q(no
losses)','Location','NorthWest');
axis([0 38 0 3e5]);

Figure(2);
plot(t(1:round(length(t)/2)), QltotN(1:round(length(t)/2)));
hold on;

plot(t(1:round(length(t)/2)), (QltotN(1:round(length(t)/2))-
Ql2N(1:round(length(t)/2))), 'k');
plot(t(1:round(length(t)/2)), (QltotN(1:round(length(t)/2))-
Ql1N(1:round(length(t)/2))), 'r');
hold off;
```

```
title('Power vcyl < 0 - One chort loading cycle');
ylabel('Energy [Ws]');
xlabel('t[s]');
legend('Qtot','Q(no pdiffmax)','Q(no
losses)','Location','NorthWest');
axis([0 38 0 3e5]);

Figure(3);
plot(t(1:round(length(t)/2)),
QltotN(1:round(length(t)/2))+QtotN(1:round(length(t)/2)));
hold on;
plot(t(1:round(length(t)/2)),
QltotN(1:round(length(t)/2))+QtotN(1:round(length(t)/2))-
Ql2N(1:round(length(t)/2))-Q2N(1:round(length(t)/2)), 'k');
plot(t(1:round(length(t)/2)),
QltotN(1:round(length(t)/2))+QtotN(1:round(length(t)/2))-
Ql1N(1:round(length(t)/2))-Q1N(1:round(length(t)/2)), 'r');
hold off;
title('Total powerusage - One chort loading cycle');
ylabel('Energy [Ws]');
xlabel('t[s]');
legend('Qtot','Q(no pdiffmax)','Q(no
losses)','Location','NorthWest');
axis([0 38 0 5e5]);

Figure(4);
plot(t(1:round(length(t)/2)),
1.7143*0.7*(QtotN(1:round(length(t)/2))-Q1N(1:round(length(t)/2))),
'b');
hold on;
plot(t(1:round(length(t)/2)),
QltotN(1:round(length(t)/2))+QtotN(1:round(length(t)/2)),'r');
hold off;
title('Power consuption for the new hydraulic system');
ylabel('Energy [Ws]');
xlabel('t[s]');
legend('Qtot new system','Qtot current
system','Location','NorthWest');
axis([0 38 0 5e5]);

% Types the differences between the curves
format short;
diff1=(max(QltotN+QtotN)-max(QltotN+QtotN-Ql1N-
Q1N))/max(QltotN+QtotN)*100
diff2=(max(QltotN+QtotN)-max(QltotN+QtotN-Ql2N-
Q2N))/max(QltotN+QtotN)*100
```

E – Test Results, Lowering Functionality

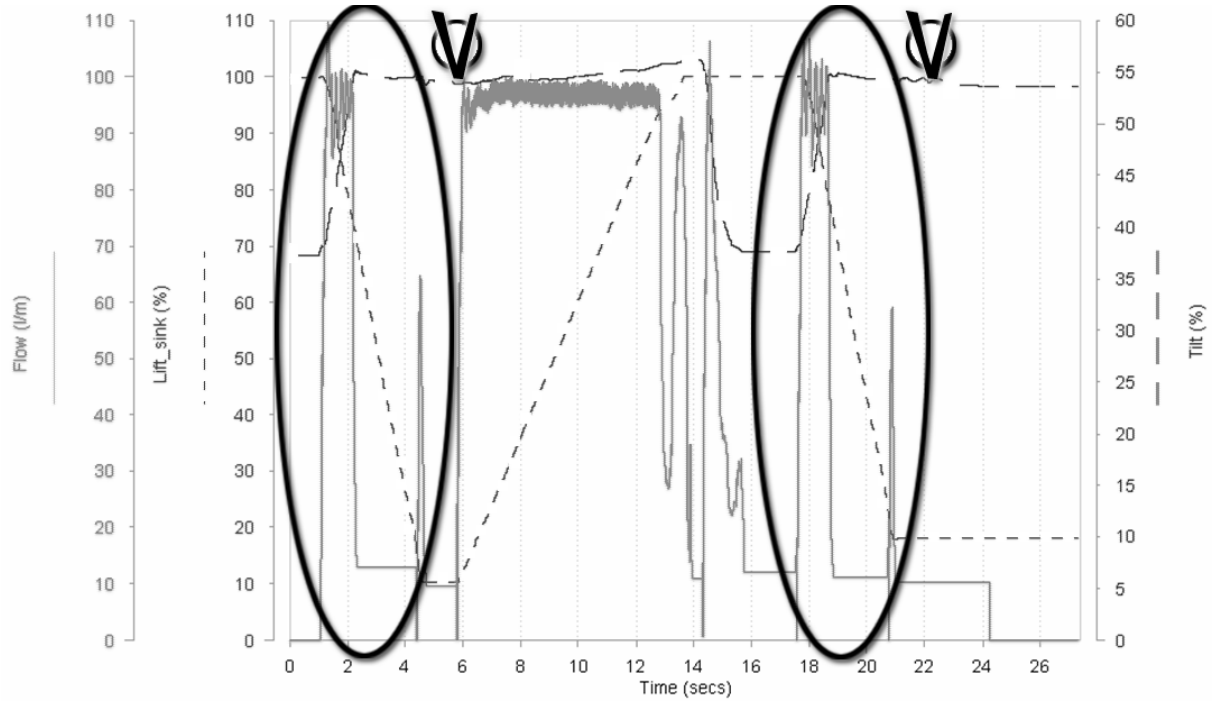


Figure 56: Test where tilt-in function works

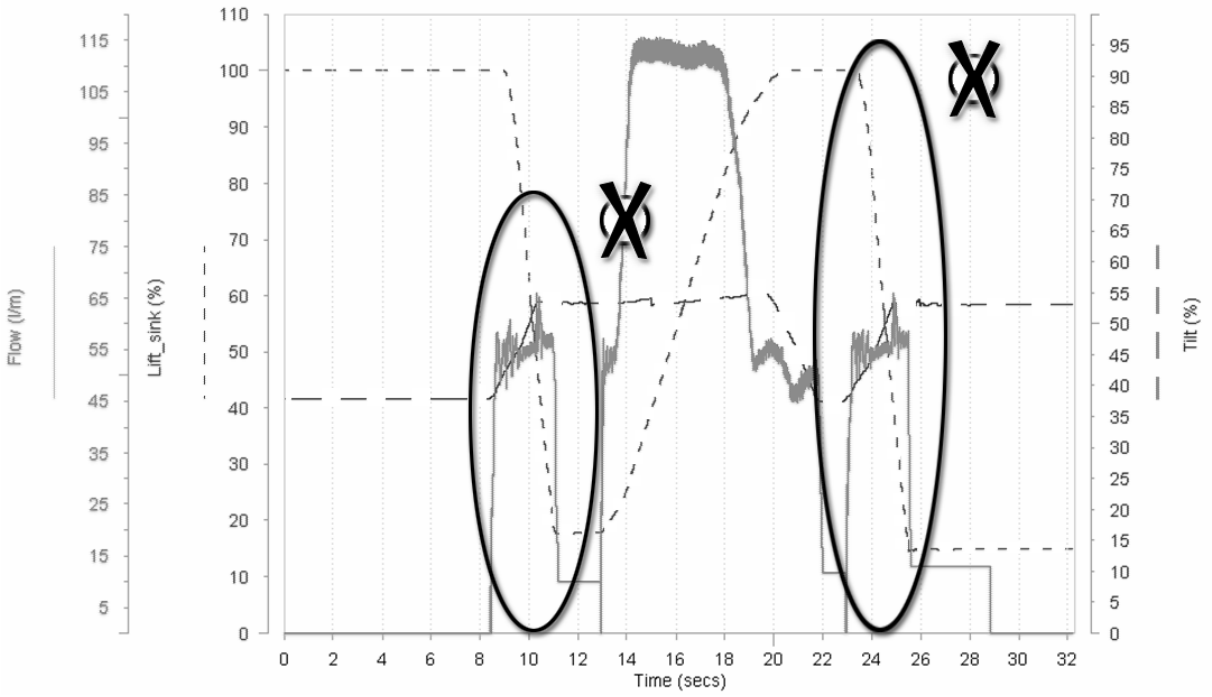


Figure 57: Test where tilt-in does not get activated

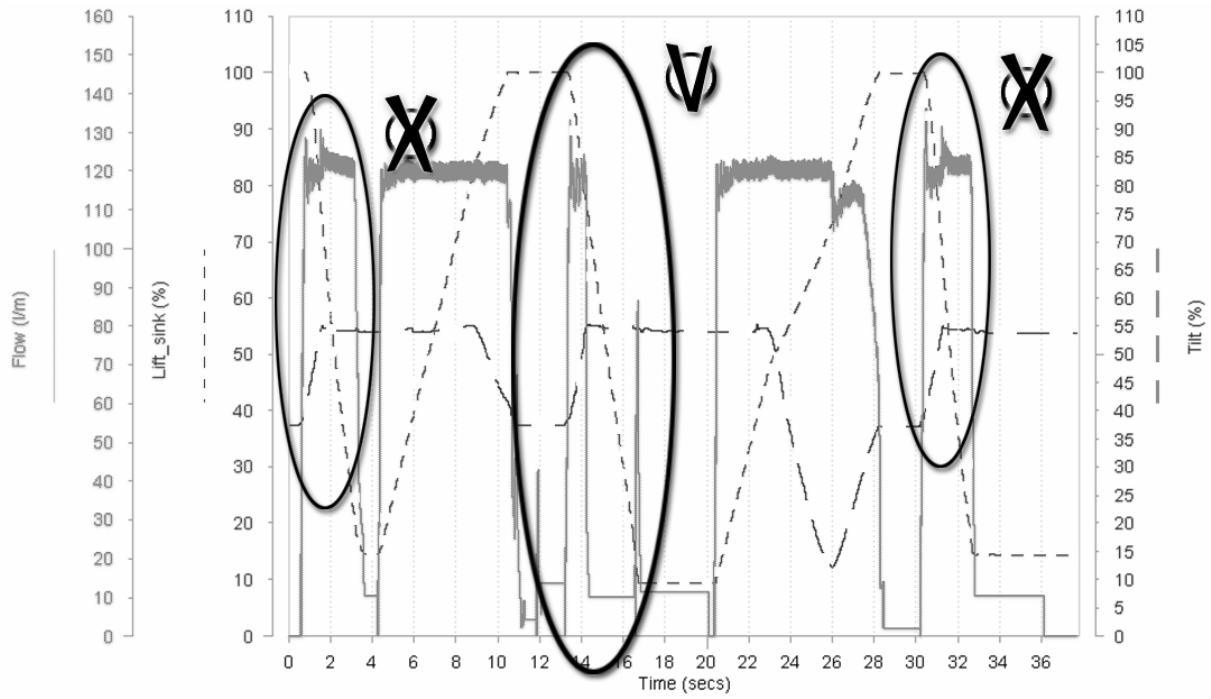


Figure 58: Test where tilt-in partly works

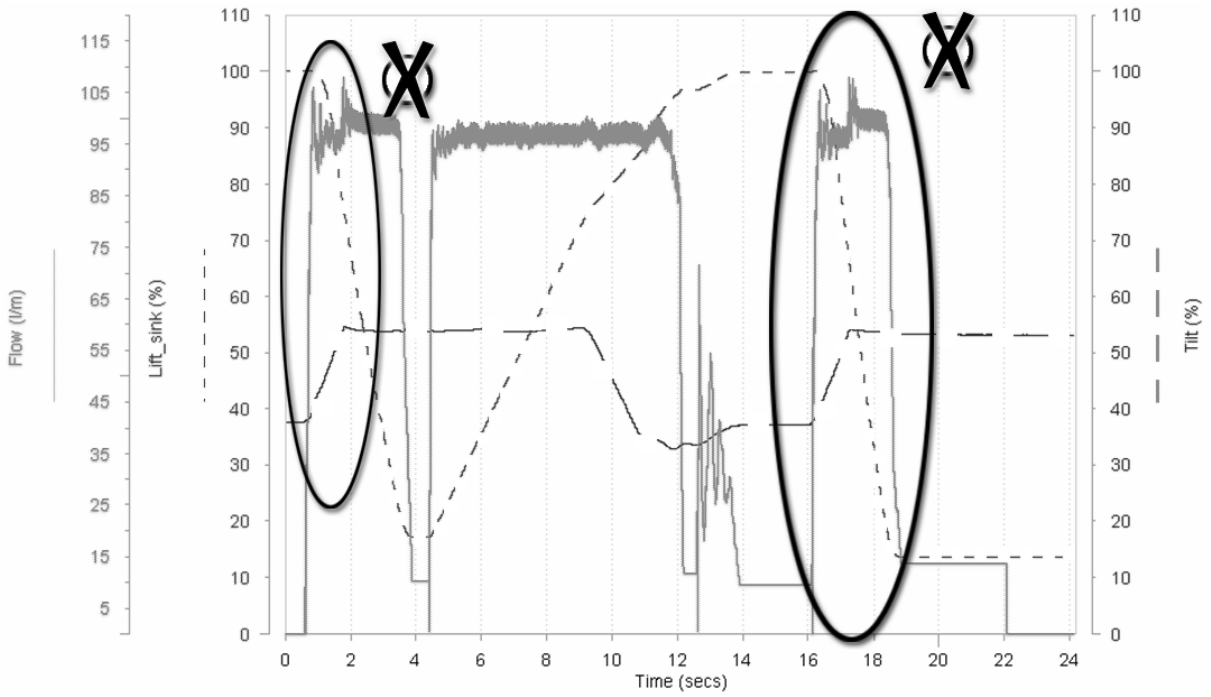


Figure 59: Test where tilt-in does not get activated

F – Pump Tests

F.1 – Definitions

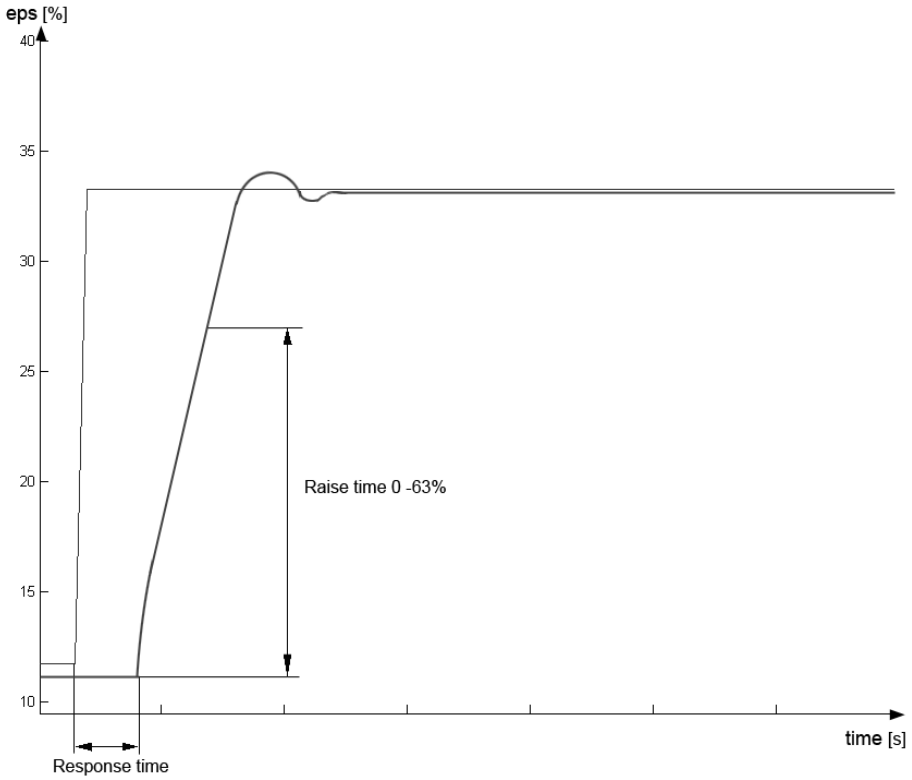


Figure 60: Definition of response and raise time

F.2 – External Pressure Feed Tests

Results – No external feed pressure

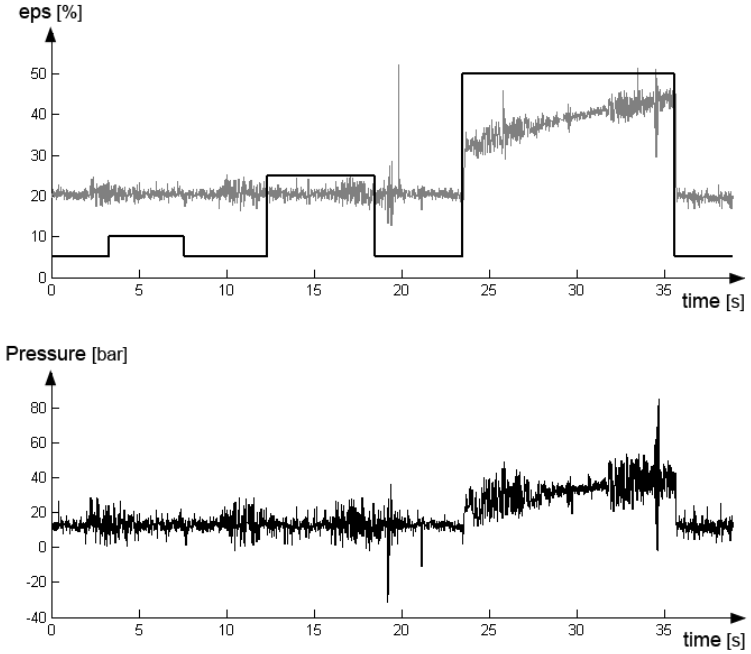


Figure 61: Plot 1 illustrates requested relative displacement versus true/measured. Plot 2 illustrates the pump pressure during the step test

Results – 35 bar external feed pressure

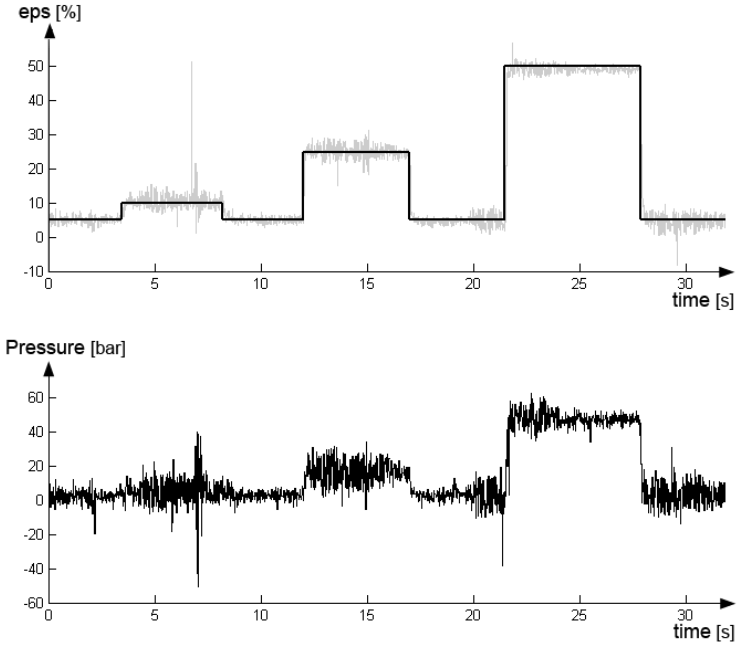


Figure 62: Plot 1 illustrates requested relative displacement versus true/measured. Plot 2 illustrates the pump pressure during the step test

Results – 70 bar external feed pressure

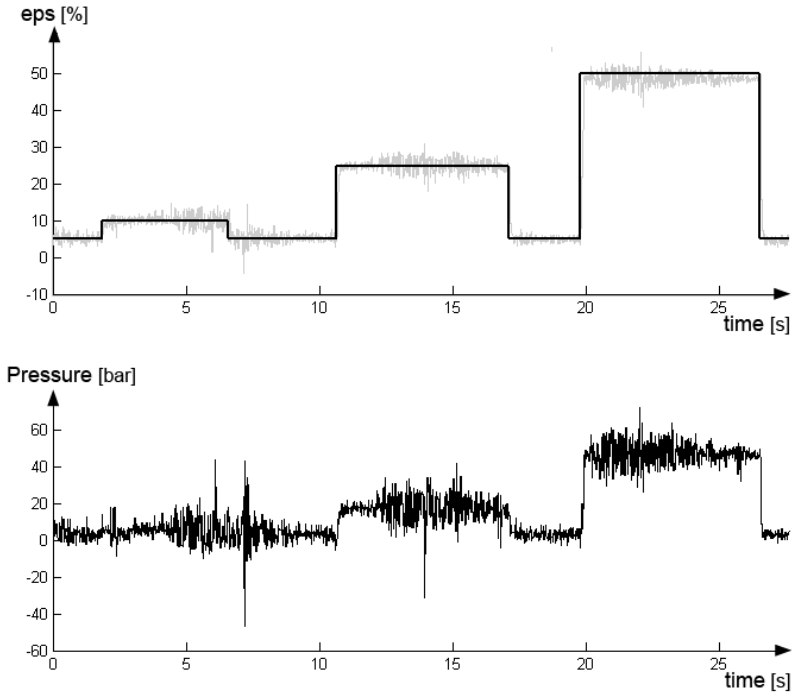


Figure 63: Plot 1 illustrates requested relative displacement versus true/measured. Plot 2 illustrates the pump pressure during the step test

Results – 150 bar external feed pressure

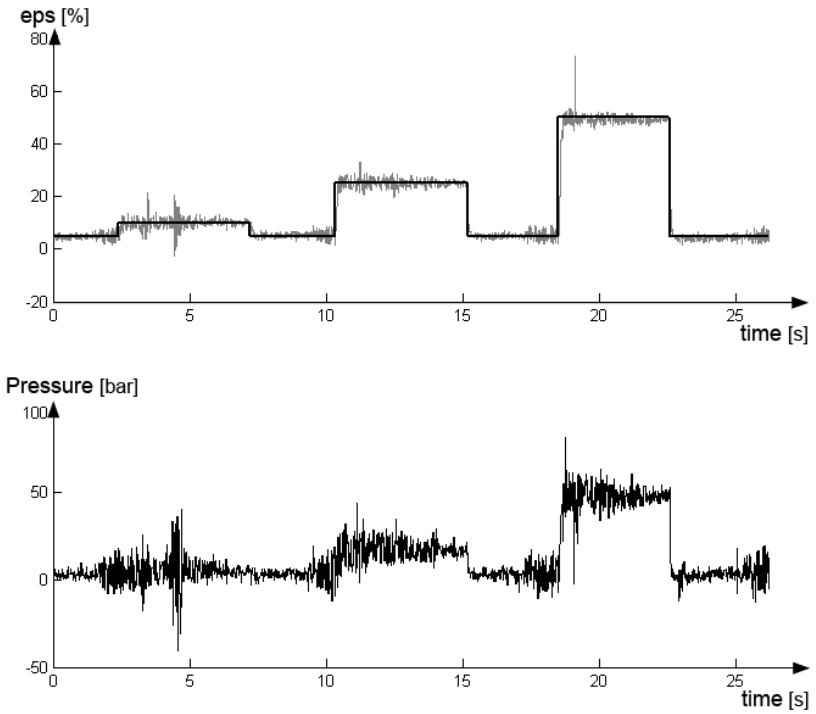


Figure 64: Plot 1 illustrates requested relative displacement versus true/measured. Plot 2 illustrates the pump pressure during the step test

G – Electrical power consumption

Components connected to the 24 V input plug (max 8 A)

Component	**Quantity**	**Total consumption**
CompactRIO	1	< 0.4 A
Pilot valves	8	< 4.8 A when all active
Pressure transducer	< 10	< 0.5 A at 5 V
Position transducer	2	< 0.2 A at 10 V
Joystick	2	< 0.1 A
		< 6.8 A

Components connected to the 12 V input plug (max 8 A)

Component	**Quantity**	**Total consumption**
WLAN Router	1	up to 2.5 A

H – Ordered Components

ELFA

Buntband med märkbricka	art. no. 55-159-60
Chassistiftdon, 20 st	art. no. 44-432-14
Hylspropp magnetventil 10 st	art. no. 43-420-02
Kabel Gul/Grön, 20 m	art. no. 55-400-54
Kabel Röd, 20 m	art. no. 55-400-26
Kabel Svart, 20 m	art. no. 55-400-00
Kabel Blå 20 m	art. no. 55-400-67
Kabel svart 20 m	art. no. 55-402-08
Kabel röd 20 m	art. no. 55-402-24
Kabel skärmad, 100m	art. no. 55-779-45
Kabelhylsdon, 20 st	art. no. 44-431-15
Kondensator 10 st	art. no. 67-808-37
Kopplingsplint 4-pol, 10 st	art. no. 48-354-27
Motstånd 100 st	art. no. 60-106-15
Mutter M8 20 st	art. no. 44-433-05
Silikonmellanlägg	art. no. 75-646-77
Skarvdon TP-kabel	art. no. 42-692-96
Slangklämma mini 10 st	art. no. 55-127-44
Slangklämma 10 st	art. no. 55-027-29
Spänningsregulator 4 st L78S10CV	art. no. 73-092-63
Spänningsregulator, 2 st L78S05CV	art. no. 73-095-60
Spänningsregulator, 2 st L78S12CV	art. no. 73-095-78
Stickpropp, 2 st	art. no. 42-100-68
Tryckvippströmställare, 2 st	art. no. 35-008-16
Universalstickpropp 24 V	art. no. 42-011-58
Uttag 2-vägs	art. no. 42-102-66

ONOFF

Router / Acesspoint

I – File descriptions

Filename	Operation	Velocity	Load	RPM
lcycle	The short duty cycle			
data	Collected points for the friction calculations			
1	Raise, lower with constant speed	Very low	Empty bucket	1280
2	Raise, lower with constant speed	Low	Empty bucket	1280
3	Raise, lower with constant speed	Normal	Empty bucket	1280
4	Raise, lower with constant speed	Fast	Empty bucket	1280
5	Raise, lower with constant speed	Max	Empty bucket	1280
6	Tilt in, out with constant speed	Very low	Empty bucket	1280
7	Tilt in, out with constant speed	Low	Empty bucket	1280
8	Tilt in, out with constant speed	Normal	Empty bucket	1280
9	Tilt in, out with constant speed	Fast	Empty bucket	1280
10	Tilt in, out with constant speed	Max	Empty bucket	1280
11	Raise, lower with constant speed	Very low	No bucket	1280
12	Raise, lower with constant speed	Low	No bucket	1280
13	Raise, lower with constant speed	Normal	No bucket	1280
14	Raise, lower with constant speed	Fast	No bucket	1280
15	Raise, lower with constant speed	Max	No bucket	1280
16	Tilt in, out with constant speed	Very low	No bucket	1280
17	Tilt in, out with constant speed	Low	No bucket	1280
18	Tilt in, out with constant speed	Normal	No bucket	1280
19	Tilt in, out with constant speed	Fast	No bucket	1280
20	Tilt in, out with constant speed	Max	No bucket	1280
21	Raise, lower with constant speed	Very low	Filled bucket	1280
22	Raise, lower with constant speed	Low	Filled bucket	1280
23	Raise, lower with constant speed	Normal	Filled bucket	1280
24	Raise, lower with constant speed	Fast	Filled bucket	1280
25	Raise, lower with constant speed	Max	Filled bucket	1280
26	Tilt in, out with constant speed	Very low	Filled bucket	1280
27	Tilt in, out with constant speed	Low	Filled bucket	1280
28	Tilt in, out with constant speed	Normal	Filled bucket	1280
29	Tilt in, out with constant speed	Fast	Filled bucket	1280
30	Tilt in, out with constant speed	Max	Filled bucket	1280
31	Raise but, tilt up a little bit Lower but, tilt down a little bit Raise but, tilt down a little bit Lower but, tilt up a little bit		Filled bucket	1680
32	Tilt up but, raise a little bit Tilt down but, lower a little bit		Filled bucket	1680

Tilt up but, lower a little bit
Tilt down but, raise a little bit

33 Raise but, tilt up a little bit — Filled bucket 680
Lower but, tilt down a little bit
Raise but, tilt down a little bit
Lower but, tilt up a little bit

34 Tilt up but, raise a little bit — Filled bucket 680
Tilt down but, lower a little bit
Tilt up but, lower a little bit
Tilt down but, raise a little bit

J – LabView Files

File Name	Description
NTCM.prj	Project explorer
NTCM (HOST).vi	VI that runs on the host computer
Valve_display.vi	Displays active valvistor elements
NTCM (FPGA).vi	VI that runs on the CompactRIO FPGA
Measure Loop Rate (FPGA).vi	Tick counter
Pulse Width Modulation (FPGA).vi	PWM signal generator
NTCM (TARGET).vi	VI that runs on the CompactRIO target
16_bits_to_signal(+-5V).vi	Converts a 16 Bit signal to Volts with ±5V range
16_bits_to_signal(+-10V).vi	Converts a 16 Bit signal to Volts with ±10V range
Bits_to_signal.vi	SubVI: Converts bits to signals
Voltage_scaling.vi	SubVI: Converts and Calibrates transducer signals
Signal_16bits.vi	Converts signal with ±10V range to 16 bit signal
Signal_to_bits.vi	SubVI: Converts signal to bits
PWM_controller.vi	Transforms desired duty cycles and periods to ticks
*CANcom.vi**	Converts Can-frames to physical signals (V, rpm etc.)
Valve_controller.vi	Contains the dll for the valve controller
Pump_controllers.vi	Contains the dll for the pump controllers

**To use the CANcom, an upgrade has to be installed. It can be found at www.ni.com, upgrade: NI-CAN 2.33*

K – Hydraulic Scheme Volvo L60:E