



KONGSBERG



## KPEC Document Collection

### KPEC

<b>Employer</b>	Kongsberg Defence & Aerospace			
<b>Group Members</b>	<b>Name</b>		<b>Initials</b>	
	Ola Pedersen Aasheim		OA	
	Salahuddin Asjad		SA	
	Hung Dinh		HD	
	Dler Hasan		DH	
	Even Gudbrandsen		EG	
	Jannik Schäffer		JS	
<b>Document information</b>	<b>Revision</b>	<b>Date</b>	<b>Approved</b>	<b>Pages</b>
	1.0	18.05.2015	KPEC	644





## Document overview

Kongsberg Programmable Engine Control is a high-end ECU designed for interfacing petrol engines with up to 8 cylinders for use in motorsports applications. The KPEC ECU is a motherboard design for interfacing the TE0720 micromodule, a Zynq SoC based processing platform for embedded applications. The KPEC ECU can interface up to 44 sensors and 24 actuators with expansion support.

This is the collection of all public documentation created by KPEC 2015. The PDF version of this document is publicly available and contains the complete documentation of the KPEC ECU.

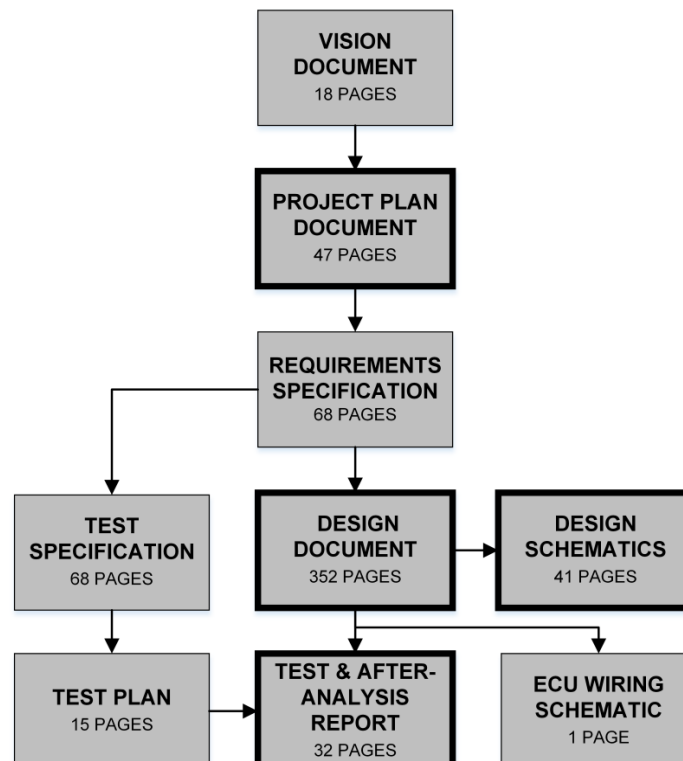
This document uses cross references and navigation buttons for easing access to document features.

### Navigational features of this document:

- Clicking the red KPEC logo in the top right corner of any page brings you back to this page.
- All chapters of documents are accessible in the navigation pane of the PDF viewer
- Every cross reference within the documents are clickable
- The content lists of each document is clickable
- Schematic navigational features:
  - Clicking the pathname in the schematics brings you to the schematic content list
  - Clicking hierarchical blocks in the schematics brings you to the corresponding schematic page



Clicking any box in the overview figure below brings you to the respective document. Important documents are emphasized.





KONGSBERG



## Vision document

### KPEC

<b>Employer</b>	Kongsberg Defence & Aerospace			
<b>Group Members</b>	<b>Name</b>		<b>Initials</b>	
	Ola Pedersen Aasheim		OA	
	Salahuddin Asjad		SA	
	Hung Dinh		HD	
	Dler Hasan		DH	
	Even Gudbrandsen		EG	
	Jannik Schäffer		JS	
<b>Document information</b>	<b>Revision</b>	<b>Date</b>	<b>Approved</b>	<b>Pages</b>
	4.0	18.05.2015	HD	18





## Abstract

This vision document is our first formative document in this project. The purpose of this vision document is first of all to present an overview of the preparation that is done concerning the startup of our Bachelor thesis. But this document will also give an idea of what must be accomplished to achieve our goals. After reading this document, one should be able to form a basic understanding of the overall objectives and framework for this project. This vision document is a basis of how the project group is going to conduct the project, but keep in mind that this document is not our final document.





## Revision Table

Version	Date	Approval	Description
1.0	05.01.2015	HD	<ul style="list-style-type: none"> <li>Created the document</li> </ul>
1.1	07.01.2015	OA	<ul style="list-style-type: none"> <li>Changed to kpec.no email addresses</li> <li>Fixed indentations on bullet lists</li> </ul>
2.0	22.01.2015	HD	<ul style="list-style-type: none"> <li>Revised and published</li> </ul>
3.0	09.03.2015	EG	<ul style="list-style-type: none"> <li>Revised project model section, Revised and published</li> </ul>
4.0	18.05.2015	HD	<ul style="list-style-type: none"> <li>Revised and published</li> </ul>

## Contents

Abstract .....	2
Revision Table .....	3
List of figures.....	4
List of tables .....	4
1. Introduction .....	5
2. Organization .....	6
2.1. Group Members.....	6
2.2. Employer .....	9
2.2.1. Background information .....	9
2.2.2. Stakeholders.....	10
2.2.3. Primary Stakeholders.....	10
2.2.4. Secondary Stakeholders .....	10
3. Assignment .....	11
3.1. General description .....	11
3.2. Detailed project description.....	11
3.2.1. Short term goals .....	11
3.2.2. Long term goals .....	11
3.3. System overview.....	12
4. Objectives .....	14



4.1.	Group values and vision.....	14
4.2.	Group objectives.....	14
4.3.	Employers objectives .....	15
5.	Project model .....	16
5.1.	Iterative evolutionary agile model.....	16
5.2.	Lifecycle assessment.....	16
6.	Time management and estimation.....	17
6.1.	Autumn .....	17
6.2.	Spring.....	18
6.3.	System for time management.....	18

## List of figures

Figure 1 - System overview of the project. ....	12
Figure 2 - Detail view of ECU .....	13
Figure 3 - Iterative evolutionary agile model.....	16

## List of tables

Table 1 - Group members .....	6
Table 2 - Primary stakeholders, advisors and sensors during the project .....	10
Table 3 - Time estimation on different parts of the project related to documentation. ....	17



## 1. Introduction

This project is the final part of our bachelor's degree at Buskerud and Vestfold university college (HBV), and is intended to build a bridge between theory and real life work situation. Therefore the project work is intended to be as close as possible to real engineering project. The vision document is written for the bachelor thesis with project name "KPEC" given by our employer Kongsberg Defence & Aerospace. The project name is an abbreviation for Kongsberg Programmable Engine Control, and the extent of the project is to design hardware and software to control an internal combustion engine (ICE). We are a group consisting of six students, where four are studying Cybernetics and Mechatronics (electrical), while two are studying Embedded Systems (Computer). This gives us an experience of working with other engineering disciplines.

This bachelor-thesis is the final part of our bachelor's degree at Buskerud and Vestfold university college (HBV). The project is intended to last from January 2015 until June 2015, and contains about 3600 working hours. Preparation to the project has been done in the period of October until December 2014, where we defined the project plan, preliminary studies review and vision document. This vision document is intended to give a first overview of the assignment, the group, scope & values, and which project model we are planning to use.

Kongsberg Defence & Aerospace (KDA) has given us an exciting and challenging project. This project is supposed to go over many years and the final goal is to have a fully working ECU (Engine Control Unit) to control an internal combustion engine (ICE) more accurately than the solutions that are currently on the market.

The project is intended to be solved by the group unaided by others, but the group will have some guidance by external resources from KDA and HBV. The project group has an office at HBV where the group work will be done. As a project group, we are responsible for successful completion of the project in terms of time, and budget. We intend to achieve this by focusing on quality and proactive project controlling.





## 2. Organization

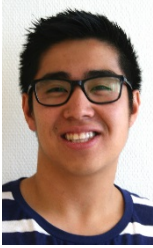

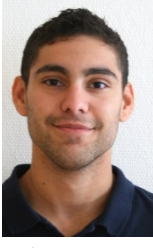
This chapter introduces the group members as well as our employer, Kongsberg Defence & Aerospace. The group consists of four students from electrical engineering, and two students from computer engineering at Buskerud and Vestfold University College (HBV).

### 2.1. Group Members

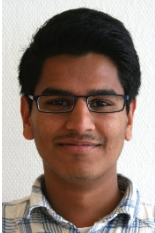
**Table 1 - Group members**

Personal information	Project role	Description
 <b>Even Gudbrandsen</b> Electrical engineering Contact: even@kpec.no Tel. 928 10 601	Project manager & Quality responsible	22 years old, from Modum. Studying cybernetics and mechatronics at HBV.  Main interests are electronics, automotive, systems engineering and outdoor life. Certified motor vehicle mechanic, and solid experience in automotive technology.
 <b>Ola P. Aasheim</b> Electrical engineering Contact: ola@kpec.no Tel. 915 52 141	Hardware design & layout responsible	21 years old, from Oslo. Studying cybernetics and mechatronics at HBV  Areas of interest undergoes electronics, programming and technology related subjects. Loves long hikes in the outdoors



 <b>Hung V. Dinh</b> Electrical engineering Contact: hung@kpec.no Tel. 905 23 885	Preliminary studies & modelling responsible	23 years old, from Nøtterøy. Studying cybernetics and mechatronics at HBV.  Main interest are electronics, training and football. Certified production electronics technician
 <b>Jannik B. Schäffer</b> Electrical engineering Contact: jannik@kpec.no Tel. 922 13 272	Test & Verification responsible	25 years old, from Larvik. Studying cybernetics and mechatronics at HBV  Interdisciplinary interest for electronics and mechanics. 5 years of working experience as a motorcycle-technician, particularly interested in hi-fi, custom vehicles and extreme sports.
 <b>Dler Hasan</b> Computer engineering Contact: dler@kpec.no Tel. 400 740 96	Marketing & requirements responsible	22 years old, from Stockholm. Studying computer engineering in embedded systems.  Experience as project manager and administration officer for the project ENT3R, helping young students with math. Very interested in programming, video editing and picture design



	Software design & interface responsible	<p>21 years, from Bergen. Studying computer engineering in embedded systems.</p> <p>I have 7 years' experience in sales in computer &amp; photography.</p> <p>Main interests are programming, cricket and football.</p>
---	---	---

Every member of the group has been assigned different project roles and responsibility areas for ensuring quality and traceability. However, it is not intended that each individual should work only with their responsibility areas. The delegation of different project roles has been done to ensure that the project outcomes are presented in a structural way.



## 2.2. Employer

Our employer is Kongsberg Defence & Aerospace (KDA), which develops and produces Defence related products for global costumers. KDA is a subsidiary of Kongsberg Gruppen ASA.

### 2.2.1. Background information

KONGSBERG is an international company who provides hi-technology systems and products for customers in oil & gas sector, maritime, military and aerospace. In 1996 Kongsberg Våpenfabrikk changed their company name to Kongsberg Gruppen ASA. The company has about 7500 employees and has headquarters in Kongsberg. Local offices in over 25 countries ensure access to all important markets and proximity to customers. Kongsberg Gruppen is divided into different divisions; Kongsberg Maritime, Kongsberg Defence Systems, Kongsberg Protech Systems and Kongsberg Oil & Gas Technologies.

Kongsberg Defence Systems is again divided into some subdivisions; Missile Systems, Naval Systems, Integrated Defence Systems, Aerostructures, Defence Communications and Space & Surveillance. Today, Kongsberg Defence & Aerospace Systems (KDA) is one of Norway's premier suppliers of defense and aerospace-related systems and products.

KDA has through the past developed many interesting systems. Many people recognize the company for its development and production of the first passive IR seeker-based missile, the Penguin and development of Protector RWS which is a remotely controlled weapon station. Development of the Penguin's successor, the Naval Strike Missile were completed in 2007 and is currently in use in the Royal Norwegian Navy's Skjold-Class Corvettes and Fritjof Nansen-Class frigates, as well as a coastal defense system for the Polish military. Currently KDA are working on several large projects like Joint Strike Missile (JSM), a land and sea based missile for integration in the new Lockheed Martin F-35 Lightning II and NASAMS aerial defense system.



### 2.2.2. Stakeholders

A stakeholder is anyone who has an interest in our project, and we have divided these stakeholders as primary and secondary. The primary stakeholders are anyone with direct interests in the project, while a secondary stakeholder is anyone which is indirectly affected by the project. During the project, we will concurrently use our advisors and resources to guide us on the way to complete the project successfully.

### 2.2.3. Primary Stakeholders

Our primary stakeholders are the group, our employer, college HBV, and our resource group. These individual will be monitoring the project work, as well as attending the presentations. For convenience, we have listed our advisors and sensors in Table 2.

**Table 2 - Primary stakeholders, advisors and sensors during the project**

Personalia	Role	Contact
Antonio L.L. Ramos	Internal advisor, HBV	Antonio.Ramos@hbv.no
Karoline Moholth	Internal sensor, HBV	Karoline.Moholth@hbv.no
Jonas Brattensborg	External advisor, KDA	Jonas.Brattensborg@kongsberg.com
Eirik Kile	External sensor, KDA	Eirik.Kile@kongsberg.com

### 2.2.4. Secondary Stakeholders

We have defined our secondary stakeholders as person who has indirectly contact with our product; this may be the user of the ECU or even the employee working with an engine controlled by our ECU. During the lifetime of the product, the product might need maintenance, upgrades and software updates; this means that we will need a maintenance team which will take care of this job. Thus the maintenance team is also a secondary stakeholder for us. We need to make a product that is easily handled and operated, so that everyone that uses the product can use it properly.





## 3. Assignment

### 3.1. General description

The project is given by KDA (Kongsberg Defence & Aerospace) and is supposed to give the students an exciting and challenging project. The task is meant to be an interdisciplinary project, consisting of students from cybernetics, electronics, hardware, software and machine. The project is supposed to go over many years, and the final goal is to have a fully working ECU for an internal combustion engine that is based on a Xilinx Zynq module.

At the end of spring 2015, KDA is expecting deliverance on electronic design for the ECU, as well as a tablet application that communicates with the Zynq module. Due to that this project is supposed to last for many years, the short term goals are the main objective for this project. However, the long term goals must also be taken into account when designing the electronic design and software; they are listed in section 3.2.2.

### 3.2. Detailed project description

#### 3.2.1. Short term goals

- Select a micro module that is based on a Xilinx Zynq SoC.
- Produce schematic design for the ECU, using Cadence Allegro.
- Consider the need of an operating system.
- Establish communication towards the Zynq and tablet
- Version control of source code using GitHub.

When designing the schematics, software to the tablet and preparations towards PCB design, the project group needs to take into account for further implementation and upgrades that might arise in during the entire lifetime of the ECU.

#### 3.2.2. Long term goals

Fully working ECU with following specifications

- Tuning from tablet
- Low-level timing and communication toward external components, which gets processed by the FPGA.
- The option for using ethanol fuel to drive the engine
- Spark plugs with integrated pressure sensors to detect Peak Pressure Point (PPP)
- Automatic adjusting the flow rate to the injectors, so that all injectors deliver the same amount of fuel.
- Automatic adjustment of fuel mapping.
- Option to reprogram flash using Ethernet/WIFI.
- Produce pneumatic valves that can be controlled by the ECU.



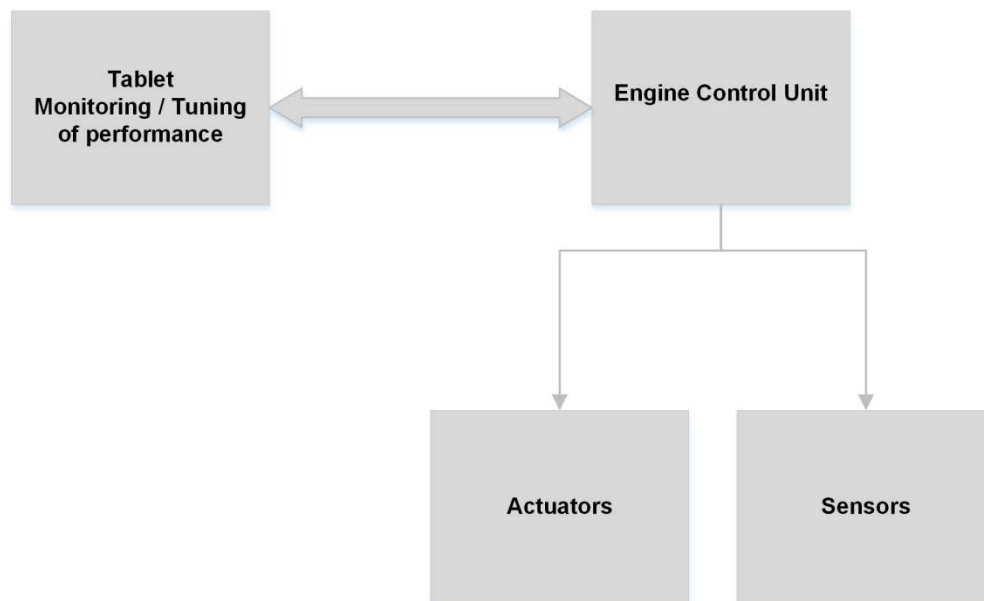
- Produce casing for ECU.

### 3.3. System overview

An ECU is an electronic unit which is used to monitor and control an internal combustion engine (ICE). The ICE is an engine where chemical energy is converted to mechanical energy, where the combustion occurs when fuel and air is mixed and compressed in a combustion chamber. When the air and fuel mixture are ignited due to spark plugs or self-ignition, the pressure in the combustion chamber are increased and apply force to a piston in an engine. The piston is then connected to a rod (crank) that converts vertical motion to rotational motion.

The ECU is supposed to be monitored and controlled by a tablet. The application installed on the tablet will have features, so the user easily can view and read values that are coming from the ECU. It can for instance show us the output values from different sensors or how economic the engine is running. It will also have features to control the ECU output power. In addition, the application will also have feature to log every change in value from a sensor, so the user can have a look at the log at a later time.

Our assignment is to design an electric design and software for use on a tablet with wireless communication for the ECU. This assignment gives us some technical and design challenges, and therefore we will focus on using Systems Engineering to accomplish the project. Figure 1 gives an overview of the system.



**Figure 1 - System overview of the project.**



A more detail view of the ECU is illustrated in Figure 2. The overall hardware mission is to design and layout all subsystems that are related to the Zynq module; the external sensors and actuators are also displayed in the figure.

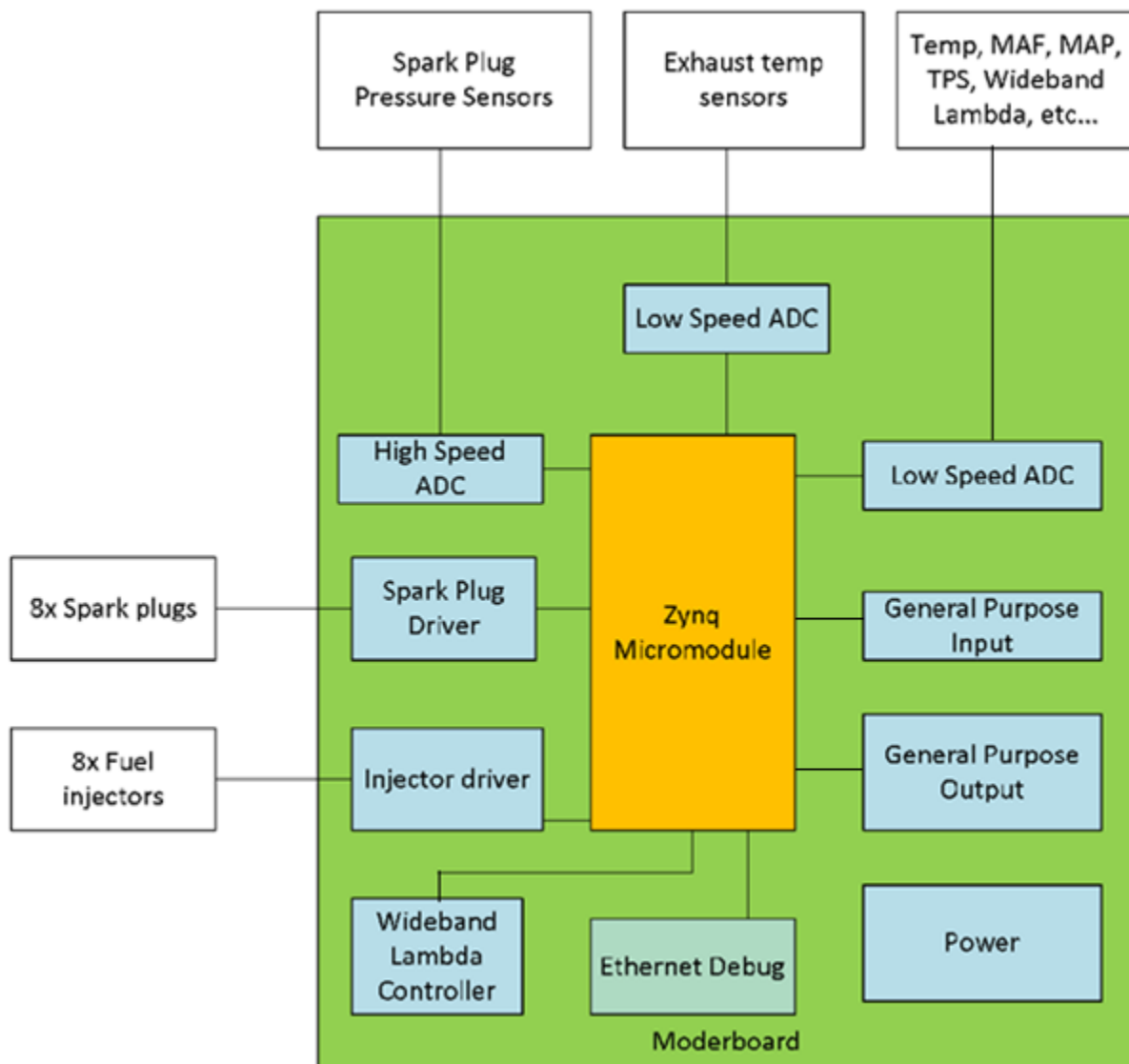


Figure 2 - Detail view of ECU<sup>1</sup>

<sup>1</sup> Figure from KDA project description



## 4. Objectives

This chapter is about the objectives and motivation for accomplishing the assignment with best possible result. To ensure quality and continuous work throughout the project, the team has defined project rules and values. The project rules are a subsidiary part of the group contract, and could be found in the appendixes.

### 4.1. Group values and vision

The group values are essential to ensure that the group dynamic is optimal during the whole project. Our values motivate us to accomplish good results, and give us a pointer on how we should collaborate with others both internally and externally. Our values are Quality, Ambitions and Passion.

The group has a vision on delivering a high quality product that exceeds KDA's requirements, and to learn how ECU works thoroughly. We want to experience to work with an authentic project in a professional environment, and gain insights in how engineering projects develops.

We will accomplish our goals and visions by keeping the ambitions and motivation high throughout the project. This is accomplished by working seriously and systematic, as well with dedication and passion towards the project. Our niche is to produce a seamless control system that is controlled wirelessly by an app with smooth and user friendly design.

### 4.2. Group objectives

- Find a solution to the project that exceeds employer requirements.
- Deliver a final result that satisfies our user and system requirements.
- Come up with smart and innovative ideas and solutions.
- Achieve a well-earned grade that satisfies all team members.
- Increase knowledge and skills within project work.
- Challenge the knowledge that has been acquired throughout the engineering course, and test it out in practice.
- Deliver a high standard documentation to capture the viewers' attention with a unique design.
- Put in the amount of work that must be done to fulfill the tasks.
- Follow the templates and design structure from the project.
- Keep updated throughout the project development.
- Help each other out in difficulties.



### 4.3. Employers objectives

We have specified the employer objectives through what we mean are the most important notes. We have to keep in mind throughout the project that we are working for a huge company, and we have to satisfy both the company and user/system requirements.

- Fully functioning Engine Control Unit (ECU).
- Build form and layout for a motherboard which micro model will be placed on.
- Implementing an FPGA module to read a low speed ADC.
- Create demo software that tunes from a tablet, (Android or IOS).
- Create an exciting interdisciplinary task.



## 5. Project model

When working on project of this extent, it is crucial to have a good platform for managing the project. By implementing Systems engineering (SE) as a project management tool, we can control the engineering project from beginning until end. SE is an approach that focuses on the big picture view of entire systems.

SE is based upon common sense for its tools and techniques, and its foundation relies on well-defined procedures. We will use SE-tools concurrently throughout the project, and we will design the system by assessing the entire life cycle. The first step on the way is to define a project model to follow throughout the project, and we chose a modified V-model.

### 5.1. Iterative evolutionary agile model

We have selected an iterative evolutionary agile model (hereafter our model) that is based on a V-model in use for our project as seen in Figure 3. Our model uses an iterative approach with iterations during the project to gain understanding of the product and to add functionality. The kernel of our model is iterative, and the project phases are performed stepwise. This implies that the first iteration is longer than the rest of the iterations, due to the fact that a concept phase, requirements phase and preliminary design phase must be completed before the design iteration could be performed.

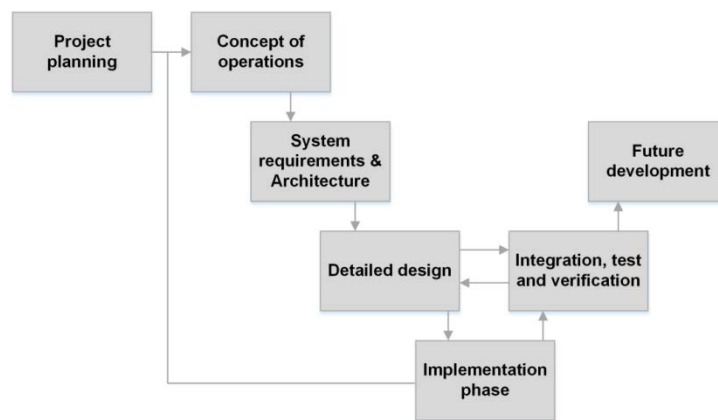


Figure 3 – Iterative evolutionary agile model

### 5.2. Lifecycle assessment

The project will be designed for lifecycle, which means that we will consider the whole lifetime from conception to disposing. We will consider utilization and upgrades as well as disposal of the system before we enter the design phase. By documenting thoroughly in the early phases of the project as well as developing subsystems simultaneous, we increase our chances of successful product development.



## 6. Time management and estimation

Bachelor thesis executed at HBV, has a weight of 20 ECTS. The EU-standard time estimation for 10 ECTS is defined to be between 250-300 hours depending on ambition level of the student. The bachelor thesis will therefore contain approximately 600 hours of project work per student. The total resources available in the group are therefore 3600 hours.

The evaluation weight of the bachelor thesis is 50% of project documentation, 25% presentations, and 25% the product itself. Therefore, the documentation of the project is crucial in gaining a decent grade. The project consists of six main reports and three presentations which we have estimated in Table 3. It is important to understand that the time estimations are only an abstract overview, but it gives us a simple outline so we can start planning.

There are a lot of preparations and practical details that must be considered during this project. Therefore, we have decided to use 15% of the available resources in the autumn and 85% during the spring of 2015. The planning phase of the project started the second week of October 2014. The italicized text in Table 3 indicates relative completion of the project.

**Table 3 - Time estimation on different parts of the project related to documentation.**

Semester	Project phase	Percentage	Hours
Autumn	Project planning phase	10%	360
Autumn	Concept of operations phase	5%	180
Spring	System requirements and architecture phase	10%	360
Spring	Detailed design HW & SW phase	40%	1440
Spring	Implementation phase	15%	540
Spring	Integration, test and verification phase	10%	360
	Administrative	10%	360
<b>Sum</b>		<b>100%</b>	<b>3600</b>

### 6.1. Autumn

During the autumn of 2014, all group members have 30 ECTS of other subjects until December. Because of this, the maximum amount of work per student is 12 hours per week. The working period stretches from week 41 until week 51 (week 49-50 is exam period). In this period, the group will be focusing on completion of vision document and project plan, as well as preliminary studies and other practical problems.



## 6.2. Spring

During the spring semester of 2014, each group members will have one compulsory subject of 10 ECTS that will have final exam in the first weeks of April. There will be approximately 18 weeks available for project work, because week 14 is Easter, and 15 is exam period. The project is to be delivered in week 20, and the last two weeks will be used for the final presentation. This gives a total of approximately 30 hours per student per week, which can be a challenge considering the additional compulsory subject.

## 6.3. System for time management

In order for the group to ensure that we have total control over our disposable time, we need a tool that can track the time. To enable us to succeed in reaching the principal objectives, the tool should support features as:

- User friendly - intuitive
- Tidy and visual for both user and Systems engineer
- Support extraction of time-data into at least M.S. Excel.
- Administration of a small-scale project from beginning to end.

In the process of finding a suitable tool, we discovered that there exist several suppliers of software solutions that claim to have revolutionized time-management. Paymo is time management software that aims to assist smaller projects like ourselves, in providing a complete solution to ensure constructive management of our time. In addition the latest version of the software offer features that appeal to us:

- Project management
- Task delegation, where the SE can administrate his resources in real time. Prioritize, deadline etc.
- Accessible interface, which provides visual data as well as the data, can be extracted to a spread-sheet like excel, Google drive, pdf, etc.
- Paymo APP, which enables the users to log time on the go with the freedom of their smart phones.

In addition the Paymo software offers an add-on for administration of invoicing, at this stage of the project we are not sure of the extent of our financial contribution, but we realize that we can benefit from the system that Paymo offers, so we can keep track of the economic aspect of the project. The system charges a monthly fee, 4.95\$/user, which sums up to an expense of 29.70\$.





KONGSBERG



## Project Plan

### KPEC

<b>Employer</b>	Kongsberg Defence & Aerospace			
<b>Group Members</b>	<b>Name</b>		<b>Initials</b>	
	Ola Pedersen Aasheim		OA	
	Salahuddin Asjad		SA	
	Hung Dinh		HD	
	Dler Hasan		DH	
	Even Gudbrandsen		EG	
	Jannik Schäffer		JS	
<b>Document information</b>	<b>Revision</b>	<b>Date</b>	<b>Approved</b>	<b>Pages</b>
	4.0	18.05.2015	HD	47





## Abstract

This is the second formative document, which is a continuation of the vision document. The overall purpose of this document is to enlighten on how the project group will work throughout the project to ensure that our goals are accomplished. After reading this document, one should be able to form a basic understanding of the routines we have defined for ensuring that the project is on-track and within the boundaries at all times.



## Revision Table

Version	Date	Approval	Description
1.0	08.01.2015	HD	<ul style="list-style-type: none"> <li>Created the document</li> </ul>
1.1	26.01.2015	EG	<ul style="list-style-type: none"> <li>Revised activity list</li> </ul>
2.0	22.01.2015	HD	<ul style="list-style-type: none"> <li>Revised and published</li> </ul>
2.1	03.02.2015	OA	<ul style="list-style-type: none"> <li>Defined iteration plan in time management section, and revised project model.</li> </ul>
2.2	09.03.2015	EG	<ul style="list-style-type: none"> <li>Revised activity list and time management (reschedule of project).</li> </ul>
3.0	16.03.2015	HD	<ul style="list-style-type: none"> <li>Revised and published</li> </ul>
3.1	20.04.2015	OA	<ul style="list-style-type: none"> <li>Revised activity list, and time management (reschedule of project, due to changes regarding external review with KDA)</li> </ul>
4.0	18.05.2015	EG	<ul style="list-style-type: none"> <li>Revised and published</li> </ul>

## Contents

Abstract .....	2
Revision Table .....	3
List of figures.....	6
List of tables .....	7
1. Introduction .....	8
2. Project scope .....	9
2.1. General description of assignment .....	9
2.1.1. Short-term goals .....	9
2.1.2. Long-term goals .....	9
2.2. Objectives .....	9
2.2.1. Group values and vision .....	10
2.2.2. Group objectives .....	10
2.2.3. Employers objectives .....	11
2.3. Deliverables .....	11
2.4. Boundaries.....	11



2.5.	Market need.....	12
3.	Organization.....	13
3.1.	Team members .....	13
3.2.	Communication.....	14
3.2.1.	Internal communication .....	14
3.2.2.	External communication .....	14
3.2.3.	Follow-up procedures .....	15
3.3.	Presentations.....	16
3.3.1.	First presentation .....	16
3.3.2.	Second presentation .....	16
3.3.3.	Final presentation .....	16
4.	Stakeholders .....	17
4.1.	Primary stakeholders.....	17
4.2.	Secondary stakeholders.....	17
4.3.	Key stakeholders .....	18
5.	Project model .....	19
5.1.	Phases during the project .....	19
5.1.1.	Project planning .....	19
5.1.2.	Concept of operations .....	20
5.1.3.	User & System requirements.....	20
5.1.4.	Detailed design.....	21
5.1.5.	Implementation .....	21
5.1.6.	Integration, test and verification .....	22
5.1.7.	Last phases of the project .....	22
5.2.	Document flow.....	22
6.	Time management.....	23
6.1.	Schedule .....	24
6.2.	Milestones.....	25
6.3.	Iterations .....	26
6.3.1.	Iteration plan.....	26
6.4.	Progress monitoring .....	27



6.5.	Detailed time estimation .....	28
7.	Financial management .....	31
7.1.	General information of the budget .....	31
7.2.	Financial Management life cycle .....	31
7.3.	Expenses in our project.....	32
7.4.	Estimation budget.....	33
7.5.	Recommendation .....	33
7.5.1.	Explanation of component choice .....	34
8.	Quality management .....	35
8.1.	General information of quality management .....	35
8.2.	Quality tool .....	36
8.2.1.	Analysis-synthesis-evaluation process.....	36
8.2.2.	How and where their implemented .....	37
8.2.3.	Reflection on the contribution of tools .....	37
8.3.	Documentation Templates .....	37
8.3.1.	Requirement Template .....	38
8.3.2.	Notification of Meeting Template.....	38
8.3.3.	Minutes Template.....	38
8.3.4.	Follow-Up Document.....	38
8.3.5.	Work Log.....	38
8.3.6.	Deviation Report.....	38
8.3.7.	Academic Report Template .....	38
8.4.	Group Contract .....	39
8.5.	Life-cycle Assessment.....	39
8.6.	Quality plan .....	40
8.6.1.	Identify the customers .....	40
8.6.2.	Determine customer needs .....	40
8.6.3.	Develop product features .....	41
8.6.4.	Develop process features.....	41
8.6.5.	Transition to operation.....	41
9.	Risk management.....	42



9.1. General information about risk .....	42
9.2. Risk analysis .....	42
9.2.1. Management.....	42
9.2.2. Resources .....	43
9.2.3. Project members .....	43
9.2.4. Software.....	43
9.2.5. Hardware.....	43
9.2.6. Implementation .....	43
9.2.7. Testing.....	44
9.2.8. Third-party involvement.....	44
9.3. Qualitative risk analysis .....	44
9.3.1. Probability .....	44
9.3.2. Impact .....	44
9.3.3. Risk matrix .....	45
9.4. Mitigation plan.....	45
Bibliography.....	47

## List of figures

Figure 1 - Cost, time and quality relationship. ....	11
Figure 2 – Marked need model .....	12
Figure 3 - Overview of stakeholders .....	18
Figure 4 - Iterative evolutionary agile model”.....	19
Figure 5 - Project document flow .....	22
Figure 6 - Project timeline .....	23
Figure 7 - GANNT-chart view .....	27
Figure 8 - Status update for project.....	27
Figure 9 - Financial Management life cycle.....	32
Figure 10 - 2-D Pie chart of our financial budget .....	33
Figure 11 - features to illustrate quality management. ....	35
Figure 12 - Analysis-synthesis-evaluation model. ....	36
Figure 13 - Quality plan.....	40



## List of tables

Table 1 - Group members and their responsibility areas .....	13
Table 2 - Overall assigned resources to different project phases.....	23
Table 3 - Scheduled progress .....	24
Table 4 - Milestone overview .....	25
Table 5 - Iteration overview.....	26
Table 6 - Detailed time estimation from Microsoft Project .....	28
Table 7 - Our estimated financial budget.....	33
Table 8 - Probability Analysis .....	44
Table 9 - Impact Analysis .....	44
Table 10 - Risk matrix .....	45
Table 11 - Color codes of Table 10.....	45
Table 12 - Mitigation codes of Table 13.....	45
Table 13 - Mitigation plan .....	46



## 1. Introduction

This is the second formative document for KPEC and it contains the project plan, which is supposed to give an overview on how the group is planning to approach and handle the task that is given.

The project plan includes goals and boundaries of the project so that we might meet the requirements of our employer and stakeholders. Additionally the project plan will also contain the different roles of each member and how the project group will communicate with each other, internal and external.

Moreover, the project plan will also include routines on how to handle problems during the project e.g. how to follow up on a hardware/software issue that might arise. Furthermore the project plan will also contain research of stakeholders, project model, time management system, financial management, quality management and risk management.

Broadly speaking the project plan is a document that will provide the group with a system on how and where to use their time and a strategy on how to approach the project using the project model as guidance.





## 2. Project scope

The project scope section defines the projects scope, which includes a description of the assignment as well as determining the objectives, deliverables, boundaries and the market need for the product.

### 2.1. General description of assignment

The project is given by KDA (Kongsberg Defence & Aerospace). The final goal, the long term goals of this assignment is to have a fully working ECU for an internal combustion engine that is based on a Xilinx Zynq circuit.

The assignment is composed of short-term goals and long-term goals. Because this project is supposed to last for many years, the short-term goals are the main objective for the project. However, the long-term goals must also be taken into account when designing schematics and software.

#### 2.1.1. Short-term goals

- Select a Micromodule that contains a Xilinx Zynq.
- Produce schematic for ECU circuitry, produced in Cadence Allegro.
- Consider the need of an operating system.
- Revision control of source code using GitHub.
- Establish communication between Zynq and tablet.

#### 2.1.2. Long-term goals

- Fully working ECU with following specifications:
- Tuning from tablet
- Low-level timing and communication toward external components, which gets processed by the FPGA.
- An option for using alternative fuel as propellant (Ethanol).
- Spark plugs with integrated pressure sensors to detect Peak Pressure Point (PPP)
- Automatic adjusting of the flow rate to the injectors, so that all injectors deliver the same amount of fuel.
- Automatic adjustment of fuel mapping.
- Option to reprogram flash using Ethernet/WIFI.
- Produce pneumatic valves that can be controlled by the ECU.
- Produce casing for ECU.
- Implementing a FPGA module to read a low speed ADC.

### 2.2. Objectives

To assure that the assignment given by KDA is completed successfully, we have defined some objectives and values that will assure that the assignment is accomplished according to KDA and the group's standards. We have also tried to define our employer's objectives for the assignment.



### 2.2.1. Group values and vision

The group values are essential to ensure that the group dynamic is optimal during the whole project. Our values motivate us to accomplish good results, and give us a pointer on how we should collaborate with others both internally and externally. Our values are Quality, Ambitions and Passion.

The group has a vision on delivering a high quality product that exceeds KDA's expectations, and to learn how ECU works thoroughly. We want to experience to work with an authentic project in a professional environment, and gain insights in how engineering projects develops.

We will accomplish our goals and visions by keeping the ambitions high throughout the project. This is accomplished by working seriously and systematic, as well as dedicating passion and time for the project. Our niche is to produce a seamless control system that is controlled wirelessly by an app with nice and user-friendly design.

### 2.2.2. Group objectives

- Find a solution to the project that exceeds employer requirements.
- Deliver a final result that satisfies our user and system requirements.
- Come up with smart and innovative ideas and solutions.
- Achieve a well-earned grade that satisfies all team members.
- Increase knowledge and skills within project work.
- Challenge the knowledge that has been acquired throughout the engineering course, and test it out practically.
- Deliver a high standard documentation to capture the viewers' attention with a unique design.
- Put in the amount of work that must be done to fulfill the tasks.
- Follow the templates and design structure from the project.
- Keep updated throughout the project development.
- Help each other out in difficulties.



### 2.2.3. Employers objectives

We have specified the employer objectives through what we mean are the most important ones. We have to keep in mind throughout the project that we are working for a large company, and we have to satisfy both the company and their requirements. Which are:

- Fully functioning Engine Control Unit (ECU).
- Build schematics for a motherboard with mounting capabilities for a FPGA micro module.
- Implementing an FPGA module to read a low speed ADC.
- Create demo software that adjusts the ECU from a tablet, (Android or IOS).
- Create an exciting interdisciplinary task.

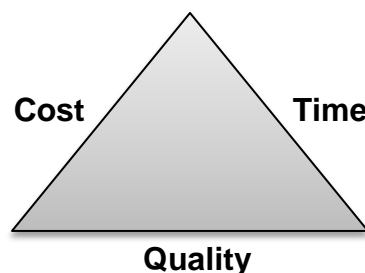
### 2.3. Deliverables

When the project is completed, KDA is expecting deliverance on schematics for the ECU circuitry; they also want software to view and adjust parameters in real time that can be monitored on a tablet.

### 2.4. Boundaries

The boundaries are the constraints that are associated with any project. These constraints are defined in Figure 1 as time, cost and quality. An example of a time constrain may be that the assignment given is too big, which is time consuming, therefore expensive and the total quality of the product will decrease. This in return will lead the project group into not being able to accomplish the assignment given. This is why the project group must focus equally between, time, cost and the quality of the product when designing, thus achieving a product that the project group may be proud of and will fulfill the requirements of the employer.

The time, cost and quality boundaries are defined in Chapter 6, 7 and 8. To handle the risks associated with the project, we have done a thoroughly risk analysis in Chapter 9. To mitigate the risks related to the project boundaries, we have defined a mitigation plan which is described in 9.4.



**Figure 1 - Cost, time and quality relationship.**



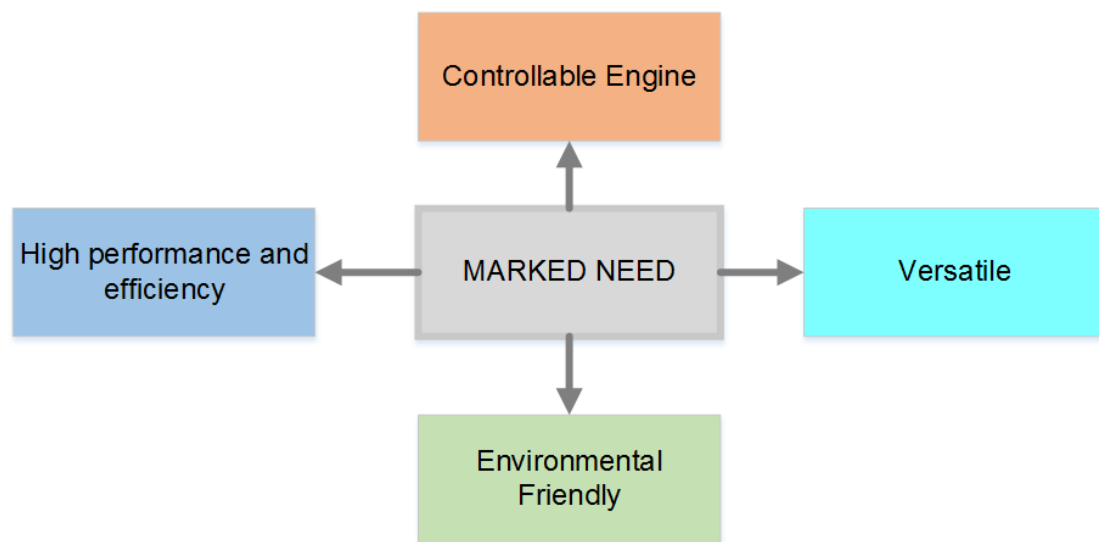
## 2.5. Market need

To ensure that the product is successfully designed and deployed in the market, we have defined the market need in Figure 2 as a part of Systems engineering processes.

By introducing this product to the market, we will be introducing innovations as Peak Pressure Point detection, which will provide better ignition, and if everything goes as planned, we will also be able to do mapping in real time, which will increase the overall effect of the motor.

If we compare our self with other ECU companies (MaxxECU, Megasquirt and Autronic), we can see that we will match their specification and applications easily, but a consequence to the extreme processing power, our ECU will be a lot more expensive than any of their high-end ECU. The main reasons for this are that we are using a Xilinx Zynq module.

From this we may draw the conclusion that this project, with the current market need may not be viable, but, is interesting for educational purposes, and will be an electro-technical study for students at HBV. As well as development, where KDA has assigned us the project where they wish that we develop their ECU with specifications, applications and requirements.



**Figure 2 – Marked need model**



### 3. Organization

The organization section gives an overview of the group members and their roles in the project. This section also contains information about the group routines regarding communication, follow-up procedures and presentations during the project.

#### 3.1. Team members

The group consists of four cybernetics-engineers and two embedded systems engineers, making this an interdisciplinary project. The group members has different backgrounds and interests, however all share dedication to technology. Table 1 gives an overview of the different group members and their responsibility areas.

**Table 1 - Group members and their responsibility areas**

Members	Roles and responsibility areas
<b>Ola P. Aasheim</b>	<b>Hardware design responsible.</b>
<b>ola@kpec.no</b>	<ul style="list-style-type: none"> <li>• Electronics design choices.</li> <li>• Hardware design document</li> </ul>
<b>Hung V. Dinh</b>	<b>Preliminary studies &amp; modeling responsible.</b>
<b>hung@kpec.no</b>	<ul style="list-style-type: none"> <li>• Concept of operations phase; FPGA / Internal combustion engines.</li> <li>• Mathematical models and simulations in MATLAB</li> <li>• Risk analysis.</li> <li>• Vision / concept of operations studies document.</li> </ul>
<b>Jannik Schäffer</b>	<b>Test &amp; Verification responsible.</b>
<b>jannik@kpec.no</b>	<ul style="list-style-type: none"> <li>• Test specification &amp; test plan.</li> <li>• Selection of sensors and actuators.</li> <li>• Testing strategy and equipment.</li> <li>• Test report and after-analysis document.</li> </ul>
<b>Dler Hasan</b>	<b>Marketing &amp; requirements responsible.</b>
<b>dler@kpec.no</b>	<ul style="list-style-type: none"> <li>• Completion of system and user requirements.</li> <li>• Creating and updating website for the group.</li> <li>• Presentation layouts and build-ups.</li> <li>• Requirements and test specification document.</li> </ul>
<b>Salahuddin Asjad</b>	<b>Software design &amp; interface responsible.</b>
<b>salahuddin@kpec.no</b>	<ul style="list-style-type: none"> <li>• High-level APP programming to PAD.</li> <li>• Low-level HW programming.</li> <li>• Interfaces between external components.</li> <li>• Software design document.</li> </ul>
<b>Even Gudbrandsen</b>	<b>Project manager &amp; quality responsible.</b>
<b>even@kpec.no</b>	<ul style="list-style-type: none"> <li>• Administrative tasks and follow-up procedures.</li> <li>• System responsible and quality control.</li> <li>• External communication.</li> <li>• Project plan document.</li> </ul>



## 3.2. Communication

Communication is important to ensure that every team member is on schedule at all times, and that misconceptions do not occur. Different communication channels will be used throughout the project for communicating internally or externally.

### 3.2.1. Internal communication

Internal communication will occur personally during the core-hours (09-14) at the office. Our main information channel for internal communication is through Facebook, where we have established a hidden group. The group members can post information and update on status of the project. Facebook Messenger provides instant user-friendly communication that works as a channel to distribute information, files and knowledge.

Common documents and presentations regarding the project will be shared through Google Drive as an organized, convenient and safe alternative to Facebook. When necessary, meetings are held on Monday mornings as board meetings. This type of meetings is equal to Scrum meetings, where all the participants are standing during the meeting to increase the efficiency. The systems engineer notifies the members on E-mail of the agenda, date and time.

### 3.2.2. External communication

External communication is communication pointed towards our stakeholders in the project. Our intention is to provide as much public information as possible during the project, in this way anybody with interest could follow us. Individual requests towards our stakeholders will occur by the project manager through e-mail. To update our stakeholders, we will provide following information regularly on our website; [www.kpec.no](http://www.kpec.no).

- Documentation of project phases.
- Description of group members
- Description of project.
- Weekly update on progress, written technically.



### **3.2.3. Follow-up procedures**

To be able to follow-up on the progress of the project and individual members of the group, the group has devised procedures to assure continuation of the project. These procedures are defined as:

- Weekly follow-up documents
- Weekly meetings
- Meetings with advisors (Internal and external)
- Internal procedures

#### **3.2.3.1. Weekly Follow-Up Documents**

This document will contain the current project status of each member; additionally the document will also include the current week workload and also next week workload. This document is also used to keep the internal advisor updated on the progress during the project.

#### **3.2.3.2. Weekly Meetings**

The group will be having weekly meetings on Monday mornings, so that everybody can be updated on the current status of the different task that has been done. This is also a place where the members can discuss and take up current issues and new issues.

#### **3.2.3.3. Meetings with Advisors**

The project group has two kinds of advisors, an internal from HBV and an external from KDA. There is going to be weekly meetings with the internal advisor, so that the advisor knows the current status and progress of the project group. External advisor meetings are done when KPEC need consultation or vice versa.

#### **3.2.3.4. Internal Follow-Up Procedures**

As a part of the project the project group recognizes that there will be a lot of issues that will arise during the project. This is why the group has made an internal follow-up procedure. In this procedure we are using Outlooks' own task handling system. This is actually just an e-mail that the group members can send to each other to notify that there is an issue that needs to be followed up. This system allows the project group to see if the issue is done, half-finished or done. Furthermore it also lets the group members see if further resources are needed to resolve the issue.



### 3.3. Presentations

There will be three presentations during the spring semester of 2015, the two first ones be divided into two parts.

- Public (max 20 minutes) presentation of the documents.
- Oral examination which is about 15 min. per. Student. The examination will include all related topics within the project.

The grade will be the evaluated result from all the presentations, there is also stated that everyone shall present something during the two first presentations.

#### 3.3.1. First presentation

First presentation ss a walkthrough of the following documents:

- Specifications
- Project plan
- Test plan

Everyone that's interested can participate in the presentation:

- Sensors both internal and external
- Supervisors both internal and external

#### 3.3.2. Second presentation

The second presentation is a walkthrough of the project, status and remaining activities.

Everyone that's interested can participate in the presentation:

- Sensors both internal and external
- Supervisors both internal and external

#### 3.3.3. Final presentation

The final presentation has duration of 1 hour, containing:

- 20 minutes sale of the product, which reviews how great the product is.
- 20 minutes technical presentation, time consumption, technical solutions etc.
- 20 minutes for questioning of the whole group united.

This will happen immediately in chronological order, and the participants are the whole group, including sensors and supervisors. Then there will be an oral examination where there is an opening for questions if necessary, every topic is applicable.





## 4. Stakeholders

Stakeholders are those who may be affected by or have an effect in the KPEC project. We can characterize stakeholders by primary- and secondary stakeholders. It is a good idea to identify and understand stakeholders, because it will first of all put more ideas on the table. But it will also increase the credibility and understandability of their interests.

### 4.1. Primary stakeholders

Primary stakeholders are those who are directly affected by the KPEC project.

- Buskerud and Vestfold University College (HBV) is one of our main stakeholder. Our school will provide with a group room for us where we can sit together and work with our project. We will also communicate with an internal advisor from HBV.
- Kongsberg Defence & Aerospace (KDA) is also one of our main stakeholder. They are providing with tools and equipment we need during our project development. Our external advisor will be from KDA, which means that there will be weekly communication between us.
- We from the KPEC team are of course stakeholders in this project.
- The examiners and advisors will also have an impact on this project. The advisors will for example bring new ideas we can implement to our project, while the examiners will evaluate our project during this semester.

### 4.2. Secondary stakeholders

Secondary stakeholders are those who have indirect relations to this project.

- We are going to develop KPEC product for Kongsberg Defence & Aerospace, but the user of the product does not need to have any relation to KDA.
- If KDA want to sell the product further to other customers, they will also become an important stakeholder.
- The KPEC product will also need maintenance during its life cycle. Therefore, maintenance team needs and worries need to be considered.
- We will also count Trenz Electronic as a stakeholder, because we will use their developed Zynq module. They also have a GitHub account where we can use some of their developed software. Therefore, they will affect our project in some ways.



### 4.3. Key stakeholders

A key stakeholder is a stakeholder that has something to gain in KPEC's success. KPEC's key stakeholders have been defined as HBV and KDA.

KPEC have done some research and identified the following stakeholders as seen in Figure 3. The figure illustrates KPEC and their relation with some of the few identified stakeholders.

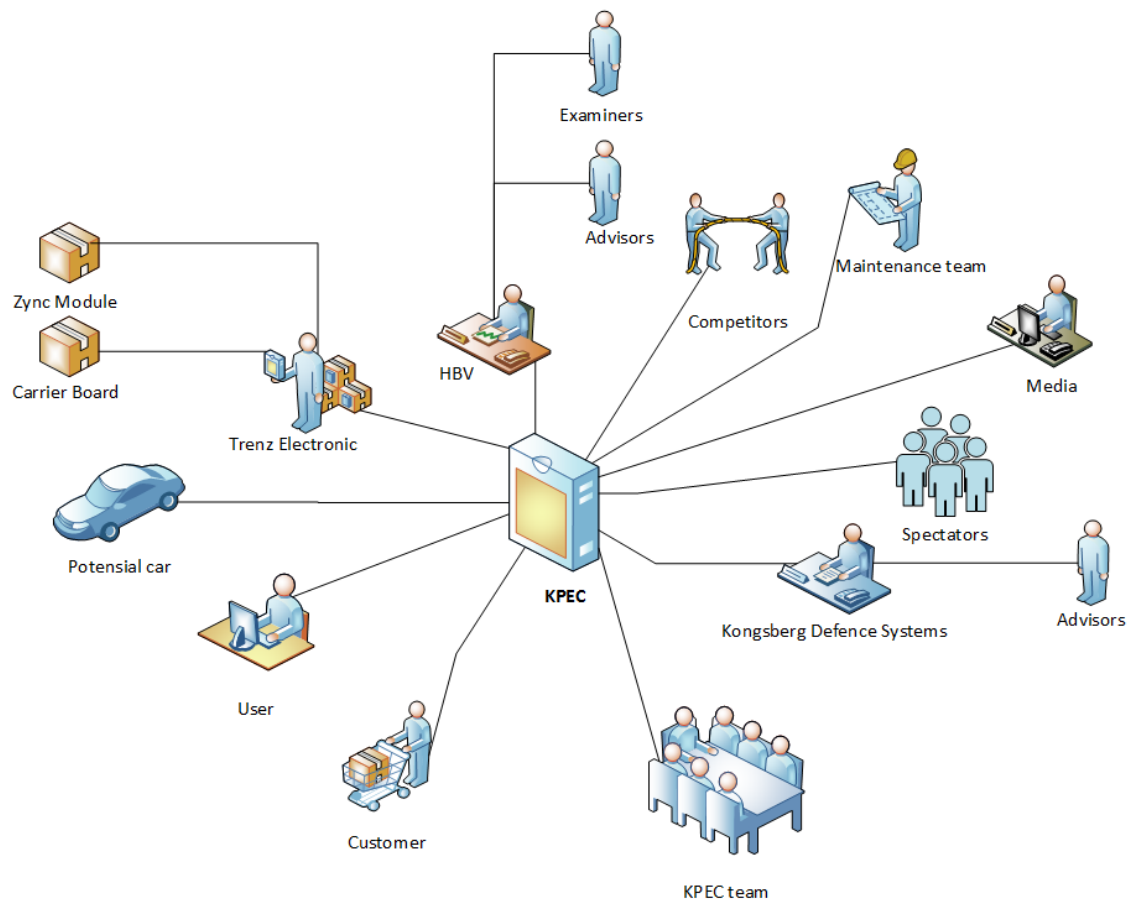


Figure 3 - Overview of stakeholders



## 5. Project model

We have selected an iterative evolutionary agile model (hereafter our model) that is based on a V-model in use for our project as seen in Figure 4. As described in the vision document, our model uses an iterative approach with iterations during the project to gain understanding of the product and to add functionality. The kernel of our model is iterative, and the project phases are performed stepwise. This implies that the first iteration is longer than the rest of the iterations, due to the fact that a concept phase, requirements phase and preliminary design phase must be completed before the design iteration could be performed.

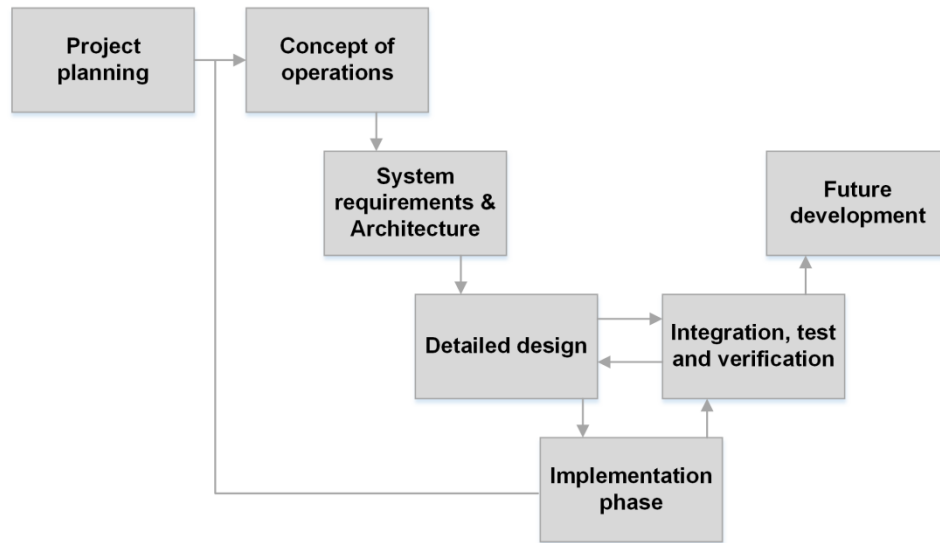


Figure 4 - Iterative evolutionary agile model"

### 5.1. Phases during the project

This section will describe each phase of the project and the documents that are generated in each phase. In our project we will focus on the design iterations, as this is the development phase and we will not have sufficient time to produce and fully test the system. The next stages are therefore emulated by the "Future development" box.

#### 5.1.1. Project planning

The project planning phase is the first phase of the project. In this phase, it is important to recognize who we are as a group, what we actually want to achieve and the ambitions we have for our own learning's sake and the end result. This phase begins with the creation of a vision document, which is an early assessment of our task and how we wish to address it. The vision document is the initial document that gives an overview and general idea of the scope of our project and what must be accomplished to achieve our goals for the project.



Another part of project planning is to establish a documentation standard and template for how the project shall be documented. Other templates such as presentation template, notification of meeting, minutes, aberration reports and weekly follow up documents were also produced in advance in order to ensure that every document follows the same standard from the beginning of the project and thus saves time when concatenating documents.

Perhaps the most important part of the project planning phase is the creation of this document, the project plan. The project plan is the main document describing how we will manage and work during our project. The project plan covers estimates of important quantities such as time and financial management. The procedures and methodology that we will use during the project is described in the project plan.

### 5.1.2. Concept of operations

The concept of operations phase is the first phase where the system designed is looked upon from a *zoom out* perspective. The general idea of this phase is to grant an understanding of the system as a whole and how it interacts with its users and its environment. Design choices are *not* made during this phase and ideally no specific limitations to the finished system is set. Systems engineering tools are the main focus of this phase in use for understanding the task.

As a part of understanding the task and system which is designed, a preliminary study is performed and documented with a following report. The preliminary studies is performed to gather required background information and to uncover new fields of studies which require further analysis in order to be able to design the system. Our group performed preliminary studies by delegating different fields of study between our group members to investigate further. The topics were divided between the members of our group based on existing knowledge in relevant subjects, as the group has members with a vast knowledge of different subjects ranging from different backgrounds.

### 5.1.3. User & System requirements

At this point in the project, the group members shall have a clear insight in *what* we are creating and using this knowledge, be able to establish the requirements for the system. Some requirements will be derived directly from the task given by our employer, KDA. The requirements given by the employer will usually be system requirements that are absolute for our system and difficult to change without negotiation. This phase still perceives the system from a *zoom out* perspective, meaning that no direct design choices are to be established by the requirements set, thus leaving room for as many different solutions as possible in the finished detailed design.

In this phase, the Requirement and Test specification document is created. A template for the identification and how the requirements shall be written is created in advance



with a dedicated guideline document to ensure that all requirements are written with a given set of rules. The reason for strict guidelines when writing the requirements is that the requirement specification is the most important document regarding design choices that may limit system functionality. Traceability in the requirements is absolute key, so therefore a unique identifier is assigned each requirement that reveals what department is responsible, the importance level that the requirement is met and the originating person who first created the requirement.

The Test specification is the most important document for when the system is designed and first produced and has to be tested, verified and validated in the future phases of the project. The test specification is written with a direct link to the Requirements specification in order to be able to verify and validate that the requirements are actually met.

#### 5.1.4. Detailed design

The detailed design phase is the first stage of development where work is performed on much lower level of perspective than earlier. In this stage, the main design work and prototyping is performed based on the system and user requirements set in the requirements specification<sup>1</sup>. As the requirements shall be written in order not to subdue any possible solutions, the actual design choices are made in this phase.

In terms of our project with electronics and software design, the detailed design phase will be performing part searches for selecting parts based on simulations and research. The electronic design process is to find actual components to be used and drawing schematics. For software will this phase be where the actual coding takes place, according to the requirements.

The Design document is generated in this phase and is a large document that justifies and describes every design choice made during the detailed design phase. As our project will likely continue over several years with involvement of many different groups, this documents serves as a very important reference in regards to allow later groups understand our design choices, possible modifications and considerations that must be made.

#### 5.1.5. Implementation

The implementation phase is where the partial designs performed during the iterations are collaborated into a complete schematic that fulfills the requirements. This is performed during multiple iterations where the scope of the project is narrowed, and functionality is added.

For the hardware part, the design process starts at concept of operations with understanding of what that is to be designed. It follows then by the requirements phase

---

<sup>1</sup> KPEC 2014, Gudbrandsen, Pedersen, Schäffer, Dinh, Hasan, Asjad, Requirement specification (Rev 4.0), HBV



where the detailed requirements for the hardware components are determined as target values for the design phase. During the design phase, components are selected and verified in the test and verification phase, and then implemented in the schematic design.

The process described in the last section is equivalent for software, except for the fact that they are designing code that they also test iteratively. This design processes can span typically from 2 hours to one week depending on the difficulty level of the task. The iterations performed for the product has a nominal length of typically two weeks (see iteration plan), and is documented with an iteration report.

#### 5.1.6. Integration, test and verification

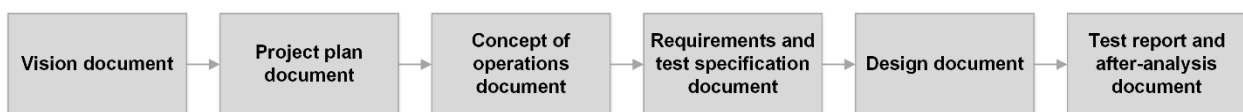
The testing phase of the project model consists of two types of testing; detail design testing (individual) and static/dynamic system testing that tests the completed product. Software and hardware will have different testing strategies due to the fact that they have two different strategies to solve technical problems. For software, testing is performed continuously throughout the project, but for hardware, testing is performed as simulations. When the project is completed, a FAT should be performed to verify that all the requirements are met.

#### 5.1.7. Last phases of the project

As the project is no longer at our hands at the time these phases would be performed, what we can do in advance for planning these is limited. We will however write a test report for software testing we have performed and a full after-analysis document about the project. As mentioned previously, the project is likely to be handed over to new students as their thesis in order to continue work towards a complete working system. Therefore, it is very important that we form documentation to create a platform as good as possible for our successors of the project.

### 5.2. Document flow

Figure 5 shows the overall document flow in our project. The documents are created chronologically across the different phases of development.



**Figure 5 - Project document flow**



## 6. Time management

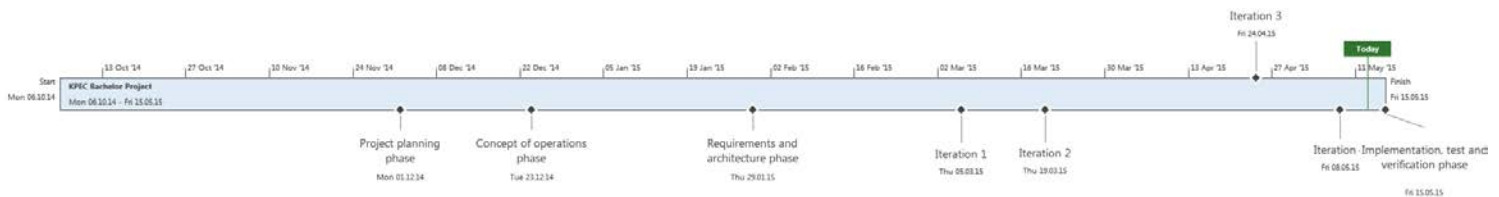
The projects duration is limited from January to June 2015. Bachelor thesis executed at HBV, has a weight of 20 ECTS. The EU-standard time estimation for 1 ECTS is defined to be between 25-30 hours depending on ambition level of the student. The bachelor thesis will therefore contain approximately 650 hours of project work per student. The total resources available in the group are therefore 3940 hours as stated in Table 2.

The project is following the project model during the project, and resources are assigned according to the overall importance of the project phase. By following the systems engineering discipline, it is important to establish a good project plan and understand the customer thoroughly. The project starts in autumn with planning phase and concept of operations, and follows with requirements and detailed design during spring semester. The schedule assumes 8-hour working days throughout the project, five days per week.

**Table 2 - Overall assigned resources to different project phases.**

Semester	Project phase	Percentage	Hours
Autumn	Project planning phase	9%	360
Autumn	Concept of operations phase	5%	180
Spring	System requirements and architecture phase	9%	360
Spring	Detailed design HW & SW phase / iterations	57%	2260
Spring	Integration, test and verification phase	5%	180
	Administrative	15%	600
<b>Sum</b>		<b>100%</b>	<b>3940</b>

Microsoft Project Professional is applied to ensure accuracy of time estimation, which is an advanced GANTT-chart with time tracking and baseline project management features. The timeline in Figure 6 gives an overview of start and finish dates as well as milestones. The detailed design phase is the phase where the most of our resources are applied, giving a total of 2260 hours.



**Figure 6 - Project timeline**



## 6.1. Schedule

**Table 3 - Scheduled progress**

Week	Project phase	Activities	Hours
41-43	Startup	<ul style="list-style-type: none"> <li>• Creation of group contract</li> <li>• Analyze assignment</li> <li>• Generate group name, logo and webpage</li> <li>• Generate vision document</li> <li>• Generate GANNT-chart</li> <li>• Create document templates</li> </ul>	180
44-48	Project planning	<ul style="list-style-type: none"> <li>• Creation of project plan document</li> </ul>	180
49-50	Exam preparations	<ul style="list-style-type: none"> <li>• No project work</li> </ul>	0
51	Concept of operations	Concept of operations document and research regarding: <ul style="list-style-type: none"> <li>• FPGA-technology</li> <li>• Internal combustion engine</li> <li>• Engine management system</li> <li>• Zynq micromodule</li> <li>• PAD-programming</li> </ul>	180
52-1	Holidays	<ul style="list-style-type: none"> <li>• No project work</li> </ul>	0
2-5	System requirements and architecture	Writing requirements & test specifications for <ul style="list-style-type: none"> <li>• System</li> <li>• Hardware</li> <li>• Software</li> <li>• Defining system architecture</li> <li>• Generate first presentation</li> </ul>	360
6-13	Detailed design	<ul style="list-style-type: none"> <li>• First iteration – design outline</li> <li>• Second iteration</li> <li>• Generate second presentation</li> <li>• Third iteration</li> </ul>	1500
14	Easter	<ul style="list-style-type: none"> <li>• Easter holiday Thursday and Friday</li> </ul>	100
15	Exam preparations	<ul style="list-style-type: none"> <li>• No project work</li> </ul>	0
16-17	Detailed design	<ul style="list-style-type: none"> <li>• Third iteration</li> <li>• Review meeting with KDA</li> <li>• Fourth iteration</li> </ul>	360
18-19	Integration, test and verification	<ul style="list-style-type: none"> <li>• Completion of software and hardware</li> <li>• Completion of design document</li> <li>• Testing and verification of requirements</li> <li>• Completion of test report and after analysis</li> </ul>	360
20	Project completion	<ul style="list-style-type: none"> <li>• Generate third presentation</li> <li>• Generate poster</li> <li>• Project completion</li> </ul>	120





## 6.2. Milestones

Milestones occur when one phase of the project model is completed and gives motivation for propulsion in the project. Table 4 gives an overview of milestones according to time estimation automatically generated in Microsoft Project.

**Table 4 - Milestone overview**

Date	Milestone
06.10.14	<b>Project startup</b> <ul style="list-style-type: none"> <li>Selection of assignment and startup of project.</li> </ul>
01.12.14	<b>Project planning phase completed</b> <ul style="list-style-type: none"> <li>Vision document and project plan completed.</li> </ul>
23.12.14	<b>Concept of operations phase completed</b> <ul style="list-style-type: none"> <li>Preliminary studies document completed.</li> </ul>
30.01.15	<b>System requirements and architecture phase completed</b> <ul style="list-style-type: none"> <li>Requirements specification and test specification completed</li> <li>First presentation completed.</li> </ul>
06.03.15	<b>First iteration completed – Detail design</b> <ul style="list-style-type: none"> <li>Design outline completed</li> </ul>
20.03.15	<b>Second iteration completed – Detail design</b> <ul style="list-style-type: none"> <li>Functionality added to design outline</li> <li>Second presentation completed</li> </ul>
17.04.15	<b>Third iteration completed – Detail design</b> <ul style="list-style-type: none"> <li>Functionality added to complete design.</li> <li>Review meeting performed with customer</li> </ul>
01.05.15	<b>Fourth iteration completed – Detail design</b> <ul style="list-style-type: none"> <li>Revision of design according to customer demands.</li> </ul>
08.05.15	<b>Integration, test and verification phase completed</b> <ul style="list-style-type: none"> <li>Completion of schematics and software design.</li> <li>Completion of design document.</li> <li>Test that schematic and software are according to requirements.</li> <li>Test report and after analysis document completed</li> <li>Hand-in of project documentation</li> </ul>
15.05.15	<b>Project completion</b> <ul style="list-style-type: none"> <li>Third presentation completed</li> <li>Advertising poster completed</li> </ul>



### 6.3. Iterations

The KPEC group is following an iterative project model and therefore, a plan regarding the design iterations must be defined. Our adapted version of the project model is an agile type, and these project models are focusing on customer satisfaction and quality. This is due to the fact that the customer objectives are changing as well as the understanding of the problem during the development of the product.

We will have multiple iterations during detailed design, where we will be evaluating the design internally or externally to our stakeholders. This is a system validation where the design is confirmed by the stakeholders to meet their requirements, and the overall quality of the system will therefore increase.

When the iterations are completed, an iteration report is generated to document the progress and deliverables for next iteration. The project is then moving towards implementation where the hardware and detail design are implemented to a finished product. Table 5 gives an overview of the iteration sequence.

#### 6.3.1. Iteration plan

**Table 5 - Iteration overview**

Phase	Week	Description	Deadline
<b>Iteration 1</b>	6-10	Generate first outline of detailed design on hardware and software. Intern review meeting with a walkthrough of requirements, software and hardware design.	05.03.15 Intern review
<b>Iteration 2</b>	11-12	Review of outline design, further development of functionality. Second presentation will be held during this iteration.	19.03.15 Presentation
<b>Iteration 3</b>	13-16	Review of second iteration, correction according to critics received at second presentation. Further development of hardware and software design. Review meeting with our customer.	16.04.15 Review meeting with customer.
<b>Iteration 4</b>	17-18	Review of third iteration, correction of design according to customer demands.	01.05.15 Intern review



## 6.4. Progress monitoring

GANNT-chart view is used to graphically monitor the progress. The project is automatically updated when tasks are marked as completed in Microsoft Project. A section of the project is viewed in Figure 7 with project phases and milestones, as well as assigned resources and priority.

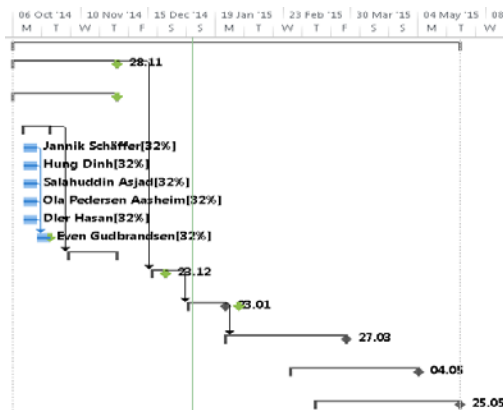
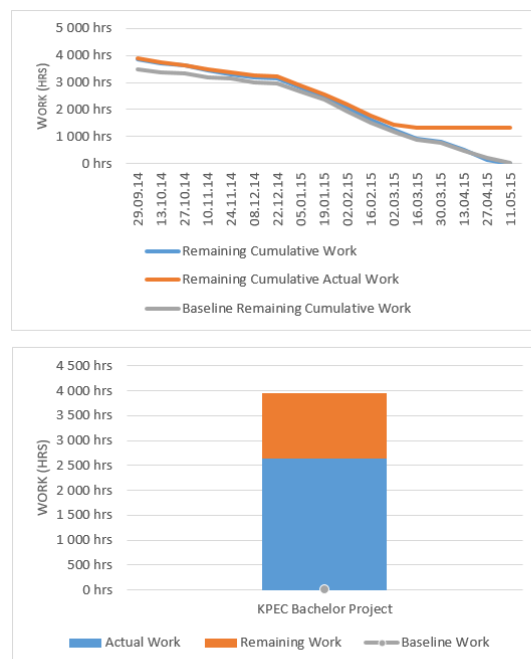


Figure 7 - GANTT-chart view

Automatic report function in Microsoft project will be used to control propulsion of the project, and will be presented to our stakeholders during presentations. This report gives an overview of work burndown, remaining and actual work performed. Figure 8 shows the status of the project.



[Try setting a baseline](#)

% Work Complete

67%

Remaining Work

1 317,7 hrs

Actual Work

2 628,8 hrs

## WORK OVERVIEW

Figure 8 - Status update for project



## 6.5. Detailed time estimation

Table 6 - Detailed time estimation from Microsoft Project

WBS	Task Name	Work	Finish
1.	<b>KPEC Bachelor Project</b>	<b>4 022,5 hrs</b>	<b>Fri 15.05.15</b>
1.1.	<b>Administrative tasks</b>	<b>674,5 hrs</b>	<b>Thu 14.05.15</b>
1.1.1.	General research	90 hrs	Wed 06.05.15
1.1.2.	Documentation templates	52,5 hrs	Fri 17.10.14
1.1.3.	Communication / Documentation	65 hrs	Thu 14.05.15
1.1.4.	Communication / follow-up	65 hrs	Mon 20.04.15
1.1.5.	Meeting intern	72 hrs	Wed 14.01.15
1.1.6.	Meeting extern	30 hrs	Thu 08.01.15
1.1.7.	Webpage update	55 hrs	Thu 23.10.14
1.1.8.	Internal control	25 hrs	Tue 17.02.15
1.1.9.	Presentation work	80 hrs	Fri 27.03.15
1.1.10.	PAYMO / Microsoft Project	60 hrs	Wed 17.12.14
1.1.11.	Revision of project plan	30 hrs	Mon 23.02.15
1.1.12.	Practical tasks	50 hrs	Thu 14.05.15
1.2.	<b>Project planning phase</b>	<b>374 hrs</b>	<b>Mon 01.12.14</b>
1.2.1.	<b>Vision document</b>	<b>72 hrs</b>	<b>Fri 24.10.14</b>
1.2.1.1	Organization	12 hrs	Fri 17.10.14
1.2.1.2	Assignment	12 hrs	Fri 17.10.14
1.2.1.3	Objectives	12 hrs	Fri 17.10.14
1.2.1.4	Project model	12 hrs	Fri 17.10.14
1.2.1.5	Time management	12 hrs	Fri 17.10.14
1.2.1.6	Completion	12 hrs	Fri 24.10.14
1.2.2.	<b>Project plan document</b>	<b>252 hrs</b>	<b>Mon 01.12.14</b>
1.2.2.1	Project scope	12 hrs	Fri 21.11.14
1.2.2.2	Organization	12 hrs	Fri 21.11.14
1.2.2.3	Stakeholders	24 hrs	Fri 28.11.14
1.2.2.4	Project model	24 hrs	Fri 21.11.14
1.2.2.5	Time management	36 hrs	Fri 28.11.14
1.2.2.6	Financial management	24 hrs	Fri 28.11.14
1.2.2.7	Risk management	36 hrs	Fri 28.11.14
1.2.2.8	Quality management	36 hrs	Mon 24.11.14
1.2.2.9	Completion	48 hrs	Mon 01.12.14
1.2.3.	Revision of vision document	50 hrs	Wed 05.11.14
1.3.	<b>Concept of operations phase</b>	<b>174 hrs</b>	<b>Tue 23.12.14</b>
1.3.1.	<b>Concept of operations document</b>	<b>126 hrs</b>	<b>Tue 23.12.14</b>
1.3.1.1	FPGA-technology	24 hrs	Fri 19.12.14
1.3.1.2	Internal Combustion engine	24 hrs	Fri 19.12.14
1.3.1.3	Engine management system	36 hrs	Thu 18.12.14
1.3.1.4	Zynq micromodule (LL)	24 hrs	Fri 19.12.14
1.3.1.5	PAD-programming (HL)	18 hrs	Tue 23.12.14
1.3.2.	<b>System engineering - exploration</b>	<b>36 hrs</b>	<b>Tue 23.12.14</b>



1.3.2.1	Defining market need	4,8 hrs	Mon 22.12.14
1.3.2.2	Customer key driver	4,8 hrs	Mon 22.12.14
1.3.2.3	Concept of operations	4,8 hrs	Mon 22.12.14
1.3.2.4	Stakeholders and their requirements	4,8 hrs	Mon 22.12.14
1.3.2.5	Lifecycle needs and concerns	4,8 hrs	Mon 22.12.14
1.3.2.6	Completion	12 hrs	Tue 23.12.14
1.3.3.	Revision of concept of operations document	12 hrs	Wed 03.12.14
1.4.	<b>Requirements and architecture phase</b>	<b>360 hrs</b>	<b>Thu 29.01.15</b>
1.4.1.	<b>Requirements specification document</b>	<b>150 hrs</b>	<b>Fri 16.01.15</b>
1.4.1.1	Define hardware requirements	50 hrs	Fri 09.01.15
1.4.1.2	Define software requirements	50 hrs	Fri 09.01.15
1.4.1.3	Define system Requirements	50 hrs	Fri 16.01.15
1.4.2.	<b>Test specification document</b>	<b>125 hrs</b>	<b>Thu 22.01.15</b>
1.4.2.1	Define hardware test specification	50 hrs	Thu 22.01.15
1.4.2.2	Define software test specification	50 hrs	Thu 22.01.15
1.4.2.3	Define system test specification	25 hrs	Tue 20.01.15
1.4.3.	<b>Systems engineering - definition</b>	<b>40 hrs</b>	<b>Mon 26.01.15</b>
1.4.3.2	Use case diagram	8 hrs	Mon 26.01.15
1.4.3.3	System context diagram	8 hrs	Wed 07.01.15
1.4.3.4	Technical budgets	8 hrs	Tue 06.01.15
1.4.3.5	Functional diagrams	8 hrs	Mon 12.01.15
1.4.3.6	System Breakdown Structure	8 hrs	Wed 21.01.15
1.4.4.	Revision of documents	45 hrs	Thu 29.01.15
1.5.	<b>Detailed design HW &amp; SW phase</b>	<b>2 260 hrs</b>	<b>Fri 08.05.15</b>
1.5.1.	Component research & selection	100 hrs	Thu 29.01.15
1.5.2.	<b>Iteration 1</b>	<b>900 hrs</b>	<b>Thu 05.03.15</b>
1.5.2.1	<b>1.1 Hardware</b>	<b>600 hrs</b>	<b>Thu 05.03.15</b>
1.5.2.1.1	Week 6: Input circuits	120 hrs	Thu 05.02.15
1.5.2.1.2	Week 7: ADC	120 hrs	Thu 12.02.15
1.5.2.1.3	Week 8: Output circuits	120 hrs	Thu 19.02.15
1.5.2.1.4	Week 9: Filter design	120 hrs	Thu 26.02.15
1.5.2.1.5	Week 10: Circuit simulations	120 hrs	Thu 05.03.15
1.5.2.2	<b>2.1 Software</b>	<b>300 hrs</b>	<b>Tue 03.03.15</b>
1.5.2.2.1	Week 6: Petalinux install/config	60 hrs	Tue 03.02.15
1.5.2.2.2	Week 7: Communication Zynq/tablet	60 hrs	Tue 10.02.15
1.5.2.2.3	Week 8: Database structure	60 hrs	Tue 17.02.15
1.5.2.2.4	Week 9: Real time communication	60 hrs	Tue 24.02.15
1.5.2.2.5	Week 10: Database design	60 hrs	Tue 03.03.15
1.5.3.	<b>Iteration 2</b>	<b>360 hrs</b>	<b>Thu 19.03.15</b>



1.5.3.1	<b>1.2 Hardware</b>	<b>240 hrs</b>	<b>Thu 19.03.15</b>
1.5.3.1.1	Week 11: Completion of presentation and documentation	120 hrs	Thu 12.03.15
1.5.3.1.2	Week 12: Intern communication & inputs	120 hrs	Thu 19.03.15
1.5.3.2	<b>2.2 Software</b>	<b>120 hrs</b>	<b>Thu 19.03.15</b>
1.5.3.2.1	Week 11: Completion of presentation and documentation	60 hrs	Thu 12.03.15
1.5.3.2.2	Week 12: Tablet application structure	60 hrs	Thu 19.03.15
1.5.4.	<b>Iteration 3</b>	<b>540 hrs</b>	<b>Fri 24.04.15</b>
1.5.4.1	<b>1.3 Hardware</b>	<b>360 hrs</b>	<b>Fri 24.04.15</b>
1.5.4.1.1	Week 13/14: Power design 1	120 hrs	Mon 30.03.15
1.5.4.1.2	Week 16: Power design 2	120 hrs	Thu 16.04.15
1.5.4.1.3	Week 17: Power design & protection circuitry	120 hrs	Fri 24.04.15
1.5.4.2	<b>2.3 Software</b>	<b>180 hrs</b>	<b>Mon 20.04.15</b>
1.5.4.2.1	Week 13/14: Tablet application design 1	60 hrs	Thu 26.03.15
1.5.4.2.2	Week 16: Tablet application design 2	60 hrs	Mon 13.04.15
1.5.4.2.3	Week 17: Tablet application design 3	60 hrs	Mon 20.04.15
1.5.5.	<b>Iteration 4</b>	<b>360 hrs</b>	<b>Fri 08.05.15</b>
1.5.5.1	<b>1.4 Hardware</b>	<b>240 hrs</b>	<b>Fri 08.05.15</b>
1.5.5.1.1	Week 18: Revision of schematic design	120 hrs	Fri 01.05.15
1.5.5.1.2	Week 19: Completion of schematics & design documentation	120 hrs	Fri 08.05.15
1.5.5.2	<b>2.4 Software</b>	<b>120 hrs</b>	<b>Fri 08.05.15</b>
1.5.5.2.1	Week 18: Revision of software design	60 hrs	Fri 01.05.15
1.5.5.2.2	Week 19: Completion of software & design documentation	60 hrs	Fri 08.05.15
1.6.	<b>Implementation, test and verification phase</b>	<b>180 hrs</b>	<b>Fri 15.05.15</b>
1.6.1.	Week 20: Test report and after-analysis document	90 hrs	Tue 12.05.15
1.6.2.	Week 20: Project completion	90 hrs	Fri 15.05.15



## 7. Financial management

### 7.1. General information of the budget

Leading the project and creating a good working atmosphere is the essence of good management. Spending, budgeting and cost estimation is vital to create an abstract overview for the company. The costs are obviously affected by the results of the systems engineering work. Therefore it's important to have clear ground to work up from, in other words a budget.

After our first meeting with the company, we agreed to that the company will be responsible for most of the expenses in this project. However, our team is willing to pay 5000 NOK each, if KDA don't cover some of our needed equipment or components. We've got the freedom when it comes to selection of some of components and equipment we are going to use. Therefore, we need to make an estimation of budget, so KDA know roughly how much money they need to spend on this project. And because KDA has given us the possibility to choose some of the components and modules, we need to have the budget clear as soon as possible.

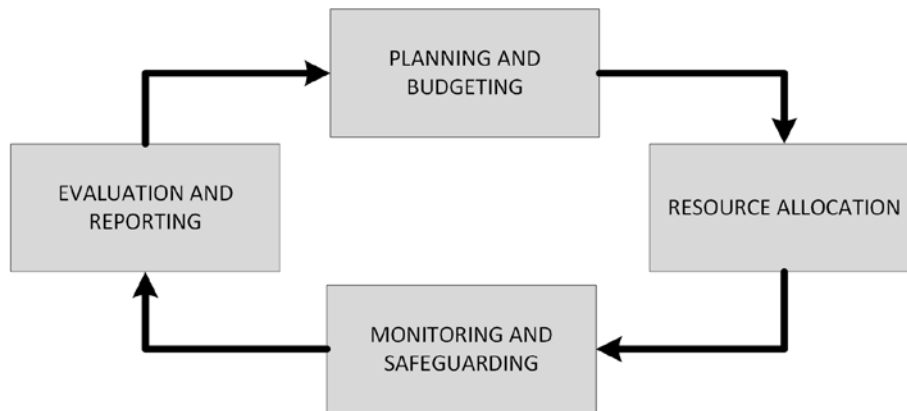
By experience, we know that there are often some hidden expenses that we don't count in the first round. Because we have not yet received the exact price of components and modules, there are some uncertainties concerning the expected expenses in this project. Therefore, we expect that there will be some changes, whenever we know more about the prices.

### 7.2. Financial Management life cycle

Planning and budgeting follows some important steps seen in Figure 9. A budget shows how much it will cost, reflects policy choices and above all provides the framework and boundaries for this project. A well-applied budget framework provides a guideline for future allocations, and will also limit the need to work with reduced budgets.

Resource allocation is the next phase of financial management life cycle. When we are done with the budget, KDA will receive the list of components and tools we will need during the project. Since KDA is our resource in KPEC financials, and since some components we will use during the project are expensive, we need to priorities.

Monitoring and safeguarding is also an important phase that reflects the work and activities that are done during the project development. It is important to have regular internal procedures, to ensure that the financial status is under control. During the project, there is a possibility that we have to face some unexpected economic problems. In addition, since most part of our economical expenses is covered by KDA, we need to be aware of different causes that can give us unexpected problems.



**Figure 9 - Financial Management life cycle**

Evaluation and reporting is the last part of the financial life cycle where we need to evaluate and analyze our work and the project itself. Since there are a many different types of engine control units out in the real world, we will compare our project against other similar products. That will for example help us to see if our work is costing too much and achieving too little.

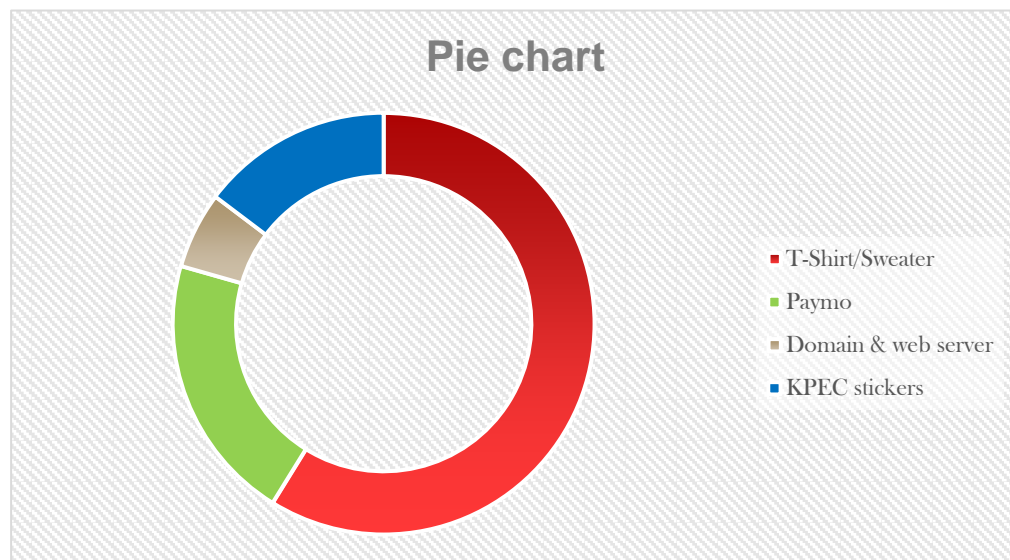
### **7.3. Expenses in our project**

We want an efficient and effective management of money. By looking at the 2-D Pie chart below, Figure 10, we can clearly identify the largest expenses. Both our computer and electrical engineering team need tools to develop the idea and satisfy the company. In order to progress throughout our project in a sparse manner, we must take into account what the price of each tool will cost. After some research and calculations, the group found the expenses for our project mentioned below.





## 7.4. Estimation budget



**Figure 10 - 2-D Pie chart of our financial budget**

Table 7 below is our current budget, notice that we will throughout the project make a more accurate budget with more precise prices. Meanwhile we have to take in account that it is important to have something to look at while our group is continuing our Systems Engineering approach.

**Table 7 - Our estimated financial budget**

Product	Amount	Total cost (NOK)
Paymo 3	6	1400 kr
Web server & domain	1	400 kr
T- shirts / Sweater	18	4000 kr
KPEC stickers	400	1000 kr
Presentation material	3	1200 kr
		<b>Total costs: <u>8000kr</u></b>

## 7.5. Recommendation

When we had our first meeting with KDA, we got some inputs of how they want the product to be, and because the final product is going to be developed in some years, the component choice is quite critical and important. We have taken into account that this project is going to be developed further after we are finished with our thesis. The components and tools we will use will then have the capacity and availability to be developed further when it is needed in the future.



### 7.5.1. Explanation of component choice

After some research and discussion, we have found some components and products that will match our needs. Some of the choices are still uncertain, so there can be some changes later.

- **Trenz Electronic TE0720 Zynq SoC** – This one was one of many choices we had. The reason that we decided to go for this module is that the chip contains everything necessary to create a Linux, Windows or Android based design. In addition, there is a wide range of reference documents about different parts of the Zynq module.
- **Trenz Electronic TE0703 Carrier board** – The TE0720 Zynq Soc will be stacked on this carrier board. In that way, all necessary inputs and output pins will be available. The carrier board also have DIP switch for choosing operational mode, JTAG & UART connection, Ethernet connection, SD-card slot and so on.
- **Printed circuit boards/components** – we have added ten exemplars of the printed circuit boards to the budget.
- **Android Tablet** – We have decided that we want to make an Android app to control the Engine Control Unit. Therefore, we need an Android tablet to communicate with the system. For that case, we want to use Samsung Galaxy Tab S 10.5". The tablet has a large and clear screen with good battery life and octa core processors.
- **GitHub** – Our team, especially the Computer Engineering students need a tool to do version control of the codes. Initially the group has decided that every member of the group will have their own account to GitHub.
- **Paymo** – Our team also needs a tool to track our work and development. Paymo is an excellent tool for time tracking and is widely used in many big companies. It is quite easy to explore the track history and it is possible to combine Paymo with Gantt charts created in Microsoft Visio.
- **T-shirts & sweaters** – We want to make it easy for other students, teachers, employees in KDA and other people to recognize us. Therefore, we want to represent ourselves by wearing identical T-shirts & sweaters with logo and group name.



## 8. Quality management

### 8.1. General information of quality management

In our quest to create a system that provides the desired functionality, some measures will be necessary. We will need a system that is able to ensure us that our product will meet certain standards, include properties as quality, but also the means to achieve it.

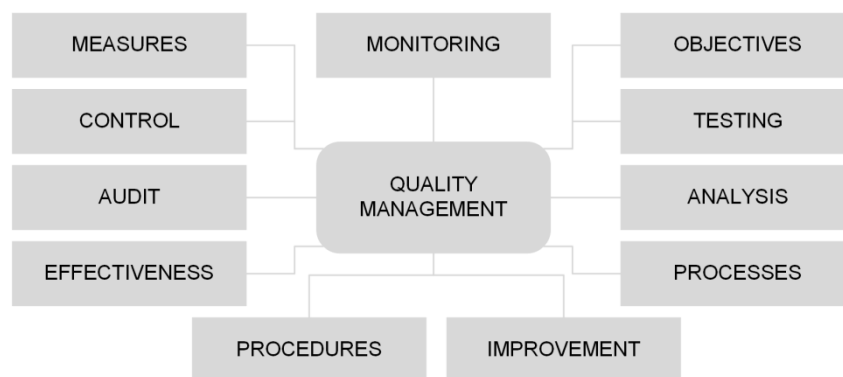
Management is the perception about doing things right, with management we make decisions and choose activities that we believe will help us in reaching our goals.

Quality management is used to coordinate activities, and to provide guidance within our project concerning quality.

Routing a plan for our quality measures includes reference to how we want things to be done, but also guidelines for our working-environments. We will need to establish what qualities our system should provide, what a quality are, and how we can achieve it.

With use of quality management (Figure 11), we want to map the reality of our project, by running tests, validation and verification of our documentation, and hopefully being able to do prototyping. Following the procedures we can identify possible errors before they arise, and prevent them from happening in the future.

Quality improvement will provide the foundation for our quality management-system, we will use lessons learned based on personal experience, but also systems developed to measure what quality is.



**Figure 11 - features to illustrate quality management.**

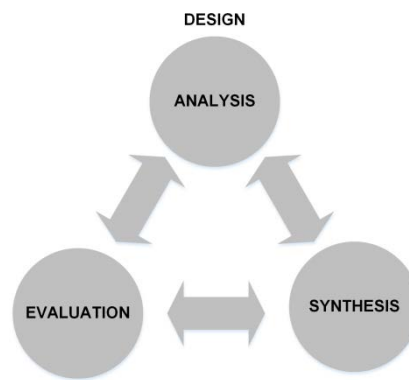


## 8.2. Quality tool

To be able to manage the quality of the project, there are some tools in the Systems Engineering kit that could be used to help the project group to achieve an overall better quality at all levels. The tools that the project group has used are as following.

- Analysis and synthesis evaluation process
- Key Performance Indicators (KPI)
- Quality function deployment (QFD)

### 8.2.1. Analysis-synthesis-evaluation process



**Figure 12 - Analysis-synthesis-evaluation model.**

The implementation of the Analysis-Synthesis-Evaluation process follows during the entire life-cycle. As an example, if you are in the design phase of the life-cycle. The same analytical approach applies, by listing a number of possible candidates' for your design, and explaining why the chosen one is desirable, and providing documentation to support your opinion. Further on in your design you should synthesis the candidate to look for features that might match up with known fully functional design, then you are validating that your candid design is feasible. If it is, you should evaluate if that design is the optimal approach for your stakeholders need. Are there any external regulations, forcing you in another direction about the design?



### 8.2.2. How and where their implemented

The project model provides an overall structure for the whole project, that's intended to solve your perception of the problem, or more precisely your team's perception. As it is normal to have a brainstorm in the very beginning of the project, that may be done in several ways.

The objective is of course to meet the user requirements, but identifying the stakeholders may be a tedious process as it is easy to only include stakeholders that you might profit from. The stakeholders are those who own issue containing systems, and your goal is to develop a system that can solve their actual problem.

Remember that we as a group may have a different perception of the problem, than the companies that uses the system, or the companies may even have their own users/customers that use uses our system. The stakeholder's objective may be to profit from the system, but the customers have a whole lot of other concerns. So the essence is to reflect on what system you are planning to develop, and what purpose it's intended to solve. The process here, and through the entire project involves the Analysis-Synthesis-Evaluation technique.

### 8.2.3. Reflection on the contribution of tools

Selection of the right tools is essential in SE, as there is no definitive approach about a problem. How do we ensure that we meet all this requirements? How do we validate that we are designing the right system for the right purpose? Did we perceive the actual requirements correct? These are all helpful questions in the SE process. Even the topics like, "system requirements", "user requirements" and so on are SE tools, used to solve problems in a structural manner, in the objective of creating a system that is optimum at the time. Conscious use of the techniques in SE is most useful, and which should be applied throughout the whole process as the project model illustrates.

## 8.3. Documentation Templates

To be able to assure that the documentation produced during the project is of quality; the project group has made different documentation templates. The reason for doing this is not only to get the quality that the group wants, but also to get a structured way to document, and to assure that all documents are the same.

The documentation template produced during the project is as following:

- Requirement template
- Meeting request template
- Minutes template
- Follow-up document
- Work log



- Deviation report
- Academic report template

### **8.3.1. Requirement Template**

To be able to assure traceability of a requirement, a system was developed to be able to trace the number ID of the requirement, from which department, importance level and who had made the requirement.

### **8.3.2. Notification of Meeting Template**

During the project, KPEC will be having regular meetings. To ensure that everybody knows where and when the meeting is going to happen, and what the content of the meeting is, a template was made to inform the participants.

### **8.3.3. Minutes Template**

During a meeting a lot of different topics are discussed and solutions made. To be able to assure that all participants or non-participants know what was concluded during the meeting, a minute template was made. The minutes also specify the attendees of the meeting and who was the secretarial.

### **8.3.4. Follow-Up Document**

As a part of a project of this magnitude, there will be a lot of problems, changes or other things that require follow up, as a consequence of this, a follow-up template was made to be able to follow up on these issues.

### **8.3.5. Work Log**

To be able to keep track on what work was done each day during the project time, a work log template was made.

### **8.3.6. Deviation Report**

The planning phase is the key to success for any project, but even a detailed plan can't count for unaccounted events that might arise during the project time. A deviation report was created to try to account for all events that were not planned, some of these event might be; overtime, change in plans, illness etc.

### **8.3.7. Academic Report Template**

A large portion of the project is to write an academic report. To be able to produce reports that are within the same standards, an academic report template was made to achieve a structured and overall quality that the team reaches for.



#### 8.4. Group Contract

In the early stages of the project, the group concluded that a group contract should be drafted. A well-defined contract reflects the ambitions of the group and also gives the guidelines on how to manage internal disagreements. Furthermore the contract also defines the different roles each member has during the project and the rules that must be followed. This assures the continuity of the project and saves time on unnecessary uncertainties.

#### 8.5. Life-cycle Assessment

Life-cycle to a product starts with the idea, the system conception, and then it follows with design and development. Then we have production, distribution and operation. The end phases of a life cycle are maintenance and support, retirement, phase-out and disposal.

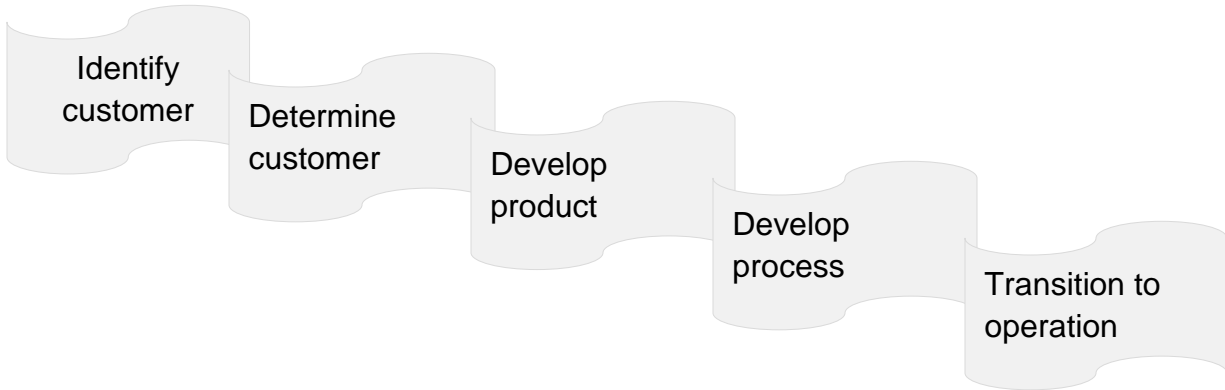
It is of importance that the group has taken the life cycle into account while designing the product because in each phase there are different issues that need to be addressed. In the early stages of the life cycle, the stakeholders in the entire life cycle must be identified and their needs and concern written down. The reason for this is that in every phase of the life cycle there are other stakeholders that have different needs and concerns that we as a project group needs to take into account before designing and development can begin. An example of a stakeholder that we need to take care of is in the distribution phase, are we making a product that is easy to pack and distribute? Or are we making a product that is easy to do maintenance and support on?

These are some of the question that might arise during the project work and that we need to have taken into consideration when designing the ECU.



## 8.6. Quality plan

To assure that the quality of the project is able to meet certain standards, we have developed a quality plan which follows 5 simple steps.



**Figure 13 - Quality plan**

### 8.6.1. Identify the customers

A customer is the recipient of goods or services, meaning that everyone that will interact with our product should be considered as a customer.

Identifying the customers is critical in the planning phase, by the use of a flow chart; one can identify who the customers are. Remember that not all customers are equally important, so be sure to weight them by their importance.

### 8.6.2. Determine customer needs

Analysing the customer needs might be an impossible task, human demands has no boundaries when it comes to functionality. Though it's critical to distinguish between what's a cool add-on feature, and what's really the core functionality that the customer is seeking.

- Perceived need
- Latent need
- Cultural need
- Innovative need
- Environment and safety

But most of all, the best approach is to pretend you are the customer, put yourself in the customers shoes. Does the application work like its intended? Easy to use? Is it adding value?





### 8.6.3. Develop product features

What features should our system have? We use tools such as brainstorming and evaluation matrices to define them. And if we are unsure about how viable the feature is, we can verify it through use of 'house of quality'. The house of quality is a brilliant tool to see the relationship between user requirements and system requirement, as it will provide the features of importance to us in reaching the development objectives.

### 8.6.4. Develop process features

With the desired features ready, we are now proceeding to realize the project. We will need to develop a process plan, which tools are we going to use? Equipment, platforms, technology, and a flowchart to provide a visual context of the process, will be useful. A process usually consists of, methods, manpower, material and machinery, make the process solutions concrete.

### 8.6.5. Transition to operation

Now that the product has been produced, the next phase of the process before the product can be fully implemented into the real world is the test phase. This is where KPEC has to test the product using test procedures created during the system conception, design and development phases to see if the requirements are met.



## 9. Risk management

### 9.1. General information about risk

The objective as for most projects are to successfully deliver what the project was intended to, but not all projects live to see another day, as issues and constraints develop later in the process. Finding integration problems late in the development process, it's a risk that can jeopardize the entire project. By identifying possible threats/problems early in the process, we will try to avoid any risks that in a large perspective would be obvious. The best way to overcome the communication difficulties inherent in a large team environment is to provide a common foundation for development and maintenance and to establish a common language for communicating across that foundation. As a result of proper implementation you may be rewarded with a successful project, meaning that you met the stakeholders actual need, if your needs were well defined you've probably had a good progression during the design process, leading to development and the final product. If the customer is happy, then you are happy, and everybody is happy.

### 9.2. Risk analysis

As a part of our risk analysis, we have done a thorough research on the risks associated with our project. With the risk analysis we can obtain a mitigation plan to mitigate the risk when they occur. We have looked into our management and we have gone as far as looking into our self as a team. We decided to divide the various risks into subgroups as follows.

Risks associated with the project:

- Management
- Resources
- Project members
- Software
- Hardware
- Implementation
- Testing
- 3rd party involvement

#### 9.2.1. Management

- There is a limited time frame with this project, so to fall behind schedule and not to be able to deliver in time is a major risk.
- Scope risk. The scope of the project is a big risk for us. An example of this is if our customer decides that they want a new feature implemented, and this change may cause the project group to fail our goals and deadlines.



- The risk of using our resources wrong. An example of this is if the project group is using it's time on the wrong implementation, making us unable to fulfill all the requirements.
- Unclearly defined requirements, leading up to not be able to deliver what the customer wants.
- Communication risk. To not be able to communicate with your project team or customer, making the project group unable to fulfill the task.

### 9.2.2. Resources

- Risk of overrunning the budget.
- Capacity risk. To not be able to solve a task without significant knowledge.
- Change in a group members availability. An example of this is if one group member can't put in the required hours.

### 9.2.3. Project members

- Illness in the team causing us to not be able to work as efficient as before because we are shorthanded.
- Discomfort in the team caused by another group member.
- Group members adding "nice to have" features, that will potentially ruin our time budget, and prevent us from delivering the main features.
- Disruption by other students
- Group member's personal equipment stops working. Keeping us from progress and valuable data.

### 9.2.4. Software

- Risk of not meeting the requirements. An example of this is if we can't establish communication with the tablet.
- Bugs in script, making us use time debugging and potentially preventing us from acquiring the desired functionalities.

### 9.2.5. Hardware

- Unable to deliver the intended schematic, making us unable to fulfill our requirement.
- Not able to verify that the requirements are met.

### 9.2.6. Implementation

- Unable to establish communication.
- The data is unable to process, due to bugs in the system.
- External equipment won't be able to interface with the ECU. Examples of this may be sensors and actuators.



### 9.2.7. Testing

- Unable to present the desired functionalities.
- Testing might give various results

### 9.2.8. Third-party involvement

- Problems with different programs. Example Cadence allegro and licensing.
- Server issues, where we get our licenses and store our data.
- Risk associated with HBV (Limitations; Room, delays, change in subjects)
- Suppliers are unable to deliver equipment needed for the project.

## 9.3. Qualitative risk analysis

We have done the following risk analysis, where risk is defined (1). Table 8 is the probability of an event happening, where we use 5 levels of probability. Table 9 defines the consequence/impact of an event happening.

Risk is defined as:

$$Risk = Impact \times Probability \quad (1)$$

### 9.3.1. Probability

**Table 8 - Probability Analysis**

Attribute: Probability		
Level	Value	Criteria
5	Near certainty	Everything points to this becoming a problem
4	Very likely	High chance of this becoming a problem
3	Likely (50/50)	There is an even chance this may turn into a problem
2	Unlikely	Risk like this may turn into a problem once in a while
1	Improbable	Not much chance this will become a problem

### 9.3.2. Impact

**Table 9 - Impact Analysis**

Attribute: Impact		
Level	Value	Criteria
5	Catastrophic	Project is at its breaking point, all measures need to be used to progress
4	Critical	Project is at a standstill, urgent measures are needed to progress
3	Moderate	Project is at a stop, measures need to be done to progress
2	Marginal	The working pace is slowed down, minimal-to none impact on progress
1	Negligible	Project will move on without problem



### 9.3.3. Risk matrix

The risk matrix is a result of (1), and can be seen in Table 10.

A risk matrix using a 5 colored system where each color represents a certain risk level. Where light green is the lowest level and red is the highest level which can be seen in Table 11. At each level, certain measures are needed to mitigate the risk.

**Table 10 - Risk matrix**

Probability	Impact				
	1	2	3	4	5
1	1	2	3	4	5
2	2	4	6	8	10
3	3	6	9	12	15
4	4	8	12	16	20
5	5	10	15	20	25

**Table 11 - Color codes of Table 10**

High risk	Not acceptable. Risk reduction must be implemented immediately
Medium/High	Not acceptable. Some risk reduction is needed.
Medium risk	Acceptable after consideration. Risk reduction measures should be assessed
Low/Medium	Acceptable. Monitoring is needed.
Low risk	Acceptable. Risk reduction measures not needed

### 9.4. Mitigation plan

The mitigation plan offers a structured set of actions that must be made in order to mitigate, if the situation reflects one of the risks stated in table 12. Which action that should be made depends on the nature of the risk and is explained in Table 12.

**Table 12 - Mitigation codes of Table 13.**

Assume/Accept	We acknowledge the risk, without any effort to control the risk
Avoid	Revision of the requirement, and or constraint is necessary to reduce or eliminate the risk.
Control	Effort has to be made, reassuring that the impact of a risk is less severe
Watch/Monitor	We acknowledge the risk, but we are aware, and want to monitor it



Table 13 - Mitigation plan

Risk	R	I	P	Mitigation
Time	6	2	3	See document: Time management system. (Watch/Monitor)
Scope	3	3	1	(Assume/Accept)
Wrong use of resources	4	4	1	(Watch/Monitor)
Unclearly defined requirements	5	5	1	Write system requirements, and ask the customer if we are on the same page. Then we'll get formal acceptance, to prevent any disputes on delivery. (Control)
Communication	4	4	1	See Regulations §8. We have no system for handling communication externally. (Assume/Accept)
Funding's	3	3	1	See document: Regulations §7. (Assume/Accept)
Capacity	15	5	3	If the task is close to solely being dependent of external resources, the project should be revised by the SE. (Control)
Change in a group members availability	2	2	1	See document: Regulations §2, 3, 4. (Assume/Accept)
Sickness	8	2	4	See document: Regulations §3. (Assume/Accept)
Discomfort	5	1	5	See document: Regulations §4. (Watch/Monitor)
Adding a "nice to have" feature	10	2	5	Assess the gain/loss ratio, we should avoid any add on features unless it is considered essential and or necessary to meet the requirements. (Watch/Monitor)
Disruption by other students	2	2	1	See document: Regulations §10.7. (Control)
Unable to meet requirements	10	5	2	Then we're screwed. (Avoid)
Bugs	10	2	5	Structure, systematic approach, readability, seeking external guidance. (Assume/Accept)
Backup	5	5	1	Make a backup plan. (Control)
Unable to deliver schematic and PCB-design	5	5	1	Proactive measures. (Assume/Accept)
Won't be able to establish communication	4	4	1	(Watch/Monitor)
Fail to process data	5	5	1	(Watch/Monitor)
External equipment interface problem	8	4	2	Be sure to verify that all equipment is compatible with each other. (Control)
Unable to show the desired functionality	12	4	3	(Avoid)
Licenses	5	5	1	Proactive, establish those things in advance. (Watch/Monitor)
Server issues	16	4	4	Make a backup plan. (Control)
Constraints regarding HBV	3	1	3	Be aware of them, delays are common. (Assume/accept)
Suppliers unable to deliver equipment	2	1	2	Order in good time. (Watch/Monitor)



## Bibliography

defence, D. o. (2001). *Systems engineering fundamentals*. Virginia: Defense acquisition university press Fort Belvoir.

Safi, J. (2014). *MPSE-2201 Systems Design & Engineering*. University, HBV. Kongsberg: Jamal Safi.

Shamieh, C. (2011). *System engineering for dummies*. Willey publishing Indiana.



KONGSBERG



# Requirements Specification

## KPEC

<b>Employer</b>	Kongsberg Defence & Aerospace			
<b>Group Members</b>	<b>Name</b>		<b>Initials</b>	
	Ola Pedersen Aasheim		OA	
	Salahuddin Asjad		SA	
	Hung Dinh		HD	
	Dler Hasan		DH	
	Even Gudbrandsen		EG	
	Jannik Schäffer		JS	
<b>Document information</b>	<b>Revision</b>	<b>Date</b>	<b>Approved</b>	<b>Pages</b>
	4.0	18.05.2015	JS	68







## Abstract

The formative document defines systems, software and hardware requirements, which is a continuation of the concept of operations document. The overall purpose of this document is to describe the needs for what stakeholders require from the system. After reading this document, one should be able to understand which requirements have to be fulfilled in order to verify that the system is built correctly.



## Revision Table

Version	Date	Approval	Description
1.0	19.01.2015	HD	<ul style="list-style-type: none"> <li>Created the document</li> </ul>
2.0	26.01.2015	DH	<ul style="list-style-type: none"> <li>Revised and published</li> </ul>
2.1	30.01.2015	HD	<ul style="list-style-type: none"> <li>Changed Created → Added in changelog. Also changed requirement 29, 30, 34, 35, 43, 81, 90-107, 109 and 110. Added requirement 111.</li> </ul>
2.2	03.03.2015	HD	Revised after 1 iteration: <ul style="list-style-type: none"> <li>Changed Created → Pending in change log.</li> <li>Changed requirement 1, 27, 49 and 55.</li> </ul>
3.0	13.03.2015	DH	<ul style="list-style-type: none"> <li>Revised and published</li> </ul>
4.0	18.05.2015	JS	<ul style="list-style-type: none"> <li>Revised and published</li> </ul>

## Contents

Abstract .....	2
Revision Table .....	3
List of tables .....	3
List of requirement tables.....	3
1. Introduction .....	7
2. Requirements Changelog.....	66

## List of tables

Table 1 – Requirements Changelog Part 1.....	66
Table 2 – Requirements Changelog Part 2.....	67
Table 3 – Requirements Changelog Part 3.....	68

## List of requirement tables

Requirement 1 – 001-SY1EG .....	8
Requirement 2 – 002-SY1EG .....	8
Requirement 3 – 003-SY1EG .....	9
Requirement 4 – 004-SY1EG .....	9



Requirement 5 - 005-SY1EG .....	10
Requirement 6 - 006-SY1EG .....	10
Requirement 7 - 007-SY3EG .....	11
Requirement 8 - 008-SY3EG .....	11
Requirement 9 - 009-SY3EG .....	12
Requirement 10 - 010-SY3EG .....	12
Requirement 11 - 011-SY3EG .....	13
Requirement 12 - 012-SY3EG .....	13
Requirement 13 - 013-SY4EG .....	14
Requirement 14 - 014-SY4EG .....	14
Requirement 15 - 015-SY5EG .....	15
Requirement 16 - 016-SY3JS .....	15
Requirement 17 - 017-SY4EG .....	16
Requirement 18 - 018-SY4EG .....	16
Requirement 19 - 019-SY5EG .....	17
Requirement 20 - 020-SY3EG .....	17
Requirement 21 - 021-SY3EG .....	18
Requirement 22 - 022-SY4EG .....	18
Requirement 23 - 023-SW4SA .....	19
Requirement 24 - 024-SW3SA .....	19
Requirement 25 - 025-SW5SA .....	20
Requirement 26 - 026-SW2SA .....	20
Requirement 27 - 027-SW2DH.....	21
Requirement 28 - 028-SW5DH.....	21
Requirement 29 - 029-SW4DH.....	22
Requirement 30 - 030-SW4SA .....	22
Requirement 31 - 031-SW3DH.....	23
Requirement 32 - 032-SW3DH.....	23
Requirement 33 - 033-SW3DH.....	24
Requirement 34 - 034-SW5SA .....	24
Requirement 35 - 035-SW5SA .....	25
Requirement 36 - 036-SW5SA .....	25
Requirement 37 - 037-SW5DH.....	26
Requirement 38 - 038-SW5SA .....	26
Requirement 39 - 039-SW4DH.....	27
Requirement 40 - 040-SW4DH.....	27
Requirement 41 - 041-SW5DH.....	28
Requirement 42 - 042-SW3DH.....	28
Requirement 43 - 043-SW5SA .....	29
Requirement 44 - 044-SW5SA .....	29
Requirement 45 - 045-SW4SA .....	30



Requirement 46 - 046-SW3SA .....	30
Requirement 47 - 047-HW2HD .....	31
Requirement 48 - 048-HW2JS .....	31
Requirement 49 - 049-HW2HD .....	32
Requirement 50 - 050-HW3JS .....	32
Requirement 51 - 051-HW3HD .....	33
Requirement 52 - 052-HW3EG .....	33
Requirement 53 - 053-HW2JS .....	34
Requirement 54 - 054-HW3HD .....	34
Requirement 55 - 055-HW3EG .....	35
Requirement 56 - 056-HW4HD .....	35
Requirement 57 - 057-HW3JS .....	36
Requirement 58 - 058-HW3JS .....	36
Requirement 59 - 059-HW3HD .....	37
Requirement 60 - 060-HW3EG .....	37
Requirement 61 - 061-HW3HD .....	38
Requirement 62 - 062-HW4EG .....	38
Requirement 63 - 063-HW3HD .....	39
Requirement 64 - 064-HW3EG .....	39
Requirement 65 - 065-HW3JS .....	40
Requirement 66 - 066-HW2EG .....	40
Requirement 67 - 067-HW3OA.....	41
Requirement 68 - 068-HW3HD .....	41
Requirement 69 - 069-HW3JS .....	42
Requirement 70 - 070-HW3HD .....	42
Requirement 71 - 071-HW3OA.....	43
Requirement 72 - 072-HW3JS .....	43
Requirement 73 - 073-HW3EG .....	44
Requirement 74 - 074-HW3JS .....	44
Requirement 75 - 075-HW3EG .....	45
Requirement 76 - 076-HW3HD .....	45
Requirement 77 - 077-HW3JS .....	46
Requirement 78 - 078-HW5HD .....	46
Requirement 79 - 079-HW3HD .....	47
Requirement 80 - 080-HW4JS .....	47
Requirement 81 - 081-HW4HD .....	48
Requirement 82 - 082-HW3HD .....	48
Requirement 83 - 083-HW3JS .....	49
Requirement 84 - 084-HW4EG .....	49
Requirement 85 - 085-HW5JS .....	50
Requirement 86 - 086-HW5OA.....	50



Requirement 87 - 087-HW3JS .....	51
Requirement 88 - 088-HW3EG .....	51
Requirement 89 - 089-HW4HD .....	52
Requirement 90 - 090-HW3OA.....	52
Requirement 91 - 091-HW4OA.....	53
Requirement 92 - 092-HW4OA.....	53
Requirement 93 - 093-HW5OA.....	54
Requirement 94 - 094-HW4OA.....	54
Requirement 95 - 095-HW3OA.....	55
Requirement 96 - 096-HW3OA.....	55
Requirement 97 - 097-HW4OA.....	56
Requirement 98 - 098-HW4OA.....	56
Requirement 99 - 099-HW5OA.....	57
Requirement 100 - 100-HW5OA.....	57
Requirement 101 - 101-HW3OA.....	58
Requirement 102 - 102-HW3OA.....	58
Requirement 103 - 103-HW3OA.....	59
Requirement 104 - 104-HW4OA.....	59
Requirement 105 - 105-HW5OA.....	60
Requirement 106 - 106-HW2OA.....	60
Requirement 107 - 107-HW2OA.....	61
Requirement 108 - 108-HW4OA.....	61
Requirement 109 - 109-HW4OA.....	62
Requirement 110 - 110-HW4OA.....	62
Requirement 111 - 111-SY3HD .....	63
Requirement 112 - 112-SY1EG .....	63
Requirement 113 - 113-SW3DH.....	64
Requirement 114 - 114-HW2HD .....	64
Requirement 115 - 115-SY3EG .....	65
Requirement 116 - 116-SW2DH.....	65



## 1. Introduction

The following section of the document lists all the requirements in tables. Please refer to *Requirements Guidelines* for a complete explanation, reason and specification for how the requirements shall be written and how they are identified.

KPEC have defined two different kinds of requirements. The requirements which are located in this document are the requirements given from our stakeholders (KDA). The detailed requirements which are obtained during the detailed design phase (can be found in the *Detailed requirements* specifications document).

The complete list of changes in requirements is found in the Requirements Changelog in the end of this document.

**Requirement 1 – 001-SY1EG**

<b>001-SY1EG</b>	<b>Design an ECU</b>	<b>12.01.2015</b>
<b>Status</b>	Changed	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To regulate the combustion process of an engine	
<b>WHY</b>	Create an exciting and multidisciplinary assignment.	
<b>Acceptance Criteria</b>	Have a full electronic design for an engine control system.	
<b>Comments</b>	Succeeded by 112-SY1EG in the requirement specification.	

**Requirement 2 – 002-SY1EG**

<b>002-SY1EG</b>	<b>FPGA based ECU</b>	<b>12.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	Implement FPGA technology in the engine control unit.	
<b>WHY</b>	To apply innovative technology in existing applications.	
<b>Acceptance Criteria</b>	FPGA based engine control unit.	
<b>Comments</b>		

**Requirement 3 – 003-SY1EG**

<b>003-SY1EG</b>	<b>Schematics and PCB-layout</b>	<b>12.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	Full design of an engine control unit	
<b>WHY</b>	To have a schematic ready for PCB layout.	
<b>Acceptance Criteria</b>	Complete schematics design file.	
<b>Comments</b>		

**Requirement 4 – 004-SY1EG**

<b>004-SY1EG</b>	<b>Tablet communication – Read</b>	<b>12.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	Communicate with the engine control unit using a tablet	
<b>WHY</b>	To be able to monitor engine parameters.	
<b>Acceptance Criteria</b>	Must be able to read IO values.	
<b>Comments</b>		



**Requirement 5 – 005-SY1EG**

<b>005-SY1EG</b>	<b>Tablet communication - Write</b>	<b>12.01.2015</b>
<b>Status</b>	Pending	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	Communicate with the engine control unit using a tablet	
<b>WHY</b>	To be able to change engine parameters.	
<b>Acceptance Criteria</b>	Must be able to adjust control parameters.	
<b>Comments</b>		

**Requirement 6 – 006-SY1EG**

<b>006-SY1EG</b>	<b>Support for up to 8 cylinder engines</b>	<b>12.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To support up to 8 cylinder engines.	
<b>WHY</b>	To be able to control a wide range of engines.	
<b>Acceptance Criteria</b>	Must be able to handle IO for up to 8 cylinders.	
<b>Comments</b>		

**Requirement 7 – 007-SY3EG**

<b>007-SY3EG</b>	<b>Engine performance tuning</b>	<b>12.01.2015</b>
<b>Status</b>	Pending	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To be able to tune engine performance.	
<b>WHY</b>	To allow the user to choose desired performance.	
<b>Acceptance Criteria</b>	Parameter change results in a system response.	
<b>Comments</b>	Long-term goals make this a level 3 priority.	

**Requirement 8 – 008-SY3EG**

<b>008-SY3EG</b>	<b>Engine performance tuning from tablet</b>	<b>12.01.2015</b>
<b>Status</b>	Pending	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To be able to tune from tablet.	
<b>WHY</b>	To allow the user to choose desired performance.	
<b>Acceptance Criteria</b>	Parameter change results in a system response.	
<b>Comments</b>	Long-term goals make this a level 3 priority.	

**Requirement 9 – 009-SY3EG**

<b>009-SY3EG</b>	<b>Communication with external devices</b>	<b>13.01.2015</b>
<b>Status</b>	Pending	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	Communication with external devices.	
<b>WHY</b>	To be able to extend functionalities.	
<b>Acceptance Criteria</b>	Must have a system for interfacing external devices.	
<b>Comments</b>	Long-term goals make this a level 3 priority.	

**Requirement 10 – 010-SY3EG**

<b>010-SY3EG</b>	<b>Petrol fuel option</b>	<b>13.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To be able to handle petrol as fuel.	
<b>WHY</b>	Petrol engines are most commonly used.	
<b>Acceptance Criteria</b>	The system must have interface towards petrol as fuel.	
<b>Comments</b>	Long-term goals make this a level 3 priority.	

**Requirement 11 – 011-SY3EG**

<b>011-SY3EG</b>	<b>Bioethanol fuel option</b>	<b>13.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To be able to handle bioethanol as fuel.	
<b>WHY</b>	Bioethanol is commonly used as an alternative fuel.	
<b>Acceptance Criteria</b>	The system must have interface towards bioethanol as fuel.	
<b>Comments</b>	Long-term goals make this a level 3 priority.	

**Requirement 12 – 012-SY3EG**

<b>012-SY3EG</b>	<b>Support for Peak Pressure Point (PPP)</b>	<b>13.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To be able to detect peak pressure point in cylinders.	
<b>WHY</b>	To obtain precise ignition timing, this can increase efficiency.	
<b>Acceptance Criteria</b>	Shall interface sensors to determine peak pressure point.	
<b>Comments</b>		

**Requirement 13 – 013-SY4EG**

<b>013-SY4EG</b>	<b>Injector flowrate adjustment</b>	<b>13.01.2015</b>
<b>Status</b>	Pending	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To adjust injector flowrate according to process data.	
<b>WHY</b>	To optimize the combustion process.	
<b>Acceptance Criteria</b>	Shall be able to adjust injector flowrate according to process data.	
<b>Comments</b>	Long-term goals make this a level 4 priority.	

**Requirement 14 – 014-SY4EG**

<b>014-SY4EG</b>	<b>Dynamic fuel map adjustment</b>	<b>13.01.2015</b>
<b>Status</b>	Pending	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To adjust fuel map dynamic according to environment.	
<b>WHY</b>	To optimize efficiency in different environments.	
<b>Acceptance Criteria</b>	Must have interface to allow mapping.	
<b>Comments</b>	Long-term goals make this a level 4 priority.	

**Requirement 15 – 015-SY5EG**

<b>015-SY5EG</b>	<b>Support for pneumatic valve control module</b>	<b>13.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To be able to interface a pneumatic valve control module.	
<b>WHY</b>	Flowrate adjustment can enhance performance.	
<b>Acceptance Criteria</b>	Verify in the design that there is an interface for an external pneumatic valve control module.	
<b>Comments</b>	Long-term goals make this a level 5 priority.	

**Requirement 16 – 016-SY3JS**

<b>016-SY3JS</b>	<b>Tablet</b>	<b>13.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	Use tablet to interface with ECU.	
<b>WHY</b>	To make the system more available and user friendly.	
<b>Acceptance Criteria</b>	Choose a tablet that supports interfacing with the ECU.	
<b>Comments</b>		

**Requirement 17 – 017-SY4EG**

<b>017-SY4EG</b>	<b>Engine start/stop</b>	<b>14.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	User	
<b>WANTS</b>	To start the engine using a switch.	
<b>WHY</b>	To be able to stop or start the engine.	
<b>Acceptance Criteria</b>	Must have IO support for ignition signal.	
<b>Comments</b>		

**Requirement 18 – 018-SY4EG**

<b>018-SY4EG</b>	<b>Engine stop when rolling over</b>	<b>14.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	User	
<b>WANTS</b>	To stop the engine when rolling over.	
<b>WHY</b>	To ensure safety if rollover occurs.	
<b>Acceptance Criteria</b>	Must have IO support for rollover signal.	
<b>Comments</b>		

**Requirement 19 – 019-SY5EG**

<b>019-SY5EG</b>	<b>Unintended startup</b>	<b>14.01.2015</b>
<b>Status</b>	Pending	
<b>WHO</b>	User	
<b>WANTS</b>	To prevent engine from unintended startup.	
<b>WHY</b>	To avoid unexpected engine behavior.	
<b>Acceptance Criteria</b>	Must have IO support for unintended startup.	
<b>Comments</b>	Long-term goals make this a level 5 priority.	

**Requirement 20 – 020-SY3EG**

<b>020-SY3EG</b>	<b>Engine acceleration</b>	<b>14.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	User	
<b>WANTS</b>	To accelerate engine when activating accelerator.	
<b>WHY</b>	To increase engine speed.	
<b>Acceptance Criteria</b>	Shall have IO support for reading accelerator value.	
<b>Comments</b>		



**Requirement 21 – 021-SY3EG**

<b>021-SY3EG</b>	<b>Fail-safe protection</b>	<b>14.01.2015</b>
<b>Status</b>	Pending	
<b>WHO</b>	User	
<b>WANTS</b>	To have an engine fail-safe protection.	
<b>WHY</b>	To prevent mechanical engine damage.	
<b>Acceptance Criteria</b>	Must have a system for engine protection.	
<b>Comments</b>		

**Requirement 22 – 022-SY4EG**

<b>022-SY4EG</b>	<b>Self-check</b>	<b>14.01.2015</b>
<b>Status</b>	Pending	
<b>WHO</b>	User	
<b>WANTS</b>	To have a self-check system for IO.	
<b>WHY</b>	For troubleshooting purposes.	
<b>Acceptance Criteria</b>	Must have a surveillance system for fault detection in IO.	
<b>Comments</b>		

**Requirement 23 – 023-SW4SA**

<b>023-SW4SA</b>	<b>GitHub - Version control</b>	<b>12.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	Version control system to record changes to code files.	
<b>WHY</b>	Can revert files back to a previous version and collaborate with other developers.	
<b>Acceptance Criteria</b>	GitHub shall be used for version control.	
<b>Comments</b>		

**Requirement 24 – 024-SW3SA**

<b>024-SW3SA</b>	<b>Ethernet communication</b>	<b>12.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To communicate between Zynq and tablet via Ethernet.	
<b>WHY</b>	Widely supported standard used for both communication and debugging.	
<b>Acceptance Criteria</b>	Ethernet communication shall work between Zynq and tablet.	
<b>Comments</b>		

**Requirement 25 – 025-SW5SA**

<b>025-SW5SA</b>	<b>Wi-Fi communication</b>	<b>12.01.2015</b>
<b>Status</b>	Pending	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To communicate between Zynq and tablet via Wi-Fi.	
<b>WHY</b>	Not all tablets/computers have support for physical access to Ethernet (RJ45 connector).	
<b>Acceptance Criteria</b>	Wi-Fi communication shall work between Zynq and tablet.	
<b>Comments</b>		

**Requirement 26 – 026-SW2SA**

<b>026-SW2SA</b>	<b>Xilinx Vivado – Zynq programming</b>	<b>12.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	Use Xilinx Vivado Design Suite to configure Zynq SoC.	
<b>WHY</b>	Free and latest version of HDL design suite from Xilinx.	
<b>Acceptance Criteria</b>	Zynq SoC is programmed through configurations from Xilinx Vivado.	
<b>Comments</b>		

**Requirement 27 – 027-SW2DH**

<b>027-SW2DH</b>	<b>Android application</b>	<b>22.01.2015</b>
<b>Status</b>	Changed	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To create an app with graphical user interface (GUI) for tablet.	
<b>WHY</b>	To allow user to interact with ECU through graphical icons and visual indicators.	
<b>Acceptance Criteria</b>	Check if the app is running on the tablet without any exception or crash.	
<b>Comments</b>	Succeeded by 113-SWDH in the requirement specification.	

**Requirement 28 – 028-SW5DH**

<b>028-SW5DH</b>	<b>Reconfigure FPGA over Ethernet</b>	<b>12.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To reprogram SPI flash in Zynq SoC over Ethernet connection.	
<b>WHY</b>	Use the network to reprogram the SPI flash so developers can collaborate.	
<b>Acceptance Criteria</b>	The old bitstream data replaced by a new bitstream file using Ethernet.	
<b>Comments</b>	Long-term goals make this a level 5 priority.	

**Requirement 29 – 029-SW4DH**

<b>029-SW4DH</b>	<b>Android Studio</b>	<b>13.01.2015</b>
<b>Status</b>	Changed	
<b>WHO</b>	Software developer	
<b>WANTS</b>	Use Android Studio project to develop the application.	
<b>WHY</b>	Standard software used for Android developing and debugging.	
<b>Acceptance Criteria</b>	Android Studio shall be used for app development and debugging.	
<b>Comments</b>	Succeeded by 1029 – SW4DH in the detailed requirements specification document.	

**Requirement 30 – 030-SW4SA**

<b>030-SW4SA</b>	<b>Android version support</b>	<b>13.01.2015</b>
<b>Status</b>	Changed	
<b>WHO</b>	Software developer	
<b>WANTS</b>	Use API version 19 or later (Android 4.4.2)	
<b>WHY</b>	Have support for the latest design features provided by Android.	
<b>Acceptance Criteria</b>	Configure the SDK and implement the right API to the app.	
<b>Comments</b>	Succeeded by 1030 – SW4SA in the detailed requirements specification document.	

**Requirement 31 – 031-SW3DH**

<b>031-SW3DH</b>	<b>History log</b>	<b>13.01.2015</b>
<b>Status</b>	Pending	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To view a log history of parameters from ECU on the tablet.	
<b>WHY</b>	To keep track of parameters from ECU.	
<b>Acceptance Criteria</b>	Display a history log on the tablet of different parameters from the ECU.	
<b>Comments</b>		

**Requirement 32 – 032-SW3DH**

<b>032-SW3DH</b>	<b>Implement Zynq module as IP</b>	<b>13.01.2015</b>
<b>Status</b>	Pending	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To implement Zynq module as IP when designing.	
<b>WHY</b>	Use IP catalog in Vivado design suite for creation, analyze and validation of complex designs.	
<b>Acceptance Criteria</b>	Shall use Vivado IP integrator to generate block diagrams.	
<b>Comments</b>		

**Requirement 33 – 033-SW3DH**

<b>033-SW3DH</b>	<b>Tuning from tablet</b>	<b>13.01.2015</b>
<b>Status</b>	Pending	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To have tuning function from the tablet.	
<b>WHY</b>	Allow the user to choose desired performance from the graphical user interface.	
<b>Acceptance Criteria</b>	Parameter change using tuning function will result into a system response.	
<b>Comments</b>	Long-term makes this a level 3 priority.	

**Requirement 34 – 034-SW5SA**

<b>034-SW5SA</b>	<b>Toolbar inside the application</b>	<b>13.01.2015</b>
<b>Status</b>	Changed	
<b>WHO</b>	User	
<b>WANTS</b>	To have a user-friendly toolbar layout.	
<b>WHY</b>	Easy navigate to settings and reload functions.	
<b>Acceptance Criteria</b>	The toolbar layout shall be properly functioning with access to the icons.	
<b>Comments</b>	Succeeded by 1034 – SW5SA in the detailed requirements specification document.	

**Requirement 35 – 035-SW5SA**

<b>035-SW5SA</b>	<b>Navigation drawer inside the application</b>	<b>13.01.2015</b>
<b>Status</b>	Changed	
<b>WHO</b>	User	
<b>WANTS</b>	To have a user-friendly navigational drawer layout.	
<b>WHY</b>	Easy access to main navigation options.	
<b>Acceptance Criteria</b>	The navigation drawer layout shall be functioning with an accessible menu.	
<b>Comments</b>	Succeeded by 1035 – SW5SA in the detailed requirements specification document.	

**Requirement 36 – 036-SW5SA**

<b>036-SW5SA</b>	<b>Low battery consumption</b>	<b>13.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Software developer	
<b>WANTS</b>	Avoid unnecessary usage of processor resources within the tablet.	
<b>WHY</b>	Unnecessary usage of processor resources will drain battery and will lead to instability.	
<b>Acceptance Criteria</b>	Inactive activities and services shall not drain battery.	
<b>Comments</b>		



**Requirement 37 – 037-SW5DH**

<b>037-SW5DH</b>	<b>Communication security</b>	<b>14.01.2015</b>
<b>Status</b>	Pending	
<b>WHO</b>	Software developer	
<b>WANTS</b>	Safe communication between ECU and tablet when Wi-Fi communication is implemented.	
<b>WHY</b>	Do not be interrupted by unknown devices while using the application.	
<b>Acceptance Criteria</b>	Use WPA2 encryption with uppercase-, lowercase- and special letters.	
<b>Comments</b>		

**Requirement 38 – 038-SW5SA**

<b>038-SW5SA</b>	<b>Tablet compatibility</b>	<b>14.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Software developer	
<b>WANTS</b>	To have compatibility to wide range of the newer tablets devices (API 19 and later).	
<b>WHY</b>	Not all tablets have the same hardware specifications.	
<b>Acceptance Criteria</b>	Design layouts for various screen sizes. Min.2	
<b>Comments</b>		

**Requirement 39 – 039-SW4DH**

<b>039-SW4DH</b>	<b>Flash write</b>	<b>14.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	Be able to write to flash memory.	
<b>WHY</b>	Be able to update the configuration and code inside flash memory.	
<b>Acceptance Criteria</b>	Must be able to write into the flash memory.	
<b>Comments</b>		

**Requirement 40 – 040-SW4DH**

<b>040-SW4DH</b>	<b>Read flash</b>	<b>14.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence Systems	
<b>WANTS</b>	To be able to read from flash memory.	
<b>WHY</b>	Be able to read the existing configuration that is stored inside flash memory.	
<b>Acceptance Criteria</b>	Must be able to read from the flash memory.	
<b>Comments</b>		

**Requirement 41 – 041-SW5DH**

<b>041-SW5DH</b>	<b>Fail-safe function for engine</b>	<b>14.01.2015</b>
<b>Status</b>	Pending	
<b>WHO</b>	Software developer	
<b>WANTS</b>	To stop the engine when a sensor is not functioning properly.	
<b>WHY</b>	To avoid that a damaged sensors can cause critical damages to the engine.	
<b>Acceptance Criteria</b>	Shall be able to detect absence from a sensor and handle the operation immediately.	
<b>Comments</b>		

**Requirement 42 – 042-SW3DH**

<b>042-SW3DH</b>	<b>Database</b>	<b>15.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Software developer	
<b>WANTS</b>	To store parameters from the engine.	
<b>WHY</b>	Be able to use the stored parameters for logging.	
<b>Acceptance Criteria</b>	Tablet shall have read and write access to database.	
<b>Comments</b>		

**Requirement 43 – 043-SW5SA**

<b>043-SW5SA</b>	<b>Phonegap</b>	<b>15.01.2015</b>
<b>Status</b>	Changed	
<b>WHO</b>	Software developer	
<b>WANTS</b>	To develop application for tablet using Phonegap.	
<b>WHY</b>	Phonegap projects are developed using widely known languages.	
<b>Acceptance Criteria</b>	Phonegap shall be used for app development and debugging.	
<b>Comments</b>	Succeeded by 1043 – SW4SA in the detailed requirements specification document.	

**Requirement 44 – 044-SW5SA**

<b>044-SW5SA</b>	<b>Real-time view</b>	<b>20.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To view parameters from ECU in real-time on tablet.	
<b>WHY</b>	Reading engine parameters are useful for monitoring and analyzing.	
<b>Acceptance Criteria</b>	The tablet screen shall be able to view different parameters within 50 ms.	
<b>Comments</b>		

**Requirement 45 – 045-SW4SA**

<b>045-SW4SA</b>	<b>Real-time view</b>	<b>20.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To view parameters from ECU in real-time on tablet.	
<b>WHY</b>	Reading engine parameters are useful for monitoring and analyzing.	
<b>Acceptance Criteria</b>	The tablet screen shall be able to view different parameters within 100 ms.	
<b>Comments</b>		

**Requirement 46 – 046-SW3SA**

<b>046-SW3SA</b>	<b>Real-time view</b>	<b>20.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To view parameters from ECU in real-time on tablet.	
<b>WHY</b>	Reading engine parameters are useful for monitoring and analyzing.	
<b>Acceptance Criteria</b>	The tablet screen shall be able to view different parameters within 200 ms.	
<b>Comments</b>		

**Requirement 47 – 047- HW2HD**

<b>047-HW2HD</b>	<b>Xilinx Zynq</b>	<b>13.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To implement a micromodule with a Xilinx Zynq in the ECU.	
<b>WHY</b>	Familiar technology to KDA.	
<b>Acceptance Criteria</b>	Design must contain a micromodule with a Xilinx Zynq.	
<b>Comments</b>		

**Requirement 48 – 048-HW2JS**

<b>048-HW2JS</b>	<b>Motherboard</b>	<b>13.01.2015</b>
<b>Status</b>	Pending	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To have an ECU motherboard that interfaces with the micromodule.	
<b>WHY</b>	In order to interface IO from an engine and process parameters with the micromodule.	
<b>Acceptance Criteria</b>	The micromodule shall mate with the motherboard.	
<b>Comments</b>		

**Requirement 49 – 049-HW2HD**

<b>049-HW2HD</b>	<b>Cadence Allegro</b>	<b>13.01.2015</b>
<b>Status</b>	Changed	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To have the design done in Cadence Allegro	
<b>WHY</b>	The software is desired by KDA.	
<b>Acceptance Criteria</b>	Schematic and PCB layout drawn in Cadence Allegro	
<b>Comments</b>	Succeeded by 114-HW2HD in the requirement specification.	

**Requirement 50 – 050-HW3JS**

<b>050-HW3JS</b>	<b>Fuel injectors</b>	<b>13.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To be able to control the fuel injectors individually.	
<b>WHY</b>	Fuel management.	
<b>Acceptance Criteria</b>	Separate drivers for each injector.	
<b>Comments</b>		

**Requirement 51 – 051-HW3HD**

<b>051-HW3HD</b>	<b>Individual ignition coils</b>	<b>13.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To be able to control the ignition coils individually.	
<b>WHY</b>	Ignition management.	
<b>Acceptance Criteria</b>	Separate drivers for each spark plug.	
<b>Comments</b>		

**Requirement 52 – 052-HW3EG**

<b>052-HW3EG</b>	<b>Exhaust temperature</b>	<b>13.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To monitor individual cylinder-exhaust temperatures.	
<b>WHY</b>	To optimize combustion regulation.	
<b>Acceptance Criteria</b>	I/O must support up to 8 sensors.	
<b>Comments</b>		



**Requirement 53 – 053-HW2JS**

<b>053-HW2JS</b>	<b>Read analog quantities</b>	<b>13.01.2015</b>
<b>Status</b>	Pending	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To measure analog quantities.	
<b>WHY</b>	To monitor the combustion process.	
<b>Acceptance Criteria</b>	The system is able to read analog quantities.	
<b>Comments</b>		

**Requirement 54 – 054-HW3HD**

<b>054-HW3HD</b>	<b>Oxygen level</b>	<b>13.01.2015</b>
<b>Status</b>	Pending	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To monitor the oxygen level in the exhaust accurately.	
<b>WHY</b>	To evaluate the quality of the combustion.	
<b>Acceptance Criteria</b>	The system is able to read oxygen level in exhaust.	
<b>Comments</b>		

**Requirement 55 – 055-HW3EG**

<b>055-HW3EG</b>	<b>Cylinder pressure characteristics</b>	<b>13.01.2015</b>
<b>Status</b>	Changed	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To monitor pressure characteristics inside the cylinders.	
<b>WHY</b>	To enhance the ignition timing in the cylinders.	
<b>Acceptance Criteria</b>	Shall be able to read the pressure characteristics inside each cylinder.	
<b>Comments</b>	Succeeded by 115-HW3EG in the requirement specification.	

**Requirement 56 – 056-HW4HD**

<b>056-HW4HD</b>	<b>Dedicated GPIO</b>	<b>13.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To have GPIO for external devices.	
<b>WHY</b>	To interface to external devices.	
<b>Acceptance Criteria</b>	Dedicated GPIO available.	
<b>Comments</b>		

**Requirement 57 – 057- HW3JS**

<b>057-HW3JS</b>	<b>Manifold air pressure</b>	<b>13.01.2015</b>
<b>Status</b>	Pending	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To monitor air pressure in the manifold.	
<b>WHY</b>	To regulate the combustion process.	
<b>Acceptance Criteria</b>	Shall be able to measure the pressure inside the inlet manifold.	
<b>Comments</b>		

**Requirement 58 – 058-HW3JS**

<b>058-HW3JS</b>	<b>Communication</b>	<b>13.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To establish communication between ECU and tablet.	
<b>WHY</b>	To communicate between system and user	
<b>Acceptance Criteria</b>	Communication interface included in the design.	
<b>Comments</b>		

**Requirement 59 – 059-HW3HD**

<b>059-HW3HD</b>	<b>CAN communication</b>	<b>13.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To have CAN communication.	
<b>WHY</b>	To enable communication with external systems.	
<b>Acceptance Criteria</b>	Shall have a CAN- transceiver implemented.	
<b>Comments</b>		

**Requirement 60 – 060-HW3EG**

<b>060-HW3EG</b>	<b>Flexi fuel</b>	<b>13.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To operate on 2 different fuels.	
<b>WHY</b>	Extended functionalities, this makes the ECU more versatile.	
<b>Acceptance Criteria</b>	System to detect fuel type and I/O to manage fuel selection.	
<b>Comments</b>	Long-term goals make this a level 3 priority.	

**Requirement 61 – 061-HW3HD**

<b>061-HW3HD</b>	<b>Peak pressure point (PPP)</b>	<b>13.01.2015</b>
<b>Status</b>	Pending	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To detect Peak Pressure Point in cylinders.	
<b>WHY</b>	To enhance the ignition timing.	
<b>Acceptance Criteria</b>	Shall be able to read pressure characteristics inside the cylinder.	
<b>Comments</b>	Long-term goals make this a level 3 priority.	

**Requirement 62 – 062-HW4EG**

<b>062-HW4EG</b>	<b>Write to flash memory</b>	<b>13.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To reprogram the flash memory.	
<b>WHY</b>	To be able to reconfigure the system with new software.	
<b>Acceptance Criteria</b>	Interface to support reprogramming of flash.	
<b>Comments</b>	Long-term goals make this a level 4 priority.	

**Requirement 63 – 063-HW3HD**

<b>063-HW3HD</b>	<b>Casing for ECU</b>	<b>13.01.2015</b>
<b>Status</b>	Pending	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To have a casing for the ECU.	
<b>WHY</b>	To protect the motherboard and improve EMC performance.	
<b>Acceptance Criteria</b>	Shall have mounting holes. Connectors must be at edge of the board.	
<b>Comments</b>	Long-term goals make this a level 3 priority.	

**Requirement 64 – 064-HW3EG**

<b>064-HW3EG</b>	<b>Number of fuel injectors</b>	<b>14.01.2015</b>
<b>Status</b>	<b>Fulfilled</b>	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To control up to 8 fuel injectors.	
<b>WHY</b>	To control up to 8 cylinder engines.	
<b>Acceptance Criteria</b>	Shall have interface to 8 fuel injectors.	
<b>Comments</b>		

**Requirement 65 – 065-HW3JS**

<b>065-HW3JS</b>	<b>Number of ignition coils</b>	<b>14.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To control up to 8 ignition coils	
<b>WHY</b>	To control up to 8 cylinder engines.	
<b>Acceptance Criteria</b>	Shall have interface for 8 ignition coils.	
<b>Comments</b>		

**Requirement 66 – 066-HW2EG**

<b>066-HW2EG</b>	<b>Low side switching</b>	<b>14.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Hardware engineer	
<b>WANTS</b>	To activate actuators by low side switching.	
<b>WHY</b>	To improve safety for end user and maintenance.	
<b>Acceptance Criteria</b>	All actuator outputs shall have low side switching.	
<b>Comments</b>		

**Requirement 67 – 067-HW3OA**

<b>067-HW3OA</b>	<b>Exhaust oxygen sensor</b>	<b>14.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Hardware engineer	
<b>WANTS</b>	To have input support for up to 2 exhaust oxygen sensors.	
<b>WHY</b>	To evaluate the quality of the combustion for up to 8 cylinder engine with catalytic converter.	
<b>Acceptance Criteria</b>	Shall interface for 2 exhaust oxygen sensors.	
<b>Comments</b>		

**Requirement 68 – 068-HW3HD**

<b>068-HW3HD</b>	<b>Exhaust temperature sensors</b>	<b>14.01.2015</b>
<b>Status</b>	Pending	
<b>WHO</b>	Hardware engineer	
<b>WANTS</b>	To have input support for up to 8 exhaust temperature sensors.	
<b>WHY</b>	To evaluate the individual quality of the combustion process for up to 8 cylinder engine.	
<b>Acceptance Criteria</b>	Shall have interface for up to 8 exhaust temperature sensors.	
<b>Comments</b>		



**Requirement 69 – 069-HW3JS**

<b>069-HW3JS</b>	<b>Camshaft position sensors</b>	<b>14.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Hardware engineer	
<b>WANTS</b>	To have input support for up to 4 camshaft position sensors.	
<b>WHY</b>	System needs to know the engine position to activate ignition and fuel injection.	
<b>Acceptance Criteria</b>	Shall interface for 4 camshaft position sensors.	
<b>Comments</b>		

**Requirement 70 – 070-HW3HD**

<b>070-HW3HD</b>	<b>Crankshaft revolutions sensors</b>	<b>14.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Hardware engineer	
<b>WANTS</b>	To have input support for crankshaft revolution sensor.	
<b>WHY</b>	System needs to know the engine position to activate ignition and to determine engine speed.	
<b>Acceptance Criteria</b>	Shall have interface for 1 crankshaft revolution sensor.	
<b>Comments</b>		

**Requirement 71 – 071-HW3OA**

<b>071-HW3OA</b>	<b>Manifold Air Flow sensor</b>	<b>14.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Hardware engineer	
<b>WANTS</b>	To have input support for Manifold Air Flow (MAF)	
<b>WHY</b>	System needs to know the amount of air injected to the engine, and the air temperature.	
<b>Acceptance Criteria</b>	Shall have interface for Manifold Air Flow (MAF)	
<b>Comments</b>		

**Requirement 72 – 072-HW3JS**

<b>072-HW3JS</b>	<b>Manifold Air Pressure</b>	<b>14.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Hardware engineer	
<b>WANTS</b>	To have input support for Manifold Air Pressure (MAP).	
<b>WHY</b>	System needs to know the pressure in the inlet manifold to determine engine load.	
<b>Acceptance Criteria</b>	Shall have interface for Manifold Air Pressure (MAP).	
<b>Comments</b>		

**Requirement 73 – 073-HW3EG**

<b>073-HW3EG</b>	<b>Throttle Position Sensor (TPS)</b>	<b>14.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Hardware engineer	
<b>WANTS</b>	To have input support for Throttle Position Sensor (TPS)	
<b>WHY</b>	System needs to know the position of the throttle plate.	
<b>Acceptance Criteria</b>	Shall have interface for TPS.	
<b>Comments</b>		

**Requirement 74 – 074-HW3JS**

<b>074-HW3JS</b>	<b>Fuel Pressure Sensor (FPS)</b>	<b>14.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Hardware engineer	
<b>WANTS</b>	To have input support for Fuel Pressure Sensor (FPS)	
<b>WHY</b>	System needs to know the fuel pressure to deliver correct fuel level.	
<b>Acceptance Criteria</b>	Shall have interface for FPS.	
<b>Comments</b>		

**Requirement 75 – 075-HW3EG**

<b>075-HW3EG</b>	<b>Oil Pressure Sensor (OPS)</b>	<b>14.01.2015</b>
<b>Status</b>	Pending	
<b>WHO</b>	Hardware engineer	
<b>WANTS</b>	To have input support for Oil Pressure Sensor (OPS)	
<b>WHY</b>	System needs to know the oil pressure to ensure that the engine is running properly.	
<b>Acceptance Criteria</b>	Shall have interface for OPS.	
<b>Comments</b>		

**Requirement 76 – 076-HW3HD**

<b>076-HW3HD</b>	<b>Coolant temperature sensor (CTS)</b>	<b>14.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Hardware engineer	
<b>WANTS</b>	To have input support for Coolant Temperature Sensor (CTS)	
<b>WHY</b>	System needs to know the coolant temperature to adjust fuel injection and ignition timing.	
<b>Acceptance Criteria</b>	Shall have interface for CTS.	
<b>Comments</b>		

**Requirement 77 – 077-HW3JS**

<b>077-HW3JS</b>	<b>Oil Temperature sensor (OTS)</b>	<b>14.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Hardware engineer	
<b>WANTS</b>	To have input support for Oil Temperature Sensor (OTS)	
<b>WHY</b>	System needs to know the oil temperature to ensure that the engine is not being overheated.	
<b>Acceptance Criteria</b>	Shall have interface for OTS.	
<b>Comments</b>		

**Requirement 78 – 078-HW5HD**

<b>078-HW5HD</b>	<b>Knock sensor</b>	<b>14.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Hardware engineer	
<b>WANTS</b>	To have input support for 2 knock sensors.	
<b>WHY</b>	System needs to know if there is any ignition knocking, which is dangerous for the engine.	
<b>Acceptance Criteria</b>	Shall have interface for 2 knock sensors.	
<b>Comments</b>		

**Requirement 79 – 079-HW3HD**

<b>079-HW3HD</b>	<b>Peak Pressure Point Sensor (PPPS)</b>	<b>14.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Hardware engineer	
<b>WANTS</b>	To have input support for up to 8 PPPS.	
<b>WHY</b>	System needs to know when there is a pressure peak inside the cylinder, so that the ignition can be done at the proper time.	
<b>Acceptance Criteria</b>	Shall have interface for up to 8 PPPS.	
<b>Comments</b>		

**Requirement 80 – 080-HW4JS**

<b>080-HW4JS</b>	<b>Rollover Sensor</b>	<b>14.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Hardware engineer	
<b>WANTS</b>	To have input support for rollover sensor.	
<b>WHY</b>	System needs to know when the car has rolled over, so the engine power can be cut.	
<b>Acceptance Criteria</b>	Shall have interface for a rollover sensor.	
<b>Comments</b>		

**Requirement 81 – 081-HW4HD**

<b>081- HW4OA</b>	<b>Industrial temperature range</b>	<b>14.01.2015</b>
<b>Status</b>	Changed	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To use components that satisfies industrial temperature range specifications.	
<b>WHY</b>	Extended operational temperature components are required to function is required in this application.	
<b>Acceptance Criteria</b>	All components used must be meet or exceed the industrial temperature range: -40°C to 85°C.	
<b>Comments</b>	Succeeded by 1081 – HW4OA in the detailed requirements specification document.	

**Requirement 82 – 082-HW3HD**

<b>082-HW3HD</b>	<b>Fuel pump</b>	<b>14.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Hardware engineer	
<b>WANTS</b>	To have output support for a fuel pump.	
<b>WHY</b>	Provide the engine with fuel.	
<b>Acceptance Criteria</b>	Shall have interface for 1 fuel pump.	
<b>Comments</b>		

**Requirement 83 – 083-HW3JS**

<b>083-HW3JS</b>	<b>Idle Air Control</b>	<b>14.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Hardware engineer	
<b>WANTS</b>	To have support for idle air valve.	
<b>WHY</b>	To allow the engine to stay idle.	
<b>Acceptance Criteria</b>	Shall have interface for 1 idle air valve.	
<b>Comments</b>		

**Requirement 84 – 084-HW4EG**

<b>084-HW4EG</b>	<b>Boost controller</b>	<b>14.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Hardware engineer	
<b>WANTS</b>	To have support for 2 boost controller.	
<b>WHY</b>	To support twin turbo engines.	
<b>Acceptance Criteria</b>	Shall have interface for 2 boost controller.	
<b>Comments</b>		



**Requirement 85 – 085-HW5JS**

<b>085-HW5JS</b>	<b>Tachometer</b>	<b>14.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Hardware engineer	
<b>WANTS</b>	To have support for RPM display unit.	
<b>WHY</b>	To measure the RPM of the engine.	
<b>Acceptance Criteria</b>	Shall have interface for a RPM display unit.	
<b>Comments</b>		

**Requirement 86 – 086-HW5OA**

<b>086-HW5OA</b>	<b>Pneumatic Valve lifters</b>	<b>14.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Hardware engineer	
<b>WANTS</b>	To have support for pneumatic valve control.	
<b>WHY</b>	This removes the timing belt and camshaft, also the valves can be tuned, allowing economic and max effect.	
<b>Acceptance Criteria</b>	Shall have interface to support Pneumatic Valve control.	
<b>Comments</b>		

**Requirement 87 – 087-HW3JS**

<b>087-HW3JS</b>	<b>CAN Bus</b>	<b>14.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Hardware engineer	
<b>WANTS</b>	To have support for CAN bus.	
<b>WHY</b>	To be able to expand functionalities and establish hierarchy communication between important part of the system.	
<b>Acceptance Criteria</b>	Shall have interface for CAN bus.	
<b>Comments</b>		

**Requirement 88 – 088-HW3EG**

<b>088-HW3EG</b>	<b>Power</b>	<b>14.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Hardware engineer	
<b>WANTS</b>	To supply the motherboard with power.	
<b>WHY</b>	To supply the electronics on the motherboard.	
<b>Acceptance Criteria</b>	Shall have interface for a power supply.	
<b>Comments</b>		

**Requirement 89 – 089-HW4HD**

<b>089-HW4HD</b>	<b>Ignition</b>	<b>14.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Hardware engineer	
<b>WANTS</b>	To have an ignition signal to activate the external circuitry.	
<b>WHY</b>	To have control of the engine start	
<b>Acceptance Criteria</b>	Shall have interface to determine ignition status.	
<b>Comments</b>		

**Requirement 90 – 090-HW3OA**

<b>090-HW3OA</b>	<b>Filtered ADC inputs</b>	<b>14.01.2015</b>
<b>Status</b>	Changed	
<b>WHO</b>	Hardware Engineer	
<b>WANTS</b>	To filter all inputs to Analog-to-Digital Converters (ADCs).	
<b>WHY</b>	No filters on inputs will cause aliasing issues due to high frequency noise.	
<b>Acceptance Criteria</b>	All ADC channels have filters.	
<b>Comments</b>	Succeeded by 1090 – HW3OA in the detailed requirements specification document.	

**Requirement 91 – 091-HW4OA**

<b>091-HW4OA</b>	<b>Electrostatic Discharge (ESD) protection</b>	<b>13.01.2015</b>
<b>Status</b>	Abandoned	
<b>WHO</b>	ISO 10605:2008(E)	
<b>WANTS</b>	To protect automotive connectors for ESD.	
<b>WHY</b>	ESD from contact with external connectors can cause damage to hardware.	
<b>Acceptance Criteria</b>	All external headers or connectors have ESD protection that meets or exceeds 15kV contact model.	
<b>Comments</b>	Abandoned: Test failed	

**Requirement 92 – 092-HW4OA**

<b>092-HW4OA</b>	<b>JTAG interface</b>	<b>13.01.2015</b>
<b>Status</b>	Changed	
<b>WHO</b>	Software developers	
<b>WANTS</b>	To access a JTAG header.	
<b>WHY</b>	JTAG is an industry standard for programming and debugging software.	
<b>Acceptance Criteria</b>	There is a standard JTAG header or test points on the finished board.	
<b>Comments</b>	Succeeded by 1092 – HW4OA in the detailed requirements specification document	

**Requirement 93 – 093-HW5OA**

<b>093-HW5OA</b>	<b>Power Good (PG) LEDs</b>	<b>13.01.2015</b>
<b>Status</b>	Changed	
<b>WHO</b>	Test personnel	
<b>WANTS</b>	To see if the different voltages on the board is functioning.	
<b>WHY</b>	It is easy to detect faults in power management with PG LEDs (Power Good LEDs).	
<b>Acceptance Criteria</b>	There are LEDs for each voltage source on the board indicating that each source is functioning.	
<b>Comments</b>	Succeeded by 1093 – HW5OA in the detailed requirements specification document	

**Requirement 94 – 094-HW4OA**

<b>094-HW4OA</b>	<b>RoHS Compliance</b>	<b>13.01.2015</b>
<b>Status</b>	Changed	
<b>WHO</b>	RoHS Directive	
<b>WANTS</b>	To ensure that all components are RoHS compliant.	
<b>WHY</b>	To reduce the amount of toxic substances used in electronics.	
<b>Acceptance Criteria</b>	All components and manufacturing methods used for the system must be RoHS compliant.	
<b>Comments</b>	Succeeded by 1094 – HW4OA in the detailed requirements specification document	

**Requirement 95 – 095-HW3OA**

<b>095-HW3OA</b>	<b>Sufficient decoupling</b>	<b>13.01.2015</b>
<b>Status</b>	Changed	
<b>WHO</b>	Hardware Engineer	
<b>WANTS</b>	To ensure that all components are decoupled.	
<b>WHY</b>	Decoupling reduces electromagnetic interference (EMI) in components and signals.	
<b>Acceptance Criteria</b>	All individual power pins are decoupled with at least one capacitor per power pin.	
<b>Comments</b>	Succeeded by 1095 – HW3OA in the detailed requirements specification document	

**Requirement 96 – 096-HW3OA**

<b>096-HW3OA</b>	<b>Power design simulations</b>	<b>13.01.2015</b>
<b>Status</b>	Changed	
<b>WHO</b>	Hardware engineer	
<b>WANTS</b>	To simulate all onboard power devices.	
<b>WHY</b>	Simulation is an important tool to design power management and verify results.	
<b>Acceptance Criteria</b>	All onboard power sources are properly simulated using SPICE software under different loading figures.	
<b>Comments</b>	Succeeded by 1096 – HW3OA in the detailed requirements specification document	

**Requirement 97 – 097-HW4OA**

<b>097-HW4OA</b>	<b>ADC filter simulations</b>	<b>13.01.2015</b>
<b>Status</b>	Changed	
<b>WHO</b>	Hardware engineer	
<b>WANTS</b>	To simulate all ADC filters.	
<b>WHY</b>	Simulation is an important tool to verify frequency response and dampening requirements in the filters.	
<b>Acceptance Criteria</b>	All onboard ADC input channel filters are properly simulated using SPICE software.	
<b>Comments</b>	Succeeded by 1097 – HW4OA in the detailed requirements specification document	

**Requirement 98 – 098-HW4OA**

<b>098-HW4OA</b>	<b>Output MOSFET drives simulations</b>	<b>13.01.2015</b>
<b>Status</b>	Changed	
<b>WHO</b>	Hardware engineer	
<b>WANTS</b>	To simulate all onboard MOSFET driver circuitry.	
<b>WHY</b>	Simulation is an important tool to verify the design.	
<b>Acceptance Criteria</b>	All onboard MOSFET drives are properly simulated using SPICE software under different loading figures.	
<b>Comments</b>	Succeeded by 1098 – HW4OA in the detailed requirements specification document	

**Requirement 99 – 099-HW5OA**

<b>099-HW5OA</b>	<b>Reset toggle</b>	<b>13.01.2015</b>
<b>Status</b>	Changed	
<b>WHO</b>	Software developer	
<b>WANTS</b>	To reset all components on the board without cutting power.	
<b>WHY</b>	Resetting components without having to cut power is beneficial for issues in software when prototyping.	
<b>Acceptance Criteria</b>	There is either a reset switch or jumper present on the board for triggering reset functions in components.	
<b>Comments</b>	Succeeded by 1099 – HW5OA in the detailed requirements specification document	

**Requirement 100 – 100-HW5OA**

<b>100-HW5OA</b>	<b>Programmable LEDs</b>	<b>13.01.2015</b>
<b>Status</b>	Changed	
<b>WHO</b>	Software developer	
<b>WANTS</b>	To be able to toggle LEDs on the board in software.	
<b>WHY</b>	Programmable LEDs are useful when testing and debugging code.	
<b>Acceptance Criteria</b>	There is at least one programmable LED on the board.	
<b>Comments</b>	Succeeded by 1100 – HW5OA in the detailed requirements specification document	



**Requirement 101 – 101-HW3OA**

<b>101-HW3OA</b>	<b>Fiducial markers</b>	<b>13.01.2015</b>
<b>Status</b>	Pending	
<b>WHO</b>	Manufacturer	
<b>WANTS</b>	To have access to at least 3 visible fiducial markers.	
<b>WHY</b>	Fiducial markers are required for pick and place (P&P) assembly.	
<b>Acceptance Criteria</b>	There are at least 3 fiducial markers visible on the board.	
<b>Comments</b>	Succeeded by 1101 – HW3OA in the detailed requirements specification document	

**Requirement 102 – 102-HW3OA**

<b>102-HW3OA</b>	<b>Thermal reliefs</b>	<b>13.01.2015</b>
<b>Status</b>	Changed	
<b>WHO</b>	Manufacturer	
<b>WANTS</b>	To ensure that component pads are thermally relieved.	
<b>WHY</b>	Without thermal relief, components may not be soldered properly to the board.	
<b>Acceptance Criteria</b>	All component pads and holes connected to large copper planes must be thermally relieved.	
<b>Comments</b>	Succeeded by 1102 – HW3OA in the detailed requirements specification document	

**Requirement 103 – 103-HW3OA**

<b>103-HW3OA</b>	<b>Part availability</b>	<b>13.01.2015</b>
<b>Status</b>	Changed	
<b>WHO</b>	Purchaser (for manufacture)	
<b>WANTS</b>	To ensure that all electronic parts are available.	
<b>WHY</b>	All parts must be available for the product to be manufactured.	
<b>Acceptance Criteria</b>	Ensure that all electrical components used in the design are available and is not at the end of its life-cycle.	
<b>Comments</b>	Succeeded by 1103 – HW3OA in the detailed requirements specification document	

**Requirement 104 – 104-HW4OA**

<b>104-HW4OA</b>	<b>Ethernet debugging</b>	<b>13.01.2015</b>
<b>Status</b>	Changed	
<b>WHO</b>	Software developer	
<b>WANTS</b>	To debug software over Ethernet	
<b>WHY</b>	Debugging over Ethernet makes it simpler during the debug process to collaborate	
<b>Acceptance Criteria</b>	Allow access to Ethernet debug features in hardware design with a physical connector.	
<b>Comments</b>	Succeeded by 1104 – HW4OA in the detailed requirements specification document	

**Requirement 105 – 105-HW5OA**

<b>105-HW5OA</b>	<b>USB programming interface</b>	<b>14.01.2015</b>
<b>Status</b>	Changed	
<b>WHO</b>	Software developer	
<b>WANTS</b>	To program the ECU using a USB port.	
<b>WHY</b>	USB is an interface used in most computers and is used much for programming and debugging.	
<b>Acceptance Criteria</b>	There is a USB interface accessible with a USB port on the finished board that has access to debugging features.	
<b>Comments</b>	Succeeded by 1105 – HW5OA in the detailed requirements specification document	

**Requirement 106 – 106-HW2OA**

<b>106-HW2OA</b>	<b>3.3V voltage level</b>	<b>14.01.2015</b>
<b>Status</b>	Changed	
<b>WHO</b>	Hardware engineer	
<b>WANTS</b>	To use a stable 3.3V DC supply on the board.	
<b>WHY</b>	3.3V is a standard operational voltage used by many integrated circuits (ICs) and signals.	
<b>Acceptance Criteria</b>	There is at least one 3.3V DC source on the board to supply all components that require 3.3V.	
<b>Comments</b>	Succeeded by 1106 – HW2OA in the detailed requirements specification document	

**Requirement 107 – 107-HW2OA**

<b>107-HW2OA</b>	<b>5V voltage level</b>	<b>14.01.2015</b>
<b>Status</b>	Changed	
<b>WHO</b>	Hardware engineer	
<b>WANTS</b>	To use a stable 5V DC supply on the board.	
<b>WHY</b>	5V is a standard operational voltage used by many integrated circuits (ICs) and signals.	
<b>Acceptance Criteria</b>	There is at least one 5V DC source on the board to supply all components that require 5V.	
<b>Comments</b>	Succeeded by 1107 – HW2OA in the detailed requirements specification document	

**Requirement 108 – 108-HW4OA**

<b>108-HW4OA</b>	<b>Electrostatic Discharge (ESD) protection</b>	<b>15.01.2015</b>
<b>Status</b>	Abandoned	
<b>WHO</b>	ISO 10605:2008(E)	
<b>WANTS</b>	To protect automotive connectors for ESD.	
<b>WHY</b>	ESD from contact with external connectors can cause damage to hardware.	
<b>Acceptance Criteria</b>	All external headers or connectors have ESD protection that meets or exceeds 25kV air discharge model.	
<b>Comments</b>	Abandoned: Test failed	

**Requirement 109 – 109-HW4OA**

<b>109-HW4OA</b>	<b>0603 SMD component size standards</b>	<b>23.01.2015</b>
<b>Status</b>	Changed	
<b>WHO</b>	Hardware engineer	
<b>WANTS</b>	To use a standard size for discrete components.	
<b>WHY</b>	The size of components are important for manufacture and rework purposes, and reduces the amount of different components in use.	
<b>Acceptance Criteria</b>	Discrete components such as resistors, capacitors etc. have a minimum size of 0603 or larger (if required).	
<b>Comments</b>	Succeeded by 1109 – HW4OA in the detailed requirements specification document	

**Requirement 110 – 110-HW4OA**

<b>110-HW4OA</b>	<b>Do not mount (DNM) components</b>	<b>23.01.2015</b>
<b>Status</b>	Changed	
<b>WHO</b>	Manufacturer	
<b>WANTS</b>	To have a clear indication of which components not to mount.	
<b>WHY</b>	In designs, many components maybe added for later modification but not mounted during first manufacturing run.	
<b>Acceptance Criteria</b>	All components that are DNM in initial design shall have visible DNM description in schematics. DNM components shall be listed. DNM components shall not be mounted on the manufactured board.	
<b>Comments</b>	Succeeded by 1110 – HW4OA in the detailed requirements specification document	

**Requirement 111 – 111-SY3HD**

<b>111-SY3HD</b>	<b>Max RPM</b>	<b>30.01.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To support engines that rotates at a speed up to 16.000 RPM	
<b>WHY</b>	To support a wide range of engines for motorsport.	
<b>Acceptance Criteria</b>	Components must be able to support engine speed to 16.000 RPM.	
<b>Comments</b>		

**Requirement 112 – 112-SY1EG**

<b>112-SY1EG</b>	<b>Design an ECU</b>	<b>03.03.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To regulate the combustion process of an engine	
<b>WHY</b>	Create an exciting and multidisciplinary assignment.	
<b>Acceptance Criteria</b>	Have a full schematic design for an engine control system.	
<b>Comments</b>		

**Requirement 113 – 113-SW3DH**

<b>113-SW3DH</b>	<b>Android application</b>	<b>03.03.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To create an app with graphical user interface (GUI) for tablet.	
<b>WHY</b>	To allow user to interact with ECU through graphical icons and visual indicators.	
<b>Acceptance Criteria</b>	Check if the app is running on the tablet without any exception or crash.	
<b>Comments</b>		

**Requirement 114 – 114-HW2HD**

<b>114-HW2HD</b>	<b>Cadence Allegro</b>	<b>03.03.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To have the design done in Cadence Allegro	
<b>WHY</b>	The software is desired by KDA.	
<b>Acceptance Criteria</b>	Schematic drawn in Cadence Allegro	
<b>Comments</b>		

**Requirement 115 – 115-SY3EG**

<b>115-SY3EG</b>	<b>Cylinder pressure characteristics</b>	<b>03.03.2015</b>
<b>Status</b>	Pending	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To monitor pressure characteristics inside the cylinders.	
<b>WHY</b>	To enhance the ignition timing in the cylinders.	
<b>Acceptance Criteria</b>	Shall be able to read the pressure characteristics inside each cylinder.	
<b>Comments</b>	Long term goal makes this a level 3 priority.	

**Requirement 116 – 116-SW2DH**

<b>116-SW2DH</b>	<b>Install PetaLinux on Zynq-TE0720</b>	<b>03.03.2015</b>
<b>Status</b>	Fulfilled	
<b>WHO</b>	Kongsberg Defence & Aerospace	
<b>WANTS</b>	To install and configure PetaLinux on Zynq-TE0720.	
<b>WHY</b>	To customize, build and deploy Embedded Linux solutions on Xilinx processing systems.	
<b>Acceptance Criteria</b>	Have the latest version of PetaLinux 2014.4 and the Zynq-TE0720.	
<b>Comments</b>		





## 2. Requirements Changelog

Table 1 – Requirements Changelog Part 1

Requirement ID	Action	Date	Responsible
001-SY1EG	Changed; Succeeded by 112-SY1EG	03.03.2015	HD
002-SY1EG	Passed	06.05.2015	EG
003-SY1EG	Passed	07.05.2015	EG
004-SY1EG	Passed	07.05.2015	EG
005-SY1EG	Not tested	12.01.2015	EG
006-SY1EG	Passed	07.05.2015	EG
007-SY3EG	Not tested	12.01.2015	EG
008-SY3EG	Not tested	12.01.2015	EG
009-SY3EG	Not tested	12.01.2015	EG
010-SY3EG	Passed	07.05.2015	EG
011-SY3EG	Passed	07.05.2015	EG
012-SY3EG	Passed	07.05.2015	EG
013-SY4EG	Not tested	13.01.2015	EG
014-SY4EG	Not tested	13.01.2015	EG
015-SY5EG	Passed	07.05.2015	EG
016-SY3EG	Passed	07.05.2015	EG
017-SY4EG	Passed	07.05.2015	EG
018-SY4EG	Passed	07.05.2015	EG
019-SY5EG	Not tested	14.01.2015	EG
020-SY3EG	Passed	07.05.2015	EG
021-SY3EG	Not tested	14.01.2015	EG
022-SY4EG	Not tested	14.01.2015	EG
023-SW4SA	Passed	06.05.2015	SA
024-SW3SA	Passed	06.05.2015	SA
025-SW5SA	Not tested	12.01.2015	SA
026-SW2SA	Passed	06.05.2015	SA
027-SW2DH	Changed; Succeeded by 113-SWDH	03.03.2015	HD
028-SW5DH	Passed	06.05.2015	DH
029-SW4DH	Changed; Succeeded by 1029-SW4DH	13.01.2015	DH
030-SW4SA	Changed; Succeeded by 1030-SW4SA	13.01.2015	SA
031-SW3DH	Passed	13.01.2015	DH
032-SW3SA	Passed	13.01.2015	DH
033-SW3DH	Not tested	13.01.2015	DH
034-SW5SA	Changed; Succeeded by 1034-SW5SA	13.01.2015	SA
035-SW5SA	Changed; Succeeded by 1035-SW5SA	13.01.2015	SA
036-SW5SA	Passed	11.05.2015	SA
037-SW5DH	Not tested	14.01.2015	DH
038-SW5SA	Passed	11.05.2015	SA
039-SW4DH	Passed	06.05.2015	SA
040-SW4DH	Passed	06.05.2015	SA



Table 2 – Requirements Changelog Part 2

Requirement ID	Action	Date	Responsible
041-SW5DH	Not tested	14.01.2015	SA
042-SW3DH	Passed	06.05.2015	DH
043-SW5SA	Changed; Succeeded by 1043-SW5SA	15.01.2015	SA
044 SW5SA	Passed	08.05.2015	SA
045 SW4SA	Passed	08.05.2015	SA
046-SW3SA	Passed	08.05.2015	SA
047-HW2HD	Passed	08.05.2015	HD
048-HW2JS	Not tested	13.01.2015	JS
049-HW2HD	Changed; Succeeded by 114-HW2HD	03.03.2015	HD
050-HW3JS	Passed	06.05.2015	JS
051-HW3HD	Passed	06.05.2015	HD
052-HW3EG	Passed	06.05.2015	EG
053-HW2JS	Not tested	13.01.2015	JS
054-HW3HD	Not tested	13.01.2015	HD
055-HW3EG	Changed; Succeeded by 115-HW3EG	03.03.2015	HD
056-HW4HD	Passed	06.05.2015	HD
057-HW3JS	Not tested	13.01.2015	JS
058-HW3JS	Passed	06.05.2015	JS
059-HW3HD	Passed	06.05.2015	HD
060-HW3EG	Passed	06.05.2015	EG
061-HW3HD	Not tested	13.01.2015	HD
062-HW4EG	Passed	06.05.2015	EG
063-HW3HD	Not tested	13.01.2015	HD
064-HW3EG	Passed	06.05.2015	EG
065-HW3JS	Passed	06.05.2015	JS
066-HW2EG	Passed	06.05.2015	EG
067-HW3OA	Passed	06.05.2015	OA
068-HW3HD	Not tested	14.01.2015	HD
069-HW3JS	Passed	06.05.2015	JS
070-HW3HD	Passed	06.05.2015	HD
071-HW3OA	Passed	06.05.2015	OA
072-HW3JS	Passed	06.05.2015	JS
073-HW3EG	Passed	06.05.2015	EG
074-HW3JS	Passed	06.05.2015	JS
075-HW3EG	Not tested	14.01.2015	EG
076-HW3HD	Passed	06.05.2015	HD
077-HW3JS	Passed	06.05.2015	JS
078-HW5HD	Passed	06.05.2015	HD
079-HW3HD	Passed	06.05.2015	HD
080-HW4JS	Passed	06.05.2015	JS



Table 3 – Requirements Changelog Part 3

Requirement ID	Action	Date	Responsible
081-HW4OA	Changed; Succeeded by 1081-HW4OA	14.01.2015	JS
082-HW3HD	Passed	06.05.2015	HD
083-HW3JS	Passed	06.05.2015	HD
084-HW3EG	Passed	06.05.2015	JS
085-HW5JS	Passed	06.05.2015	EG
086-HW5OA	Passed	06.05.2015	JS
087-HW3JS	Passed	06.05.2015	OA
088-HW3EG	Passed	06.05.2015	JS
089-HW4HD	Passed	06.05.2015	EG
090-HW3OA	Changer; Succeeded by 1090-HW3OA	14.01.2015	HD
091-HW4OA	Failed	06.05.2015	OA
092-HW4OA	Changed; Succeeded by 1092-HW4OA	13.01.2015	OA
093-HW5OA	Changed; Succeeded by 1093-HW5OA	13.01.2015	OA
094-HW4OA	Changed; Succeeded by 1094-HW4OA	13.01.2015	OA
095-HW3OA	Changed; Succeeded by 1095-HW3OA	13.01.2015	OA
096-HW3OA	Changed; Succeeded by 1096-HW3OA	13.01.2015	OA
097-HW4OA	Changed; Succeeded by 1097-HW4OA	13.01.2015	OA
098-HW4OA	Changed; Succeeded by 1098-HW4OA	13.01.2015	OA
099-HW5OA	Changed; Succeeded by 1099-HW5OA	13.01.2015	OA
100-HW5OA	Changed; Succeeded by 1100-HW5OA	13.01.2015	OA
101-HW3OA	Changed; Succeeded by 1101-HW3OA	13.01.2015	OA
102-HW3OA	Changed; Succeeded by 1102-HW3OA	13.01.2015	OA
103-HW3OA	Changed; Succeeded by 1103-HW3OA	13.01.2015	OA
104-HW4OA	Changed; Succeeded by 1104-HW4OA	13.01.2015	OA
105-HW5OA	Changed; Succeeded by 1105-HW5OA	14.01.2015	OA
106-HW2OA	Changed; Succeeded by 1106-HW2OA	14.01.2015	OA
107-HW2OA	Changed; Succeeded by 1107-HW2OA	14.01.2015	OA
108-HW4OA	Failed	06.05.2015	OA
109-HW4OA	Changed; Succeeded by 1109-HW4OA	16.01.2015	OA
110-HW4OA	Changed; Succeeded by 1110-HW4OA	16.01.2015	OA
111-SY3HD	Passed	06.05.2015	HD
112-SY1EG	Passed	06.05.2015	HD
113-SW3DH	Passed	06.05.2015	HD
114-HW2HD	Passed	06.05.2015	HD
115-SY3EG	Not tested	03.03.2015	HD
116-SW2DH	Passed	06.05.2015	DH



KONGSBERG



## Test Specification

### KPEC

<b>Employer</b>	Kongsberg Defence & Aerospace			
<b>Group Members</b>	<b>Name</b>		<b>Initials</b>	
	Ola Pedersen Aasheim		OA	
	Salahuddin Asjad		SA	
	Hung Dinh		HD	
	Dler Hasan		DH	
	Even Gudbrandsen		EG	
	Jannik Schäffer		JS	
<b>Document information</b>	<b>Revision</b>	<b>Date</b>	<b>Approved</b>	<b>Pages</b>
	4.0	18.05.2015	JS	68





## Abstract

The formative document defines systems, software and hardware test specification, which is a continuation of the concept of operations document. The overall purpose of this document is to describe the needs for how to test the requirements. After reading this document, one should be able to understand how to test each requirement in order to verify that the requirement is tested.



## Revision Table

Version	Date	Approval	Description
1.0	14.01.2015	JS	<ul style="list-style-type: none"> <li>Created the document</li> </ul>
2.0	26.01.2015	DH	<ul style="list-style-type: none"> <li>Revised and published</li> </ul>
2.1	30.01.2015	HD	<ul style="list-style-type: none"> <li>Changed test requirement 29, 30, 34, 35, 43, 81, 90-107, 109 and 110.</li> </ul>
2.2	03.03.2015	HD	Revised after 1 iteration: <ul style="list-style-type: none"> <li>Changed test 1, 27, 49 and 55.</li> </ul>
3.0	13.03.2015	DH	<ul style="list-style-type: none"> <li>Revised and published</li> </ul>
4.0	18.05.2015	DH	<ul style="list-style-type: none"> <li>Revised : Test update</li> </ul>

## Contents

Abstract .....	2
Revision Table .....	3
List of tables .....	3
List of test tables.....	3
1. Introduction .....	7
2. Test results tables .....	66

## List of tables

Table 1 - Requirement test status Part 1 .....	66
Table 2 - Requirement test status Part 2.....	67
Table 3 - Requirements Changelog Part 3.....	68

## List of test tables

Test 1 - 001-SY1EG.....	8
Test 2 - 002-SY1EG.....	8
Test 3 - 003-SY1EG.....	9
Test 4 - 004-SY1EG.....	9
Test 5 - 005-SY1EG.....	10
Test 6 - 006-SY1EG.....	10
Test 7 - 007-SY3EG.....	11



Test 8 - 008-SY3EG.....	11
Test 9 - 009-SY3EG.....	12
Test 10 - 010-SY3EG.....	12
Test 11 - 011-SY3EG.....	13
Test 12 - 012-SY3EG.....	13
Test 13 - 013-SY4EG.....	14
Test 14 - 014-SY4EG.....	14
Test 15 - 015-SY5EG.....	15
Test 16 - 016-SY3JS.....	15
Test 17 - 017-SY4EG.....	16
Test 18 - 018-SY4EG.....	16
Test 19 - 019-SY5EG.....	17
Test 20 - 020-SY3EG.....	17
Test 21 - 021-SY3EG.....	18
Test 22 - 022-SY4EG.....	18
Test 23 - 023-SW4SA.....	19
Test 24 - 024-SW3SA.....	19
Test 25 - 025-SW5SA.....	20
Test 26 - 026-SW2SA.....	20
Test 27 - 027-SW2DH.....	21
Test 28 - 028-SW5DH.....	21
Test 29 - 029-SW4DH.....	22
Test 30 - 030-SW4SA.....	22
Test 31 - 031-SW3DH.....	23
Test 32 - 032-SW3SA.....	23
Test 33 - 033-SW3DH.....	24
Test 34 - 034-SW5SA.....	24
Test 35 - 035-SW5SA.....	25
Test 36 - 036-SW5SA.....	25
Test 37 - 037-SW5DH.....	26
Test 38 - 038-SW5SA.....	26
Test 39 - 039-SW4DH.....	27
Test 40 - 040-SW4DH.....	27
Test 41 - 041-SW5DH.....	28
Test 42 - 042-SW3DH.....	28
Test 43 - 043-SW5SA.....	29
Test 44 - 044-SW5SA.....	29
Test 45 - 045-SW4SA.....	30
Test 46 - 046-SW3SA.....	30
Test 47 - 047-HW2HD.....	31
Test 48 - 048-HW2JS.....	31



Test 49 - 049-HW2HD.....	32
Test 50 - 050-HW3JS .....	32
Test 51 - 051-HW3HD.....	33
Test 52 - 052-HW3EG.....	33
Test 53 - 053-HW2JS.....	34
Test 54 - 054-HW3HD.....	34
Test 55 - 055-HW3EG.....	35
Test 56 - 056-HW4HD.....	35
Test 57 - 057-HW3JS.....	36
Test 58 - 058-HW3JS.....	36
Test 59 - 059-HW3HD.....	37
Test 60 - 060-HW3EG.....	37
Test 61 - 061-HW3HD.....	38
Test 62 - 062-HW4EG .....	38
Test 63 - 063-HW3HD.....	39
Test 64 - 064-HW3EG .....	39
Test 65 - 065-HW3JS.....	40
Test 66 - 066-HW2EG.....	40
Test 67 - 067-HW3OA.....	41
Test 68 - 068-HW3HD.....	41
Test 69- 069-HW3JS.....	42
Test 70 - 070-HW3HD .....	42
Test 71 - 071-HW3OA .....	43
Test 72 - 072-HW3JS.....	43
Test 73 - 073-HW3EG .....	44
Test 74 - 074-HW3JS.....	44
Test 75 - 075-HW3EG.....	45
Test 76 - 076-HW3HD.....	45
Test 77 - 077-HW3JS.....	46
Test 78 - 078-HW5HD.....	46
Test 79 - 079-HW3HD.....	47
Test 80 - 080-HW4JS.....	47
Test 81 - 081-HW4JS.....	48
Test 82 - 082-HW3HD .....	48
Test 83 - 083-HW3JS.....	49
Test 84 - 084-HW4EG.....	49
Test 85 - 085-HW5JS.....	50
Test 86 - 086-HW5OA.....	50
Test 87 - 087-HW3JS.....	51
Test 88 - 088-HW3EG.....	51
Test 89 - 089-HW4HD.....	52





Test 90 - 090-HW3OA .....	52
Test 91 - 091-HW4OA .....	53
Test 92 - 092-HW4OA .....	53
Test 93 - 093-HW5OA .....	54
Test 94 - 094-HW4OA .....	54
Test 95 - 095-HW3OA .....	55
Test 96 - 096-HW3OA .....	55
Test 97 - 097-HW4OA .....	56
Test 98 - 098-HW4OA .....	56
Test 99 - 099-HW5OA .....	57
Test 100 - 100-HW5OA .....	57
Test 101 - 101-HW3OA .....	58
Test 102 - 102-HW3OA .....	58
Test 103 - 103-HW3OA .....	59
Test 104 - 104-HW4OA .....	59
Test 105 - 105-HW5OA .....	60
Test 106 - 106-HW2OA .....	60
Test 107 - 107-HW2OA .....	61
Test 108 - 108-HW4OA .....	61
Test 109 - 109-HW4OA .....	62
Test 110 - 110-HW4OA .....	62
Test 111 - 111-SY3HD .....	63
Test 112 - 112-SY1EG .....	63
Test 113 - 113-SW3DH .....	64
Test 114 - 114-HW2HD .....	64
Test 115 - 115-SY3EG .....	65
Test 116 - 116-SW2DH .....	65



## 1. Introduction

The following section of the document lists all the test specifications in tables. Please refer to *Requirements Guidelines* for a complete explanation, reason and specification for how the requirements shall be written and how they are identified.

There are two kind of test specification; there is one for the requirement specification and one for the detailed requirement specification. This document contains the test for the requirement specification.

The complete list of changes in requirements is found in the **Table 1 - Requirement test status** in the end of this document.

**Test 1 - 001-SY1EG**

<b>001-SY1EG</b>	<b>Design an ECU</b>	<b>14.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	Have a full electronic design for an engine control system.	
<b>Test</b>	Verify that the design files contain all the component and interfaces to regulate the combustion process in an engine.	
<b>Comments</b>	Succeeded by 112-SY1EG in the test specification.	

**Test 2 - 002-SY1EG**

<b>002-SY1EG</b>	<b>FPGA based ECU</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	FPGA based engine control unit.	
<b>Test</b>	Verify that the ECU is based on a FPGA.	
<b>Comments</b>	The Trenz te0720 micro module utilize FPGA technology	

**Test 3 - 003-SY1EG**

<b>003-SY1EG</b>	<b>Schematic design</b>	<b>07.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Complete schematics design file.	
<b>Test</b>	Verify that schematic for ECU design is complete.	
<b>Comments</b>	The schematic design is reviewed by KDA and KPEC, this is the first revision of the design, more rigorous testing will be necessary on the PCB to reveal bugs. At this stage the schematic design is complete.	

**Test 4 - 004-SY1EG**

<b>004-SY1EG</b>	<b>Tablet communication - Read</b>	<b>07.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Must be able to read IO values	
<b>Test</b>	Send signal to an input, check if tablet can read.	
<b>Comments</b>	Long term goal. The internal temperature sensor implemented in the XADC, transmit data to the tablet application.	

**Test 5 - 005-SY1EG**

<b>005-SY1EG</b>	<b>Tablet communication - Write</b>	<b>14.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	Must be able to adjust control parameters.	
<b>Test</b>	Verify that a change in control parameter through the tablet, changes the parameter on the measuring device.	
<b>Comments</b>	Long term goal. Not possible to test at this stage of the project.	

**Test 6 - 006-SY1EG**

<b>006-SY1EG</b>	<b>Support for up to 8 cylinder engines</b>	<b>07.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Must be able to handle IO for up to 8 cylinders.	
<b>Test</b>	Verify that there is IO support for up to: <ul style="list-style-type: none"><li>• 8 Ignition coils</li><li>• 8 Fuel injections</li><li>• 4 camshaft sensor</li><li>• 8 cylinder pressure sensors</li><li>• 8 exhaust temperature sensors</li><li>• 2 lambda sensors</li></ul>	
<b>Comments</b>	The schematic design has implemented all this sensors and actuators.	

**Test 7 - 007-SY3EG**

<b>007-SY3EG</b>	<b>Engine performance tuning</b>	<b>14.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	Parameter change results in a system response.	
<b>Test</b>	Verify that the engine characteristics alter when changing regulation parameters.	
<b>Comments</b>	Long-term goals make this a level 3 priority.	

**Test 8 - 008-SY3EG**

<b>008-SY3EG</b>	<b>Engine performance tuning from tablet</b>	<b>14.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	Parameter change results in a system response.	
<b>Test</b>	Verify that the engine characteristics alter when changing regulation parameters via tablet.	
<b>Comments</b>	Long-term goals make this a level 3 priority.	

**Test 9 - 009-SY3EG**

<b>009-SY3EG</b>	<b>Communication with external devices</b>	<b>14.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	Must have a system for interfacing external devices.	
<b>Test</b>	Verify that there is a system, and that it is working as intended. E.g. CAN-BUS.	
<b>Comments</b>	Long-term goals make this a level 3 priority.	

**Test 10 - 010-SY3EG**

<b>010-SY3EG</b>	<b>Petrol fuel option</b>	<b>07.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	The system must have interface towards petrol as fuel.	
<b>Test</b>	Consult the documentation to verify that we have IO to support petrol as fuel.	
<b>Comments</b>	Long-term goals make this a level 3 priority. There are implemented IO in the schematic design to drive an external fuel pump, there is also implemented a flex fuel sensor input to determine whether or not the fuel supplied is petrol.	

**Test 11 - 011-SY3EG**

<b>011-SY3EG</b>	<b>Bioethanol fuel option</b>	<b>07.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	The system must have interface towards bioethanol as fuel	
<b>Test</b>	Consult the documentation to verify that we have IO to support bioethanol as fuel.	
<b>Comments</b>	Long-term goals make this a level 3 priority. There are implemented IO in the schematic design to drive an external fuel pump, there is also implemented a flex fuel sensor input to determine whether or not the fuel supplied is bioethanol.	

**Test 12 - 012-SY3EG**

<b>012-SY3EG</b>	<b>Support for Peak Pressure Point (PPP)</b>	<b>07.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Shall interface sensors to determine peak pressure point.	
<b>Test</b>	Verify that there is IO for PPP measuring devices.	
<b>Comments</b>	There are both IO and circuitry to support PPP measurement with help of the Cylinder Pressure Sensors(CPS).	



**Test 13 - 013-SY4EG**

<b>013-SY4EG</b>	<b>Injector flowrate adjustment</b>	<b>15.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	Shall be able to adjust injector flowrate according to process data.	
<b>Test</b>	Change the process data, and verify that the injector-signal alters duration.	
<b>Comments</b>	Long-term goals make this a level 4 priority.	

**Test 14 - 014-SY4EG**

<b>014-SY4EG</b>	<b>Dynamic fuel map adjustment</b>	<b>15.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	Must have interface to allow mapping.	
<b>Test</b>	Change environmental parameters to verify that the map changes accordingly.	
<b>Comments</b>	Long-term goals make this a level 4 priority.	

**Test 15 - 015-SY5EG**

<b>015-SY5EG</b>	<b>Support for pneumatic valve control module</b>	<b>07.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Verify in the design that there is an interface for an external pneumatic valve control module.	
<b>Test</b>	Consult the documentation to verify that there is an interface to handle communication with external module.	
<b>Comments</b>	Long-term goals make this a level 5 priority. There are implemented a dedicated can transceiver in the schematic design P. 29. for the external valve module.	

**Test 16 - 016-SY3JS**

<b>016-SY3JS</b>	<b>Tablet</b>	<b>07.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Choose a tablet that supports interfacing with the ECU.	
<b>Test</b>	Install application on tablet and try to establish connection to ECU. Verify that there is connection.	
<b>Comments</b>	Samsung galaxy tab S 10.5" was the tablet chosen to run the application, the tablet runs the Android kitkat 4.4.2 which support hybrid application development. Communication is established.	

**Test 17 - 017-SY4EG**

<b>017-SY4EG</b>	<b>Engine start/stop</b>	<b>07.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Must have IO support for ignition signal.	
<b>Test</b>	Consult the documentation to verify that there is dedicated IO to handle engine start/stop.	
<b>Comments</b>	IO and circuitry are implemented in the schematic design P.11	

**Test 18 - 018-SY4EG**

<b>018-SY4EG</b>	<b>Engine stop when rolling over</b>	<b>07.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Must have IO support for rollover sensor.	
<b>Test</b>	Consult the documentation to verify that there is dedicated IO to handle roll over.	
<b>Comments</b>	The rollover sensor has been implemented in the schematic design P.10 with use of an accelerometer.	

**Test 19 - 019-SY5EG**

<b>019-SY5EG</b>	<b>Unintended startup</b>	<b>19.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	Must have IO support for unintended startup.	
<b>Test</b>	Activate driveline, see if engine starts. Simulate rollover, see if engine starts.  The point in both cases, is to have systems to prevent the engine to start when the driveline is engaged and when the car has rolled.	
<b>Comments</b>	Long-term goals make this a level 5 priority.	

**Test 20 - 020-SY3EG**

<b>020-SY3EG</b>	<b>Engine speed</b>	<b>07.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Shall have IO support for reading engine speed value.	
<b>Test</b>	Consult the documentation to verify that there is dedicated IO to handle engine speed measurement.	
<b>Comments</b>	There is implemented a driver circuit to provide an external circuitry with tachometer data on P.1 in the schematic design.	

**Test 21 - 021-SY3EG**

<b>021-SY3EG</b>	<b>Fail-safe protection</b>	<b>19.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	Must have a system for engine protection	
<b>Test</b>	Remove one of the essential sensors, and verify that the system enable fail-safe routine.	
<b>Comments</b>	Long term goal.	

**Test 22 - 022-SY4EG**

<b>022-SY4EG</b>	<b>Self-check</b>	<b>19.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	Must have a surveillance system for fault detection in IO.	
<b>Test</b>	Run self-check and verify that all components are responding.	
<b>Comments</b>	Long term goal.	

**Test 23 - 023-SW4SA**

<b>023-SW4SA</b>	<b>GitHub – Version control</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Github shall be used for version control.	
<b>Test</b>	Consult the KPEC GitHub account to verify the presence of our SW files.	
<b>Comments</b>	The SW files for the entire project are present in the Github account, files are dated back to 5/2 on the day of inspection; 30.04.15.	

**Test 24 - 024-SW3SA**

<b>024-SW3SA</b>	<b>Ethernet communication</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Ethernet communication shall work between Zynq and tablet.	
<b>Test</b>	Verify that the Ethernet communication is established between Zynq and tablet, transmit and receive data. For example program the Zynq to read a text-string on the tablet.	
<b>Comments</b>	Communication is established, the micro module are generating numbers which are routed through the Zynq. The numbers are visually represented on the tablet.	

**Test 25 - 025-SW5SA**

<b>025-SW5SA</b>	<b>WI-Fi communication</b>	<b>20.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	Wi-Fi communication shall work between Zynq and tablet.	
<b>Test</b>	Verify that the Wi-Fi communication is established between Zynq and tablet, transmit and receive data. For example program the Zynq to read a text-string on the tablet.	
<b>Comments</b>	This feature cannot be tested before the PCB circuit has been realized.	

**Test 26 - 026-SW2SA**

<b>026-SW2SA</b>	<b>Xilinx Vivado – Zynq programming</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Zynq SoC is programmed through configurations from Xilinx Vivado.	
<b>Test</b>	Verify that the Zynq accept configurations from Xilinx Vivado.	
<b>Comments</b>	The test engineer has visually inspected the command ifconfig, to verify that an IP address is established and static.	

**Test 27 - 027-SW2DH**

<b>027-SW2DH</b>	<b>Android application</b>	<b>22.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	Check if the app is running on the tablet without any exception or crash.	
<b>Test</b>	Verify that there is an app on the tablet running without any bugs.	
<b>Comments</b>	Succeeded by 113-SWDH in the test specification.	

**Test 28 - 028-SW5DH**

<b>028-SW5DH</b>	<b>Reconfigure FPGA over Ethernet</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	The old bitstream data replaced by a new bitstream file using Ethernet.	
<b>Test</b>	Update SPI flash over Ethernet.	
<b>Comments</b>	On inspection, the SPI flash was updated. A file was transferred from an FTP server to the SPI flash over Ethernet. The transfer succeeded.	



**Test 29 - 029-SW4DH**

<b>029-SW4DH</b>	<b>Android Studio</b>	<b>20.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	Android Studio shall be used for app development and debugging.	
<b>Test</b>	Verify that the tablet can run the developed application.	
<b>Comments</b>	Succeeded by 1029 – SW4DH in the detailed test requirements specification document.	

**Test 30 - 030-SW4SA**

<b>030-SW4SA</b>	<b>Android version support</b>	<b>20.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	Configure the SDK and implement the right API to the app.	
<b>Test</b>	Verify that the tablet is running the supported features for API 19.	
<b>Comments</b>	Succeeded by 1030 – SW4SA in the detailed test requirements specification document.	

**Test 31 - 031-SW3DH**

<b>031-SW3DH</b>	<b>History log</b>	<b>20.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	Display a history log on the tablet of different parameters from the ECU.	
<b>Test</b>	Verify that a parameter history log from the ECU can be read on the tablet.	
<b>Comments</b>	The tablet receives data from the TE0720, verified.	

**Test 32 - 032-SW3SA**

<b>032-SW3SA</b>	<b>Implement FPGA module as IP</b>	<b>06.05.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	Shall use Vivado IP integrator to generate block diagrams.	
<b>Test</b>	Use the IP catalog in Vivado to create block diagrams. Run synthesis to verify that the implemented IP is working successfully.	
<b>Comments</b>	Synthesis run smoothly, processing system successfully implemented with IP blocks.	

**Test 33 - 033-SW3DH**

<b>033-SW3DH</b>	<b>Tuning from tablet</b>	<b>20.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	Parameter change using tuning function will result into a system response.	
<b>Test</b>	Change process parameters via the graphical user interface to verify that it alters through ECU to engine.	
<b>Comments</b>	Long-term makes this a level 3 priority. Not ready to test yet (06.05.15)	

**Test 34 - 034-SW5SA**

<b>034-SW5SA</b>	<b>Toolbar inside the application</b>	<b>20.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	The toolbar layout shall be properly functioning with access to the icons.	
<b>Test</b>	Verify that the application has a toolbar. Verify that there is access to the icons.	
<b>Comments</b>	Succeeded by 1034 – SW5SA in the detailed test requirements specification document.	

**Test 35 - 035-SW5SA**

<b>035-SW5SA</b>	<b>Navigation drawer inside the application</b>	<b>20.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	The navigation drawer layout shall be functioning with an accessible menu.	
<b>Test</b>	Verify that the navigation drawer is functioning and responding correctly.	
<b>Comments</b>	Succeeded by 1035 – SW5SA in the detailed test requirements specification document.	

**Test 36 - 036-SW5SA**

<b>036-SW5SA</b>	<b>Low battery consumption</b>	<b>11.05.2015</b>												
<b>Status</b>	Passed													
<b>Acceptance Criteria</b>	Inactive activities and services shall not drain battery.													
<b>Test</b>	Use Traceview to profile performance activities.													
<b>Comments</b>	<p>The application has been simulated with android debugging device; DDMS. Three scenarios was tested: 1.Full operation 2.Standby with Zynq communication 3.standby without Zynq communication.</p> <table> <tr> <th>App</th><th>Cpu</th><th>State</th></tr> <tr> <td>16%</td><td>3%</td><td>Full operation</td></tr> <tr> <td>3%</td><td>1%</td><td>Stby w/Zynq com.</td></tr> <tr> <td>1%</td><td>0%</td><td>Stby wo/Zynq com.</td></tr> </table>		App	Cpu	State	16%	3%	Full operation	3%	1%	Stby w/Zynq com.	1%	0%	Stby wo/Zynq com.
App	Cpu	State												
16%	3%	Full operation												
3%	1%	Stby w/Zynq com.												
1%	0%	Stby wo/Zynq com.												

**Test 37 - 037-SW5DH**

<b>037-SW5DH</b>	<b>Communication security</b>	<b>20.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	Use WPA2 encryption with uppercase-, lowercase- and special letters.	
<b>Test</b>	Verify that the acceptance criteria are met and that the communication only can be established when it is supposed to.	
<b>Comments</b>	Not possible without the physical circuit board. The test board does not contain a wifi module.	

**Test 38 - 038-SW5SA**

<b>038-SW5SA</b>	<b>Tablet compatibility</b>	<b>11.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Design layouts for various screen sizes. Min.2	
<b>Test</b>	Test the application on min. 2 different tablets to verify that it is operational.	
<b>Comments</b>	The application runs smoothly on both 10.5" and 12" tablets.	

**Test 39 - 039-SW4DH**

<b>039-SW4DH</b>	<b>Flash write</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Must be able to write into the flash memory.	
<b>Test</b>	Use the terminal, write something to the memory and then access the address location in the flash memory and read the content.	
<b>Comments</b>	The test engineer have sent a file to the flash memory through the ftp server with the use of the putty terminal. Flash write successful.	

**Test 40 - 040-SW4DH**

<b>040-SW4DH</b>	<b>Read flash</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Must be able to read from the flash memory.	
<b>Test</b>	Use the terminal, access the address location in the flash memory, and then read the content.	
<b>Comments</b>	The test engineer have read the same file as we uploaded to the flash memory, flash read successful.	

**Test 41 - 041-SW5DH**

<b>041-SW5DH</b>	<b>Fail-safe function for engine</b>	<b>06.05.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	Shall be able to detect absence from a sensor.	
<b>Test</b>	Verify that the ECU disables IO to the fuel pump, ignition and injectors when critical sensor is disconnected or out of specified range. Verify that the ECU prioritize this interrupt over other normal interrupts.	
<b>Comments</b>	Not possible to test without a physical circuitry.	

**Test 42 - 042-SW3DH**

<b>042-SW3DH</b>	<b>Database</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Shall have read and write access to database.	
<b>Test</b>	Verify that you can read from the database. Verify that you can write to the database.	
<b>Comments</b>	The tablet applications runs on data from the database, and the Zynq write errors to the database.	

**Test 43 - 043-SW5SA**

<b>043-SW5SA</b>	<b>Phonegap</b>	<b>20.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	Phonegap shall be used for app development and debugging.	
<b>Test</b>	Verify that the tablet can run the developed application.	
<b>Comments</b>	Succeeded by 1043 – SW4DH in the detailed test requirements specification document.	

**Test 44 - 044-SW5SA**

<b>044-SW5SA</b>	<b>Real-time view</b>	<b>08.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	The tablet screen shall be able to view different parameters within 50 ms.	
<b>Test</b>	Verify that the acceptance criteria are met using wireshark software.	
<b>Comments</b>	The application are simulated in the Firefox web browser, a performance check are executed in order to reveal the update speed of the data viewed in the application. The average update speed is 43.7fps, which translates to $1/43.7\text{Hz}=22.8\text{ms}$ . The application supports out definition of real time view.	



**Test 45 - 045-SW4SA**

<b>045-SW4SA</b>	<b>Real-time view</b>	<b>08.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	The tablet screen shall be able to view different parameters within 100 ms.	
<b>Test</b>	Verify that the acceptance criteria are met using wireshark software.	
<b>Comments</b>	The application are simulated in the Firefox web browser, a performance check are executed in order to reveal the update speed of the data viewed in the application. The average update speed is 43.7fps, which translates to $1/43.7\text{Hz}=22.8\text{ms}$ . The application supports out definition of real time view.	

**Test 46 - 046-SW3SA**

<b>046-SW3SA</b>	<b>Real-time view</b>	<b>08.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	The tablet screen shall be able to view different parameters within 200 ms.	
<b>Test</b>	Verify that the acceptance criteria are met using wireshark software.	
<b>Comments</b>	The application are simulated in the Firefox web browser, a performance check are executed in order to reveal the update speed of the data viewed in the application. The average update speed is 43.7fps, which translates to $1/43.7\text{Hz}=22.8\text{ms}$ . The application supports out definition of real time view.	

**Test 47 - 047-HW2HD**

<b>047-HW2HD</b>	<b>Xilinx Zynq</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Design must contain a micromodule with a Xilinx Zynq.	
<b>Test</b>	Consult the documentation to verify that there is a micromodule with a Xilinx Zynq.	
<b>Comments</b>	The design implements a Xilinx zynq, ref. design document, chapter 6. P. 153.	

**Test 48 - 048-HW2JS**

<b>048-HW2JS</b>	<b>Motherboard</b>	<b>20.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	The micromodule shall mate with the motherboard.	
<b>Test</b>	Verify that the motherboard mates with the connectors of the micromodule.	
<b>Comments</b>	Not possible to test before a physical circuit board are manufactured.	

**Test 49 - 049-HW2HD**

<b>049-HW2HD</b>	<b>Cadence Allegro</b>	<b>20.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	Schematic and PCB layout drawn in Cadence Allegro.	
<b>Test</b>	Verify that the design files are .dsn and that the board file is .brd format.	
<b>Comments</b>	Succeeded by 114-HW2HD in the test specification.	

**Test 50 - 050-HW3JS**

<b>050-HW3JS</b>	<b>Fuel injectors</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Separate drivers for each injector.	
<b>Test</b>	Consult the documentation to verify that there are separate drivers for each injector.	
<b>Comments</b>	8. Individual injector drivers are implemented in the schematic design p.34.	

**Test 51 - 051-HW3HD**

<b>051-HW3HD</b>	<b>Individual ignition coils</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Separate drivers for each spark plug.	
<b>Test</b>	Consult the documentation to verify that there are separate drivers for each spark plug.	
<b>Comments</b>	8. Individual coil drivers are implemented in the schematic design p.32.	

**Test 52 - 052-HW3EG**

<b>052-HW3EG</b>	<b>Exhaust temperature</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	I/O must support up to 8 exhaust temperature sensors.	
<b>Test</b>	Consult the documentation to verify that there is IO for up to 8 exhaust temperature sensors.	
<b>Comments</b>	A temperature driver IC is implemented in the schematic design P.8. 8 channels are assigned exhaust temperature.	

**Test 53 - 053-HW2JS**

<b>053-HW2JS</b>	<b>Read analog quantities</b>	<b>20.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	The system is able to read analog quantities.	
<b>Test</b>	Verify that the ECU is able to read analog quantities, either by reading IO using Vivado or the application on the tablet.	
<b>Comments</b>	Not possible to test before a physical circuit board are manufactured.	

**Test 54 - 054-HW3HD**

<b>054-HW3HD</b>	<b>Oxygen level</b>	<b>20.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	The system is able to read oxygen level in exhaust.	
<b>Test</b>	Verify that the ECU is able to read oxygen level in exhaust, either by reading IO using Vivado or the application on the tablet.	
<b>Comments</b>	Not possible to test before a physical circuit board are manufactured.	

**Test 55 – 055-HW3EG**

<b>055-HW3EG</b>	<b>Cylinder pressure characteristics</b>	<b>20.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	Shall be able to read the pressure characteristics inside each cylinder.	
<b>Test</b>	Verify that the ECU is able to read the pressure characteristics inside each cylinder, either by reading IO using Vivado or the application on the tablet.	
<b>Comments</b>	Succeeded by 115-HW3EG Not possible to test before a physical circuit board are manufactured.	

**Test 56 - 056-HW4HD**

<b>056-HW4HD</b>	<b>Dedicated GPIO</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Dedicated GPIO available.	
<b>Test</b>	Consult the documentation to verify that there is dedicated GPIO available.	
<b>Comments</b>	3 dedicated analog input channels are assigned general input in the schematic design P.16. 2 Digital inputs P.11. 4 dedicated low side drivers P. 36. 4 General purpose digital output pins P.26.	

**Test 57 - 057-HW3JS**

<b>057-HW3JS</b>	<b>Manifold air pressure</b>	<b>20.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	Shall be able to measure the pressure inside the inlet manifold.	
<b>Test</b>	Verify that the ECU is able to measure the pressure inside the inlet manifold, either by reading IO using Vivado or the application on the tablet.	
<b>Comments</b>	Not possible to test before a physical circuit board are manufactured.	

**Test 58 - 058-HW3JS**

<b>058-HW3JS</b>	<b>Communication</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Communication interface included in the design.	
<b>Test</b>	Consult the documentation to verify that there is a communication interface in the design.	
<b>Comments</b>	There are 4 individual communication platforms implanted in the schematic design, Ethernet & wifi interface, USB interface, Can interface and JTAG interface. P. 26.	

**Test 59 - 059-HW3HD**

<b>059-HW3HD</b>	<b>CAN communication</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Shall have a CAN-transceiver implemented.	
<b>Test</b>	Consult the documentation to verify that there is a CAN-transceiver implemented in the design.	
<b>Comments</b>	Can transceiver implemented in the schematic design P. 29.	

**Test 60 - 060-HW3EG**

<b>060-HW3EG</b>	<b>Flexi fuel</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	System to detect fuel type and I/O to manage fuel selection.	
<b>Test</b>	Consult the documentation to verify that there is a system to detect fuel and IO to manage fuel selection.	
<b>Comments</b>	In the schematic design there is implemented a dedicated digital flex fuel input pin in the x-bay connector P.41. IO to manage fuel selection is not necessarily a matter for the ECU, but it is possible to configure general purpose IO to manage the fuel selection.	



**Test 61 - 061-HW3HD**

<b>061-HW3HD</b>	<b>Peak pressure point (PPP)</b>	<b>20.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	Shall be able to read pressure characteristics inside the cylinder.	
<b>Test</b>	Verify that the ECU is able to measure the PPP, either by reading IO using Vivado or the application on the tablet.	
<b>Comments</b>	Long-term goals make this a level 3 priority. Not possible to test before a physical circuit board are manufactured.	

**Test 62 - 062-HW4EG**

<b>062-HW4EG</b>	<b>Write to flash memory</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Interface to support reprogramming of flash.	
<b>Test</b>	Consult the schematics to verify that there is an interface to reprogram flash.	
<b>Comments</b>	Long-term goals make this a level 4 priority. There are implemented Ethernet, wifi and JTAG which can be used to flash the micromodule. P.27 and 30.	

**Test 63 - 063-HW3HD**

<b>063-HW3HD</b>	<b>Casing for ECU</b>	<b>20.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	Shall have mounting holes. Connectors must be at edge of the board.	
<b>Test</b>	Check the gerber files to verify that there is mounting holes and that connectors must be at the edge of the board.	
<b>Comments</b>	Long-term goals make this a level 3 priority. Not included in the deliverables this year.	

**Test 64 - 064-HW3EG**

<b>064-HW3EG</b>	<b>Number of fuel injectors</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Shall have interface to 8 fuel injectors.	
<b>Test</b>	Consult the schematics to verify that there are interface to support 8 fuel injectors.	
<b>Comments</b>	8 fuel injector drivers implemented in the schematic design P. 34.	

**Test 65 - 065-HW3JS**

<b>065-HW3JS</b>	<b>Number of ignition coils</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Shall have interface for 8 ignition coils.	
<b>Test</b>	Consult the schematics to verify that there is interface for 8 ignition coils.	
<b>Comments</b>	8 coil drivers implemented in the schematic design P. 32.	

**Test 66 - 066-HW2EG**

<b>066-HW2EG</b>	<b>Low side switching</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	All actuator outputs shall have low side switching.	
<b>Test</b>	Check the schematics to verify that all signal switching performed on the low side of the actuator.	
<b>Comments</b>	All actuator drivers are per definition low side switches.	

**Test 67 - 067-HW3OA**

<b>067-HW3OA</b>	<b>Exhaust oxygen sensor</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Shall interface for 2 exhaust oxygen sensors.	
<b>Test</b>	Consult the schematics to verify that there are interface for 2 exhaust oxygen sensors.	
<b>Comments</b>	The lambda sensor interface are implemented in the schematic design P.1 and 2.	

**Test 68 - 068-HW3HD**

<b>068-HW3HD</b>	<b>Exhaust temperature sensors</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Shall have interface for up to 8 exhaust temperature sensors.	
<b>Test</b>	Consult the schematics to verify that there are interface to up to 8 exhaust temperature sensors.	
<b>Comments</b>	There are 8 dedicated temperature sensor channels in the schematic design P. 8.	

**Test 69– 069-HW3JS**

<b>069-HW3JS</b>	<b>Camshaft position sensors</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Shall interface for 4 camshaft position sensors.	
<b>Test</b>	Consult the schematics to verify that there are interface for 4 camshaft position sensors.	
<b>Comments</b>	There are 4 dedicated input channels with level shift implemented in the schematic design P.13.	

**Test 70 - 070-HW3HD**

<b>070-HW3HD</b>	<b>Crankshaft revolutions sensors</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Shall have interface for 1 crankshaft revolution sensor.	
<b>Test</b>	Consult the schematic to verify that there is an interface for 1 crankshaft revolution sensor.	
<b>Comments</b>	The flywheel sensor interface is implemented in the schematic design on P. 12.	

**Test 71 - 071-HW3OA**

<b>071-HW3OA</b>	<b>Manifold Air Flow sensor</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Shall have interface for Manifold Air Flow (MAF).	
<b>Test</b>	Consult the schematics to verify that there is an interface for Manifold Air Flow (MAF).	
<b>Comments</b>	MAF digital input pin on P. 11.	

**Test 72 - 072-HW3JS**

<b>072-HW3JS</b>	<b>Manifold Air Pressure</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Shall have interface for Manifold Air Pressure (MAP).	
<b>Test</b>	Consult the schematics to verify that there is an interface for MAP sensor.	
<b>Comments</b>	MAP analog input on P. 17.	

**Test 73 - 073-HW3EG**

<b>073-HW3EG</b>	<b>Throttle Position Sensor (TPS)</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Shall have interface for TPS.	
<b>Test</b>	Consult the schematics to verify that there is an interface for TPS.	
<b>Comments</b>	Interface for TPS are implemented in the schematic design on P. 35.	

**Test 74 - 074-HW3JS**

<b>074-HW3JS</b>	<b>Fuel Pressure Sensor (FPS)</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Shall have interface for FPS.	
<b>Test</b>	Consult the schematics to verify that there is an interface for FPS.	
<b>Comments</b>	Fuel pressure sensor interface implemented in the schematic design on P.17	

**Test 75 - 075-HW3EG**

<b>075-HW3EG</b>	<b>Oil Pressure Sensor (OPS)</b>	<b>20.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	Shall have interface for OPS.	
<b>Test</b>	Consult the schematic to verify that there is an interface for OPS.	
<b>Comments</b>	Oil pressure sensor interface implemented on P. 17.	

**Test 76 - 076-HW3HD**

<b>076-HW3HD</b>	<b>Coolant temperature sensor (CTS)</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Shall have interface for CTS.	
<b>Test</b>	Consult the schematics to verify that there is an interface for CTS.	
<b>Comments</b>	Schematic implementation of CTS on P.8.	



**Test 77 - 077-HW3JS**

<b>077-HW3JS</b>	<b>Oil Temperature sensor (OTS)</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Shall have interface for OTS.	
<b>Test</b>	Consult the schematics to verify that there is an interface for OTS.	
<b>Comments</b>	Schematic implementation of OTS on P.8.	

**Test 78 - 078-HW5HD**

<b>078-HW5HD</b>	<b>Knock sensor</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Shall have interface for 2 knock sensors.	
<b>Test</b>	Consult the schematics to verify that there are interface for knock sensors.	
<b>Comments</b>	The knock sensor interface has been implemented in the schematic design on P. 9.	

**Test 79 - 079-HW3HD**

<b>079-HW3HD</b>	<b>Peak Pressure Sensor (PPP)</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Shall have interface for up to 8 PPS.	
<b>Test</b>	Consult the schematics to verify that there are interface for 8 PPS.	
<b>Comments</b>	The Cylinder pressure interface circuit has been implemented in the schematic design on P. 19 and 20.	

**Test 80 - 080-HW4JS**

<b>080-HW4JS</b>	<b>Rollover Sensor</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Shall have interface for a rollover sensor.	
<b>Test</b>	Consult the schematics to verify that there is an interface for a rollover sensor.	
<b>Comments</b>	There are no dedicated external rollover sensor interface, there are however implemented an internal gyroscope to cover the same functionality. If an external sensor is necessary, the general input pins can be assigned.	

**Test 81 - 081-HW4JS**

<b>081- HW4OA</b>	<b>Industrial temperature range</b>	<b>24.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	All components used must be meet or exceed the industrial temperature range: -40°C to 85°C.	
<b>Test</b>	Verify that all components in BOM meets or exceeds operational temperature of -40°C to 85°C (Industrial temperature range).	
<b>Comments</b>	Succeeded by 1081 – HW4OA in the detailed test requirements specification document.	

**Test 82 - 082-HW3HD**

<b>082-HW3HD</b>	<b>Fuel pump</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Shall have interface for 1 fuel pump.	
<b>Test</b>	Consult the schematics to verify that there is an interface for a fuel pump.	
<b>Comments</b>	Low side switch implemented in the schematic design on P. 36.	

**Test 83 - 083-HW3JS**

<b>083-HW3JS</b>	<b>Idle Air Control</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Shall have interface for 1 idle air valve.	
<b>Test</b>	Consult the schematics to verify that there is an interface for idle air valve.	
<b>Comments</b>	There are general purpose low side switches implemented on P.36 in the schematic design.	

**Test 84 - 084-HW4EG**

<b>084-HW4EG</b>	<b>Boost controller</b>	<b>20.01.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Shall have interface for 2 boost controller.	
<b>Test</b>	Consult the schematics to verify that there are interface for 2 boost controller.	
<b>Comments</b>	There are implemented a boost controller circuitry in the schematic design, one dedicated and 4 general circuits.	

**Test 85 - 085-HW5JS**

<b>085-HW5JS</b>	<b>Tachometer</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Shall have interface for a RPM display unit.	
<b>Test</b>	Consult the documentation to verify that there is an interface for a RPM display unit.	
<b>Comments</b>	Tachometer analog output are available in the schematic design on P. 1.	

**Test 86 - 086-HW5OA**

<b>086-HW5OA</b>	<b>Pneumatic Valve lifters</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Shall have interface to support Pneumatic Valve control.	
<b>Test</b>	Consult the documentation to verify that there is an interface to support Pneumatic Valve control.	
<b>Comments</b>	On P.29 in the schematic design, there is implemented a dedicated Can transceiver for the valve control module.	

**Test 87 - 087-HW3JS**

<b>087-HW3JS</b>	<b>CAN Bus</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Shall have interface for CAN bus.	
<b>Test</b>	Consult the schematics to verify that the interface for CAN bus.	
<b>Comments</b>	Can bus transceiver implemented on P. 29. In the schematic design.	

**Test 88 - 088-HW3EG**

<b>088-HW3EG</b>	<b>Power</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Shall have interface for a power supply.	
<b>Test</b>	Consult the schematic to verify that there is an interface for a power supply.	
<b>Comments</b>	There are four power lines from VBAT through the x-bay connector in the schematic design on P. 41.	

**Test 89 - 089-HW4HD**

<b>089-HW4HD</b>	<b>Ignition</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Shall have interface to determine ignition status.	
<b>Test</b>	Consult the schematics to verify that there is an interface to determine ignition status.	
<b>Comments</b>	Ignition signal interface implemented in the schematic design on P. 11.	

**Test 90 - 090-HW3OA**

<b>090-HW3OA</b>	<b>Filtered ADC inputs</b>	<b>20.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	All ADC channels have filters.	
<b>Test</b>	Verify that schematics have filters on ADC inputs Verify that filters are mounted on ECU motherboard	
<b>Comments</b>	Succeeded by 1090 – HW3OA in the detailed test requirements specification document.	

**Test 91 - 091-HW4OA**

<b>091-HW4OA</b>	<b>Electrostatic Discharge (ESD) protection</b>	<b>06.05.2015</b>
<b>Status</b>	Failed	
<b>Acceptance Criteria</b>	All external headers or connectors have ESD protection that meets or exceeds 15kV contact model.	
<b>Test</b>	Consult the schematics that all headers or connectors have either components with integrated ESD measures or dedicated ESD components such as TVS diodes/arrays that meets or exceeds 15kV contact model.	
<b>Comments</b>	All signal tracks have ESD protection, but the power lines are not directly protected with any ESD measures.	

**Test 92 - 092-HW4OA**

<b>092-HW4OA</b>	<b>JTAG interface</b>	<b>21.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	There is a standard JTAG header or test points on the finished board.	
<b>Test</b>	Verify that JTAG is accessible through test points or a header in schematics and on the ECU motherboard	
<b>Comments</b>	Succeeded by 1092 – HW4OA in the detailed test requirements specification document.	



**Test 93 - 093-HW5OA**

<b>093-HW5OA</b>	<b>Power Good (PG) LEDs</b>	<b>21.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	There are LEDs for each voltage source on the board indicating that each source is functioning.	
<b>Test</b>	Consult the schematics if the power source has a PG function. Verify that there is a LED connected to the PG output (typically open collector configuration) in schematics and on the physical board.	
<b>Comments</b>	Succeeded by 1093 – HW5OA in the detailed test requirements specification document.	

**Test 94 - 094-HW4OA**

<b>094-HW4OA</b>	<b>RoHS Compliance</b>	<b>21.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	All components and manufacturing methods used for the system must be RoHS compliant.	
<b>Test</b>	Verify that materials used for production is RoHS compliant Verify that all components in Bill of Materials (BOM) are RoHS compliant	
<b>Comments</b>	Succeeded by 1094 – HW4OA in the detailed test requirements specification document.	

**Test 95 - 095-HW3OA**

<b>095-HW3OA</b>	<b>Sufficient decoupling</b>	<b>21.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	All individual power pins are decoupled with at least one capacitor per power pin.	
<b>Test</b>	Verify that all ICs have decoupling capacitors in schematics Verify that decoupling capacitors and/or filters are placed close to power pins on the ECU motherboard	
<b>Comments</b>	Some inputs may have additional filtering on power lines. Succeeded by 1095 – HW3OA in the detailed test requirements specification document.	

**Test 96 - 096-HW3OA**

<b>096-HW3OA</b>	<b>Power design simulations</b>	<b>21.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	All onboard power sources are properly simulated using SPICE software under different loading figures.	
<b>Test</b>	Verify simulation results by varying input voltages and loading figures. Verify that power devices meets voltage ripple and noise requirements.	
<b>Comments</b>	Succeeded by 1096 – HW3OA in the detailed test requirements specification document.	

**Test 97 - 097-HW4OA**

<b>097-HW4OA</b>	<b>ADC filter simulations</b>	<b>21.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	All onboard ADC input channel filters are simulated using SPICE software.	
<b>Test</b>	Verify that simulations are using correct components. Verify simulation results by varying input voltages and frequencies. Verify that the signal is dampened at least <b>half</b> of the ADC sampling frequency.	
<b>Comments</b>	Succeeded by 1097 – HW4OA in the detailed test requirements specification document.	

**Test 98 - 098-HW4OA**

<b>098-HW4OA</b>	<b>Output MOSFET drives simulations</b>	<b>21.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	All onboard MOSFET drives are properly simulated using SPICE software under different loading figures.	
<b>Test</b>	Verify that simulations are using correct components Verify simulation results by varying input voltages Verify simulation results by varying duty cycle (if appropriate)	
<b>Comments</b>	Succeeded by 1098 – HW4OA in the detailed test requirements specification document.	

**Test 99 - 099-HW5OA**

<b>099-HW5OA</b>	<b>Reset toggle</b>	<b>21.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	There is either a reset switch or jumper present on the board for triggering reset functions in components.	
<b>Test</b>	Verify that a reset switch or jumper is present on the ECU motherboard. Verify that the system resets when switch/jumper is triggered.	
<b>Comments</b>	Succeeded by 1099 – HW5OA in the detailed test requirements specification document.	

**Test 100 - 100-HW5OA**

<b>100-HW5OA</b>	<b>Programmable LEDs</b>	<b>21.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	There is at least one programmable LED on the board.	
<b>Test</b>	Verify that there is at least one programmable LED on the ECU motherboard. Verify that each LED is possible to toggle individually in software.	
<b>Comments</b>	Succeeded by 1100 – HW5OA in the detailed test requirements specification document.	

**Test 101 - 101-HW3OA**

<b>101-HW3OA</b>	<b>Fiducial markers</b>	<b>21.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	There are at least 3 fiducial markers visible on the board.	
<b>Test</b>	Consult the documents to verify the acceptance criteria. Verify that there is at least 3 visible fiducials on the ECU motherboard.	
<b>Comments</b>	Succeeded by 1101 – HW3OA in the detailed test requirements specification document.	

**Test 102 - 102-HW3OA**

<b>102-HW3OA</b>	<b>Thermal reliefs</b>	<b>21.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	All component pads and holes connected to large copper planes must be thermally relieved.	
<b>Test</b>	Control that component pads and plated holes are thermally relieved in board file (PCB design file)	
<b>Comments</b>	Results from manufacture may cover this test. Succeeded by 1102 – HW3OA in the detailed test requirements specification document.	

**Test 103 - 103-HW3OA**

<b>103-HW3OA</b>	<b>Part availability</b>	<b>21.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	Ensure that all electrical components used in the design are available and is not at the end of its life-cycle.	
<b>Test</b>	Check part availability and manufacture status for each component on Bill of Materials (BOM)	
<b>Comments</b>	Succeeded by 1103 – HW3OA in the detailed test requirements specification document.	

**Test 104 - 104-HW4OA**

<b>104-HW4OA</b>	<b>Ethernet debugging</b>	<b>21.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	Allow access to Ethernet debug features in hardware design with a physical connector.	
<b>Test</b>	Verify that there is a connector for Ethernet on the ECU motherboard. Verify that debug features are accessible using Ethernet.	
<b>Comments</b>	Succeeded by 1104 – HW4OA in the detailed test requirements specification document.	

**Test 105 - 105-HW5OA**

<b>105-HW5OA</b>	<b>USB programming interface</b>	<b>21.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	There is a USB interface accessible with a USB port on the finished board that has access to debugging features.	
<b>Test</b>	Verify that there is a connector for USB on the ECU motherboard Verify that debug features are accessible using USB	
<b>Comments</b>	Succeeded by 1105 – HW5OA in the detailed test requirements specification document.	

**Test 106 - 106-HW2OA**

<b>106-HW2OA</b>	<b>3.3V voltage level</b>	<b>21.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	There is at least one 3.3V DC source <b>on</b> the board to supply all components that require 3.3V.	
<b>Test</b>	Verify that voltage sources supplies sufficient current using power budget calculations. Use voltmeter/scope to measure and verify that voltage is approximately 3.3V, preferably under varying loads.	
<b>Comments</b>	Succeeded by 1106 – HW2OA in the detailed test requirements specification document.	

**Test 107 - 107-HW2OA**

<b>107-HW2OA</b>	<b>5V voltage level</b>	<b>21.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	There is at least one 5V DC source on the board to supply all components that require 5V.	
<b>Test</b>	Verify that voltage sources supplies sufficient current using power budget calculations. Use voltmeter/scope to measure and verify that voltage is approximately 5V, preferably under varying loads.	
<b>Comments</b>	Succeeded by 1107 – HW2OA in the detailed test requirements specification document.	

**Test 108 – 108-HW4OA**

<b>108-HW4OA</b>	<b>Electrostatic Discharge (ESD) protection</b>	<b>06.05.2015</b>
<b>Status</b>	Failed	
<b>Acceptance Criteria</b>	All external headers or connectors have ESD protection that meets or exceeds 25kV air discharge model.	
<b>Test</b>	Consult the schematics that all headers or connectors have either components with integrated ESD measures or dedicated ESD components such as TVS diodes/arrays that meets or exceeds 25kV air discharge model.	
<b>Comments</b>	All signal paths have ESD protection, but the power lines are not protected.	



**Test 109 - 109-HW4OA**

<b>109-HW4OA</b>	<b>0603 SMD component size standards</b>	<b>23.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	Discrete components such as resistors, capacitors etc. have a minimum size of 0603 or larger (if required).	
<b>Test</b>	Verify that all SMD discrete components of standard sizing is at standard size 0603 or larger.	
<b>Comments</b>	Succeeded by 1109 – HW4OA in the detailed test requirements specification document.	

**Test 110 – 110-HW4OA**

<b>110-HW4OA</b>	<b>Do not mount (DNM) components</b>	<b>23.01.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	All components that are DNM in initial design shall have visible DNM description in schematics. DNM components shall be listed. DNM components shall not be mounted on the manufactured board.	
<b>Test</b>	Verify that all components with BOM_IGNORE property have visible DNM described in schematics. Verify that all DNM components are listed in BOM_IGNORE list. Verify that the DNM components are not mounted on the manufactured board.	
<b>Comments</b>	Succeeded by 1110 – HW4OA in the detailed test requirements specification document.	

**Test 111 - 111-SY3HD**

<b>111-SY3HD</b>	<b>Max RPM</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Components must be able to support engine speed to 16.000 RPM	
<b>Test</b>	Check datasheets to confirm that all components supports 16.000 RPM.	
<b>Comments</b>	All the driver circuitry are rated for high switching frequencies.	

**Test 112 - 112-SY1EG**

<b>112-SY1EG</b>	<b>Design an ECU</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Have a full schematic design for an engine control system.	
<b>Test</b>	Verify that the design files contain all the component and interfaces to regulate the combustion process in an engine.	
<b>Comments</b>	The schematic design has implemented all necessary components to control a combustion engine.	

**Test 113 - 113-SW3DH**

<b>113-SW3DH</b>	<b>Android application</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Check if the app is running on the tablet without any exception or crash.	
<b>Test</b>	Verify that there is an app on the tablet running without any bugs.	
<b>Comments</b>	The KPEC android application are functional.	

**Test 114 - 114-HW2HD**

<b>114-HW2HD</b>	<b>Cadence Allegro</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Schematic drawn in Cadence Allegro.	
<b>Test</b>	Verify that the design files are .dsn .	
<b>Comments</b>	All schematic design files are opj. Files (Cadence allegro)	

**Test 115 - 115-SY3EG**

<b>115-SY3EG</b>	<b>Cylinder pressure characteristics</b>	<b>03.03.2015</b>
<b>Status</b>	Not tested	
<b>Acceptance Criteria</b>	Shall be able to read the pressure characteristics inside each cylinder.	
<b>Test</b>	Verify that the ECU is able to read the pressure characteristics inside each cylinder, either by reading IO using Vivado or the application on the tablet.	
<b>Comments</b>	Long term goal makes this a level 3 priority. Not possible to test before the circuitry has been realized on a physical circuit board.	

**Test 116 - 116-SW2DH**

<b>116-SW2DH</b>	<b>Install PetaLinux on Zynq-TE0720</b>	<b>06.05.2015</b>
<b>Status</b>	Passed	
<b>Acceptance Criteria</b>	Have the latest version of Petalinux 2014.4 on the Zynq-TE0720	
<b>Test</b>	Verify that the latest stable version of Petalinux is installed from the terminal. Currently 2014.4	
<b>Comments</b>	The terminal was properly installed using Petalinux version 2014.4	



## 2. Test results tables

Table 1 – Requirement test status Part 1

Requirement ID	Test status	Date	Responsible
001-SY1EG	Changed; Succeeded by 112-SY1EG	03.03.2015	HD
002-SY1EG	Passed	06.05.2015	JS
003-SY1EG	Passed	07.05.2015	JS
004-SY1EG	Passed	07.05.2015	JS
005-SY1EG	Not tested	12.01.2015	JS
006-SY1EG	Passed	07.05.2015	JS
007-SY3EG	Not tested	12.01.2015	JS
008-SY3EG	Not tested	12.01.2015	JS
009-SY3EG	Not tested	12.01.2015	JS
010-SY3EG	Passed	07.05.2015	JS
011-SY3EG	Passed	07.05.2015	JS
012-SY3EG	Passed	07.05.2015	JS
013-SY4EG	Not tested	13.01.2015	JS
014-SY4EG	Not tested	13.01.2015	JS
015-SY5EG	Passed	07.05.2015	JS
016-SY3EG	Passed	07.05.2015	JS
017-SY4EG	Passed	07.05.2015	JS
018-SY4EG	Passed	07.05.2015	JS
019-SY5EG	Not tested	14.01.2015	JS
020-SY3EG	Passed	07.05.2015	JS
021-SY3EG	Not tested	14.01.2015	JS
022-SY4EG	Not tested	14.01.2015	JS
023-SW4SA	Passed	06.05.2015	DH
024-SW3SA	Passed	06.05.2015	DH
025-SW5SA	Not tested	12.01.2015	DH
026-SW2SA	Passed	06.05.2015	DH
027-SW2DH	Changed; Succeeded by 113-SWDH	03.03.2015	HD
028-SW5DH	Passed	06.05.2015	DH
029-SW4DH	Changed; Succeeded by 1029-SW4DH	13.01.2015	DH
030-SW4SA	Changed; Succeeded by 1030-SW4SA	13.01.2015	DH
031-SW3DH	Passed	13.01.2015	DH
032-SW3SA	Passed	13.01.2015	DH
033-SW3DH	Not tested	13.01.2015	DH
034-SW5SA	Changed; Succeeded by 1034-SW5SA	13.01.2015	DH
035-SW5SA	Changed; Succeeded by 1035-SW5SA	13.01.2015	DH
036-SW5SA	Passed	11.05.2015	DH
037-SW5DH	Not tested	14.01.2015	DH
038-SW5SA	Passed	11.05.2015	DH
039-SW4DH	Passed	06.05.2015	DH
040-SW4DH	Passed	06.05.2015	DH



Table 2 – Requirement test status Part 2

Requirement ID	Test status	Date	Responsible
041-SW5DH	Not tested	14.01.2015	DH
042-SW3DH	Passed	06.05.2015	DH
043-SW5SA	Changed; Succeeded by 1043-SW5SA	15.01.2015	DH
044 SW5SA	Passed	08.05.2015	DH
045 SW4SA	Passed	08.05.2015	DH
046-SW3SA	Passed	08.05.2015	DH
047-HW2HD	Passed	06.05.2015	JS
048-HW2JS	Not tested	13.01.2015	JS
049-HW2HD	Changed; Succeeded by 114-HW2HD	03.03.2015	HD
050-HW3JS	Passed	06.05.2015	JS
051-HW3HD	Passed	06.05.2015	JS
052-HW3EG	Passed	06.05.2015	JS
053-HW2JS	Not tested	13.01.2015	JS
054-HW3HD	Not tested	13.01.2015	JS
055-HW3EG	Changed; Succeeded by 115-HW3EG	03.03.2015	HD
056-HW4HD	Passed	06.05.2015	JS
057-HW3JS	Not tested	13.01.2015	JS
058-HW3JS	Passed	06.05.2015	JS
059-HW3HD	Passed	06.05.2015	JS
060-HW3EG	Passed	06.05.2015	JS
061-HW3HD	Not tested	13.01.2015	JS
062-HW4EG	Passed	06.05.2015	JS
063-HW3HD	Not tested	13.01.2015	JS
064-HW3EG	Passed	06.05.2015	JS
065-HW3JS	Passed	06.05.2015	JS
066-HW2EG	Passed	06.05.2015	JS
067-HW3OA	Passed	06.05.2015	JS
068-HW3HD	Not tested	14.01.2015	JS
069-HW3JS	Passed	14.01.2015	JS
070-HW3HD	Passed	14.01.2015	JS
071-HW3OA	Passed	06.05.2015	JS
072-HW3JS	Passed	06.05.2015	JS
073-HW3EG	Passed	06.05.2015	JS
074-HW3JS	Passed	06.05.2015	JS
075-HW3EG	Not tested	14.01.2015	JS
076-HW3HD	Passed	06.05.2015	JS
077-HW3JS	Passed	06.05.2015	JS
078-HW5HD	Passed	06.05.2015	JS
079-HW3HD	Passed	06.05.2015	JS
080-HW4JS	Passed	06.05.2015	JS



Table 3 – Requirements test status Part 2

Requirement ID	Test status	Date	Responsible
081-HW4OA	Changed; Succeeded by 1081-HW4OA	14.01.2015	JS
082-HW3HD	Passed	06.05.2015	JS
083-HW3JS	Passed	06.05.2015	JS
084-HW4EG	Passed	06.05.2015	JS
085-HW5JS	Passed	06.05.2015	JS
086-HW5OA	Passed	06.05.2015	JS
087-HW3JS	Passed	06.05.2015	JS
088-HW3EG	Passed	06.05.2015	JS
089-HW4HD	Passed	06.05.2015	JS
090-HW3OA	Changed; Succeeded by 1090 – HW3OA	13.01.2015	JS
091-HW4OA	Failed	06.05.2015	JS
092-HW4OA	Changed; Succeeded by 1092-HW4OA	13.01.2015	JS
093-HW5OA	Changed; Succeeded by 1093-HW5OA	13.01.2015	JS
094-HW4OA	Changed; Succeeded by 1094-HW4OA	13.01.2015	JS
095-HW3OA	Changed; Succeeded by 1095-HW3OA	13.01.2015	JS
096-HW3OA	Changed; Succeeded by 1096-HW3OA	13.01.2015	JS
097-HW4OA	Changed; Succeeded by 1097-HW4OA	13.01.2015	JS
098-HW4OA	Changed; Succeeded by 1098-HW4OA	13.01.2015	JS
099-HW5OA	Changed; Succeeded by 1099-HW5OA	13.01.2015	JS
100-HW5OA	Changed; Succeeded by 1100-HW5OA	13.01.2015	JS
101-HW3OA	Changed; Succeeded by 1101-HW3OA	13.01.2015	JS
102-HW3OA	Changed; Succeeded by 1102-HW3OA	13.01.2015	JS
103-HW3OA	Changed; Succeeded by 1103-HW3OA	13.01.2015	JS
104-HW4OA	Changed; Succeeded by 1104-HW4OA	14.01.2015	JS
105-HW5OA	Changed; Succeeded by 1105-HW5OA	14.01.2015	JS
106-HW2OA	Changed; Succeeded by 1106-HW2OA	14.01.2015	JS
107-HW2OA	Changed; Succeeded by 1107-HW2OA	15.01.2015	JS
108-HW4OA	Failed	16.01.2015	JS
109-HW4OA	Changed; Succeeded by 1109-HW4OA	16.01.2015	JS
110-HW4OA	Changed; Succeeded by 1110-HW4OA	16.01.2015	JS
111-SY3HD	Passed	06.05.2015	HD
112-SY1EG	Passed	06.05.2015	HD
113-SW3DH	Passed	06.05.2015	HD
114-HW2HD	Passed	06.05.2015	HD
115-SY3EG	Not tested	03.03.2015	HD
116-SW2DH	Passed	06.05.2015	DH



KONGSBERG



## Test plan

### KPEC

<b>Employer</b>	Kongsberg Defence & Aerospace			
<b>Group Members</b>	<b>Name</b>		<b>Initials</b>	
	Ola Pedersen Aasheim		OA	
	Salahuddin Asjad		SA	
	Hung Dinh		HD	
	Dler Hasan		DH	
	Even Gudbrandsen		EG	
	Jannik Schäffer		JS	
<b>Document information</b>	<b>Revision</b>	<b>Date</b>	<b>Approved</b>	<b>Pages</b>
	3.0	18.05.2015	JS	15







## Revision Table

Version	Date	Approval	Description
1.0	11.03.2015	JS	<ul style="list-style-type: none"> <li>Created the document</li> </ul>
1.1	13.03.2015	HD	<ul style="list-style-type: none"> <li>Added chapter 4 and 5.</li> </ul>
2.0	16.03.2015	JS	<ul style="list-style-type: none"> <li>Revised and published</li> </ul>
3.0	18.05.2015	JS	<ul style="list-style-type: none"> <li>Revised and published</li> </ul>

## Contents

Revision Table .....	2
List of figures.....	3
1. Introduction .....	4
1.1. Resources.....	4
2. The scope of testing.....	4
2.1. Test methods .....	5
2.1.1. Static vs. Dynamic.....	5
2.1.2. Alpha vs. Beta .....	6
2.1.3. Test features.....	6
2.1.4. White box vs black box testing.....	7
3. Hardware testing .....	8
3.1. Simulation.....	8
3.2. Review .....	10
3.2.1. Internal review .....	10
3.2.2. KDA review .....	11
4. Software testing.....	11
4.1. Static testing .....	11
4.2. Dynamic testing .....	12
5. Manufacture testing.....	12
5.1. Flying probe testing .....	12
5.2. Bed of nails testing .....	13



6. Excluded features .....	13
7. Test acceptance .....	13
8. Goal .....	14
Bibliography .....	15

## List of figures

Figure 1 - Static vs Dynamic rate of success.....	5
Figure 2 - Bode plot of the Lambda filter circuit.....	6
Figure 3 - White box VS black box.....	7
Figure 4 - Signal flow Lambda circuit.....	8
Figure 5 - Undesirable step response .....	9
Figure 6 - Lambda driver step response .....	9
Figure 7 - Proper step response on lambda filter .....	10



## 1. Introduction

The test plan document is intended to provide information for the test personnel to be able to complete the test-phase of the KPEC project. The document shall allow its reader to get an understanding of how KPEC will execute their tests. The reason for testing is to reveal faults that might compromise the experience of the system

### 1.1. Resources

The resources needed to complete the tests are essentially related to KPEC's own personell where we have established a dedicated test and verification responsible. His job will be to administer resources along with the systems engineer, to ensure that we effectively can verify & validate each requirement specified in the test specification document. External resources may be applicable for static testing of the design documents, as our knowledge as students are restricted.

## 2. The scope of testing

Testing is initiated to determine whether we're building what we set out to build, and if we're building it right. KPEC are continuously testing to verify design elements on a daily basis, to increase the performance of the system.

Testing is initiated to provide objective knowledge about how the system performs and how different parts of the system perform. The system that KPEC has set out to design will have a horizon of several years; it is therefore especially meaningful to conduct testing to evaluate the quality of the existing design, before making decisions whether or not to proceed with the project. Test results are intended to provide the stakeholders with objective knowledge about the system performance, for which they can access the quality for the specific system test.

The purpose of testing is to reveal possible error within the system, for which the designer can initiate measures to correct the error. It's in the stakeholders best interest that the system KPEC delivers, has executed rigorous testing to verify the system qualities and weaknesses. Subsystems that are questionable should not be a part of the deliverables, as this is in conflict with the general result that stakeholders will expect.



## 2.1. Test methods

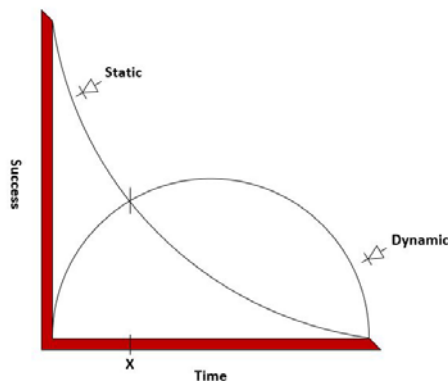
### 2.1.1. Static vs. Dynamic

There are many features in this project that applicable for testing whether it is hardware or software, they are performed in various ways to accomplish the same purpose.

A system is usually tested implicitly, electronic hardware designers will consult their requirement documentation to evaluate solutions; the process is time consuming and full of tradeoffs. Hardware designers can similarly benefit from simulations to validate parts of their circuit design. This is called static testing, where the circuit performance is evaluated by humans, and decisions are made on subjective experiences. There are other forms of static testing as well; walkthrough, where the designers can lead the members of the development-team through the process, in which the members can provide feedback. For software designers it's essential to compile their code, run it and evaluate the results. The iteration and thought process that static testing can provide is useful in earlier phases of the project.

As the project evolves, fewer and fewer problems are revealed by static testing, that's where dynamic methods are useful. Dynamic testing is executed when systems are getting operational, in order to test that the interaction between design elements will perform as expected. The hardware part of KPEC can't test dynamic features because we will not produce any physical product, our deliverable is a schematic design. All our testing will therefore be done by inspection, reviews and walkthroughs. Meanwhile our software part will be doing static and dynamic testing to verify and validate that their code works properly.

Static and dynamic debugging has different rates of success evaluated in the time span of the project. Figure 1 is an illustration of this case.



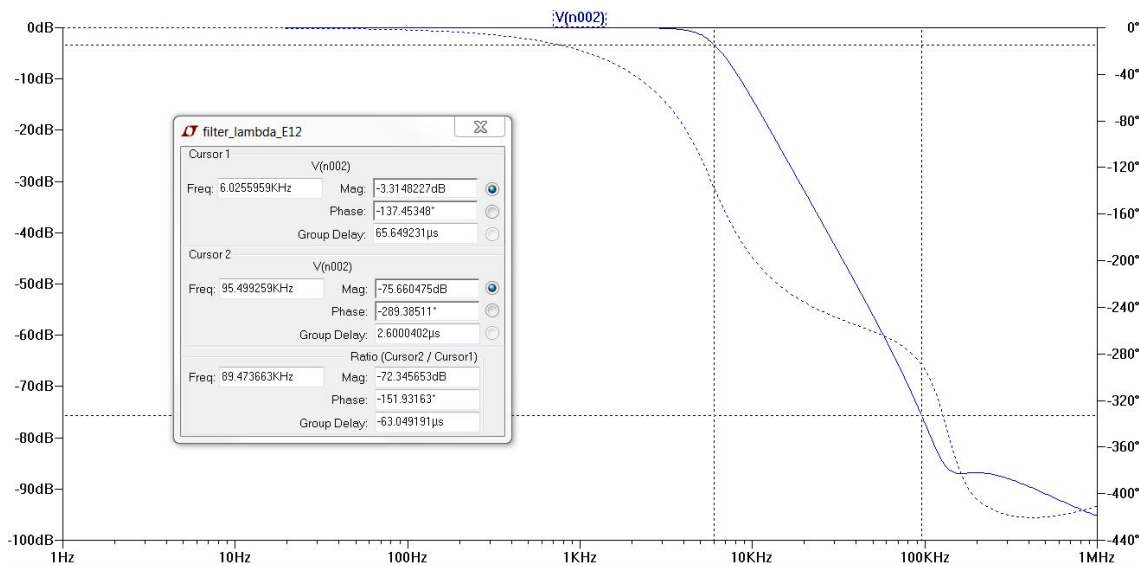
**Figure 1 - Static vs Dynamic rate of success**

While static testing is useful in the beginning, there is a point in time X where dynamic testing is the best suited tool to detect errors in the design.



### 2.1.2. Alpha vs. Beta

Alpha testing is a form of internal testing at KPEC, the mission is to verify and validate the system before the system is transitioned into beta testing. Alpha testing is usually conducted on smaller parts of the system before they are merged together. A typical alpha test could be to design an anti-aliasing filter, and then build the circuit in any circuit design tool where different simulations can be executed. Typical simulation for the filter would be a bode plot in the frequency domain as seen in Figure 2.



**Figure 2 - Bode plot of the Lambda filter circuit**

Beta testing is more rigorous as it includes external stakeholders, and can be considered a form of acceptance test. Beta testing will be executed on the total system deliverables, the system contains all the physical attributes, but there is uncertain whether or not the system is functional in every aspect. To our project this could translate into a finished ECU circuit board, all the circuitry to operate the engine are present, the beta testing will then reveal whether or not the ECU can perform as intended with all or parts of its subsystems operational. It is essential to notify that KPEC will produce a schematic design and not be delivering any PCB design, this suggests that we will only conduct Alpha testing.

### 2.1.3. Test features

As mentioned previously we have created a test specification document that are divided into three sub categories. These are;

- System requirements
- Hardware requirement
- Software requirements



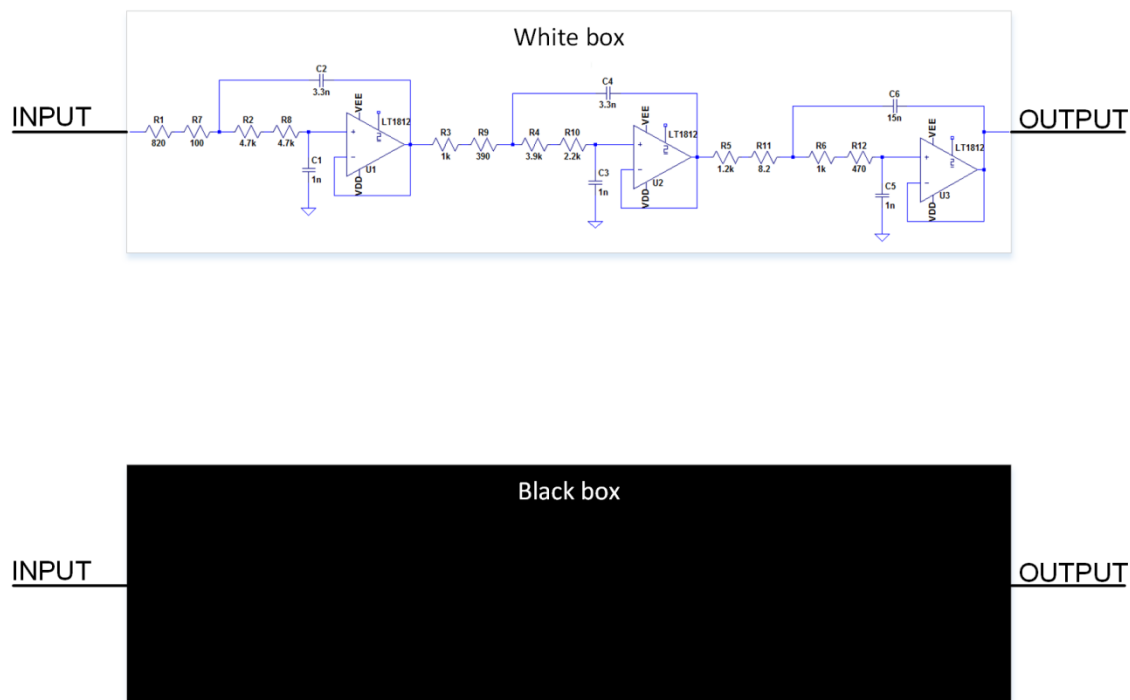
Where hardware and software requirements reflects the functionalities of the system requirements. We will be doing a lot of static testing, which translates to visual inspection of produced documentation. There will be done some dynamic testing from software when their coding is being tested on the Micromodule for communication with the tablet to verify that the code and application works. The deliverables will include schematic design features, software related source codes and a functional tablet application, which implies that a lot of testing will be necessary.

#### 2.1.4. White box vs black box testing

There are two different test scenarios; white box and black box testing. In the white box<sup>1</sup> testing method the person that tests the system knows how the system works, and therefore designs tests that test the internal structure. In software it is used to determine if the internal structure of the code is working properly. While in hardware this is similar to testing circuit nodes to determine if the circuit behaves properly.

Black box<sup>2</sup> testing consists of testing the output response of the system for a given input, without knowledge about the internal structure. The person who execute the test is only required to know what the system should do and do not need knowledge of how the system does it.

Figure 3 illustrates the white and black box testing principles.



**Figure 3 - White box VS black box**

<sup>1</sup> White-box testing

<sup>2</sup> Black-box testing

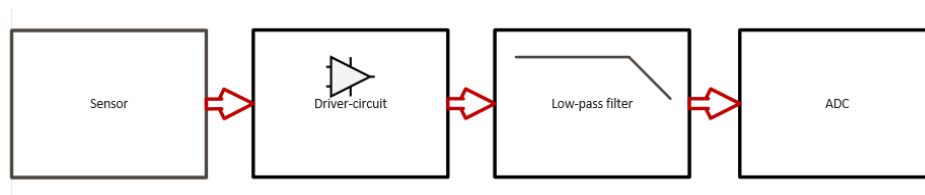


### 3. Hardware testing

Electronic development can be demanding without the aid of computer aided tools, software is available on the market that enables engineers to develop complex systems. One of such tools are the linear technology spice software (LTspice), the software is excellent for circuit testing in many ways. It's usual that engineers break complex systems into smaller parts, when a parts of a circuitry has been developed, the behavioral of the circuit can be simulated with the LTspice software. An additional way to test our design is to have a software/hardware review; where we present the technical solutions we have made to KDA and get feedback. Simulations and review meetings will be discussed further in chapter 3.1 and chapter 3.2.

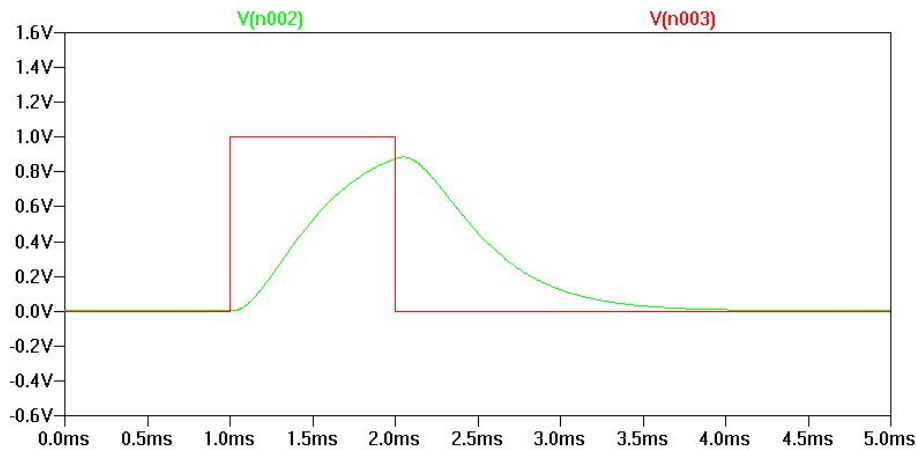
#### 3.1. Simulation

Since we can't fully test our ECU before it has been produced, we have to simulate the subsystems of the ECU, these subsystems are considered as: sensors side and actuator side. Simulations will be executed to see if the circuitry response is in accordance with what we expect.



**Figure 4 - Signal flow Lambda circuit**

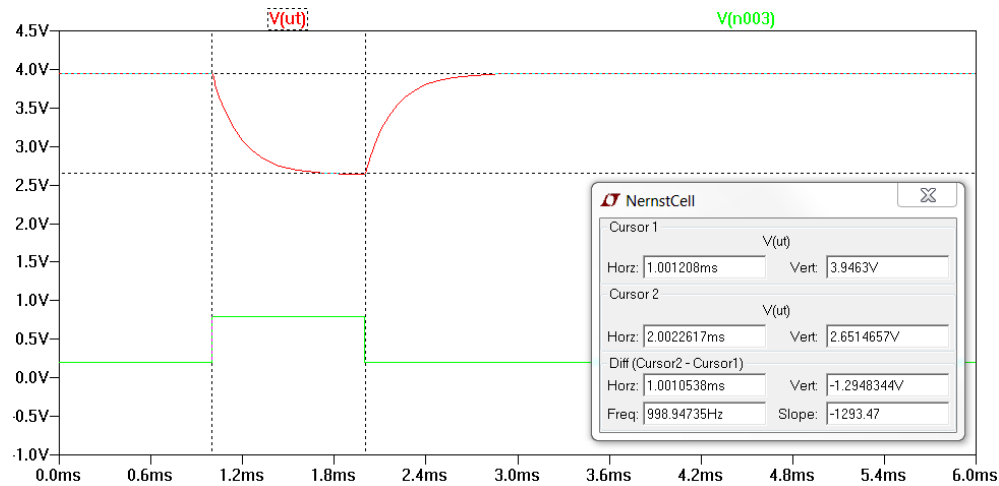
An example of testing that KPEC executed is the design of active filters for the Lambda sensor interface seen in Figure 4. During development of the filter, some requirements were given and some were implicit. One of the weaknesses with active filter design is that when the cut off frequency becomes small enough, the step response drops significantly and the process of designing the filter becomes more difficult. The problem with this is that the ADC is sampling the sensor-signal delayed by the filter; such data acquisition is useless in a closed loop control circuit Figure 5.



**Figure 5 - Undesirable step response**

The test simulation reveals an undesirable step response of the Lambda filter, if this filter had been implemented the signal would have been delayed by more than 1ms for 1V step impulse, and the output would never reach the set value in the given time. The filter is also highly nonlinear, as it is “coloring” the impulse.

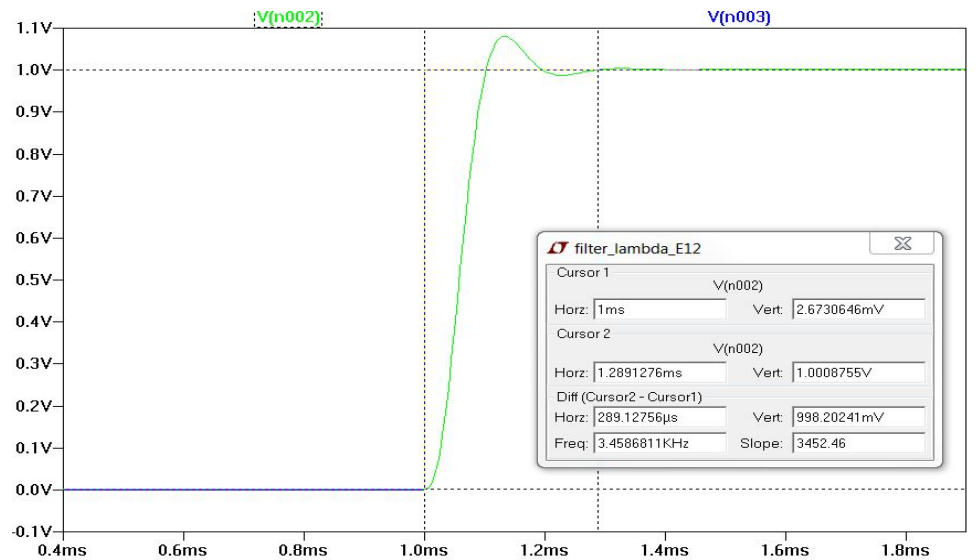
One solution is to run a dynamic test on the Lambda driver, by simulating it in LTspice. When running a step on the Lambda driver we can find the time-delay introduced by the circuit, and thus verifying that our active filter will be agile enough to process the Lambda signal.



**Figure 6 - Lambda driver step response**

The simulation is valuable in many ways; first of all we are validating the step response of the Lambda driver circuit Figure 6. It's behavioral is steady, precise and quite slow. The time it takes from the step is initiated until it reaches its new value is 1ms. Now that we know the new requirement of the filter, we can proceed with the design. The filter will now need to delay data less than the Lambda driver circuit, to ensure that no distortion is inflicted by the filter.





**Figure 7 - Proper step response on lambda filter**

Figure 7 illustrates a proper filter design, its Butterworth characteristics recognized by a small overshoot followed by ringing is the signature of a Butterworth filter. What's important to notice here, is that the filter step response in Figure 7 is much quicker than the driver circuit response in Figure 6. Actually it's more than three times faster, with its Butterworth signature, so by static testing through simulation, we have validated that indeed this filter design will be sufficient.

## 3.2. Review

A review is a multidiscipline meeting to assure that a design has been implemented properly. In such a meeting, representatives from hardware and software are present to provide feedback on implementation to assure that the design meet the requirements. The design is sent in advance so that the participants can look and study the design and come with comments on how to improve or if it's implemented correctly. KPEC has two types of reviews, internal and KDA review. These topics will be discussed further in chapter 3.2.1 and 3.2.2.

### 3.2.1. Internal review

In the internal review members of hardware from KPEC are the representatives. The schematics are distributed in advance so that the members can study and come with improvements or new solutions to the overall design. After an internal review the design is re-designed and implemented with the improvements or solutions that came up at the meeting.



### 3.2.2. KDA review

At this meeting, representatives from KDA and KPEC are the participants. KDAs individual disciplines hardware, software and mechanical department can contribute. To conduct an efficient review, members of the meeting have been assigned parts of the design to study in more detail. In this way the design will undergo real examination and errors will be found and corrected. It's KPECs responsibility to take initiative for these reviews.

## 4. Software testing

Software testing consists of two parts, static testing and dynamic testing and does not consist of circuit simulations or inspections like in hardware, a more hands down approach is used. The assignment for software for this bachelor thesis consists of 3 parts, top design, Zynq design and application development. These tests are used to verify and validate the requirements set by our employer.

### 4.1. Static testing

In the static testing, the code itself is compiled to see if there are any issues with the code.

Some examples of static testing:

- When the hardware design for the Zynq module is created it has to be validated in Xilinx Vivado. The program will then check if all necessary connectors are included.
- When setting up PetaLinux we are following a guide to configure the PetaLinux operating system. The last command is: PetaLinux build which build the operating system and pack every other PetaLinux setup files into two files. The PetaLinux build command will also provide a build log, so we can see that everything went successfully or if something was left out due to different issues.
- When we design the application in C++ we are also checking for syntax- or other kind of errors. After running the compile command, the terminal will tell whether the compilation went successfully or if something went wrong.



## 4.2. Dynamic testing

In the dynamic testing the code itself is tested in the intended environment, for the software part of the system, it is to implement the code in the Zynq module and the creation of an application.

Some examples on dynamic testing:

- After setting up the PetaLinux operating system we have to test it on our Micromodule. When we use JTAG communication, the Zynq will provide with a log to verify if everything went as expected. This is to validate that the processing system, programmable logic and other components are running as desired.
- After the applications in C++ are designed we can run the applications on our Micromodule. This is important because the Micromodule are not running on the same architecture as our computers. So we will then need to evaluate the binary file to verify that the test went as expected.
- To test if we have real-time communication we will use Wireshark network analyzer to calculate the time it takes to send packages from server to client.
- When creating the Android application, we will test the application on different emulators with different screen sizes and specifications to verify that the application is supported on other hardware as well.

## 5. Manufacture testing

When the PCB are manufactured, different testing methods are used in order to find defective boards. PCBs are crafted with small tolerances, and it is expected that shorted lines and open circuits can occur. Finding defective boards during or right after manufacture saves substantial amounts of time compared to manual testing of the boards when they arrive from factory.

There are different ways boards are tested, depending on quality requirements and manufacture volume. This section describes the two main testing methods used for PCBs.

### 5.1. Flying probe testing

Flying probe testing is commonly used for both large and small volume production runs. The way flying probe testing functions, is that conductive needles are placed at all pads on the populated board and controlled whether they conduct as supposed to, are open or shorted to other nets. The netlist is generated by the gerber files (production files) given to the manufacturer from the customer.

The flying probe tester machines are typically large and completely automated, making it very fast, accurate and save manpower. Flying probe testing is used both on



populated and unpopulated boards. The amount of components can be considerable, testing the boards both before and after components are mounted can save substantial amounts in terms of manufacturing costs.

## 5.2. Bed of nails testing

Beds of nails testing are most commonly used in volume manufacturing or high value board manufacture for populated boards. The bed of nails is an active device that must be custom built and programmed for each design. Bed of nails testing allows making contact to each test points on the device under test simultaneously. The boards can be tested for complete functionality and powered by the bed of nails, taking measurements of the boards and logging results. Bed of nails testing is very powerful as it can reveal design errors on the finished boards in as well as manufacturing errors. In-circuit programming can also be implemented in addition to testing for electrical characteristics, effectively increasing the manufacturing process.

## 6. Excluded features

As mentioned earlier the duration of our project restricts the development process, we will not be able to run tests on a final product. Our mission will then be to test functionality that is available to us through static testing and some dynamic testing regarding software. As a consequence of this we will provide the next generation with our test procedures and documentation regarding the design, and to provide them with as much information as possible, but in the end, the next generation will need to take necessary measures to test the project the way they may seem fit using our information or not.

To summarize, there is no physical product to test on. Hardware will only execute static testing like inspection, simulations and reviews. While software will be doing static testing using inspection and compiling as test procedures. Additionally they will also be performing dynamic testing on the physical Zynq module by implementing the code followed by debugging.

## 7. Test acceptance

During the test-phase of the project, a document will be created.

The official document shall be written in accordance with KPEC standards for reports; the document will include the test requirement ID, description of the test, result of the test and a comment section. The document shall provide the information about the tests and the result of these, multiple tests might be necessary to cover one requirement.

The test phase has been scheduled for week 19-20 and the duration of the test phase is 10 days. During these days we will go through the requirements and test specifications to verify and validate that we are meeting our requirements set by our employer. Additionally we will



be going through the design requirement and test requirement to verify that we have been able to meet our own requirements.

The result of the test phase is a formal document over the test that illustrates how we performed the different test and to determine which test has passed/failed. This allows us to see if we have met our employer requirements and our own.

## 8. Goal

KPEC has formed a risk analysis which reveals that testing might give various results, that combined with challenges related to the duration of the project may present itself as an issue we must deal with. At the beginning we had ambitions and thought we would be able to design a schematic and PCB layout, but we quickly realized that this was not possible. KPEC and KDA agreed upon that finishing a schematic design was a reasonable goal for the duration of this project. This is why the goal will be to ensure that the deliverables are working, and what we have made are in accordance with the system requirements and this is accomplished with testing. By testing the product that we have, we can be able to verify and validate that what we deliver is working and that it meet requirements embedded with the task description and our own requirements.



## Bibliography

Luke Engelbert-Fenton, C. M. (2012, 10 20). *Test Plan*. Hentet 01 23, 2015 fra Design & Implementation of an Electronic ID: [http://projects-web.engr.colostate.edu/ece-sr-design/AY13/inventory/ECE401\\_EIDI\\_Test\\_Plan.pdf](http://projects-web.engr.colostate.edu/ece-sr-design/AY13/inventory/ECE401_EIDI_Test_Plan.pdf)

*WikiHow*. (2014, 06). Hentet 01 23, 2015 fra <http://www.wikihow.com/Write-a-Test-Plan>

Wikipedia. (2015, 03 01). *Black-box testing*. Hentet fra [http://en.wikipedia.org/wiki/Black-box\\_testing](http://en.wikipedia.org/wiki/Black-box_testing)

Wikipedia. (2015, 01 31). *White-box testing*. Hentet fra [http://en.wikipedia.org/wiki/White-box\\_testing](http://en.wikipedia.org/wiki/White-box_testing)



KONGSBERG



## Design document

### KPEC

<b>Employer</b>	Kongsberg Defence & Aerospace			
<b>Group Members</b>	<b>Name</b>		<b>Initials</b>	
	Ola Pedersen Aasheim		OA	
	Salahuddin Asjad		SA	
	Hung Dinh		HD	
	Dler Hasan		DH	
	Even Gudbrandsen		EG	
	Jannik Schäffer		JS	
<b>Document information</b>	<b>Revision</b>	<b>Date</b>	<b>Approved</b>	<b>Pages</b>
	3.0	18.05.2015	SA	352





## Abstract

This is the design document, which is a thorough description of the results carried out during the design phase of the project. The overall purpose of this document is to authenticate the results for future project use for Kongsberg Defence & Aerospace. After reading this document, one should be able to have a clear understanding of the detailed design of hardware and software for this project.





## Revision Table

Version	Date	Approval	Description
1.0	09.02.2015	EG	<ul style="list-style-type: none"> <li>Created the document</li> </ul>
2.0	16.03.2015	HD	<ul style="list-style-type: none"> <li>Revised and published</li> </ul>
2.1	18.03.2015	HD	<ul style="list-style-type: none"> <li>Added chapter 2.6.6</li> </ul>
2.2	24.03.2015	JS	<ul style="list-style-type: none"> <li>Revision of introduction in chapter 2.9.4</li> <li>Added simulation results from step response in chapter 2.9.4.3</li> <li>Added chapters 2.9.4.4 – 2.9.4.9</li> </ul>
2.3	27.03.2015	HD	<ul style="list-style-type: none"> <li>Added chapters 2.9.6 – 2.9.8</li> </ul>
2.4	15.03.2015	OA	<ul style="list-style-type: none"> <li>Changed chapters 4.2 and 4.3</li> <li>Added chapters 2.13, 2.14 and 3.9.2</li> </ul>
2.5	22.04.2015	SA	<ul style="list-style-type: none"> <li>Changed chapter 7.5</li> </ul>
2.6	28.04.2015	HD	<ul style="list-style-type: none"> <li>Revised 3.1.2, 3.2.2 and 3.7</li> </ul>
2.7	28.04.2015	OA	<ul style="list-style-type: none"> <li>Revised 2.2, 2.3, 2.4, 2.10, 4.3, 4.4 and 3.1.2</li> </ul>
2.8	06.05.2015	HD	<ul style="list-style-type: none"> <li>Added chapter 6, 7, 5.7, 3.10.3, 2.10.5, 3.8.1 and 12.</li> <li>Added Appendix and chapter 12.</li> <li>Revised 2.8.</li> <li>Merged 2.9.7, 2.11, 2.12 into 2.11.</li> </ul>
2.9	08.05.2015	OA	<ul style="list-style-type: none"> <li>Revised introduction</li> <li>Added chapter 1.1-1.3</li> <li>Revised chapter 1.4</li> </ul>
2.10	10.05.2015	HD	<ul style="list-style-type: none"> <li>Revised chapter 9</li> <li>Revised chapter 10.4 and 10.5</li> </ul>
2.11	12.05.2015	DH	<ul style="list-style-type: none"> <li>Revised chapter 1.5</li> <li>Added chapter 11</li> </ul>
2.12	13.05.2015	HD	<ul style="list-style-type: none"> <li>Revised 3.8</li> <li>Added chapter 3.12</li> <li>Updated pictures</li> </ul>
3.0	18.05.2015	SA	<ul style="list-style-type: none"> <li>Revised and published</li> </ul>



## Contents

Abstract .....	2
Revision Table .....	3
List of figures.....	11
List of tables .....	16
List of codes .....	18
Introduction .....	20
1. System overview .....	22
1.1. KPEC ECU specification summary.....	22
1.2. Role of the Engine Control Unit (ECU) .....	23
1.3. Combustion process overview .....	25
1.4. Hardware design overview .....	27
1.5. Software design overview.....	32
2. TE0720 Micromodule interfacing.....	35
2.1. TE0720 B2B interfacing .....	36
2.1.1. Connector top.....	37
2.1.2. Connector bottom .....	39
2.1.3. Connector left .....	41
3. Sensor interfaces .....	42
3.1. Temperature measurements.....	42
3.1.1. Thermistor measurements .....	43
3.1.2. Thermocouple measurements .....	44
3.2. Temperature sensor interfacing.....	44
3.2.1. LTC2983 pinout description .....	45
3.2.2. LTC2983 thermistor interface .....	47
3.2.3. LTC2983 thermocouple interface .....	48
3.2.4. LTC2983 diode interface .....	49
3.3. Barometric air pressure measurements.....	50
3.3.1. MS5611-01BA03-50 On-board barometer .....	50
3.4. Position and orientation measurements .....	51
3.4.1. MPU-9150 Motion Processing Unit (MPU) .....	51



3.4.2.	MPU-9151 schematic implementation .....	53
3.5.	Flywheel sensor interface .....	54
3.5.1.	Zero crossing.....	54
3.5.2.	Adaptive peak threshold.....	54
3.5.3.	Bias voltage source.....	55
3.5.4.	Mode selection.....	55
3.5.5.	ESD protection .....	57
3.5.6.	Output voltage.....	58
3.6.	Camshaft sensor interface .....	59
3.6.1.	ESD protection .....	60
3.6.2.	Low-pass filtering .....	60
3.6.3.	Input buffer .....	62
3.7.	Analog data acquisition .....	63
3.7.1.	Nyquist sampling theorem.....	63
3.7.2.	Oversampling.....	64
3.7.3.	Time budget.....	65
3.7.4.	Cylinder Pressure interface .....	66
3.7.5.	Fluid pressure interface .....	85
3.7.6.	Manifold air pressure (MAP) interface .....	89
3.7.7.	General purpose analog in filter.....	90
3.7.8.	Zynq-7000 XADC .....	91
3.8.	Lambda sensor interface .....	94
3.8.1.	Wideband linear lambda sensor .....	96
3.8.2.	Bosch lambda sensor LSU 4.9.....	97
3.9.	Knock sensor interface.....	112
3.9.1.	TPIC8101 .....	112
3.10.	Digital interface.....	115
3.10.1.	Flex fuel sensor .....	115
3.10.2.	Ignition signal circuitry.....	119
3.10.3.	Brake pedal sensor circuitry .....	121
3.10.4.	Manifold Air Flow (MAF) sensor.....	122



3.10.5.	General purpose digital inputs .....	123
3.11.	Battery monitor.....	124
3.11.1.	Implementation of the battery monitor.....	124
3.11.2.	Simulation of the battery monitor .....	126
3.12.	Accelerator pedal.....	130
4.	Drivers .....	131
4.1.	Ignition coil drive.....	131
4.1.1.	Power stage.....	132
4.1.2.	Driver stage .....	133
4.2.	Fuel Injection driver .....	134
4.2.1.	Power stage.....	135
4.2.2.	Driver stage .....	136
4.3.	Boost controller driver .....	137
4.3.1.	Boost controller selection.....	137
4.4.	Fuel pump driver.....	140
4.4.1.	Fuel pump safety.....	140
4.5.	Throttle body motor controller .....	142
4.5.1.	MC33931 H-bridge controller .....	142
4.5.2.	Throttle body position feedback.....	146
4.6.	Multipurpose MOSFET .....	147
4.7.	General purpose low-side switches .....	148
4.8.	Multipurpose stage driver.....	149
4.9.	PWM - Pulse Width Modulation.....	151
5.	Communication interfaces.....	153
5.1.	USB 2.0 interface .....	153
5.1.1.	USB to dual UART interface.....	153
5.1.2.	Design specifics for FT2232HL.....	154
5.2.	JTAG .....	157
5.2.1.	JTAG implementation on the ECU.....	158
5.2.2.	JTAG enable jumper .....	158
5.3.	Ethernet .....	160



5.4.	Wi-Fi.....	162
5.4.1.	MRF24WG0MB pinout description .....	163
5.5.	CAN.....	164
5.5.1.	Implementation of CAN in our project.....	164
5.6.	Valve module.....	166
5.6.1.	Pneumatic valve concept evaluation.....	167
5.6.2.	Valve module interface requirements .....	169
5.6.3.	CAN bus implementation .....	172
5.7.	General purpose digital I/O .....	174
6.	Power design .....	175
6.1.	Power design strategies .....	175
6.1.1.	Voltage divider .....	176
6.1.2.	Linear regulator.....	177
6.1.3.	Switched-mode Power Supply (SMPS) .....	178
6.2.	Automotive power design considerations.....	179
6.2.1.	Cold crank cycle .....	179
6.2.2.	Cold cranking power budget .....	183
6.2.3.	Load dumps .....	184
6.3.	ECU power input protection design.....	185
6.3.1.	LT4363 surge stopper .....	185
6.4.	Main ECU power supplies design .....	198
6.4.1.	Power budget and configuration .....	198
6.4.2.	3.3 V supply .....	200
6.4.3.	5V Supply.....	208
6.4.4.	-5 V supply .....	214
6.4.5.	15V supply .....	222
6.4.6.	-15 V Supply.....	228
6.4.7.	Additional notes.....	236
7.	Miscellaneous features .....	237
7.1.	RTC (Real Time Clock) battery .....	237
7.2.	Programmable LEDs .....	237



7.3.	Test points .....	239
7.4.	0 $\Omega$ resistors .....	239
7.5.	Reset button and jumper configurations.....	240
7.6.	PGOOD LED's .....	241
7.7.	General purpose inputs and outputs .....	242
8.	Derating of electrical components.....	243
8.1.	Derating factors .....	244
8.2.	Component derating tables .....	245
8.2.1.	Capacitors.....	245
8.2.2.	Resistors .....	247
8.2.3.	Transistors.....	247
8.2.4.	Diodes .....	248
8.2.5.	Magnetic components.....	248
8.2.6.	Integrated circuits.....	249
8.2.7.	Optocouplers .....	249
8.3.	Power dissipation derating of components.....	250
9.	Software design.....	251
9.1.	Zynq module (TE0720) .....	251
9.2.	Carrier board (TE0703) .....	251
9.3.	Tablet device .....	251
10.	Zynq design .....	252
10.1.	Configuring custom hardware .....	252
10.1.1.	Implement Hardware .....	253
10.1.2.	Export hardware.....	254
10.2.	Software Development Kit (SDK) .....	256
10.2.1.	Hardware platform specification .....	256
10.2.2.	Build Bootloader .....	257
10.3.	PetaLinux operating system.....	258
10.3.1.	Why Processing System and PetaLinux?.....	259
10.3.2.	Device-tree configuration.....	260
10.3.3.	Building PetaLinux .....	261



10.3.4.	Mount the NAND flash (eMMC) on bootup .....	262
10.4.	Real-time communication .....	263
10.4.1.	Server-side communication .....	264
10.5.	Parameter storage .....	266
10.5.1.	Storage in TSV format.....	266
10.6.	Makefile .....	270
11.	Android application design.....	271
11.1.	PhoneGap.....	272
11.2.	Application development workflow .....	273
11.2.1.	Setup the development environment.....	273
11.2.2.	Creating the application.....	275
11.2.3.	Run, debug and test the application .....	278
11.2.4.	Create a release version of the application.....	283
11.3.	Application code structure .....	284
11.3.1.	Navigation.....	285
11.3.2.	Dashboard.....	286
11.3.3.	Diagrams .....	289
11.3.4.	WebSocket connection inside application.....	292
11.3.5.	Performance analysis .....	294
11.3.6.	Application configuration files.....	297
11.4.	Sensors- and drivers overview .....	300
11.5.	Configuration manuals and recommendations .....	301
11.5.1.	PetaLinux startup guide .....	301
11.5.2.	Android development configuration guide .....	304
11.5.3.	GitHub startup guide .....	305
12.	ECU Connectivity .....	306
12.1.	ECU header .....	306
12.2.	ECU connectors .....	308
12.2.1.	Common connector - BAY X (48 pins).....	309
12.2.2.	Bank 1 connector - Bay Y (53 pins) .....	309
12.2.3.	Bank 2 connector - Bay Z (53 pins) .....	309



12.3.	Connector and header numbering system.....	310
12.4.	Schematic implementation.....	311
12.5.	Practical implementation.....	312
Appendix I - List of abbreviations.....		314
Appendix II - Power budget.....		319
Appendix III - Recommended automotive sensors, actuators and parts .....		321
Appendix IV - Signal names and description .....		323
Appendix V - Board-to-Board connector tables.....		328
Appendix VI - ECU connector tables.....		337
Connection and description table for common - bay X.....		337
Connection and description table for bank 1 - bay Y.....		339
Connection and description table for bank 2 - bay Z.....		341
Bibliography .....		343





## List of figures

Figure 1 - Overview model .....	27
Figure 2 - Hardware implementation.....	28
Figure 3 - Hardware design status .....	31
Figure 4 - Software implementation .....	33
Figure 5 - Software design status.....	34
Figure 6 - TE0720 Micromodule .....	35
Figure 7 - TE0720 bottom with B2B connectors .....	37
Figure 8 - Connector top schematic implementation .....	38
Figure 9 - Connector bottom schematic implementation .....	40
Figure 10 - Connector left schematic implementation .....	41
Figure 11 - NTC thermistor resistance characteristic .....	43
Figure 12 - LTC2983 schematic implementation .....	45
Figure 13 - Cold junction diode .....	48
Figure 14 - Diode configuration .....	49
Figure 15 - Barometer implementation .....	50
Figure 16 - Inertial module measurements.....	52
Figure 17 - MPU-9150 schematic implementation .....	53
Figure 18 - MAX9924 Mode A2/C .....	55
Figure 19 - Mode selection GND/VCC.....	56
Figure 20 - EXT voltage divider .....	56
Figure 21 - Bias voltage .....	57
Figure 22 - ESD protection on input .....	58
Figure 23 - Voltage level-shifting .....	58
Figure 24 - Simplified block diagram of the system .....	59
Figure 25 - ESD Protection .....	60
Figure 26 - Low-pass filter implementation .....	60
Figure 27 - Filter simulation and connection.....	61
Figure 28 - Unit step on filter .....	62
Figure 29 - Schematic connections.....	62
Figure 30 - Low-pass filter camshaft.....	62
Figure 31 - Amplitude spectrum of a sampled signal .....	63
Figure 32 - Antialiasing with use of a filter .....	64
Figure 33 - Practical use of the anti-aliasing filter .....	64
Figure 34 - 16-bit, $F_s = 192$ kHz, 12 <sup>th</sup> order filter .....	70
Figure 35 - 16-bit, $F_s = 510$ kHz, 6 <sup>th</sup> order filter .....	71
Figure 36 - Schematic overview of ADS8568 .....	72
Figure 37 - Decoupling bipolar high voltage.....	73
Figure 38 - Schematic view of the hardware configuration.....	75
Figure 39 - REFIO decoupling equivalent of 500 nF.....	75
Figure 40 - Decoupling REFX pins.....	76



Figure 41 - Signal path .....	76
Figure 42 - Filter solutions.....	77
Figure 43 - Filter comparison .....	77
Figure 44 - 6th order Butterworth filter simulation .....	78
Figure 45 - Butterworth filter design with Sallen-key topology .....	79
Figure 46 - Cylinder pressure filter Bode plot.....	81
Figure 47 - Fired and motored cylinder pressures .....	82
Figure 48 - Comparison of knock versus thermal shock.....	82
Figure 49 - CPS filter step response to 3.5V.....	83
Figure 50 - Voltage divider to level shift input signal.....	86
Figure 51 - frequency response of the filter .....	86
Figure 52 - Filter response on transient input.....	88
Figure 53 - Filter schematic for liquid pressure sensor .....	88
Figure 54 - Manifold air pressure implementation.....	89
Figure 55 - DC inputs analog filter .....	90
Figure 56 - General purpose analog filter .....	90
Figure 57 - XADC functional diagram.....	91
Figure 58 - XADC differential passive filters example.....	92
Figure 59 - Feedback block diagram of lambda sensor interface .....	94
Figure 60 - LSU 4.9 equivalent circuitry .....	97
Figure 61 - Virtual ground circuitry.....	98
Figure 62 - Simulation of virtual ground in LTSpice .....	99
Figure 63 - DAC for pump current control.....	99
Figure 64 - Pump current circuitry .....	100
Figure 65 - Simulation of pump current circuitry.....	101
Figure 66 - Nernst cell circuitry .....	102
Figure 67 - Simulation of Nernst cell amplified output vs Nernst cell voltage .....	103
Figure 68 - Schematic design of heating circuitry .....	104
Figure 69 - Simulation of heater current sensing circuitry.....	105
Figure 70 - Two-plane crankshaft on an 8 cylinder engine .....	106
Figure 71 - 4 <sup>th</sup> order Butterworth filter design with Sallen-key topology.....	108
Figure 72 - Bode-plot of Lambda filter .....	108
Figure 73 - Step response in Lambda driver circuit .....	109
Figure 74 - Lambda filter step response .....	109
Figure 75 - Schematic implementation for analog outputs .....	110
Figure 76 - Complete system simulation.....	110
Figure 77 - Monte Carlo simulation of complete system with 10% tolerances .....	111
Figure 78 - Simplified schematic of TPIC8101 .....	113
Figure 79 - Knock sensor interface .....	113
Figure 80 - Flex fuel interface circuitry .....	117
Figure 81 - Model of buffer circuit with low pass filter.....	117



Figure 82 - Basic diagram ignition signal.....	119
Figure 83 - Simulation of ignition interface circuit .....	120
Figure 84 - Ignition signal circuitry .....	120
Figure 85 - Mounting location and basic equivalent circuitry .....	121
Figure 86 - Brake pedal depressed signal circuitry .....	121
Figure 87 - Schematic implementation of MAF sensor .....	122
Figure 88 - General purpose inputs and flex fuel sensor interface .....	123
Figure 89 - Difference amplifier .....	125
Figure 90 - Battery monitor voltage translator circuit .....	126
Figure 91 - LTSpice schematic drawing.....	126
Figure 92 - Amplifier nominal input .....	127
Figure 93 - Amplifier nominal output.....	127
Figure 94 - Minimum and maximum values .....	128
Figure 95 - Low-side topology .....	131
Figure 96 - Simplified block diagram of ignition coil .....	132
Figure 97 - Coil driver schematic .....	133
Figure 98 - Flyback diode configuration .....	133
Figure 99 - Simplified wiring diagram of injector .....	135
Figure 100 - Injection driver schematic.....	136
Figure 101 - Twin turbo configuration.....	139
Figure 102 - Boost controller and fuel pump .....	141
Figure 103 - MC33931 schematic implementation .....	142
Figure 104 - MC33931 current to voltage feedback.....	144
Figure 105 - Feedback circuit simulation.....	145
Figure 106 - Voltage feedback simulation results .....	145
Figure 107 - Throttle body feedback implementation .....	146
Figure 108 - Block diagram of NCV8401ADTRKG .....	148
Figure 109 - General purpose low-side switch.....	148
Figure 110 - Block diagram of LM5112 .....	149
Figure 111 - Stage driver configuration .....	150
Figure 112 - PWM block diagram .....	151
Figure 113 - FT2232 schematic implementation .....	154
Figure 114 - USB connector and signal lines.....	155
Figure 115 - FT2232HL configuration data EEPROM.....	155
Figure 116 - 12 MHz oscillator for USB clock.....	156
Figure 117 - Data transmission LEDs.....	156
Figure 118 - JTAG header Schematic.....	158
Figure 119 - JTAG enable jumper .....	158
Figure 120 - Ethernet interface jack .....	160
Figure 121 - Wi-Fi module schematic implementation .....	162
Figure 122 - CAN Transceiver schematic.....	164



Figure 123 - Traditional valve mechanism principle .....	166
Figure 124 - Timing belt .....	167
Figure 125 - Pneumatic valve control module concept .....	168
Figure 126 - CAN bus data frame .....	172
Figure 127 - Voltage translator .....	174
Figure 128 - Voltage divider circuit .....	176
Figure 129 - Linear regulator.....	177
Figure 130 - Buck converter principle circuit .....	178
Figure 131 - Cold crank cycle.....	180
Figure 132 - Starter motor start-up voltage drop during cold cranking .....	181
Figure 133 - Audi A4 battery voltage testing .....	182
Figure 134 - Battery charging voltage .....	182
Figure 135 - Starter motor crank loading.....	183
Figure 136 - Voltage spike caused by load dump .....	184
Figure 137 - LT4363 package pinout.....	186
Figure 138 - LT4363 simulation circuit .....	187
Figure 139 - Zener voltage regulator .....	190
Figure 140 - Zener emitter follower voltage regulator .....	190
Figure 141 - Power dissipation in R3 .....	191
Figure 142 - IRFS4127PBF Safe Operating Area (SOA) .....	193
Figure 143 - LT4363 pass transistor power dissipation simulation result .....	194
Figure 144 - Reverse polarity protection circuit.....	196
Figure 145 - ECU power protection circuit schematic implementation .....	197
Figure 146 - Power configuration drawing.....	199
Figure 147 - 3.3 V 5 A supply .....	200
Figure 148 - 3.3 V power supply schematic implementation.....	204
Figure 149 - Crank test simulation in LTSpice.....	205
Figure 150 - Load test simulation in LTSpice .....	206
Figure 151 - LTM4609 5V supply .....	208
Figure 152 - Ripple comparison of buck and buck/boost mode .....	211
Figure 153 - LTM4609 crank test simulation.....	212
Figure 154 - LTM4609 load test simulation .....	213
Figure 155 - 5 V power supply schematic implementation.....	213
Figure 156 - LT3724 negative 5 V supply.....	214
Figure 157 - Crank test simulation .....	218
Figure 158 - Load test simulation.....	219
Figure 159 - -5 V power supply schematic implementation.....	219
Figure 160 - 15 V configurations .....	222
Figure 161 - Load test on 15 V power supply.....	225
Figure 162 - Regulation to sudden load changes .....	225
Figure 163 - 15 V power supply schematic implementation.....	227



Figure 164 - N15V power supply configurations.....	228
Figure 165 - Load test for -15 V power supply .....	233
Figure 166 - Regulation to sudden changes in load.....	234
Figure 167 -15 V power supply schematic implementation.....	235
Figure 168 - Low-ESR capacitors to help with fast load transients .....	236
Figure 169 - RTC battery holder implementation.....	237
Figure 170 - Programmable LEDs schematic implementation.....	238
Figure 171 - Test point pads layout.....	239
Figure 172 - 0 $\Omega$ resistors implementation example.....	239
Figure 173 - Reset switch and jumper header.....	240
Figure 174 - PGOOD LEDs implementation.....	241
Figure 175 - Top level overview .....	251
Figure 176 - The board selection in Vivado .....	252
Figure 177 - Configured IP Processing System .....	254
Figure 178 - Build hardware platform.....	256
Figure 179 - Design process diagram .....	257
Figure 180 - Flow Steps Diagram .....	258
Figure 181 - Overall network communication.....	260
Figure 182 - PetaLinux Activity Diagram.....	261
Figure 183 - Sequence diagram of WebSocket communication .....	263
Figure 184 - Example of parameter storage.....	269
Figure 185 - Log storage structure.....	269
Figure 186 - Overall structure .....	271
Figure 187 - Application structure.....	272
Figure 188 - Development process for Android .....	273
Figure 189 - Android SDK setup .....	274
Figure 190 - Most important PhoneGap commands .....	275
Figure 191 - Android application stack.....	275
Figure 192 - From sketch to functional application.....	276
Figure 193 - Activity diagram of the application use .....	277
Figure 194 - Activity diagram for our Android application.....	279
Figure 195 - CPU usage while fully running the application.....	280
Figure 196 - CPU usage while minimized with Zynq connection.....	281
Figure 197 - CPU usage while minimized and unconnected with Zynq .....	282
Figure 198 - Navigation toolbar.....	285
Figure 199 - Screenshot of the dashboard design.....	286
Figure 200 - CanvasJS customized design.....	289
Figure 201 - Performance results .....	296
Figure 202 - Sensors/Drivers overview.....	300
Figure 203 - Necessary 32-bits libraries.....	302
Figure 204 - 3D CAD model of Molex 34763-0001 .....	307



Figure 205 - Overview of header connector bays.....	307
Figure 206 - 3D Section view of the ECU header.....	308
Figure 207 - V8 engine sensors and actuators typical location.....	308
Figure 208 - Bay X connector 48 pins. ....	309
Figure 209 - Bay Y connector 53 pins. ....	309
Figure 210 - Unique numbering system for KPEC connectors.....	310
Figure 211 - XC4 location in bay X connector and underside of ECU header.....	310
Figure 212 - Schematic implementation of ECU connector.....	311
Figure 213 - Mounting of ECU header towards PCB.....	312
Figure 214 - Correct mounting of wires in mating connector.....	312
Figure 215 - Wiring schematics for sensor and actuator connection to ECU.....	313
Figure 216 - Connection of mating connectors to ECU header. ....	313
Figure 217 - Bay X pin locations.....	337
Figure 218 - Bay Y pin locations.....	339
Figure 219 - Bay Z pin locations.....	341

## List of tables

Table 1 - Document structure.....	21
Table 2 - Temperature sensor characteristics.....	42
Table 3 - LTC2983 input channels configuration.....	46
Table 4 - MPU-9150 pins and signal description.....	53
Table 5 - MAX9924 operating modes.....	54
Table 6 - Power operational range.....	73
Table 7 - Preselected hardware configuration.....	75
Table 8 - Butterworth filter table.....	79
Table 9 - Discrete components CPS filter.....	80
Table 10 - XADC channel assignments.....	93
Table 11 - Pugh-matrix for selection of wideband lambda sensor.....	96
Table 12 - Pugh-matrix for selection of lambda sensor interface. ....	96
Table 13 - LSU 4.9 signal overview. ....	97
Table 14 - Maximum typical levels for DAC.....	101
Table 15 - Net aliases for signals with description and characteristics.....	103
Table 16 - Lambda value comparison for gasoline vs bioethanol.....	105
Table 17 - Pin configuration TPIC8101.....	114
Table 18 - Specifications for the sensor.....	116
Table 19 - Flex fuel sensor signal description.....	116
Table 20 - Saturated versus Peak and Hold (P&H).....	134
Table 21 - Turbocharger vs supercharger.....	137
Table 22 - Typical characteristics of a solenoid boost controller.....	138



Table 23 - Typical characteristics of fuel pumps .....	140
Table 24 - MC33931 pinout description .....	143
Table 25 - TTL vs CMOS .....	149
Table 26 - USB 2.0 signal descriptions.....	153
Table 27 - FT232HL Dual UART pin configuration.....	153
Table 28 - JTAG Signal Description .....	157
Table 29 - Ethernet PHY LED connections.....	161
Table 30 - 32-bit data package assignment .....	171
Table 31 - Pin configuration.....	174
Table 32 - Effective power at 6 and 12 V .....	184
Table 33 - LT4363 pin description.....	186
Table 34 - Power calculation summary.....	198
Table 35 - LTC3891 pin description.....	200
Table 36 - Voltage thresholds .....	203
Table 37 - LTM4609 pin description.....	208
Table 38 - LTM4609 Max/Min .....	211
Table 39 - LT3724 pin discription.....	214
Table 40 - Maximum and minimum rating for the LT3724 .....	217
Table 41 - LT3580 pin description.....	222
Table 42 - LT8580 pin description.....	228
Table 43 - Power design summary.....	236
Table 44 - Programmable LEDs pin configuration.....	238
Table 45 - General purpose inputs and outputs.....	242
Table 46 - Derating factors.....	244
Table 47 - Capacitor derating table.....	245
Table 48 - Resistor derating table.....	247
Table 49 - Transistor derating table.....	247
Table 50 - Diode derating table .....	248
Table 51 - Magnetic component derating table.....	248
Table 52 - Integrated circuit derating table .....	249
Table 53 - Optocoupler derating table .....	249
Table 54 - Power dissipation parameter description.....	250
Table 55 - Required Hardware for PetaLinux .....	253
Table 56 - Progress from Vivado to SDK .....	255
Table 57 - First Draft Analyse low performance .....	294
Table 58 - JQuery Performance Code.....	296
Table 59 - Physical specifications for mating connectors.....	307
Table 60 - Power budget 3.3 V .....	319
Table 61 - Power budget 5 V .....	319
Table 62 - Power budget -5 V .....	319
Table 63 - Power budget 15 V .....	320





Table 64 - Power budget -15 V .....	320
Table 65 - Cold crank power budget .....	320
Table 66 - Parameter explanation for table 6 .....	320
Table 67 - Recommended sensors, actuators and parts for KPEC ECU. ....	321
Table 68 - Power signal description .....	323
Table 69 - XADC channel connections .....	323
Table 70 - Signal description list .....	324
Table 71 - Connector top pinout .....	328
Table 72 - Connector bottom pinout.....	331
Table 73 - Connector left pinout .....	334
Table 74 - Connection and description table for bay X.....	337
Table 75 - Connection and description table for bay Y.....	339
Table 76 - Connection and description table for bay Z.....	341

## List of codes

Code 1 - Configure system-top.dts .....	260
Code 2 - C: Part of the startup code .....	262
Code 3 - Bash: Environmental variable for websocketd .....	264
Code 4 - Bash: Run websocketd with our application .....	264
Code 5 - C: Generate dummy values for temperature sensors.....	265
Code 6 - C: Structure for storing the parameters .....	267
Code 7 - C: Time check and creation of thread.....	267
Code 8 - C: Store the data into the flash memory.....	268
Code 9 - Set up current time in PetaLinux.....	268
Code 10 - JavaScript: View the content from sensor_data.txt.....	269
Code 11 - Makefile: Create executable application.....	270
Code 12 - Source to settings64.sh .....	270
Code 13 - Java: CordovaApp.java .....	279
Code 14 - jQuery event handling .....	284
Code 15 - JavaScript: Cylinder view for coolant temperature.....	288
Code 16 - JQuery: Window onLoad .....	290
Code 17 - Diagram Design.....	290
Code 18 - Update Diagram .....	291
Code 19 - JavaScript: Part of the code to fetch the data from the socket.....	293
Code 20 - jQuery performance code .....	295
Code 21 - XML: config.xml configuration file .....	297
Code 22 - XML: AndroidManifest.xml configuration file.....	298
Code 23 - DSL: build.gradle file for Android application.....	299
Code 24 - Bash: 32-bits libraries .....	302





Code 25 - Bash: Permission command .....	303
Code 26 - Bash: Source into Vivado and/or SDK folders .....	303
Code 27 - Bash: From CentOS.....	304
Code 28 - Bash: From Ubuntu .....	304
Code 29 - Bash: Install PhoneGap CLI.....	304
Code 30 - Bash: Configure GitHub alias.....	305
Code 31 - Bash: Clone the KPEC repository.....	305
Code 32 - Bash: Make the files ready to be pushed.....	305
Code 33 - Bash: Add a text (Important for the traceability) .....	305
Code 34 - Bash: Push your content to private repository .....	305
Code 35 - Bash: To download the code from private to the host machine .....	305



## Introduction

This is the design document for KPEC and it contains the results encountered during the design phase of the project. This document is supposed to give a thorough understanding of the detailed design of hardware and software for this project.

An important notion about the design document is that it is the most important document in respect to further development of the KPEC ECU. The design document is the primary source of information regarding any technical aspect of the design, and is therefore essential for future hardware and software development. With this in mind, the design document brings up introductory basics in the beginning of each subchapter about that particular part of the design. The design document a key part of the product itself.

The project is following an iterative evolutionary agile model with the basic principle that the world is changing, and that customer objectives changes throughout the development phase. To ensure overall quality and customer satisfaction, frequent iterations with our customer with delivery of parts of software and hardware is vital. Design choices according to system requirements are validated during review meetings and discussed to ensure that the finished product is delivered according to customers objectives.

We are following a Systems Engineering approach during the iterations in the design phase. A systems engineering approach focuses on the customer need in the concept of operations domain, follows with requirements and then design to fulfil the requirements. Each design action is tested against its requirement as a verification method, and if the test succeeds the design is implemented in the final design. If the test fails, the design or requirement is changed according to the result of system validation with customer.

This project model allows us to work in parallel with software and hardware design with daily design iterations. There will be two major system iterations, where our customer will give us feedback on the hardware and software design. For further information regarding our project model and iteration plan, refer to the project plan document<sup>1</sup>.

---

<sup>1</sup> KPEC 2014, Gudbrandsen, Pedersen, Schäffer, Dinh, Hasan, Asjad, Project plan (Rev 4.0), HBV.



The first chapter of this document summarizes the design in short with little background information. In order to get a full understanding of the different subsystems, each part of the design is described in greater detail in the later chapters in this document. Table 1 provides a graphical structural overview of the main chapters in the document. This design document is made for interactivity. All references are clickable and will navigate to the referenced part of the document when clicked in the PDF version.

**Table 1 - Document structure**

The first chapter of this document focuses on a system overview of the ECU. This provides the reader with general specifications of the KPEC ECU, the purpose of the ECU, and basic concepts regarding combustion theory. The chapter extends to an overview presentation of the hardware and software solutions.

The second chapter presents the required information for interfacing the TE0720, with a thorough description of assigned signals towards the TE0720 connectors.

The third chapter describes all the sensor interfaces, and the fourth chapter describes the driver interfaces.

The fifth chapter describes the external communication interfaces which are designed for the ECU. Chapter six describes the power design for internal and external circuitry.

Chapter seven describes miscellaneous features which are important for the practical utilization of the ECU. Chapter eight defines external requirements (KDA) regarding derating of electrical components.

Chapter nine gives an extended overview of the software design of the project. Chapter ten describes the initiation of the Zynq module, and especially the installation of PetaLinux operating system. Chapter eleven describes the design and development of the tablet application.

Chapter twelve provides the reader with practical information regarding implementation of the ECU in a vehicle. This chapter defines the pinout assignment of the ECU-connector, and is intended to make a transition to real life application from a system perspective.

The appendixes of the design document includes a list of abbreviations, power budget, recommended sensors (for purchaser), signal names and description, board-to-board connection tables, and ECU connector tables. Attached to the document as PDF files is the complete schematic design and application circuitry for the ECU.

Introduction
1. System overview
2. TE0720 Micromodule interfacing
3. Sensor interfaces
4. Drivers
5. Communication interfaces
6. Power design
7. Miscellaneous features
8. Derating of electrical components
9. Software design
10. Zynq design
11. Android application design
12. ECU Connectivity
Appendix



## 1. System overview

The engine management system interfaces multiple sensors and actuators to control the combustion process of an internal combustion engine (ICE). To monitor the combustion process accurately, one should have a thorough understanding of the mechanical aspects of an engine, as well as some theory regarding combustion. KPEC's FPGA based ECU will be designed for a turbocharged 4-stroke petrol/ethanol engine, with support for up to 8 cylinders at 16.000 RPM. For more information regarding system requirements to our ECU, refer to our requirements specification<sup>2</sup>.

### 1.1. KPEC ECU specification summary

The KPEC ECU is a highly user configurable, high performance Programmable Engine Control Unit. The KPEC ECU is designed for petrol engines with up to 8 cylinders. Engine tuning and parameter real time view is achieved wirelessly via Wi-Fi communication directly to the user's phone or tablet device through a dedicated app. The KPEC ECU can interface a vast array of automotive performance sensors, with 17 different sensor interfaces for a total of 44 sensors. There are 5 different high power drivers for a total of 24 low side switched output stages. The main specs of the KPEC ECU are:

- High performance Zynq-7000 All Programmable SoC
  - Artix-7 FPGA fabric for signal interfacing and data acquisition
  - Dual core ARM Cortex-A9 @ 800 MHz running PetaLinux
- 1 GB DDR3 SDRAM
- 4 GB internal NAND Flash memory for data storage and logging
- Wi-Fi
  - 2.4 GHz IEEE 802.11 b/g/n
  - Wi-Fi Direct
- 2x High speed CAN 2.0B transceivers
- 9-axis MEMS accelerometer, gyroscope and magnetometer
- On-board high precision barometer
- 16-channel 12-bit differential ADC for internal analog signals and inputs
  - Manifold Air Pressure sensors (MAP)
  - Oil and fuel pressure
  - Throttle body feedback inputs
  - Lambda probe measurements (O<sub>2</sub>)
  - Accelerator pedal sensors
  - Battery voltage monitoring
- 8x High speed 8-channel 16-bit ADC optical cylinder pressure sensor interface
- High precision 24-bit 20-channel ADC temperature measurements

---

<sup>2</sup> KPEC 2014, Gudbrandsen, Pedersen, Schäffer, Dinh, Hasan, Asjad, Requirements specification (Rev 4.0), HBV.



- 8x Exhaust temperature sensors
  - Oil, coolant and air temperature sensors
  - Internal ECU temperature
- Fully configurable dual wideband Bosch LSU 4.9 Lambda interface
- Variable reluctance sensor interface for flywheel speed and position sensing
- Digital or analog Mass Air Flow sensor (MAF)
- Dual knock sensor interface
- 4x Camshaft position hall effect sensor interface
- Flex fuel sensor interface for automatic E85 fuel recognition and support
- High current throttle body H-bridge driver
- High power, high performance low side switches
  - 8x High current IGBT F1 grade ignition coil drivers
  - Boost controller drivers
  - 8x Fuel injector drivers
  - Fuel pump driver
- General purpose I/O providing excellent expandability options
  - 2x 0-12V General purpose analog outputs
  - 3x 0-5V General purpose analog inputs
  - 4x General purpose low side switch output drivers
  - 2x 5V General purpose digital inputs
  - 4x 5V bi-directional digital I/O
- Protected ignition and brake signal inputs
- High efficiency multi-level on-board power management
- Military grade power protection circuitry for maximum reliability
- Ethernet, USB and JTAG interfaces for programming and debugging

## 1.2. Role of the Engine Control Unit (ECU)

All vehicles manufactured today rely on an engine control unit as the heart of the automotive control system cluster. The use of electronic engine control systems arose in the late 80s with the revolution in microprocessors and embedded technology. In many ways, the advancement in embedded electronics has revolutionized the automotive industry. Environmental, safety and durability requirements increases for each year. The use of sensors to collect knowledge of every aspect of engine operation and status, allows a processing unit to optimize the entire combustion process. In comparison to vehicles back in the 70s, the electronic control of modern cars has cut emissions by 90%, achieved at least a 50% increase in power, uses 25% less fuel and operates up to three times as long before requiring service<sup>3</sup>.

---

<sup>3</sup> Volvo Personbilar Sverige AB 2005, Grunder motorstyrssystem VT2201.



An ECU is able to provide the driver with useful data and warnings at much higher rates than earlier. The ECU is programmed especially for the particular type of engine and desirable performance, whether it is maximum power in a race car, or maximizing fuel efficiency in an ordinary car. Programmable ECUs are a branch of ECUs that allows the user to tweak and reconfigure engine performance. Programmable ECUs are typically found in motorsports applications, where the user must configure the ECU for maximum performance with non-standard parts, as the cars used are heavily modified.

The major issue with the automotive industry and development today is cost optimization. Everything is manufactured for achieving lowest possible manufacturing costs, which leaves less room for new features unless they are an absolute requirement from new regulations. KPEC ECU is a low volume, high cost product thought for use in motorsports applications. The KPEC ECU employs several features not seen in any ECUs today.

The main difference with KPEC ECU from any existing solutions today is the high measurement accuracy of the unit. The KPEC ECU employs more sensors than existing ECUs, with the primary focus of monitoring each individual engine cylinder. Monitoring each individual cylinder allows the ECU to optimize the combustion process in each cylinder, as differences occur with wear of actuators and mechanical parts. The process can be described as dynamic mapping, where the ECU adapt to changes in the engine over time, instead of following a static table of settings for all conditions.

The KPEC ECU is designed for interfacing optical cylinder pressure sensors (CPS), which is not used outside laboratory tuning and measurements today. Including cylinder pressure sensor interface in the ECU allows the user to tune engine performance even further than previously possible. Monitoring the pressure changes in each individual engine cylinder reveals important information about engine condition. Accurate cylinder pressure measurements reveals issues such as engine knocking, and can adjust ignition more precisely for the cylinder where the knock occurs. Other issues such as leakage in engine valves are hard to detect in today's engine management systems, which can be detected by cylinder pressure measurements. As the engine mechanical condition changes with wear, the ECU can adapt to the changes with the information collected from the cylinder pressure sensors.



### 1.3. Combustion process overview

A basic combustion engine consists of engine block, piston, rod, crankshafts, camshafts and valves. It converts chemical energy (compressed fuel and air mixture) to mechanical energy by rapid expansion in the combustion chamber after ignition occurs. A normal combustion sequence utilizes a combustion pressure of 60 bar in 2 ms at a maximum of 2000 °C<sup>4</sup>. The combustion process occurs during four phases, intake, compression, combustion and exhaust during 720° of crankshaft motion.

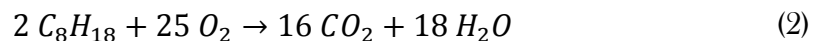
During the intake stroke, the piston goes down (until 180°) and air and petrol enters the combustion chamber. During compression stroke, the piston goes up to TDC (Top Dead Center) and air and fuel are compressed to typically 13 bar at 360° crank revolution. During combustion stroke, the spark plug ignites the air and fuel mixture and the piston is forced down and produces mechanical power until 540°. During exhaust stroke, the piston goes up and forces the remaining exhaust (unburnt fuel and air) out of the combustion chamber at 720°. The process repeats itself for each 720°.

The combustion stroke does only occur 1 time per 720° crankshafts revolutions. This implies that power is only produced between 360° - 540°, one cylinder engine has therefore poor efficiency. The displacement between power strokes depends on the number of cylinders,  $n$ ;

$$PS_D = \frac{720^\circ}{n} \quad (1)$$

This implies that an engine with eight cylinders is capable of producing twice the power of a four cylinder. Torque and power are mechanical parameters that are unique for each engine produced. To increase engine performance, turbocharging or supercharging increases power and torque output to 50% compared to a naturally aspired engine.

An ideal combustion process gives the following chemical equation<sup>5</sup>;



However, this equation represents a theoretical model with stoichiometric air to fuel ratio, defined as

---

<sup>4</sup> Volvo Personbilar Sverige AB 2005, Grunder motorstyrsystem VT2201.

<sup>5</sup> David Giessel 2002, Physics Behind Modern 4 Stroke engines.



$$\lambda = 1.0 = \frac{14.7 \text{ kg air}}{1 \text{ kg petrol}} \quad (3)$$

14 kilograms of dry air at 20°C equals approximately 12.2 m<sup>3</sup> of air. If the air to fuel ratio deviates from the stoichiometric ratio, we have either lean ( $\lambda > 1$ ) or rich ( $\lambda < 1$ ) mixture, and this gives an impact of the efficiency of the engine and the exhaust composition. Unwanted exhaust gases like *HC*, *CO* and *NO<sub>x</sub>* occurs during every combustion process due to the fact that the world is not ideal.

To increase the speed of an internal combustion engine, the amount of air and fuel injected to the engine must be increased. The combustion velocity is constant 30 ms, and the combustion time from TDC to end of the stroke is approximately 2 ms. This implies that the ignition timing must be advanced when the engine speed is increased in order to maintain efficiency. The composition of air and fuel must also be corrected according to external parameters as temperature, humidity and pressure.

To control the combustion process, there are three main sensor parameters for an engine to run; load, position and speed. Load indication is determined by the pressure in the inlet manifold (wide open throttle gives high load). Position of the crankshaft is used to determine the position of 90° before TDC (for ignition sequence). Speed of the crankshaft is used to determine ignition timing advancement and mixture of fuel and air by adjusting the activation of fuel injection.

Fuel injection activation time is basically determined by a fuel lookup table with correction factors (air/engine temperature, barometric pressure, manifold pressure, lambda compensation, ethanol fuel content, battery voltage compensation). The timing when the ignition should occur is determined by the lookup table, and is compensated by the acceleration desire, engine load and engine temperature.

Activation of the ignition system is determined by a trigger signal when the flywheel is located 90° BTDC. This activated a sequence (fixed for motor configuration) of activations of the ignition coils, so that the air and fuel mixture of the cylinder which is at TDC are ignited. The ignition timing must be compensated for to ensure that the ignition is occurring at the right instance according to load and engine speed. The interface design towards these sensors and actuators are described in the detail in this document.



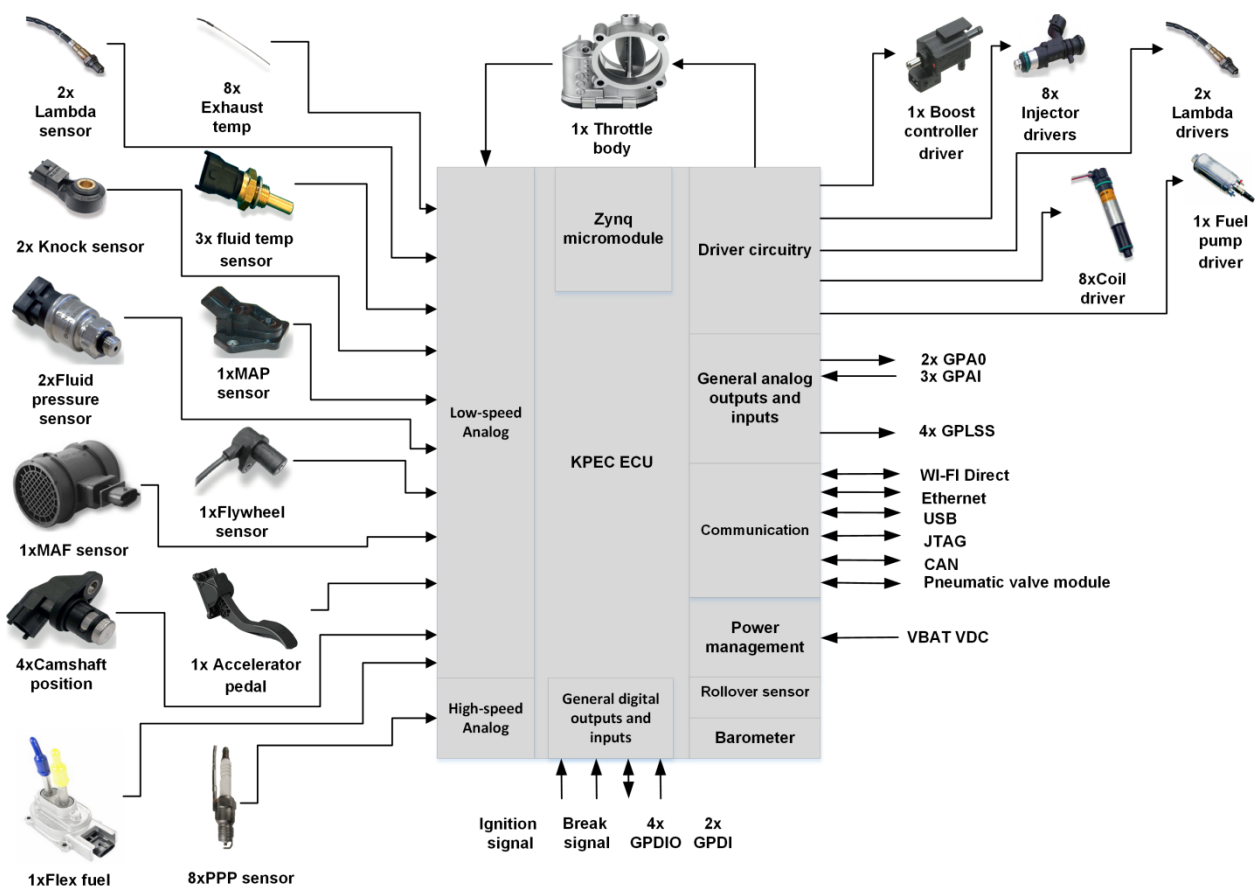


## 1.4. Hardware design overview

This section provides the reader some basic information about the hardware design part of this project.

The KPEC ECU is based on a Trenz Electronics TE0720 micromodule, a powerful Zynq-7000 SoC based processing platform. The hardware design is principally a motherboard design for interfacing this platform to all sensors and actuators in order to be able to electronically control the combustion process of the engine. Details of the TE0720 interfacing are described in chapter 0.

Figure 1 shows hardware overview with the external sensors, actuators and interfaces towards the ECU and the Zynq micromodule.



**Figure 1 - Overview model**

As seen on Figure 1, the KPEC ECU can interface 17 different sensors, 44 in total on the input side. There are 6 different actuators types, 24 in total on the driver side including all general purpose I/O. There are 6 different communication interfaces.

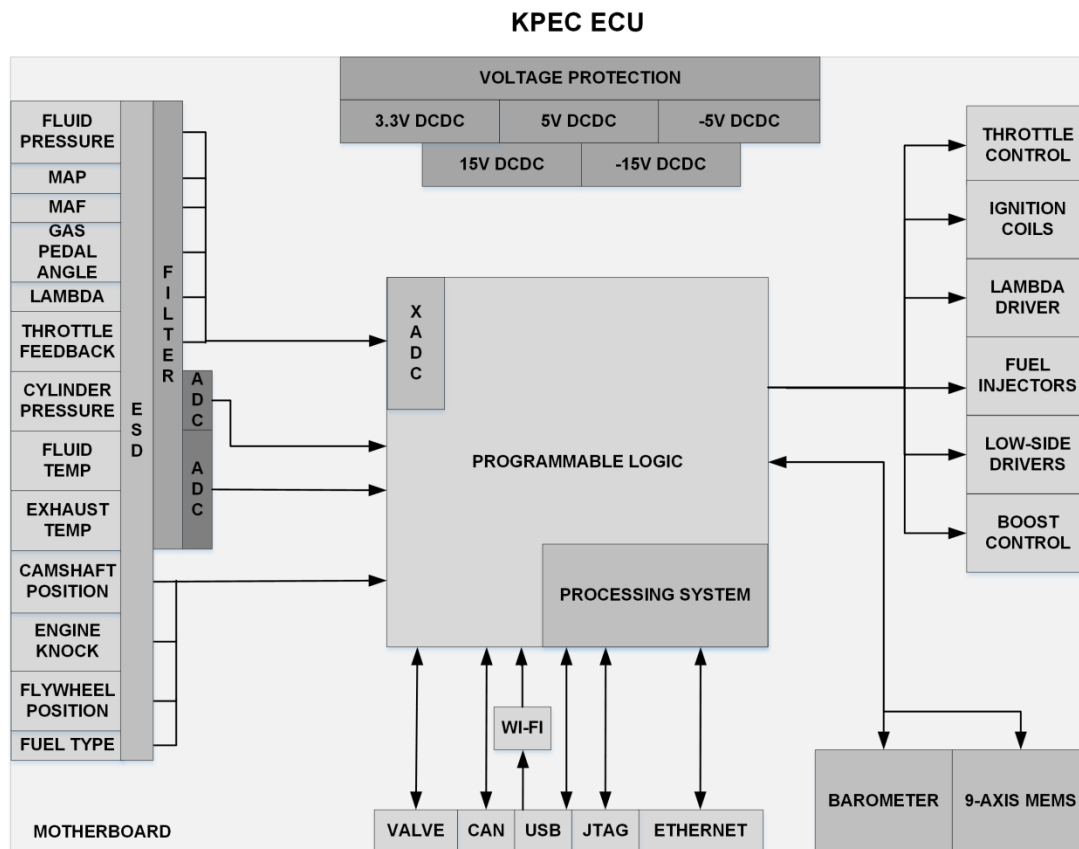
The main challenge of interfacing so many different sensors, is that they exhibit different characteristics and we need to handle all of them, some are analog, some are



digital. While some sensors give out a 0 - 5 V output, other gives out a digital output with varying frequencies and/or duty cycles. Most actuators are generally in principle controlled in the same way, but the drivers needs to be individually configured and sized for the actuator it controls.

Some sensors that outputs 0 - 5 V needs to go through an analog filter to filtrate any noise that is absorbed in the signal as it passes through wires in the noisy engine bay of the vehicle. Non-filtered signals can provide wrong readings as any noise carried in the signal is seen as a part of the signal by the receiver. After the filtration the signal goes through an analog to digital converter (ADC) to convert from the continuous analog plane to discrete digital representation. This has to be done because the ECU can only process data digitally. There are challenges in choosing ADCs and designing filters, if not done correctly the signal that we want to process will be distorted and unusable.

While sensors that exhibits digital outputs can be fed directly into the ECU without any filtration and conversion, as these are only two-state signals and robust to noise in nature.



**Figure 2 - Hardware implementation**



Figure 2 shows a detailed view of the internal hardware subsystems that must be designed in order to interface the different sensors and actuators that makes up the engine control system.

The sensor interfaces are shown to the left in Figure 2, with filters and ADCs as described above. The Zynq System on a Chip (SoC) is portrayed in the middle. The Zynq-7000 SoC is a dual core ARM Cortex-A9 processor combined with FPGA logic in a single package. The FPGA has an internal 16 channel 12-bit ADC, marked named *XADC*, which are used for measuring analog sensor values in the ECU.

The actuators are to the right in Figure 2. We have 6 different actuators that must be controlled. Using sensor data, the ECU processes and calculates when to activate each actuator. All the actuator uses a low-side switching topology, which means that the switch is between the load (the actuator) and battery ground.

There are 6 different communication interfaces shown in the bottom of Figure 2. The interfaces such as USB, JTAG and Ethernet are thought used primarily for software development. CAN (Controller Area Network) is an automotive standard communication bus, used as high speed communication line between the ECU and other external modules in the car, such as ABS, ESP etc. The valve interface is a dedicated CAN bus used to communicate with an external pneumatic valve module. The Wi-Fi is for wireless tablet communication when the user reads engine parameters and performs tuning in real time.

A challenge when designing automotive electronics is the harsh nature of an engine bay where the ECU is typically placed. Whenever selecting components, considerations about vibration, moisture and temperature must be taken into account. An enclosed casing helps with providing protection against the outside environment, but at the same time offer a challenge with power dissipation as the electronics disperse heat during operation.



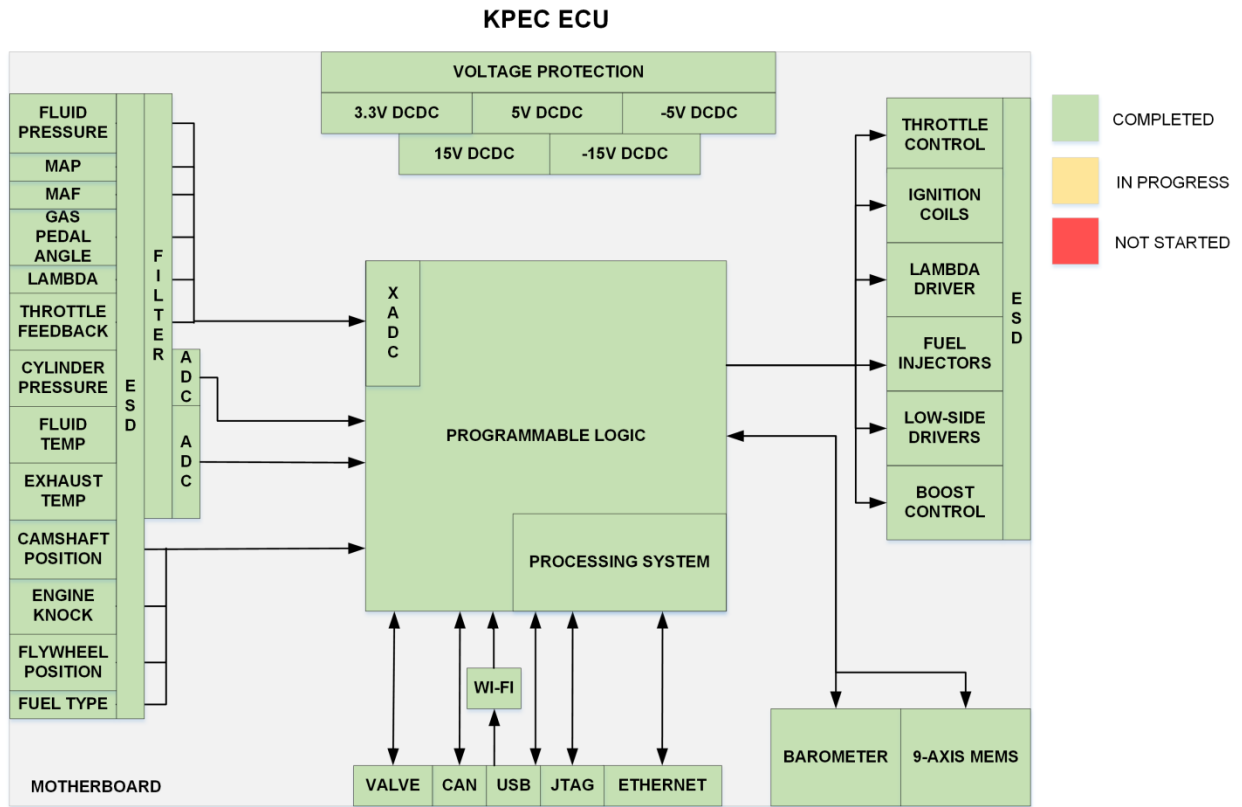
The motherboard consists of many different types of components, such as power components, signal conditioning components and the Zynq micromodule. These devices need to be powered by different voltage levels depending on their use and operation. We have 5 different power supplies where the 3.3 V supplies the Zynq micromodule and most on-board components. Positive 5 V and negative 5 V deliver power to primarily the different interfacing components and external sensors. The remaining two power supplies, the positive 15 V and negative 15 V supplies an ADC which is used to gather information from the cylinder pressure sensors, as well as analog signal outputs. Since a car only has one power source, the battery, means that we have to derive all the different voltage levels from this source. The power supply voltage in a car varies, with possible transients and noise. During our power design for the ECU, this must be taken into account in order to protect the delicate components in the ECU. The protection circuitry handles events such as cold crank, overvoltage, overcurrent, load dump and reverse voltage.

Cold crank occurs when it's cold outside and the starter motor draws a large current of several hundred amperes. The massive load causes the battery voltage, which is nominally 12 V, to drop by several volts down to as low as 6 V. We need to assure that power supplies function at all times, before, during and after cranking.

Load dump occurs when an inductive load is suddenly disconnected in a system. The inductor holds large amount of current in the inductor windings, and when suddenly disconnected, the discharge of the inductor causes large voltage spikes in the system of up to 120 V. Special protection circuitry must be designed for such transients, as they easily are able to cause harm to electronic components.

Reverse voltage happens if the user connects the battery terminals the wrong way.

There is an also many extra features that is not a direct requirement in order to allow further development and changes to the ECU after manufacture. Such features are described in more detail in the design chapters.



**Figure 3 - Hardware design status**

Figure 3 shows the current status in development for each of the subsystems. The first design of every subsystem has been finished and reviewed. Most of the system was reviewed at an external review meeting at KDA. The power design was not finished by that point, but is thoroughly simulated and internally reviewed before schematic implementation for delivery. The KPEC ECU will undergo another review at KDA after delivery, and after approval, the design is ready for PCB layout.



## 1.5. Software design overview

This section will provide with some high level understanding of the software part of this year's project. In the software part of the document, we are using UML and regular diagrams to simplify the complex approach. Since this project will last for several years, we are writing in a way that the next year's students can understand our work without frustration. The software design document is divided into three subgroups, Software design, Zynq design and Android application design. The chapter software design is a soft introduction to the software detail design. This chapter introduces the three main components we are using to develop our system for this year and can be read more about in details in chapter 9.

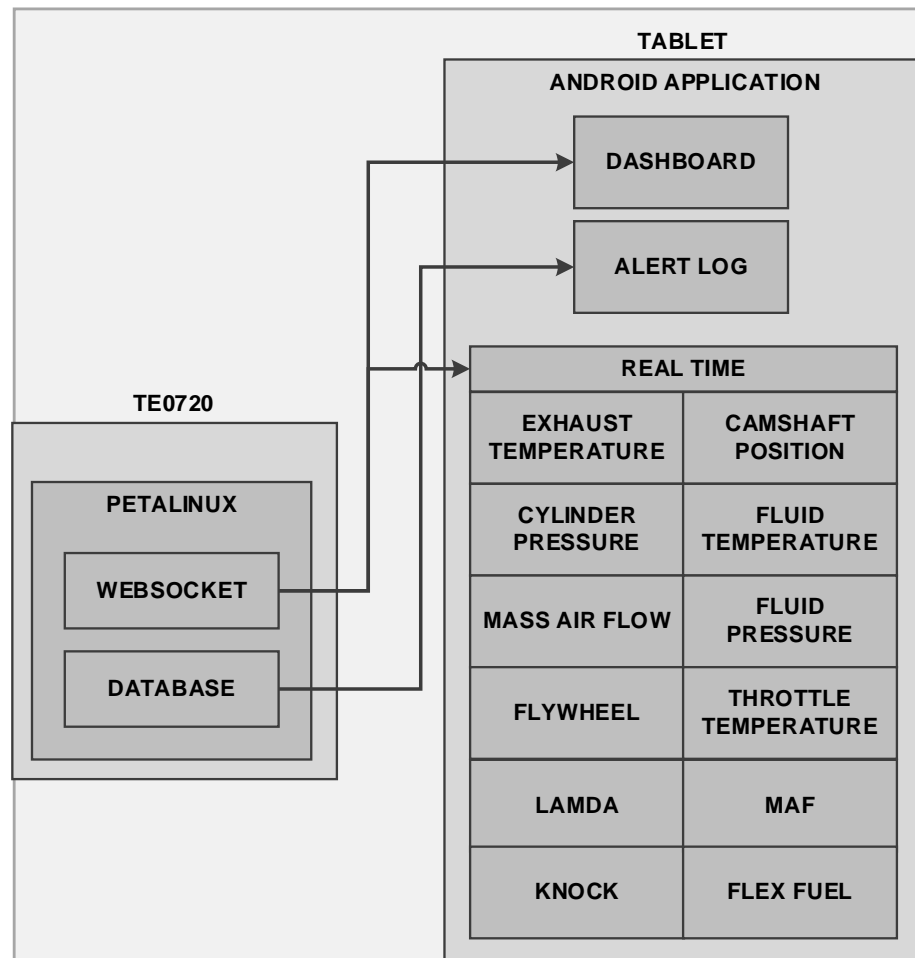
The main software task for this year's project is to setup the TE0720 Micromodule properly and get PetaLinux up and running. PetaLinux is an embedded Linux operating system widely used on Zynq devices, which runs on top of the powerful dual core ARM Cortex-A9 processor. To setup the TE0720 with PetaLinux, we have to go through several steps. The first step is to design the TE0720 to satisfy the requirements. This is done by using the software provided by Xilinx. Here we have to specify which hardware components of the TE0720 we are using. We are then able to use our design to develop a PetaLinux project, which will operate as a fully embedded Linux system. And after our operating system, PetaLinux, is ready and set up as we want, we are developing user applications to run in PetaLinux. The user application will control the serving of data to the Android application through Ethernet interface. So this is a kind of modular system, which consists of different subsystems. You can read more about this in chapter 10.

The last chapter of the software document is about the Android application we are developing and its design decisions. The Android application communicates with our TE0720 to receive parameters from different sensors. We are using a hybrid application framework to develop our Android application. The framework is called PhoneGap and makes it possible to develop applications for several mobile platforms right from the same code structure. So instead of using different programming languages to develop separate applications for each platform, we are using HTML, JavaScript and CSS, which is supported by all mobile platforms, to create the hybrid application.

The application will display real-time parameters from different kind of sensors. These parameters are displayed in either dynamic graphs or from the dashboard. You can read more about our Android application in details in chapter 11.

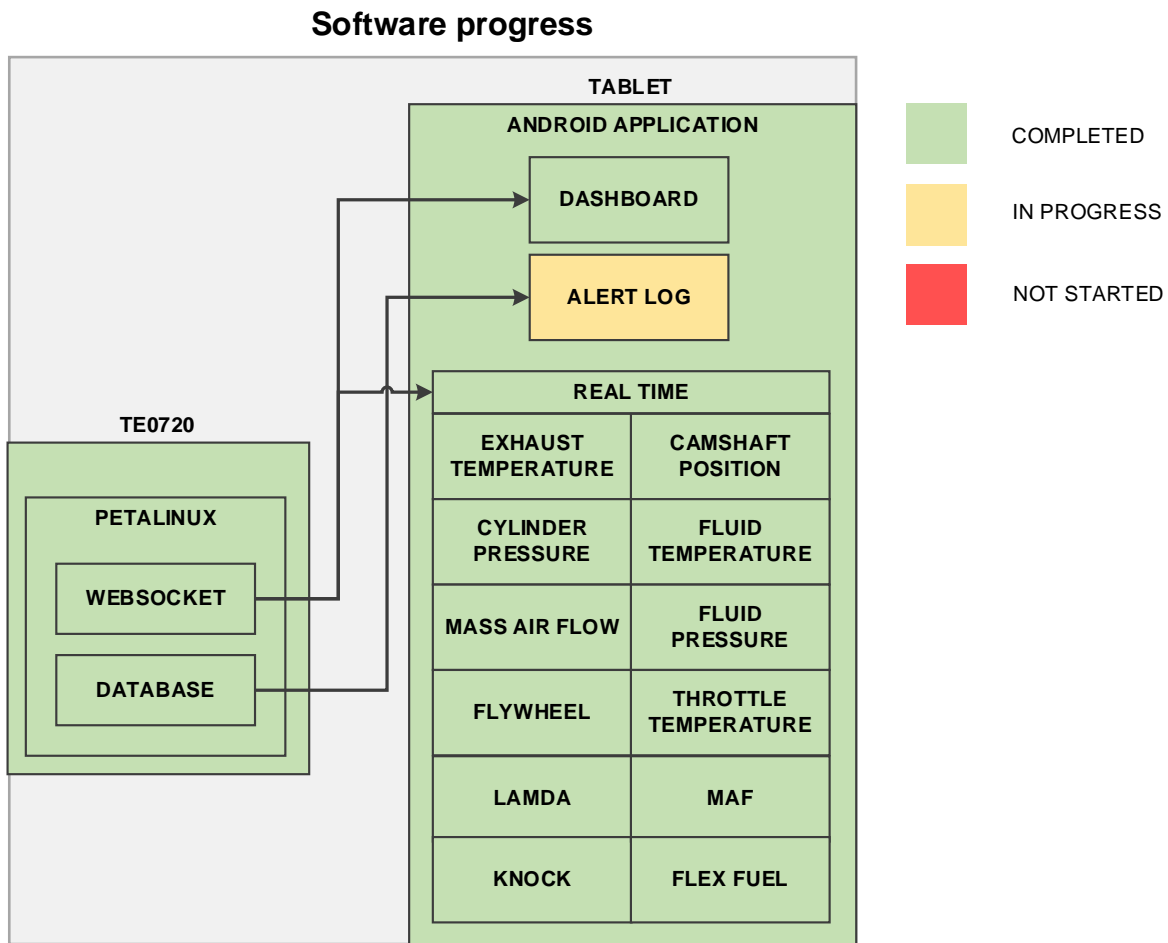


Figure 4 shows a detailed view of the software subsystems as an overall overview of the software development.



**Figure 4 - Software implementation**

As shown in Figure 4, the TE0720 to the left side consists of the operating system PetaLinux which sends parameters to the tablet through WebSocket connection and is storing different errors coming from the sensors into the database. The tablet to the right side receives the parameters from TE0720 and is displaying all the parameters shown inside the real time box in Figure 4.



**Figure 5 - Software design status**

Figure 5 shows the status in development for each of the subsystems. We are finished with all of this year's tasks that we were asked to accomplish by KDA. We also got an optional task from KDA to create log from the parameters on the TE0720, and display that on the tablet. We have done a great deal of that part, but we were not able to finish the overall task. The TE0720 is able to store the parameters into the database, but unfortunately we didn't have enough time to implement a viewable table of the log into the application.

Additionally we have added a configuration manual and recommendations chapter see 11.5, that guides the developers to configure necessary tools for further development on the project.





## 2. TE0720 Micromodule interfacing

The main data processing unit for our system is the TE0720 Micromodule from Trenz Electronics GmbH. The TE0720 is a modular platform for use in demanding data processing applications where real time processing of data is crucial. The heart of the TE0720 is a Xilinx Zynq SoC (System on a Chip), which is an FPGA combined with a high end dual core ARM Cortex A-9 processor.

Traditional ECUs take use of  $\mu$ Cs (Microcontrollers) which are much simpler to program, cheaper and easier to implement in hardware. Their main drawback is limited resources in terms of sequential processing at low clock rates. In order to increase efficiency of ICEs, sensor data must be collected with greater accuracy and processing capabilities. FPGAs are especially efficient in data collection and digital signal processing (DSP). Their nature of true parallel processing makes them excellent in applications of heavy data acquisition and real time processing. Combining FPGA fabric with a Processing System (PS) allows running an operating system with user applications on the same chip as logic blocks implemented in the Programmable Logic (PL) with extremely fast signal intercommunication.



**Figure 6 - TE0720 Micromodule<sup>6</sup>**

Figure 6 shows the TE0720 micromodule. The module itself provides everything required to run the system, including on-board power supplies. FPGAs require multiple voltage levels and external components to function. The TE0720 implements all required components for the FPGA to operate on a very small board, at only 4x5 cm in size. The module is able to operate from a single 3.3 V supply in order to simplify implementation.

---

<sup>6</sup> Figure from Trenz Electronics TE0720 product page



Some key specifications of the TE0720 are<sup>7</sup>:

- Xilinx Zynq Z020 SoC
  - Dual core ARM Cortex A-9 processor @ up to 800 MHz
  - 85K logic cells Artix-7 FPGA fabric
  - Internal ADC with support for 16 external multiplexed channels
- 1 GB DDR3 SDRAM
- On-board power management
  - 1.0 V
  - 1.5 V
  - 1.8 V
- 32 MB Quad SPI flash memory
- 4 GB NAND flash memory
- Gigabit Ethernet transceiver
- System management controller
- Hermaphroditic Board-to-Board (B2B) connectors
  - 2x 100 pin
  - 1x 60 pin
  - 152 FPGA I/O
  - 14 MIO

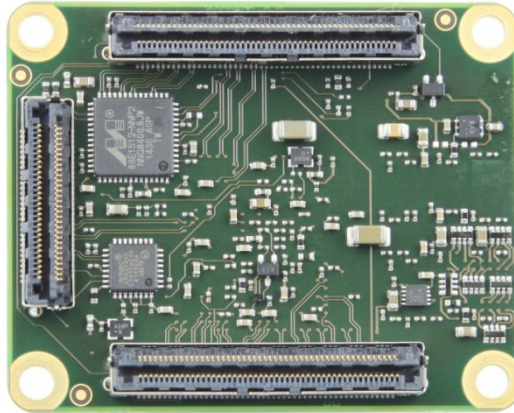
Using a module simplifies hardware design tremendously, as FPGA layout and design is a hefty topic within electronic design. Using a module ready to be set up from a development board also allows us to create and set up parts of the software prior to ECU motherboard manufacture. The ECU motherboard interfacing the B2B connectors on the module is the actual hardware design that is performed in our project by the hardware engineers.

## 2.1. TE0720 B2B interfacing

As mentioned briefly above, the TE0720 will interface the ECU motherboard using Board-to-Board (B2B) connectors. This section leaps into detail of the board level internal communication between the TE0720 and the ECU motherboard components. Complete list of signal name descriptions and pin connections can be found in Appendix IV and Appendix V. We have chosen to follow Trenz Electronics' nomenclature when it comes to labelling the B2B connectors.

---

<sup>7</sup> TE0720 User manual



**Figure 7 - TE0720 bottom with B2B connectors<sup>8</sup>**

The underside of the TE0720 with the B2B connectors is shown in Figure 7. TE uses the notation connector *top*, *bottom* and *left* when referring to these, corresponding to the position of the connectors in Figure 7. Connector top and bottom are 100 pin connectors, while connector left is a 60 pin connector. The connections for the different I/O pins are spread out on these connectors along with power pins and interfaces for other board components such as the Ethernet PHY. The connectors used are hermaphroditic, meaning that there is no difference in the mating connectors between the boards. The only difference is that the connectors on the ECU motherboard are rotated 180 degrees, thus mirroring the contacts. Take note that all odd numbered pins on the TE0720 become even numbered pins on the ECU motherboard and vice versa. The FPGA banks are powered by external supplies.

### 2.1.1. Connector top

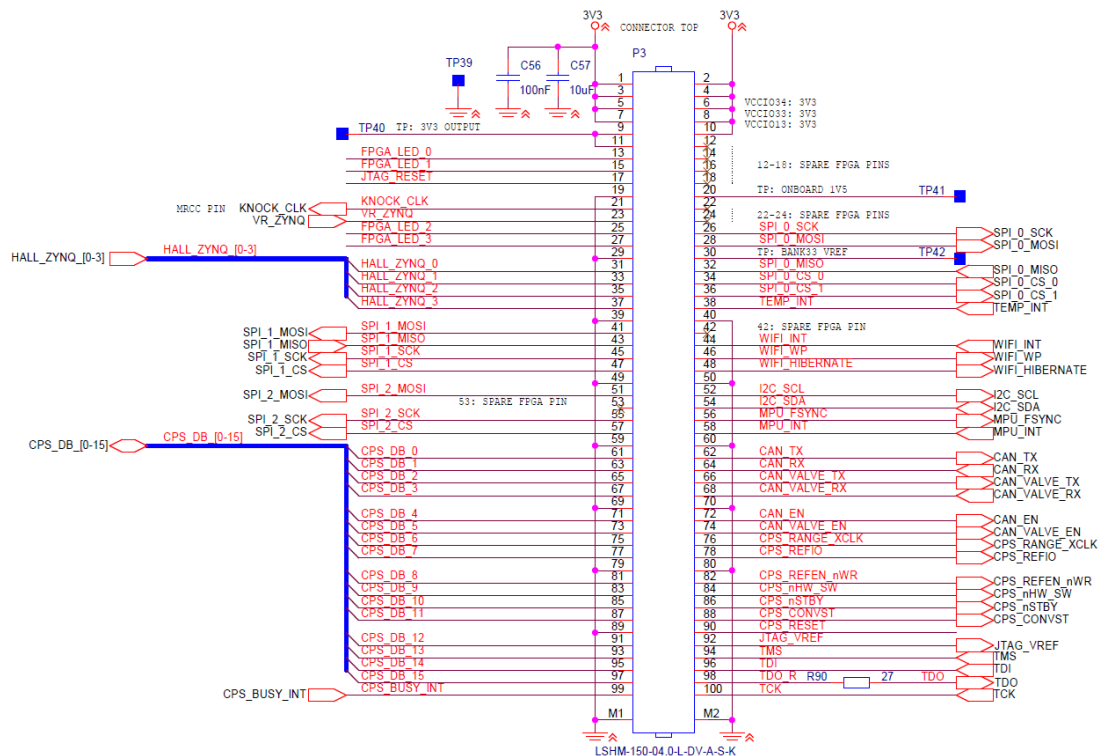
Connector top has FPGA bank 13 and bank 33 pins available. Connector top is the main connector for all internal communication interfaces such as all SPI buses and the I2C bus. The components connected to connector top are:

- Module power input (3.3 V)
- FPGA bank 13, 33 and 44 voltage supply (3.3 V)
- SPI bus 0, 1 and 2
  - Knock sensor interface
  - Temperature sensor interface
  - Wi-Fi module interface
  - Lambda and GPAO DAC
- I2C bus
  - 9-axis Accelerometer, magnetometer and gyroscope MEMS sensor
  - Barometer

<sup>8</sup> Figure from Trenz Electronics TE0720 product page



- CPS ADC interface
- Hall sensor digital inputs
- JTAG interface
- TE0720 reset input
- CAN interface
- Pneumatic Valve Module CAN interface



**Figure 8 - Connector top schematic implementation**

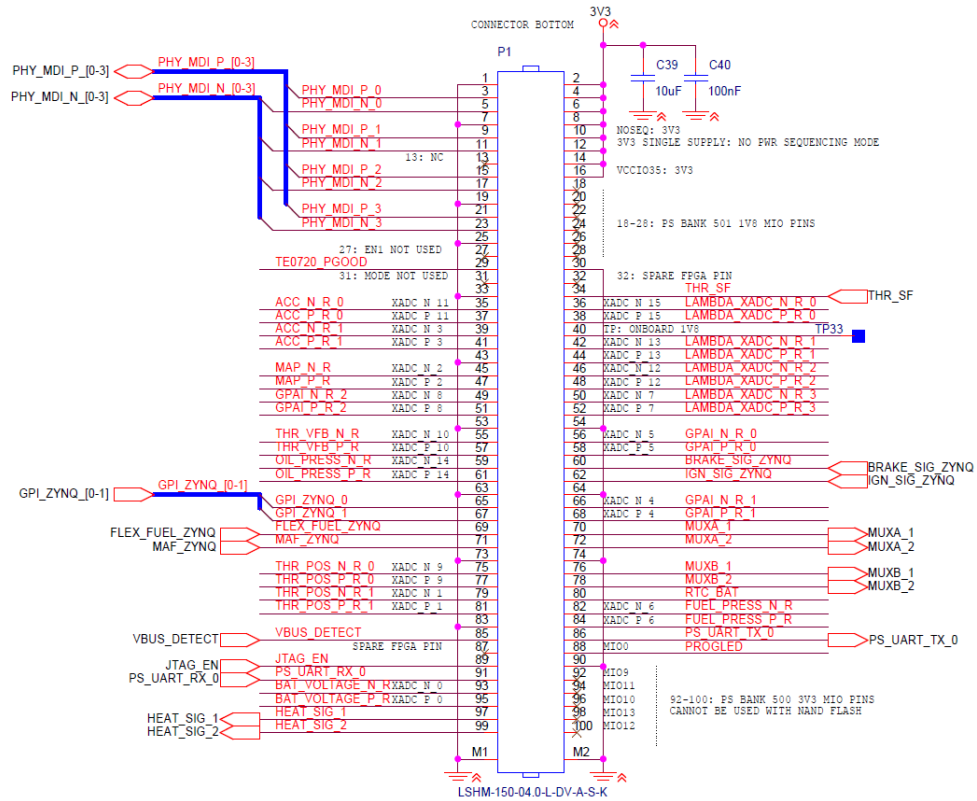
The schematic implementation of connector top is shown in Figure 8. See Table 71 in Appendix V for a complete list of all signals and their connection to connector top.



### 2.1.2. Connector bottom

Connector bottom is connected to many of the ECU analog sensors as the internal XADC is available to this connector. There are also power inputs and other special function pins on this connector. The components connected to connector bottom are:

- Module power input (3.3 V)
- FPGA bank 35 voltage supply (3.3 V)
- RTC battery input
- MIO0 (bank 500) 3.3 V I/O
  - Pin 0: Programmable LED
  - Pin 9-13: Not used
  - Pin 14-15: UART0 (Not usable at the same time as NAND flash)
- MIO1 (bank 501) 1.8 V I/O
  - pin 40-46: not used
- Ethernet PHY
- XADC analog sensor inputs (see 3.7.8 for details)
  - Battery voltage
  - Throttle body position feedback
  - Accelerator pedal position
  - Fuel pressure
  - Lambda sensor
  - Throttle body H-bridge driver feedback
  - Manifold Air Pressure
  - General purpose analog inputs
- Digital sensor inputs
  - Manifold air pressure sensor
  - Flex fuel sensor
  - Brake signal input
  - Ignition signal input
  - General purpose digital inputs
- Throttle body status flag input
- Lambda MUX channel control outputs
- VBUS input signal
- TE0720 PGOOD output (connected to green LED)
- NOSEQ input (pulled hard to 3.3 V)
- JTAG enable input



**Figure 9 - Connector bottom schematic implementation**

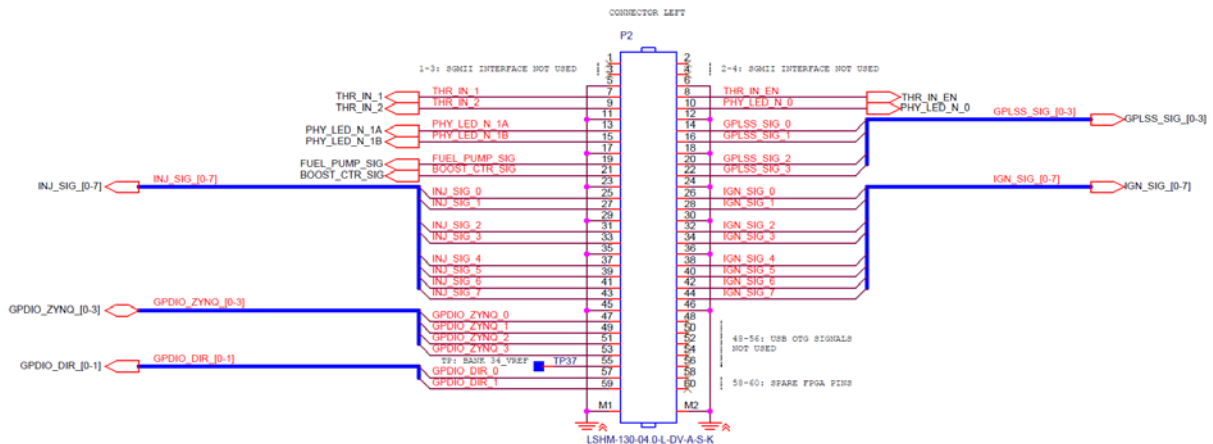
The schematic implementation of connector bottom is shown in Figure 9. See Table 72 in Appendix V for a complete list of all signals and their connection to connector bottom.



### 2.1.3. Connector left

Connector left contains FPGA bank 34 pins and is the smallest of the connectors with 60 pins. The connector is primarily used for driver circuitry control signals and therefore most of the outputs that are controlling low side switching circuits in the ECU. There are no power pins on connector left aside from GND pins. Components connected to connector left are:

- Ethernet SGMII interface (not used)
- USB OTG interface (not used)
- Ethernet PHY RJ45 jack LEDs (must be internally routed to these pins)
- Throttle body H-bridge
- Low side switch controls
  - Boost controller
  - Fuel pump
  - Fuel injectors
  - Ignition coils
  - General purpose low side switches
- General purpose digital I/O



**Figure 10 - Connector left schematic implementation**

The schematic implementation of connector left is shown in Figure 8. See Table 73 in Appendix V for a complete list of all signals and their connection to connector left.



### 3. Sensor interfaces

In order to control the combustion process of a modern internal combustion engine (ICE), we must gather different parameters of the current status of the engine. This is performed using an array of different sensors, all of which data is collected and processed in the ECU to create a response in the regulated system. The sensors collect information such as pressure and temperature in fluids and gases, position and speed of rotating bodies and information about the combustion process itself. All of which is needed in order to ensure an efficient and stable combustion process in the ICE.

This section describes the different sensors and how they are interfaced in hardware. If the ICE is seen as a black box, the sensors can be seen as the *inputs* of the ECU.

#### 3.1. Temperature measurements

Temperature measurements in different parts of the engine of different parameters are extremely important. Information about temperatures in coolant, oil, fuel, air and exhausts are absolutely vital in order to be able to regulate the combustion process. There are different types of sensors that are used for different types of temperature measurements, depending on the range of temperatures to be measured. One of the biggest issues with temperature sensors are nonlinearity. The fact that the sensors all have linearity errors must be compensated for. Some temperature sensor types are almost linear, while others are completely nonlinear. Noise in temperature measurements must also be taken into account, as noisy environments such as a car engine bay can be able to completely destroy the measured signal from the sensor if not properly conditioned.

**Table 2 - Temperature sensor characteristics<sup>9</sup>**

Property	Thermistor	Thermocouple	RTD	IC
Resolution	Very high	Average	High	High
Range	Small	Very broad	Broad	Limited
Output	Nonlinear	Nonlinear	Almost linear	Linear
Accuracy	Very high	Limited	High	Limited

The different characteristics of the different types of temperature sensors used are shown in Table 2. The following text will focus primarily on thermistors and thermocouples as these are the types of temperature sensors that we interface with our ECU.

---

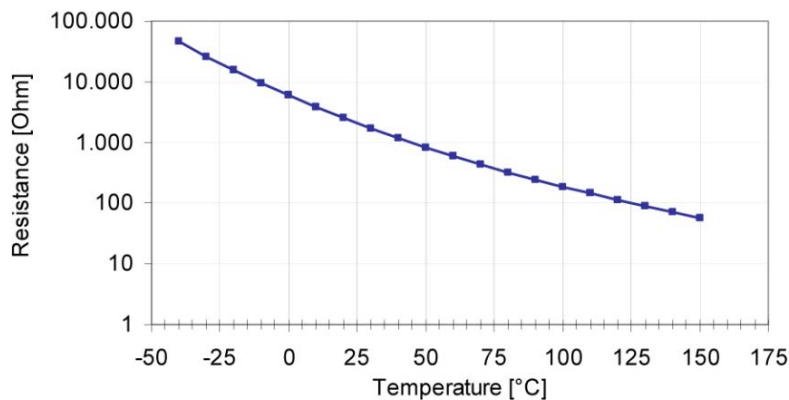
<sup>9</sup> Jounach, M p. 233





### 3.1.1. Thermistor measurements

The thermistor is a very cheap and fairly reliable type of temperature sensor. The thermistor is a resistance based temperature sensor. It only has two leads and can be seen as a resistor with resistance varying with the ambient temperature of the device. There are two main types of thermistors, Positive Temperature Coefficient (PTC) thermistors and Negative Temperature Coefficient (NTC), with the latter most commonly used. Negative temperature coefficient means that the resistance of the thermistor decreases with increasing temperature, as opposed to PTC thermistors where the resistance increases with temperature. There is a wide use of thermistors in engine measurements due to their rugged nature and low price. The nonlinearity of the thermistor is compensated for by using a map of the typical resistance-to-temperature relationship. Such a relationship is shown in Figure 11.



**Figure 11 - NTC thermistor resistance characteristic<sup>10</sup>**

The characteristic in Figure 11 is from Bosch NTC M12-H<sup>11</sup>, which is a thermistor we will interface with our ECU. As the name implies, this is a NTC thermistor. The resistance value of the thermistor drops significantly with temperature. NTC M12-H is typically used for measuring the temperature in liquids such as coolant (water), oil and fuel. The range of this particular thermistor is -40 to 150 °C.

<sup>10</sup> Figure from BOSCH NTC M12-H Datasheet

<sup>11</sup> Bosch NTC M12-H Datasheet



### 3.1.2. Thermocouple measurements

A thermocouple is another inexpensive and reliable way of measuring temperature. A thermocouple works by measuring the voltage difference in a pair of conductors when the conductors are exposed to different temperatures. There are many types of thermocouple types depending on the alloy of the metals in the conductors. A major benefit of thermocouples is the wide range of measurable temperatures, which often range over several hundred degrees Celsius. Type K thermocouples can measure over 1000 °C. Disadvantages of thermocouples are the relatively low accuracy and nonlinearities. Thermocouples require a cold junction measurement to use as reference in order to be able to determine temperature correctly. Originally, the cold junction was a physical ice bath, but today, normally a different type temperature sensor measures the cold junction in order to be able to perform cold junction compensation in software.

We use Bosch TCP K thermocouple probe for measuring exhaust temperature in our application. This is a type K thermocouple capable of measuring a range of -200 to 1300 °C. There will be eight thermocouple inputs on our ECU in order to be able to measure exhaust temperatures of each engine cylinder independently. The reason for monitoring exhaust temperatures of each cylinder is due to possible differences in the cylinders. As fuel functions as a cylinder coolant, cylinders running hotter than others may need increased injector opening times to ensure proper cooling of the cylinder. The wear and tear of the engine may create changes in each cylinder that can be measured and compensated for in the ECU. Individual monitoring of each cylinder temperature also makes it possible to detect and pinpoint faults easier.

### 3.2. Temperature sensor interfacing

We decided to use LTC2983<sup>12</sup> which is a special purpose IC recently released by Linear Technology. LTC2983 is a high accuracy temperature sensor interface IC complete with compensation for the most commonly used types of temperature sensors. In regard to thermocouples, the IC automatically performs cold junction compensation based on an external temperature measurement device, and also linearizes the results.

LTC2983 allows for multiple types of input sensors connected to the different channels simultaneously. There are 20 input channels on the IC, which are multiplexed to three internal 24-bit  $\Delta\Sigma$  ADCs. There is also embedded fault detection circuitry.

Configuration and readouts from the IC is performed using SPI.

---

<sup>12</sup> LTC2983 Datasheet



### 3.2.1. LTC2983 pinout description

This section is a brief description regarding the reason for the components surrounding the LTC2983<sup>13</sup>.

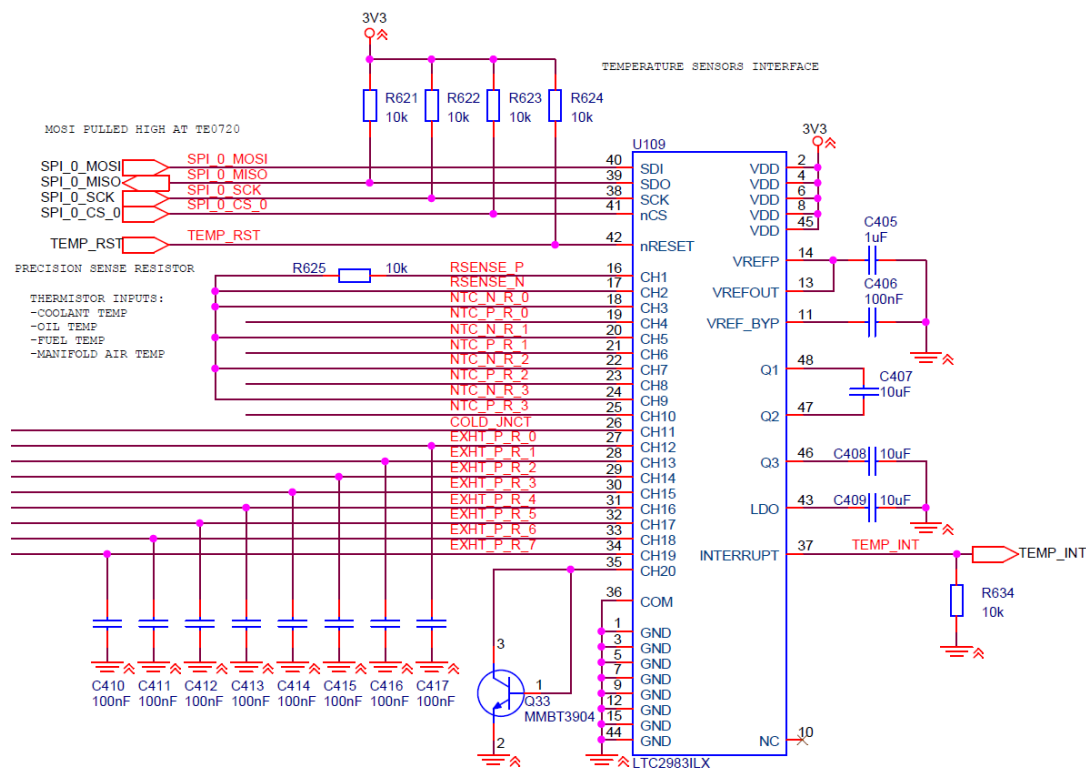


Figure 12 - LTC2983 schematic implementation

Figure 12 shows the schematic with the LTC2983 implemented. The IC itself is powered using 3.3V and the IC supplies all excitation currents required for all temperature sensors. Thermistors require an excitation current in order to create a measurable voltage drop over each sensor, while thermocouples do not.

#### 3.2.1.1. Communication

SPI interface is shown in the top left part of the schematic in Figure 12. LTC2983 supports SPI clock rates of up to 2 MHz. The reset input is an active low, and is therefore pulled up to VDD in order to avoid false triggering. Interrupt pin is low when the device is busy during startup or conversion, and turns high when device is ready. The LTC2983 is connected to SPI\_0 bus, at CS pin 0. SPI\_0 is a shared bus with the TPIC8101 knock sensor interface, which is at CS\_1.

<sup>13</sup> LTC2983 Datasheet



### 3.2.1.2. Decoupling

Each VDD-pin is decoupled with a 100 nF ceramic capacitor (not shown in Figure 12). The VREFP and VREFOUT require an external filtering capacitor selected to be 1  $\mu$ F. Outputs Q1, Q2 and Q3 are external bypass pins for the integrated charge pump. These pins are only for the internal supply. The LDO (Low Drop Out regulator) output is decoupling to ground for the internal LDO.

### 3.2.1.3. Input channels

The LTC2983 has 20 input channels configurable individually for different types of temperature sensors. The COM pin of the package is a common analog input for all channels performing single ended temperature sensor measurements. This pin is connected to ground in our design.

**Table 3 - LTC2983 input channels configuration**

Channel	LTC2983 pin no.	Assignment
1	16	R <sub>SENSE</sub> <sup>+</sup>
2	17	R <sub>SENSE</sub> <sup>-</sup>
3	18	THERMISTOR0-
4	19	THERMISTOR0+
5	20	THERMISTOR1-
6	21	THERMISTOR1+
7	22	THERMISTOR2-
8	23	THERMISTOR2+
9	24	THERMISTOR3-
10	25	THERMISTOR3+
11	26	Cold Junction Diode
12	27	THERMOCOUPLE0+
13	28	THERMOCOUPLE1+
14	29	THERMOCOUPLE2+
15	30	THERMOCOUPLE3+
16	31	THERMOCOUPLE4+
17	32	THERMOCOUPLE5+
18	33	THERMOCOUPLE6+
19	34	THERMOCOUPLE7+
20	35	DIODE0



### 3.2.2. LTC2983 thermistor interface

The thermistor inputs are configured for Bosch NTC-M12-H<sup>14</sup> and Bosch HFM 5<sup>15</sup> thermistors. They are configured for differential inputs all sharing a single precision  $10.0\text{ k}\Omega \pm 0.1\%$   $R_{sense}$  resistor. See Table 3 for LTC2983 pin assignments. When performing software setup, the sense resistor channel pointer must be configured to channel 1 and 2 in accordance to Table 3. The sense resistor must be configured for sharing between all thermistor channels. Auto ranged excitation current of the sense resistor can be used. Refer to the LTC2983 datasheet for detailed description on channel configuration for differential measurement of thermistors.

Note that Bosch motorsport thermistors are special purpose nonstandard thermistors, and their characteristics must be manually programmed to the internal memory of the LTC2983 before use. Refer to NTC-M12-H and HFM 5 datasheets for thermistor resistance characteristics.

The thermistor interface has 100 pF decoupling capacitors on each differential line input as recommended in the LTC2983 datasheet. In addition, the cutoff frequency of the low pass filter can be changed if required by replacing the  $0\text{ }\Omega$  series resistors on the inputs.

All thermistor inputs are ESD protected at 30 kV Air and contact discharge.

---

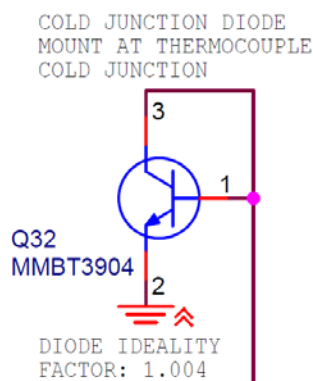
<sup>14</sup> Bosch NTC-M12-H Datasheet

<sup>15</sup> Bosch HFM 5 Datasheet



### 3.2.3. LTC2983 thermocouple interface

The thermocouple inputs are configured in hardware for Bosch TCP-K<sup>16</sup>, but are not limited to this thermocouple. The Bosch TCP-K is a standard K-type thermocouple which is supported directly by the LTC2983. Each thermocouple requires a cold junction measurement in order to perform cold junction compensation, but multiple thermocouples can share a single cold junction input. Referring to Table 3, channel 11 of the LTC2983 is connected to a diode for cold junction measurement. The diode is used for cold junction compensation for all eight thermocouples that are connected to the ECU.



**Figure 13 - Cold junction diode**

The cold junction diode is in reality a BJT transistor. The reason for this is the ideality factor. The ideality factor of a diode is a measurement of how much it deviates from the ideal diode equation, in order to be able to compensate. The reason for using a transistor instead of the diode is that typical 2-terminal diodes have significantly higher ideality factor.<sup>17</sup> The MMBT3904<sup>18</sup> used for diode measurements in our design has an ideality factor of 1.004<sup>19</sup>, which is supported for automatic compensation in the LTC2983.

As seen in Table 3, channels 12 to 19 of the LTC2983 are configured for thermocouple inputs. These inputs are single ended measurements with all thermocouples connected to COM ground. For use with Bosch TCP-K, all these input channels are configured to a K-type thermocouple. The cold junction channel pointer must address the cold junction diode located at channel 11.

<sup>16</sup> Bosch TCP-K Datasheet

<sup>17</sup> LTC2997 Datasheet

<sup>18</sup> MMBT3904 Datasheet

<sup>19</sup> LTC2997 Datasheet



The protection RC-network on each thermocouple input also acts as a low pass filter. The values for these components were selected to 1 kΩ resistor and 100 nF capacitor.

The equation for the cut-off frequency in a RC low pass filter is given by (4):

$$f_c = \frac{1}{2\pi RC} \quad (4)$$

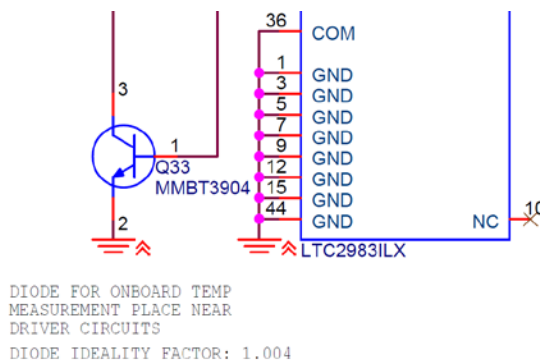
Inserting values into (5) gives:

$$f_c = \frac{1}{2\pi \cdot 1k\Omega \cdot 100nF} = 1591.55 \text{ Hz} \quad (5)$$

All thermocouple inputs are ESD protected at 30 kV Air and contact discharge.

### 3.2.4. LTC2983 diode interface

The spare channel on the LTC2983 not used for temperature sensors are connected to a transistor for measuring diode temperature in the same way the cold junction diode works as described in 3.2.3.



**Figure 14 - Diode configuration**

The transistor is configured in the same way as the cold junction diode. The transistor is a small SOT-23 package and should be placed at a location on the circuit board where ambient temperature is measured. Such a location would typically be close to temperature sensitive circuits. Measuring the ambient temperature in the enclosed ECU can be important in order to perform critical shutdown if overheating. As with the cold junction diode, the ideality factor of the diode must be programmed into the LTC2983s memory in order to compensate correctly.

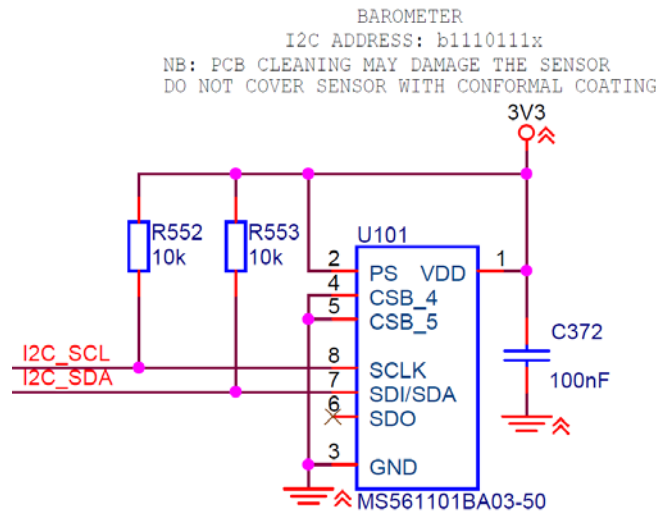


### 3.3. Barometric air pressure measurements

The ECU is required to know the air density in order to establish how much fuel that has to be injected into the engine. The combustion process is carefully regulated by constantly monitoring the air density in order to calculate the optimal amount for fuel that must be injected.

#### 3.3.1. MS5611-01BA03-50 On-board barometer

We selected MS5611-01BA03-50<sup>20</sup> barometer from Measurement Specialties Inc. for measuring the air pressure from an on-board sensor. The sensor has a small footprint and is surface mounted. The sensor is configured and communicated with I2C bus.



**Figure 15 - Barometer implementation**

The barometer is a small 8-pin package. The schematic implementation of the barometer is shown in Figure 15. The SCLK and SDA pins are for the I2C interface. The Protocol Select (PS) pin is pulled high in order to activate the I2C interface mode, as the MS5611-01BA03-50 supports both SPI and I2C interfaces. The I2C bus is a shared bus with the MPU-9150 motion processing unit. The I2C interface on the MS5611-01BA03-50 is in itself very simple with few configuration commands, and all calculations must be performed by the Zynq-7000 in our ECU. The CSB\_4 and CSB\_5 pins are internally connected and sets the 7<sup>th</sup> bit in the I2C address. The address bit is the complimented value of CSB\_4 and CSB\_5. The pins are pulled low, hence the 7<sup>th</sup> address bit in the I2C address is 1. The I2C address is b1110111x. The last bit in the I2C address, denoted *x* is set to 0 for write and 1 for read commands. The *b* prefix designates that it is a binary value.

<sup>20</sup> MS5611-01BA03-50 Datasheet





Handling of the sensor must be performed with care. Electronics for use in harsh environments such as an ECU typically has a conformal coating. Conformal coating of the barometer may render it unusable as it uses the surrounding air for monitoring pressure. In this regard, the ECU cannot be completely hermetically sealed, as the pressure of the inside of the ECU must equal the outside air pressure. PCB cleaning solutions may also harm the sensor. The barometer should be covered before cleaning and applying conformal coating to the PCB.

### 3.4. Position and orientation measurements

It is important to be able to monitor the ECUs orientation at any time. Especially in motorsport applications, the risk of the car to roll over is very high. If the ECU detects rollover or collision, it can cut fuel supply and engine power. It is also important to prevent accidental start of the engine if the car is flipped upside down.

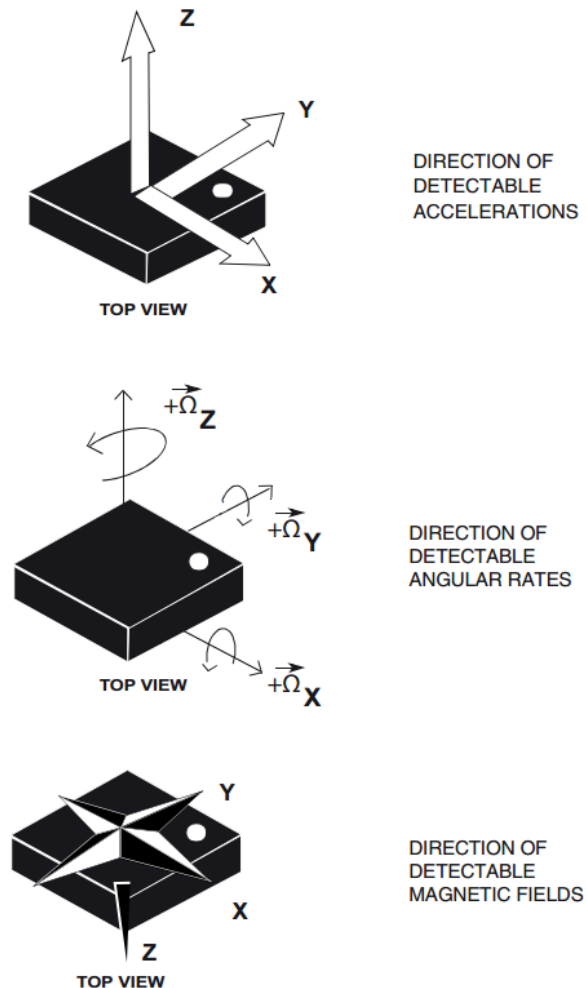
There are two main ways this is done in today's systems. Both rely on MEMS (Micromechanical Systems). One method uses a tilt-switch which is a device integrated in semiconductor packages. The device is in principle a simple switch triggered by gravitational forces.

The second way is to use an accelerometer. The accelerometer measures acceleration in several axes. For use as a tilt sensor, it most importantly measures acceleration due to gravitational forces. Accelerometers come typically in small QFN or LGA PCB packages making them easy to place on high density circuit boards such as in handheld devices. They also consume little energy, making them suitable in battery critical applications.

The tilt-switch is mechanically the most reliable option; however, these are components typically larger than accelerometers, come in strange packages and are relatively expensive. They are also much less configurable in the way they function. Bosch, alongside with other corporations, manufacture both tilt sensors and accelerometers for automotive use. Due to part availability, price and configurability, an accelerometer was selected the best option.

#### 3.4.1. MPU-9150 Motion Processing Unit (MPU)

There were several choices to make when selecting which accelerometer to use. The MPU-9150 from Invensense was selected. Invensense is a leading manufacturer in MEMS sensors for small applications. The MPU-9150 is Motion Processing Unit (MPU), meaning that it implements accelerometer, gyroscope and magnetometer in a single package.



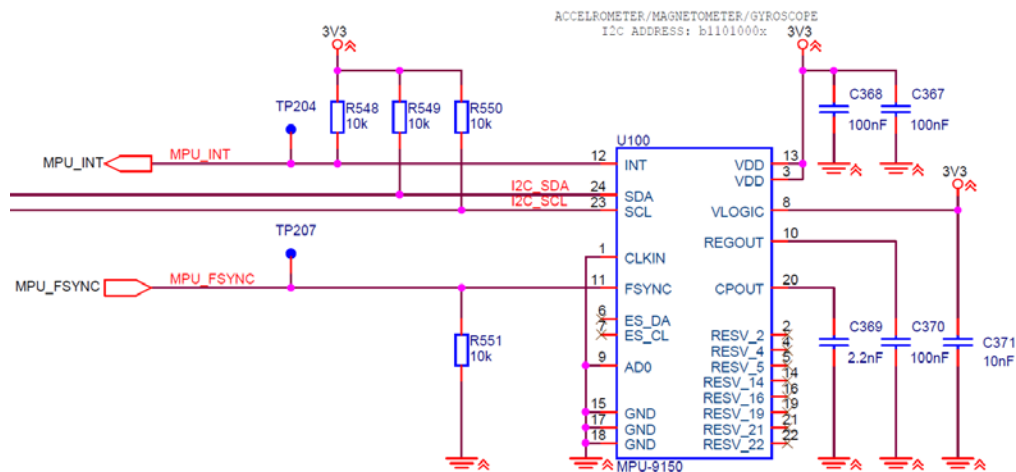
**Figure 16 - Inertial module measurements<sup>21</sup>**

The MPU-9150 is able to measure acceleration, angular rates and magnetic fields for a total of 9 degrees of freedom described in Figure 16. The gyroscope and magnetometer does not serve critical functions in the ECU in the same order as the accelerometer. The reason for why they are implemented is intended for possible future use for enhancing the user experience of the ECU. As the ECU will be an interactive device with Wi-Fi communication for handheld devices and computers, the inertial module can be used to display interesting information such as compass (magnetometer), and angular measurements for example in steep hills or corners of a race track. If combined with GPS, it can also be used for accurate positioning and navigation systems.

<sup>21</sup> Figure from LSM9DS0 Datasheet



### 3.4.2. MPU-9151 schematic implementation



### Figure 17 - MPU-9150 schematic implementation

Figure 17 shows the schematic implementation of the MPU-9150. The MPU-9150 communicates using an I2C interface. The AD0 pin sets the 7<sup>th</sup> bit in the I2C address. The address bit is the value of AD0. The pin is pulled low, hence the 7<sup>th</sup> address bit in the I2C address is 0. The I2C address is b1101000x. The last bit in the I2C address, represented by  $x$  in the address is set to 0 for write and 1 for read commands. The  $b$  prefix describes that it is a binary value.

### Table 4 - MPU-9150 pins and signal description

Pin	Description
INT	Interrupt output
SDA	Serial Data (I2C)
SCL	Serial Clock (I2C)
CLKIN	External clock input (Not used)
FSYNC	Frame synchronization input
ES_DA	Auxiliary I2C Serial Data (Not used)
ES_CL	Auxiliary I2C Serial Clock (Not used)
AD0	I2C address bit
VLOGIC	Interface logic voltage supply
REGOUT	Internal regulator capacitor connection
CPOUT	Charge pump capacitor connection
RESV_x	Reserved pins (NC)
VDD	Power supply

Table 4 shows pin descriptions of signal pins in the MPU-9150. 2.2 nF capacitor coupled to CPOUT is for the internal charge pump. The 100 nF on REGOUT is for the internal voltage regulator. VLOGIC and VDD pins are decoupled according to datasheet recommendations. The IC supports different VLOGIC voltages for compatibility with different I2C bus levels. The I2C bus in the KPEC ECU is at 3.3 V logic level.



### 3.5. Flywheel sensor interface

To be able to measure the speed of the crankshaft we are using an inductive speed sensor. To interface the inductive sensor, we are using a specialized IC from Maxim Integrated that is made for interfacing inductive sensors (or variable reluctance sensor). The sensor we are interfacing is giving out a sinusoidal signal with a max voltage of 200 mVp-p. In Bosch datasheet, it is stated that the sensor consist of a bar magnet with a soft magnetic pole pin with two connections, and the operation of the sensors is when a ferromagnetic ring gear turns past the sensor a voltage is generated. The voltage amplitude and frequency varies with the motors RPM. Since the output of the sensor is a sinusoidal, to be able to use any of the information, we have to transform sinusoidal to a square wave signal, using a comparator with hysteresis.

The IC we are using (MAX9924UAUB<sup>22</sup>) have 4 different modes and are seen in Table 5.

**Table 5 - MAX9924 operating modes**

OPERATING MODE	SETTING		DEVICE FUNCTIONALITY		
	ZERO_EN	INT_THRS	ZERO CROSSING	ADAPTIVE PEAK THRESHOLD	BIAS VOLTAGE SOURCE
<b>A1</b>	VCC	VCC	ENABLED	ENABLED	EXTERNAL
<b>A2</b>	GND	GND	ENABLED	ENABLED	INTERNAL REF
<b>B</b>	VCC	GND	ENABLED	DISABLED	EXTERNAL
<b>C</b>	GND	VCC	DISABLED	DISABLED	EXTERNAL

The modes are set by applying different voltages to the pins ZERO\_EN and INT\_THRS. With each mode different functionalities are enabled and disabled.

#### 3.5.1. Zero crossing

The zero-crossing signal corresponds to missing tooth in the crankshaft and is used to determine the position of the crankshaft. The signal from the differential is level-shifted to the bias voltage, this means that the zero crossing signal is simply the bias voltage.

#### 3.5.2. Adaptive peak threshold

In modes A1 and A2, adaptive peak threshold is enabled. Adaptive peak threshold is the upper trigger lever for the comparator and gives a new upper trigger level per cycle. The threshold voltage is 1/3 of the peak of the previous cycle. If the amplitude of the signal voltage rises, so does the threshold voltage, and vice versa for a decrease in amplitude of the signal voltage. Inside the IC there is a built in watchdog that drops the threshold voltage to a default minimum threshold, this is done if the amplitude of the signal voltage remains lower than the adaptive peak threshold for more than 85 ms.

<sup>22</sup> MAX9924UAUB datasheet



### 3.5.3. Bias voltage source

The bias reference voltage is used reference all internal electronics, and should be set to  $V_{CC}/2$  using an external resistor-divider circuit. The minimum threshold, adaptive upper threshold and zero crossing signals is all referenced to the bias voltage. In mode A1, C and B the bias voltage is set by external components in a resistor-divider. In mode A2, the bias is set internally to 2.5 V, this reduces external components. The buffer in the IC eliminates the loading effect of the resistors. The bias voltage should be bypassed with a 10  $\mu$ F and 100 nF capacitors in parallel; this is to ensure a smooth and stable bias voltage.

### 3.5.4. Mode selection

We have chosen to implement two modes, A2 and C. We choose A2 because this is the most “plug-and-play” mode. It does not require any more components, because the bias voltage at 2.5 V is set internally and EXT is not connected. The adaptive peak threshold and zero-crossing signal is enabled, which allows us to interface with many different kinds of sensors. This is because of the adaptive peak threshold and zero crossing. This means that *a//DNM* components are not mounted.

When changing to mode C, the EXT needs to be applied with a voltage. The EXT pin gives us the hysteresis values for the upper and lower trigger levels of the comparator.

As a default configuration we have connected the IC as mode A2, this is because we want to interface the variety of sensors that are out there. We have also made it available to connect the IC as mode C; this allows more control of the parameters. These parameters are bias voltage and hysteresis trigger levels. Figure 18 shows the schematic connection of modes A2 and C.

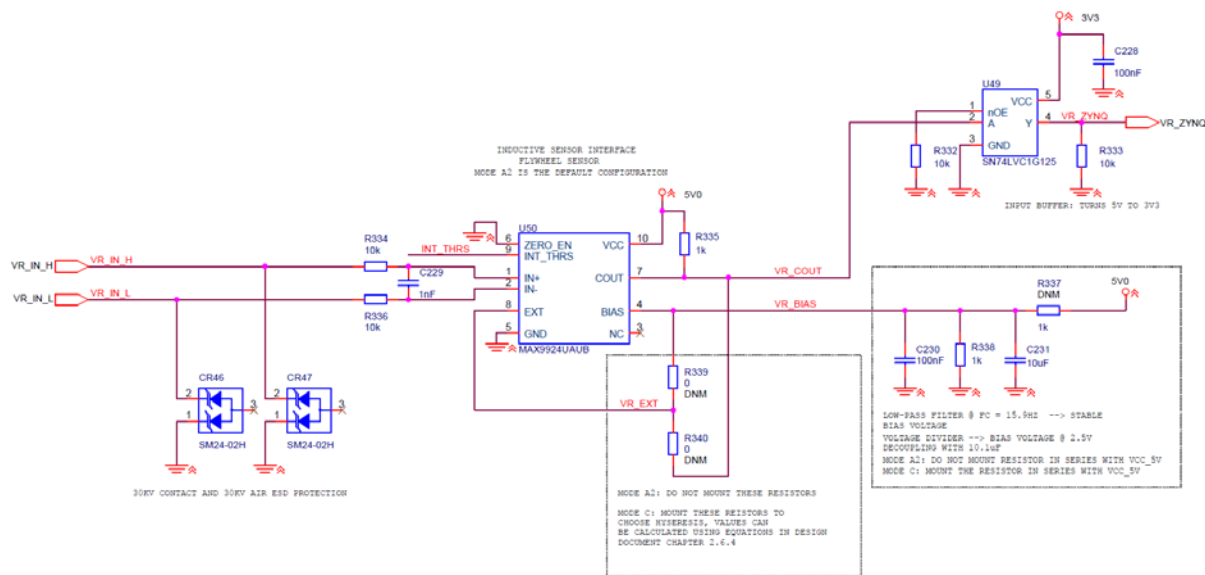
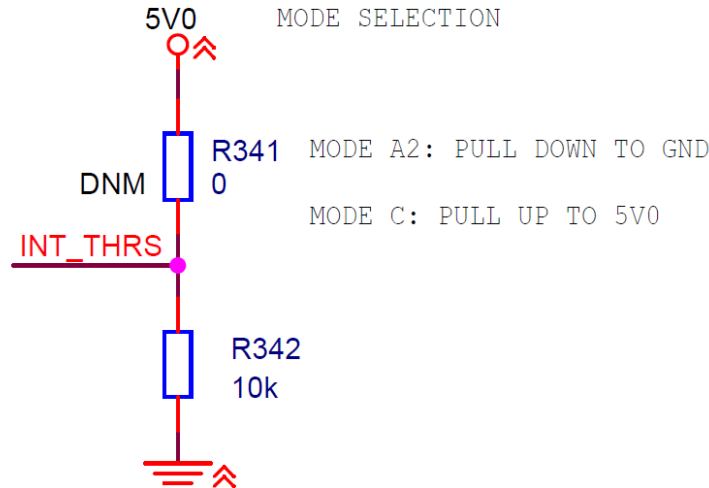


Figure 18 - MAX9924 Mode A2/C

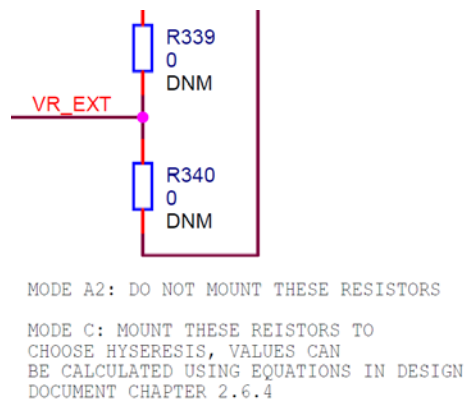


Since the default configuration in the schematic is set to A2, to be able to change mode to C, there are some minor changes that need to be implemented. First of all the 0  $\Omega$  resistor between INT\_THRS and GND must be soldered off, the 0  $\Omega$  resistor between INT\_THRS and 5V0 must be mounted, see Figure 19.



**Figure 19 - Mode selection GND/VCC**

When changing to mode C, the EXT pin needs to be applied with a voltage via a voltage divider. The EXT pin gives us the hysteresis values for the upper and lower trigger levels of the comparator. The equations for these levels are given in (6) and (7). Additionally the EXT pin must be connected between the voltage divider, seen in Figure 18. The two resistors must be soldered on as seen in Figure 20.



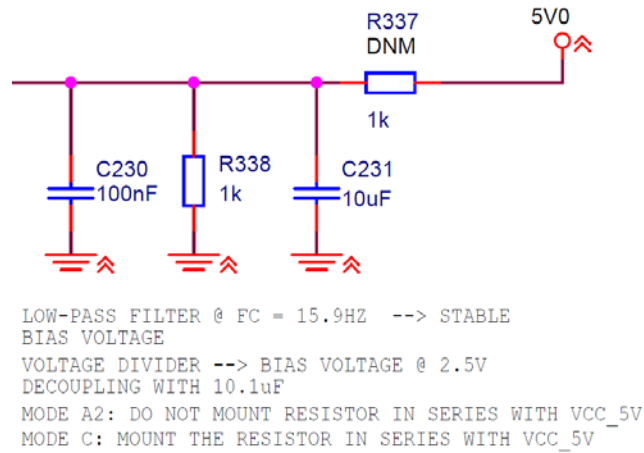
**Figure 20 - EXT voltage divider**

$$V_{TU} = \left( \frac{R_1(V_{PULLUP} - V_{BIAS})R_2}{R_1 + R_2 + R_{PULLUP}} \right) + V_{BIAS} \quad (6)$$



$$V_{TL} = \frac{R_2}{R_1 + R_2} \cdot V_{BIAS} \quad (7)$$

The last thing to do before completing the change from mode A2 to mode C is to solder on the 1 kΩ resistor that is in series with 5V0 seen in Figure 21.

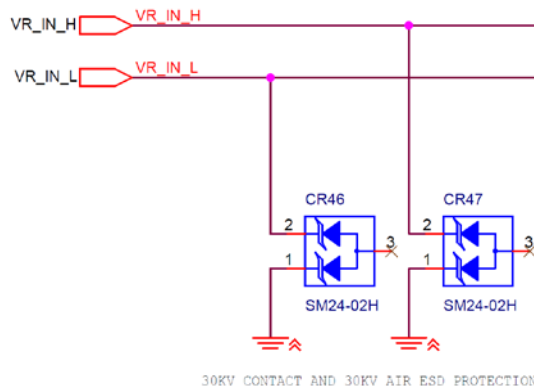


**Figure 21 - Bias voltage**

When the 1 kΩ resistor that is DNM is not connected, the 1 kΩ resistor (in parallel with 10 μF and 100 nF) pulls down the VBIAS pin to ground, thus enabling mode A2. The DNM resistor that is in series with 5V0 and the 10 μF makes a low-pass filter which smooth's and stabilizes the voltage bias at a cut-off frequency of 15.9 Hz.

### 3.5.5. ESD protection

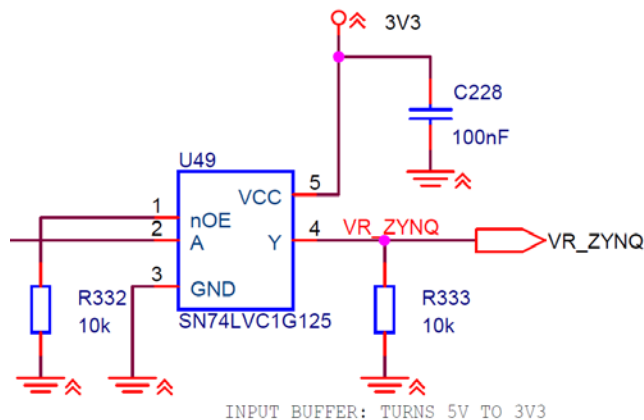
The input of the circuitry is protected with a TVS which will protect the rest of the circuitry from ESD. The TVS protects from transients and is rated for 30 kV contact and 30 kV air. Figure 22 shows the schematic on how the ESD protection was implemented.



**Figure 22 - ESD protection on input**

### 3.5.6. Output voltage

The output (COUT) gives out digital signal with amplitude from 0 - 5 V; we need to level-shift this to 3.3 V, because our ECU can't handle 5 V. We did this by using an input buffer that takes 5 V and level-shifts it to 3.3 V. The implementation is seen in Figure 23.



**Figure 23 - Voltage level-shifting**



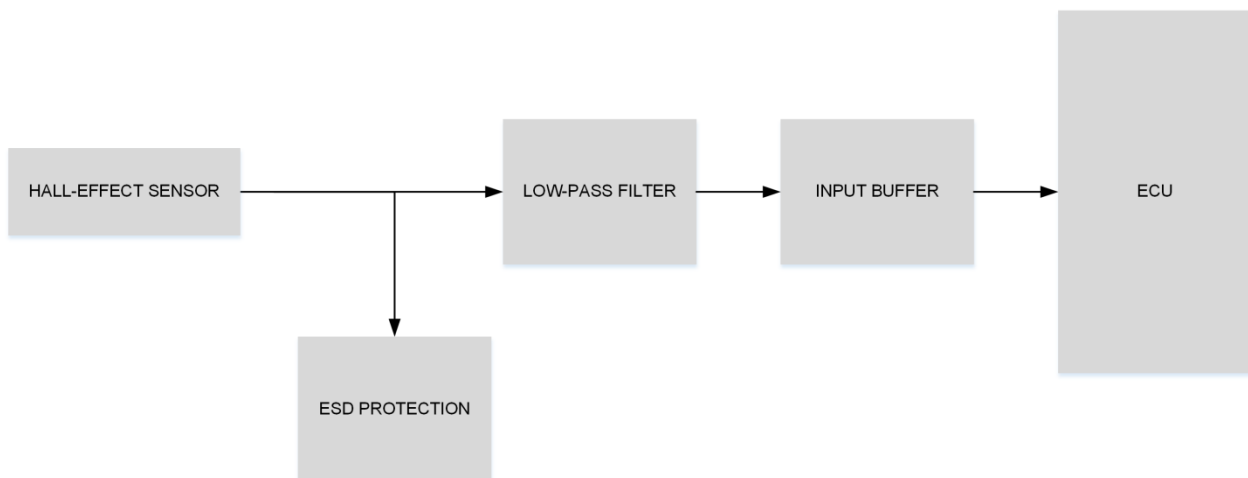


### 3.6. Camshaft sensor interface

During the combustion process the valves moves up and down. The process is as following; when the valve is pushed down by the camshaft, air and fuel mixture enters the combustion chamber. The valves are spring loaded, and go up as the camshaft no longer forces them open. Right before Top Dead Centre (TDC) the mixture is ignited and the piston in the combustion chamber is pushed down. When the piston is pushed up again by the crankshaft, exhausts from the combustion are fed out into the exhaust system. Then, the process is repeated. The camshaft is what pushes the valves up and it is important that we know the position and speed of this camshaft, so that we can perform timing of the combustion process.

We are interfacing sensors from Bosch Motorsport, and we used the Mini-HA-P as reference when designing the interface. The HA-P is sensor is built on the principles of the Hall Effect. The Hall Effect principle is that a voltage is produced when a metallic material is entering and cutting the magnetic field made by the Hall Effect sensor. We are using this sensor to determine the position and speed of the crankshaft, by installing a Hall Effect sensor close to the camshaft. In our application we have 4 of these. The way it works is that the camshaft has a number of tracks with different sizes, and when these tracks are cutting the magnetic field produced by the Hall Effect sensor, we get a digital signal, which is either *high* or *low*. By measuring the time and pattern of these HIGH pulses we can determine the frequency and position of the camshaft.

Since the output signal of our sensor is digital, this means we can input these signal directly into our ECU, after some signal processing has been done. A simplified block diagram shows the system in Figure 24.



**Figure 24 - Simplified block diagram of the system**



### 3.6.1. ESD protection

To protect all signal lines that goes out of the ECU from ESD, we are using the DRTR5V0U4TS<sup>23</sup>, which is a 4 channel ESD protection chip, which is rated to 25 kV contact and 30 kV Air and is powered from a 5 V supply that is provided by the ECU. Figure 25 shows how the input signal is protected from ESD in the schematic.

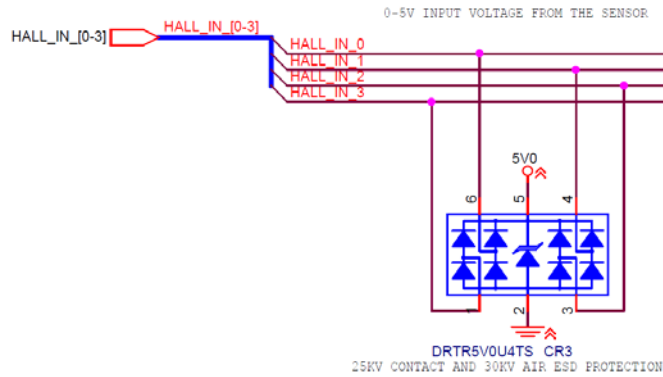


Figure 25 - ESD Protection

### 3.6.2. Low-pass filtering

The maximum frequency of our Hall-effect sensor is 10 kHz, and since we want information from 0 - 10 kHz we are filtering everything above this maximum frequency. We have implemented a first order RC filter with a cut-off frequency at 13.2 kHz. We have the cut-off frequency at 13.2 kHz because if we put it at 10 kHz we will have some attenuation at 10 kHz, this is due to that at the cut-off frequency the attenuation is already -3 dB or 0.707 of the amplitude. We don't want to lose any information at 10 kHz and that's why we moved the cut-off frequency to 13.2 kHz.

Figure 26 shows how this was implemented in schematics.

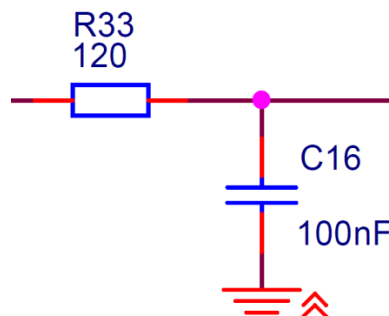


Figure 26 - Low-pass filter implementation

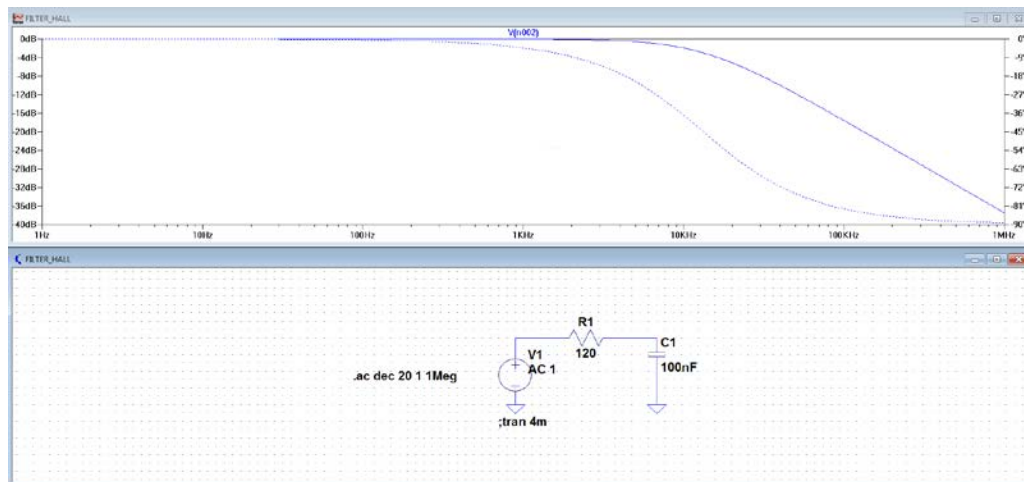
<sup>23</sup> DRTR5V0U4TS



A simulation was performed to see if the RC filter did what it was made for. LTSpice is used to perform the simulation. By using a capacitor of 100 nF and 120  $\Omega$  we then calculated the cut-off frequency using (8).

$$f_c = \frac{1}{2\pi RC} \quad (8)$$

We determined these values by choosing existing capacitor values already used in the schematics and then we calculated to the nearest E12 resistor value. The connection and simulation is seen in Figure 27.



**Figure 27 - Filter simulation and connection**

The simulation parameters are as following. We are using a voltage generator with AC amplitude of 5 V. A simulation of AC analysis with decade as type of sweep, 20 numbers of points per decade, start frequency at 1 and stop frequency at 1 GHz. The conclusion with this simulation is that the cut-off frequency is at 13.2 kHz and at 10 kHz we have attenuation of 0.8, which means we get 4 V out when the amplitude is 5 V. To be able to see if the filter is able to follow a transient input into the filter we have simulated this and can be seen in Figure 28, where the red line represents the unit step that changes from a low state to a high state. The pink line represents the filter trying to follow the unit step.

From the simulation figure we can see that the filter is following the unit step all the time, except for when transition from low to high, but it quickly converges towards the low and high states.

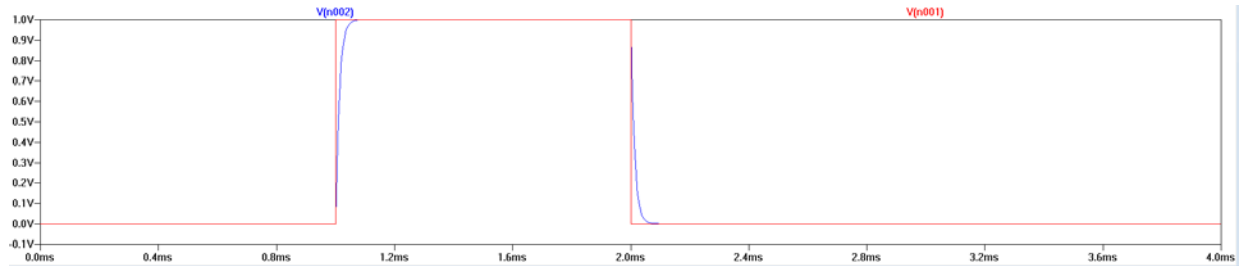


Figure 28 - Unit step on filter

### 3.6.3. Input buffer

The ECU can only handle a maximum voltage of 3.3 V on the input. Since the Hall Effect sensor outputs a 0 - 5 V we have to level shift this voltage to 3.3 V. This is done by the input buffer.

The entire schematic is seen in Figure 29, and low pass filter implementation in Figure 30

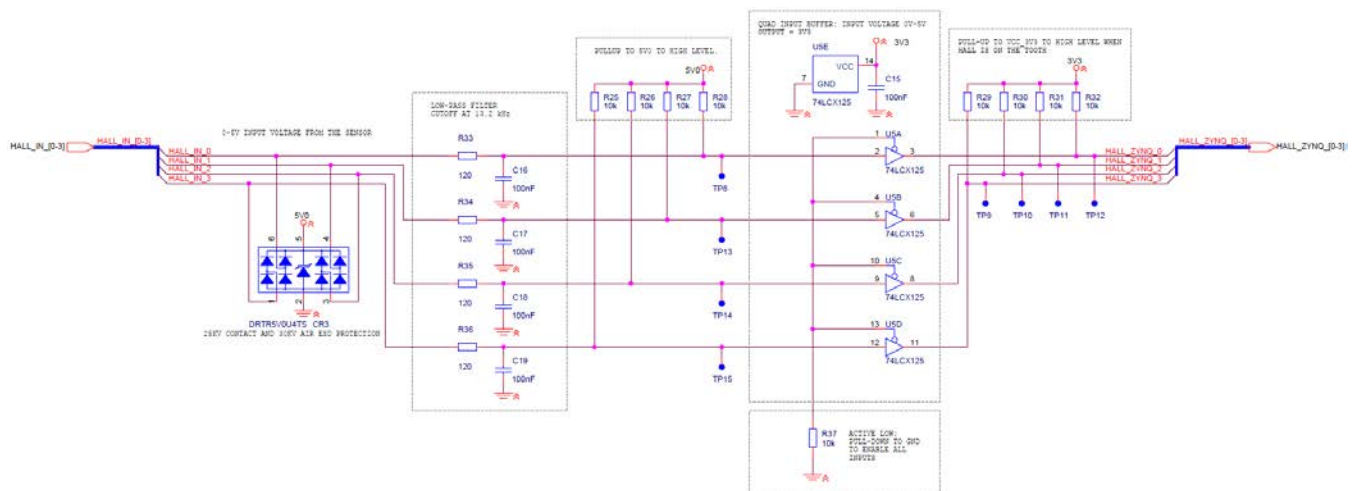


Figure 29 - Schematic connections

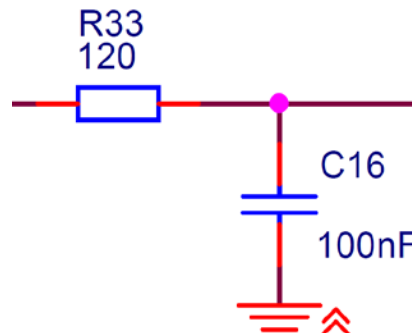


Figure 30 - Low-pass filter camshaft



### 3.7. Analog data acquisition

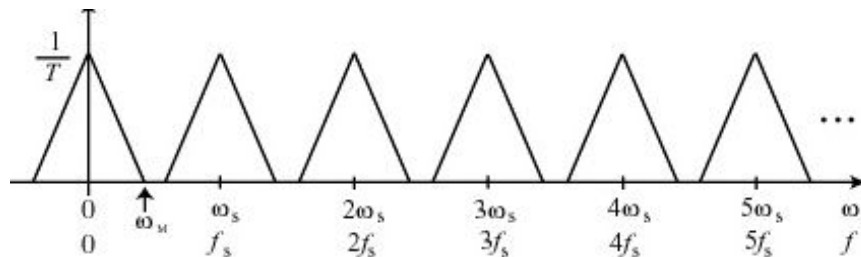
This section will discuss sensors that output a DC voltage in the range of 0 - 5 V. To be able to interface these kinds of sensors, we need to take the analog values and convert them to digitally represented values using an ADC so that the ECU can process the data. Three individual ADCs will be necessary to interface all our sensors, where one is internally implemented on the Micromodule and the others will be implemented on the motherboard. We will see it's not a simple matter of only sending analog quantities into an ADC, before that some filtration is needed.

Chapter 3.7.1 and 3.7.2 discusses the theoretical part of sampling theorems. The remaining chapters discuss solutions for sensors that is rated for analog output voltages in the range of 0 - 5 V.

#### 3.7.1. Nyquist sampling theorem

Choosing the right CPS (Cylinder Pressure Sensor) ADC (Analog-to-Digital Converter) will need a strategic approach as there are some concerns that must be taken care of.

The Nyquist theorem states that the sampling frequency of the ADC should be  $2f_s > \omega_m$  where  $f_s$  is the sampling frequency, and  $\omega_m$  is the maximum frequency component of the measuring device.

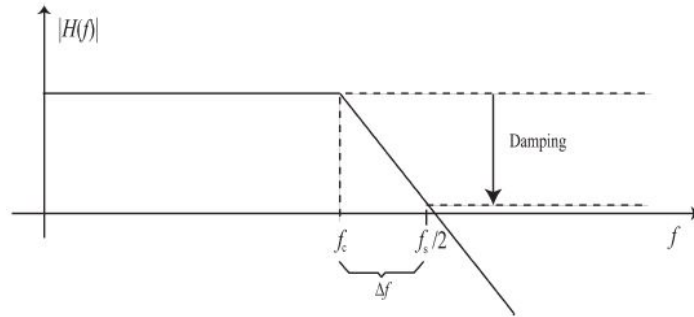


**Figure 31 - Amplitude spectrum of a sampled signal<sup>24</sup>**

We are considering using the Optrand optical spark plug pressure sensor, where the frequency response is 1 - 30 kHz. That implies that a minimum sampling frequency of 60 kHz should be implemented, the categorization of ADCs makes the 96 kHz sampling ADC sufficient to meet the Nyquist theorem, see Figure 31.

The sampling theorem states that; for a time-continuous signal to be reconstructed, all frequency components will need to be less than  $\frac{f_s}{2}$ .

<sup>24</sup> Wikimedia Project, Sampling

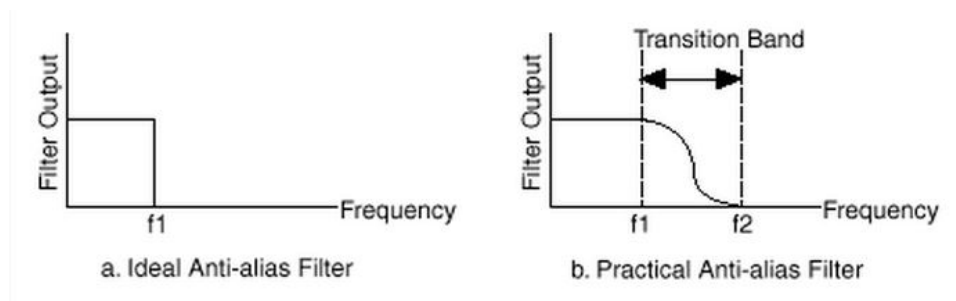


**Figure 32 - Antialiasing with use of a filter<sup>25</sup>**

The resolution of the ADC is directly related to the filter design. To avoid aliasing the measured signal must be dampened below the noise-level of the ADC, which translates to a dampening factor of  $20 \log 2^{-n} @ \frac{f_s}{2}$  where  $n$  is the bit resolution of the ADC. This is illustrated in Figure 32.

### 3.7.2. Oversampling

To avoid aliasing of a signal, an analog anti-aliasing filter is added before the ADC. The purpose is to meet the Nyquist criterion and the noise level requirement as discussed. An ideal filter will cut off all the unwanted frequencies at exactly the desired frequencies, but unfortunately the world is not ideal. We will need to oversample, to move the boundaries of the Nyquist criterion upwards in the frequency spectrum.



**Figure 33 - Practical use of the anti-aliasing filter<sup>26</sup>**

As seen by Figure 33, half the Nyquist frequency ( $\frac{f_s}{2}$ ) representing  $f_2$  is shifted upwards to allow sampling without aliasing. Another feature of oversampling is that it alters the requirement of the filter, when sampling just above the Nyquist criterion the filter will require a high roll-off rate. As when the signal is oversampled, a first or second order filter may be sufficient.

<sup>25</sup> S. Gudvangen, Introduction to quantisation

<sup>26</sup> National instruments, What are Anti-Aliasing Filter and Why are They Used?



For our application and specifically interfacing the CPS sensor with the most adequate ADC can be challenging. Based on the discussed knowledge, time budget, and bitrate analysis we wish to narrow down our field of interest.

### 3.7.3. Time budget

Based on a study of an application report<sup>27</sup>, we discovered that a time budget were a reasonable approach to detect the processing boundaries implicit in the requirements.

The following calculations were made to determine the time frame at hand, given that an 8 cylinder engine should be able to run at speeds up to 16.000 RPM.

$$\frac{Min}{16.000 \text{ rev}} \cdot \frac{rev}{360 \text{ CA}} \cdot \frac{60 \text{ s}}{min} = 10.41 \frac{\mu s}{CA} \quad (9)$$

Which gives us the time frame of *each* Crank Angle (CA) as the engine speed is 16 kRPM.

$$\frac{60^\circ \text{ CA}}{\text{knock window}} \cdot \frac{10.41 \mu s}{CA} = 625 \frac{\mu s}{\text{knock window}} \quad (10)$$

Gives us the time frame of the knock window with a 16 kRPM engine speed, given that knock will only occur during a 60° window 10 - 70° CA after TDC.

Another perspective of the cylinder pressure processing is that it will need to be shifted from the time domain with use of fast Fourier transform (FFT) into the frequency domain, where the knock frequencies will appear at certain modes. What all of this is implying is that we will be restricted to sequential processing, as the digital signal processing (DSP) is done in the 800 MHz dual-core ARM® Cortex-A9 part of the Zynq SoC. Since the Zynq-7000 breaks the digital signal processing bottleneck with software acceleration, it enables the processor to perform the FFT 45% faster with implementation of ARM NEON library. The result is that it can process 4K2K Ultra-HDTV applications, and surely will handle our 8 channel FFT without any constraints.

<sup>27</sup> Engine knock detection using spectral analysis techniques with a TMS320



### 3.7.4. Cylinder Pressure interface

Cylinder pressure is an important factor in the combustion process, and by monitoring the pressure inside the cylinders an optimal engine control can be achieved. The cylinder pressure information is used to determine exhaust gas recirculation, air/fuel mixture and to determine and eliminate knocking. Furthermore by determining where the maximum peak pressure point inside the cylinder is, this information can then be used to determine the timing of the ignition for max or economic fuel efficiency.

We are interfacing a pressure sensor from Optrand<sup>28</sup> which is integrated inside a spark plug. The sensor is built up from multiple optical fibers which are positioned in front of a flexing metal and the pressure that is induced is proportional to the intensity of the reflected light. The output voltage of the sensor is 0.5 – 4.5 V.

In today's market these kinds of pressure measurements is not used to regulate the combustion process in an engine. The main reasons for this are that this is really expensive, where prices for a single spark plug with integrated pressure sensor can cost up to 900 US dollars, with a modification (custom made spark plugs for your engine) fee of 250 US dollars<sup>29</sup>, and the implementations that has been done has only been done under safe environment in laboratories.

Since the ECU only handles digital signals meaning that an ADC must be used to convert the analog quantities over to the digital domain. Furthermore to avoid aliasing, an anti-aliasing filter is added before the ADC. Chapter 3.7.4.1 and 3.7.4.2 discusses the ADC that is used, chapter onwards discuss the implementation of this ADC, while chapter 3.7.4.4 discuss the anti-aliasing filter.

#### 3.7.4.1. ADC resolution

The next step is to determine the resolution of the ADC, we know the output voltage of the transducer and should choose a resolution above the industrial norm (10-bit) and well below the boundaries of the hardware (1.25 Gbps), leaving the software with margin to process the data.

The signal we wish to measure has a bandwidth of 0 – 30 kHz and could be measured with the Optrand Calplug transducer. The Optrand transducer has given a Sensitivity of 1.51 mV/psi and a range of 0 - 3000 psi (200 bar).

One aspect of choosing the ADC requirements is to take into consideration the effective resolution of the ADC. Ideally we would have full N of bits available, to process something meaningful, let's take an example;

The transducer output voltage is 0.5 - 5 V

---

<sup>28</sup> Optrand pressure sensors

<sup>29</sup> Raph Vlodarczyk, Optrand inc.





$$Range = (5 - 0.5)V = 4.5 V \quad (11)$$

$$\Delta = \frac{Range}{n} = \frac{4.5 V}{2^{16}} = 68 \frac{\mu V}{step} \quad (12)$$

Now in the ideal world, we would have 68  $\mu V$ /step of the 16-bit ADC, without considering constraints inherent in the ADC, LSB step-size, signal amplitude. Different manufacturers are providing information about the Signal-to-Noise-Ratio (SNR) in dB, which is a measure of how small a signal the ADC can resolve from the sensor. Ideally this would result in  $20 \log 2^{16} = 96 \text{ dB}$  for the 16-bit converter.

In reality we need to take into consideration the constraints as mentioned, the ADC manufacturers provide this data in either of two ways; Signal to Noise And Distortion (SINAD)<sup>30</sup> which is the ratio of the wanted signal to the distortion, and the Effective number of bits (ENOB)<sup>31</sup> which as the name suggest is a measure of the effective number of bits, after the constraints has been withdrawn, and can be calculated;

$$ENOB = \frac{SINAD - 1.76}{6.02} \quad (13)$$

Where

$$SINAD = \frac{rms \text{ SIGNAL}}{rms \text{ NOISE}} \quad (14)$$

Let's do another example to illustrate the effect;

The ADS8568, SAR, 8 ch, 510 kSPS, simultaneous sampling ADC has given a SNR of 91.5 dB.

SINAD is given to be 90 dB that suggest that this ADC has excellent AC performance. Let's calculate the Effective number of bits;

$$ENOB = \frac{90 \text{ dB} - 1.76}{6.02} = 14.65 \text{ bit} \quad (15)$$

---

<sup>30</sup> Understanding SINAD

<sup>31</sup> Understanding ENOB



This may not sound much of a difference, but a quick calculation below may put this in perspective.

$$2^{16} = 65536 \quad (16)$$

$$2^{14.65} = 25709 \quad (17)$$

$$2^{16} - 2^{14.65} = 39826 \text{ steps} \quad (18)$$

60% of the original step resolution has vanished in circuitry constraints.

If we now go back to the Optrand transducer, we can collect information about sensitivity and pressure range:

Optrand CALplug<sup>32</sup>:

Sensitivity: 1.51 mV/step

Pressure range: 0 - 200 bar (about 3000 psi)

Let's check to see if the 16-bit ADC, with distortion will have sufficient step resolution to detect each level-shift of the transducer:

$$\frac{2^{14.65} \text{ step}}{3000 \text{ psi}} = 8.57 \text{ step/psi} \quad (19)$$

So far so good, by the looks of it there is margin to resolve the transducer pressure range.

The resolution of the 16-bit ADC sounds beneficial, let's find out if our hardware can handle such data rates within our time frame.

To calculate the amount of data processed/time unit we will need to determine a plausible sampling rate. We know from conventional engine management systems that one sample/degree is sufficient to maintain regulation, but our implementation of CPS increases the requirements for resolution. From external resources<sup>33</sup> 800 samples per period is assumed extreme precision, so let's see which sample rates that might be applicable.

First let's find the frequency of our engine at maximum RPM

<sup>32</sup> CALplug Datasheet

<sup>33</sup> John errington's data conversion website



$$\frac{16.000 \text{ rev/min}}{60 \text{ sec}} = 267 \text{ Hz} = 267 \text{ rev/sec} \quad (20)$$

If we test a couple sample rates we can evaluate their results.

$$f_s = 192 \text{ KHz}$$

$$\frac{192 \text{ K}}{267} = 719 \text{ Samples/rev} \quad (21)$$

$$\frac{719 \text{ Samples/rev}}{360^\circ} = 2 \text{ Sampels/degree} \quad (22)$$

Already better than conventional engine management systems, let's try next level ADC sampling rate.

$$f_s = 510 \text{ kHz}$$

$$\frac{510 \text{ k}}{267} = 1910 \text{ Sampels/rev} \quad (23)$$

$$\frac{1910 \text{ S/r}}{360^\circ} = 5.3 \text{ Sampels/degree} \quad (24)$$

So we have found a couple of sampling rates that might be usable, given that we can process their data within our time budget and implement hardware that doesn't cause too many constraints.

The amount of data that the Zynq is able to handle is listed to be in the area of 1.25 Gbit/s, translated into the time budget of our 8 cylinder engine it means:

$$1.25 \text{ Gbit/s} = 1250 \text{ Mbit/s} \quad (25)$$

The bitrate with the given resolution and two different sampling rates is as follows<sup>34</sup>:

---

<sup>34</sup> Bitrate A/D conversion



$$r_b = \frac{n}{T} = \frac{16}{(1/192k)} = 3 \text{ Mbit/s} \quad (26)$$

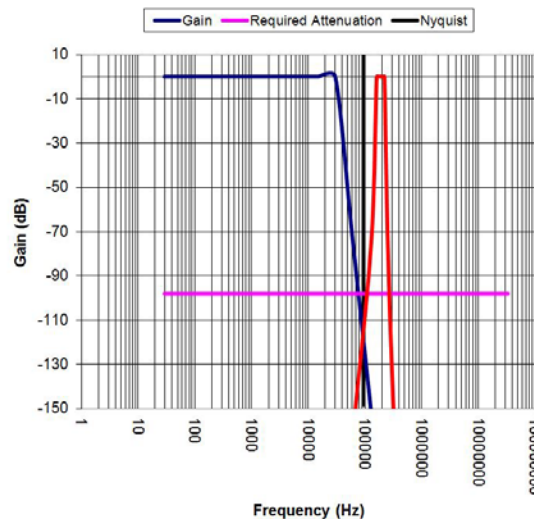
$$r_b = \frac{n}{T} = \frac{16}{(1/510k)} = 8.16 \text{ Mbit/s} \quad (27)$$

With a 16-bit ADC, and a sampling rate of up to 510 kHz we have verified that the bitrate is well below what the Zynq is capable of. Then we know that the oversampling should not cause any process constraints on the system.

In addition we are well above the Nyquist sampling criterion, giving us a minimum sampling frequency of 60 kHz, twice the frequency response of the transducer (0 – 30 kHz) but also leaving us space to add a filter design that has a moderate roll-off rate.

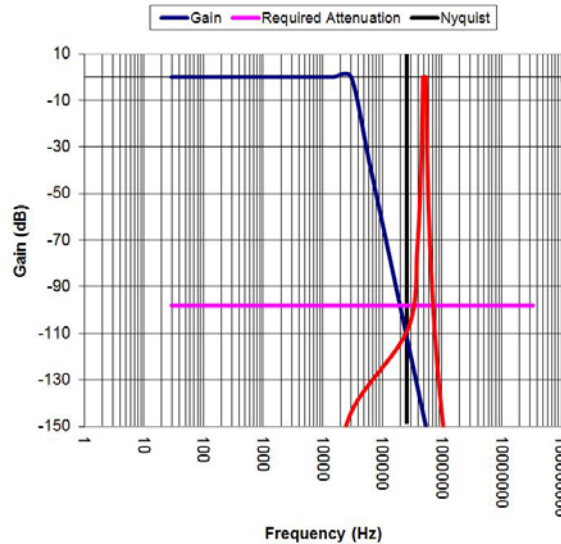
We could see that both sampling rates will cause sufficient sampling resolution with respectively 719 and 1910 samples/rev in worst case scenario at 16.000 RPM. So by the looks of it, they will both do the job, but if we choose the 510 kHz ADC we are given some bandwidth on filter design.

This is confirmed by anti-aliasing simulation in Figure 34 and Figure 35. In order to meet Nyquist criterion with a 192 kHz sampling frequency (Figure 34), it will require a quite steep 12<sup>th</sup> order filter.



**Figure 34 - 16-bit,  $F_s = 192 \text{ kHz}$ , 12<sup>th</sup> order filter**

Oversampling at 510 kHz (Figure 35) requires half the roll-off rate by implementing a 6<sup>th</sup> order filter. We can verify that we will indeed have the sufficient damping at half the sampling frequency which Nyquist formulated.



**Figure 35 - 16-bit,  $F_s = 510$  kHz, 6<sup>th</sup> order filter**

#### 3.7.4.2. **Choosing ADC**

Most general semiconductor dealers offer the possibility to perform a parametric search where KPEC prefers a part supplier called Digi-Key. They are offering a variety of electronics from many of the leading manufacturers; along with a search engine that makes it easy to navigate and assess their specifications, availability and price.

There are more than 16 400 different ADCs in the Digi-Key catalogue, so it could be useful to narrow the window a bit. We are searching for 16-bit ADCs with  $\geq 500$  kSPS and 8 channel input. This shortens the list down to 34.

Based on the accessible information, tools and features chose to go for Texas instruments, their solutions are clever, the availability is good, the communication is done by SPI or parallel which is necessary at such data rates, they support our temperature requirements and they can handle parallel sampling. On the down side it will require a split rail converter as their solutions requires both positive and negative voltages.

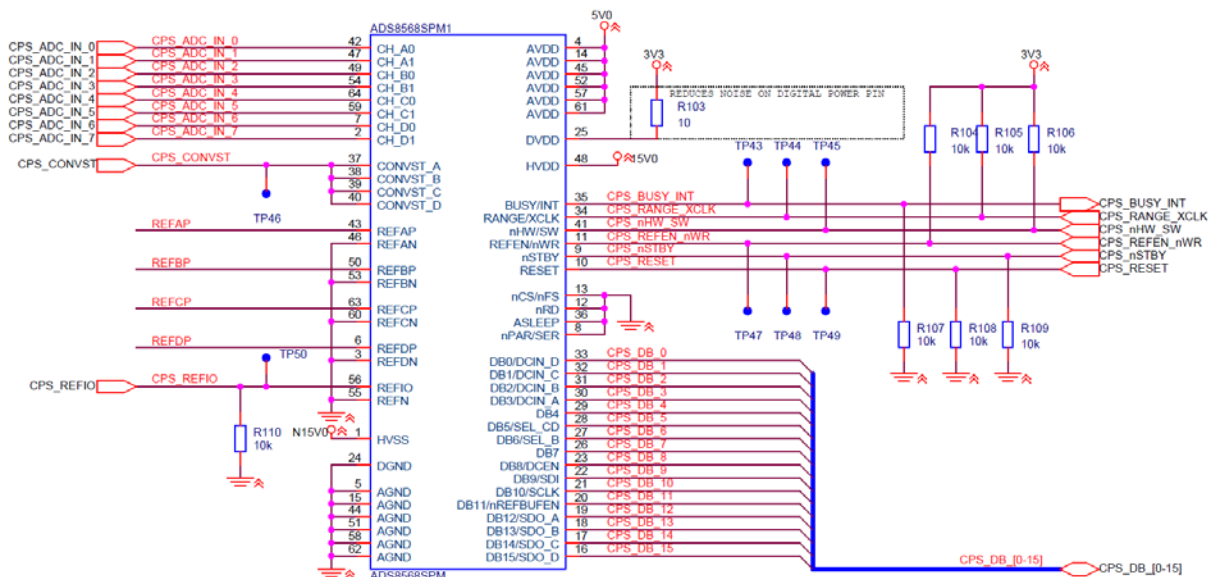
The specific ADC that will meet our requirements is available in several different device-packages and packaging. To make the circuitry serviceable in the future, we prefer the QFP package above the permanent QFN packages, making them difficult to replace.

The ADC for the cylinder pressure measurement application is the Texas Instruments ADS8568SPM.



### 3.7.4.3. Cylinder pressure ADC configuration

This chapter is a technical discussion of the ADS8568<sup>35</sup> ADC configuration; this part will only address the hardware implementation. Some features of this ADC can be implemented through hardware or software configuration. KPEC has chosen to reduce some of the configuration features as the purpose for this ADC is given through CPS (Cylinder Pressure Sensor) data acquisition.



**Figure 36 - Schematic overview of ADS8568**

The schematic overview of the ADC itself can be seen in Figure 36, where there are some hardware decisions that will need to be accounted for. There are a total of 64 pins on this package where 8 of them are Analog Input (AI) from the CPS sensors, 16 of them are Digital Output (DO) which represents the 16 bits of the ADC, where port 16 is the Most Significant Bit (MSB) and port 33 is the Least Significant Bit (LSB). Depending of the configuration and mode the ADC is operated in, most of these ports have added functionality, and this is a clever way to save I/O on the interfacing controller.

<sup>35</sup> ADS8568 Datasheet



### 3.7.4.3.1. Power

Firstly the ADC will need to be powered up with a total of 4 different sources; this is because the internal circuitry on the ADC consists of separate parts like analog data acquisition, internal I/O buffers for digital interface and the high voltage supplies to drive the analog input circuitry. Though the ADC theoretically can be driven until the maximum power rating, it's never good practice; we have listed the power requirements in Table 6 which is the functional operation range.

**Table 6 - Power operational range**

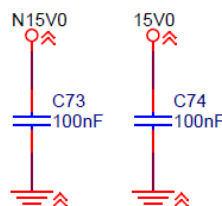
	Min. (V)	Typ. (V)	Max. (V)
<b>AVDD</b>	4.5	5	5.5
<b>DVDD</b>	2.7	3.3	5.5
<b>HVDD</b>	5	15	16.5
<b>HVSS</b>	-16.5	-15	-5

The typical values are favourable, and will be the target for the power design. The datasheet<sup>36</sup> has specified that a linear regulator is recommended to generate the analog supply voltage. Texas Instruments (TI) has also suggested a split rail converter to generate the bipolar supplies.

### 3.7.4.3.2. Power decoupling

Decoupling is necessary in circuit design, and especially on the power terminals. Potential ripple in the power lines can alter the performance of the circuitry. The problem is known and is often handled with the use of one or more decoupling capacitors. The following power decoupling implementations have been made;

- A ceramic bypass cap of 1  $\mu$ F is placed in between pin number 24 and 25 of the digital power input.
- Each of the six analog supply pins uses a ceramic decoupling capacitor of 1  $\mu$ F
- The high voltage analog power decoupling capacitors are 100 nF between pin number 1, 48 and GND. Where a schematic implementation is shown in Figure 37.



**Figure 37 - Decoupling bipolar high voltage**

<sup>36</sup> ADS8568 Datasheet



#### 3.7.4.3.3. Configuration

The ADC has an extended array of functionalities that are accounted for in the datasheet, most of the features can be selected either in the hardware or software. Where possible we have chosen to preselect some features, which will be permanent in hardware, the main objective is to reduce the amount of control settings as there is a fixed amount of I/O that the controller can support.

To support our 0.5 - 5 V CPS analog signal, the ADC is capable of adjusting its input boundaries all the way up to  $\pm 12$  V. VREF can be chosen to be either 2.5 V/3 V in the software configuration (Config bit C13), furthermore the ADC can be configured to support analog input signals in the range  $\pm 2V_{REF}$  or  $\pm 4V_{REF}$ .

If  $V_{REF} = 3$  V and configured in the  $\pm 2V_{REF}$  mode, the ADC is able to support  $\pm 6V_{REF}$  which should be sufficient in handling the input signal of 0.5 - 5 V.

The  $\pm 2V_{REF}$  mode can be selected with controller input on the RANGE/XCLK and nHW/SW pins with reference to the datasheet<sup>37</sup>.

#### 3.7.4.3.4. Communication

The ADC can be operated in either serial or parallel communication with the controller, to support the full data acquisition performance the ADC will need to be operated in parallel mode.

Pin 8 nPAR/SER is preselected *low* to allow parallel interface with the controller.

To enable the parallel output pins, pin 12 nRD is preselected *low*.

The chip select pin 13 nCS/nFS shall also be held *low* to enable the parallel interface.

If the device is running at low data rates in battery applications, there is a feature to reduce the power consumption of the ADC, our application are none of those things and will be operated in normal mode.

Pin 36 ASLEEP is preselected *low* to allow the ADC to run in normal mode.

The schematic configuration is illustrated in Figure 38 and the pins preselected are listed in Table 7.

---

<sup>37</sup> ADS8568 Datasheet



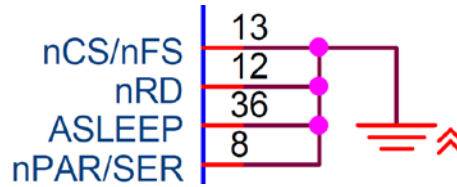


Figure 38 - Schematic view of the hardware configuration

Table 7 - Preselected hardware configuration

Pin number	Name	Type	State
8	$\overline{PAR/SER}$	DI	Low
12	$\overline{RD}$	DI/DI	Low
13	$\overline{CS/FS}$	DI/DI	Low
36	ASLEEP	DI	Low

#### 3.7.4.3.5. Ref decoupling

To allow input voltage ranges on the analog channels, as sensors may have different voltage ratings, the ADC is equipped with a configurable VREF followed by a DAC that allows tuning of the REF voltage down to 2.92 mV/step. The buffered DAC output is provided at REFIO pin, which should be decoupled with 100 nF (minimum), but a 470 nF capacitor is recommended for best performance.

To reduce the BOM, and make the design more viable we decided not to implement the 470 nF capacitor, but instead use 5 capacitors in parallel to generate the equivalent of 500 nF with the use 100 nF capacitors. The schematic implementation is illustrated in Figure 39.

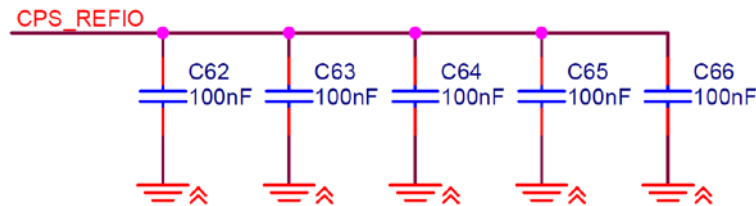
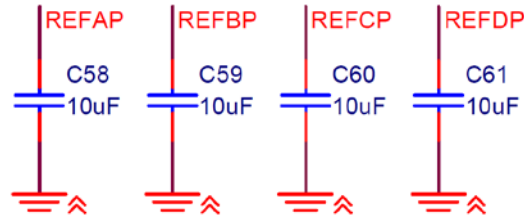


Figure 39 - REFIO decoupling equivalent of 500 nF

The REFIO is then distributed to each of the eight ADC stages, where the signal is buffered through a voltage follower. The buffered reference signal will again need to be decoupled through the REFA, REFB, REFC, REFD pins with a 10  $\mu$ F capacitor. The schematic decoupling is illustrated in Figure 40.

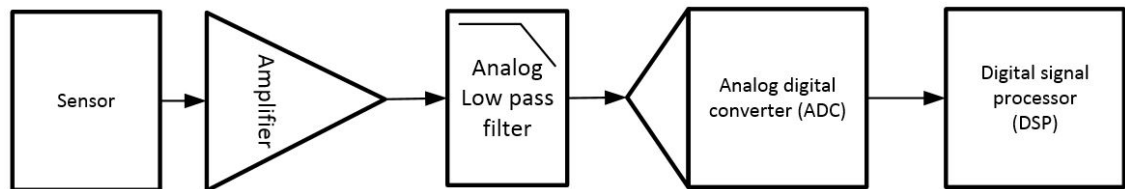


**Figure 40 - Decoupling REFX pins**

#### 3.7.4.4. Filter implementation

In general it's common to implement a low-pass filter in the circuitry when interfacing a sensor; sensors may have different output signals from DC to AC or PWM and thus altering the filter requirements.

The low-pass filtering can either be done by an analog filter before the ADC, or in digital signal processing where various active filters like the analog Bessel, Butterworth and Chebyshev are available, but also more complicated signal processing like FIR-filters and FFT can be implemented. Figure 41 illustrates the principals of signal acquisition from various sensors. Some filters like the CPS filter will be applicable for a more thorough evaluation, while DC-signal temperature sensors are less demanding.



**Figure 41 - Signal path**

We have made a decision and chosen an ADC for our peak pressure point detection. To avoid anti-aliasing, and optimize the signal processing we will need to match the transducer output with the input of the ADC. In terms of physical applications, this means that we will need a low-pass filter and possibly an amplifier.

Based on our knowledge from signal processing, design tools, transducer and ADC datasheets; we have established some filter requirements.

- The bandwidth of the transducer is 0 - 30 kHz.
- The ADC sampling rate is 500 kHz and the resolution is 16 bits.

So the cut-off frequency for our filter should be 30 kHz.



The attenuation at half the Nyquist frequency should be  $20 \log 2^{-16} = -96 \text{ dB}$ , calculated based on the ADC resolution of  $n$  bits.

And the stopband should be less than half the Nyquist frequency of 250 kHz, so we are choosing 240 kHz as the stopband frequency of our filter design.

With help from the TI designer tool, we have entered the filter requirements and it immediately suggested a number of possible filter designs.

Solutions: ( 7 found )										
Select	Filter Response	Color	Order	Max Q	Att (dB)	Passband Ripple (dB)	Group Delay (usec)	Group Delay Flatness (usec)	Settling Time (usec)	Step Response Overshoot (%)
Select	0.2dB Chebyshev	Green	5	3.707	-101.08	0.276	43.615	24.331	211.269	14.48
Select	Butterworth	Blue	6	1.93	-108.16	0.031	21.778	1.274	129.657	14.24
Select	Transitional Gaussian to 6dB	Magenta	6	3.45	-106.67	0.543	19.772	0.388	120.049	5.15
Select	Linear Phase 0.05°	Black	7	2.02	-100.41	0.516	16.948	0.146	49.399	0.00
Select	Linear Phase 0.5°	Green	7	2.66	-103.41	0.616	17.379	0.722	62.645	0.00
Select	Transitional Gaussian to 12dB	Yellow	7	3.73	-108.06	0.570	18.060	0.546	73.672	0.15
Select	Bessel	Red	8	1.23	-98.77	0.514	17.050	0.069	36.421	2.50

Figure 42 - Filter solutions

As seen in Figure 42 they are listed by filter order, where the least order filters often are favourable due to physical circuitry, but also signal conditioning where we want the signal to be as little affected by the filter as possible.

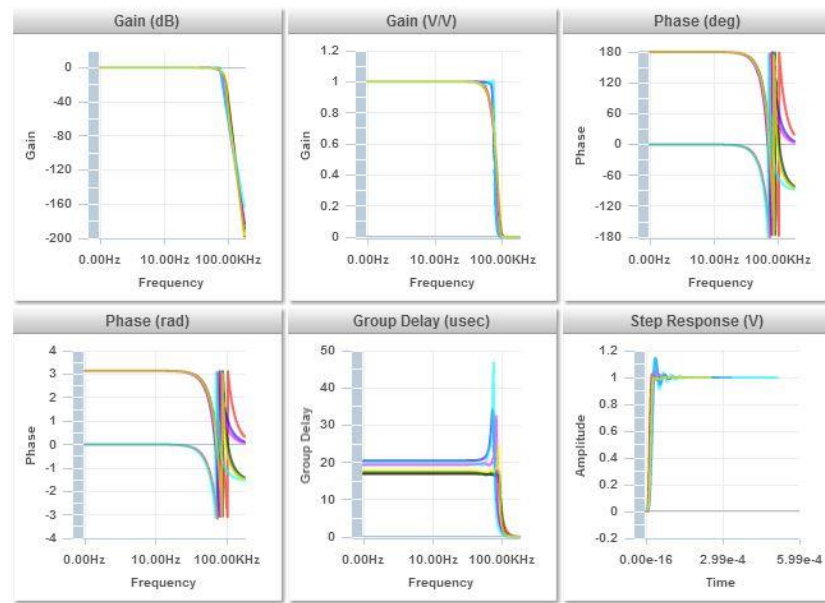


Figure 43 - Filter comparison



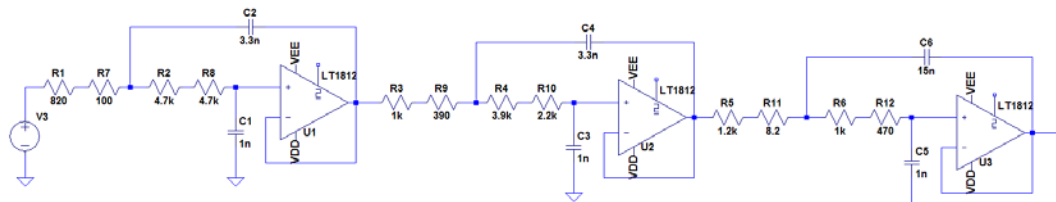
Figure 43 illustrates the filters in comparison to each other; reference to the colour coding can be seen in Figure 42. The filter response in Figure 43 is illustrative, as we can compare the signalling through each filter, and make a decision based on filter characteristic.

Through the Gain plot, we can see that all the filters are very stable; they do not alter the gain through the frequency of interest (0 - 30 kHz) which is critical. We are dependent on a fixed gain into the ADC, due to resolution and linearity-error.

Further down there is the phase response plot, where the designer can choose from either a linear or a phase shifting filter. It is not required to phase shift the output of the transducer, so a regular filter is sufficient.

And there is the step-response plot of the filters; some of them are more agile than others. We can see from the time scale, that they are all responding good through a really small time-window, but the Chebyshev filter for instance is causing far more overshoot than its opponents.

Based on this data, we are going for the Butterworth filter that has a stable gain, is non-phase shifted and has an agile step response, along with its predicted 6-order design. The filter is also properly attenuated at 240 kHz by -101 dB, which is more than the requirement of -96 dB.



**Figure 44 - 6th order Butterworth filter simulation**

In order to achieve the circuit in Figure 44, there is a process of computing all the component values. Texas instruments has developed an application report on low-pass filter design<sup>38</sup>, with help of six equations and a table, the discrete components can be calculated with the equations below.

$$R_1 = mR \quad (28)$$

$$R_2 = R \quad (29)$$

<sup>38</sup> Active low-pass Filter Design

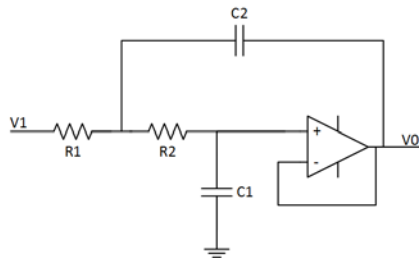


$$C_1 = C \quad (30)$$

$$C_2 = nC \quad (31)$$

$$f_c = \frac{1}{2\pi RC\sqrt{mn}} \quad (32)$$

$$Q = \frac{\sqrt{mn}}{m+1} \quad (33)$$



**Figure 45 - Butterworth filter design with Sallen-key topology**

The filter is built in stages of the circuit given in Figure 45, where the specifications for the filter are known. Table 8 is used to determine Q values for the individual stages in the filter design.

**Table 8 - Butterworth filter table**

Filter order	Stage 1 (Q)	Stage 2 (Q)	Stage 3 (Q)
2	0.7071		
3	1.000		
4	0.5412	1.3065	
5	0.6180	1.6181	
6	0.5177	0.7071	1.9320

Stage 1 of the CPS filter is calculated as follows, where the capacitors in (34) and (35) are chosen in the E12 standard.

$$C_1 = 1nF \quad (34)$$

$$n = \frac{C_2}{C_1} = \frac{3.3nF}{1nF} = 3.3 \quad (35)$$



If we solve (33) for the parameter  $m$  we get (36)

$$m^2 + m\left(2 - \frac{n}{Q^2}\right) + 1 = 0 \quad (36)$$

We are designing a 6<sup>th</sup> order filter in accordance with what we discovered earlier, so  $Q = 0.5177$  for the first filter stage. We can now use Table 8 to find the  $Q$  value for the first stage in order to calculate  $m$  in (37).

$$m = 0.0978 \vee 10.214 \quad (37)$$

We can now solve (32) for  $R$  in order to determine  $R_2$  for the first filter stage.

$$R = \frac{1}{2\pi \cdot C \cdot \sqrt{mn} \cdot f_c} = \frac{1}{2\pi \cdot 1 \text{ nF} \cdot (\sqrt{10.214 \cdot 3.3}) \cdot 30 \text{ kHz}} = 913 \Omega \quad (38)$$

From (28) we can now calculate  $R_1$  in (39)

$$R_1 = mR = 10.214 \cdot 913 \Omega = 9325 \Omega \quad (39)$$

The  $R_1$  resistance is achieved with two 4.7 k $\Omega$  resistors to form a series resistance of 9.4 k $\Omega$ . The  $R_2$  resistor is similarly achieved with one 820  $\Omega$  and one 100  $\Omega$  resistor in series to form 920  $\Omega$  of series resistance.

Instead of doing all the calculations for each filter stage by hand, we developed a Matlab script to simplify the process. We can now simply just enter the values for  $C_1, C_2, Q, f_c$  for the script to compute the resistor values  $R_1$  and  $R_2$  for each filter stage. This process enables us to quickly compute discrete components for any filter design, when the requirements for the filter are known.

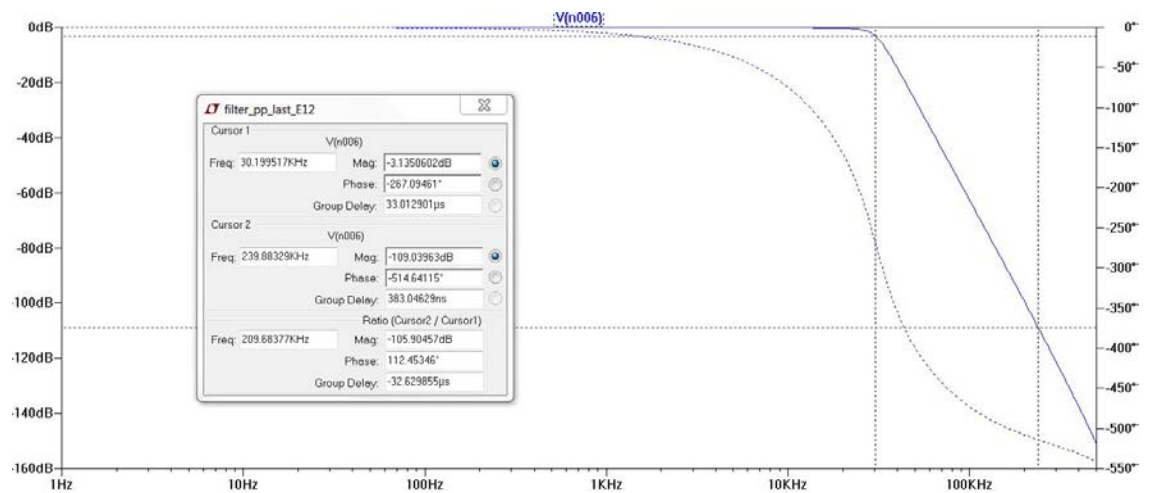
**Table 9 - Discrete components CPS filter**

	Stage 1	Stage 2	Stage 3
$C_1$	1 nF	1 nF	1 nF
$C_2$	3.3 nF	3.3 nF	3.3 nF
$Q$	0.5177	0.7071	1.932
$f_c$	30 kHz	30 kHz	30 kHz
$R_1$	9325 $\Omega$	6105 $\Omega$	1466 $\Omega$
$R_2$	913 $\Omega$	1397 $\Omega$	1279 $\Omega$



Table 9 lists all the discrete components in the CPS filter design, the table is divided into three different stages which represents the foundation for the circuit designed in Figure 44. Further on the circuit has been developed in LTSpice to simulate it with real components and verify that the characteristic of the filter meets the requirements. In this filter design we have used a different op-amp than the TI tool suggested, and thus causing for different circuitry. The existing circuitry has been tweaked to contemplate the E12 series, as it is desirable to keep the Bill of Materials (BOM) list as short as possible

With the circuit in Figure 44, we can use the LTSpice program to create a Bode-plot of the filter response as seen in Figure 46.

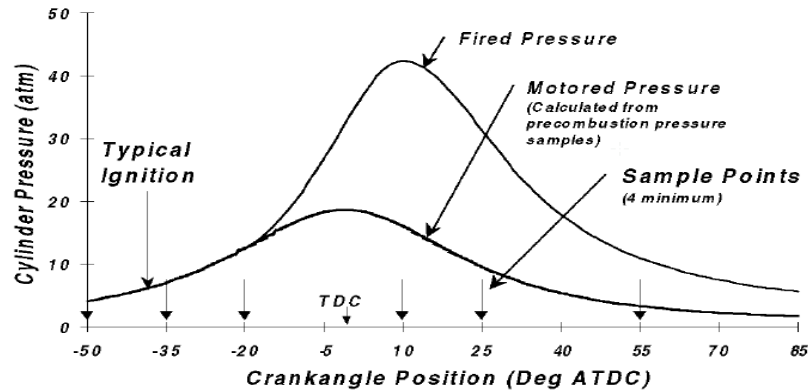


**Figure 46 - Cylinder pressure filter Bode plot**

The filter is almost linear in the frequency range of the CPS sensor, which is 0 - 30 kHz. The upper frequency component we're interested in is located around 18 kHz, which is the upper knocking mode in the frequency spectrum. The cut-off frequency is located at 30 kHz with -3 dB attenuation. The roll off rate is 60 dB/decade before it reaches -109 dB at 240 kHz which is below the -96 dB requirement at 250 kHz found earlier.

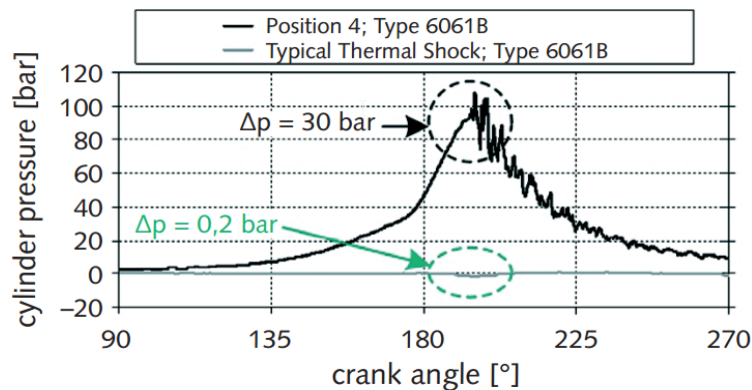
Further on we can evaluate the step response of the CPS filter, but before that we'll need some knowledge about the cylinder pressure in a combustion engine. Engines during normal combustion will maximum have a cylinder pressure of 60 bar<sup>39</sup> as seen in Figure 47

<sup>39</sup> Volvo Personbilar Sverige AB 2005, Grunder motorstyrsystem VT2201



**Figure 47 - Fired and motored cylinder pressures<sup>40</sup>**

But during knocking there is another set of pressures introduced in the combustion chamber as seen in Figure 48, the pressures that arise during knocking can potentially damage the engine and that's one of the reasons we wish to avoid knocking.



**Figure 48 - Comparison of knock versus thermal shock<sup>41</sup>**

The peak pressures points that can arise during knocking can reach up to 110 bar as seen in Figure 48, which is almost twice the pressure obtained during normal operation. In order to maintain regulation the CPS filter will need to pass such signals, (42) give us the output voltage of the pressure sensor for the worst case knocking scenario.

$$\frac{110 \text{ bar}}{0.06894 \text{ bar/psi}} = 159 \text{ psi} \quad (40)$$

<sup>40</sup> Figure from SAE technical paper 2000-01-0932

<sup>41</sup> Figure from Kistler, Pressure indication during knocking conditions 920-349e-11.06



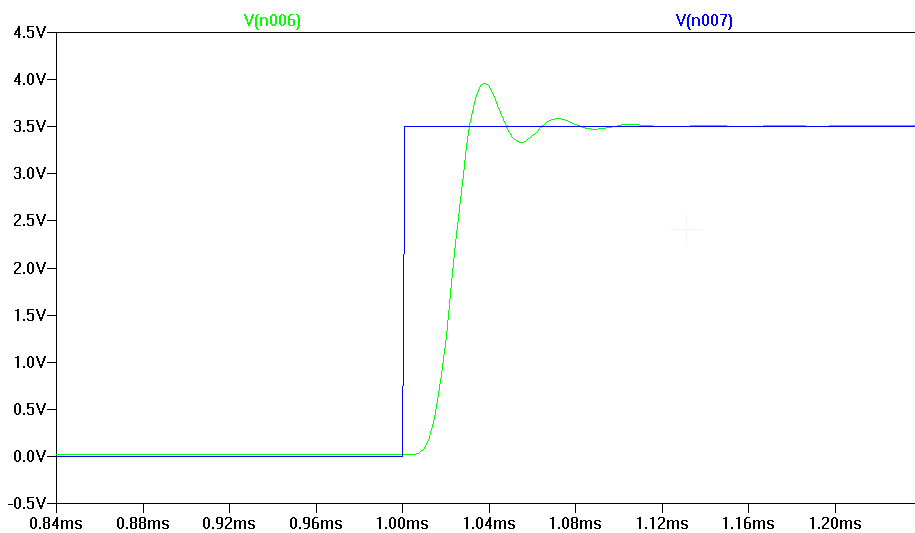


$$P \cdot PSI + 0.5 = V_{out} \quad (41)$$

$$1596 \text{ psi} \cdot 0.00151 + 0.5 \text{ V} = 2.91 \text{ V} \quad (42)$$

This is the CPS sensor output voltage at 110 bar.

We can now perform a step response on the CPS filter to see if the filter with its given circuitry will pass the sensor output. The simulation is illustrated in Figure 49, to 3.5 V step.



**Figure 49 - CPS filter step response to 3.5V**

The filter responds well with its characteristic Butterworth response, there is a slight overshoot followed by some oscillations before the output of the filter stabilizes with minimum offset error. The overshoot is continuous and not cut-off which means that the filter is not saturated, we can calculate the cylinder pressure that this 3.5 V step will translate to as seen in equation (45)<sup>42</sup>.

$$psi = \frac{V_{out} - 0.5}{0.00151} \quad (43)$$

$$\frac{3.5 \text{ V} - 0.5}{0.00151} = 1987 \text{ psi} \quad (44)$$

<sup>42</sup> Optrand, autopsi pressure sensor operating instructions



$$1987 \text{ psi} \cdot \frac{0.06894 \text{ bar}}{\text{psi}} = 137 \text{ bar} \quad (45)$$

Now we have found that the filter will pass the 110 bar cylinder peak pressure with sensor output voltage at 2.91 V, the simulation is executed with a step response of 3.5 V indicating a cylinder pressure of 137 bar. This is good news because it allows us to use op-amps with smaller supply voltages  $\pm 5$  V, which again improves linearity and reduces the power supply requirements.



### 3.7.5. Fluid pressure interface

To be able to determine the pressure of fuel and oil, we are using a fluid fuel pressure sensor from Bosch, the PSS-10<sup>43</sup>. The working operation of this sensor is that the sensor has a sensing element with a constant area, this element responds to a force that is applied to this area. The force that is applied deflects on a diaphragm, bellows or a Bourdon tube. The resulting deflection is then converted to a electrical signal. The sensor is operated on a 5 V supply and the output voltage has a range from 0.5 - 4.5 V. The sensor has a response time of 1.5 ms and can handle pressures from 0.5 - 11 bar.

The oil that circulates inside the engine is pumped around the engine by an oil pump that is located in the oil sump. The oil is pumped and pressurized, and then the oil is moved around the engine, lubricating major mechanical parts like cylinders and valves. Additionally the oil is used to seal and transfer unwanted objects inside the engine, as well as a coolant. We want to measure the oil pressure to assure that the oil pump is working correctly; assuring that the engine is working properly.

We want to measure the pressure of the fuel that is located near the fuel rail. The fuel is pressurized by the fuel pump (4 - 8 bar) and then transferred to the fuel rail. Inside the fuel rail there is a regulator that lets out excess pressure, the pressure in the fuel rail should be between 3 - 6 bar. It is important that the pressure inside the fuel rail is constant, because this information is used by the injectors to determine the injection time of the injectors.

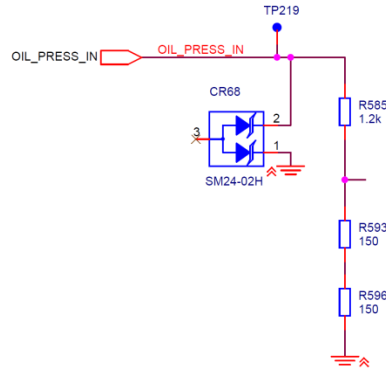
These liquid pressure sensors need to be filtered in order remove any high frequency noise absorbed in the signal lines from the sensor, before going to the XADC.

#### 3.7.5.1. Implementation

The PSS-10 outputs a voltage signal that goes from 0.5 - 4.5 V, while the XADC that is on the Zynq only handles 0 - 1 V on the input. This implies that we'll need to level shift the voltage signals from 0.5 - 4.5 V to 0 - 1 V. This is achieved by a simple voltage divider as shown in Figure 50. When we first calculated the voltage divider we got these values to be 1.2 k $\Omega$  and 300  $\Omega$ . We knew that to scale 0.5 - 4.5 V to 0 - 1 V we would need voltage division ratio of 0.222. As a precaution to protect the ECU, we want to limit the input values to be a bit lower than 1 V and a bit above 0 V. A ratio of 0.2 is appropriate. (46) is used to calculate  $R_2$ , by first selecting a value for  $R_1$ . The result is seen in (47).

---

<sup>43</sup> Bosch, PSS-10 datasheet



**Figure 50 - Voltage divider to level shift input signal**

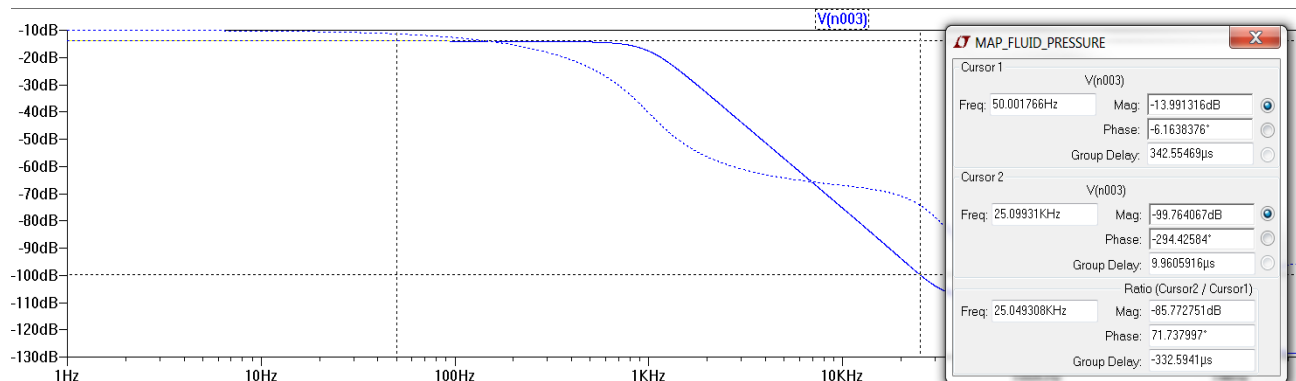
$$\frac{R_2}{R_1 + R_2} = 0.2 \quad (46)$$

Solving for  $R_2$ , we get:

$$R_2 = \frac{0.2R_1}{1 - 0.2} = \frac{240}{0.8} = 300\Omega \quad (47)$$

The filter that is added after the voltage divider has these parameters. Sampling rate at 50 kHz, with minimum -74 dB attenuation at the stopband; 25 kHz (This satisfies the Nyquist-Shannon theorem), 3<sup>rd</sup> order Butterworth response with Sallen-key topology (unity gain) and a cut-off frequency at 1 kHz. The requirements are obtained by using the anti-aliasing filter tool from Texas Instruments<sup>44</sup>.

The frequency response of the filter is seen in Figure 51.



**Figure 51 - frequency response of the filter**

<sup>44</sup> Texas Instruments; Anti-Aliasing Calculation Tool for A to D Converters



As seen in Figure 51, the frequency response starts at approximate -13.9 dB, at 1 kHz the attenuation is -17.6 dB (-3.7 dB), at the stopband (25 kHz) we have -99.8 dB attenuation. At 1 kHz we should have -3 dB, but we are having -3.7 dB, this is not a problem because we want to have as much attenuation after 1 kHz as possible.

Due to the simulation setup, where the voltage divider is also taken into account when simulating the step response, it adds some attenuation in the pass-band to the filter as intended. By calculating the loading effect of the voltage divider in dB, we can subtract that from the overall frequency response. First we need to calculate the effect of the voltage divider has on the filter, by using (48).

$$\frac{V_{MAX} + V_{MIN}}{5 V} = offset\ effect \quad (48)$$

The offset effect added by the voltage divider is then calculated to be:

$$Offset\ effect = \frac{0.9 V - 0.1 V}{5 V} = 0.2 \quad (49)$$

In dB this can be calculated with

$$Offset\ effect_{dB} = 20 \log 0.2 = -13.98\ dB \quad (50)$$

And by using (50) we find that the attenuation of the offset is -13.98 dB. By subtracting this from the overall response we get is;

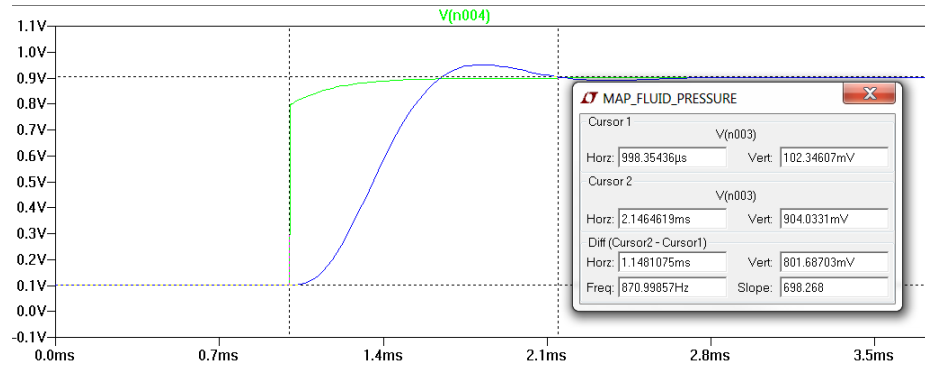
- Frequency response starts at 0 dB
- Cut-off frequency (-3 dB) at 1 kHz
- -74 dB at -17 kHz (which is before the stopband at 25 kHz)

The filter meets the requirements for the frequency response (frequency domain), now we need to check the filter for worst case in the time domain, by testing the filter on response on a transient input.

Filter calculations procedures can be found in chapter 3.7.4.4.

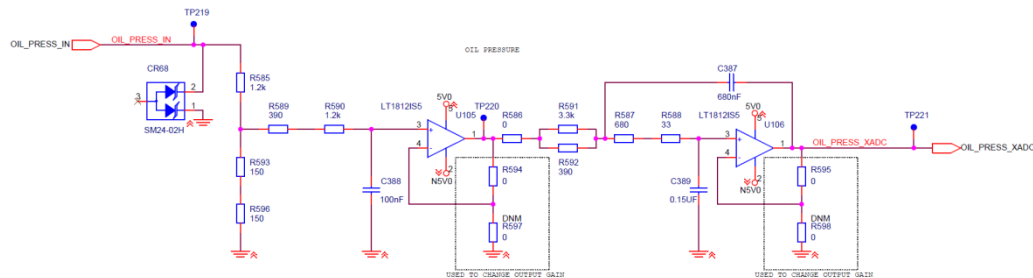


Figure 52 shows the filter response on a transient input.



**Figure 52 - Filter response on transient input**

As can be seen in Figure 52, the filter stabilizes at around 1.2 ms, since the response of the sensor is 1.5 ms we can see that the filter stabilizes before the response of the sensor. With this we can conclude that the filter is good enough for our sensor and all requirements have been met.



**Figure 53 - Filter schematic for liquid pressure sensor**

To the left of the schematic in Figure 53 is the voltage divider that we have implemented to level-shift the input voltage signal from the sensor from 0.5 - 4.5 V to 0.1 - 0.9 V, onwards to the right is the 3<sup>rd</sup> order Butterworth filter with Sallen-key topology. The components that are used are approximated to the closed E12 value. We are using the LT1812 operational amplifier because this is also used in other filters.



### 3.7.6. Manifold air pressure (MAP) interface

To measure the pressure inside the manifold, we are using a pressure sensor from Bosch, the PSA-C.<sup>45</sup> The information that is obtained by this sensor is directly fed into the ECU. The data is used to calculate the air density and can be used to determine the air mass flow into the engine. Furthermore the information is used to calculate the amount of fuel needed in the combustion process. The sensor uses a piezo-resistive sensor element that when a mechanical strain is applied to this element, results into change in electrical resistivity. This sensor has a built in electronics for signal-amplification and temperature compensation. The sensor handles pressures from 0.2 to 2.5 bar and has a response time of 10 ms. The output of the sensor is 0.3 - 4.8 V with a voltage supply of 5 V.

#### 3.7.6.1. Implementation

Before the signal is fed into the ECU, we need to filter the signals to remove any unwanted high-frequency noise before the ADC. The filter has these parameters and is as following; Sampling rate of 50 kHz, 3<sup>rd</sup> order, Butterworth response with Sallen-key topology (unit gain), cut-off frequency at 1 kHz and -74 dB attenuation at 25 kHz (This satisfies the Nyquist-Shannon theorem). The voltage output from the sensor is 0.3 - 4.8 V and needs to be level-shifted to 0 - 1 V.

Since this filter has the same parameters as the fluid pressure sensors. By using the filter design that we designed for fluid pressure sensor we don't need to design a new filter. The implementation is shown in Figure 54. The details regarding this implementation can be found in chapter 3.7.5.1.

Filter calculations procedures can be found in chapter 3.7.4.4.

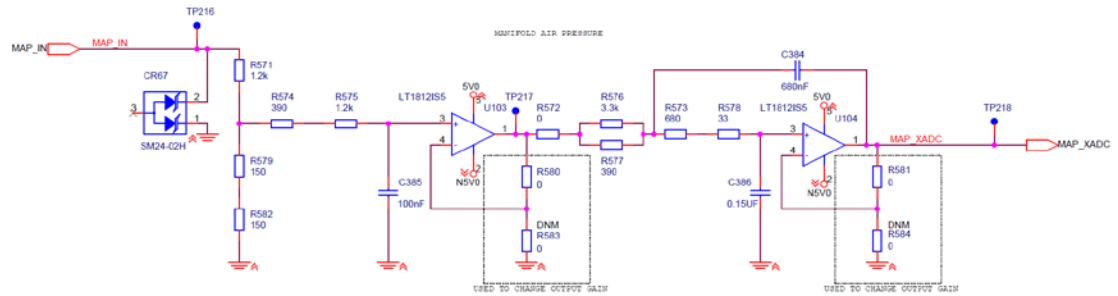


Figure 54 - Manifold air pressure implementation

<sup>45</sup> Pressure sensor Air PSA-C datasheet



### 3.7.7. General purpose analog in filter

As we have available input to the XADC we need to create a general purpose filter that can be used to filter out unwanted noise before the XADC, if the input is going to be used. Since all channels are sampled at around 50 kHz, we need to have 73 dB attenuation before Nyquist, which is at 25 kHz. To be able to do this we need a 3<sup>rd</sup> order filter. The filter is a 3<sup>rd</sup> order Butterworth filter with Sallen-Key topology. We have implemented two types of 3<sup>rd</sup> order filter. One with a cut-off frequency at 1 kHz and a second at a cut-off frequency at 320 Hz, both filters has a stopband at 24 kHz. The attenuation at 24 kHz is 81 and 83 dB for the two cut-off frequencies, which meets the requirements of 73 dB attenuation at 25 kHz.

The filter with 320 Hz is used in general voltage applications to filter out noise and give us the DC signal that we need. While the 1 kHz filter is used for new sensors that need to be interfaced. All filters in the design have the flexibility to change cut-off frequency and gain of each stage of the filter by modifying the resistors and adding values for the 0  $\Omega$  resistors. Filter calculations can be found in chapter 3.7.4.4.

The implemented DC filter is seen in Figure 55 and sensor filter in Figure 56.

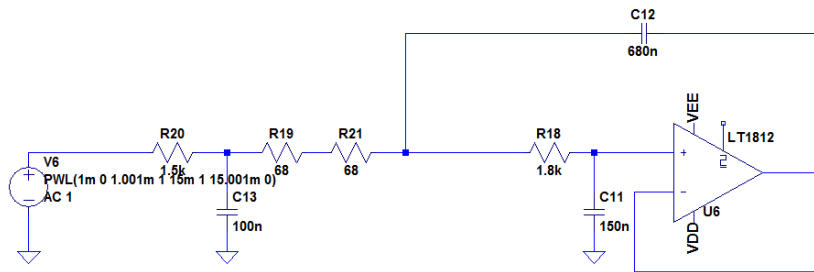


Figure 55 - DC inputs analog filter

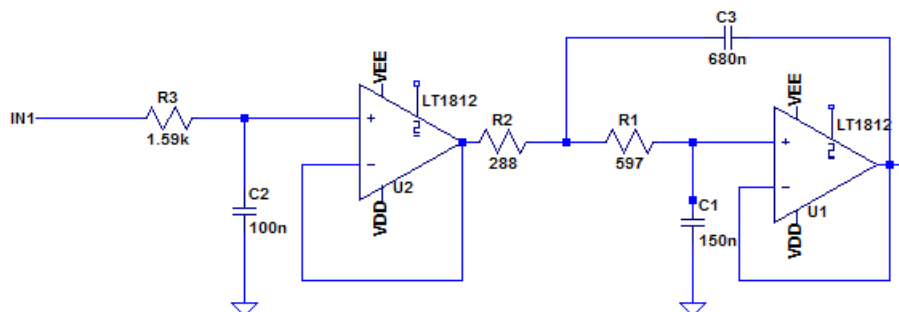


Figure 56 - General purpose analog filter





### 3.7.8. Zynq-7000 XADC

The Zynq-7000 SoC has an internal dual 12 bit 1 MSPS internal ADC. Note that this chapter only describes briefly the hardware setup for the XADC. For full setup of the XADC and use of internal sensor data, see latest version of UG480<sup>46</sup> from Xilinx. The dual ADCs have multiplexed inputs that are expanded into 17 external ADC channels. One of the channels is a dedicated channel and not accessible on the B2B connectors of the TE0720. The remaining 16 ADC channels are auxiliary channels that can be used as regular I/O if not used as ADC input channels. All 16 auxiliary input channels on the KPEC ECU are used. The remaining channels that are not used for dedicated sensor inputs can be used as general purpose analog inputs.

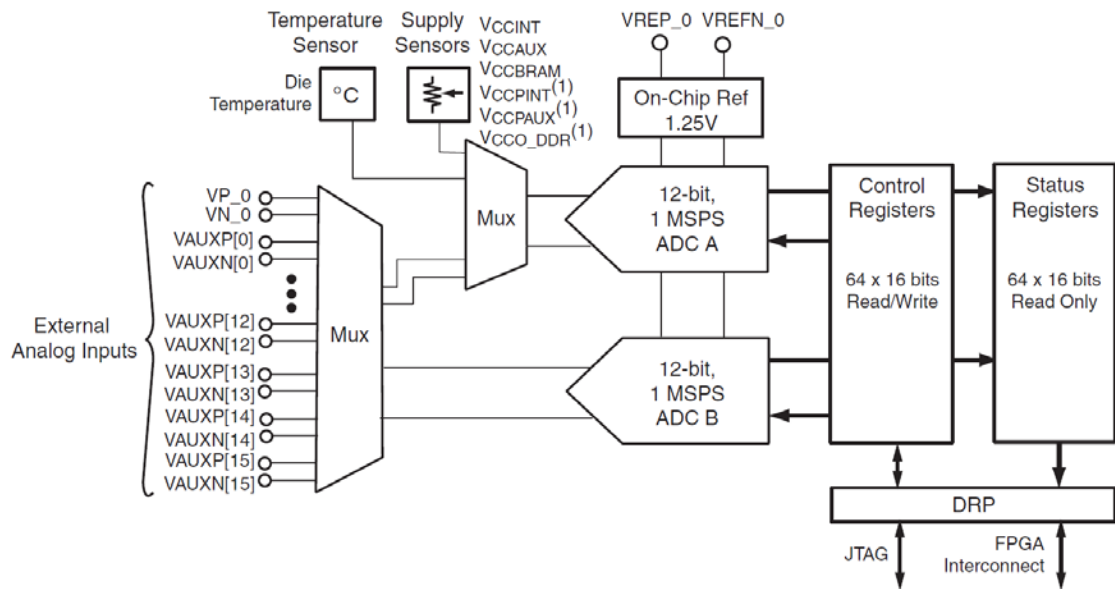


Figure 57 - XADC functional diagram<sup>47</sup>

Figure 57 shows a functional diagram of the XADC. In addition to external analog inputs (auxiliary inputs), the XADC can monitor the power supplies connected to the Zynq SoC as well as the internal temperature of the IC. The XADC is powered from a dedicated 1.8V LDO on the TE0720 for noise performance. The 1.8 V on-board LDO also powers a 1.25 V voltage reference IC providing external reference voltage for the XADC.

#### 3.7.8.1. XADC hardware configuration

The XADC have differential inputs on all channels for noise suppression as the ADC can reject common mode noise on input signals. The XADC can be configured for both unipolar and bipolar input signals, with the limitation that the *maximum* peak-to-peak voltage  $V_{p-p}$  does not exceed 1 V. The auxiliary XADC

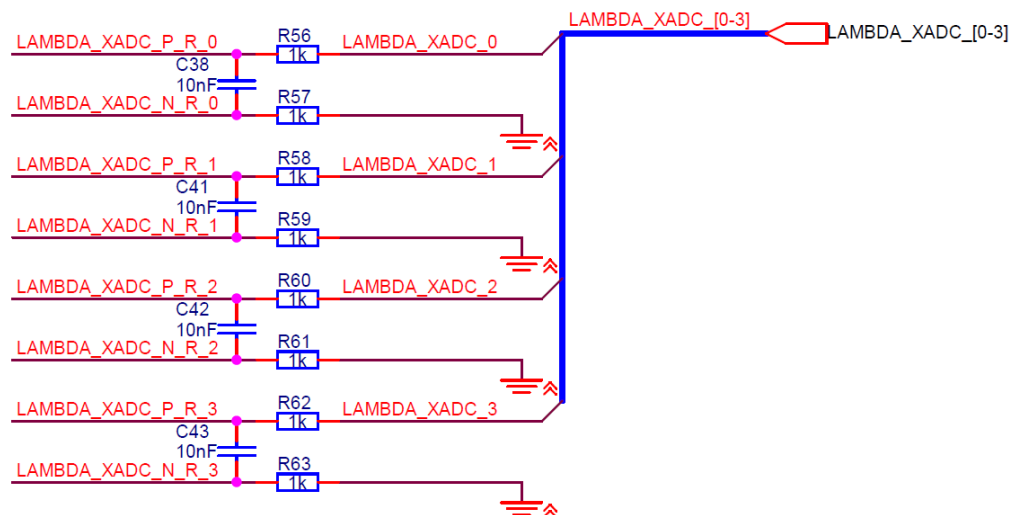
<sup>46</sup> Xilinx UG480 User Guide

<sup>47</sup> Figure from Xilinx UG480



channels are configured for unipolar inputs in the KPEC ECU, meaning that the maximum input voltage of the positive side of the differential signals cannot exceed 1 V with 0 V GND potential. Because all input channels are differential inputs, each channel requires two connections per analog input. Therefore, the 16 auxiliary channels use a total of 32 inputs on the TE0720 B2B connector.

The sensor data measured with the XADC in the KPEC ECU are all single ended signals. In order to couple these to differential inputs, a dedicated ground line must be routed with the positive signal as a differential trace pair from the end of each active filter. The active filters should be placed as close to the B2B connector as possible, thus making the ground differential lines short traces on the board. At the end of each analog differential pair is a passive filter.



**Figure 58 - XADC differential passive filters example**

An example of the passive filters at the end of the differential input signals are shown in Figure 58. The large 1 k $\Omega$  resistors limit the inrush current to the XADC from the output of the active filters. The required data acquisition time is dependent on the input source resistance and the size of the internal sample capacitor  $C_{SAMPLE}$ .  $C_{SAMPLE}$  is typically in the range of a few pF, so even a low capacitance 10 nF capacitor between the input pairs should be sufficient in order to not suppress minimum data acquisition requirements of the XADC.

### 3.7.8.2. XADC channel assignment and sampling

As mentioned above, all the XADC channels are used for analog inputs in the KPEC ECU. Most of the inputs are designed for special types of automotive sensors, while 3 inputs can be used as general purpose analog inputs. Keep in mind that the cut-off frequency of the general purpose analog input filters are 1 kHz.

**Table 10 - XADC channel assignments**

<b>XADC channel</b>	<b>Signal</b>	<b>Filter description</b>
<b>0</b>	BAT_VOLTAGE	<b>3.7.7</b>
<b>1</b>	THR_POS_1	<b>3.7.7</b>
<b>2</b>	MAP_XADC	<b>3.7.6</b>
<b>3</b>	ACC_SIG_1	<b>3.7.7</b>
<b>4</b>	GPAI_1	<b>3.7.7</b>
<b>5</b>	GPAI_0	<b>3.7.7</b>
<b>6</b>	FUEL_PRESS_XADC	<b>3.7.5</b>
<b>7</b>	LAMBDA_XADC_3	<b>3.8.2.5</b>
<b>8</b>	GPAI_2	<b>3.7.7</b>
<b>9</b>	THR_POS_0	<b>3.7.7</b>
<b>10</b>	THR_VFB	<b>3.7.7</b>
<b>11</b>	ACC_SIG_0	<b>3.7.7</b>
<b>12</b>	LAMBDA_XADC_2	<b>3.8.2.5</b>
<b>13</b>	LAMBDA_XADC_1	<b>3.8.2.5</b>
<b>14</b>	OIL_PRESS_XADC	<b>3.7.5</b>
<b>15</b>	LAMBDA_XADC_0	<b>3.8.2.5</b>

Table 10 shows the channel setup for the different XADC input channels. All input signals are connected to the XADC as differential inputs shown in Figure 58.

It is possible to sample pairs of input channels simultaneously with the two internal ADCs for achieving the highest sampling rate across channels as possible. This is described as *Simultaneous Sampling Mode* in UG480. Auxiliary input channels 0 - 7 are sampled by ADC A, while channels 8 - 15 are sampled by ADC B in Figure 57. This means that XADC channels  $n$  and  $n + 8$  are sampled at the same time. Channel 0 is sampled simultaneously with channel 8, channel 1 and 9, and so on. On chip temperature and voltage supply monitoring can also be added to the sampling sequence in Simultaneous Sampling Mode.



### 3.8. Lambda sensor interface

The scope of using a lambda sensor is to monitor the amount of oxygen ( $O_2$ ) in the exhaust when the combustion process is completed. The amount of oxygen in the exhaust after combustion is a result of the air/fuel mixture. The emissions of a perfect combustion process are only  $CO_2$ ,  $H_2O$ , and  $N_2$ , if the combustion is stoichiometric ( $\lambda = 1.0$ ). The definition of the lambda 1 is 14.7 kg air to 1 kg fuel, the mixture is said to be lean if the amount of air is greater than 14.7 kg. The mixture is said to be rich if the amount of air is less than 14.7 kg for 1 kg of fuel.

It is not always desirable that the lambda ratio should be equal to 1.0. Maximum performance is obtained with a rich mixture, and best fuel economy is obtained with lean mixture. The air/fuel ratio must therefore be continually regulated (injector time and air flow) according to engine conditions.

The lambda sensor must be mounted in the exhaust manifold of a naturally aspirated engine, or after the downpipe of a turbo engine. The measured exhaust gases must be as hot as possible to ensure correct reading, because the sensor output is only accurate when the exhaust temperature is exactly 780 °C.

Wideband lambda sensors are currently the most accurate way to measure ultra-lean air/fuel mixtures (ranges from 0.7 to 40). The sensor element is made of a ceramic element Zirconium Dioxide ( $ZrO_2$ ), and consists of two sensing elements (cells); Nernst cell and pump cell.

The Nernst cell utilizes electrochemistry, and the cell produces a voltage according to the level of oxygen present in the emissions. If the ratio of air and fuel is stoichiometric, the Nernst reference voltage is approximately 450 mV. If the mixture is rich the reference voltage is above 450 mV, and if the mixture is lean, the reference voltage is below 450 mV. Figure 59 gives a feedback block diagram of the interface with the pump current as the plant, and the Nernst cell as the sensor element.

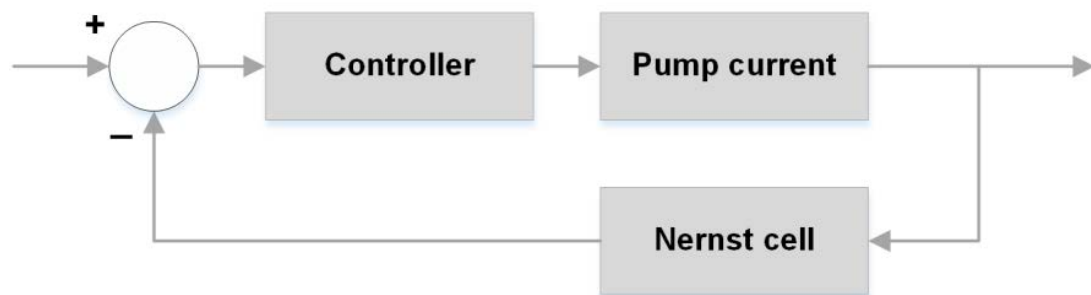


Figure 59 - Feedback block diagram of lambda sensor interface



It is not adequate to only use the Nernst cell reference as a measurement of air to fuel mixture. The pump cell current determines the lambda value along with correction factors (temperature, resistance and voltage levels). The intention of the pump cell is to pump free oxygen away from the Nernst cell, and to consume the free oxygen. The control unit needs to control and measure the pump current (and direction) required to produce a change in the Nernst cell voltage. If the pump current is zero, then a stoichiometric ratio is present. The current direction determines the rich or lean state according to a lookup table provided by Bosch.

It is highly complex to interface wideband lambda sensors. The accuracy of the Nernst cell reading is dependent of the temperature of the lambda sensor which needs to be held constant. The nominal heater voltage should not exceed 7.5 V due to the fact that the pump/Nernst cell is ceramic elements, which will be destroyed if subjected to a rapid temperature change. The activation of the heating element should be performed in multiple stages to ensure maximum durability of the lambda sensor. The following sections documents design decisions for 2 lambda sensors.



### 3.8.1. Wideband linear lambda sensor

There are multiple suppliers of lambda sensors, and a comparison of different lambda sensors available on the market is shown in Table 11. The importance factors are weighted according to both the system requirements and the engineering perspective.

**Table 11 - Pugh-matrix for selection of wideband lambda sensor.**

Criteria	Importance	Bosch LSU 4.9	Denso O2	Delphi WRO2
Accuracy	5	5	3	3
Available documentation	4	3	2	3
Long term availability	5	4	4	3
Ease of implementation	3	3	4	4
Durability	3	5	3	4
Cost (1=Expensive)	2	2	4	4
<b>SUM</b>	<b>22</b>	<b>3.9</b>	<b>3.3</b>	<b>3.4</b>

The decision based matrix implies that Bosch LSU 4.9 is the best lambda sensor to interface for our engine control unit. The Bosch LSU 4.9 utilizes a robust design which has been developed over multiple years, has excellent accuracy and sensitivity. However, it is quite complex to interface, and Bosch recommends to only use this sensor with their lambda sensor interface IC CJ125 (unavailable to buy for small projects). To determine which solution that suits our design, we compare them in Table 12, where design on open source material is found to be beneficial. This design is based on Megasquirt, RusEFI, and Bosch automotive datasheets and modified to interface the Zynq XADC.

**Table 12 - Pugh-matrix for selection of lambda sensor interface.**

Criteria	Importance	IC - Bosch CJ125	External - Bosch LT4 / ALM	"Open Source" design
Risk (5=Low risk)	5	5	4	2
Available documentation	4	2	3	3
Long term availability	5	2	3	5
Ease of implementation	4	3	4	3
Durability	4	5	3	3
Cost (1=Expensive)	3	3	1	5
<b>SUM</b>	<b>25</b>	<b>3,9</b>	<b>3,6</b>	<b>4</b>



### 3.8.2. Bosch lambda sensor LSU 4.9

To read oxygen level from a Bosch LSU 4.9, we need to monitor the Nernst cell, pump cell and the heater temperature. In Figure 60, an equivalent circuitry of the sensor is displayed with net-aliases for this design, and the Bosch net-aliases and wiring colours according to the datasheet<sup>48</sup>.

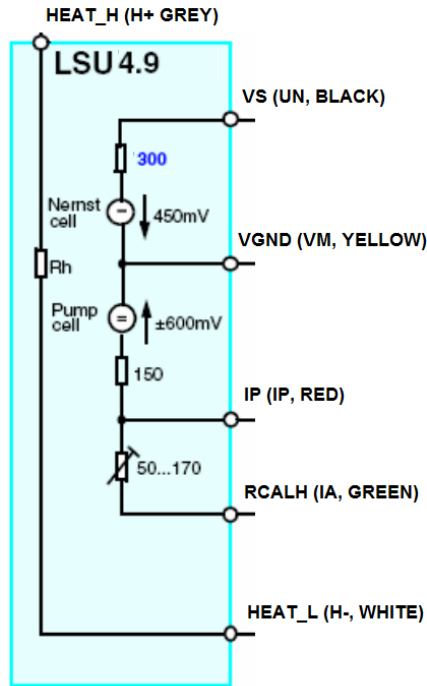


Figure 60 - LSU 4.9 equivalent circuitry<sup>49</sup>

Table 13 gives an overview of the different net aliases, a description of their names and functions, and electrical characteristics that must be interfaced.

Table 13 - LSU 4.9 signal overview.

Net alias	Name	Function	Electrical characteristics
VS	Nernst cell voltage input	Nernst cell reference voltage	450 mV ( $\lambda = 1.0$ ), $\pm 0.4$ mA
VGND	Virtual ground	Ground for pump current control and Nernst cell.	2 V above ground. Max 32 mA
IP	Pump cell current input	Pump cell source/sink amplified current.	0.6 - 4.4 V, max $\pm 30$ mA (peak)
RCALH	Calibration resistance	Unique internal resistive value (temperature dep.)	5 - 200 $\Omega$ (Can be connected directly to IP).
HEAT_H	Heater element high side.	High side of heater element – $V_{BAT}$ (fused)	Max 13 V – $I_{NOM}$ 2 A – $I_{MAX}$ 10 A
HEAT_L	Heater element low side.	Low side switch for activation time (PWM)	Duty cycle activation – max 63%

<sup>48</sup> Bosch Motorsport 2015

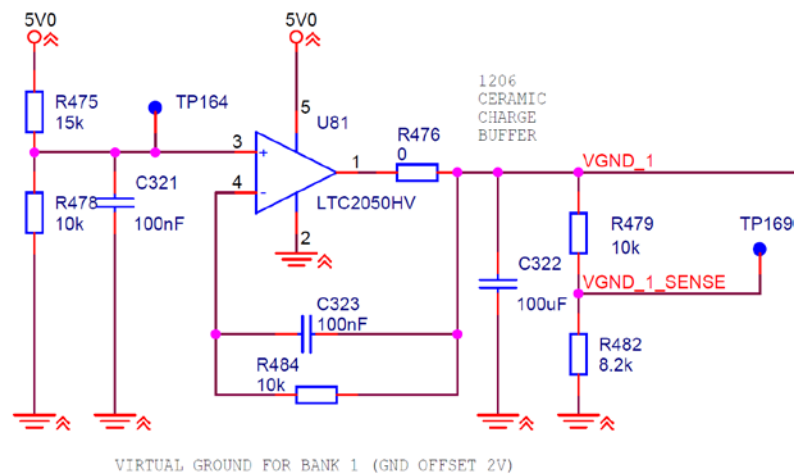
<sup>49</sup> Figure from CJ125 Datasheet



### 3.8.2.1. Virtual ground circuit

The functionality of the sensor interface where described in the introduction. To enable sinking and sourcing of the pump cell current of the LSU 4.9, the pump/Nernst cell reference ground needs to be held 2 V above signal ground. This can be defined as virtual ground, and is realizable with an operational amplifier such as Linear Technology LTC2050HV<sup>50</sup> operational amplifier, which is versatile and used throughout this design. This operational amplifier offers good DC performance, and its small package size (SOT-23) makes it beneficial in this design.

To create the 2 V offset, we use a simple voltage divider with a 5 V source, and this gives an input of 2 V to the non-inverting input, as shown in Figure 61.



**Figure 61 - Virtual ground circuitry**

The operational amplifier and the 100  $\mu$ F capacitor provide a short term charge buffer<sup>51</sup> which provides stability and high power virtual ground for the sensor. The 100 nF capacitors stabilize the input signal reference.

Because this is the common ground level for the pump and Nernst cell, it's vital that the FPGA knows the real voltage output of the VGND, so that the lambda value can be calibrated according to this voltage level. This is provided by the voltage divider which gives an output value of 0.9 V when the VGND is 2 V. To verify that the circuitry works as intended, a simulation is performed in LTSpice as shown in Figure 62.

<sup>50</sup> Linear Technologies 2015, LTC2050HV

<sup>51</sup> TechEdge 2015, WBO2 DIY,



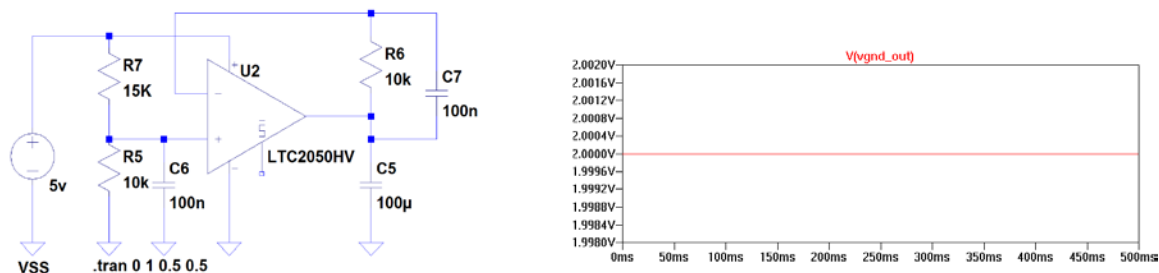


Figure 62 - Simulation of virtual ground in LTSpice

### 3.8.2.2. Pump current control

The pump current control circuitry must be able to sink and source current in the pump cell to push or pull oxygen from the Nernst cell. The pump current  $I_P$  can be considered as a regulated plant and the Nernst cell is the sensor as shown in Figure 59. This implies that the plant gain parameter  $I_P$  must be controlled bi-directionally according to the Nernst cell voltage reading. The circuitry consists of multiple stages, a DAC, a current amplifying current, and a voltage converting circuitry to enable this functionality.

To generate analog voltage levels from the Zynq to the current amplifier, a DAC must be selected with accurate resolution. A 12-bit DAC with clock rates up to 40 MHz and SPI-compatibility is therefore preferable. The selected component TI DAC124S085 (in Figure 63) fulfils these requirements. To get more accurate resolution, a LDO which converts voltage from 5 V to 3.3 V is applied as a reference voltage which gives  $\frac{3.3V-0V}{2^{12}} = 0.8 \frac{mV}{bit}$  resolution.

This DAC also provides two extra outputs which are assigned to analog outputs for general purpose which are described in section 3.8.2.6. The Zynq is provided with calibration voltage from the voltage divider which is ideally 0.7 V when the input is 0.5 V, this is to correct for aberrations from theoretical output voltages. Nominal voltages and numerical values for digital implementation are provided in Table 14.

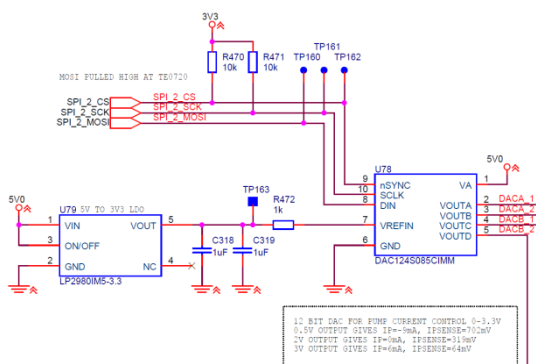
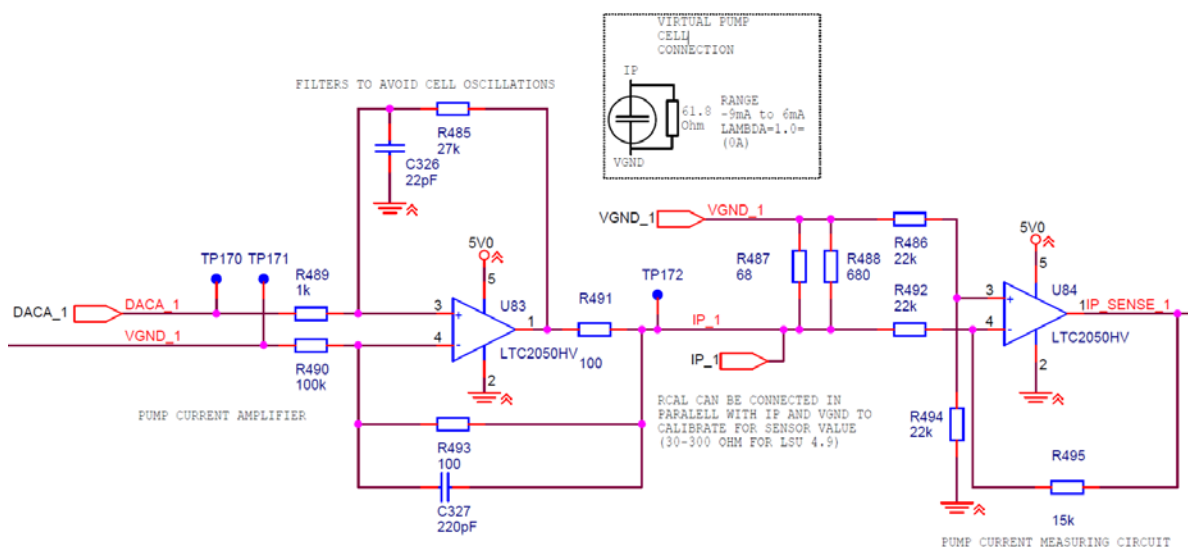


Figure 63 - DAC for pump current control



The pump current control circuitry in Figure 64 consists of an amplifying circuit and a current sensing circuit to determine the current flowing in the pump cell (feedback path). The output voltage of the DAC determines the direction and magnitude of the pump current (IP), which is referenced to the virtual ground (2 V). This implies that when the DAC output is 2 V, there is no current supplied to the pump cell. The 220 pF and 22 pF capacitor stabilizes the circuitry due to oscillations that occur in the cell, as well as external signal noise.



**Figure 64 - Pump current circuitry**

The resistors coupled in parallel (68  $\Omega$  and 680  $\Omega$ ) are matched to the pump cell impedance which the lambda sensor is connected in parallel. The IP\_SENSE\_1 signal is a voltage output signal which is provided over the shunt resistor (0 - 1 V for XADC). The IP\_SENSE\_1 and DACA\_1\_SENSE must be handled by the Zynq for sensor free-air calibration to determine the normalized pump current which then can derive the lambda stoichiometric value which ideally is 0 A.

Bosch recommends a continuous current value for the pump cell of 20  $\mu\text{A}$ <sup>52</sup>, but the outer range of pump current values is defined to -9 - 6 mA. Normal startup value for the lambda sensor is typically 20  $\mu\text{A}$ , and must be regulated according to the Nernst cell voltage. The DAC is a rail to rail type, but offers linearity errors in LSB near 0 V. The regulation span from the DAC is therefore designed between 0.5 - 3 V for maximum lifetime of sensor. Table 14 gives an overview of the maximum voltages levels of the DAC.

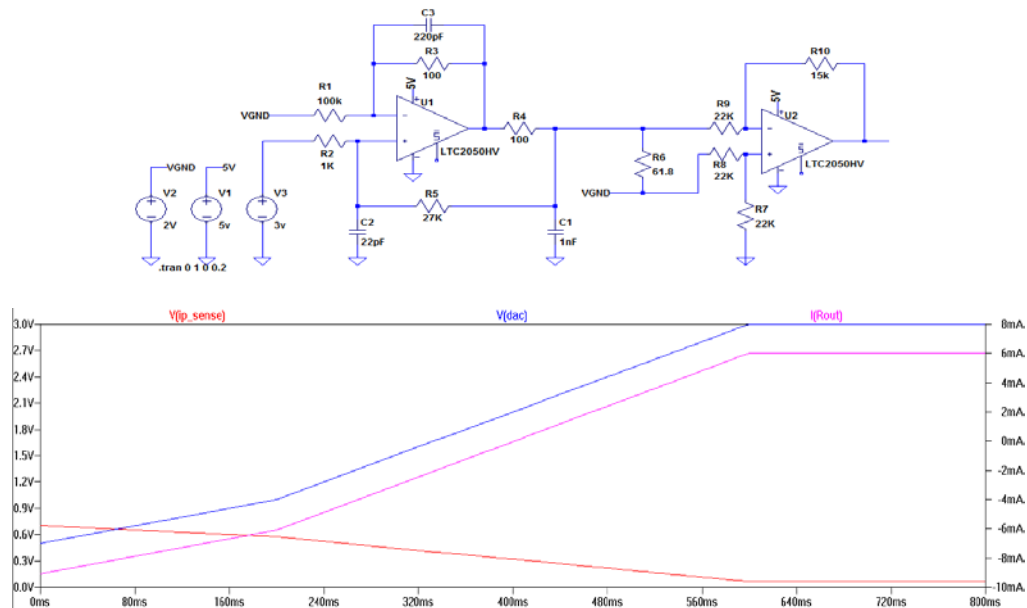
<sup>52</sup> Bosch LSU 4.9 2005

**Table 14 - Maximum typical levels for DAC**

Relative $\lambda$ -value	Pump Current output response (IP)	DAC output voltage (To current amplifier)	DAC input numerical value (From FPGA)	IP_SENSE output voltage (to XADC)	DACA_SENSE output voltage (to XADC)
$\lambda < 0.65$	6 mA	3 V	3724	64 mV	744 mV
$0.65 \leq \lambda < 1$	3 mA	2.5 V	3103	192 mV	620 mV
$\lambda = 1$	<b>0 mA</b>	<b>2 V</b>	<b>2482</b>	<b>319 mV</b>	<b>500 mV</b>
$\lambda > 1$	-9 mA	0.5 V	621	702 mV	124 mV

The pump current must be regulated continuously according to the nernst voltage reference. The nernst cell spans from 0.2 - 0.8 V. When the nernst voltage is 0.45 V, the pump current should be around 0 A, but to verify this, a small current of  $\pm 20 \mu\text{A}$  (DAC numerical value 2507) must be applied. When there is a change in the nernst voltage, the lambda value could be determined as around stoichiometric point.

When the nernst voltage is above 0.45 V (rich mixture), the pump current should be incrementally applied (from 2482) until the target value of 0.45 V is obtained. When the nernst voltage is below 0.45 V (lean mixture), the pump current should be detrimentally applied (from 2482) until the target value is obtained. The lambda value is rapidly changing, and should have adequate refreshment rates to ensure best results. A simulation of the pump current circuitry is performed in Figure 65 which verifies the functionality.

**Figure 65 - Simulation of pump current circuitry**



### 3.8.2.3. Nernst cell monitoring

The Nernst cell voltage needs to be monitored according to the presence of oxygen in the exhaust, as this is the regulation parameter. When the pump current is changed, an equivalent response should be monitored by the Nernst cell voltage. This circuitry is also referenced to virtual ground just as the pump cell. A differential amplifying circuit is designed as shown in Figure 66, which is connected to a CMOS low voltage MUX.

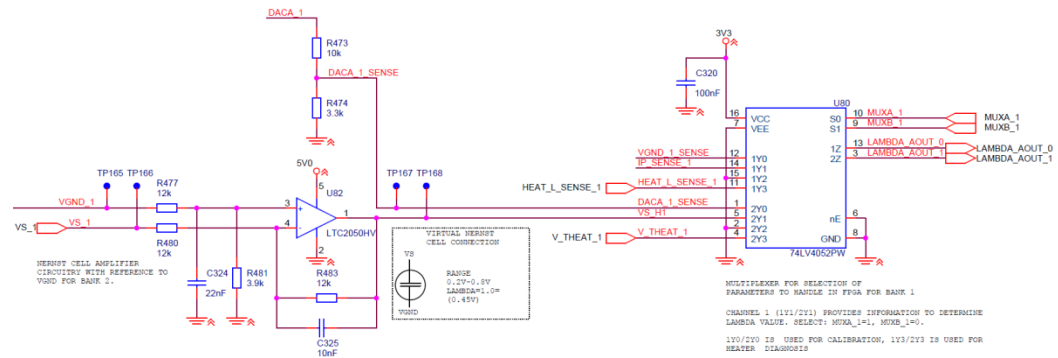


Figure 66 - Nernst cell circuitry

The Nernst voltage range is typically between 0.2 - 0.8 V (according to the oxygen level). The output signal must therefore be designed within the upper and lower supply rail of the operational amplifier (LTC2050HV). The output signal VS\_1 provides the XADC with analog voltage level in the range of 781 mV (Rich mixture) to 181 mV (Lean mixture). The target value output of the amplifier is 531 mV due to the Nernst cell stoichiometric voltage is 0.45 V.

The Nernst cell element generates noise when the temperature of the sensor is below the operation value (780 °C). To avoid signal noise, the 10 nF and 22 nF capacitors filter the signal. The connection RCAL can be connected directly to the VGND/IP for calibration purposes to determine the internal resistance of the Nernst cell. The lambda interface must support two cylinder bank engines, so that a total of twelve signals must be handled by the Zynq XADC. To avoid using twelve pins of the XADC, filters and multiplexer is beneficial.

The 74LV4052<sup>53</sup> is beneficial in this design, and is recommended in applications with multiple analog voltage inputs. This multiplexer/demultiplexer is a dual four-channel type with common select logic. This implies that 1Yn and 2Yn is activated simultaneously on the output channels 1Z and 2Z when they are selected via S0 and S1. The channels 1Y1 and 2Y1 are the highest priority during normal operation. This gives information of the pump current (IP\_SENSE\_1) and the Nernst cell

<sup>53</sup> 74LV4052 Datasheet

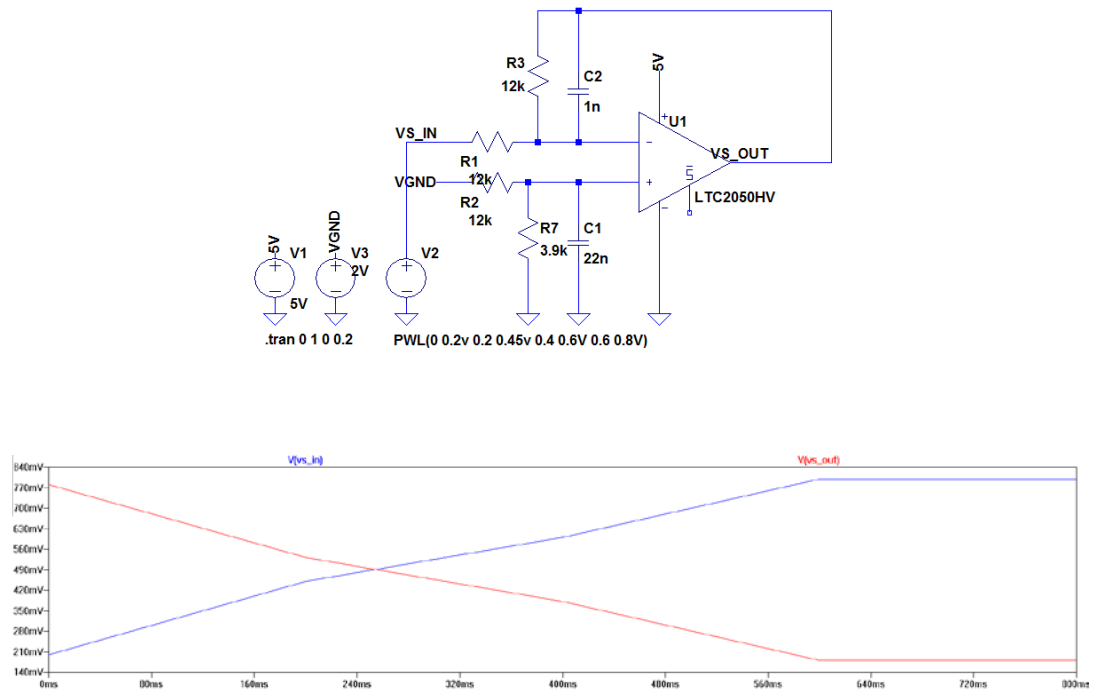


reference voltage (VS\_1). Table 15 gives an overview of the signals that is multiplexed to the XADC.

**Table 15 - Net aliases for signals with description and characteristics.**

Net alias	Description	Electrical characteristics
VGND_x	Virtual ground	DC 0 - 1 V
DACA_x	Voltage input for current amplifier.	DC 0 - 1 V
IP_SENSE_x	Voltage output of pump current amplifier circuit.	DC 0 - 1 V
VS_x	Voltage output of Nernst cell amplifier.	0.8 V (Rich) 0.2 V (Lean)
V_THEAT_x	Voltage output of heater circuit.	DC 0 - 1 V
HEAT_L_SENSE_x	Voltage input of heater circuit for diagnosis and temperature determination.	$0.053 \cdot V_{BAT}$ (normally 0.7 V)

A simulation of the Nernst cell amplified output is performed in Figure 67. The output voltage is plotted versus input Nernst cell voltage which verifies the design. This voltage is typically varying according to the oxygen level in the exhaust, as well as the temperature of the sensor, and other correction factors. Bosch recommends a minimum sampling rate of 2 kHz for the Nernst cell, and recommends 20 kHz for precision usage.

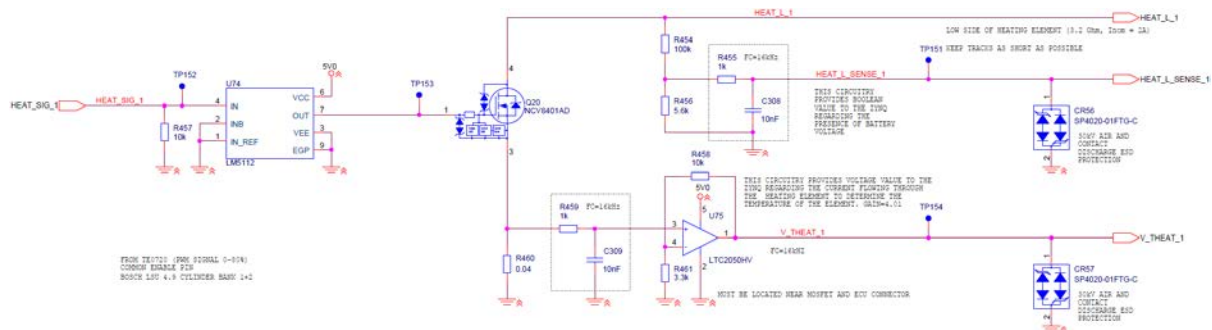


**Figure 67 - Simulation of Nernst cell amplified output vs Nernst cell voltage**



#### 3.8.2.4. Heater driver circuitry

The heater element must be controlled accurately so that a normal operating temperature can be obtained within 20 seconds after startup. The normal operating temperature is recommended by Bosch to 780 °C; however the activation logic for this circuitry is not trivial.



### Figure 68 - Schematic design of heating circuitry

Figure 68 shows the schematic design for the heater circuitry of bank 1 sensor. The heating element has nominal impedance of  $3.2\ \Omega$  at room temperature, and increases with the temperature. The changing impedance with temperature can be used to determine target value of  $780\ ^\circ\text{C}$  by measuring the voltage drop over the heating element. The circuitry utilizes a voltage divider on the low side of the element, for Boolean diagnostic sensing purposes. The voltage will be  $0.053\ V_{BAT}$  when the heating element is powered by  $V_{BAT}$ .

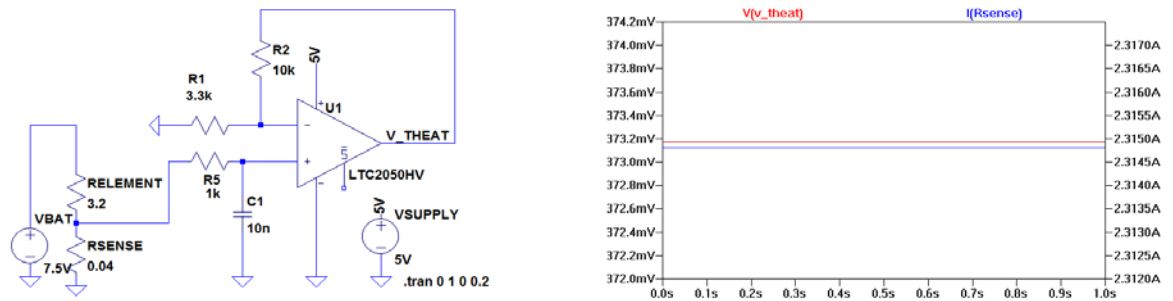
To control the activation of the heating element, the high current MOSFET (NCV8401AD<sup>54</sup>) is selected with internal protection and a pre-driver circuitry which utilizes PWM signals as input. This circuitry is used in injection and ignition drivers, and offers high switching frequency and high power which is beneficial in this design. However, Bosch recommends a heater activation frequency of approximately 100 Hz. The Nernst cell and pump cell is ceramic elements that are sensitive to shock and sudden temperature. The activation sequence shall therefore be performed in step, and the nominal voltage should never exceed 63%  $V_{BAT}$

The heating element should use engine chassis ground, to avoid oscillations on the virtual ground reference. To monitor the heating current, the small voltage across the shunt resistor ( $0.04\ \Omega$ ) is sensed and amplified by the operational amplifier LTC2050HV. Due to noise on the chassis ground, the input of the operational amplifier is low pass filtered. Bosch states that the average heating current is  $2\text{ A}$ ; this gives nominal power dissipation over the shunt resistor of  $P = I^2 R = 160\text{ mW}$ .

54 NCV8401AD Datasheet



The sensing voltages V\_THEAT\_1 and HEAT\_L\_SENSE\_1 in combination determine the heater current, and are used as a basis for determining the heaters warm up phase. After operation temperature is obtained, a heating maintenance program must be obtained to regulate the temperature of the sensor to around 780 °C. The input signals towards the ECU; VS\_x, IP\_x, VGND\_x is ESD protected for up to 30 kV for contact and air discharge. A simulation is performed in figure to verify that the circuitry works as intended.



**Figure 69 - Simulation of heater current sensing circuitry**

When the lambda sensor is used in engines that use bioethanol as fuel, a software calibration must be performed due to the fact that bioethanol (E85) contains less energy than petrol. This implies that the engine needs more bioethanol than petrol to operate stoichiometric and hence give maximum performance. A lambda value of 1.0 for petrol engines is equal to 1.5 for bioethanol, which Table 16 gives an overview.

**Table 16 - Lambda value comparison for gasoline vs bioethanol<sup>55</sup>**

Lambda value	Gasoline	Bioethanol (E85)
0.7	10.3	6.8
0.9	13.2	8.8
1.0	14.7	9.8
1.1	16.2	10.8
1.3	19.1	12.7
1.5	22.1	14.7

<sup>55</sup> MoTec 2015



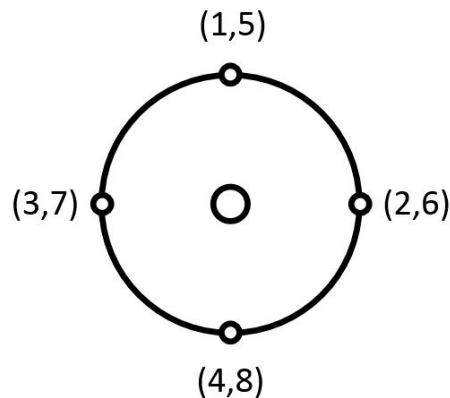
### 3.8.2.5. *Lambda filter design*

We have identified another sensor that are applicable for custom filter designs, while the rest is assumed general input DC signals where we don't anticipate their frequency to extend beyond 10 Hz.

From basic engine theory, we know that for a four-cycle engine it takes two complete revolutions to complete one single combustion, adding up to  $720^\circ$  of crankshaft angle.

The idea is that if we know the firing frequency in the worst case scenario of the engine, we can use this knowledge to figure out how many times/sec an exhaust valve opens and thus generating gases that we wish to measure with our Lambda probe. With knowledge of this frequency we can choose a reasonable sampling rate  $f_s$ , which then in turn set the filter requirements.

Figure 70 is an illustration of an 8 cylinder V configuration crankshaft, with cylinder position is illustrated in Figure 70. We can see that the pistons 1&5, 2&6, 4&8, 3&7 are pairs connected to the crankshaft, and will occur at top dead center (TDC) respectively in each cylinder block, while the crankshaft are turning through its cycle.



**Figure 70 - Two-plane crankshaft on an 8 cylinder engine**

Due to the V configuration, ignition cannot occur for each cylinder pair at the same time, as it would cause the engine to counteract itself. The ignition firing<sup>56</sup> order for this two-plane 8 cylinder V engine is :1, 5, 4, 8, 6, 3, 7, 2 shifted at  $90^\circ$  of rotation adding up to  $720^\circ$  of crankshaft displacement.

We now know the firing order, the *next* phase in the combustion process will be to open the exhaust valve in each respective cylinder, and thus getting rid of the combustion gases that we want to measure to maintain regulation of the engine.

The equation for finding ignition<sup>57</sup> frequency is given below

<sup>56</sup> What-When-how

<sup>57</sup> Performance ignition systems





$$f_{IGN} = \frac{\text{Number of cylinders}}{120} \cdot RPM \quad (51)$$

So the ignition frequency at worst case scenario for our 8 cylinder engine is

$$f_{IGN} = \frac{8}{120} \cdot 16.000RPM = 1067Hz \quad (52)$$

To ensure proper data acquisition, we will benefit from an analog low-pass filter that will attenuate high frequency components, it is desirable but challenging to attenuate those signals before the signal is fed into the ADC. An alternative is to implement digital filters which can achieve almost ideal filter topologies, like Butterworth, Chebyshev, brick wall etc. As with all electronics there are trade-offs and digital filtering is no exception. Digital filters produce excellent filters at the cost of acquisition-time, so does analog filters but when designed properly they can have a response time as low as in the  $\mu s$ -range.

The desired bandwidth is 2 kHz for the low-pass filter, which will pass the Lambda signal of interest.

Tustin's<sup>58</sup> method provide guidelines for sampling time, which should be  $5\omega_s < f_s < 100\omega_s$  where  $\omega_s$  is the signal bandwidth and  $f_s$  is the sampling frequency.

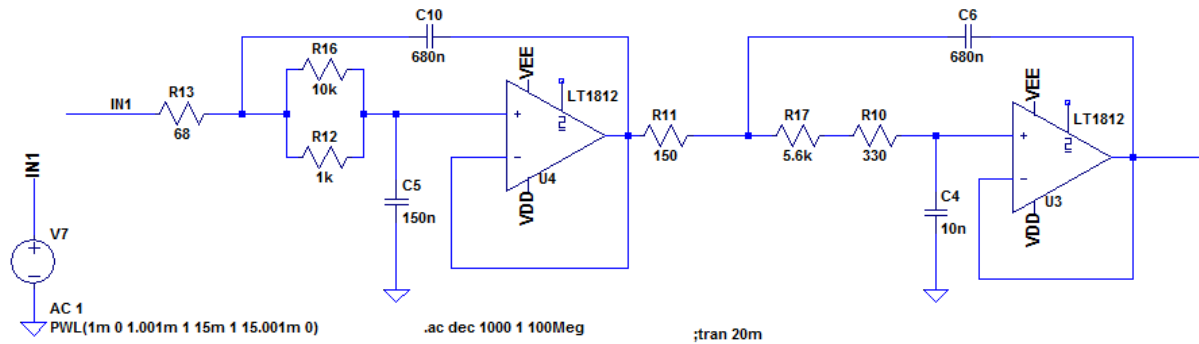
For good resolution we're sampling  $f_s = \omega_s \cdot 25 = 50 \text{ kHz}$ .

Below Nyquist is  $20 \text{ kHz} < \frac{f_s}{2}$  which will be the stopband frequency, and the attenuation of the filter will be at least  $20 \log 2^{-12} = -72 \text{ dB}$  at this frequency.

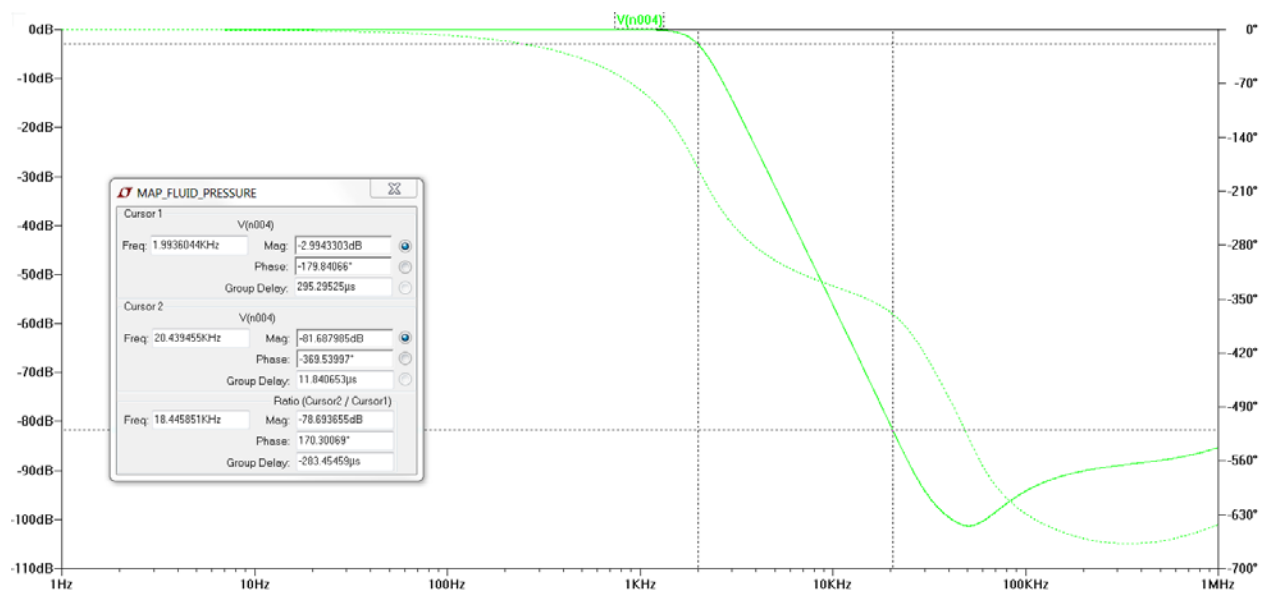
The resulting filter design and Bode plot shown in Figure 71 and Figure 72 respectively verifies that we are within the requirements of this certain filter, and should be able to avoid anti-aliasing and high frequency components in the sampled signal.

To cause continuity and reduce BOM, were reusing the components from the CPS filter, and using circuitry from the E12 standard. A general description on how we went about calculating the filter can be found in 3.7.4.4

<sup>58</sup> University of Trento - Sampling



**Figure 71 - 4<sup>th</sup> order Butterworth filter design with Sallen-key topology**

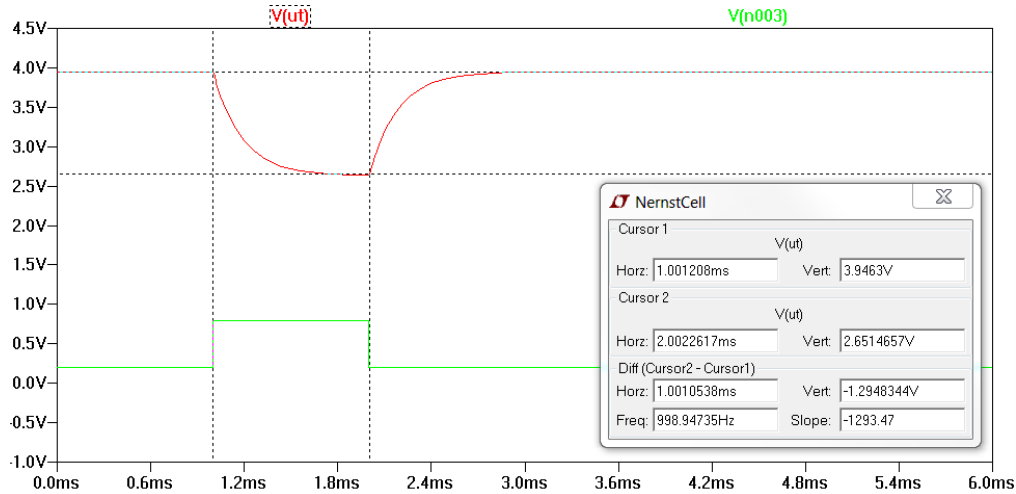


**Figure 72 - Bode-plot of Lambda filter**

The Bode-plot in Figure 72 verifies the filter requirements with a cut-off frequency of 2 kHz with -3 dB attenuation, and finally the stopband is found at 20 kHz with -81 dB attenuation, within the -72 dB requirement for this filter. The roll-off rate; 80 dB/decade is less than the roll-off rate of the CPS filter, as expected due to the 3<sup>rd</sup> order design of the Lambda filter.

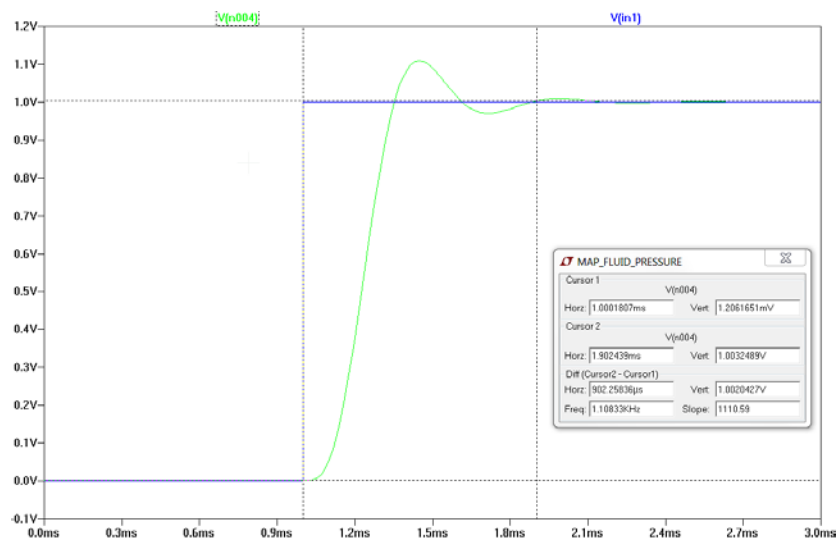
When measuring signals that are intended to provide feedback in a closed loop regulation, there is a time constraint as well. We have verified that frequency components of interest will pass the filter.

The XADC implemented in the Zynq SoC is more than capable of handling the signal, but there is no use for the Lambda filter being able to meet the acquisition time of the XADC, if the Lambda driver is considerably slower. Let's simulate a step response on the Lambda driver to find out in Figure 73.



**Figure 73 - Step response in Lambda driver circuit**

The simulation in Figure 73 verifies what we expected; the rise-time of the Lambda driver circuit is almost 1 ms, so if our low-pass filter has a better response we should be good.



**Figure 74 - Lambda filter step response**

Now in Figure 74 we can see that the filter responds to a step response in 902  $\mu$ s. In comparison with the Lambda driver circuit which has a response of 1 ms, the filter response time is sufficient. From this we can conclude that the filter will be quick enough to respond to the Lambda sensor.



### 3.8.2.6. General Purpose Analog Outputs

The quad DAC offers two unassigned outputs DACA\_2 and DACB\_2. To fulfil the requirement that analog outputs should be provided as signal for external devices, we have designed an amplifier that provides 0 - 13.6 V output. The operational amplifier LT1468<sup>59</sup> supports voltages in the range of  $\pm 15$  V, as well as it provides relatively high output current. The input to the amplifier from the DAC is in the range of 0 - 3.3 V. The input signal is amplified 4.1 times to the range of 0 - 13.6 V which is equal to a regulated  $V_{BAT}$  from the alternator. The circuitry is ESD protected and the schematic implementation is shown in Figure 75.

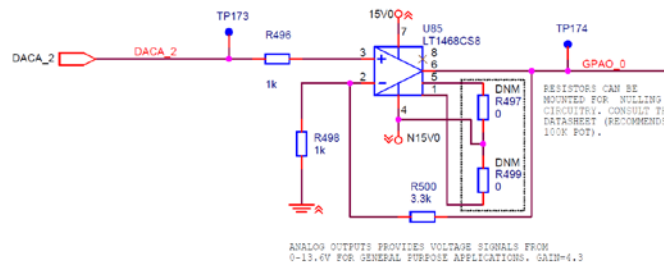


Figure 75 - Schematic implementation for analog outputs

### 3.8.2.7. System performance

To evaluate the total system performance of the Nernst cell, pump current and virtual ground circuitry combined, a simulation of the system is provided in Figure 76. This simulation is concerned with worst case simulation of the virtual ground part of the lambda sensor interface. The simulated loading effect is hence minimal.

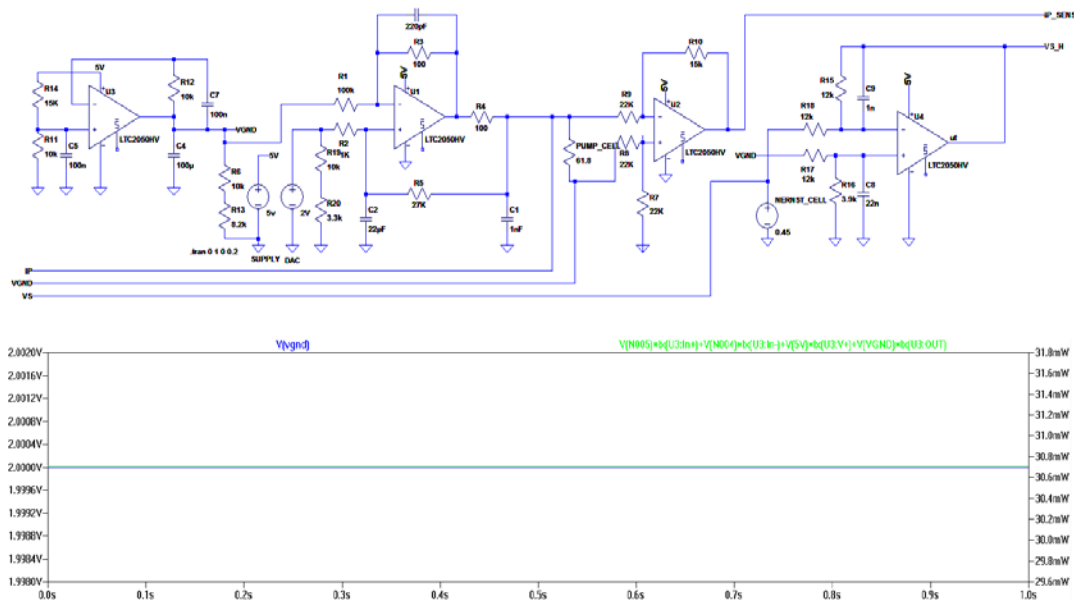
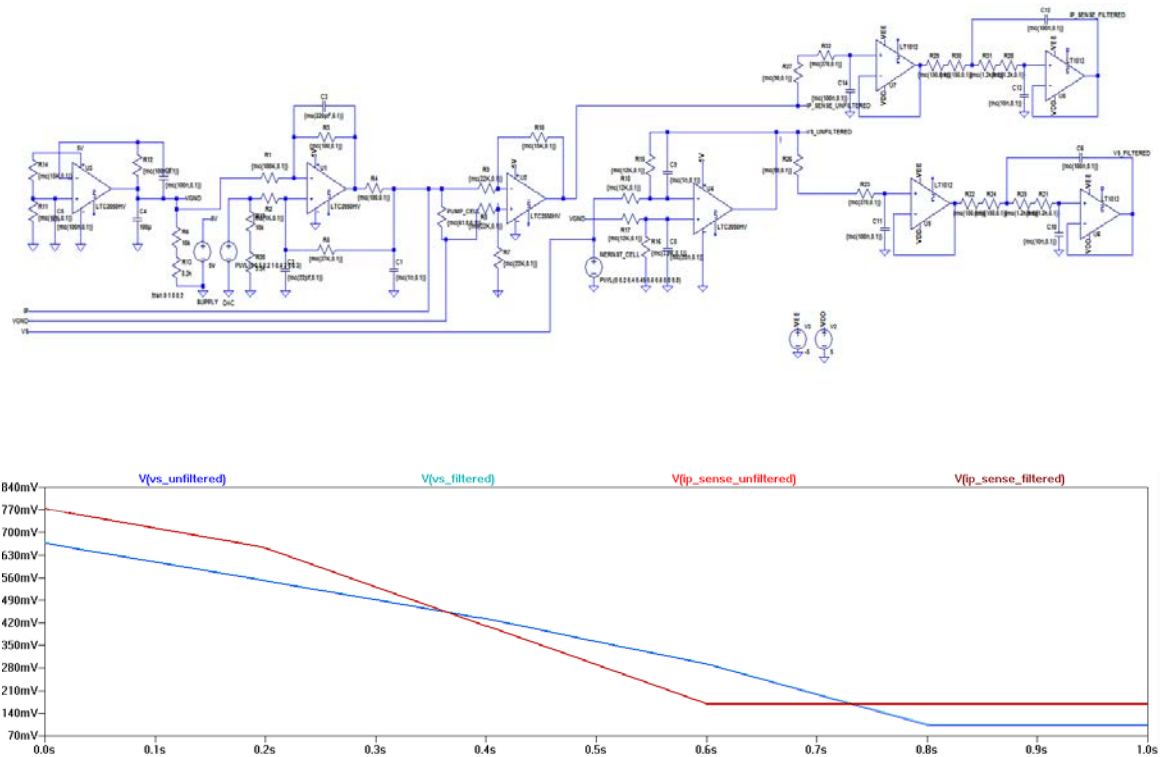


Figure 76 - Complete system simulation

<sup>59</sup> LT1468 Datasheet



The previous simulation considered worst case loading of charge buffer capacitor for virtual ground. This simulation proved that the capacitor withstands the loading effects of continuous regulation. To ensure that the circuitry works as intended with varying tolerances on the resistors and capacitors, a Monte Carlo analysis are performed. The Monte Carlo analysis utilizes statistical analysis of the circuitry by adjusting the circuit parameters to worst case values. For our discrete components, a worst case value is 10% with a nominal tolerance of 1% per component. The results are shown in Figure 77.



**Figure 77 - Monte Carlo simulation of complete system with 10% tolerances**

The simulation plots in Figure 77 show minimal deviation in the circuitry with 10% tolerances on all discrete components. This implies that the design withstands eventual discrete value changes due to thermal environmental changes. The bandwidth of the circuitry is therefore not limited by the multiple op-amp stages. The lambda sensor interface is implemented to the schematic design in the sensor hierarchy, as well as the driver hierarchy. The lambda sensor interface consists of one hierarchical block for each cylinder block. Refer to Bosch datasheet for look-up tables and other useful information.



### 3.9. Knock sensor interface

Engine knocking is a phenomenon that occurs when the fuel and air mixture in an engine cylinder detonates prematurely or during regular ignition. When the piston moves upwards in the cylinder, pressure increases. When the fuel/air mixture is ignited by the spark plug, the ignition forms a flame front in the cylinder, further increasing the overall pressure. At high pressures, the fuel/air mixture might self-ignite and cause detonation in another part of the cylinder. This causes audible sounds, hence the name engine knocking. Knocking occurs when ignition from the spark plug does not occur at the optimal point. Knocking in engines must be prevented as it can completely destroy the engine over time if not handled.

Engine knock is sensed by a piezoelectric sensor mounted directly to the engine block. The vibration that propagates from the detonation is picked up by the sensor and causes a generated voltage in the sensor due to the piezoelectric effect.

Knocking occurs on the frequency of 6.3 kHz, 10.4 kHz, 13.1 kHz, 14.4 kHz and between 18 - 19 kHz<sup>60</sup>. We are using a knock sensor from Bosch, the knock sensor KS-P<sup>61</sup>, which has a frequency range of 1 - 20 kHz.

To be able to interface with this sensor, we are using a custom IC that is used for knock interface. More about this IC can be read further down in chapter 3.9.1.

#### 3.9.1. TPIC8101

The TPIC8101<sup>62</sup> is a dual-channel knock sensor interface that has programmable gain and programmable band-pass filter. Furthermore the TPIC8101 can communicate with the TE0720 through SPI and it can also output an analog value if needed.

The principle of the TPIC8101 is as following:

The signal from the knock sensor is sent to an amplifier input, the gain on this stage can be set externally by choosing values of the resistors. The signal is then sent through an anti-aliasing filter to restrict the bandwidth to satisfy the sampling theorem see chapter 3.7. The signals are then transformed over to the digital world, using a 10 bit ADC with a maximum sampling frequency of 200 kHz. The digital approximation is then sent through the internal gain stage. The gain stage setting is selectable up to 64 values from 0.111 to 2.0. The signal is then filtered with a programmable band-pass filter (1.22-19.98 kHz); this is to extract frequencies of interest and to filter out any noise from the engine and other noise sources. The output from the band-pass filter is then full-wave rectified and then integrated using an integrator time constant by the SPI and integration time. The integrated signal is then fed to the TE0720 and the data is

---

<sup>60</sup> SAE technical paper series 2000-01-0932

<sup>61</sup> Bosch KS-P datasheet

<sup>62</sup> TPIC8101 datasheet



then used to adjust the ignition timing. The principle of TPIC8101 is illustrated in Figure 78.

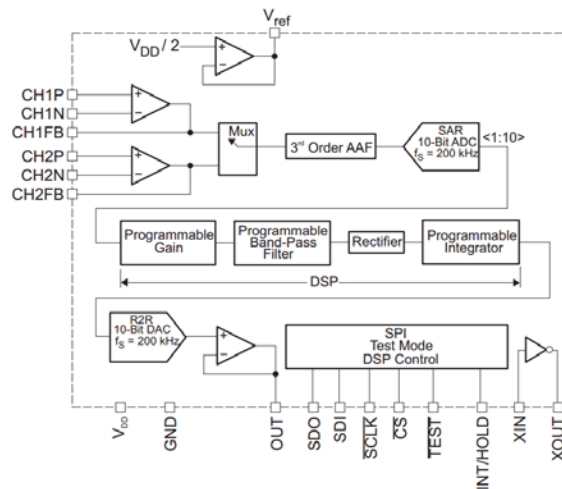


Figure 78 - Simplified schematic of TPIC8101<sup>63</sup>

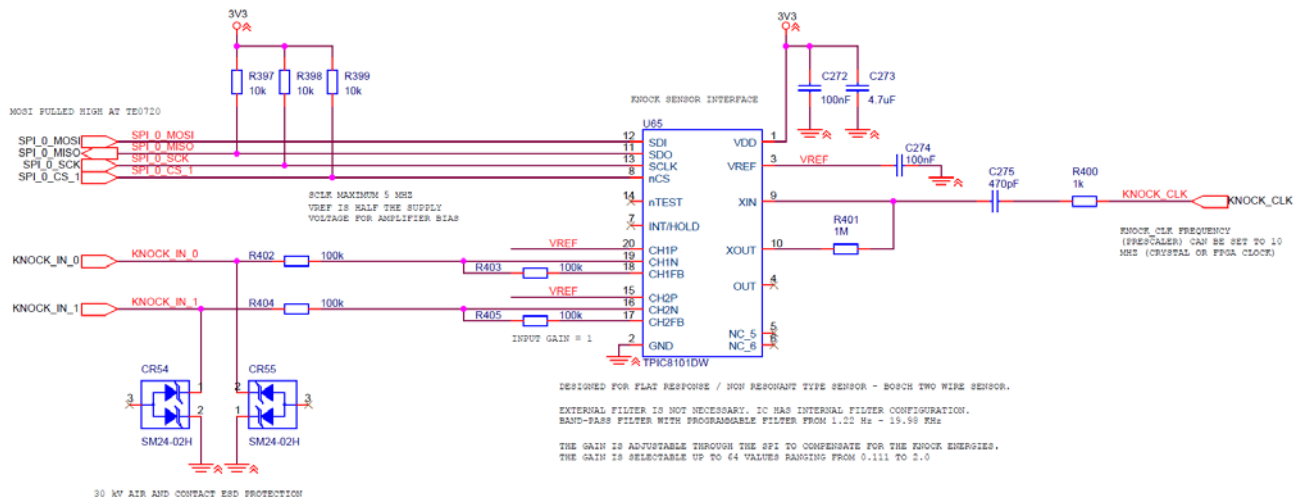


Figure 79 - Knock sensor interface

As seen from Figure 79, the input is protected from ESD using TVS diodes, the TVS is rated with 30 kV air and contact discharge. The supply voltage is decoupled with a 100 nF and 4.6 μF capacitors. The external gain is set by the 4 external resistors to give a gain of 1. By choosing values of resistors to unity gain, allows the gain to be set internally. Figure 79 shows the finished schematic for the knock sensor interface. The clock signal from the TE0720 is used to set the internal sampling frequency of the ADC.

<sup>63</sup> Figure from TPC8101 datasheet



Figure 78 shows a simplified schematic of the TPIC8101, where CH1P/N and CH2P/N is the input for the piezoelectric sensors and CH1/2FB is the feedback pin of the input amplifier. VDD is the supply voltage for the device and GND is the ground pin. VREF is the internal reference voltage that is set by a capacitor connected to ground. The out pin is the analog output of the integrated signal if an analog output is used. The INT/HOLD pin selects if the senses signal is integrated or to hold for data acquisition. SDI and SDO is the serial input/output for the SPI bus and SCLK is the SPI clock. The TEST pin is used to activate test mode and is active low. XIN and XOUT are the input and output of the oscillator which is coming from the ECU. A more detailed pin description is found in Table 17.

**Table 17 - Pin configuration TPIC8101**

PIN		
NAME	NO.	DESCRIPTION
VDD	1	5 V input supply
GND	2	Ground connection
VREF	3	Supply reference generated with external bypass capacitor
OUT	4	Analog output
NC	5	No connect
	6	
INT/HOLD	7	Selectable for integrate (High) or hold (low) mode
nCS	8	Chip select for SPI communication
XIN	9	Inverter input for oscillator
XOUT	10	Inverter output for oscillator
SDO	11	Serial data output for SPI
SDI	12	Serial data input for SPI
SCLK	13	SPI clock
nTEST	14	Test mode (active low), open for normal operation
CH2P	15	Positive input for amplifier 2
CH2N	16	Negative input for amplifier 2
CH2FB	17	Output of amplifier 2, for feedback
CH1FB	18	Output of amplifier 1, for feedback
CH1N	19	Positive input for amplifier 1
CH1p	20	Negative input for amplifier 1

As mentioned above, the communication interface for the TPIC8101 knock sensor interface is SPI. The SPI is connected to SPI\_0 bus, which is a shared bus with the LTC2983 temperature sensor interface. The TPIC8101 is connected to CS\_1 on SPI\_0 bus.





### 3.10. Digital interface

This section provides design descriptions regarding interfacing towards digital sensors. These are the flex fuel sensor (ethanol content), ignition signal, brake pedal circuitry, mass air flow and general purpose digital inputs.

#### 3.10.1. Flex fuel sensor

Ethanol fuel is currently being used in high performance motorsports applications due to increased performance when used in race engines. A combustion engine can run on clean ethanol, but requires high customization of internal mechanical components as well as actuators to handle the higher octane rates that ethanol provides. For practical purposes, this translates to extensive wear on sealing rings and gaskets, due to the absence of oil (which petroleum is a product of).

To combine these two properties for ethanol and petrol, a synthetic blend is provided commonly known as E85. The E85-term means that 85% of the fuel is ethanol and 15% are petrol; this increases the octane rating of the fuel to approximately 110 octane (98 is normal rating). Increase of octane level is beneficial in engine tuning due to the fact that the flame front of the combustion process travels at a higher velocity, and increases performance by 30%<sup>64</sup>. Ethanol fuel cools the combustion chamber, and is more environmentally friendly due to cleaner combustion than petroleum based fuel.

Due to the higher octane rating, and the swifter combustion process, the ignition timing needs to be provided in advance to ensure that the engine is running most efficiently (heat and performance). E85 has good ability to suppress detonation, which means that knocking occurs more controlled, and a more aggressive ECU-program can be applied. The timing must be advanced in increments from 1° to 13° BTDC, to ensure that maximum performance is obtained. The stoichiometric air to fuel ratio is obtained at 9.7: 1, and this implies that the engine requires 50% more fuel when running on E85 compared to petrol.

The ECU must be able to determine the amount of ethanol in the fuel provided to the engine, to correct the ignition timing and fuel basic table to ensure maximum performance and efficiency. There are multiple vendors that offer digital flex fuel sensors which provide the ECU with information regarding the ethanol content in the fuel. GM/Continental delivers a sensor (PN 13577379) that is well documented for similar projects, which is beneficial to interface for this ECU. The mechanical specifications for this sensor are given in Table 18.

---

<sup>64</sup> Hi Octane Racing 2015

**Table 18 - Specifications for the sensor<sup>65</sup>.**

Specification	Value
Measuring range	0-100% (ethanol content)
Sensor accuracy	5%
Max temperature error	1.5%
Output characteristic	Linear
Operating temperature	Environment -40°C → +125°C, Fuel -40°C → +125°C
Maximum fuel pressure	10 bar (0.1 bar)
Drop in sensor	0.1 bar
Maximum flow	200 l/h
Signal output voltage	5 V
Supply voltage	6 - 18 V
Response time	< 250 ms

Due to the relative low maximum rates for fuel pressure makes this sensor beneficial to install in the fuel return line after the fuel rail (low fuel pressure), in which the temperature of the fuel can be measured for maximum safety. The flex fuel sensor is a digital type which has battery and ground supply voltages, and an output signal that the signal description is given by Table 19.

**Table 19 - Flex fuel sensor signal description**

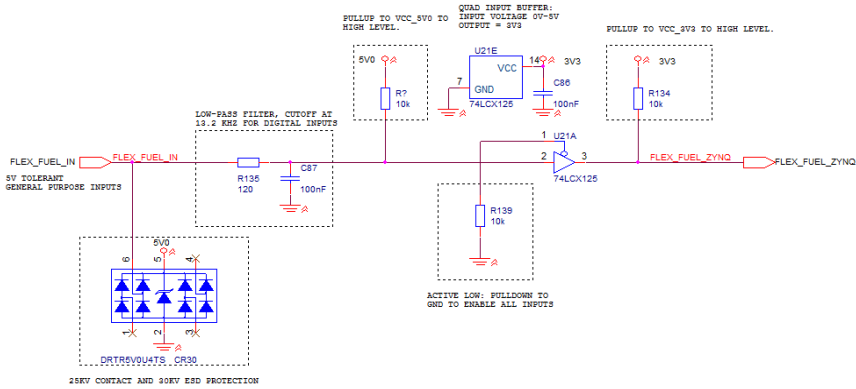
Description	Electrical characteristics	Practical range
Ethanol content	Frequency: 50 Hz to 150 Hz	0 % to 100 %
Fuel temperature	Period: 1 ms to 5 ms	-40°C to 125°C

### 3.10.1.1. Schematic design

Table 19 implies that there are two quantities that need to be measured; frequency and period of the signal. The output voltage is 5 V, with varying frequency and period. Due to the fact that the Zynq module operates on 3.3 V, there must be a level shifting from 5 V to 3.3 V. The options for performing this level shifting are to use an optocoupled device or a buffer to perform this task. Due to the fact that buffers for the same type of signals are used for interfacing hall-effect sensors, it's beneficial to continue the use of this buffer, the 74LCX125<sup>66</sup>, which is shown in Figure 80.

<sup>65</sup> Zeitronix Ethanol Content Analyzer product page

<sup>66</sup> 74LCX125 Datasheet

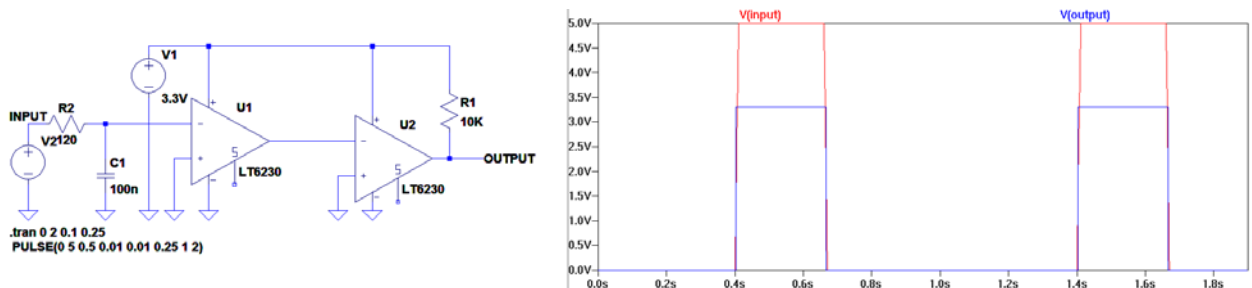


**Figure 80 - Flex fuel interface circuitry**

The buffer 74LCX125<sup>67</sup> requires a low signal on the pin number 1 to activate the output of the buffer. To avoid using multiple I/O on the Zynq to activate the buffer, a pull down resistor to ground makes the buffer activated without the need of any external activation signals. The sensor is designed as an open collector output, which yields that a pull up resistor (10 kΩ) to VCC is required. The output signal of the buffer circuit is therefore level shifted to 3.3V, with varying frequency and period. The low pass filter on the input filters the high frequent noise (above 13.2 kHz) that is typically generated in automotive environments; the cut-off frequency is defined by (53).

$$F_c = \frac{1}{2\pi RC} = \frac{1}{2\pi \cdot 120 \Omega \cdot 100 \text{ nF}} = 13.2 \text{ kHz} \quad (53)$$

A model of the circuit are designed in LTSpice with two equivalent rail to rail operational amplifiers (LT6230) that provides the level shifting action, and inverts the output signal to a voltage level that the TE0720 is able to read. Figure 81 shows the model of the circuit, and the simulation performed with a PWM signal as input. This circuitry is equivalent to the quad buffer used, and therefore verifies that it works as intended.



**Figure 81 - Model of buffer circuit with low pass filter.**

<sup>67</sup> Fairchild semiconductor 2013, 74LCX125



The parameters of the sensor is described in Table 18; the frequency of the signal determines the ethanol percentage between 50 - 150 Hz. (0-100%). The pulse width determines the fuel temperature between -40 °C and +125 °C (1 - 5 ms). The signals must be handled in **FPGA** as two separate variables that can be used as a compensation for the ignition timing. The timing should be advanced by 13° BTDC to ensure that maximum performance and efficiency is obtained with 100% ethanol<sup>68</sup>. This gives a compensation factor as shown in (54)

$$FFCOMP = \frac{-13^\circ - 0^\circ}{100\% - 0\%} = -0.13^\circ/\% \quad (54)$$

This implies that when the engine is running on E85 (currently most used), the ignition sequence from cylinder 1 should be advanced by -11.05° (BTDC). There is electronics inside the flex fuel sensor that can utilize self-diagnostics to ensure that the sensor is functioning as expected. The sensor will give an output frequency of 170 Hz when there is an internal fault, or the fuel is polluted (oil, water, etc.), which must be taken into account in software.

The temperature information from the sensor is obtained by computing the reciprocal of the frequency, and this information is useful for safety purposes. When the engine is stationary in a hot environment, the fuel temperature returning from the engine can increase to its auto ignition temperature; this implies that the user should receive a warning when the temperature rises above 95 °C. The input pin of the circuitry is ESD protected according to the requirements.

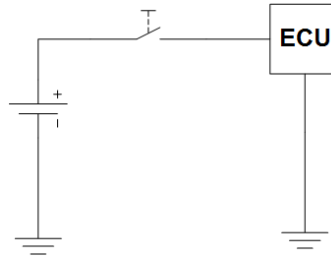
---

<sup>68</sup> Megasquirt 2015, flexfuel



### 3.10.2. Ignition signal circuitry

The user needs to activate the ECU to ensure that the engine should start or stop according to the user's request. Due to the fact that the motorsports solution today utilizes an ignition key lock to activate or deactivate the ignition, the interface towards this transducer should be considered as a manual switch configuration, for example a single-pole, single-throw switch as shown in Figure 82.



**Figure 82 - Basic diagram ignition signal**

The circuitry after the ignition switch should also activate a main relay which supplies the actuators with positive supply voltage. Between the battery and the switch, there should be a fuse which protects the circuitry. The battery voltage can fluctuate severely due to activation of starter engine (high load), and can therefore fall to 6 V at the lowest, and 16 V at the most (overloading generator). The inductive loads in the vehicle (ignition coils, fuel injectors and solenoids) as well as the generators induces high frequency noise which makes this circuitry non trivial to implement in our design.

The ignition signal needs to be level shifted from battery voltage to 3.3 V which the Zynq is able to read as logic 0 or 1. When the ignition signal is activated, the output circuitry which controls the ignition coils and injectors must be activated. There are multiple ways to interface the ignition signal, but to protect the Zynq against transients and highly fluctuating voltages, an optocoupler is ideal. There are only two states of this circuitry (on or off), so no information will be clocked through the optocoupler, and hence limited wear will occur.

The optocoupler CNY17-3 from Vishay contains an input diode and a phototransistor with base connection on the output. To define an activation threshold, a 5.1 V Zener diode is implemented to avoid unwanted activations by bouncing or floating inputs. The input should be protected with a resistor to ensure that the forward current is limited. According to Ohm's law, the limit resistor can be determined for worst case input voltage by (55) ( $I_F$  is 4 mA for 1.05 V according to graph in datasheet<sup>69</sup>).

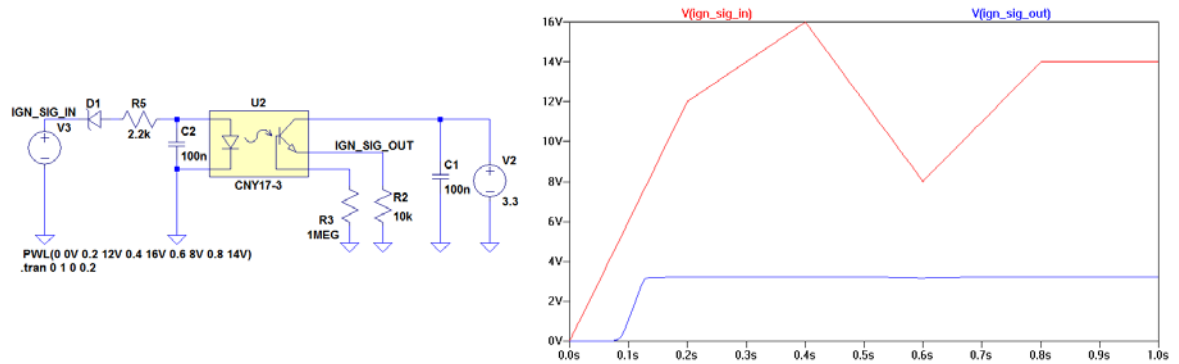
<sup>69</sup> Vishay semiconductors 2014, CNY17-3



$$R_{IN} = \frac{V_{BATT} - V_Z - V_F}{I_F} \approx 2.2 \text{ k}\Omega \quad (55)$$

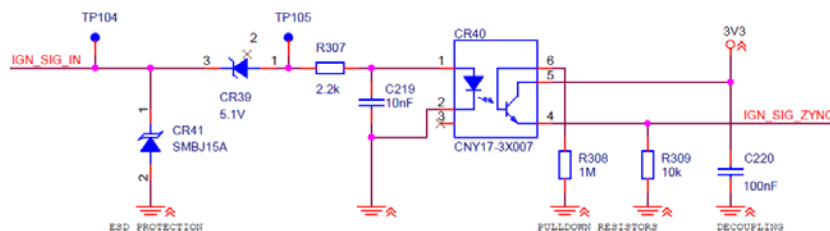
The output of the circuitry utilizes the internal voltage source (3.3 V) which will be sinked to ground when the diode is conducting. This implies that there are two ways to design the output circuitry; active high or active low by the placement of the resistor (Emitter or Collector biased).

To ensure that maximum safety is obtained, the active high is the best solution (Emitter resistor). The worst case scenario is if the optocoupler fails, and the internal ignition signal is left low when the configuration is active low (the engine will still run). The output of the circuit is pulled down to ground by a 10 kΩ resistor that is recommended for this application. A simulation is performed in Figure 83 to verify the functionalities.



**Figure 83 - Simulation of ignition interface circuit**

The simulation performed in Figure 83 verifies the functionality which proves that the circuit works as intended for the voltage range between 6 - 16 V. The capacitors acts as decoupling for the optocoupler, and the 1 MΩ resistor pulls the base input of the phototransistor to ground. The circuitry is realized in Figure 84, including ESD protection that protects the circuit according to the requirements.

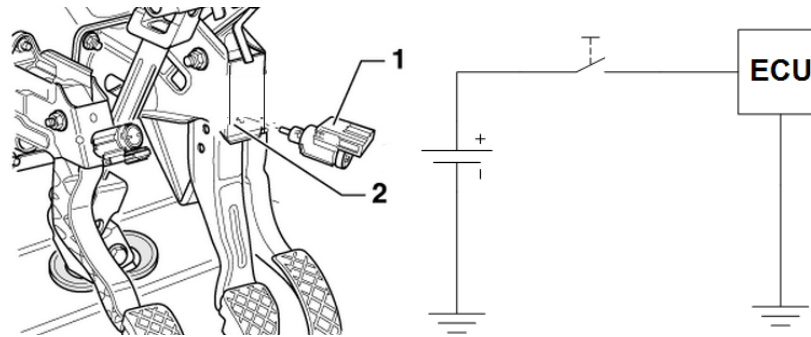


**Figure 84 - Ignition signal circuitry**



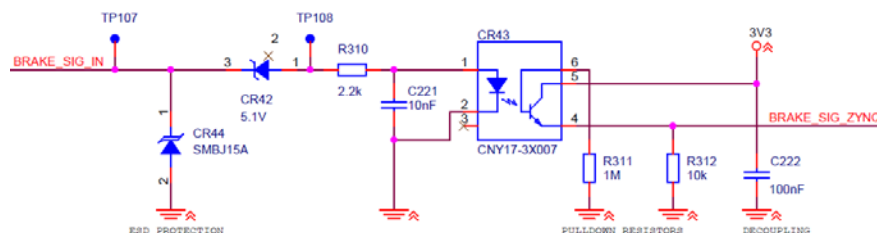
### 3.10.3. Brake pedal sensor circuitry

The brake pedal signal acts as a safety barrier if unintended behaviour of the engine occurs. This unintended behaviour includes engine rev-up when stationary, as well as acceleration when driver request coasting or deceleration. When the KPEC ECU is tested with software, any unintended behaviour should have an emergency switch that disables the ignition and injection systems. In this debugging mode, an active Boolean value from the brake pedal sensor should shut down the entire system. Typical mounting location and equivalent circuitry are shown in Figure 85



**Figure 85 - Mounting location and basic equivalent circuitry**

The brake pedal depressed signal has equivalent characteristics to the ignition signal, where the battery voltage can fluctuate and contain a lot of noise. To avoid increasing the BOM, a circuitry that is equal to the ignition signal circuitry is duplicated with changed net aliases. The output signal of this circuitry is 3.3 V when the brake pedal is depressed, as shown in Figure 86. The input signal should be fused to avoid overloading the circuitries, simulations of the circuitry is performed in Figure 83.



**Figure 86 - Brake pedal depressed signal circuitry**



#### 3.10.4. Manifold Air Flow (MAF) sensor

The mass airflow sensor is used to measure the flow rate of the air that is entering the engine. While the MAP sensor can also be used to determine the mass airflow rate to the engine, a MAF sensor is used to provide redundancy. The MAF sensor is mounted right behind the air filter, while the MAP is inside the air manifold. The physical airflow measuring unit is provided in terms as kg/hour where 20 kg/hour is a normal air flow value at idle (approximately 850 RPM).

In cars with turbo the pressure after the turbo is much higher, thus the MAF and MAP has different values for pressure (Pressure can be derived from airflow). In cars without turbo, both MAF and MAP are not needed. Traditional both sensors are installed to acts like a redundancy if one sensor fails.

There are two types of mass airflow sensor; these are the vane meter and the hot wire. We are interfacing a hot wire from Bosch, the HFM 6<sup>70</sup>, which outputs a digital signal. More precisely a PWM signal with frequency proportional to the mass airflow rate entering the engine.

The PWM signal that the HFM 6 outputs has is signal that is 0 - 5 V. We need to level shift this to 0 - 3.3 V so that the ECU can handle the information and to the required calculations. The signal is first filtered at a cut-off frequency at 13.2 kHz and then level shifted to 0 - 3.3 V. A similar circuitry to the flex fuel sensor interface is therefore duplicated to achieve this functionality as shown in Figure 87.

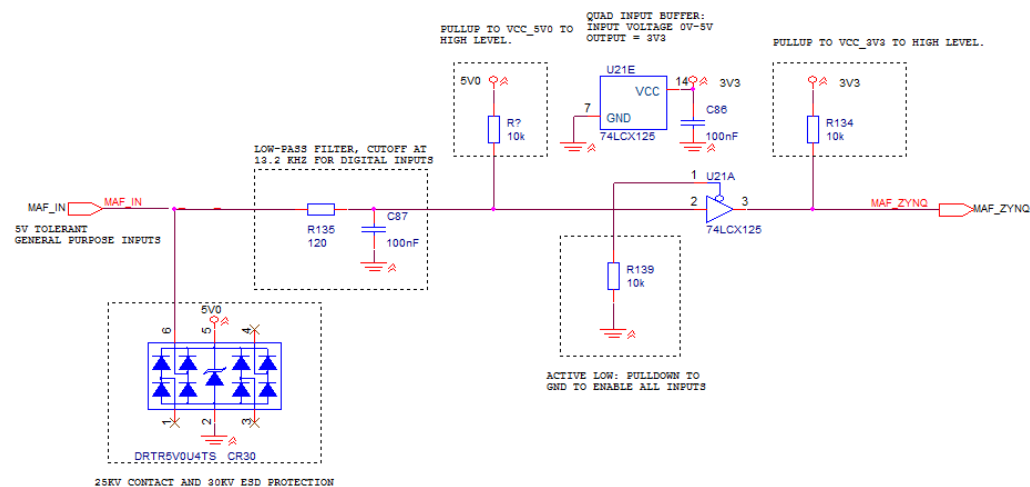


Figure 87 - Schematic implementation of MAF sensor

<sup>70</sup> Mass airflow sensor, HFM 6 datasheet

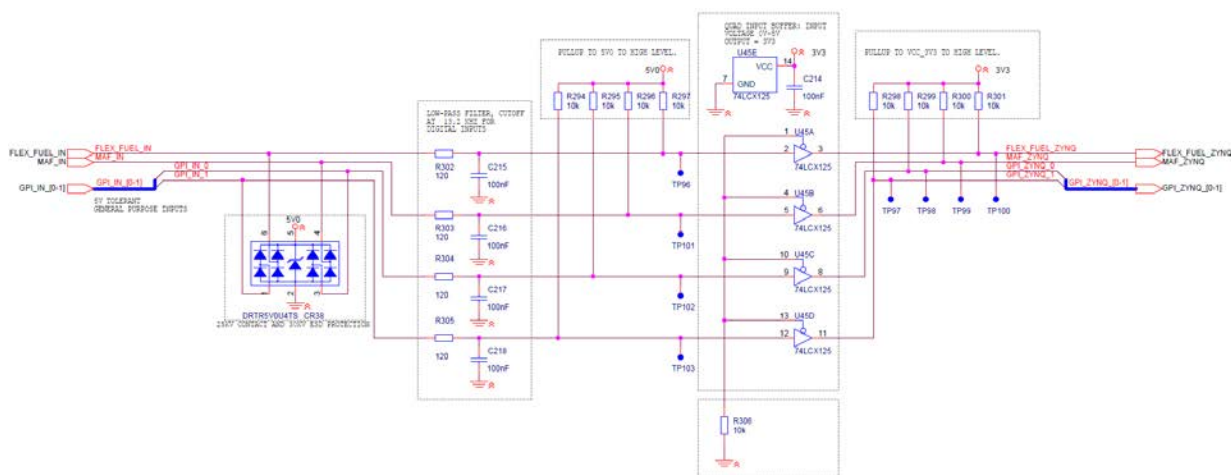




### 3.10.5. General purpose digital inputs

The general purpose inputs are based on the flex fuel and hall-effect sensor interface, to be able to read logic levels (digital) from external devices. These inputs can increase the functionality of the ECU after deployment if necessary. The filter is designed with cut-off frequency at 13.2 kHz which makes this inputs versatile for 0 - 5 V reading of different sensors or switches.

Figure 88 combines the general purpose and flex fuel interface, and utilizes the remaining two inputs to the 74LCX125. The ESD-protection circuitry protects the buffer from electrostatic discharge according to the requirements.



### Figure 88 - General purpose inputs and flex fuel sensor interface



### 3.11. Battery monitor

Battery voltage of the system can be an important factor to measure as this makes the ECU able to detect faults in the alternator charging the battery, or faults with the battery itself. Several parts of an alternator can fail. If the regulator controlling the charging voltage fails, the charging voltage can reach levels much higher than what the car battery and components are rated for, causing damage. Another scenario is that the alternator does not charge the battery at all, causing the battery to drain completely until unusable. The battery itself can also fail and lose its ability to hold charge. These events can be detected and reported with the help of a small monitoring circuit in the ECU. The battery voltage is constantly varying, and the range specified as functional for the KPEC ECU is a battery voltage ranging from 6 to 19 V. If outside these limits, the KPEC ECU will not function.

#### 3.11.1. Implementation of the battery monitor

As the battery voltage is an analog DC signal, the voltage must be read by an ADC. For measuring battery voltage, the internal 12 bit XADC on the Zynq SoC on the TE0720 is used, however, the XADC only supports a voltage input range of 0 - 1 V, while the specified battery range is 6 - 19 V. This requirement makes a voltage translator required. For this, a difference amplifier can be used.

In order not to exceed the maximum input voltage of the LTC2050HV operational amplifier, the input voltage is divided by a simple voltage divider. The maximum input voltage is defined as 19 V.

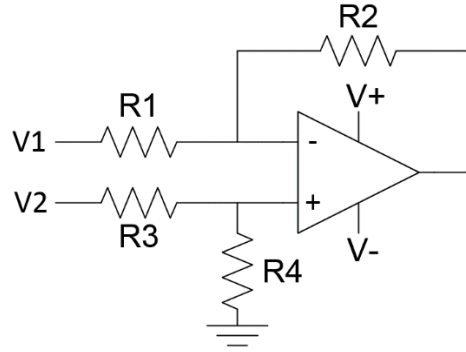
The voltage divider formula is given in (56)

$$V_{out} = \frac{R_2}{R_1 + R_2} V_{in} \quad (56)$$

In order not to exceed the rail voltage of the XADC, 900 mV maximum input was selected in order to allow some headroom for component tolerances.

The resistor value ratio is found to be 0.2742 by rearranging (56) and inserting values for maximum in- and output voltage. The closest E12 resistor series values that gives this ratio is:

$$\begin{aligned} R_1 &= 18 \text{ k}\Omega \\ R_2 &= 6.8 \text{ k}\Omega \end{aligned} \quad (57)$$



**Figure 89 - Difference amplifier**

The circuit for a difference amplifier is shown in Figure 89. The equation for calculating the voltage output of the difference amplifier is given in (58)

$$V_{out} = \left( \frac{R_1 + R_2}{R_3 + R_4} \right) \frac{R_4}{R_1} V_2 - \frac{R_2}{R_1} V_1 \quad (58)$$

In the case where  $R_1 = R_3$  and  $R_2 = R_4$ , (58) can be reduced to (59)

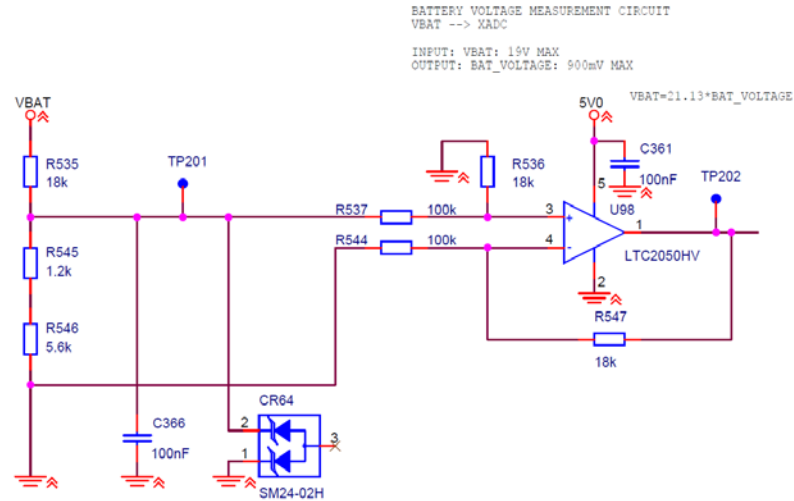
$$V_{out} = \frac{R_2}{R_1} (V_2 - V_1) \quad (59)$$

The maximum input to the amplifier was deduced to be approximately 5 V. In order to get a maximum output of 900 mV to the XADC, the following gain is calculated:

$$A_v = \frac{V_{out}}{V_{in}} = \frac{0.9 \text{ V}}{5 \text{ V}} = 0.18 \quad (60)$$

We reference the amplifier input  $V_1$  to ground (0 V), thus the gain of the amplifier is directly multiplied with the input voltage  $V_2$ . Standard E12 resistor series that matches the resistor ratio of 0.18 is:

$$\begin{aligned} R_1 &= 18 \text{ k}\Omega \\ R_2 &= 100 \text{ k}\Omega \end{aligned} \quad (61)$$

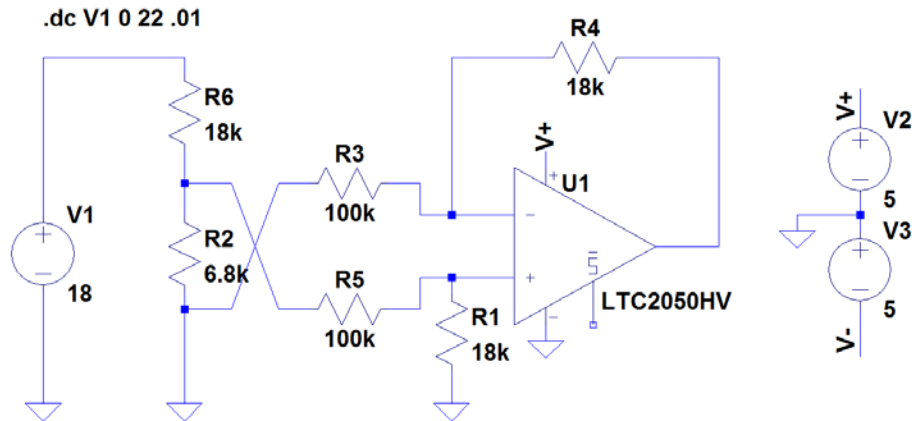


**Figure 90 - Battery monitor voltage translator circuit**

Figure 90 shows the schematic implementation of the voltage translator. The LTC2050HV operational amplifier was picked due to its use elsewhere in the design. The clamping diode is to ensure that the amplifier input does not exceed 5 V in case of input voltages above 19 V

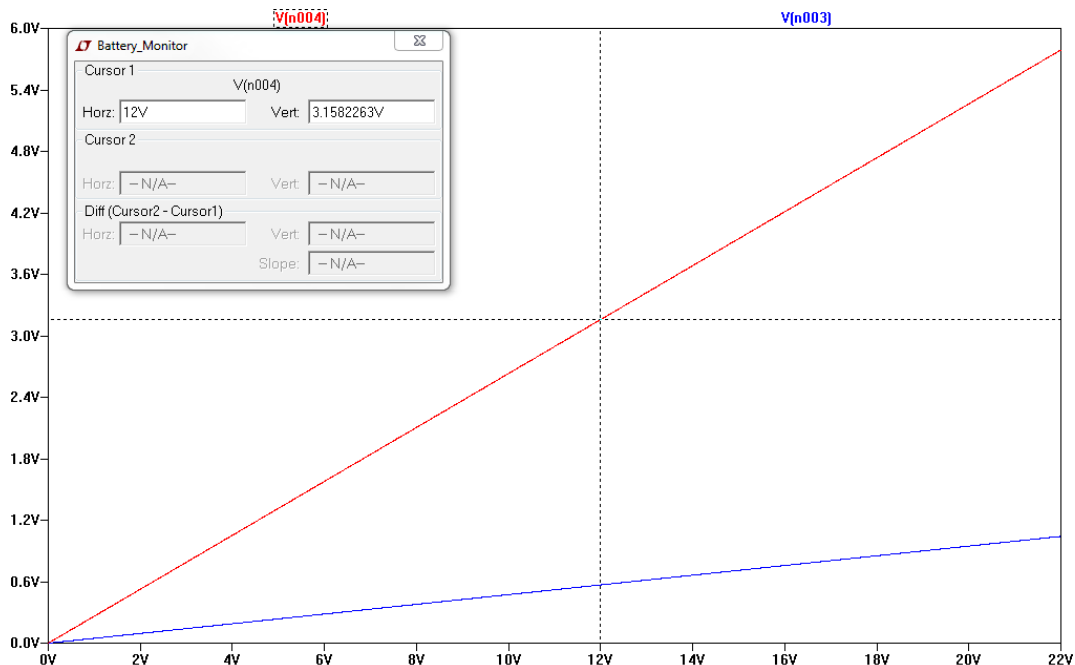
### 3.11.2. Simulation of the battery monitor

In order to verify that the circuit functions as calculated in 3.11.1, the circuit is simulated using LTSpice.



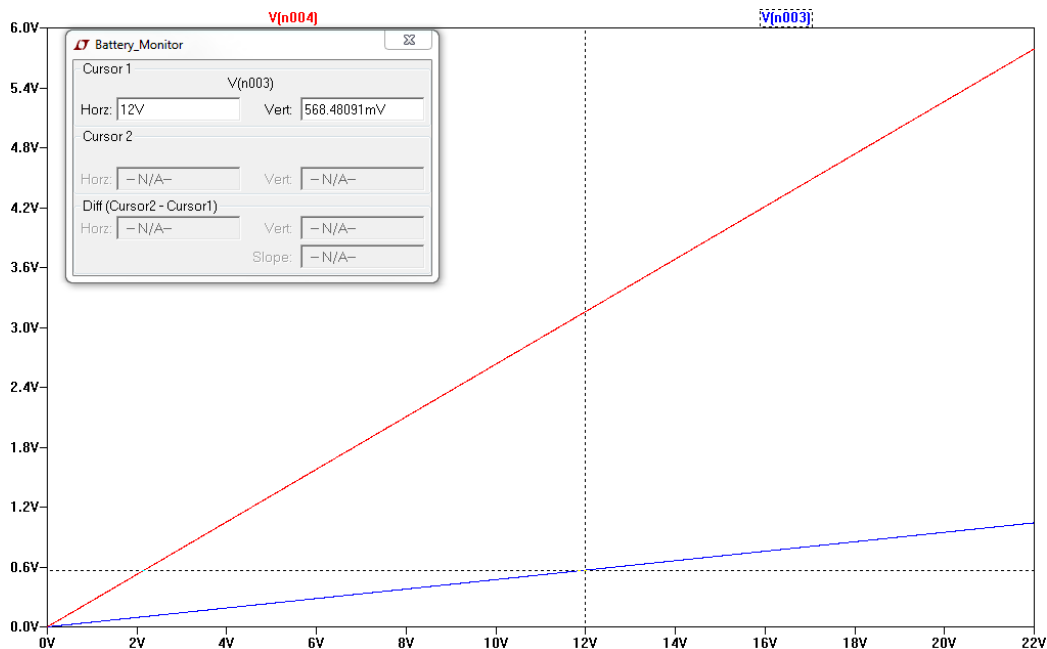
**Figure 91 - LTSpice schematic drawing**

Figure 91 shows the circuit drawn in LTSpice. The voltage source V1 represents the battery voltage, and was swept with 0 - 22 V. The nominal ideal battery value for a car battery is 12 V, but due to charge level and loading, this is not a true figure.



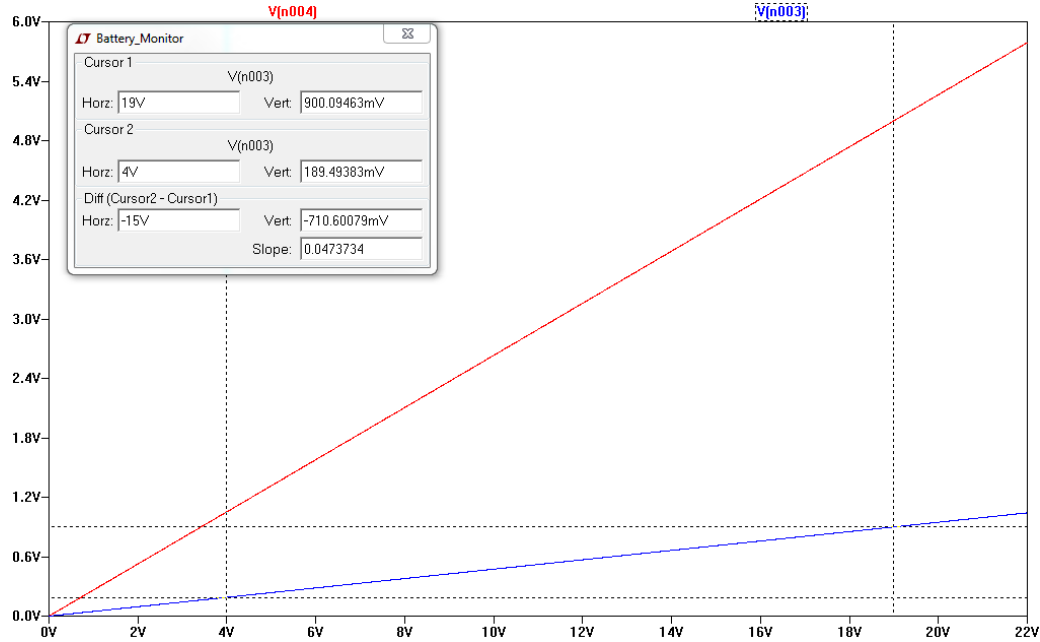
**Figure 92 - Amplifier nominal input**

Figure 92 shows the input of the amplifier with a nominal battery voltage of 12 V. The voltage is at approximately 3.16 V, verifying that the voltage divider circuit functions as intended.



**Figure 93 - Amplifier nominal output**

Figure 93 shows the nominal output of the amplifier at 12 V battery voltage. The nominal output voltage read by the XADC is approximately 568.5 mV.



**Figure 94 - Minimum and maximum values**

Finally, Figure 94 shows the output voltage levels for the defined minimum and maximum levels. For a battery voltage of 19 V, the amplifier output is 900 mV. For the minimum battery voltage of 6 V, the output is approximately 189.5 mV.

There is a clear linear relationship between the in- and output voltages of the circuit, which makes it easy to calculate battery voltage from measured XADC values. A formula for calculating the battery voltage from the numerical XADC output can be deduced.

With values measured in Figure 93, the relationship between battery voltage and amplifier output voltage is:

$$\frac{V_{BAT}}{V_{out}} = \frac{12 \text{ V}}{0.568 \text{ V}} \approx 21.13 \quad (62)$$

The voltage measured by the XADC is:

$$V_{ADC} = \frac{N}{2^{12}} \quad (63)$$



As the XADC 12 bits and  $N$  is the number output by the XADC converted to a decimal value.

As the output of the amplifier is the same as the XADC input,  $V_{out} = V_{ADC}$ , combining (62) and (42) yields

$$V_{BAT} = 21.13 \cdot \frac{N}{2^{12}} \quad (64)$$



### 3.12. Accelerator pedal

The accelerator pedal is used by the driver to regulate how much torque from the engine that is given for a given angular displacement of the pedal. A general accelerator from Bosch consists of an accelerator pedal and a potentiometer or a non-contact hall sensor, which acts as an angular positioning sensor. When the pedal is pressed down, the sensors gather information about position and movement of the pedal. This information is then used to calculate the amount of torque and accordingly adjust the throttle device and injection system.<sup>71</sup>

The interface for the accelerator pedal is exactly the same as the implementation for battery monitor. For implementation details see 3.11.1. The only different in the implementation of the accelerator pedal from the battery monitor, is that there are no voltage divider on the input, although the option to change voltage divider ratio, is implemented by using two 0  $\Omega$  resistors in a voltage divider network, where one is not mounted.

With this a relationship between the accelerator pedal output voltage and amplifier output voltage is:

$$\frac{V_{ACC}}{V_{out}} = \frac{5\text{ V}}{0.9\text{ V}} = 5.56 \quad (65)$$

The voltage measured by the XADC is:

$$V_{ADC} = \frac{N}{2^{12}} \quad (66)$$

As the XADC 12 bits and N is the number output by the XADC converted to a decimal value.

As the output of the amplifier is the same as the XADC input  $V_{out} = V_{ADC}$ , combining (65) and (66) yields

$$V_{ACC} = 5.56 \cdot \frac{N}{2^{12}} \quad (67)$$

---

<sup>71</sup> Bosch, Accelerator-pedal module





## 4. Drivers

This section is dedicated towards the topic of the actuator driver circuitry. These drivers receive a command and then activate the actuators. A standard in the automotive industry is to use a low-side switch to activate the actuator, and this is a standard KPEC is following. The low-side switching method consists of having the switch between the device and ground; this is illustrated in Figure 95.



Figure 95 - Low-side topology

### 4.1. Ignition coil drive

As discussed and agreed upon with KDA, we are going to interface Bosch array of actuators and sensors. A demand from our employers is that we need to interface with an engine that can have a max speed of 16.000 RPM. With this requirement, we needed to find an ignition coil that can handle such speeds. We searched through Bosch arrays of actuators and managed to find a single ignition coil that could handle such extreme speeds. This is the Single Fire Coil S19<sup>72</sup>.

To be able to accommodate this sensor we need to find a low side switch driver which either is a MOSFET or an IGBT.

The MOSFET is preferred in high frequency applications (>200 kHz), wide line or load variations, long duty cycles, low voltage applications (<250 V) and low power applications (<500 W output power). The IGBT is preferred in low frequency applications (<20 kHz), low duty cycle, narrow or small line or load variations, high voltage applications (> 1000 V) and high power applications (>5 kW).

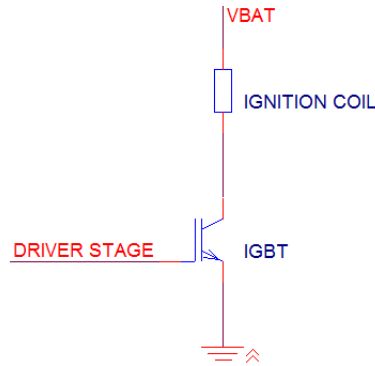
The ignition coil driver is divided into two parts, the power stage and the driver stage. The power stage will be discussed in 4.1.1. The driver stage will be discussed in 4.1.2.

<sup>72</sup> Single fire coil S19 datasheet



#### 4.1.1. Power stage

In the ignition application we require a switch that can handle great breakdown voltages (300 - 600 V) that has a low duty cycle and that can handle low frequency switching; this is the main reason for why selecting an IGBT. A simplified block diagram of the ignition coil is illustrated in Figure 96.



**Figure 96 - Simplified block diagram of ignition coil**

We found out that there were two options: 1) Choose a standalone IGBT and design the protection circuitry around it, or 2) Choose an IGBT that is special purpose made for ignition applications.

In the datasheet for Single Fire Coil S19, Bosch recommended that we use the IGBT IRG4BC40S<sup>73</sup>. This device is a standalone component and does not offer any means to limit the spike voltage that can damage the IGBT and the coil isolation. Furthermore the device does not have any ESD protection circuitry. This means that we have to design our own protection circuitry around the IGBT. This is time consuming process which we do not have resources for and that's why we opted for option 2, an IGBT that is custom made for this specific application.

We decided to go for the device ISL9V5045S3ST<sup>74</sup>, we went for this device because it has the highest breakdown voltage and it could handle the current requirement (25 A) that the coil ignition is rated for. This device also offer us the opportunity to reduce the spike voltage to a voltage that will not harm our IGBT, using a voltage clamp circuit, furthermore it also gives us ESD protection of up to 4 kV. We have chosen a low-side switch that can be able to handle breakdown voltages up to 505 V, which offers sufficient protection for sudden voltage spikes across the inductive load.

<sup>73</sup> IRG4BC40S Datasheet

<sup>74</sup> ISL9V5045S3ST Datasheet

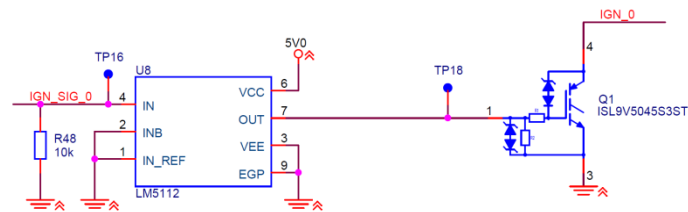


#### 4.1.2. Driver stage

The driver stage is supposed to drive the power stage, meaning that driver stage opens and closes the power stage switch when the coil is fully charged, and this creates the spark in the spark plug, this is illustrated in Figure 96.

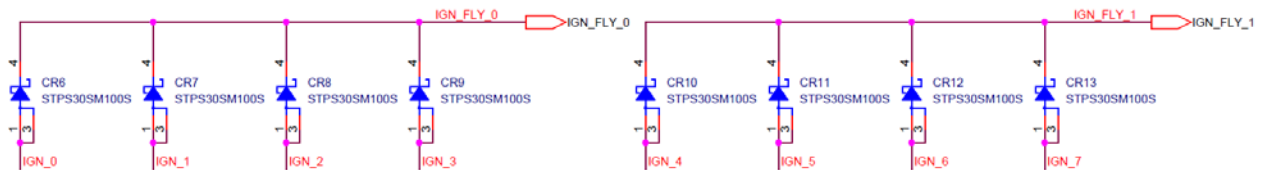
Figure 97 illustrated the connections done in schematic. The LM5112<sup>75</sup> which is a single MOSFET pre-driver, it has two in signals and 1 output signal. These signals are obtained from the ECU.

The requirements that we have for a driver circuit like this is that it has to be available, automotive qualified, high output current and fast switching frequency. For these sets of requirements, we ended up on the LM5112 which is single low-side gate driver; this driver is also the multipurpose stage driver that we are using for most of the low-side switch configurations. The stage driver is discussed more in chapter 4.8



**Figure 97 - Coil driver schematic**

As a protection towards our ECU, we have also added a flyback diode. The purpose with the diode is to eliminate flyback, which is a sudden voltage spike that is induced across an inductive load when the supply voltage is removed<sup>76</sup>. Some coil drivers may already have a flyback diode integrated in the component and some may not have. As a precaution we are adding it, the flyback diode is a Schottky diode (STPS30SM100S<sup>77</sup>) from STMicroelectronics and have a reverse voltage of 100 V and a forward current of 30 A. Also the forward voltage is approximately 0.7 V. The Schottky diode is connected in parallel with the load, see Appendix VI for connection of flyback diodes. Since there are 8 coils, we must have 8 flyback diodes. We have divided them into two groups, BANK1 and BANK2. The implementation is seen in Figure 98.



**Figure 98 - Flyback diode configuration**

<sup>75</sup> LM5112 Datasheet

<sup>76</sup> Wikimedia Commons: Flyback Diode

<sup>77</sup> STPS30SM100S Datasheet



## 4.2. Fuel Injection driver

The fuel injection driver is the actuator whose job is to supply the cylinders with fuel. There are two types of injectors, saturated and peak and hold. The different between these injectors is that in a saturated type the impedance is high ( $> 10 \Omega$ ), while in peak and hold have low impedance ( $< 3 \Omega$ ). Additionally the saturated is very easy to drive, there is only need of  $V_{BAT}$  (typical 12 V) through the injector, while in a peak and hold (P&H) injector require a high peak current to open and a lower current to hold open. These values are typical 3 A and 1 A respectively, this means that there will require circuitry to charge and hold a constant current to control the injector<sup>78</sup>.

Advantages and disadvantages for the saturated type and peak and hold (P&H) can be seen in Table 20.

**Table 20 - Saturated versus Peak and Hold (P&H)**

Type	Advantages	Disadvantages
<b>Saturated</b>	<ul style="list-style-type: none"> <li>• Used for almost all production systems.</li> <li>• High availability.</li> <li>• Easy to drive and configure.</li> <li>• Inexpensive.</li> </ul>	<ul style="list-style-type: none"> <li>• Slower response</li> <li>• Lower flow rates</li> </ul>
<b>Peak and hold (P&amp;H)</b>	<ul style="list-style-type: none"> <li>• Faster response</li> <li>• Large flow rates</li> </ul>	<ul style="list-style-type: none"> <li>• Requires more circuitry and more complicated setup.</li> <li>• Expensive.</li> <li>• Low availability.</li> </ul>

An important setting when dealing with injectors is the injector dead-time also known as battery voltage compensation (BA). BA is the time delay for an injector being energized and the injector opens. Similarly there is a time delay for de-energizing and injector closing. The open time of the injector is significant longer than the closing time, as a result of this, less fuel will flow through the injector for a given pulse width, decreasing the efficiency of the engine. To remove this problem, an increase in pulse width will compensate for the dead time, thus optimizing the combustion process in the engine. The dead time for an injector is a function of battery voltage, differential fuel pressure and type of injector and will be dealt by software<sup>79</sup>.

Since KPEC is developing an ECU for motorsport, we went for the saturated type. The main criteria are that the saturated type is more common, inexpensive and less complicated to interface.

<sup>78</sup> Jeff Krummen, Engine calibration

<sup>79</sup> Linkecu : Battery-compensation



The injection driver is composed of the injector, power stage and a driver stage, which is discussed further in 4.2.1 and 4.2.2. A typical injector has a resistance of  $12\ \Omega$  and we are powering the injector with  $V_{BAT}$  (typ. 12 V), using ohm's law we get that the current through the injector is 1 A. With this we can conclude that our power stage must be able handle 1 A running through it.

#### 4.2.1. Power stage

The power stage of the gasoline injector is either a MOSFET or an IGBT. Since the fuel injector is a low voltage application (typ. 12 V), we are using a MOSFET. A simplified wiring diagram of the injector setup is shown in Figure 99. Since this is a saturated type injector, to drive this, all we need to do is power it with  $V_{BAT}$  and switch the MOSFET on/off. The on and off time determine the flow of the injector. We are going to use a MOSFET that is used as a multipurpose low-side switch. The criterion for using this multipurpose switch is that it has the required protecting and a low  $R_{DS(ON)}$  (less heat) and can manage high currents. The multipurpose low-side switch is discussed further in 4.6.

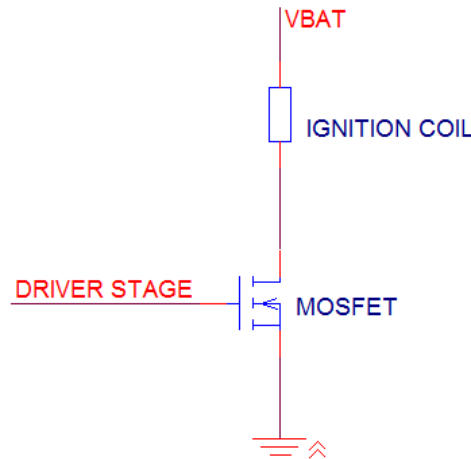


Figure 99 - Simplified wiring diagram of injector



#### 4.2.2. Driver stage

As seen in Figure 99, the driver stage is what activates the MOSFET. The driver stage that we have chosen is a multipurpose stage driver that we are using for almost all low-side switches. Figure 100 illustrates the wiring done in schematic for the injection driver.

The stage driver is discussed more in detail in chapter 4.8.

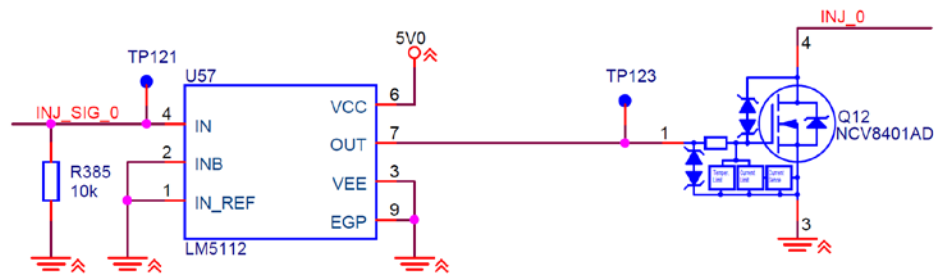


Figure 100 - Injection driver schematic



### 4.3. Boost controller driver

The principle of the boost controller is to control the boost level produced in the intake manifold of a turbocharger or supercharger to increase the efficiency of the engine. The difference between a turbocharger and a supercharger is that in a supercharger there is a compressor which is driven mechanically (using belts, chains, shafts or gears); this increases the pressure or density of air and is then supplied to the combustion engine. While in a turbocharger the exhaust from the combustion process is used to drive a turbine, which increases the pressure of air that is induced and this compressed air is then fed into the combustion engine. A comparison of the turbocharger and supercharger can be seen in Table 21.

**Table 21 - Turbocharger vs supercharger**

Type	Advantages	Disadvantages
<b>Turbocharger</b>	<ul style="list-style-type: none"><li>• Does not put a load on the engine</li><li>• Cheaper</li><li>• Longer life time</li><li>• Commonly used</li></ul>	<ul style="list-style-type: none"><li>• Lower response</li></ul>
<b>Supercharger</b>	<ul style="list-style-type: none"><li>• Faster response</li><li>• Expensive</li></ul>	<ul style="list-style-type: none"><li>• Mechanically driven by the engine</li><li>• Utilizes 20% of engine power</li><li>• Lower life time</li></ul>

In our application we are going to interface with a turbocharger, this is due to that it's the most commonly used, it's cheap, has longer life time and it doesn't consume any power from the engine. To be exact we are going to interface a boost controller that will regulate the turbocharger. The main job for the boost controller is to regulate the pressured air inside the intake manifold.

#### 4.3.1. Boost controller selection

There are two different kind of boost controller that is used. The air control solenoid and the stepper motor. Their principle is the same, although they handle it differently. The operation is as following: The exhaust gas that is produced after the combustion process is fed into the turbine, and the exhaust makes the turbine spin. The spinning of the turbine increases the air pressure of the air that is fed into the turbine. The pressurized air is then fed into the combustion engine increasing the efficiency of the engine. The application of the boost controller is to limit the exhaust going into the turbine when there is too much boost pressure going into the engine. Thus decreasing the speed of the turbine and the pressurized air goes down.

In choosing a boost controller type we looked at different criterion. The most critical criteria that we looked at were availability, meaning that there is enough documentation



to be able to interface the component, also price and complexity of the interface, where important for choosing the type. With these criteria we decided to implement the solenoid type boost controller. The documentation for the stepper type is non-existent, this making the interfacing work almost impossible for us. The price ranges for the stepper vs solenoid is also huge. The stepper one cost from 700 US dollar and the solenoid one cost around 10 US dollar. As a bonus, the solenoid one is also the most commonly used boost controller which make the component available for us.

A typical solenoid boost controller has 4 connection, a vacuum supply (VAC), variable control pressure (OUT), vent connection (ATM) and electrical connection. Typical characteristics of a solenoid boost controller can be seen in Table 22.

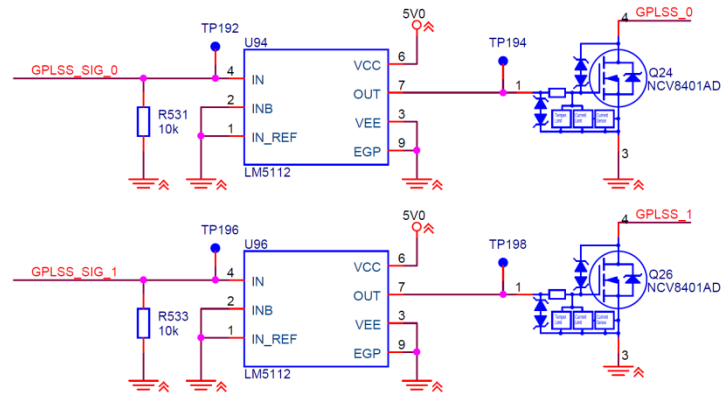
The solenoid is controlled via the ECU (Engine control unit), using a control current. More precise the solenoid is controlled by a current with constant frequency; this is also referred as a PWM (Pulse width modulation). The on and off time of the solenoid is called the duty cycle. Figure 102 shows the schematic for the boost controller. The boost controller consists of a driver stage and a driver stage. These stages are discussed in 4.6 and 4.8.

**Table 22 - Typical characteristics of a solenoid boost controller**

Typical characteristics		
Characteristics	Value	Unit
Rated voltage	12	V
Operating voltage	10 - 16	V
Resistance	11 - 16	$\Omega$
Inductivity	40	mH
Pulse duty factor	20 - 95	%
Frequency	250 - 300	Hz
Ambient temperature	-30 - 120	$^{\circ}$ C

If a twin turbo is used, two turbo controllers is needed, in the schematic for the ECU we have implemented only one turbo. If two boost controllers are needed we have made 4 dedicated low-side switches that can be used to drive any actuator. Figure 101 shows the dedicated low-side switches.





**Figure 101 - Twin turbo configuration**

The solenoid is controlled through PWM, and this signal is produced from the FPGA. Further information about this topic is discussed further in 4.9.



#### 4.4. Fuel pump driver

The purpose of the fuel pump is to provide the engine with fuel. We are interfacing the fuel pump with information from Bosch. Typical characteristics are seen in Table 23.

**Table 23 - Typical characteristics of fuel pumps**

Typical characteristics		
Characteristics	Value	Unit
Supply voltage	6 - 16.5	V
Rated voltage	13.8	V
Load current	6 - 19	A
Non-return valve	External/internal	N/A
Fuel filtering	External/internal	N/A

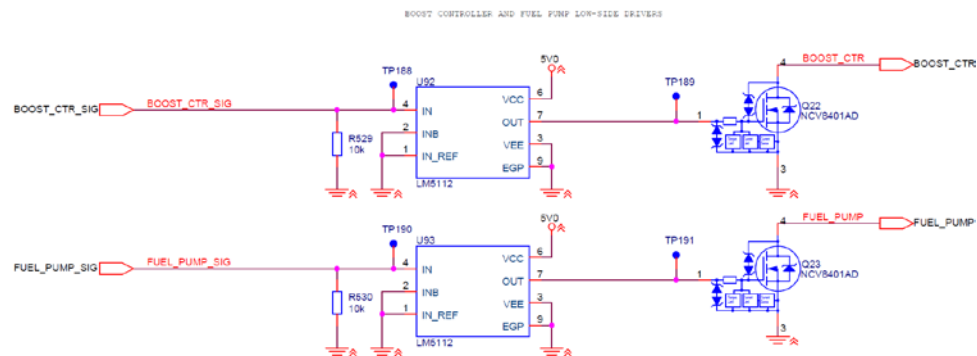
The way we are going to interface with the fuel pump. Meaning we are going to operate a switch between the fuel pump and supply voltage. A standard in the automotive is that the fuel pump electrical wires go through a relay. We are then going to activate this relay so that it closes/opens which leads to the fuel pump being activated. The switch that is used is going to be connected in a low-side configuration. Here we have the choice between an IGBT and an MOSFET. In this application we are handling a 12 V power supply and the switch needs to stay either on or off, there is no need for a PWM. We decided to go for the multipurpose MOSFET which is discussed in 4.6.

##### 4.4.1. Fuel pump safety

It is important to know about some features regarding the fuel pump and when to activate the fuel pump. The fuel pump relay is controlled by the ECU. There are two parameters that should be taken into account before activating the fuel pump; the ignition key on signal and RPM input. Once the ignition is turned the fuel pump relay should be active for 1 - 2 seconds, this is called the prime and is to give the initial fuel to the engine. After the prime the fuel pump relay should be turned off. The fuel pump relay can only be activated (after the prime) if the ignition is on and there is an RPM signal present, this is a security issue and is used to protect driver and passengers. If the car stalls or gets into an accident, we want to disconnect the fuel pump.



Figure 102 shows the wiring done in schematic for boost controller and fuel pump. The MOSFET and driver stage is discussed further in 4.6 and 4.8.



**Figure 102 - Boost controller and fuel pump**



## 4.5. Throttle body motor controller

The throttle body has a valve that regulates the amount of air that is fed to the engine. The position of the valve is regulated by the angle of the gas pedal controlled by the driver. When the gas pedal is depressed by the driver, the throttle body opens the air valve, supplying air to the engine for combustion. The throttle body uses a geared permanent magnet DC motor for controlling the valve angle in the throttle body. The position of the valve in the throttle body can be read as a feedback signal generated by two sets of metallic wipers creating contact with a carbon track, creating a voltage signal. The two sets wipers functions principally equal to how two potentiometers work, and the values can be compared to each other to improve readouts as the wiper tracks are worn over time.

In order to control the DC motor in both directions, an H-bridge circuit is used. The H-bridge allows switching pairs of transistors on or off in order to control the direction the current flows through the motor, thus controlling the direction the motor rotates.

### 4.5.1. MC33931 H-bridge controller

The MC33931<sup>80</sup> is an H-bridge power IC designed especially for throttle control, but can drive any H-bridge motor control application with current draws of up to 5 A. The H-bridge is internally implemented in the IC.

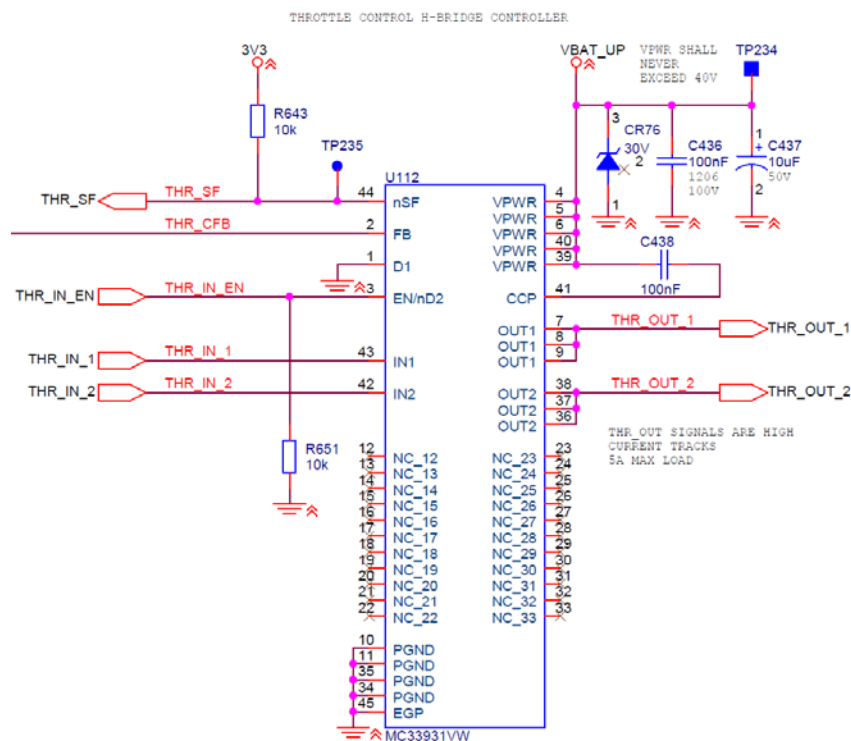


Figure 103 - MC33931 schematic implementation

<sup>80</sup> MC33931 Datasheet



Figure 103 shows the schematic implementation of the MC33931. The reason for the many no-connects (NCs) is that the package is thermally connected to a large surface underneath the package that must be coupled to a ground plane. The small package dissipates large amounts of heat due to the large currents that are switched in the internal H-bridge.

**Table 24 - MC33931 pinout description**

Pin	Description
nSF	Status Flag
FB	Feedback
D1	H-bridge output disable
EN/nD2	H-bridge output disable
IN1	OUT1 control input
IN2	OUT2 control input
CCP	Charge Pump Capacitor
OUT1	H-bridge side 1 output
OUT2	H-bridge side 2 output

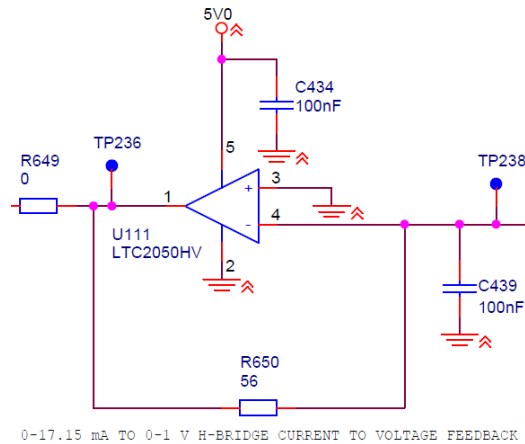
The pin functions are described in Table 24. As seen in Figure 103, the set nSF pin is pulled high, as this is an open drain output. The pin will be pulled low if fault detection circuitry in the IC is triggered. The D1 pin is pulled hard to a low level, making the output enable controlled by the EN/nD2 signal alone. IN1 and IN2 are the inputs for controlling the H-bridge outputs. The CPP pin is for the charge pump capacitor, which must be a value between 30 - 100 nF<sup>81</sup>. The OUT1 and OUT2 pins are the H-bridge outputs. In the schematic, the signal outputs of the H-bridge are defined as 1 and 2, corresponding to the input numbers, where control input IN1 controls OUT1 and control input IN2 controls OUT2.

<sup>81</sup> MC33931 Datasheet



#### 4.5.1.1. MC33931 current feedback (FB)

The FB-pin on the MC33931 is a current feedback pin, allowing to monitor the amount of current in the device. The FB-pin supplies current equal to 0.24% of the high side current in the H-bridge.



**Figure 104 - MC33931 current to voltage feedback**

Figure 104 shows the circuit for converting the output current of the FB pin on the MC33931 to a voltage readable by the Zynq XADC embedded on the TE0720. The XADC only handles input voltages of 0 - 1 V, while the FB-pin on the MC33931 outputs a current feedback in the range of 0 - 17.5 mA, corresponding to 0 - 6 A of high side current.

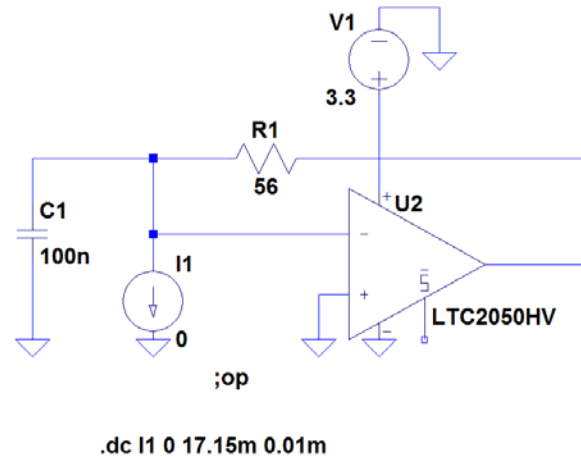
$$V_{out} = R_f I_f \quad (68)$$

By Ohms law (68), the voltage can be giving by multiplying the feedback resistor with the current flowing to the input of the LTC2050HV operational amplifier.

$$V_{out} = 56 \, \Omega \cdot 17.5 \, \text{mA} = 0.98 \, \text{V} \quad (69)$$

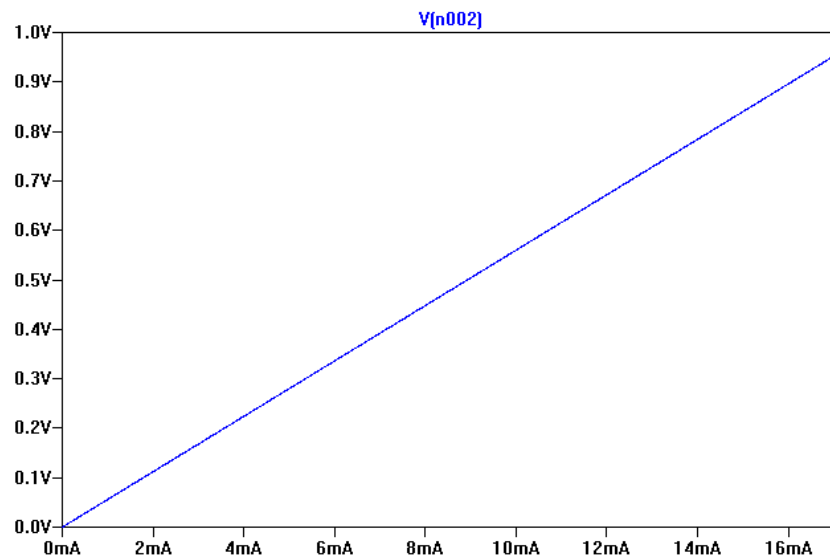
By inserting the values from Figure 104 into (68), we yield a voltage range of 0 - 0.98 V for the current range 0 - 17.5 mA using a 56  $\Omega$  resistor, which is a standard E12 resistor value.

The circuit is simulated using LTSpice in order to verify that it operates properly.



**Figure 105 - Feedback circuit simulation**

Figure 105 shows the drawn simulation circuit. The circuit is drawn exactly as in the schematics, with the only difference being that a current source is added to represent the feedback signal. The simulation command is a dc sweep, which performs a sweep of the current source I1 from 0 - 17.5 mA with a step of 0.01 mA.



**Figure 106 - Voltage feedback simulation results**

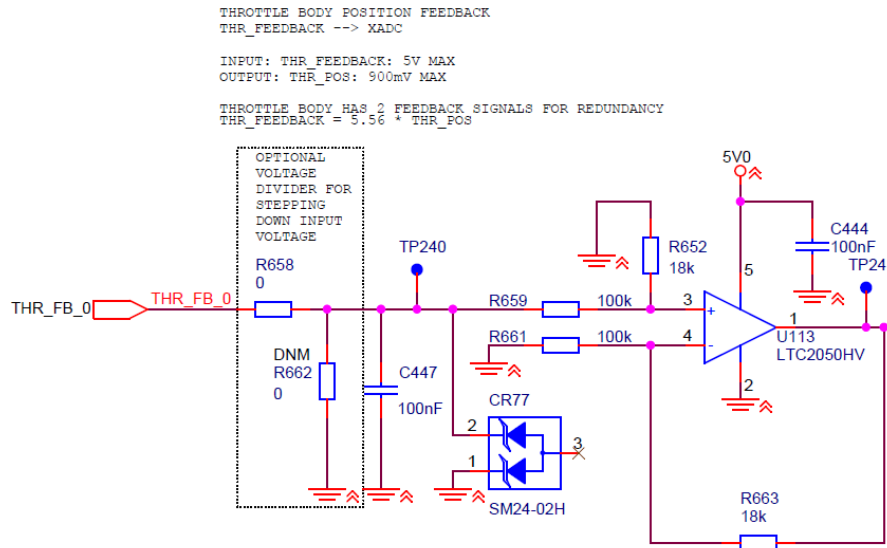
Figure 106 shows the simulated results of the circuit. As expected, the circuit produces a linear relationship between voltage output and current input. In order to determine current output of the high side in the H-bridge, a simple mathematical relationship between feedback current and output voltage can be made. Take note that this is based on the maximum output value, and that the typical value must be measured on the finished board. The simulations serve as a good indicator, but component tolerances and other non-idealities will change the actual output read by the XADC.



#### 4.5.2. Throttle body position feedback

As mentioned in the introduction to this section, the throttle body itself has a feedback mechanism in order to be able to measure the throttle body position. The sets of wipers causes the resistance output of the tracks to change, and the position can be calculated. In order to measure the change in resistance, a voltage is applied to the tracks, and changes with resistance changes the output voltage, as the tracks and the sets of wipers acts as a potentiometer, which is equivalent to a variable voltage divider circuit.

The voltage range is simply selected based on the input voltage on the tracks. The recommended voltage level for use on these tracks is 5 V as the circuitry for reading the 5 V values are based on the 5 V to 1 V input buffer described in 3.11.2. If 12 V is applied to the tracks, resistors for implementing a voltage divider on the input can be added. This implies placing resistors on the input of the buffer in order to divide down the voltage of the circuit. The input to the buffer shall always be lower than 5 V. It is not recommended to reach the rail voltage of the input buffer due to limitations of the operational amplifiers.



**Figure 107 - Throttle body feedback implementation**

The implementation of one of the throttle feedback circuits are shown in Figure 107. There are two such circuits implemented in the schematics. The circuit is designed for a maximum 5 V input to the voltage buffer, but the 0  $\Omega$  resistors on the input can be configured as a voltage divider to allow for higher voltage inputs, as long as the input to the buffer never exceeds 5 V.





#### 4.6. Multipurpose MOSFET

While developing the ECU, we found out that there are many sensors and actuators that need to be interfaced. There are 17 different sensors, 44 sensors total and 5 different actuators and 24 actuators in total.

Most of the actuators need to be activated. As previously said, this is done by low-side switching. This is the main reason for choosing a multipurpose MOSFET that can be used as low-side switching in applications under 250 V. Some minimum criterions for the multipurpose MOSFET are that it needs to be able to handle the range of current requirements of the different actuators and have ESD protection.

Other characteristics that are important in choosing a MOSFET is  $P_D$ , this parameter tells us how much the MOSFET is able to dissipate power. For this reason it's important to choose an MOSFET with low Drain-to-Source On-Resistance  $R_{DS(ON)}$  as possible. An example of this is if 2 A is running through the MOSFET and the  $R_{DS(ON)}$  is 100 mΩ, using the power formula we get that there is 0.4 W that is being dissipated by the MOSFET. If the MOSFET can't dissipate this heat, then the heat that is produced will destroy the MOSFET, thus ending an important feature of our application.

We ended upon a device that is custom made for this kind of applications; switching a variety of resistive, inductive and capacitive loads and that is automotive rated. This device is the NCV8401ADTRKG<sup>82</sup>, which have a  $V_{DSS}$  that is clamped to 42 V, which tells how much voltage that can be applied without causing avalanche breakdown. A typical  $R_{DS(ON)}$  23 mΩ and a maximum drain current of 33 A, which is also the limited at the same value.

The device also offers overvoltage protection, ESD protection, temperature limit and current limiting. This is illustrated in Figure 108.

This MOSFET has a temperature limit at 175 °C, if exceeded the device will turn off. The current limit is at 33 A and the overvoltage protection clamps any voltage above 42 V.

---

<sup>82</sup> NCV8401ADTRKG datasheet

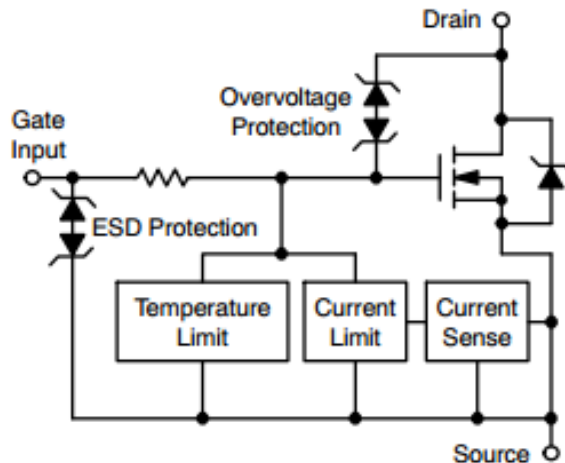


Figure 108 - Block diagram of NCV8401ADTRKG<sup>83</sup>

#### 4.7. General purpose low-side switches

To be able to interface towards actuators that need a low-side configuration in the future, we have chosen to implement 4 general purpose low-side switches. These may be used to interface a tachometer or another boost controller for twin turbo engines.

The GPLSS (General Purpose Low-Side Switch) implementation is seen in Figure 109.

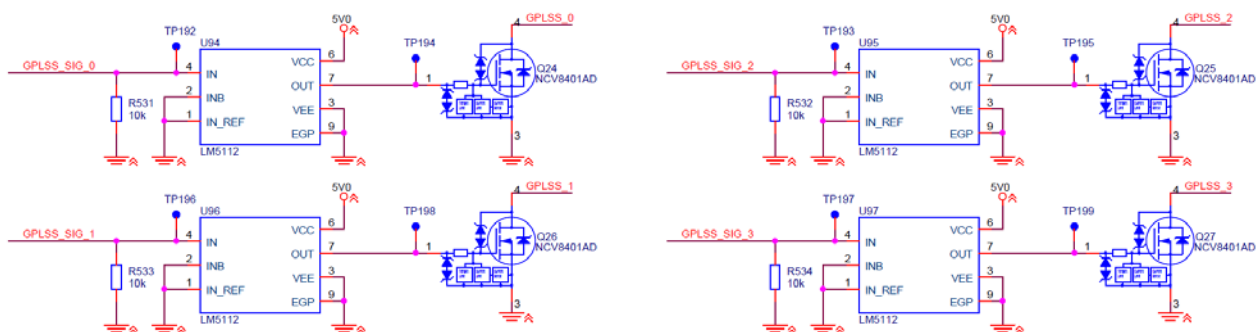


Figure 109 - General purpose low-side switch

<sup>83</sup> Figure from NC8404ADTRK datasheet



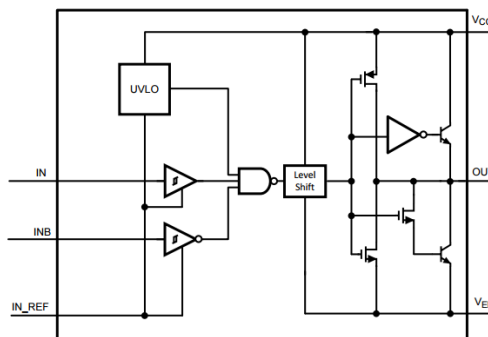
#### 4.8. Multipurpose stage driver

The stage driver is produced by Texas Instruments and is a single 7 A MOSFET gate driver, which has an operating voltage from 3.5 - 14 V. It has a 7 A sink / 3 A source current. The LM5112<sup>84</sup> is automotive qualified and is built to drive N-channel enhancement-mode MOSFETs by inducing high peak current pulses during switching. The LM5112 is CMOS but have compatibility for TTL input thresholds. For a CMOS device a logic low is anywhere  $0 - \frac{1}{3} V_{DD}$  and logic high is from  $\frac{2}{3} V_{DD} - V_{DD}$ , where  $V_{DD}$  is the supply voltage (typ. 2 - 15 V). While in a TTL device a logic low is anywhere between 0 - 0.8 V and 2 V -  $V_{CC}$  is logic high (typical  $V_{CC}$  is in range 4.75 - 5.25 V). A comparison between TTL and CMOS is seen in Table 25.

**Table 25 - TTL vs CMOS**

Input threshold	Advantages	Disadvantages
<b>TTL</b>	<ul style="list-style-type: none"> <li>Robust</li> </ul>	<ul style="list-style-type: none"> <li>Price</li> <li>Draw more power at rest</li> </ul>
<b>CMOS</b>	<ul style="list-style-type: none"> <li>Price</li> <li>Draw less power at rest</li> <li>Less parts</li> </ul>	<ul style="list-style-type: none"> <li>Susceptible to ESD</li> </ul>

We chose to use the LM5112 that comes with CMOS threshold because CMOS is the most common used, and is currently phasing out TTL. The CMOS is also easier to implement, because it require less components around the device. A simplified block diagram of the LM5112 is seen in Figure 110.



**Figure 110 - Block diagram of LM5112<sup>85</sup>**

The LM5112 has inverting and non-inverting input pins to satisfy for both inverting and non-inverting gate drive. The signal that is used to drive the gates is a PWM signals that is generated by the ECU; this is discussed more in chapter 4.9. The OUT pin is the

<sup>84</sup> LM5112, Stage driver

<sup>85</sup> Figure from LM5112 datasheet



output pin to the gate of the MOSFET, this pin is held low until input is present and VDD is above UVLO (Under-voltage lockout) threshold. VDD is the supply voltage which provides power to the IC. VEE is common ground reference for input and output circuits. The INB pin is connected to IN REF which is coupled to GND; this is because the INB pin is not used in this application.

This driver is chosen as the multipurpose stage driver for most of the low-side switches; the reason for this is that there is sufficient with information about this device, it is automotive rated and is custom made for this application. The 7 A peak current is sufficient for driving all the low-side switches and the enable speed is adequate. Since there are a lot of low-side switches, we wanted to find one stage driver that could handle them all. This way we can lessen the BOM (Bill of materials) and reuse the component in other features.

There is no need for protection circuitry around this device because our power stages has sufficient protection, thus any harm towards the stage driver will not occur.

The LM5112 is bypassed with a standard 100 nF capacitor and a 22  $\mu$ F low ESR capacitor. The 100 nF and 22  $\mu$ F is in parallel. The 100 nF is used to reduce unwanted AC signals (noise) that is riding on the DC signals. The 22  $\mu$ F bulk capacitor is used to provide energy to the component, to compensate for the energy that is pulled instantaneously for short periods. We chose this specific value of 22  $\mu$ F, because it was recommended by the manufacturer. The configuration for the stage driver is seen on Figure 111.

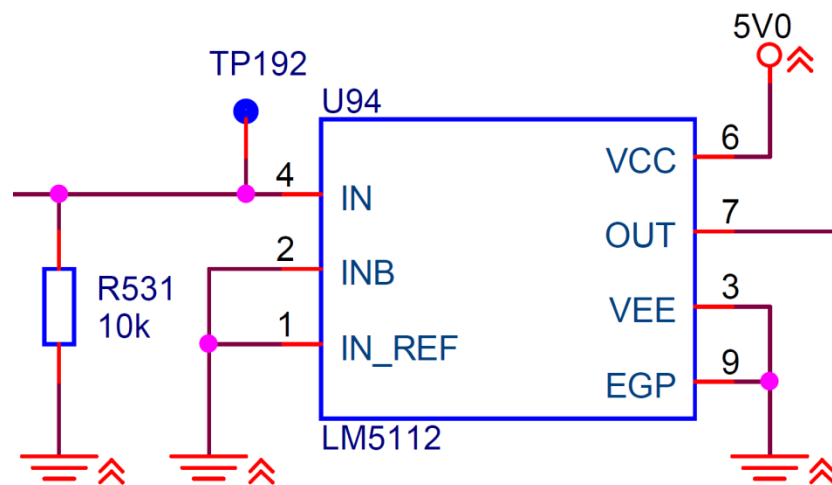


Figure 111 - Stage driver configuration



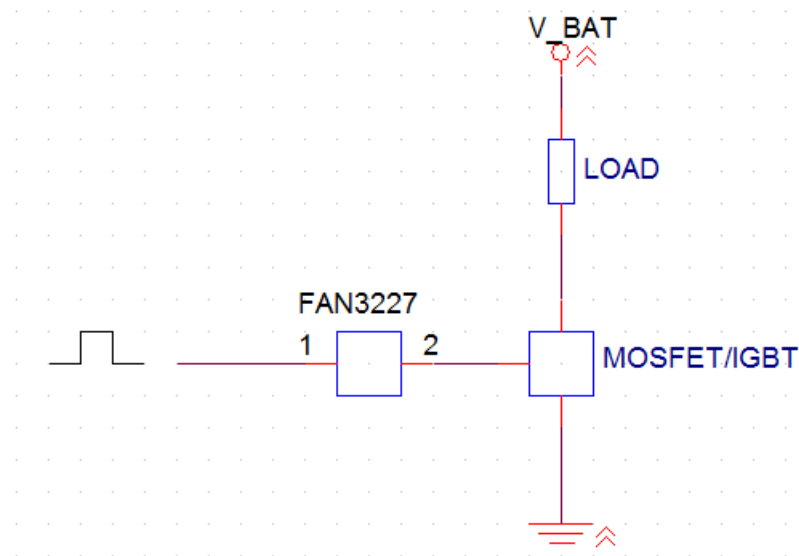
#### 4.9. PWM – Pulse Width Modulation

The PWM is a voltage and current signal with constant frequency and is used to switch on/off the low-side switches. The principle of the PWM is that the average voltage and current that is induced to the load is controlled by turning the switch that is configured in a low-side configuration, between the load and ground, on and off in a fast rate. The longer the switch is on, the higher is the power to the load and vice versa.

The duty cycle of a PWM is the percentage of on time on an interval called for a period. 100% duty cycle is fully on and 0% duty cycle is closed.

The advantages with using PWM are that the power loss in the low-side switch is really low. This is due to when the switch is off, there is practically no current, and when the switch is on, and the power is being transferred to the load, there is almost no voltage drop across the switch.

The way we are using PWM is to send the signal through the stage driver and to the power stage. The stage driver activates the gate, and PWM signal switches the switch on and off, thus transferring power to the load. This is illustrated in Figure 112.



**Figure 112 - PWM block diagram**

The simplest form of PWM requires two signals, a sawtooth or triangle signal, a carrier signal and a comparator. A comparator is used to compare the two signals, and when the signal carrier is higher than the sawtooth or triangle the PWM signal goes high and when the carrier signal is lower than the sawtooth or triangle the PWM is low. This method is called for the interseptive method. There are many methods for PWM control, such as delta and delta-sigma. Which method has not been taken into account for during this project.



We decided that the PWM should be generated in the FPGA; this is due to complex hardware solutions. If we were to implement an IC to generate the PWM, we would need additional components that need to interface with the IC for generation of sufficient PWM signal that would be used.

This is where we have an advantage, the flexibility of the FPGA. With the FPGA we can create any digital circuit, so that we can reduce the complexity of hardware. With less hardware components we also have less power loss in the design. According to Suryakant Behera<sup>86</sup>, who wrote a thesis about this matter, a digital PWM is less sensitive to noise and immune to temperature and voltage changes. They are also much more flexible than analog PWM devices.

Suryakant Behera, also says that the advantages with a FPGA based design is that it is easily to modify, because in the FPGA there are interconnecting logic blocks and these can be easily modified. Additionally it gives us the advantage that we can implement the PWM within a short time. Furthermore an implementation of a PWM signal in a FPGA design cost less.

---

<sup>86</sup> Suryakant Behera, FPGA based PWM



## 5. Communication interfaces

This section contains detailed descriptions of interfaces on the ECU that makes up for all external communication to other devices and systems. Communication interfaces are used for programming and debugging the ECU, as well as communicating with the finished unit when implemented for use.

### 5.1. USB 2.0 interface

Universal Serial Bus (USB) is a widely used interface for communication between computers and many types of devices. Implementing a USB interface on our ECU will allow the software developers to communicate with the ECU, using a standard USB cable connected directly to the USB port. We have selected to use a micro-B type USB connector on our ECU. We have chosen to use a FTDI FT2232HL<sup>87</sup> dual channel USB controller. FT2232HL allows for USB full speed communication, meaning that it can handle USB 2.0 data rates of up to 480 Mbit/s. See Table 26 for a description of signals present in the USB interface.

**Table 26 - USB 2.0 signal descriptions**

Signal name	Description
VBUS	5 V+
D-	Data-
D+	Data+
GND	Ground
ID	Host/device identifier

#### 5.1.1. USB to dual UART interface

Both channels of the FT2232HL USB interface controller is configured as a regular UART, allowing for using the USB port as a debugging port to the MRF24WG0MB Wi-Fi module and regular UART towards the TE0720. There are LEDs connected to dedicated pins on the FT2232HL that are active low outputs, which switch on LEDs to indicate whether data is transmitted or received. The MRF24WG0MB Wi-Fi module is described more in detail in 5.4.1. Table 27 shows the signal connections for the two UART channels of the FT2232HL towards the TE0720 and the MRF24WG0MB.

**Table 27 - FT2232HL Dual UART pin configuration**

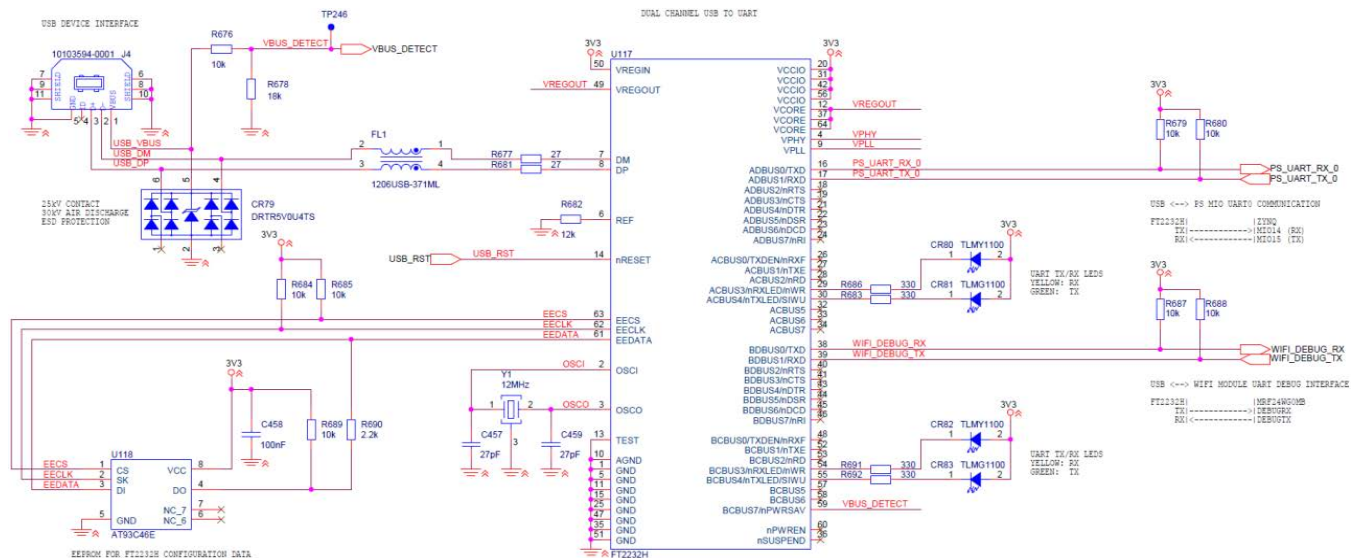
Device	UART Channel	Signal name	Device pin
TE0720	ADBUS0/TXD	PS_UART_RX_0	MIO14
	ADBUS1/RXD	PS_UART_TX_1	MIO15
MRF24WG0MB	BDBUS0/TXD	WIFI_DEBUG_RX	DEBUGRX
	BDBUS1/RXD	WIFI_DEBUG_TX	DEBUGTX

<sup>87</sup> FT2232H Datasheet



### 5.1.2. Design specifics for FT2232HL

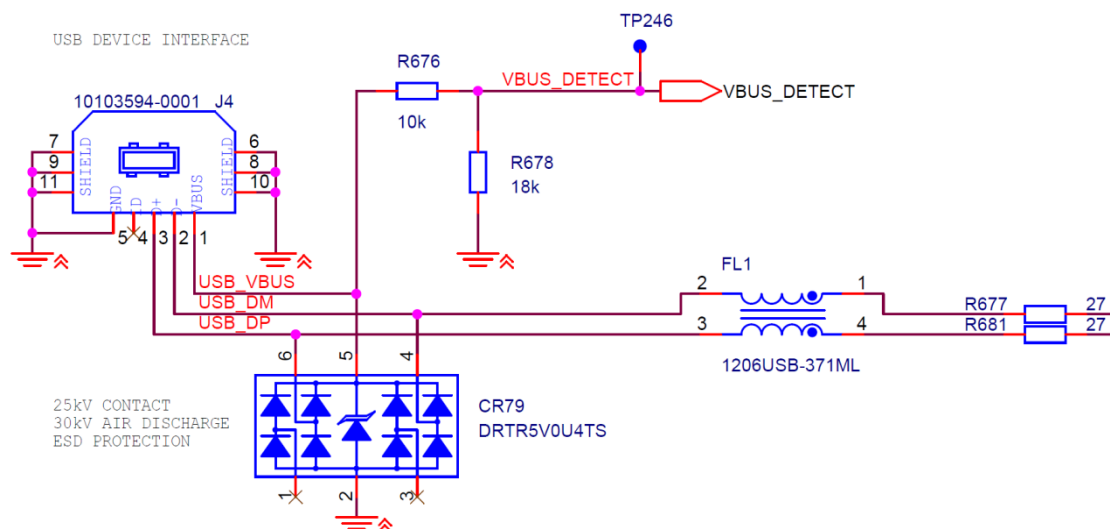
The FT2232HL is a widely used and inexpensive USB interface controller. It is very versatile in its operation and can be configured for virtually any standard communication interface, and is typically used in almost any application for adding USB connectivity. Due to its popular use, FT2232HL is very well documented by the manufacturer FTDI, and is also found in many reference designs we can base our implementation on in order to ensure that we implement the hardware design of the controller correctly. The FT2232HL will be powered by the on-board 3.3 V source, i.e. self-powered mode. There is an internal LDO to supply the 1.8 V processor core, which requires an external filter capacitor. The rest of the device, including all I/O is powered by the 3.3 V supply. Figure 113 shows the complete schematic implementation of the FT2232HL USB interface.



**Figure 113 - FT2232 schematic implementation**

The USB is implemented as a device port, and operates in self-powered mode. Self-powered mode operation implies that the USB interface is not powered from the 5 V VBUS which are one of the channels in the USB, and requires that the interface IC is powered by a source external to the USB interface. As it is configured as a device port, it is given that it can only be connected to a USB host controller as normally found in computers. Device mode is recognized when connected to a host device by the dedicated ID pin that was introduced with mini- and micro-USB connectors. The ID pin is left floating, indicating a device interface to the host.

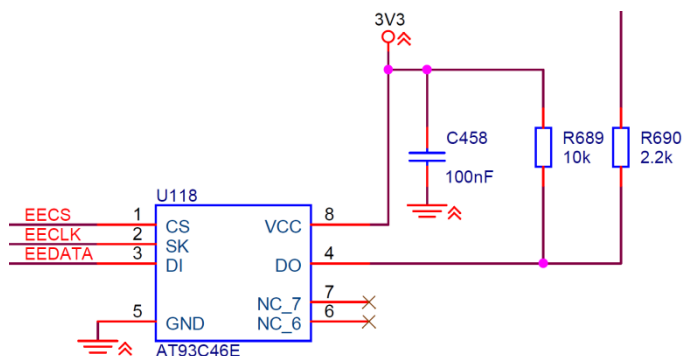




### Figure 114 - USB connector and signal lines

Figure 114 shows the connector and USB differential lines connected to the FT2232HL. A 5 V TVS array suitable for USB is used in order to protect against ESD. The VBUS\_DETECT signal is connected to both the FT2232HL and the TE0720 in order to signal that a USB host is connected to the USB port. The resistors are simple voltage dividers in order to step the voltage down to a 3.3 V level from the 5 V supplied by the USB. There are also signal termination and common mode choke coils in order to prevent signal reflections and reduce common mode noise in the signal in compliance to the USB 2.0 standard<sup>88</sup>.

An external EEPROM is required to hold necessary configuration data, which are accessed directly from the FT2232HL with dedicated I/O lines. The EEPROM is a 1 kB 16-bit word device (16x64 b) from Atmel, the AT93C46E<sup>89</sup>, using an industry standard three-wire serial interface. The schematic implementation for the configuration data EEPROM is shown in Figure 115.



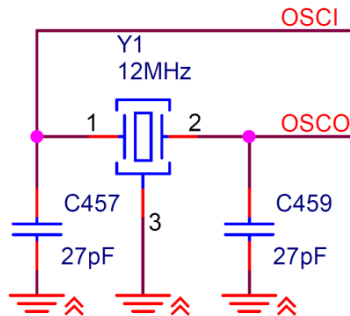
### Figure 115 - FT2232HL configuration data EEPROM

<sup>88</sup> USB Implementers Forum, Inc; Universal Serial Bus Specification

89 AT93C46E Datasheet

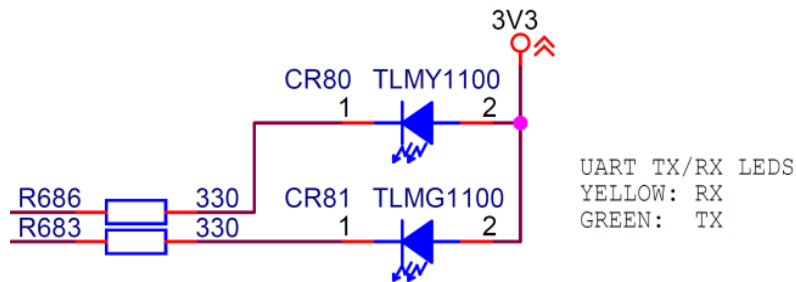


The device also requires an external 12 MHz crystal oscillator for precision clock that is required in USB interfaces. The schematic implementation of the 12 MHz oscillator is shown in Figure 116.



**Figure 116 - 12 MHz oscillator for USB clock**

As an additional feature, there is indicator LEDs connected to the FT2232HL in order to see that data is either transmitted or received. The schematic implementations of the LEDs are shown in Figure 117.



**Figure 117 - Data transmission LEDs**

There are one yellow and one green surface mount LED for each UART. The yellow LED indicates that data is received to the FT2232HL, i.e. transmitted from the targeted device, and the green LED indicates that data is sent from the FT2232HL, i.e. received by the targeted device.



## 5.2. JTAG

JTAG is a standard interface used in most ICs containing any form of programmable logic. JTAG is the common name for IEEE Std. 1149.1, which specifies standard Test Access Port (TAP) and Boundary-Scan Architecture. From its creation, JTAG has become a de facto standard debugging interface and is also used for both development and factory programming of components.

Often, a circuit board can contain more than one component using JTAG. JTAG allows for daisy-chaining these components, and to access all through a single JTAG interface. There is always recommended to include a dedicated header accessible for JTAG-debugging in every design, or at least to make it available through test points in space constrained designs. Table 28 shows standard JTAG interface I/O, although different manufacturers of ICs often have additional pins for extended debugging functionality.

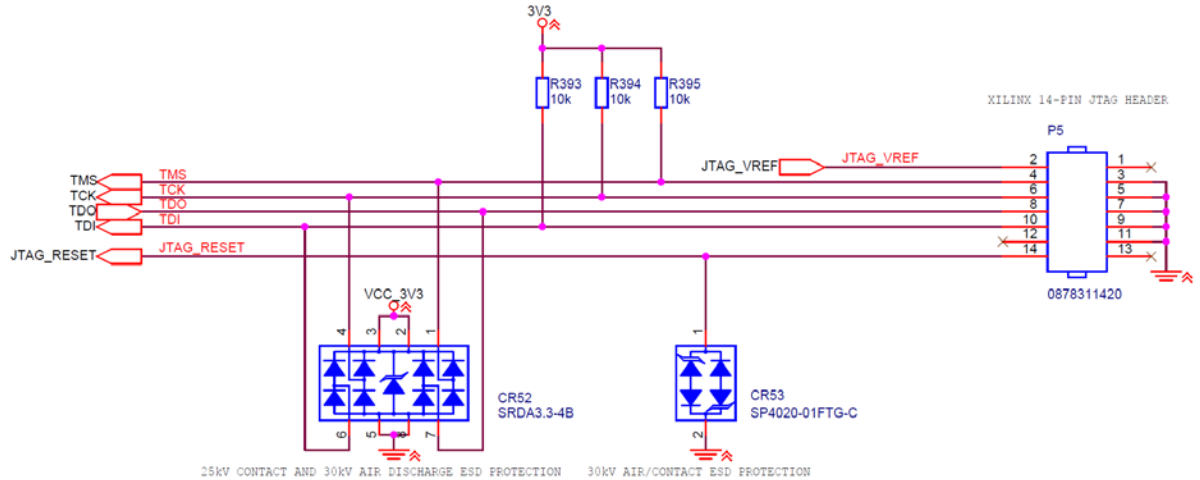
**Table 28 - JTAG Signal Description**

Signal name	Description
TDI	Test Data in
TDO	Test Data Out
TCK	Test Clock
TMS	Test Mode Select
TRST	Test Reset



### 5.2.1. JTAG implementation on the ECU

JTAG is directly accessible through a standard 14-pin Xilinx JTAG interface. The Xilinx JTAG interface uses a voltage reference output from the board in addition to the standard JTAG I/O as described in Table 28.

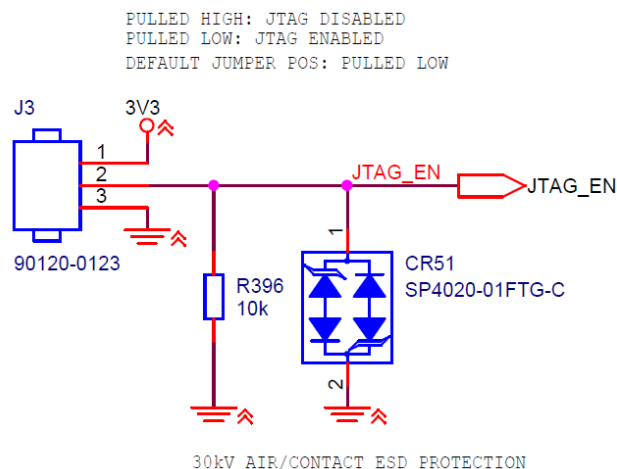


### Figure 118 - JTAG header Schematic

Platform Cable USB II is the JTAG programmer supplied by Xilinx. The Xilinx 14-pin JTAG interface uses a 2 mm pitched connector. JTAG is often implemented with standard 2.54 mm pitch headers, but the Xilinx JTAG programmer is not. We implemented the 87831-1420 connector from Molex in order to match the connector from the Xilinx Platform Cable USB II.

### 5.2.2. JTAG enable jumper

Access to JTAG functions can be toggled by placing a jumper on the ECU.



### Figure 119 - JTAG enable jumper



The *JTAG\_EN* signal is connected to the TE0720 and enables JTAG when pulled low. If the jumper is not present, the default state is to enable JTAG. JTAG can be enabled by placing the jumper on positions 2 and 3, shorting *JTAG\_EN* to GND. By placing the jumper on position 1 and 2, *JTAG\_EN* is forced to a logic *high* signal, thus disabling JTAG.

A 27  $\Omega$  series termination resistor is placed on the TDO line in accordance to manufacturer recommendations.

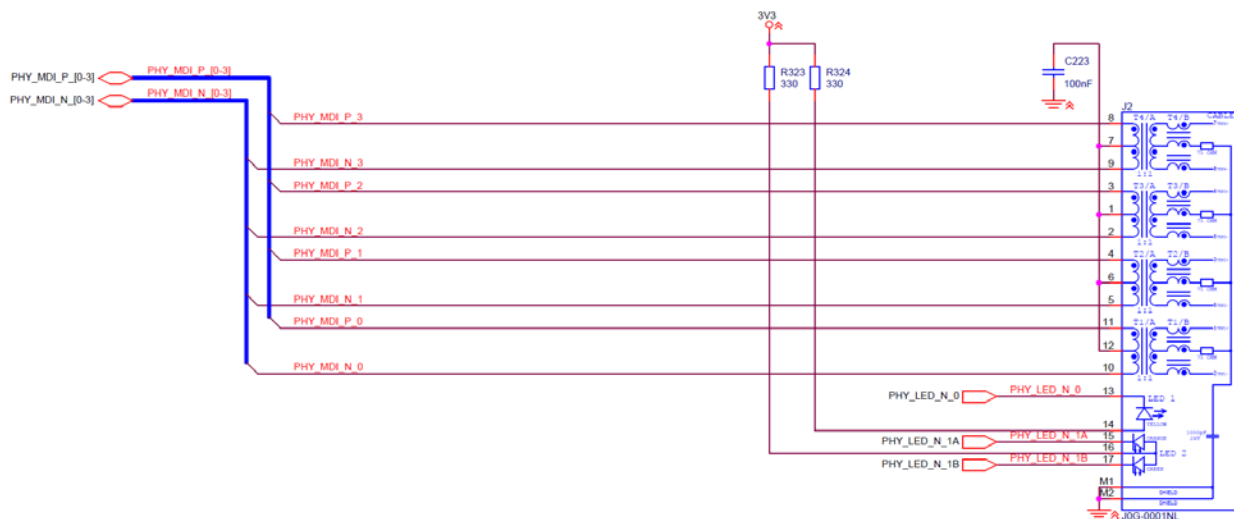


### 5.3. Ethernet

Ethernet is a communication standard found in all local and wide area networks. The Ethernet standard is described by IEEE Std. 802.3. Ethernet is most often physically interconnected using the well-known RJ45 connector used with Cat 5e or Cat 6 cables for Gigabit Ethernet speeds.

The controller for the Ethernet PHY is located on the TE0720 module itself. Thus it is very easy to set up for use hardware wise, as the signals can be made directly accessible from a regular RJ45 jack on the circuit board without further changes in hardware. The controller used for the PHY is a Marvell 88E1512<sup>90</sup> controller supporting up to Gigabit Ethernet speeds. Debugging features can be made available through Ethernet in the software setup of the module.

The internet specification requires transformer coupling between nodes, usually referred to as *magetics*. Many RJ45 jacks have this implemented in the jack itself to save board space, simplify layout and reduce BOM. For these reasons, we selected J0G-0001NL<sup>91</sup> from Pulse Electronics which has implemented magetics and Gigabit Ethernet support.



**Figure 120 - Ethernet interface jack**

As seen in Figure 120, the four pairs of signal lines are differential signals. The reason for using differential signalling is to allow the very high speed that makes up Gigabit Ethernet. Differential signals are really mirrored signals; however, noise affecting both signals will not be mirrored. When subtracting the signals at the receiving end of the signal, common mode noise can be rejected. Noise becomes a gradually worsening problem with longer distance transmission, especially at high speeds.

<sup>90</sup> 88E1512 Datasheet

<sup>91</sup> J0G-0001NL Datasheet



As the casing of the Ethernet jack is shielded, ESD should not really be much of a problem. As the signals from the jack is directly connected to a component on the TE0720, it is very important to protect it as much as possible, being a component that is close to impossible to repair and the entire module is relatively expensive to replace.

The LEDs on the Ethernet jack are connected to FPGA output pins. As the signals from the Ethernet PHY for controlling the Ethernet connector LEDs are internally routed and not directly available on the B2B connectors, the signals for the LEDs must be rerouted through the FPGA in order for the LEDs to function. There is a single yellow LED which indicates communication, while the second LED is a dual colour green and orange LED that can switch colour depending on the connection speed. Normally, one colour is used for Gigabit (1000 Mbps) speed and the second colour is used for 100 Mbps connection speed. The LEDs are pulled high by default and must be pulled low in order to light. The connections to the Zynq SoC are shown in Table 29.

**Table 29 - Ethernet PHY LED connections**

<b>Zynq pin</b>	<b>Schematic net name</b>
B34_L7_N	PHY_LED_N_0
B34_L18_P	PHY_LED_N_1A
B34_L18_N	PHY_LED_N_1B



## 5.4. Wi-Fi

Wi-Fi is a wireless communication standard for transmitting and receiving data over local area network wirelessly. The benefit of implementing Wireless LAN (WLAN) in our ECU is to allow tablet, phones and other handheld devices to communicate with the ECU without the need of physically tapping into the CAN bus. All Wi-Fi certified products follow the IEEE 802.11 standard.

Wi-Fi requires high frequency analog transmission in the 2.4 GHz band, which makes special care necessary in order to maintain signal fidelity. To ease the design process of implementing high frequency analog interfaces, a module solution was chosen. The module selected for use in our application is the Microchip MRF24WG0MB<sup>92</sup>. The module is soldered directly onto the circuit board and communicates with the TE0720 through SPI.

MRF24WG0MB supports IEEE 802.11b/g, including Wi-Fi Direct. Wi-Fi Direct allows two units to communicate directly between their Wi-Fi interfaces without the use of a separate access point. The penalty for using Wi-Fi direct is a slight reduction in transfer speeds between nodes, but it should be sufficient for use in our application.

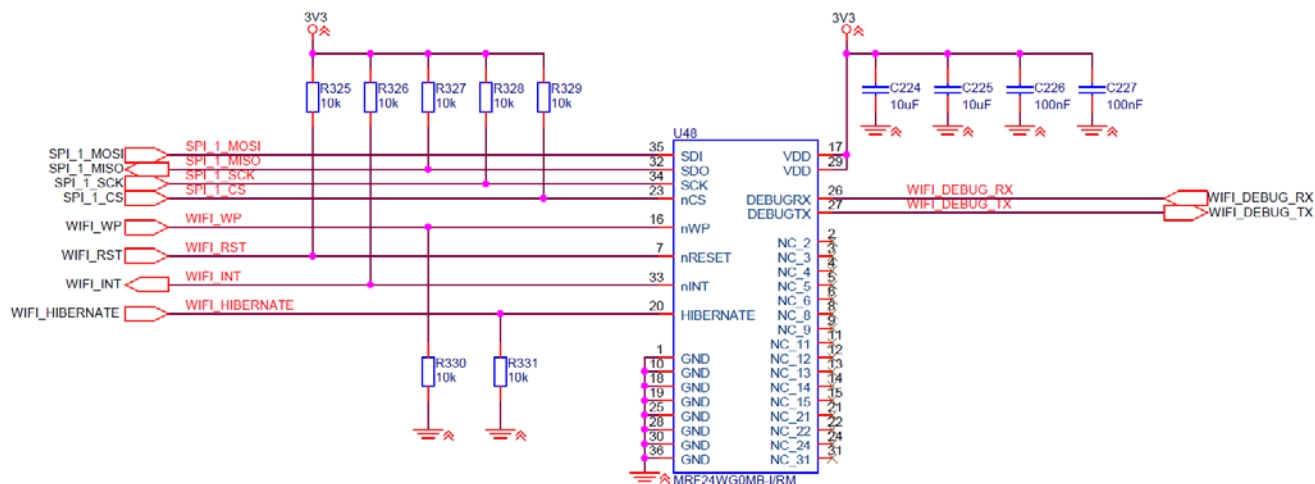


Figure 121 - Wi-Fi module schematic implementation

<sup>92</sup> MRF24WG0MB Datasheet





#### 5.4.1. MRF24WG0MB pinout description

Figure 121 shows the schematic drawing implementing the Wi-Fi module. The module current consumption is fairly large at 240 mA for transmitting data and 156 mA for receiving data. The power input pins (VDD pins) are located on opposing sides of the module, and therefore require the large decoupling capacitors per pin.

The SPI interface can be operated with a clock frequency (SCK) of up to 25 MHz. The SPI interface is connected to SPI\_1 bus which is dedicated only to use for this device. There is a UART compatible debug interface. The debug interface is connected to the USB controller to allow direct access to a host computer, described in 5.1.1.

The Write Protect (WP) pin is protection for the internal flash memory of the module. It is normally pulled low to trigger write protection, but by pulling this pin high, the firmware of the module can be updated after implementation.

The HIBERNATE pin can be pulled high in order to deactivate the module, thus saving power in the system.

The Reset pin is an active low pin and is triggered by global reset of the ECU.

The interrupt pin is an open drain output and is pulled low to signal an interrupt to the TE0720.



## 5.5. CAN

Controller Area Network (CAN) is a bus interface for communication between different modules in a system. CAN bus have become a de facto standard in the automotive industry. In modern cars, most electronic control modules are individually connected to CAN bus, making intercommunication between the modules possible. Implementation of CAN bus is very important for troubleshooting vehicles, as the mechanic can connect to the car network through a standard OBD-II connector with diagnostic equipment. CAN uses differential signalling in order to establish message based communication at high speeds. The CAN 2.0 specification developed by Bosch is an open standard and defines default word lengths. As the ECU is the most important control system in an automobile and is crucial in engine diagnostics, it is no question whether or not to implement a CAN interface in our design.

### 5.5.1. Implementation of CAN in our project

Often, CAN controllers are embedded in microcontrollers used in ECUs. We do not have access to the CAN controller implemented in the Processing System of the Zynq-7000, as the interface pins are used for other communication standards. It is possible to implement external CAN controllers that can handle the entire CAN 2.0 protocol; however, this will limit us to the embedded functionality of the controller. By implementing it entirely in VHDL in the Programmable Logic (PL) layer of the Zynq SoC, we have complete extendibility and versatility to our design and protocol, as well as limiting the need for hardware components down to a single, inexpensive CAN transceiver. The transceiver is needed to send and receive the differential signals of the CAN bus.

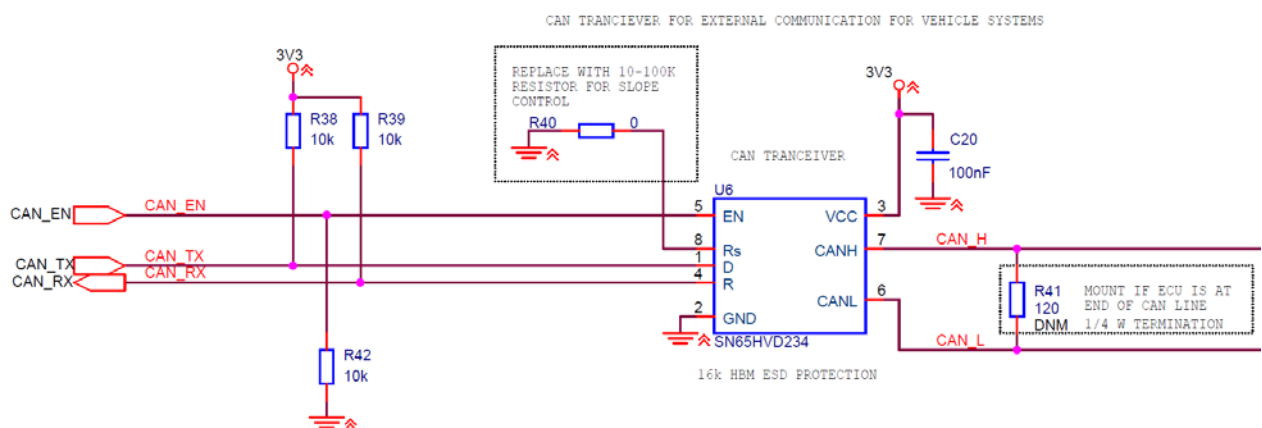


Figure 122 - CAN Transceiver schematic



Figure 122 shows the implementation of the CAN transceiver in our design. We have selected the SN65HVD234<sup>93</sup> from Texas Instruments. This is a widely used CAN transceiver with built in ESD protection for the level that we require in our application. The transceiver is a simple serial interface on the controller side of the system, denoted with signals CAN\_TX (Transmit) and CAN\_RX (Receive).

The Rs pin on the controller is pulled hard to ground, setting the device in high speed mode with no slope on the output signals. As commented in the schematic, the value of the pull-down resistor can be changed in order to adjust the slope of the output differential signal. The value to slope ratio is calculated using a formula found in the SN65HVD234 Datasheet. An option that it is not designed for is a hard pull up to VCC, which would enable device low power mode. This is not needed for our application.

The CAN standard requires a 120  $\Omega$  termination resistor at each end of the signal line. Where this termination is embedded is depending on the CAN layout in the vehicle. Often the end termination is embedded in the ECU, but in other cases it is an external insert at the end of the signal lines in order to allow expansion to the CAN bus. Due to not having any knowledge of the CAN bus in the vehicle the ECU are used, the termination resistor is not mount as a default and can be added only if needed.

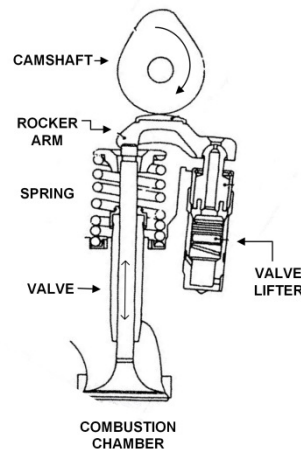
---

<sup>93</sup> SN65HVD234 Datasheet



## 5.6. Valve module

The pneumatic valve module (PVM) is an extension to the ECU that may be added to the engine control system in the future. The module should be capable of controlling the engine valves electronically using pneumatics to activate mechanical valve actuators. Valves are controlled by the engine camshaft by physically pushing spring loaded valves in order to control air/fuel intake and exhaust outlet, as shown in Figure 123.



**Figure 123 - Traditional valve mechanism principle<sup>94</sup>**

The engine speed varies according to the throttle position and engine load, and the valve opening and closing activation frequency also varies. If the valve closing and activation frequency is increased, a problem occurs with mechanical valve lifters; valve float. When valve float occurs, the valve remains open when it should be closed due to the “low” tension in the valve spring which reduces engine performance.

When speed increases, the air/ fuel mixture enters the combustion chamber later than it should. This decreases the performance, and hence especially the engine output torque. In automotive legacy systems, there is no possibility to change the valve timing during normal operation due to mechanical connection via the timing belt, as shown in Figure 124. In modern engines, the valve timing of inlet and outlet camshafts can be mechanically controlled individually to enhance the injection of air and fuel mixture to the engine. This increases performance, engine torque, and lowers the fuel consumption and the emissions.

<sup>94</sup> Figure from Volvo Personbilar Sverige AB, Grunder motorstyrsystem VT2201.



**Figure 124 - Timing belt<sup>95</sup>**

Controlling the valves electronically allows fully adjustable open- and closing times of each individual valve. This implies that the camshaft is no longer needed, thus eliminating losses of mechanical energy expended by the valve train (timing belt and sprockets). Precision timing of valves can increase overall fuel efficiency in the engine by optimizing the combustion process to a greater extent than in today's engines. There is however a major concern regarding the startup position of the valves (and engine) when using pneumatically controlled valves.

#### **5.6.1. Pneumatic valve concept evaluation**

Exactly how the design of the valve module will be performed is of course not determined by KPEC, as this is a possible task for future bachelor thesis groups. There are two possible communication concepts towards the pneumatic valves; module based or slave based.

##### **5.6.1.1. Slave based pneumatic valve control**

This implies ECU controlled operation of the activation of valve solenoids to open or close the valves. This mode relies on that the calculations and decisions for valve timings are performed accurately by the ECU, where timing is critical. The main information are transmitted to the valve module contains data of exactly which valve that must be opened and closed at any time. This makes the time delay from when data is received and processed from ECU a crucial factor.

In this operational mode, the ECU needs a dedicated channel for information flow to the PVM, as interruptions from other nodes in a communication network could cause time delays leading to catastrophic engine failures. A valve failing to close at the correct time causes the collision with the piston in the combustion chamber, possibly destroying the valve and results to complete engine breakdown.

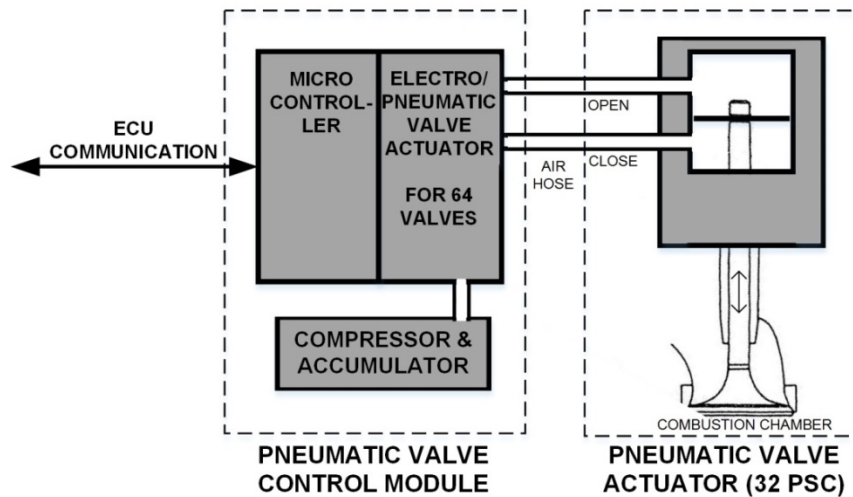
The constant dataflow of this operational mode will require the most from the communication interface between the ECU and the PVM, but requires less computational power from the PVM.

<sup>95</sup> Figure from Volvo Personbilar Sverige AB, Grunder motorstyrssystem VT2201.



#### 5.6.1.2. *Module based pneumatic valve module*

This implies that a dedicated module is implemented around the cylinder head to control an electro pneumatic control system for enabling or disabling the individual valve lifters as shown in Figure 125. This application requires a separate PCB which includes a microcontroller with fixed basic software stored in EEPROM. The calculations for valve timings and compensation are performed primarily in the PVM.



**Figure 125 - Pneumatic valve control module concept**

The ECU sends parameters such as position of engine crankshaft (critical), coolant temperature, velocity, exhaust gas temperatures and manifold air pressure in order for the PVM to optimally adjust valve timing. For startup the timing of when valves are opened and closed are computed in the PVM, while the ECU only feeds the required parameters and desired driving profile.

Driving profiles may be different modes of operation such as low emission or high performance drive. This mode of operation will demand more computational power from the processing system in the PVM, but will lessen the data rate and critical timing requirements of the interface between the ECU and PVM. Based on this assumption, this may be the best alternative mode of operation considering safety and adaptability.



### 5.6.2. Valve module interface requirements

It is challenging to estimate the requirements for the valve interface as it is not being developed at this stage, and design decisions for the PVM other than the interface to the ECU are not the scope of this project. The actual amount of data needed for the PVM controlled operation is also harder to estimate due to greater uncertainties about the required timing of when data has to be sent to the PVM. The PVM controlled operational mode should be less demanding on the interface. Based on these assumptions, an interface capable of handling the first mode shall also be able to handle the second mode of operation. Therefore, these calculations are only based on the required data from the ECU controlled operation.

#### 5.6.2.1. Engine valve update rate

The requirement for the KPEC ECU is to be able to handle engine speeds of up to at least 16 000 RPM. This is seen as worst case scenario in timing critical applications for the ECU.

$$\frac{16.000 \text{ RPM}}{60 \text{ s}} \cong 267 \frac{\text{rev}}{\text{s}} \quad (70)$$

First off is to convert 16 000 RPM (Revolutions per Minute) to 267 rev/s (Revolutions per second) as seen in (70).

A complete combustion cycle is completed in the span of 2 revolutions in a 4 stroke engine. In the first revolution, the intake valves are opened and closed, while the exhaust valves are opened and closed in the second revolution.

$$\frac{4 \text{ activations}}{2 \text{ rev}} = 2 \frac{\text{activations}}{\text{rev}} \quad (71)$$

Each valve opening or closing is computed as activations in (71). Note that this is the activation rate of the valves of a single cylinder.



The activation rate in (71) must be multiplied by the total amount of cylinders in the engine. In order to compute the activation rate of valves, (70) and (71) is combined.

$$8 \text{ cyl} \cdot 2 \frac{\text{activations}}{\text{rev}} \cdot 267 \frac{\text{rev}}{\text{s}} = 4272 \frac{\text{activations}}{\text{s}} \quad (72)$$

Equation (72) concludes that there will be approximately 4 272 valve activations per second for an 8 cylinder engine running at 16 000 RPM.

#### 5.6.2.2. **Data package size estimation**

It is difficult to estimate the size of each data package as we do not know the actual data that will be sent to the module at the point of implementation. The estimate is based *only* on the information needed in order to determine which valve to open or close, and no other information such as diagnostics or other parameters that may be needed etc. Therefore, this estimate shall only be used as a bare minimum requirement for the interface, meaning that there must be much overhead in the capabilities of the interface beyond this estimation.

The main features of each data package are as follows:

- Protocol bits
- Engine position
- Engine speed
- Valve number
- Valve open or close

The amount of protocol bits in a serial data transmission are typically 4. The first and last two bits are reserved for start- and stop-bits, while one bit is a parity bit.

Engine position is determined by the position of the crank shaft. Information about crankshaft position makes the ECU and PVM knowledgeable about the exact position of each piston in each cylinder. With a resolution of the crankshaft angle of  $0.1^\circ$ , a full revolution requires 3600 data points.

$$2^n = 3600 \quad (73)$$

$$n = 11.814 \cong 12 \text{ bits} \quad (74)$$





In order to be able to express this number, a minimum of 12 bits are required as seen in (74).

Engine velocity can be expressed by revolutions per second as seen in (70). Supporting up to 16 000 RPM means representing an engine speed of up to 267 rev/s.

$$2^n = 267 \quad (75)$$

$$n = 8.0607 \cong 9 \text{ bits} \quad (76)$$

In order to express the full range engine speed of 16 000 RPM, minimum of 9 bits are required as seen in (76).

Valve number requires 5 bits as required to express 32 in binary, while valve open or close is simply a single bit.

**Table 30 - 32-bit data package assignment**

	Protocol	Engine position	Engine speed	Valve# on/off	-
Bit	0 - 3	4 - 15	16 - 24	25-31	32

Table 30 shows what may be a typical 32-bit data package with all data bit calculations combined in this section.

In order to calculate the data rate required for the interface, the 32-bit data package required for each valve activation is multiplied with the valve activation rate.

$$32 \frac{\text{bit}}{\text{activations}} \cdot 4272 \frac{\text{activations}}{\text{s}} = 136704 \frac{\text{bit}}{\text{s}} \cong 136.7 \frac{\text{kbit}}{\text{s}} \quad (77)$$

Equation (77) shows the estimated data rate required from the PVM interface. As this estimate is made on rough assumptions, care must be taken when selecting an interface. As mentioned earlier, use of different communication protocols may add even more protocol bits, thus drastically increasing the required data rate. The PVM may also have to give a feedback signal to the ECU, making bidirectional communication a requirement.



### 5.6.3. CAN bus implementation

The use of CAN protocol could be a good solution for the PVM. CAN 2.0 specified by ISO 11898-6:2013 supports data rates of up to 1 Mbit/s, which suggestively is sufficient for our application.

The CAN protocol used in automotive applications adds several bits more to the protocol, which then again adds to the previous data rate requirements calculated in 5.6.2.2.

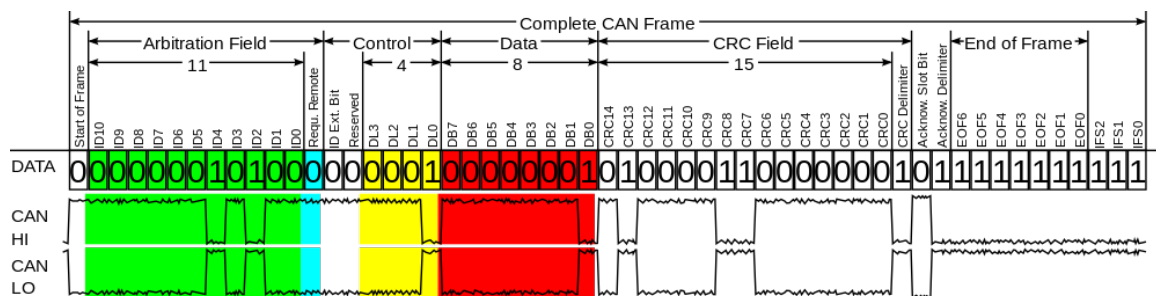


Figure 126 - CAN bus data frame<sup>96</sup>

CAN bus is message based and therefore require some sort of identifier for each message. There are two formats, one with 11 identifier bits, shown in green in Figure 126. The other format uses 29 identifier bits. The other bits are for several other control bits such as data transmission length and an identifier extension bit for separating messages with 11 or 29 identifier bits.

The data of the actual message is shown as 8 bits marked red in Figure 126. CAN 2.0 supports data fields of size ranging from 0 - 64 bits long. Previous estimations made in 5.6.2.2 suggested that the minimum amount of bits required for a message could be 32, although this number included protocol bits that would be disregarded using CAN. The main point is that a data field of up to 64 bits should support any data that would be needed for the PVM in any operating mode.

<sup>96</sup> Figure from Wikimedia Commons / CAN bus



### 5.6.3.1. CAN data rate requirements

Using CAN protocol for the messages will change the estimate made in 5.6.2.2. Subtracting the protocol bits would result in a minimum word length of 27 bits. The closest byte is still 32 bits (4 bytes), so this estimate continues to build on that number.

The CAN protocol alone requires a minimum of 44 bits using a 11 bit identifier (CAN 2.0A), and 64 bits for the 29 bit identifier (CAN 2.0B)<sup>97</sup>.

Recalling the result in (72), which yielded 4 272 activations/s at worst case of 16 000 RPM engine speed, the new data rate requirement can be calculated. The first assumption is transmitting 32 bits of data using an 11 bit identifier.

$$(32 + 44) \frac{\text{bit}}{\text{s}} \cdot 4272 \frac{\text{activations}}{\text{s}} = 324672 \frac{\text{bit}}{\text{s}} \cong 324.7 \frac{\text{kbit}}{\text{s}} \quad (78)$$

Looking at the result in (78), even exaggerating the required amounts of data bits yields a result well within the capabilities of the CAN bus, recalling that the maximum data rate supported by CAN 2.0 is 1 Mbit/s.

For good measure, the most extreme case possible can also be calculated. This means using the 29 bit identifier and transmitting a 64 bit message

$$(64 + 64) \cdot 4272 \frac{\text{activations}}{\text{s}} = 546816 \frac{\text{bit}}{\text{s}} \cong 546.8 \frac{\text{kbit}}{\text{s}} \quad (79)$$

The most extreme case, although unlikely to be applicable is still well within the boundaries of the CAN bus as calculated in (79). However, using up to more than half of the total bandwidth of an existing CAN bus with many nodes could result in issues with other modules. Therefore, the PVM should use a dedicated interface.

When cranking the engine, the PVM should have a default state after handshake command is send through CAN. This state should put the valves in a center position with injection and ignition disabled until the engine position is confirmed by the PVM. When the valves are synchronized with the crankshaft, a timestamp should be provided to enable ignition and injection, and hence start the engine. Messages on the CAN-bus will then only regulate the position of the valves according to engine load.

<sup>97</sup> ISO 11898-6:2013



## 5.7. General purpose digital I/O

Our system operates at 3.3 V and to be able to communicate with other systems that operate on 5V we need to level shift 3.3 V to 5 V and vice versa. To do this we use a voltage translator. The device we are using is from Texas Instruments, the SN74LVCT45<sup>98</sup> which is a dual bit dual-supply transceiver. The device has dual supply,  $V_{CCA}$  and  $V_{CCB}$  and 4 inputs, A1, A2, B1 and B2. The principle operation of this device is that A1 and A2 tracks  $V_{CCA}$  while B1 and B2 tracks  $V_{CCB}$ . The component also has a DIR pin, which is the direction pin. This pin is HIGH if you want to translate from voltage A to voltage B, and LOW if you want to go from B to A. We are connecting  $V_{CCA}$  with 5 V and  $V_{CCB}$  with 3.3 V, this allows us to translate and communicate with other systems that operate at other voltages. We are implementing 4 general purpose digital I/O's. As default all I/O's are pulled down to ensure low on all pins. The DIR pin is also pulled down, enabling voltage translation from B to A; this implies that 3.3 V get translated to 5 V. The implementation is seen on Figure 127 and a more detailed pin configuration is seen on Table 31.

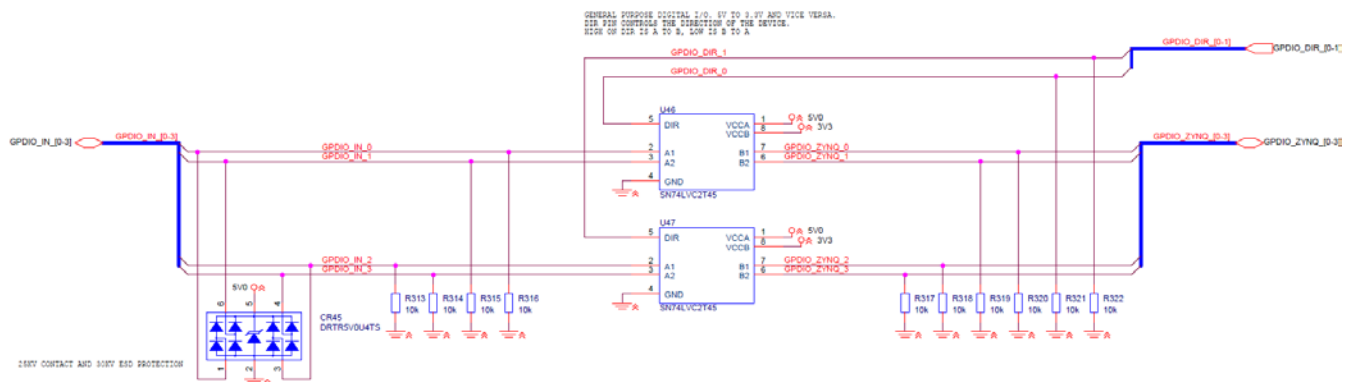


Figure 127 - Voltage translator

Table 31 - Pin configuration

PIN	TYPE	DESCRIPTION
NO.	NAME	
1	$V_{CCA}$	A-port supply voltage
2	A1	Input/output referenced to $V_{CCA}$
3	A2	Input/output referenced to $V_{CCA}$
4	GND	Ground
5	DIR	Direction control input
6	B1	Input/output referenced to $V_{CCB}$
7	B2	Input/output referenced to $V_{CCB}$
8	$V_{CCB}$	B-port supply voltage

The inputs are ESD protected at 30 kV air and contact discharge.

<sup>98</sup> SN74LVCT45, voltage translator



## 6. Power design

The power design is an important part of any electronic design. Generally, in electronics such as battery powered devices, everything is powered from the same voltage source. However, a voltage source such as a battery is not an ideal source, meaning that the voltage across the battery cells varies with battery charge. In automotive applications, everything is powered from the same voltage source, the car battery. For small vehicles, the car battery voltage is nominally 12 V; however the actual voltage range delivered to the electronic systems varies much from this depending on different factors. Under normal operating conditions, the car battery voltage ranges between 9 - 16 V depending on temperature and charge levels. Car batteries are tested by the SAE J537 standard, which specifies that the car battery voltage shall never be less than 7.2 V at -18 °C. Bad batteries with failing battery cells or extreme temperatures during cold cranking may reduce the battery voltage even further. Cold cranking is described more in detail in 6.2.1.

The different components on the circuit board have different operational voltages. Some components, such as ADCs, operational amplifiers and FPGAs often require more than one operational voltage level in order to function. The individual supply voltages must be of stable nature in order for the components to function properly. Regulated supplies compensating for the varying input voltage must be designed in order to deliver steady supply voltages to the board components under all conditions.

There are different ways to generate multiple stable voltage levels from a single source, which is the main focus in this chapter. Other considerations regarding power design such as protection circuitry is also described in this chapter

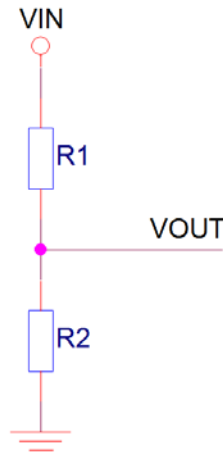
### 6.1. Power design strategies

There are multiple ways of designing voltage supplies. There are pros and cons to every design topology, and different design strategies must be weighed against each other. When simulating and perform basic circuit analysis, we assume the use of an ideal voltage supply. An ideal supply is capable of producing a completely steady voltage level with zero ripple and noise, regardless of the current load. Ideal supplies have no voltage or current limits, meaning that we can expend an infinite amount of power for our circuit. Of course, this is not possible in the real world. Different types of voltage regulators pose different challenges regarding design complexity and effectiveness. Generally, a voltage regulator that converts an input voltage to a steady voltage output is named DCDC converter, as it converts one DC voltage to another. The subchapters in this section describe a few types of voltage regulators, and where and why they are used in our design.



### 6.1.1. Voltage divider

A voltage divider does not regulate output voltage. The output voltage of a voltage divider is simply just some ratio of the input voltage given by the resistance ratio of the resistors used in the design.



**Figure 128 - Voltage divider circuit**

Figure 128 shows the basic voltage divider circuit. The equation for calculating the output voltage for a given input voltage is:

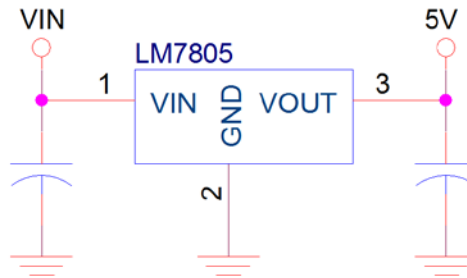
$$V_{out} = \frac{R_2}{R_1 + R_2} V_{in} \quad (80)$$

Voltage dividers cannot be used for regulating supplies because the output is directly proportional to the input with the ratio of the dividing resistors. The output cannot be larger than the input. With a stable voltage input, a voltage divider could theoretically provide a stable output level. The current that is pulled from the circuit will increase the power loss in R1 in Figure 128 and thus be very inefficient for any notable supplied current. The power dissipated in R1 would also generate heat and require large power resistors. For these reasons, voltage dividers are never used as voltage supplies, but rather only used stepping down signals with a known voltage level and very low power.



### 6.1.2. Linear regulator

Linear regulators are fairly simple in use, very inexpensive and require little external components in order to operate. A linear regulator will give a constant, often fixed voltage level for as long as the input to the regulator remains above a minimum level with respect to the output voltage. Linear regulators induce very little noise to the circuit and provide very stable output voltages.



**Figure 129 - Linear regulator**

A typical circuit implementation of a linear regulator is shown in Figure 129. The main drawback of linear regulators is efficiency. The linear regulators efficiency decreases with the difference between in- and output voltage. The voltage has to typically be at least 2 V higher than the output voltage. An exemption to this is Low-Dropout regulators (LDOs) where the input voltage can much closer to the output voltage than in classic linear regulators. A voltage input above the output voltage is essentially *wasted*, thus making them unusable for high current applications, as well as low power designs. Linear regulators are good for use in low current and low noise circuits, for example as voltage reference for an ADC or DAC. An example where a linear regulator is used for a voltage reference is described in 3.8.2.2. An example of how inefficient a linear regulator is can be calculated as follows:

$$P = (V_{in} - V_{out}) \cdot I_{out} \quad (81)$$

For example, if  $V_{in}$  in Figure 129 is 12 V, typical for a car battery and the current draw is 1 A, the power dissipated in the 5 V regulator can be found by inserting values into (81):

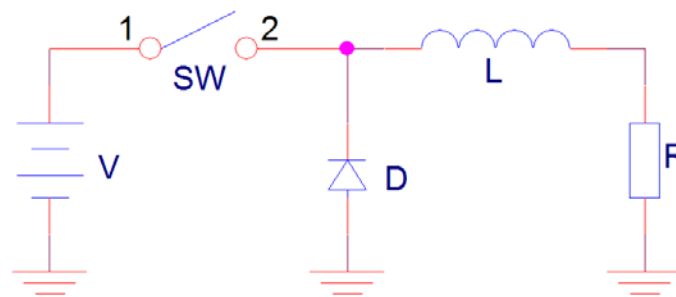
$$P = (12 - 5V) \cdot 1 A = 7 W \quad (82)$$



7 W of wasted power would generate much heat and require a heatsink for the regulator. This would then require much board space and increase the temperature of the entire board if placed in an enclosure.

### 6.1.3. Switched-mode Power Supply (SMPS)

Switched-mode power supplies (SMPS), often called DCDC converters is a range of different types of DC voltage converters, with a switching element, a conductor and a diode as the main components of the principle design. There are many different topologies and types depending on the application and the desired output voltage in regards to the input voltage. The most common types of SMPS DCDC converters are buck and boost type supplies.



**Figure 130 - Buck converter principle circuit**

Generally, the converter functions by rapidly charging and discharging an inductor by switching the supply current on and off, inducing a voltage across the load different to the input voltage. A great benefit of SMPS DCDC converters is that it is possible to invert and boost voltage as well as stepping down, by using different design topologies. Boost type converters can either invert or generate a higher output voltage than the input voltage, without the use of a transformer. The buck converter only steps down the output voltage with respect to the input voltage. It is possible to combine the two main topologies into a single buck-boost converter for maintaining a stable output voltage in applications where the input voltage fluctuates both above and below the output voltage of the converter. The switching element is most often a MOSFET switched on and off by a controller IC. The controller IC monitors the output voltage of the device and regulates it by adjusting the duty cycle of the switching MOSFET.

The main benefit of SMPS DCDC converters are high efficiency. The resistive elements in the current path of the converter are primarily the switching element and the inductor. The switching element used in practical applications are power MOSFETs. In practical applications, an efficiency of 80 - 95% is achievable.

There are also several drawbacks of using SMPSs in the design. The switching element generates much noise. The current output itself carries a ripple due to the charge and





discharge cycle of the inductor, which in itself can be seen as supply noise. In order to minimize output current ripple, large filtering capacitors are used. The controller IC requires multiple external components. Many SMPS controllers embed some of the external components, such as the inductor in order to lessen the use of board space. High current supplies may require fairly large external components due to power loss and current capabilities, and increase cost of the design.

The noise generated by switching supplies is problematic in mixed signal circuitry, as the analog signals are vulnerable to noise. Because of the efficiency of SMPSs, they must be used for all the main supply voltages in the design capable of delivering several watts of power with low supply losses.

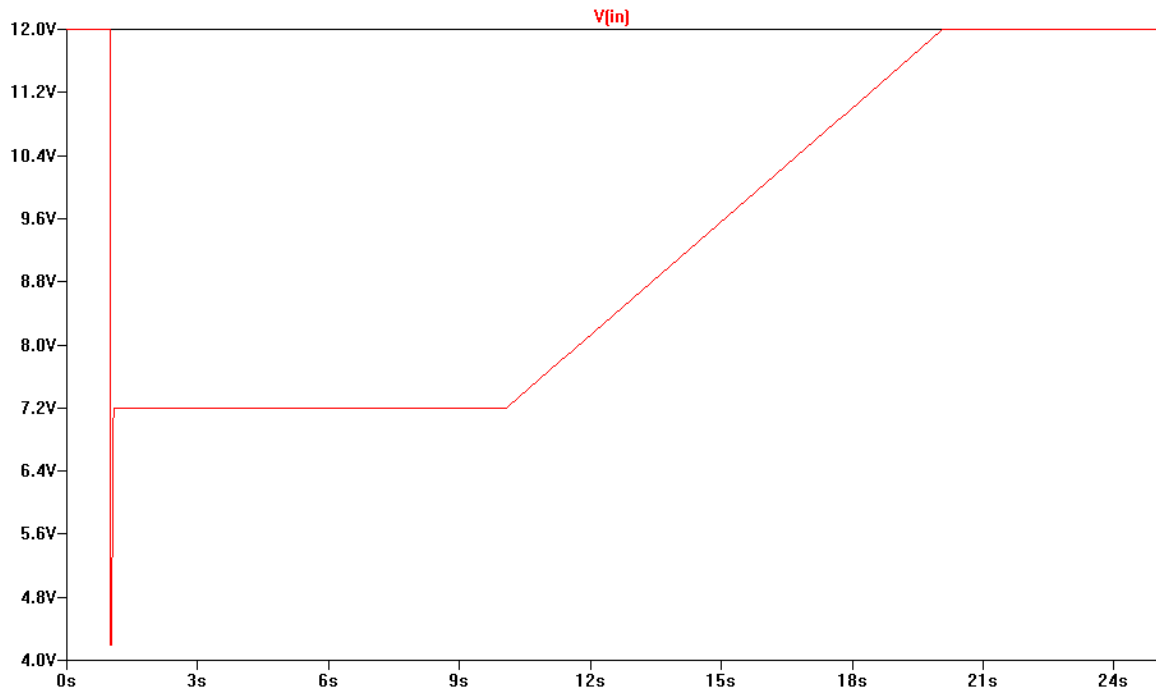
## 6.2. Automotive power design considerations

There are many considerations that must be taken into account when designing power management for electronics in automotive use. Automotive standards are strict in respect to high reliability and endurance in extreme conditions. As mentioned in the introduction to this chapter, the car battery voltage is not a stable, ideal DC voltage source. During normal operation, the car battery is constantly charged by the alternator. The alternator voltage is slightly higher than the nominal cell voltage of the battery in order to be able to charge the battery. The battery voltage is typically 12 V, while the alternator charging voltage is in the vicinity of 13 - 15 V. There are cases where the voltage can fluctuate even more than this, such as during cold weather and cold cranking the engine. Another thing is the noise generated by the alternator, which creates high frequency noise on the battery terminals, which must be filtered at the power input to the ECU.

### 6.2.1. Cold crank cycle

As described in the introduction, the requirements for new car batteries is tested in accordance to SAE J537 standard, which specifies that the car battery voltage shall never be less than 7.2 V at -18 °C. However, old batteries or batteries with damaged battery cells may in some cases reach voltage levels below this. Testing for temperatures as far down as -29 °C is optional according to SAE J537, and generally produces even worse results.

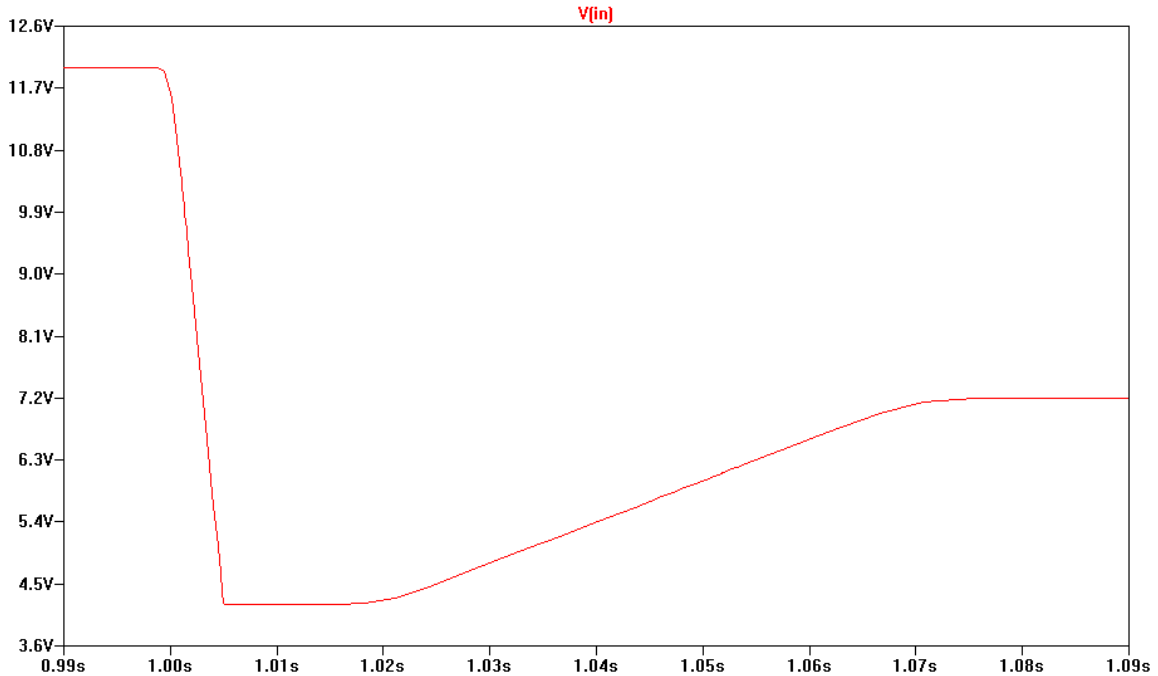
A special case when the battery voltage drops significantly is during cold cranking. When the engine is powered up, the electric starter engine cranks the engine until it can inject fuel and fire the coils to start running. Especially during cold start of the engine, the electric starter motor draws up to several hundred amperes from the battery, depending on its size. The sudden loading of the battery causes the battery voltage to drop significantly for some time.



**Figure 131 - Cold crank cycle**

Figure 131 shows a cold crank cycle. Note that the battery voltage drops further than 7.2 V required by SAEJ537. The cycle begins with a nominal 12 V battery level. When the starter engine starts, the large current draw causes the battery voltage to drop for some time, before it slowly returns to its initial 12 V value.

Especially critical is the initial start of the starter motor itself, as the moment it starts to turn requires an even higher peak input current. The start-up current of the starter motor causes the battery voltage to drop even further for a small timeframe and is seen as a transient load to the battery.



**Figure 132 - Starter motor start-up voltage drop during cold cranking**

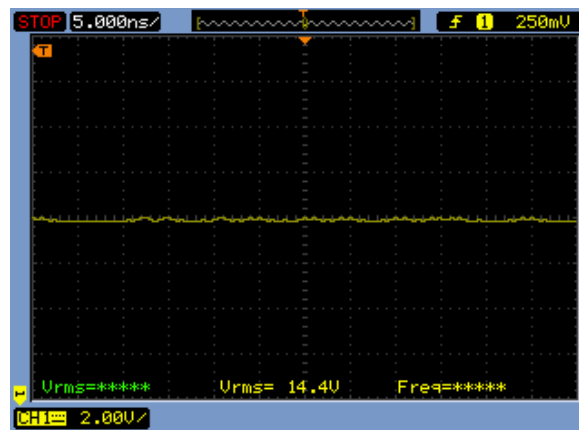
A close up of the initial voltage drop can be seen in Figure 132. As the starter motor starts to turn, the battery voltage drops down to about 4.2 V in about 5 ms, stays at this level for 15 ms and then slowly rising to 7.2 V in about 50 ms. Voltage drops as far as 4.2 V would really never occur unless during extreme temperatures with a severely discharged battery. Therefore, 4.2 V is not an absolute requirement of the KPEC ECU; although the power design circuitry is designed for handling voltages below the specified minimum level in SAE J537.

These crank conditions would almost never occur in real life, and is only a basis for testing the power circuitry for worst-case scenarios. In order to demonstrate a more typical use, we measured the crank voltage drop during normal conditions on an actual vehicle.



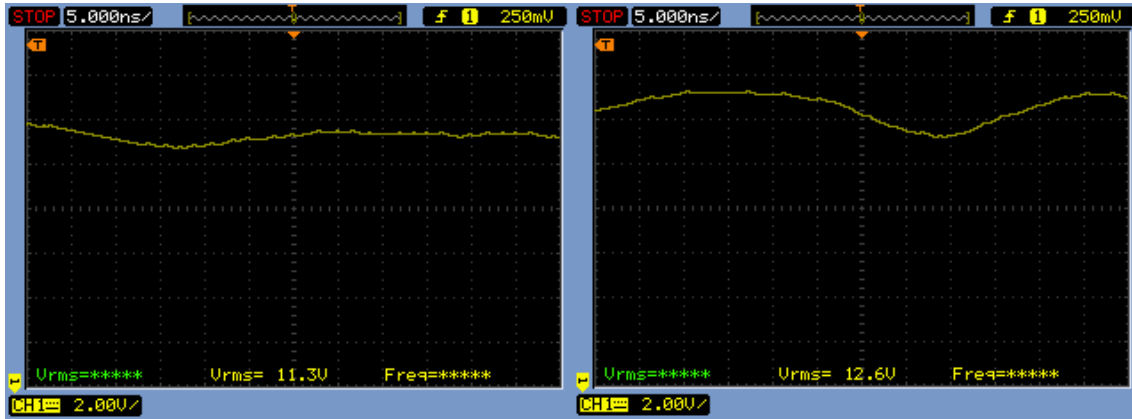
**Figure 133 - Audi A4 battery voltage testing**

A digital oscilloscope was connected to a 2007 Audi A4 2.0 diesel engine. The battery was fairly large and well charged. The engine was initially cold, and the outside air temperature was 17 °C and the weather was dry.



**Figure 134 - Battery charging voltage**

When the car engine was idling, the dynamo had a charge voltage of 14.4 V measured with the oscilloscope as seen in Figure 134.



**Figure 135 - Starter motor crank loading**

Figure 135 shows two oscilloscope measurements of cranking the engine. A very brief drop down to 9 V was observed but not recorded, while the typical cranking voltage drop was about 11 - 12 V as shown in Figure 135. Note that the battery voltage fluctuates rapidly when cranking the engine, compared to the measurement in Figure 134.

### 6.2.2. Cold cranking power budget

During cold cranking the battery voltage drops to 4.2 V for some milliseconds before slowly rising to 6 V, although SAEJ537 says that the battery voltage would never go lower than 7.2 V. As a precaution for worst case scenario we will let the battery voltage drop to 6 V. By doing this we have to calculate how much power the DCDC converters needs at 6 V. This information is used to design the protection circuit that will protect the DCDC converters and the entire motherboard. Table 32 shows the relationship between how much power is drawn depending on the efficiency of the DCDC converters. The relationship that is used is seen in (83).

$$W_{ACTUAL} = \frac{W_{DCDC}}{Efficiency} \quad (83)$$

As an example, let's say that the 3.3 V supply outputs 14.9 W. To determine how much the 3.3 V supply is actual draws from the battery (83), can be used if the efficiency for the DCDC converter is known. In this case the efficiency of the 3.3 V supply is 0.91%, by using (83), we can determine the actual power that is drawn from the battery is 16.3 W.

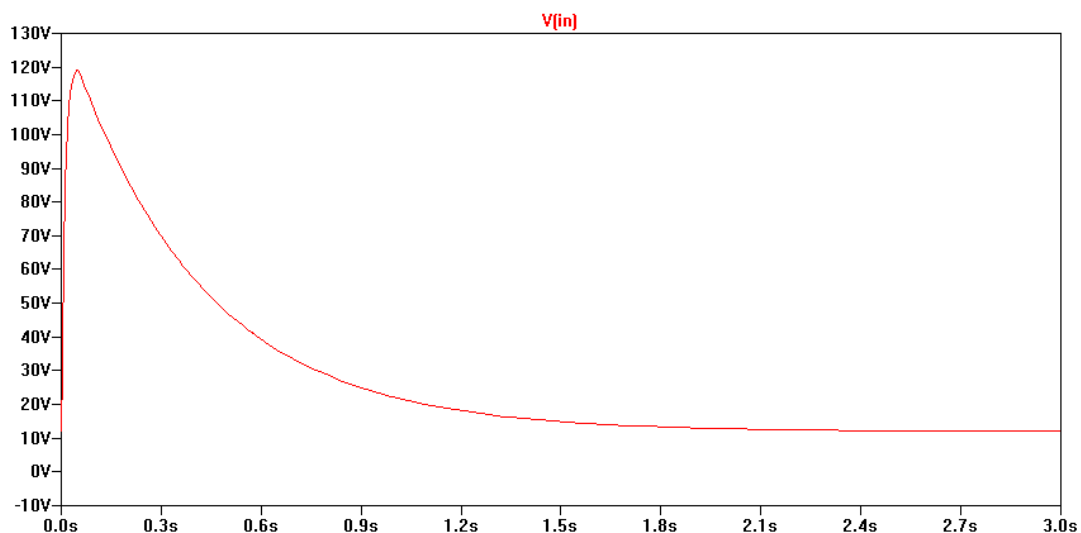
As seen in Table 32, during cold cranking (6 V) the DCDC converter draws 6.3 A from the battery, while at nominal voltage (12 V) its only 3.15 A.

**Table 32 - Effective power at 6 and 12 V**

V	A	W	W (Effective)	A @ 6 V	A @ 12 V
3.3	4.5	14.85	16.32	2.72	1.36
5	3	15	17.1	2.84	1.42
-5	0.6	3	3.41	0.57	0.28
15	0.025	0.375	0.43	0.071	0.04
-15	0.025	0.375	0.43	0.071	0.04
SUM	8.15	33.6	37.62	6.27	3.13

### 6.2.3. Load dumps

A phenomenon called a load dump occurs when an inductive load is suddenly disconnected in a system. The inductor holds large amount of current in the inductor windings, and when suddenly disconnected, the discharge of the inductor causes large voltage spikes in the system. In automotive systems, this occurs when the alternator disconnects, causing the alternator windings to rapidly discharge while the alternator itself continue to supply magnetic energy to the coils. The transients can be as high as 120 V, and therefore very damaging to low voltage semiconductor electronics such as the ECU.

**Figure 136 - Voltage spike caused by load dump**

The waveforms and magnitudes of the load dumps that automotive rated electronics are tested for are described in SAE J1113/11. Figure 136 shows a simulated typical voltage spike that can be caused by a load dump. The large voltage spike is fairly short-lived and usually decays within 500 ms, but a transient of this size and duration carries more than enough energy to cause damage to delicate semiconductor circuitry. Therefore, protection circuitry must be implemented in the power design of the ECU.



### 6.3. ECU power input protection design

Due to events such as load dumps described in 6.2.3, the ECU requires input protection circuitry. In addition to load dumps, reverse polarity protection is also required in case the user of the system connects the battery the wrong way. Load dumps are transient events, while the reverse polarity protection must be able to withstand a continuous negative voltage applied to the power input of the ECU. A set of requirements for the protection circuitry can be derived from these expected scenarios;

- Must withstand load dump transients of up to 120 V with a 500 ms fall time.
- Must withstand negative (reverse) battery voltage.
- Must function during critical engine cold crank cycles.

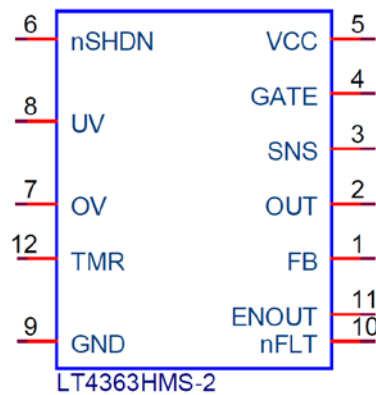
There are several ways to design for such events. Load dumps can be regulated by a Zener regulator circuit; reverse polarity protection can be a simple Schottky diode in series with the input voltage supply signal. The issue with such solutions is that the power dissipated is the voltage drop carried by the diode junctions is very high in high current applications. There is also no way to control a Zener regulator from overloading as it will attempt to regulate input voltage until the components reach their respective breakdown levels for power dissipation, effectively destroying the components. In order to increase efficiency and controllability during the events listed above; several types of power controllers have been developed for this purpose. Linear Technology has developed several controllers for such protection applications. As using their components allows simulation using LTSpice, they are a preferred components manufacturer.

#### 6.3.1. LT4363 surge stopper

The LT4363<sup>99</sup> is a high voltage and current limiting protection controller for use in automotive and avionic surge protection. The IC handles current monitoring, over- and under voltage monitoring. The working principle of the protection circuit is a switch opened or closed by a controller depending on the input voltage and passing current. In practice, the switch is an N-channel MOSFET, as these handle large current voltages with high efficiency compared to Zener regulators and Schottky diodes in series as mentioned above. The MOSFET used for controlling current supply to the rest of the ECU is named the *pass transistor*. For the test conditions above, the LT4363 was the most fitting choice in our application.

---

<sup>99</sup> LT4363 Datasheet



**Figure 137 - LT4363 package pinout**

The pinout of the LT4363 is shown in Figure 137. The LT4363 itself is a small 12-pin MSOP package, but the external power components require some board area. The output of the controller can be controlled by an external clamp in overvoltage events until the pass transistor is closed.

**Table 33 - LT4363 pin description**

Pin	Description
<b>nSHDN</b>	Active low shutdown pin
<b>VCC</b>	Power input
<b>UV</b>	Undervoltage threshold level. Adjusted by bias resistors to input power
<b>OV</b>	Overvoltage threshold level. Adjusted by a voltage divider from input power.
<b>TMR</b>	Timer capacitor pin. Value of capacitor determines fault warning time and cool down period
<b>GND</b>	Ground
<b>GATE</b>	Current passing MOSFET gate drive output
<b>SNS</b>	Current sensing input. Sense resistor is placed between SNS and OUT
<b>OUT</b>	Output voltage sense input. Sets the fault timer current
<b>FB</b>	Feedback input pin. A voltage divider sets the output adjustable clamping voltage
<b>ENOUT</b>	Enable output. Open collector and is held low until OUT is 0.5 V of VCC and 3 V above GND
<b>nFLT</b>	Fault open collector output. Is pulled low when the current pass transistor is about to be shut down.



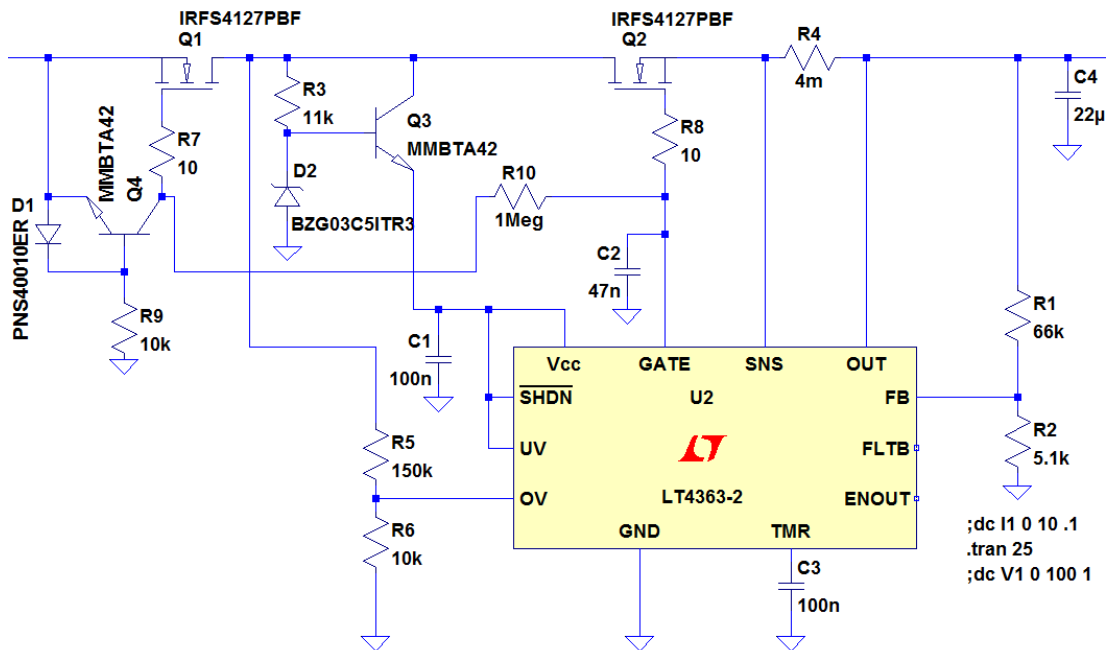


Table 33 shows the functionality of each pin on the LT4363. In order to select the external components, some parameters must be decided. The LT4363 is highly configurable, and the requirement parameters are used for calculating external control input component values.

The following subchapters describe the mathematics and reasoning behind selection the external components for use in the protection circuit. First of all, we know that the input battery voltage shall not exceed 18 V to the ECU due to measurement circuits and other circuitry relying on the protected battery input voltage. The overvoltage input prevents the controller from re-opening the MOSFET if the input voltage is too high even after the cool down timer has finished.

The parameters used for the design is as follows:

- Output clamping voltage of 18 V
- Maximum 6 A pass current
- 20 V input overvoltage limit
- No undervoltage limit



**Figure 138 - LT4363 simulation circuit**

The circuit implemented for simulation is shown in Figure 138. The reference designators for each component will be used in the equations for calculating the values below. Refer to Figure 138 for viewing circuit implementation of the calculated values.



### 6.3.1.1. *LT4363 control resistors selection*

The value of the feedback resistors  $R_1$  and  $R_2$  sets the adjustable output clamping voltage. The equation for finding these resistance values is given in (84). The output clamped voltage should be 18 V.

$$V_{REG} = \frac{1.275 \text{ V} \cdot (R_1 + R_2)}{R_2} = 18 \text{ V} \quad (84)$$

During overvoltage, the current through  $R_1$  and  $R_2$  should be about 250  $\mu\text{A}$ . The value of  $R_2$  can be calculated to be

$$R_2 = \frac{1.275 \text{ V}}{250 \mu\text{A}} = 5.1 \text{ k}\Omega \quad (85)$$

By rearranging (84) and inserting the value from (85),  $R_1$  can be found.

$$R_1 = \frac{(18 \text{ V} - 1.275 \text{ V}) \cdot 5.1 \text{ k}\Omega}{1.275 \text{ V}} = 66.9 \text{ k}\Omega \quad (86)$$

The closest E12 resistor series resulting in the answer from (86) is two 33 k $\Omega$  in series. Hence, the value for  $R_1$  is 66 k $\Omega$ .

For measuring overcurrent, the value of the shunt resistor  $R_4$  is found. The supplied current is at a critical stage during cold crank. For the On-board DCDC converters to deliver the same amount of power during cold cranking as during nominal battery voltage, the current drawn from the battery is much higher than normal. This is described in 6.2.2. The current sense resistor must be selected in order to allow the increased current draw from the battery during cold crank.

The current sense resistor can be calculated by inserting values into (87):

$$R_{SNS} = \frac{25 \text{ mV}}{I_{LIM}} = \frac{25 \text{ mV}}{6 \text{ A}} = 4.1667 \text{ m}\Omega \quad (87)$$



For the results in (87), a shunt resistor of 4 mΩ was selected, referenced  $R_4$  in Figure 138. This should allow about 6.25 A passing through the shunt resistor before entering overcurrent protection mode and disconnecting the current supply of the ECU. Overcurrent protection is useful for preventing short circuits inside the ECU without blowing external fuses, or protecting the ECU if a wrong valued fuse is used.

The overvoltage function is controlled by the OV input on the LT4363. The LT4363 enters overvoltage protection mode and clamps the output load to 18 V as specified above. The OV pin is a comparator input that activates for voltage values above 1.275 V. The LT4363 does not reattempt to turn on the pass transistor as long as the input on OV is above 1.275 V. A simple voltage divider formed by  $R_5$  and  $R_6$  in Figure 138 sets the trigger level for overvoltage condition.

The overvoltage is triggered at 20V inputs or higher as specified in the beginning of this section, for an output voltage to the OV input of 1.275 V at 20 V, the simple voltage divider calculation can be given by (88):

$$20\text{ V} \cdot \frac{R_6}{R_5 + R_6} = 1.275\text{ V} \quad (88)$$

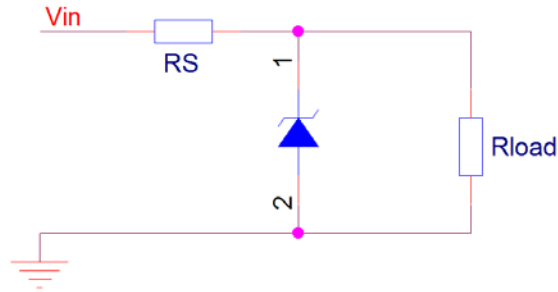
$R_6$  was chosen to be 10 kΩ. By rearranging (88), the value for  $R_5$  can be found.

$$R_5 = \frac{200\text{ k}\Omega - 12.74\text{ k}\Omega}{1.275} = 146.86\text{ k}\Omega \quad (89)$$

The closest E12 series standard value for  $R_5$  is 150 kΩ.

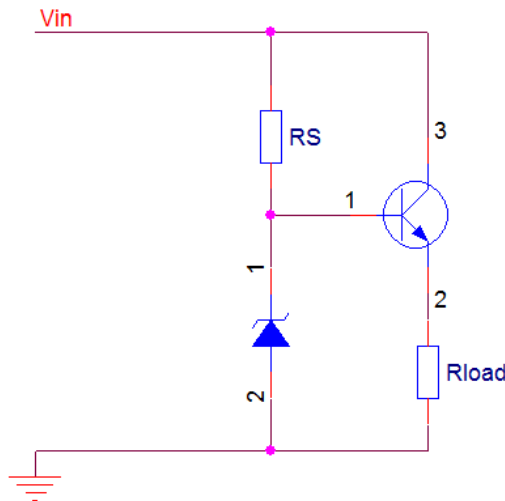
#### 6.3.1.2. **LT4363 supply voltage regulation**

Recalling the initial requirements for the protection circuit described in 6.3, the circuit must handle load dumps of typically 120 V. The LT4363 can handle up to 80 V on the input-side, so for managing transients above this, the input voltage must be regulated. There are two good ways of doing this with few components.



**Figure 139 - Zener voltage regulator**

Figure 139 shows a simple Zener diode voltage regulator and is a suggested solution in the datasheet. The Zener has a rated reverse breakdown voltage, so any voltage levels applied on  $V_{in}$  drops across the Zener as it passes its breakdown voltage and carries current in reverse. Efficiency is not really much of an issue in the power supply for the LT4363, as the current required is typically 0.7 mA at 12 V<sup>100</sup> voltage supply. The problem with the circuit in Figure 139, is that the voltage drop across the series resistor  $RS$  becomes too large at low input voltages such as during cold cranking. The LT4363 require a minimum of 4 V on  $V_{cc}$  to function. The drop across  $RS$  can cause the voltage supply to reach levels below this.



**Figure 140 - Zener emitter follower voltage regulator**

A way to solve this is to apply a Zener emitter follower voltage regulator. The circuit for a Zener emitter follower circuit is shown in Figure 140. The voltage equals  $V_{in}$  for voltages below the Zener breakdown voltage, but for any values above, the Zener conducts in reverse, and controls the emitter voltage of the BJT NPN transistor. The circuit is not very efficient at high loads, which does not matter much in this

<sup>100</sup> LT4363 Datasheet



application. The main advantage is that the voltage difference between input and output voltages are only affected by the PN-junction voltage drop in the BJT for values below the Zener diode breakdown voltage. The drop is typically 0.5 - 0.7 V depending on the transistor, which is acceptable during cold cranking conditions. The RS series resistor must carry enough current in order to provide enough base drive current,  $I_B$  for the transistor. The conducted current may be high at high input voltages, and simulation provided information that several resistors in parallel should be used for spreading the power dissipation across several resistors, as small surface mount components can't handle much power dissipation.

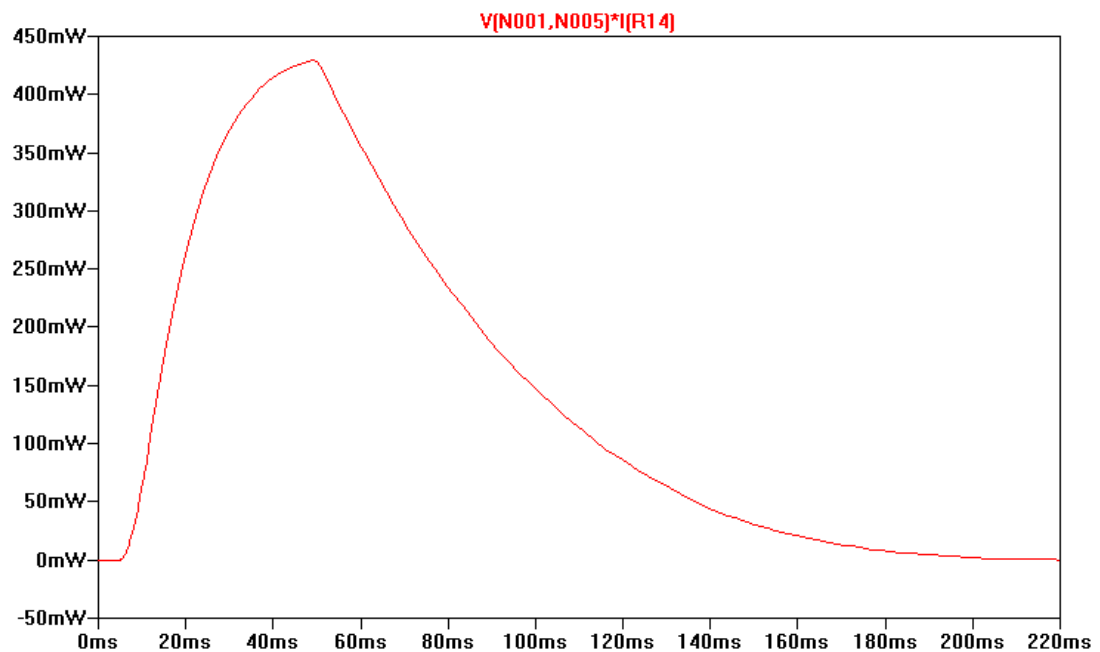


Figure 141 - Power dissipation in R3

It is desired to keep the resistance of  $R_3$ , the series resistor of high value in order to decrease power dissipation. Figure 141 shows the power dissipation in the 11 k $\Omega$  resistor at a 120 V input transient. Standard 0603 resistors are only rated for 100 mW. As resistors are derated by 50%<sup>101</sup>, 10 parallel resistors would be required in order to handle this power dissipation. The solution to this is to increase component size. 1206 resistors are typically rated at 250 mW; therefore 5 in parallel should be sufficient for handling the power dissipated, also considering derating. The closest E12 standard value using equal resistance values for parallel resistors are five 56 k $\Omega$  resistors, giving  $R_3$  a value of 11.2 k $\Omega$ .

<sup>101</sup> See component derating: Chapter 8



The transistor selected for the regulator is the MMBTA42<sup>102</sup> a widely available high voltage transistor produced by several component manufacturers. It is capable of handling 300 V between collector and emitter, making it a fitting choice for this design.

### 6.3.1.3. *Current pass transistor design*

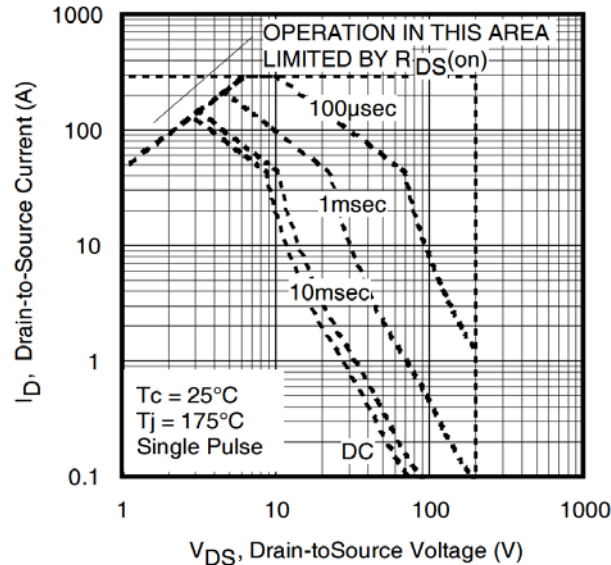
One of the most important components is the pass transistor,  $Q_2$  in Figure 138. An N-channel power MOSFET typically has a low  $R_{DS(ON)}$ , which is the internal on-resistance of the device and responsible for the power loss caused by the pass current. As the power loss and voltage drop in the protection circuit mainly depends on the pass transistors, high current MOSFETS with low  $R_{DS(ON)}$  are desirable. N-channel MOSFETs typically has a lower  $R_{DS(ON)}$  than P-channel MOSFETs. A good power MOSFET selected for use in the protection circuit is the IRFS4127PBF<sup>103</sup> from International Rectifier.

The MOSFET must handle the voltage drop caused by clamping the overvoltage during an overvoltage event. When overvoltage is detected, capacitor  $C_3$  at the TMR pin is charged. When the TMR voltage reaches 1.275 V, the FLT pin is pulled low indicating fault condition. The pass transistor is turned off when the voltage reaches 1.375 V. The value of  $C_3$  is critical for ensuring that the current pass transistor  $Q_2$  remains within its Safe Operating Area (SOA). The SOA describes the region the MOSFET can operate within for voltage and current loads for different durations. During a transient event, the voltage drop across the MOSFET  $V_{DS}$  is the input voltage minus the clamping voltage, creating a massive increase in the power that must be dissipated in the MOSFET for a short duration until it is turned off by the TMR function.

---

<sup>102</sup> MMBTA42 Datasheet

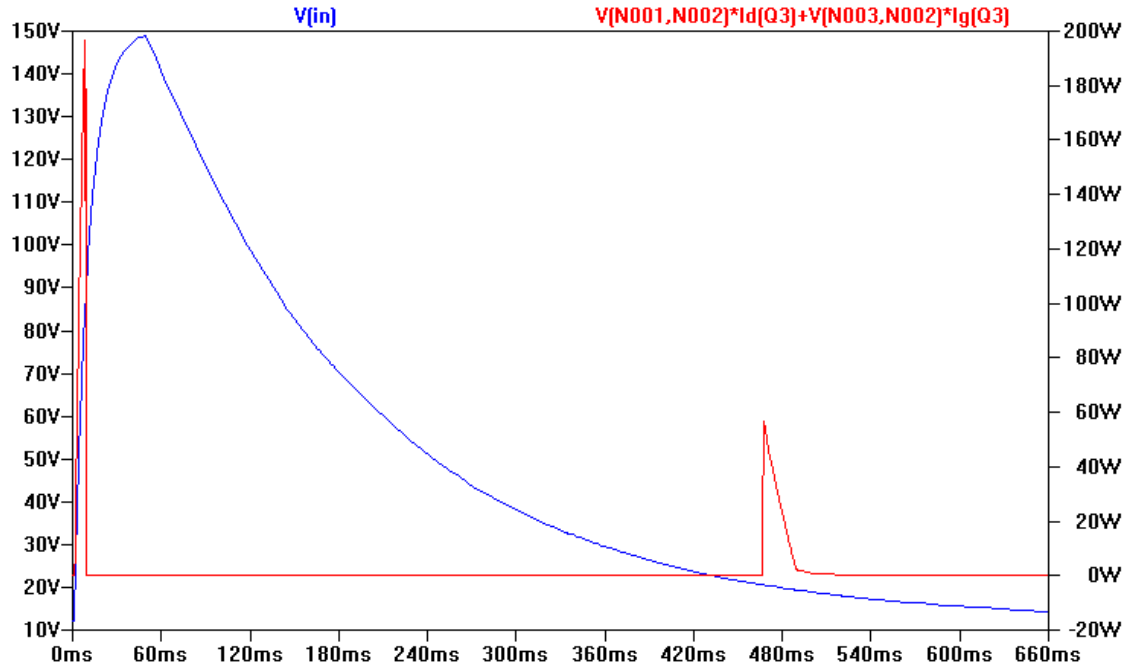
<sup>103</sup> IRFS4127PBF Datasheet



**Figure 142 - IRFS4127PBF Safe Operating Area (SOA)<sup>104</sup>**

For the SOA is used for non-interrupted operation during short transients. For very short transients, the MOSFET can handle the entire voltage drop and current load as long as it remains within the specified SOA. The SOA for the IRFS4127PBF is shown in Figure 142. Using information about SOA, the time the MOSFET must be shut down for different expected transient voltage can be calculated. For example, an instant clamping of 120 V input would cause about 102 V drop across  $Q_2$  due to the output being clamped to 18 V. Knowing that the regular current load is maximum 3 A at 12 V, the current load is about 2 A at 18 V clamped output. With this information, the SOA for this particular event can be found using Figure 142. At this load level, the MOSFET must turn off in about 200  $\mu$ s in order to operate within SOA. As there is a certain rise time for any voltage transient, the real SOA must be computed for different cases. This is best performed by simulation as the mathematics of computing the SOA requirements for a given transient curve is tedious. By LTSpice simulation of typical road vehicle load dumps, 100 nF for  $C_3$  TMR capacitor was deemed a fitting choice as this allows the ECU to operate during short transient events, and able to fully withstand longer load dump transients of up to 150 V by shutting the current pass transistor within time to remain inside the SOA.

<sup>104</sup> Figure from IRFS4127PBF Datasheet



**Figure 143 - LT4363 pass transistor power dissipation simulation result**

Figure 143 shows the simulation result of a 150 V input transient. The blue line represents the voltage transient at the input, while the red line is the power dissipation in the MOSFET,  $Q_2$ . The current pass transistor remains on for about 6 ms before shutting down. As the voltage drop across  $Q_2$  increase with the ramping transient, the power dissipation in the MOSFET increases rapidly. The peak power dissipation is 196 W before shutting down. During the transient, the circuit remains off until the transient voltage drops below the OV threshold set at about 20 V. The second power peak happens at this stage. The time between shutting down and re-enabling the ECU battery supply allows  $Q_2$  to cool. Short before shutdown, a warning timer pulls the FLT pin low in order to signal an impending shutdown of the pass transistor. For a given TMR capacitor of 100 nF, the warning time can be given by:

$$t_{warn} = 100 \text{ nF} \cdot \frac{100 \text{ mV}}{6 \text{ }\mu\text{A}} = 1.67 \text{ ms} \quad (90)$$

Simulations of different load dump scenarios proved that the warning time of 1.67 ms found in (90) is sufficient, however this should be vigorously tested in practice after board manufacture before fully approving the design, as the Spice simulations works with many ideal factors which are not true in real life.





The cool down time of the current pass transistor can be calculated as follows;

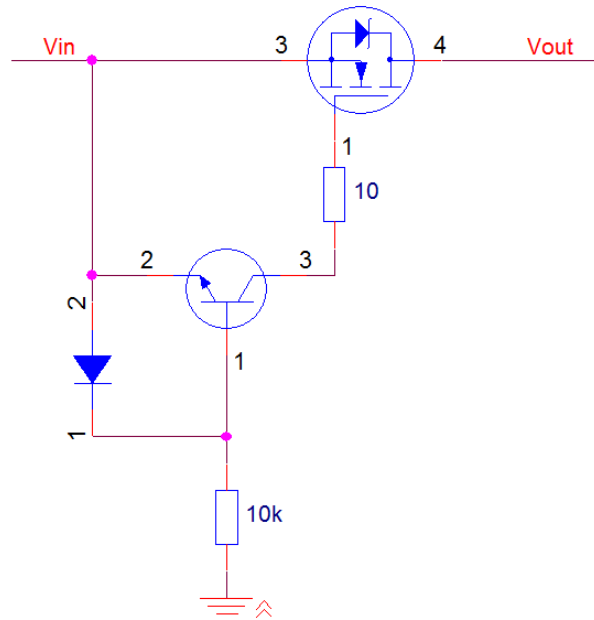
$$t_{cool} = 100 \text{ nF} \cdot \frac{2.925 \text{ V} + 3.8 \text{ V}}{2 \text{ }\mu\text{A}} = 336.25 \text{ ms} \quad (91)$$

Note that the cool down time in (91) is the minimum time before turning on the pass transistor. If overvoltage condition persists past  $t_{cool}$ , the pass transistor is inhibited from turning on.

#### **6.3.1.4. LT4363 reverse polarity protection**

Reverse polarity protection on VBAT input pins on the ECU connector must be implemented in case the user of the system swap the battery terminals at the car battery. Applying reverse voltage to fragile logic elements can damage them; therefore this must be protected against in the ECU.

The reverse protection circuit operates without any logic elements and is a standalone circuit in principle. There are two primary ways of implementing reverse voltage protection. The first method is using a P-channel MOSFET. As mentioned above, P-channel MOSFETs generally has a larger  $R_{DS(ON)}$  which would mean larger power losses. As the pass transistor is an N-channel transistor, it would also mean adding yet another component to the BOM. Another way of providing reverse input protection is placing an N-channel MOSFET in back-to-back configuration with the current pass transistor, with an additional BJT to regulate the gate voltage of the MOSFET.

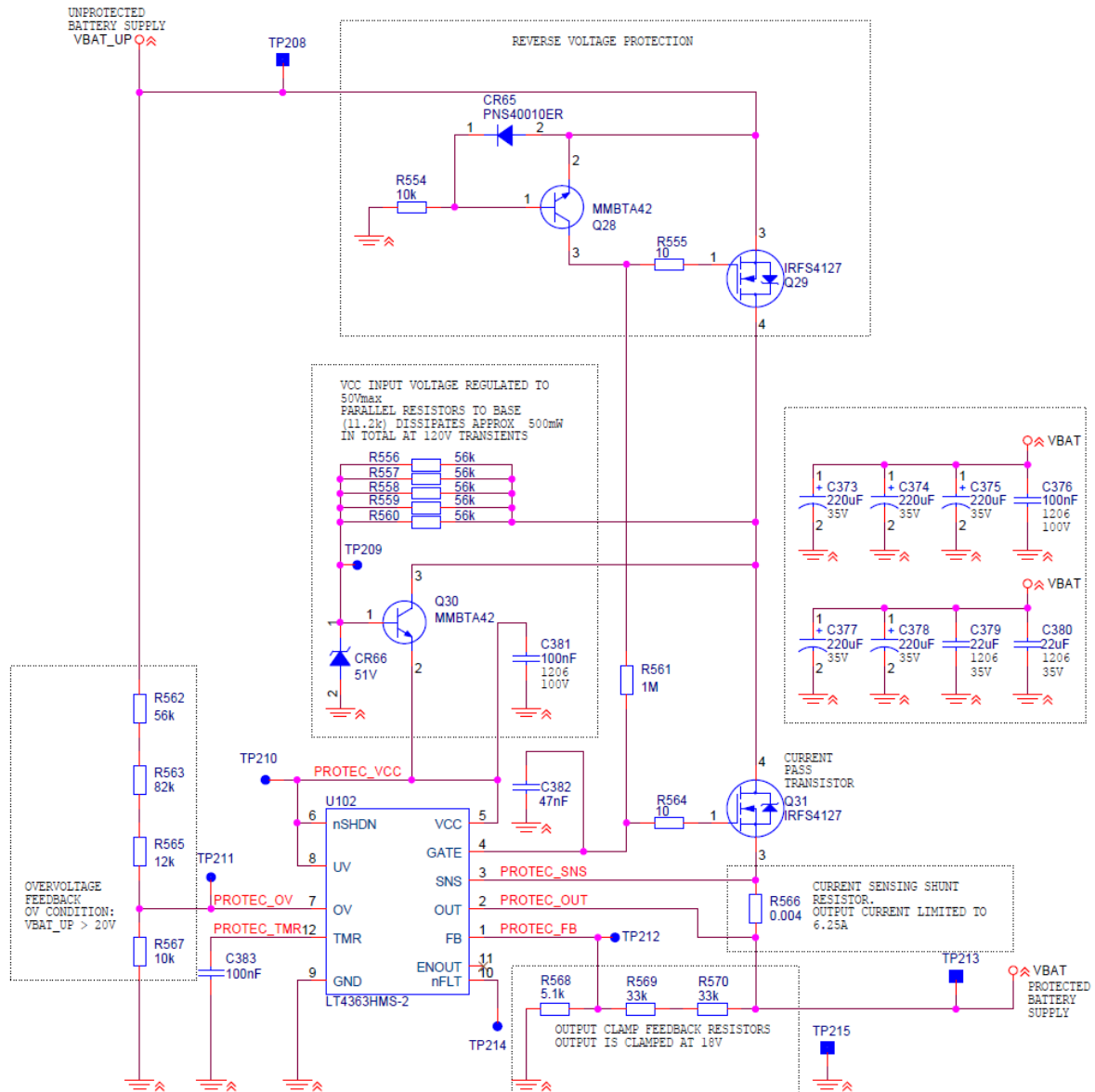


**Figure 144 - Reverse polarity protection circuit**

Figure 144 shows the principle reverse protection circuit.

When  $V_{in}$  is positive, the transistor emitter-base voltage is negative with respect to ground due to the forward voltage drop across the diode. This causes the BJT to remain closed, and the gate of the MOSFET can be opened to allow current flow.

When  $V_{in}$  is negative, the ground is at a higher level than  $V_{in}$  with respect to ground. The emitter-base diode blocks the negative voltage seen from the collector and the BJT turns on by the higher voltage referenced at the base than emitter. The negative voltage is then applied directly to the MOSFET, forcing it off as the gate voltage is approximately the same as the source voltage. The current flow is blocked when the MOSFET is in off-state, protecting any circuitry behind the MOSFET from voltage inputs below ground level.



**Figure 145 - ECU power protection circuit schematic implementation**

The schematic implementation of the ECU power protection circuit is shown in Figure 145. The 1 MΩ resistor between the back-to-back configured N-channel MOSFETs ensures that the gate voltages remain equal. The 47 nF capacitor on the gate drive pin of the LT4363 reduces inrush current to the gates of the MOSFETs.

The output bulk capacitance must be at least 10 times larger than the ceramic input capacitance of the DCDC converters. The total input capacitance of the DCDC converters are 110 μF, thus a minimum output capacitance of 1.1 mF is required. In order to meet this requirement, five 220 μF aluminium electrolytic capacitors were placed in parallel with two 22 μF ceramic capacitors, as electrolytic capacitors typically have a slower discharge rate than ceramic capacitors.



## 6.4. Main ECU power supplies design

This section contains the design decisions and test/simulations done to our power design. Different topologies are used for the different DCDC converters. After reading this section the reader should have a clear understanding about the design decisions, test/simulations and what topology has been done for the different power supplies.

### 6.4.1. Power budget and configuration

While designing the ECU, we had to estimate the power that was required from the ECU. The procedure for this was that we looked at each component and looked at their minimum, typical and maximum values. We added these numbers into a matrix and calculated the different total power that was needed. We also looked at which types of voltages that was needed, after going through the entire design, we concluded that we had to have 5 different power supplies. A 3.3 V supply to supply the Zynq and on-board components while positive and negative 5 V supplies to provide power to interfacing components. We also needed positive and negative 15 V to provide power to our external ADC and some operational amplifiers. Table 34 shows the power calculation that each power supply needs to supply to provide enough power to the ECU. This table is only a summary of the calculation; the entire calculation matrix is found in Appendix II.

**Table 34 - Power calculation summary**

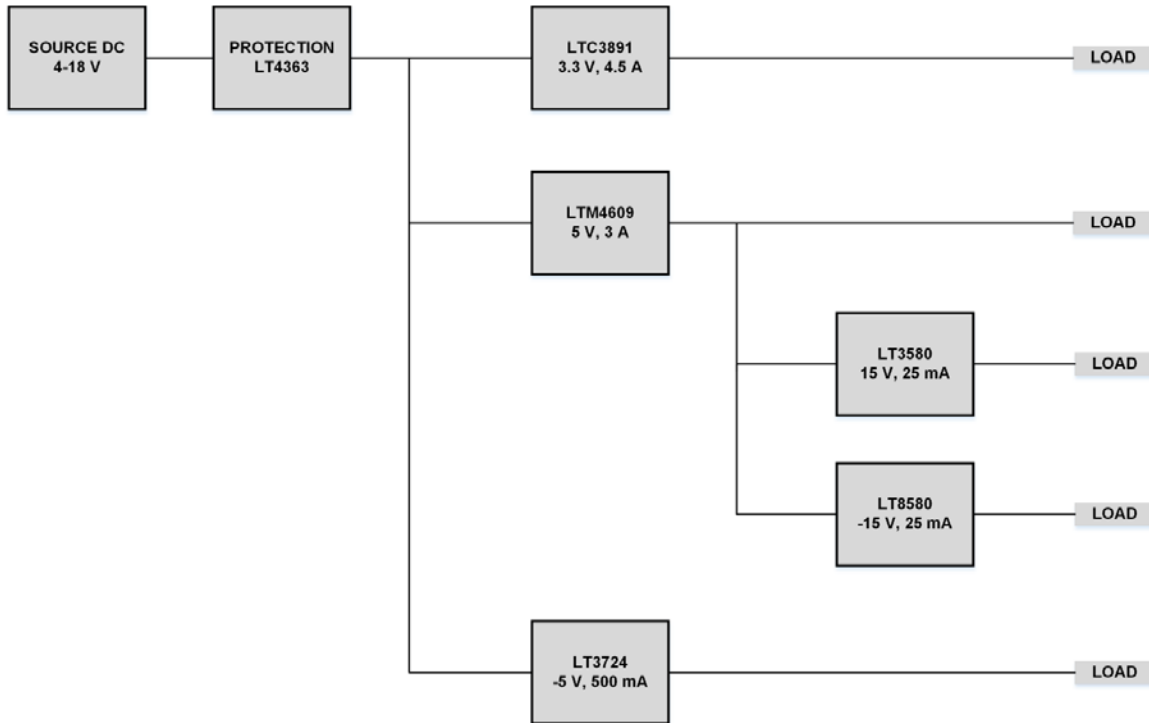
VOLTAGE	POWER (W)			AMPERES (mA)		
	MIN	TYP.	MAX	MIN	TYP.	MAX
<b>3.3</b>	0.16	5.4	9.4	47	1643	2841
<b>5</b>	0.039	3.7	6.1	7.8	780	1305
<b>-5</b>	0.0	0.08	1.0	0.0	131	265
<b>15</b>	0.0	0.15	0.20	0.0	10.1	13.6
<b>-15</b>	0.0	0.16	0.21	0.0	10.5	14.0
<b>SUM</b>	<b>0.19</b>	<b>9.49</b>	<b>17.1</b>	<b>54.8</b>	<b>2580</b>	<b>4444</b>

We are designing our power supplies after maximum rating. From which we are scaling the maximum rating with a fixed constant of 1.5. This enables us to have a buffer which is important when designing power supplies.

We have 5 power supplies that we need to design and also some requirements to follow. Some of these requirements are that the power supplies need to survive a cold crank cycle (discussed in 6.2.1), provide enough power to the ECU and that voltage regulation is within 5% at all times (Specified in derating 8.2.6. After some discussions and research we found out that since the positive and negative 15 V supplies does not output much power to the ECU



components, therefore we can connect them to the 5 V supply, leaving the cold crank problem to this power supply instead. This means that we now have 3 power supplies that need to survive a cold crank cycle test. A configuration drawing was drawn and can be seen in Figure 146.



**Figure 146 - Power configuration drawing**

As seen on Figure 146, the 3.3 V, 5 V and -5 V supplies is required to pass the cold crank cycle test to be approved.

Additionally, since the power supplies are taking their power from the battery, we need some sort of protection to ensure that our power supplies do not take damage if some transient occurs from the battery, load dump, or over and under voltage and even reverse voltage by reversing the battery poles. The protection circuitry is discussed in 6.3. The different power designs are discussed in detail in 6.4.2 (3.3 V), 6.4.3, (5 V), 6.4.4 (-5 V), 6.4.5 (15 V) and 6.4.6 (-15 V).



### 6.4.2. 3.3 V supply

The LTC3891<sup>105</sup> from Linear Technology benefits from a step-down topology, which translates the wide input voltage range; 6 - 18 V down to a reliable 3.3 V on the output of the power controller. Notice that the LTC3891 is a controller; it implies that the switching element is located outside the IC. The power design in Figure 147 shows both the inductor and the two switching MOSFETs, which the LTC3891 controls. A pin description is given in Table 35

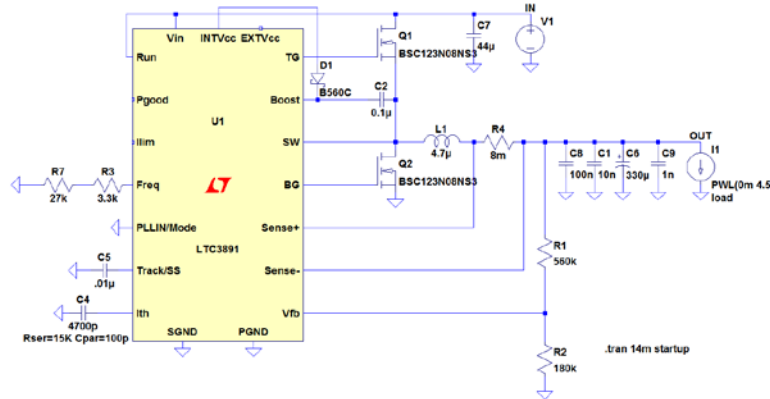


Figure 147 - 3.3 V 5 A supply

Table 35 - LTC3891 pin description

Name	NO.	Description
PLINN/MODE	1	External synchronization input to phase detector and forced continuous mode input
SGND	2	Small-signal ground
RUN	4	Digital run control input
SENSE-	5	The (-) input to the differential current comparator
SENSE+	6	The (+) input to the differential current comparator
VFB	7	Feedback voltage from resistive divider from output
ITH	8	Error amplifier outputs and switching regulator compensation point
PGOOD	9	Open-Drain Logic Output
TG	10	High current gate drives for top N-channel MOSFET
SW	11	Switch node connection to inductor
BOOST	12	Bootstrapped supply to the topside floating driver
BG	13	High current gate drive for bottom N-channel MOSFET
INTVCC	14	Output of the internal Linear Low Dropout Regulator
EXTVCC	15	External power input to an internal LDO connected to INTVCC
PGND	16	Driver power ground
VIN	17	Main supply pin
ILIM	18	Current comparator sense voltage range inputs
TRACK/SS	19	External tracking and soft start input
FREQ	20	The frequency control pin for the internal VCO

<sup>105</sup> LTC3891 Datasheet, 3.3 V supply



#### 6.4.2.1. **LTC3891 design decisions**

Both the MOSFETS are n-channel devices from Infineon, that are optimized for DCDC converters, that means that they have very low impedance which allows them to dissipate just a small amount of power, the result is better efficiency and power capabilities.

The wide input voltage is filtered through the input capacitor of 44  $\mu\text{F}$  (see 6.4.7 for calculations) before it supplies the top MOSFET and the VIN pin on the controller, the voltage on the RUN pin can be chosen to shut down the main control loop of the controller or to disable the whole controller and most of its circuitry. For this application the controller is activated continuously.

We have left the EXTVcc pin floating to activate the internal LDO (Low Dropout regulator), the Vin pin supplies the internal LDO to deliver 5.1 V on the INTVcc pin. INTVcc supplies both the top and bottom MOSFET, and most of the other circuitry.

The controller operates in buck mode, because the voltage on the output is always lower than the supplied voltage. The top MOSFET is PWM modulated to charge the inductor through each cycle. When the top MOSFET is turned off, the bottom MOSFET opens, to drain the inductor if a reverse current appear before the next cycle is initiated. This is to prevent the buck controller to perform as a boost controller, as any voltage in the inductor when the next cycle on the top MOSFET is initiated, will add on the MOSFET voltage, and thus acting as a boost converter.

The sense resistor can be chosen by using (92).

$$R_{sense} = \frac{V_{sense(max)}}{I_{MAX} + \frac{\Delta I_L}{2}} \quad (92)$$

To ensure full current capability over the entire temperature range, the minimum value for the  $V_{sense(max)}$  in the electrical characteristics should be chosen for the given  $I_{LIM}$ , which is floating in our case.  $I_{MAX}$  is not listed in the electrical characteristics table but can be found with simulation to be 7.5 A, the inductor size can be found in (93). With an inductor ripple of approximately 200 mA, we get:

$$L = \frac{1}{f \cdot \Delta I_L} V_{out} \left(1 - \frac{V_{out}}{V_{in}}\right) \quad (93)$$



Inserting numbers to (93) we find the inductor value;

$$L = \frac{1}{250 \text{ kHz} \cdot 200 \text{ mA}} \cdot 3.3 \text{ V} \left( 1 - \frac{3.3 \text{ V}}{12 \text{ V}} \right) = 47.85 \mu\text{H} \quad (94)$$

The closest value of 47.85  $\mu\text{H}$  is 47  $\mu\text{H}$ , which gives us a inductor ripple of 229 mA.

From the datasheet, the minimum  $V_{sense(MAX)}$  is 64 mV and the  $R_{sense}$  resistor can be calculated using (95).

$$R_{sense} = \frac{64 \text{ mV}}{7.5 \text{ A} + \frac{229 \text{ mA}}{2}} \cong 8 \text{ m}\Omega \quad (95)$$

The voltage feedback  $V_{fb}$  is used to configure the output voltage of the controller, this power is designed for 3.3 V and therefore the voltage divider can be calculated with (96).

$$V_{out} = 0.8 \text{ V} \left( 1 + \frac{R_B}{R_A} \right) \quad (96)$$

$$R_B = 180 \text{ k}\Omega \left( 1 - \frac{3.3 \text{ V}}{0.8 \text{ V}} \right) = 560 \text{ k}\Omega \quad (97)$$

We can see from (97) that the resistors  $R_A$  is 180 k $\Omega$  and  $R_B$  is 560 k $\Omega$  will set the output voltage to 3.3 V

Both the frequency, input voltage and the inductor affect the current ripple as seen in (93). The frequency as implied earlier is set to approximately 250 kHz with a 40 k $\Omega$  resistor, which increase the inductor size but also increase the efficiency of the circuit due to lower switching in the MOSFETs and thus reducing the power dissipation in the MOSFETs.

The controller support different modes of operation from the PLLIN/MODE pin, which can be connected to an external clock to achieve synchronization of an array of power supplies. For now, we have chosen to operate the controller in burst mode, which determines the response of the controller at light loads, there is also possible to operate the controller in continuous inductor current mode and pulse skipping mode.





In order to avoid large surge currents when the LTC3891 is powered up, there is an option to enable a soft start function, which the designer can choose the amount of time for the controller to reach its regulated value, in our case 3.3 V. The soft start function can be achieved with a capacitor between TRACK/SS and ground, and the value for the capacitor can be calculated with (98).

$$t_{ss} = \frac{C_{ss} \cdot 0.8 \text{ V}}{10 \mu\text{A}} \quad (98)$$

If we desire a soft start time of 800  $\mu\text{s}$ , we find in (99) that the soft start capacitor is;

$$C_{ss} = \frac{t_{ss} \cdot 10 \mu\text{A}}{0.8 \text{ V}} = \frac{800 \mu\text{s} \cdot 10 \mu\text{A}}{0.8 \text{ V}} = 10 \text{ nF} \quad (99)$$

The ITH pin is connected to a comparator located inside the LTC3891; the comparator measures the voltage at  $V_{fb}$  which is measuring the output voltage at the sense resistor and comparing it to an internal 0.8 V reference. Whenever the load current increases, the  $V_{fb}$  will decrease slightly compared to the 0.8 V reference and thus increase the ITH voltage. The ITH voltage can then be measured by external circuitry to monitor the load current of the regulator.

The maximum and minimum values for the regulator are listed in Table 36.

**Table 36 - Voltage thresholds**

	Min.	Typ.	Max.	Unit
Vin	4.5	12	18	V
Vout	3.2	3.3	3.4	V
Iout	-	5	-	A





#### 6.4.2.2. Crank test simulation

To test the power circuit under cranking conditions, we are simulating that the input voltage is initially 12 V, and rapidly drops to 4 V when the cranking is initiated, after some time the engine ignites and start to generate power to the battery, and the battery voltage returns to 12 V. The battery voltage profile is a bit exaggerated, but is intended to simulate cranking during cold winter weather with a weakened battery.

We are running test at 4 V to assure that we will handle a voltage drop to 6 V. If the supply manages to regulate when the voltage drops to 4 V, we have assured that the supply will handle a drop to 6 V, which is in accordance to SAEJ537.

The reason for this though battery profile, is based on the desire to operate the core function circuitry of the ECU during cold cranking, so the engine is able to start and thus generating enough power to supply the remaining power supplies and circuitry.

The 3.3 V power supplies the Trenz micromodule, which controls the regulation of the combustion engine, it is therefore essential that the 3.3 V power will manage the demanding circumstances that appear in automotive applications.

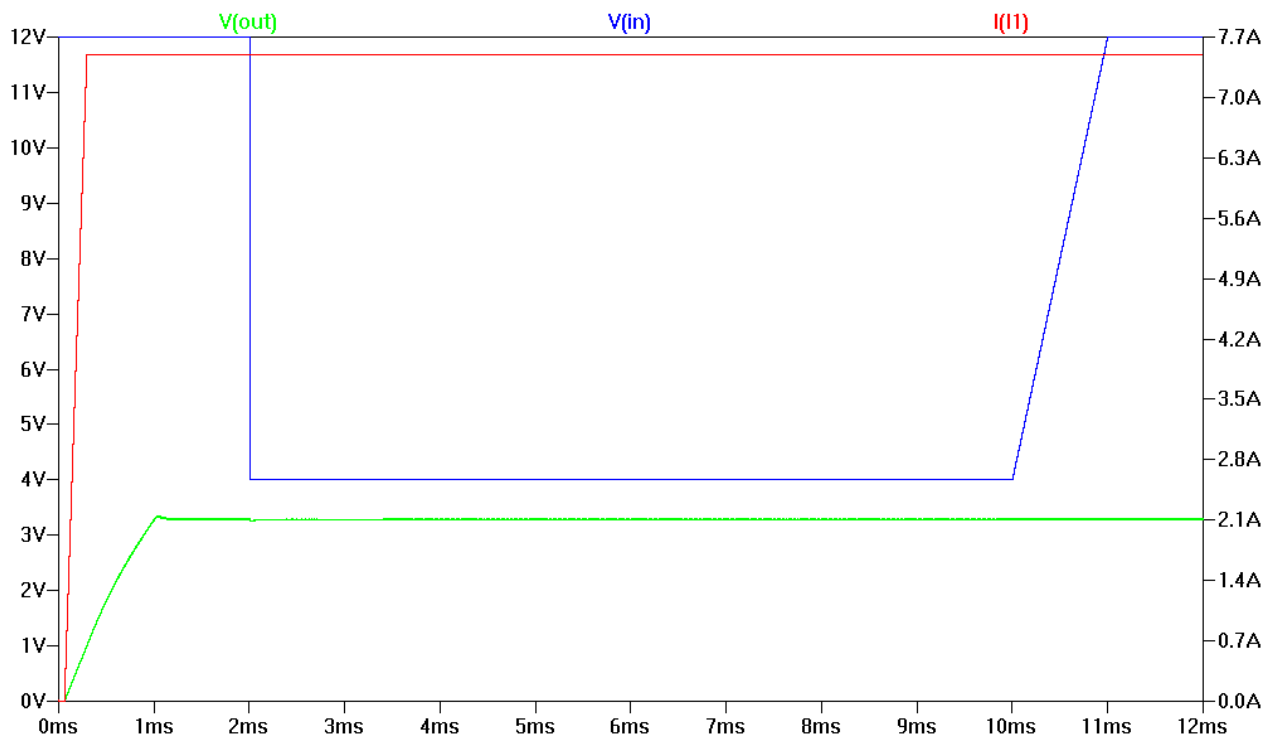


Figure 149 - Crank test simulation in LTSpice

Figure 149 shows the simulation executed in LTSpice of the circuit in Figure 147, where the blue line indicates the battery voltage during cold cranking. The red line indicates the current capabilities during full load, and finally the green line indicates the output voltage of 3.3 V during cold cranking with full load.



#### 6.4.2.3. Load test simulation

Figure 150 shows the load simulation. The power is fired up and stabilized with 5 A load. The blue line indicates the output voltage of the 3.3 V power and the green line indicates the amount of current that the load dissipate. The 3.3 V power supply is operating normally when suddenly a 8 A load is connected to the power supply, a cursor is pointed at the bottom peak level of the output voltage, where we can see from the box that the voltage level is 3.16 V. To have good regulation the voltage regulation cannot be off with more than 5% from desired value.

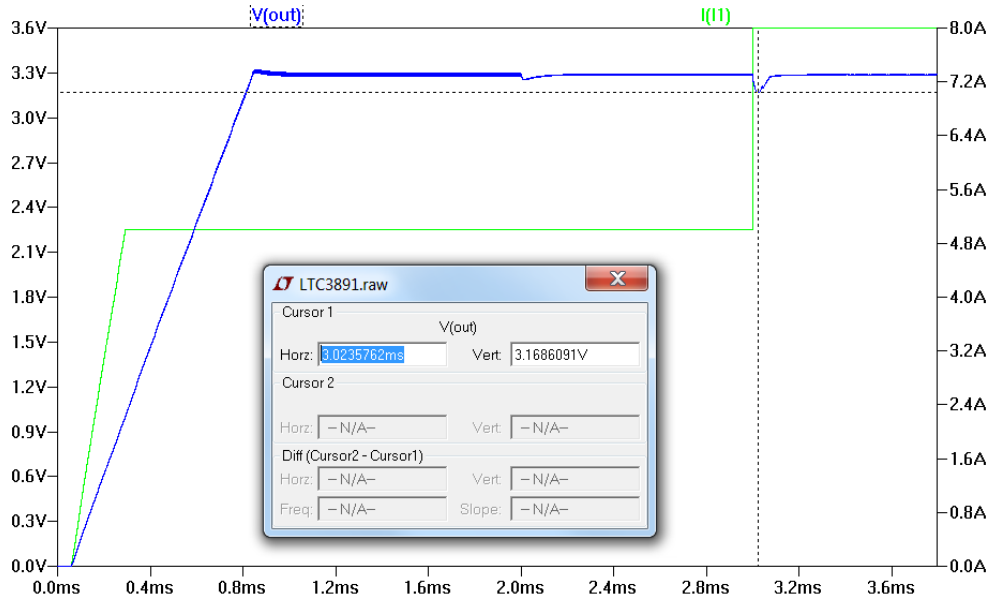


Figure 150 - Load test simulation in LTSpice

$$3.3 \text{ V} \cdot 0.05 = 0.165 \text{ V} \quad (100)$$

$$3.3 \text{ V} - 0.165 \text{ V} = 3.135 \text{ V} \quad (101)$$

$$3.16 \text{ V} > 3.135 \text{ V} \quad (102)$$

By looking at Equation (100), (101) and (102) we can see that the power supply is within 5% regulation for an 8 A load.



#### 6.4.2.4. Temperature

The efficiency of the controller determines how much power it dissipates, and thus how much temperature we can expect from the IC. From the datasheet we can see that the efficiency is listed to be approximately 91% for an input voltage of 12 V.

The estimated input power can be calculated, because we know the output power;  
 $3.3\text{ V} \cdot 5\text{ A} = 16.5\text{ W}$ .

$$\frac{16.5\text{ W}}{0.91\%} = 18.1\text{ W} \quad (103)$$

We can see from (103) that the input power is approximately 18.1 W, the power dissipated in the circuit must be;

$$P_D = P_{out} - P_{in} = 18.1\text{ W} - 16.5\text{ W} = 1.6\text{ W} \quad (104)$$

The equation to calculate the junction temperature of the IC is seen in (105).

$$T_J = T_A + (P_D \cdot \theta_{JA}) \quad (105)$$

Where  $\theta_{JA} = 38\text{ }^{\circ}\text{C/W}$ , and  $T_A = \text{ambient temperature}$ .

$$T_J = 30\text{ }^{\circ}\text{C} + \left(1.6\text{ W} \cdot 38\frac{^{\circ}\text{C}}{\text{W}}\right) = 91\text{ }^{\circ}\text{C} \quad (106)$$

The junction temperature of the LTC3891 can reach temperatures as high as 91 °C when running a 5 A load in an ambient temperature of 30 °C.



### 6.4.3. 5V Supply

The LTM4609<sup>106</sup> as seen in Figure 151 is a buck-boost converter, contrary to the 3.3 V LTC3891, this controller has its MOSFET drivers implemented inside the chip. The LTM4609 can handle a wide  $V_{IN}$  range of 6 - 18 V, and the output voltage  $V_{OUT}$  can be programmed in the range of 0.8 - 34 V, which makes this controller highly versatile to implement. The buck-boost topology enables the controller to regulate a fixed DC voltage on the  $V_{OUT}$  pin, while the voltage on the  $V_{IN}$  pin drops below the regulated  $V_{OUT}$  pin. Pin description is given in Table 37.

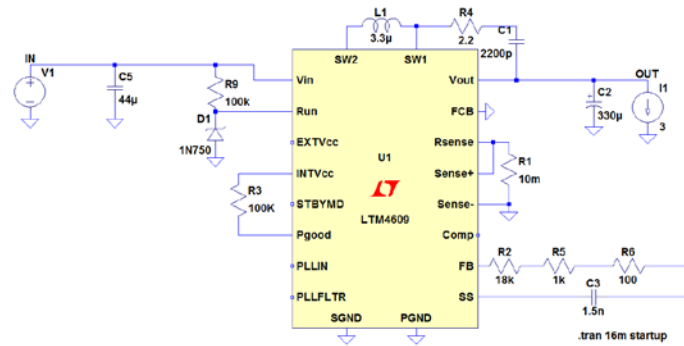


Figure 151 - LTM4609 5V supply

Table 37 - LTM4609 pin description

Name	NO.	Description
VIN	Bank 1	Power input pin
VOUT	Bank 5	Power output pin
PGND	Bank 6	Power ground for both input and output
SW1	Bank 4	Switch nodes. Power inductor is connected between SW1 and SW2
SW2	Bank 2	Switch nodes. Power inductor is connected between SW1 and SW2
RSENSE	Bank 3	Sensor resistor pin
RSENSE+	A4	Positive input to the current sense and reverse current detect comparators
RSENSE-	A5	Negative input to the current sense and reverse current detect comparators
EXTVCC	F6	External VCC input
INTVCC	F5	Internal 6 V regulator output
PLINN	B9	External clock synchronization
PLLFLTR	B8	Lowpass filter of the phase-locked loop is tied to this pin
SS	A6	Soft-start pin
STBYMD	A10	LDO control pin
VFB	B6	Negative input of the error amplifier
FCB	A9	Forced continuous control input
SGND	A7	Signal ground pin
COMP	B7	Current control threshold and error amplifier compensation point
PGOOD	B5	Open drain logic output
RUN	A8	Run control pin

<sup>106</sup> LTM4609, Positive 5V supply



#### 6.4.3.1. *LTM4609 design decisions*

Since the LTM4609 has its switching MOSFETS inside the chip, it has very few external components. The chip itself is only 15 mm x 15 mm x 2.62 mm; the result is that the 5 V regulator is a very space effective solution.

The regulator is supplied on the  $V_{IN}$  pin, in the range of 6 - 18 V; a decoupling capacitor of 44  $\mu$ F (see 6.4.7 for capacitor calculations) is placed on the supply pin to reduce ripple feed into the regulator. The RUN pin can be connected to an external controller, which can activate the 5 V power supply. For the controller inside the power supply to be active, the voltage supplied to the RUN pin must be in the range of 1.6 - 6 V, whenever the voltage drops below 1.6 V the controller will shut down.

The INTVcc pin is power from an internal low dropout regulator (LDO), which generates 6 V from the  $V_{IN}$  pin. The internal LDO supplies both the INTVcc pin, the controller and internal circuitry. As known, LDOs produce a stable supply voltage with low ripple at the cost of continuous power dissipation; this may reduce the efficiency of the power supply by a fraction.

An alternative is to either supply the LTM4609 from its own  $V_{OUT}$  if the voltage is programmed in the range of  $5.7\text{ V} < V_{OUT} < 7\text{ V}$  which is the most efficient solutions.

If an external supply is available in the 5.5 - 7 V range, it is possible to use this as an external supply on the EXTVcc pin, given that the power is compatible with the internal MOSFETs.

If the EXTVcc is left floating the controller will automatically use the internal LDO as supply. Whenever a voltage rise above 5.7 V on the EXTVcc pin, the internal LDO will be switched off, and the EXTVcc thereby supplying internal power.

Several control options are available for the STBYMD pin, which can be retrieved from the datasheet. When the STBYMD pin is left floating or decoupled to ground using a capacitor, the LTM4609 allows for external power on/off control.

For visual verification, a diode can be connected to the PGOOD pin to evaluate whether or not the voltage regulation is within 7% regulation. Whenever  $V_{OUT}$  is within 7% of the desired value, the PGOOD pin will internally be connected to ground.

Both the PLLIN and the PLLFLTR pin are used to determine the switching frequency of the internal MOSFETs, the LTM4609 can be synchronized with other switching devices, by adding an external clock to the PLLIN pin, the top MOSFET



in the controller will trigger on the rising edge of the clock signal applied. The second alternative is to configure the desired operating frequency, by the use of the internal clock. The datasheet has specified a voltage range between 0.5 - 2 V on the PLLFLTR pin that enables the designer to choose from 200 - 400 kHz of switch frequency.

If the PLLFLTR is left floating, the switch frequency will be approximately 180 kHz.

The inductor between SW1 and SW2 are of importance to reduce the switching ripple generated by the MOSFETs, the inductor can be selected by solving (107) and (108).

$$L_{BOOST} \geq \frac{V_{in}^2 (V_{OUT(max)} - V_{in})}{V_{OUT(max)}^2 \cdot f \cdot I_{OUT(max)} \cdot Ripple\%} \quad (107)$$

$$L_{BUCK} = \frac{V_{OUT}(V_{IN(max)} - V_{OUT})}{V_{IN(max)} \cdot f \cdot I_{OUT(max)} \cdot Ripple\%} \quad (108)$$

We have done the calculation, resulting in inductors within the range of 10 - 21  $\mu\text{H}$ . Simulation of circuit with the given inductor values, results in poor regulation of the output voltage. A decision was made to choose the recommended inductor value provided in the datasheet of 3.3  $\mu\text{H}$ , which provided a beneficial regulation of the output voltage. In addition, a RC circuit can be added between the SW1 pin and  $V_{OUT}$  in order to reduce the switching noise.

The nominal  $V_{IN} = 13 \text{ V}$  and nominal  $V_{OUT} = 5 \text{ V}$ , cause the foundation for the decoupling calculation, as the regulator normally will operate in buck mode rather than boost mode. When the  $V_{IN}$  voltage reaches approximately 4 - 6 V the regulator will operate in buck/boost mode which is less effective, cause more switching and results in higher voltage ripple. Unfortunately there is no equation to derive a decoupling capacitor for this mode, so we will use the buck mode Equation (109) and keep in mind the poor regulation in buck/boost mode.

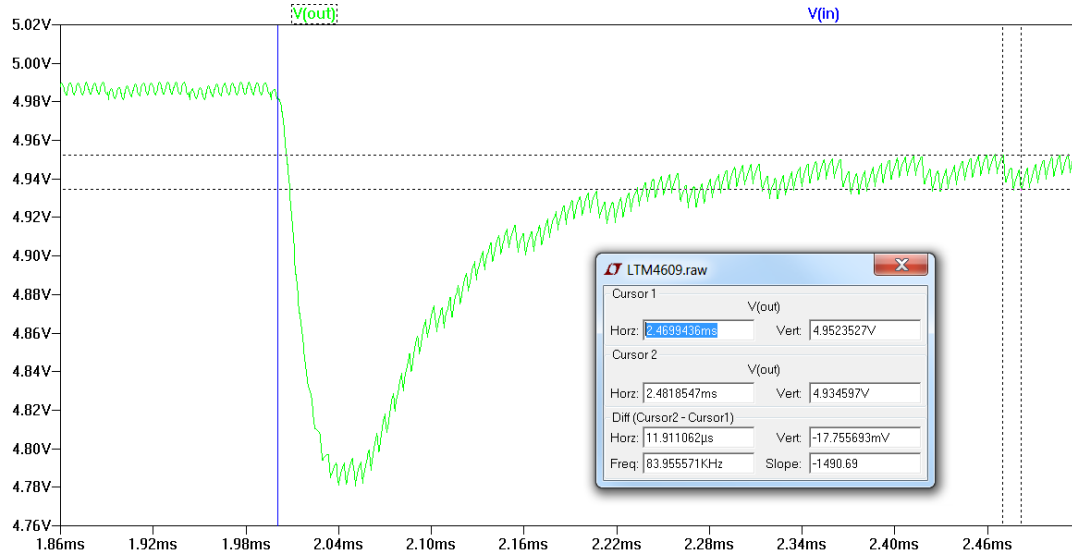
$$V_{RIPPLE,BUCK} = \frac{V_{OUT}(V_{IN(max)} - V_{OUT})}{8 \cdot L \cdot C_{OUT} \cdot V_{IN(max)} \cdot f^2} \quad (109)$$

$$V_{RIPPLE,BUCK} = \frac{5 \text{ V} \cdot (18 - 5) \text{ V}}{8 \cdot 3.3 \mu\text{H} \cdot 470 \mu\text{F} \cdot 18 \text{ V} \cdot 180^2 \text{ kHz}} = 8.98 \text{ mV} \quad (110)$$





With an output bulk capacitor of 470  $\mu\text{F}$  and an input capacitor of 44  $\mu\text{F}$  (see 6.4.7) we can see from (110) that it will produce a ripple of less than 10 mV during buck operation, as we discussed earlier we do expect the ripple to increase slightly when operated in buck/boost mode.



**Figure 152 - Ripple comparison of buck and buck/boost mode**

The simulation in Figure 152 confirms our suspicions, the blue line is the  $V_{IN}$  voltage, representing the voltage dropping from 12 V to 4 V, which means that the regulator goes from buck to buck/boost mode. As the regulator operates in buck/boost mode we can measure the ripple, from the cursor window we can see that the new ripple is 17.75 mV. We wish to maintain regulation within 5% of the 5 V output voltage, with the resulting operation range of 4.75 V to 5.25 V. The decoupling capacitor of 470  $\mu\text{F}$  was selected after running simulations with different kind of capacitors.

For light load operation, the LTM4609 provide features to improve efficiency, there are three modes to choose from, which can be selected by applying a voltage to the FCB pin. Information about the three modes can be found in the datasheet. When the FCB pin is below 0.75 V the controller will operate as a continuous, PWM current mode synchronous switching regulator.

The maximum and minimum values for the 5 V power supply is listed in Table 38.

**Table 38 - LTM4609 Max/Min**

	Min.	Typ.	Max.	Unit
<b>Vin</b>	4	12	18	V
<b>Vout</b>	4.89	5	5.43	V
<b>Iout</b>	0	2	8	A



#### 6.4.3.2. Crank test simulation

The crank test simulation is executed to evaluate how the power supply will respond during cold cranking. The blue line represents the battery voltage which suddenly drops when the starter motor is applied, and drops from 12 V to 6 V and stays there until the engine fires and start to generate its own power. The green line represents the output voltage of the 5 V power supply, regulation is within 5% even when the voltage drops rapidly, which proves that the power supply will maintain 5 V during cold cranking.

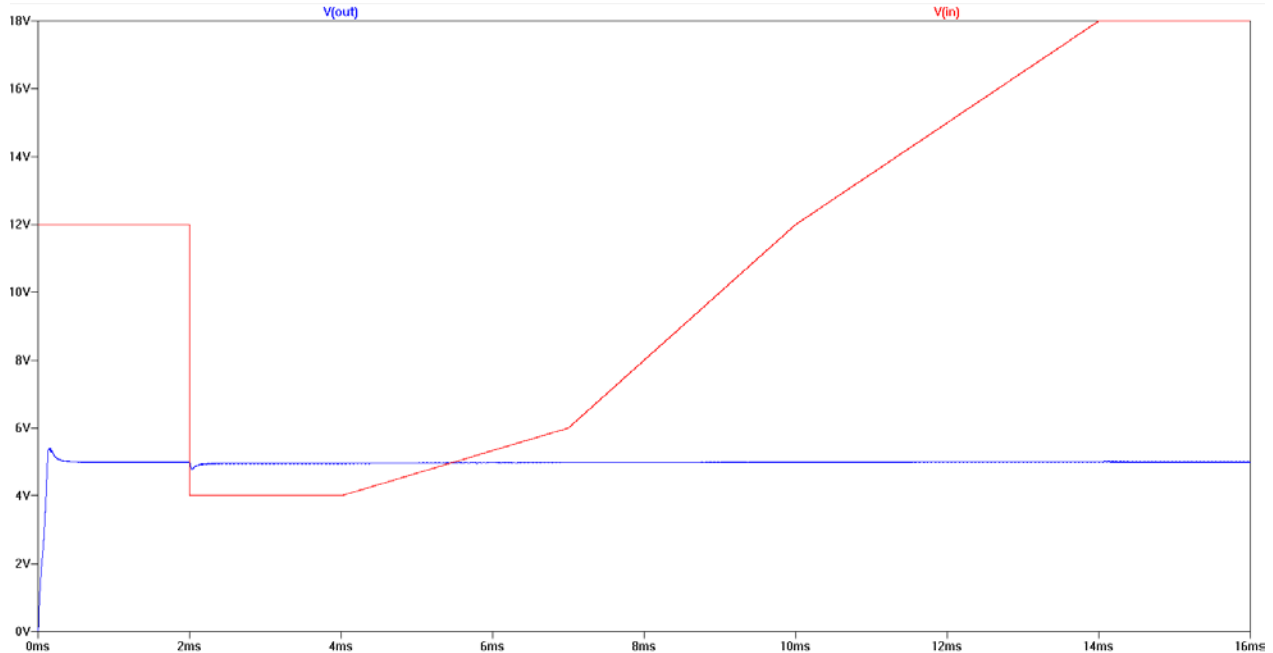


Figure 153 - LTM4609 crank test simulation

#### 6.4.3.3. Load test simulation

To see how the 5 V power supply responds to different load levels, we can simulate the power supply in LTSpice to evaluate its results. The red line represents the load current on the output pin of the supply, the loading is increased in 2 ms steps from 0 to 8 A, where the green line is the output voltage. The LTM4609 starts with 0 loading and quickly reach regulation to the desired value of 5 V, after regulation has stabilized, a 4 A load is applied, the output voltage falls down to 4.89 V as seen in the cursor box, the controller regains control and regulate the voltage back to 5 V. The voltage drop is well within 5% regulation, and so is the regulation through as the loading is further increased up to 8 A.

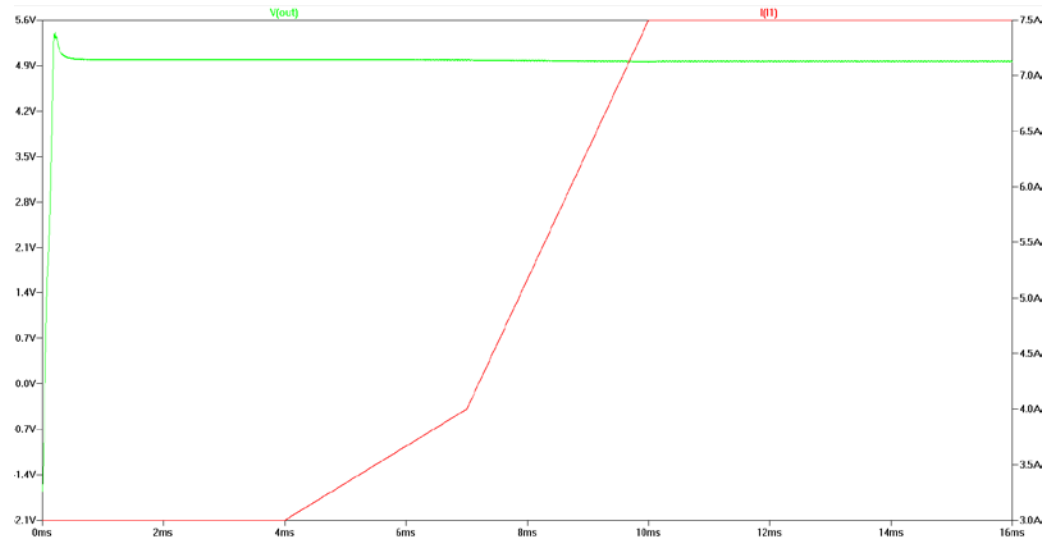


Figure 154 - LTM4609 load test simulation

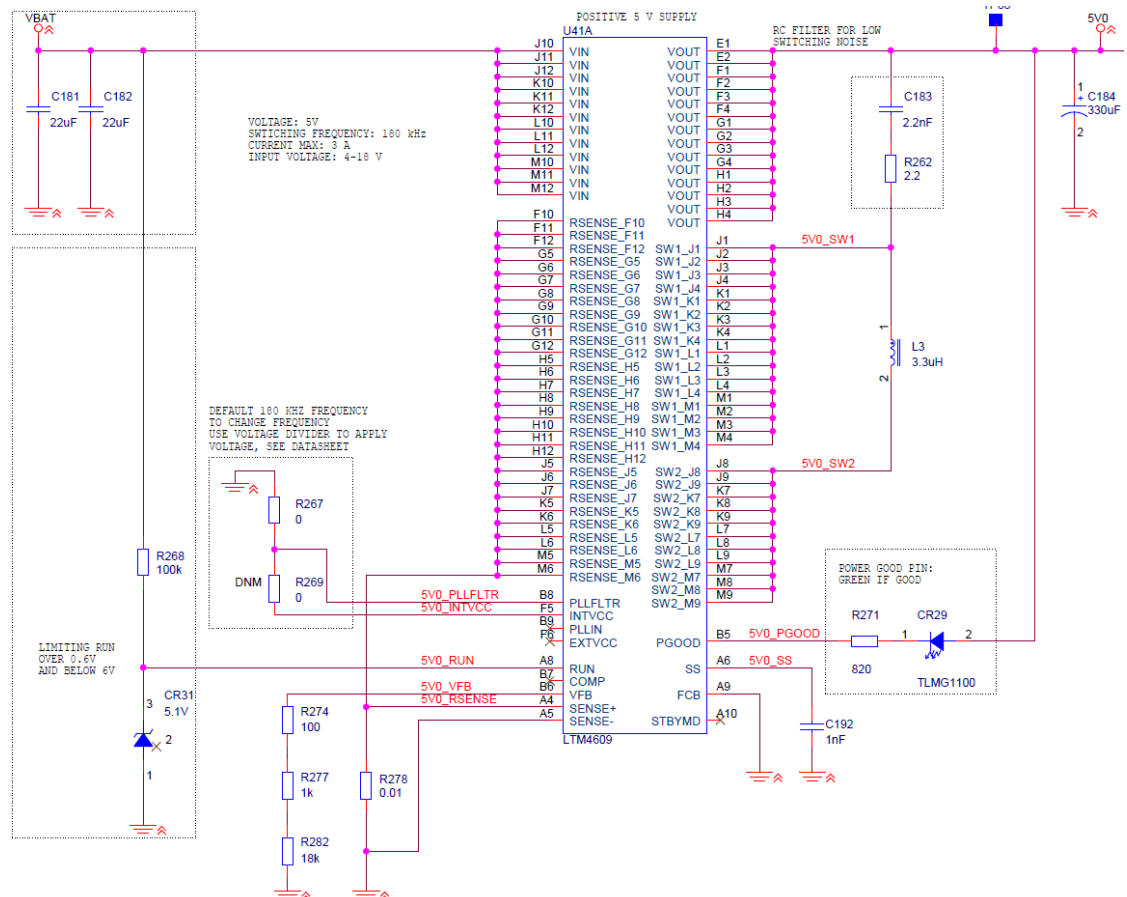
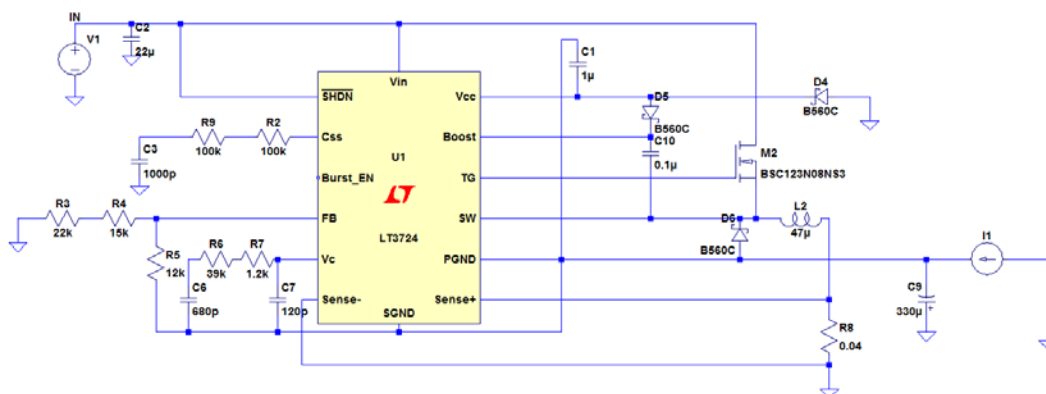


Figure 155 - 5 V power supply schematic implementation



#### 6.4.4. -5 V supply

The LT3724<sup>107</sup> as seen in Figure 156 is implemented in a TSSOP package, it's an inverting buck/boost controller that has a wide 6 - 18 V input voltage and delivers -5 V at the output pin. The controller is designed for low to medium power applications, making the controller suitable for our use. Compared to other topologies, this converter is interesting because it only use one MOSFET where the other is replaced with a freewheeling diode. The negative voltage is regulated with PWM switching of the MOSFET, the controller has a sense resistor in the path, which is used to compare the inline voltage drop with an internal reference voltage, thus the controller can respond to loading currents to maintain steady -5 V. A pin description is given in Table 39.



**Figure 156 - LT3724 negative 5 V supply**

### Table 39 - LT3724 pin discription

Name	NO.	Description
VIN	1	Power supply pin
NC	2	No connect
nSHDN	3	Shutdown pin, active low.
CSS	4	Soft-start pin
BURST_EN	5	Enable or disable Burst Mode operation
VFB	6	Output voltage feedback pin
VC	7	Output of the error amplifier
SGND	8	SGND pin is the low noise ground reference
SENSE-	9	Negative input for current sense amplifier
SENSE+	10	Positive input for current sense amplifier
PGND	11	High-current ground reference for internal low side switch and VCC regulator circuit
VCC	12	Internal bias supply
NC	13	No connect
SW	14	Switch pin
TG	15	Bootstrapped gate drive for N-channel MOSFET
BOOST	16	Supply for the bootstrapped gate drive
Exposed pad	17	Exposed leadframe, internally connected to SGND pin

<sup>107</sup> LT3725, negative 5 V supply



#### 6.4.4.1. **LT3724 design decisions**

The LT3724 can be configured to operate both as step up and step down controller, positive SEPIC controller or inverter which is our use of the controller.

The controller is supplied through the wide 6 - 18 V input range on the  $V_{IN}$  pin; the controller is decoupled with 22  $\mu$ F low ESR input capacitor (see 6.4.7 for calculations).

In order to protect the power supply, the controller features an undervoltage protection. The inverted SHDN can be programmed into a preferred threshold voltage, in which the source must exceed, before the controller is enabled. The undervoltage lockout ensures that the power supply is power down during undervoltage, and can be programmed with use of a voltage divider as illustrated in the datasheet. As the power is to be implemented in automotive applications, it is expected a certain undervoltage operation during cold start cranking. The inverted SHDN pin can then be configured to not shutdown the controller, and thus operate down to 4 V, it is accomplished by a pull up resistor to  $V_{in}$ , the SHDN pin current cannot exceed 1 mA above 5 V, therefore a 1 M $\Omega$  resistor is recommended.

In order to decrease the initial surge current, when all electronics is powered up, the LT3724 features a slow start configuration on the  $C_{SS}$  pin. The time delay can be chosen by calculating (111).

$$C_{SS} = \frac{2 \mu A \cdot t_{SS}}{V_{OUT}} \quad (111)$$

If we choose the soft start time to be 2.5 ms, we calculated the capacitor value in (112) to be;

$$C_{SS} = \frac{2 \mu A \cdot 2.5 ms}{5 V} = 1 nF \quad (112)$$

In series with the recommended 200 k $\Omega$  resistor, the controller will regulate the output voltage to 95% of its pre-set voltage in 2.5 ms. During low current loading, the regulator features a BURST\_EN pin, which can be configured to allow the controller to be more efficient. This is achieved through reduced switching, but also increases the ripple, the BURST\_EN pin is connected to ground, to disable the feature, more information can be found in the datasheet.



In order to set the output voltage of the regulator, the controller uses a closed loop feedback, which links the output voltage to the FB pin using a voltage divider. The feedback voltage is compared to the internal 1.231 V reference, whenever the error amplifier sense a difference, the threshold current is compensated which cause the regulator to alter the switch frequency in order to reduce the error between the feedback pin and the internal reference. The output voltage can be programmed by choosing the resistors in (26).

$$R_2 = R_1 \cdot \left( \frac{V_{OUT}}{1.231 V} - 1 \right) \quad (113)$$

$$R_2 = 12 k\Omega \cdot \left( \frac{5 V}{1.231 V} - 1 \right) = 36740 \Omega \quad (114)$$

If we choose the 12k and 37k resistors, we can evaluate the output voltage in (115).

$$-1.231 V \cdot \left( \frac{37 k\Omega}{12 k\Omega} + 1 \right) = -5.02 V \quad (115)$$

The given feedback resistors will set the controller output voltage to -5 V like we designed it to.

The switching current that operates the MOSFET can be monitored directly on the  $V_C$  pin, the voltage represents the switching current and is the result of both the internal error amplifier and the soft start circuit. Transient response of the regulator can be designed by altering the filter capacitors and resistor, the circuit represents the dominant pole for the controller regulation, and thus stability and response can be designed. The datasheet does not specify the foundation for the calculation, so knowledge about regulation will be necessary to configure another pole location for the filter. The capacitors  $C_{C1}$  and  $C_{C2}$  with values 680 pF and 120 pF and the resistor  $R_C = 40.2 k\Omega$  is retrieved from the application note in the datasheet.

Measurement of the inductor current is used in the regulation to detect loading currents on the output pin; the current through the sense resistor is proportional to the current in the inductor.  $R_{SENSE}$  can be calculated using (116).



$$R_{sense} = \frac{100 \text{ mV}}{I_{OUT(Max)}} \quad (116)$$

The maximum output current is 2.5 A, and the sense resistor is given in (117).

$$R_{sense} = \frac{100 \text{ mV}}{2.5 \text{ A}} = 0.04 \Omega \quad (117)$$

Both the signal ground SGND and the power ground PGND pins are connected to the output, because the controller operates as an inverter.

During start up and or cold start cranking, the MOSFET switch on for longer periods than normal regulation, as the MOSFET charges the inductor, the freewheeling Schottky diode enables discharge of the inductor by opening during the MOSFET off time. During heavy switching, the SW pin helps the Schottky diode to keep virtual ground to allow charging of the boost inductor.

The TG pin is the gate switching pin of the controller, it usually switches high currents at very high rates, and it's recommended that the PCB trace for this pin is minimum 0.5 mm.

The regulator features a boost function, which utilize a boost capacitor. When the MOSFET is turned off, an internal BJT which the collector is connected to the SW PIN, pull the SW pin low in order to charge the boost capacitor from the  $V_{CC}$  pin which also supplies most of the internal IC functions. When the MOSFET switches on, the MOSFET is biased from the boost capacitor.

The maximum and minimum values for the -5 V power supply is listed in Table 40.

**Table 40 - Maximum and minimum rating for the LT3724**

	Min.	Typ.	Max.	
<b>Vin</b>	4	12	18	V
<b>Vout</b>	-5.26	-5	4.82	V
<b>Iout</b>		400	1000	mA



#### 6.4.4.2. Crank test simulation

The LT3724 is applied to a cold start crank test seen in Figure 157, the battery voltage is 12 V before it drops to 4 V for a period, before the engine starts and slowly starts to generate power. The green line represents the output voltage, as we can see by the cursor box the voltage slightly dips outside 5% regulation when  $V_{IN}$  drops drastically, but soon regain regulation when  $V_{IN}$  is more realistic. The blue line indicates the input voltage and the green line indicating the output voltage. As can be seen on Figure 157 the converter maintains voltage regulating during and after crank.

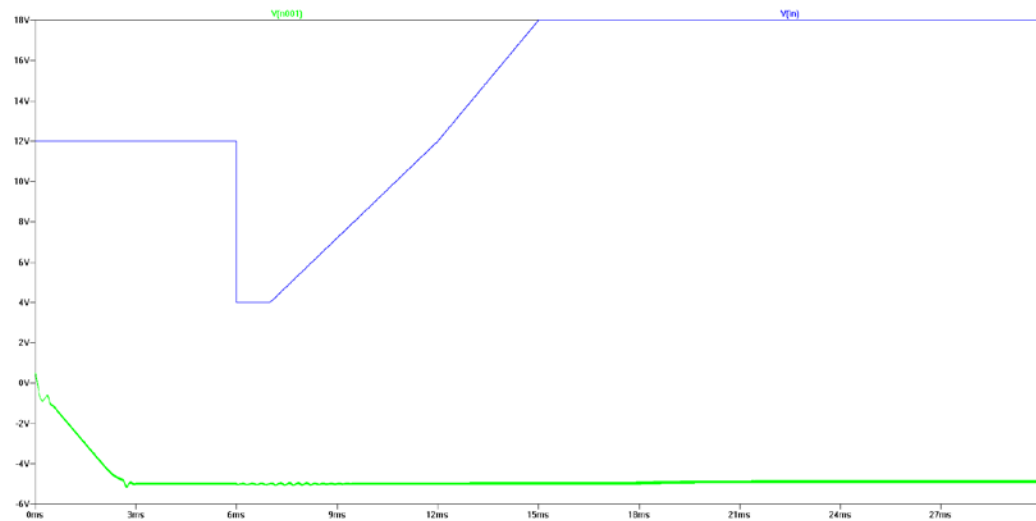


Figure 157 - Crank test simulation

#### 6.4.4.3. Load test simulation

The load test simulation as seen in Figure 158, exerts the LT3724 to an suddenly applied load, to evaluate its results. The 400 mA load is applied at 4 ms after the controller has stabilized at -5 V, the sudden loading of the output pin cause a slight overshoot before it reach 5% regulation with good margin. The controller is then exerted to an increasing load up to 1 A, well within the 400 mA maximum current rating.



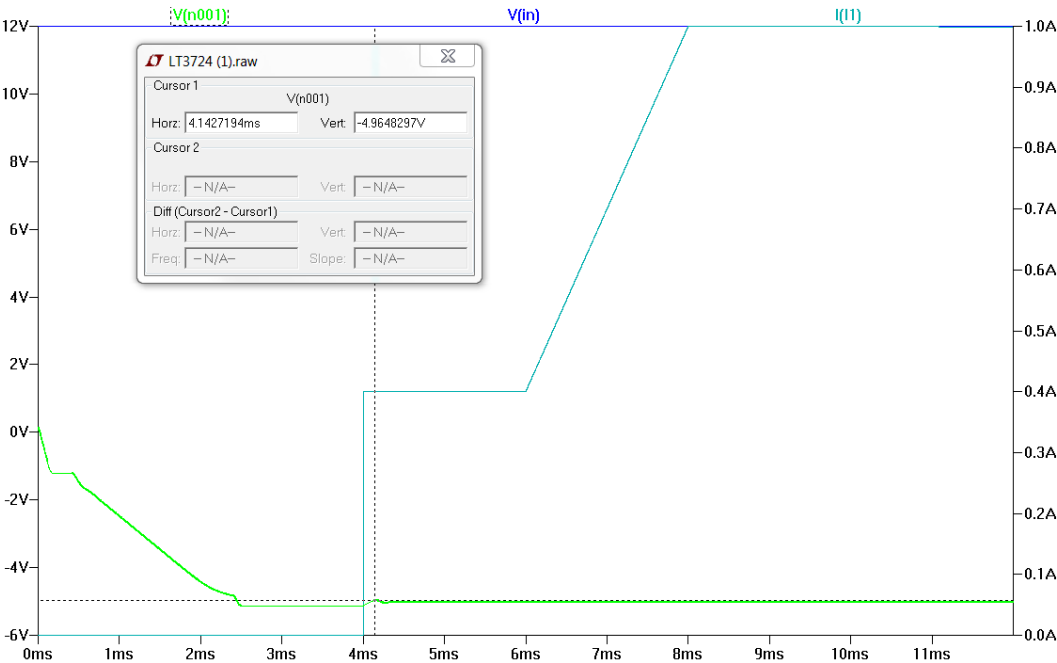


Figure 158 - Load test simulation

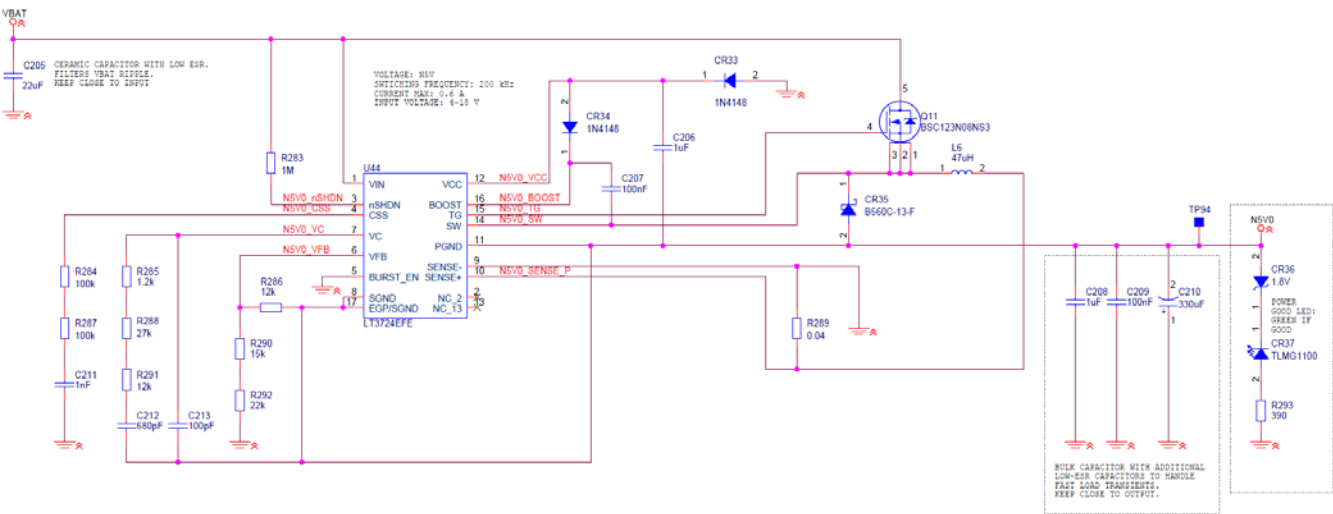


Figure 159 - -5 V power supply schematic implementation



#### 6.4.4.4. Temperature

The thermal behaviour of the switching MOSFET can be critical; electronics are affected by high temperatures, as their efficiency often drops and their behaviour changes. Thermal prediction for the MOSFET is useful, because that is the device that dissipates most of the power in the power supply circuit. Power dissipation is analogue with temperature, so let's calculate the conduction losses of the MOSFET in (119) with use of (118).

$$P_{COND} = (I_{OUT(MAX)})^2 \left( \frac{V_{OUT}}{V_{IN}} \right) R_{DS(ON)} \quad (118)$$

$$P_{COND} = (1 \text{ A})^2 \cdot \left| \frac{-5 \text{ V}}{4 \text{ V}} \right| \cdot 0.0123 \Omega = 15.3 \text{ mW} \quad (119)$$

To calculate the maximum transition losses, (120) and (121) is used.

$$P_{TRAN} = K \cdot V_{in}^2 \cdot I_{OUT(MAX)} \cdot C_{RSS} \cdot f_{SW} \quad (120)$$

$$P_{TRAN} = 2 \cdot (12 \text{ V})^2 \cdot 1 \text{ A} \cdot 15 \text{ pF} \cdot 200 \text{ kHz} = 864 \mu\text{W} \quad (121)$$

The total power dissipated in the MOSFET is the sum of both  $P_{COND}$  and  $P_{TRAN}$  is given in (122) and (123).

$$P_{FET(TOT)} = P_{COND} + P_{TRAN} \quad (122)$$

$$P_{FET(TOT)} = 15.3 \text{ mW} + 864 \mu\text{W} = 16.1 \text{ mW} \quad (123)$$

The power dissipation should be less than 3% of the total output power for the regulator to be efficient. With the use of (124) we can calculate the efficiency.

$$\eta = \frac{P_{FET(TOT)}}{P_{OUT}} = \frac{16.1 \text{ mW}}{5 \text{ W}} = 0.32\% \quad (124)$$



Now that we have found the power dissipation in the MOSFET and verified that the efficiency is well within 3%, we can use (125) to find the junction temperature of the MOSFET.

$$T_J = T_A + P_{FET(TOT)} \cdot \theta_{JA} \quad (125)$$

Where  $\theta_{JA}$  the thermal resistance of the MOSFET is package and  $T_A$  is the ambient temperature of the MOSFET.  $\theta_{JA}$  is found in the Infineon BSC123N08NS3 datasheet to be  $18 \frac{K}{W}$ , and the ambient temperature is set to  $40^\circ C$ .

$$T_J = 40^\circ C + 16.1 mA \cdot 18 \frac{K}{W} = 40.3^\circ C \quad (126)$$

The junction temperature of the MOSFET is found to be  $40.3^\circ C$  in (126). The power dissipated in the MOSFET is small, which lowers the temperature.



#### 6.4.5. 15V supply

The LT3580<sup>108</sup> is a boost/inverting DC/DC converter, we are using the LT3580 in a boost topology, transforming 5 V to 15 V, to supply components on the motherboard. The 15 voltage supply configuration can be seen in Figure 160. This DCDC converter is an integrated converter, meaning that the switching devices are internally inside the component, as a consequence of this, not many external components are needed to properly regulate 15 V. A pin description is given in Table 41.

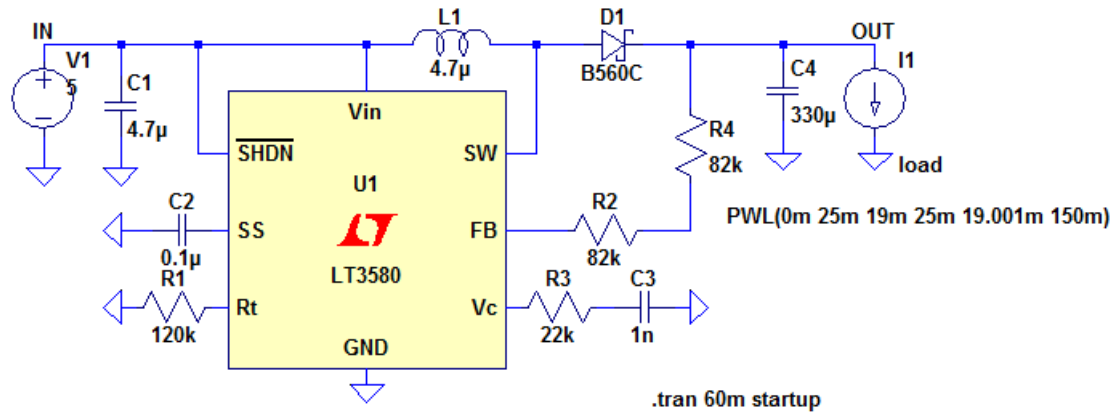


Figure 160 - 15 V configurations

Table 41 - LT3580 pin description

Name	NO.	Description
FB	1	Positive and negative feedback
VC	2	Error amplifier output pin
VIN	3	Input supply
SW	4	Switch pin
nSHDN	5	Shutdown pin, active low
RT	6	Timing resistor pin. Adjusts the switching frequency
SS	7	Soft-start pin
SYNC	8	Synchronize the switching frequency to an outside clock
GND	9	Ground

<sup>108</sup> LT3580, Boost converter



#### 6.4.5.1. LT3580 design decisions

The input is bypassed with a 4.7  $\mu\text{F}$  capacitor with low ESR (see 6.4.7 for calculations) and the output is bypassed with a 330  $\mu\text{F}$  capacitor to filter the ripple, the capacitors was chosen after running simulations and recommended values from the datasheet. The shutdown pin is active low, meaning that when a voltage on the input crosses the enable voltage (typically 1.2 V) the converter is then activated and regulation can start. This pin is connected directly to the input voltage. The 15 V power supply is connected to the 5 V supply, this reduces the complexity of the power supply, due to the elimination of the cranking problem.

The switching frequency is chosen to be at approximately 700 kHz with the resistor of 120 k $\Omega$  connected between  $R_T$  and ground. The LT3580 uses a single resistor to provide feedback to the feedback pin to enable exact regulation of 15 V on the output. In Figure 160, it can be seen that two resistors are used; this is to approximate to the nearest E12 resistor values. The real value of this resistor should be 164 k $\Omega$ ; the error introduced by changing to E12 is 0%. To change the output voltage, a new resistor  $R_{FB}$  needs to be calculated using (127).

The compensation network that is used is to smooth out the transients on the output voltage. The compensation network is built up from capacitors and resistors and is connected to the VC pin, which is an error amplifier output pin. The compensation network was first set to the recommended values in the datasheet, and then changed, using simulations, to give sufficient transient response on the output.

The SW pin is the switch pin, and is connected to the collector of the internal NPN power switch.

The inductor that is used was calculated by using (128) for duty cycle and (129) for inductor value. Where  $V_D$  is the forward voltage drop across the diode and  $V_{CESAT}$  is typically 300 mV at 1.5 A. Where  $\eta$  is the efficiency of the converter, which is approximately 88% for boost applications.

$$R_{FB} = \frac{V_{OUT} - 1.215 \text{ V}}{83.3 \mu\text{A}} \quad (127)$$

$$DC \cong \frac{V_{OUT} - V_{IN} + V_D}{V_{OUT} + V_D - V_{CESAT}} \quad (128)$$

$$L > \frac{DC \cdot V_{IN}}{2(f_{SW}) \left( I_{LIM} - \frac{V_{OUT} \cdot I_{OUT}}{V_{IN} \cdot \eta} \right)} \quad (129)$$



We calculated that we needed an inductor that is bigger than 2.9  $\mu\text{H}$ ; we chose to use a 4.7  $\mu\text{H}$ , due to the availability of this component.

The diode that is used is a B560C from (Diodes Incorporation<sup>109</sup>) is a diode with a breakdown voltage at 60 V and can handle currents up to 5 A. We choose to use this because this is the diode that we are using for the other DCDC converters and to not add to the BOM, we chose to use the same.

The LT3580 is configured to properly work with a wide input range from 4 - 14 V, and has an output of 15 V at 150 mA constant load. The ripple at the output voltage is 422  $\mu\text{V}$ .

The SS pin stands for soft-start and is connected by an external capacitor between the SS pin and ground. The capacitor is slowly charged up to 2.1 V by an internal resistor. While the capacitor is charged up to 2.1 V the current limit also increases. The soft-start is set to approximate 35 ms. The soft-start limits the limit peak switch current during start-up, thus eliminating the high start-up current that is inherent in switching regulators. To determine the soft-start time, use (130).

$$T_{SS} = \frac{2.1 \text{ V} \cdot C_{SS}}{6 \mu\text{A}} \quad (130)$$

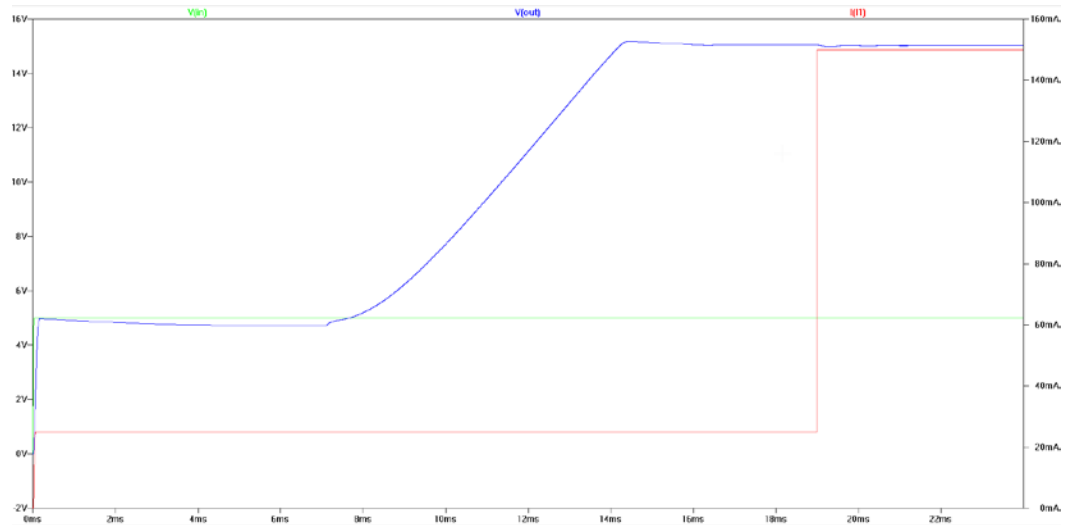
#### 6.4.5.2. Load test simulation

To test this power supplies we are running test on, using LTSpice. The test that is done is the crank test and load test. During the crank test we are running the power supply at an input voltage at 12 V for a while and then suddenly dropping the input to 4 V. We want to see if the power supply can handle the sudden change in input voltage. The 15 V power supply is connected to the 5 V supply, thus this supply does not need to be run with a crank test. A load test has been simulated. The load test consist of running the supply at nominal load (which is 25 mA), then the load will increase. The purpose with this test is to see if the power supply can handle more loads on the output and still continue regulating as intended.

The load test can be seen in Figure 161. The load starts at 25 mA, we let the supply stabilize to 15 V before letting the load increase to 150 mA. This is illustrated in Figure 162.

---

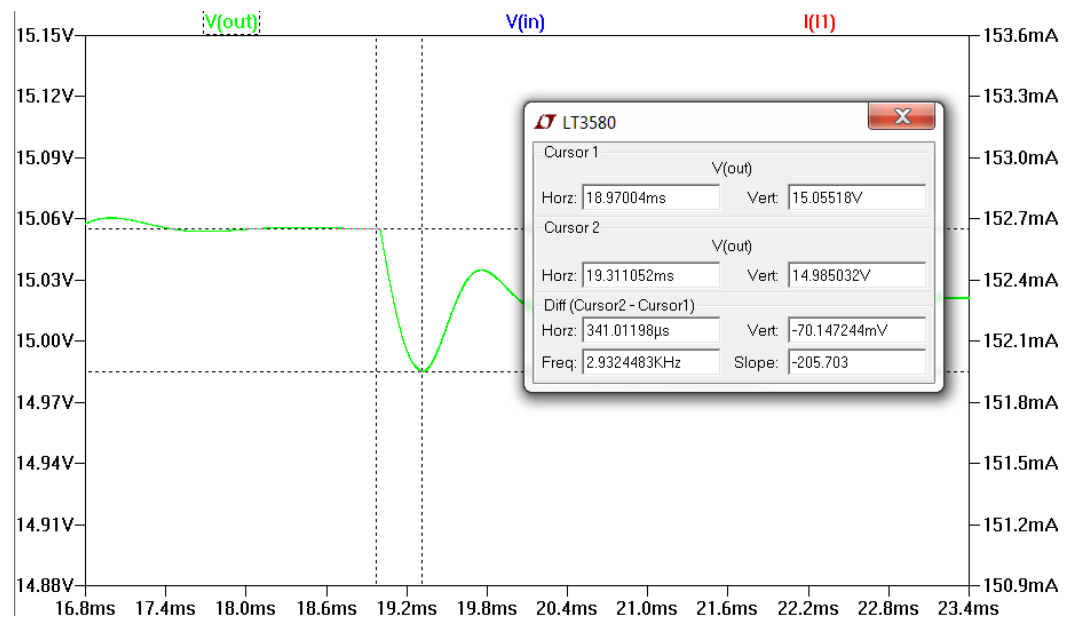
<sup>109</sup> B560C Datasheet



**Figure 161 - Load test on 15 V power supply**

The voltage drops from 15 V to 14.98 V and returns back to 15 V in 861  $\mu$ s.

From the derating tables, we are allowed to have 5% error in the regulation; this means that the output voltage can drop to a minimum value of 14.25 V and a maximum value of 15.75 V. With this we can conclude that we have a 15 V power supply that will meet our requirements.



**Figure 162 - Regulation to sudden load changes**



#### 6.4.5.3. Temperature

To calculate the heat dissipated in the converter, equation (131),(132),(133),(134) and (135) can be used.

$$\text{Average Input Current} = I_{IN} = \frac{V_{OUT} \cdot I_{OUT}}{V_{IN} \cdot \eta} \quad (131)$$

$$\text{Switch } I^2R \text{ Loss: } P_{SW} = D_C \cdot I_{IN}^2 \cdot R_{SW} \quad (132)$$

$$\text{Base Drive Loss (AC): } P_{BAC} = 13 \text{ ns} \cdot I_{IN} \cdot V_{OUT} \cdot f \quad (133)$$

$$\text{Base Drive Loss (DC): } P_{BDC} = \frac{V_{IN} \cdot I_{IN} \cdot D_C}{40} \quad (134)$$

$$\text{Input Power Loss: } P_{INP} = 7 \text{ mA} \cdot V_{IN} \quad (135)$$

Where,  $R_{SW}$  is the switch resistance (typically 200 mΩ at 1.5 A),  $D_C$  is the duty cycle, see (128) and  $\eta$  is the efficiency (typically 88%). The total power dissipated is then given by (136).

$$P_{TOT} = P_{SW} + P_{BAC} + P_{BDC} + P_{INP} \quad (136)$$

The total power dissipated in the LT8580 is then calculated to be 0.185 W.

To calculate the junction temperature, which is the highest operating temperature of the LT8580, equation (137) can be used.

$$T_J = T_A + \theta_{JA} \cdot P_{TOT} \quad (137)$$

Where  $T_J$  is the junction temperature,  $T_A$  is the ambient temperature and  $\theta_{JA}$  is the thermal resistance from the silicon junction to the ambient air.

The junction temperature is then calculated to be 37 °C.



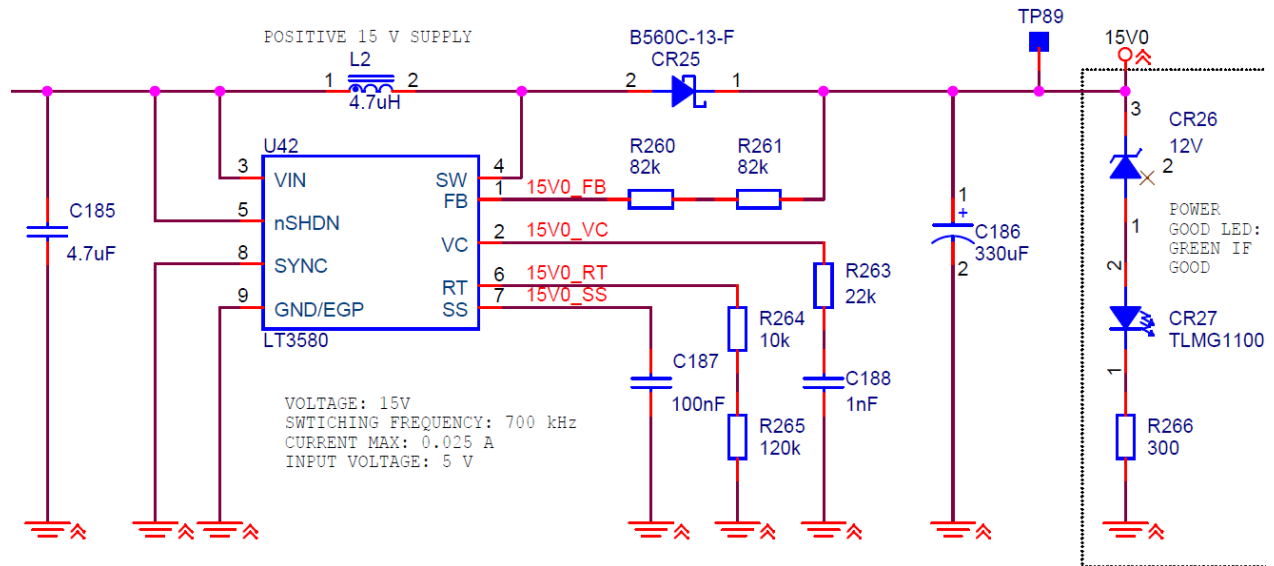


Figure 163 - 15 V power supply schematic implementation



#### 6.4.6. -15 V Supply

The LT8580<sup>110</sup> is a boost/sepic/inverting DCDC converter. We are using the LT8580 with the dual inductor inverting topology. This is in principle the same as the sepic topology by with an inductor in series with the output and the diode connected between the output and ground. The -15 V is also connected to the 5 V supply. The 5 V input supply is transformed to give an output voltage of -15 V. The configurations for the -15 V power supply is seen in Figure 164. A pin description is given in Table 42.

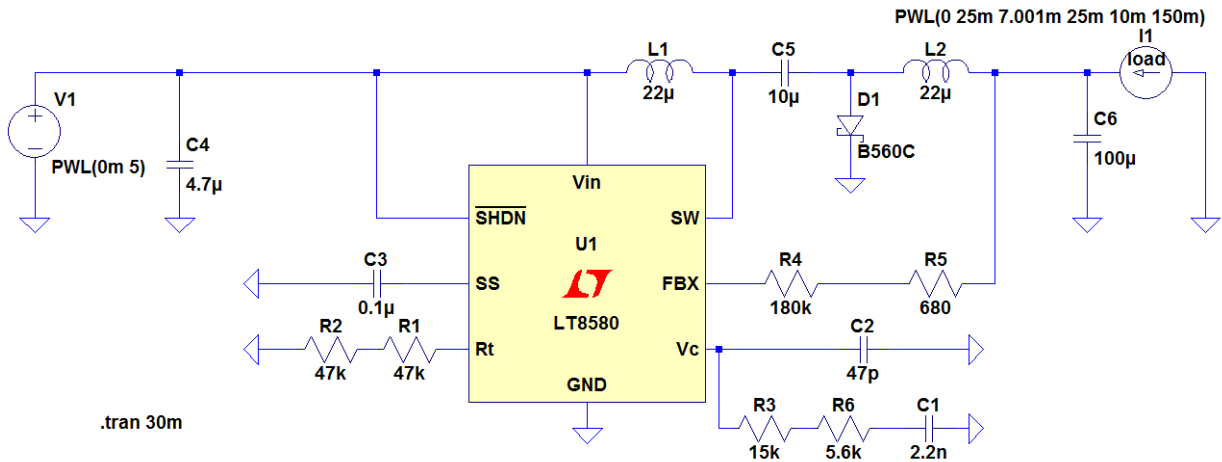


Figure 164 - N15V power supply configurations

Table 42 - LT8580 pin description

Name	NO.	Description
FBX	1	Positive and negative feedback pin
VC	2	Error amplifier output pin
VIN	3	Input supply pin
SW	4	Switch pin
nSHDN	5	Shutdown pin, active low
RT	6	Timing resistor pin. Adjusts the switching frequency
SS	7	Soft-start pin
SYNC	8	Synchronize the switching frequency to an outside clock
GND	9	Ground

<sup>110</sup> LT8580 Datasheet



#### 6.4.6.1. LT8580 Design decisions

The input voltage range is between 4.5 - 20 V and the output is -15 V. The switching frequency of this converter is set at 0.9 MHz and is set by the resistor of 94 kΩ between  $R_T$  and ground. The resistor has an original value of 94 kΩ, and on Figure 164 the resistor is replaced by two resistors of 47 kΩ, due to choosing the nearest E12 resistor value. The error from changing this resistor from 94 kΩ to two 47 kΩ is 0%. The input is filtered with a 4.7 μF capacitor (see 6.4.7 for calculations).

Next step in the design is to calculate the duty cycle, which is given in (138).

$$D_C = \frac{|V_{OUT}| + 0.5V}{V_{IN} + |V_{OUT}| + 0.5V - 0.4V} \quad (138)$$

Inserting values for  $V_{IN}$  and  $V_{OUT}$  and we find out that the duty cycle is approximately 80%.

We have now the necessary information to calculate and choose inductors (L1 and L2). To do this we calculate the typical, minimum and maximum inductor size. See (139), (140) and (141)

$$L_{TYP} = \frac{(V_{IN} - 0.4) \cdot D_C}{f_{OSC} \cdot 0.3 A} \quad (139)$$

$$L_{MIN} = \frac{(V_{IN} - 0.4) \cdot (2D_C - 1)}{1.25 \cdot (D_C - 300 ns \cdot f_{OSC}) \cdot f_{osc}(1 - D_C)} \quad (140)$$

$$L_{MAX} = \frac{(V_{IN} - 0.4) \cdot D_C}{f_{OSC} \cdot 80 A} \quad (141)$$

Inserting values into the equations above, we calculated that the values for typical, minimum and maximum inductors to be 13 μH, 19 μH and 49 μH. It was recommended in the LT8580 that the inductor should have values over the typical and minimum values but lower than the maximum value. There was also the choice between coupled inductor or uncoupled. For the coupled case the inductors  $L = L1 = L2$  (calculated) and for the uncoupled  $L = L1 \parallel L2$ . The benefits for choosing the coupled inductors is that they require less board area on the PCB, they are inexpensive and are easier to integrate, but however there is a limited selection of high-power off-the-shelf inductors, which is a



problem. The only solution than is to design a coupled inductor with all electrical parameters. As for choosing two uncoupled inductors often offers a broad off-the-shelf selection, since the parameters of L1 and L2 does not need to be identical, which also provides greater flexibility. We choose to use an uncoupled version due to the flexibility and off-the-shelf availability; this is because we want to be able to do as much re-work as possible on the motherboard.

Since we went for uncoupled inductors  $L = L1 \parallel L2$ , after careful consideration, checking availability on databases and availability on the market we went with having an L approximately 47  $\mu\text{H}$ . By choosing L1 and L2 to be equal, we get that L1 and L2 needs to be 22  $\mu\text{H}$ .

Next step is to calculate the ripple current in (142), to be able to determine the maximum output current.

$$I_{RIPPLE} = \frac{(V_{IN} - 0.4 V) \cdot D_C}{f_{OSC} \cdot L} \quad (142)$$

$I_{RIPPLE}$  was calculated to be 49.2  $\mu\text{A}$ .

The maximum current is given by (143)

$$I_{OUT} = \left(1 A - \frac{I_{RIPPLE}}{2}\right) \cdot (1 - D_C) \quad (143)$$

The maximum output current was calculated to be 200 mA.

We need to determine the reverse voltage that our diode needs to handle. This is done by using (144).

$$V_R = V_{IN} + V_{OUT} \quad (144)$$

$V_R$  is calculated to be 20 V. This means that the maximum reverse voltage that our diode needs to handle is 20 V. The diode that is used is a B560C from (Diodes Inc.) is a diode with a breakdown voltage at 60 V and can handle currents up to 5 A. We choose to use this because this is the diode that we are using for the other DCDC converters and to not add to the BOM, we chose to use the same.



In this topology the output is disconnected from the input by having a capacitor between L1 and L2. In the datasheet it is recommended that this capacitor is larger than 1  $\mu\text{F}$ . A 10  $\mu\text{F}$  capacitor was found to be suitable after running simulation and confirming that the transient response is still within calculations.

To determine the in capacitor and out capacitor equation (145) and (146) is used.

$$C_{IN} \geq C_{VIN} + C_{PWR} \frac{1A \cdot D_C}{40 \cdot f_{OSC} \cdot 0.005 \cdot V_{IN}} + \frac{I_{RIPPLE}}{8 \cdot f_{OSC} \cdot 0.005 \cdot V_{IN}} \quad (145)$$

$$C_{OUT} \geq \frac{I_{RIPPLE}}{8 \cdot f_{OSC} \cdot 0.005 \cdot |V_{OUT}|} \quad (146)$$

$C_{IN}$  and  $C_{OUT}$  is calculated to be larger than 91 nF and 0.87  $\mu\text{F}$ . After running simulations and doing research we came with the conclusion that a value of 4.7  $\mu\text{F}$  for  $C_{IN}$  and 300  $\mu\text{F}$  for  $C_{OUT}$  was sufficient for our application.

As previous mentioned the switching frequency is at 0.9 MHz and that the resistor connected between  $R_T$  and ground has a value of 94 k $\Omega$ . To be able to change the switching frequency (147) used.

$$R_T = \frac{85.5}{f_{OSC}} - 1 = 94 \text{ k}\Omega \quad (147)$$

To change the output voltage, equation (148) is needed to calculate the new  $R_{FBX}$  which is a single resistor to choose the output voltage.

$$R_{FBX} = \frac{|V_{OUT}| + 3 \text{ mV}}{83.3 \mu\text{A}} = 180.1 \text{ k}\Omega \quad (148)$$

The LT8580 also has a soft-start pin, and has a capacitor of 0.1  $\mu\text{F}$  connected between this pin and ground. The soft-start is at 35 ms and is given by (149)

$$T_{SS} = \frac{2.1 \text{ V} \cdot C_{SS}}{6 \mu\text{A}} \quad (149)$$



The  $V_C$  pin the error amplifier output pin, the compensation network is connected to this pin. The network consists of a series resistor-capacitor network in parallel with a capacitor. The compensation network was first set to the recommended values in the datasheet, and then changed, using simulations, to give sufficient transient response on the output.

Now that we have designed the converter, we need to run test to verify that it meets our requirements.



#### 6.4.6.2. Load test simulation

As with the 15 V supply, there is no need running a crank test, due to that the -15 V and 15 V is connected on the 5 V power supply. A load test will be done to see if the power supply manages to regulate with varying loads.



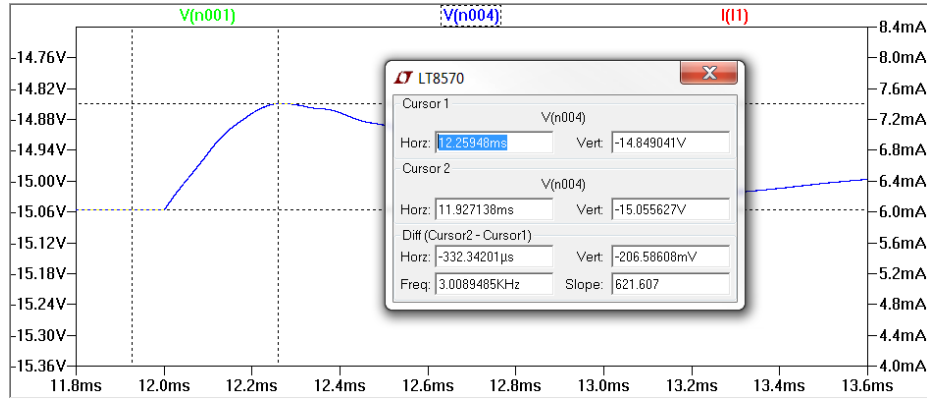
**Figure 165 - Load test for -15 V power supply**

The load is at the nominal value of 25 mA until the power supply has stabilized at -15 V, then the load suddenly changes to 150 mA, as can be seen in Figure 165. The regulation drops, but quickly starts to stabilize to the expected output voltage. The input voltage is set at a constant 5 V.

In Figure 166 it can be seen that when the load changes from 25 mA to 150 mA, the regulation drops to -14.85 V and it takes the power supply approximate 1.4 ms to stabilize back to -15 V. As with the 15 V power supply, we are allowed to have 5% deviation from the expected output voltage. This gives us the minimum output voltage of -14.75 V and a maximum output voltage of -15.25 V.

The regulator is within the range of the deviation, thus verifying that we can use this power supply. The ripple on the output voltage is 216  $\mu$ V.

The -15 V power supply can supply a maximum of 150 mA at 5 V. The input range of this power supply can vary between 4.5 - 20 V at a constant load of 150 mA. The calculation showed that the DCDC converter could handle 200 mA, to be sure, we recommend to never load this converter with more than 150 mA.



**Figure 166 - Regulation to sudden changes in load**

#### 6.4.6.3. Temperature

To calculate the heat dissipated in the converter, equation (150),(151),(152),(153) and (154) can be used.

$$\text{Average Input Current} = I_{IN} = \frac{V_{OUT} \cdot I_{OUT}}{V_{IN} \cdot \eta} \quad (150)$$

$$\text{Switch } I^2R \text{ Loss: } P_{SW} = D_C \cdot I_{IN}^2 \cdot R_{SW} \quad (151)$$

$$\text{Base Drive Loss (AC): } P_{BAC} = 20 \text{ ns} \cdot I_{IN} \cdot V_{OUT} \cdot f \quad (152)$$

$$\text{Base Drive Loss (DC): } P_{BDC} = \frac{V_{IN} \cdot I_{IN} \cdot D_C}{40} \quad (153)$$

$$\text{Input Power Loss: } P_{INP} = 6 \text{ mA} \cdot V_{IN} \quad (154)$$

Where,  $R_{SW}$  is the switch resistance (typically 530 mΩ at 0.75 A),  $D_C$  is the duty cycle, see (138) and  $\eta$  is the efficiency (typically 85%). The total power dissipated is then given by (155).

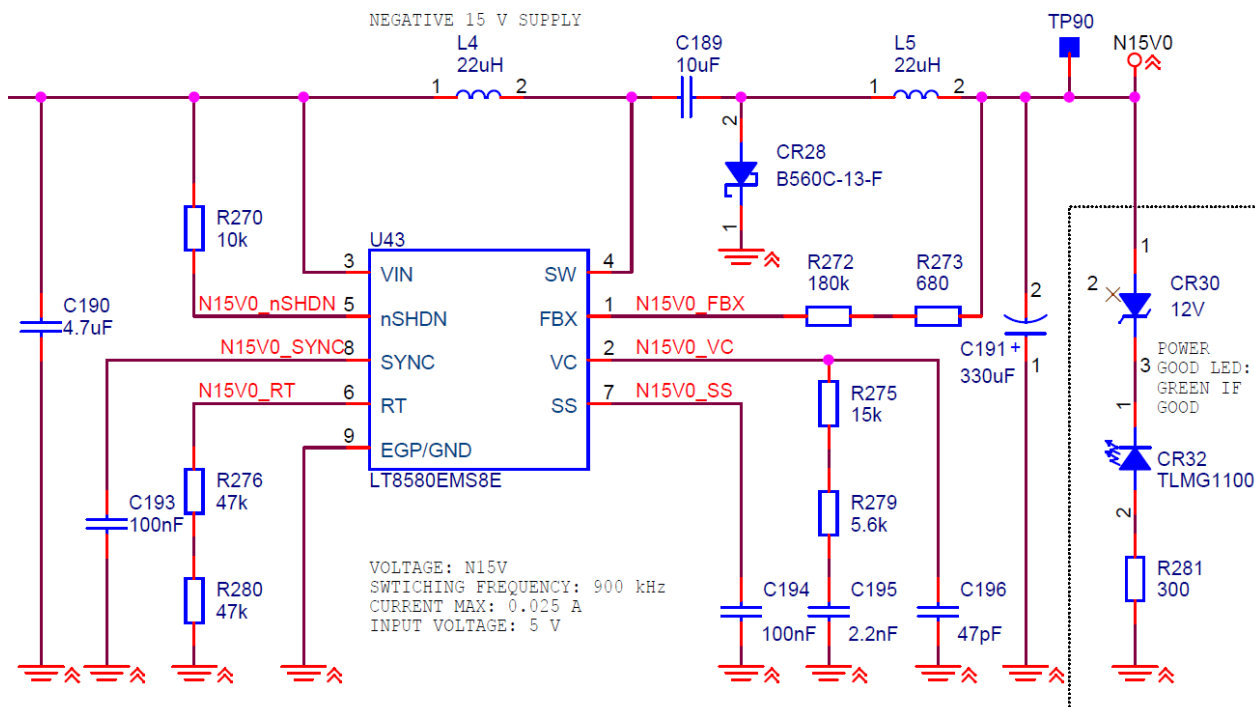
$$P_{TOT} = P_{SW} + P_{BAC} + P_{BDC} + P_{INP} \quad (155)$$

The total power dissipated in the LT8580 is than calculated to be 0.23 W.



$$T_J = T_A + \theta_{JA} \cdot P_{TOT} \quad (156)$$

The junction temperature is then calculated to be 39 °C.



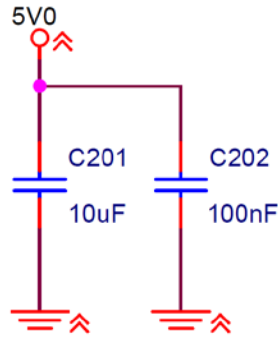
### Figure 167 -15 V power supply schematic implementation



#### 6.4.7. Additional notes

All DCDC converters have an output bulk capacitor to handle sudden load changes. Additional to these bulk capacitors, there are also added two low-ESR ceramic capacitors to help deal with fast load transients due to that the bulk capacitors are too slow. All capacitors output capacitors where chosen after running simulations using LTSpice.

The implementation is seen in Figure 168.



**Figure 168 - Low-ESR capacitors to help with fast load transients**

All input capacitors of DCDC converters was calculated using (157)<sup>111</sup>

$$C_{MIN} > \frac{I_{OUT} \cdot D_C (1 - D_C) \cdot 1000}{f_{SW} \cdot V_{P(MAX)}} \quad (157)$$

Where  $D_C$  is given in (158)

$$D_C = \frac{V_{OUT}}{V_{IN} \cdot \eta} \quad (158)$$

To summarize the power design, Table 43 gives the specifications of each DCDC converter.

**Table 43 - Power design summary**

V	$V_{MIN}$ (V)	$V_{TYP}$ (V)	$V_{MAX}$ (V)	$I_{MIN}$ (mA)	$I_{TYP}$ (mA)	$I_{MAX}$ (mA)	$f_{SW}$ (kHz)
3.3	3.23	3.29	3.31	47	2900	4500	350
5	4.8	4.9	5.1	7.8	1226	3000	180
-5	-5.05	-5.03	-4.98	0	200	600	200
15	15.07	15.07	15.07	0	14	25	700
-15	-15.03	-15.02	-14.98	0	14	25	900

<sup>111</sup> Texas Instrument, Input and output Capacitor selection

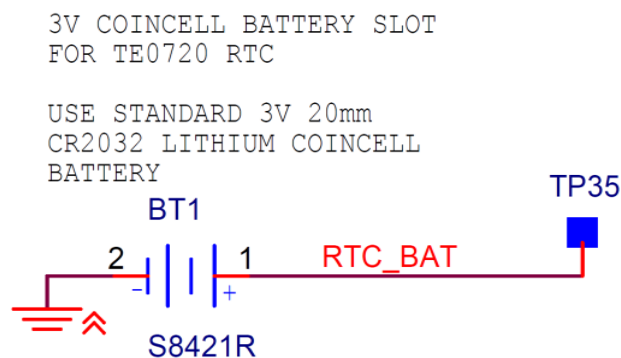


## 7. Miscellaneous features

This section contains features that is not specified under the requirement of the product, but which we saw essential to the product as it provides flexibility during testing and for future upgrades. These features include RTC battery, programmable LED's, test points, 0  $\Omega$  resistors, reset jumper, PGOOD LED's and general purpose inputs and outputs.

### 7.1. RTC (Real Time Clock) battery

When the ECU is disconnected from the battery, the ECU will shut down and the internal system clock will reset to its default state. To prevent this from happen, there has been implemented a RTC battery holder. The RTC battery holder will be implemented on the ECU and will be providing power to the ECU's system clock so that it remembers the time and date. The implementation is seen in Figure 169.

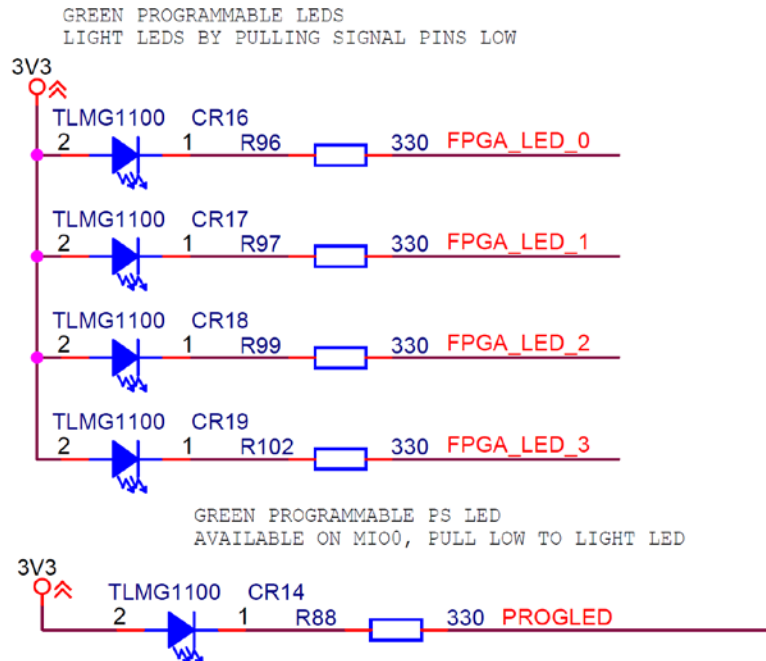


**Figure 169 - RTC battery holder implementation**

The battery that must be inserted is a standard 3 V, 20 mm CR2032 coin cell battery.

### 7.2. Programmable LEDs

Programmable LEDs provide visual assistance when programming and testing software code. Being able to toggle LEDs from different I/O pins on the TE0720 allows visual confirmation that the code functions as intended. When testing new features, the LEDs can be used for displaying different status flags or interrupt events from board components. The TE0720 itself has 3 LEDs. The KPEC ECU motherboard has a total of 5 programmable LEDs. 4 of the LEDs are connected to the programmable logic, and the last LED is connected directly to MIO pin 0 of the processing system on the Zynq.



**Figure 170 - Programmable LEDs schematic implementation**

Figure 170 shows the schematic implementation of the programmable LEDs. The LEDs are pulled high with common anode connection, and the connected pin must be pulled individually low in order to light them in software, sinking current from the 3.3 V supply through the LEDs. The current passing through the LEDs is given by (159):

$$I_D = \frac{V_{in} - V_D}{R_D} = \frac{3.3 \text{ V} - 2.1 \text{ V}}{330 \Omega} = 3.63 \text{ mA} \quad (159)$$

The average forward voltage drop  $V_D$  of the LED is 2.1 V. Table 44 shows the connection of the LEDs to the Zynq SoC. All the FPGA LEDs are connected to bank 33 pins, while MIO0 is in bank 500.

**Table 44 - Programmable LEDs pin configuration**

ZYNQ pin	LED
MIO0	PROGLED
B33_L4_P	FPGA_LED_0
B33_L4_N	FPGA_LED_1
B33_L14_P	FPGA_LED_2
B33_L14_N	FPGA_LED_3



### 7.3. Test points

To be able to test the design once it has been produced, it is important to provide the one who is testing with enough test points, so that there is possible to access the test point with test probes or other test equipment. Test points are essentially a pad that is connected to the trace. A test point does not affect the performance of the circuit, thus it is beneficial to provide with sufficient test points, so that the most part of the PCB can be tested. Test points are inserted on inputs, outputs and essential points of the circuit, this provides the ability to test, debug and verify that a circuit on the PCB is working. It also gives the flexibility to probe internal traces that are hard to access on the PCB. There are two kinds of test points that are used. A circle test point is used for signal lines and square test points are used for power lines.



Figure 171 - Test point pads layout

### 7.4. 0 $\Omega$ resistors

0  $\Omega$  resistors are used in large quantity over the entire design. These resistors are there to provide flexibility to the design. 0  $\Omega$  is essential a short, but it leaves a footprint so that other resistors can be soldered on. This provides the design with great flexibility due to the amount of configuration possibilities that are granted. As an example of the usage of 0  $\Omega$  resistors are in the negative feedback path of the filters throughout the design, this grants the user the ability to change the gain of the filter on every stage. This is illustrated on Figure 172.

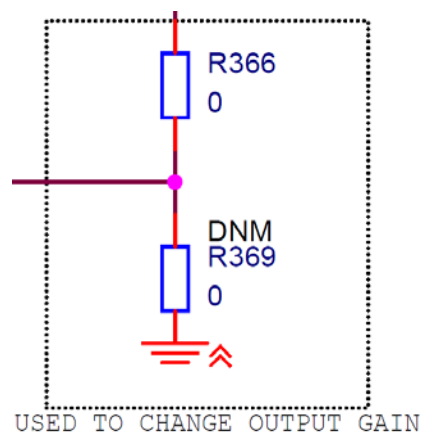
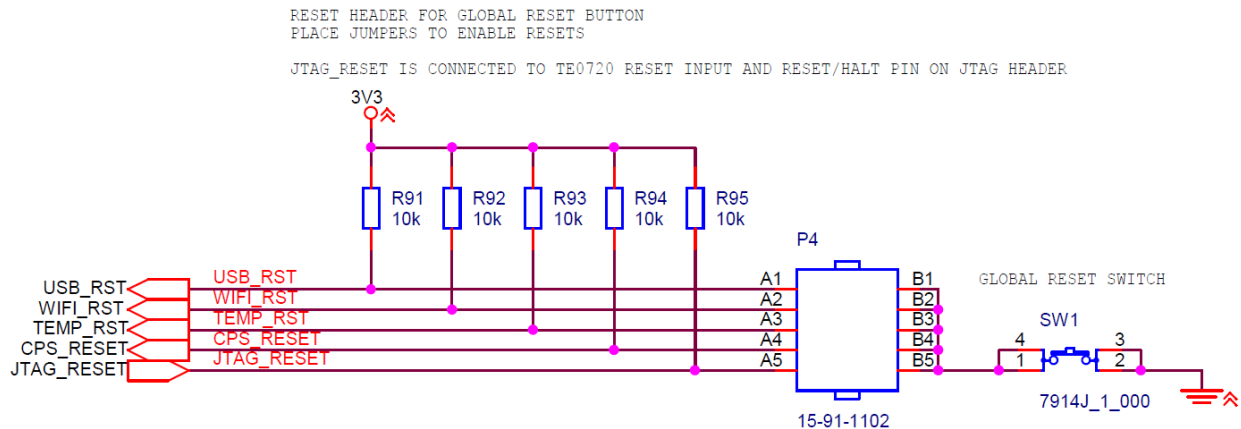


Figure 172 - 0  $\Omega$  resistors implementation example



## 7.5. Reset button and jumper configurations

Several of the on-board components have a reset functions. These tend to be active low signal inputs, where the signals are actively pulled high by a weak pull up resistor to  $V_{DD}$ . In order to trigger a reset, the signal on the reset input of the ICs are typically pulled low for a given minimum time before returning to pulled high state. A way of performing this is to use a tactile switch which grounds the reset pins of the components when depressed. The KPEC ECU has a global reset switch that can be disconnected from different components by placing or removing jumpers.



**Figure 173 - Reset switch and jumper header**

Figure 173 shows the schematic implementation of the reset toggle circuitry. All reset signals are pulled high at the header in order to not trigger false resets. The reset connector is a standard 8 position, 4 row 2 column 2.54 mm pitched header. When jumpers are placed across the header, the respective reset signals are connected to the tactile switch. The JTAG\_RESET signal is connected both to the JTAG reset position on the JTAG header as well as the reset signal input to the TE0720. Any jumper connecting signals across the reset header will also toggle reset if JTAG reset is used on the Xilinx JTAG header described in 5.2



## 7.6. PGOOD LED's

The PGOOD (Power Good) is a pin on the 3.3 V and 5 V power supplies; this pin is an open drain output and is used to tie a LED to ground when the power supply is working as intended. With this LED the tester/user can see when the power supplies (3.3 V and 5 V which are the most critical supplies) are working properly. A green light indicating that the supply is working and no light indicates errors in voltage regulation or worst case malfunction. Additional LED's are connected on those power supplies that does not have PGOOD pin to act as an indicator that the power supplies are working, this implementation is done on the negative 5 V, positive 15 V and negative 15 V. The LED is connected in back to back configuration with a Zener and a current limiter resistor in series. Both configurations are seen in Figure 174. Furthermore the PGOOD pin on the TE0720 lights a green LED on the ECU motherboard when *all* supplies on the module signals PGOOD.

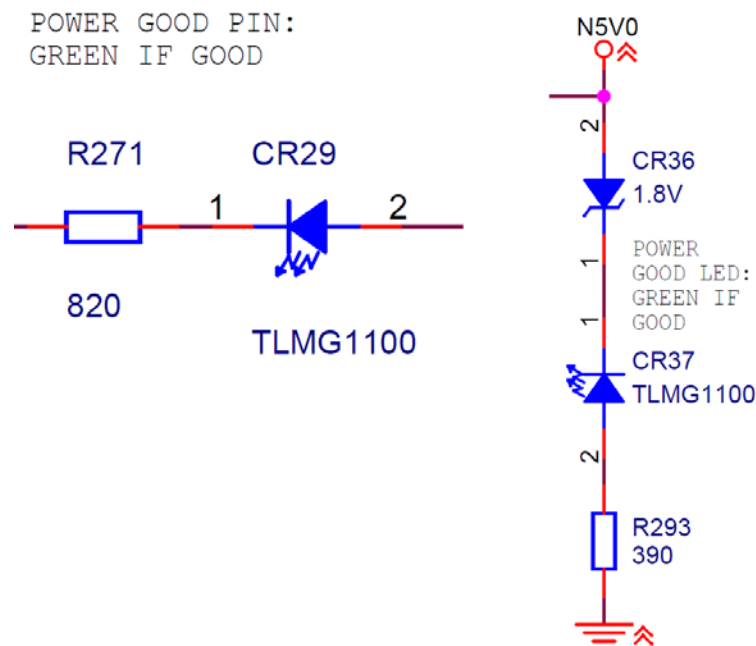


Figure 174 - PGOOD LEDs implementation



## 7.7. General purpose inputs and outputs

To be able to upgrade or expand further features we have implemented different kinds of general purpose inputs and outputs. These general purpose inputs and outputs are divided into two subgroups, analog and digital. Where the 3 analog inputs are used to interface other sensors with a 0 - 5 V output range. While the 2 analog outputs can be used to communicate with many types of displays such as lambda display and RPM meter or any application that has an input range from 0 - 12 V. There are also 2 dedicated digital inputs which are used to interface digital sensors. Additional we have added 4 general digital inputs and outputs which can be used to communicate with other 5 V systems. To be able to extend the ECU capabilities to handle more actuators (if needed), we have implemented 4 general purpose low-side switches that can be used to interface new actuators such as a second boost controller if a twin turbo engine is being used.

Table 45 Shows the different general purpose inputs and outputs and where to read more about them in detail.

**Table 45 - General purpose inputs and outputs**

Interfacing type	Description	Section
<b>GPAI</b>	General purpose analog input	3.7.7
<b>GPAO</b>	General purpose analog output	3.8.2.6
<b>GPD I</b>	General purpose digital input	3.10.5
<b>GPDIO</b>	General purpose digital in- and output	5.7
<b>GPLSS</b>	General purpose low-side switch	4.7





## 8. Derating of electrical components

Derating is a set of requirements that specify design tolerances for different types of components in the electric design. All components used in the electronic design of the KPEC ECU must follow derating requirements. The basic principle of derating components is to ensuring that they operate at less than maximum ratings in order to extend lifetime. Allowing large tolerances from manufacturer's specifications ensures higher reliability for the finished product.

The derating data in this section is based on MIL-HDBK-978-B<sup>112</sup>, MIL-HDBK-217F<sup>113</sup> and recommendations from KDA. The operating conditions that are taken into consideration for the KPEC ECU and background for derating values are for operating under *normal* conditions.

*Normal* operating conditions are defined as follows;

- Used in applications usually accessible for maintenance or replacement
- Used in an uncontrolled temperature environment with a range of -40 °C to 85 °C
- Used in an environment having minimal to medium controlled shock, vibration or pressure
- Shelf life for a minimum of 10 years
- Application life span of 5 to 10 years

As the specifications of the derating data are based on are military standards, derating *normal* operating conditions are well within the requirements for our application.

---

<sup>112</sup> NASA Parts Application Handbook MIL-HDBK-978-B

<sup>113</sup> Department of Defense: Reliability Prediction of Electronic Equipment MIL-HDBK-217F



## 8.1. Derating factors

The derating data is displayed and calculated with the use of different factors explained in Table 46.

**Table 46 - Derating factors**

Derating factor	Description
$B_{max}$	Maximum load given from manufacturer
$B_0$	Maximum load after derating
$S_D$	Derating factor

The maximum load for a stress factor for a component can be calculated by using (160):

$$B_0 = B_{max} \cdot S_D \quad (160)$$

The stress factors are used to calculate the derating limits. There are several stress factors that can affect the lifetime of a component. Most types of components are to some extent degraded over time, which may lead to severe failures after extended use.

Typical stress factors for an electrical component are:

- Current
- Voltage
- Power
- Temperature
- Frequency



## 8.2. Component derating tables

The following section initiates tables to describe derating for different components and subtypes of these. These must be followed in order to ensure high reliability in the finished ECU.

### 8.2.1. Capacitors

There are many types of capacitors made from different materials. The different types have wide ranges of capacitances and maximum ratings, as well as different usage areas. Only the three most commonly used capacitors are described in this section, as other types are not applicable for use in our design.

**Table 47 - Capacitor derating table**

Type	Derating Parameter	Derating Level
Ceramic (X7R, X5R, C0G, NP0)	DC voltage	60%
	Temp. from max limit	$T_{\max} - 10\text{ }^{\circ}\text{C}$
Electrolytic Aluminium	DC voltage	70%
	Temp. from max limit	$T_{\max} - 10\text{ }^{\circ}\text{C}$
Electrolytic Tantalum	DC voltage	60%
	Temp. from max limit	$T_{\max} - 10\text{ }^{\circ}\text{C}$
Solid Tantalum/Aluminium	DC voltage	70%
	Temp. from max limit	$T_{\max} - 10\text{ }^{\circ}\text{C}$
	Reverse voltage	50%

Table 47 shows derating data for different types of capacitors. The derating level implies the percentage of the maximum rating specified in the component datasheet.

#### 8.2.1.1. Tantalum capacitors

Tantalum capacitors are a type of electrolytic capacitors and are polarized. They come in medium to large capacitance values. Many low-ESR types can be useful where relatively large capacitance values are required, but also faces strict ESR-requirements.

There are two main weaknesses that must be taken into care when using tantalum capacitors:

- Withstands little ripple current as this generates heat in the capacitor body. This can have a drastic effect on lifetime or even risk of destroying the capacitor if used for application with typically large ripple current, such as switched-mode power supplies.
- Withstands low inrush current, and should not be used where large transients may occur.



### 8.2.1.2. *Aluminium electrolyte capacitors*

Aluminium electrolyte capacitors usually come in large capacitances. The capacitance is often large compared to unit volume in comparison other types of capacitors. They are usable for large voltage ranges and come in a wide array of package and capacitance values. Aluminium Capacitors are polarized, and may explode if the polarity is reversed in the circuit they are used. Aluminium electrolyte capacitors are often used for temporary energy storage in driver circuitry and smoothing of rectified DC voltage. They are almost always used with switched-mode supplies.

Many types of aluminium electrolyte capacitors use a liquid electrolyte. The electrolyte slowly evaporates over time. Over time, this increases ESR and lowers capacitance<sup>114</sup>, which affects overall performance and reliability of the system. Temperature is an important factor for loss in evaporated electrolyte. While the capacitors themselves tends to be reliable, the limited shelf life (also while unpowered) may affect overall product lifetime.

### 8.2.1.3. *Ceramic capacitors*

Ceramic capacitors are not polarized and usually come in small capacitances with small footprints. They are used very much in filters and decoupling. There are two main classes of ceramic capacitors in use. The two types are described by IEC 60384-8<sup>115</sup> and IEC 60384-9<sup>116</sup>.

- Class 1 are high stability and low losses. Often denoted NP0 or C0G and used in resonant circuit with high stability demands.
- Class 2 are more common in use and have generally higher capacitance values than Class 1. The most used are denoted X5R and X7R which are describing the temperature ratings and stability. Class 2 ceramic capacitors are generally used for decoupling, filters and smoothing signal from power supplies.

Generally, ceramic capacitors are best to use anywhere possible, but high voltage and/or high capacitance requirements may necessitate the use of electrolytic capacitors described in 8.2.1.1 and 8.2.1.2.

It is important to note that capacitance ratings are not stable at increasing voltages. As X7R rating of class 2 capacitors only specifies temperature rating, many capacitors can drop significantly in capacitance at rated voltage compared to zero

---

<sup>114</sup> Vishay 28356: Introduction Aluminium Capacitors

<sup>115</sup> International standard IEC 60384-8

<sup>116</sup> International standard IEC 60384-9



voltage. Use of aerospace- and military rated components can be an assurance to ensure that the capacitors are of sufficient quality in high reliability design.

### 8.2.2. Resistors

Resistors are not as prone to errors due to voltage effects as capacitors, and are generally static in resistance. Resistors are very reliable components, with power dissipation normally specified in the datasheets at room temperature to 70 °C. This section will not go into detail of different resistor types, as the only applicable type in our design is the thick film CRCW-series<sup>117</sup> resistors from Vishay. These resistors are fairly precise at 1% maximum error, and qualified for automotive and military use.

**Table 48 - Resistor derating table**

Type	Derating Parameter	Derating Level
Resistor (all types)	Power dissipation	50%
	Temp. from max limit	T <sub>max</sub> - 30 °C

Resistors have maximum voltages that must not be exceeded in order to ensure reliability. The maximum power dissipation must be derated according to Table 48. If ambient operational temperatures exceed 70 °C, additional derating may be required.

### 8.2.3. Transistors

Transistor derating data is given by Table 49. There are not that many considerations to make in addition to the derating data. Note that SMD transistors switching large current loads must be sufficiently coupled to large planes for proper heat sinking.

**Table 49 - Transistor derating table**

Type	Derating Parameter	Derating Level
Transistor (All types)	Power dissipation	70%
	Breakdown voltage	75%
	Max junction temp.	T <sub>max</sub> - 10 °C
Thyristor	Power dissipation	50%
	Peak reverse voltage	60%
	Peak forward voltage	75%
	Max junction temp.	T <sub>max</sub> - 10 °C
	On-state current	70%
	Off-state current	70%

<sup>117</sup> Vishay D/CRCW e3 Datasheet



### 8.2.4. Diodes

Diodes are derated in similar ways as transistors, with temperature as the most important factor for estimating reliability. The derating data is given in Table 50.

**Table 50 - Diode derating table**

Type	Derating Parameter	Derating Level
Signal/switch	Forward current	90%
	Reverse voltage	75%
	Surge current	80%
	Temp. from max limit	$T_{\max} - 10\text{ }^{\circ}\text{C}$
Voltage regulator, voltage reference, rectifier, schottky	Power dissipation	70%
	Reverse voltage	75%
	Temp. from max limit	$T_{\max} - 10\text{ }^{\circ}\text{C}$
TVS	Power dissipation	70%
	Average current	80%
	Temp. from max limit	$T_{\max} - 10\text{ }^{\circ}\text{C}$
LED	Average fwd. current	50%
	Temp. from max limit	$T_{\max} - 10\text{ }^{\circ}\text{C}$

### 8.2.5. Magnetic components

In addition to derating characteristics, self-heating from the component should not exceed  $40\text{ }^{\circ}\text{C}$  from ambient temperature. The derating data for magnetic components are shown in Table 51.

**Table 51 - Magnetic component derating table**

Type	Derating Parameter	Derating Level
Beads	Power dissipation	50%
	DC current	90%
	Hot spot temp.	$T_{\max} - 10\text{ }^{\circ}\text{C}$
Transformers, coils, inductors	Power dissipation	50%
	Peak reverse voltage	60%
	Max junction temp.	$T_{\max} - 30\text{ }^{\circ}\text{C}$



### 8.2.6. Integrated circuits

Integrated circuits are in many categories and wide array of packages. Therefore, specific derating data must always be controlled against datasheet recommendations, especially for power ICs as they often dissipate much more power and may need additional derating. Other than specified in Table 52, the following should be considered:

- Max frequency should not exceed 80% of maximum frequency (digital)
- Input voltage (analogue) should not exceed 70% of maximum rating

**Table 52 - Integrated circuit derating table**

Type	Derating Parameter	Derating Level
IC (MOS/bipolar)	Supply voltage	±5% (or specified)
	Output current	80%
	Fan-out	80%
	Frequency	80%
	Max junction temp.	T <sub>max</sub> - 10 °C

### 8.2.7. Optocouplers

Optocouplers in general can be unreliable, and should not be used where possible. Alternatives for isolation circuits for digital signals exist, but are not applicable in every type of signal circuit.

The derating data for optocouplers are shown in Table 53.

**Table 53 - Optocoupler derating table**

Type	Derating Parameter	Derating Level
Photodiode, phototransistor	Reverse voltage	70%
	Current	75%
	Power dissipation	80%
	Max junction temp.	T <sub>max</sub> - 10 °C



### 8.3. Power dissipation derating of components

This section describes the steps for calculating power derating in components. Note that these parameters are not always given in the component datasheets. Some may be given by a figure or a graph.

**Table 54 - Power dissipation parameter description**

Parameter	Description
$T_{A,max}$	Component maximum ambient temperature
$P_D$	Power dissipation in the component
$P_{D,max}$	Maximum power dissipation in the component
$P_{0D}$	Derated maximum power dissipation
$T_{J,max}$	Maximum junction temperature
$\theta_{JA}$	Thermal resistance, junction to ambient
$\theta_{JC}$	Thermal resistance, junction to case
$S_D$	Derating factor

The parameters in Table 54 are required for calculating the power dissipation in components.

$$P_{D,max} = \frac{T_{J,max} - T_{A,max}}{\theta_{JA}} W \quad (161)$$

The maximum power dissipation can be calculated by inserting parameters into (161).

The derating factor  $S_D$  is often given as a percentage.

$$P_{0P} = P_{D,max} \cdot \frac{S_D}{100} W \quad (162)$$

The new maximum allowed power dissipation at maximum ambient temperature can be calculated using (162). Note that (162) is just a rewritten form of (160), with the derating factor given as a percentage as in the derating tables in section 8.2. The maximum junction temperature can be computed using (163).

$$T_{J,operating} = T_{A,max} + P_{0P} \cdot \theta_{JA} \quad (163)$$





## 9. Software design

This chapter gives an overview of the main parts in the software system. Here we will describe design decisions, progress and challenges we have encountered, and how we solved them. More details about these parts will follow further down in the chapter. The following diagram, Figure 175, is an overview of the top level structure for this year's project.

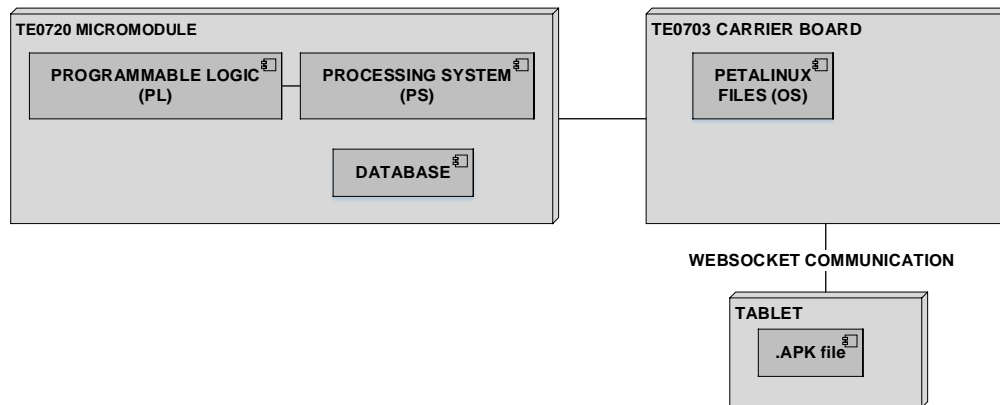


Figure 175 - Top level overview

### 9.1. Zynq module (TE0720)

The TE0720 Micromodule from Trenz Electronic is our main hardware for this project. The TE0720 is the heart of our engine control unit. It consists of two main components, programmable logic (PL) and processing system (PS). The programmable logic handles the FPGA bitstream while the processing system, which runs ARM dual-core Cortex-A9, will run the PetaLinux platform. The module also consists of RAM and flash memory to store the code and configuration.

### 9.2. Carrier board (TE0703)

The carrier board (TE0703) is used to provide us required connections and extensions to TE0720. This gives us the benefit to work with software development while the motherboard is under development by the hardware engineers. But the carrier board will not be a part of the finished engine control unit. For this year we are using the carrier board to get access to Ethernet communication for our TE0720. We are also storing the required files, to get PetaLinux up and running, into a SD card which is connected to the carrier board.

### 9.3. Tablet device

While the TE0720 handles the processing, a tablet will be used to see the results from an Android application. For this year we will set up the Graphical User-Interface (GUI) through an Android application, which will read different parameters coming from the TE0720. The application will then view the parameters in different views such as a dashboard, various diagrams or in digital format.



## 10. Zynq design

This section guides us through the process of using Xilinx Vivado to create an ARM Cortex-A9 based processor design targeting our Zynq SoC (System On Chip). We are using Xilinx Vivado 2014.4<sup>118</sup> to create the hardware system with help of IP cores. We are also using Software Development Kit (SDK) to create the applications needed to be run on top of the hardware platform. We will describe the decisions we have taken concerning the development process in order to install and configure PetaLinux on TE0720.

### 10.1. Configuring custom hardware

The first step of building the PetaLinux OS (operating system) for the TE0720 development board was to select the right development board, in order to implement the specifications needed. The board files includes all the basic components of hardware, design tools, IP, and pre-verified reference design including a targeted design, this enables a complete embedded processing platform. The development board, TE0720, is bought from Trenz Electronic. Therefore we had to download the re-customized design board manually from their website in order to choose the correct layout in Vivado.

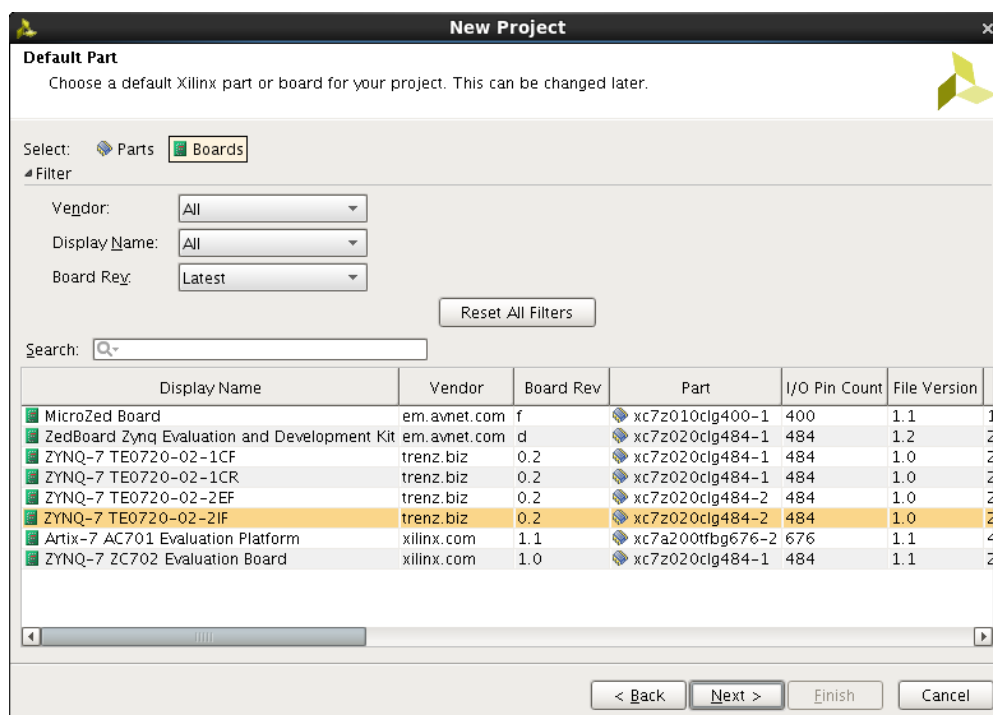


Figure 176 - The board selection in Vivado

<sup>118</sup> Xilinx Inc, Xilinx Vivado and SDK 2014.4



### 10.1.1. Implement Hardware

In order to build PetaLinux using Vivado design, there are a number of hardware and peripheral IP configuration, modifications required to ensure that the hardware platform is Linux-ready. The TE0720 contains many different components that can be designed in several ways. To be able to design the TE0720, we have used Xilinx Vivado 2014.4. Xilinx Vivado supports IP integrator, which allows us to design the TE0720 using block diagrams. Since this project is supposed to last for several years, it is a good way to design the TE0720 by IP integrators. Designing this way will make it easier to upgrade or modify the existing design later in the project phase.

The ARM processing system is pre-configured with the I/O peripherals that are connected on the development board. By selecting the development board as shown on Figure 176, we are able to choose the correct layout for our TE0720 in Vivado. In order to build the operating system PetaLinux, we have configured the IP Block Design. Table 55 shows us what type of configurations we did in order to build a proper hardware design in Xilinx Vivado.

**Table 55 - Required Hardware for PetaLinux**

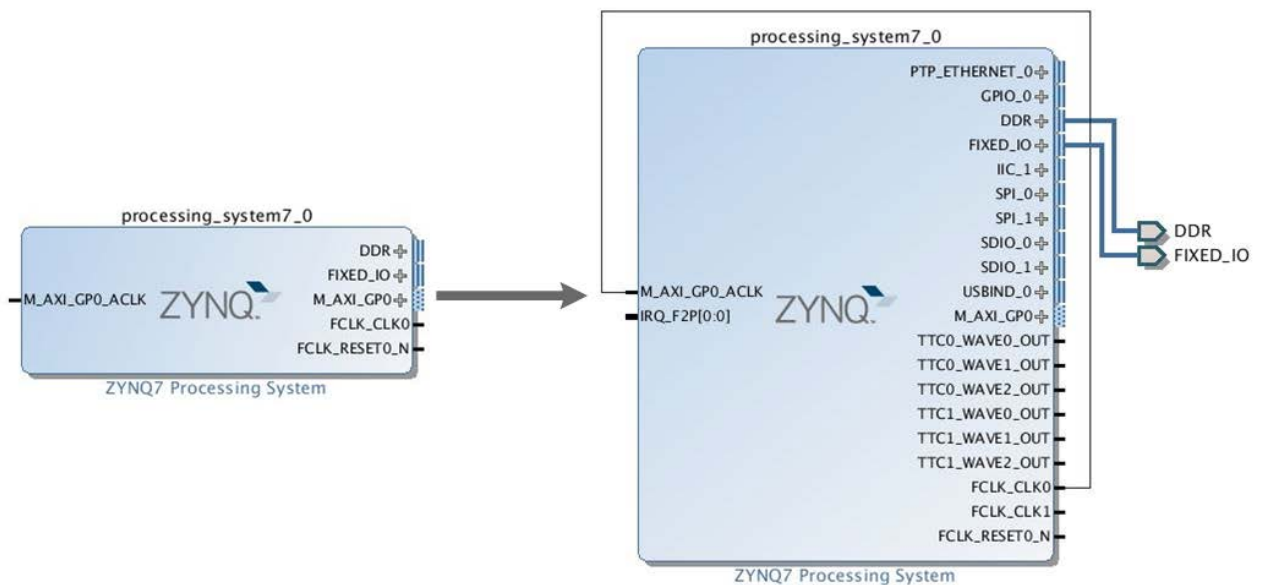
What	Why
Triple Timer Counter (TTC)	Use TTC as timers or output waveforms for EMIO and MIO pins
External Memory Controller with at least 32MB of memory	Required because we must have high performance executions
UART (Serial Communication)	Required, this is necessary to communicate through UART
Ethernet(Network Communication)	Optional, this is necessary to get network communication
SD Card (Boot Kernel)	Required if we wish to deploy kernel from SD card
Non-volatile memory (Optional) e.g. QSPI FLASH, SD/MMC	To store parameters from the ECU and other configurations and code



### 10.1.2. Export hardware

We need to generate a HDL file that is required for implementation, simulation and synthesis for the IP block design. In this design we only use the processing system and there is nothing designed in the programmable logic, but we still need to do this step in order to connect to the design on the top level.

Figure 177 shows the processing system without any configuration to the left. After configuring the processing system to our needs, we get the IP block at the right side. We can then use this design to setup PetaLinux.



**Figure 177 - Configured IP Processing System**



In order to open the integrated IP block design in SDK, we have to run block design automation. Zynq block automation applies current board preset and generates external connections for FIXED\_IO, Trigger and DDR interfaces. The reason we have to adjust the PL-PS configuration part is because we must choose the correct hardware layout to build PetaLinux OS. After these configurations given in Table 56, we can then export everything we have customized in Vivado to SDK, so we can build the PetaLinux bootloader. The design result is shown in Figure 177 after the configuration is done.

**Table 56 - Progress from Vivado to SDK**

What	Why
Run Block Automation	Applies current board preset and generates external connections
Validate Design	Check for errors or critical warnings in IP block design
Generate Output Products	Synthesis, implementation and simulation will be generated
Create HDL wrapper	Copy generated wrapper to allow user edits
Run Implementation p.1	Generate bitstream
Run Implementation p.2	Open implemented design
Export Hardware	Export hardware platform for software development tools
Launch SDK	Open SDK for further development



## 10.2. Software Development Kit (SDK)

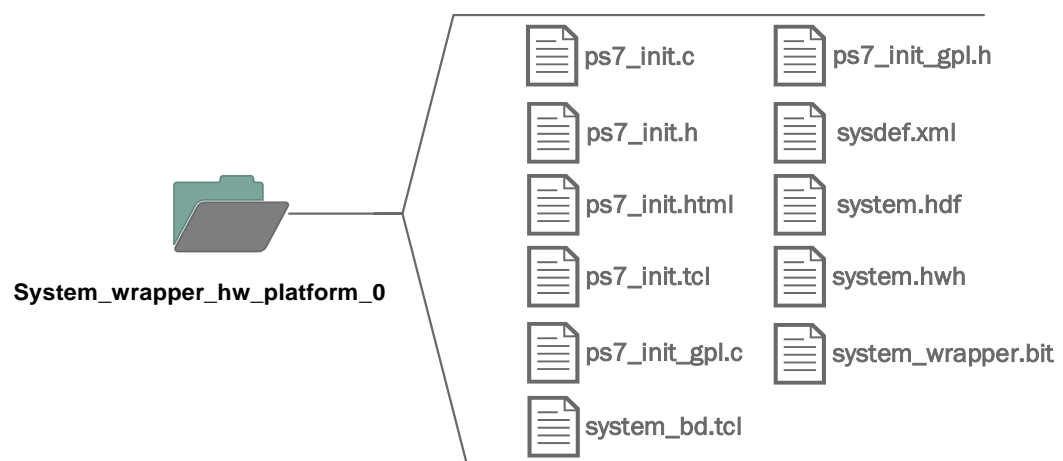
We are using Xilinx Software Development Kit (SDK) to continue on the design which was created in Xilinx Vivado, to further develop our system. SDK can interface the embedded hardware directly from Vivado with hardware and software co-debug capabilities. So we are using our IP block design to create applications for our design. SDK has many advantages such as editors, compilers, build tools, flash memory management and JTAG/GDP debug integration. We will build C/C++ program code using SDK and also use this software to analyse our system for potential bottlenecks.

### 10.2.1. Hardware platform specification

When exporting the hardware from Vivado, it generates a code and other files from the created hardware design, that are required for us to write, debug and deploy software applications.

There are two important files that are created, ps7\_init.c and ps7\_init.h. These files contain the initialization code for the TE0720 processing system and initialization settings for clocks, DDR, Phase Loop and MIOs. SDK uses these settings when initializing the processing system so that the applications can be ran on top of the processing system.<sup>119</sup>

- The system-wrapper.bit file is a bitstream corresponding to the hardware description. The FPGA bitstream programs a FPGA device with the hardware created by the hardware designer.
- TE0720 initialization files:
  - Ps7\_init.tcl: Contains the clock, Phase Loop and DDR initialization code.
  - Ps7\_init.c/.h: Used during the FSBL application project creation.



**Figure 178 - Build hardware platform**

<sup>119</sup> Xilinx: *Hardware Platform Specification*.



### 10.2.2. Build Bootloader

Inside the Software Development Kit (SDK), we have to create the First Stage Bootloader (FSBL). FSBL is a part of the Linux startup, which first of all configures the FPGA with HW bitstream and loads the operating system PetaLinux image from the non-volatile memory (NAND/SD/QPSI) to RAM and starts executing it.

The boot image can be deployed to the SD card. This will give us the opportunity to boot PetaLinux from the SD card and easier customize the operating system after we power up the board. In order to boot the operating system on the SD card, we have to build two files (image.ub and BOOT.bin). The BOOT.bin file contain FSBL image, FPGA bitstream and U-boot. To boot PetaLinux, we need a way to load the image.ub into the DDR and run U-Boot with the correct parameters.

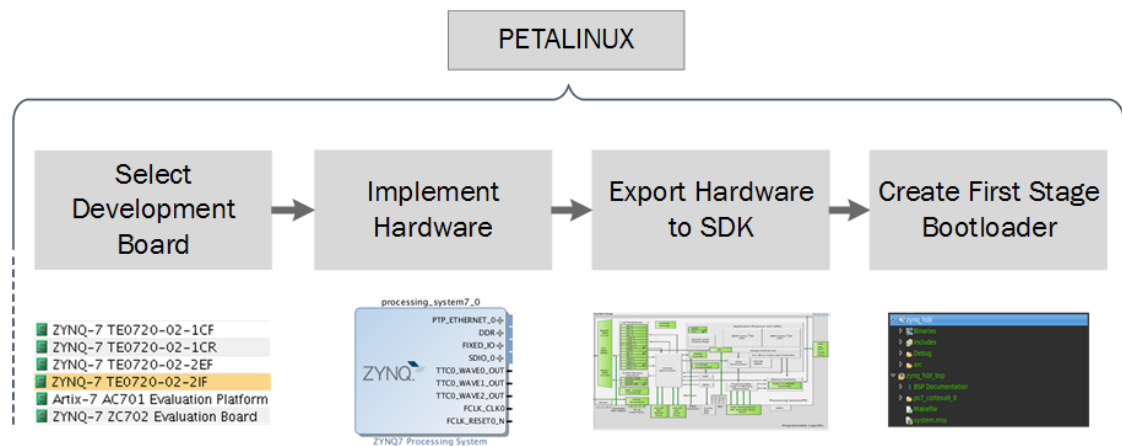


Figure 179 - Design process diagram



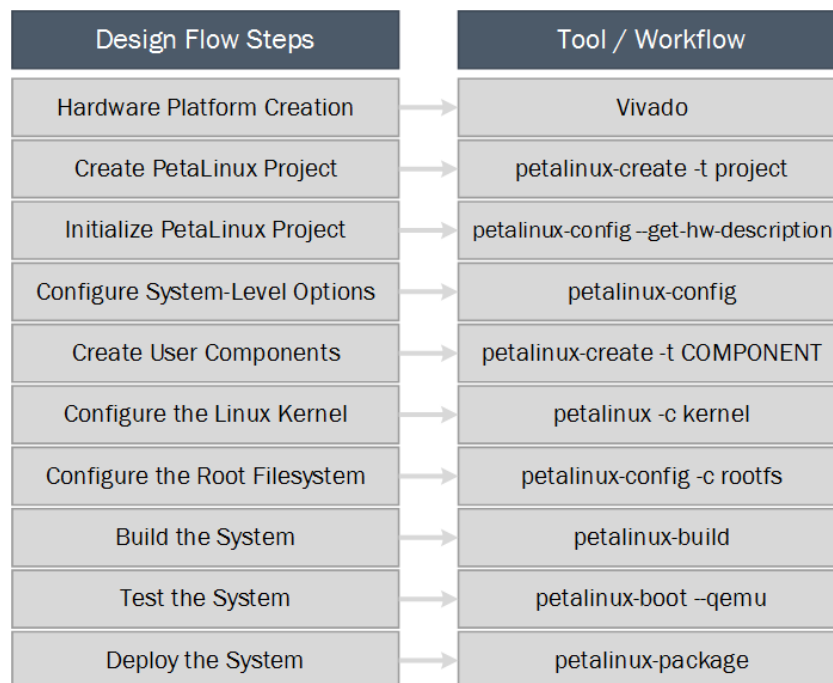
### 10.3. PetaLinux operating system

PetaLinux offers everything necessary to build, deploy and customize embedded Linux solutions on Xilinx processing systems.

The key features for PetaLinux are:

- Pre-configured binary bootable images
- Fully customizable Linux for TE0720
- PetaLinux SDK which includes tools to build a customized operating system for TE0720.

PetaLinux has a wide range of support, both from Trenz Electronic and Xilinx, which is good for further development of this project. Figure 180 below provides a design workflow, demonstrating the tasks that must be accomplished in order to build and configure PetaLinux for TE0720.



**Figure 180 - Flow Steps Diagram**





### 10.3.1. Why Processing System and PetaLinux?

We have decided to use a fully embedded Linux operating system on our TE0720 called PetaLinux. Since the micromodule contains both Processing System and Programmable Logic, it gives us many different ways to design the module. It is possible to use only the Processing System or the Programmable Logic individually. But in most cases it might be a beneficial to take use of both of them in the design.

While the Programmable Logic can process several tasks simultaneously, it can sometimes be too complex and hard to make algorithms for some tasks. The algorithms are mostly coded in VHDL or Verilog. Instead it can be possible to do the same processing in Processing System in a much simplified way, by sacrifice some of the performance of the Programmable Logic. That is because the Processing System handles the tasks in a sequential way. The algorithms that are running in Processing System are mostly written in either C or C++.

Since we are using PetaLinux on top of the dual-core ARM Cortex A9, we can also take use of hyperthreading in Linux. PetaLinux also has an integrated embedded web server called  $\mu$ Web<sup>120</sup>, which can alternatively help with interfacing between the TE0720 and the tablet. Another advantage that comes by using PetaLinux, is that the Linux filesystem makes it easier to handle the storage and file structure.

The Processing System is also very flexible, which makes it possible to run dual operating systems on two separated cores. It is also possible to run PetaLinux in a single core and using bare-metal applications in the second one.<sup>121</sup> This gives us different options to choose from.

---

<sup>120</sup> Workware,  $\mu$ Web

<sup>121</sup> Multi-OS Support (AMP & Hypervisor)



### 10.3.2. <sup>122</sup>Device-tree configuration

In order to setup the Ethernet communication for the TE0720, we had to do some adjustments inside the device-tree configuration. Device tree is a simple tree structure of nodes and properties. TE0720 uses a Marvell Alaska 88E1512<sup>123</sup> Gigabit Ethernet PHY. The Ethernet physical interface is connected to the Zynq Ethernet0 PS GEM0 (MIO Pins MIO16 - MIO27). Therefore we must choose the compatible device 88E1512 to enable Ethernet connection as shown in Code 1.

#### Code 1 - Configure system-top.dts

```
&gem0 {
    phy-handle = <&phy0>;
    phy-mode = "rgmii-id";
    ps7_ethernet_0_mdio: mdio {
        phy0: phy@0 {
            compatible = "marvell,88E1512";
            device_type = "ethernet-phy";
            reg = <0>;
        };
    };
};
```

We had to configure the rootfs by changing the automatic IP address to static IP address. The reason for this change was because we wanted the IP address to be the same after every startup, also called static IP address.

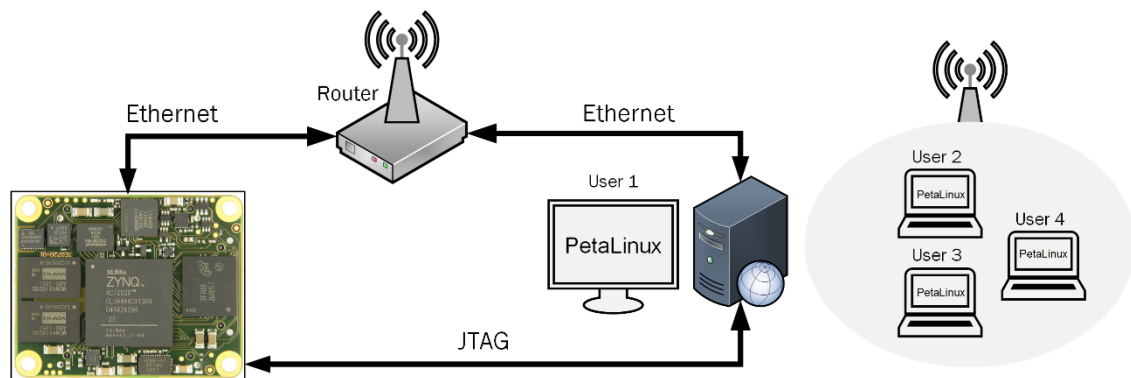


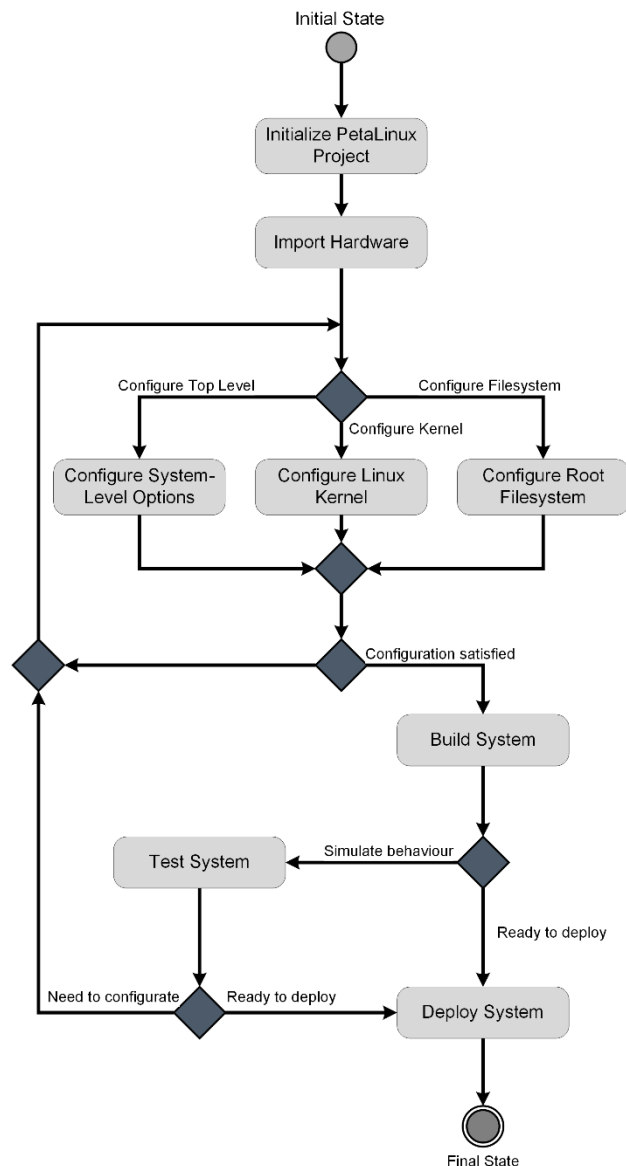
Figure 181 - Overall network communication

<sup>123</sup> 88E1512 Datasheet



### 10.3.3. Building PetaLinux

To simplify the PetaLinux workflow structure, we have gathered the most important activities into an activity diagram. This gives us an abstract overview of which state and command is necessary to build our version of PetaLinux. To create everything needed to boot up PetaLinux platform on TE0720, we are using PetaLinux tool which is downloaded from Xilinx website.<sup>124</sup> We are also using Linux host machine to develop PetaLinux, because PetaLinux tools requires several 32-bits libraries which is only found on a Linux host computer<sup>125</sup>.



**Figure 182 - PetaLinux Activity Diagram**

<sup>124</sup> Xilinx Inc, PetaLinux Tools Download

<sup>125</sup> Xilinx Inc, PetaLinux Tools Documentation



#### 10.3.4. Mount the NAND flash (eMMC) on bootup

Since we are going to store and read information from the NAND flash on the Zynq module, we have created a small startup application to run a small C code. As shown in Code 2, the C code runs five Linux commands to accomplish this. After executing the code, the NAND flash memory is then mounted to the /mnt/emmc folder in the root file system. The startup code will also copy over the libraries and code files to make the parameter logging up and running as expected. Since we are using the  $\mu$ Web server to share the database from the NAND flash memory, we are creating a symbolic link from the sensor\_data.txt file inside the NAND flash memory to the httpd folder inside /home. So each time the data is added to the sensor\_data.txt file inside NAND flash memory, the data will also be available from the /home/httpd folder. It would probably be more efficient to make a hard link instead of symbolic link from the flash memory to the httpd folder, because it would then just use a pointer to fetch the data from the original file inside flash memory. But since the PetaLinux file system and flash memory are using two different file formats and to different partitions, it is not possible to use a hard link.

##### Code 2 - C: Part of the startup code

```
strcpy( command, "mkdir /mnt/emmc; mount /dev/mmcblk1p1 /mnt/emmc;  
cp -rf /mnt/emmc/KPEC /home/httpd; ln -s /mnt/emmc/sensor_data.txt  
/home/httpd/KPEC/sensor_data.txt;  
PATH=$PATH:/mnt/emmc/websocket;" );  
system( command );
```



## 10.4. Real-time communication

We want to establish communication between the TE0720 Zynq SoC and a tablet. One of the requirements from KDA is that the tablet shall receive real-time parameters from the Zynq module. There are different protocols to establish machine to machine (M2M) communication. Most of these protocols share a common problem, which is the latency due to network overhead. The HTTP protocol, which is one of the most used protocols for network communication, has more overhead per message. There are some re-engineered HTTP protocol such as HTTP 2.0 and SPDY which will give lower latency, but instead we are using WebSocket for data transferring. WebSocket connections are bi-directional and full-duplex. After the initial handshake request/response, there is almost no per message overhead. Although the WebSocket protocol is not supported by all web browsers today, there is native support for the latest Android browsers. There are also libraries for both PhoneGap and Android Studio to establish WebSocket communication.

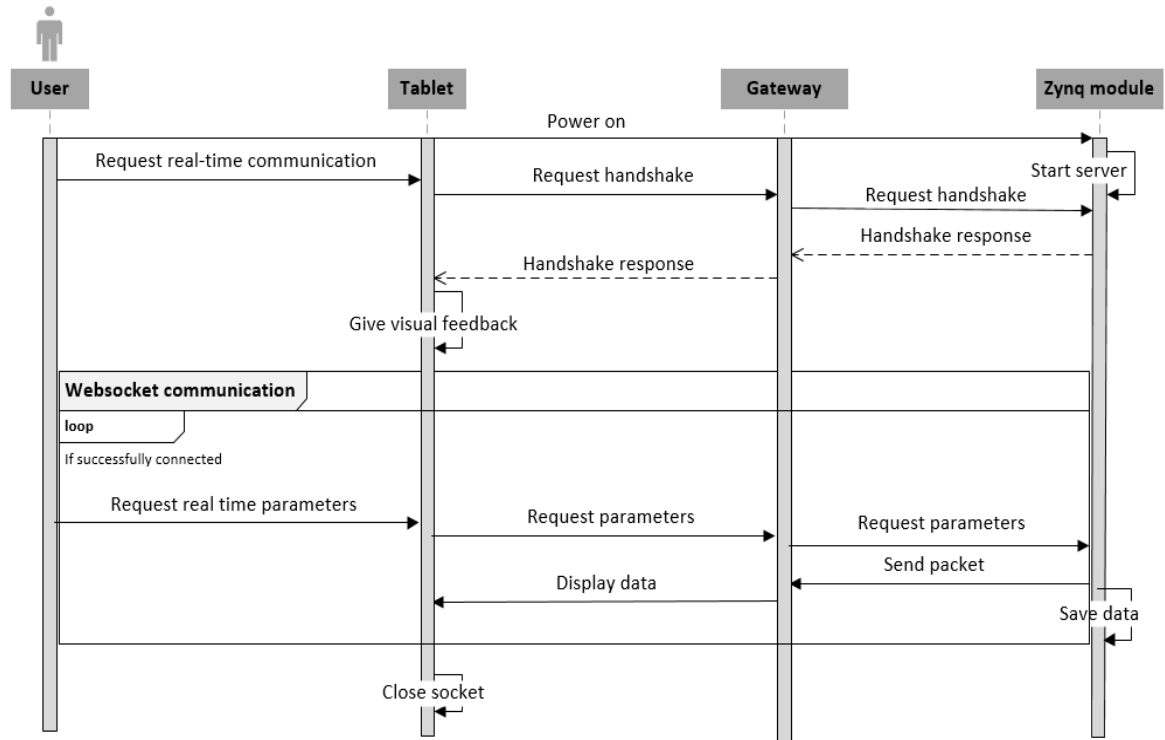


Figure 183 - Sequence diagram of WebSocket communication



#### 10.4.1. Server-side communication

We have chosen to implement the server-side of socket communication to the TE0720. The reason for that is because the TE0720 will handle connection requests coming from clients. This choice only affects the connection establishment in the beginning phase. After the connection is established, both server and client are able to send and receive data, and the server-client choice will not matter. On the server-side of the communication we are using a daemon tool (process that runs in the background) called `websocketd`<sup>126</sup>, to implement WebSockets in an easier way. The tool allows us to use socket connection between different machines, and supports a widely range of languages. Instead of caring about the frame, buffer and packets, `websocketd` will handle that for us as a background process.

To initialize the `websocketd` tool, we need to create an environmental variable for `websocketd`, as shown in Code 3. This makes it possible to use the variable `websocketd` from the terminal which is stored inside the `websocket` folder in the flash memory.

##### Code 3 - Bash: Environmental variable for `websocketd`

```
PATH=$PATH:/mnt/emmc/websocket ;
```

After the environmental setup is ready, we can then start our application from the terminal. Code 4 shows how we are doing that. The `websocketd` daemon will run in the background, while our application called `websocket_data`, will use the `websocketd` daemon to send parameters to the tablet. We are also specifying that we are using the port number 8080 for the WebSocket communication.

##### Code 4 - Bash: Run `websocketd` with our application

```
websocketd --port=8080 websocket_data
```

Since we are not dealing with real data from the sensors, we have to generate some random values from the TE0720. There are several possibilities to do that, but we are simply using the random function in C to generate the dummy parameters. Code 5 shows us how we are generating the dummy values for the temperature sensors. To make it easy for the Android application to identify the parameters, we are using the first part of the message as identification. In the case of Code 5, the identifier is called `Temp` to make it clear that the following parameters are for temperature sensors. The code also checks for error, which you can read about in chapter 10.5.

---

<sup>126</sup> `websocketd`, WebSocket communication

**Code 5 - C: Generate dummy values for temperature sensors**

```
void rand_temperature_sensor(int EXTS[8], int FTS, int CTS, int OTS){
    int i;
    for (i = 0; i < 8; i++){
        EXTS[i] = rand() % 50 + 850;
        if(EXTS[i]>EXTS_limit){
            char name[6];
            sprintf(name, "EXTS%d", i+1);
            detect_error(name, EXTS[i], ((EXTS[i]>1050) ? 2 : 1), i);
        }
    }

    FTS = rand() % 5 + 97;
    if(FTS > FTS_limit) detect_error("FTS", FTS, ((FTS>120) ? 2 : 1), 8);
    CTS = rand() % 4 + 88;
    if(CTS > CTS_limit) detect_error("CTS", CTS, ((CTS>110) ? 2 : 1), 9);
    OTS = rand() % 5 + 90;
    if(OTS > OTS_limit) detect_error("OTS", OTS, ((OTS>120) ? 2 : 1), 10);

    printf("%s,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d\n", "Temp", EXTS[1], EXTS[2], EXTS[3], EXTS[4], EXTS[5], EXTS[6], EXTS[7], EXTS[8], FTS, CTS, OTS);
}
```



## 10.5. Parameter storage

Another important feature of the ECU product is to make it possible to view a log of the parameters coming from the sensors. The idea about storing the parameters is that when the TE0720 is receiving a value that is out of the limit, it will be captured and stored with the sensor name, parameter, and the time at that point for later analysis. This feature is supposed to be implemented as a long-term goal, but we have been working on it this year to find an idea of solution for this feature.

### 10.5.1. Storage in TSV format

Since the system will store different parameters, it is important to have a structured way of storing the data. In embedded Linux environment SQLite and MySQL are widely used for storing the data in a rational way. We have tried out these two alternatives, but they were not matching our needs. MySQL uses too much resource from the processing system, and is server-based. So after storing a bunch of data, the latency of loading the content from the database will increase greatly. SQLite database is also a rational database, which is often used in embedded systems. Like MySQL it also uses SQL database engine, is serverless and is written in C language. SQLite is mostly used as a local database and not so much used as a server/client database. For instance we could use SQLite database locally for our Android application, where the parameters would be stored inside the tablet. We didn't chose this option because using this way, would make the exact tablet dependent to the system. All the data would be inside the tablet and not in TE0720. So we tried to use the SQLite as a server/client database. Since the database model is developed in C language, we had to use a code converter in JavaScript to make the database readable from the Android application. We found out early that to convert the database to JavaScript, increased the latency dramatically as the row of the database increased. After storing about 7000 rows with parameters into the SQLite database, it took about 15 minutes to view the result.

Therefore we have changed the database model into a single text file in tab-separated value (TSV) format. The reason for trying out SQL database in the first place was because we wanted to use SQL statements to fetch the data from the SQLite database. This is not possible using a pure text file in TSV format. To make that happen, we can now use a plugin in JavaScript that enables us to use SQL statements from a TSV format text file.<sup>127</sup> There are several options to store the database into, but we are using the NAND flash, which is a non-volatile flash memory, to store all the parameters. The file is stored into /mnt/emmc and is called sensor\_data.txt.

---

<sup>127</sup> Andrey Gershun, AlaSQL.js





### Code 6 - C: Structure for storing the parameters

```
struct arg_struct{
    char *sensorname;
    double sensorvalue;
    int faultcode;
    int index;
};
```

We are using struct to handle several variables when using threads. As the Code 6 shows, a complete struct will then include the sensor name, sensor value, fault code and the index number of pthread\_mutex.

Each sensor is given a variable called is\_writing[index], which is a boolean to control whether the parameter is writing to the file. This is to avoid that a busy thread gets interrupted. We also don't want to write the sensor fault several times per second. In order to avoid that, we are using a timer for each sensor called writing\_time[index], that will keep track of the last time the parameter was written to the file for each sensor. Code 7 shows how the struct is made by the sensor data. After the thread is created, we can then start write\_sens\_error function which is shown in Code 8.

### Code 7 - C: Time check and creation of thread

```
void detect_error(char *sensorname, double sensorvalue, int faultcode,
int index){
    pthread_t thread_error;
    struct arg_struct args;
    args.sensorname = malloc(7);
    strcpy(args.sensorname, sensorname);
    args.sensorvalue = sensorvalue;
    args.faultcode = faultcode;
    args.index=index;
    clock_t t=clock();

    if(is_writing[index]==0&&((float)t-(float)writingtime[index])>5000){
        is_writing[index]=1;
        writingtime[index]=clock();
        pthread_create(&thread_error, NULL, &write_sens_error, (void
        *)&args);
        pthread_join(thread_error, NULL);
    }
    free(args.sensorname);
}
```



After the thread is started, it will then start the function in Code 8. Here the file `sensor_data.txt` is opened and the arguments from the struct are then inserted into the file. The `is_writing[index]` boolean is then set to 0, so that the next sensor parameters are ready to be stored into the file after the time limit which is set in Code 7.

#### Code 8 - C: Store the data into the flash memory

```
void *write_sens_error(void *arguments){  
  
    pthread_mutex_lock(&mutex_writing);  
    struct arg_struct *args = arguments;  
    struct tm *local;  
    time_t t;  
    t = time(NULL);  
    local = localtime(&t);  
    FILE *fp;  
    fp = fopen("/mnt/emmc/sensor_data.txt", "a");  
    fprintf(fp, "%s %s %f %s %d %s %s", args->sensorname, "\t",  
    args->sensorvalue, "\t", args->faultcode, "\t", asctime(local));  
    fclose(fp);  
    is_writing[args->index]=0;  
    pthread_mutex_unlock(&mutex_writing);  
    pthread_exit(NULL);  
    return NULL;  
}
```

Since we also want to store the time to that point when the error was detected, we need to set the system clock time correctly. The carrier board we are using for software development doesn't have backup power, so the Real-Time-Clock is reset every time the TE0720 is powered off. This will however change when the real motherboard is used, which will have backup power implemented.

But for now we have to set the hwclock after each bootup. The command for doing that correctly is shown in Code 9, where `<current time>` is changed with the time in `hh:mm:ss` format and `<current date>` in `dd/mm/yyyy`.

#### Code 9 - Set up current time in PetaLinux

```
date -s '<current date> <current time>'
```



### Code 10 - JavaScript: View the content from sensor\_data.txt

```
<script>var statement;</script>
<button id="where Faultcode = 1" class="btn btn-default btn-lg"
onclick="button_click(this.id)">Only warnings</button>
<button id="where Faultcode = 2" class="btn btn-default btn-lg"
onclick="button_click(this.id)">Only faults</button>
<button id="" class="btn btn-default btn-lg" onclick="button_click(this.id)">
View all</button>

<script>
function button_click(clicked_id){
    statement = clicked_id;
    alasql('SELECT * FROM
TSV("sensor_data.txt",{headers:true})',[],function(res){
    var s = "<table id='res' class='table table-striped table-bordered'
cellspacing='0'
width='100%'><thead><th>Sensorname<th>Sensorvalue<th>Time<th>Faultcode<tbody>";
    for(var i=0;i<res.length;i++) {
        s += '<tr>';
        s += '<td>'+res[i].Sensorname;
        s += '<td>';
        s += '>'+res[i].Sensorvalue;
        s += '<td>';
        s += '>'+res[i].Time;
        s += '<td>';
        s += '>'+res[i].Faultcode;
    }
    s += '</table>';
    document.getElementById('res').innerHTML = s;
});
</script>
```

To show the content from sensor\_data.txt, we are using a combination of JavaScript and HTML. The library alasql allows us to use SQL statements on a regular text file. We can therefore separate the data inside the text file, such as display only the data from the last hour, or display only the faults and not the warnings et cetera. Code 10 shows how the three different buttons adds three different SQL statements. We can for instance press the “Only faults” button, which will add ‘where Faultcode = 2’ to the SQL statement, and only output the faults.

sensor_data.txt
Sensor name
Sensor value
Time
Faultcode

Figure 185 - Log storage structure

FTS	121.000000	2	Mon May 11 17:06:35 2015
CTS	151.000000	2	Mon May 11 17:06:35 2015
FTS	121.000000	2	Mon May 11 17:06:39 2015
CTS	151.000000	2	Mon May 11 17:06:39 2015
KS1	20514.000000	1	Mon May 11 17:06:40 2015
KS2	20597.000000	1	Mon May 11 17:06:40 2015
FTS	121.000000	2	Mon May 11 17:06:43 2015
CTS	151.000000	2	Mon May 11 17:06:44 2015
KS1	20529.000000	1	Mon May 11 17:06:48 2015
KS2	20584.000000	1	Mon May 11 17:06:48 2015
FTS	121.000000	2	Mon May 11 17:06:48 2015
CTS	151.000000	2	Mon May 11 17:06:48 2015
FTS	121.000000	2	Mon May 11 17:06:53 2015
CTS	151.000000	2	Mon May 11 17:06:53 2015

Figure 184 - Example of parameter storage



## 10.6. Makefile

Since we are not developing on the same architecture as the ARM architecture, we have to do cross compiling, from the host Linux machine, to get an executable file for the TE0720. The target architecture is the ARM processor inside the TE0720. We will therefore include arm-xilinx-linux-gnueabi-gcc, so the file which is created after make will be executable on the TE0720.<sup>128</sup> The name of the application will then be websocket\_data.

### Code 11 - Makefile: Create executable application

```
SOURCE=websocket_data.c
CC=arm-xilinx-linux-gnueabi-gcc
LDFLAGS=-lpthread
PROGRAM=websocket_data

all: $(PROGRAM)

$(PROGRAM): $(SOURCE)
    $(CC) $(SOURCE) -o $(PROGRAM) $(LDFLAGS)

clean:
    rm $(PROGRAM)
```

To make it possible to cross compile for Xilinx ARM processors, we also need to source into the settings in the Vivado folder before we are going to make the executable application.

### Code 12 - Source to settings64.sh

```
source /opt/Xilinx/Vivado/2014.4/settings64.sh
```

---

<sup>128</sup> Xilinx, Cross-compile to ARM



## 11. Android application design

While our Zynq board will produce high amount of data from the engine control unit, we need a client that will receive the needed data and then analyse and review these parameters from the Zynq. That is why we need to create an Android application for that purpose. The application will have support for real-time view where the user of the application can view parameters generated from TE0720 in real time right from the tablet.

The long-term goal of the application is to allow the user to take control of the engine performance and view each sensor information individually. The application is a vital part of this project because it is the selling key. Therefore the application must be interactive, functional and simple to understand. Figure 186 shows the overall structure of the software system. It shows the connection between the TE0720 and the Android application.

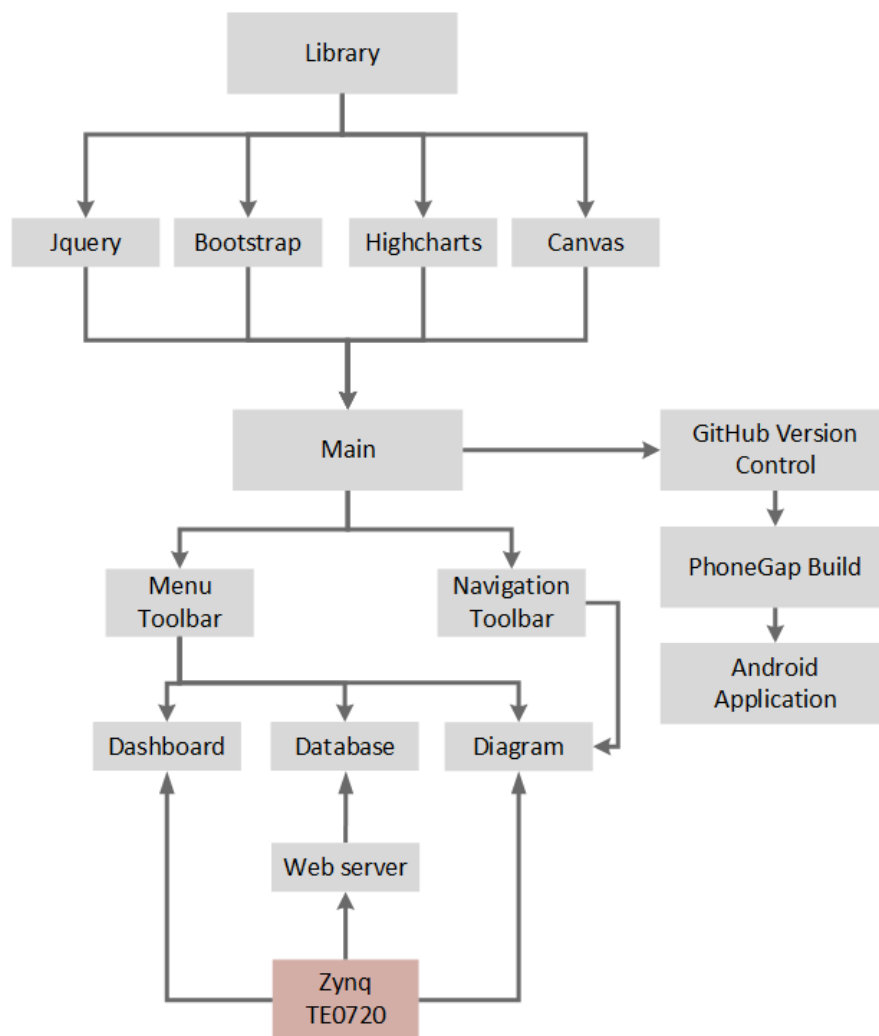
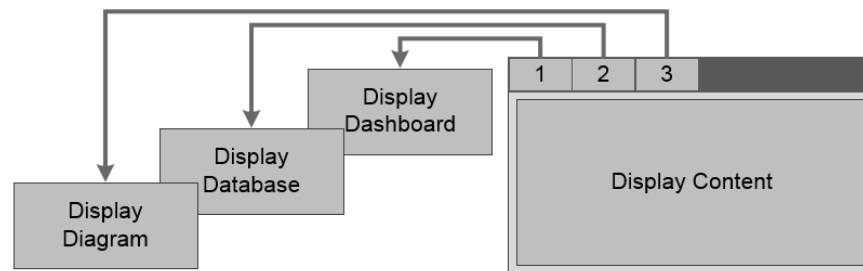


Figure 186 - Overall structure



The code structure of the application is made in a way so future developers can start right on the code and further develop the application. We decided to use PhoneGap because it is a free and open source framework that allows creating mobile apps using standardized web APIs for multiple platforms (7 different types). The platform that we are using to test the application is on android, due to that in this project we are using a Samsung Galaxy Tab S which is one of the latest high-end Android tablets. The final application will be accessible to hundreds of millions of users across a wide range of devices from phone to tables and platforms.



**Figure 187 - Application structure**

### 11.1. PhoneGap

We are using PhoneGap<sup>129</sup> to build our Android application. PhoneGap is an open-source framework for cross-platform development by using HTML, CSS and JavaScript. A PhoneGap project contains four main functional parts to ensure the implementation of the HTML and JavaScript code:

- A native browser embedded in the project.
- Structured API so we can access the native functionality for a mobile device from the code.
- API for writing native plugins.
- A file storage, which contains HTML, JavaScript, CSS and other resources of the application.

The huge benefit of PhoneGap, is that the application can be directly tested on different browsers on the host computer such as Google Chrome, Safari, Internet Explorer et cetera. This saves us a lot of time instead of loading the code into PhoneGap cloud every time we make a small change.

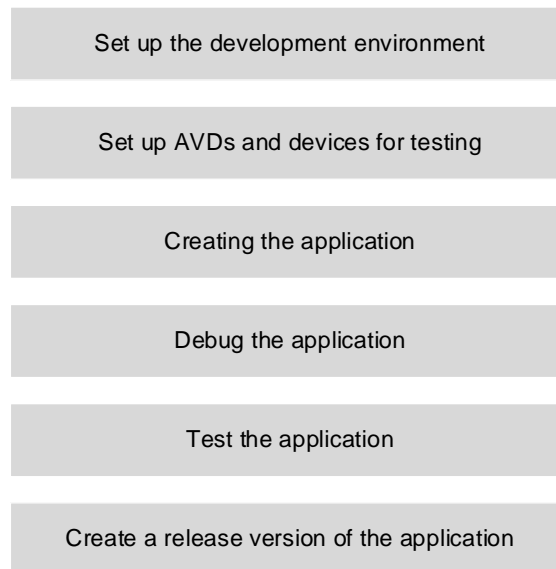
---

<sup>129</sup> PhoneGap, Hybrid application



## 11.2. Application development workflow

There are many different ways to develop applications for portable devices today. Since we are developing an application for an Android tablet, we had to configure the host machine to make it possible to develop application for Android based devices. The main development steps for developing an Android application using PhoneGap are shown in Figure 188.



**Figure 188 - Development process for Android**

### 11.2.1. Setup the development environment

The first step for developing an Android application is to configure the host machine we are going to use for developing the application. We are using Linux and Windows platform to develop the Android application. To configure the development environment we need to install the Android Software Development Kit (SDK). The Android SDK includes several necessary tools to build our Android application, such as SDK Platform, Google API, system image, and Android SDK tools.

The target tablet we are developing for is a Samsung Galaxy Tab S, which is currently running on 4.4.2 version of Android KitKat. The current latest API for Android development is version 22. By developing application from the API version 22, the application will only support a few tablets at this point because it requires that the Android version on the tablet is upgraded to Android 5.1.1 from 4.4.2. So instead of developing from API 22, we are using version API 19 so all the functions Android SDK is providing, supports the tablet we are targeting to. As Figure 189 shows us, we have installed all the necessary tools for API 19. For further development of this application, it will be necessary to upgrade the API to the latest version, and change the



target device in the Android Manifest and build gradle which you can read about in chapter 11.3.6.

To make it possible to run our application on the tablet, we had to enable the tablet to debugable mode. There is an option called USB debugging in developer options in settings on the Android tablet.

Tools			
<input type="checkbox"/>	Android SDK Tools	24.1.2	Installed
<input type="checkbox"/>	Android SDK Platform-tools	22	Installed
<input type="checkbox"/>	Android SDK Build-tools	22.0.1	Installed
<input type="checkbox"/>	Android SDK Build-tools	21.1.2	Not installed
<input type="checkbox"/>	Android SDK Build-tools	20	Not installed
<input type="checkbox"/>	Android SDK Build-tools	19.1	Installed
<input type="checkbox"/>	Android 5.1.1 (API 22)		
<input type="checkbox"/>	Android 5.0.1 (API 21)		
<input type="checkbox"/>	Android 4.4W.2 (API 20)		
<input type="checkbox"/>	Android 4.4.2 (API 19)		
<input type="checkbox"/>	SDK Platform	19	4 Installed
<input type="checkbox"/>	Samples for SDK	19	6 Installed
<input type="checkbox"/>	ARM EABI v7a System Image	19	2 Installed
<input type="checkbox"/>	Intel x86 Atom System Image	19	2 Installed
<input type="checkbox"/>	Google APIs (x86 System Image)	19	13 Installed
<input type="checkbox"/>	Google APIs (ARM System Image)	19	13 Installed
<input type="checkbox"/>	Glass Development Kit Preview	19	11 Not installed
<input type="checkbox"/>	Sources for Android SDK	19	2 Installed
<input type="checkbox"/>	Android 4.3.1 (API 18)		
<input type="checkbox"/>	Android 4.2.2 (API 17)		
<input type="checkbox"/>	Android 4.1.2 (API 16)		
<input type="checkbox"/>	Android 4.0.3 (API 15)		
<input type="checkbox"/>	Android 2.3.3 (API 10)		
<input type="checkbox"/>	Android 2.2 (API 8)		
<input type="checkbox"/>	Extras		
<input type="checkbox"/>	Android Support Repository	14	Not installed
<input type="checkbox"/>	Android Support Library	22.1.1	Installed
<input type="checkbox"/>	Google Play services	24	Not installed
<input type="checkbox"/>	Google Repository	17	Not installed
<input type="checkbox"/>	Google Play APK Expansion Library	3	Not installed
<input type="checkbox"/>	Google Play Billing Library	5	Not installed
<input type="checkbox"/>	Google Play Licensing Library	2	Not installed
<input type="checkbox"/>	Android Auto API Simulators	1	Not installed
<input type="checkbox"/>	Google USB Driver	11	Not compatible with Linux
<input type="checkbox"/>	Google Web Driver	2	Installed
<input type="checkbox"/>	Intel x86 Emulator Accelerator (HAXM in	5.3	Not compatible with Linux

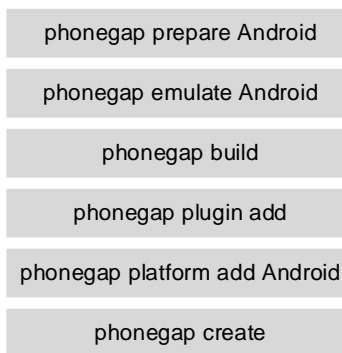
Figure 189 - Android SDK setup





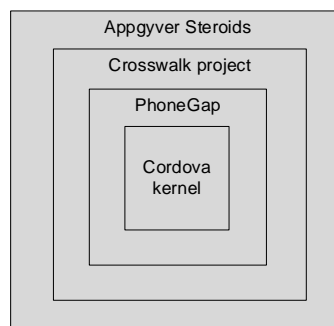
### 11.2.2. Creating the application

PhoneGap project can either be built from Cordova or PhoneGap commands. PhoneGap is a distribution of Apache Cordova, so they use very similar commands to generate and build applications. We are only using PhoneGap commands, to avoid potential conflicts between different commands. If we for instance use the create command for PhoneGap to create our application and use build command for Cordova to build the application, it will not build the project correctly. Therefore we are only using PhoneGap commands, and the most used commands are shown in, Figure 190.<sup>130</sup> But as Figure 191 shows, the PhoneGap application is based on the Cordova kernel. So when running PhoneGap commands in the terminal, the build log will show a few Cordova instances.



**Figure 190 - Most important PhoneGap commands**

When creating a PhoneGap application that is targeting the Android version 4.4.2 (API 19), it uses the Chrome render instead of the stock webkit, which is used on the tablets that are running Android version below 4.4. Therefore we are also implementing Crosswalk-project framework, so the tablets that runs on older version of Android, also can run our application with Chrome rendering. This decreases the animation rendering time significantly on older devices.



**Figure 191 - Android application stack**

<sup>130</sup> PhoneGap, Commands

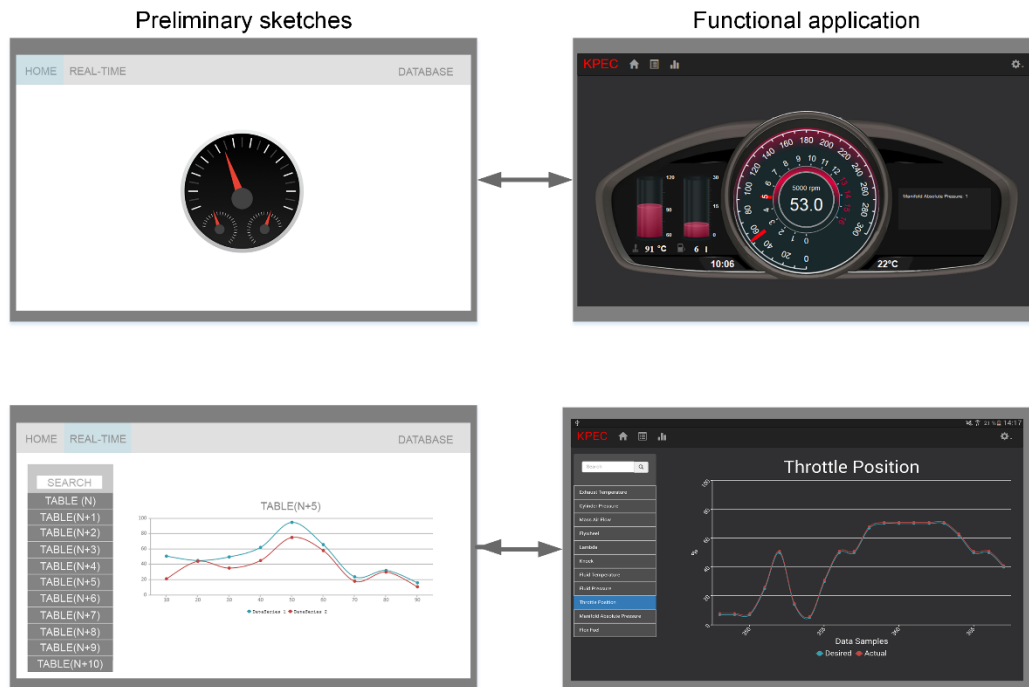


### 11.2.2.1. Design & layout

Before we started to develop the tablet application, we made some drawings on paper. We have chosen to use a dark theme, because we believe that it is a part of the racing feeling that will target the desired stakeholders. Before we developed the application, we also made preliminary sketches to define the application content and flow structure. The goal is to build an interactive and easy-to-understand application.

The application is therefore divided into two main parts, the dashboard and the 12 different diagrams. Both the dashboard and the diagrams receives data from TE0720 in real-time. The dashboard is centred on the layout to give the user a nice overview. Dashboard consists of different parameters that are sent from TE0720 to illustrate engine performance. Dashboard is designed in a sporty way to capture the stakeholders' interest, and to give the user a feeling that the Android application has something to do with motorsport. The diagram runs in real-time and the user has the opportunity to navigate between them. Each diagram is also functional so the user can decide which sensor to view.

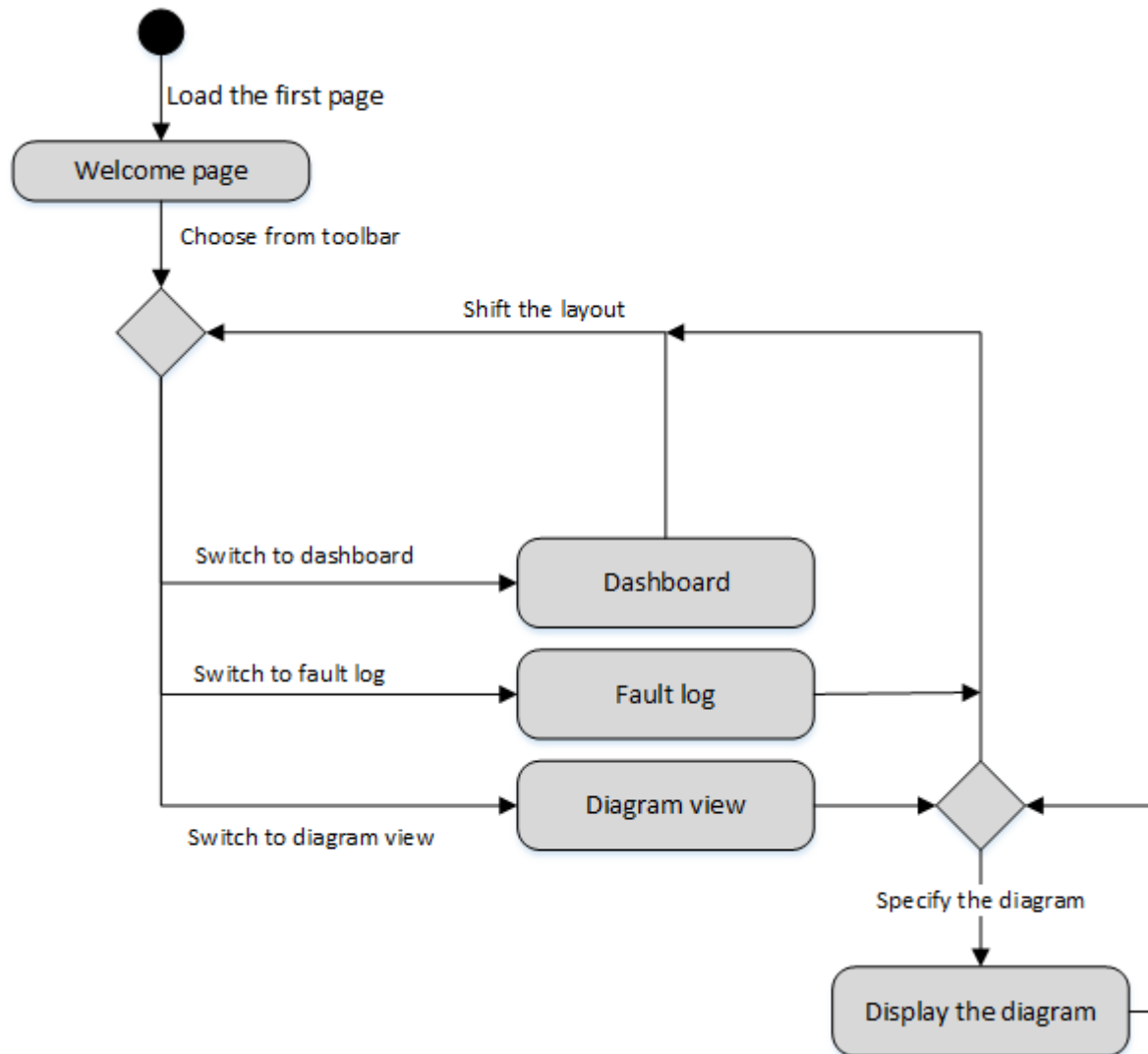
Figure 192 below shows the comparison between the preliminary sketches and the final application.



**Figure 192 - From sketch to functional application**



The activity diagram in Figure 193 shows our application overflow. From the starting node, we go to a welcome page. At that point, only the top toolbar is shown where it is possible to choose between the dashboard, fault log and diagram view. When choosing the diagram view, the navigational drawer will show up at the left side. It is then possible to specify which diagram is to be displayed. The user can switch to another layout at any point of the application run time.



**Figure 193 - Activity diagram of the application use**



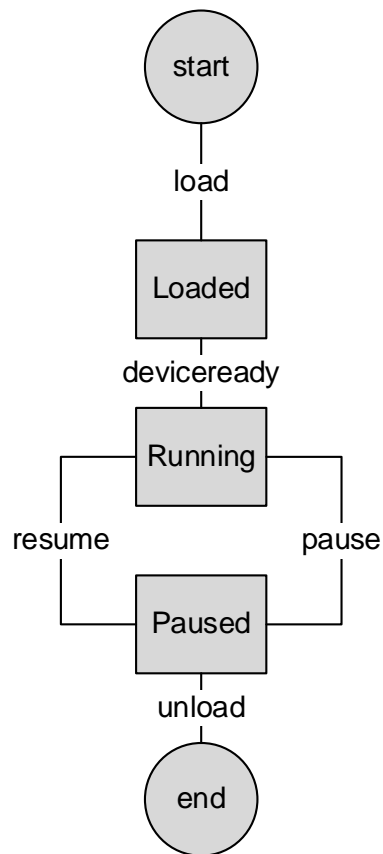
### 11.2.3. Run, debug and test the application

During the development process, we have to run and debug the application many times. PhoneGap makes it easy to create an Android application file (apk) from their cloud based website. For small changes we are using the regular web browser on our computers, but we are debugging bigger changes on our target devices by creating the application and open it inside the tablet.

Another good thing with PhoneGap is that we don't need to update the build.gradle each time we want to make a new version of the application. Each time we have updated the config.xml file, we are copying the file into the www folder. Then we have to run the command `phonegap build` so the application is aware of the changes in config.xml. This can for instance be after installing a new plugin into PhoneGap, or changing the application framework. Phonegap will automatically update the AndroidManifest and build.gradle files from the config.xml. You can read more about these configuration files in chapter 11.3.6.

The tablet application always runs in a specified activity, dependent of what the tablet is showing on the screen. Every time the application is started, it has to load the content to make the application ready for use. It will start from the activity `onDOMContentLoaded` where the initial page is loaded and parsed. It will then go to the next activity, which is the `onNativeReady`. In this activity, all the native functionality like camera, vibration and network are set up. After getting the native functionality ready, it is time to create objects for all cordova JavaScripts. When all these activities are finished, we will go to `onDeviceReady` activity, which indicates that the application is ready for use. The application will then run as normal, and the user can start interact with the application.

When the application is minimized by the system, the application will go to `onPause` activity. This is to prevent that the diagrams and dashboard are still rendering, while they are inactive. The next time we start the application, it will go to `onResume` activity, to reload the previous content of the application. In the end the user can also terminate the task by exiting the application. The application will then go to `unload` or `onDestroy` activity. In most cases, we are working with `deviceready`-, `pause`- and `resume` activities, which are the most important activities to make the application work in a stable way.



**Figure 194 - Activity diagram for our Android application**

In most programming languages, applications are started from a `main()` method. But when it comes to Android, the application is started from an activity instance by invoking specific phases of its lifecycle. To initialize our Android application in the very beginning, we are running `onCreate` activity from Java. In our application, the `onCreate` activity phase loads into `onStart` activity in PhoneGap, which will again load into `index.html`.

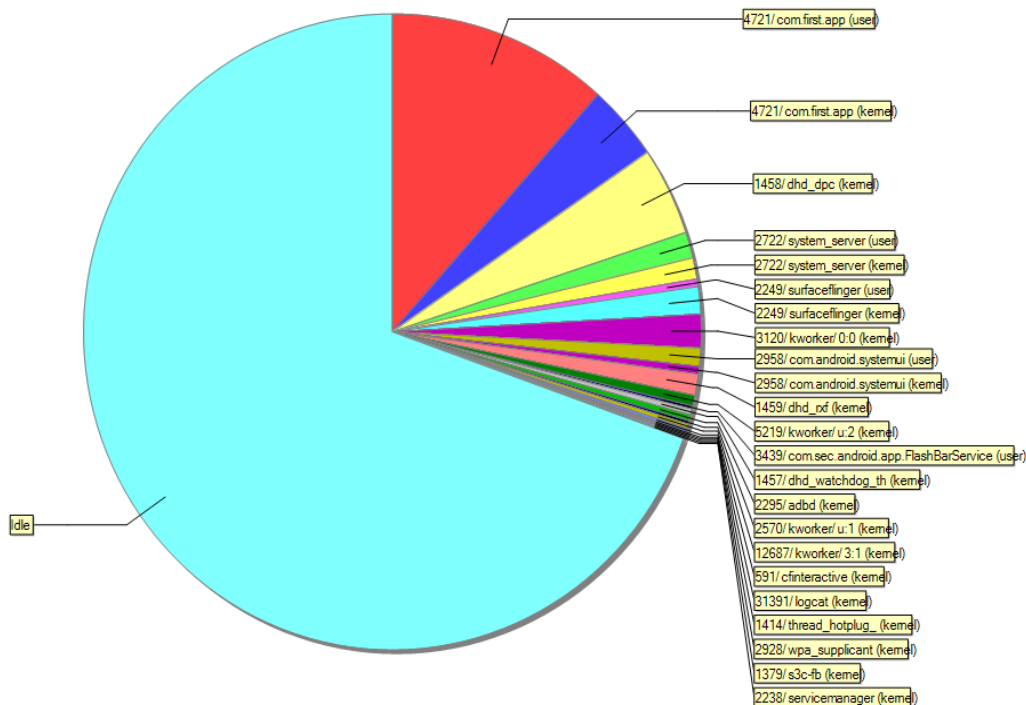
**Code 13 - Java: CordovaApp.java**

```
public class CordovaApp extends CordovaActivity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        super.init();
        // Set by <content src="index.html" /> in config.xml
        loadUrl(launchUrl);
    }
}
```



We are using Dalvik Debug Monitor Server (DDMS)<sup>131</sup> to keep track of the resources our application is using while running on the tablet. DDMS is a debugging tool, that is a part of the Android SDK, is a useful tool to get real-time log about network resource, threads, processes, logcat, CPU usage and much more. This is a very useful tool because DDMS can find weaknesses that are not easy to find while running the application normally. Since the target platform is a portable device, it is important that we are debugging properly, so the application doesn't overload the CPU, memory or other resources.

Because our tablet application involves networking and animation rendering, it is important to keep track of the CPU usage to minimize the battery consumption. After implementing all the functions into our application, we are running DDMS to find out the CPU usage of the application in different states. In Figure 195, we can see that our application (com.first.app) is using some CPU resource while running the application in normal mode. This is while the application is receiving parameters from the Zynq module, and while the dashboard is presented on the tablet. In that point of time the kernel usage of the application was about 3% while the user usage of the application was about 16%. This is a reasonable amount of CPU usage compared with other installed application.

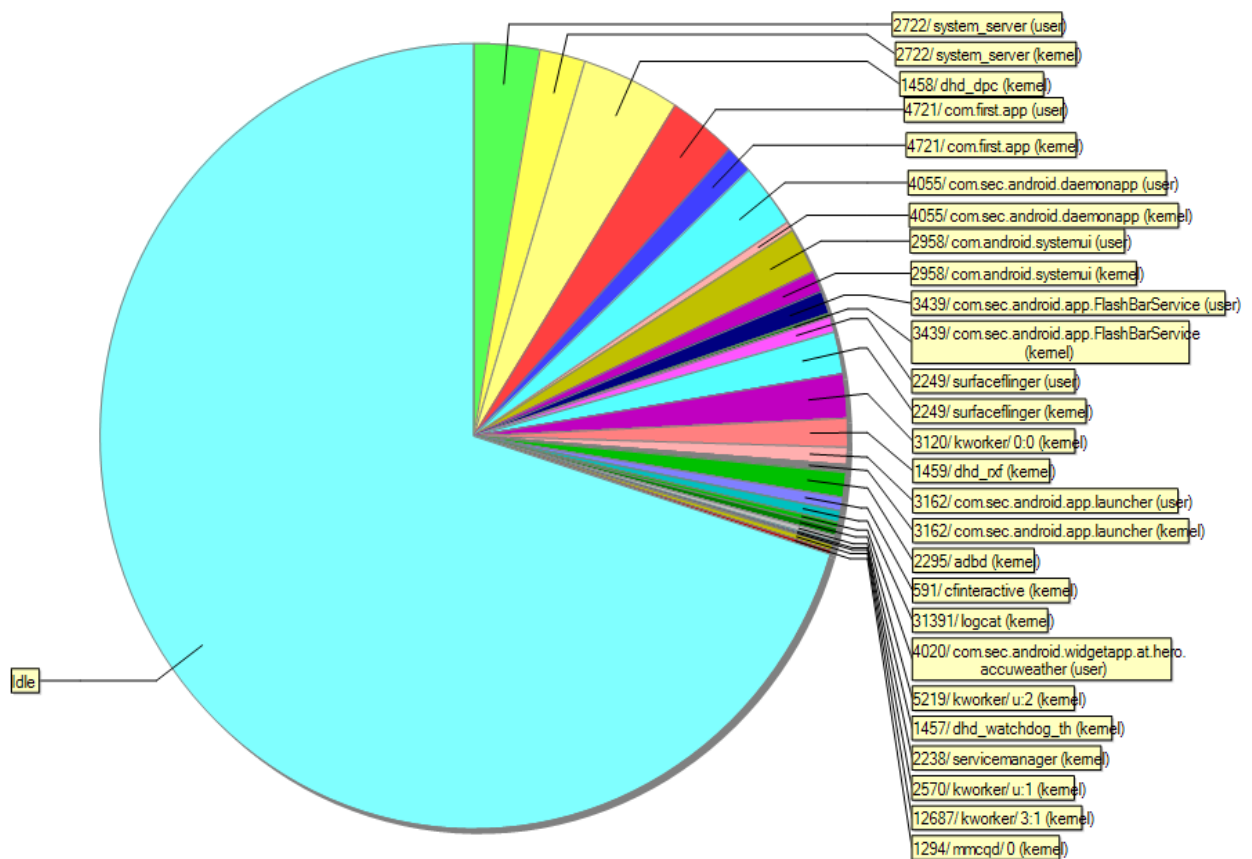


**Figure 195 - CPU usage while fully running the application**

131 Android, DDMS



We also want the application to have a low power consumption while the application is inactive. If the application is minimized after being connected to the Zynq, the application will remain connected with the Zynq, but the rendering of the graphs and the dashboard will be set to idle. This is to prevent that the user of the application has to connect to the Zynq again, when the application is resumed again. Figure 196 shows that the application (com.first.app) uses less CPU resource when it is running in minimized mode than in normal mode. The CPU usage in a certain point of time is about 3% from the user and about 1% from the application kernel.



**Figure 196 - CPU usage while minimized with Zynq connection**



When the socket gets unconnected between the tablet and the Zynq module, while the application is set to minimized mode (onPause), Figure 197 shows that the CPU usage will reduce. In a certain point of time the CPU usage was calculated to be 1% from the user and 0% from the kernel.

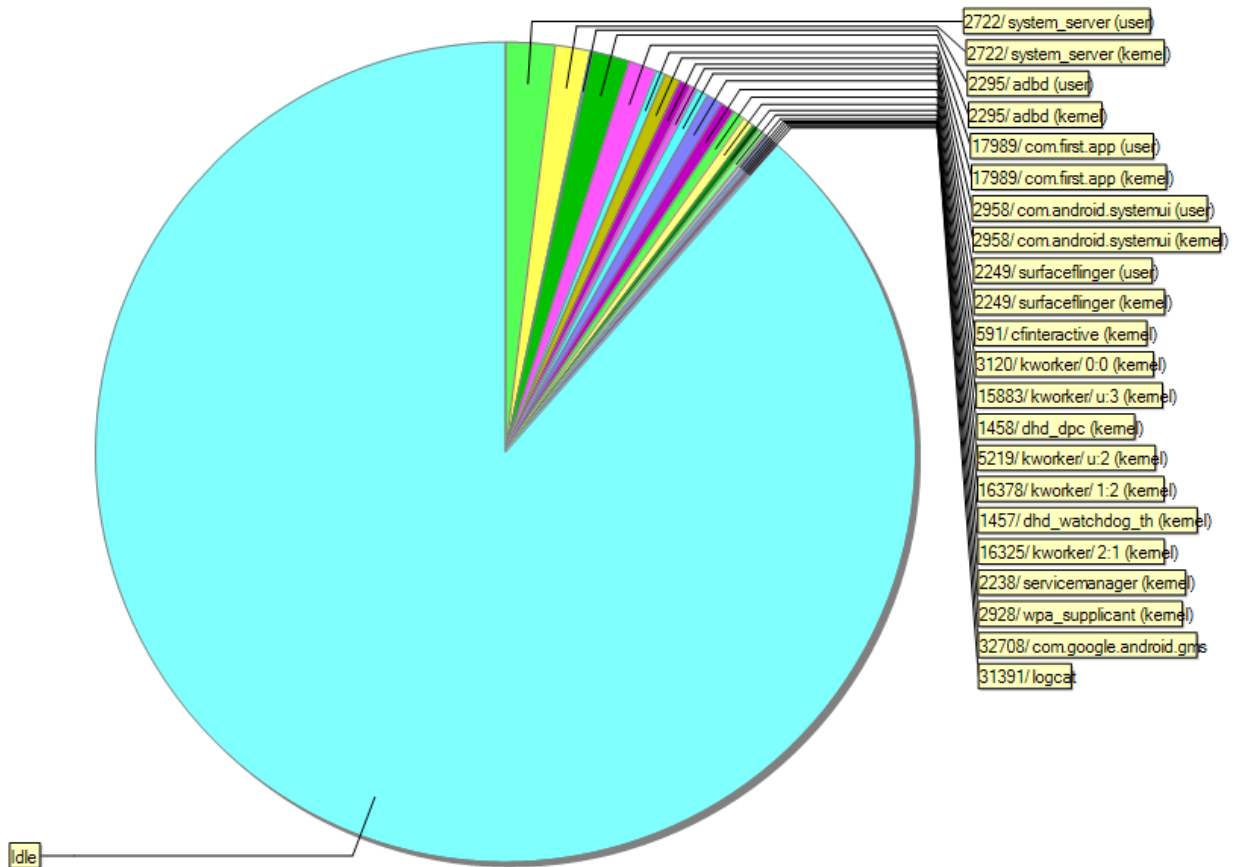


Figure 197 - CPU usage while minimized and unconnected with Zynq





#### 11.2.4. Create a release version of the application

After debugging and testing, we can make a release version of the application. The difference between the debuggable and release version of the application is that, the debuggable application makes it possible to use DDMS and TraceView to find errors and performance issues. The debuggable application also uses more battery resource than a release version. When converting to release version we have to remove the debugging and logging settings and then modify the `AndroidManifest.XML` and `build.gradle`. We also have to create a signed key to make it a valid release application. Since we are using PhoneGap cloud, it is quite easy to generate a key for the application. We just need to upload the keystore file, that is generated using Android SDK<sup>132</sup>, and then use that as a signed key when uploading to PhoneGap build cloud. After creating an apk file of the release model, we can install the application on other devices and also upload it to Play Store.

---

<sup>132</sup> Android, Creating own keystore



### 11.3. Application code structure

We have developed the application code in a flexible and editable way, which makes it easy for further developers to start where we left. Index.html is the main file and is responsible of holding the libraries such as jQuery, Bootstrap, highcharts and canvas. The main file is also connecting dashboard.html, navigationtoolbar.html and database.html together to make them navigable. We started with using href links to switch between the three navigable pages. The transition between the pages took about 4-5 seconds when we tested the application on the tablet. A href link is basically a URL that contains the point to the desired destination. Using href links between the layouts was a problem, because the application was not feeling as a regular native application. So we decided to use jQuery event handling that now allows us to load the page less than 0.1 seconds. This was a vast improvement, which increased the application performance and the native feeling.

In the beginning, we made 12 code files which were identical. Here we decided which table was going to be shown or hidden individually for each diagram. This made the code unstructured and decreased the app performance because of that. The structure was also against the code law DRY, don't repeat yourself. So to prevent this, with few adjustments, we gave each button diagram a unique ID, and then implemented jQuery with event handling to fetch each diagram individually. We then replaced the 12 copies, with Code 14. The benefit of the solution now is that user can navigate through the 12 different diagrams faster and the developers can add or delete diagrams easier.

#### Code 14 - jQuery event handling

```
<script>
$(function(){
  $('[data-row]').on('click', function() {
    var row = $(this).attr('data-row');
    $('.active').removeClass('active');
    $('#table' + row).addClass('active');
  });
});
</script>
```

The library that is implemented is based on three principals. The first principle is the speed and functionality. Here we have implemented jQuery because it makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy to use API that works across multiple browsers. The second principle is the design and layout of the application. Here we have implemented Bootstrap because it makes it possible to build a front-end framework for developing responsive applications with powerful design. The third principle is divided into two



parts, highcharts.js and canvas.js. We used highcharts library to build the dashboard, and then canvas library to build the 12 different tables.

### 11.3.1. Navigation

In order to make the application interactive and toggleable, navigation is an important part of the user experience. We decided to implement two types of navigations because we want the user to control and browse the application freely.

#### 11.3.1.1. Navigation menu

The navigation menu allows the user to navigate through different pages such as dashboard, database log and real-time statistics. The menu is fetched from Bootstrap as a component and has been modified for our application. We have made the navigation bar modifiable and flexible, so future software developers can add/delete pages more easily.

The adjustments that were made to the navigation menu:

- Replaced text with icons, because text can be a language barrier.
- Removed the search bar, because we have few pages to navigate between.
- Changed from white- to black theme.
- Added the project logo.

#### 11.3.1.2. Navigation toolbar

The navigation toolbar allows the user to navigate through 12 different diagrams. There is also possible to search for the diagrams. Navigating this way makes the application more interactive and structured, because only one diagram can be shown at a time.

The toolbar is static, and positioned on the left side of the screen, to create space for the diagrams that are displayed, while navigating. The toolbar is also a Bootstrap component that has been modified for our application.

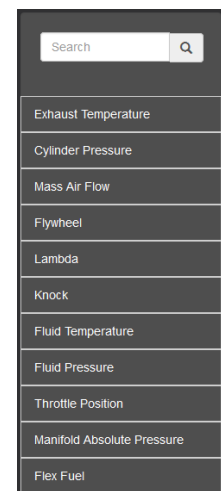


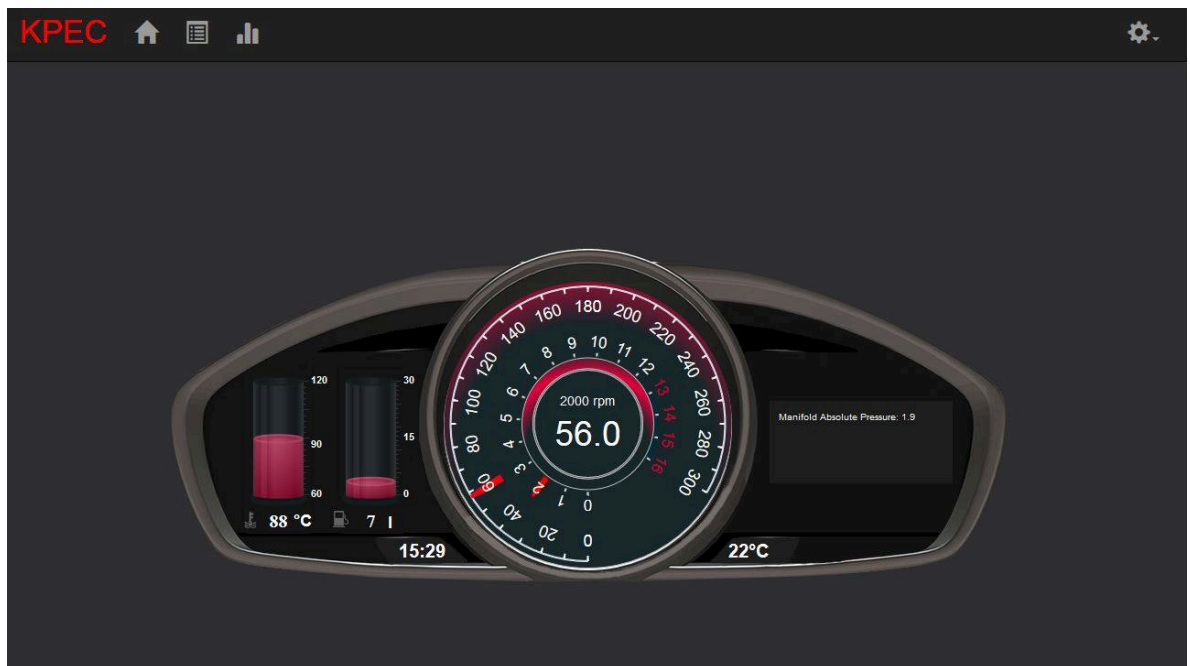
Figure 198 - Navigation toolbar



### 11.3.2. Dashboard

To visualize the most common parameters from sensors, we have implemented a dashboard view into our Android application. These parameters are normally found in a car dashboard today. The dashboard is based on five different main components.

- Speedometer for speed in km/h and rpm view
- Gage view of fuel content
- Gage view of coolant temperature
- Display area
- Clock view



**Figure 199 - Screenshot of the dashboard design**

The first component is the speedometer which is divided into two components. The inner component shows the speed and rpm in a digital format, which is also present in many car dashboards today. Instead of reading the speed or rpm from the analog gage, the exact value can be retrieved from the digital display. Additionally there is the frame around the inner component, which also shows the speed and rpm, but in analog format.

The second and third component is the gage window for fuel content and coolant temperature, which is quite similar. Normally the coolant temperature in a car is about 80 - 95 °C. The analog temperature gage is scaled for temperatures in the range of 60 - 120 °C, temperatures outside the scale will only be visual in the digital display. There is



also a digital format below the gage to show the exact temperature. To the right side of the coolant temperature display, we have the gage window for the fuel content. In a typical racing car the fuel tank holds around 30 litres, therefore the cylinder limit is set from 0 - 30 litres. Here also there is a digital format below the cylinder view. Both the speedometer and the two cylinder views are created from JavaScript and HTML. The display limit for these components is easy to change, which is a benefit because the limit can depend from a car to another.

The fourth component is the display area (on the right side), which can display parameters from other sensors or additional warnings. It can for instance display a critical warning if a sensor is not responding as expected. The last component is the clock view, which shows the system clock of the tablet. On some tablets, the system toolbar on top of the screen is hidden, when an application is running. Therefore displaying a system clock on the dashboard is a nice to have feature.



Code 15 shows a part of the code that is used to create the cylinder view for the coolant temperature. The design of the cylinder can easily be adjusted from this JavaScript code. The data format is in json, so after receiving the parameter from the WebSocket, we need to convert it to json format before making it available for the cylinder function. The variable name CTS is a global variable coming from a JavaScript file called connect\_socket.js, which can be read about later in chapter 11.3.4.

#### Code 15 - JavaScript: Cylinder view for coolant temperature

```
animation: {
    duration: 80
},
type: 'cylinder',
renderAt: 'chart-container1',
id: 'fluid_temperature_coolant',
width: '70',
height: '180',
dataFormat: 'json',
dataSource: {
    "chart": {
        "caption": "",
        "subcaption": "",
        "subcaptionFontBold": "1",
        "lowerLimit": "60",
        "upperLimit": "120",
        "numberSuffix": "",
        "bgColor": "#000000",
        "showBorder": "0",
        "thmFillColor": "#993333",
        "BgAlpha": "91.9",
        "showTickMarks": "1",
        "showTickValues": "0",
        "baseFontColor": "FFFFFF"
    },
    "value": CTS },
    "events": {
        "rendered" : function (evtObj, argObj){
            var intervalVar = setInterval(function () {
                FusionCharts.items["fluid_temperature_water"].feedData("&value="+CTS);
            }, 0);
        }
    }
})
.render();
});
```

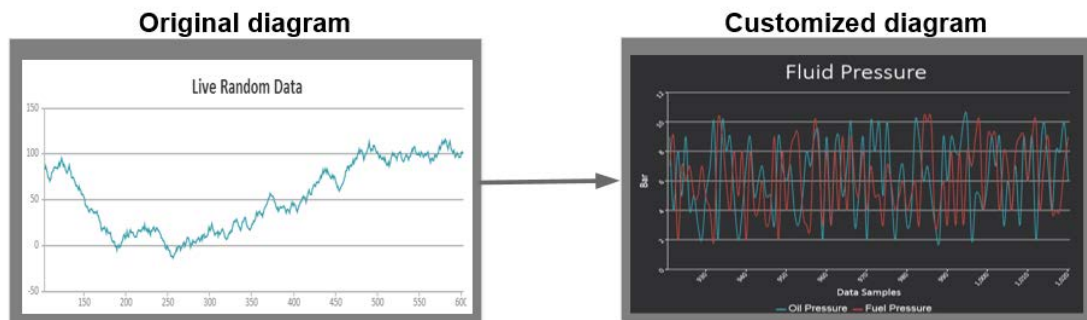


### 11.3.3. Diagrams

We are interfacing 12 different tables that are receiving parameters in real-time. The problem we have been working on during the development of the Android application is that PhoneGap does not use the native hardware acceleration by standard in every function. This means that the application performance is decreased dramatically, because the diagrams are running at high animation rendering on the tablet.

We managed to solve this problem by using canvas.js to display the diagrams instead of highchart.js, because canvas.js require less hardware effects than highchart.js, which increases the application performance. The solution was based on speed performance rather than fanciful diagrams, because we wanted to satisfy our real-time requirements.

After we selected canvas.js to show the diagrams, we made an efficient algorithm that is based on jQuery library to increase the application performance. The reason we went for jQuery is because it takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that can be called with a few lines as seen in Code 14.



**Figure 200 - CanvasJS customized design**



### 11.3.3.1. Canvas diagrams

Canvas is basically a container for graphics, and we combine JavaScript with Canvas to draw the dynamic graphics to display the diagrams. By using canvas.js, we had to modify many things internal in canvas.js library such as text, colour and body position, to make it viewable for the user. The diagrams are positioned on the right side of the screen to create a relationship between the component navigation toolbar. Most of the components are put close to each other to create short and comfortable eye movements.

In order make the code flexible and efficient, we manage to implement the same code structure individually for each diagram, with some manual adjustments. We will take the knock diagram as an example and explain the code segments briefly.

#### Code 16 - JQuery: Window onLoad

```
$(window).on("load", function() {  
    var dps1 = []; // Knock Bank 1  
    var dps2 = []; // Knock Bank 2  
    // More code goes here..  
});
```

The window load is a jQuery event that executes when the whole complete page is fully loaded, including all frames, objects and drawings. This includes a fast respond time that satisfies our user, and solved the 4-5 seconds load delay that we specified in chapter 11.3. The declared variable called dps inside the jQuery event, stores data into an array.

#### Code 17 - Diagram Design

```
var chart = new CanvasJS.Chart("knock",  
{  
    // More code goes here..  
});
```

This is an important function that gives us the possibility to design the framework of the diagram individually. The function in Code 17 is where we declares each component that is displayed such as text position, interactive functionalities and let the user manually select which line to view. Instead of putting all the diagrams inside one place and make it unstructured, we have placed each diagram code into separated files.





### Code 18 - Update Diagram

```
var updateChart = function (count) {  
  // More code goes here..  
    chart.render();  
  };  
  
  // generates first set of dataPoints  
  updateChart(dataLength);  
  
  // update chart after specified time.  
  setInterval(function(){updateChart()}, updateInterval);
```

The function `updateChart` is responsible for updating the diagram and make it dynamic. When the data is being loaded into the diagram, it slowed down the application performance. To prevent the issue, we have implemented `chart.render()` method. This method converts the lag into continuous data transitions that allow more comfortable view for the user.

Instead of fetching massive amount of data that can slow down the application performance, we choose the data length by setting this as parameter inside `updateChart` method. `setInterval` method is responsible for updating the chart after a specific time. These methods combined all together, creates a dynamic diagram that moves in real-time.



#### 11.3.4. WebSocket connection inside application

Since we are using WebSockets to communicate between TE0720 and the tablet, we have to make a WebSocket connection every time the application starts. Code 19 shows a part of the code inside the `connect_socket.js` file. The file is sourced inside the `index.html` page, and since the PhoneGap application is set to load the content from the `index.html` first when the application starts, the code inside `connect_socket.js` will also be loaded.

All the variables for the sensors are global so they can be reached to any file inside the application project. The IP address we are using to connect with the TE0720 is 192.168.0.10, and the socket is set to the standard 8080 port. When the application receives data to the 8080 port, it will identify what type of data that is received. The data coming from TE0720 is in comma separated format. The first word of the message sent from TE0720, is for identification. For instance the message can start with the word Temp. We will then know that the message from TE0720 will contain the data from the temperature sensors. The message will in other words contain the parameters for exhaust temperature from cylinder 1 - 8, followed by fluid temperature, coolant temperature and oil temperature.

When we are going to use the corresponding parameters in other files, we can then use the same variables such as `EXTS[6]`, `FTS`, `CTS` and so on. Doing it this way, will keep the code structure simple rather than making a new socket connection and an `onMessage()` method for every single JavaScript file we need to read the parameters. This way it will also make it much easier if we want to change the structure of the socket communication. We can for example modify the structure of the socket so the parameters are sent separately, with each having an independent identifier.

**Code 19 - JavaScript: Part of the code to fetch the data from the socket**

```
var EXTs = [];  
var dash_kmt;  
var CRANK;  
var FFS;  
var FTS;  
var CTS;  
var OTS;  
var OPS;  
var FPS;  
var ws = new WebSocket('ws://192.168.0.10:8080/');  
  
ws.onmessage = function(event) {  
var message = event.data;  
var message_split = message.split(",");  
  
switch(message_split[0]){  
  
    case "Temp":  
        for (i = 1; i <= 8; i++) {  
            EXTs[i] = parseFloat(message_split[i]);  
        }  
        FTS = parseFloat(message_split[9]);  
        CTS = parseFloat(message_split[10]);  
        OTS = parseFloat(message_split[11]);  
        break;  
    case "dash":  
        dash_kmt = parseFloat(message_split[1]);  
        FFS = parseFloat(message_split[2]);  
        CRANK = parseFloat(message_split[3]);  
        OPS = parseFloat(message_split[4]);  
        FPS = parseFloat(message_split[5]);  
        break;  
    }  
};
```



### 11.3.5. Performance analysis

In order to satisfy our requirement from requirement specification document, ID-044, 045 and 046, respectively 50, 100 and 200 milliseconds, we have performed some analytic tests to capture the application performance in frames per second and milliseconds. The first application draft we made was running at average 6.8 fps (147 ms). After some adjustments inside the application code, we went from 6.8 fps to 43.8 fps (20.8 ms). Calculations were made, and we found out that the application performance has been increased by 544 % (From 6.8 fps to 43.8 fps). This means that we have met requirement ID-044. Equation (164) shows how to convert frames per seconds too milliseconds.

$$\frac{1 \text{ Second}}{\text{Frames Per Second}} \cdot 1000 = \text{Milliseconds} \quad (164)$$

We have performed different analyse tests on how fast the application performance is running, and we manage to increase the application performance by analysing diagram close. The in-built tool in Firefox so-called “Inspect Element (Q)”<sup>133</sup> gave us the opportunity to measure the application performance in different scenarios. The tests consist of how many frames per second and milliseconds each diagram uses. Table 57 shows the results from the first draft compared to the second draft which is shown in Table 58.

**Table 57 - First Draft Analyse low performance**

Duration (Minutes)	Description	Frames/Second	Milliseconds
10 m	Dashboard	6 fps	167 ms
10 m	Exhaust Temperature	7 fps	143 ms
10 m	Cylinder Pressure	7 fps	143 ms
10 m	Mass Air Flow	7 fps	143 ms
10 m	Flywheel	10 fps	100 ms
10 m	Lambda	7 fps	143 ms
10 m	Knock	6 fps	167 ms
10 m	Fluid Temperature	9 fps	111 ms
10 m	Fluid Pressure	10 fps	100 ms
10 m	Throttle Position	7 fps	143 ms
10 m	Manifold Absolute Pressure	6 fps	167 ms
10 m	Flex Fuel	6 fps	167 ms
<b>Average</b>	<b>Everything</b>	<b>6.8 fps</b>	<b>141 ms</b>

<sup>133</sup> Mozilla, Inspect Element



The application was running with low performance due to that every dynamic diagram in the background was receiving data samples simultaneously. The algorithm behind this had to be modified. After some internal code analysis, we found out that the solution was to change the jQuery code we made in first draft, because it was running the diagrams in parallel without handling the data in order. The old jQuery code worked like this: Wait for the user to click, then fetch the diagram, and then wait for the click again. The problem was that it didn't check if the diagrams in the background were running or not.

#### Code 20 - jQuery performance code

```
$(function(){
    var pres_row = 0;
    $('[data-row]').on('click', function() {
        var row = $(this).attr('data-row');
        $('.active').removeClass('active');
        $('#table' + row).addClass('active');

        if(row !== pres_row){
            // clear the currently running interval if any
            if('diagramInterval' + pres_row in window) {
                clearInterval(window['diagramInterval' + pres_row]);
            }
            pres_row = row;
            // load again only if not loaded
            if('diagram' + pres_row in window) {
                window['diagram' + pres_row]();
            }
            else {
                $.getScript("SensorTables/diagram" + pres_row + ".js", function(){
                    window['diagram' + pres_row]();
                });
            }
        }
    });
});
```

This code shown in Code 20 is responsible for only displaying the current diagram that is being viewed by the user, and hold the rest diagrams idle. Now the user can navigate freely through the diagrams without any delay in high speed. The jQuery code works like: Wait for the user to click, then after the click it fetches the diagram, and then it holds the diagram that is running in background idle. This all leads to the second application draft, where the performance has been improved.



Table 58 - JQuery Performance Code

Duration (Minutes)	Description	Frames/Second	Milliseconds
10 m	Dashboard	20 fps	50 ms
10 m	Exhaust Temperature	45 fps	22 ms
10 m	Cylinder Pressure	50 fps	20 ms
10 m	Mass Air Flow	48 fps	21 ms
10 m	Flywheel	50 fps	20 ms
10 m	Lambda	54 fps	19 ms
10 m	Knock	48 fps	21 ms
10 m	Fluid Temperature	55 fps	18 ms
10 m	Fluid Pressure	52 fps	19 ms
10 m	Throttle Position	50 fps	20 ms
10 m	Manifold Absolute Pressure	49 fps	20 ms
10 m	Flex Fuel	48 fps	21 ms
<b>Average</b>	<b>Everything</b>	<b>43.8 fps</b>	<b>20.8 fps</b>

Figure 201 below, shows the performance results in a statistical overview. These results come from the two tables above, Table 57 and Table 58.

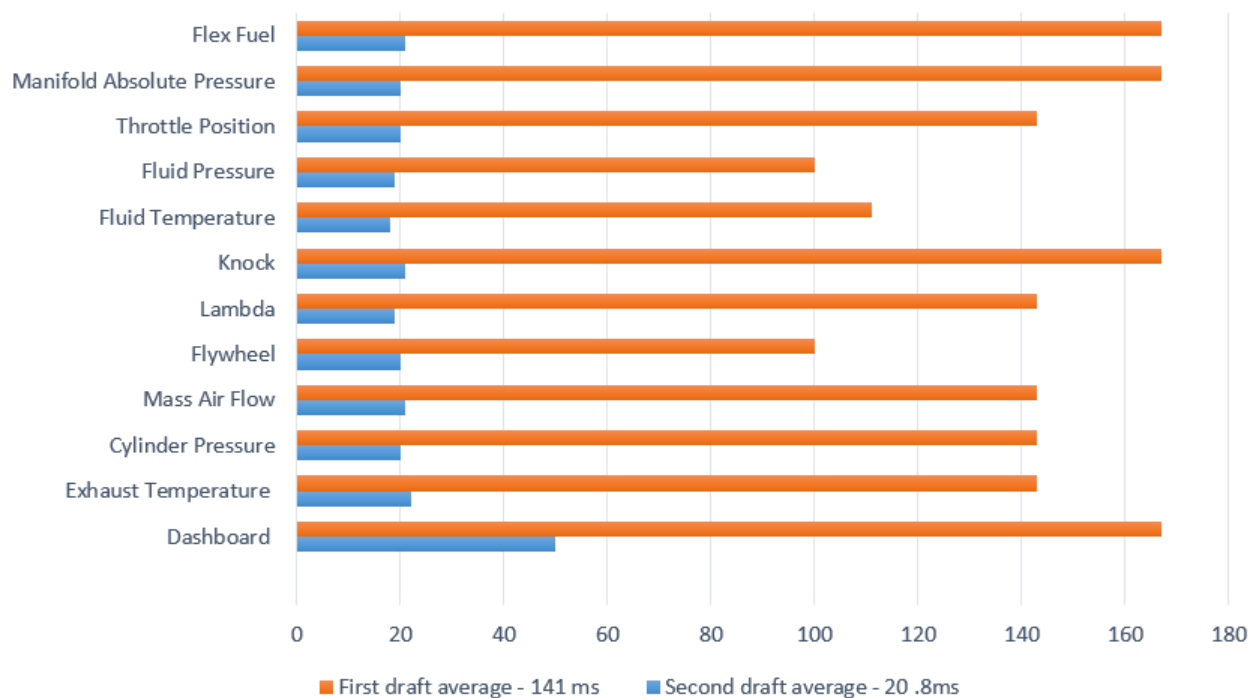


Figure 201 - Performance results



### 11.3.6. Application configuration files

When creating a PhoneGap application, it has to contain several configuration files. Most of them are created for specifying metadata about the application. The main configuration file for PhoneGap is `config.xml` which handles all the plugins and native functionality. However for Android platform, `AndroidManifest.xml` is also an important configuration file which handles all required permissions, activities and application specifications. More about these configurations can be read in the following chapters.

#### 11.3.6.1. *Config.xml*

To make it easy to handle the behaviour of the application, we are using `config.xml` to enable plugins and other preferences. `config.xml` is a global configuration file, like Android Manifest and gradle files. Instead of making xml file for each mobile platform, we are just using the `config.xml` file to configure the behaviour of the application. Code 21 shows our `config.xml` file. This file includes the application name and description, which is first of all used to set the application name for navigating to the application on the tablet. The application and description in `config.xml` is also used when publishing to an application store like Play Store. We have also set `index.html` as our start page. It is important to declare this part, so that the application can know which page to load when the application is started. Additionally have declared plugins and preferences to make the application behave like we want it to. After we run the PhoneGap build command, the `config.xml` will copy the configuration from the `config.xml` into wherever they are needed, mostly in `AndroidManifest.xml` and Gradle files.

#### Code 21 - XML: `config.xml` configuration file

```
<?xml version='1.0' encoding='utf-8'?>
<widget id="com.KPEC.app" version="0.0.1"
xmlns="http://www.w3.org/ns/widgets"
xmlns:cdv="http://cordova.apache.org/ns/1.0">
  <name>KPEC</name>
  <description> Be the one to control the engine.</description>
  <author href="http://www.kpec.no">KPEC team </author>
  <content src="index.html" />
  <access origin="*" />
  <plugin name="InAppBrowser"
value="org.apache.cordova.InAppBrowser" />
  <preference name="exit-on-suspend" value="true" />
  <preference name="KeepRunning" value="false"/>
  <preference name="LoadingDialog" value="Starting up,Getting things
ready for you!"/> <!--While application startup-->
  <preference name="LoadingPageDialog" value="Loading,Please wait!"/>
  <preference name="android-minSdkVersion" value="17"/>
  <preference name="android-targetSdkVersion" value="19"/>
</widget>
```



### 11.3.6.2. Android Manifest

The AndroidManifest.xml file provides essential information about the application. It needs to be properly built to run the application. This file describes the components of the application. All the attribute names beginning with android are universal and will apply to all the layouts in the Android application. For instance in Code 22, we have set the hardwareAccelerated attribute to be true. This means that the entire application will use the webkit, and not only on a few layouts. We also need to specify the permissions in the AndroidManifest.xml file. For this application we are only asking the user to give the application permission to internet connection. For further development and permissions for functions like GPS, messaging and device power management may be specified inside the manifest file. When the user installs the release application from Play Store, the user will then get a list of all these permissions that the application is requiring from the tablet.

#### Code 22 - XML: AndroidManifest.xml configuration file

```
<?xml version='1.0' encoding='utf-8'?>
<manifest
  android:hardwareAccelerated="true"
  android:versionCode="1"
  android:versionName="0.0.1" package="com.first.app"
  xmlns:android="http://schemas.android.com/apk/res/android">
  <supports-screens
    android:anyDensity="true"
    android:largeScreens="true"
    android:normalScreens="true"
    android:resizeable="true"
    android:smallScreens="true"
    android:xlargeScreens="true" />
  <uses-permission android:name="android.permission.INTERNET" />
  <application
    android:hardwareAccelerated="true"
    android:icon="@drawable/icon"
    android:label="@string/app_name">
    <activity
      android:configChanges="orientation|keyboardHidden|keyboard|screenSize|locale"
      android:label="@string/activity_name"
      android:launchMode="singleTop"
      android:name="CordovaApp"
      android:theme="@android:style/Theme.Black.NoTitleBar"
      android:windowSoftInputMode="adjustResize">
      <intent-filter android:label="@string/launcher_name">
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity> </application>
  <uses-sdk android:minSdkVersion="19" android:targetSdkVersion="19" />
</manifest>
```





### 11.3.6.3. *build.gradle*

Like a regular native application, PhoneGap also uses Gradle to build the application. The `build.gradle` in Code 23 is created in Domain Specific Language (DSL), and is an advanced build toolkit to create custom build logic through plugins. Most of the structure is generated by PhoneGap, but we had to update the SDK- and build tool version to our needs. Here we could also set the minimum SDK version required to run this application, but since we are only developing one type of the application, we don't find it necessary to implement that. Although by setting the minimum SDK version, we can create different kind of applications dependent of the tablet hardware. We could for instance create an additional application that will support the older versions (lower API level) of tablets with limited animation rendering.

#### Code 23 - DSL: `build.gradle` file for Android application

```
buildscript {
    repositories {
        mavenCentral()
    }

    dependencies {
        classpath 'com.android.tools.build:gradle:0.10.+
    }
}

apply plugin: 'android-library'

android {
    compileSdkVersion 19
    buildToolsVersion "19.0.0"

    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_7
        targetCompatibility JavaVersion.VERSION_1_7
    }

    sourceSets {
        main {
            manifest.srcFile 'AndroidManifest.xml'
            java.srcDirs = ['src']
            resources.srcDirs = ['src']
            aidl.srcDirs = ['src']
            renderscript.srcDirs = ['src']
            res.srcDirs = ['res']
            assets.srcDirs = ['assets']
        }
    }
}
```



## 11.4. Sensors- and drivers overview

Each diagram out of 12 diagrams that are made in the application is referenced to the sensors/drivers overview. This template is made to make it easier and more flexible for future software developers to understand which sensors and drivers must be implemented in the application. In order to create 12 different diagrams, we have been using Figure 202, as a template for the application development. The architecture of the database structure is divided into two parts. The first part consists of sensors, and the second part consists of drivers.

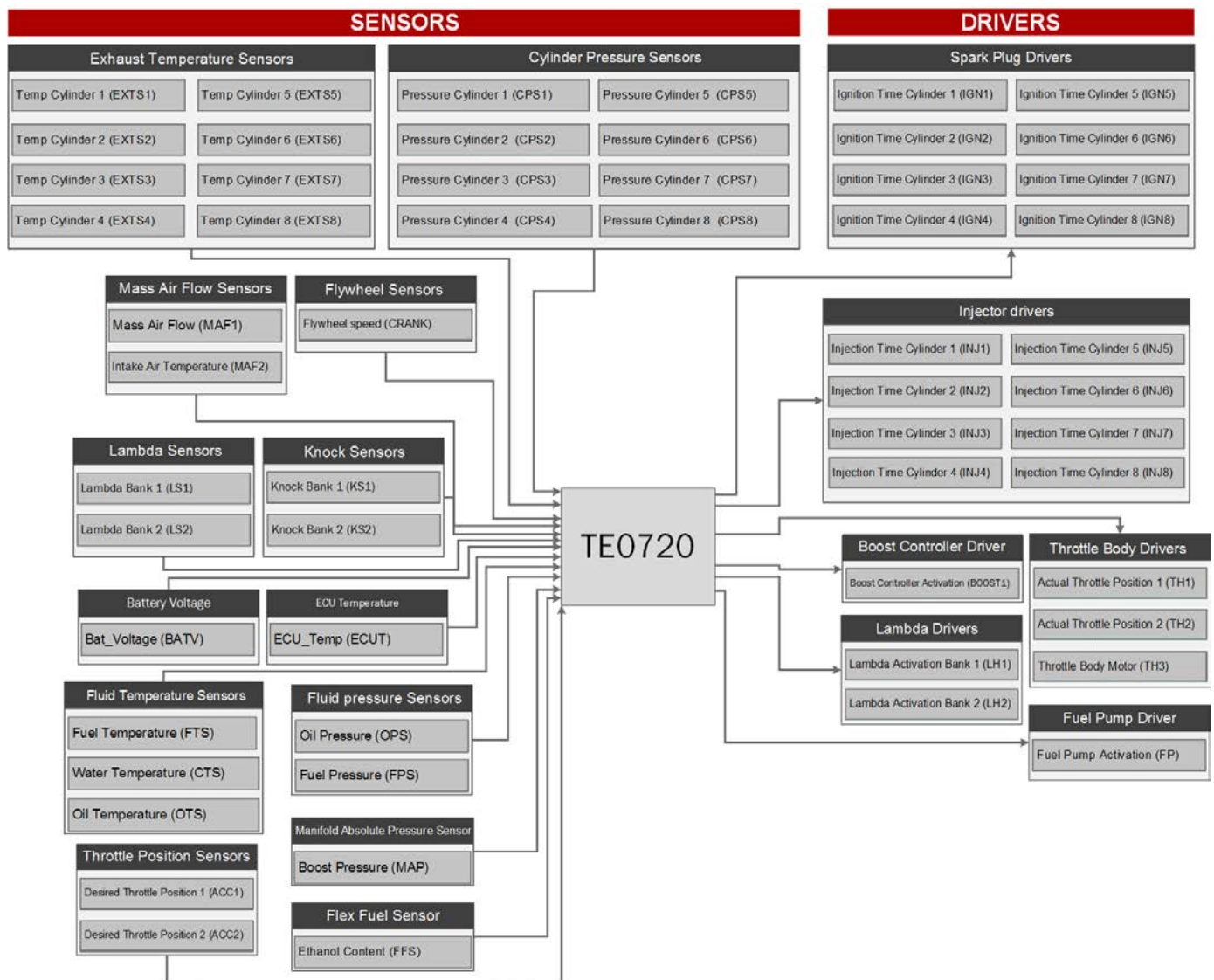


Figure 202 - Sensors/Drivers overview



## 11.5. Configuration manuals and recommendations

This chapter contains a manual to configure PetaLinux and Android application development. Additionally this chapter contains a guide on how to use GitHub.

### 11.5.1. PetaLinux startup guide

Certain things have to be done on the host machine, to further develop PetaLinux Operating System.

The host machine must have a supported Linux Operating System installed. Xilinx is recommending several operating systems. We are using both Linux Ubuntu Desktop 14.4 (Debian based) and CentOS 6.6. Final (RedHat Enterprise based). We were not able to install the PetaLinux Tools CentOS 7 when we configured our host machines in January 2015. The Linux operating system can also be installed using VMware or VirtualBox (On top of a Windows Operating System). Optionally the Linux OS can also be installed beside the Windows OS (dual-boot).

Minimum workstation requirements:

- 4 GB RAM (recommended minimum for Xilinx tools)
- Pentium 4 2GHz CPU clock or equivalent
- 5 GB free HDD space

Supported OS:

- RHEL 5.9 (32-bit or 64-bit)
- RHEL 6 (32-bit or 64-bit)
- SUSE Enterprise 11 (32-bit or 64-bit)
- CentOS 6 (64-bit)
- Ubuntu 14.04 (64 bit)

After a Linux operating system is successfully installed into the host machine, there are some important Xilinx applications and Linux libraries that must be installed, in order to build and further develop PetaLinux.

Dependent on which operating system is installed; several libraries need to be installed before installing PetaLinux Tools. Most of the necessary commands are listed below in Figure 203. The list to the left is for CentOS platform using yum packages, while the list to the right is for Ubuntu platform using deb packages.



Tool/Library	CentOS 6 (64-bit) (yum install)	Ubuntu 14.04 (64 bit) (sudo apt-get)
Dos2unix	dos2unix	tofrodo
ip	iproute	iproute
gawk	gawk	gawk
gcc	gcc	gcc
git	git	git-core
make	gnutls-devel	make
netstat	net-tools	net-tools
ncurses	ncurses-devel	ncurses-dev, libncurses5-dev
tftp server	tftp-server	tftpd
zlib	zlib-devel	zlib1g-dev
flex	flex	flex
bison	bison	bison
32bit libs	libstdc++.i686, glibc.i686, libgcc.i686, libgomp.i686, ncurses-libs.i686, zlib.i686	setup 32bit libs support in ubuntu, lib32z1, lib32ncurses5, lib32bz2-1.0, ia32gcc1, lib32stdc++6, libselinux1

**Figure 203 - Necessary 32-bits libraries**

We found out that some of the 32 bits libraries for both CentOS and Ubuntu were deprecated. Therefore we used the command in Code 24.

#### Code 24 - Bash: 32-bits libraries

```
su -c 'yum -y install --skip-broken glibc.i686 arts.i686
audiofile.i686 bzip2-libs.i686 cairo.i686 cyrus-sasl-lib.i686 dbus-
libs.i686 directfb.i686 esound-libs.i686 fltk.i686 freeglut.i686
gtk2.i686 hal-libs.i686 imlib.i686 lcms-libs.i686 lesstif.i686
libacl.i686 libao.i686 libattr.i686 libcap.i686 libdrm.i686
libexif.i686 libgnomecanvas.i686 libICE.i686 libieee1284.i686
libsigc++20.i686 libSM.i686 libtool-ltdl.i686 libusb.i686
libwmf.i686 libwmf-lite.i686 libX11.i686 libXau.i686 libXaw.i686
libXcomposite.i686 libXdamage.i686 libXdmcp.i686 libXext.i686
libXfixes.i686 libXkbfile.i686 libxml2.i686 libXmu.i686 libXp.i686
libXpm.i686 libXScrnSaver.i686 libxslt.i686 libXt.i686 libXtst.i686
libXv.i686 libXxf86vm.i686 lzo.i686 mesa-libGL.i686 mesa-libGLU.i686
nas-libs.i686 nss_ldap.i686 cdk.i686 openldap.i686 pam.i686
popt.i686 pulseaudio-libs.i686 sane-backends-libs-gphoto2.i686 sane-
backends-libs.i686 SDL.i686 svgalib.i686 unixODBC.i686 zlib.i686
compat-expat1.i686 compat-libstdc++-33.i686 openal-soft.i686 alsa-
oss-libs.i686 redhat-lsb.i686 alsa-plugins-pulseaudio.i686 alsa-
plugins-oss.i686 alsa-lib.i686 nspluginwrapper.i686 libXv.i686
libXScrnSaver.i686 qt.i686 qt-x11.i686 pulseaudio-libs.i686
pulseaudio-libs-glib2.i686 alsa-plugins-pulseaudio.i686'
```



After the Linux libraries are installed, we can then start installing the necessary Xilinx software.

- Vivado 2014.4 WebPack (Free version). Remember to choose the installation with the SDK included.<sup>134</sup>
- PetaLinux Tools 2014.4. This tool has to be installed, to access all the PetaLinux commands from the terminal.<sup>135</sup> The PetaLinux commands are supported on both bash and shell command-line-interface.

Note that the version number of the Vivado, SDK and the PetaLinux Tools must be the same. Also note that PetaLinux Tools are upgraded to newer version after each second release. (2014.2 , 2014.4...). This means that Vivado upgrades like 2014.1 and 2014.3 are not supported with PetaLinux 2014.4 Tools.

After downloading the required tools, the tools are usually installed into the /opt folder. Note that this is a root folder, so if it you want to give the permissions to a normal user, use the command in Code 25 from your Linux terminal.

#### Code 25 - Bash: Permission command

```
chown -R username:username <path/to/the/folder>
```

To start the Xilinx application, it is needed to source the settings inside the Vivado and/or SDK folders. Use the commands in Code 26.

#### Code 26 - Bash: Source into Vivado and/or SDK folders

```
source /opt/Xilinx/2014.4/Vivado/settings64.sh  
source /opt/Xilinx/2014.4/SDK/settings64.sh
```

Then just type vivado for Xilinx Vivado or xsdk for the Xilinx SDK to start the corresponding applications.

Note that you have to source into the Vivado settings first before sourcing into PetaLinux settings. Otherwise the Linux host machine will not be able to find the PetaLinux commands.

---

<sup>134</sup> Xilinx, Install Vivado and SDK

<sup>135</sup> Xilinx, Install PetaLinux Tools



### 11.5.2. Android development configuration guide

Unlike development of PetaLinux, you can use Windows, Linux or OS X to setup the host machine for Android application development.

There are several important tools that need to be installed for further development of the PhoneGap application.

The first step is to install Android SDK.<sup>136</sup> The Android SDK tool can either be installed with Android Studio development platform or as a stand-alone tool. To further develop our PhoneGap application we only need to install the Android SDK, but the Android Studio can also be used as a pure code editor if desired.

The next step is to install the required PhoneGap tools.<sup>137</sup> Since the PhoneGap requires NodeJS, it has to be installed first.

#### Code 27 - Bash: From CentOS

```
su -c yum install npm
su -c yum install nodejs
```

#### Code 28 - Bash: From Ubuntu

```
sudo apt-get install npm
sudo apt-get install nodejs
```

After installing NodeJS, use the Terminal or cmd to install PhoneGap CLI using the command in Code 29.

#### Code 29 - Bash: Install PhoneGap CLI

```
npm install -g phonegap
```

Adobe PhoneGap provides a way for users to create mobile applications using technologies such as HTML, CSS, and JavaScript. In order to build the application we have been using PhoneGap build, because it is a cloud service for compiling PhoneGap applications. PhoneGap build will always be built against the required SDK for the platform it is targeting. In order to build the application, zip the folder or commit your folder to GitHub, either way PhoneGap build handles the building process of the application.

---

<sup>136</sup> Android, Download Tools

<sup>137</sup> PhoneGap, Install PhoneGap



### 11.5.3. GitHub startup guide

In order to start working on the code that is made by the software developers from KPEC. There are some important steps that must be followed. First create a GitHub<sup>138</sup> account. Then install Git Bash (available for Mac OS X, Windows and Linux)<sup>139</sup>. Keep in mind that the repository is private, so only invited members can contribute. After installation and being added as a contributor to the private repository, start Git Bash and use these commands:

#### Code 30 - Bash: Configure GitHub alias

```
git config --global user.name "YOUR NAME"
git config --global user.email "YOUR EMAIL ADDRESS"
```

Upload the entire code from GitHub on host machine. Keep in mind that if the operating system is Linux based, connecting over SSH is a must, otherwise the code content cannot be fetched:

#### Code 31 - Bash: Clone the KPEC repository

```
git clone https://github.com/LocalHawk/KongsbergECU.git
```

Whenever a change has been implemented to the code, we strongly recommend doing a commit, because of traceability. A commit, is basically a changelog, that can be tracked by the command git log, or viewing it on GitHub website. Code 32 to Code 34 is used for uploading while Code 35 is used for downloading the content from the repository.

#### Code 32 - Bash: Make the files ready to be pushed

```
git add --all
```

#### Code 33 - Bash: Add a text (Important for the traceability)

```
git commit -m "Note your changes here.."
```

#### Code 34 - Bash: Push your content to private repository

```
git push origin master
```

#### Code 35 - Bash: To download the code from private to the host machine

```
git pull origin master
```

<sup>138</sup> GitHub, Official website

<sup>139</sup> GitHub, Download Git Bash terminal



## 12. ECU Connectivity

This section of the design document discusses important factors relating the design of the engine control unit that is important to consider for future development of the ECU. This includes signal description, pinout lists and application circuitry for the ECU which is essential for software developers, as well as test personnel and assemblers.

### 12.1. ECU header

The KPEC ECU is an advanced unit which handles data from an extended amount of sensors compared to solutions on the market today. The ECU connector needs to connect the engine sensors towards the engine control unit.

The I/O requirement for the connector is to handle 93 in- and output signals, as well as 5 V supply voltage to sensors and battery supply for the ECU. The total I/O count for the ECU is 103 with power supplies and signals, which is the minimum pin count requirement for the connector. The operating environment for the ECU is typically in the engine bay which implies that there is a risk for water splash towards the unit. The ECU connector requires minimum IP67 rating, and should be suited for automotive applications.

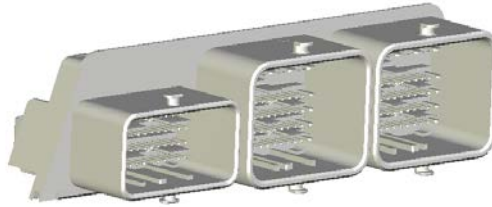
Military-standard connectors are typically round and for signal applications, this implies that multiple connectors must be fitted to the casing for different power requirements. The ECU's 8 low side drivers which charges the ignition coils inductor requires a maximum current of 20 A. The ECU is also designed to a wide variety of engine configuration, from four to eight cylinders, where a four cylinder engine is a beneficial starting point for software developers in the project. We have therefore chosen a connector from Molex that exceeds our requirements.

Molex's three-bay CMC header<sup>140</sup> with support for 154 pins (48 + 53 + 53) complies with RoHS and industrial standards, and is therefore beneficial in this design. The connector counterparts from the electrical harness for this solution are sealed with silicon gaskets which comply with IP67 rating. Due to the fact that four cylinder engines are most commonly used in racing and for testing purposes, a natural distribution of the connectors is by common signals and cylinder banks. A 3D CAD drawing of the connector is shown in Figure 204.

---

<sup>140</sup> Molex CMC datasheet



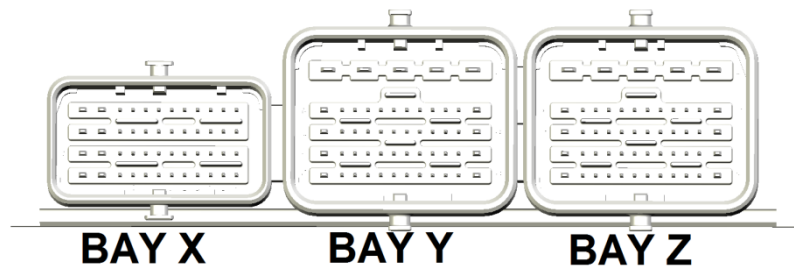


**Figure 204 - 3D CAD model of Molex 34763-0001**

This connector consists of three connector bays (slots), hence with three different current limits. Table 59 gives a description of the physical specifications of the mating connectors for the ECU header. This must be seen in compliance with Figure 205.

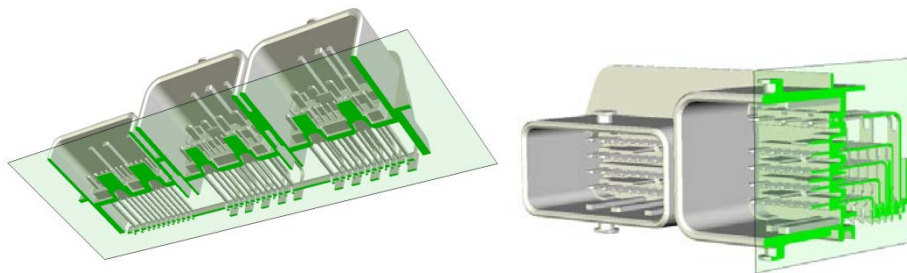
**Table 59 - Physical specifications for mating connectors**

BAY		BAY X	BAY Y	BAY Z
<b>COLOR MARK</b>		White	Brown	Black
<b>ASSIGNMENT</b>		Common for bank 1 & bank 2	Bank 1 sensors and actuators.	Bank 2 sensors and actuators.
<b>TOTAL PINS</b>		<b>48</b>	<b>53</b>	<b>53</b>
<b>Signal</b>	<b>&lt;2.5 A</b>	40	40	40
<b>Low Power</b>	<b>&lt;12 A</b>	8	8	8
<b>High Power</b>	<b>&lt;21 A</b>	0	5	5



**Figure 205 - Overview of header connector bays.**

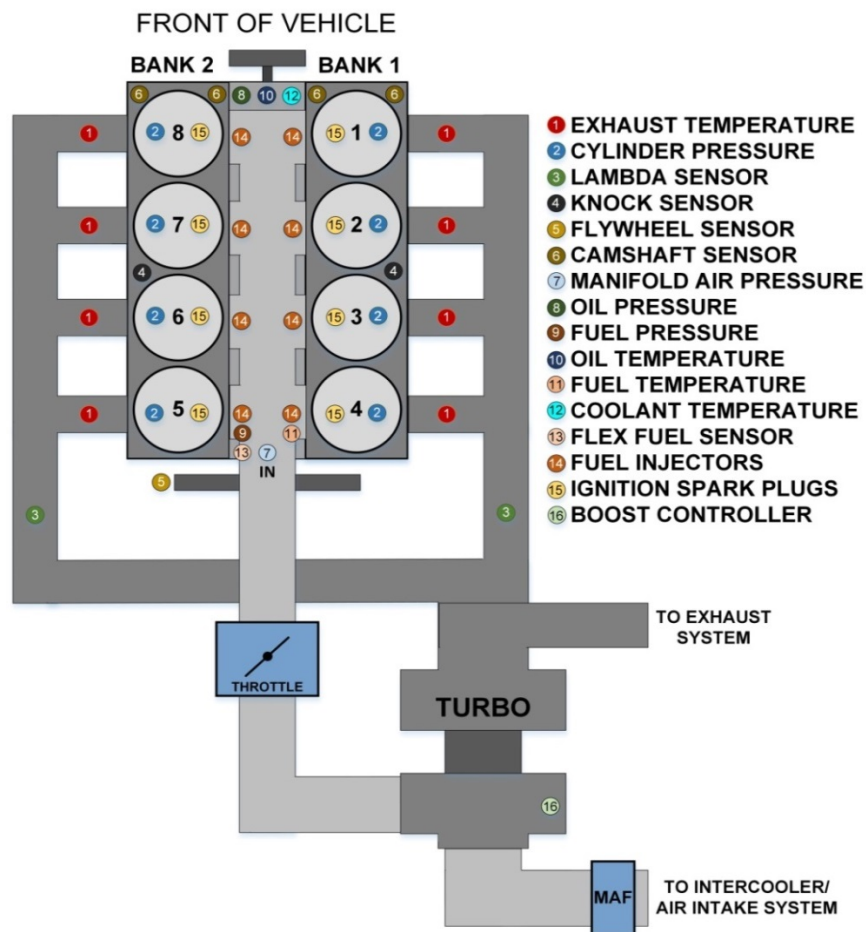
As Table 59 implies, there is both a visual and physical protection against user faults such as wrong connections of sensors and actuators for bank 1 and bank 2. The digital common sensors are insulated in the common bay x to avoid noise. A section view of the connector is provided in Figure 206, which displays the internal construction of the electrical conductors which is casted in the housing. Due to the high number of pins in the connectors, a unique numbering system must be defined to ensure that correct wires are connected to correct sensors and internal signals in the ECU.



**Figure 206 - 3D Section view of the ECU header**

## 12.2. ECU connectors

The counterpart to the ECU header is the engine harness connectors which collect signals from the sensors, and supply actuators with enable signals. One of our product concepts is to have a versatile engine management system that can easily be adapted for different engine configurations. To enable this functionality, the bays of the header are assigned to different cylinder banks. Figure 207 gives an overview of the location and amount of the different sensors and actuators.



**Figure 207 - V8 engine sensors and actuators typical location.**



### 12.2.1. Common connector - BAY X (48 pins)

The common connector connects the common sensors and actuators for the engine as communication, digital inputs, pressure sensors and supply voltage. As well as general purpose outputs and inputs, which are separated by analog and digital signals in the connector. This connector contains 48 slots for wires, where 8 of these are designed for high current (12 A). These are assigned for power supply and ground, which provides a maximum of 48 A delivered to the ECU. For pinout description, refer to Appendix VI.

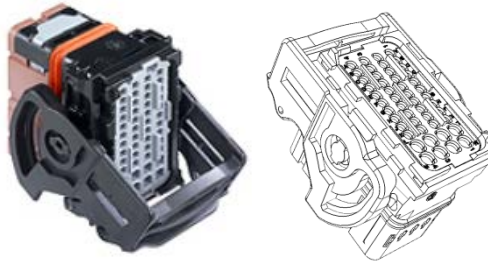


Figure 208 - Bay X connector 48 pins.<sup>141</sup>

### 12.2.2. Bank 1 connector - Bay Y (53 pins)

The bank 1 connector connects the sensors and actuators for cylinder bank 1 of a V-engine. The sensors that are specific for bank 1 is exhaust temperature, cylinder pressure, knock, hall and lambda sensors as well as injectors, fuel pump, boost controller, throttle controller and ignition coils. This connector has 5 high current pins that handle continuous 21 A loads, which are dedicated for our ignition coil low side switches. There is also a dedicated pin for flyback diode connection. There is 8 pin that handles 12 A continuous loads, which are dedicated for fuel injectors and low side switches.

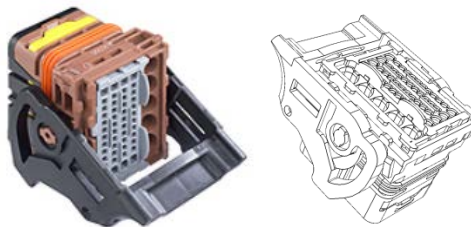


Figure 209 - Bay Y connector 53 pins.

### 12.2.3. Bank 2 connector - Bay Z (53 pins)

The bank 2 connector connects the bank 2 sensors and actuators specific for an eight cylinder V-engine. This connector has the same specification and pin dedication as the bank 1 connector as mentioned above, with other casing. If the ECU are used to interface a 4 cylinder inline or boxer engine, there is no need to connect the bank 2

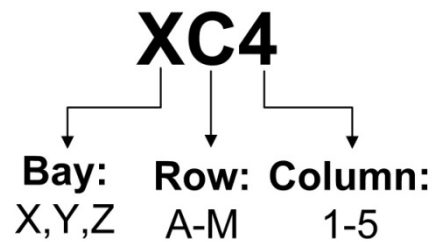
<sup>141</sup> Molex CMX datasheet



connector (bay Z). There should have a cover (silicone sealant) which protects against moisture to ensure maximum lifetime.

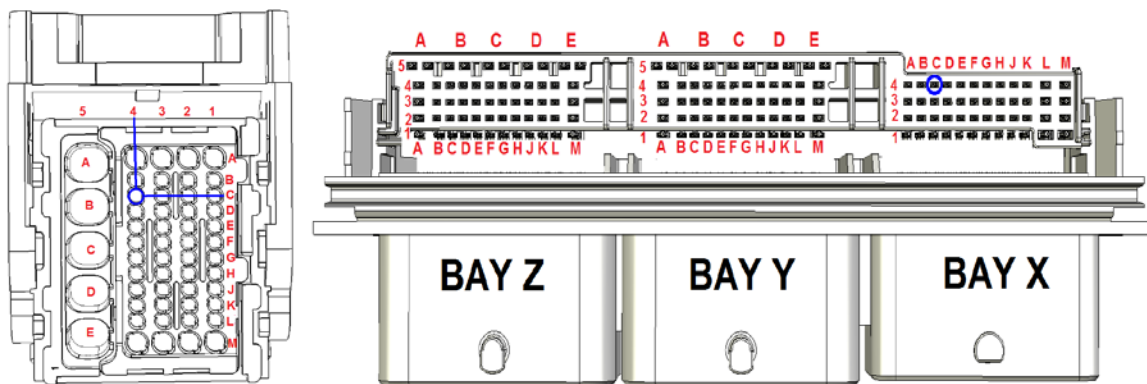
### 12.3. Connector and header numbering system

The PCB header contains 154 pins distributed over three bays; it is not intuitive to number this connector from 0 - 153 for the assemblers of this system. The group has therefore defined our own numbering system for the connectors X, Y and Z, to easier assign signals and supply voltages to the sensors and actuators. The numbering system refers to the connector seen from above (from wiring harness) towards the header, and consists of two letters and an integer as shown in Figure 210.



**Figure 210 - Unique numbering system for KPEC connectors**

The corresponding pin location in the harness connector and underside of ECU header are provided in Figure 211. Note that this is seen vertically from the harness side with the connector mounting hoop unlocked.



**Figure 211 - XC4 location in bay X connector and underside of ECU header.**

For the 48 pins connector, the row and column pin location is casted in the plastic which don't confuse the assemblers and test personnel. The connectors must be manually mounted on the wiring harness with terminals according to the cross section of the wires. The 2.5 A signal line requires 0.75 mm<sup>2</sup> cable, 12 A requires 2.5 mm<sup>2</sup> cables and 21 A requires 5 mm<sup>2</sup> cables. There are special tools required for mounting



the wiring harness to the connectors and the ECU header to the PCB. Refer to the datasheets to ensure that the connectors and headers are mounted correctly.

## 12.4. Schematic implementation

The ECU connector is implemented in the schematic design by three blocks analogous to the X, Y and Z bay. The 10 isolated pins in the blocks (Y/Z 1-5 A&B) is dedicated for high current driver circuitry (21 A continuously). The pins are partitioned into A&B due to better connection and stability towards the PCB. The schematic implementation is shown in Figure 212.

The risk of connecting the wiring harness in incorrect manner is greatest between bay Y and Z for the bank 1 and bank 2 sensors and actuators. To avoid damages on the ECU motherboard, the driver circuitry are assigned on the same pin locations in the bays. If misconnection occurs, the extra sensors that are assigned on bay X are connected directly to ground. This improves safety for the end user, as well as the ECU motherboard. The connection of the wiring harness towards the connectors must be performed by qualified personnel only.

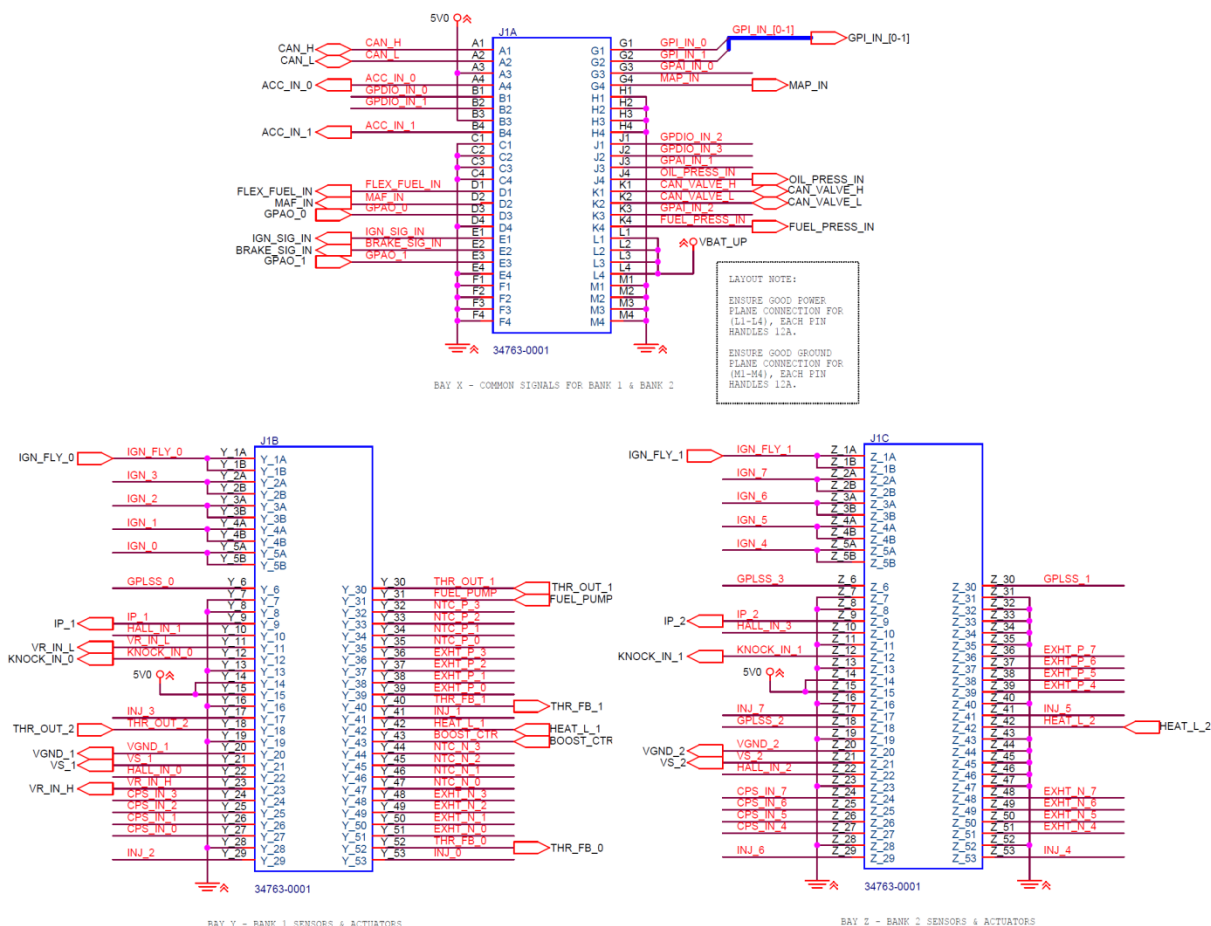
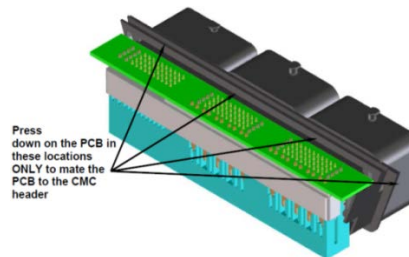


Figure 212 - Schematic implementation of ECU connector



## 12.5. Practical implementation

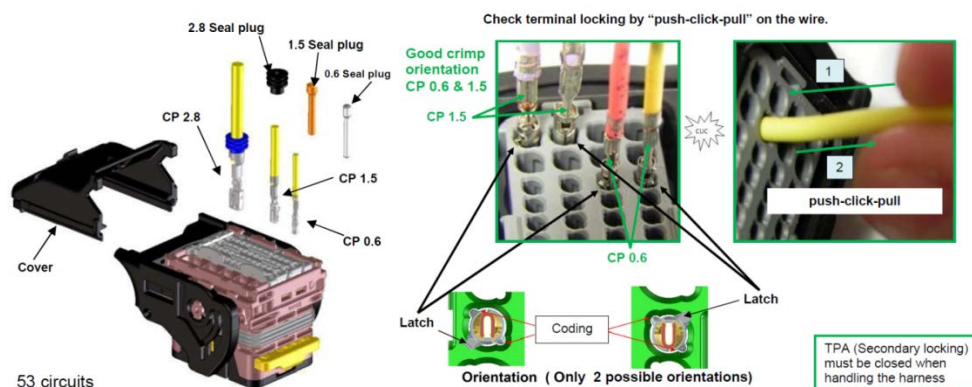
The ECU header needs to be mounted on the PCB in a special manner to ensure that the connection and stability are obtained throughout the lifetime of the ECU. The header is first pressed down in the holes of the PCB, and then soldered towards the pads on the PCB as shown in Figure 213. There should be adequate sealant between the connector and the ECU casing to ensure that moist cannot enter the unit.



**Figure 213 - Mounting of ECU header towards PCB<sup>142</sup>**

When the ECU is ready for implementation in a vehicle, a wiring harness needs to be connected to the mating connectors. This harness needs to be designed for the specific automotive application, where EMI standards should be fulfilled to ensure that the ECU works as intended under all conditions. The wiring harness should have twisted signal lines towards digital sensors, and adequate coating (asphalt wiring tape) to avoid short circuitry towards signal lines, ground or supply.

High quality wires used in the harness decreases the risk of electrical faults in the system. The cross-section rating of the different signals should be fulfilled to ensure that no heat is developed in the wires when loaded. The wires must be connected individually in the mating connector, as shown in Figure 214. The slots in the connector which is not assigned should be sealed to avoid moist entering the ECU.



**Figure 214 - Correct mounting of wires in mating connector<sup>143</sup>**

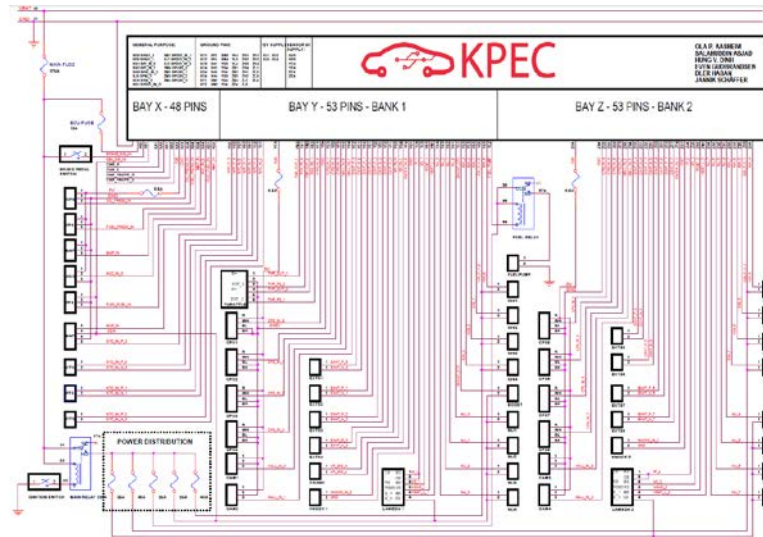
<sup>142</sup> Figure from Molex CMC datasheet

<sup>143</sup> Figure from Molex CMC datasheet



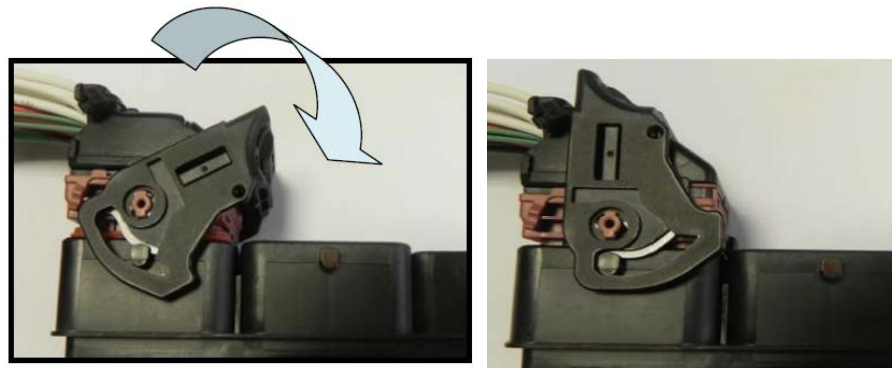


The sensor supply and ground signals in the wiring harness can be extended with branches which ensures that all sensors have the same potential. Sensors should have separate ground wires to ensure that noise and potential difference don't destroy the sensor signal. The sensor ground should never be connected directly to chassis ground (engine block, body work). The 5V sensor supply has no internal protection in the ECU, so this supply lines must be protected with external fuse. Wiring schematics for connection of wiring harness towards sensors and actuators are provided in Appendix VI.



**Figure 215 - Wiring schematics for sensor and actuator connection to ECU.**

When the wiring harness is connected towards the sensors and actuators and mated into the connector, connection should be performed as shown in Figure 216. Datasheets specific for each external sensor and actuator should be considered before any connections towards the ECU is performed.



**Figure 216 - Connection of mating connectors to ECU header.<sup>144</sup>**

<sup>144</sup> Figure from Molex CMC datasheet



## Appendix I - List of abbreviations

Abbreviation	Description
<b>ABS</b>	Anti-lock Braking System
<b>AC</b>	Alternating Current
<b>ADC</b>	Analog-to-Digital Converter
<b>AI</b>	Analog Input
<b>ARM</b>	Advanced RISC Machines
<b>AP</b>	All Programmable
<b>API</b>	Application Programming Interface
<b>APK</b>	Android Application Package
<b>AVD</b>	Android Virtual Device
<b>B2B</b>	Board to Board
<b>BA</b>	Battery Voltage Compensation
<b>BGA</b>	Ball Grid Array
<b>BJT</b>	Bipolar Junction Transistor
<b>BLDC</b>	Brushless Direct Current
<b>BOM</b>	Bill Of Materials
<b>BTDC</b>	Before Top Dead Centre
<b>BW</b>	Bandwidth
<b>CA</b>	Crankshaft Angle
<b>CAD</b>	Computer Aided Design
<b>CAN</b>	Controller Area Network
<b>CM</b>	Common Mode
<b>CMOS</b>	Complementary Metal-Oxide Semiconductor
<b>CPS</b>	Cylinder Pressure Sensor
<b>CPU</b>	Central Processing Unit
<b>CTS</b>	Cylinder Temperature sensor
<b>CS</b>	Chip Select
<b>CSS</b>	Cascading Style Sheets
<b>DAC</b>	Digital-to-Analog Converter
<b>DC</b>	Direct Current
<b>DDR</b>	Double Data Rate
<b>DDMS</b>	Dalvik Debug Monitor Server
<b>DM</b>	Differential Mode
<b>DMA</b>	Direct Memory Access
<b>DNM</b>	Do Not Mount
<b>DSL</b>	Domain Specific Language
<b>DSP</b>	Digital Signal Processor
<b>ECU</b>	Engine Control Unit
<b>EEPROM</b>	Electrically Erasable Programmable Read-Only Memory





<b>EN</b>	Enable
<b>ENOB</b>	Effective number of bits
<b>EMC</b>	Electromagnetic Compatibility
<b>EMI</b>	Electromagnetic Interference
<b>ESD</b>	Electrostatic Discharge
<b>ESP</b>	Electronic Stability Program
<b>ESR</b>	Equivalent Series Resistance
<b>FFCOMP</b>	Flex Fuel Compensation
<b>FFT</b>	Fast Fourier transform
<b>FIFO</b>	First In First Out
<b>FILO</b>	First In Last Out
<b>FIR</b>	Finite impulse response
<b>FPGA</b>	Field Programmable Gate Array
<b>FPS</b>	Frames Per Second
<b>FTS</b>	Fuel Temperature Sensor
<b>FSBL</b>	First Stage Bootloader
<b>GBW</b>	Gain Bandwidth
<b>GND</b>	Ground
<b>GPIO</b>	General Purpose Input/Output
<b>GPLSS</b>	General Purpose Low Side Switch
<b>GPS</b>	Global Positioning System
<b>GS</b>	Gate-Source
<b>HREF</b>	Hyperlink Reference
<b>HTML</b>	HyperText Markup Language
<b>HTTP</b>	HyperText Transfer Protocol
<b>Hz</b>	Hertz
<b>I2C</b>	Inter-Integrated Circuit
<b>IC</b>	Integrated Circuit
<b>ICE</b>	Internal Combustion Engine
<b>IEC</b>	International Electrotechnical Commission
<b>IEEE</b>	Institute of Electrical and Electronics Engineers
<b>IGBT</b>	Insulated-gate bipolar transistor
<b>INT</b>	Interrupt
<b>I/O</b>	Input/Output
<b>IP</b>	Intellectual property
<b>IP</b>	Internal Protection Rating
<b>ISO</b>	International Organization for Standardization
<b>JTAG</b>	Joint Test Action Group
<b>JSON</b>	JavaScript Object Notation
<b>KDA</b>	Kongsberg Defence & Aerospace



<b>KPEC</b>	Kongsberg Programmable Engine Control
<b>LDO</b>	Low-Dropout Regulator
<b>LED</b>	Light Emitting Diode
<b>LGA</b>	Land Grid Array
<b>LSB</b>	Least Significant Bit
<b>LVMOS</b>	Low-Voltage Complementary Metal-Oxide Semiconductor
<b>LVDS</b>	Low-Voltage Differential Signalling
<b>LVTTL</b>	Low-Voltage Transistor-Transistor Logic
<b>M2M</b>	Machine to Machine
<b>MAF</b>	Manifold Air Flow
<b>MAP</b>	Manifold Air Pressure
<b>MEMS</b>	Micromechanical Systems
<b>MIO</b>	Multiplexed Input/Output
<b>MOSFET</b>	Metal-Oxide-Semiconductor Field-Effect Transistor
<b>MSB</b>	Most-Significant-bit
<b>MSOP</b>	Mini Small Outline Package
<b>MUX</b>	Multiplexer
<b>N/A</b>	Not Applicable
<b>NC</b>	No-Connect
<b>NOSEQ</b>	No sequencing
<b>NTC</b>	Negative Temperature Coefficient
<b>OS</b>	Operating System
<b>OTG</b>	On The Go
<b>P&amp;H</b>	Peak and hold
<b>PCA</b>	Printed Circuit Assembly
<b>PCB</b>	Printed Circuit board
<b>PGOOD</b>	Power Good
<b>PL</b>	Programmable Logic
<b>PLL</b>	Phase-Locked Loop
<b>PHY</b>	Physical layer
<b>PPP</b>	Peak Pressure Point
<b>PS</b>	Processing System
<b>PSI</b>	Pounds per square inch
<b>PVM</b>	Pneumatic Valve Module
<b>PWM</b>	Pulse Width Modulation
<b>QSPI</b>	Quad Serial Peripheral Interface
<b>RAM</b>	Random Access Memory
<b>RC</b>	Resistor-Capacitor
<b>RDS(ON)</b>	Resistance Drain-to-Source Switch on
<b>RISC</b>	Reduced Instruction Set Computing



<b>RoHS</b>	Restrictions of Hazardous Substances
<b>ROM</b>	Read-Only Memory
<b>ROOTFS</b>	Root filesystem
<b>RPM</b>	Revolutions per minute
<b>RTC</b>	Real Time Clock
<b>RTD</b>	Resistance Temperature Detector
<b>RX</b>	Receive
<b>SAE</b>	Society of Automotive Engineers
<b>SAR</b>	Successive approximation register
<b>SCL</b>	Serial Clock
<b>SCK</b>	Serial Clock
<b>SDA</b>	Serial Data
<b>SDI</b>	Serial Data In
<b>SDK</b>	Software Development Kit
<b>SDO</b>	Serial Data Out
<b>SIGMII</b>	Serial Gigabit Media Independent Interface
<b>SINAD</b>	Signal to noise and distortion ratio
<b>SMD</b>	Surface Mount Device
<b>SMPS</b>	Switch-Mode Power Supplies
<b>SMT</b>	Surface Mount Technology
<b>SOA</b>	Safe Operating Area
<b>SoC</b>	System on a Chip
<b>SOIC</b>	Small Outline Integrated Circuit
<b>SOT</b>	Small Outline Transistor
<b>SPI</b>	Serial Peripheral Interface
<b>SPDY</b>	Pronounced Speedy
<b>SPICE</b>	Simulation Program with Integrated Circuit Emphasis
<b>SPs</b>	Samples Per second
<b>SS</b>	Slave Select
<b>TAP</b>	Test Access Port
<b>TCK</b>	Test Clock
<b>TDC</b>	Top Dead Centre
<b>TDI</b>	Test Data In
<b>TDO</b>	Test Data Out
<b>TI</b>	Texas instruments
<b>TMS</b>	Test Mode Select
<b>TPS</b>	Throttle Position Sensor
<b>TRST</b>	Test Reset
<b>TSV</b>	Tab-Separated Values
<b>TTC</b>	Triple Timer Counter



<b>TTL</b>	Transistor-Transistor Logic
<b>TVS</b>	Transient Voltage Suppressor
<b>TX</b>	Transmit
<b>UART</b>	Universal Asynchronous Receiver/Transmitter
<b>U-BOOT</b>	Universal Bootloader
<b>URL</b>	Uniform Resource Locator
<b>USB</b>	Universal Serial Bus
<b>UVLO</b>	Under-voltage lockout
<b>VCC</b>	Voltage Collector-Collector
<b>VDD</b>	Voltage Drain-Drain
<b>VEE</b>	Voltage Emitter-Emitter
<b>VHDL</b>	Very High Speed Integrated Circuit Hardware Description Language
<b>VSS</b>	Voltage Source-Source
<b>VR</b>	Variable Reluctance
<b>Wi-Fi</b>	Wireless Fidelity
<b>WLAN</b>	Wireless Local Area Network
<b>WP</b>	Write Protect
<b>QFN</b>	Quad Flat No-lead
<b>QFP</b>	Quad Flat Package
<b>XML</b>	Extensible Markup Language



## Appendix II - Power budget

Table 60 - Power budget 3.3 V

Part number	Description	QTY	3.3 V (mA)		
			MIN	TYP	MAX
FT2232HL-R	Dual channel USB controller	1	0	100	150
J0G-0001NL	1Gbps Ethernet connector	1	0	8	12
LTC2983ILX#PBF	20ch temperature interface	1	0	15	20
MS5611-01BA03-50	Barometer	1	0	1.4	2.5
LSM9DS0TR	Accelero-, gyro-, magnetometer	1	0	6.5	10
SN65HVD234	ESD protected CAN transceiver	1	0	6	10
MRF24WG0MB-I/RM	WiFi module	1	0	237	300
MC33926PNB	H-bridge, Control and Power Stage	1	0	20	0
TLMx1100	Various LEDs	2	0	5	0
TE0720	Zynq micromodule	1	47	925	1849
TPIC8101	Knock sensor interface	1	0	0	20
SN74LVC1G125	Single Input buffer	1	0	100	150
74LCX125	quad input buffer	2	0	100	150
ADS8568	ADC PP	1	0	0.5	2
AT93C46E	EEPROM	1	0	0.5	2
10k	Pull-up	40	0	0.33	0.33
SUM		57	47	1643.1	2840.7

Table 61 - Power budget 5 V

Part number	Description	QTY	5 V (mA)		
			MIN	TYP	MAX
LM5112	Ignition coil driver	24	0	1	2
PSS-10	Fluid Pressure sensor	2	0	8	12
PSA-C	Manifold air pressure sensor	1	0	9	14
MAX9924UAB+	VR interface IC	1	0	2.6	5
Mini-HA-P	Hall-Effect Speed Sensor Mini-HA-P	4	0	10	15
LT1812IS5	Operational amp	50	0	2.3	4.6
ADS8568	ADC PP	1	0	36.6	48.4
LTC2050HV	Lambda	16	0	1	1.6
DAC124S085C1MM	DAC for pump current ctr	1	0	0.36	0.68
CD74HC4052	Multiplexer	2	0	50	75
10k	Pull-up	40	0	0.5	0.5
PSI-A	Pressure sensor	8	0	50	85
SUM		150	15.6	780.16	1305.68

Table 62 - Power budget -5 V

Part number	Description	QTY	-5 V (mA)		
			MIN	TYP	MAX
LT1812IS5	Operational amplifier	38	0	0	4.6
LTC2050HV	Operational amplifier	16	0	1	1.6
SUM		144	0	16	246



Table 63 - Power budget 15 V

Part number	Description	QTY	15 V (mA)		
			MIN	TYP	MAX
ADS8568	ADC	1	0	2.3	3.2
LT1468	Operational amplifier	2	0	3.9	5.2
<b>SUM</b>		<b>144</b>	<b>0</b>	<b>10.1</b>	<b>13.6</b>

Table 64 - Power budget -15 V

Part number	Description	QTY	-15 V (mA)		
			MIN	TYP	MAX
ADS8568	ADC	1	0	2.7	3.6
LT1468	Operational amplifier	2	0	3.9	5.2
<b>SUM</b>		<b>3</b>	<b>0</b>	<b>10.5</b>	<b>14</b>

NB! Power budget has not been properly scaled. DCDC converters are designed to 1.5 times  $I_{MAX}$ .

Table 65 - Cold crank power budget

V	A	W	$W_{EFF}$	$A_{6V}$	$A_{12V}$
<b>3.3</b>	4.5	14.85	16.32	2.72	1.36
<b>5</b>	3	15	17.1	2.84	1.42
<b>-5</b>	0.6	3	3.41	0.57	0.28
<b>15</b>	0.025	0.375	0.43	0.071	0.04
<b>-15</b>	0.025	0.375	0.43	0.071	0.04
<b>SUM</b>	<b>8.15</b>	<b>33.6</b>	<b>37.62</b>	<b>6.27</b>	<b>3.13</b>

Table 66 - Parameter explanation for table 6

<b>V</b>	<b>DCDC output voltage</b>
<b>A</b>	DCDC output current
<b>W</b>	DCDC output power
$W_{EFF}$	DCDC input power
$A_{6V}$	DCDC input current at 6V supply
$A_{12V}$	DCDC input current at 12V supply



## Appendix III - Recommended automotive sensors, actuators and parts

Table 67 gives an overview of the recommended parts used for this design.

**Table 67 - Recommended sensors, actuators and parts for KPEC ECU.**

QTY	SCHEMATIC ABBREVIATION	DESCRIPTION	TYPE	MODEL	MANUFACTURER	MPN
2	LAMBDA	Lambda sensor	Analog sensor	LSU 4.9	BOSCH	0 258 017 025
8	CPS	Cylinder pressure sensor	Analog sensor	-	OPTRAND	AUTOPSI
1	FPS	Fuel pressure sensor	Analog sensor	PSS-10	BOSCH	B 261 209 341
1	MAP	Manifold air pressure sensor	Analog sensor	PSA-C	BOSCH	0 281 002 389
1	OPS	Oil pressure sensor	Analog sensor	PSS-10	BOSCH	B 261 209 341
8	EXTS	Exhaust temperature sensor	Analog sensor	TCP-K	BOSCH	B 261 209 385
1	FTS	Fuel temperature sensor	Analog sensor	NTC-M12-H	BOSCH	0 281 002 170
1	OTS	Oil temperature sensor	Analog sensor	NTC-M12-H	BOSCH	0 281 002 170
1	CTS	Coolant temperature sensor	Analog sensor	NTC-M12-H	BOSCH	0 281 002 170
2	KNOCK	Knock sensor	Analog sensor	KS-P	BOSCH	0 261 231 120
1	ACC	Accelerator pedal position sensor	Analog sensor	-	BOSCH	0 280 752 214
1	CRANK	Flywheel sensor	Analog sensor	IA-C	BOSCH	0 261 210 136
4	CAM	Camshaft sensor	Digital sensor	HA-P	BOSCH	1 234 482 092
1	MAF	Manifold air flow sensor	Digital sensor	HFM 6	BOSCH	0 281 002 764
1	FFS	Flex fuel sensor	Digital sensor	-	GM	12570260
8	INJ	Fuel injectors	PWM Actuator	EV14	BOSCH	0 280 158 XXX
8	IGN	Ignition coils	PWM Actuator	S19	BOSCH	0 221 B00 113-02
1	BOOST	Boost controller	PWM Actuator	TCV	PIERBURG	VO 30670448
1	THROTTLE	Throttle body	PWM Actuator	ETM-82	BOSCH	0 280 750 101
1	FUELPUMP	Fuel pump	Actuator	FP200	BOSCH	0 580 254 044
1	IGN_SW	NO ignition switch	Switch		WÜRTH	-
1	BRK_SW	NO brake pedal switch	Switch		AFI	SKU 9997-SW9018
10	FUSE	Fuse with	Fuse		WÜRTH	-



		holders (0.5A-175A)			
2	RELAYS	Relay (200A/30A)	Relay	WÜRTH	-
100		Wires (0.75mm, 2.5mm, 5mm)	Wire	WÜRTH	-
10		Accessories (tape, connectors)		WÜRTH	-
2		Lambda connector	Connector	BOSCH	1928404682 / D261205356-01
2		Pressure sensor connector	Connector	BOSCH	D 261 205 339-1
1		MAP connector	Connector	BOSCH	D 261 205 289-01
8		Exhaust temperature connector	Connector	BOSCH	B 261 209 159-01
3		Temperature sensor connector	Connector	BOSCH	D 261 205 337-01
2		Knock sensor connector	Connector	BOSCH	D 261 205 337-01
1		Flywheel sensor Connector	Connector	BOSCH	D 261 205 335-01
4		Camshaft sensor connector	Connector	BOSCH	F 02U B00 555-01
1		Manifold air flow sensor connector	Connector	BOSCH	1 928 403 836
1		Throttle body connector	Connector	BOSCH	D 261 205 358-01





## Appendix IV - Signal names and description

**Table 68 - Power signal description**

Powers	
<b>VBAT</b>	Protected car battery voltage; variable source at 12 V nominal, ECU operational for ranges 6 - 18 V.
<b>VBAT_UP</b>	Unprotected car battery input voltage
<b>3V3</b>	3.3 V $\pm 5\%$
<b>5V0</b>	5.0 V $\pm 5\%$
<b>N5V0</b>	Negative 5.0 V $\pm 5\%$ referenced to GND.
<b>15V0</b>	15.0 V $\pm 5\%$
<b>N15V0</b>	Negative 15.0 V $\pm 5\%$ referenced to GND.
<b>GND</b>	Ground reference.
<b>3.3VIN</b>	TE0720 3.3 V supply input
<b>VIN</b>	TE0720 supply input, Single 3.3 V supply mode used.
<b>VCCIO13</b>	TE0720 FPGA bank 13 supply, connected to 3.3 V.
<b>VCCIO33</b>	TE0720 FPGA bank 33 supply, connected to 3.3 V.
<b>VCCIO34</b>	TE0720 FPGA bank 34 supply, connected to 3.3 V.
<b>VCCIO35</b>	TE0720 FPGA bank 35 supply, connected to 3.3 V.
<b>VCCO_MIO0_500</b>	TE0720 PS bank 500 supply, internally connected to 3.3 V

**Table 69 - XADC channel connections**

XADC Connections	
XADC channel	Signal name
0	BAT_VOLTAGE
1	THR_POS_1
2	MAP_OUT
3	ACC_SIGNAL_1
4	GPAI_1
5	GPAI_0
6	FUEL_PRESSURE_OUT
7	LAMBDA_FOUT_3
8	GPAI_2
9	THR_POS_0
10	THROTTLE_VFB
11	ACC_SIGNAL_0
12	LAMBDA_FOUT_2
13	LAMBDA_FOUT_1
14	OIL_PRESSURE_OUT
15	LAMBDA_FOUT_0



Table 70 - Signal description list

Signals	
Signal net name	Description
ACC_IN_[0-1]	ECU connector accelerator pedal signals.
ACC_N_R_[0-1]	Accelerator pedal XADC negative differential input.
ACC_P_R_[0-1]	Accelerator pedal XADC positive differential input.
ACC_SIG_[0-1]	Accelerator pedal analog input connected to XADC channels 3 and 11.
ANALOG_OUT_[0-1]	ECU connector General Purpose Analog output signals.
BAT_VOLTAGE	Battery voltage analog input, connected to XADC channel 0.
BAT_VOLTAGE_R_N	Battery voltage XADC negative differential input.
BAT_VOLTAGE_R_P	Battery voltage XADC positive differential input.
BOOST_CTR	ECU connector boost controller signal.
BOOST_CTR_SIG	Signal for low side switch for activating boost controller.
BRAKE_SIG_IN	ECU connector brake pedal signal.
BRAKE_SIG_ZYNQ	Brake pedal digital signal input, high if brake pedal activated.
CAN_EN	CAN transceiver enable signal.
CAN_H	ECU connector CAN Bus High signal.
CAN_L	ECU connector CAN Bus Low signal.
CAN_RX	CAN Bus transceiver receive signal output.
CAN_TX	CAN Bus transceiver transmit signal input.
CPS_BUSY_INT	Cylinder Pressure sensor ADC busy status and interrupt output.
CPS_CONVST	Cylinder Pressure sensor ADC conversation start signal on all channels.
CPS_DB_[0-15]	Cylinder Pressure sensor ADC 16 bit output data bus.
CPS_IN_[0-7]	ECU connector cylinder pressure signals.
CPS_nHW_SW	Cylinder Pressure sensor ADC hardware/software mode select.
CPS_nSTBY	Cylinder Pressure sensor ADC standby mode input, ADC is powered down when input is low.
CPS_RANGE_XCLK	Peak Pressure sensor ADC; HW mode: Analog voltage input range select. SW mode: External clock input.
CPS_REFEN_nWR	Cylinder Pressure sensor ADC internal/external reference select.
CPS_REFIO	Cylinder Pressure sensor ADC analog reference input/output.
CPS_RESET	Cylinder Pressure sensor ADC active high reset input.
EXHT_N_[0-7]	ECU connector negative side thermocouple exhaust temperature signals.
EXHT_P_[0-7]	ECU connector positive side thermocouple exhaust temperature signals.
FLEX_FUEL_IN	ECU connector flex fuel sensor signal.
FLEX_FUEL_ZYNQ	Digital flex fuel sensor input. 5V tolerant input buffer can be used as GPI if not used for flex fuel sensor.
FPGA_LED_[0-3]	On-board LED control signals, pull low to light LEDs
FUEL_PRESS_IN	ECU connector fuel pressure signal.
FUEL_PRESS_N_R	Fuel pressure sensor XADC negative differential input.
FUEL_PRESS_P_R	Fuel pressure sensor XADC negative positive input.
FUEL_PRESS_XADC	Analog fuel pressure sensor input, connected to XADC channel 6.



<b>FUEL_PUMP</b>	ECU connector fuel pump signal.
<b>FUEL_PUMP_SIG</b>	Signal for low side switch for activating fuel pump.
<b>GPAI_[0-2]</b>	General Purpose Analog Inputs, connected to XADC channels 4, 5 and 8.
<b>GPAI_N_R_2</b>	General Purpose Analog Input XADC negative differential input.
<b>GPAI_P_R_2</b>	General Purpose Analog Input XADC positive differential input.
<b>GPAI_IN_[0-2]</b>	ECU connector General Purpose Analog Input signals.
<b>GPDI0_DIR_[0-1]</b>	Direction control signal to voltage translator
<b>GPDI0_IN_[0-3]</b>	General purpose inputs/outputs for 3.3 V to 5 V
<b>GPDI0_ZYNQ_[0-3]</b>	General purpose inputs/outputs for 3.3 V to 5.5 V
<b>GPI_IN_[0-1]</b>	ECU connector General Purpose digital input signals.
<b>GPI_ZYNQ_[0-1]</b>	General Purpose Digital inputs, 5 V input tolerant buffers, 3.3 V towards TE0720.
<b>GPLSS_[0-3]</b>	ECU connector General Purpose Low Side Switch signals.
<b>GPLSS_EN</b>	Active high signal for enabling all general purpose low side switch output pre driver stages.
<b>GPLSS_SIG_[0-3]</b>	Signal for low side switch for activating general purpose low side switch outputs.
<b>HALL_IN_[0-3]</b>	ECU connector camshaft hall sensor signals.
<b>HALL_ZYNQ_[0-3]</b>	Hall sensor interface output signals carrying camshaft position data.
<b>HEAT_L_[1-2]</b>	ECU connector lambda heater signals.
<b>HEAT_SIG_[1-2]</b>	Signal for low side switch for activating lambda heater elements.
<b>I2C_SDA</b>	I2C bus Serial Data line.
<b>I2C_SCL</b>	I2C bus Serial Clock signal.
<b>IGN_EN</b>	Active high enable signal for all ignition coil low side switch pre driver stages.
<b>IGN_[0-7]</b>	ECU connector ignition coil signals.
<b>IGN_SIG_[0-7]</b>	Signal for low side switch for activating spark plug coils.
<b>IGN_SIG_IN</b>	ECU connector ignition ON signal.
<b>IGN_SIG_ZYNQ</b>	Ignition ON signal input, optocoupled 12 V input buffer.
<b>INJ_[0-7]</b>	ECU connector fuel injector signals.
<b>INJ_SIG_[0-7]</b>	Signals for low side switches for activating fuel injectors.
<b>IP_[1-2]</b>	ECU connector lambda pump cell current signals.
<b>JTAG_EN</b>	JTAG enable signal. JTAG is enabled when pulled low. Value can be changed with onboard jumper.
<b>JTAG_RESET</b>	JTAG reset input signal, triggers global reset of all board components including TE0720.
<b>JTAG_VREF</b>	JTAG voltage reference signal, 3.3 V from TE0720.
<b>KNOCK_CLK</b>	Knock sensor interface clock input.
<b>KNOCK_IN_[0-1]</b>	ECU connector Knock sensor signals.
<b>LAMBDA_XADC_[0-3]</b>	Lambda interface analog multiplexer outputs, connected to XADC channels 7, 12, 13 and 15.
<b>LAMBDA_XADC_N_R_[0-3]</b>	Lambda interface analog MUX outputs XADC negative differential input.
<b>LAMBDA_XADC_P_R_[0-3]</b>	Lambda interface analog MUX outputs XADC positive differential input.
<b>MAF_IN</b>	ECU connector Manifold Air Flow signal.



<b>MAF_ZYNQ</b>	Digital Manifold Air flow sensor input.
<b>MAP_IN</b>	ECU connector Manifold Air Pressure signal.
<b>MAP_N_R</b>	MAP sensor XADC negative differential input.
<b>MAP_P_R</b>	MAP sensor XADC positive differential input.
<b>MAP_XADC</b>	Manifold Air Pressure analog input, connected to XADC channel 2.
<b>MPU_FSYNC</b>	Motion Processing Unit Frame Synchronization input signal.
<b>MPU_INT</b>	Motion Processing Unit interrupt output signal.
<b>MUXA_1</b>	Lambda 1 analog MUX line select S0.
<b>MUXA_2</b>	Lambda 2 analog MUX line select S1.
<b>MUXB_1</b>	Lambda 1 analog MUX line select S0.
<b>MUXB_2</b>	Lambda 2 analog MUX line select S1.
<b>NC</b>	Not Connected, marked as a cross on part pins in schematic.
<b>NOSEQ</b>	No sequencing of TE0720 supplies due to 3.3 V single supply. Pulled high to 3.3 V.
<b>NTC_N_[0-3]</b>	ECU connector negative side NTC thermistor temperature signals.
<b>NTC_P_[0-3]</b>	ECU connector positive side NTC thermistor temperature signals.
<b>OIL_PRESS_IN</b>	ECU connector oil pressure signal.
<b>OIL_PRESS_N_R</b>	Oil pressure sensor XADC negative differential input.
<b>OIL_PRESS_P_R</b>	Oil pressure sensor XADC positive differential input.
<b>OIL_PRESS_XADC</b>	Oil Pressure analog input, connected to XADC channel 14.
<b>PHY_LED_N_0</b>	Ethernet jack yellow LED, pull low for activation.
<b>PHY_LED_N_1A</b>	Ethernet jack dual LED orange side, pull low for activation. NB: Do not pull low at same time as green LED
<b>PHY_LED_N_1B</b>	Ethernet jack dual LED green side, pull low for activation. NB: Do not pull low at same time as orange LED
<b>PHY_MDI_P_[0-3]</b>	Positive side of gigabit Ethernet PHY differential lines.
<b>PHY_MDI_N_[0-3]</b>	Negative side of gigabit Ethernet PHY differential lines.
<b>PROGLED</b>	Onboard programmable LED control signal. Pull low to light LEDs. Connected to MIO pin 0.
<b>PS_UART_RX_0</b>	PS UART0 receive input.
<b>PS_UART_TX_0</b>	PS UART0 transmit output.
<b>RTC_BAT</b>	Positive battery voltage from coin cell battery for maintaining Real Time Clock while non-powered.
<b>SPI_0_CS_0</b>	SPI 0 chip select. Connected to CS pin on Wi-Fi module.
<b>SPI_0_CS_1</b>	SPI 0 chip select. Connected to CS pin on knock sensor interface.
<b>SPI_0_MISO</b>	SPI 0 Master Input Slave Output.
<b>SPI_0_MOSI</b>	SPI 0 Master Output Slave Input.
<b>SPI_0_SCK</b>	SPI 0 Serial Clock.
<b>SPI_1_CS</b>	SPI 1 chip select. Connected to CS pin on Wi-Fi module.
<b>SPI_1_MISO</b>	SPI 1 Master Input Slave Output.
<b>SPI_1_MOSI</b>	SPI 1 Master Output Slave Input.
<b>SPI_1_SCK</b>	SPI 1 Serial Clock.
<b>SPI_2_CS</b>	SPI 2 chip select. Connected to CS pin on Lambda DAC.



<b>SPI_2_MOSI</b>	SPI 2 Master Output Slave Input, Note that Lambda DAC is 3-wire SPI with no MISO signal.
<b>SPI_2_SCK</b>	SPI 2 Serial Clock.
<b>TCK</b>	JTAG Test Clock.
<b>TDI</b>	JTAG Test Data Input.
<b>TDO</b>	JTAG Test Data Output.
<b>TE0720_PGOOD</b>	TE0720 Power good signal. Connected to a green LED on the ECU motherboard that lights when all DCDC converters on TE0720 are good.
<b>TEMP_INT</b>	Temperature interface LTC2983 interrupt output.
<b>TEMP_RST</b>	Temperature interface LTC2983 active low reset input.
<b>THR_CFB</b>	Throttle H-bridge controller current output feedback signal, 0.24% of H-bridge high side output current.
<b>THR_FB_[0-1]</b>	ECU connector throttle body feedback signals.
<b>THR_IN_[1-2]</b>	Throttle body H-bridge input control signals. Each signal controls corresponding output. Forward mode: 1=HIGH, 2=LOW. Reverse mode: 1=LOW, 2=HIGH.
<b>THR_IN_EN</b>	Active high enable signal for throttle body H-bridge.
<b>THR_OUT_[1-2]</b>	ECU connector throttle body H-bridge driver signals.
<b>THR_POS_[0-1]</b>	Throttle body position feedback signals, connected to XADC channels 1 and 9.
<b>THR_POS_N_R_[0-1]</b>	Throttle position XADC negative differential input.
<b>THR_POS_P_R_[0-1]</b>	Throttle position XADC positive differential input.
<b>THR_SF</b>	Throttle body H-bridge controller status flag input, signal low if any faults with throttle H-bridge controller.
<b>THR_VFB</b>	Throttle H-bridge controller current output voltage feedback signal, connected to XADC channel 10.
<b>THR_VFB_N_R</b>	Throttle feedback XADC negative differential input.
<b>THR_VFB_P_R</b>	Throttle feedback XADC positive differential input.
<b>TMS</b>	JTAG Test Mode Select.
<b>VBUS_DETECT</b>	USB Host connected if VBUS is high. USB controller is connected to PS UART0 and WiFi UART debug.
<b>VGND_[1-2]</b>	ECU connector lambda virtual ground signals.
<b>VR_IN_H</b>	ECU connector crankshaft variable reluctance sensor high signal.
<b>VR_IN_L</b>	ECU connector crankshaft variable reluctance sensor low signal.
<b>VR_ZYNQ</b>	Variable reluctance sensor interface output carrying crankshaft speed and position data.
<b>VS_[1-2]</b>	ECU connector lambda Nernst cell voltage signals.
<b>WIFI_DEBUG_RX</b>	WiFi module debug UART interface receive input.
<b>WIFI_DEBUG_TX</b>	WiFi module debug UART interface transmit output.
<b>WIFI_HIBERNATE</b>	WiFi module sleep mode signal, sleep mode active when signal input is HIGH.
<b>WIFI_INT</b>	WiFi module interrupt output, open drain output, pulls LOW to signal interrupt.
<b>WIFI_RST</b>	WiFi module reset input signal, connected to global reset switch.
<b>WIFI_WP</b>	WiFi module internal flash memory Write Protect, used to update firmware stored in flash.



## Appendix V - Board-to-Board connector tables

**Table 71 - Connector top pinout**

Pin	Net	Type	Bank	FPGA	Schematic net name
1	VIN	Power input			3V3
3	VIN	Power input			3V3
5	VIN	Power input			3V3
7	VIN	Power input			3V3
9	3.3V	O			NC
11	3.3V	O			NC
13	B33_L4_P	DIFFIO	33	W20	FPGA_LED_0
15	B33_L4_N	DIFFIO	33	W21	FPGA_LED_1
17	RESIN	Reset input			JTAG_RESET
19	GND				GND
21	B33_L13_P	DIFFIO_CC	33	W17	KNOCK_CLK
23	B33_L13_N	DIFFIO_CC	33	W18	VR_ZYNQ
25	B33_L14_P	DIFFIO_CC	33	W16	FPGA_LED_2
27	B33_L14_N	DIFFIO_CC	33	Y16	FPGA_LED_3
29	GND				GND
31	B13_L5_P	DIFFIO	13	U12	HALL_ZYNQ_0
33	B13_L5_N	DIFFIO	13	U11	HALL_ZYNQ_1
35	B13_L6_P	DIFFIO	13	U10	HALL_ZYNQ_2
37	B13_L6_N	DIFFIO	13	U9	HALL_ZYNQ_3
39	GND				GND
41	B13_L1_P	DIFFIO	13	V10	SPI_1_MOSI
43	B13_L1_N	DIFFIO	13	V9	SPI_1_MISO
45	B13_L12_P	DIFFIO_CC	13	Y9	SPI_1_SCK
47	B13_L12_N	DIFFIO_CC	13	Y8	SPI_1_CS
49	GND				GND
51	B13_L14_P	DIFFIO_CC	13	AA7	SPI_2_MOSI
53	B13_L14_N	DIFFIO_CC	13	AA6	NC
55	B13_L13_P	DIFFIO_CC	13	Y6	SPI_2_SCK
57	B13_L13_N	DIFFIO_CC	13	Y5	SPI_2_CS
59	GND				GND



61	B13_L4_P	DIFFIO	13	V12	CPS_DB_0
63	B13_L4_N	DIFFIO	13	W12	CPS_DB_1
65	B13_L3_P	DIFFIO	13	W11	CPS_DB_2
67	B13_L3_N	DIFFIO	13	W10	CPS_DB_3
69	GND				GND
71	B13_L10_P	DIFFIO	13	Y11	CPS_DB_4
73	B13_L10_N	DIFFIO	13	Y10	CPS_DB_5
75	B13_L2_P	DIFFIO	13	V8	CPS_DB_6
77	B13_L2_N	DIFFIO	13	W8	CPS_DB_7
79	GND				GND
81	B13_L23_P	DIFFIO	13	V7	CPS_DB_8
83	B13_L23_N	DIFFIO	13	W7	CPS_DB_9
85	B13_L24_P	DIFFIO	13	W6	CPS_DB_10
87	B13_L24_N	DIFFIO	13	W5	CPS_DB_11
89	GND				GND
91	B13_L19_P	DIFFIO	13	R6	CPS_DB_12
93	B13_L19_N	DIFFIO	13	T6	CPS_DB_13
95	B13_L22_P	DIFFIO	13	U6	CPS_DB_14
97	B13_L22_N	DIFFIO	13	U5	CPS_DB_15
99	B13_IO0	IO	13	R7	CPS_BUSY_INT
2	VCCIO34	I/O Supply	34		3V3
4	VCCIO34	I/O Supply	34		3V3
6	VCCIO33	I/O Supply	33		3V3
8	VCCIO13	I/O Supply	13		3V3
10	VCCIO13	I/O Supply	13		3V3
12	B33_L7_P	DIFFIO	33	AA22	NC
14	B33_L7_N	DIFFIO	33	AB22	NC
16	B33_L8_P	DIFFIO	33	AA21	NC
18	B33_L8_N	DIFFIO	33	AB21	NC
20	1.5V	O			NC
22	B33_L11_P	DIFFIO_CC	33	Y19	NC
24	B33_L11_N	DIFFIO_CC	33	AA19	NC



26	B33_L12_P	DIFFIO_CC	33	Y18	SPI_0_SCK
28	B33_L12_N	DIFFIO_CC	33	AA18	SPI_0_MOSI
30	B33_VREF	IO_VREF	33	V19/V15	NC
32	B33_L17_P	DIFFIO	33	AA17	SPI_0_MISO
34	B33_L17_N	DIFFIO	33	AB17	SPI_0_CS_0
36	B33_L18_P	DIFFIO	33	AA16	SPI_0_CS_1
38	B33_L18_N	DIFFIO	33	AB16	TEMP_INT
40	GND				GND
42	B13_L7_P	DIFFIO	13	AA12	NC
44	B13_L7_N	DIFFIO	13	AB12	WIFI_INT
46	B13_L8_P	DIFFIO	13	AA11	WIFI_WP
48	B13_L8_N	DIFFIO	13	AB11	WIFI_HIBERNATE
50	GND				GND
52	B13_L11_P	DIFFIO_CC	13	AA9	I2C_SCL
54	B13_L11_N	DIFFIO_CC	13	AA8	I2C_SDA
56	B13_L9_P	DIFFIO	13	AB10	MPU_FSYNC
58	B13_L9_N	DIFFIO	13	AB9	MPU_INT
60	GND				GND
62	B13_L20_P	DIFFIO	13	T4	CAN_TX
64	B13_L20_N	DIFFIO	13	U4	CAN_RX
66	B13_L17_P	DIFFIO	13	AB7	CAN_VALVE_TX
68	B13_L17_N	DIFFIO	13	AB6	CAN_VALVE_RX
70	GND				GND
72	B13_L16_P	DIFFIO	13	AB5	CAN_EN
74	B13_L16_N	DIFFIO	13	AB4	CAN_VALVE_EN
76	B13_L18_P	DIFFIO	13	Y4	CPS_RANGE_XCLK
78	B13_L18_N	DIFFIO	13	AA4	CPS_REFIO
80	GND				GND
82	B13_L15_P	DIFFIO	13	AB2	CPS_REFEN_nWR
84	B13_L15_N	DIFFIO	13	AB1	CPS_nHW_SW
86	B13_L21_P	DIFFIO	13	V5	CPS_nSTBY
88	B13_L21_N	DIFFIO	13	V4	CPS_CONVST
90	B13_IO25	IO	13	U7	CPS_RESET





92	VREF_JTAG	O 3.3V			JTAG_VREF
94	TMS	JTAG	0		TMS
96	TDI	JTAG	0		TDI
98	TDO	JTAG	0		TDO
100	TCK	JTAG	0		TCK

**Table 72 - Connector bottom pinout**

Pin	Net	Type	Bank	FPGA	Schematic net name
1	GND				GND
3	PHY_MDI0_P	DIFFIO			PHY_MDI_P_0
5	PHY_MDI0_N	DIFFIO			PHY_MDI_N_0
7	GND				GND
9	PHY_MDI1_P	DIFFIO			PHY_MDI_P_1
11	PHY_MDI1_N	DIFFIO			PHY_MDI_N_1
13	NC				GND
15	PHY_MDI2_P	DIFFIO			PHY_MDI_P_2
17	PHY_MDI2_N	DIFFIO			PHY_MDI_N_2
19	GND				GND
21	PHY_MDI3_P	DIFFIO			PHY_MDI_P_3
23	PHY_MDI3_N	DIFFIO			PHY_MDI_N_3
25	GND				GND
27	EN1	I			NC
29	PGOOD	O			TE0720_PGOOD
31	MODE	IO			NC
33	GND				GND
35	B35_L10_N	DIFFIO_ADC	35	A19	ACC_N_R_0
37	B35_L10_P	DIFFIO_ADC	35	A18	ACC_P_R_0
39	B35_L9_N	DIFFIO_ADC	35	A17	ACC_N_R_1
41	B35_L9_P	DIFFIO_ADC	35	A16	ACC_P_R_1
43	GND				GND
45	B35_L7_N	DIFFIO_ADC	35	B15	MAP_N_R
47	B35_L7_P	DIFFIO_ADC	35	C15	MAP_P_R



49	B35_L2_N	DIFFIO_ADC	35	D17	GPAI_N_R_2
51	B35_L2_P	DIFFIO_ADC	35	D16	GPAI_P_R_2
53	GND				GND
55	B35_L8_N	DIFFIO_ADC	35	B17	THR_VFB_N_R
57	B35_L8_P	DIFFIO_ADC	35	B16	THR_VFB_P_R
59	B35_L21_N	DIFFIO_ADC	35	E20	OIL_PRESS_N_R
61	B35_L21_P	DIFFIO_ADC	35	E19	OIL_PRESS_P_R
63	GND				GND
65	B35_L11_N	DIFFIO_CC	35	C18	GPI_ZYNQ_0
67	B35_L11_P	DIFFIO_CC	35	C17	GPI_ZYNQ_1
69	B35_L23_N	DIFFIO	35	F22	FLEX_FUEL_ZYNQ
71	B35_L23_P	DIFFIO	35	F21	MAF_ZYNQ
73	GND				GND
75	B35_L5_N	DIFFIO_ADC	35	E18	THR_POS_N_R_0
77	B35_L5_P	DIFFIO_ADC	35	F18	THR_POS_P_R_0
79	B35_L3_N	DIFFIO_ADC	35	D15	THR_POS_N_R_1
81	B35_L3_P	DIFFIO_ADC	35	E15	THR_POS_P_R_1
83	GND				GND
85	B35_L6_N	DIFFIO	35	F17	VBUS_DETECT
87	B35_L6_P	DIFFIO	35	G17	NC
89	JTAG_EN				JTAG_EN
91	MIO14	MIO	500	B6	PS_UART_RX_0
93	B35_L1_N	DIFFIO_ADC	35	E16	BAT_VOLTAGE_N_R
95	B35_L1_P	DIFFIO_ADC	35	F16	BAT_VOLTAGE_P_R
97	B35_L19_N	DIFFIO	35	H20	HEAT_SIG_1
99	B35_L19_P	DIFFIO	35	H19	HEAT_SIG_2
2	VIN	Power input			3V3
4	VIN	Power input			3V3
6	VIN	Power input			3V3
8	NOSEQ	input			3V3
10	VCCIO35	I/O Supply			3V3
12	VCCIO35	I/O Supply			3V3



14	3.3VIN	Power input			3V3
16	3.3VIN	Power input			3V3
18	MIO45	MIO	501	B9	NC
20	MIO44	MIO	501	E13	NC
22	MIO43	MIO	501	B11	NC
24	MIO42	MIO	501	D8	NC
26	MIO41	MIO	501	C8	NC
28	MIO40	MIO	501	E14	NC
30	GND				GND
32	B35_L16_N	DIFFIO	35	C22	NC
34	B35_L16_P	DIFFIO	35	D22	THR_SF
36	B35_L24_N	DIFFIO_ADC	35	G22	LAMBDA_XADC_N_R_0
38	B35_L24_P	DIFFIO_ADC	35	H22	LAMBDA_XADC_P_R_0
40	1.8V	O			NC
42	B35_L18_N	DIFFIO_ADC	35	B22	LAMBDA_XADC_N_R_1
44	B35_L18_P	DIFFIO_ADC	35	B21	LAMBDA_XADC_P_R_1
46	B35_L15_N	DIFFIO_ADC	35	A22	LAMBDA_XADC_N_R_2
48	B35_L15_P	DIFFIO_ADC	35	A21	LAMBDA_XADC_P_R_2
50	B35_L22_N	DIFFIO_ADC	35	G21	LAMBDA_XADC_N_R_3
52	B35_L22_P	DIFFIO_ADC	35	G20	LAMBDA_XADC_P_R_3
54	GND				GND
56	B35_L17_N	DIFFIO_ADC	35	D21	GPAI_N_R_0
58	B35_L17_P	DIFFIO_ADC	35	E21	GPAI_P_R_0
60	B35_L13_N	DIFFIO_CC	35	B20	BRAKE_SIG_ZYNQ
62	B35_L13_P	DIFFIO_CC	35	B19	IGN_SIG_ZYNQ
64	GND				GND
66	B35_L14_N	DIFFIO_CC	35	C20	GPAI_N_R_1
68	B35_L14_P	DIFFIO_CC	35	D20	GPAI_P_R_1
70	B35_L4_N	DIFFIO	35	G16	MUXA_1
72	B35_L4_P	DIFFIO	35	G15	MUXA_2
74	GND				GND
76	B35_L12_N	DIFFIO_CC	35	C19	MUXB_1
78	B35_L12_P	DIFFIO_CC	35	D18	MUXB_2



80	VBAT	VBAT input			RTC_BAT
82	B35_L20_N	DIFFIO_ADC	35	F19	FUEL_PRESS_N_R
84	B35_L20_P	DIFFIO_ADC	35	G19	FUEL_PRESS_P_R
86	MIO15	MIO	500	E6	PS_UART_TX_0
88	MIO0	MIO	500	G6	PROGLED
90	GND				GND
92	MIO9	MIO	500	C4	NC
94	MIO11	MIO	500	B4	NC
96	MIO10	MIO	500	G7	NC
98	MIO13	MIO	500	A6	NC
100	MIO12	MIO	500	C5	NC

Table 73 - Connector left pinout

Pin	Net	Type	Bank	FPGA	Schematic net name
1	SIN_N	SGMII_RX			NC
3	SIN_P	SGMII_RX			NC
5	GND				GND
7	B34_L1_P	DIFFIO	34	J15	THR_IN_1
9	B34_L1_N	DIFFIO	34	K15	THR_IN_2
11	GND				GND
13	B34_L18_P	DIFFIO	34	P20	PHY_LED_N_1A
15	B34_L18_N	DIFFIO	34	P21	PHY_LED_N_1B
17	GND				GND
19	B34_L20_P	DIFFIO	34	P17	FUEL_PUMP_SIG
21	B34_L20_N	DIFFIO	34	P18	BOOST_CTR_SIG
23	GND				GND
25	B34_L10_P	DIFFIO	34	L21	INJ_SIG_0
27	B34_L10_N	DIFFIO	34	L22	INJ_SIG_1
29	GND				GND
31	B34_L13_P	DIFFIO_CC	34	M19	INJ_SIG_2
33	B34_L13_N	DIFFIO_CC	34	M20	INJ_SIG_3
35	GND				GND



37	B34_L21_P	DIFFIO	34	T16	INJ_SIG_4
39	B34_L21_N	DIFFIO	34	T17	INJ_SIG_5
41	B34_L15_P	DIFFIO	34	M21	INJ_SIG_6
43	B34_L15_N	DIFFIO	34	M22	INJ_SIG_7
45	GND				GND
47	B34_L17_P	DIFFIO	34	R20	GPDIO_ZYNQ_0
49	B34_L17_N	DIFFIO	34	R21	GPDIO_ZYNQ_1
51	B34_L23_P	DIFFIO	34	R18	GPDIO_ZYNQ_2
53	B34_L23_N	DIFFIO	34	T18	GPDIO_ZYNQ_3
55	B34_VREF	IO_VREF	34	M16/P15	NC
57	B34_L14_P	DIFFIO_CC	34	N19	GPDIO_DIR_0
59	B34_L14_N	DIFFIO_CC	34	N20	GPDIO_DIR_1
2	SOUT_N	SGMII_TX			NC
4	SOUT_P	SGMII_TX			NC
6	GND				GND
8	B34_L7_P	DIFFIO	34	J18	THR_IN_EN
10	B34_L7_N	DIFFIO	34	K18	PHY_LED_N_0
12	GND				GND
14	B34_L2_P	DIFFIO	34	J16	GPLSS_SIG_0
16	B34_L2_N	DIFFIO	34	J17	GPLSS_SIG_1
18	GND				GND
20	B34_L4_P	DIFFIO	34	L17	GPLSS_SIG_2
22	B34_L4_N	DIFFIO	34	M17	GPLSS_SIG_3
24	GND				GND
26	B34_L5_P	DIFFIO	34	N17	IGN_SIG_0
28	B34_L5_N	DIFFIO	34	N18	IGN_SIG_1
30	GND				GND
32	B34_L12_P	DIFFIO_CC	34	L18	IGN_SIG_2
34	B34_L12_N	DIFFIO_CC	34	L19	IGN_SIG_3
36	GND				GND
38	B34_L8_P	DIFFIO	34	J21	IGN_SIG_4
40	B34_L8_N	DIFFIO	34	J22	IGN_SIG_5



42	B34_L9_P	DIFFIO	34	J20	IGN_SIG_6
44	B34_L9_N	DIFFIO	34	K21	IGN_SIG_7
46	GND				GND
48	OTG D+	DIFFIO			NC
50	OTG D-	DIFFIO			NC
52	OTG ID	I			NC
54	VBUS_V_EN	O			NC
56	USB_VBUS	I			NC
58	B34_L22_P	DIFFIO	34	R19	NC
60	B34_L22_N	DIFFIO	34	T19	NC



## Appendix VI - ECU connector tables

Caution; maximum values for each pin must be taken into account when ECU is connected.

### Connection and description table for common - bay X

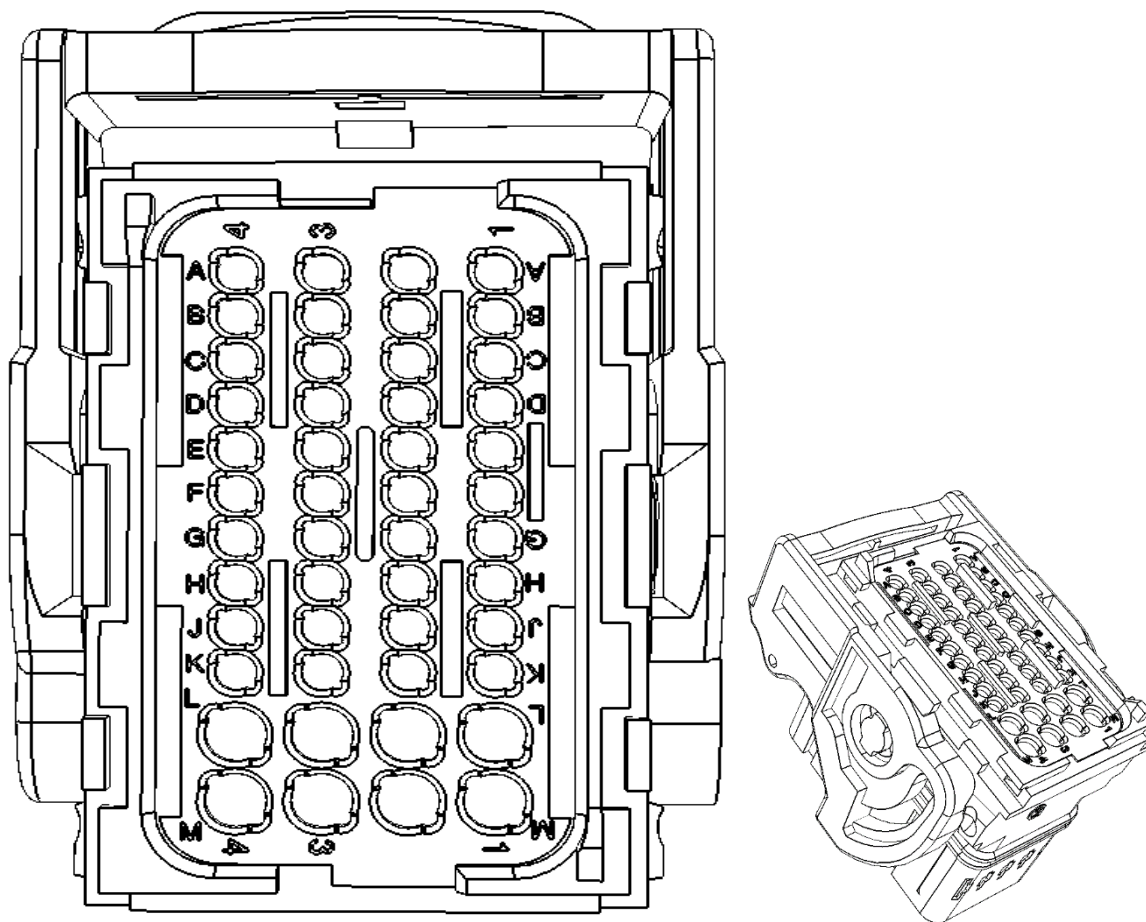


Figure 217 - Bay X pin locations

Table 74 - Connection and description table for bay X

PIN #	H #	NETALIAS	DESCRIPTION	TYPICAL VALUES	MAX. VALUES
XA1	A1	CAN_H	Network to external devices, high signal	2.5V-3.5V	5V/2.5A
XA2	A2	CAN_L	Network to external devices, low signal	1V-2V	5V/2.5A
XA3	A3	5V0	5V supply voltage to sensors.	5V	5V/2.5A
XA4	A4	ACC_IN_0	Accelerator pedal signal 1 (0-100%)	0.5V-4.5V	5V/2.5A
XB1	B1	GPDIO_IN_0	General purpose digital config. IN/OUTPUT	0V/5V	5V/2.5A
XB2	B2	GPDIO_IN_1	General purpose digital config. IN/OUTPUT	0V/5V	5V/2.5A
XB3	B3	5V0	5V supply voltage to sensors.	5V	5V/2.5A



XB4	B4	<b>ACC_IN_1</b>	Accelerator pedal signal 2 (0-100%)	0.5V-4.5V	<b>5V/2.5A</b>
XC1	C1	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
XC2	C2	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
XC3	C3	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
XC4	C4	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
XD1	D1	<b>FLEX_FUEL_IN</b>	Ethanol sensor input - Digital (0-100%)	50-150Hz	<b>5V/2.5A</b>
XD2	D2	<b>MAF_IN</b>	Airflow sensor input - Digital (0-700kg/h)	2-10kHz	<b>5V/2.5A</b>
XD3	D3	<b>GPAO_0</b>	General analog output signal 1 (DAC)	0-13.6V	16V/2.5A
XD4	D4	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
XE1	E1	<b>IGN_SIG_IN</b>	Ignition activated signal - Digital	0 / 12V	16V/2.5A
XE2	E2	<b>BRAKE_SIG_IN</b>	Brake pedal depressed signal - Digital	0 / 12V	16V/2.5A
XE3	E3	<b>GPAO_1</b>	General analog output signal 2 (DAC)	0-13.6V	16V/2.5A
XE4	E4	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
XF1	F1	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
XF2	F2	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
XF3	F3	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
XF4	F4	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
XG1	G1	<b>GPI_IN_0</b>	General digital input signal 1 - PWM	0-13kHz	<b>5V/2.5A</b>
XG2	G2	<b>GPI_IN_1</b>	General digital input signal 2 - PWM	0-13kHz	<b>5V/2.5A</b>
XG3	G3	<b>GPAI_IN_0</b>	General analog input 1 - filtered Fc=1 kHz	0.5V-4.5V	<b>5V/2.5A</b>
XG4	G4	<b>MAP_IN</b>	Manifold Air Pres. sensor input (0.2-5 Bar)	0.3-4.8V	<b>5V/2.5A</b>
XH1	H1	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
XH2	H2	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
XH3	H3	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
XH4	H4	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
XJ1	J1	<b>GPDIO_IN_2</b>	General purpose digital config. IN/OUTPUT	0V/5V	<b>5V/2.5A</b>
XJ2	J2	<b>GPDIO_IN_3</b>	General purpose digital config. IN/OUTPUT	0V/5V	<b>5V/2.5A</b>
XJ3	J3	<b>GPAI_IN_1</b>	General analog input 2 - filtered Fc=1 kHz	0.5V-4.5V	<b>5V/2.5A</b>
XJ4	J4	<b>OIL_PRESS_IN</b>	Oil Pressure sensor input (0.5-11 Bar)	0.5V-4.5V	<b>5V/2.5A</b>
XK1	K1	<b>CAN_VALVE_H</b>	Network to external devices, high signal	2.5V-3.5V	<b>5V/2.5A</b>
XK2	K2	<b>CAN_VALVE_L</b>	Network to external devices, low signal	1V-2V	<b>5V/2.5A</b>
XK3	K3	<b>GPAI_IN_2</b>	General analog input 3 - filtered Fc=1 kHz	0.5V-4.5V	<b>5V/2.5A</b>
XK4	K4	<b>FUEL_PRESS_IN</b>	Fuel Pressure sensor input (0.5-11 Bar)	0.5V-4.5V	<b>5V/2.5A</b>
XL1	L1	VBAT	Battery supply voltage	10.5-16V	<b>18V/12A</b>
XL2	L2	VBAT	Battery supply voltage	10.5-16V	<b>18V/12A</b>
XL3	L3	VBAT	Battery supply voltage	10.5-16V	<b>18V/12A</b>
XL4	L4	VBAT	Battery supply voltage	10.5-16V	<b>18V/12A</b>
XM1	M1	GND	Chassis ground for sensors & actuators	0V	0V/12A
XM2	M2	GND	Chassis ground for sensors & actuators	0V	0V/12A
XM3	M3	GND	Chassis ground for sensors & actuators	0V	0V/12A
XM4	M4	GND	Chassis ground for sensors & actuators	0V	0V/12A





### Connection and description table for bank 1 - bay Y

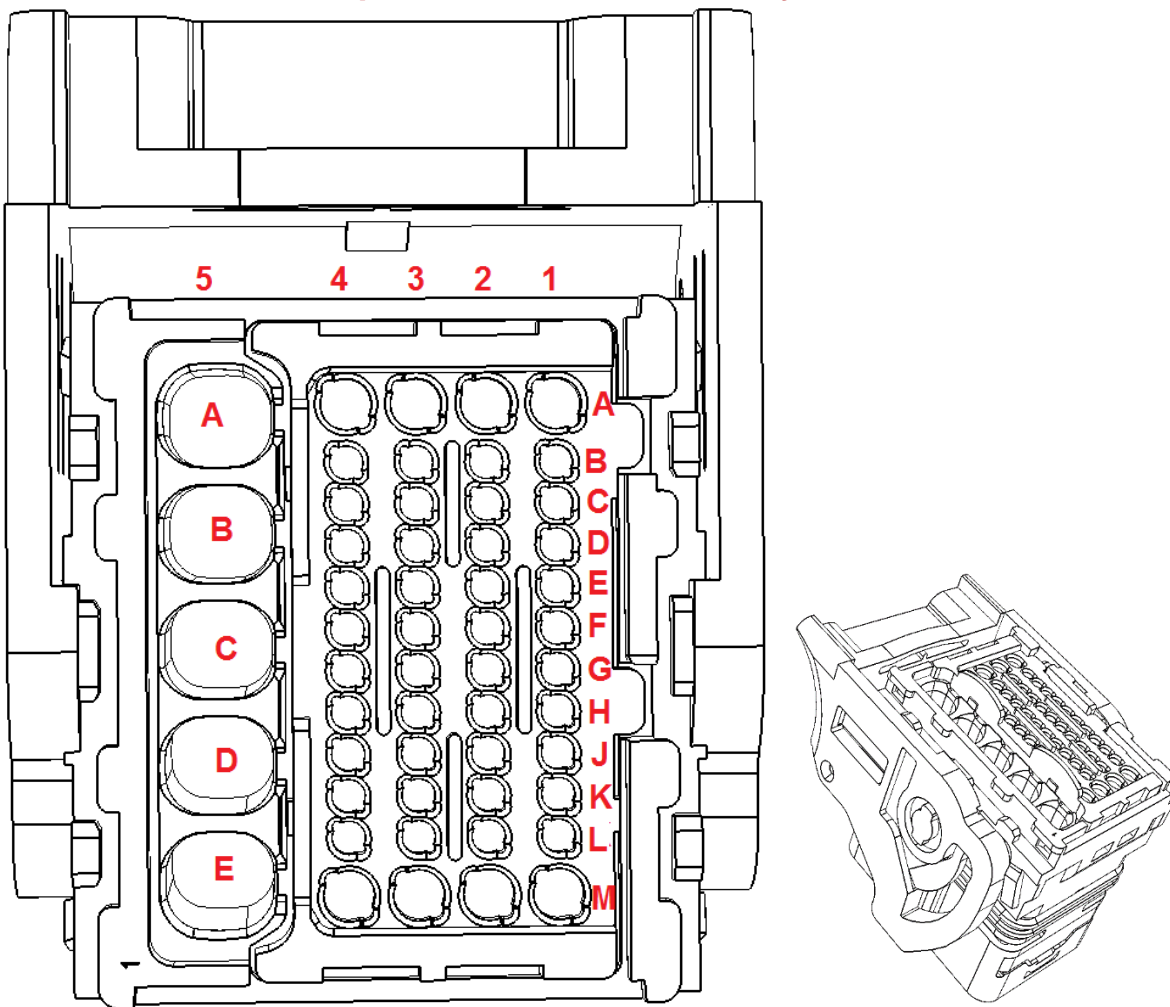


Figure 218 - Bay Y pin locations

Table 75 - Connection and description table for bay Y

PIN #	H #	SIGNAL NAME	DESCRIPTION	TYPICAL VALUES	MAX. VALUES
YA1	Y_53	INJ_0	Output power to fuel injector 1 PWM <7ms	0-250 Hz	16V/12A
YA2	Y_41	INJ_1	Output power to fuel injector 2 PWM <7ms	0-250 Hz	16V/12A
YA3	Y_29	INJ_2	Output power to fuel injector 3 PWM <7ms	0-250 Hz	16V/12A
YA4	Y_17	INJ_3	Output power to fuel injector 4 PWM <7ms	0-250 Hz	16V/12A
YA5	Y_5A	IGN_0	Output power to ignition coil 1 PWM 0.4ms	0-2.5kHz	16V/21A
YB1	Y_52	THR_FB_0	Throttle position signal 1 HIGH (0-100%)	0.5V-4.5V	5V/2.5A
YB2	Y_40	THR_FB_1	Throttle position signal 2 LOW (0-100%)	4.5-0.5V	5V/2.5A
YB3	Y_28	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
YB4	Y_16	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
YB5	Y_4A	IGN_1	Output power to ignition coil 2 PWM 0.4ms	0-2.5kHz	16V/21A



YC1	Y_51	<b>EXHT_N_0</b>	Exhaust temperature sensor 1 (-) 0-1300°C	0mV	<b>3.3V/2.5A</b>
YC2	Y_39	<b>EXHT_P_0</b>	Exhaust temperature sensor 1 (+) 0-1300°C	0-50mV	<b>3.3V/2.5A</b>
YC3	Y_27	<b>CPS_IN_0</b>	Cylinder pressure sensor 1 (0-200Bar)	0.5V-4.5V	<b>5V/2.5A</b>
YC4	Y_15	5V0	5V supply voltage to sensors.	5V	<b>5V/2.5A</b>
YC5	Y_3A	<b>IGN_2</b>	Output power to ignition coil 3 PWM 0.4ms	0-2.5kHz	<b>16V/21A</b>
YD1	Y_50	<b>EXHT_N_1</b>	Exhaust temperature sensor 2 (-) 0-1300°C	0mV	<b>3.3V/2.5A</b>
YD2	Y_38	<b>EXHT_P_1</b>	Exhaust temperature sensor 2 (+) 0-1300°C	0-50mV	<b>3.3V/2.5A</b>
YD3	Y_26	<b>CPS_IN_1</b>	Cylinder pressure sensor 2 (0-200Bar)	0.5V-4.5V	<b>5V/2.5A</b>
YD4	Y_14	5V0	5V supply voltage to sensors.	5V	<b>5V/2.5A</b>
YD5	Y_2A	<b>IGN_3</b>	Output power to ignition coil 4 PWM 0.4ms	0-2.5kHz	<b>16V/21A</b>
YE1	Y_49	<b>EXHT_N_2</b>	Exhaust temperature sensor 3 (-) 0-1300°C	0mV	<b>3.3V/2.5A</b>
YE2	Y_37	<b>EXHT_P_2</b>	Exhaust temperature sensor 3 (+) 0-1300°C	0-50mV	<b>3.3V/2.5A</b>
YE3	Y_25	<b>CPS_IN_2</b>	Cylinder pressure sensor 3 (0-200Bar)	0.5V-4.5V	<b>5V/2.5A</b>
YE4	Y_13	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
YE5	Y_1A	<b>IGN_FLY_0</b>	Flyback diode protection for bank 1 HS	12V	<b>16V/21A</b>
YF1	Y_48	<b>EXHT_N_3</b>	Exhaust temperature sensor 4 (-) 0-1300°C	0mV	<b>3.3V/2.5A</b>
YF2	Y_36	<b>EXHT_P_3</b>	Exhaust temperature sensor 4 (+) 0-1300°C	0-50mV	<b>3.3V/2.5A</b>
YF3	Y_24	<b>CPS_IN_3</b>	Cylinder pressure sensor 4 (0-200Bar)	0.5V-4.5V	<b>5V/2.5A</b>
YF4	Y_12	<b>KNOCK_IN_0</b>	Knock sensor input bank 1 (1-20kHz)	0.5V-4.5V	<b>5V/2.5A</b>
YG1	Y_47	<b>NTC_N_0</b>	Coolant tempsensor NTC (-) -40-150°C	45k-47Ω	<b>3.3V/2.5A</b>
YG2	Y_35	<b>NTC_P_0</b>	Coolant tempsensor NTC (+) -40-150°C	45k-47Ω	<b>3.3V/2.5A</b>
YG3	Y_23	<b>VR_IN_H</b>	Crankshaft pos/vel sensor (+) 0-16kRpm	0-200Vpp	<b>5V/2.5A</b>
YG4	Y_11	<b>VR_IN_L</b>	Crankshaft pos/vel sensor (-) 0-16kRpm	0-200Vpp	<b>5V/2.5A</b>
YH1	Y_46	<b>NTC_N_1</b>	Fuel tempsensor NTC (-) -40-150°C	45k-47Ω	<b>3.3V/2.5A</b>
YH2	Y_34	<b>NTC_P_1</b>	Fuel tempsensor NTC (+) -40-150°C	45k-47Ω	<b>3.3V/2.5A</b>
YH3	Y_22	<b>HALL_IN_0</b>	Camshaft position bank 1 (intake) <10kHz	0.4V-5V	<b>5V/2.5A</b>
YH4	Y_10	<b>HALL_IN_1</b>	Camshaft position bank 1 (exhaust) <10kHz	0.4V-5V	<b>5V/2.5A</b>
YJ1	Y_45	<b>NTC_N_2</b>	Oil tempsensor NTC (-) -40-150°C	45k-47Ω	<b>3.3V/2.5A</b>
YJ2	Y_33	<b>NTC_P_2</b>	Oil tempsensor NTC (+) -40-150°C	45k-47Ω	<b>3.3V/2.5A</b>
YJ3	Y_21	<b>VS_1</b>	Lambda bank 1 - Nernst cell input (%O2)	0.2-0.8V	<b>3.3V/2.5A</b>
YJ4	Y_9	<b>IP_1</b>	Lambda bank 1 - Pump current output (mA)	(-)9-6mA	<b>3.3V/2.5A</b>
YK1	Y_44	<b>NTC_N_3</b>	MAF GND (signal GND)	45k-47Ω	<b>3.3V/2.5A</b>
YK2	Y_32	<b>NTC_P_3</b>	MAF tempsensor NTC (+) -40-130°C	45k-47Ω	<b>3.3V/2.5A</b>
YK3	Y_20	<b>VGND_1</b>	Lambda bank 1 - Seperate signal ground	2V	<b>3.3V/2.5A</b>
YK4	Y_8	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
YL1	Y_43	<b>BOOST_CTR</b>	Output power Turbo Control Valve 20-100%	250-300Hz	16V/2.5A
YL2	Y_31	<b>FUEL_PUMP</b>	Output power to fuel pump relay (on/off)	0V/12V	16V/2.5A
YL3	Y_19	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
YL4	Y_7	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
YM1	Y_42	<b>HEAT_L_1</b>	Lambda bank 1 - Heater element activation	0-7.5V	<b>16V/12A</b>
YM2	Y_30	<b>THR_OUT_1</b>	Output power to throttle motor (-) PWM	0-12V	<b>16V/12A</b>
YM3	Y_18	<b>THR_OUT_2</b>	Output power to throttle motor (+) PWM	0-12V	<b>16V/12A</b>
YM4	Y_6	<b>GPLSS_0</b>	General power output 1 (Low Side Switch)	0-12V	<b>16V/12A</b>



## Connection and description table for bank 2 - bay Z

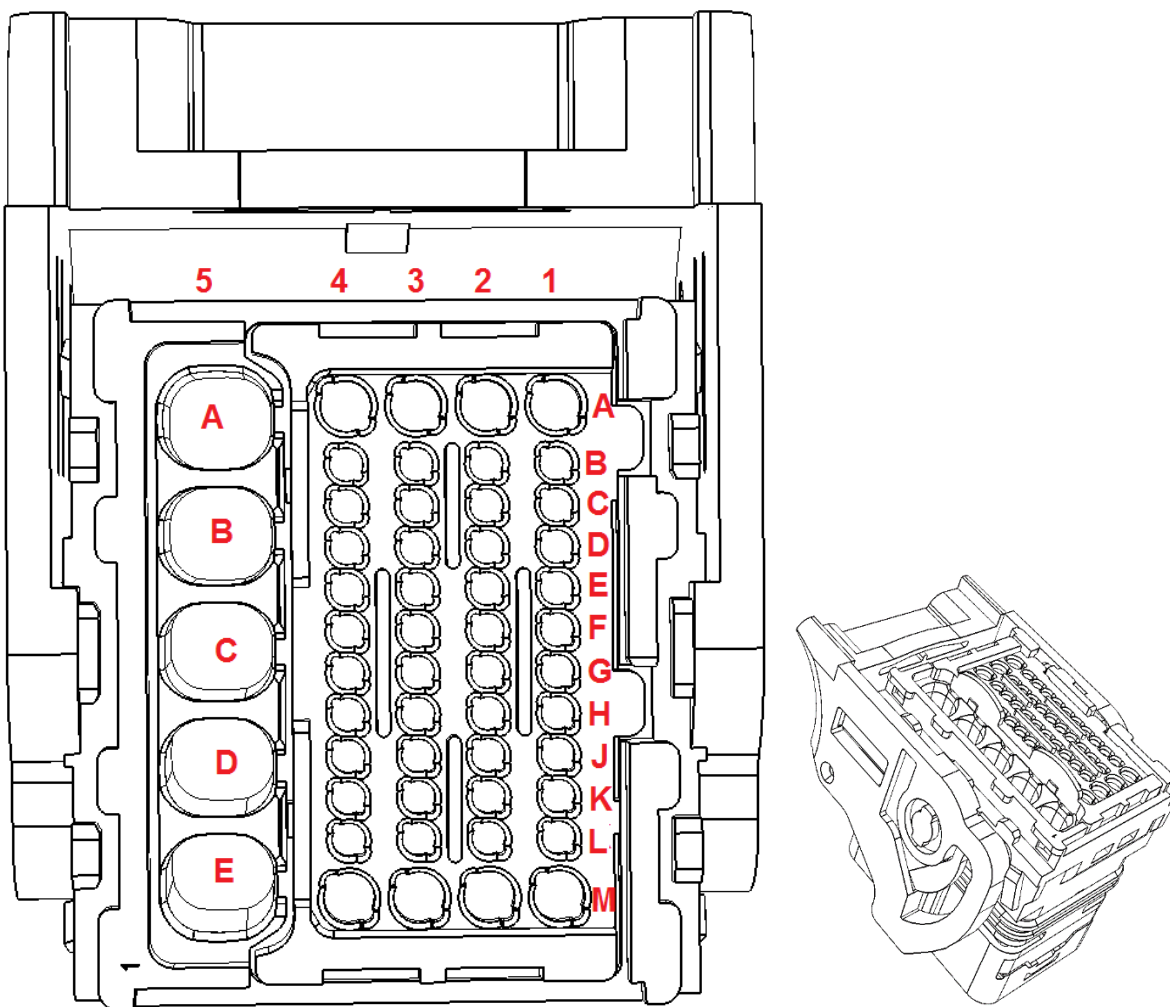


Figure 219 - Bay Z pin locations

Table 76 - Connection and description table for bay Z

PIN #	H #	SIGNAL NAME	DESCRIPTION	TYPICAL VALUES	MAX. VALUES
ZA1	Z_53	INJ_4	Output power to fuel injector 5 PWM <7ms	0-250 Hz	16V/12A
ZA2	Z_41	INJ_5	Output power to fuel injector 6 PWM <7ms	0-250 Hz	16V/12A
ZA3	Z_29	INJ_6	Output power to fuel injector 7 PWM <7ms	0-250 Hz	16V/12A
ZA4	Z_17	INJ_7	Output power to fuel injector 8 PWM <7ms	0-250 Hz	16V/12A
ZA5	Z_5A	IGN_4	Output power to ignition coil 5 PWM 0.4ms	0-2.5kHz	16V/21A
ZB1	Z_52	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
ZB2	Z_40	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
ZB3	Z_28	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
ZB4	Z_16	GND	Chassis ground for sensors & actuators	0V	0V/2.5A



ZB5	Z_4A	<b>IGN_5</b>	Output power to ignition coil 6 PWM 0.4ms	0-2.5kHz	<b>16V/21A</b>
ZC1	Z_51	<b>EXHT_N_4</b>	Exhaust temperature sensor 5 (-) 0-1300°C	0mV	<b>3.3V/2.5A</b>
ZC2	Z_39	<b>EXHT_P_4</b>	Exhaust temperature sensor 5 (+) 0-1300°C	0-50mV	<b>3.3V/2.5A</b>
ZC3	Z_27	<b>CPS_IN_4</b>	Cylinder pressure sensor 5 (0-200Bar)	0.5V-4.5V	<b>5V/2.5A</b>
ZC4	Z_15	5V0	5V supply voltage to sensors.	5V	<b>5V/2.5A</b>
ZC5	Z_3A	<b>IGN_6</b>	Output power to ignition coil 7 PWM 0.4ms	0-2.5kHz	<b>16V/21A</b>
ZD1	Z_50	<b>EXHT_N_5</b>	Exhaust temperature sensor 6 (-) 0-1300°C	0mV	<b>3.3V/2.5A</b>
ZD2	Z_38	<b>EXHT_P_5</b>	Exhaust temperature sensor 6 (+) 0-1300°C	0-50mV	<b>3.3V/2.5A</b>
ZD3	Z_26	<b>CPS_IN_5</b>	Cylinder pressure sensor 6 (0-200Bar)	0.5V-4.5V	<b>5V/2.5A</b>
ZD4	Z_14	5V0	5V supply voltage to sensors.	5V	<b>5V/2.5A</b>
ZD5	Z_2A	<b>IGN_7</b>	Output power to ignition coil 8 PWM 0.4ms	0-2.5kHz	<b>16V/21A</b>
ZE1	Z_49	<b>EXHT_N_6</b>	Exhaust temperature sensor 7 (-) 0-1300°C	0mV	<b>3.3V/2.5A</b>
ZE2	Z_37	<b>EXHT_P_6</b>	Exhaust temperature sensor 7 (+) 0-1300°C	0-50mV	<b>3.3V/2.5A</b>
ZE3	Z_25	<b>CPS_IN_6</b>	Cylinder pressure sensor 7 (0-200Bar)	0.5V-4.5V	<b>5V/2.5A</b>
ZE4	Z_13	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
ZE5	Z_1A	<b>IGN_FLY_1</b>	Flyback diode protection for bank 2 HS	12V	<b>16V/21A</b>
ZF1	Z_48	<b>EXHT_N_7</b>	Exhaust temperature sensor 8 (-) 0-1300°C	0mV	<b>3.3V/2.5A</b>
ZF2	Z_36	<b>EXHT_P_7</b>	Exhaust temperature sensor 8 (+) 0-1300°C	0-50mV	<b>3.3V/2.5A</b>
ZF3	Z_24	<b>CPS_IN_7</b>	Cylinder pressure sensor 8 (0-200Bar)	0.5V-4.5V	<b>5V/2.5A</b>
ZF4	Z_12	<b>KNOCK_IN_1</b>	Knock sensor input bank 2 (1-20kHz)	0.5V-4.5V	<b>5V/2.5A</b>
ZG1	Z_47	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
ZG2	Z_35	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
ZG3	Z_23	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
ZG4	Z_11	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
ZH1	Z_46	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
ZH2	Z_34	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
ZH3	Z_22	<b>HALL_IN_2</b>	Camshaft position bank 2 (intake) <10kHz	0.4V-5V	<b>5V/2.5A</b>
ZH4	Z_10	<b>HALL_IN_3</b>	Camshaft position bank 2 (exhaust) <10kHz	0.4V-5V	<b>5V/2.5A</b>
ZJ1	Z_45	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
ZJ2	Z_33	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
ZJ3	Z_21	<b>VS_2</b>	Lambda bank 2 - Nernst cell input (%O2)	0.2-0.8V	<b>3.3V/2.5A</b>
ZJ4	Z_9	<b>IP_2</b>	Lambda bank 2 - Pump current output (mA)	(-)9-6mA	<b>3.3V/2.5A</b>
ZK1	Z_44	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
ZK2	Z_32	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
ZK3	Z_20	<b>VGND_2</b>	Lambda bank 2 - Seperate signal ground	2V	<b>3.3V/2.5A</b>
ZK4	Z_8	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
ZL1	Z_43	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
ZL2	Z_31	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
ZL3	Z_19	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
ZL4	Z_7	GND	Chassis ground for sensors & actuators	0V	0V/2.5A
ZM1	Z_42	<b>HEAT_L_2</b>	Lambda bank 2 - Heater element activation	0-7.5V	<b>16V/12A</b>
ZM2	Z_30	<b>GPLSS_1</b>	General power output 2 (Low Side Switch)	0-12V	<b>16V/12A</b>
ZM3	Z_18	<b>GPLSS_2</b>	General power output 3 (Low Side Switch)	0-12V	<b>16V/12A</b>
ZM4	Z_6	<b>GPLSS_3</b>	General power output 4 (Low Side Switch)	0-12V	<b>16V/12A</b>



## Bibliography

- Analog devices. (2008, 10). *Understanding SINAD and ENOB*. Retrieved from <http://www.analog.com/media/en/training-seminars/tutorials/MT-003.pdf>
- Android. (n.d.). *Signing Your Applications*. Retrieved from Developer Android: <http://developer.android.com/tools/publishing/app-signing.html>
- Android. (n.d.). *Using DDMS*. Retrieved from Developer Android: <http://developer.android.com/tools/debugging/ddms.html>
- Atmel. (2015). *AT93C46E Datasheet*. Retrieved from <http://www.atmel.com/Images/Atmel-5207-SEEPROM-AT93C46E-Datasheet.pdf>
- Behera, S. (2010). *FPGA based PWM*. Retrieved from [http://ethesis.nitrkl.ac.in/1985/1/FPGA\\_based\\_PWM\\_techniques\\_for\\_controlling\\_Inverter.pdf](http://ethesis.nitrkl.ac.in/1985/1/FPGA_based_PWM_techniques_for_controlling_Inverter.pdf)
- Bemporad, A. (2010). *University of Trento*. Retrieved 03 10, 2015, from <http://cse.lab.imtlucca.it/~bemporad/teaching/ac/pdf/AC2-03-Sampling.pdf>
- Bosch. (2005). *LSU 4.9*. Retrieved from [https://www.rbracing-rsr.com/downloads/wiring\\_pdfs/bosch\\_lsu49.pdf](https://www.rbracing-rsr.com/downloads/wiring_pdfs/bosch_lsu49.pdf)
- Bosch. (2014, 12 10). *Pressure sensor Fluid PSS-10 datasheet*. Retrieved from [http://www.bosch-motorsport.de/media/catalog\\_resources/Pressure\\_Sensor\\_Fluid\\_PSS-10\\_Datasheet\\_51\\_en\\_2780784395pdf.pdf](http://www.bosch-motorsport.de/media/catalog_resources/Pressure_Sensor_Fluid_PSS-10_Datasheet_51_en_2780784395pdf.pdf)
- Bosch. (2015, 01 21). *Lambda Sensor LSU 4.9*. Retrieved from [http://www.bosch-motorsport.de/media/catalog\\_resources/Lambda\\_Sensor\\_LSU\\_49\\_Datasheet\\_51\\_en\\_2779147659pdf.pdf](http://www.bosch-motorsport.de/media/catalog_resources/Lambda_Sensor_LSU_49_Datasheet_51_en_2779147659pdf.pdf)
- Bosch. (2015, 01 21). *Pressure Sensor Air PSA-C*. Retrieved from [http://www.bosch-motorsport.de/media/catalog\\_resources/Pressure\\_Sensor\\_Air\\_PSA-C\\_Datasheet\\_51\\_en\\_2779841291pdf.pdf](http://www.bosch-motorsport.de/media/catalog_resources/Pressure_Sensor_Air_PSA-C_Datasheet_51_en_2779841291pdf.pdf)
- Bosch Engineering. (2015, 01 21). *Temperature Sensor NTC M12-H Datasheet*. Retrieved from [http://www.bosch-motorsport.com/media/catalog\\_resources/Temperature\\_Sensor\\_NTC\\_M12-H\\_Datasheet\\_51\\_en\\_2782610059pdf.pdf](http://www.bosch-motorsport.com/media/catalog_resources/Temperature_Sensor_NTC_M12-H_Datasheet_51_en_2782610059pdf.pdf)
- Bosch Engineering. (2015, 01 21). *Thermocouple Probe TCP K Datasheet*. Retrieved from [http://www.bosch-motorsport.com/media/catalog\\_resources/Thermocouple\\_Probe\\_TCP\\_K\\_Datasheet\\_51\\_en\\_10753994123pdf.pdf](http://www.bosch-motorsport.com/media/catalog_resources/Thermocouple_Probe_TCP_K_Datasheet_51_en_10753994123pdf.pdf)



- Bosch Engineering GmbH. (2006, 04). *CJ125 Datasheet*. Retrieved from [http://www.bosch-semiconductors.de/media/pdf\\_1/einzeldownloads/engine\\_management/cj125\\_product\\_info.pdf](http://www.bosch-semiconductors.de/media/pdf_1/einzeldownloads/engine_management/cj125_product_info.pdf)
- Bosch Engineering. (n.d.). *Hot-film air-mass meter HFM 5 Datasheet*. Retrieved from [http://www.maseratilife.com/forums/attachments/coupe-spyder-gs/10210d1326782931-diy-maf-sensor-cleaning-bosch-hfm-5-sensors\\_airmass.pdf](http://www.maseratilife.com/forums/attachments/coupe-spyder-gs/10210d1326782931-diy-maf-sensor-cleaning-bosch-hfm-5-sensors_airmass.pdf)
- Bosch. (n.d.). *Mass airflow sensor HFM 6*. Retrieved from [http://rb-aa.bosch.com/boaasocs/modules-pdf/Bosch\\_AA\\_Sensoren\\_PDFGen.dll?db=BOAAMDB02PRD7&MV\\_ID=868&CL\\_ID=20&Prod\\_ID=547&filename=0281002764.pdf](http://rb-aa.bosch.com/boaasocs/modules-pdf/Bosch_AA_Sensoren_PDFGen.dll?db=BOAAMDB02PRD7&MV_ID=868&CL_ID=20&Prod_ID=547&filename=0281002764.pdf)
- Bosch Motorsport. (2015). *Bosch Motorsport*. Retrieved from [http://www.bosch-motorsport.de/media/catalog\\_resources/Lambda\\_Sensor\\_LSU\\_49\\_Datasheet\\_51\\_en\\_2779147659pdf.pdf](http://www.bosch-motorsport.de/media/catalog_resources/Lambda_Sensor_LSU_49_Datasheet_51_en_2779147659pdf.pdf)
- Chistiansen, D., & Alexander, C. (2004). *Standard Handbook of Electrical Engineering*. Macgraw-Hill.
- Department of Defense. (1991, 12 02). *Reliability Prediction of Electronic Equipment MIL-HDBK-217F*. Retrieved from <http://www.sre.org/pubs/Mil-Hdbk-217F.pdf>
- Diodes Incorporated. (2014, 04 02). *B560C*. Retrieved from Diodes: <http://www.diodes.com/datasheets/ds13012.pdf>
- EFI. (2015, 01 05). *rusEfi - an open source DIY ECU*. Retrieved from [http://rusefi.com/wiki/index.php?title=Main\\_Page](http://rusefi.com/wiki/index.php?title=Main_Page)
- Errington, J. (2014). *John errington's data conversion website*. Retrieved 02 10, 2015, from <http://www.skillbank.co.uk/SignalConversion/rate.htm>
- Fairchild Semiconductor Corporation. (2014, 94 28). *AN-8208 Introduction to Automotive Ignition Systems*. Retrieved from <https://www.fairchildsemi.com/application-notes/AN/AN-8208.pdf>
- Fairchild Semiconductor. (2012, 01). *ISL9V5045S3ST Datasheet*. Retrieved from [https://www.fairchildsemi.com/datasheets/IS/ISL9V5045S\\_F085.pdf](https://www.fairchildsemi.com/datasheets/IS/ISL9V5045S_F085.pdf)
- Fairchild Semiconductor. (2013). *74LCX125*. Retrieved from Fairchild: <https://www.fairchildsemi.com/datasheets/74/74LCX125.pdf>
- Fairchild Semiconductor. (2013, 12). *74LCX125 Datasheet*. Retrieved from <https://www.fairchildsemi.com/datasheets/74/74LCX125.pdf>





- Fairchild Semiconductor Corporation. (2002). *MMBT3904 Datasheet*. Retrieved from <https://www.fairchildsemi.com/datasheets/MM/MMBT3904.pdf>
- Freescale Semiconductor. (2012). *MC33931 Datasheet*. Retrieved from [http://cache.freescale.com/files/analog/doc/data\\_sheet/MC33931.pdf](http://cache.freescale.com/files/analog/doc/data_sheet/MC33931.pdf)
- FTDI Chip. (2012). *FT2232H Datasheet*. Retrieved from [http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS\\_FT2232H.pdf](http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT2232H.pdf)
- G.Horner, T. (1995). *Engine knock detection using spectral analysis techniques with a TMS320 DSP*. Texas Instruments.
- Gershun, A. (2015, 5 8). *agershun/alasql*. Retrieved from Github: <https://github.com/agershun/alasql>
- Grippo, B. B. (2014). *MegaSquirt ECU*. Retrieved February 3, 2015, from <http://www.megamanual.com/PWC/>
- Gudvangen, S. (2014). *Introduction to quantisation*.
- Hi Octane racing. (2011, 06 03). *THE TRUTH ABOUT E85 FUEL*. Retrieved from Hi Octane Racing: <http://hioctaneracing.com/blog/2011/7/3/the-truth-about-e85-fuel.html>
- IEEE Standards Association. (2012). IEEE Std. 802.11-2012. Piscataway, USA.
- IEEE Standards Association. (2012). IEEE Std. 802.3-2012. Piscataway, USA.
- IEEE Standards Association. (2013). IEEE Std. 1149.1-2013. Piscataway, USA.
- Inc, X. (2014, 11 25). *PetaLinux Tools Documentation - Reference Guide*. Retrieved from [www.Xilinx.com:  
http://www.xilinx.com/support/documentation/sw\\_manuals/petalinux2014\\_4/ug1144-petalinux-tools-reference-guide.pdf](http://www.xilinx.com/support/documentation/sw_manuals/petalinux2014_4/ug1144-petalinux-tools-reference-guide.pdf)
- Inc, X. (n.d.). *Multi-OS Support (AMP & Hypervisor)*. Retrieved from [www.Xilinx.com:  
http://www.wiki.xilinx.com/Multi-OS+Support+%28AMP+%26+Hypervisor%29](http://www.wiki.xilinx.com/Multi-OS+Support+%28AMP+%26+Hypervisor%29)
- International Electrotechnical Commission (IEC). (2005). IEC 60384-8. Geneva, Switzerland.
- International Electrotechnical Commission (IEC). (2005). IEC 60384-9. Geneva, Switzerland.
- International Organization for Standardization (ISO). (2013). ISO 11898-6:2013. Geneva, Switzerland.
- International Rectifier. (2000). *IRG4BC40S Datasheet*. Retrieved from <http://www.irf.com/product-info/datasheets/data/irg4bc40s.pdf>



- International Rectifier. (2008). *IRFS4127PBF Datasheet*. Retrieved from <http://www.irf.com/product-info/datasheets/data/irfs4127pbf.pdf>
- Jacobs, C. (1999). Performance ignition systems. HPBooks.
- Jounaeh, M. (2013). *Fundamentals of Mechatronics, SI*. Stamford: Cengage Learning.
- Krummen, J. (2011, N/A N/A). *Ecus and engine calibration*. Retrieved 02 18, 2015, from sae: [http://www.sae.org/students/presentations/ecus\\_and\\_engine\\_calibration\\_201\\_by\\_jeff\\_krummen.pdf](http://www.sae.org/students/presentations/ecus_and_engine_calibration_201_by_jeff_krummen.pdf)
- Likely, C. (n.d.). *Achiving EMC for DC-DC converters*. Retrieved from compliance-club: [http://www.compliance-club.com/archive/old\\_archive/021132.htm](http://www.compliance-club.com/archive/old_archive/021132.htm)
- Linear Technology. (1998). *LT1468 Datasheet*. Retrieved from <http://cds.linear.com/docs/en/datasheet/1468fb.pdf>
- Linear Technology. (1999). *LTC2050/LTC2050HV Datasheet*. Retrieved from <http://cds.linear.com/docs/en/datasheet/2050fc.pdf>
- Linear Technology. (1999). *LTC2050/LTC2050HV Datasheet*. Retrieved from <http://cds.linear.com/docs/en/datasheet/2050fc.pdf>
- Linear Technology. (2007). *LT3580 Datasheet*. Retrieved from Linear: <http://cds.linear.com/docs/en/datasheet/3580fg.pdf>
- Linear Technology. (2011). *LT3724 Datasheet*. Retrieved from Linear: <http://cds.linear.com/docs/en/datasheet/3724fd.pdf>
- Linear Technology. (2011). *LT4363 Datasheet*. Retrieved from <http://cds.linear.com/docs/en/datasheet/4363fb.pdf>
- Linear Technology. (2011). *LTC2997 Datasheet*. Retrieved from <http://cds.linear.com/docs/en/datasheet/2997fa.pdf>
- Linear Technology. (2012). *LTC3891 Datasheet*. Retrieved from Linear: <http://cds.linear.com/docs/en/datasheet/3891fa.pdf>
- Linear Technology. (2014). *LT8580 Datasheet*. Retrieved from Linear: <http://cds.linear.com/docs/en/datasheet/8580f.pdf>
- Linear Technology. (2014). *LTC2983 Datasheet*. Retrieved from <http://cds.linear.com/docs/en/datasheet/2983f.pdf>
- Linear Technology. (2014). *LTM4609 Datasheet*. Retrieved from Linear: <http://cds.linear.com/docs/en/datasheet/4609ff.pdf>





- Linear Technology. (2015). *LTC2050HV*. Retrieved from Linear:  
<http://cds.linear.com/docs/en/datasheet/2050fc.pdf>
- Linear Technology. (2015, 03 05). *LTSpice IV Simulation tool*. Retrieved from  
<http://www.linear.com/designtools/software/>
- Link Engine management. (2002). *Battery compensation*. Retrieved from  
<http://www.linkecu.com/support/glossary/battery-compensation>
- m8rge. (2014, 12 17). *cwebsocket*. Retrieved 2 25, 2015, from GitHub:  
<https://github.com/m8rge/cwebsocket>
- Management, L. E. (N/A, N/A N/A). *Battery compensation*. Retrieved 02 18, 2015, from Linkecu:  
<http://www.linkecu.com/support/glossary/battery-compensation>
- Marvell. (2015). *Alaska Gigabit Ethernet Transceivers*. Retrieved from  
<http://www.marvell.com/transceivers/alaska-gbe/>
- Mason, M. (2009, 09 24). *Electronic design*. Retrieved 03 03, 2015, from  
<http://electronicdesign.com/analog/abcs-adcs>
- Maxim. (2012, 3 12). *MAX9924-MAX9927 Datasheet*. Retrieved from  
<http://datasheets.maximintegrated.com/en/ds/MAX9924-MAX9927.pdf>
- Measurement Specialties. (2012, 10 26). *MS5611-01BA03 Datasheet*. Retrieved from  
<http://www.meas-spec.com/downloads/MS5611-01BA03.pdf>
- Megasquirt. (2015). *Flexfuel*. Retrieved from <http://www.megamanual.com/flexfuel.htm>
- MegaSquirt. (2015, 03 16). *megamanual*. Retrieved from <http://www.megamanual.com/PWC/>
- Microchip Technology Inc. (2012). *MRF24WG0MA/MB Datasheet*. Retrieved from  
<http://ww1.microchip.com/downloads/en/DeviceDoc/70686B.pdf>
- Molex. (2008). *2.00 mm Milli-Grid Connector overview*. Retrieved from  
<http://www.literature.molex.com/SQLImages/kelmscott/Molex/PDF/Images/987650-1991.PDF>
- Molex Inc. (2015, 04 24). *Molex CMC header*. Retrieved from Molex:  
[http://www.molex.com/webdocs/datasheets/pdf/en-us/0347630001\\_PCB\\_HEADERS.pdf](http://www.molex.com/webdocs/datasheets/pdf/en-us/0347630001_PCB_HEADERS.pdf)
- Molex Inc. (2015, 04 28). *Molex CMC Receptacle*. Retrieved from Molex:  
[http://www.molex.com/webdocs/datasheets/pdf/en-us/0643212019\\_CRIMP\\_HOUSINGS.pdf](http://www.molex.com/webdocs/datasheets/pdf/en-us/0643212019_CRIMP_HOUSINGS.pdf)



- Motec. (2015). *Bosch LSU 4.9 Datasheet*. Retrieved from <http://www.motec.com/filedownload.php?docid=3525>
- MoTec. (n.d.). *Bosch LSU 4.9 Sensor Motec*. Retrieved from <http://www.motec.com/filedownload.php?docid=3525>
- Mozilla. (2015, 04 6). *Page Inspector*. Retrieved from Developer Mozilla: [https://developer.mozilla.org/en-US/docs/Tools/Page\\_Inspector](https://developer.mozilla.org/en-US/docs/Tools/Page_Inspector)
- National Aeronautics and Space Administration (NASA). (n.d.). *NASA Parts Application Handbook MIL-HDBK-978-B*. Retrieved from [https://nepp.nasa.gov/files/21529/MIL-HDBK-978B\\_vol1.pdf](https://nepp.nasa.gov/files/21529/MIL-HDBK-978B_vol1.pdf)
- National Instruments. (2014, 03 05). *digital.ni*. Retrieved 02 19, 2015, from digital.ni.com: <http://digital.ni.com/public.nsf/allkb/2D038D3AE1C35011862565A8005C5C63>
- National instruments. (2014, 3 20). *What are Anti-Aliasing Filters and Why are They Used?* Retrieved 06 02, 2015, from National Instruments Knowledgebase: <http://digital.ni.com/public.nsf/allkb/68F14E8E26B3D101862569350069E0B9>
- National Semiconductor Corporation. (2002). *AN31 Op Amp Circuit Collection*. Retrieved from <http://www.ti.com/ww/en/bobpease/assets/AN-31.pdf>
- NXP. (2013, 07 01). *74LV4052 Datasheet*. Retrieved from [http://www.nxp.com/documents/data\\_sheet/74LV4052.pdf](http://www.nxp.com/documents/data_sheet/74LV4052.pdf)
- On Semiconductor. (2011, 2). *AND8202/D Datasheet*. Retrieved from [http://www.onsemi.com/pub\\_link/Collateral/AND8202-D.PDF](http://www.onsemi.com/pub_link/Collateral/AND8202-D.PDF)
- On Semiconductor. (2011). *Low-side self-protected MOSFET*. Retrieved from [http://www.onsemi.com/pub\\_link/Collateral/AND8202-D.PDF](http://www.onsemi.com/pub_link/Collateral/AND8202-D.PDF)
- On Semiconductor. (2012, 7). *NCV8401*. Retrieved from [http://www.onsemi.com/pub\\_link/Collateral/NCV8401-D.PDF](http://www.onsemi.com/pub_link/Collateral/NCV8401-D.PDF)
- Onsemi. (2015). *Automotive Driver Requirements, Topologies and Applications*. Retrieved from Onsemi: [http://www.onsemi.com/pub\\_link/Collateral/TND384-D.PDF](http://www.onsemi.com/pub_link/Collateral/TND384-D.PDF)
- Optrand Inc. (2009). *CALplug Datasheet*. Retrieved from <http://www.optrand.com/CALplug.htm>
- Optrand Inc. (2009). *Optrand pressure sensor*. Retrieved from <http://www.optrand.com/catalog/opcat2001large.pdf>
- Optrand Inc. (2009, 09 18). *Optrand pressure sensor*. Retrieved 03 2015, 04, from [http://www.optrand.com/autopsi\\_man091809.pdf](http://www.optrand.com/autopsi_man091809.pdf)



- Philips Semiconductors. (2000). *MMBTA42 Datasheet*. Retrieved from [http://www.nxp.com/documents/data\\_sheet/MMBTA42.pdf](http://www.nxp.com/documents/data_sheet/MMBTA42.pdf)
- PhoneGap. (n.d.). *The command-line interface*. Retrieved from PhoneGap documentation: [http://docs.phonegap.com/en/4.0.0/guide\\_cli\\_index.md.html#The%20Command-Line%20Interface](http://docs.phonegap.com/en/4.0.0/guide_cli_index.md.html#The%20Command-Line%20Interface)
- Pulse Electronics. (2011). *J0G-0001NL Datasheet*. Retrieved from <http://productfinder.pulseeng.com/products/datasheets/J423.pdf>
- Pulse Electronics Corporation. (2011). *PulseJack™ J0G-0001NL Datasheet*. Retrieved from <http://productfinder.pulseeng.com/products/datasheets/J423.pdf>
- Racing, H. O. (2015). *Hi Octane Racing*. Retrieved from <http://hioctaneracing.com/blog/2011/7/3/the-truth-about-e85-fuel.html>
- SAE International. (2000, 6 9). *Cylinder-pressure-based engine*. Retrieved from <http://am.delphi.com/pdf/techpapers/2000-01-0932.pdf>
- SAE International. (2011). *SAE J537 Standard*. Retrieved from <http://standards.sae.org/wip/j537/>
- SAE International. (2012). *SAE J1113/11 Standard*. Retrieved from [http://standards.sae.org/j1113/11\\_201201/](http://standards.sae.org/j1113/11_201201/)
- STMicroelectronics. (2011). *STPS30SM100S Datasheet*. Retrieved from <http://www.st.com/web/en/resource/technical/document/datasheet/CD00228797.pdf>
- STMicroelectronics. (2013). *LSM9DS0 Datasheet*. Retrieved from <http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/DM00087365.pdf>
- Syfer Pty Ltd. (2015, 02 10). *PetaLinux Image with Custom Application*. Retrieved from <http://syfer.com.au/assets/s502-00001-a.pdf>
- TechEdge. (2015). *WB02 DIY*. Retrieved from <http://wbo2.com/2y/lmschem.htm>
- Texas Instruments. (2005, 02 03). *Anti-aliasing filter tool*. Retrieved from <http://www.ti.com/tool/ANTIALIASINGCALC>
- Texas Instruments. (2006, 02). *Input and output capacitor selection*. Retrieved from ti: <http://www.ti.com/lit/an/slta055/slta055.pdf>
- Texas Instruments. (2011, 10). *ADS8568 Datasheet*. Retrieved from <http://www.ti.com/lit/ds/sbas543a/sbas543a.pdf>
- Texas Instruments. (2011, 10). *ADS8568 Datasheet*. Retrieved 02 17, 15, from <http://www.ti.com/lit/ds/symlink/ads8568.pdf>



- Texas Instruments. (2014, 12). *TPIC8101 Datasheet*. Retrieved from <http://www.ti.com/lit/ds/symlink/tpic8101.pdf>
- Texas Instruments. (2015). *SN65HVD234 Datasheet*. Retrieved from <http://www.ti.com/general/docs/lit/getliterature.tsp?genericPartNumber=sn65hvd234&fileType=pdf>
- Texas Instruments. (n.d.). *LM5112*. Retrieved from ti: <http://www.ti.com/lit/ds/symlink/lm5112.pdf>
- Texas Instruments. (n.d.). *SN74LVC2T45*. Retrieved from Ti: <http://www.ti.com/lit/ds/symlink/sn74lvc2t45.pdf>
- The-Crankshaft Publishing. (n.d.). *Firing Order of Cylinders (Automobile)*. Retrieved 02 19, 2015, from What-When-how: <http://what-when-how.com/automobile/firing-order-of-cylinders-automobile/>
- Trenz Electronic GmbH. (2014, 09 26). *TE0720 User Manual*. Retrieved from [http://www.trenz-electronic.de/fileadmin/docs/Trenz\\_Electronic/TE0720-GigaZee/documents/TE0720%20User%20Manual-v43-20140926\\_1036.pdf](http://www.trenz-electronic.de/fileadmin/docs/Trenz_Electronic/TE0720-GigaZee/documents/TE0720%20User%20Manual-v43-20140926_1036.pdf)
- Trenz Electronics. (2014). *High-Speed I/O*. Retrieved 02 10, 2015, from Trenz Wiki: <https://wiki.trenz-electronic.de/pages/viewpage.action?pageId=9273744>
- Trenz Electronics. (n.d.). *TE0720 User Manual*. Retrieved from Trenz Wiki: <https://wiki.trenz-electronic.de/display/TE0720/TE0720+User+Manual>
- USB Implementers Forum, Inc. (2000, 04 27). *Universal Serial Bus Specification*. Retrieved from [http://www.usb.org/developers/docs/usb20\\_docs/usb\\_20\\_031815.zip](http://www.usb.org/developers/docs/usb20_docs/usb_20_031815.zip)
- Vishay. (2012, 10 02). *Optocoupler CNY17*. Retrieved from <http://www.vishay.com/docs/83606/cny17.pdf>
- Vishay Intertechnology Inc. (2009, 08 10). *Transient Voltage Suppressors (TVS) for Automotive Electronic Protection*. Retrieved from <http://www.vishay.com/docs/88490/tvs.pdf>
- Vishay Intertechnology Inc. (2012, 06 12). *D/CRCW e3 Datasheet*. Retrieved from <http://www.vishay.com/docs/20035/dcrcwe3.pdf>
- Vishay Intertechnology Inc. (2014, 12 04). *23356 Aluminum Capacitors Introduction*. Retrieved from <http://www.vishay.com/docs/28356/alucapsintroduction.pdf>
- Vishay Semiconductor. (2014). *CNY17-3*. Retrieved from <http://www.vishay.com/docs/83606/cny17.pdf>
- Walnes, J. (n.d.). *WebSockets the unix way*. Retrieved from websocketd: <http://websocketd.com/>



- WBo2. (2012, 03 23). *Lambda Module Schematic Guide*. Retrieved from WBo2:  
<http://wbo2.com/2y/lmschem.htm>
- Wikibooks. (2014, 11 1). *Analog and digital conversion*. Retrieved from  
[http://en.wikibooks.org/wiki/Analog\\_and\\_Digital\\_Conversion/Resolution\\_and\\_Bitrate](http://en.wikibooks.org/wiki/Analog_and_Digital_Conversion/Resolution_and_Bitrate)
- Wikimedia Foundation Inc. (2013, 05 17). *Sampling*. Retrieved 02 06, 2015, from Wikipedia:  
<http://nn.wikipedia.org/wiki/Sampling>
- Wikimedia Foundation Inc. (2014, 02 11). *Control Systems/Examples/Second Order Systems*. Retrieved 02 07, 2015, from Wikibooks:  
[http://en.wikibooks.org/wiki/Control\\_Systems/Examples/Second\\_Order\\_Systems](http://en.wikibooks.org/wiki/Control_Systems/Examples/Second_Order_Systems)
- Wikimedia Foundation Inc. (2014, 08 19). *Sampling*. Retrieved 02 06, 2015, from Wikipedia:  
[http://no.wikipedia.org/wiki/Sampling\\_%28signalbehandling%29](http://no.wikipedia.org/wiki/Sampling_%28signalbehandling%29)
- Wikimedia Foundation Inc. (2015, 03 01). *Black-box testing*. Retrieved from Wikipedia:  
[http://en.wikipedia.org/wiki/Black-box\\_testing](http://en.wikipedia.org/wiki/Black-box_testing)
- Wikimedia Foundation Inc. (2015, 02 27). *CAN bus*. Retrieved 03 05, 2015, from Wikipedia:  
[http://en.wikipedia.org/wiki/CAN\\_bus](http://en.wikipedia.org/wiki/CAN_bus)
- Wikimedia Foundation Inc. (2015, 01 31). *White-box testing*. Retrieved from Wikipedia:  
[http://en.wikipedia.org/wiki/White-box\\_testing](http://en.wikipedia.org/wiki/White-box_testing)
- Wikimedia Foundation Inc. (n.d.). *Flyback diode*. Retrieved from Wikipedia:  
[http://en.wikipedia.org/wiki/Flyback\\_diode](http://en.wikipedia.org/wiki/Flyback_diode)
- Wikimedia Project. (2013). *Sample*. Retrieved from wikipedia:  
<http://nn.wikipedia.org/wiki/Sampling>
- Workware. (n.d.). *Overview*. Retrieved from  $\mu$ web:  
<http://www.workware.net.au/workware/index.html>
- Xilinx 2014. (2014, 01 02). *Hardware Platform Specification*. Retrieved 03 13, 2015, from  
[http://www.xilinx.com/support/documentation/sw\\_manuels/xilinx2014\\_4/SDK\\_Doc/concepts/sdk\\_c\\_hwspec.htm](http://www.xilinx.com/support/documentation/sw_manuels/xilinx2014_4/SDK_Doc/concepts/sdk_c_hwspec.htm)
- Xilinx. (2014, 11 24). *Xilinx All Programmable*. Retrieved from www.xilinx.com:  
<http://www.xilinx.com/support/download.html>
- Xilinx All Programmable. (n.d.). *PetaLinux Tools*. Retrieved from www.xilinx.com:  
<http://www.xilinx.com/tools/petalinux-sdk.htm>
- Xilinx Inc. (n.d.). *Install Xilinx Tools*. Retrieved from Wiki Xilinx:  
<http://www.wiki.xilinx.com/Install+Xilinx+Tools>



- Xilinx Inc. (2015, 01 16). *DS593 Platform Cable USB II Datasheet*. Retrieved from [http://www.xilinx.com/support/documentation/data\\_sheets/ds593.pdf](http://www.xilinx.com/support/documentation/data_sheets/ds593.pdf)
- Xilinx Inc. (2015, 02 04). *UG480 Zynq-7000 AP SoC XADC*. Retrieved from [http://www.xilinx.com/support/documentation/user\\_guides/ug480\\_7Series\\_XADC.pdf](http://www.xilinx.com/support/documentation/user_guides/ug480_7Series_XADC.pdf)
- Zaremba, D. (n.d.). *AND8202/D*. Retrieved 02 18, 2015, from Onsemi: [http://www.onsemi.com/pub\\_link/Collateral/AND8202-D.PDF](http://www.onsemi.com/pub_link/Collateral/AND8202-D.PDF)
- Zeitronix. (2015). *Zeitronix Ethanol Content*. Retrieved from <http://www.zeitronix.com/Products/ECA/ECA.shtml>
- Zeitronix. (n.d.). *Zeitronix Ethanol Content Analyzer*. Retrieved from <http://www.zeitronix.com/Products/ECA/ECA.shtml>



KONGSBERG



## Test & After-analysis report

### KPEC

<b>Employer</b>	Kongsberg Defence & Aerospace			
<b>Group Members</b>	<b>Name</b>		<b>Initials</b>	
	Ola Pedersen Aasheim		OA	
	Salahuddin Asjad		SA	
	Hung Dinh		HD	
	Dler Hasan		DH	
	Even Gudbrandsen		EG	
	Jannik Schäffer		JS	
<b>Document information</b>	<b>Revision</b>	<b>Date</b>	<b>Approved</b>	<b>Pages</b>
	2.0	18.05.2015	OA	32





## Abstract

The test & after-analysis report describes results encountered during the implementation, test & verification phase of the KPEC project. This document provides the reader with test results, general and individual evaluation of the work performed during the bachelor project. The test report will focus on the successful test results, as well as a detailed discussion of the circumstances of the tests that failed. Testing is important in engineering development, and saves project overall cost and quality dramatically when done correctly. This document is closely linked to the test plan.





## Revision Table

Version	Date	Approval	Description
1.0	06.05.2015	JS	<ul style="list-style-type: none"> <li>Created the document</li> </ul>
2.0	18.05.2015	OA	<ul style="list-style-type: none"> <li>Revised and published</li> </ul>

## Contents

Abstract .....	2
Revision Table .....	3
List of figures.....	4
List of tables .....	4
1. Test report.....	5
1.1. Product overview .....	5
1.2. Testing procedures.....	5
1.2.1. Static testing.....	6
1.2.2. Review meetings.....	6
1.2.3. Simulations and $\alpha$ -testing.....	6
1.2.4. Excluded tests .....	7
1.3. Test results.....	7
1.3.1. Successful test requirements.....	7
1.3.2. Aberrations.....	8
1.4. Recommendations.....	11
2. After analysis.....	12
2.1. Administrative analysis .....	12
2.1.1. Time estimation .....	12
2.1.2. Scheduling.....	12
2.1.3. Meetings .....	13
2.1.4. Presentations .....	13
2.2. Technical solutions analysis .....	14
2.2.1. Hardware design .....	14
2.2.2. Software design .....	15



2.3.	Total project evaluation.....	15
2.4.	Individual evaluation .....	16
2.4.1.	Even Gudbrandsen .....	16
2.4.2.	Ola Pedersen Aasheim .....	17
2.4.3.	Hung V. Dinh .....	18
2.4.4.	Jannik B. Schäffer .....	19
2.4.5.	Salahuddin Asjad .....	20
2.4.6.	Dler Hasan.....	21
3.	Conclusion.....	22
	Appendix I – Requirements test results.....	23
	Appendix II – Time estimation versus actual work .....	26
	Appendix III – Bill of materials .....	29

## List of figures

Figure 1 - Tests planned vs. Tests executed .....	5
Figure 2 - Requirement 091-HW4OA failed the test.....	8
Figure 3 - Requirement 108-HW4OA failed the test.....	9
Figure 4 - Requirement 1081-HW4OA failed the test .....	10

## List of tables

Table 1 – Requirements test results.....	23
Table 2 – Detailed requirements test results .....	25
Table 3 - Time estimation versus actual work.....	26
Table 4 - BOM created in OrCAD Allegro to view component parameters .....	29



## 1. Test report

Testing is executed in order to reveal defects within the circuit and software design. The KPEC ECU is based on a thorough requirements and test specification defined before the design process. The tests performed in this document are therefore linked to the test specification where all requirements have status fields which reveal the requirement status. Some requirements are tested as static test, and some are not performed due to long term goal. Long term goal requirements should be tested when the complete system are implemented in a PCB design.

### 1.1. Product overview

The KPEC Engine Control Unit is a high-end product designed for interfacing petrol engines with up to 8 cylinders for use in motorsports applications. The KPEC ECU is a motherboard design for interfacing the TE0720 micromodule, a Zynq SoC based processing platform for embedded applications. The KPEC ECU is a highly user configurable, high performance programmable ECU. Engine tuning and parameter real time view is achieved wirelessly via Wi-Fi communication directly to the user's tablet device through a dedicated application.

The KPEC ECU can interface a vast array of automotive performance sensors, with 17 different sensor types of interfaces for up to a total of 44 sensors. There are 5 different high power drivers for a total of 24 low side switched actuator drive output stages.

### 1.2. Testing procedures

The tests performed at this point of the project lifecycle are performed as static testing. The design document has a thorough walkthrough of each part of the schematic and software design, where simulations are performed and described to verify the design. Figure 1 gives a graphical view of the tests that has been executed in green, and the tests that need to be performed when the product is finished are shown in blue. Red indicates requirements that failed when tested.

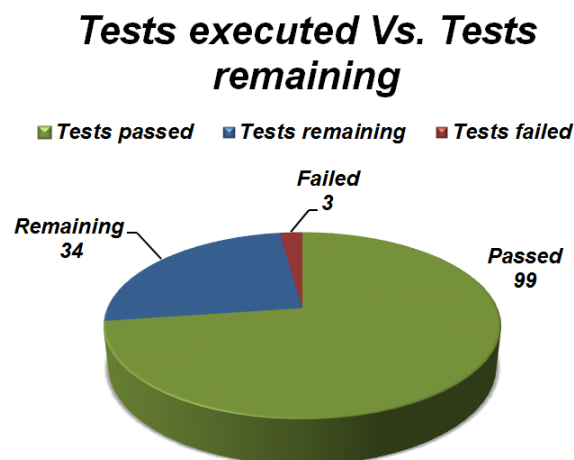


Figure 1 - Tests planned vs. Tests executed



### 1.2.1. Static testing

Static testing was performed during week 19 - 20 in the test phase of the project. Static testing was accomplished thru visual inspection of the design compared to the requirement specification according to the test specification document. The intention of the static testing is to check the requirement at hand against the designed feature for completeness and deviations. Iterative static testing increases the chance of project success when bugs and design glitches are revealed, are relatively cheap and easy compared to other alternatives.

### 1.2.2. Review meetings

Review meetings were held in between design iterations, and repeated several times during the design phase in accordance to the iterative project model. Review meetings include all the project members where each designer presents his thoughts on the sub design decisions for the other members to reflect and receive feedback on the design. Each review meeting is documented with an iteration report which contains review points of the current design to enhance the design.

### 1.2.3. Simulations and $\alpha$ -testing

Simulations and  $\alpha$ -testing are related to each other; however,  $\alpha$ -testing is best described as an internal acceptance test of the design for the team members. The KPEC team have performed both simulations and  $\alpha$ -testing during the project.

Simulations proved to be useful during the design development where practical testing were not possible. Physical behavior of the circuits, electrical characteristics etc. were implemented in a mathematically based simulation, using software provided by Linear Technology called LTSpice<sup>1</sup>.

The Embedded Linux operating system (PetaLinux) has only been tested on the TE0720. Software has performed their tests in level-structure, which means that the most vital tests are executed on a physical device to show real environment results, and the less important tests ran on a computer.

The software code is version controlled and well-documented so future developers can easily access the code on GitHub. The software engineers have done over 100 commits to increase the traceability. A graphical hierarchy is made on GitHub, which shows the folder structure, to make it easy for future developers to find the corresponding files.

Simulations are extremely powerful when the designer is aware of the boundaries of the simulation, unfortunately the models that form the foundation for the physical representation are not always either complete or available in its full extent. Simulations

---

<sup>1</sup> LTSpice IV 4.22n



are time consuming, some hardware manufacturers offer online tools in order to provide fast to market solutions.

#### **1.2.4. Excluded tests**

Many of the test features cannot be tested at this stage of the project, and should be succeeded along with the continuance of the project. Such features include dynamic testing, regulation testing when the ECU is operational, communication, tuning etc.

### **1.3. Test results**

The following section describes the test results encountered during the project. For individual test result of each requirement, consult Appendix I in this document.

#### **1.3.1. Successful test requirements**

The KPEC team performed tests of the short term requirements of the project according to their acceptance criteria. The test results uncovered that only 3 out of 133 tests failed, where two of them were project requirements and one were internal requirements. The importance of the requirements are scaled from 1 to 5 where 1 is the most important. Risk evaluation of three failing requirements indicates that the ECU does not predict any severe functional deficiencies.

There are still 34 requirements that remain untested, due to the fact that they are long-term goals and are not a part of this project. A second iteration of testing evaluation will be necessary when the design has been realized into a printed circuit assembly.



### 1.3.2. Aberrations

Requirement 091-HW4OA about electrostatic discharge protection failed the test as shown in Figure 2.

091-HW4OA	Electrostatic Discharge (ESD) protection	06.05.2015
Status	Failed	
Acceptance Criteria	All external headers or connectors have ESD protection that meets or exceeds 15kV contact model.	
Test	Consult the schematics that all headers or connectors have either components with integrated ESD measures or dedicated ESD components such as TVS diodes/arrays that meets or exceeds 15kV contact model.	
Comments	All signal tracks have ESD protection, but the power lines are not directly protected with any ESD measures.	

**Figure 2 – Requirement 091-HW4OA failed the test.**

The acceptance criteria states that all external headers and connectors should have ESD protection.

However, a protection circuitry has been developed to protect the ECU from reverse polarity, load dump transients of up to 150 V and overcurrent/short circuit. The challenge is that no general solution is offered to protect power lines of ESD known by us or KDA. There are however custom (expensive) solutions available on the market provided by companies such as Protek devices.



Requirement 108-HW4OA about electronic discharge protection also failed the test as shown in Figure 3.

108-HW4OA	Electrostatic Discharge (ESD) protection	06.05.2015
Status	Failed	
Acceptance Criteria	All external headers or connectors have ESD protection that meets or exceeds 25kV air discharge model.	
Test	Consult the schematics that all headers or connectors have either components with integrated ESD measures or dedicated ESD components such as TVS diodes/arrays that meets or exceeds 25kV air discharge model.	
Comments	All signal paths have ESD protection, but the power lines are not protected.	

**Figure 3 - Requirement 108-HW4OA failed the test**

Requirement 091-HW4OA is similar to requirement 108-HW4OA, requirement 091-HW4OA states that 15 kV contact protection is necessary while requirement 108-HW4OA states that 25 kV air protection is necessary. All signal lines which are directly connected to the user are ESD protected, however, not all the protection circuitry handles 25 kV, but can manage up to 4 kV. The ECU will primarily be used in an EPA environment which reduces the risk of damage to the system, and hence can lower the formal protection level.



Requirement 1081-HW4OA regarding component temperature range also failed its test as shown in Figure 4. This requirement is succeeded from initial requirement 081-HW4OA. This requirement is derived from the electronics design standards provided by our customer KDA.

<b>1081- HW4OA</b>	<b>Industrial temperature range</b>	<b>08.05.2015</b>
<b>Status</b>	Failed	
<b>Acceptance Criteria</b>	All components used must be meet or exceed the industrial temperature range: -40°C to 85°C.	
<b>Test</b>	Verify that all components in BOM meets or exceeds operational temperature of -40°C to 85°C (Industrial temperature range).	
<b>Comments</b>	Of all the 1540 individual components, only one failed the temperature requirement. The RJ45 connector J0G-001NL has the commercial temperature rating; 0-70 °C	

**Figure 4 - Requirement 1081-HW4OA failed the test**

This test was performed in Cadence Allegro<sup>2</sup>, where we created a Bill Of Materials (BOM) report shown in Appendix III. The report revealed that 25 components were not listed for the Industrial or Military temperature range, which are respectively -40 - 85 °C and -55 - 125 °C. Of the 25 components, only the RJ 45 connector turned out to be rated for the commercial temperature range; 0 - 70 °C. The requirement is rated 4 on a scale of 1 to 5, where 1 is absolutely necessary and 5 is nice to have features.

The consequence that the RJ45 port is not able to meet the temperature requirement is not expected to decrease the performance of the ECU. The RJ45 port is implemented to enable debugging and reconfiguration of the ECU, however such activities should be carried out in an ESD/EPA protected area with qualified personnel only.

<sup>2</sup> Allegro Design Entry CIS 16.6-S038





## 1.4. Recommendations

Tests have revealed that the product has been designed in a viable matter so far, it's necessary to distinguish between short-term and long-term requirements for this project;

We have now executed the short-term tests for the project where the long-term tests are succeeded along with the continuance of the project. We are happy to discover that only 3 out of 133 tests failed, where two of them were project requirements and 1 where internal requirements. The importance of the requirements is estimated as 4 on a scale from 1 to 5 where 1 is the most significant. Risk evaluation of three failing requirements indicates that the ECU does not predict any severe functional deficiencies.

There are still 34 requirements that remain untested, they are long-term goals and are not a part of this report. Another evaluation will be necessary when the design has been realized into a printed circuit board.

The impression KPEC has is that we have exceeded our stakeholder KDAs expectations, after the external review meeting where conducted at their facilities. The test phase has been a success.

As mention initially in this report, the test phase is by no means over. Important  $\beta$ -testing, factory acceptance test, customer acceptance test will be necessary at some point in the continuance of the project. To ease troubleshooting, and testing, we are passing forward the simulation files from LTSpice, for which the simulation circuit can be used to gather knowledge about node voltages, currents and signal behavior. In the schematic design, there have been implemented test points throughout the circuit, the test points can be used to compare the physical design with the simulation.

Another feature that should be tested is temperature vulnerable components like MOSFETs and high frequency devices. In the design document, some worst case temperature calculations has been made, the temperature can be measured with a IR thermal camera and compared with the theoretical values for an ECU that is operational.



## 2. After analysis

This section provides information regarding evaluation of the bachelor project KPEC. The first section will cover administrative and technical solution analysis, while the second section will give an individual evaluation of the effort sacrificed to accomplish this project.

### 2.1. Administrative analysis

#### 2.1.1. Time estimation

The original time estimation were performed during the autumn of 2014, two months before official project startup and the project manager based his assumptions on previous bachelor projects and the project model. This estimation were based on 600 hour work per student and this gives 3600 hours in total, which can be seen in the vision document<sup>3</sup>.

However, after first presentation and initiation of the design phase of the project, the project manager experienced that the design of the system were challenging, and the administrative tasks were more time consuming than expected. To correct for this, the group decided to reduce the assignment and our project model to a more agile and iterative model. The resource estimation was changed to focus more on the design, and we added an iteration plan. A detailed overview of the estimated work versus actual work could be found in the Appendix II.

#### 2.1.2. Scheduling

The group realized in January that the projects were not going to be finalized on schedule due to the extent of the deliverables. We contacted our employer KDA. This was a tough decision, but later proved to be the only right thing to do. KDA made it clear to us that any deliverables should be of a certain quality rather than delivering quick design solutions. The group and KDA agreed on that the PCB layout design should be performed at a later stage of this project. The main deliverables where then reduced to schematic design.

The project has been rescheduled four times due to design challenges. A key tool to monitor the progress has been Microsoft Project<sup>4</sup> with weekly reports, as well as Paymo<sup>5</sup> and the iterations performed as part of the project model. The project started out with some basic tasks in our activity lists with estimated work hours. This way of scheduling where not ideal, as the progress of the project were hard to monitor.

The project manager defined iteration intervals, and then made weekly activities for the group to focus on in collaboration. The team gained better progress by thorough design

---

<sup>3</sup> KPEC 2015, Gudbrandsen, Pedersen, Schäffer, Dinh, Hasan, Asjad, Vision document (Rev 4.0), HBV

<sup>4</sup> Microsoft Project 2013 15.0.4709.1000

<sup>5</sup> Paymo Paymoapp 3



iterations. Deliverable list were generated for the next review meeting. The first iteration had therefore a long interval of 5 weeks prior to the introduction of iterations.

### **2.1.3. Meetings**

Whiteboard meetings were held each Monday morning by the project manager to inform the team members of upcoming tasks and late tasks. Internal review meetings were held during the design process where requirements, schematics and software code were reviewed in collaboration. Weekly meetings with our internal advisor has been held each Thursday with status report and technical issues.

The group has been in one external review meeting with our employer KDA. This meeting was particularly useful due to the fact that four experienced engineers reviewed our schematic design and gave us feedback regarding possible issues. The software department had an external session with our external sensor and advisor where they presented their tablet application and software design. All meetings (status/review) has a minute which documents each decision and topics that has been addressed.

### **2.1.4. Presentations**

The group has performed two public presentations at this time; the third presentation will be accomplished after the documentation is handed in. The first presentation was held 29.01.15 at D101, and the second presentation was held 19.03.15 at D101.

During the first presentation, we presented the group and the product. The group focused on the concept of operations and the administrative planning phase of the project. Attendance at the first presentation was very good both from students and other resources from the campus. The group got feedback regarding project model and documentation which has been revised after the presentation. The group's overall evaluation of the first presentation is that it went successfully.

The second presentation focused on the current status of the project and current technical solutions (for software and hardware) to the assignment. Attendance at the second presentation was very good both from students and other resources from the campus. The group received positive feedback regarding the current status, and was encouraged to keep up the work. The group's overall evaluation of the second presentation is that it went successfully.

The third presentation will be held 02.06.15, 11.30 at B120 HBV Kongsberg. This presentation (40 minutes) will contain on a sales pitch of the KPEC ECU, as well as focus on technical solutions and the project management process.



## 2.2. Technical solutions analysis

### 2.2.1. Hardware design

The hardware design process was initiated after a thorough walkthrough of the concept of operations and the requirements specification. The group generated a detailed overview of the sensors and actuators that needed to be interfaced towards the TE0720. Each interface was distributed among the hardware group with one week deadline. Some of the interfaces (ADC, filter design and lambda interface) required more resources than estimated, and were especially challenging due to complexity.

Bosch motorsport was defined as a baseline manufacturer of sensors and actuators, which we wanted to interface. Their physical characteristics are equivalent with other suppliers which makes the ECU versatile. Bosch provides datasheets for each sensor and actuator with physical and electrical characteristics, but without application circuitry. Bosch utilizes a lot of resources regarding market regulation and restrictions regarding sales of components for small scale projects. This made some of the design process especially challenging.

The lambda sensor interface was first based on a special IC produced by Bosch due to the complexity of the sensor. The group contacted (by telephone and mail) Bosch sales offices in Norway, Germany and Canada, where all rejected our request to purchase low quantity of this IC. Bosch and their subsidiary distributors do not trade with projects or companies related to the defence industry.

We needed to reverse engineer the IC according to functional descriptions in datasheets (for IC and sensor), as well as blueprints and multiple master thesis regarding Lambda sensors. The circuits were reconstructed in LTSpice, and simulated to unveil their electrical properties. The analog circuits were then analytically adapted to interface the Zynq XADC. The functionality of the interface was verified through advanced analysis, to ensure that the circuitry works as intended.

The active filter design towards the sensors were very challenging due to lack of experience of filter design, and little external resources available regarding practical implementation of filters. The resources were extended to two design engineers who proved to accomplish this task and verify the functionality of the filter.

The hardware team has accomplished all the assignment goals, and provided a comprehensive appendix for ECU connectivity. This section makes the practical implementation of the ECU easier for the end user with connection tables, signal description and general fitting instructions. The hardware team's impression is that we have exceeded the expectations of our stakeholders, and are really proud of the end result.



### 2.2.2. Software design

The software design process was initiated after a thorough walkthrough of the concept of operations and the requirements specification. The group generated a detailed overview of the sensors and actuators that needed to be interfaced towards the TE0720. The software developers have cooperated on the tasks such as installing a complete embedded Linux operating system (PetaLinux) on the TE0720, and build an Android application which gives the user the ability to analyze real time parameters from the TE0720. This took more time than estimated, and were especially challenging due to complexity.

In order to fulfill the requirements as much as possible, the software group worked in parallel with one week deadline on each task. We solved our problems by communicating with Trenz Electronics and Xilinx, which gave us valuable information. We had to configure the operating system (PetaLinux) properly in order to build user applications; we spent a great amount of time on configuring the operating system (PetaLinux). This made some of the design process especially challenging.

The process of building a dedicated operating system for TE0720 is not a straightforward procedure, which our stakeholder KDA predicted they should assist given the advanced level of technology. The software department has faced several challenges with both PetaLinux and application development, commitment and hard work have enabled them to solve their obstacles without any direct help from KDA.

The software team has accomplished all the assignment goals successfully, included most of the optional goals. We have implemented and configured PetaLinux in our TE0720, and build an interactive Android application, which interfaces the TE0720 sending data in real-time with Ethernet communication. The software team's impression is that we have exceeded the expectations of our stakeholders, and we are really proud of the deliverables.

### 2.3. Total project evaluation

Section 2.1 and 2.2 described the hardware and software design processes, as well as challenges we had to handle during this project. The design document describes thoroughly the design decisions made in the schematic and software design with theory and simulations as verification. The design of the KPEC ECU has been derived through practical experience and learning by doing approach. It is important to emphasize that the majority of what we have accomplished in this project is self-taught. This is a direct cause of commitment and hard work, of which we are very proud of.



## 2.4. Individual evaluation

### 2.4.1. Even Gudbrandsen

My role during this project has been the project manager & quality responsible. I have also performed hardware design of lambda sensor interface, digital interface, PVM communication and ECU connectivity. The group were established during the late summer of 2014, where we sent an application to KDA regarding the possibilities for performing a bachelor thesis. We received a positive response with an exciting interdisciplinary tasks for both cybernetics and computer engineers, and initiated the planning phase of the project in the beginning of October 2014.

This assignment has undoubtedly been the most challenging assignment during the bachelor study for me. The group work has been challenging due to lack of practical experience regarding electronics design and project management. I have a technical background as a certified vehicle technician and have good experience with automotive electronics. This project has really tested my knowledge level, as well as it has been extremely exciting due to the fact that automotive electronics and technology is my main interest.

My role as a project manager has been to keep a systematic overview of the hardware and software part of the project. This role has been very challenging; especially to keep track of progress, as well as strategic planning to ensure that all our goals were met. I have encouraged the team members to perform the required work with enthusiasm, as well as making decisions regarding design choices and other issues which enrolled during the project. The group has worked jointly on our office during the whole project; my opinion is that this has raised the quality of the product.

My personal learning outcome from the bachelor project has been the analytical approach to solving complex problems with the help of project management tools. The project model proved to be extremely helpful with the design iterations, as well as milestones which ensured propulsion in the project. Another aspect of the project was the realistic experience of a development project for a professional customer. Some of the challenges required external expertise, so I have been in telephone contact with Bosch (Germany), Molex (Norway), Continental (Germany) and Optrand (USA). This project has enhanced my communication skills, as well as extended my contact network.

I will highly recommend other students to work with an interdisciplinary project. I've gained a much better understanding of the software development process, and their perspective of hardware. The communication with the internal and external advisor has been excellent. I have sacrificed a lot of dedication and working hours to this project, which makes me extra proud of the final result we have accomplished. This has been a great period of my life, and I feel very fortunate to been allowed to be member of such a resourceful group.



#### 2.4.2. Ola Pedersen Aasheim

My role in KPEC is *Hardware Design and Layout responsible*, due to my prior knowledge about electronics design from LocalHawk 2014. The thesis is done in cooperation with the same department I was employed by during the project. It has been a challenging, yet exciting role in the project.

All electrical engineers in the group has worked large parts of the hardware design, and as the responsible I have been required to understand every hardware design aspect to quite some detail, partake in design discussion across all subsystems and delegate tasks. One of the greatest challenges with this role is the lack of experience and prior knowledge when delegating work. When I and other group members have designed something entirely new, it is almost impossible to estimate the amount of hours required to finish the task.

The cooperation with our employer has been nothing but pleasant. We have had several meetings both externally and in our office with external advisor, sensor and other KDA employees over the course of the project. They have granted us guidance where needed, but in all major cases I would claim the actual design work and decisions have been entirely up to ourselves.

An interdisciplinary project is something I will highly recommend to other students when doing their bachelor thesis. I see myself as a hobbyist developer within both programming and electronics with previous programming experience. This makes it exciting to work alongside software engineers, as the hardware design choices we make highly affects their work when creating software.

The Systems Engineering approach has been very helpful designing hardware for future use and development. I feel that I have managed to keep close cooperation with our software engineers, much due to the interest I in general carry for software design. The different way of thinking expressed in software engineering contrary to hardware engineering, opens up creative solutions to hardware problems when addressed from a software engineer's viewpoint.

The only thing I really wanted more of in this project is time. I would love to continue development of the KPEC ECU until project completion of the circuit board and software. It is very exciting to be a founding member of a project that is likely to be seen and worked on by many people in the future over several years.



### 2.4.3. Hung V. Dinh

I have a technical background as a certified production electronics technician. My responsibilities during this project were to make sure that the preliminary studies was performed and documented. Additionally I was also responsible for modelling and simulations during the project. As the project progressed I have also been given the responsible to make sure that the documentation is written accordingly to KPEC standard. My main hardware responsible was to design and implement the driver circuitry and power design (in cooperation with Jannik), with some design implementation towards sensors.

I have only good experiences with working in group with other disciplinarians. I have learned a lot about software related topics, such as PetaLinux, WebSocket etc. In my opinion our group have worked really well, we have had discussions and helped each other to be able to achieve a mutual goal. I feel like that we as a group have worked incredibly well, all members were motivated and gave everything to finish our goals.

To follow a project model was a bit weird at first, but I quickly recognized how important it was to have a good model. The model gave our group a structured way to work, starting with research, decisions, simulation/test and finally to implement. The project model enabled us to be able to learn from our mistakes, thus helping us progress towards our goal.

Our communications towards our internal and external advisors and sensor have been amazing. We are having regular contact with our external advisor through email and in person. We received tips and guidance when we asked for it. During the project we had weekly meetings with our internal advisor gave us the ability to get guidance for documentation and technical aspects, this was something that we really appreciated.

During our design of the ECU we documented in parallel in the design process. We ran simulations and test on every circuit that was possible. The design process started with research, then component selection, simulation and test and then finished with the schematic implementation.

Some of the biggest challenges for us were to get enough information about specific components that we needed. It was fairly difficult sometimes due to confidential information. Other challenges was that we did not have the required expertise, as a consequence of this we had to spend much time to expand our knowledge in order to complete our goals. In the end I think that our group has done a splendid job, we have done the research and have managed to complete a finished design of the ECU.

This project has really given me a real life experience on how it is to work in an engineering environment, it was challenging, difficult but also really rewarding. I believe that this experience will benefit me in future career.





#### 2.4.4. Jannik B. Schäffer

I've been the test and verification responsible in this project, in addition to hardware engineer. I have an interdisciplinary interest for electronics and mechanics.

My learning outcome as a result of this project has been extraordinary, to work in an interdisciplinary environment with 5 dedicated fellows for more than 6 months, has been both challenging and rewarding. There are some strong personalities on this team, and the discussions occasionally reached new heights when we we're struggling to solve a problem. The landscape working environment and strict scheduling, brought us closer as a team, I 'am confident that this project is realized through the dedication that this group has been able to conduct.

We adapted the project model to fit our project, with our interpretation of the V-model. We created an iterative evolutionary agile model that I feel encouraged progress rather than constrain us.

In the initial phase of the project, we conducted project planning which helped us to navigate the direction of this project, we established structure and boundaries for our project, but as we anticipated not all tasks are equally easy to estimate, rescheduling was necessary several times over the duration of the project, internal structure enabled us to follow up on late tasks in order to achieve the deliverables.

That the project has been conducted in a realistic matter will be useful experience when we are eventually entering the job market. We have a real customer, who delivers global world-leading technology. To develop viable technology, we had to reach out of Norway. During the project we have discussed solutions with distributors in Switzerland, Germany and England, in addition to Norwegian distributors. Some of the distributors have even taken the effort to send us free samples of their products, which we appreciate and will keep in mind for the future.

KDA have been the ultimate stakeholder in my opinion, they have dedicated equipment and resources to assist the bachelor thesis. To have KDA developers to evaluate and provide feedback on the design has been valuable experience to us in the position as project engineers.



#### 2.4.5. Salahuddin Asjad

In the bachelor project, I have the role as software design & interface responsible. I have been working on both the development of the Zynq module (TE0720) design and the Android application. In partnership my colleague Dler Hasan, I have also developed the website. Since we are only two software engineers in the group, we have been working closely together with regular internal iterations. Both software- and electrical engineers, have been working hard together with a good communication level, to reach the goals required by Kongsberg Defence & Aerospace.

The project work has helped me by setting personal goals and deadlines. We have also had four main iterations, where we have reviewed each other's work. The feedback from the group has helped me to quality assure and re-think some aspects of my work to make it even better. It is important to have a structured project model that helps for the project development. And I think our project model has done that.

We have mostly been working with the documentation in parallel with the development of both Zynq module and the Android application. Since this project will last for several years, the documentation is done in a detailed level to help the future developers to start from where we left. I have used the external resources carefully, and only used trusted resources to avoid potential incorrect material.

The communication towards both the external and internal stakeholders have been significant. The internal and external advisors have provided with the required advises and support. Because of the advertising from t-shirts and stickers, many people around have also noticed about our bachelor project, which I think is a positive sign for future development of this project.

Since we haven't learned about either Linux or Android application development during the previous engineering courses, it has given us several interesting challenges during the project. Due to hard work, we have proudly managed to solve the challenges we have met with minimal help from others.

I'm very happy with the energy and effort we have put into this bachelor project. This project has given me new ways to think and also a very positive understanding of how it is to work in a multidisciplinary environment. I'm confident that I will have use for what I have learned during this project.



#### 2.4.6. Dler Hasan

I have mainly had the responsibility of marketing and requirements, and to keep a strong traceability of requirement specification and test specification documents. Other tasks are design and layout structure of the website, implementing and configuring the Embedded Linux operating system (PetaLinux). As well as building a graphical user interface for an Android application and setup real-time communication between the TE0720 and the tablet, and create visual diagrams in order to keep a relationship between hardware and software.

I believe this group experience has given me the opportunity to work gladly with other engineers in the future. Everyone has taken the responsibility without hesitations, and been working constantly towards completing different tasks. Systems Engineering approach has taught me how to design and manage complex parts of KPEC by following the iterative project model. The project model has kept software- and hardware engineers close during the development, and this given me a clear and understandable perspective of how to work with hardware engineers in the future.

Every design decision are described briefly so customers and future engineers that want to start on the project can easily-understand. I have inserted all the challenges and problems I have encountered during the design phase while documenting it in parallel. Every source that I have been using, I've sourced, and can easily be find below the document.

My biggest challenge was the process of implementing and configuring PetaLinux on TE0720, because most of the information on Trenz-Electronic wasn't updated. However, after establishing a good communication with Trenz-Electronic, we managed to solve the challenges efficiently. Another challenge was to increase the Android application performance on the tablet, but we manage to solve the challenge just by doing some adjustment in the code, we then increased the Android application performance by 544%!

I have received valuable and meaningful information both from our internal- and external advisors, which have helped me in becoming a better engineer. I am well satisfied over working with collaborative team members, and I believe that this project would not come to its existence without each member contribution. This project has given me valuable lesson; which how to solve a difficult challenge just by following a structured and organized plan. I would gladly continue working in this project and look forward to see how far this project will go, and of course my team members in the future.



### 3. Conclusion

In conclusion, this document describes the results that KPEC has been encountering during the implementation, test & verification phase of the project. The test report has been focusing on both the successful and less successful test results, as well as a detailed discussion of the circumstances, and the product overview referenced to a statistical diagram which show tests that are executed versus tests remaining.

We have described four important test principles: static testing, reviewing meetings, simulation and  $\alpha$ -testing. We have also mentioned excluded tests which were not relevant for KPEC's project life-time. KPEC has been working constantly by following a well-structured and iterative project model. We are well satisfied over the accomplishments we have achieved, whereas mentioned only 3 out of 133 requirements have failed, with each of them discussed in detail. Due to the limited duration and the complexity of this project, we have clearly distinguished between short-term and long-term requirements.

After analysis covers the administrative and technical solution analysis and the effort sacrificed to complete the initial part of this project. Software and hardware has written about the challenges and difficulties they have been facing, and how they were dealt with. The individual evaluations of the project clearly reflect an overall satisfaction with the task and performance of the group. Without doubt, the project has been a great learning opportunity for the group. We feel that the interactions with our advisors and sensors have been very enjoyable, as they have shown great interest in following the project providing help underway.

We strongly believe that we have delivered beyond initial expectations due to our hard work. The result of this project is reflected by the performance of each individual in KPEC, which have been working hard and cooperatively every day to overcome challenges. We are very proud and satisfied of what we have been able to achieve.



## Appendix I – Requirements test results

Table 1 – Requirements test results

Requirement ID	Test status	Date	Responsible
001-SY1EG	Changed; Succeeded by 112-SY1EG	03.03.2015	HD
002-SY1EG	Passed	06.05.2015	JS
003-SY1EG	Passed	07.05.2015	JS
004-SY1EG	Passed	07.05.2015	JS
005-SY1EG	Not tested	12.01.2015	JS
006-SY1EG	Passed	07.05.2015	JS
007-SY3EG	Not tested	12.01.2015	JS
008-SY3EG	Not tested	12.01.2015	JS
009-SY3EG	Not tested	12.01.2015	JS
010-SY3EG	Passed	07.05.2015	JS
011-SY3EG	Passed	07.05.2015	JS
012-SY3EG	Passed	07.05.2015	JS
013-SY4EG	Not tested	13.01.2015	JS
014-SY4EG	Not tested	13.01.2015	JS
015-SY5EG	Passed	07.05.2015	JS
016-SY3EG	Passed	07.05.2015	JS
017-SY4EG	Passed	07.05.2015	JS
018-SY4EG	Passed	07.05.2015	JS
019-SY5EG	Not tested	14.01.2015	JS
020-SY3EG	Passed	07.05.2015	JS
021-SY3EG	Not tested	14.01.2015	JS
022-SY4EG	Not tested	14.01.2015	JS
023-SW4SA	Passed	06.05.2015	DH
024-SW3SA	Passed	06.05.2015	DH
025-SW5SA	Not tested	12.01.2015	DH
026-SW2SA	Passed	06.05.2015	DH
027-SW2DH	Changed; Succeeded by 113-SWDH	03.03.2015	HD
028-SW5DH	Passed	06.05.2015	DH
029-SW4DH	Changed; Succeeded by 1029-SW4DH	13.01.2015	DH
030-SW4SA	Changed; Succeeded by 1030-SW4SA	13.01.2015	DH
031-SW3DH	Not tested	13.01.2015	DH
032-SW3SA	Not tested	13.01.2015	DH
033-SW3DH	Not tested	13.01.2015	DH
034-SW5SA	Changed; Succeeded by 1034-SW5SA	13.01.2015	DH
035-SW5SA	Changed; Succeeded by 1035-SW5SA	13.01.2015	DH
036-SW5SA	Passed	11.05.2015	DH
037-SW5DH	Not tested	14.01.2015	DH
038-SW5SA	Passed	11.05.2015	DH
039-SW4DH	Passed	06.05.2015	DH
040-SW4DH	Passed	06.05.2015	DH
041-SW5DH	Not tested	14.01.2015	DH
042-SW3DH	Passed	06.05.2015	DH
043-SW5SA	Changed; Succeeded by 1043-SW5SA	15.01.2015	DH
044-SW5SA	Passed	08.05.2015	DH
045-SW4SA	Passed	08.05.2015	DH
046-SW3SA	Passed	08.05.2015	DH
047-HW2HD	Passed	06.05.2015	JS
048-HW2JS	Not tested	13.01.2015	JS
049-HW2HD	Changed; Succeeded by 114-HW2HD	03.03.2015	HD
050-HW3JS	Passed	06.05.2015	JS
051-HW3HD	Passed	06.05.2015	JS



052-HW3EG	Passed	06.05.2015	JS
053-HW2JS	Not tested	13.01.2015	JS
054-HW3HD	Not tested	13.01.2015	JS
055-HW3EG	Changed; Succeeded by 115-HW3EG	03.03.2015	HD
056-HW4HD	Passed	06.05.2015	JS
057-HW3JS	Not tested	13.01.2015	JS
058-HW3JS	Passed	06.05.2015	JS
059-HW3HD	Passed	06.05.2015	JS
060-HW3EG	Passed	06.05.2015	JS
061-HW3HD	Not tested	13.01.2015	JS
062-HW4EG	Passed	06.05.2015	JS
063-HW3HD	Not tested	13.01.2015	JS
064-HW3EG	Passed	06.05.2015	JS
065-HW3JS	Passed	06.05.2015	JS
066-HW2EG	Passed	06.05.2015	JS
067-HW3OA	Passed	06.05.2015	JS
068-HW3HD	Not tested	14.01.2015	JS
069-HW3JS	Passed	06.05.2015	JS
070-HW3HD	Passed	06.05.2015	JS
071-HW3OA	Passed	06.05.2015	JS
072-HW3JS	Passed	06.05.2015	JS
073-HW3EG	Passed	06.05.2015	JS
074-HW3JS	Passed	06.05.2015	JS
075-HW3EG	Not tested	14.01.2015	JS
076-HW3HD	Passed	06.05.2015	JS
077-HW3JS	Passed	06.05.2015	JS
078-HW5HD	Passed	06.05.2015	JS
079-HW3HD	Passed	06.05.2015	JS
080-HW4JS	Passed	06.05.2015	JS
081-HW4OA	Changed; Succeeded by 1081-HW4OA	14.01.2015	JS
082-HW3HD	Passed	06.05.2015	JS
083-HW3JS	Passed	06.05.2015	JS
084-HW4EG	Passed	06.05.2015	JS
085-HW5JS	Passed	06.05.2015	JS
086-HW5OA	Passed	06.05.2015	JS
087-HW3JS	Passed	06.05.2015	JS
088-HW3EG	Passed	06.05.2015	JS
089-HW4HD	Passed	06.05.2015	JS
090-HW3OA	Changed; Succeeded by 1090-HW3OA	13.01.2015	JS
091-HW4OA	Failed	06.05.2015	JS
092-HW4OA	Changed; Succeeded by 1092-HW4OA	13.01.2015	JS
093-HW5OA	Changed; Succeeded by 1093-HW5OA	13.01.2015	JS
094-HW4OA	Changed; Succeeded by 1094-HW4OA	13.01.2015	JS
095-HW3OA	Changed; Succeeded by 1095-HW3OA	13.01.2015	JS
096-HW3OA	Changed; Succeeded by 1096-HW3OA	13.01.2015	JS
097-HW4OA	Changed; Succeeded by 1097-HW4OA	13.01.2015	JS
098-HW4OA	Changed; Succeeded by 1098-HW4OA	13.01.2015	JS
099-HW5OA	Changed; Succeeded by 1099-HW5OA	13.01.2015	JS
100-HW5OA	Changed; Succeeded by 1100-HW5OA	13.01.2015	JS
101-HW3OA	Changed; Succeeded by 1101-HW3OA	13.01.2015	JS
102-HW3OA	Changed; Succeeded by 1102-HW3OA	13.01.2015	JS
103-HW3OA	Changed; Succeeded by 1103-HW3OA	13.01.2015	JS
104-HW4OA	Changed; Succeeded by 1104-HW4OA	14.01.2015	JS
105-HW5OA	Changed; Succeeded by 1105-HW5OA	14.01.2015	JS
106-HW2OA	Changed; Succeeded by 1106-HW2OA	14.01.2015	JS
107-HW2OA	Changed; Succeeded by 1107-HW2OA	15.01.2015	JS



108-HW4OA	Failed	06.05.2015	JS
109-HW4OA	Changed; Succeeded by 1109-HW4OA	16.01.2015	JS
110-HW4OA	Changed; Succeeded by 1110-HW4OA	16.01.2015	JS
111-SY3HD	Passed	06.05.2015	HD
112-SY1EG	Passed	06.05.2015	HD
113-SW3DH	Passed	06.05.2015	HD
114-HW2HD	Passed	06.05.2015	HD
115-SY3EG	Not tested	03.03.2015	HD
116-SW2DH	Passed	06.05.2015	DH

Table 2 – Detailed requirements test results

Requirement ID	Test status	Date	Responsible
1029-SW4DH	Passed	08.05.2015	DH
1030-SW4SA	Not tested	13.01.2015	DH
1034-SW5SA	Passed	13.01.2015	DH
1035-SW5SA	Passed	13.01.2015	DH
1043-SW5SA	Passed	08.05.2015	DH
1081-HW4OA	Failed	08.05.2015	JS
1090-HW3OA	Passed	08.05.2015	JS
1092-HW4OA	Not tested	13.01.2015	JS
1093-HW5OA	Passed	08.05.2015	JS
1094-HW4OA	Passed	08.05.2015	JS
1095-HW3OA	Passed	08.05.2015	JS
1096-HW3OA	Passed	08.05.2015	JS
1097-HW4OA	Passed	08.05.2015	JS
1098-HW4OA	Passed	08.05.2015	JS
1099-HW5OA	Not tested	13.01.2015	JS
1100-HW5OA	Not tested	13.01.2015	JS
1101-HW3OA	Not tested	13.01.2015	JS
1102-HW3OA	Not tested	13.01.2015	JS
1103-HW3OA	Passed	08.05.2015	JS
1104-HW4OA	Not tested	14.01.2015	JS
1105-HW5OA	Not tested	14.01.2015	JS
1106-HW2OA	Not tested	14.01.2015	JS
1107-HW2OA	Not tested	15.01.2015	JS
1109-HW4OA	Passed	08.05.2015	JS
1110-HW4OA	Passed	08.05.2015	JS





## Appendix II – Time estimation versus actual work

Table 3 - Time estimation versus actual work

WBS	Task Name	Estimated Work	Actual work
1.	<b>KPEC Bachelor Project</b>	<b>4 022,5 hrs</b>	<b>4031 hrs</b>
1.1.	<b>Administrative tasks</b>	<b>674,5 hrs</b>	<b>620 hrs</b>
1.1.1.	General research	90 hrs	85,5 hrs
1.1.2.	Documentation templates	52,5 hrs	56 hrs
1.1.3.	Communication / Documentation	65 hrs	64 hrs
1.1.4.	Communication / follow-up	65 hrs	72 hrs
1.1.5.	Meeting intern	72 hrs	22 hrs
1.1.6.	Meeting extern	30 hrs	44 hrs
1.1.7.	Webpage update	55 hrs	48 hrs
1.1.8.	Internal control	25 hrs	21 hrs
1.1.9.	Presentation work	80 hrs	77 hrs
1.1.10.	PAYMO / Microsoft Project	60 hrs	56 hrs
1.1.11.	Revision of project plan	30 hrs	30 hrs
1.1.12.	Practical tasks	50 hrs	30 hrs
1.2.	<b>Project planning phase</b>	<b>374 hrs</b>	<b>303 hrs</b>
1.2.1.	<b>Vision document</b>	<b>72 hrs</b>	<b>84 hrs</b>
1.2.1.1	Organization	12 hrs	12 hrs
1.2.1.2	Assignment	12 hrs	12 hrs
1.2.1.3	Objectives	12 hrs	12 hrs
1.2.1.4	Project model	12 hrs	12 hrs
1.2.1.5	Time management	12 hrs	24 hrs
1.2.1.6	Completion	12 hrs	12 hrs
1.2.2.	<b>Project plan document</b>	<b>252 hrs</b>	<b>211,86 hrs</b>
1.2.2.1	Project scope	12 hrs	36 hrs
1.2.2.2	Organization	12 hrs	36 hrs
1.2.2.3	Stakeholders	24 hrs	15 hrs
1.2.2.4	Project model	24 hrs	28,5 hrs
1.2.2.5	Time management	36 hrs	12 hrs
1.2.2.6	Financial management	24 hrs	24 hrs
1.2.2.7	Risk management	36 hrs	24 hrs
1.2.2.8	Quality management	36 hrs	24 hrs
1.2.2.9	Completion	48 hrs	24 hrs
1.2.3.	Revision of vision document	50 hrs	7 hrs
1.3.	<b>Concept of operations phase</b>	<b>174 hrs</b>	<b>186,6 hrs</b>
1.3.1.	<b>Concept of operations document</b>	<b>126 hrs</b>	<b>150 hrs</b>
1.3.1.1	FPGA-technology	24 hrs	30 hrs
1.3.1.2	Internal Combustion engine	24 hrs	30 hrs
1.3.1.3	Engine management system	36 hrs	36 hrs
1.3.1.4	Zynq micromodule (LL)	24 hrs	30 hrs
1.3.1.5	PAD-programming (HL)	18 hrs	24 hrs
1.3.2.	<b>System engineering - exploration</b>	<b>36 hrs</b>	<b>9,6 hrs</b>
1.3.2.1	Defining market need	4,8 hrs	0 hrs
1.3.2.2	Customer key driver	4,8 hrs	0 hrs
1.3.2.3	Concept of operations	4,8 hrs	4 hrs
1.3.2.4	Stakeholders and their requirements	4,8 hrs	3,18 hrs
1.3.2.5	Lifecycle needs and concerns	4,8 hrs	2,37 hrs
1.3.2.6	Completion	12 hrs	0 hrs





1.3.3.	Revision of concept of operations document	12 hrs	27 hrs
1.4.	<b>Requirements and architecture phase</b>	<b>360 hrs</b>	<b>324 hrs</b>
1.4.1.	<b>Requirements specification document</b>	<b>150 hrs</b>	<b>149 hrs</b>
1.4.1.1	Define hardware requirements	50 hrs	45,15 hrs
1.4.1.2	Define software requirements	50 hrs	56 hrs
1.4.1.3	Define system Requirements	50 hrs	44 hrs
1.4.2.	<b>Test specification document</b>	<b>125 hrs</b>	<b>121,5 hrs</b>
1.4.2.1	Define hardware test specification	50 hrs	52,6 hrs
1.4.2.2	Define software test specification	50 hrs	48,26 hrs
1.4.2.3	Define system test specification	25 hrs	20,64 hrs
1.4.3.	<b>Systems engineering - definition</b>	<b>40 hrs</b>	<b>28 hrs</b>
1.4.3.2	Use case diagram	8 hrs	9 hrs
1.4.3.3	System context diagram	8 hrs	5 hrs
1.4.3.4	Technical budgets	8 hrs	0 hrs
1.4.3.5	Functional diagrams	8 hrs	9 hrs
1.4.3.6	System Breakdown Structure	8 hrs	5 hrs
1.4.4.	Revision of documents	45 hrs	19,5 hrs
1.5.	<b>Detailed design HW &amp; SW phase</b>	<b>2 260 hrs</b>	<b>2387 hrs</b>
1.5.1.	Component research & selection	100 hrs	35,75 hrs
1.5.2.	<b>Iteration 1</b>	<b>900 hrs</b>	<b>1067,2 hrs</b>
1.5.2.1	<b>1.1 Hardware</b>	<b>600 hrs</b>	<b>648,25 hrs</b>
1.5.2.1.1	Week 6: Input circuits	120 hrs	134 hrs
1.5.2.1.2	Week 7: ADC	120 hrs	120,5 hrs
1.5.2.1.3	Week 8: Ouput circuits	120 hrs	141,5 hrs
1.5.2.1.4	Week 9: Filter design	120 hrs	124,1 hrs
1.5.2.1.5	Week 10: Circuit simulations	120 hrs	128,15 hrs
1.5.2.2	<b>2.1 Software</b>	<b>300 hrs</b>	<b>383 hrs</b>
1.5.2.2.1	Week 6: Petalinux install/config	60 hrs	83,55 hrs
1.5.2.2.2	Week 7: Communication Zynq/tablet	60 hrs	85,1 hrs
1.5.2.2.3	Week 8: Database structure	60 hrs	74,55 hrs
1.5.2.2.4	Week 9: Real time communication	60 hrs	73,75 hrs
1.5.2.2.5	Week 10: Database design	60 hrs	66 hrs
1.5.3.	<b>Iteration 2</b>	<b>360 hrs</b>	<b>359 hrs</b>
1.5.3.1	<b>1.2 Hardware</b>	<b>240 hrs</b>	<b>233,5 hrs</b>
1.5.3.1.1	Week 11: Completion of presentation and documentation	120 hrs	127,5 hrs
1.5.3.1.2	Week 12: Intern communication & inputs	120 hrs	106 hrs
1.5.3.2	<b>2.2 Software</b>	<b>120 hrs</b>	<b>125,5 hrs</b>
1.5.3.2.1	Week 11: Completion of presentation and documentation	60 hrs	71,75 hrs
1.5.3.2.2	Week 12: Tablet application structure	60 hrs	53,75 hrs
1.5.4.	<b>Iteration 3</b>	<b>540 hrs</b>	<b>594 hrs</b>
1.5.4.1	<b>1.3 Hardware</b>	<b>360 hrs</b>	<b>386 hrs</b>
1.5.4.1.1	Week 13/14: Power design 1	120 hrs	151,4 hrs
1.5.4.1.2	Week 16: Power design 2	120 hrs	123,2 hrs
1.5.4.1.3	Week 17: Power design & protection circuitry	120 hrs	111,4 hrs



1.5.4.2	<b>2.3 Software</b>	<b>180 hrs</b>	<b>207,8 hrs</b>
<b>1.5.4.2.1</b>	Week 13/14: Tablet application design 1	60 hrs	82 hrs
<b>1.5.4.2.2</b>	Week 16: Tablet application design 2	60 hrs	66,2 hrs
<b>1.5.4.2.3</b>	Week 17: Tablet application design 3	60 hrs	59,6 hrs
1.5.5.	<b>Iteration 4</b>	<b>360 hrs</b>	<b>367 hrs</b>
<b>1.5.5.1</b>	<b>1.4 Hardware</b>	<b>240 hrs</b>	<b>239,3 hrs</b>
<b>1.5.5.1.1</b>	Week 18: Revision of schematic design	120 hrs	123 hrs
<b>1.5.5.1.2</b>	Week 19: Completion of schematics & design documentation	120 hrs	116,3 hrs
1.5.5.2	<b>2.4 Software</b>	<b>120 hrs</b>	<b>127,6 hrs</b>
<b>1.5.5.2.1</b>	Week 18: Revision of software design	60 hrs	67,1 hrs
<b>1.5.5.2.2</b>	Week 19: Completion of software & design documentation	60 hrs	60,5 hrs
1.6.	<b>Implementation, test and verification phase</b>	<b>180 hrs</b>	<b>182 hrs</b>
<b>1.6.1.</b>	Week 20: Test report and after-analysis document	90 hrs	91,1 hrs
<b>1.6.2.</b>	Week 20: Project completion	90 hrs	90,9 hrs



## Appendix III – Bill of materials

Table 4 - BOM created in OrCAD Allegro to view component parameters

QTY	MPN	Status	RoHS Status	Package	DNM	Min temp	Max temp
1	ADS8568SPM	ACTIVE	COMPLIANT	LQFP64		-40DEGC	+125DEGC
1	S8421-45R	ACTIVE	COMPLIANT			-40DEGC	+105DEGC
15	SM24-02HTG	ACTIVE	COMPLIANT	SOT23-3		-40DEGC	+125DEGC
10	DRTR5V0U4TS-7	ACTIVE	COMPLIANT	SOT23-6		-65DEGC	+150DEGC
8	STPS30SM100SG-TR	ACTIVE	COMPLIANT	D2PAK		-65DEGC	+150DEGC
13	TLMG1100-GS08	ACTIVE	COMPLIANT	0603		-40DEGC	+100DEGC
4	B560C-13-F	ACTIVE	COMPLIANT	SMC		-55DEGC	+125DEGC
4	BZX84-B5V1,215	ACTIVE	COMPLIANT	SOT23-3		-65DEGC	+150DEGC
2	BZX84-A12,215	ACTIVE	COMPLIANT	SOT23-3		-65DEGC	+150DEGC
2	1N4148W-7-F	ACTIVE	COMPLIANT	SOD123		-65DEGC	+150DEGC
1	MMSZ4678T1G	ACTIVE	COMPLIANT	SOD-123		-55DEGC	+150DEGC
2	CNY17-3X007T	ACTIVE	COMPLIANT	SMD-6		-55DEGC	+100DEGC
2	SMBJ15A-E3/52	ACTIVE	COMPLIANT	SMB		-55DEGC	+150DEGC
6	SP4020-01FTG-C	ACTIVE	COMPLIANT	SOD323		-40DEGC	+125DEGC
7	SRDA3.3-4BTG	ACTIVE	COMPLIANT	SO8		-40DEGC	+125DEGC
2	PESD5Z12,115	ACTIVE	COMPLIANT	SOD-523		-65DEGC	+150DEGC
1	PNS40010ER,115	ACTIVE	COMPLIANT	SOD123W		-55DEGC	+175DEGC
1	BZG03C51TR3	ACTIVE	COMPLIANT	DO-214AC		-65DEGC	+150DEGC
1	BZX84-C30,215	ACTIVE	COMPLIANT	SOT23-3		-65DEGC	+150DEGC
2	TLMY1100-GS08	ACTIVE	COMPLIANT	0603		-40DEGC	+100DEGC
20	C0603C684K4PACTU	ACTIVE	COMPLIANT	0603		-55DEGC	+125DEGC
240	C0603C104J3RACTU	ACTIVE	COMPLIANT	0603		-55DEGC	+125DEGC
16	C0603C154K5RACTU	ACTIVE	COMPLIANT	0603		-55DEGC	+125DEGC
24	T491D476M020AT	ACTIVE	COMPLIANT	7343-31		-55DEGC	+125DEGC
31	C0603C103J5RACTU	ACTIVE	COMPLIANT	0603		-55DEGC	+125DEGC
16	C1206C106M3RACTU	ACTIVE	COMPLIANT	1206		-55DEGC	+125DEGC
15	C0603C105K8PACTU	ACTIVE	COMPLIANT	0603		-55DEGC	+85DEGC
16	C0603C332K5RAC	ACTIVE	COMPLIANT	0603		-55DEGC	+125DEGC
8	C0603C153J5RACTU	ACTIVE	COMPLIANT	0603		-55DEGC	+125DEGC
25	C0603C102F5GACTU	ACTIVE	COMPLIANT	0603		-55DEGC	+125DEGC
7	C3216X5R1V226M160AC	ACTIVE	COMPLIANT	1206		-55DEGC	+85DEGC
3	C1210C475K5RACTU	ACTIVE	COMPLIANT	1210		-55DEGC	+125DEGC
5	MAL214699602E3		COMPLIANT	1012		-55DEGC	+125DEGC
1	C0603C472K5RAC	ACTIVE	COMPLIANT	0603		-55DEGC	+125DEGC
3	C0603C222J5RACTU	ACTIVE	COMPLIANT	0603		-55DEGC	+150DEGC
3	C0603C102K5RACTU	ACTIVE	COMPLIANT	0603		-55DEGC	+125DEGC
1	C0603C470J5GAC	ACTIVE	COMPLIANT	0603		-55DEGC	+125DEGC
1	C0603C681J1GACTU	ACTIVE	COMPLIANT	0603		-55DEGC	+120DEGC
11	C0603C101J5GAC7411	ACTIVE	COMPLIANT	0603		-55DEGC	+125DEGC



4	C0805C475K4PACTU	ACTIVE	COMPLIANT	0805		-55DEGC	+85DEGC
1	C0603C471K1RACTU	ACTIVE	COMPLIANT	0603		-55DEGC	+125DEGC
2	C1206C107M9PACTU	ACTIVE	COMPLIANT	1206		-55DEGC	+85DEGC
2	C0603C223K1RACTU	ACTIVE	COMPLIANT	0603		-55DEGC	+125DEGC
2	C0603C220J5GACTU	ACTIVE	COMPLIANT	0603		-55DEGC	+125DEGC
2	C0603C221J5GAC	ACTIVE	COMPLIANT	0603		-55DEGC	+125DEGC
5	EEEFK1V221AP	ACTIVE	COMPLIANT	0810		-55DEGC	+105DEGC
3	C1206C104K1RAC	ACTIVE	COMPLIANT	1206		-55DEGC	+125DEGC
1	C0603C473J4RACTU	ACTIVE	COMPLIANT	0603		-55DEGC	+125DEGC
1	T521D106M050ATE090	ACTIVE	COMPLIANT	7343-31		-55DEGC	+125DEGC
2	C0603C270F5GACTU	ACTIVE	COMPLIANT	0603		-55DEGC	+125DEGC
1	1206USB-371MLC	ACTIVE	COMPLIANT			-40DEGC	+85DEGC
1	34763-0001		COMPLIANT			-40DEGC	+105DEGC
1	J0G-0001NL	ACTIVE	COMPLIANT			0DEGC	+70DEGC
1	90120-0123	ACTIVE	COMPLIANT			-55DEGC	+125DEGC
1	10103594-0001LF	ACTIVE	COMPLIANT			-55DEGC	+85DEGC
1	DO5040H-473MLD	ACTIVE	COMPLIANT			-40DEGC	+85DEGC
1	LPS3015-472MRC	ACTIVE	COMPLIANT			-40DEGC	+85DEGC
1	DR1050-3R3-R	ACTIVE	COMPLIANT			-40DEGC	+125DEGC
2	DO3308P-223MLD	ACTIVE	COMPLIANT			-40DEGC	+85DEGC
1	MSS1048-473MLD	ACTIVE	COMPLIANT			-40DEGC	+85DEGC
2	BLM18PG471SN1D	ACTIVE	COMPLIANT	0603		-55DEGC	+125DEGC
2	LSHM-150-04.0-L-DV-A-S-K-TR	ACTIVE	COMPLIANT			-55DEGC	+125DEGC
1	LSHM-130-04.0-L-DV-A-S-K-TR	ACTIVE	COMPLIANT			-55DEGC	+125DEGC
1	15-91-1102	ACTIVE	COMPLIANT			-40DEGC	+105DEGC
1	87831-1420	ACTIVE	COMPLIANT			-55DEGC	+105DEGC
8	ISL9V5045S3ST_F085	ACTIVE	COMPLIANT	D2PAK3		-40DEGC	+175DEGC
3	BSC123N08NS3GATMA1	ACTIVE	COMPLIANT	TDSON-8		-55DEGC	+150DEGC
16	NCV8401ADTRKG	ACTIVE	COMPLIANT	DPAK3		-40DEGC	+150DEGC
2	MMBTA42,215	ACTIVE	COMPLIANT	SOT23-3		-65DEGC	+150DEGC
2	IRFS4127PBF		COMPLIANT	D2PAK		-55DEGC	+175DEGC
2	MMBT3904	ACTIVE	COMPLIANT	SOT23-3		-55DEGC	+150DEGC
13	CRCW060318K0FKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
95	CRCW06030000Z0EAHP	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
12	CRCW06031K50FKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
18	CRCW060368R0FKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
22	CRCW0603100KFKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
56	CRCW06030000Z0EAHP	ACTIVE	COMPLIANT	0603	DNM	-55DEGC	+155DEGC
135	CRCW060310K0FKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
8	CRCW0603120RFKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
1	CRCW1206120RFKEA	ACTIVE	COMPLIANT	1206	DNM	-55DEGC	+155DEGC
1	CRCW1206120RFKEA	ACTIVE	COMPLIANT	1206		-55DEGC	+155DEGC





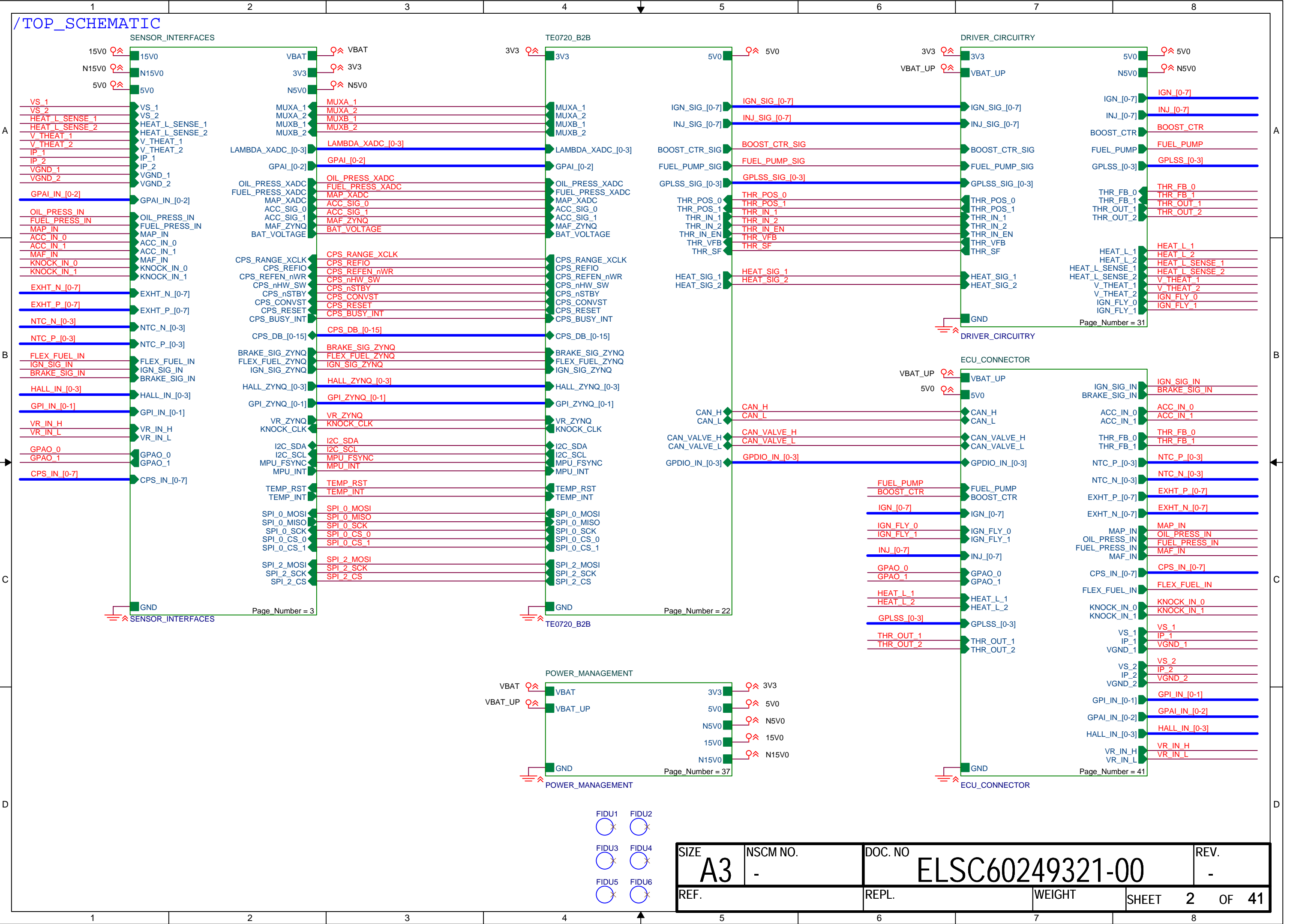
83	CRCW06031K00FKEB	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
17	CRCW0603330RFKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
3	CRCW060327R0FKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
3	CRCW060310R0FKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
9	CRCW0603820RFKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
13	CRCW0603100RFKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
16	CRCW06034K70FKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
21	CRCW0603390RFKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
10	CRCW06033K90FKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
11	CRCW06032K20FKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
22	CRCW06031K20FKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
8	CRCW0603470RFKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
1	WSL20108L000FEA		COMPLIANT	2010		-65DEGC	+170DEGC
1	CRCW0603560KFKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
2	CRCW0603180KFKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
3	CRCW060382K0FKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
1	CRCW06032R20JNEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
8	TNPW060322K0BEEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
1	CRCW0603120KFKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
2	CRCW0603300RFKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
9	CRCW0603680RFKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
6	CRCW060315K0FKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
2	CRCW060347K0FKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
1	WSL2010R0100FEA		COMPLIANT	2010		-65DEGC	+170DEGC
8	CRCW06035K60FKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
5	CRCW06031M00FKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
10	CRCW060312K0FKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
3	CRCW060327K0FKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
3	WSLP1206R0400FEA	ACTIVE	COMPLIANT	1206		-65DEGC	+170DEGC
1	CRCW06031K00FKEB	ACTIVE	COMPLIANT	0603	DNM	-55DEGC	+155DEGC
12	CRCW06033K30FKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
6	CRCW060333R0FKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
16	CRCW0603150RFKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
2	MCT06030D8201BP500	ACTIVE	COMPLIANT	0603		-55DEGC	+125DEGC
6	CRCW120656K0FKEA	ACTIVE	COMPLIANT	1206		-55DEGC	+155DEGC
1	WSL20104L000FEA		COMPLIANT	2010		-65DEGC	+170DEGC
1	CRCW06035K10FKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
2	CRCW060333K0FKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
1	RT0603BRD0710KL	ACTIVE	COMPLIANT	0603		-55DEGC	+125DEGC
1	CRCW060356R0FKEA	ACTIVE	COMPLIANT	0603		-55DEGC	+155DEGC
1	7914J-1-000	ACTIVE	COMPLIANT			-55DEGC	+125DEGC
16	LTC2050HVHS5#PBF	ACTIVE	COMPLIANT	SOT23-5		-40DEGC	+125DEGC
50	LT1812IS5#TRMPBF	ACTIVE	COMPLIANT	TSOT23-5		-40DEGC	+85DEGC



2	74LCX125MTCX	ACTIVE	COMPLIANT	TSSOP14	-40DEGC	+85DEGC
2	SN65HVD234DG4	ACTIVE	COMPLIANT	SO8	-40DEGC	+125DEGC
24	LM5112MY/NOPB	ACTIVE	COMPLIANT	MSOP8	-40DEGC	+125DEGC
1	LTC3891IFE#PBF		COMPLIANT	TSSOP20-EP	-40DEGC	+125DEGC
1	LTM4609IV#PBF	ACTIVE	COMPLIANT	LGA141	-40DEGC	+85DEGC
1	LT3580IMS8E#PBF	ACTIVE	COMPLIANT	MSOP8	-40DEGC	+125DEGC
1	LT8580EMS8E#PBF		COMPLIANT	MSOP8-EP	-40DEGC	+125DEGC
1	LT3724EFE#PBF		COMPLIANT	TSSOP16-EP	-40DEGC	+125DEGC
2	SN74LVC2T45DCTR	ACTIVE	COMPLIANT	SSOP8	-40DEGC	+85DEGC
1	MRF24WG0MB-I/RM	ACTIVE	COMPLIANT		-40DEGC	+85DEGC
1	SN74LVC1G125DCKR	ACTIVE	COMPLIANT	SC70-5	-40DEGC	+125DEGC
1	MAX9924UAUB+	ACTIVE	COMPLIANT	uMAX10	-40DEGC	+125DEGC
1	TPIC8101DW	ACTIVE	COMPLIANT	SO20	-40DEGC	+125DEGC
1	DAC124S085C1MM/NOPB	ACTIVE	COMPLIANT	VSSOP10	-40DEGC	+105DEGC
1	LP2980IM5-3.3/NOPB	ACTIVE	COMPLIANT	SOT23-5	-40DEGC	+125DEGC
2	74LV4052PW,118		COMPLIANT	TSSOP16	-40DEGC	+125DEGC
2	LT1468CS8#PBF	ACTIVE	COMPLIANT	SO8	0DEGC	+70DEGC
1	MPU-9150	ACTIVE	COMPLIANT	QFN24	-40DEGC	+85DEGC
1	MS561101BA03-50	ACTIVE	COMPLIANT	QFN8	-40DEGC	+85DEGC
1	LT4363HMS-2#PBF		COMPLIANT	MSOP12	-40DEGC	+125DEGC
1	LTC2983ILX#PBF	ACTIVE	COMPLIANT	LQFP48	-40DEGC	+85DEGC
1	MC33931VW	ACTIVE	COMPLIANT	HSOP44	-40DEGC	+125DEGC
1	FT2232HL-REEL	ACTIVE	COMPLIANT	LQFP64	-40DEGC	+85DEGC
1	AT93C46EN-SH-T	ACTIVE	COMPLIANT	SO8	-55DEGC	+125DEGC
1	ECS-120-18-5G3XDS-TR	ACTIVE	COMPLIANT		-40DEGC	+125DEGC



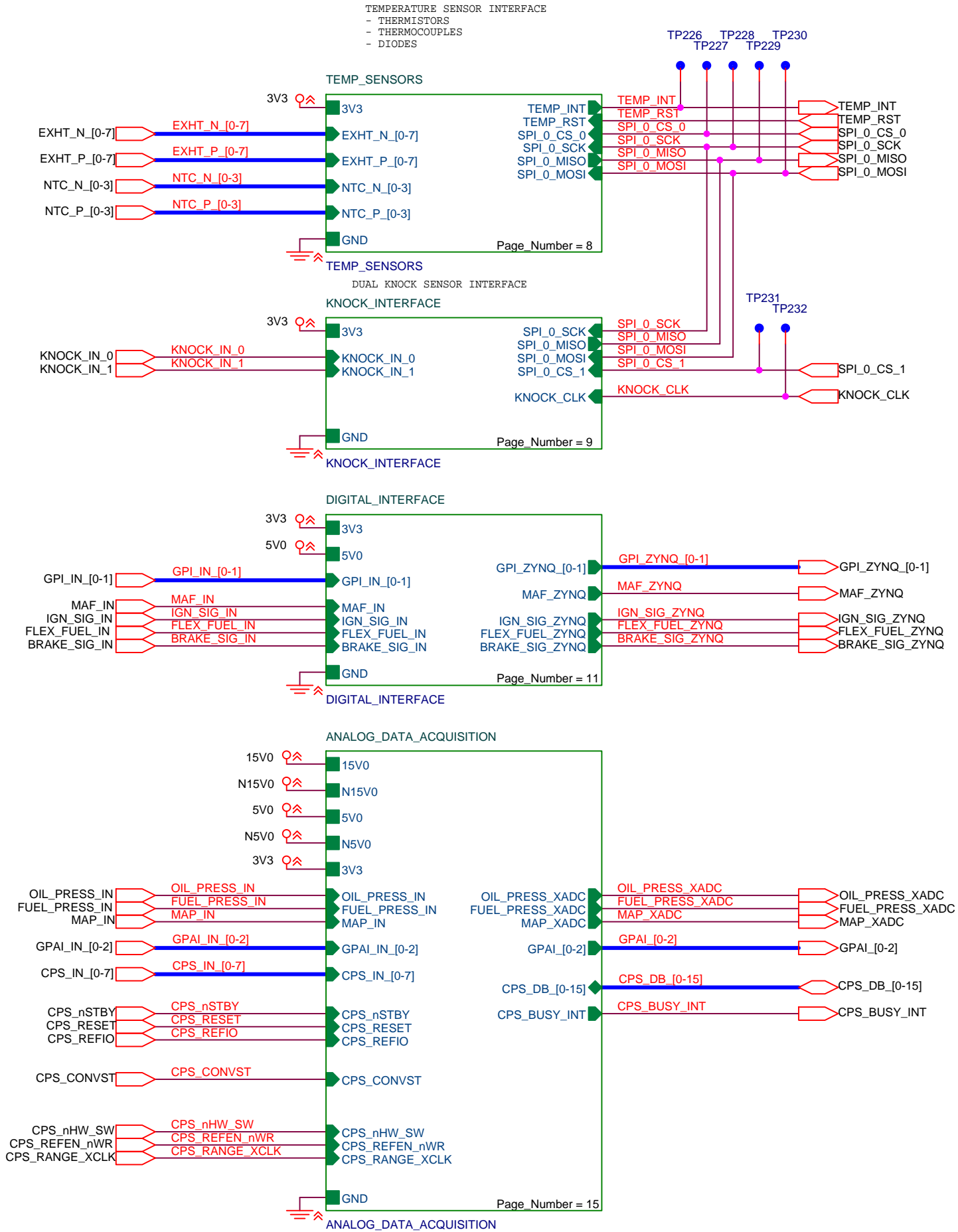
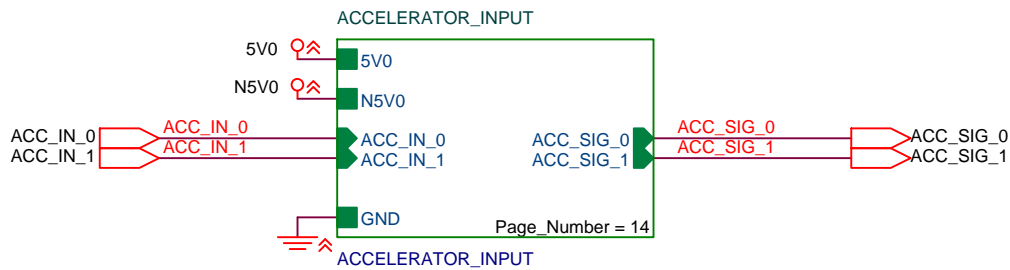
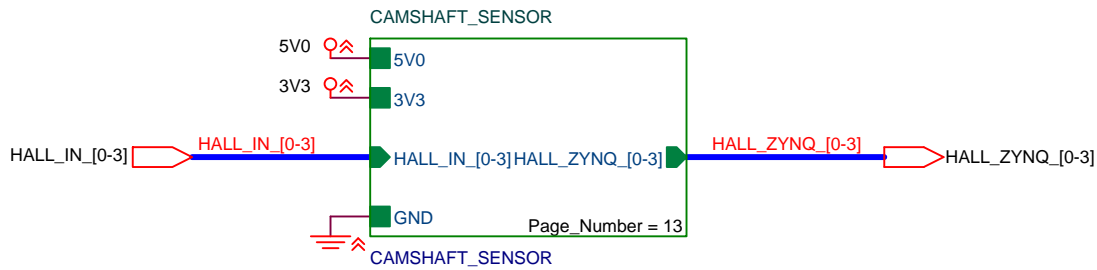
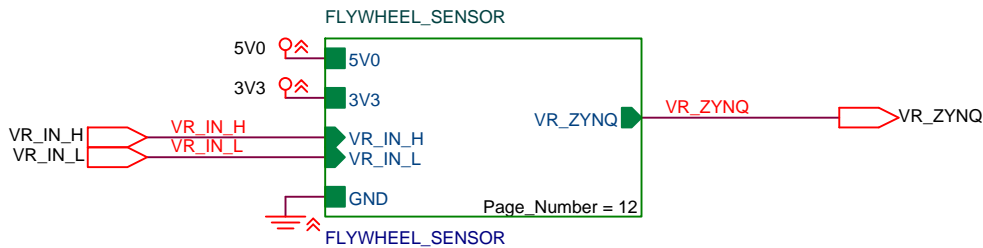
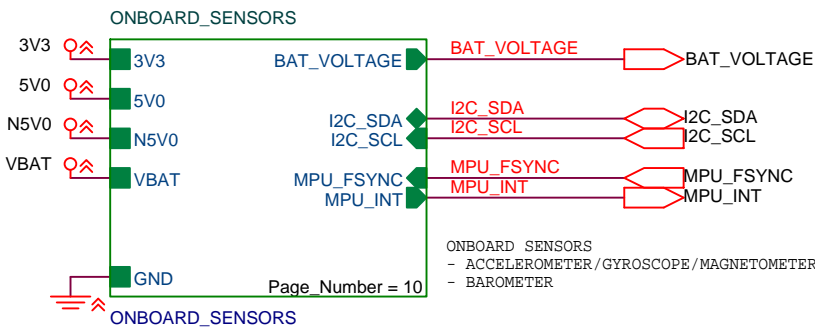
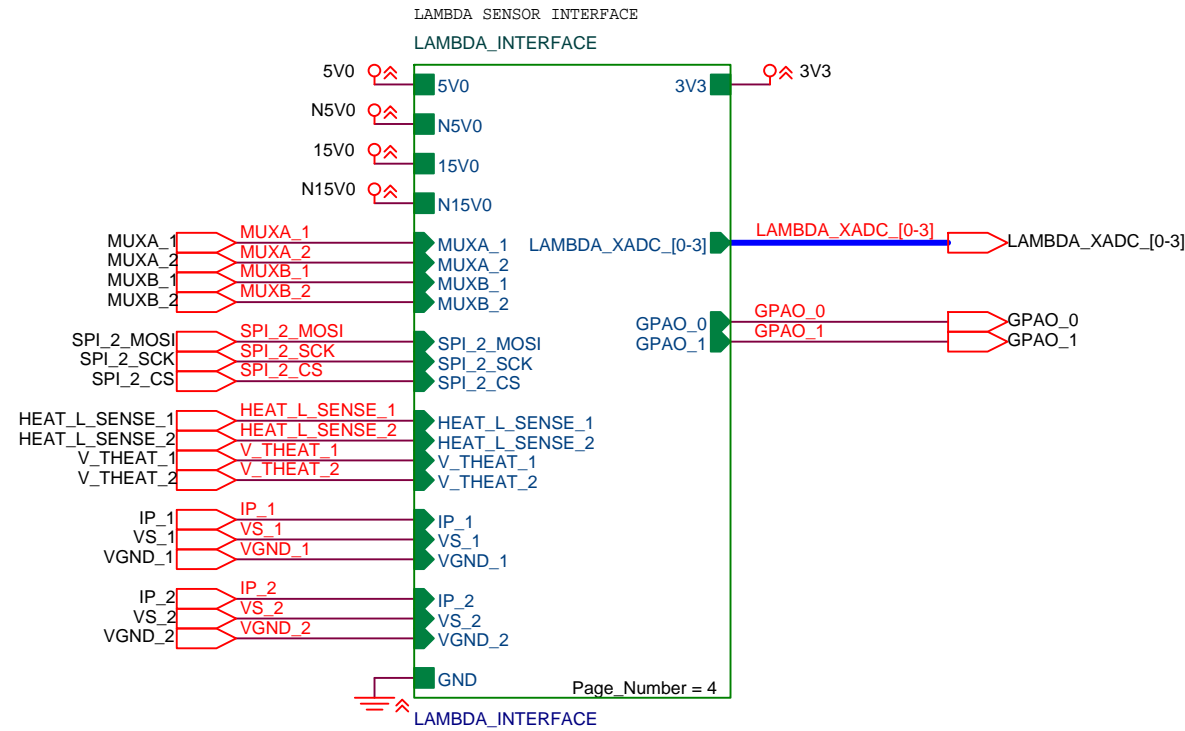
Kongsberg Programmable Engine Control (KPEC)						REVISIONS					
ECO. NO.		REV. BY		CHECKED		REV. DATE		QTY.		REV.	
#####		N.N.		N.N.		(YYYY-MM-DD)		####		####	
Page	Hierarchical block										
1.	CONTENTS										
2.	TOP_SCHEMATIC										
3.	SENSOR_INTERFACES										
4.	LAMBDA_INTERFACE										
5.	LAMBDA_FILTER										
6.	LAMBDA_SENSOR_BANK_1										
7.	LAMBDA_SENSOR_BANK_2										
8.	TEMP_SENSORS										
9.	KNOCK_INTERFACE										
10.	ONBOARD_SENSORS										
11.	DIGITAL_INPUTS										
12.	FLYWHEEL_SENSOR										
13.	CAMSHAFT_SENSOR										
14.	ACCELERATOR_INPUT										
15.	ANALOG_DATA_ACQUISITION										
16.	GPAI_INTERFACE										
17.	PRESSURE_INTERFACE										
18.	CPS_INTERFACE										
19.	CPS_FILTER_BANK1										
20.	CPS_FILTER_BANK2										
21.	CPS_ADC										
22.	TE0720_B2B										
23.	CONNECTOR_TOP										
24.	CONNECTOR_BOTTOM										
25.	CONNECTOR_LEFT										
26.	ECU_COM										
27.	ETHERNET_WIFI										
28.	USB_INTERFACE										
29.	CAN_INTERFACES										
30.	JTAG_INTERFACE										
31.	DRIVER_CIRCUITRY										
32.	COIL_DRIVER										
33.	LAMBDA_HEATER_DRIVER										
34.	INJECTION_DRIVER										
35.	THROTTLE_CONTROL										
36.	LOW_SIDE_DRIVERS										
37.	POWER_MANAGEMENT										
38.	DCDC_3V3										
39.	DCDC_5V0_15V0_N15V0										
40.	DCDC_N5V0										
41.	ECU_CONNECTOR										
<div></div>											
Design description											
This is the schematics for the KPEC ECU designed for interfacing petrol engines with up to 8 cylinders for use in motorsports applications. The KPEC ECU is a motherboard design for interfacing the Trenz Electronics TE0720 micromodule, a Xilinx Zynq SoC based processing platform for embedded applications. The KPEC ECU is designed for interfacing primarily Bosch Motorsports sensors and actuators for high performance engine control, as well as Optrand spark plug pressure sensors for measuring cylinder pressure. The KPEC ECU can interface up to 44 sensors and 24 actuators with support for further expansion modules.											
The KPEC 2015 group members are: Ola P. Aasheim Salahuddin Asjad Hung V. Dinh Even Gudbrandsen Dler Hasan Jannik Schäffer											
CAD SOFTWARE: cadence											
PROJECTION		SCALE		DEPARTMENT		<div></div>					
DATE 2015-05-18		KPEC									
DRAWN KPEC											
DESIGN KPEC											
CHECK		DRAWING TITLE EL.SCHEMATIC Kongsberg Programmable Engine Control									
APPROVED											
SIZE A3		NSCM NO. -		DOC. NO ELSC60249321-00				REV. -			
REF.		REPL.		WEIGHT		SHEET 1		OF 41			



SIZE	NSCM NO.	DOC. NO	REV.
A3	-	ELSC60249321-00	-
REF.	REPL.	WEIGHT	SHEET 2 OF 41



/SENSOR\_INTERFACES



SIZE	NSCM NO.	DOC. NO	REV.
A3	-	ELSC60249321-00	-
REF.	REPL.	WEIGHT	SHEET 3 OF 41

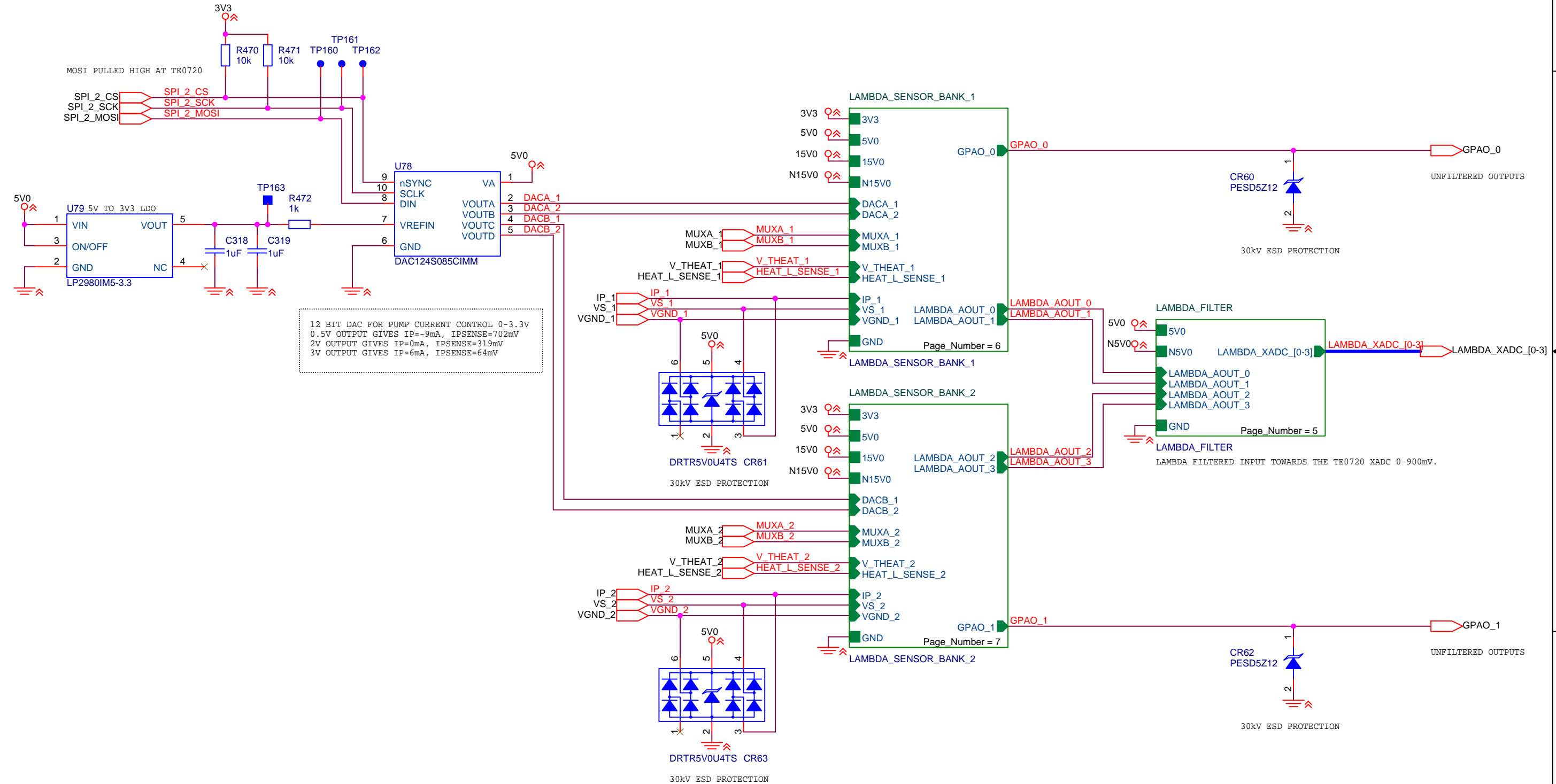
/SENSOR\_INTERFACES/LAMBDA\_INTERFACE

THE BOSCH LSU 4.9 IS HIGHLY COMPLEX TO INTERFACE DUE TO LACK OF AVAILABLE DOCUMENTATION REGARDING ITS ELECTRICAL CHARACTERISTICS. THE KEY CONCEPT IS THAT THE NERNST CELL GENERATES A VOLTAGE ACCORDING TO THE O2 PRESENT AT THE CELL. THE PUMP CELL PUMPS OXYGEN AWAY FROM THE NERNST CELL, AND BY DETERMINING THE AMOUNT OF CURRENT REQUIRED TO CHANGE THE NERNST CELL VOLTAGE A LOOKUP TABLE CAN DETERMINE THE LAMBDA VALUE. THIS DESIGN IS BASED UPON THE CJ125/CJ135 DATASHEET, AND OPEN SOURCE MATERIAL.

HEATING ELEMENT FOR THE LSU 4.9 SHOULD BE ACTIVATED STEPWISE TO AVOID DAMAGE TO THE SENSOR. HEATER CIRCUITRY IS LOCATED IN THE DRIVER\_INTERFACES. V\_THEAT/HEAT\_L\_SENSE\_1 PROVIDES VOLTAGE READINGS FROM THE DRIVER CIRCUITRY.

INPUT FROM BOSCH LSU 4.9:  
VGND\_1 (YELLOW) IS COMMON SIGNAL GROUND (2V ABOVE GND)  
VS\_1 (BLACK) IS NERNST CELL VOLTAGE SIGNAL (0.2V-0.8V ACC TO O2 LEVEL PRESENT)  
IP\_1 (RED) IS PUMP CELL CURRENT (-9mA TO 6mA)

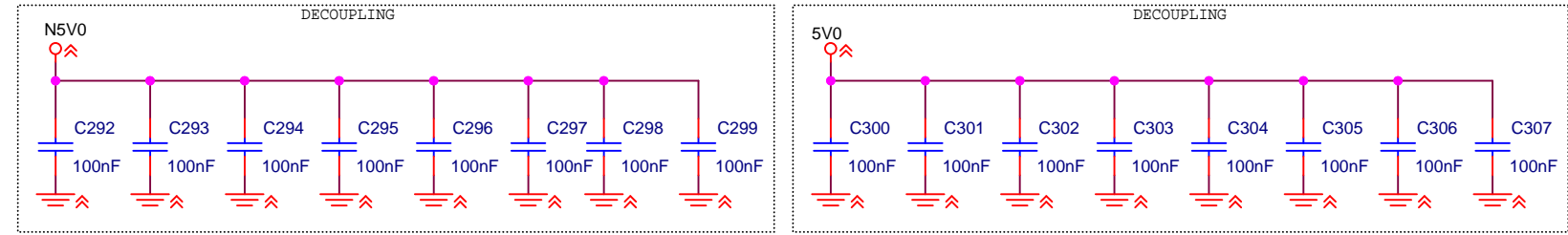
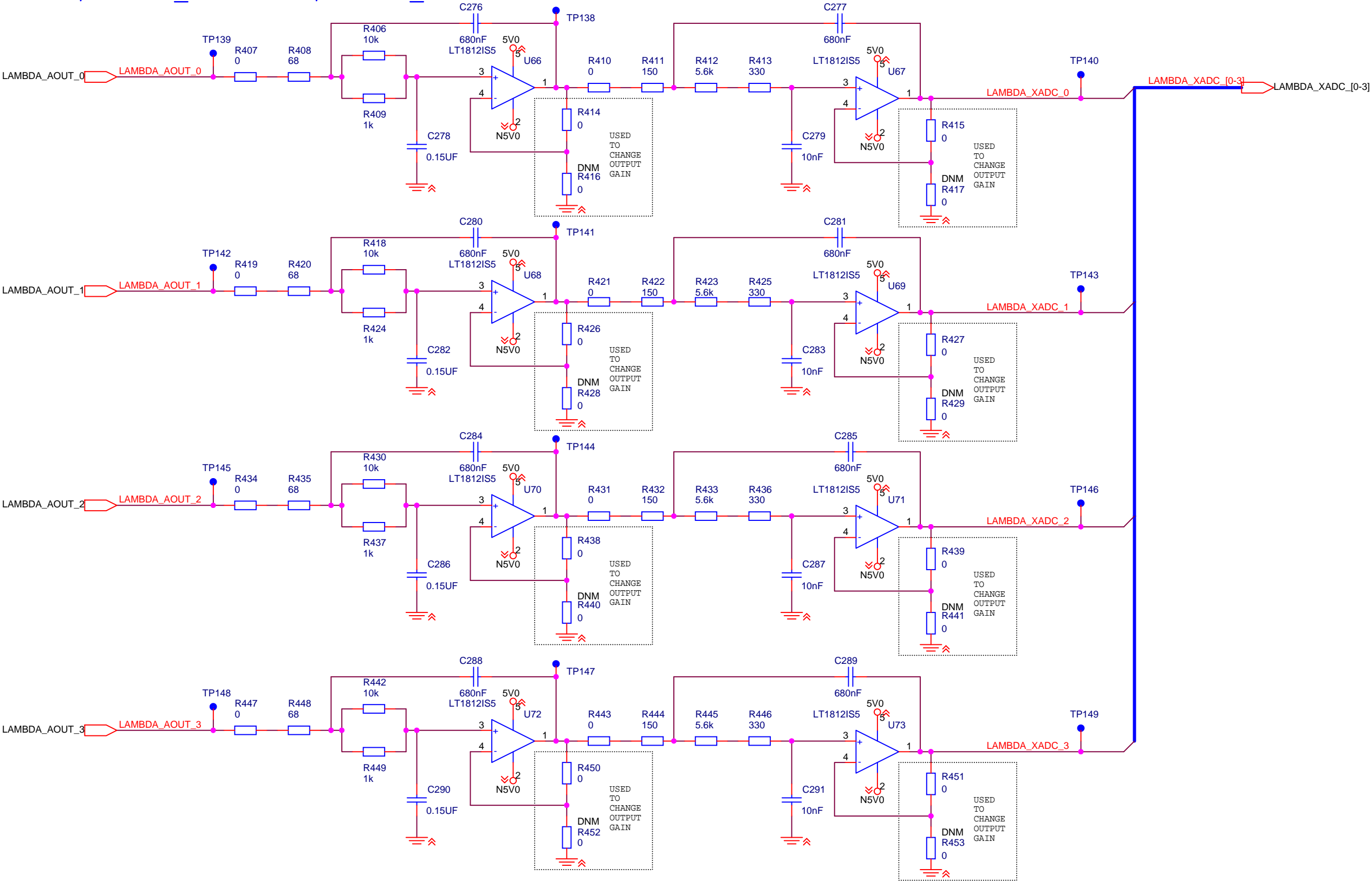
RCALH (GREEN) COULD BE CONNECTED DIRECTLY TO VGND FOR SENSOR CALIBRATION.



SIZE	NSCM NO.	DOC. NO.	REV.
A3	-	ELSC01234567-89	-
REF.	REPL.	WEIGHT	SHEET 4 OF 41

/SENSOR\_INTERFACES/LAMBDA\_INTERFACE/LAMBDA\_FILTER

CUTOFF FREQUENCY: 2kHz  
STOPBAND: 20kHz  
ATTENUATION: 81 dB  
4-TH ORDER FILTER



SIZE A3	NSCM NO. -	DOC. NO ELSC60249321-00	REV. -
REF.	REPL.	WEIGHT	SHEET 5 OF 41

/SENSOR\_INTERFACES/LAMBDA\_INTERFACE/LAMBDA\_SENSOR\_BANK\_1

A

B

C

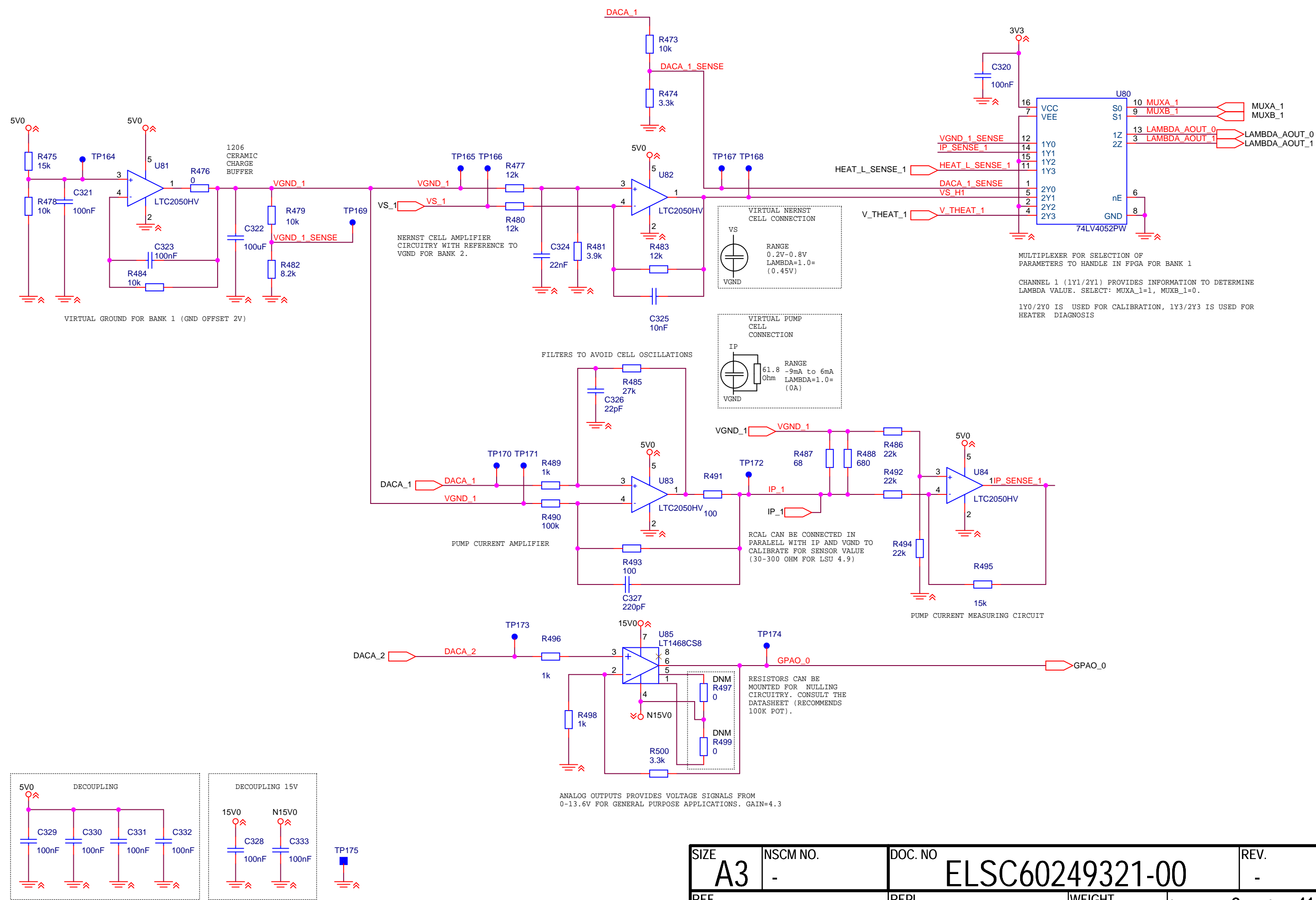
D

A

B

C

D



SIZE	NSCM NO.	DOC. NO	REV.
A3	-	ELSC60249321-00	-
REF.	REPL.	WEIGHT	SHEET 6 OF 41

/SENSOR\_INTERFACES/LAMBDA\_INTERFACE/LAMBDA\_SENSOR\_BANK\_2

A

B

C

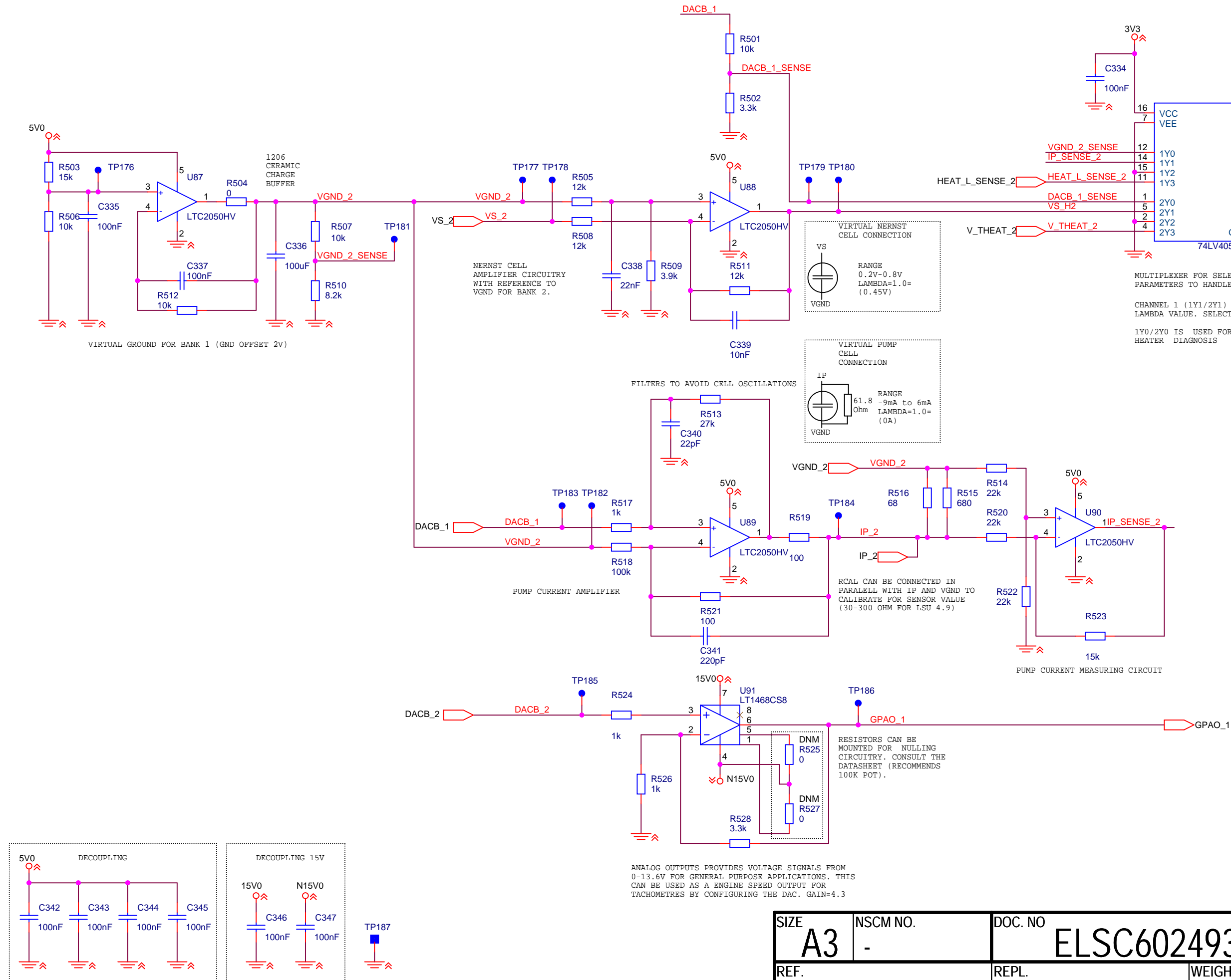
D

A

B

C

D



ANALOG OUTPUTS PROVIDES VOLTAGE SIGNALS FROM 0-13.6V FOR GENERAL PURPOSE APPLICATIONS. THIS CAN BE USED AS A ENGINE SPEED OUTPUT FOR TACHOMETRES BY CONFIGURING THE DAC. GAIN=4.3

SIZE A3	NSCM NO. -	DOC. NO ELSC60249321-00	REV. -
REF.	REPL.	WEIGHT	SHEET 7 OF 41

/SENSOR\_INTERFACES/TEMP\_SENSORS

A

B

C

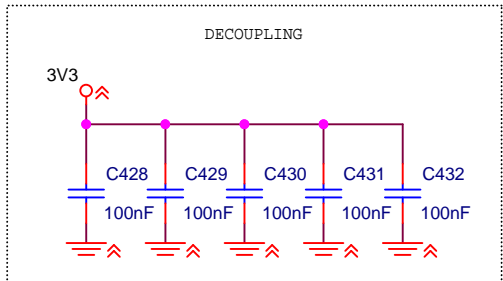
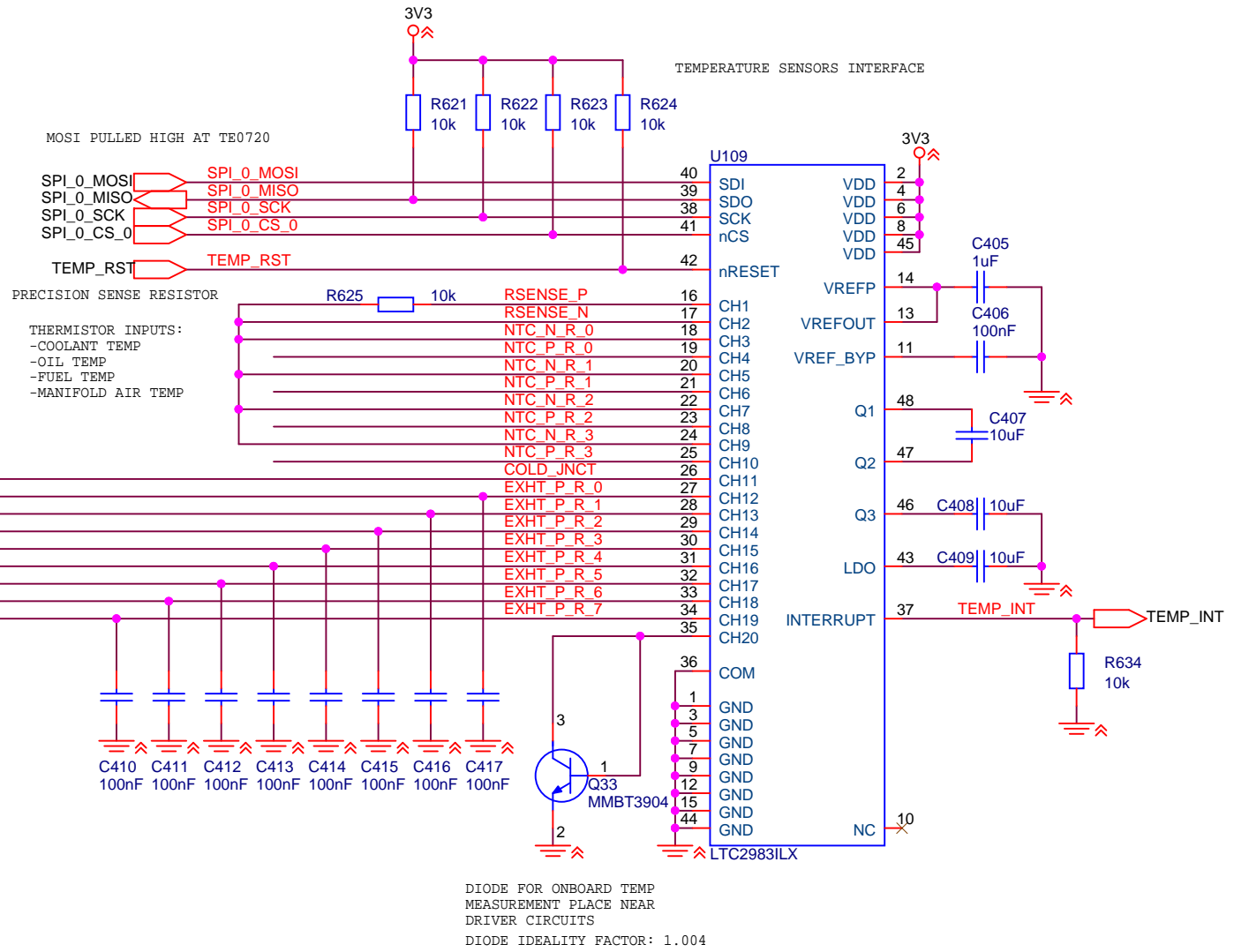
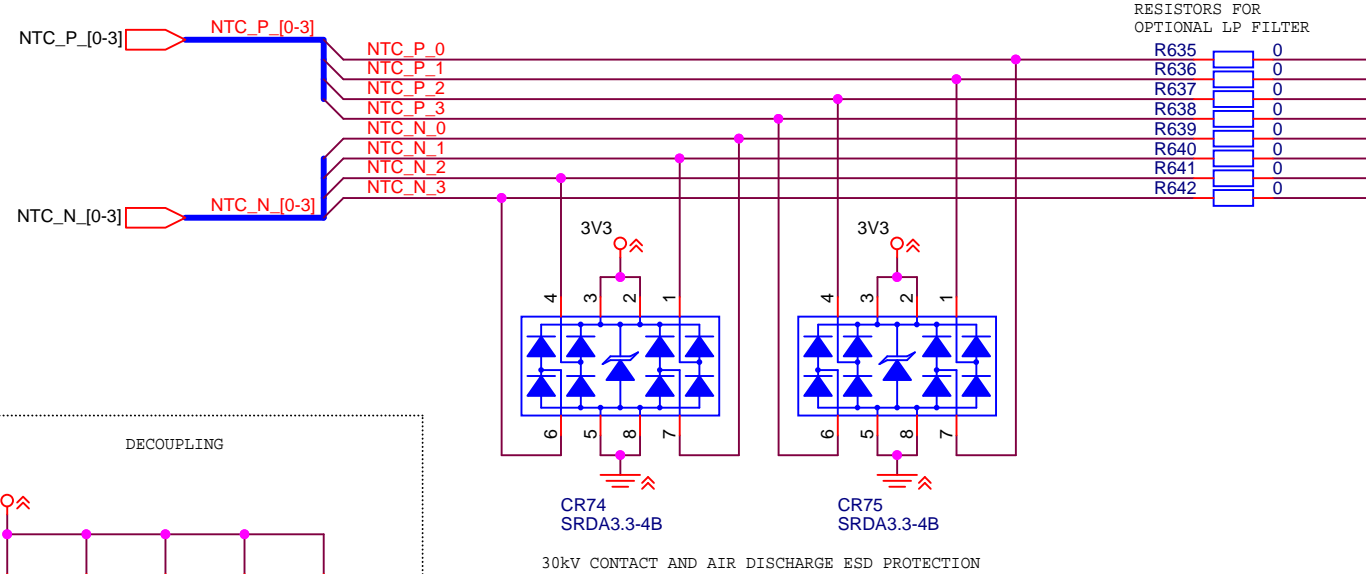
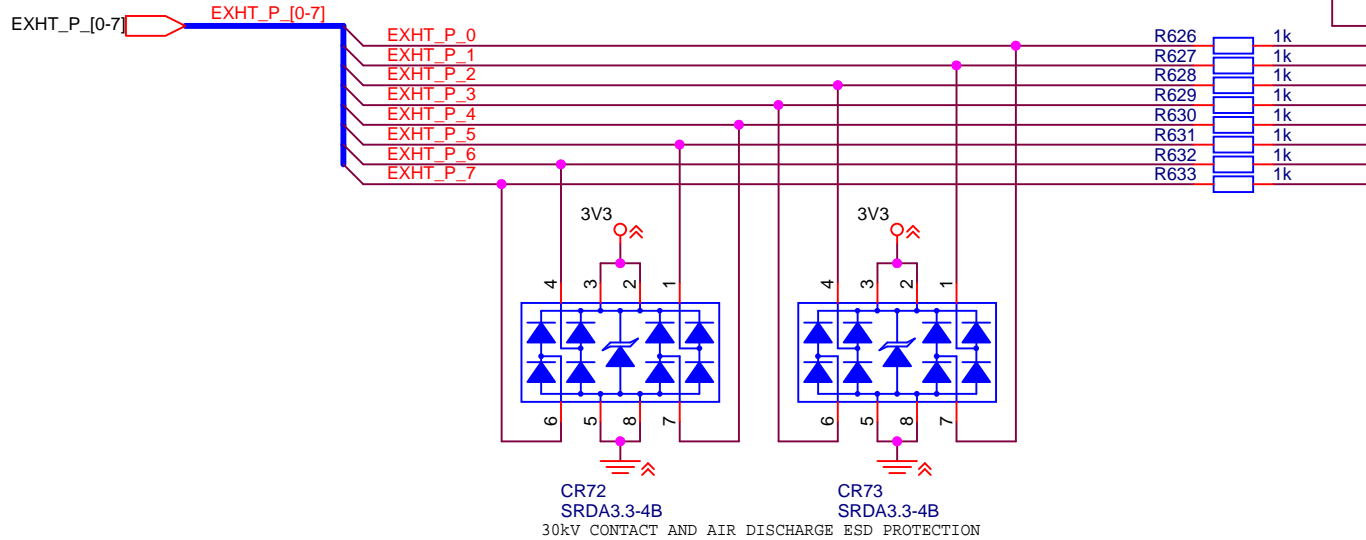
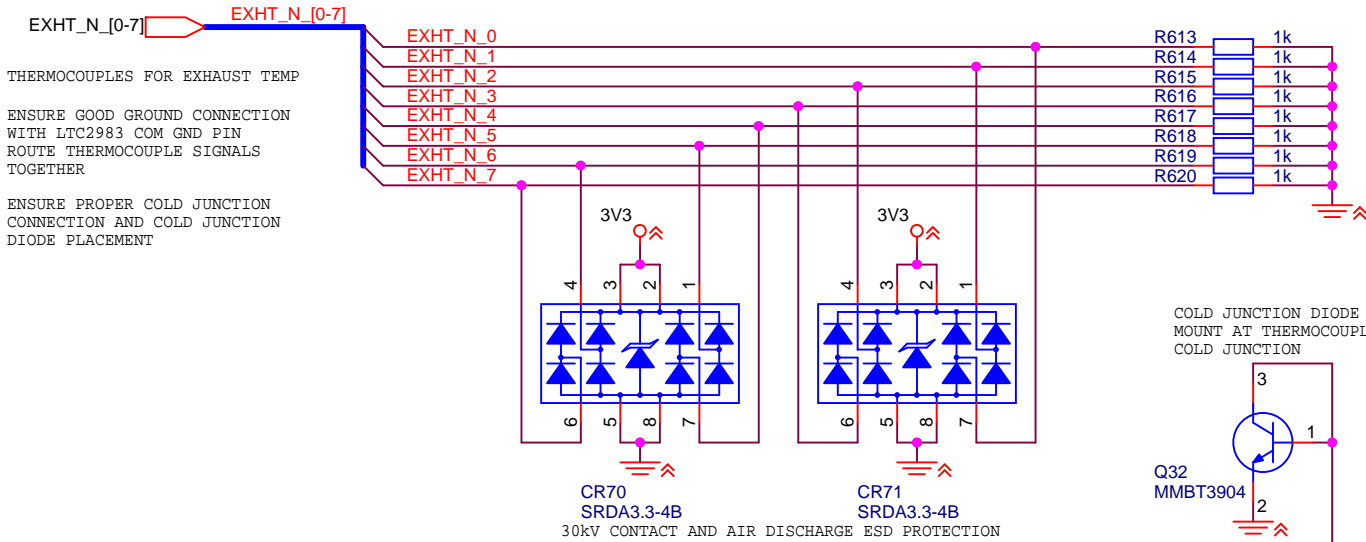
D

A

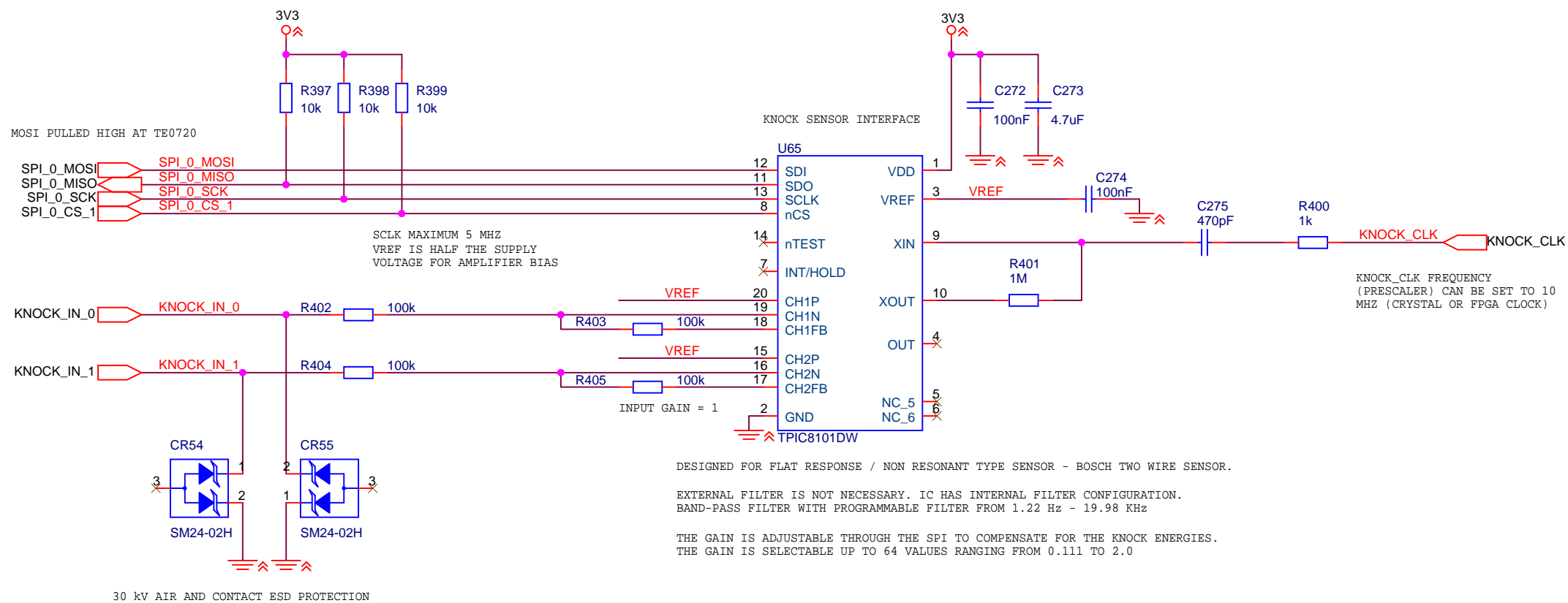
B

C

D

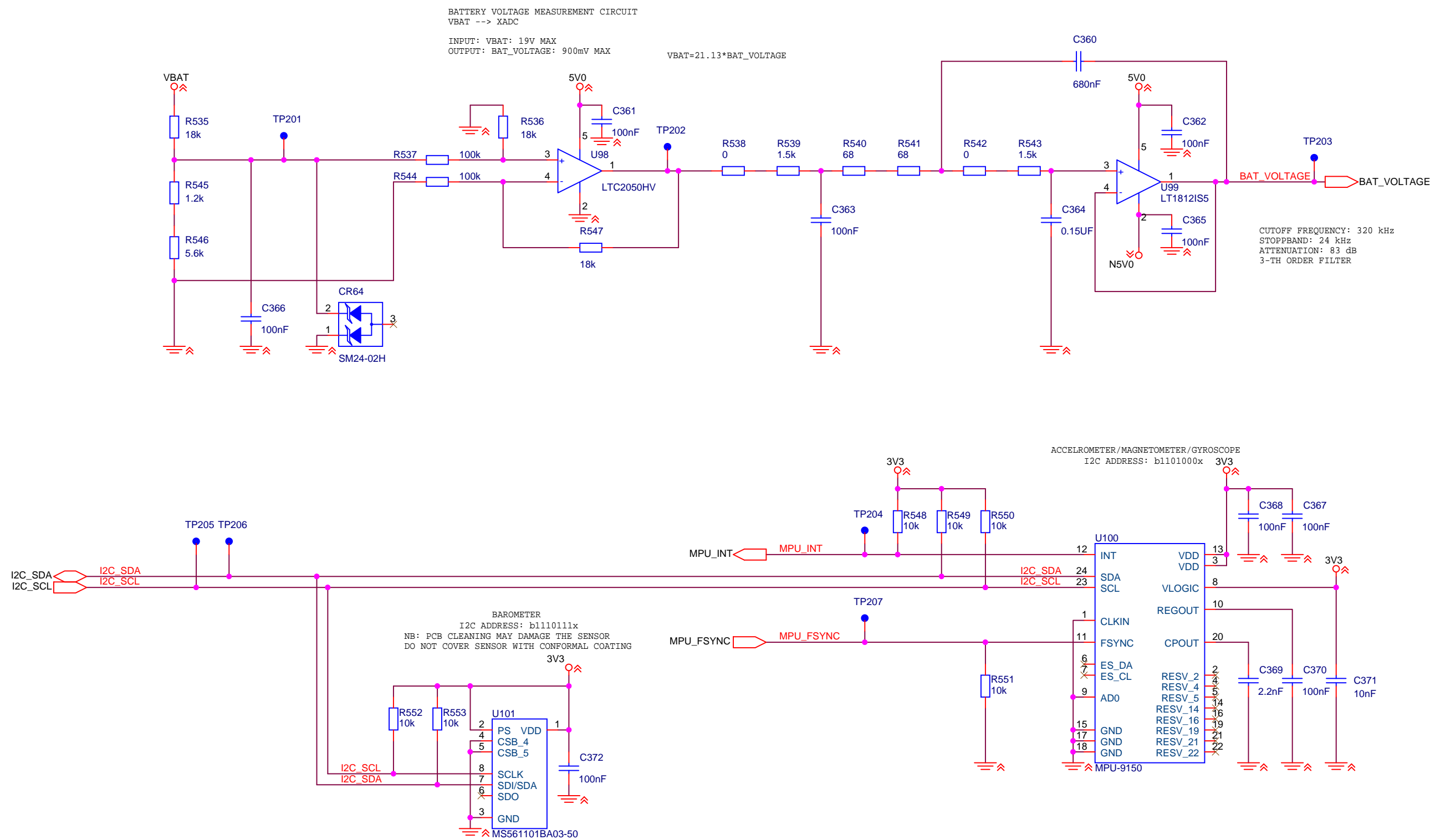


SIZE	NSCM NO.	DOC. NO	REV.
A3	-	ELSC60249321-00	-
REF.	REPL.	WEIGHT	SHEET 8 OF 41



SIZE	NSCM NO.	DOC. NO	REV.
A3	-	ELSC60249321-00	-
REF.	REPL.	WEIGHT	SHEET 9 OF 41

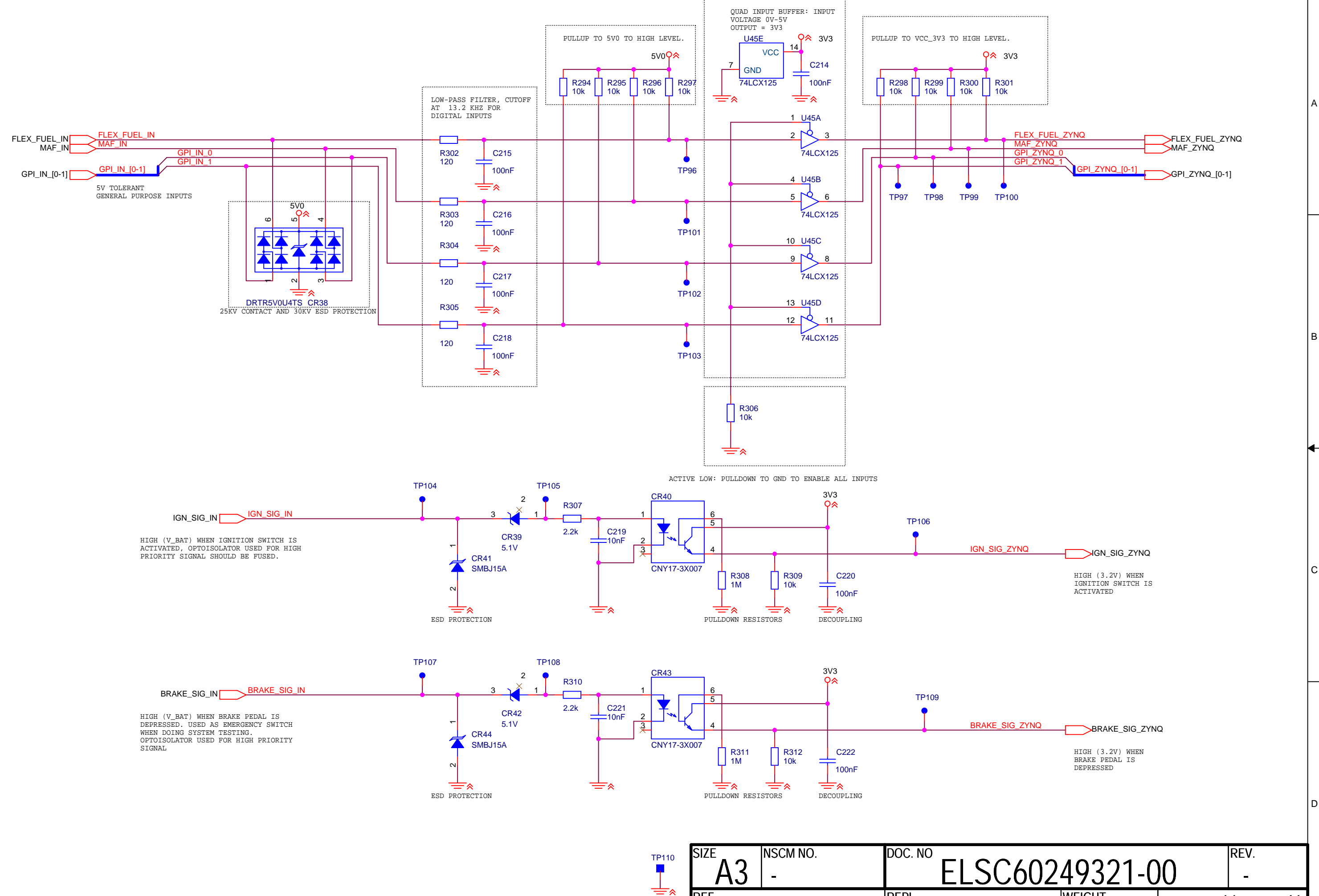




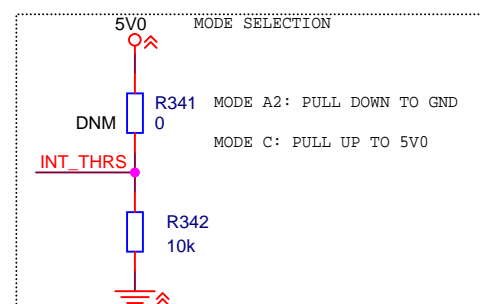
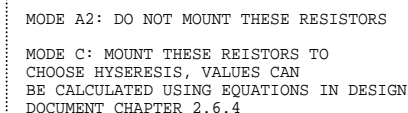
SIZE	NSCM NO.	DOC. NO.	REV.
A3	-	ELSC60249321-00	-
REF.	REPL.	WEIGHT	SHEET 10 OF 41



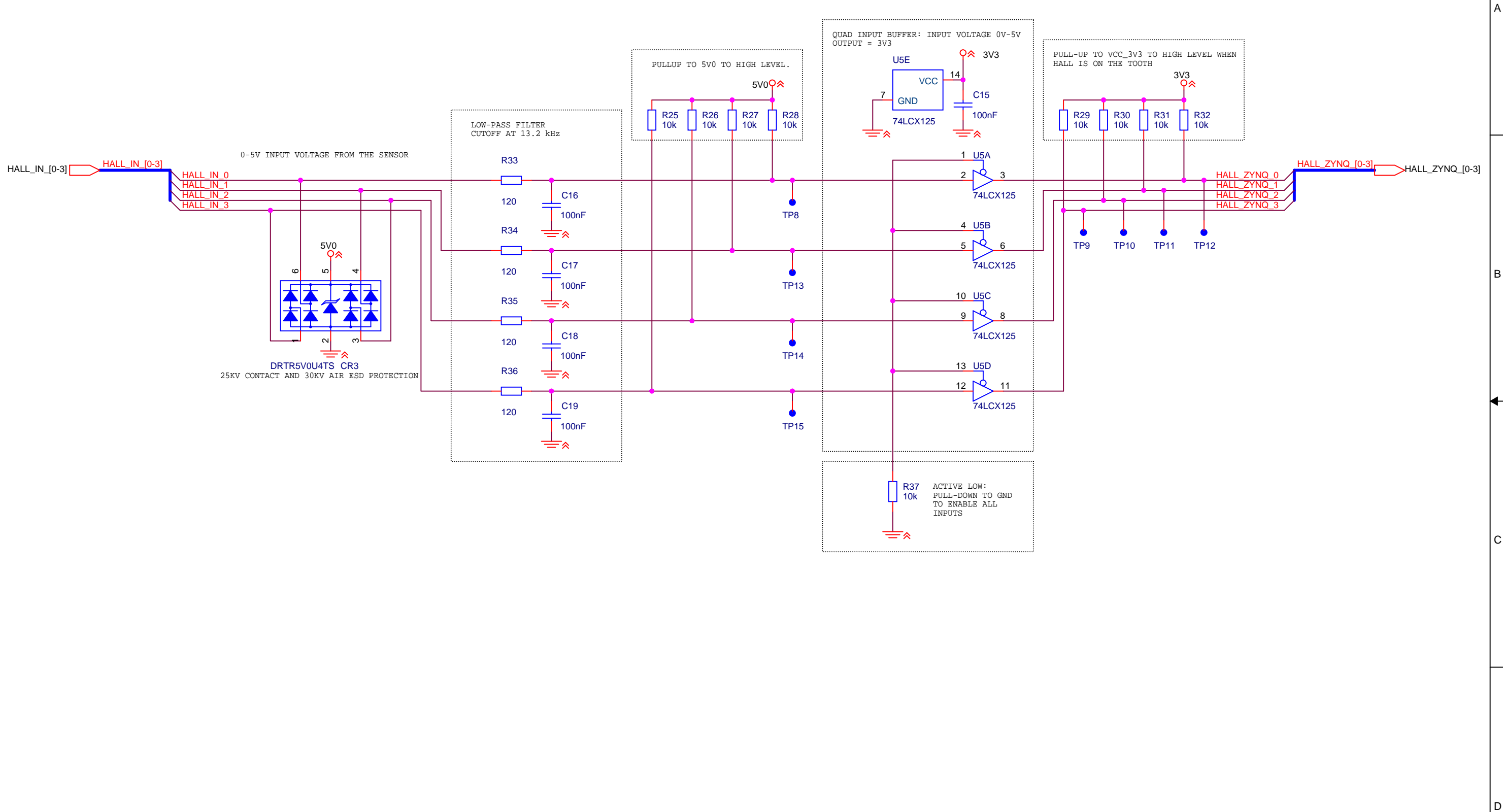
/SENSOR\_INTERFACES/DIGITAL\_INTERFACE



SIZE	NSCM NO.	DOC. NO	REV.
A3	-	ELSC60249321-00	-
REF.	REPL.	WEIGHT	SHEET 11 OF 41

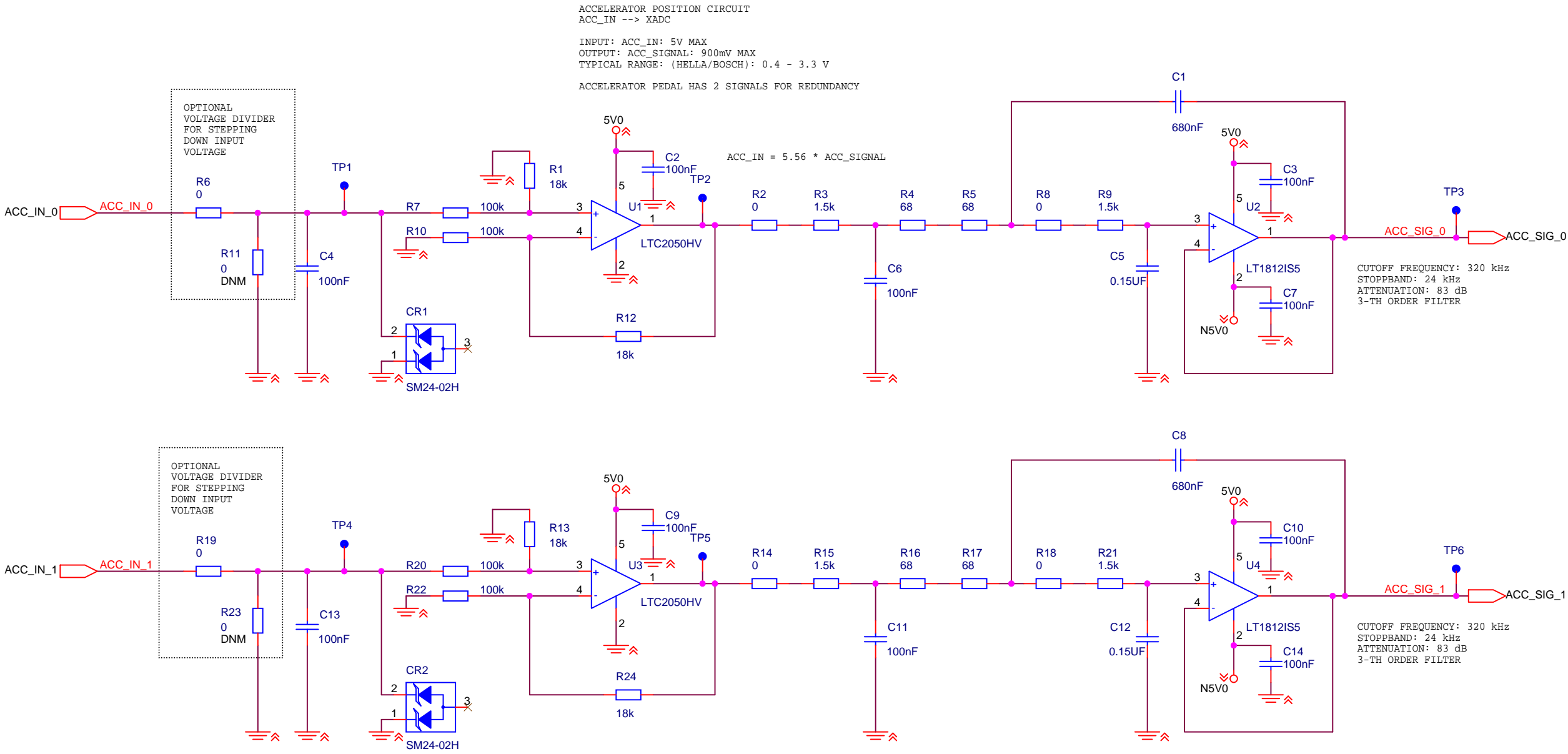


SIZE <b>A3</b>	NSCM NO. -	DOC. NO <b>ELSC60249321-00</b>	REV. -
REF.	REPL.	WEIGHT	SHEET <b>12</b> OF <b>41</b>

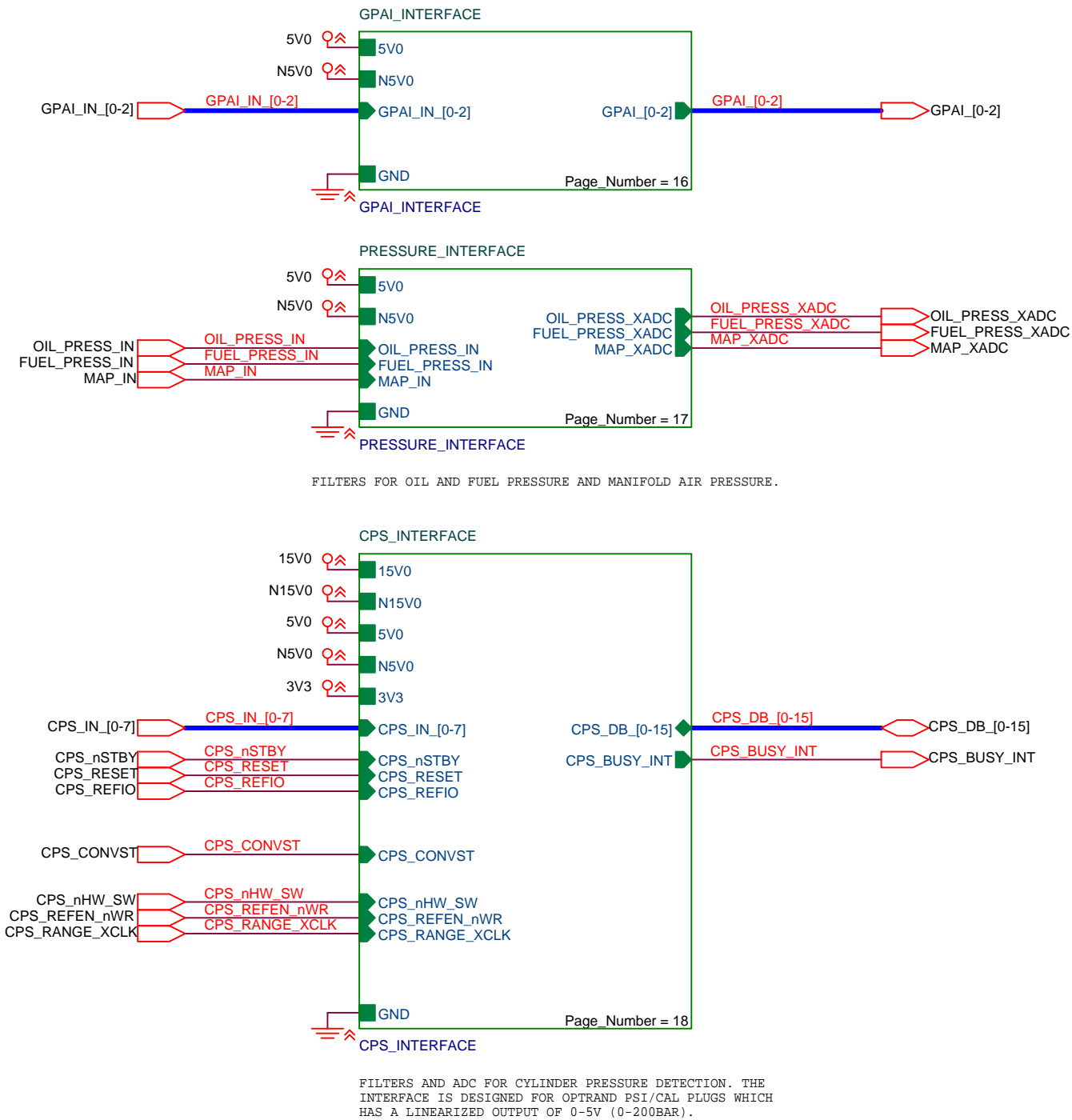


SIZE A3	NSCM NO. -	DOC. NO ELSC60249321-00	REV. -
REF.	REPL.	WEIGHT	SHEET 13 OF 41

/SENSOR\_INTERFACES/ACCELERATOR\_INPUT

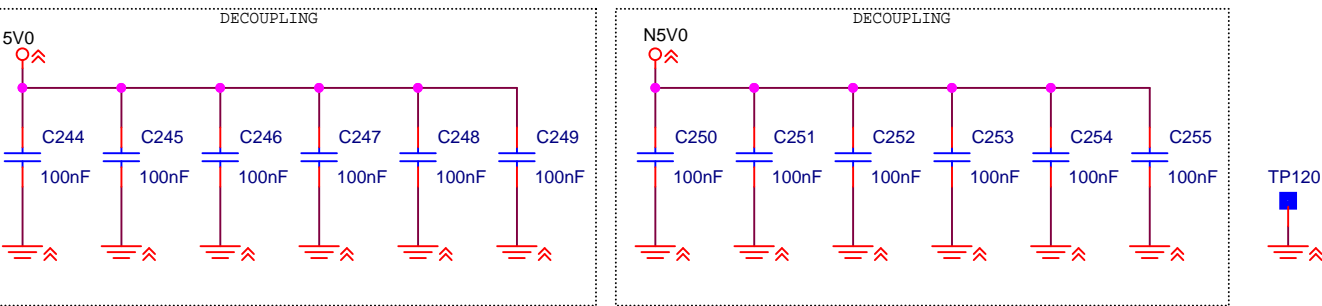
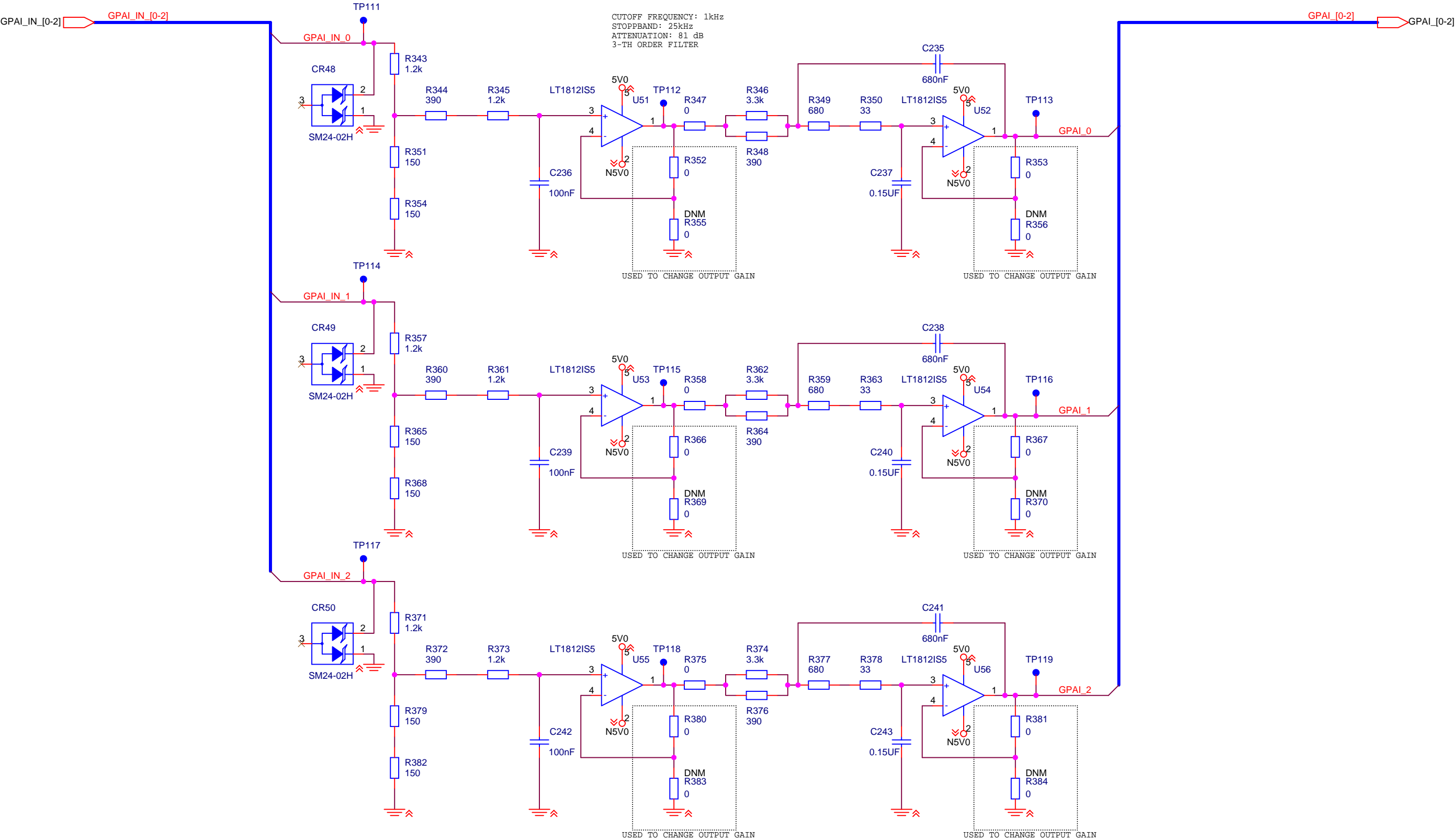


SIZE	NSCM NO.	DOC. NO.	REV.
A3	-	ELSC60249321-00	-
REF.	REPL.	WEIGHT	SHEET 14 OF 41



SIZE	NSCM NO.	DOC. NO	REV.
A3	-	ELSC60249321-00	-
REF.	REPL.	WEIGHT	SHEET 15 OF 41

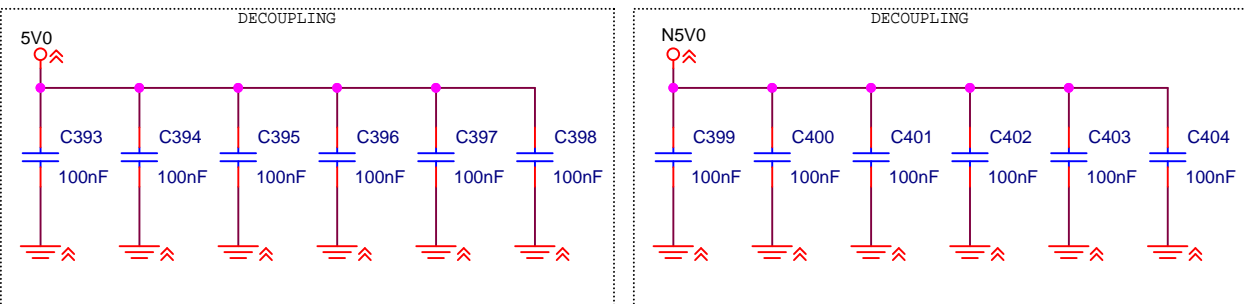
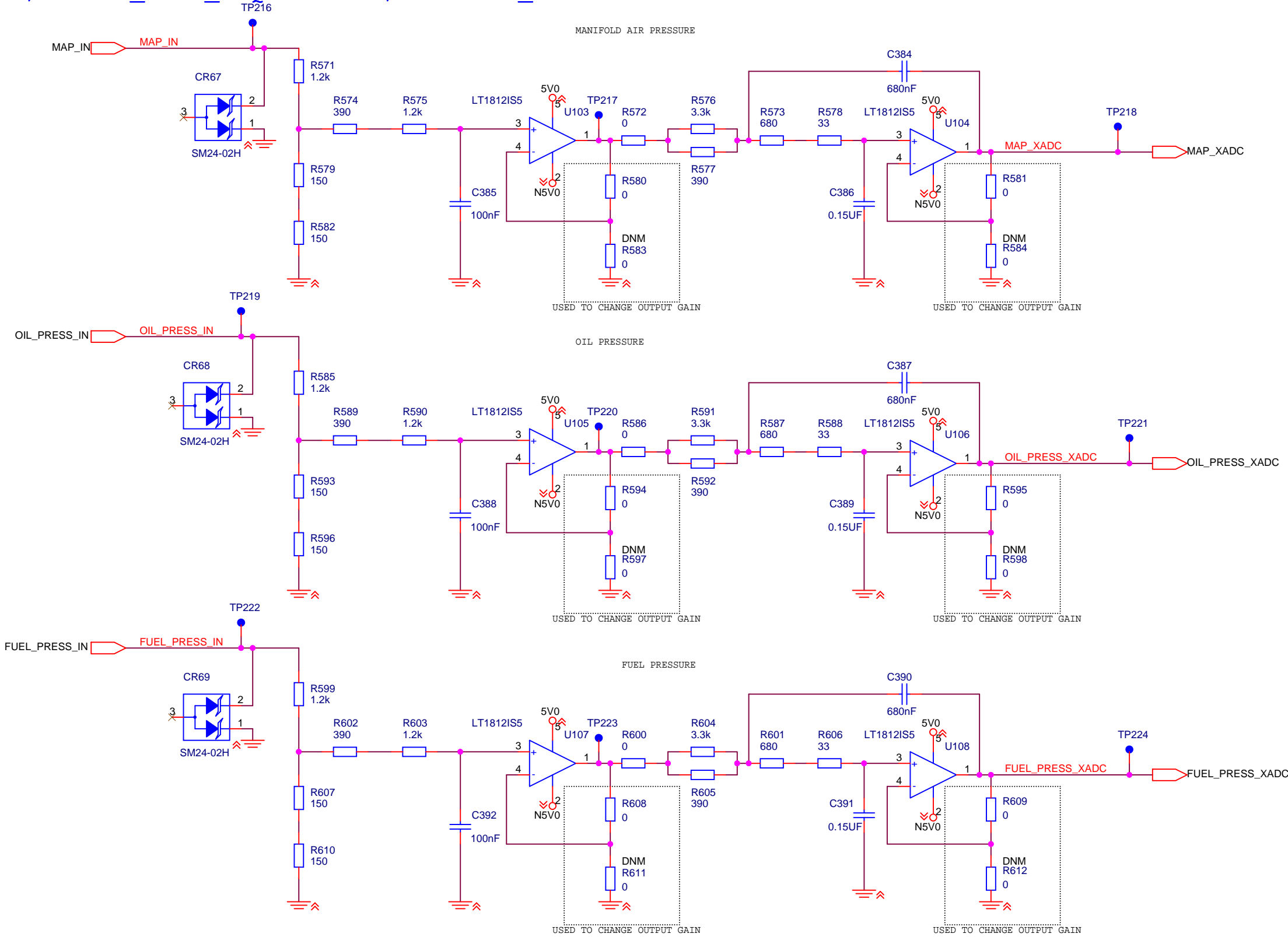
/SENSOR\_INTERFACES/ANALOG\_DATA\_ACQUISITION/GPAI\_INTERFACE



SIZE	NSCM NO.	DOC. NO.	REV.
A3	-	ELSC60249321-00	-
REF.	REPL.	WEIGHT	SHEET 16 OF 41

/SENSOR\_INTERFACES/ANALOG\_DATA\_ACQUISITION/PRESSURE\_INTERFACE

CUTOFF FREQUENCY: 1kHz  
STOPPBAND: 24kHz  
ATTENUATION: 81 dB  
3-TH ORDER FILTER



SIZE	NSCM NO.	DOC. NO	REV.
A3	-	ELSC60249321-00	-
REF.	REPL.	WEIGHT	SHEET 17 OF 41

/SENSOR\_INTERFACES/ANALOG\_DATA\_ACQUISITION/CPS\_INTERFACE

A

B

C

D

A

B

C

D

CPS\_IN\_[0-7]

CPS\_IN\_[0-7]

5V0

CR20  
DRTR5V0U4TS

CPS\_FILTER\_BANK1

5V0

N5V0

CPS\_IN\_0

CPS\_IN\_1

CPS\_IN\_2

CPS\_IN\_3

TP51

TP52

TP53

TP54

TP55

GND

CPS\_FILTER\_BANK1

Page\_Number = 19

CPS\_FILTER\_BANK2

5V0

N5V0

CPS\_IN\_4

CPS\_IN\_5

CPS\_IN\_6

CPS\_IN\_7

TP56

TP57

TP58

TP59

GND

CPS\_FILTER\_BANK2

Page\_Number = 20

CR21  
DRTR5V0U4TS

CPS\_ADC

15V0

N15V0

5V0

3V3

CPS\_ADC\_IN\_0

CPS\_ADC\_IN\_1

CPS\_ADC\_IN\_2

CPS\_ADC\_IN\_3

CPS\_ADC\_IN\_4

CPS\_ADC\_IN\_5

CPS\_ADC\_IN\_6

CPS\_ADC\_IN\_7

CPS\_ADC\_IN\_0

CPS\_ADC\_IN\_1

CPS\_ADC\_IN\_2

CPS\_ADC\_IN\_3

CPS\_ADC\_IN\_4

CPS\_ADC\_IN\_5

CPS\_ADC\_IN\_6

CPS\_ADC\_IN\_7

CPS\_DB\_[0-15]

CPS\_DB\_[0-15]

CPS\_DB\_[0-15]

CPS\_BUSY\_INT

CPS\_BUSY\_INT

CPS\_BUSY\_INT

CPS\_nSTBY

CPS\_nSTBY

CPS\_nSTBY

CPS\_RESET

CPS\_RESET

CPS\_RESET

CPS\_REFIO

CPS\_REFIO

CPS\_REFIO

CPS\_CONVST

CPS\_CONVST

CPS\_CONVST

CPS\_nHW\_SW

CPS\_nHW\_SW

CPS\_nHW\_SW

CPS\_REFEN\_nWR

CPS\_REFEN\_nWR

CPS\_REFEN\_nWR

CPS\_RANGE\_XCLK

CPS\_RANGE\_XCLK

CPS\_RANGE\_XCLK

GND

CPS\_ADC

Page\_Number = 21

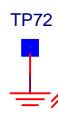
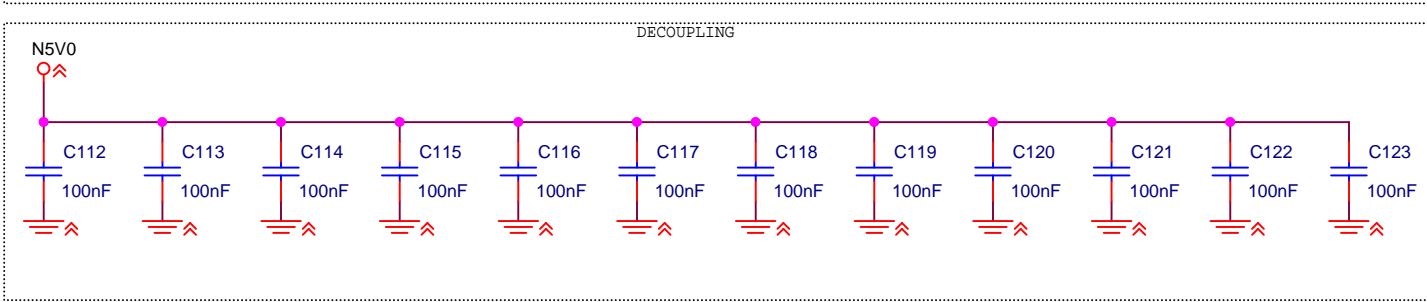
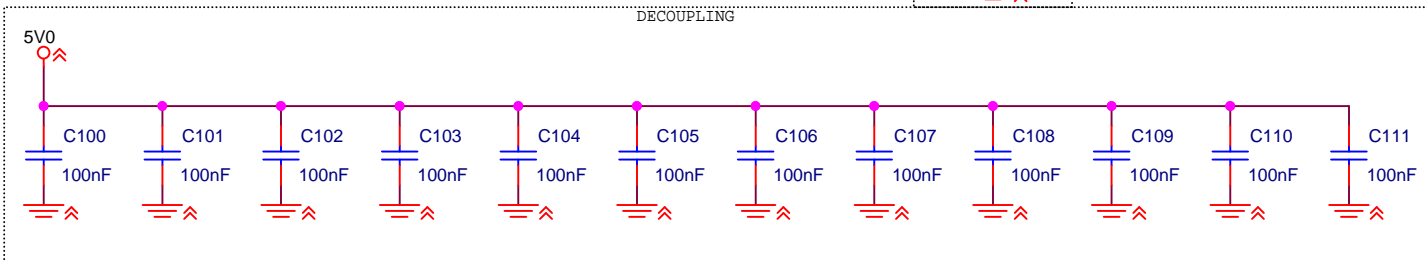
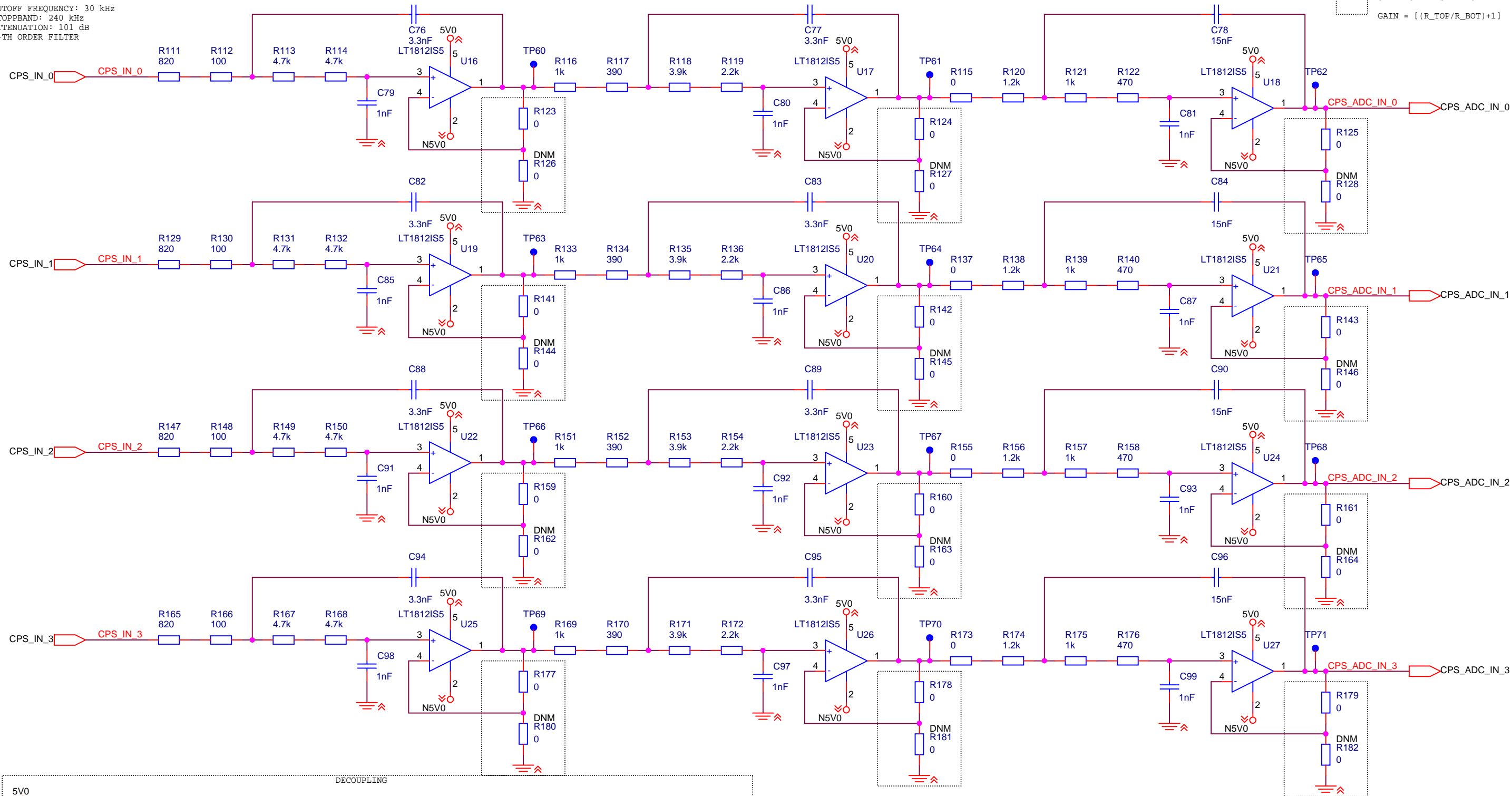
SIZE	NSCM NO.	DOC. NO	REV.
A3	-	ELSC60249321-00	-
REF.	REPL.	WEIGHT	SHEET 18 OF 41



/SENSOR\_INTERFACES/ANALOG\_DATA\_ACQUISITION/CPS\_INTERFACE/CPS\_FILTER\_BANK1

CUTOFF FREQUENCY: 30 kHz  
STOPPBAND: 240 kHz  
ATTENUATION: 101 dB  
6-TH ORDER FILTER

USED TO CHANGE OUTPUT GAIN  
UNITY GAIN AS DEFAULT  
GAIN = [(R\_TOP/R\_BOT)+1]



SIZE	NSCM NO.	DOC. NO	REV.
A3	-	ELSC60249321-00	-
REF.	REPL.	WEIGHT	SHEET 19 OF 41

# /SENSOR\_INTERFACES/ANALOG\_DATA\_ACQUISITION/CPS\_INTERFACE/CPS\_FILTER\_BANK2

CUTOFF FREQUENCY: 30 kHz  
STOPPBAND: 240 kHz  
ATTENUATION: 101 dB  
6-TH ORDER FILTER

USED TO CHANGE OUTPUT GAIN  
UNITY GAIN AS DEFAULT  
GAIN =  $[(R_{TOP}/R_{BOT})+1]$

A

B

C

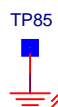
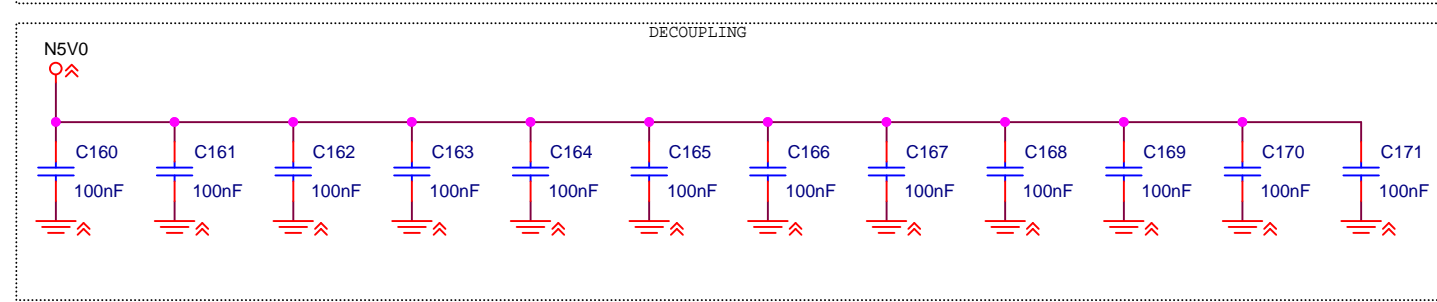
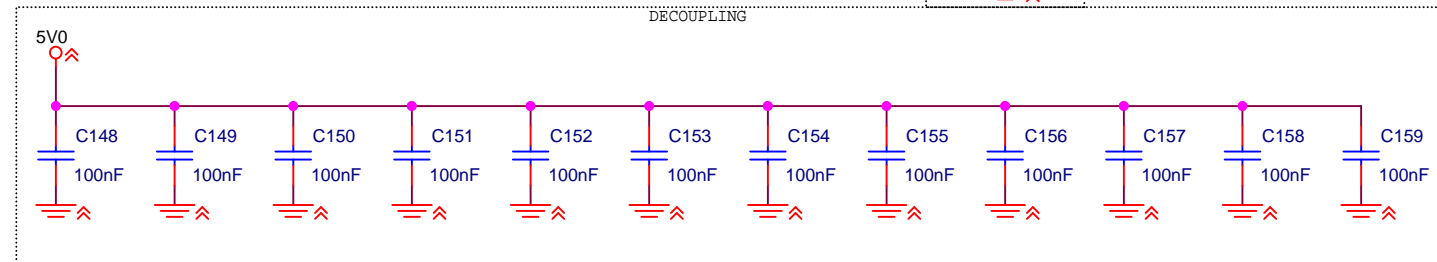
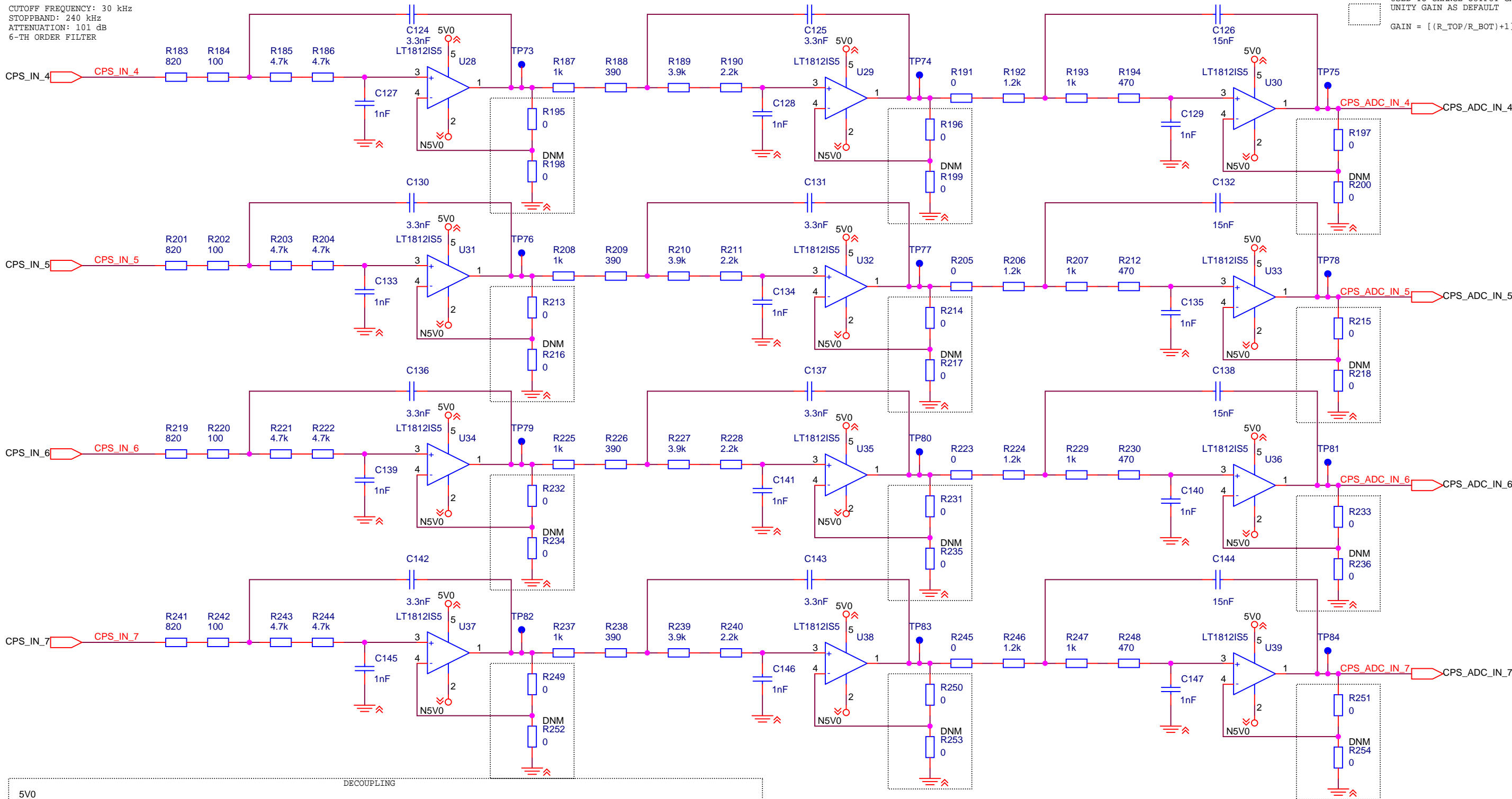
D

A

B

C

D



SIZE <b>A3</b>	NSCM NO. -	DOC. NO. <b>ELSC60249321-00</b>	REV. -
REF.	REPL.	WEIGHT	SHEET 20 OF 41

A

A

B

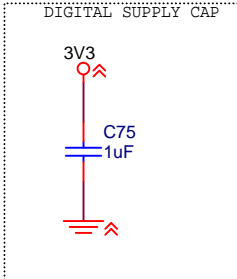
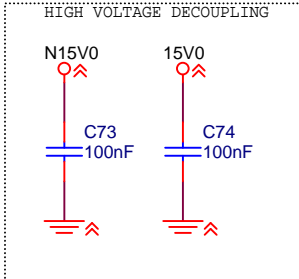
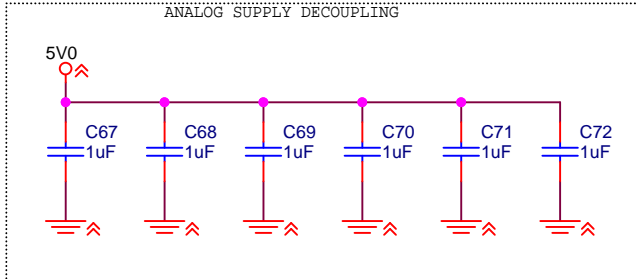
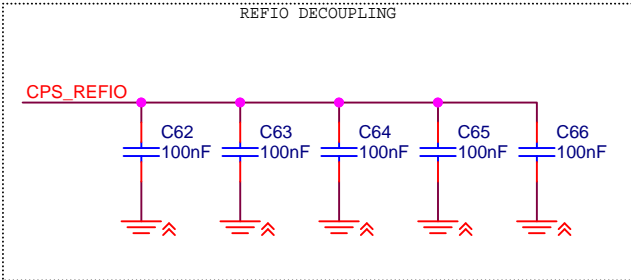
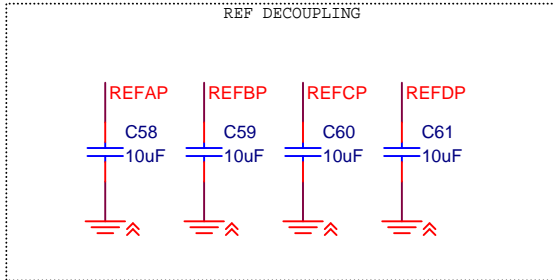
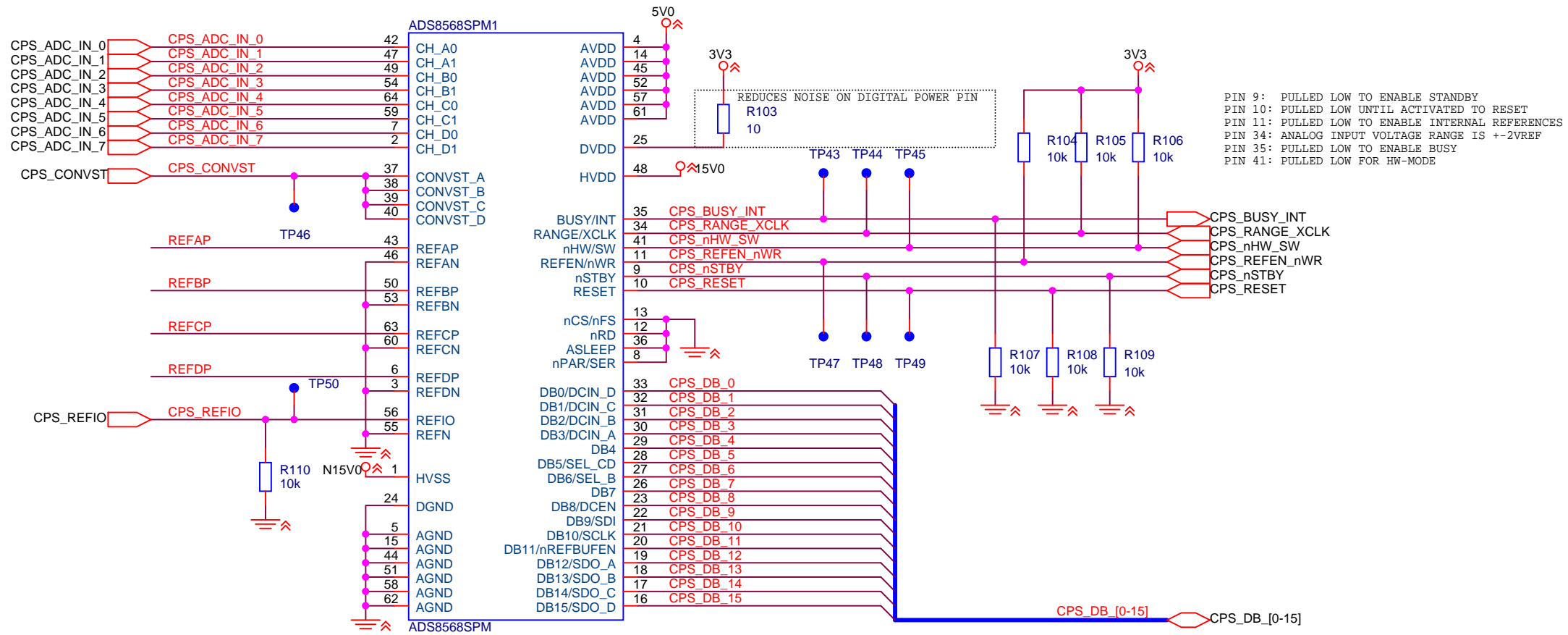
B

C

C

D

D



SIZE	NSCM NO.	DOC. NO.	REV.
A3	-	ELSC60249321-00	-
REF.	REPL.	WEIGHT	SHEET 21 OF 41

A

B

C

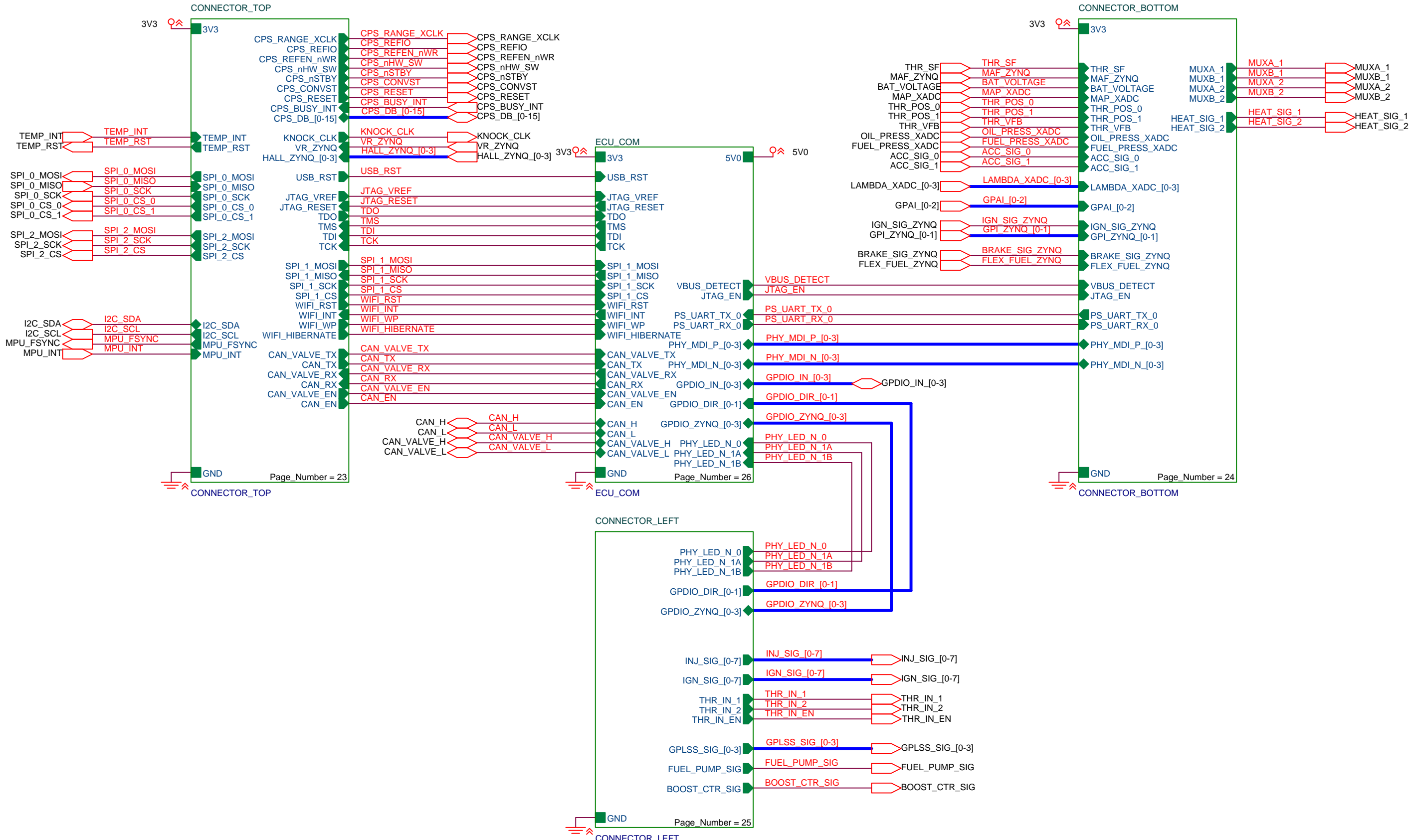
D

A

B

C

D



SIZE	NSCM NO.	DOC. NO	REV.
A3	-	ELSC60249321-00	-
REF.	REPL.	WEIGHT	SHEET 22 OF 41

A

B

C

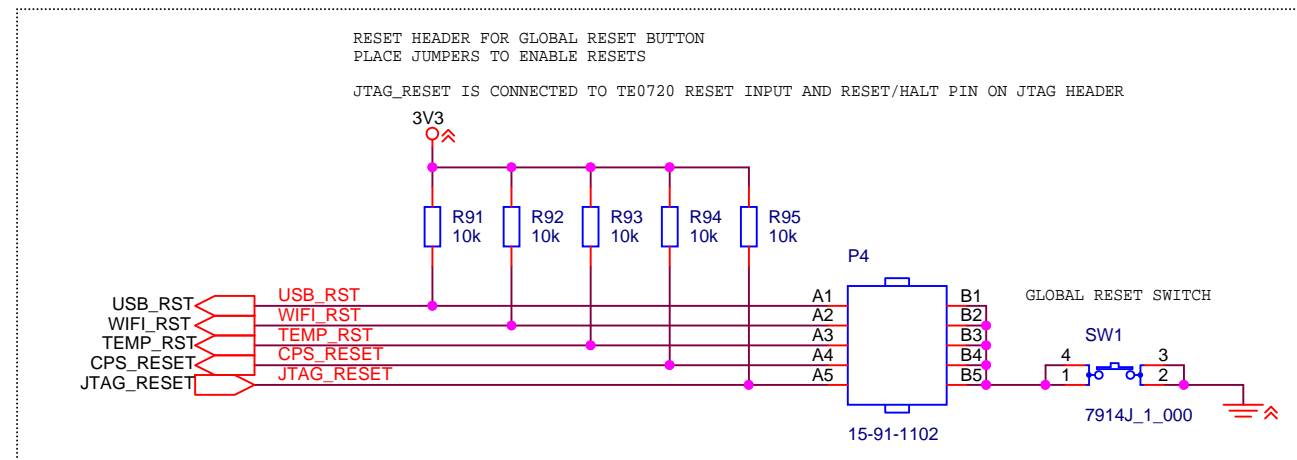
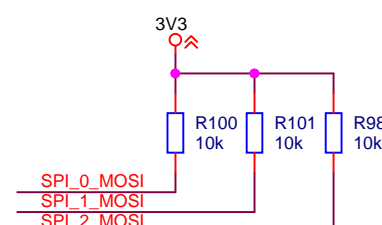
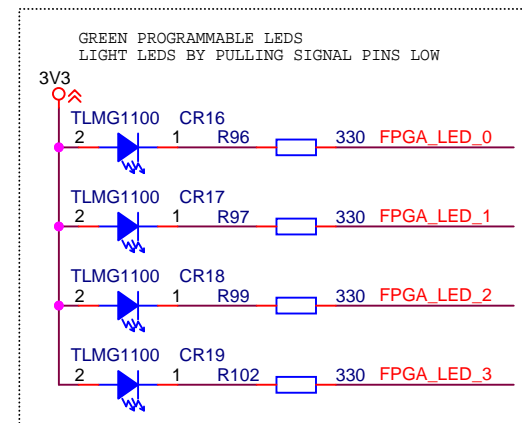
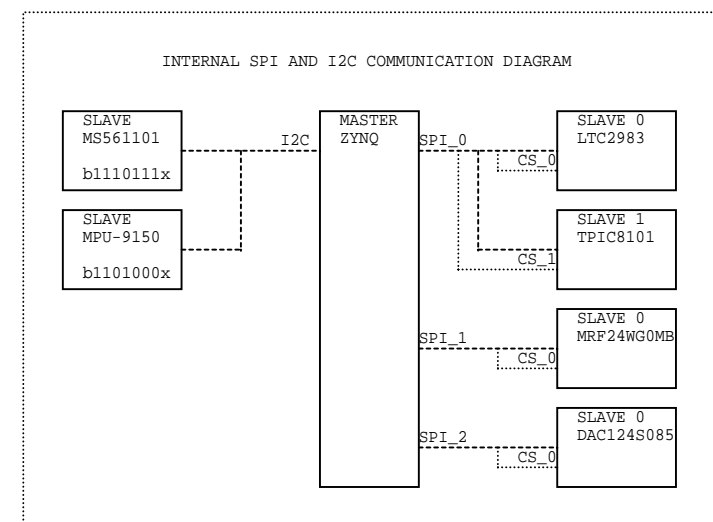
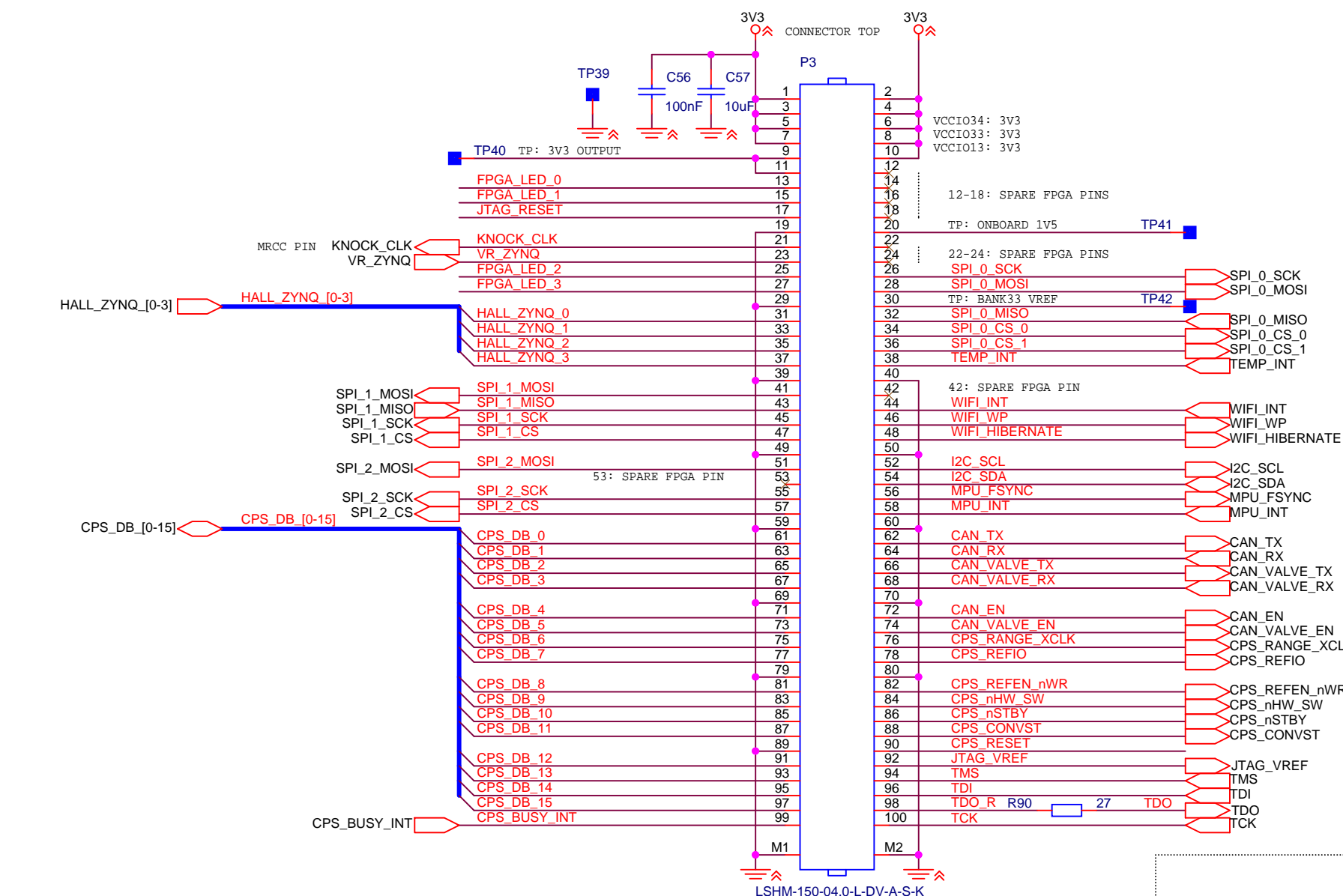
D

A

B

C

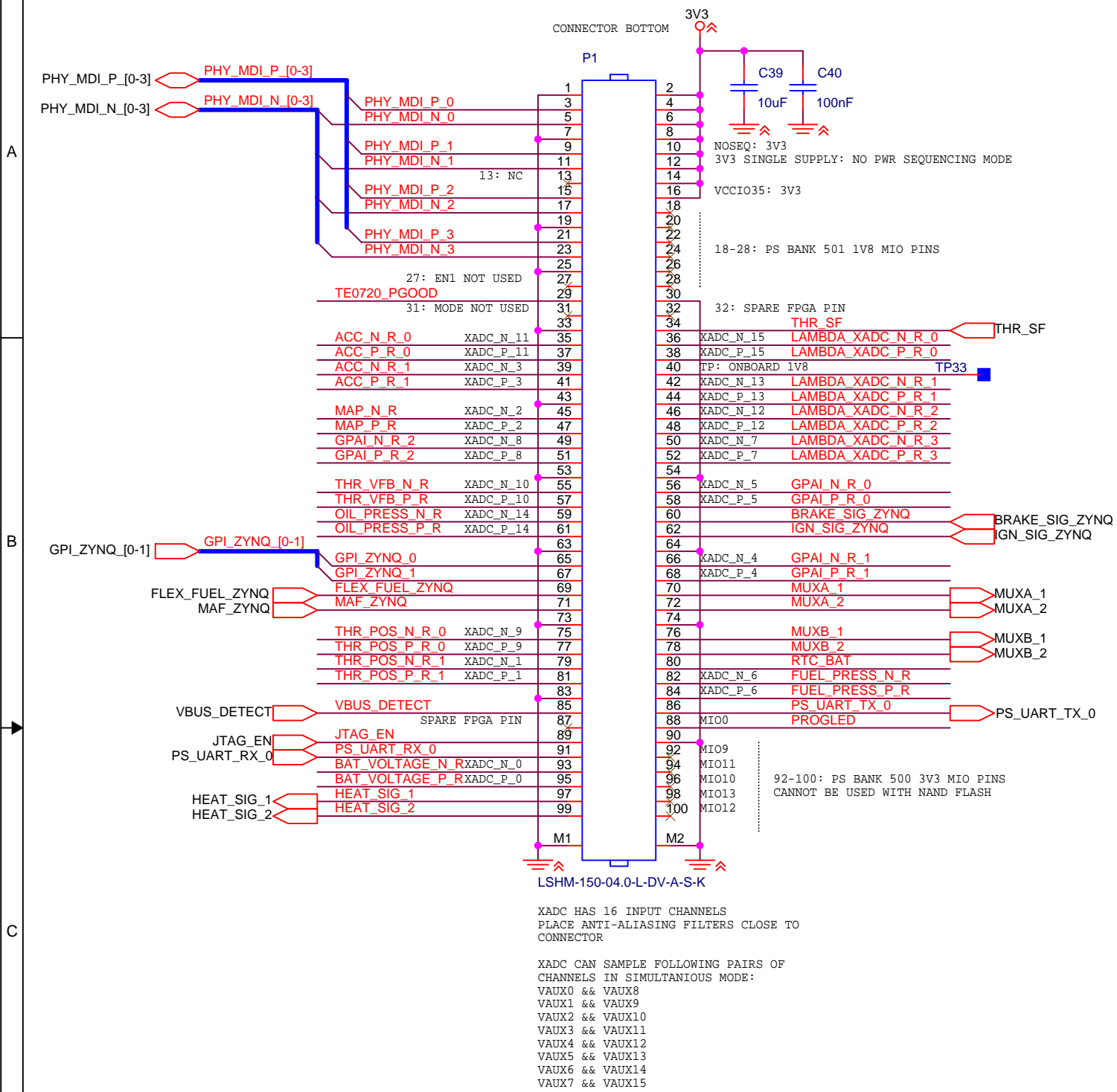
D

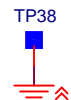
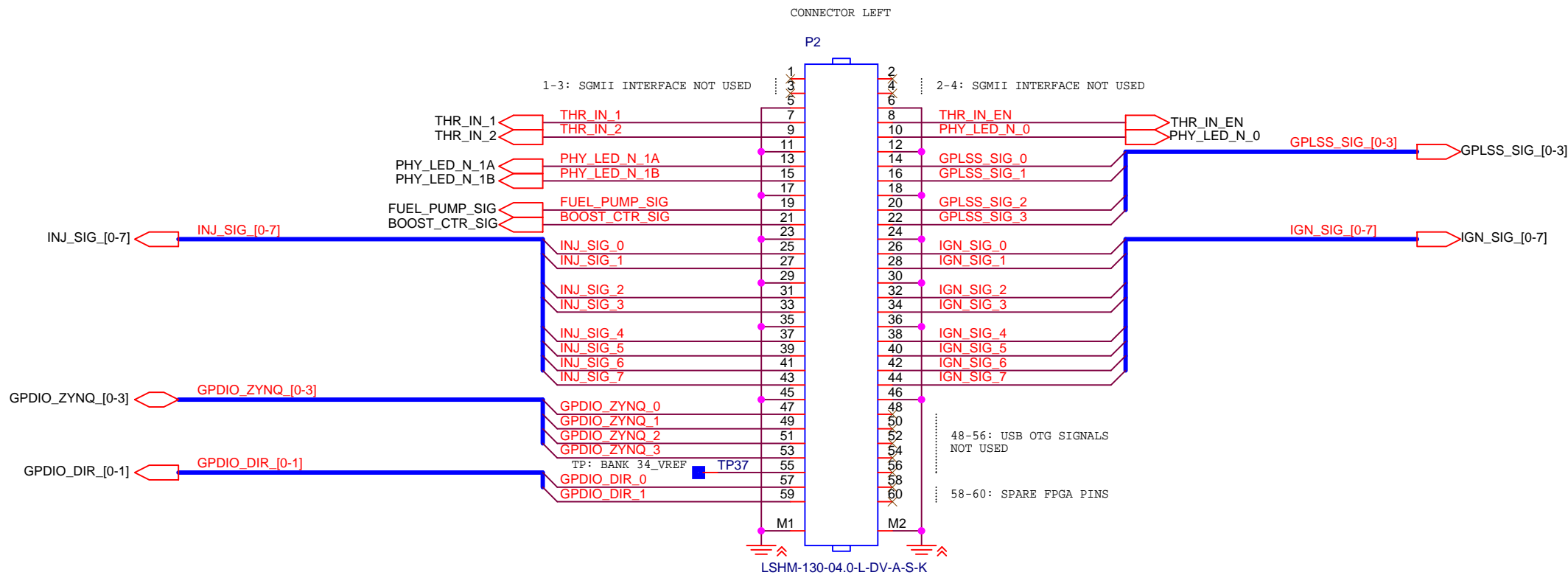


SIZE	NSCM NO.	DOC. NO	REV.
A3	-	ELSC60249321-00	-
REF.	REPL.	WEIGHT	SHEET 23 OF 41



/TE0720\_B2B/CONNECTOR\_BOTTOM

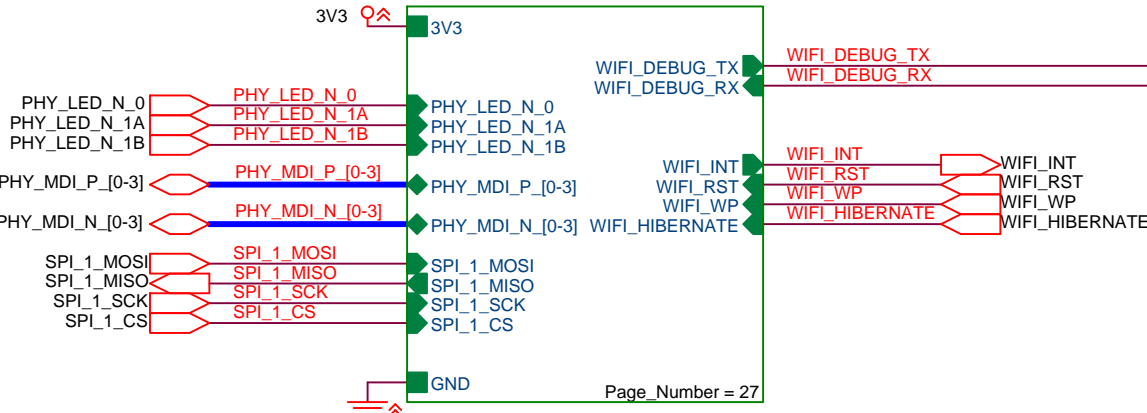




SIZE	NSCM NO.	DOC. NO	REV.
A3	-	ELSC60249321-00	-
REF.	REPL.	WEIGHT	SHEET 25 OF 41

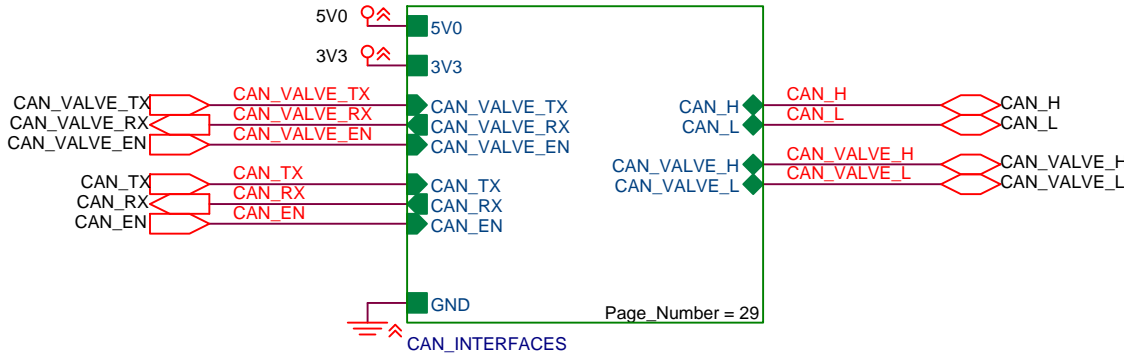
ETHERNET AND WIFI INTERFACES

ETHERNET\_WIFI



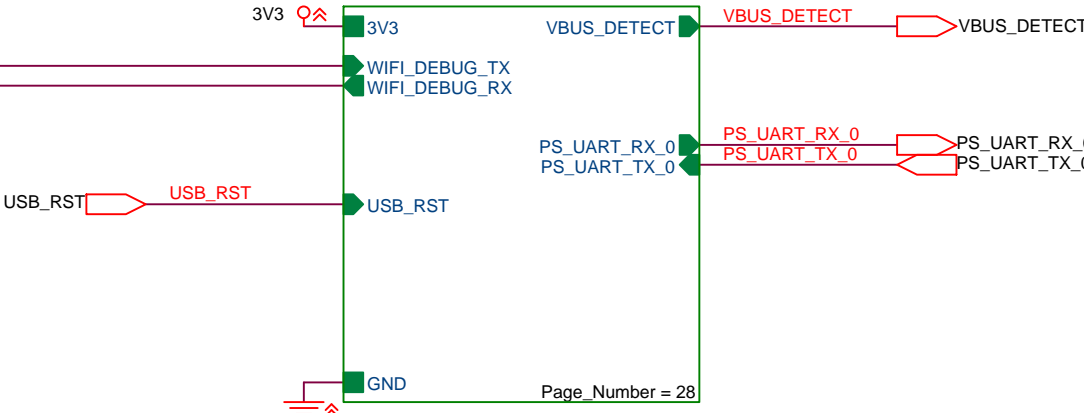
CAN BUS INTERFACES  
- CAN COMMUNICATION BUS  
- PNEUMATIC VALVE MODULE CAN BUS

CAN\_INTERFACES



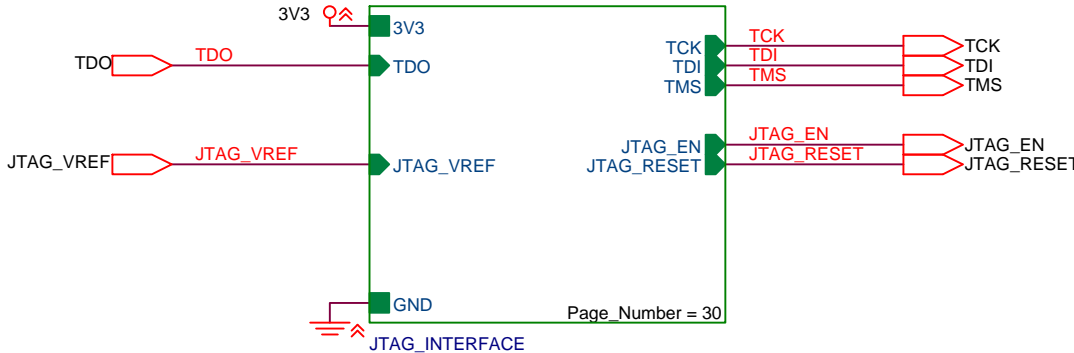
USB INTERFACES  
- USB TO ZYNQ PS (UART0)  
- USB TO WIFI DEBUG (UART)

USB\_INTERFACE

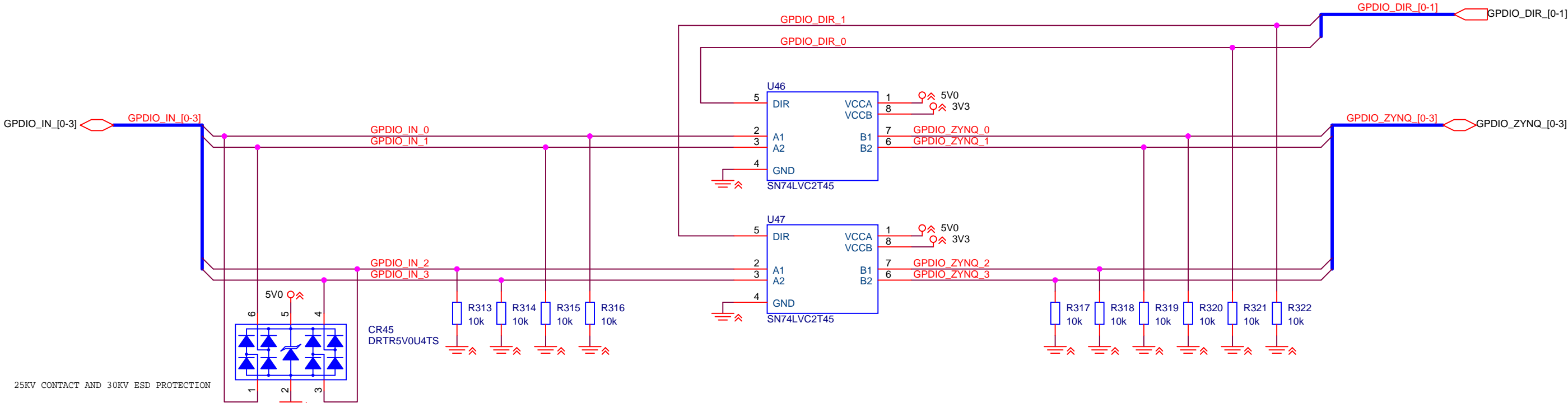


JTAG INTERFACE

JTAG\_INTERFACE

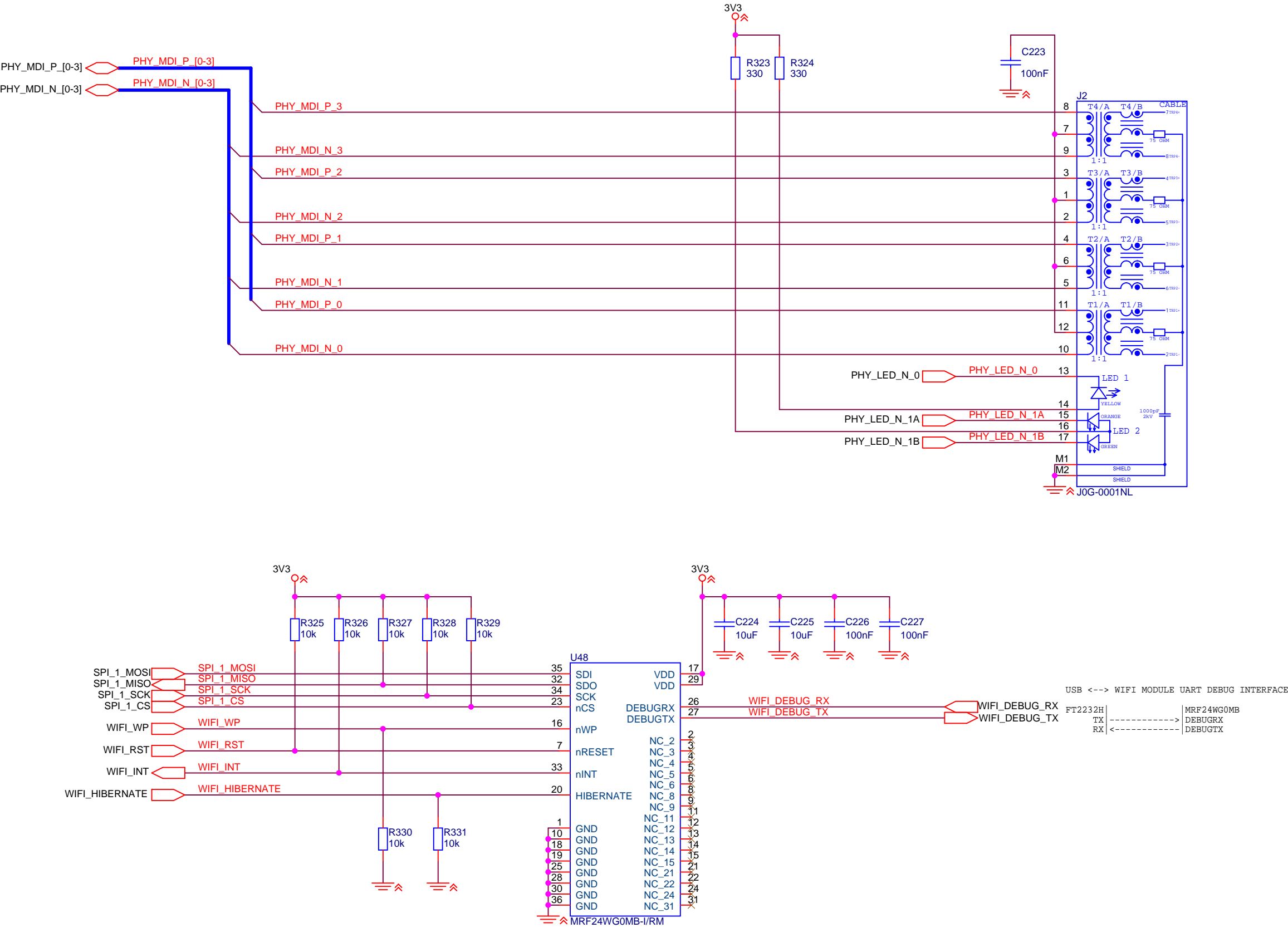


GENERAL PURPOSE DIGITAL I/O. 5V TO 3.3V AND VICE VERSA.  
DIR PIN CONTROLS THE DIRECTION OF THE DEVICE.  
HIGH ON DIR IS A TO B, LOW IS B TO A



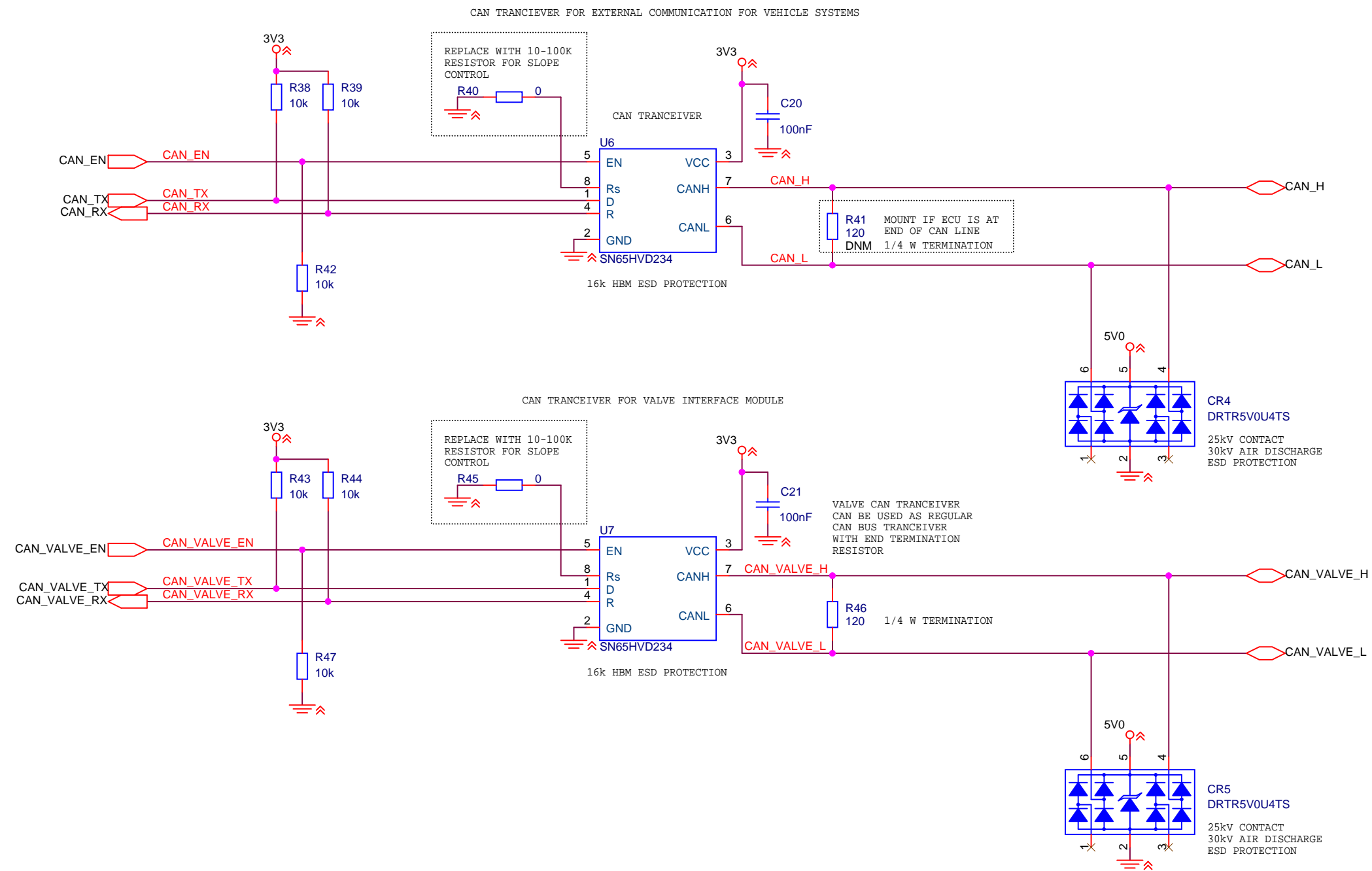
SIZE A3	NSCM NO. -	DOC. NO ELSC60249321-00	REV. -
REF.	REPL.	WEIGHT	SHEET 26 OF 41





## A

D

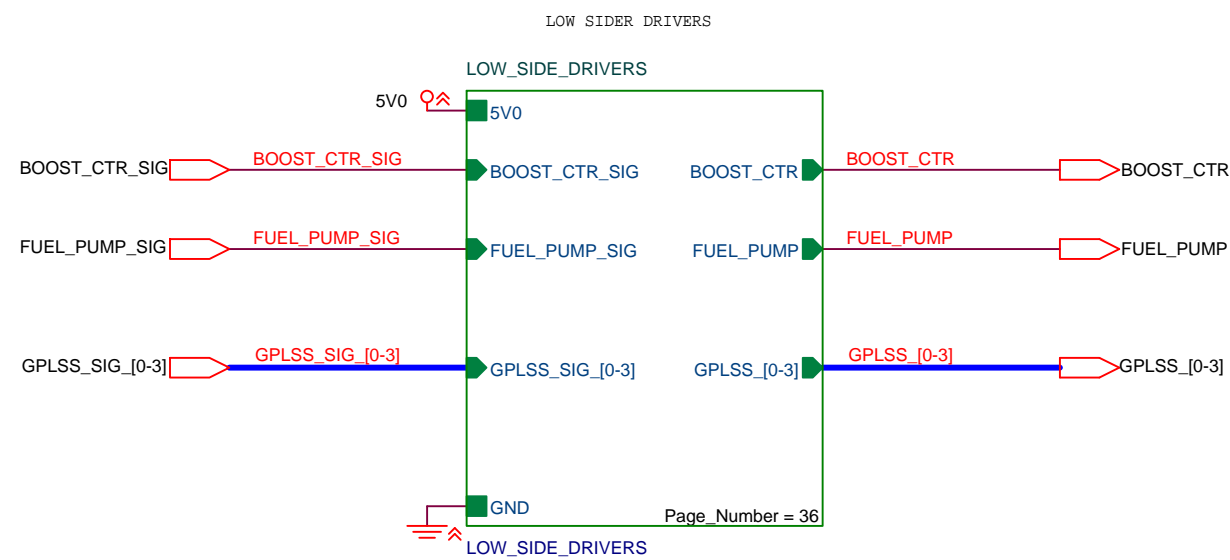
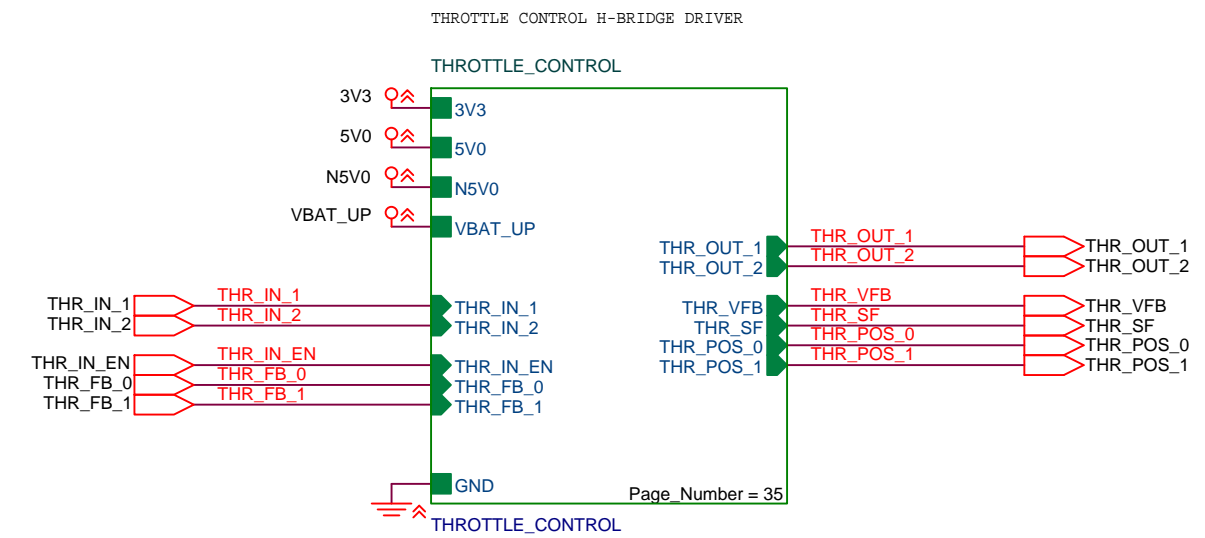
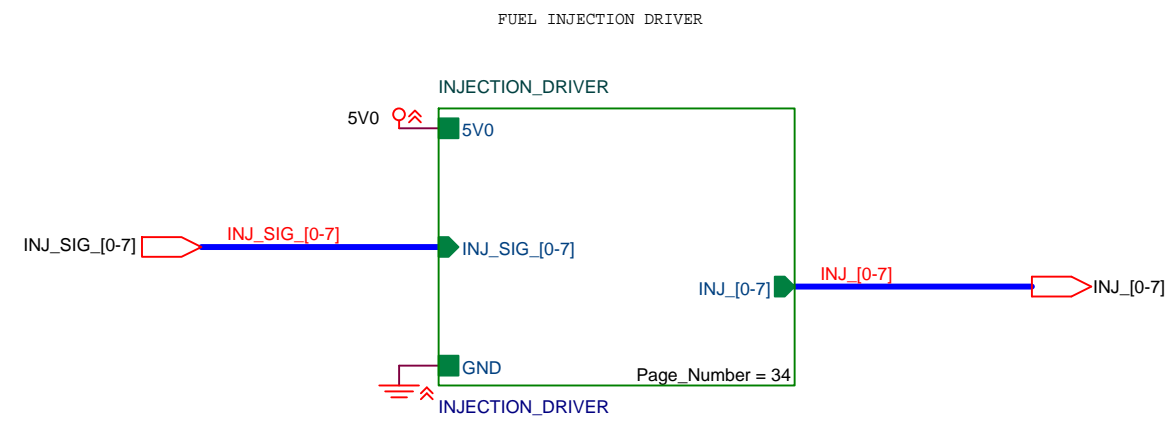
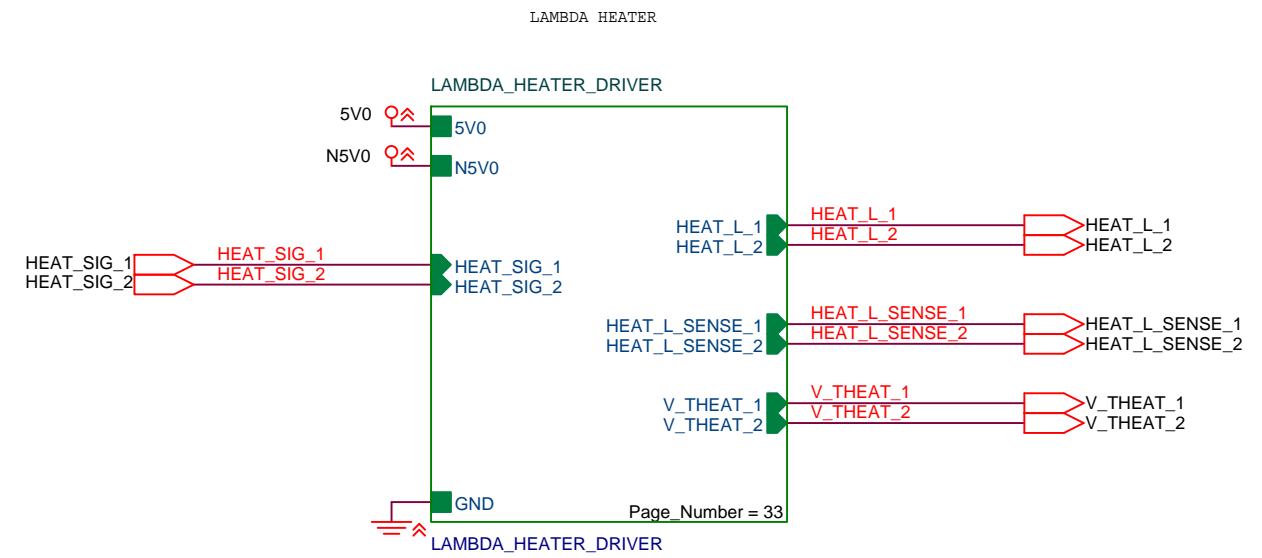
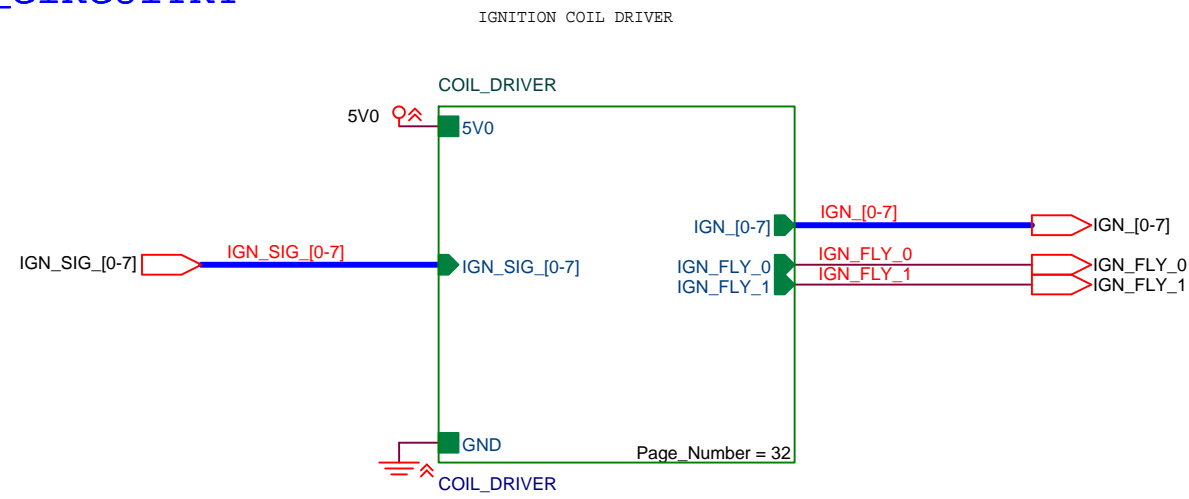


SIZE	NSCM NO.	DOC. NO	REV.
A3	-	ELSC60249321-00	-
REF.	REPL.	WEIGHT	SHEET 29 OF 41



3

```
/DRIVER_CIRCUITRY
```



SIZE <b>A3</b>	NSCM NO. -	DOC. NO <b>ELSC60249321-00</b>	REV. -
REF.	REPL.	WEIGHT	SHEET <b>31</b> OF <b>41</b>

/DRIVER\_CIRCUITRY/COIL\_DRIVER

A

B

C

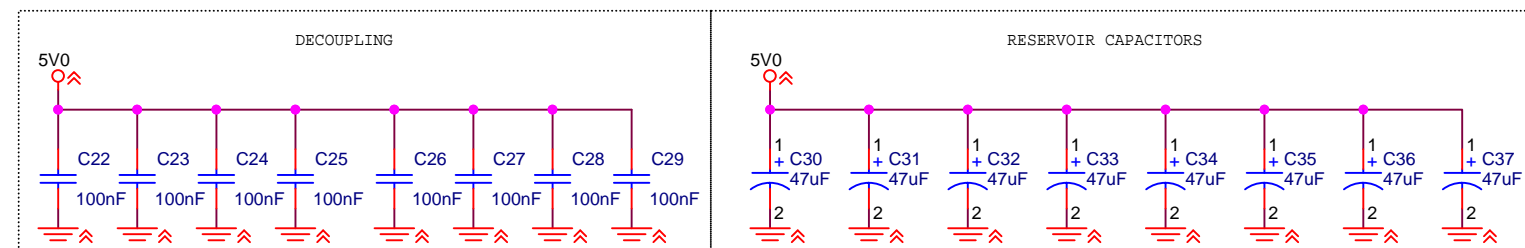
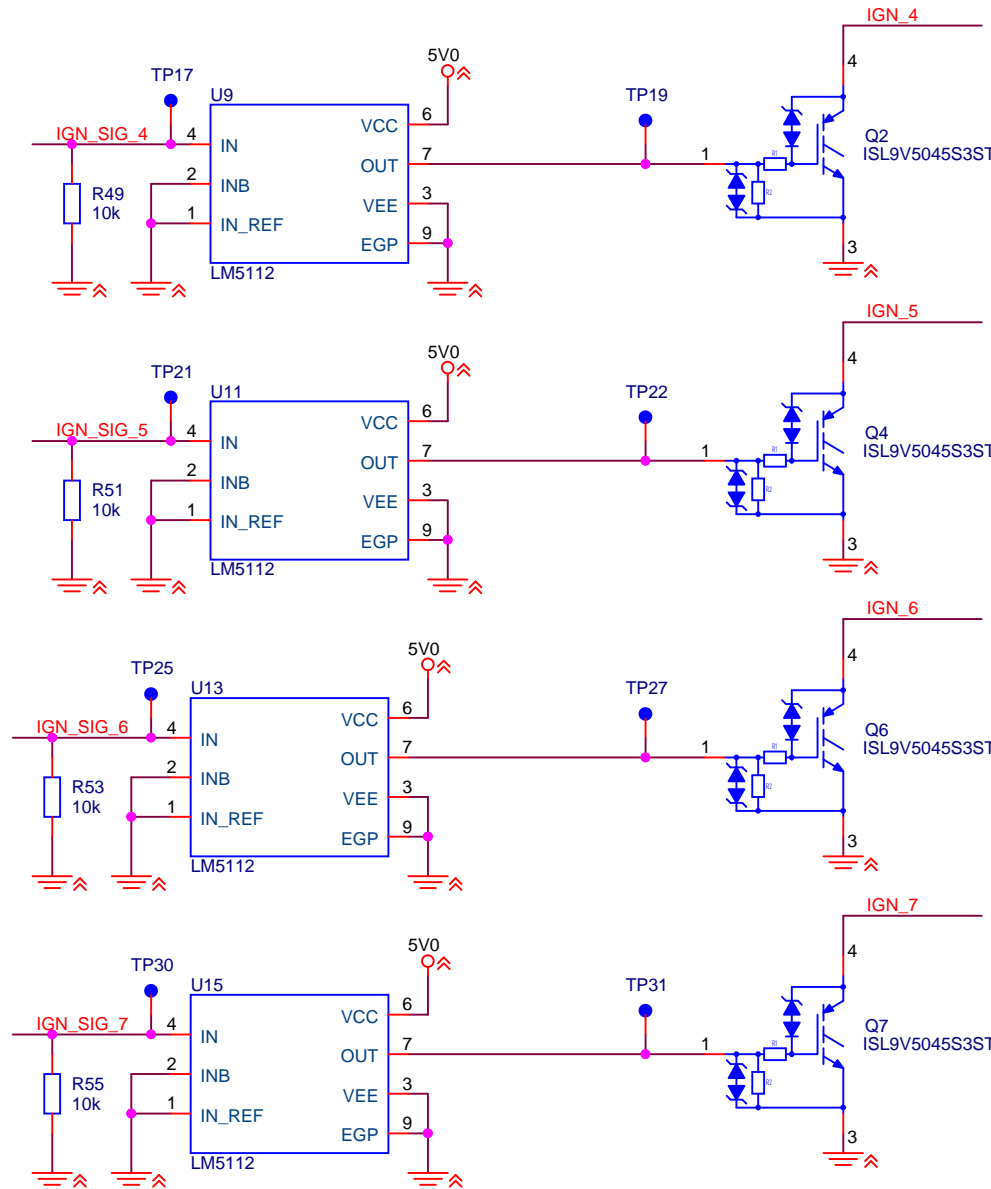
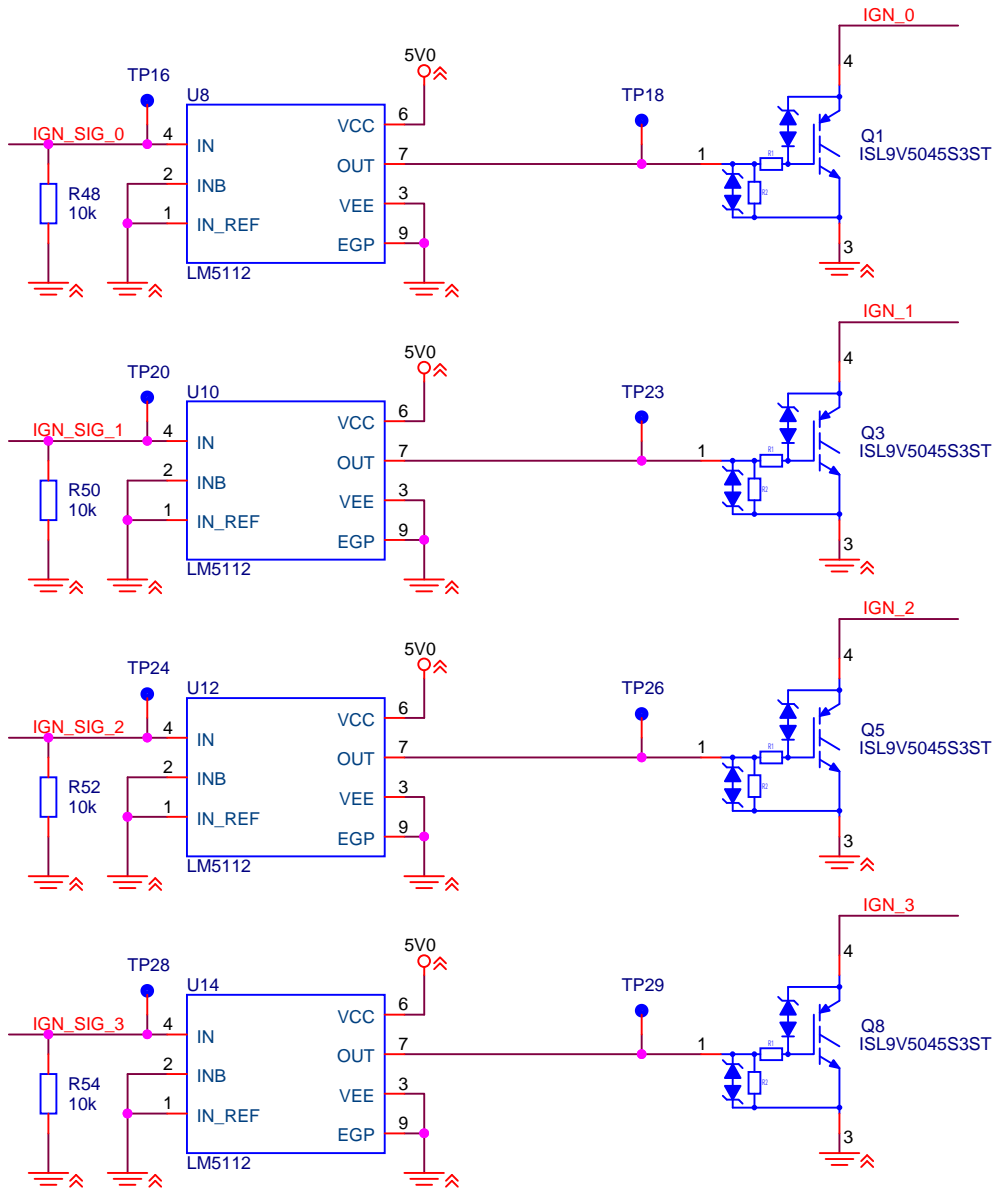
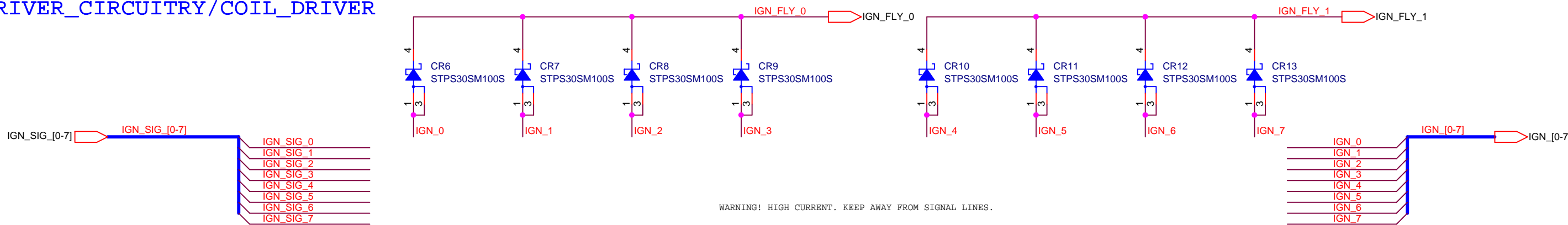
D

A

B

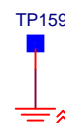
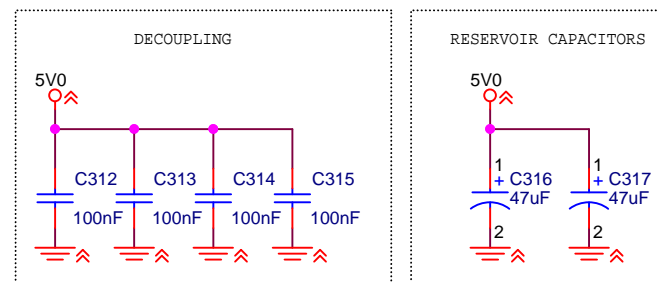
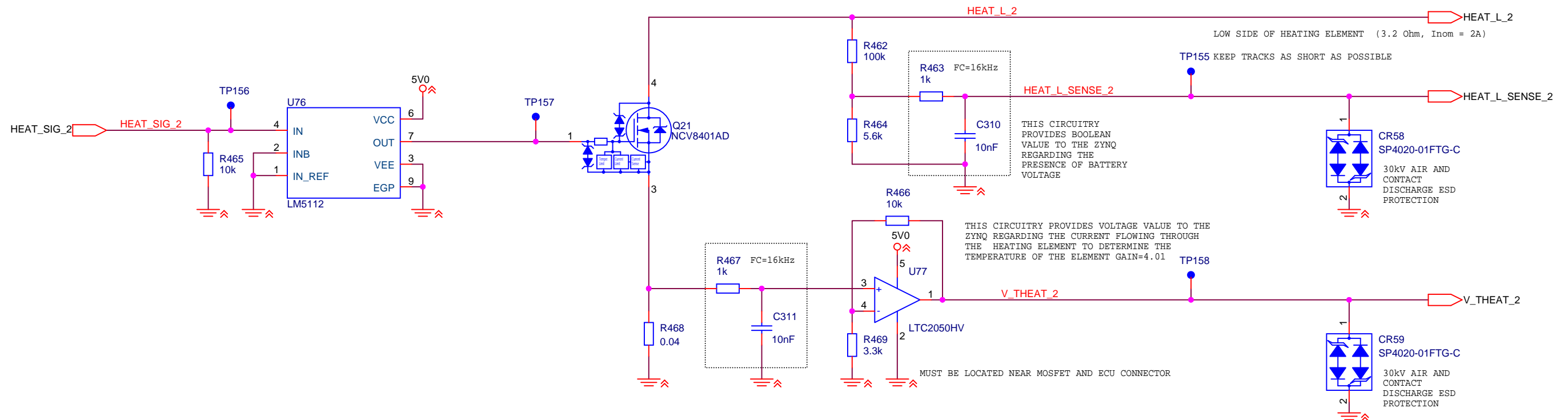
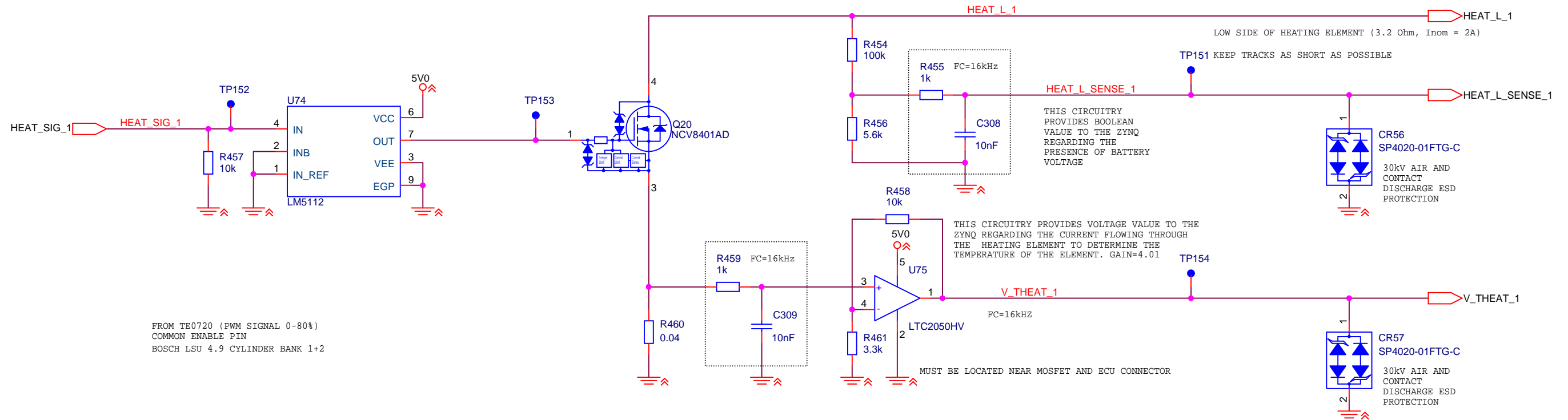
C

D



SIZE	NSCM NO.	DOC. NO	REV.
A3	-	ELSC60249321-00	-
REF.	REPL.	WEIGHT	SHEET 32 OF 41

/DRIVER\_CIRCUITRY/LAMBDA\_HEATER\_DRIVER



SIZE	NSCM NO.	DOC. NO	REV.
A3	-	ELSC60249321-00	-
REF.	REPL.	WEIGHT	SHEET 33 OF 41

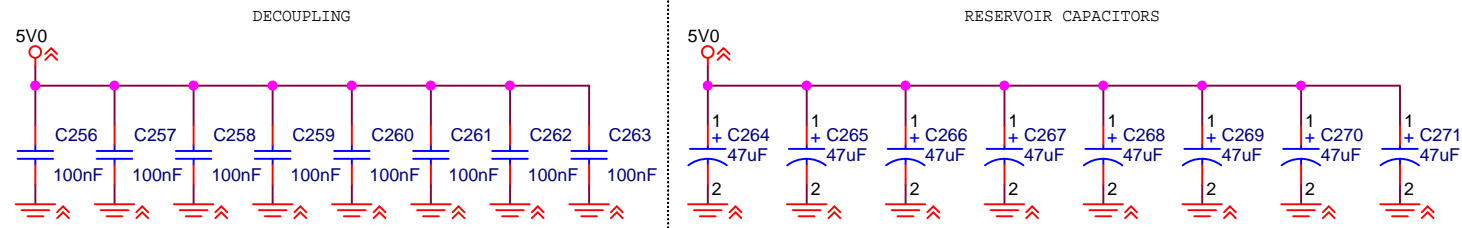
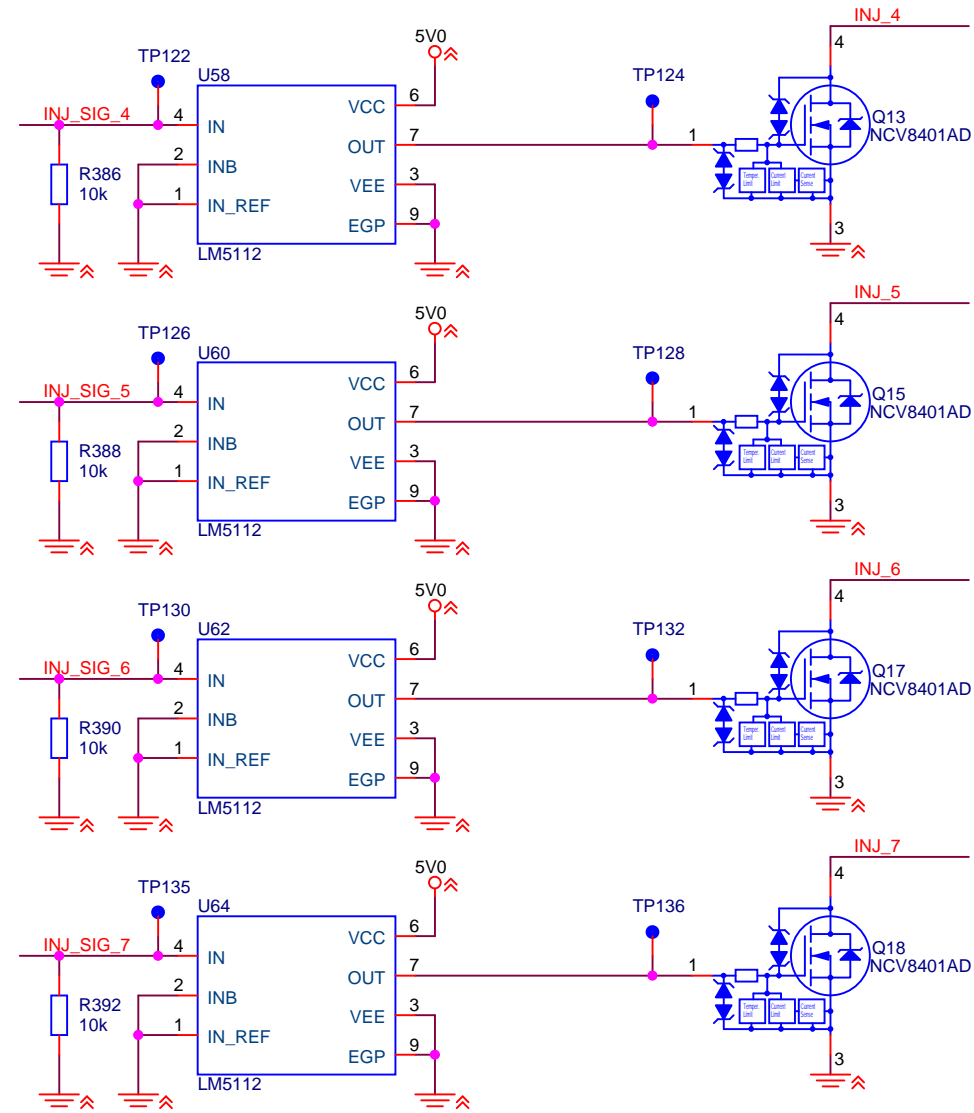
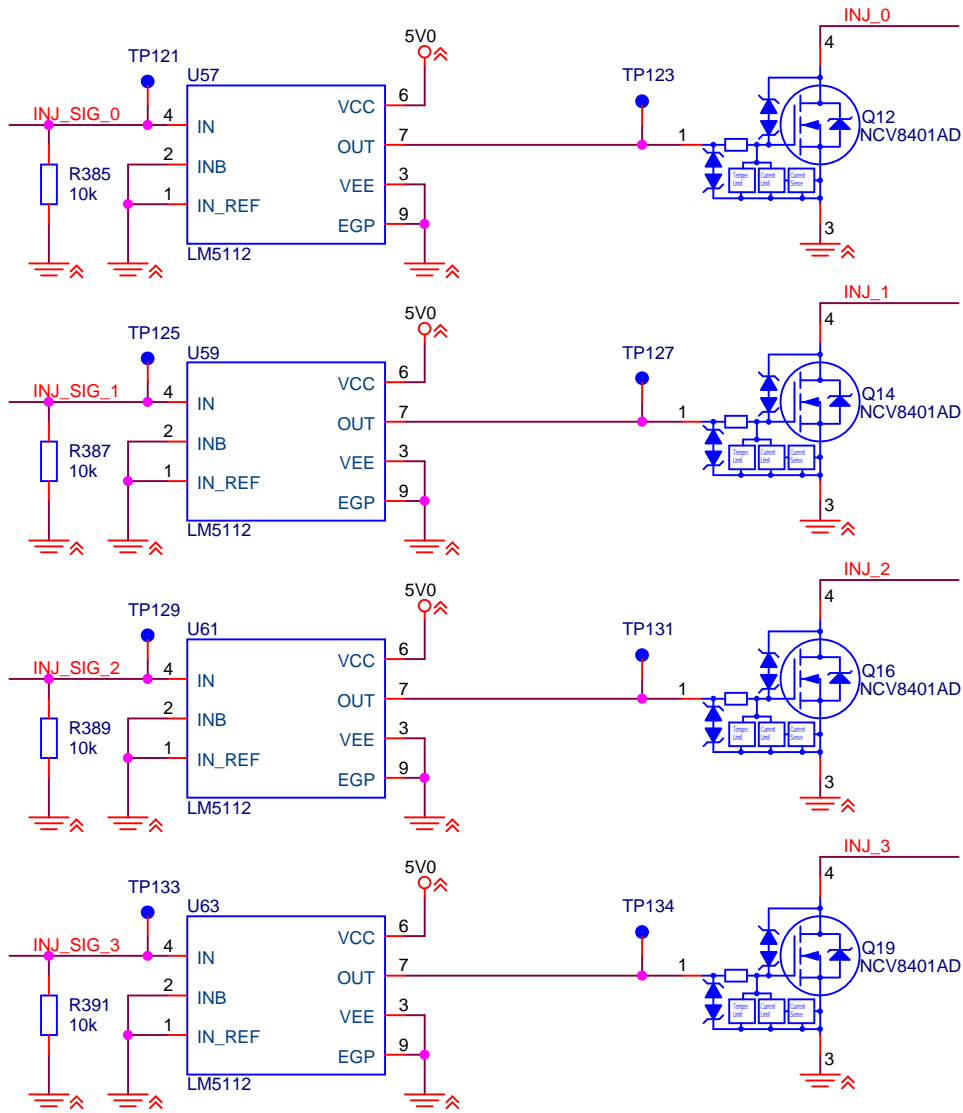
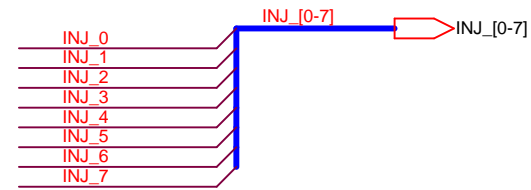
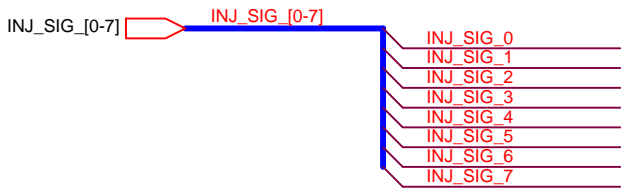
/DRIVER\_CIRCUITRY/INJECTION\_DRIVER

A

B

C

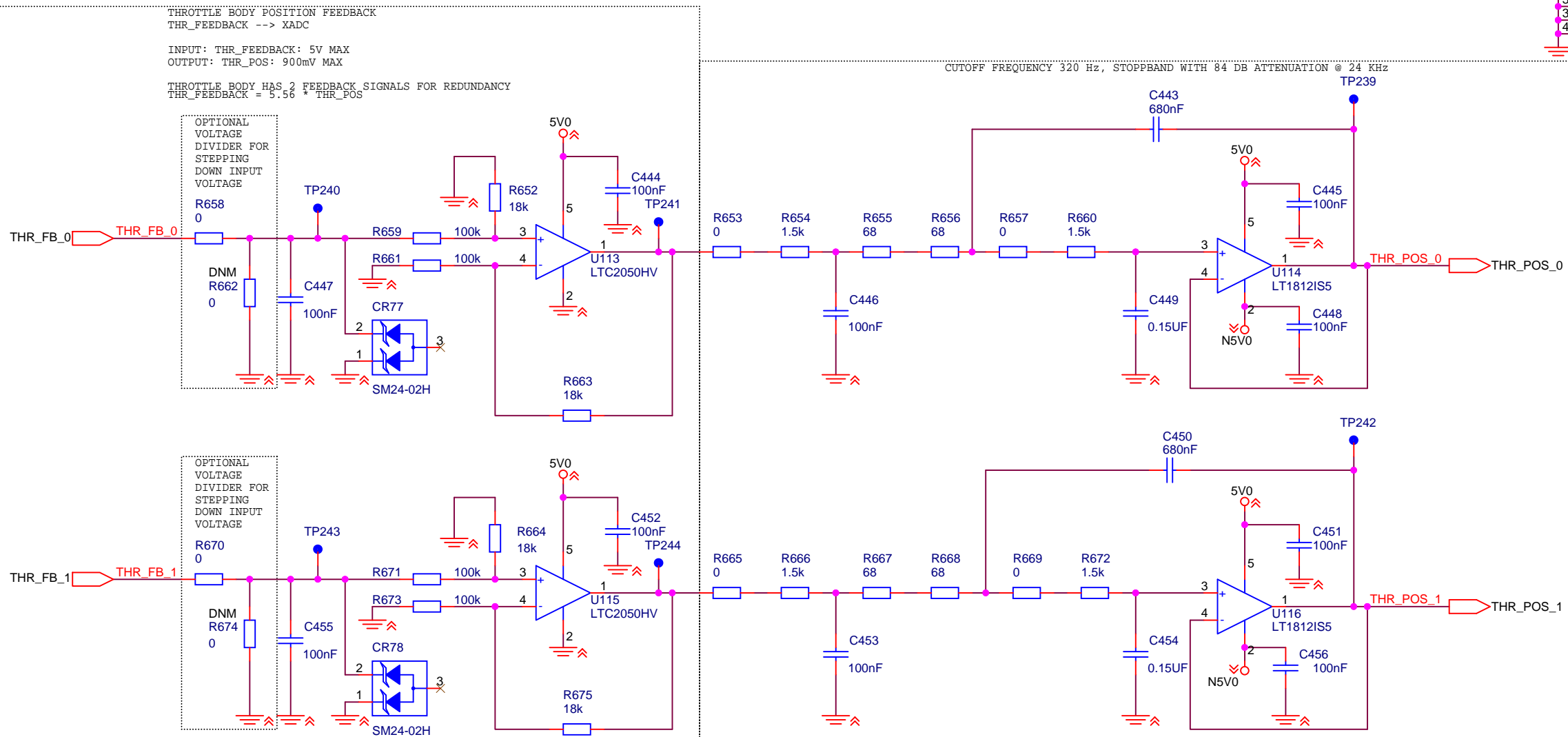
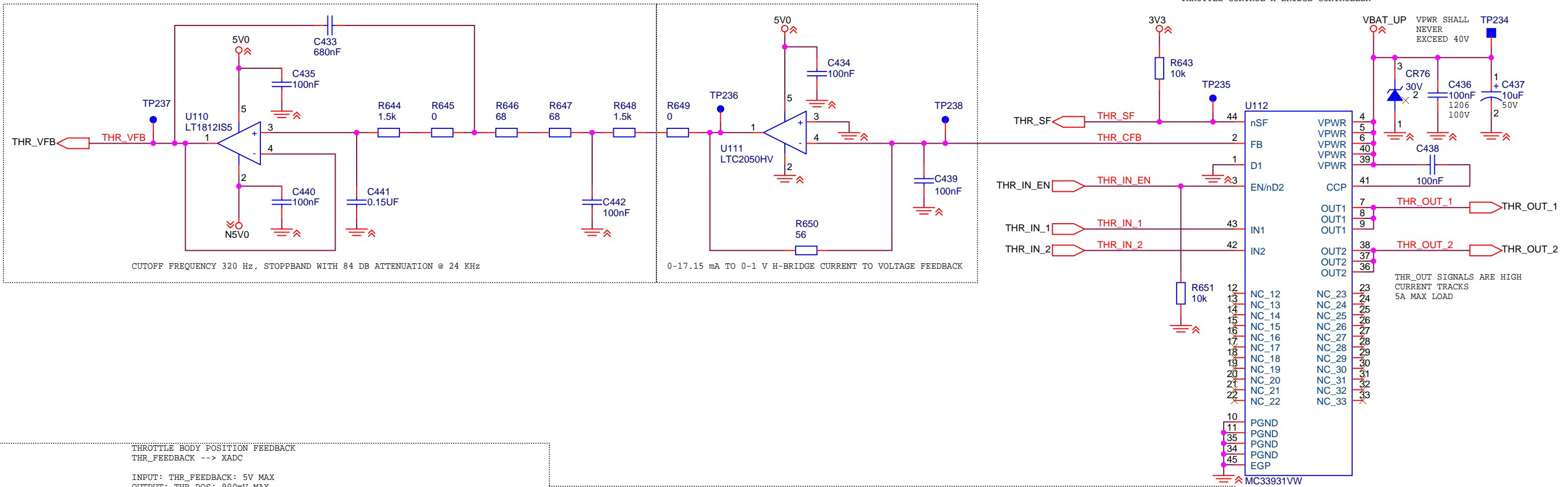
D



SIZE	NSCM NO.	DOC. NO	REV.
A3	-	ELSC60249321-00	-
REF.	REPL.	WEIGHT	SHEET 34 OF 41

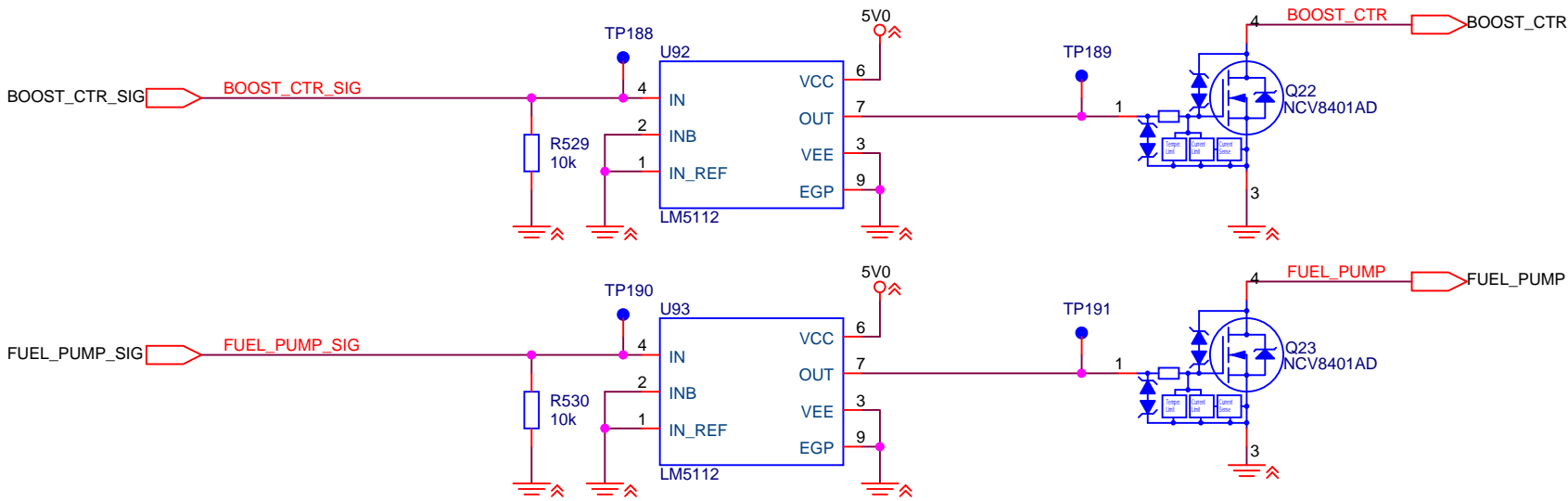


/DRIVER\_CIRCUITRY/THROTTLE\_CONTROL

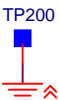
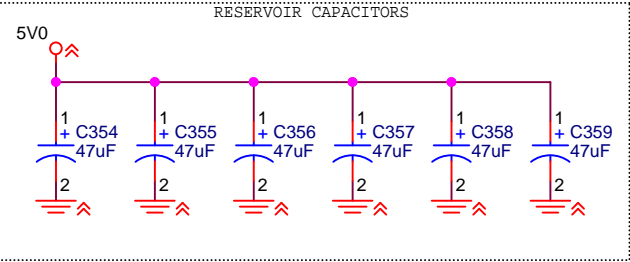
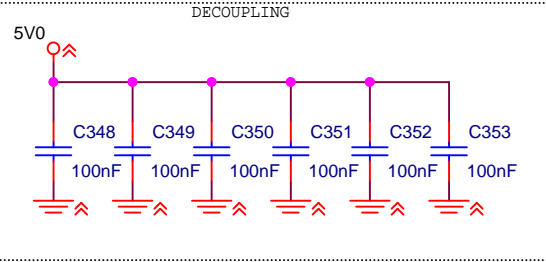
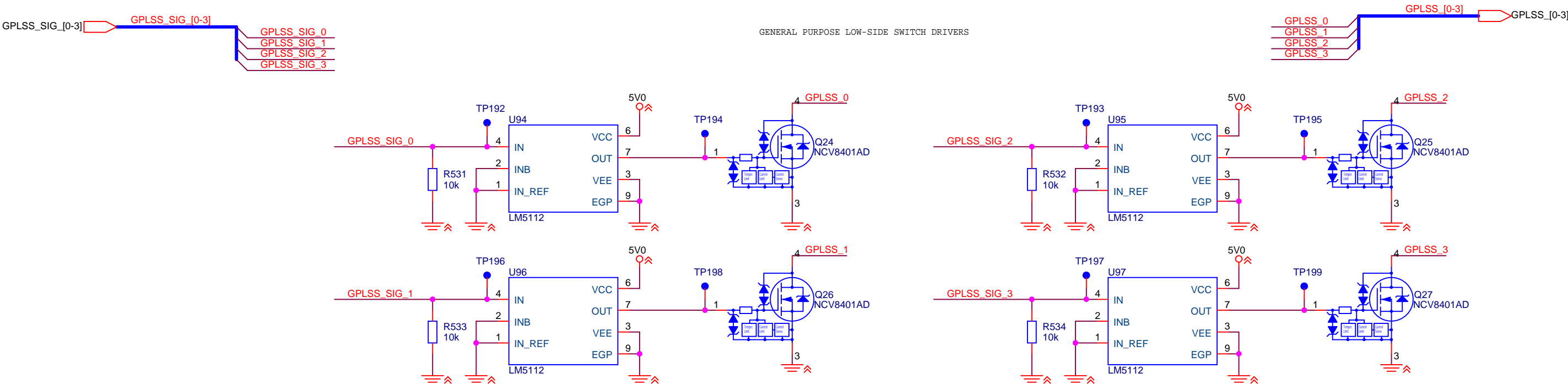


SIZE	NSCM NO.	DOC. NO.	REV.
A3	-	ELSC60249321-00	-
REF.	REPL.	WEIGHT	SHEET 35 OF 41

BOOST CONTROLLER AND FUEL PUMP LOW-SIDE DRIVERS

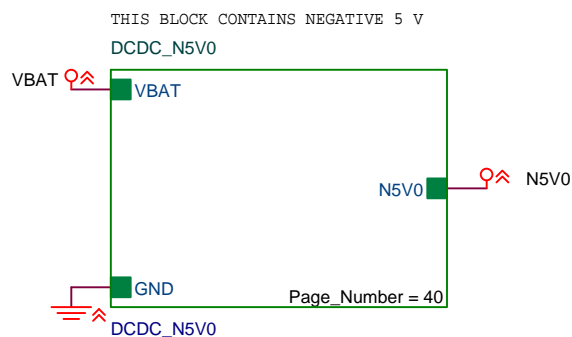
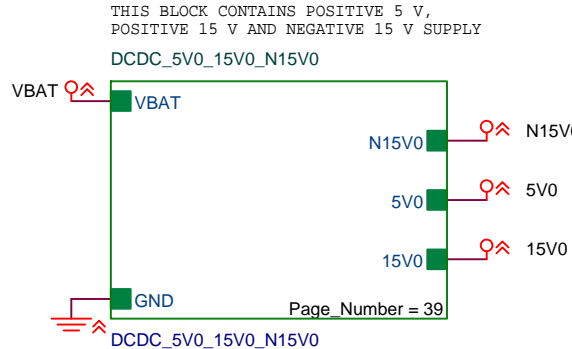
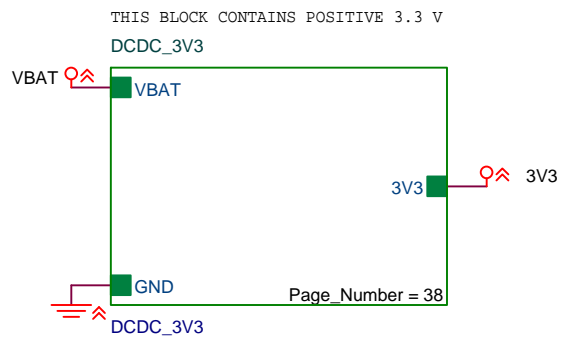
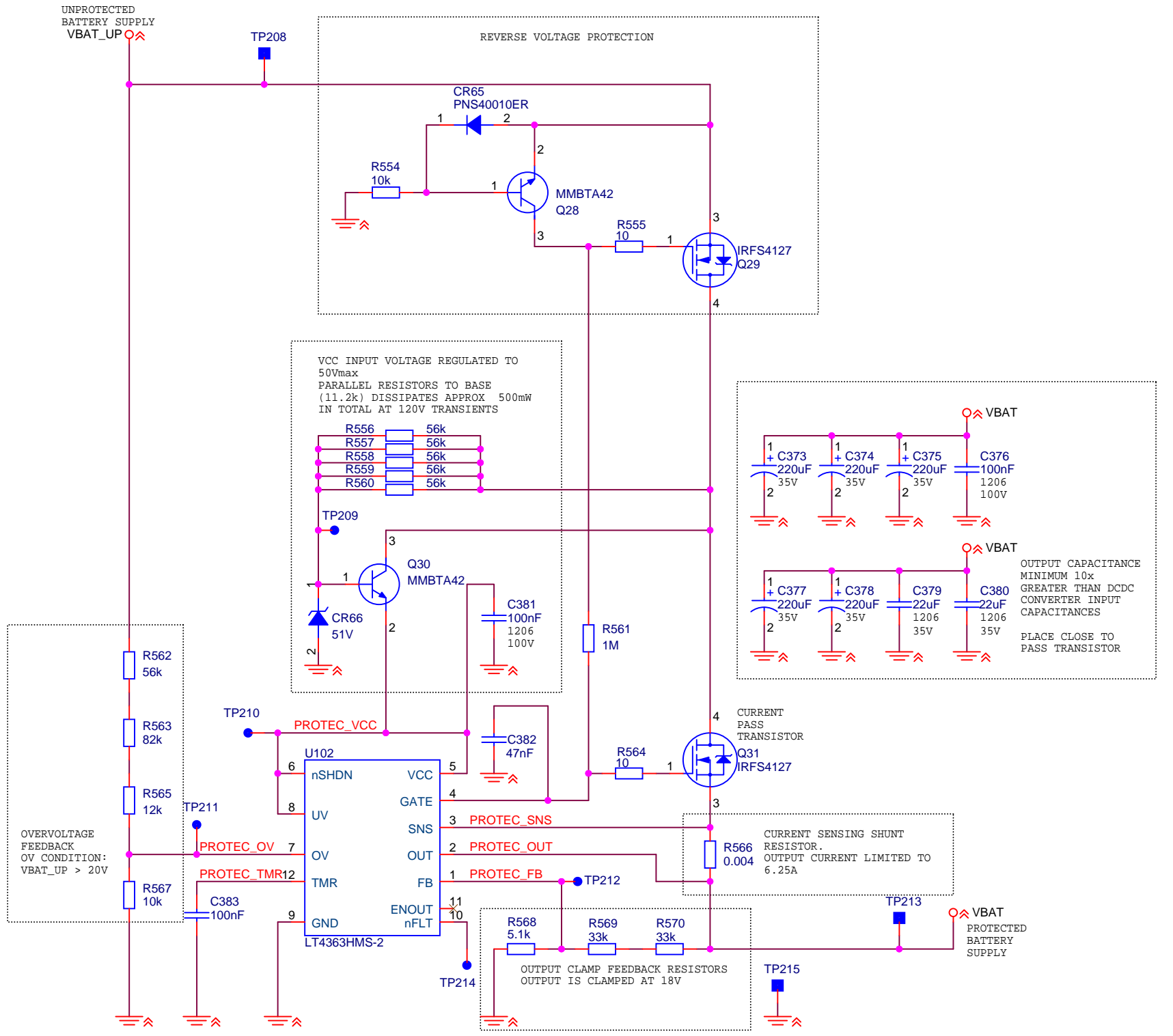


GENERAL PURPOSE LOW-SIDE SWITCH DRIVERS



SIZE	NSCM NO.	DOC. NO	REV.
A3	-	ELSC60249321-00	-
REF.	REPL.	WEIGHT	SHEET 36 OF 41

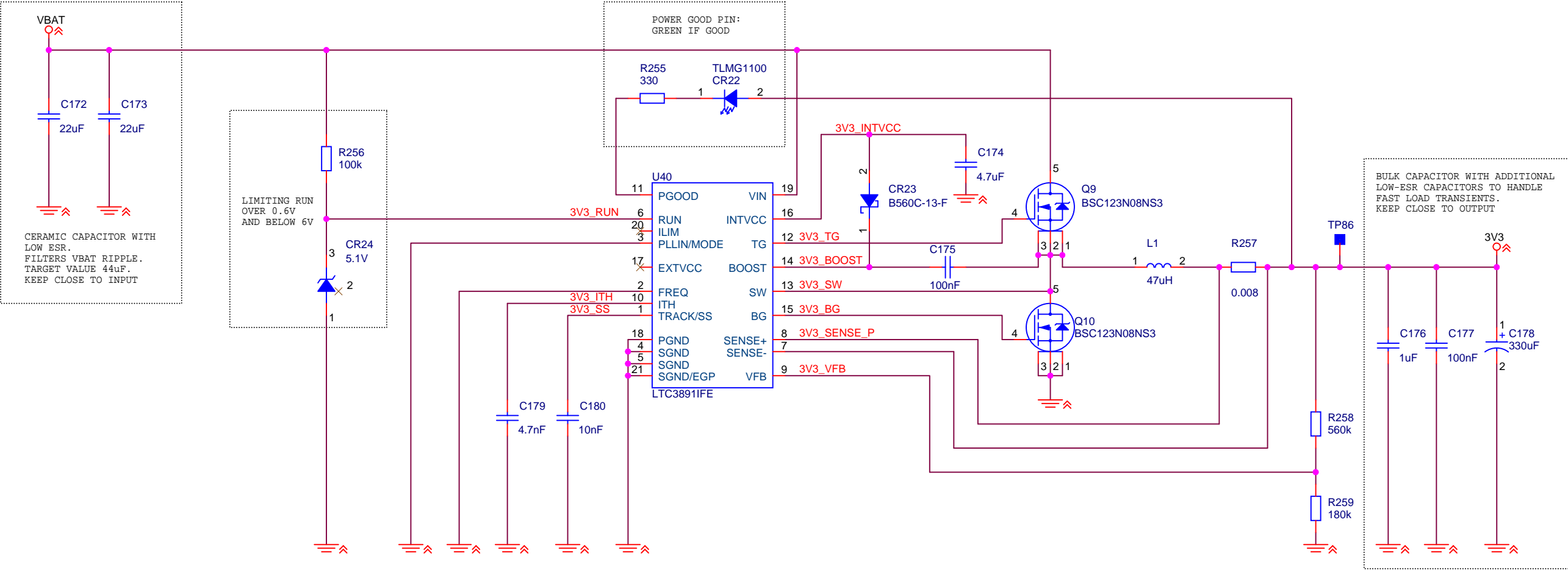
VBAT PROTECTION CIRCUITRY  
PROTECTS ECU POWER SUPPLIES AGAINST:  
- REVERSE POWER INPUT POLARITY  
- LOAD DUMP TRANSIENTS OF UP TO 150V  
- OVERCURRENT/INTERNAL SHORT CIRCUIT



SIZE A3	NSCM NO. -	DOC. NO ELSC60249321-00	REV. -
REF.	REPL.	WEIGHT	SHEET 37 OF 41

/POWER\_MANAGEMENT/DCDC\_3V3

VOLTAGE: 3.3V  
SWITCHING FREQUENCY: 350 kHz  
CURRENT MAX: 4.5 A  
INPUT VOLTAGE: 4-18 V

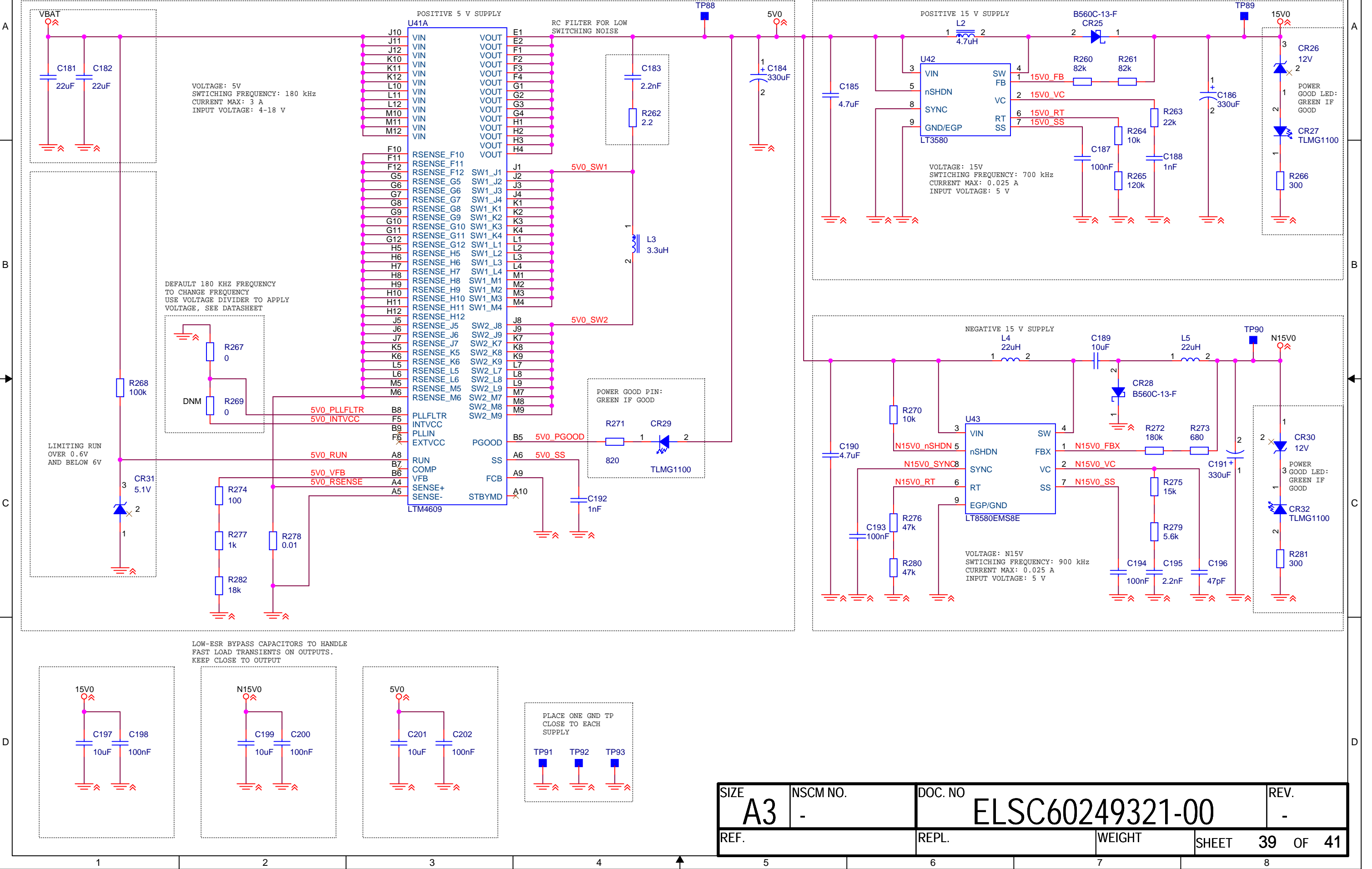


TP87

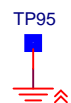
SIZE	NSCM NO.	DOC. NO.	REV.
A3	-	ELSC60249321-00	-
REF.	REPL.	WEIGHT	SHEET 38 OF 41

/POWER\_MANAGEMENT/DCDC\_5V0\_15V0\_N15V0

CERAMIC CAPACITOR WITH LOW ESR.  
FILTERS VBAT RIPPLe.  
TARGET VALUE 44uF.  
KEEP CLOSE TO INPUT SUPPLY



SIZE	NSCM NO.	DOC. NO	REV.
A3	-	ELSC60249321-00	-
REF.	REPL.	WEIGHT	SHEET 39 OF 41



/ECU\_CONNECTOR

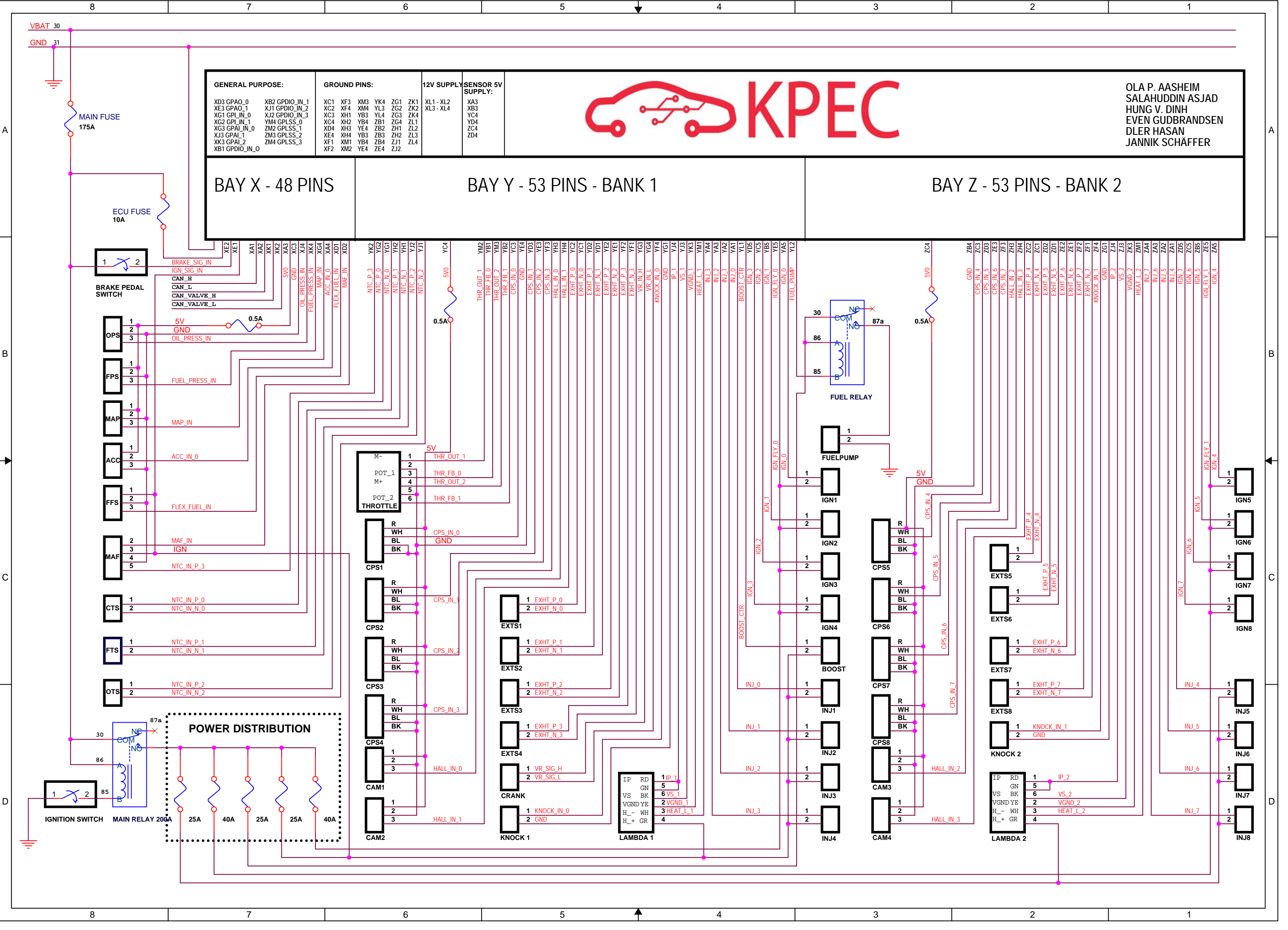
REFER TO KPEC DESIGN DOCUMENT FOR PINOUT AND  
CONVERSION TABLES FOR MORE INFORMATION

CONNECTION OF SENSORS AND ACTUATORS TOWARDS  
ZYNQ MODULE MUST BE PERFORMED BY QUALIFIED  
PERSONELL ONLY.



SIZE	NSCM NO.	DOC. NO	REV.
A3	-	ELSC60249321-00	-
REF.	REPL.	WEIGHT	SHEET 41 OF 41





**GENERAL PURPOSE:**

XD3 GPAO\_0  
XE3 GPAO\_1  
XG1 GPI\_IN\_0  
XG2 GPI\_IN\_1  
XG3 GPAI\_IN\_0  
XK3 GPAI\_1  
XB1 GPDIO\_IN\_0

**GROUND PINS:**

XC1 XF3  
XC2 XF4  
XC3 XH1  
XC4 XH2  
XD4 XH3  
XF1 XM1  
XF2 XM2

XM3 YK4  
XM4 YL3  
YB3 YL4  
YB4 ZB1  
YB3 ZB2  
YB4 ZB3  
YB4 ZB4  
YE4 ZJ2

**12V SUPPLY:**

XL1 - XL2  
XL3 - XL4

**SENSOR 5V SUPPLY:**

XA3  
XB3  
YC4  
YD4  
ZC4  
ZD4

OLA P. AASHEIM  
SALAHUDDIN ASJAD  
HUNG V. DINH  
EVEN GUDBRANDSEN  
DLER HASAN  
JANNIK SCHÄFFER

BAY X - 48 PINS

BAY Y - 53 PINS - BANK 1

BAY Z - 53 PINS - BANK 2