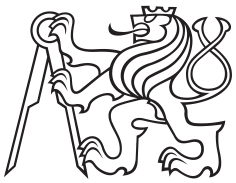
Master Thesis



Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Control Engineering

Simulator of TSI 1.5 combustion engine

Bc. Jan Vojnar

Supervisor: Ing. Michal Sojka, Ph.D.
August 2023

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Vojnar Jan** Personal ID number: **484305**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Control Engineering**

Study program: **Cybernetics and Robotics**

Branch of study: **Cybernetics and Robotics**

II. Master’s thesis details

Master’s thesis title in English:

Simulator of TSI 1.5 combustion engine

Master’s thesis title in Czech:

Simulátor spalovacího motoru TSI 1.5

Guidelines:

TSI 1.5 combustion engine simulator
1. Learn about the function of passenger car internal combustion engines and the previous project of the TSI 1.4 (EA211 evo family.) petrol engine simulator.
2. Design the simulator hardware for the newer TSI 1.5 engine. This, unlike the TSI 1.4 engine, uses Active Cylinder Technology (ACT), which needs to be simulated, and also differs in the connection of some sensors, which now use the SENT digital bus. The simulator will be connected to a Bosch engine control unit. Base your solution on a Xilinx SoC platform (CPU + FPGA).
3. Implement the basic software to run the HW simulator. The SW will be controlled via a GUI ideally implemented in Vector Canoe environment that will communicate with the simulator via a serial link protocol (USB).
4. Perform functionality testing on the CIIRC breadboard with the electronics of the whole car and also on the breadboard at Škoda Auto. Try to eliminate as many errors reported by the engine control unit as possible so that the vehicle electronics do not signal serious errors.
5. Document the results thoroughly and describe in detail the problems found and not solved.

Bibliography / sources:

[1] T. Procházka, “Manuál pro zážehový motor”, technical report, ČVUT, 2019.
[2] Škoda Auto, “1.5 l TSI 110 kW ACT - four-cylinder petrol engine of the EA211 evo line”, Self-Study Programme, 2018
[3] After Sales Training, “SSP169 - 1.5 L TSI ENGINE”, 2017, SEAT S.A.

Name and workplace of master’s thesis supervisor:

Ing. Michal Sojka, Ph.D. Embedded Systems CIIRC

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **14.02.2023** Deadline for master's thesis submission: **14.08.2023**

Assignment valid until:
by the end of summer semester 2023/2024

Ing. Michal Sojka, Ph.D.	prof. Ing. Michael Šebek, DrSc.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Ing. Michal Sojka, Ph.D., for his guidance and support. He has been very helpful and patient in providing valuable feedback and suggestions.

I would also like to thank my friend, Ing. Jaroslav Klapálek, for his continuous support and friendship. He has been a great source of motivation for me.

Lastly, I would like to thank my family for their love and care. They have always been there for me, especially during the challenging times.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of the information used within it in accordance with the Methodical Instructions for Observing the Ethical Principles in the Preparation of University Theses.

In Prague, August 14, 2023

Abstract

This diploma thesis aims to create a signal simulator of the engine sensors and actuators for Engine Control Unit (ECU) testing. The simulator consists of hardware and software components that simulate electronic signals of the 1.5 L 110 kW ACT petrol engine from the EA211 EVO family developed by Volkswagen Group. We designed the hardware part as a modular development board consisting of the circuits based on the measurements of the sensors and actuator signals, including the Active Cylinder Management (ACT) simulation circuit that mimics the ACT used in the engine. The results show that the simulator can generate accurate simulations of the ECU signals. The final version of the simulator will be used by Škoda Auto a.s. to test the ECU units of the EA211 EVO engines. This hardware is an important contribution to simplifying and reducing vehicle testing costs. The simulator has the potential to replace or complement the physical engine simulators for testing vehicle electronics during automotive development or to work as the engine simulator for virtual car simulation development.

Keywords: Engine, Schematic, PCB, Simulation, FPGA, Hardware

Supervisor: Ing. Michal Sojka, Ph.D.
CIIRC, CTU in Prague,
Jugoslávských partyzánů 1580/3,
160 00 Prague 6

Abstrakt

Tato diplomová práce si klade za cíl vytvořit simulátor signálů senzorů a akčních členů motoru pro testování řídící jednotky motoru (ECU). Simulátor se skládá z hardwarových a softwarových komponent, které simulují elektronické signály 1,5 L 110 kW ACT benzinového motoru z rodiny EA211 EVO vyvinuté skupinou Volkswagen Group. Hardwarovou část jsme navrhli jako modulární vývojovou desku sestávající se z elektrických obvodů založených na měřeních signálů senzorů a akčních členů na fyzickém simulátoru motoru. Součástí je i simulační obvod aktivního řízení válců (ACT), který napodobuje ACT používané v motoru. Výsledky ukazují, že simulátor dokáže generovat přesné simulace signálů ECU. Finální verze simulátoru bude použita společností Škoda Auto a.s. k testování jednotek ECU motorů řady EA211 EVO. Tento hardware je důležitým přínosem pro zjednodušení a snížení nákladů na testování vozidel. Simulátor má potenciál nahradit nebo doplnit fyzické simulátory motorů pro testování elektroniky během automobilového vývoje nebo pracovat jako simulátor motoru pro vývoj simulace virtuálního vozidla.

Klíčová slova: Motor, Schéma, DPS, Simulace, FPGA, Hardware

Překlad názvu: Simulátor spalovacího motoru TSI 1.5

Contents

Figures

Tables

Chapter 1

Introduction

The automotive industry constantly develops more advanced and efficient vehicles to meet customers' demands and environmental regulations. One of the significant components of a vehicle is the engine, which provides the power and performance of the car. The engine is controlled by an Electronic Control Unit (ECU), which regulates various parameters, such as fuel injection, ignition timing, air intake, and exhaust emission, based on measurements of sensors mounted to the engine. The ECU is a complex device that needs to be calibrated and validated for different operating conditions and scenarios.

To test the ECU, it is necessary to have a representation of the engine and its sensors' behavior and response. This can be achieved using a physical engine simulator, a hardware device that mimics the engine signals and communicates with the ECU. The physical engine simulator can verify the functionality and reliability of the ECU without requiring an actual engine or a vehicle.

However, developing a physical engine simulator is a challenging and expensive task that requires sophisticated hardware and software components and usually accommodates real parts of the tested car. Such a physical simulator is generally built into a rack tower. Moreover, the physical engine simulator may be unable to capture all possible variations and uncertainties of the engine operation. Therefore, there is a need for an alternative approach that can provide a flexible simulation of the engine signals for testing the ECU.

This thesis aims to design and develop a low-cost simulator of electrical signals for testing the ECU of a newly developed 1.5 L 110 kW ACT petrol engine of the EA211 EVO family. It is a new engine generation that offers improved fuel efficiency, lower emissions, and higher performance compared to the previous generation EA211. The major goals of the diploma thesis are:

- Familiarize with the principle of the combustion engine and 1.5 L 110 kW ACT petrol engine from the EA211 EVO family.
- Design a schematic and a printed circuit of the hardware simulator of the 1.5 L 110 kW ACT petrol engine.
- Create basic software to run the hardware and test the proper function of the designed circuits.

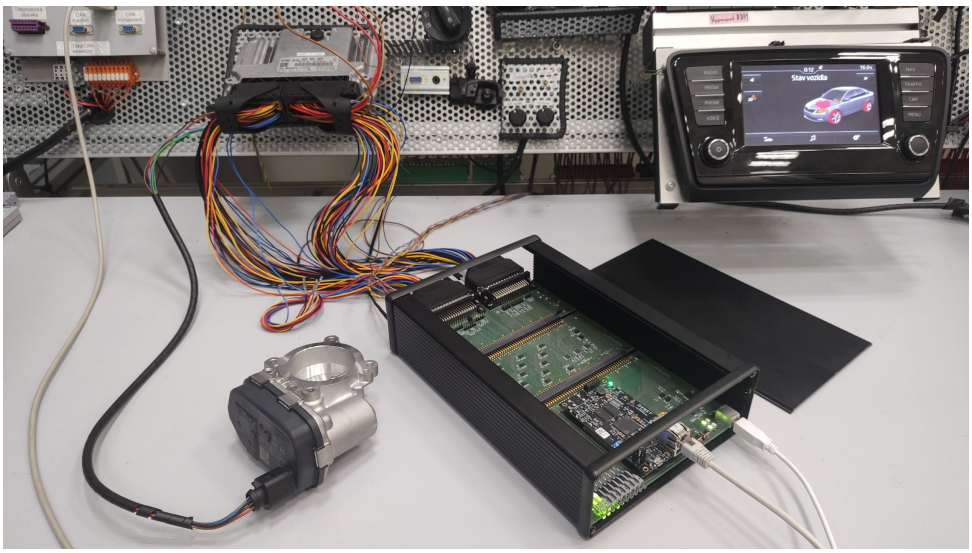


Figure 1.1: The photo of the developed simulator connected to the ECU and the Škoda breadboard. The simulator enclosure is disassembled to show the 50-pin D-Sub connectors.

The electrical signals simulator can potentially replace or complement the physical engine simulator for testing vehicle electronics during development. The output of this work will be used by Škoda Auto a.s., a leading car manufacturer in the Czech Republic. This work is an important contribution to simplifying and reducing the testing costs of vehicles.

This thesis builds on the results of the work that is described in a manual [1] created by Ing. Tomáš Procházka, which focuses on building a predecessor hardware of the this work – a simulator for a 1.4 L petrol engine from the EA211 engine family and its ECU developed by the Bosch Group. Most of the knowledge about basic engine sensors can be used on 1.5 L petrol engines too, but the newer petrol engine accommodates modern sensors with digital interfaces instead of analog ones. Therefore, the hardware of the previous simulator can't be reused. The ECU for the 1.5 L petrol engine is more complex with more safety features. The new ECU checks the behavior of the motor sensors and actuators more tightly, so an accurate simulation of most electrical signals is needed.

The structure of the thesis is as follows. Chapter 2 provides a theoretical background on petrol engines, standard sensor and control unit interfaces, and EA211 EVO engine sensors characteristics needed to design the simulator hardware. Chapter 3 describes developing and designing a simulator hardware interface between the electrical signals simulation system and ECU, including signal conditioning, communication interface, and power supply. Chapter 4 describes the basic implementation of the simulator firmware, including signal processing and user interface. Chapter 5 evaluates the functionality of the presented hardware and firmware connected to an actual ECU. The thesis concludes in Chapter 6 with a summary of the main results and directions for future work.

Chapter 2
Background

Petrol engines are one of the most common types of internal combustion engines that power various vehicles and machines nowadays. They work by burning a mixture of air and petrol in the cylinders, creating pressure that moves the pistons and turns the crankshaft. Various sensors and communication interfaces are used to control and optimize the performance of petrol engines.

In this chapter, we explore the inner workings of petrol engines, focusing on the 1.5 L 110 kW ACT petrol engine from the EA211 EVO family developed by Volkswagen Group. We also discuss the several types of automotive communication interfaces, such as basic on/off signal, Pulse-Width Modulation (PWM), Controller Area Network (CAN), Local Interconnect Network (LIN), and Single Edge Nibble Transmission (SENT), and how they are used to transmit data between the engine and other components. I also describe the sensors of the 1.5 L 110 kW ACT petrol engine from the EA211 EVO family, such as the temperature, pressure, oxygen, knock, and camshaft position sensors, and how they measure various parameters and provide feedback to the engine management system.

In the final section of this chapter, I describe the hardware components and software frameworks that I used to design the hardware of the simulator and run the software of the simulator. The hardware design is based on FPGA+ARM Zynq-7000 System on Chip (SoC) series from Xilinx. I specifically use a system on module (SoM) MicroZed from Avnet, which is based on Zynq-7000 SoC. The FPGA configuration is created using Vivado 2018.3 by Xilinx. On the ARM CPU is running Linux distribution by Xilinx, which is booted up with u-boot bootloader and prepared with Buildroot framework. The simulator hardware is designed using KiCAD 7.0 environment.

2.1 Petrol Engines

Petrol engines are a type of internal combustion engines that use petrol as their fuel to generate mechanical power. They operate on the principle of converting the chemical energy of petrol and air mixture into kinetic energy in a combustion chamber. The burning of the mixture results in the expansion of high-pressure gas, which pushes a piston connected to the crankshaft to

produce rotary motion. The crankshaft then drives the car's wheels or other devices that use the engine's power.

There are various types of petrol engines, but the most used type is the four-stroke engine, which was designed by Nicolaus Otto in 1861. A four-stroke engine has four phases of operation: intake, compression, power, and exhaust. More details of the engine operation and its theory are described in [2].

1. During the intake stroke, the piston moves down and sucks a mixture of air and petrol into the cylinder.
2. During the compression stroke, the piston moves up and compresses the mixture, increasing its pressure and temperature.
3. During the power stroke, a spark plug ignites the mixture with electricity discharge, causing an explosion. The expanding high-pressure gas from the explosion pushes the piston down and generates power.
4. During the exhaust stroke, the piston moves up and expels the burnt gases from the cylinder.

One of the significant challenges of petrol engines is to achieve high efficiency and low emissions while maintaining satisfactory performance. To address this challenge, various technologies have been developed and implemented in the modern petrol engines, such as turbocharging, direct injection, variable valve timing, or Miller cycle.

2.1.1 EA211 EVO Engine Family

One of the most modern petrol engines that incorporate these technologies is from the EA211 EVO engine family developed by the Volkswagen Group. The EA211 EVO engine family is a successor of the EA211 engine family, which was introduced in 2011. The EA211 EVO family consists of inline-three or inline-four petrol engines with variable valve timing and direct injection. They also feature a turbocharger with variable turbine geometry or wastegate, an intercooler for the charged air, and a cylinder deactivation system called Active Cylinder Technology (ACT) for reducing fuel consumption [3].

The EA211 EVO engine family includes a 1.5 L four-cylinder engine that has two variants: a 96 kW version and a 110 kW version. The 96 kW version uses variable turbine geometry and Miller cycle combustion. The 110 kW version uses turbocharging with wastegate and conventional combustion. Only the 110 kW version of the motor is simulated by the presented electrical signal simulator and discussed in more detail later in this chapter.

The 110 kW variant of the 1.5 L EA211 EVO engine has an aluminum alloy cylinder block with plasma-coated liners. The bore diameter is 74.5 mm, and the stroke is 85.7 mm long. The resulting displacement is 1498 cm^3. The compression ratio is 10.5:1; other parameters can be found in [4, p. 9]. The cylinder head is made of aluminum alloy and has an integrated exhaust

manifold. The engine has four valves per cylinder, two intakes, and two exhausts. The variable valve timing is controlled by hydraulic camshaft adjusters on both cams.

The fuel system consists of a high-pressure fuel pump driven by an intake camshaft. The maximal pressure generated by a high-pressure pump is 350 bar, according to [4, p. 23].

The engine features an electrically controlled water pump with two ball valves. The cylinder head cooling circuit and the block cooling are separate. Two ball valves are open, closed, or partially open depending on the five modes of the water pump. The water pump modes are described in detail in [4, p. 24].

2.2 Automotive Communication Interfaces

Automotive communication interfaces are the methods of communication between various electronic components and systems in a vehicle. They enable the exchange of data, signals, and commands among various controllers, sensors, actuators, or displays. Automotive interfaces can be classified into two main categories: analog and digital.

In analog interfaces, continuous voltage or current levels represent information, while digital interfaces use discrete binary values (0 or 1) to encode information. The engine control unit of 1.5 L 110 kW EA211 EVO engine uses analog interfaces such as on/off switching, which uses simple high or low voltage levels to indicate or to control the state of the device, and Pulse Width Modulation (PWM), which uses variable duty-cycles of pulses to represent information.

From digital interfaces, it uses a Controller Area Network (CAN), Local Interconnect Network (LIN), and Single Edge Nibble Transmission (SENT), which use different physical layers and protocols to transmit data packets over serial buses.

Each interface has its advantages and disadvantages regarding speed, reliability, cost, and complexity. In this section, we discuss the characteristics of each interface in more detail.

2.2.1 Analog

The simplest form of an analog signal is a continuous-time continuous-value signal, for example, a signal produced in thermometer devices in the engine, where the actual measured temperature is represented as a voltage level. This signal is noise-sensitive, so devices using such analog signals usually have slowly changing output signal amplitudes. The voltage noise in the system can mask any fast-changing information.

On/Off Switching. A special case of an analog signal is on/off switching, where the signal can be continuous in time, but its value is classified into two states, high voltage level and low voltage level. The on/off signal is

more robust to noise than continuous value signals but cannot transmit more than two states. For example, this signal type can indicate exceeding some threshold, an actual state of a binary value sensor like a hall sensor, or it can be used to control actuators like fans or valves, where we do not need to control their speed or position.

Pulse Width Modulation (PWM). To transmit more information, we can use a PWM signal. PWM signals are analog signals representing value with a variable duty cycle of pulses. The duty cycle is a ratio of the pulse duration at a high voltage level to the duration at a low voltage level. For example, a 50% duty cycle means that the duration of a high voltage level is half of the signal period, and the duration of a low voltage pulse is the other half of the period [5]. This signal is more robust to the noise but requires more complex circuitry and processing than simple signals with continuous-value and time or on/off signals. For example, PWM signals are used to control the amount of delivered power to the actuators and control their speed or to transmit the position to an actuator from a sensor.

2.2.2 Controller Area Network (CAN)

One of the communication interfaces used in this work is the CAN bus. The CAN bus is a serial communication bus that allows microcontrollers and other devices to exchange data without a host computer. It is a message-based protocol originally designed for multiplex electrical wiring within automobiles to save on copper, but it has been adopted in various other fields, such as industrial automation, building automation, and medical automation [6]. Currently, the CAN bus is usually used as a class C protocol operating at bus speeds up to 1 Mb/s for real-time communication in powertrain control, but it can also be used as a class B protocol operating at 125 kb/s for communication between electronics in a vehicle body domain [7].

The basic principle of the interface is that it uses a two-wire bus for communication between the microcontrollers and devices. Each message has an identifier and a data payload. The messages are broadcast to all nodes on the network, and each node decides whether to accept or ignore the message based on the message identifier. The identifier also determines the priority of the message, as the CAN bus uses a bitwise arbitration scheme to resolve conflicts when multiple nodes attempt to transmit at the same time. The node with the lowest identifier value wins the arbitration and continues to transmit, while the others stop and wait until the bus is idle again. This ensures that the high-priority messages are delivered with minimal delay [8].

The International Organization for Standardization (ISO) defines the CAN bus standard in ISO 11898. It specifies the electrical characteristics, signaling, timing, error detection and correction, and power management of the CAN bus. The explanation of CAN standard and more information about CAN bus used in the automotive field for vehicle diagnostics can be found in [9].

The CAN bus has several advantages over other communication interfaces, such as simplicity, low cost, high reliability, high immunity to noise and

interference, and data consistency across the network.

2.2.3 Local Interconnect Network (LIN)

Another communication interface used in this work is the LIN bus. The LIN bus is a broadcast serial bus system that is used in the automotive electronics to communicate with sensors and actuators. It has been standardized internationally by ISO since 2016 in the ISO 17987 series. The LIN bus is a class A protocol operating at a maximum bus speed of 19200 baud over a maximum cable length of 40 meters.

The basic principle of the interface is that it uses a single wire for communication between the master and slave devices. The master device sends out a request frame on the bus, and the slave device responds with a response frame.

Due to its single-wire architecture, LIN is more susceptible to electromagnetic interference than two-wire buses like a CAN bus, limiting the data rate and the recommended number of nodes connected to a bus [7].

2.2.4 Single Edge Nibble Transmission (SENT)

The communication interface that mostly replaced the analog interface of modern sensors of EA211 EVO engines is SENT. Unlike the CAN or LIN bus, SENT uses a point-to-point scheme for transmitting a single value from a sensor to a controller. It is intended to allow the transmission of high-resolution data with low system cost. It is a one-way, asynchronous voltage interface that requires three wires: a signal line (low state $< 0.5\,\mathrm{V}$, high state $> 4.1\,\mathrm{V}$), a power supply line ($5\,\mathrm{V}$), and a ground line. SENT uses PWM to encode four bits (one nibble) per symbol [10].

The SENT standard is defined by the SAE International, formerly named the Society of Automotive Engineers, in SAE J2716 standard [11]. The SENT message protocol is usually proprietary, so only the basics are covered here.

The basic unit of time in SENT is called a tick, where a tick can last between 3–90 µs, at the sender's option. Each message is preceded by a calibration pulse with a high period of 56 ticks for framing and calibration of tick length. After the calibration pulse, each nibble is transmitted with a fixed-width low signal, followed by a variable-length high period. The low-signal period is 5 (or more) ticks in length, while the high-signal period can vary, for a total time between the falling edges of 12–27 ticks (representing nibbles in range 0–15) [10].

The SENT protocol has two types of data transport: fast data channel and slow data channel. Fast data channel is transmitted in units of four bits (one nibble) for which the interval between two falling edges of the modulated signal with a constant amplitude voltage is evaluated. A SENT message is 32 bits long and consists of the following components: 24 bits of signal data that represents two measurement channels of three nibbles each (such as pressure and temperature), four bits for CRC error detection, and four bits of status/communication information. Slow data channel messages can carry a

wide range of other information. Slow data channel messages are transmitted serially, one or two bits per fast channel message, encoded in up to two most significant bits of the Status nibble [10, 12].

In terms of reliability, SENT does not have built-in correction mechanisms like a CAN bus. Reliability is achieved only with checksum error detection. However, the SENT protocol has higher-resolution transmission capability with low system cost compared to CAN or LIN communication.

2.3 Parts of 1.5 L EA211 EVO Petrol Engine

One of the main components of a modern engine is the system of sensors and actuators that monitor and control various aspects of its operation. Sensors are devices that measure physical quantities such as temperature, pressure, speed, position, and oxygen levels and convert them into electrical signals using protocols described in the previous section that the ECU can process. Actuators are devices that receive electrical signals from the ECU and perform mechanical actions such as opening or closing valves, adjusting fuel injection, and changing spark timing. This section describes the major sensors and actuators used in the 1.5 L 110 kW ACT EA211 EVO motor.

2.3.1 Mechanics and Timing

The 1.5 L 110 kW ACT EA211 EVO petrol engine is a modern and efficient engine with various technologies and systems to improve its performance, fuel economy, and emissions. One of these systems is the mechanics and timing of the engine, which controls the movement and synchronization of the pistons, valves, and camshafts. In this subsection, I describe the major sensors and actuators that are involved in the mechanics and timing of the engine, including Active Cylinder Management (ACT) actuators.

Engine speed sensor (G28)

An *engine speed sensor* is a device that measures the rotational speed of the crankshaft. The sensor sends a signal to the ECU, which is used to determine the amount of fuel and ignition timing for the engine. The *engine speed sensor* works by using a hall effect sensor to detect the teeth or notches on the crankshaft. As the crankshaft rotates, the sensor generates a pulse for each tooth or notch. The frequency of these pulses corresponds to the engine speed [13, 14].

Camshaft position sensor (G1002, G1003)

The *camshaft position sensors* measure the angular position of the camshaft for engine timing purposes. There are two sensors: the first sensor is placed on the intake camshaft (G1002), and the second one is placed on the exhaust camshaft (G1003). Both seem to be hall sensor based, equipped with a tone

wheel mount on the camshaft with notches that the sensor measures. The sensors send a signal to the ECU, which uses it to determine the camshaft position [13, 14].

Camshaft control valves (N318, N727)

The 1.5 L 110 kW ACT EA211 engine has variable valve timing on both the intake and exhaust sides. On both sides, vane-type variators with hydraulic drive actuated by solenoid valves. [13, 14].

Active Cylinder Management (ACT) actuators (N583, N587, N591, N595)

The 1.5 L 110 kW ACT EA211 engine is equipped with Active Cylinder Management, which is able to disable the second cylinder and third cylinder to reduce fuel consumption. This system closes the intake and exhaust channels on the second and third cylinders while deactivating the fuel injection and ignition for these cylinders [4].

The second and third cylinder is deactivated by adjusting their intake and exhaust valves to zero stroke. Exhaust valves are disconnected first, followed by intake valves. As a result, there is only air without gas in the combustion chamber.

To adjust the stroke of intake and exhaust cams, camshafts are equipped with ACT actuators and movable camholders for the second and third cylinders. The movable camholders can be moved between two positions: normal stroke and zero stroke [4, 13].

ACT actuators are based on mechanisms similar to solenoid actuators. Each ACT consists of two separate actuators: the first is used to move the camholder from a normal stroke to zero strokes, and the second one is used to move it back. The ECU controls the actuators with ground signals while measuring the electromagnetic feedback signals of each actuator coil.

2.3.2 Lubrication

The oil circuit in the 1.5 L 110 kW ACT EA211 EVO engine uses a vane cell oil pump with continuous adjustment. The oil is led from the oil pump to the oil filter, and then to the engine block, the cylinder head, and the cylinder head covers. The ECU monitors the oil pressure, oil temperature, and oil level with multiple sensors that are described in this section.

Oil pressure sensor (G10)

The *oil pressure sensor* is a sensor that measures oil pressure in the engine. It is located on the cylinder head. It uses the SENT interface to communicate with ECU. The measured oil pressure is sent in the fast channel, and the measured oil temperature is sent in the slow channel of the SENT communication. This

sensor helps to monitor the engine condition and optimize the performance and fuel efficiency of the vehicle [13].

Oil level and oil temperature sensor (G266)

The *oil level and oil temperature sensor* is an important component of the EA211 EVO engines as it helps to optimize the performance and durability of the engine. The sensor also warns the driver when the oil level is too low, or the oil temperature is too high, which can damage the engine. The sensor is located in the oil pan and measures both the oil level and the oil temperature of the engine.

The sensor has a thermistor that measures the oil temperature and a float that moves up or down with the oil level. The sensor has only single output and converts these two values into a single signal. The ECU converts this information back into oil level and oil temperature [13, 14].

2.3.3 Cooling

The 1.5 L 110 kW ACT EA211 family engine cooling circuit is divided into two circuits: the main circuit and the secondary circuit. The purpose of the main circuit is to regulate the temperature of the engine components and provide heat to the system. The purpose of the secondary circuit is to regulate of inlet air temperature to 15 °C above the ambient temperature to avoid overheating the turbocharger. Both circuits are fitted with various sensors and actuators. The most important ones are described in this section.

Coolant temperature sensors (G62, G82, G83)

The *coolant temperature sensors* are devices that measure the temperature of the engine coolant and send a signal to the ECU. The ECU uses this information to adjust the fuel injection, ignition timing, cooling fan operation, and other engine functions [13].

The *coolant temperature sensor (G62)* is located on the cylinder head and measures the temperature of the coolant leaving the engine. It is the main sensor that controls the engine performance and emissions. It also sends a signal to the instrument cluster to display the coolant temperature. It is a negative temperature coefficient (NTC) sensor, which means that its resistance decreases as the temperature increases. The signal voltage ranges from 5 V at -40 °C to 0.25 V at 130 °C [13, 14].

The *engine outlet coolant temperature sensor (G82)* measures the temperature of the coolant entering the water pump and cooling fan. It is the NTC sensor and has a similar voltage range and characteristics as the *G62* sensor.

The *radiator outlet coolant temperature sensor (G83)* is located on the radiator outlet and measures the temperature of the coolant returning to the engine. It is used to monitor the efficiency of the radiator and detect any overheating or cooling issues. It also sends a signal to the ECU to activate the high coolant temperature warning light if the coolant temperature exceeds

120 °C. The *G83* sensor is an active linear temperature sensor, which means it has an integrated circuit that converts resistance into linear voltage analog signals. The signal voltage ranges from 0.5 V at -40 °C to 4.5 V at 150 °C.

Engine temperature regulation module (GX33)

The *engine temperature regulation module* is responsible for regulating the engine temperature by controlling the coolant flow through the engine. It is mounted to the coolant temperature regulation module and actuates the rotary piston that opens or closes the inlets and outlets of regulating module in order to distribute the coolant in all the engine components.

2.3.4 Air Inlet

The air inlet circuit is designed to secure the proper flow of air into the engine combustion chambers and optimize the air temperature according to engine requirements. There are a few modules controlled by ECU which are described in this section.

Charge pressure regulating module (GX34)

This module consists of an actuator that opens the wastegate if the pressure is too high. The wastegate regulates the pressure generated by the turbocharger [13]. The charged air from the turbocharger flows through the intercooler, which is designed to cool down the air and minimize the thermal strain on the *throttle valve* and on the *intake manifold pressure sensor*.

Charge pressure sensor (GX26)

The *charge pressure sensor* measures the charge pressure generated by the turbocharger. It is connected to the ECU with a SENT interface, similar to many other pressure sensors in the EA211 EVO engine. The measured pressure is sent in the fast channel, and the measured temperature is sent in the slow channel of the SENT communication.

Throttle valve module (GX3)

The *throttle valve* is one of the major parts regulating the air flowing to the engine combustion chamber. The valve is regulated from ECU based on the *charge pressure sensor* and the *intake manifold pressure sensor*.

Intake manifold pressure sensor (GX9)

The pressure of charged air regulated with the *throttle valve module* is then measured with the *intake manifold pressure sensor*. The sensor communicates with ECU using the SENT interface similarly to the *charge pressure sensor*. Intake air then flows through intake valves into the combustion chamber of cylinders, where it is mixed with fuel.

2.3.5 Fuel System

The fuel system of the EA211 EVO engines consists of two parts: the low-pressure and high-pressure parts. The low-pressure part delivers fuel from the fuel tank to the high-pressure pump, which is driven by the intake camshaft. The high-pressure pump feeds the fuel to the fuel common rail and generates the required injection pressure for the four fuel injectors.

The ECU regulates the fuel pressure and injection timing according to the engine load and speed. The fuel system is designed to achieve optimal combustion efficiency and low emissions.

The summary of each component is discussed in the following sections, for detailed information about the interconnection of the discussed parts, consult [4, 13, 14].

Fuel metering valve (N290)

The *fuel metering valve* is a device that controls the amount of fuel entering the high-pressure pump in the EA211 EVO engines. It is mounted on the high-pressure pump body. The *fuel metering valve* is controlled by the ECU, which sends a PWM signal to the valve based on the engine load and speed. The valve adjusts the opening and closing of the plunger, which regulates the fuel flow to the high-pressure pump.

Fuel pressure sensor (G247)

The *fuel pressure sensor* measures pressure in the common rail and sends it to ECU. It uses analog signals with continuous value and time to encode information [4].

Fuel Injectors (N30, N31, N32, N33)

The *fuel injectors* are electronically controlled devices that can spray a precise dose of fuel. The injectors are mounted on the cylinder head and spray the fuel directly into the combustion chamber. A dose of fuel mixes with the charge air compressed by the air intake components directly in a combustion chamber [4, 13]. The *fuel injectors* are controlled by the ECU using an analog pulse signal.

Activated charcoal filter valve (N80)

The *activated charcoal filter solenoid valve* is a device that allows gas vapors to escape the high-pressure system. The fumes are led into the intake manifold and purged into the engine to prevent fuel vapors from escaping to the atmosphere [4, 13].

2.3.6 Other

The EA211 EVO engine family uses various sensors that are not directly related to the oil, cooling, intake, or fuel systems but are important for the engine's performance and operation. These sensors include the acceleration pedal position sensor, the knock sensor, the gearbox neutral position sensor, and others. These sensors provide signals to the engine management system, which adjusts the engine parameters accordingly. The following subsections describe these sensors.

Accelerator pedal module (GX2)

The *acceleration pedal position sensor* is located on the pedal assembly and measures the angle of the pedal, which corresponds to the driver's demand for power. This device consists of two measurement units, which use an analog signal to encode the angle of the pedal in a complementary way. When the value of one channel increases, the value of the second channel decreases.

Knock sensor (G61)

The *knock sensor* is mounted on the engine block and detects any abnormal combustion (knock) that may occur in the cylinders. It is a piezoelectric vibration sensor, and its output is analog voltage, depending on the level of vibrations.

The EA211 EVO engine family uses a *knock sensor* that is mounted on the engine block near the third cylinder. The ECU monitors the *knock sensor* signal and compares it with a predefined threshold value. If the signal exceeds the threshold, the ECU retards the ignition timing of the affected cylinder by a certain amount, depending on the engine load and speed.

Gearbox neutral position sensor (G701)

The *gearbox neutral position sensor* is installed on the transmission and indicates whether the gearbox is in the neutral position. It uses analog pulses to inform the ECU when the gear is changed.

Lambda probe (GX7, GX10)

Lambda sensors, also known as oxygen sensors, are used to measure the remaining oxygen content in the exhaust gas and provide an electric signal to the engine control unit to adjust the air-fuel ratio [15]. The optimal air-fuel ratio for a gasoline engine is 14.7 kg of air to 1 kg of fuel, which is known as the stoichiometric mixture [16, 17]. The principle of a *lambda sensor* is based on an oxygen comparison measurement. This means that the remaining oxygen content in the exhaust gas is compared with the oxygen content of the ambient air. If the remaining oxygen content in the exhaust gas is high, the sensor generates a low voltage level. On the contrary, when the remaining oxygen content is low, the sensor generates a high voltage level. [15, 17].

2.4 Hardware Components and Software Frameworks

The simulator that I developed consists of two main parts: the hardware design and the software implementation. In this section, I explain the choices and details of the hardware components and software frameworks that I used for both parts.

The hardware design is based on a hybrid architecture that combines a Field-Programmable Gate Array (FPGA) and an Application-Specific Integrated Circuit (ASIC) on a single chip. This chip is from Zynq-7000 System on Chip (SoC) series produced by Xilinx. The FPGA part allows me to program custom logic circuits for the simulator, while the ASIC part provides a dual-core ARM processor for running the software. To simplify the hardware development, I used a System on Module (SoM) called MicroZed from Avnet, which is a small board that contains the Zynq-7000 SoC and other essential components [18].

The FPGA configuration is created using Vivado 2018.3, which is a software tool by Xilinx that enables me to design, synthesize, and program the FPGA. On the ARM processor, I installed a Linux distribution that is customized by Xilinx for the Zynq-7000 SoC. The Linux distribution is booted up with a u-boot bootloader, which is software that loads the Linux kernel and other files from a memory card. To prepare the Linux distribution, I used the Buildroot framework, which is software that automates the process of creating embedded Linux systems.

The simulator hardware is designed using KiCAD 7.0 environment, which is a software tool that allows me to create schematic diagrams and printed circuit boards (PCBs) for the simulator.

2.4.1 MicroZed

One of the key components of the simulator hardware is the System on Module (SoM) which provides the main processing unit and memory for the simulator. The SoM that I chose for this project is MicroZed 7020 SoM Module XC7Z020-1CLG400C revision G from Avnet. This SoM is based on the Zynq-7000 SoC series from Xilinx, which combines a dual-core ARM Cortex-A9 processor and a programmable logic (PL) fabric on a single chip. The MicroZed 7020 SoM Module offers several advantages for the simulator hardware design, such as:

- High-performance and low-power consumption processor that can run Linux and other software applications for the simulator.
- A Large and flexible PL fabric that can be configured to implement custom logic circuits for the simulator, such as sensors, actuators, and communication interfaces.

- A rich set of peripherals and interfaces, such as Ethernet, USB, UART, SD card, QSPI flash, and Pmod connectors, that can be used to connect to other devices and components of the simulator.
- A small form factor and a low cost, making it suitable for embedded applications.
- A carrier board connector typical for all MicroZed boards. For the development of later revisions of the simulator or its production version, we can use any SoM from the MicroZed family.

I decided to use the MicroZed 7020 SoM Module because it was the latest and most advanced Zynq-7000 SoM available on the market at this work's design date. The main parameters of the MicroZed 7020 SoM Module are summarized in the Table 2.1.

Parameter	Value
SoC	XC7Z020-1CLG400C
Processor	Dual-core ARM Cortex-A9 @ 667 MHz
Memory	1 GB DDR3 SDRAM, 128 Mb QSPI flash
Ethernet	10/100/1000 Mbps
USB	USB 2.0 OTG
UART	USB-UART bridge
SD card	Micro SD card slot
User I/O	100 pins (50 per connector)
Pmod	2x6 Digilent Pmod compatible interface
JTAG	Xilinx PC4 JTAG configuration port

Table 2.1: Main parameters of the MicroZed 7020 SoM Module

I could not rely on newer SoMs, like the Kria K26 [19], becoming available later, as they have different specifications and compatibility issues. Moreover, I had previous experience with this MicroZed SoM, as I had already researched the development of carrier boards for this SoM. This helped me to speed up the hardware development process, as I already knew some of the design considerations and challenges involved.

The reason for using the computing module with FPGA is that many signals from the engine need accurate timing and the FPGA design can achieve that. The basic communication interfaces and simulation of the timing engine signals are needed to be accurate in the time domain. The critical signals for the ECU to operate correctly are signals that include crankshaft position, camshaft position, and simulation of wheel sensors. Simulating these signals as a program written in C running in the Linux distribution on ARM CPU was not an option, as it would introduce too much latency and jitter.

To program the FPGA part of the MicroZed 7020 SoM Module, I used Vivado 2018.3, a software tool by Xilinx that enables me to design, synthesize, and program the FPGA. Vivado 2018.3 is compatible with the Zynq-7000 SoC series and supports various features and functionalities for FPGA development.

Xilinx FPGAs and Vivado are among the best solutions on the market, as they offer high performance, flexibility, and reliability for embedded applications. Xilinx also provides excellent documentation and tutorials on using Vivado and FPGA programming, which helped me learn the basics and essential techniques of FPGA development.

MicroZed software

To prepare the Linux distribution for the ARM processor of the MicroZed 7020 SoM Module, I used the Buildroot framework, which is a software that automates the process of creating embedded Linux systems. Buildroot framework allows me to configure and customize the Linux kernel, the bootloader, the root filesystem, and the software packages I want to include in the Linux distribution. Buildroot framework also cross-compiles the Linux kernel and the software packages for the target architecture, which is ARM in this case [20].

Buildroot. I reused the old configuration of the Buildroot framework that was created for the previous simulator of the 1.4 L engine from the EA211 family. However, the configuration was outdated and needed some refactoring. I upgraded the Buildroot framework to the latest stable version that can be considered safe for production use, which was `2022.02.2` at the time of development of this work. This version of the Buildroot framework supports the latest features and bug fixes for embedded Linux development. I followed the Buildroot manual to update and modify the configuration according to my needs, which can be found on its web page [20].

Bootloader. I used u-boot version `xilinx-v2019.1`, a modified version of u-boot by Xilinx that supports the Zynq-7000 SoC series. U-boot is software that loads the Linux kernel and other files from a memory card and boots up the Linux distribution on the ARM processor. U-boot also provides some commands and utilities for testing and debugging the hardware.

Linux kernel. I used Xilinx Linux distribution with version `xilinx-2021.1`, a customized version of Linux by Xilinx that supports the Zynq-7000 SoC series. Xilinx Linux distribution includes some drivers and patches specific to the Zynq-7000 SoC series, such as AXI DMA, AXI GPIO, AXI Ethernet, AXI VDMA, and others.

2.4.2 Hardware Design Tools

To design the schematics and the PCB for the simulator hardware, I used KiCAD 7.0, which is a software tool that allows me to create schematic diagrams and Printed Circuit Boards (PCBs) for the simulator. KiCAD 7.0 is free and open-source software that supports various features and functionalities for Electronic Design Automation (EDA), such as schematic capture, PCB layout, basic 3D viewer, and Gerber file generation. KiCAD 7.0 also has a

large library of components and footprints that I can use for the simulator hardware design. I chose KiCAD 7.0 because it was the only usable software for large-scale projects that I had access to when I started to work on the thesis.

Another suitable and even better software for EDA is Eagle from Autodesk, which is a popular and powerful software that offers more advanced features and capabilities for EDA, such as auto-routing, simulation, and collaboration. However, at that time, Eagle was undergoing a major transition from standalone software to a part of Fusion 360, which is a cloud-based platform by Autodesk that integrates various design tools. This change created some difficulties in getting a license for Eagle, as Autodesk was adapting it for use in Fusion 360. The Fusion 360 had some features of Eagle implemented, but they were not stable and complete at that time.

Low-cost EDA software is not usable for projects with higher part density on PCB or with a lot of signal lines to route as creating schematics or PCB designs with such tools is extremely time-consuming. Advanced EDA software suites like Altium Designer are too expensive to buy for this project, as they cost thousands of dollars per year. Therefore, I decided to use KiCAD 7.0 as the best available option for EDA, even though it is potentially time-consuming to work with.

Chapter 3

Hardware Design

The hardware design of the electrical signal simulator for the 1.5 L 110 kW ACT engine from EA211 EVO family is one of the main goals and contributions of this thesis. In this chapter, I present the schematics and the PCB design of the simulator hardware, which consists of various components and circuits that are responsible for generating and processing the input and output signals of the Engine Control Unit (ECU) and other control units.

The simulator hardware is designated as a development board, which is split into three pieces interconnected with connectors. The development board is based on a System on Module (SoM) called MicroZed 7020 SoM Module XC7Z020-1CLG400C revision G from Avnet, which provides the main processing unit and memory for the simulator.

In the first section, I describe the structure and layout of the development board. The layout allows disconnecting major parts of the boards for development or debugging purposes. The development board consists of three interconnected pieces:

- The first piece is a board for MicroZed, a power supply for MicroZed, a user interface, and a galvanically isolated USB-UART.
- The second piece is a board with all the processing circuits and interfaces needed to receive and transmit signals to and from ECU.
- The third piece is a board with two large 50-pin D-sub connectors for selected signals from ECU.

In the second section, I describe the power sources that are used to supply the simulator hardware. There are three types of power sources: one for the MicroZed SoM, one for the ACT simulation, and optionally fitted linear voltage regulators for the simulator electronics. The MicroZed SoM requires a specific power sequence to power up, which is achieved by using an integrated power supply with power management designed for FPGAs. The ACT simulation requires a high voltage and high current power source, which is provided by using a DC-DC step-up voltage regulator. The optionally fitted power sources are used to power the simulator electronics in case it is not desirable for the simulator electronics to be powered from the MicroZed supply in case of noise or interference.

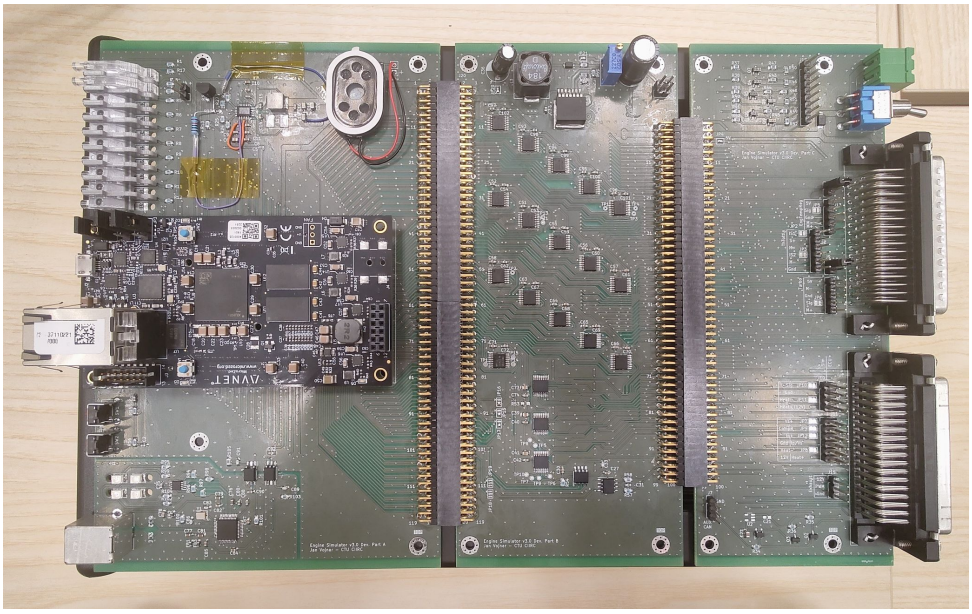


Figure 3.1: The photo shows assembled and tested development board of the simulator. As we can see, it consists of three parts connected with large connectors from the left: the computing part with the MicroZed, the signal processing part, and the connector part. This board does not have fully fitted all components like a part of the MicroZed power supply, LEDs, and LED light guides. The MicroZed power supply is reworked because the 5 V channel of the power supply failed during testing, which is discussed later in Chapter 5.

In the third section, I describe the signal processing that is performed by the simulator hardware. I explain how I analyzed the characteristics of the ECU signals, such as voltage levels, frequency ranges, waveforms, and protocols. Then, I describe each circuit type used in processing these input and output signals of the ECU, such as level translators, Digital-to-Analog Converters (DAC), and other interfaces. I also describe the circuits that are used for simulating other control units that need any simulated information from the ECU, such as the Anti-lock Breaking System (ABS) and Electronic Stability Program (ESP) unit.

3.1 Overall Architecture

The development board is the core of the simulator hardware, as it contains all the components and circuits that are required for generating and processing the input and output signals of the ECU and other control units. A photo of the fitted and tested development board is in Figure 3.1. The development board is designed as a modular system that consists of three pieces that are interconnected with connectors. Each piece has a specific function and role in the simulator hardware:

- The first piece is a board that has MicroZed connectors to connect

MicroZed SoM, a power supply for MicroZed, a user interface with LEDs and a speaker, and a 120-pin connector to output every unused MicroZed pin to the second board. It also has a galvanically isolated USB-UART interface that allows the user to safely communicate with the MicroZed board from a computer, even when the simulator is connected to the 12V auto battery or Hardware-in-the-Loop (HIL) system.

- The second piece is a board that has a 120-pin connector to connect the first board, all processing circuits, analog signal generators, LIN and CAN interfaces that are connected to these MicroZed lines, and a 100-pin connector to carry processed signals from the second board.
- The third piece is a board that has a 100-pin connector to connect the second board and two large 50-pin D-sub connectors for selected signals from ECU. It also has a circuit for encoding a wheel spinning signal, a circuit to decode a single brake signal into two mutually inverted signals, and multiple small connectors to optionally connect engine modules that are difficult to simulate, in case we decide not to simulate them or to use them for debugging and development purposes.

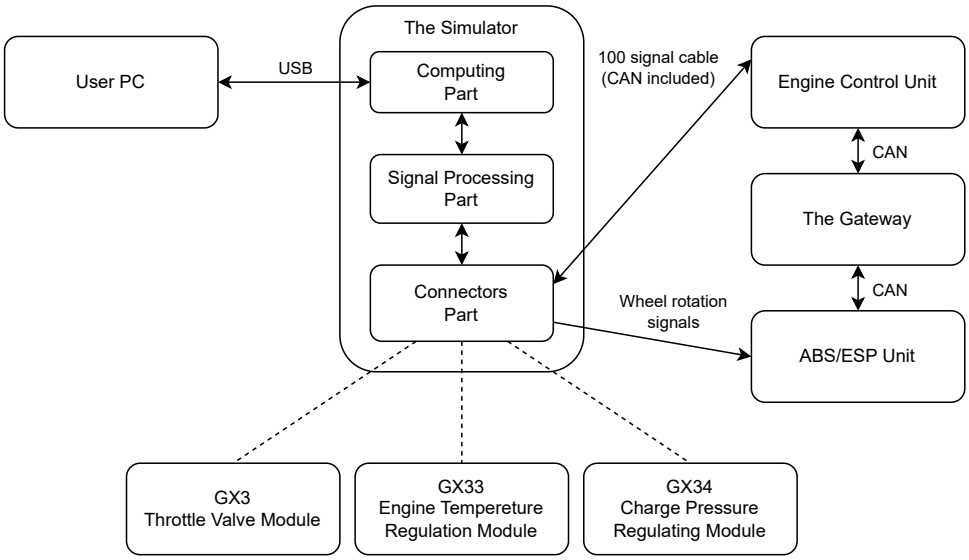


Figure 3.2: Diagram illustrates interconnections between the simulator and car components and the user. The direction of the arrow illustrates the direction of the signals. Solid links illustrate mandatory connections, and dashed links illustrate optional connections.

Figure 3.2 shows an example of how the development board is connected to the car devices. The ECU is connected to the development board via two large 50-pin D-sub connectors that carry selected signals from ECU. The ECU and the gateway unit are connected to the simulator via the CAN interface, which is also connected to the MicroZed board if any interaction is needed. The ABS/ESP unit is connected to the simulator via a circuit for encoding

the wheel sensor signals. These connections allow the simulator to simulate various sensors and actuators for the engine control unit and other systems and communicate with other control units via CAN bus.

3.2 Power Sources

One of the important aspects of the simulator hardware design is the power sources that are used to supply the different components and circuits of the simulator. The power sources have to meet the requirements of the simulator, such as voltage levels, current ratings, efficiency, stability, and safety. This section describes the types and details of the power sources I used for the simulator hardware design. In total, there are five types of them:

1. A switching voltage regulator based on MAX20808 from Analog Devices. This regulator is used as the power supply for the MicroZed SoM, which requires a specific power sequence to power up. This regulator has features needed to implement the proper power sequence to start up MicroZed.
2. A switching voltage regulator based on LM2586SX-ADJ from Texas Instruments. This regulator is used as the voltage source for the ACT feedback pulses, which require a high voltage and high current to simulate the cylinder deactivation status. The regulator can boost the input voltage from 12 V to 24 V with a maximum output current of 3 A [21]. In this design, the regulator output voltage is adjustable from 16 V to roughly 20 V.
3. An optional 3.3 V automotive grade linear regulator based on TPS7B8833 from Texas Instruments. This regulator can power the processing part of the simulator electronics. The regulator can provide a stable and low-noise 3.3 V output with a maximum output current of 500 mA. The regulator also has features that ensure its safety and robustness, such as overcurrent protection, overvoltage protection, and thermal shutdown [22].
4. An optional 5 V automotive grade linear regulator based on TPS7B8850 from Texas Instruments. This regulator can power the processing part and, optionally, the connector part of the simulator electronics. The regulator can provide a stable and low-noise 5 V output with a maximum output current of 500 mA [22]. The regulator also has the same features as the previous TPS7B8833.
5. A small linear regulator based on LP2981-5.0 from Texas Instruments. This regulator provides an optional power supply for the brake decoder circuit of the connector part. The regulator can provide a stable 5 V output with a maximum output current of 100 mA. The regulator also has features that improve its performance and reliability, such as low dropout voltage, low quiescent current, overcurrent protection, and thermal shutdown [23].

3.2.1 MicroZed Power Supply

The MicroZed SoM requires a soft-start power supply and a proper power sequence to start up, as specified in the MicroZed Carrier Design Guide [24] and the MicroZed Hardware User Guide. The power architecture of MicroZed and its carrier board is illustrated in Figure 3.3. The power-up sequence consists of the following steps:

1. 5 V power supply for the MicroZed SoM must be applied and stabilized.
2. The `PWR_EN` signal must be asserted high to enable power management of the MicroZed SoM, USB, Ethernet, and Pmod peripherals.
3. After the 1.8 V of MicroZed SoM is stable, the `VCCIO_EN` is asserted high. That enables the I/O bank voltage regulator of the carrier board. It is the 3.3 V power supply in the case of this design.
4. After all I/O bank power supplies are stable, the `PG_MODULE` must be asserted high, and MicroZed is ready to boot up.

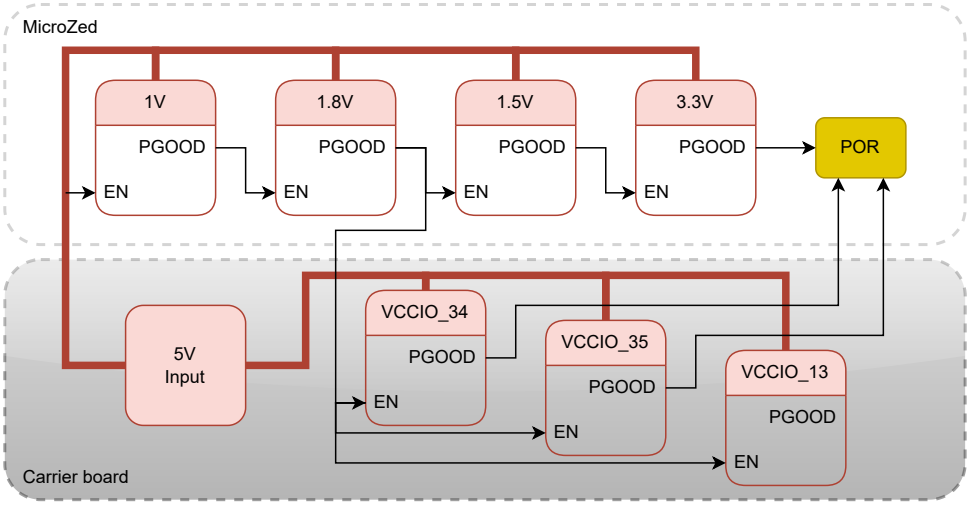


Figure 3.3: Diagram illustrates the power source architecture of the MicroZed SoM and carrier board. We can see the sequence of power regulators that leads to activating the Power-On Reset (POR) function of the MicroZed to reset and boot up the board. The `PGOOD` symbol means the power-good output and the `EN` symbol means the power-enable input.

To implement this power sequence for the simulator hardware design, I used a step-down switching voltage regulator based on MAX20808 from Analog Devices. This regulator has two regulator channels that are configured as a 3.3 V source and a 5 V source in this design.

The 5 V source is enabled as soon as 12 V input is present and provides power to the 5 V supply for the MicroZed board. The power-good signal of the 5 V source of MAX20808 is connected to the `PWR_EN` of the MicroZed board, so when 5 V voltage is stable, it enables power supplies of the MicroZed

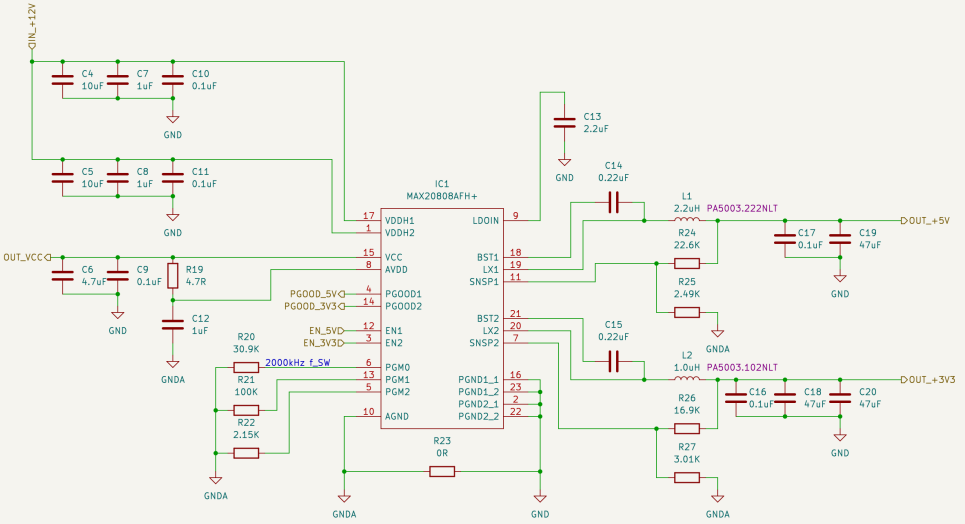
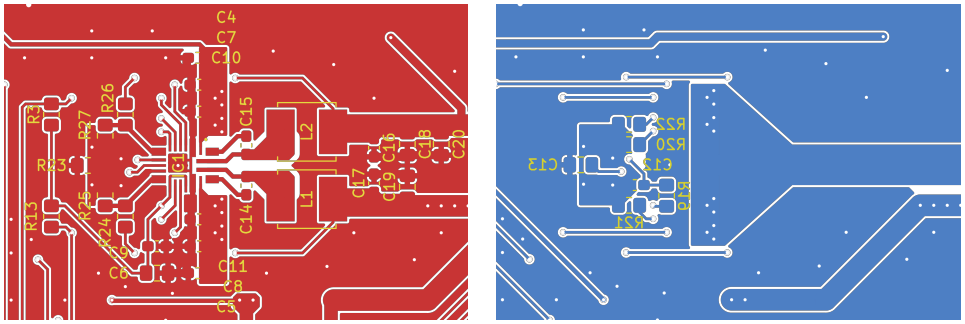


Figure 3.4: The schematic of MAX20808 circuit used in this work.



(a) : Front layer. **(b) :** Back layer.

Figure 3.5: Layout of the MAX20808 circuit (top view).

board. After the 1.8 V power supply of the MicroZed board is stable, it turns on the `VCCIO_EN` signal, which enables IO bank voltage sources of a carrier board. In the case of this design, these sources are provided by the 3.3 V source of MAX20808. When the 3.3 V source is stable, MAX20808 turns on the power-good signal connected to `PG_MODULE`, and the MicroZed board is started up.

The schematic of the power supply design for MicroZed can be seen in Figure 3.4. I used the typical application circuit for dual-output function as the basis of my design. To configure the MAX20808 switch regulator for 5 V and 3.3 V output, I used reference design component values from its datasheet [25] in Table 6 – Reference Design Examples. The layout requirements of the MAX20808 circuit require tight placement of SMD components. To be able to solder them by hand, I designed the layout that can be seen in Figure 3.5.

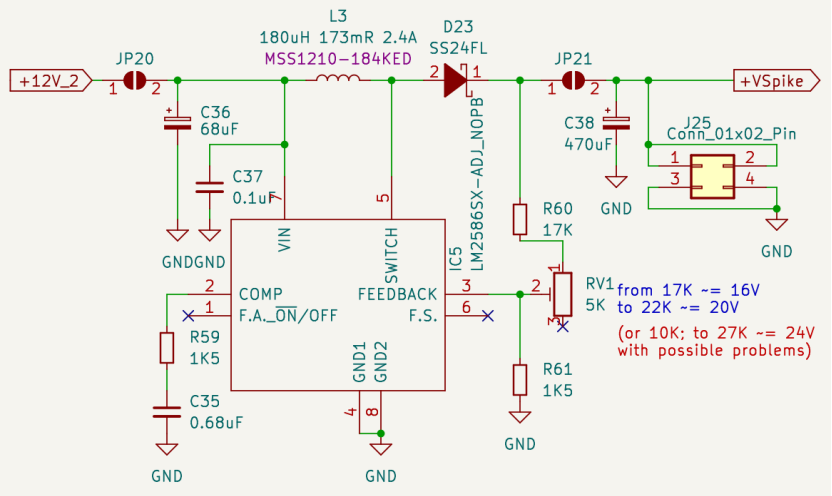


Figure 3.6: The schematic of ACT voltage regulator circuit.

3.2.2 ACT Power Supply

One of the features of the EA211 EVO engine is the Active Cylinder Technology (ACT), which can deactivate and reactivate two of the four cylinders depending on the driving conditions. This feature improves fuel efficiency and reduces the emissions of the engine.

To simulate the ACT feedback pulses, which indicate the cylinder deactivation status to the ECU, I used a step-up switching voltage regulator based on LM2586SX-ADJ from Texas Instruments. This regulator can boost the input voltage from 12 V to 16–18 V with a maximum output current of 3 A. The output voltage can be adjusted by changing the feedback resistor divider ratio using the trimmer potentiometer.

To design this circuit, I used Texas Instruments WEBENCH Power Designer[1], which is an online tool that helps me to create and optimize power supply circuits. WEBENCH Power Designer allows me to enter the input and output specifications, select the components, simulate the performance, and generate the schematic and bill of materials for the circuit if needed. I followed the design guidelines and recommendations provided in the LM2586 datasheet [21] to ensure the proper operation and stability of the circuit. The schematics of the designed ACT voltage regulator can be seen in Figure 3.6.

The circuit is simple and does not require any hard-to-achieve layout considerations. However, I used a tighter layout and a short switching loop to minimize the parasitic inductance and capacitance that could affect the switching performance and efficiency of the circuit. Instead of using a resistor voltage divider from resistors, I added a trimmer potentiometer to adjust the output voltage from 16 V to 18 V. The part values of the circuit were selected with adjustable voltage in mind so that I could fine-tune the output voltage according to the ACT feedback measurements and the ECU response. If the voltage output of the designed circuit will not be enough, I included

[1] `https://webench.ti.com/power-designer/switching-regulator`

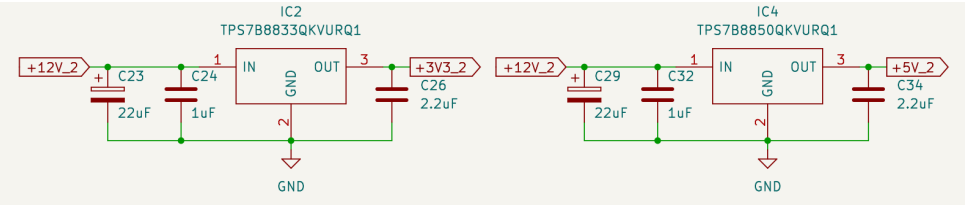


Figure 3.7: The schematic of 3.3 V and 5 V linear voltage regulator on the processing part of the simulator.

solder jumpers on the input and output of the circuit, which are labeled `JP20` and `JP21` in the schematic shown in Figure 3.6. Solder jumpers can be disconnected with a soldering iron to disconnect the ACT voltage regulator from the simulator circuit fully. An additional connector `J25` can be used to connect an alternative power source.

3.2.3 Optional Power Supplies

Besides the power sources for the MicroZed board and the ACT simulation, some other power sources are used for the simulator hardware. These power sources are optional and can be used to power the processing and connector parts of the simulator electronics in case it is not desirable for any reason for the simulator electronics to be powered from the MicroZed power supply. These power sources are based on linear regulators, which regulate the output voltage by dissipating excess power as heat. Linear regulators have advantages over switching regulators, such as lower noise, higher stability, and a more straightforward design. However, linear regulators also have disadvantages, such as lower efficiency, higher heat generation, and higher dropout voltage.

The optional regulators for the processing part of the simulator electronics are TPS7B88 from Texas Instruments with fixed voltage at 3.3 V and 5 V. This regulator can provide a stable and low-noise voltage output with a maximum output current of 500 mA. The regulator also has features that ensure its safety and robustness, such as over-current protection, under-voltage protection, over-voltage protection, and thermal shutdown.

The used circuits are the same as shown in Figure 3.7. The part values are selected according to the regulator's datasheet [22]. It is necessary to disconnect solder jumper `JP13` in case the 5 V regulator is fitted and solder jumper `JP14` in case the 3.3 V regulator is fitted. If one or both regulators are not fitted, the solder jumper belonging to the unfitted regulator must be closed. In a closed state, the power to the voltage line is provided from the MicroZed power source.

The last optional regulator is the linear regulator LP2981-5.0 from Texas Instruments. This regulator provides an optional power supply for the brake decoder circuit of the connector part. The regulator can provide a stable 5 V output with a maximum output current of 100 mA. Its circuit is shown in Figure 3.8. The solder jumper `JP4` must be opened if the regulator is fitted. If the regulator is not fitted, the solder jumper must be closed to provide

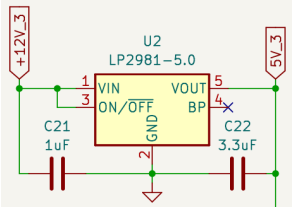


Figure 3.8: The schematic of 5 V linear voltage regulator on the connectors part of the simulator.

a power supply from the processing part of the simulator. In case the 5 V linear regulator of the processing part is fitted, and solder jumper `JP13` is open, then the brake circuit is powered with the optional 5 V regulator of the processing board. Also, if both optional 5 V regulators are not fitted, and `JP4` and `JP13` are closed, the brake circuit is powered from the MicroZed part power supply.

3.3 Signal Processing

One of the main tasks of the simulator hardware is to process the input and output signals of the ECU. These signals are generated by various sensors and actuators in the engine and other systems, representing the car's physical parameters and states. Designing the simulator hardware to generate these signals requires integrating information from all available sources. The initial source of information was the documentation of cars that use EA211 EVO engines, which Škoda engineers provided. However, this source was insufficient to obtain all the details and characteristics of the signals. Therefore, we had to resort to measuring and capturing the signals with an oscilloscope and a logic analyzer. The information obtained from the measurements was the most valuable and useful for the design.

Analyzing the signals generated by sensors, actuators, and the ECU was a complicated and tedious process because the only available device to measure signals on was a professional physical engine simulator in Škoda laboratory in Mladá Boleslav. The documentation of the physical engine simulator in Škoda or technical documentation of the sensors, actuators, and ECU is proprietary, and these resources are utterly inaccessible to us. We made many visits to measure signals where we were unsure of their character or behavior. We also consulted with Škoda engineers and technicians to obtain more information and clarification about the signals. We recorded and analyzed the signals using an oscilloscope and a logic analyzer, and I tried to understand their voltage levels, frequency ranges, waveforms, and protocols. Every simulated or measured signal of the ECU, their code, deduced voltage range character, and implemented simulation circuit type is entered in the table in Appendix B, which was gradually filled in and refined over roughly half a year. The result of signal analysis is much simpler than the analysis process itself. The simulator uses only four different types of circuits to process or generate the necessary

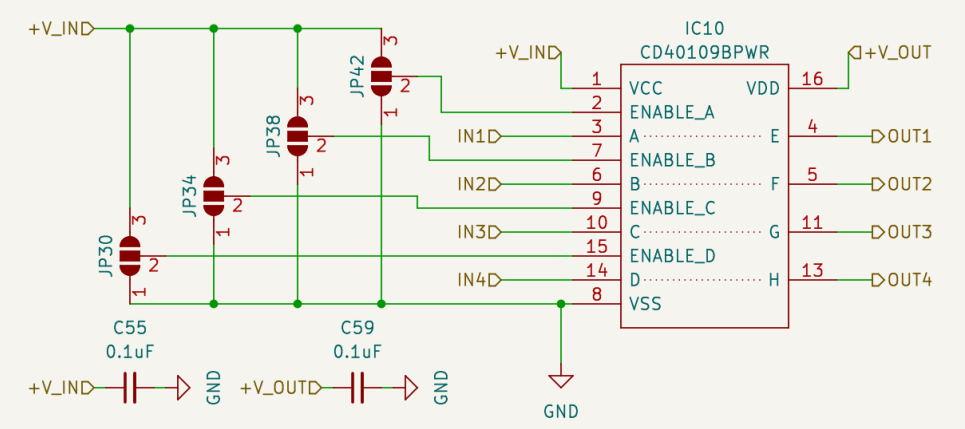


Figure 3.9: The schematic of a level translator circuit. From what voltage to what voltage depends on the circuit it is connected in.

signal. Two of them are protocol signals generated using integrated solutions.

This section describes each circuit type I used to process these signals from and to the ECU. These circuit types are level translators, Digital-to-Analog Converters (DACs) to generate analog signals, a LIN interface circuit, and a CAN bus transceiver circuit.

3.3.1 Signal Level Translator

One of the circuit types that I used to process the signals from and to the ECU is the voltage level translator. It is a device that converts a signal from one voltage level to another voltage level without changing its frequency or waveform. Signal level translators are needed because different devices and components may use different voltage levels for their signals and may not be compatible. For example, the MicroZed board uses 3.3 V for its signals in this design, while the ECU and the engine sensors and actuators use 5 V and 12 V for their signals. If these signals are directly connected, they may cause damage or malfunction to the devices and components.

I used integrated high-speed circuits CD40109B from Texas Instruments to implement voltage-level translators for the simulator hardware. These circuits have four independent non-inverting level translators that translate signals from low voltage to high voltage or vice versa. The level translator is uni-directional, and it does not matter which voltage is connected first. The circuits have a typical propagation delay of 20 ns and a maximum switching frequency of 10 MHz [26]. These specifications are sufficient for translating the signals used for the simulator hardware, as they have frequency ranges from tens of Hz to tens of kHz.

The circuits also have output-enable pins that can control the output state of each translator. Disabling the output of the level translator, which is connected to the ECU, appears that the signal is disconnected for the ECU. This can be useful for debugging or testing the simulator. I used a level translator circuit, as shown on the schematic in Figure 3.9. We can see the

entire circuit in Appendix C.

This universal circuit module in the schematic can be connected to different voltage sources. In the schematics of the simulator, I used it in four cases:

1. To translate signals from Microzed I/O banks, which are powered with 3.3 V to the 5 V voltage level of most of the digital sensors inputs of the ECU. 5 V level is used by the SENT interface or by wheel sensors, crankshaft, and camshaft hall sensors use this voltage level too.

2. To translate signals from Microzed I/O banks, which are powered with 3.3 V to the 12 V level. This voltage level is used by an *oil level and temperature sensor (G266)*, and *clutch position sensor (G476)*, and it can be used to simulate signals on contacts of automotive wiring.

3. To translate the ECU outputs with 5 V level to the 3.3 V of the MicroZed I/O banks. The ECU uses 5 V level to control valves, ignition coils with the output stage, and a coolant pump for the low-temperature circuit.

4. To translate the ECU output with 12 V level to 3.3 V of the Microzed I/O banks. The ECU uses 12 V signal to control electric motors using PWM or on/off signal, heating of lambda probes, and camshaft control valves.

3.3.2 DAC Circuit

The DAC circuit generates analog signals for the ECU. I use DAC circuits to simulate the inputs from various analog sensors. The DAC circuit consists of three DAC128S085 chips connected to a chain. DAC chips are 12-bit micro-power octal digital-to-analog converters with rail-to-rail outputs. The DAC128S085 chips have the following features [27]:

- They can operate from a 2.7 V to 5.5 V supply and consume low power.
- They have a 3-wire serial interface compatible with common SPI, QSPI, MICROWIRE, and DSP interfaces.
- They have a daisy-chain capability, which allows multiple DACs to be connected to a single SPI with a single chip select pin.

The DAC circuit uses only the `MOSI` (Master Out, Slave In) and `SCLK` (Serial Clock) pins from the SPI interface. It uses the `DOUT` (Data Out) pin of each chip for the chaining. The circuits are connected together using Data pins. The chip-select pins (`SYNC`) of all three chips are connected together, similarly to `SCLK` pins. The data input pin (`DIN`) of the first chip, the `SCLK`, and the `SYNC` line are connected to the MicroZed SPI interface (`MOSI` and `SCLK`) or FPGA-related pins through the connector between the Microzed and Processing parts. To select SPI peripheral or FPGA pins, solder-jumpers `JP15`, `JP16`, `JP17` must be bridged according to the choice. The output pins (`VOUTA` to `VOUTH`) of all circuits are connected directly to the outputs of the processing part of the simulator. The DAC circuit provides 24 analog outputs,

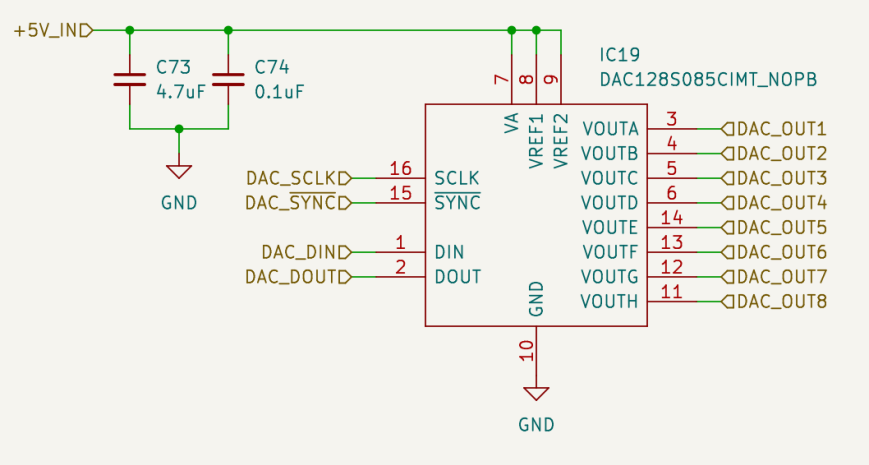


Figure 3.10: The schematic of a DAC circuit. This is a single DAC circuit. In the whole schematic of the simulator, It is used three times.

which are used to simulate all analog sensors, that are connected to the ECU, like three *coolant temperature sensors (G62, G82, G83)*, *fuel pressure sensor (G247)*, and *Knock sensor (G61)*. Figure 3.10 shows the schematic of a single DAC chip in the circuit, and we can see the entire circuit in Appendix C.

3.3.3 ACT Simulator Circuit

Active Cylinder Technology (ACT) is a system that enables the deactivation of individual cylinders in a multi-cylinder engine to reduce fuel consumption and emissions, in our case, engines from the EA211 EVO family. The ACT system consists of ACT actuators that control movable camholders on camshafts to enable or disable intake and exhaust valves. The ACT actuators are based on solenoid actuators and generate the short voltage feedback pulse when an actuator completes the move of its camholder.

The ACT actuator is connected to the car battery and the ECU – the ECU command to move the ACT actuator is a short pulse to the common ground. This pulse actuates the ACT actuator and moves the camholder on the camshaft to turn on the intake/exhaust valves. When the camholder is in its end position, it generates a feedback pulse in the ACT actuator, which the ECU can measure. The feedback pulse has a shape similar to a positive voltage spike with an amplitude ranging from 16 V to 30 V with a duration of approximately 20 ms. An oscilloscope measurement of the simulated ACT signal is shown in Figure 3.11. This simulated signal is sufficient for the ECU but does not simulate the ACT pulse accurately. The measurements of the ACT signal from the running car engine are not available to the date of this work.

The ACT simulator circuit can measure the ACT control pulses from the ECU and generate short pulses that mimic the feedback pulses of the real ACT actuators. The ACT simulator circuit works with the MicroZed SoM, where the simulation program can run, or with the FPGA that can be configured

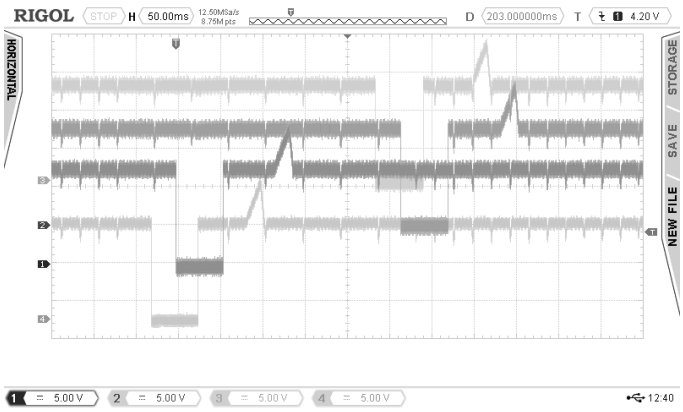


Figure 3.11: The example of the ACT control and feedback pulses were measured on the physical simulator in Škoda Auto a.s. in Mladá Boleslav with an oscilloscope Rigol DS4014E.

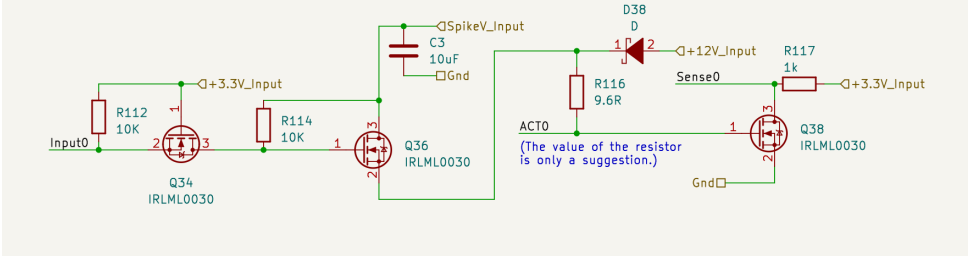


Figure 3.12: The schematic of a single ACT circuit. The ACT circuit simulator includes eight circuits. Resistor `R116` simulates the load of the ACT actuator coil, and its value will be adjusted during development. The ACT pin of the ECU is connected to the wire in the middle of the figure labeled `ACT0`. The part of the circuit which senses the control pulses is on the right side of it. The MicroZed control and measure pins for the ACT simulation are connected to wires with labels The `sense0` and `input0`.

for it in case that a precise and timely response is required. The circuit of a single ACT channel is shown in Figure 3.12, where we can see the ACT input in the middle, the sense circuit on the left, and the circuit to generate ACT feedback pulses on the right. The circuit uses a MOSFET to sense a pulse with a low voltage level to generate an inverted positive pulse with 3.3 V level for the MicroZed. The feedback circuit uses a MOSFET to switch voltage from the ACT voltage regulator on and off. The pulse generating and timing must be programmed into the MicroZed.

3.3.4 LIN Circuit

The LIN circuit is a transceiver for the simulator to connect it to the LIN bus, a low-cost communication network for automotive applications. The LIN circuit acts as a slave node, which receives and transmits data from and to the master node, which is the ECU. The LIN circuit consists of the NCV7329

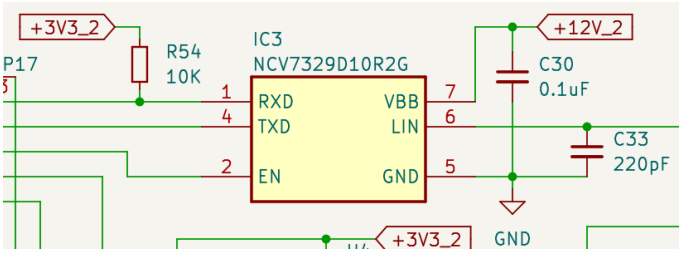


Figure 3.13: The schematic of a LIN transceiver circuit configured as a slave to simulate sensors with LIN interface.

chip, a stand-alone LIN transceiver manufactured by ON Semiconductor. The LIN circuit is the typical application circuit from its datasheet [28] as shown in Figure 3.13. The NCV7329 chip has the following features:

- It complies with ISO 17987-4 and SAE J2602 standards for LIN bus communication.
- It can be powered directly from a car battery and consumes low power.
- It has protection features such as thermal shutdown, under-voltage protection, and bus pins protection against transients.

3.3.5 CAN Circuit

The controller area network is a serial communication protocol that enables data exchange between different devices in a vehicle. The CAN bus consists of two differential lines that carry the CAN high and low signals.

I used the SN65HVD232 from Texas Instruments to communicate with or listen to the ECU unit and other control units connected to the same CAN bus. The SN65HVD232 is a 3.3 V CAN transceiver that is compatible with the ISO 11898-2 standard and can operate at data rates up to 1 Mb/s. The circuit uses the minimal configuration recommended by the datasheet [29], which includes a decoupling capacitor between `VCC` and `GND` pins, and a split termination of the CAN bus. The minimal recommended circuit I used is shown in Figure 3.14. The circuit also provides protection features such as thermal shutdown, open circuit fail-safe, and cross-wire protection.

The processing board has two CAN transceivers. The first is for general use. It is connected to the ECU and to the CAN peripheral of the MicroZed. The second CAN is unpopulated and is connected to the free connector on the connectors part of the simulator board and to the MicroZed's FPGA pins. To communicate with the second CAN transceiver from MicroZed, it is necessary to create a CAN interface using the FPGA part.

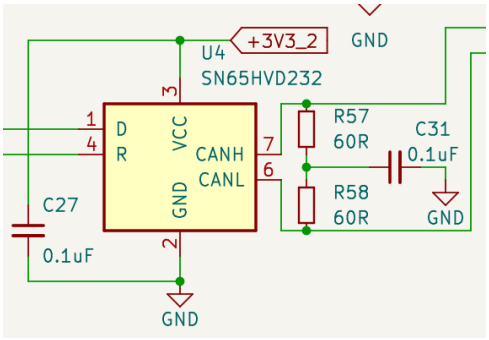


Figure 3.14: The schematic of a CAN transceiver circuit with split termination.

Chapter 4

Simulator Software

The core component of the electronic signal simulator that simulates the engine sensors and actuators for the ECU is the simulator software. In this chapter, I describe the software components of the simulator. The software runs on the MicroZed SoM (with the exception of GUI, which runs on the user's PC). The simulator software consists of the FPGA configuration, programs running on ARM CPU, and the GUI. The FPGA configuration is created with Vivado 2018.3 (a toolchain from Xilinx). The Linux distribution with Linux kernel from Xilinx runs on the ARM CPU of the MicroZed board and contains the simulator program and a few tools for testing. The GUI is intended to run on a user's PC and is realized in CANoe from Vector, a software tool for developing, testing, and analyzing networked systems.

The simulator software of this work is based on the simulator of a 1.4 L petrol engine of the EA211 family created by Ing. Tomáš Procházka. The capabilities of the previous simulator are similar to the new simulator but the older simulator has limited functionality and count of the IO. The previous hardware was built to simulate only a set of the sensors used with 1.4 petrol engine and it can't be modified to simulate different or more signals without reworking of the printed circuit board.

One of the goals of this work is to create a minimal working program to operate the new simulator and verify the functionality of the new hardware. The minimal working program can also serve as a starting point for future development and improvement of the simulator software.

4.1 FPGA Configuration

The FPGA part of the Microzed is used to generate signals to simulate engine timing signals *engine speed sensor (G28)*, *intake camshaft position sensor (G1002)*, *exhaust camshaft position sensor (G1003)*, which needs to be generated precisely. It also measures control pulses from the ECU of *intake camshaft control valve (N318)*, *exhaust camshaft control valve (N727)* which are part of the variable valve timing. There are implemented SENT interfaces for SENT sensors like *oil Pressure sensor (G10)*, *charge pressure sensor (GX26)*, *intake manifold pressure sensor (GX9)*. There are two other SENT sensors which are not been implemented yet due to incomplete information

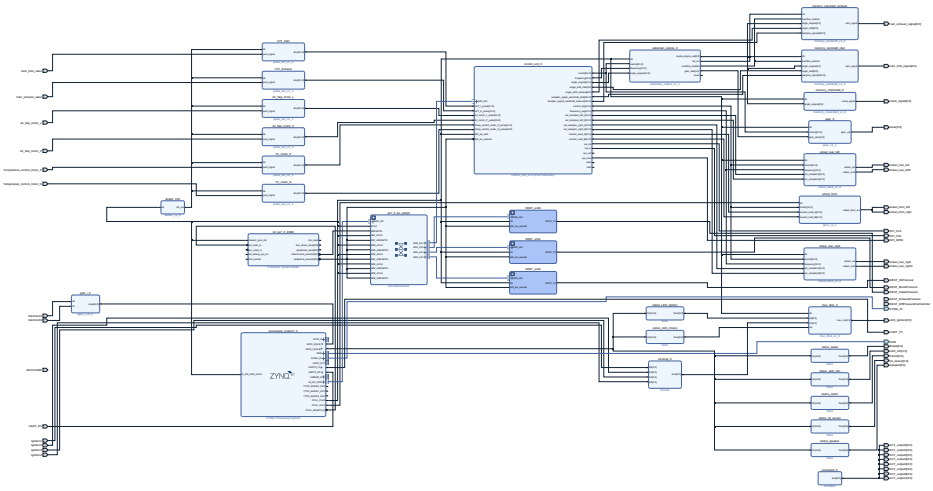


Figure 4.1: The illustration of design for the FPGA of the simulator to illustrate the complexity.

about their protocol and identification at the date of this work. Measurement and control of the FPGA-related pins are also part of the FPGA design. We can see the main diagram of the FPGA design in Figure 4.1.

4.2 Firmware

The MicroZed firmware is based on a Linux distribution with the Linux kernel modified by Xilinx, which supports the Zynq-7000 SoC. To prepare the firmware, we used the Buildroot framework, a tool that automates building a complete Linux system into an image that can be booted up using the u-boot bootloader. Buildroot is configured to include an external package tree, which contains the simulator program `motor-sim` and other programs used to develop or test newly developed functionalities. The Linux system prepared with Buildroot is packed into image for use with u-boot bootloader modified by Xilinx. The description of the structure of the building system we use to create the image of the prepared Linux distribution with the installed simulator programs is as follows:

`build` Contains the building system for automating the creation of the Linux image based on the Makefile

- `build/buildroot` Contains build artifacts and configurations of the Buildroot.
- `build/linux-xlnx` Contains configured and prepared Linux distribution modified by Xilinx.

`build/u-boot-xlnx` Contains building artifacts and the configuration of u-boot modified by Xilinx.

`external` Contains the external package tree definition of the simulator software. Subdirectory `packages` contains package definitions.

`overlay_fs` Contains files that need to be copied into the root directory structure of the prepared Linux system.

`source` Contains sources of the components to prepare and build the Linux system. These are cloned repositories using `git` command configured as submodules to easily manage updates.

`source/buildroot` Contains source files of the Buildroot.

`source/linux-xlnx` Contains source files of the Linux distribution.

`source/u-boot-xlnx` Contains source files of u-boot bootloader.

The output of the build process is the Linux image that is copied on the SD card and inserted into the MicroZed. After the start of the simulator, the u-boot bootloader copies the Linux image into memory and boots it up. After boot-up the simulation program `motor-sim` is configured to start. For development, it is easier to use `ssh` software to connect to the Linux from the local network and control it directly, which is possible after boot-up.

The `motor-sim` program manages the control of FPGA design parts, simulates the basic engine model, and implements the Modbus communication interface to communicate with the GUI reused from the 1.4 L engine simulator. This simulator board also contains new DAC chips described in Section 3.3.2, for which I implemented a library to control them. The reused code was reworked to use the new input and output pins configuration of the simulator hardware designed in this work.

4.3 GUI

To control the simulator software running on the MicroZed, Ing. Tomáš Procházka developed the GUI program, which is used for a user to interact with the simulator. The GUI provides features such as the RPM and the velocity gauges to show the actual engine and the vehicle state, panels to change the actual gear of the gearbox, the brake control to slow down the vehicle, and indicators to show the actual status of a car electrical terminals related to the engine operation. The user interface of the program is shown in Figure 4.2.

The GUI communicates with the simulator hardware using the USB cable connected to the USB port of the galvanically isolated USB-UART interface located on the MicroZed part of the simulator board. The `motor-sim` program exchanges the state of the engine and vehicle with the GUI using the Modbus interface.

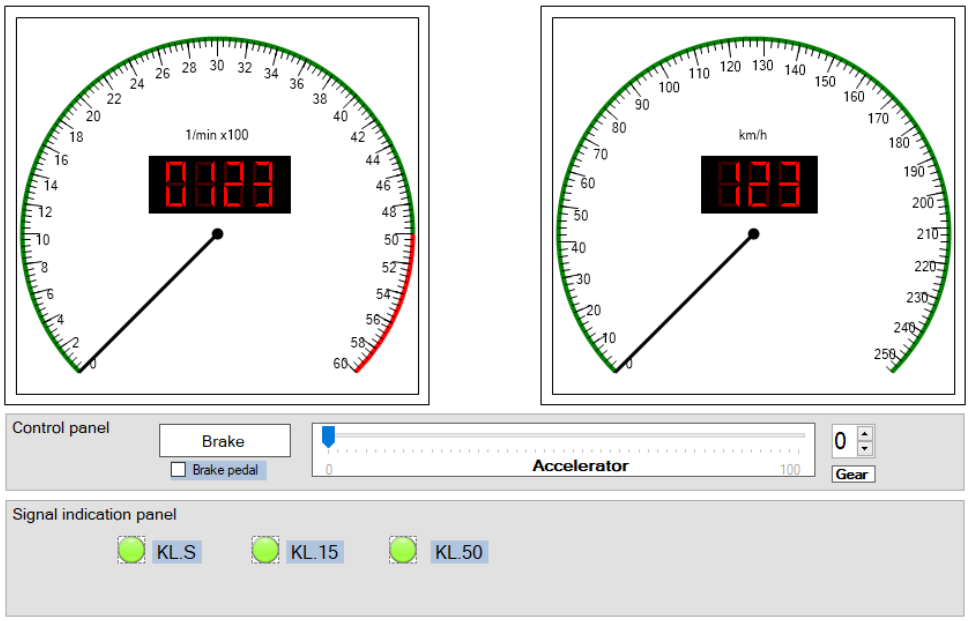


Figure 4.2: The screenshot of the GUI design with controls to operate the engine simulation. The GUI on the screenshot is nonfunctional but with all user interface components visible to illustrate all functions.

Chapter 5

Evaluation

The main goal of this diploma thesis was to design a hardware electrical signal simulator of the 1.5 L 110 kW ACT engine from EA211 EVO family for the ECU of this motor. The simulator consists of three main parts: the MicroZed part, the processing part, and the connectors part. The MicroZed SoM connected to the MicroZed part of the simulator runs FPGA design and the simulation software. The actual FPGA design simulates the crankshaft and camshaft signals of the motor and wheel sensors. The simulation software controls the FPGA design parameters and DAC circuits to generate analog signals. The signal translators and DAC circuits are located in the processing part of the simulator. The connectors part of the simulator has a wheel sensor circuit that translates signals from MicroZed to signal levels expected from the ABS/ESP unit that processes the signals and updates the car's velocity on the car's instrument cluster. I evaluate the simulator's functionality by testing each part separately, by testing the parts while connected together, and by comparing the simulated signals with the real signals measured on parts of a Škoda HIL in Mladá Boleslav. In the final evaluation, the simulator is connected to the Škoda breadboard located in Czech Institute of Informatics, Robotics and Cybernetics (CIIRC), CTU in Prague to test the simulator software. The Škoda breadboard in our laboratory in CIIRC is shown with the connected simulator in Figure 5.1.

5.1 Simulator Electronics Testing

The first part I tested was the MicroZed board, which runs the simulation software and controls the FPGA design. The MicroZed board requires a power supply that can deliver 3.3 V and 5 V voltages with sufficient current. I designed a power supply circuit based on the MAX20808 device, which has features such as soft start, under-voltage protection, and power-good output, which are essential for the optimal operation and safety of the simulator. However, during the testing process, I encountered a problem with the 5 V voltage channel of the designed power supply circuit. It turned out that the 5 V branch was sensitive to the input voltage level and became unstable when the input voltage dropped below a certain threshold. This caused the power-good signal to switch off correctly, but low input voltage disrupted the

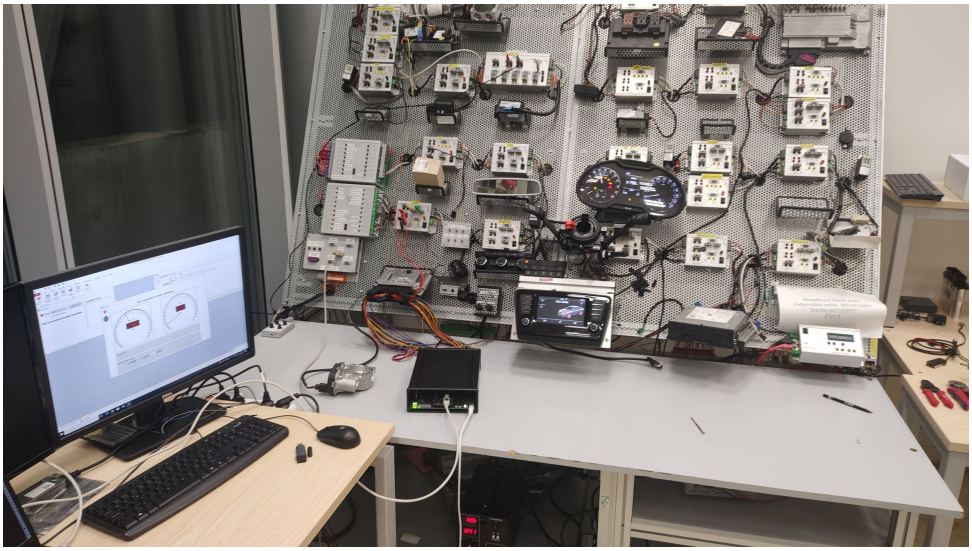


Figure 5.1: The photo of the Škoda breadboard with connected simulator that was designed, assembled and tested in this work. The simulator runs the engine simulation with the started engine with the first gear engaged and 60 % of the accelerator pedal applied. The instrument cluster and the GUI running on the PC on the left side show the engine's actual RPM and the car's velocity.

regulation of the 5 V channel, and the regulation circuit of the 5 V channel was damaged in the process.

To solve the damaged 5 V channel of the power source, I replaced the 5 V channel with a 5 V switching power supply module and added the MCP120-485 voltage watchdog to generate the power-good signal for the MicroZed SoM. After this modification, the power supply circuit worked as expected and provided stable voltages for the MicroZed SoM. After the power source testing, I assembled the rest of the MicroZed board, especially the USB-UART circuit required to control the simulation software from the user interface. As expected, the assembled USB-UART circuit worked alone, but successful communication with the simulator can be tested by running a Linux distribution with serial utilities for the serial interface testing. I prepared the Linux image necessary to boot up the Linux distribution with simulation software on the MicroZed SoM. After powering up the MicroZed part of the simulator from the 12 V laboratory source, the MicroZed SoM successfully booted up the prepared Linux image. Testing the MicroZed serial interface connected to the USB-UART circuit was also successful, and the galvanic isolation circuit translated signals correctly.

The next parts I assembled and tested were the processing and connectors parts of the simulator. The processing part was tested alone, powered by the laboratory power source. The signal generator was connected to the translator circuit input, and the oscilloscope was connected to the translator circuit output. I used the signal generator to generate voltage levels and the oscilloscope to verify the outputs. I used this setup to test all the translator circuits. All voltage translators worked as expected. The assembly of DAC

circuits, LIN bus, and CAN bus circuits were only visually inspected due to the lack of a method to test them alone without connected MicroZed SoM. The connector part was also assembled and visually inspected. The brake circuit on the connectors part was tested with a signal generator and an oscilloscope and worked as expected without signal distortions.

The results of this electronics testing showed the issue with the power supply designed to power the MicroZed SoM, and need to be redesigned in the future. The power supply issue was repairable and did not create any obstacles that needed to be addressed before the rest of the evaluation. The rest of the electronics operated within the specifications of the circuits and without any signal distortions.

5.2 Simulation Testing

After testing each part separately, I assembled the simulator parts together. To connect the simulator, the ECU, and the essential components of the Škoda breadboard, I constructed a cable according to the diagram in Figure 3.2 and connected the simulator to the Škoda breadboard. I connected to the simulator over the ethernet to directly control the simulation and have access to the immediate output of the simulation. I connected the USB cable from the simulator to the PC with a running GUI.

At first I tested whether the GUI can connect to the simulator, where the server that should respond is running. The GUI was not able to connect to the server. After diagnosing the isolation sub-circuit of the USB-UART circuit and FPGA configuration, I found no problem that should cause the connection problems. When I connected the simulator to another PC with a running serial terminal, I could communicate in both directions without problems. The problem can lie in the software refactoring and reconfiguring for the new hardware. Due to the limited time and missing method to control the simulator without using the user interface, I decided to test the simulation directly with changing parameters in the source code. The issue with the Modbus communication will be addressed in the future.

5.3 Experimental Evaluation

I modified the `motor-sim` program to be able to program its behavior instead of the control from the GUI. After starting the simulation, the program waits a few seconds and then tries to start the engine, press the clutch, engage the first gear, release the clutch, and set the acceleration pedal to 60 %. With this process, the simulated engine could start, and the engine's RPM stabilized at 3800 RPM. The simulated model of the vehicle accelerated to almost 30 km/h and remained at stable velocity according to the output log of the simulator. In this state, I could measure the most important outputs of the simulator, including the crankshaft and camshaft signals. I measured signals on the output connector of the processing part, where the signals should have the

correct voltage levels. The crankshaft measurement is shown in Figure 5.2. I compared the current wave with the one measured in Škoda laboratory in Mladá Boleslav, shown in Figure 5.3. The comparison of the simulated crankshaft signal with the crankshaft signal measured in a Škoda laboratory showed that they were very similar in shape and period. The simulation of the crankshafts and camshafts of the 1,5 L motor of EA211 EVO family is measured in Figure 5.4.

The GUI running on a user's PC does currently not have a connection to control the simulator, but it can successfully receive messages from the CAN interface of the vehicle. The gauges in the GUI indicated 3779 RPM and velocity 29 km/h as shown in Figure 5.5, which was equivalent to the values shown by the instrument cluster and similar to the reported RPM and velocity of the simulator. I searched the source code to find the voltage levels of the DAC configured by the simulator. The measure had a small positive offset up to 112 mV depending on the set value. This issue should be addressed in future development. The simulated engine works and the motor sensors report correct signals. I downloaded the Diagnostic Trouble Codes (DTC) reported by ECU after the ECU was turned on (i) without running the simulator, and (ii) with running the simulator. A comparison of the reported issues is shown in Table 5.1.

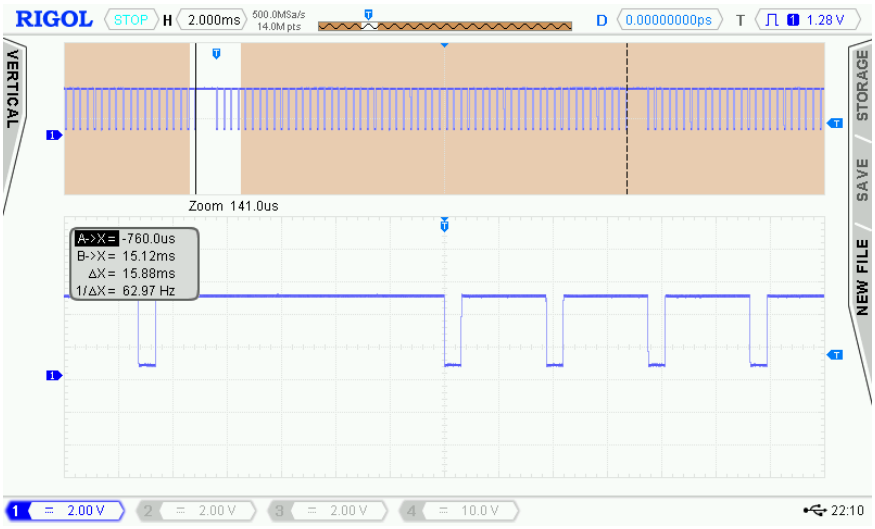


Figure 5.2: The oscilloscope screenshot illustrating the generated crankshaft signal with detail and measurement of the length of the signals period.

The results of the testing of the simulator software show the crankshaft and camshaft sensors simulated with FPGA design work correctly according to oscilloscope measurement, and the wheel sensor simulation worked with the instrument cluster without a problem. The sensors simulated with DACs have issues with uncalibrated DAC value and offset. The DTC report shows there are a lot of problems, but these reports can be misleading. The ECU fault detection does not work exactly. It estimates the possible errors based on the sensors' data and the actuator's feedback. In our case, many of the data

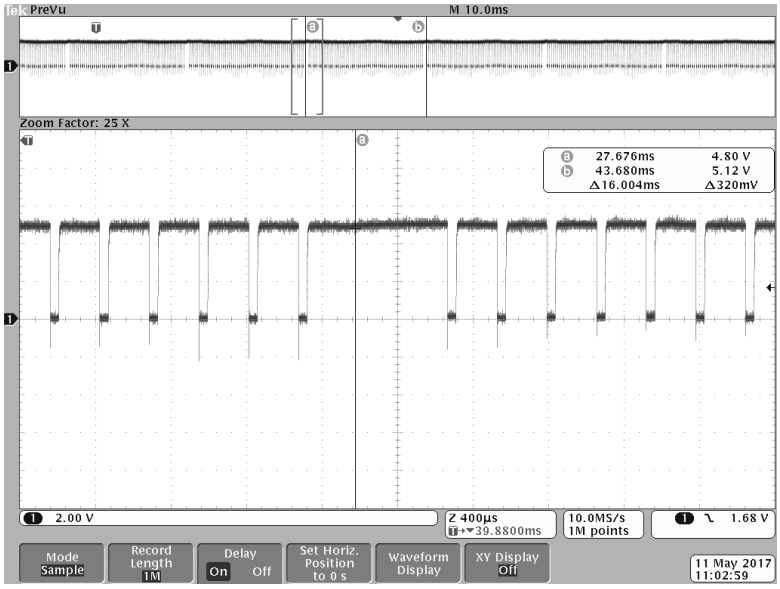


Figure 5.3: The oscilloscope screenshot illustrating the measured crankshaft signal with detail and measurement of the length of the signals period. The signal was measured on the HIL on the psychical engine simulator in the Škoda laboratory in Mladá Boleslav.

signals are not implemented yet, or their implementation is faulty because of the bad reconfiguration of the new simulator hardware. This can lead to a DTC report with faults that should not be present according to real-life measurements. This means that simulator software should be rewritten, and each simulated signal should be revised and reconfigured.

The overall results are promising. The results of the hardware testing are very good. We now have a working hardware platform to develop the simulator software. The simulator has issues with the power supply, but the simulation capabilities are fully working. The software testing results shows a lot of issues, and the overall simulation works only partially.

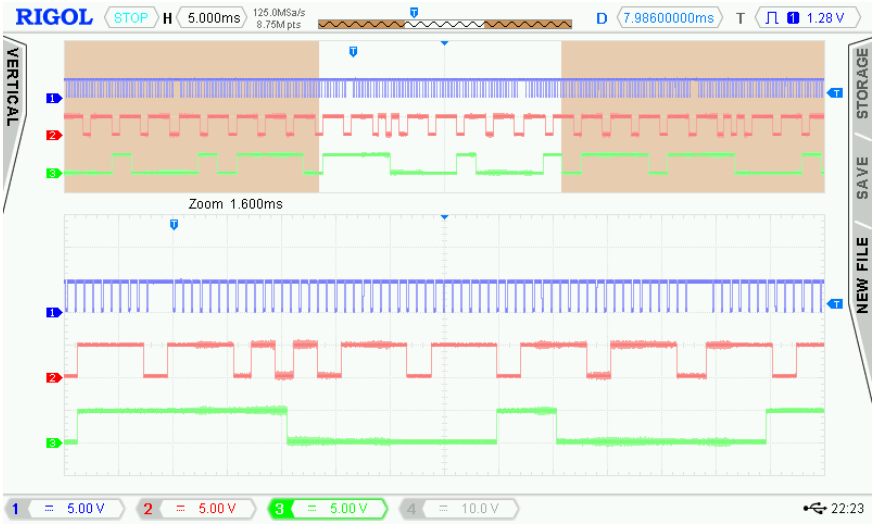


Figure 5.4: The oscilloscope screenshot illustrating the generated crankshaft and camshafts signal measured together. The first blue signal is the crankshaft position sensor simulation. The second red signal simulates the intake camshaft hall sensor output. The green signal is the simulation of the exhaust camshaft hall sensor output. The camshafts have different notch wheels measured by a hall sensor, the exhaust notch wheel has fewer notches, and the measured signals are longer.

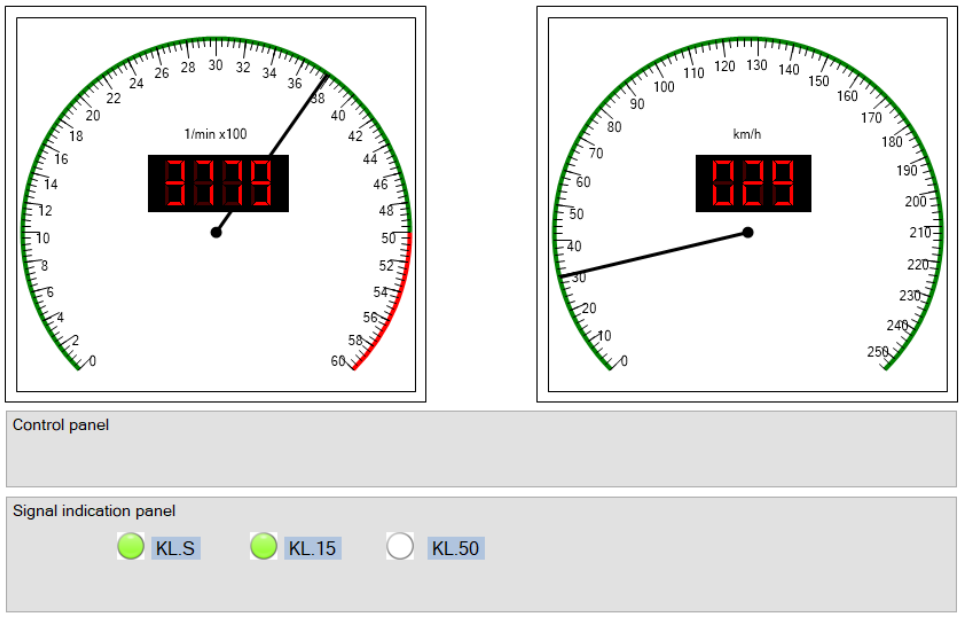


Figure 5.5: The screenshot of the running GUI. The control inputs of the simulator is hidden because the GUI is not connected to the simulator. Gauges show the value received from the CAN bus of the car.

Code	Fault description
001242	<DTC is not defined in the database>
005d05	Engine Oil Temperature Sensor Circuit low
0068f5	Sensor/switch 1 for engine oil pressure No communication
0076f0	EMC/PCM Power Relay Control Circuit Open
00940e	<DTC is not defined in the database>
009410	Lost Communication With Turbocharger/Supercharger Boost Sensor "A"
000afd	Throttle/Pedal Pos. Sens./Switch D Circuit Low Input
000aff	Throttle/Pedal Pos. Sens./Switch E Circuit Low Input
0011af	Cooling Fan 1 Control Circuit
001234	ABS brake control module Read out DTC
001243	<DTC is not defined in the database>
001557	Starter activation, return message terminal 50 Short circuit to B+
001bee	Invalid Data Received From Steering Angle Sensor Module
00221e	Engine Control Module (ECM) disabled
003923	Fuel Pump Module Control Circuit/Open
00436e	Terminal 30 Open circuit
004387	Ignition Switch Run/ Start Position Curciut High
004506	Brake Switch "A" Circuit Low
00468e	Coolant Pump "B" Control Circuit/Open
0046be	O2 Sensor Heater Contr. Circ.(Bank1(1)Sensor 1)
0046c0	O2 Sensor Negative Current Control Circuit Bank 1 Sensor 1 open
004b20	"A" Camshaft Position Slow Response (Bank 1)
004f88	Power supply terminal 15 Implausible
00552b	Cam Shift Actuator Outlet "B" Cylinder 2 Range/Performance
00552c	Cam Shift Actuator Outlet "B" Cylinder 3 Range/Performance
0064f6	Cam Shift Actuator "B" Cylinder 2 Range/Performance
0064f7	Cam Shift Actuator "B" Cylinder 3 Range/Performance
00723d	Engine Oil Pressure Sensor/Switch "A" Range/Performance
0077f9	ECM/PCM Power Relay De-Energized Performance - Too Late
00781b	ECM/PCM Power Relay De-Energized Performance - Too Early
007d2f	Turbocharger Boost Control Position Sensor Circuit - Low
008a55	<DTC is not defined in the database>
008a67	Exhaust Pressure Sensor "A"Low
00931f	Turbocharger/Supercharger Wastegate "A" Stuck Closed
009320	Turbolader/Kompressor Ladedruckbegrenzung 1 klemmt offen
009372	Turbocharger/Supercharger Boost Control Solenoid "A" Circuit/Open
0093e6	Coolant flow control element position sensor short circuit to ground
0093f4	Vacuum Pump Control Circuit/Open
006764	Voltage terminal 15 Implausible
005764	Lost Communication With Anti-Lock Brake System (ABS) Control Module
001266	Lost Communication With Restraints Control Module
004e9b	Turbocharger Boost Sensor (A) Circ. High Input

Table 5.1: The table contains a formatted copy of all DTC reported by ECU. The rows highlighted with blue color were only present when the simulator was disabled. The rows highlighted with green color were only present when the simulator was running. The rest of the rows were present in both cases.

Chapter 6
Conclusion

This thesis presents a complete hardware design of the development board of the engine simulator for the 1.5 L 110 kW ACT petrol engine with Active Cylinder Management (ACT) from the EA211 EVO family and documents characteristics of the Engine Control Unit's (ECU) signals. It also presents basic simulation software and the cable solution to connect the simulator to the ECU of the motor and the rest of the necessary control units of a car. Although the work aimed to create specific hardware with a single application, two years of experience with an automotive control unit simulation development led to designing a modular development board that is easy to work with. The designed board is modular and widely reconfigurable. The parts of the simulator can be easily replaced. The modularity allows measuring generated and received signals at any level of their processing.

The hardware of this simulator is more of a development tool than the hardware of the final simulator. It contains many features and parts that may not be needed after the development of the simulator is finished. The added features and redundant I/O make the development and debugging of the simulator easier and cost-effective.

The outcome of this work will be used to develop the final engine simulator that will be used in Škoda a.s. for ECU testing or simulating the motor in a car testing breadboards. It can be potentially used as a low-cost replacement of the physical simulation in Hardware-in-the-Loop (HIL) vehicle testing or as the engine simulator for developing the virtual car simulation.

6.1 Future Work

In the future, I will address the hardware issues discovered in the Chapter 5. I will also rewrite and recreate the FPGA design and software of the simulator to fit the new hardware board instead of trying to adapt the old simulation to new hardware. This will lead to the verification of every simulated signal. The new software will be modular to be able to replace implementations of the parts of the simulation easily. The graphic user interface will be modified to communicate using compatible, easy-to-use protocols.

Appendix A

Bibliography

[1] I. T. Procházka, *Manuál pro zážehový motor*. ČVUT, Feb 2019.

[2] R. Balmer, *Modern Engineering Thermodynamics*. Academic Press, Dec 2010.

[3] F. Eichler, W. Demmelbauer-Ebner, J. Theobald, B. Stiebels, H. Hoffmeyer, and M. Kreft, *Der neue EA211 TSI®evo von Volkswagen/The New EA211 TSI®evo from Volkswagen*, Jan 2016, pp. I–1.

[4] Škoda Group, *1.5 l TSI 110 kW ACT - four-cylinder petrol engine of the EA211 evo line*, Jan 2018.

[5] Wikipedia contributors, "Pulse-width modulation — Wikipedia, the free encyclopedia," 2023, Accessed: Aug. 7, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Pulse-width_modulation&oldid=1168055433

[6] ——, "Can bus — Wikipedia, the free encyclopedia," 2023, Accessed: Aug. 7, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=CAN_bus&oldid=1169325463

[7] C. Lupini, *Vehicle Multiplex Communication: Serial Data Networking Applied to Vehicular Engineering*, ser. Premiere Series Bks. SAE International, 2004. [Online]. Available: https://books.google.cz/books?id=HuKbEAAAQBAJ

[8] S. Corrigan, "Introduction to the controller area network (can)," 2002, Accessed: Aug. 7, 2023. [Online]. Available: https://www.ti.com/lit/an/sloa101b/sloa101b.pdf

[9] P. Subke, *Diagnostic Communication with Road-Vehicles and Non-Road Mobile Machinery*. SAE International, 2020.

[10] W. contributors, "Sent (protocol) — Wikipedia, the free encyclopedia," 2022, Accessed: Aug. 7, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=SENT_(protocol)&oldid=1123500057

[11] V. A. F. D. C. Standards, *SENT - Single Edge Nibble Transmission for Automotive Applications*, Apr 2016. [Online]. Available: https://doi.org/10.4271/J2716_201604

[12] T. White, "A tutorial for the digital sent interface," 2014, Accessed: Aug. 7, 2023. [Online]. Available: https://www.renesas.com/us/en/document/whp/tutorial-digital-sent-interface-zssc416xzssc417x

[13] SEAT, *SSP169 - 1.5 L TSI ENGINE*, Nov 2017.

[14] Škoda Group, *OCTAVIA III - Current Flow Diagram*, Feb 2017.

[15] Wikipedia contributors, "Oxygen sensor — Wikipedia, the free encyclopedia," 2023, Accessed: Aug. 7, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Oxygen_sensor&oldid=1149586112

[16] ——, "Air–fuel ratio — Wikipedia, the free encyclopedia," 2023, Accessed: Aug. 7, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Air%E2%80%93fuel_ratio&oldid=1154372432

[17] Bosch, "Lambda sensors – quick and easy testing and replacement," 2018, Accessed: Aug. 7, 2023. [Online]. Available: https://www.boschaftermarket.com/xrm/media/images/country_specific/sg/services_and_support_6/downloads_18/lambda_sensors.pdf

[18] Avnet, "Microzed boards," Accessed: Aug. 7, 2023. [Online]. Available: https://www.avnet.com/wps/portal/us/products/avnet-boards/avnet-board-families/microzed/

[19] Xilinx, "Kria K26 System-on-Module," 2023, Accessed: Aug. 7, 2023. [Online]. Available: https://www.xilinx.com/products/som/kria/k26c-commercial.html

[20] The Buildroot developers, "Buildroot - making embedded linux easy – manual," Accessed: Aug. 7, 2023. [Online]. Available: https://buildroot.org/downloads/manual/manual.html

[21] Texas Instruments, "LM2586 4-V to 40-V, 3-A Step-Up Wide VIN Flyback Regulator," 2019, Accessed: Aug. 7, 2023. [Online]. Available: https://www.ti.com/lit/ds/symlink/lm2586.pdf

[22] ——, "TPS7B88-Q1 500-mA, 40-V, Low-Dropout Regulator," 2021, Accessed: Aug. 7, 2023. [Online]. Available: https://www.ti.com/lit/ds/symlink/tps7b88-q1.pdf

[23] ——, "LP2981 100-mA Ultra-Low Dropout Regulators With Shutdown," 2016, Accessed: Aug. 7, 2023. [Online]. Available: https://www.ti.com/lit/ds/symlink/lp2981.pdf

[24] Avnet, "MicroZed™ Carrier Design Guide," 2014, Accessed: Aug. 7, 2023. [Online]. Available: https://www.avnet.com/

wps/wcm/connect/onesite/3136bd0e-5e90-4e77-bb68-32da19078aba/5274-MicroZed-Carrier-Design-Guide-rev-1-5-V1.pdf

[25] Analog Devices, "Dual-Output 4A, 3MHz, 2.7V to 16V Step-Down Switching Regulator," 2022, Accessed: Aug. 7, 2023. [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/MAX20808.pdf

[26] Texas Instruments, "CMOS Quad Low-to-High Voltage Level shifter," 2003, Accessed: Aug. 7, 2023. [Online]. Available: https://www.ti.com/lit/ds/symlink/cd40109b.pdf

[27] ——, "DAC128S085 12-Bit Micro-Power OCTAL Digital-to-Analog Converter With Rail-to-Rail Outputs," 2015, Accessed: Aug. 7, 2023. [Online]. Available: https://www.ti.com/lit/ds/symlink/dac128s085.pdf

[28] ON Semiconductor, "NCV7329 Stand-alone LIN Transceiver," 2018, Accessed: Aug. 7, 2023. [Online]. Available: https://www.onsemi.com/pdf/datasheet/ncv7329-d.pdf

[29] Texas Instruments, "SN65HVD23x 3.3-V CAN Bus Transceivers," 2018, Accessed: Aug. 7, 2023. [Online]. Available: https://www.ti.com/lit/ds/symlink/sn65hvd230.pdf

Appendix B

ECU and Parts Signals

Table B.1: ECU and Parts Signals

Part	Part Pin	KiCAD Name	ECU Pin	ECU Direction	Low[V]	High[V]	Signal Type	Circuit Type
G10.8	5V	G10_G31_GX34_5V	A26	Power	—	—	—	—
G10.8	SENT	G10_OilPressureSensor_SENT	A27	Input	0	5	SENT	Trans. 3.3V to 5V
G10.8	GND	G10_G31_GX34_Gnd	A29	Ground	—	—	—	—
G1002.1	Signal	G1002_IntakeCamshaftPosition_Signal	A42	Input	0	5	Pulse	Trans. 3.3V to 5V
G1002.1	Vcc	G71_G1002_G1003_5V	A45	Power	—	—	—	—
G1002.1	Masse	G71_G1002_G1003_Gnd	A60	Ground	—	—	—	—
G1003.7	Signal	G1003_ExhaustCamshaftPosition_Signal	A43	Input	0	5	Pulse	Trans. 3.3V to 5V
G1003.7	Vcc	G71_G1002_G1003_5V	A45	Power	—	—	—	—
G1003.7	Masse	G71_G1002_G1003_Gnd	A60	Ground	—	—	—	—
G1037.6	5V	G28_G247_G450_G1037_5V	A25	Power	—	—	—	—
G1037.6	GND	G247_G450_G1037_Gnd	A38	Ground	—	—	—	—
G1037.6	SENT	G1037_DiffPressureSensorParticleFilter_SENT	A40	Input	0	5	SENT	Trans. 3.3V to 5V
G247.8	5V	G28_G247_G450_G1037_5V	A25	Power	—	—	—	—
G247.8	GND	G247_G450_G1037_Gnd	A38	Ground	—	—	—	—
G247.8	Signal	G247_FuelPressureSensor_Signal	A56	Input	0	5	Analog	DAC 5V
G266.32	Output	G266_OilLevelOilTemp_Output	K43	Input	0	12	Protocol	Trans. 3.3V to 12V
G28.9	Masse	G28_Crankshaft_Gnd	A07	Ground	—	—	—	—
G28.9	Signal	G28_Crankshaft_Signal	A08	Input	0	5	Pulse	Trans. 3.3V to 5V
G28.9	Vcc	G28_G247_G450_G1037_5V	A25	Power	—	—	—	—
G31.3	5V	G10_G31_GX34_5V	A26	Power	—	—	—	—
G31.3	SENT	G31_BoostPressureSensor_SENT	A28	Input	0	5	SENT	Trans. 3.3V to 5V
G31.3	GND	G10_G31_GX34_Gnd	A29	Ground	—	—	—	—
G450.2	5V	G28_G247_G450_G1037_5V	A25	Power	—	—	—	—
G450.2	GND	G247_G450_G1037_Gnd	A38	Ground	—	—	—	—
G450.2	Analoger Ausgang	G450_ExhaustPressure_Analog	A39	Input	0	—	Analog	DAC 5V
G61.3	Sensor	G61_KnockSensor_A	A58	Input	1.5	+-1	Analog	—
G61.3	Sensor	G61_KnockSensor_B	A59	Input	1.5	+-0,2	Analog	—
G62.10	1	G62_G82_Gnd	A53	Ground	—	—	—	—
G62.10	2	G62_CoolantTempSensor	A54	Input	0	5	Analog	DAC 5V
G701.7	Gnd	G701_TransmissionNeutralPosition_Gnd	K16	Ground	—	—	—	—
G701.7	Signal	G701_TransmissionNeutralPosition_Signal	K38	Input	0	5	Protocol	Trans. 3.3V to 5V
G701.7	5V	G701_TransmissionNeutralPosition_5V	K39	Power	—	—	—	—
G71.7	SENT	G71_IntakePressureSensor_SENT	A44	Input	0	5	SENT	Trans. 3.3V to 5V
G71.7	5V	G71_G1002_G1003_5V	A45	Power	—	—	—	—

Continued on next page

Table B.1 – continued from previous page

Part	Part Pin	KiCAD Name	ECU Pin	ECU Direction	Low[V]	High[V]	Signal Type	Circuit Type
G71.7	GND	G71_G1002_G1003_Gnd	A60	Ground	—	—	—	—
G82.5	1	G62_G82_Gnd	A53	Ground	—	—	—	—
G82.5	2	G82_CoolantTempAfterMotorSensor	A55	Input	0	5	Analog	DAC 5V
G83.6	2	G83_CoolantTempAfterCooler_Gnd	K74	Ground	—	—	—	—
G83.6	1	G83_CoolantTempAfterCooler	K75	Input	0	—	Analog	DAC 5V
GX10.6	U_Heiz -	GX10_Lambda1BeforeCat_Heat-	K07	Output	0	12	—	Trans. 12V to 3.3V
GX10.6	Ip/Vs	GX10_Lambda1BeforeCat_IpPerVs	K54	Ground	—	—	—	—
GX10.6	Vs +	GX10_Lambda1BeforeCat_Vs+	K55	Input	0	—	Analog	DAC 5V
GX10.6	Kompensations-R	GX10_Lambda1BeforeCat_CompR	K56	Input	0	—	Analog	DAC 5V
GX10.6	Ip +	GX10_Lambda1BeforeCat_Ip+	K57	Input	0	1	Analog	DAC 5V
GX10.6	U_Heiz +	—	Outside	Power	—	—	—	—
GX2.4	Kanal 1 Masse	GX2_AcceleratorPedal_Ch1_Gnd	K10	Ground	—	—	—	—
GX2.4	Kanal 2 Masse	GX2_AcceleratorPedal_Ch2_Gnd	K11	Ground	—	—	—	—
GX2.4	Kanal 1 +5V	GX2_AcceleratorPedal_Ch1_5V	K31	Output	—	—	—	—
GX2.4	Kanal 1 Sensorsig.	GX2_AcceleratorPedal_Ch1_Signal	K32	Input	0	—	Analog	DAC 5V
GX2.4	Kanal 2 Sensorsig.	GX2_AcceleratorPedal_Ch2_Signal	K33	Input	0	—	Analog	DAC 5V
GX2.4	Kanal 2 +5V	GX2_AcceleratorPedal_Ch2_5V	K34	Power	—	—	—	—
GX3.8	M-	GX3_Throttle_M-	A01	Output	0	12	—	Trans. 12V to 3.3V
GX3.8	S+	GX3_Throttle_S+	A11	Power	—	—	—	—
GX3.8	IS1	GX3_Throttle_IS1	A12	Input	0	—	Analog	DAC 5V
GX3.8	IS2	GX3_Throttle_IS2	A13	Input	0	—	Analog	DAC 5V
GX3.8	S-	GX3_Throttle_S-	A14	Ground	—	—	—	—
GX3.8	M+	GX3_Throttle_M+	A16	Output	0	12	—	Trans. 12V to 3.3V
GX33.3	M+	GX33_EngineTempReg_M+	A03	Output	0	12	—	Trans. 12V to 3.3V
GX33.3	Signal	GX33_EngineTempReg_Signal	A09	Input	0	5	Analog	DAC 5V
GX33.3	5V	GX33_EngineTempReg_5V	A10	Power	—	—	—	—
GX33.3	M-	GX33_EngineTempReg_M-	A17	Output	0	12	—	Trans. 12V to 3.3V
GX33.3	GND	GX33_EngineTempReg_Gnd	A22	Ground	—	—	—	—
GX34.5	M+	GX34_TurboAirFlap_M+	A02	Output	0	12	—	Trans. 12V to 3.3V
GX34.5	M-	GX34_TurboAirFlap_M-	A18	Output	0	12	—	Trans. 12V to 3.3V
GX34.5	5V	G10_G31_GX34_5V	A26	Power	—	—	—	—
GX34.5	GND	G10_G31_GX34_Gnd	A29	Ground	—	—	—	—
GX34.5	Signal	GX34_TurboAirFlap_Signal	A41	Input	0	—	Analog	DAC 5V
GX41.5	SENT	GX41_ExhaustTempSensor_SENT	K15	Input	0	5	SENT	Trans. 3.3V to 5V
GX41.5	GND	GX41_ExhaustTempSensor_Gnd	K19	Ground	—	—	—	—

Continued on next page

Table B.1 – continued from previous page

Part	Part Pin	KiCAD Name	ECU Pin	ECU Direction	Low[V]	High[V]	Signal Type	Circuit Type
GX41.5	Usignal 5V	GX41_ExhaustTempSensor_5V	K20	Output	—	—	—	—
GX44.4	5V	GX44_AirPressureSensorActiveCoal_5V	K17	Output	—	—	—	—
GX44.4	Drucksignal	GX44_AirPressureSensorActiveCoal_Signal	K18	Input	0	5	—	Trans. 3.3V to 5V
GX44.4	NTC	GX44_AirPressureSensorActiveCoal_NTC	K80	Input	0	5	Analog	DAC 5V
GX44.4	GND	GX44_AirPressureSensorActiveCoal_Gnd	K81	Ground	—	—	—	—
GX7.2	U_Heiz -	GX7_Lambda1AfterCat_Heat-	K73	Output	0	12	—	Trans. 12V to 3.3V
GX7.2	Sondensignal	GX7_Lambda1AfterCat_Signal	K77	Input	0	1	Analog	DAC 5V
GX7.2	Sondenmasse	GX7_Lambda1AfterCat_Gnd	K78	Ground	—	—	—	—
GX7.2	U_Heiz +	—	Outside	Power	—	—	—	—
J883	PWM	J883_ExhaustFlap_PWM	K91	Output	0	12	PWM	Trans. 12V to 3.3V
N127.6	Zuendsignal	N127_Ignition_2_Signal	A51	Output	0	5	Pulse	Trans. 5V to 3.3V
N290.6	2	N290_FuelMeteringValve_2(L)	A32	Output	0	5	Pulse	Trans. 5V to 3.3V
N290.6	1	N290_FuelMeteringValve_1(H)	A47	Output	0	5	Pulse	Trans. 5V to 3.3V
N291.6	Zuendsignal	N291_Ignition_3_Signal	A52	Output	0	5	Pulse	Trans. 5V to 3.3V
N292.4	Zuendsignal	N292_Ignition_4_Signal	A36	Output	0	5	Pulse	Trans. 5V to 3.3V
N30.11	Plus	N30_FuelInjectionValve_1+	A33	Output	0	5	Pulse	Trans. 5V to 3.3V
N30.11	Minus	N30_FuelInjectionValve_1-	A35	Output	0	5	Pulse	Trans. 5V to 3.3V
N31.8	Minus	N31_FuelInjectionValve_2-	A20	Output	0	5	Pulse	Trans. 5V to 3.3V
N31.8	Plus	N31_FuelInjectionValve_2+	A49	Output	0	5	Pulse	Trans. 5V to 3.3V
N318.5	Minus	N318_CamshaftExhaustAdj-	A30	Output	0	12	On/Off	Trans. 12V to 3.3V
N32.8	Plus	N32_FuelInjectionValve_3+	A34	Output	0	5	Pulse	Trans. 5V to 3.3V
N32.8	Minus	N32_FuelInjectionValve_3-	A50	Output	0	5	Pulse	Trans. 5V to 3.3V
N33.4	Minus	N33_FuelInjectionValve_4-	A19	Output	0	5	Pulse	Trans. 5V to 3.3V
N33.4	Plus	N33_FuelInjectionValve_4+	A48	Output	0	5	Pulse	Trans. 5V to 3.3V
N428.6	Minus	N428_ValveOilPressureReg-	A04	Output	0	12	—	Trans. 12V to 3.3V
N583.1	Suple 1	N583_InletCamshaftActuator_2L	K50	Output	0	12 (>12)	ACT	ACT
N583.1	Suple 2	N583_InletCamshaftActuator_2N	K72	Output	0	12 (>12)	ACT	ACT
N587.1	Suple 2	N591_ExhaustCamshaftActuator_2N	K27	Output	0	12 (>12)	ACT	ACT
N587.1	Suple 1	N591_ExhaustCamshaftActuator_2L	K93	Output	0	12 (>12)	ACT	ACT
N591.1	Suple 1	N591_InletCamshaftActuator_3N	K28	Output	0	12 (>12)	ACT	ACT
N591.1	Suple 2	N591_InletCamshaftActuator_3L	K71	Output	0	12 (>12)	ACT	ACT
N595.1	Suple 1	N591_ExhaustCamshaftActuator_3N	K25	Output	0	12 (>12)	ACT	ACT
N595.1	Suple 2	N591_ExhaustCamshaftActuator_3L	K94	Output	0	12 (>12)	ACT	ACT
N70.6	Zuendsignal	N70_Ignition_1_Signal	A37	Output	0	5	On/Off	Trans. 5V to 3.3V
N727.2	Minus	N727_CamshaftIntakeAdj-	A15	Output	0	12	On/Off	Trans. 12V to 3.3V

Continued on next page

Table B.1 – continued from previous page

Part	Part Pin	KiCAD Name	ECU Pin	ECU Direction	Low[V]	High[V]	Signal Type	Circuit Type
N80.4	Minus	N80_EMValveActiveCoal-	A06	Output	0	12	On/Off	Trans. 12V to 3.3V
V468.1	Signal	V468_CoolantPumpLowTemp_Signal	A21	Output	0	5	—	Trans. 5V to 3.3V
Outside	—	GND	K01	Ground	—	—	—	—
Outside	—	GND	K02	Ground	—	—	—	—
Outside	—	12V	K05	Power	—	—	—	—
Outside	—	12V	K06	Power	—	—	—	—
Outside	—	V550_RadiatorShutterServomotor_LIN	K08	IO	0	12	LIN	LIN
Outside	—	J743_MechatronicsDoubleClutch_TipUp	K30	Output	—	—	—	—
Outside	—	Kl50	K44	Input	0	12	On/Off	—
Outside	—	J538_FuelPumpControlUnit_PWM	K48	Output	0	12	PWM	Trans. 12V to 3.3V
Outside	—	Starter2	K49	Output	0	12	On/Off	Trans. 12V to 3.3V
Outside	—	—	K52	—	—	—	—	—
Outside	—	G476_ClutchPositionSensor_KUP	K59	Input	0	12	On/Off	Trans. 3.3V to 12V
Outside	—	J527_SteeringColumnElectronicsControlUnit	K61	Input	—	—	—	—
Outside	—	F.1_BrakeLight_BLS	K62	Input	0	12	On/Off	Trans. 3.3V to 5V
Outside	—	F.1_BrakeLight_BTS	K63	Input	0	12	On/Off	Trans. 3.3V to 5V
Outside	—	J533_Diagnostic_CANL	K67	IO	—	—	CAN	—
Outside	—	J533_Diagnostic_CANH	K68	IO	—	—	CAN	—
Outside	—	Starter1	K70	Output	0	12	On/Off	Trans. 12V to 3.3V
Outside	ILS/PN	G476_ClutchPositionSensor_ILS	K79	Input	0	12	On/Off	Trans. 3.3V to 12V
Outside	—	DL1_Datalogger1_CAN14H	K82	—	—	—	—	—
Outside	—	DL1_Datalogger1_CAN14L	K83	—	—	—	—	—
Outside	—	Kl50R	K85	Input	0	12	On/Off	—
Outside	—	Kl15	K87	Input	0	12	On/Off	Trans. 3.3V to 12V
Outside	—	VX57_RadiatorFan_PWM	K89	Output	0	12	PWM	Trans. 12V to 3.3V
Outside	—	Kl30	K92	Power	11	15	—	—

Appendix C

Schematics

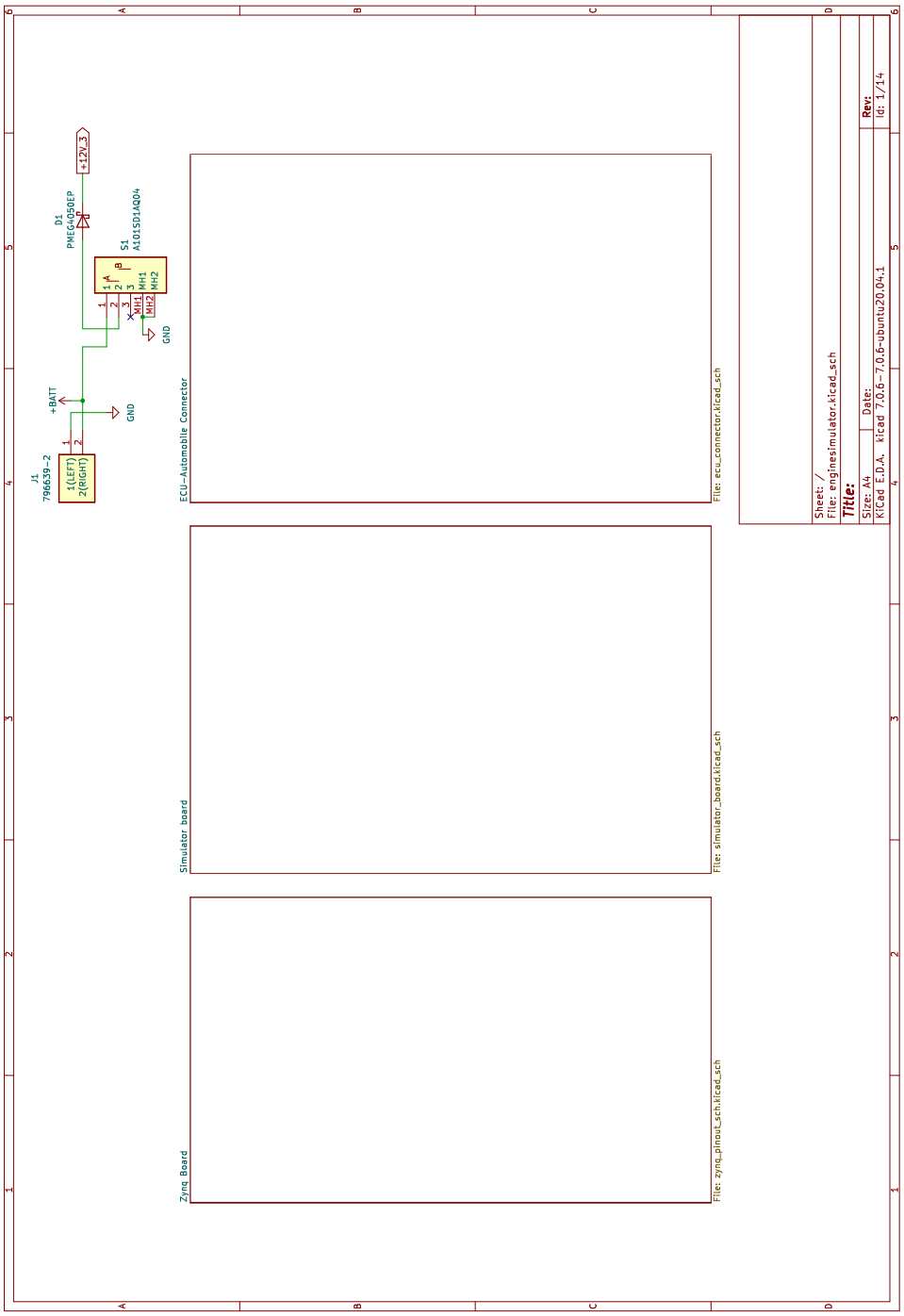


Figure C.1: Board layout

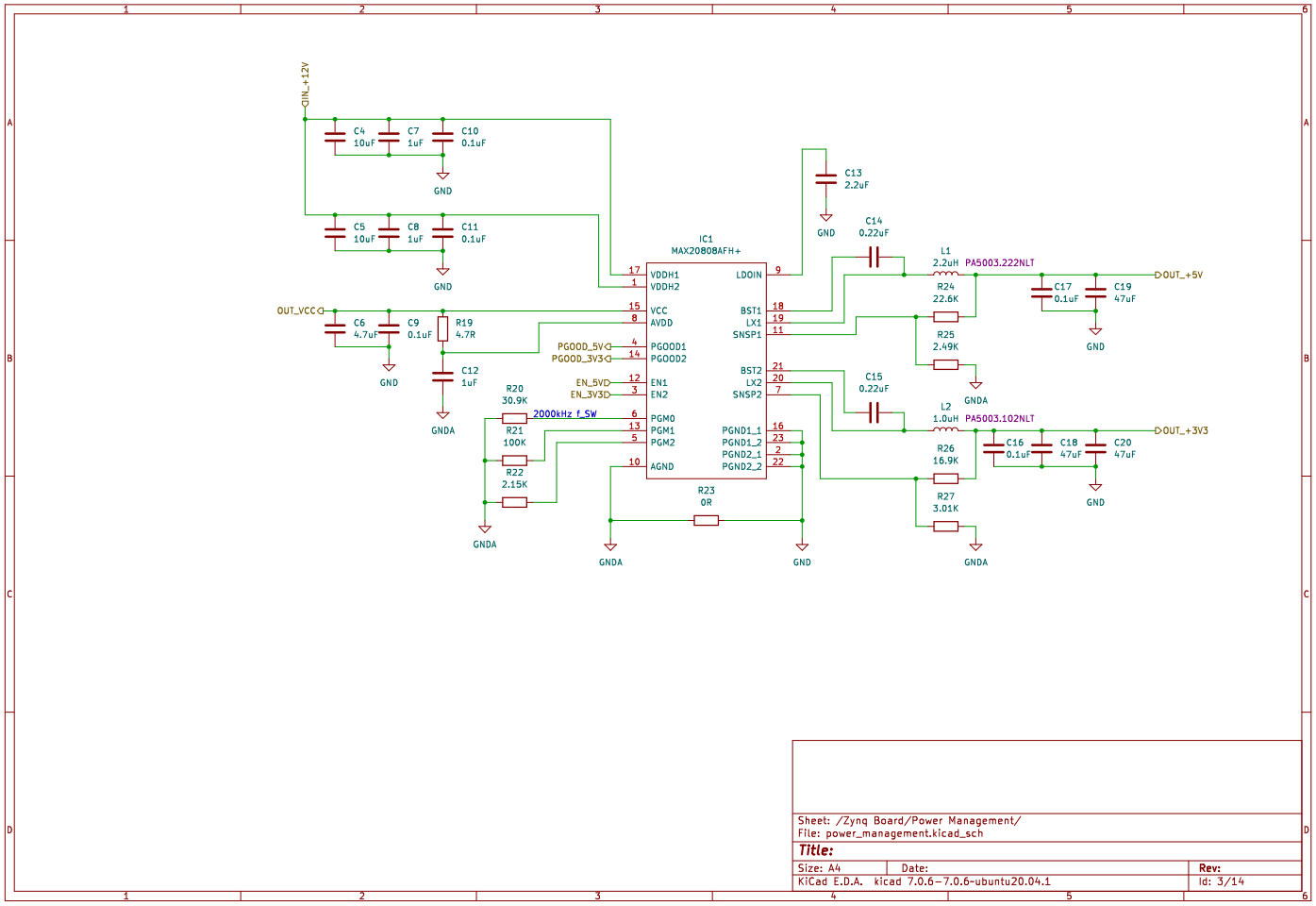
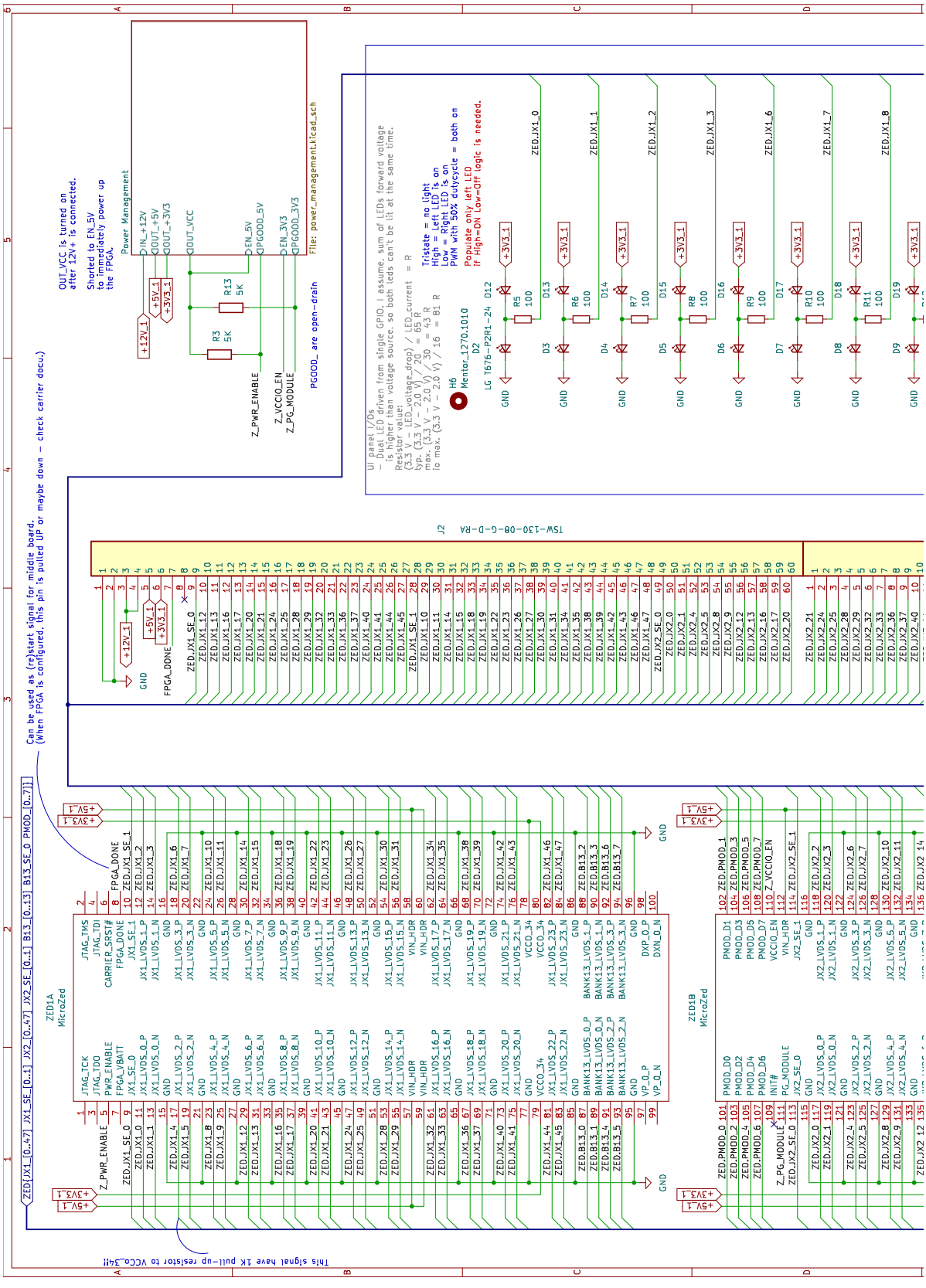


Figure C.2: Power supply of MicroZed


Power Management
Can be used as (re)start signal for middle board.
(When FPGA is configured, this pin is pulled UP or maybe down - check carrier docu.)
This signal have 1K pull-up resistor to VCCo_34!
OUT_VCC is turned on
after 12V+ is connected.
Shorted to EN_5V
to immediately power up
the FPGA.
PGOOD_ are open-drain
UI panel I/Os
- Dual LED driven from single GPIO. I assume, sum of LEDs forward voltage is higher than voltage source, so both leds can't be lit at the same time.
Resistor value:
(3.3 V - LED_voltage_drop) / LED_current = R
typ. (3.3 V - 2.0 V) / 20 = 65 R
max (3.3 V - 2.0 V) / 30 = 43 R
lo max (3.3 V - 2.0 V) / 16 = 81 R
Tristate = no light
High = Left LED is on
Low = Right LED is on
PWM with 50% dutycycle = both on
Populate only left LED
if High=ON Low=Off logic is needed.
ZED1A
MicroZed
ZED1B
MicroZed
TSW-130-08-G-D-RA
J2
Mentor_1270.1010
H6
LG T676-P2R1-24
D2
R3
5K
R13
5K
R5
100
R6
100
R7
100
R8
100
R9
100
R10
100
R11
100
ZED.JX1_SE_0
ZED.JX1_SE_1
ZED.JX2_SE_0
ZED.JX2_SE_1
Z_PWR_ENABLE
Z_VCCIO_EN
Z_PG_MODULE
FPGA_DDNE
FPGA_DONE
GND

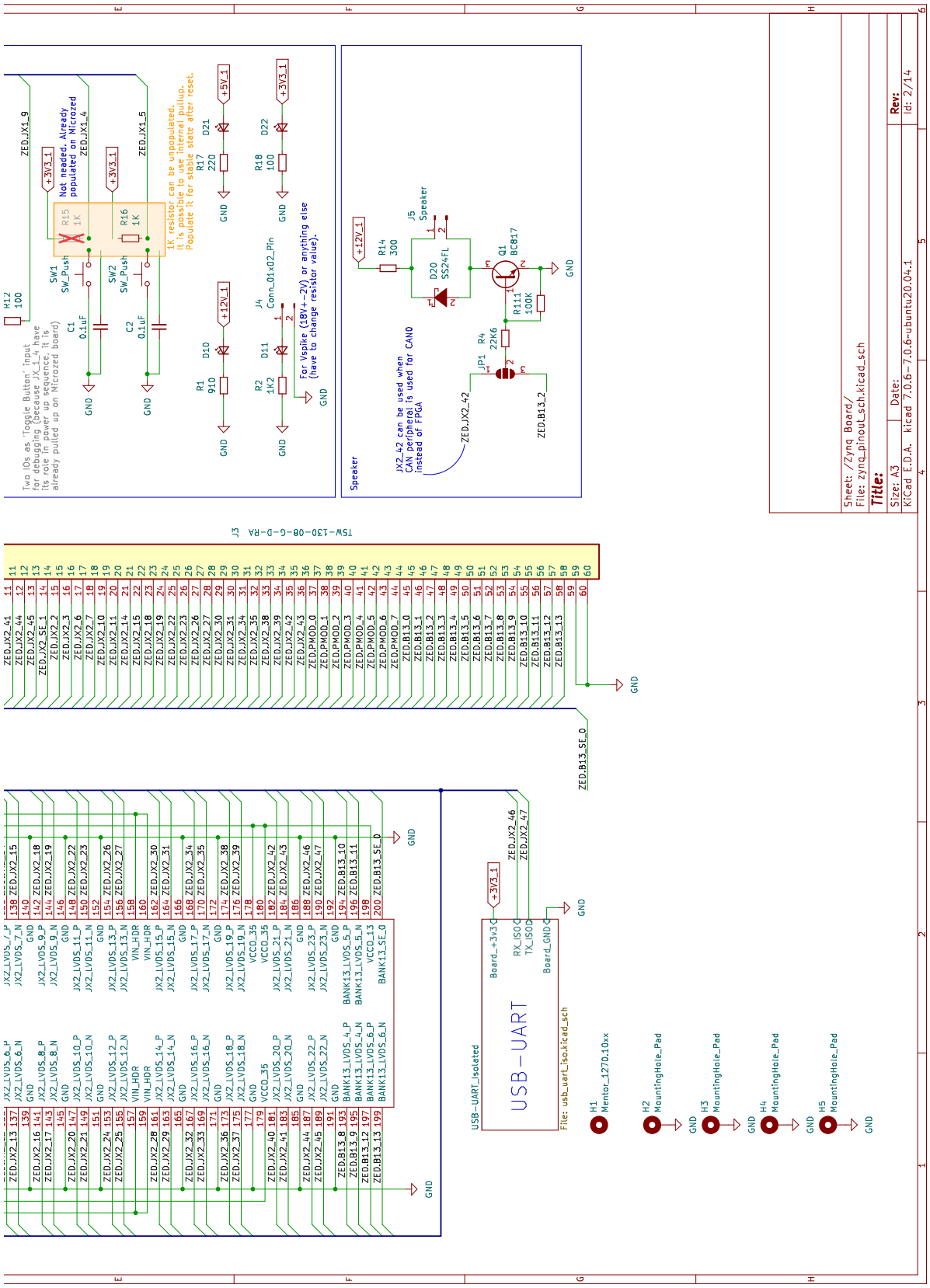


Figure C.3: MicroZed

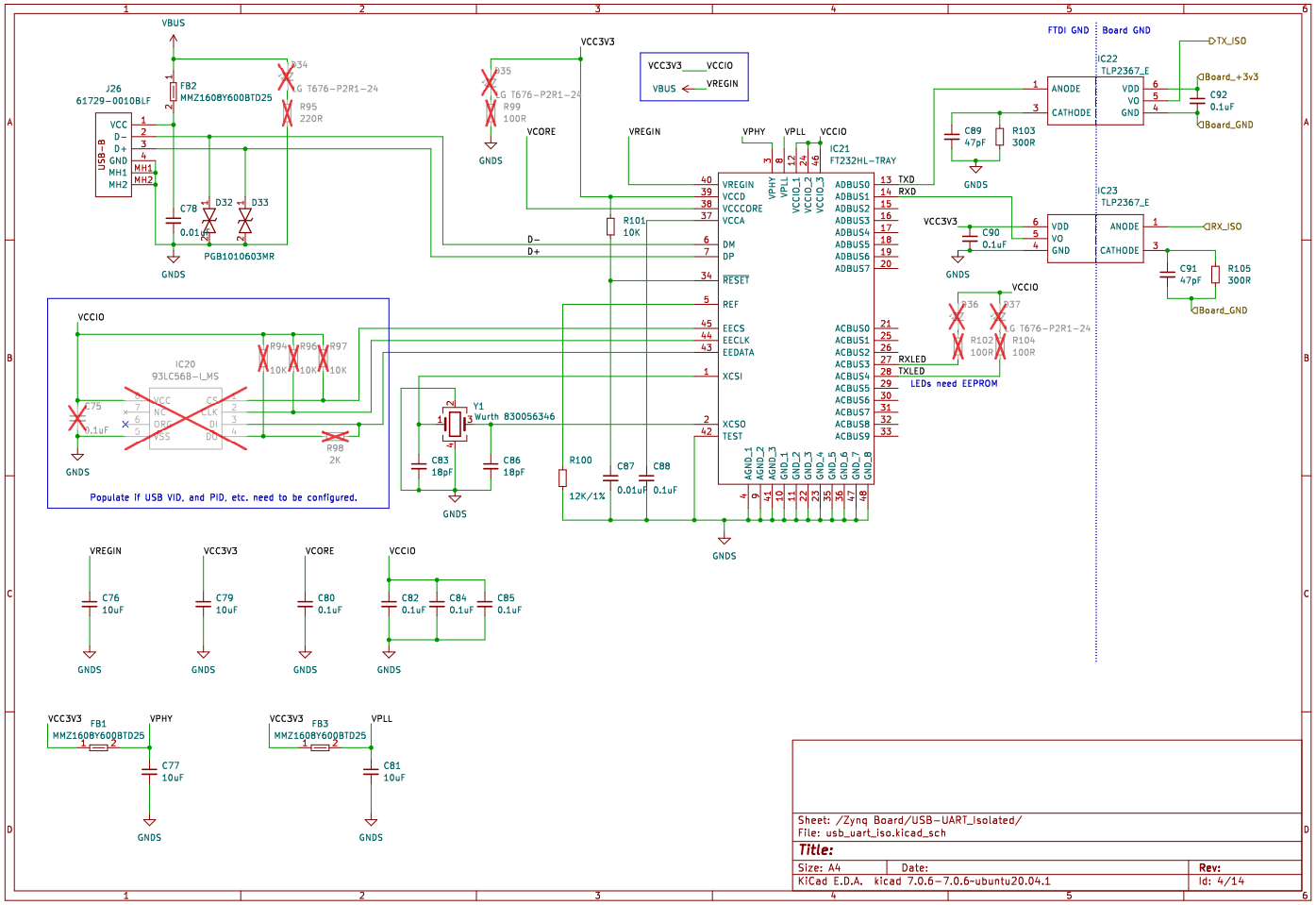


Figure C.4: USB-UART for MicroZed

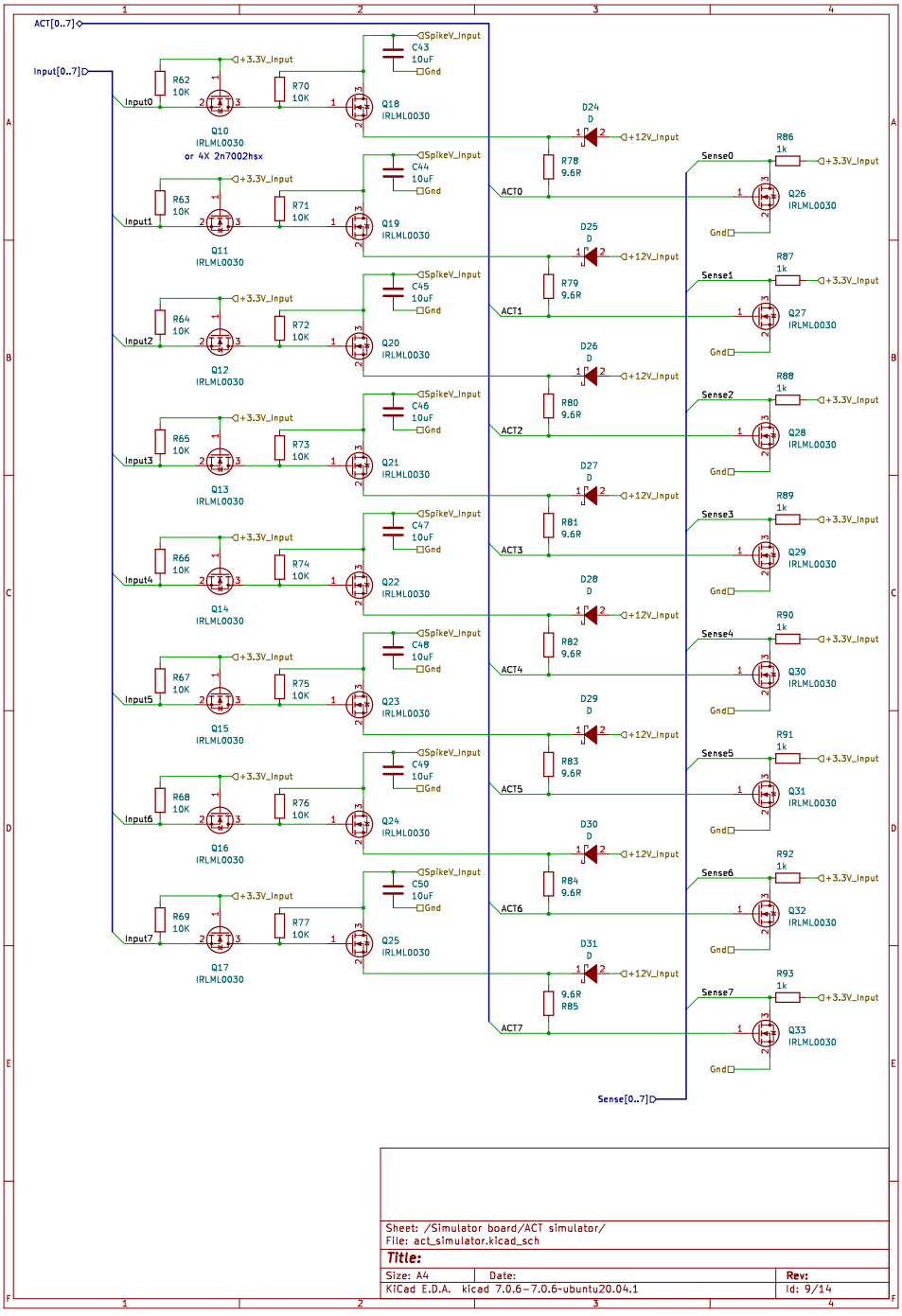
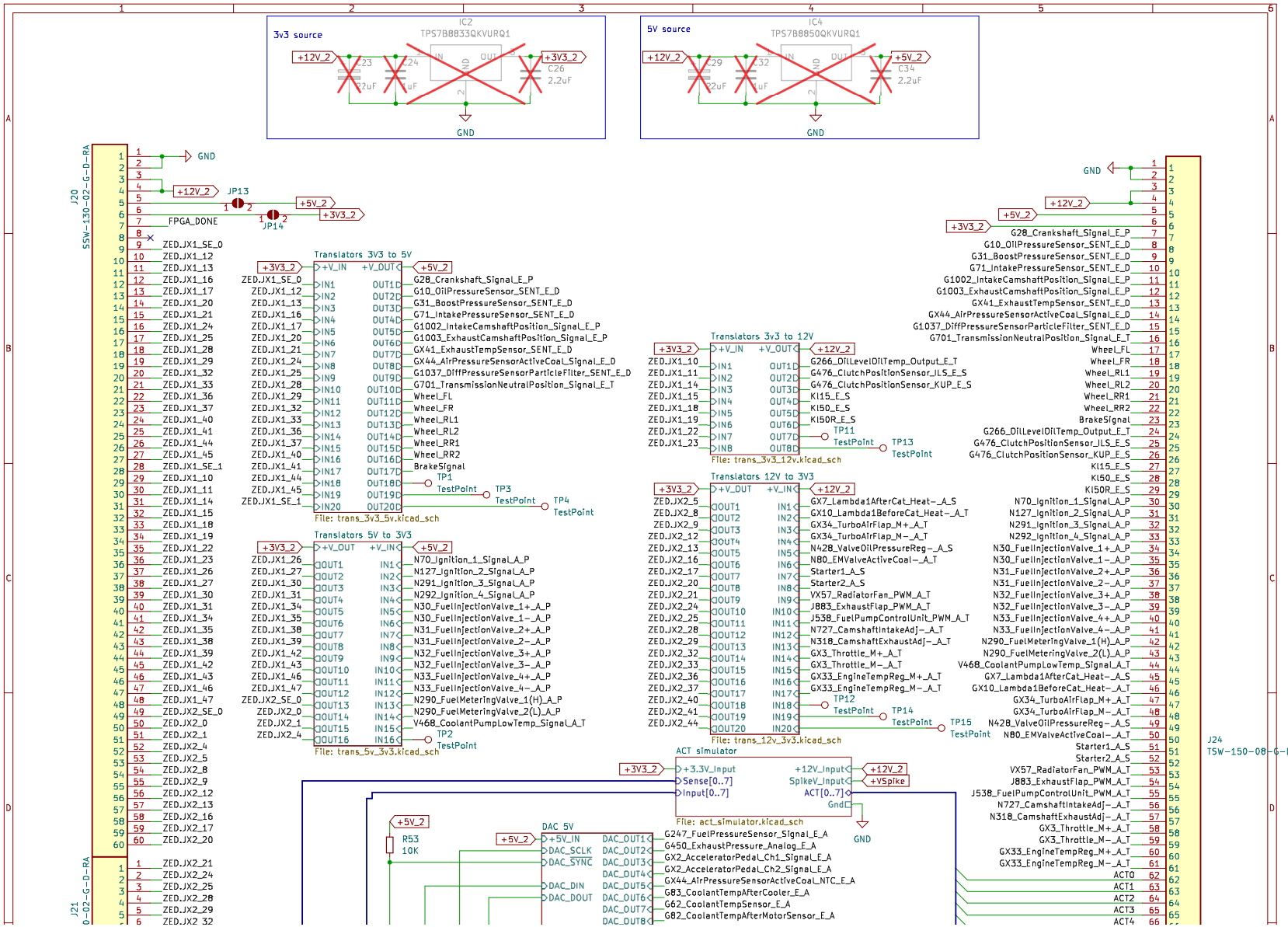


Figure C.5: Active Cylinder Management Simulator


3v3 source
IC2
TPS7B8833QKVURQ1
+12V_2
+3V3_2
C23
22uF
C24
1uF
C26
2.2uF
GND
5V source
IC4
TPS7B8850QKVURQ1
+5V_2
C29
22uF
C32
1uF
C34
2.2uF
J20
SSW-130-02-G-D-RA
GND
JP13
JP14
FPGA_DONE
ZED.JX1_SE_0
ZED.JX1_12
ZED.JX1_13
ZED.JX1_16
ZED.JX1_17
ZED.JX1_20
ZED.JX1_21
ZED.JX1_24
ZED.JX1_25
ZED.JX1_28
ZED.JX1_29
ZED.JX1_32
ZED.JX1_33
ZED.JX1_36
ZED.JX1_37
ZED.JX1_40
ZED.JX1_41
ZED.JX1_44
ZED.JX1_45
ZED.JX1_SE_1
ZED.JX1_10
ZED.JX1_11
ZED.JX1_14
ZED.JX1_15
ZED.JX1_18
ZED.JX1_19
ZED.JX1_22
ZED.JX1_23
ZED.JX1_26
ZED.JX1_27
ZED.JX1_30
ZED.JX1_31
ZED.JX1_34
ZED.JX1_35
ZED.JX1_38
ZED.JX1_39
ZED.JX1_42
ZED.JX1_43
ZED.JX1_46
ZED.JX1_47
ZED.JX2_SE_0
ZED.JX2_0
ZED.JX2_1
ZED.JX2_4
ZED.JX2_5
ZED.JX2_8
ZED.JX2_9
ZED.JX2_12
ZED.JX2_13
ZED.JX2_16
ZED.JX2_17
ZED.JX2_20
J21
0-02-G-D-RA
ZED.JX2_21
ZED.JX2_24
ZED.JX2_25
ZED.JX2_28
ZED.JX2_29
ZED.JX2_32
Translators 3V3 to 5V
+V_IN
+V_OUT
G28_Crankshaft_Signal_E_P
G10_OilPressureSensor_SENT_E_D
G31_BoostPressureSensor_SENT_E_D
G71_IntakePressureSensor_SENT_E_D
G1002_IntakeCamshaftPosition_Signal_E_P
G1003_ExhaustCamshaftPosition_Signal_E_P
GX41_ExhaustTempSensor_SENT_E_D
GX44_AirPressureSensorActiveCoal_Signal_E_D
G1037_DiffPressureSensorParticleFilter_SENT_E_D
G701_TransmissionNeutralPosition_Signal_E_T
Wheel_FL
Wheel_FR
Wheel_RL1
Wheel_RL2
Wheel_RR1
Wheel_RR2
BrakeSignal
TP1
TP2
TP3
TP4
TestPoint
File: trans_3v3_5v.kicad_sch
Translators 5V to 3V3
N70_Ignition_1_Signal_A_P
N127_Ignition_2_Signal_A_P
N291_Ignition_3_Signal_A_P
N292_Ignition_4_Signal_A_P
N30_FuelInjectionValve_1+_A_P
N30_FuelInjectionValve_1-_A_P
N31_FuelInjectionValve_2+_A_P
N31_FuelInjectionValve_2-_A_P
N32_FuelInjectionValve_3+_A_P
N32_FuelInjectionValve_3-_A_P
N33_FuelInjectionValve_4+_A_P
N33_FuelInjectionValve_4-_A_P
N290_FuelMeteringValve_1(H)_A_P
N290_FuelMeteringValve_2(L)_A_P
V468_CoolantPumpLowTemp_Signal_A_T
Translators 3v3 to 12V
G266_OilLevelOilTemp_Output_E_T
G476_ClutchPositionSensor_ILS_E_S
G476_ClutchPositionSensor_KUP_E_S
KI15_E_S
KI50_E_S
KI50R_E_S
TP11
TP13
File: trans_3v3_12v.kicad_sch
Translators 12V to 3V3
GX7_Lambda1AfterCat_Heat-_A_S
GX10_Lambda1BeforeCat_Heat-_A_T
GX34_TurboAirFlap_M+_A_T
GX34_TurboAirFlap_M-_A_T
N428_ValveOilPressureReg-_A_S
N80_EMValveActiveCoal-_A_T
Starter1_A_S
Starter2_A_S
VX57_RadiatorFan_PWM_A_T
J883_ExhaustFlap_PWM_A_T
J538_FuelPumpControlUnit_PWM_A_T
N727_CamshaftIntakeAdj-_A_T
N318_CamshaftExhaustAdj-_A_T
GX3_Throttle_M+_A_T
GX3_Throttle_M-_A_T
GX33_EngineTempReg_M+_A_T
GX33_EngineTempReg_M-_A_T
TP12
TP14
TP15
File: trans_12v_3v3.kicad_sch
ACT simulator
+3.3V_Input
Sense[0..7]
Input[0..7]
+12V_Input
SpikeV_Input
ACT[0..7]
Gnd
+VSpike
File: act_simulator.kicad_sch
R53
10K
DAC 5V
+5V_IN
DAC_SCLK
DAC_SYNC
DAC_DIN
DAC_DOUT
DAC_OUT1
DAC_OUT2
DAC_OUT3
DAC_OUT4
DAC_OUT5
DAC_OUT6
DAC_OUT7
DAC_OUT8
G247_FuelPressureSensor_Signal_E_A
G450_ExhaustPressure_Analog_E_A
GX2_AcceleratorPedal_Ch1_Signal_E_A
GX2_AcceleratorPedal_Ch2_Signal_E_A
GX44_AirPressureSensorActiveCoal_NTC_E_A
G83_CoolantTempAfterCooler_E_A
G62_CoolantTempSensor_E_A
G82_CoolantTempAfterMotorSensor_E_A
J24
TSW-150-08-G-D-RA
ACT0
ACT1
ACT2
ACT3
ACT4

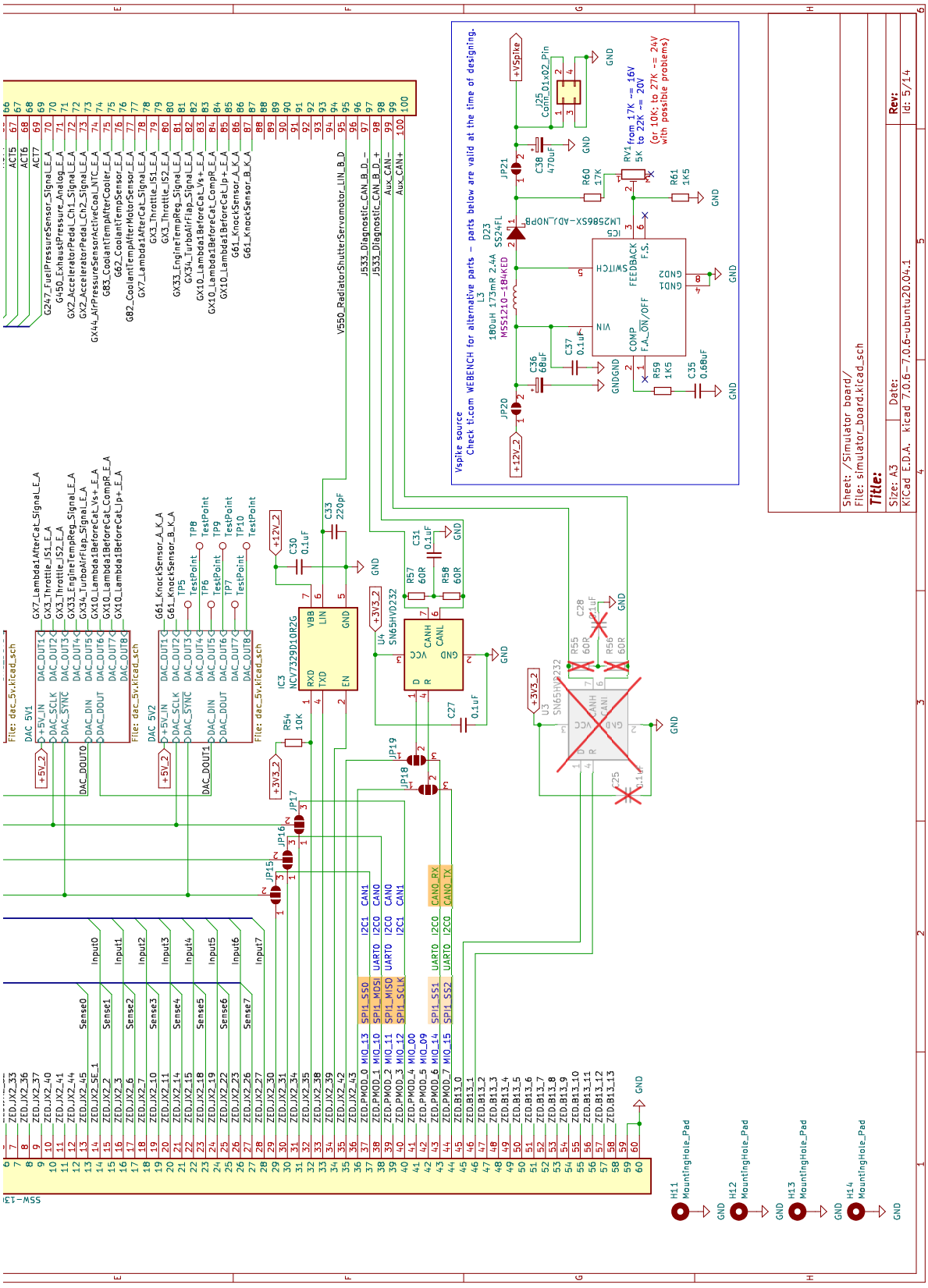
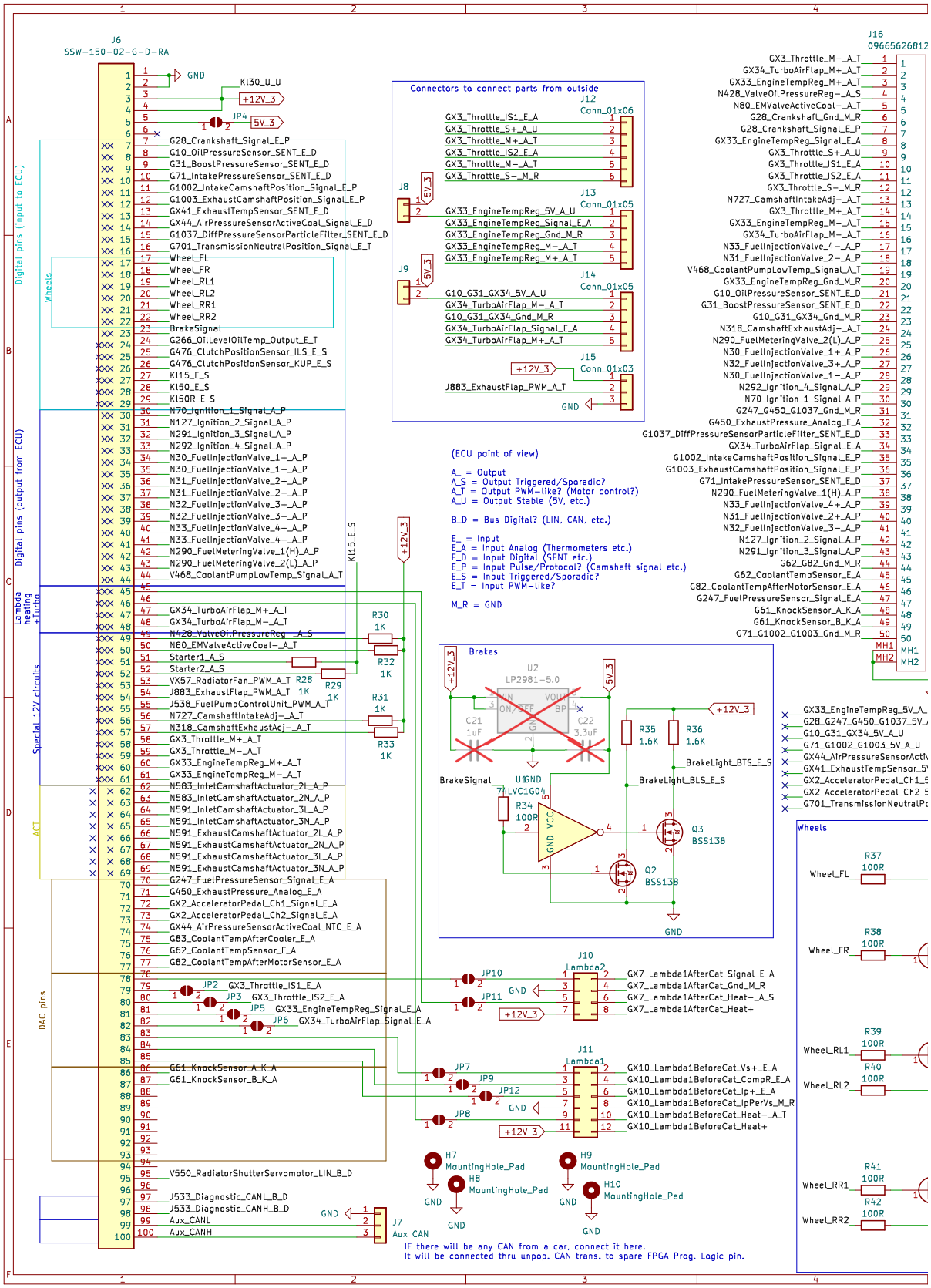


Figure C.6: Simulator board


J6
SSW-150-02-G-D-RA
J16
09665626812
Connectors to connect parts from outside
J12
Conn_01x06
J13
Conn_01x05
J14
Conn_01x05
J15
Conn_01x03
J8
J9
(ECU point of view)
A_ = Output
A_S = Output Triggered/Sporadic?
A_T = Output PWM-like? (Motor control?)
A_U = Output Stable (5V, etc.)
B_D = Bus Digital? (LIN, CAN, etc.)
E_ = Input
E_A = Input Analog (Thermometers etc.)
E_D = Input Digital (SENT etc.)
E_P = Input Pulse/Protocol? (Camshaft signal etc.)
E_S = Input Triggered/Sporadic?
E_T = Input PWM-like?
M_R = GND
Brakes
U2
LP2981-5.0
U1
74LVC1G04
Q3
BSS138
Q2
BSS138
Wheels
J10
Lambda2
J11
Lambda1
J7
Aux CAN
IF there will be any CAN from a car, connect it here.
It will be connected thru unpop. CAN trans. to spare FPGA Prog. Logic pin.
Digital pins (input to ECU)
Digital pins (output from ECU)
Special 12V circuits
ACT
DAC pins
Lambda heating +Turbo

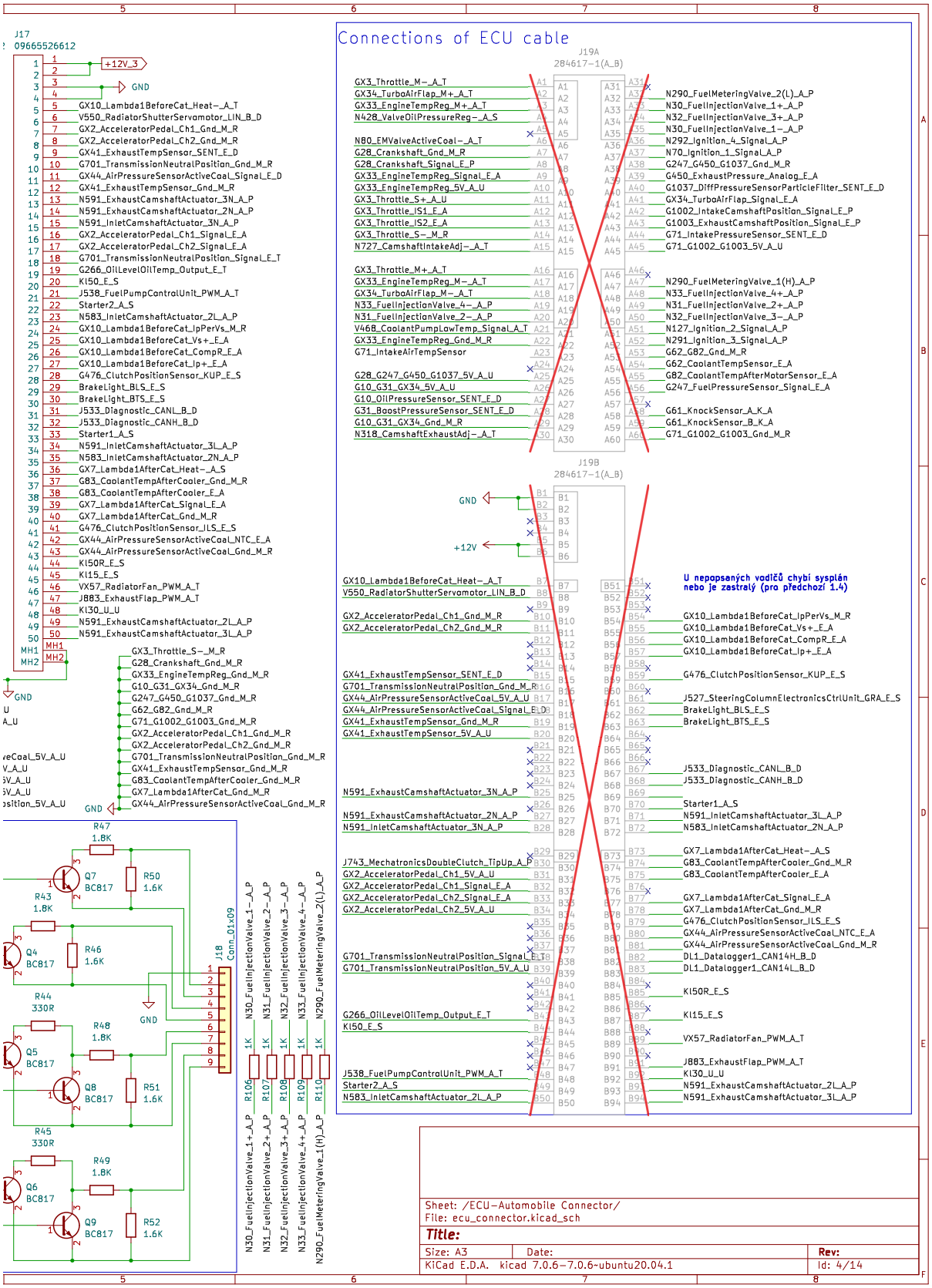


Figure C.7: ECU-Automobile Connector

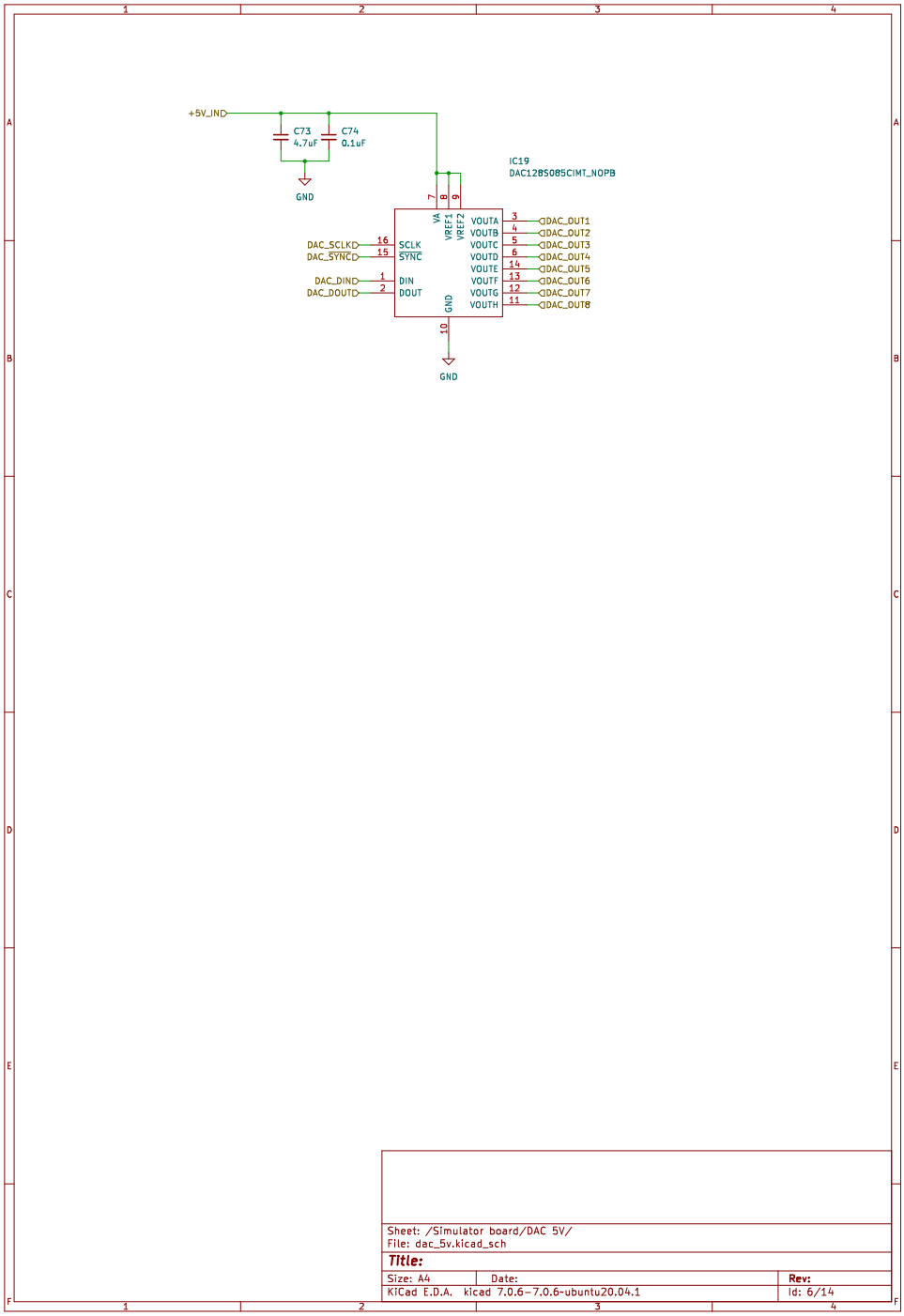


Figure C.8: Digital-to-Analog Converter 5 V (part 1)

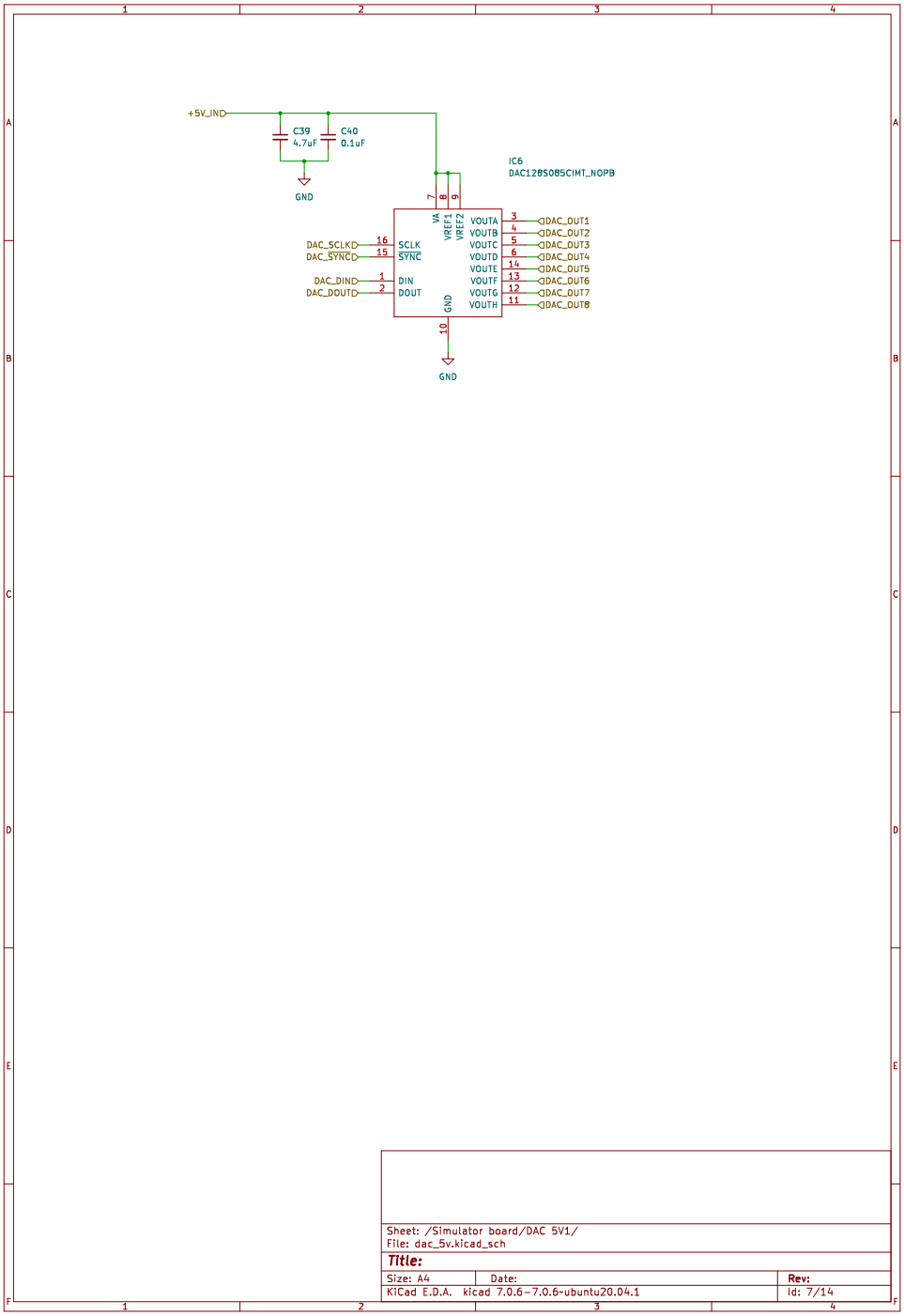


Figure C.9: Digital-to-Analog Converter 5 V (part 2)

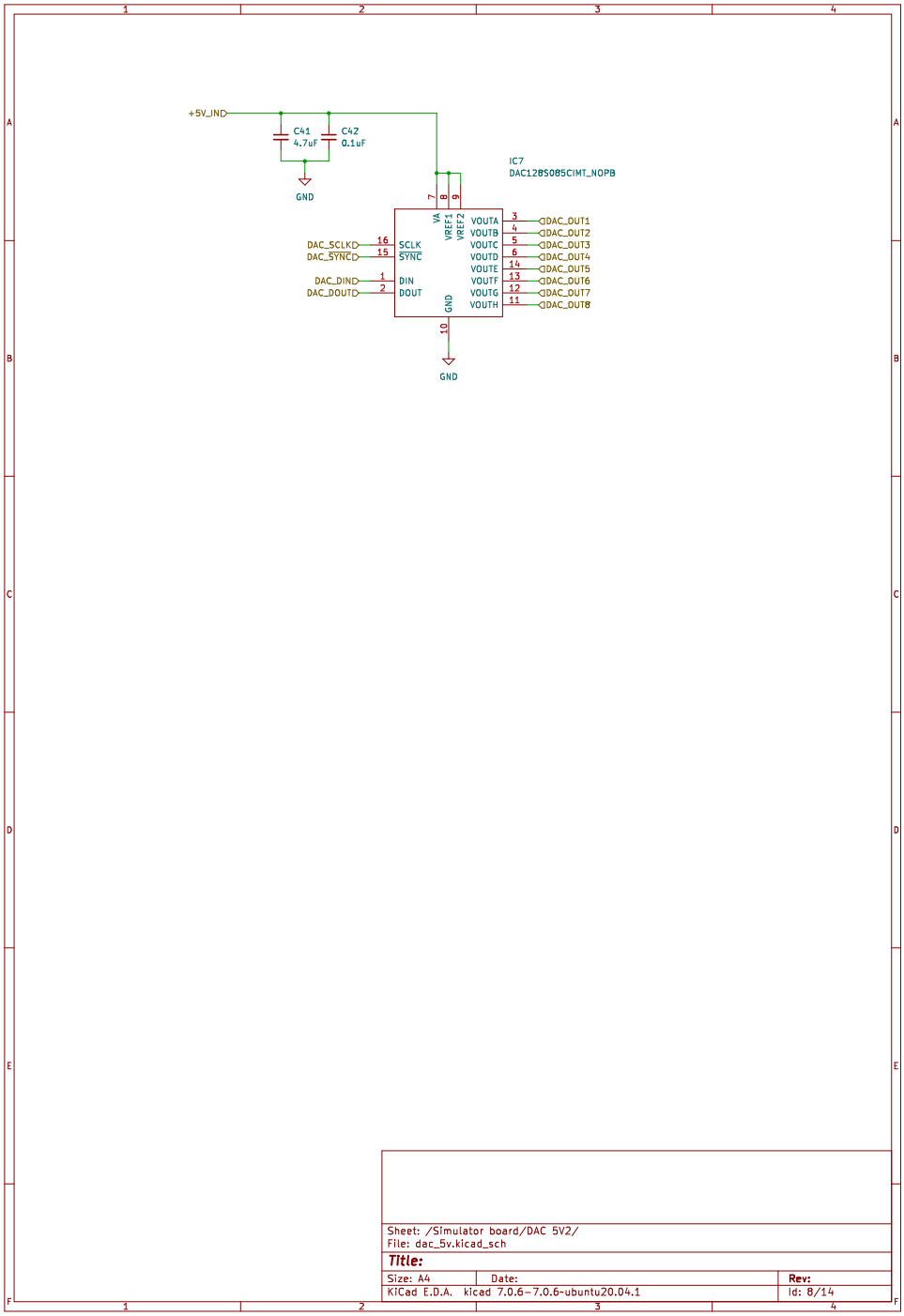


Figure C.10: Digital-to-Analog Converter 5 V (part 3)

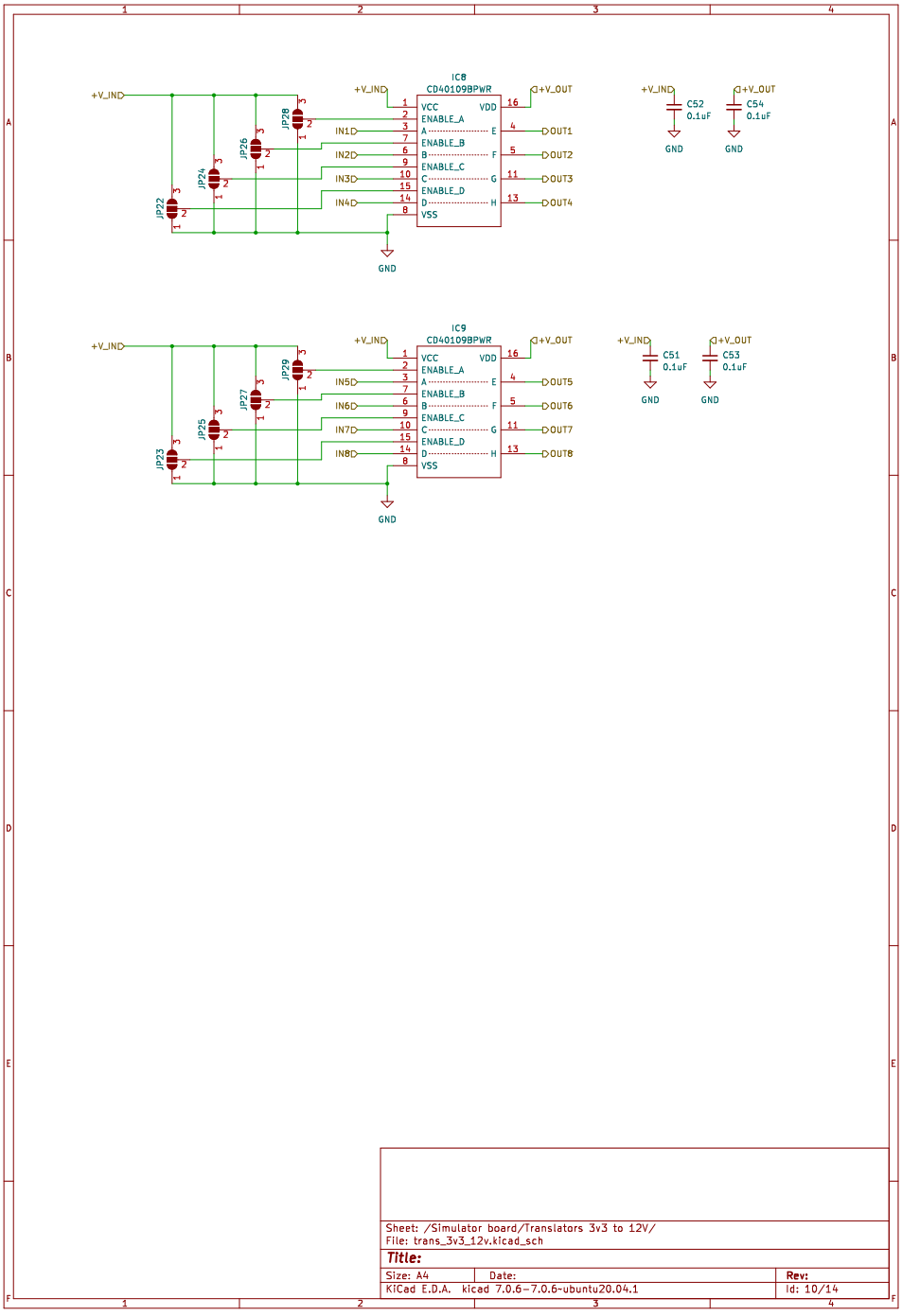


Figure C.11: Translators from 3.3 V to 12 V

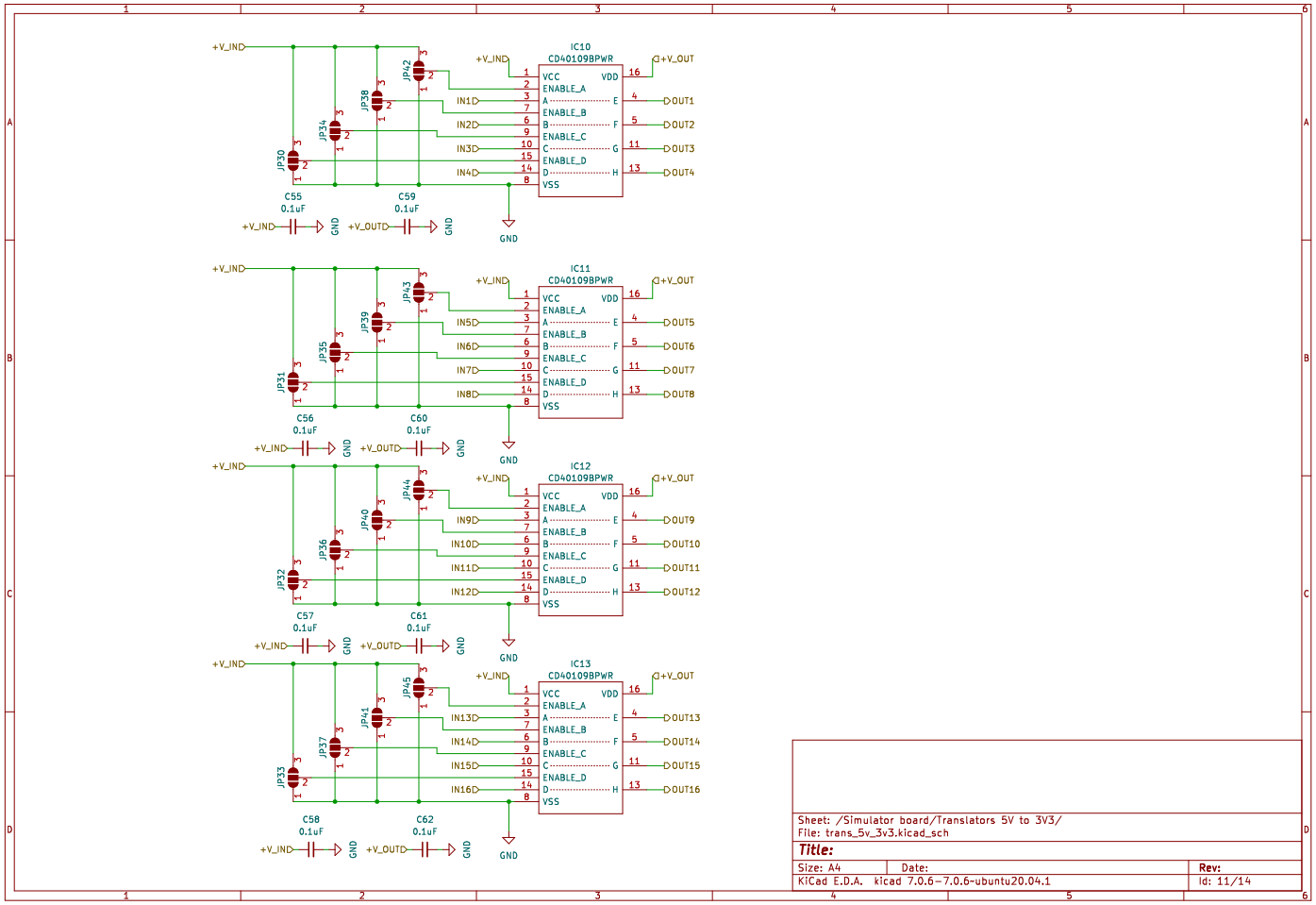


Figure C.12: Translators 5 V to 3.3 V

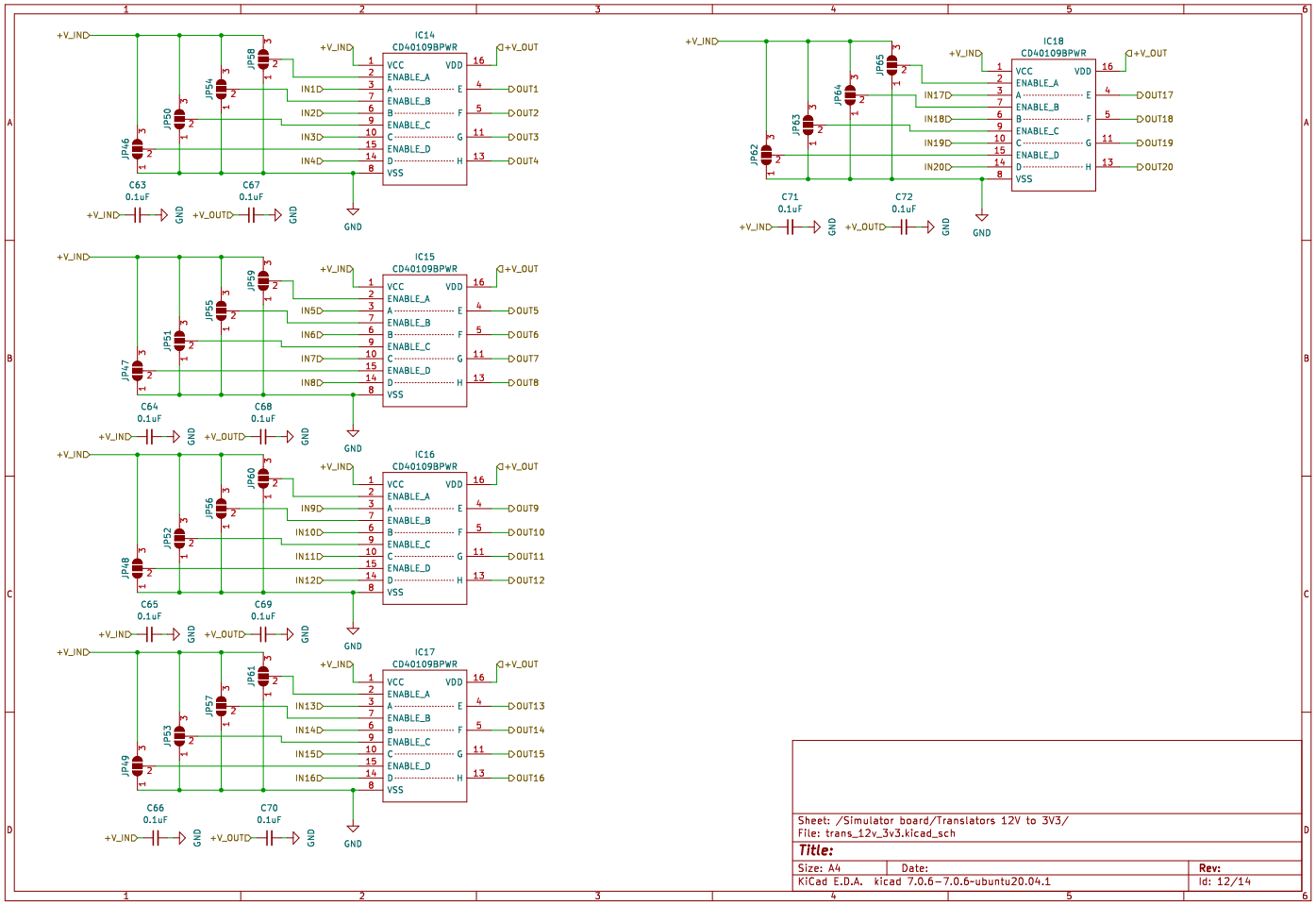


Figure C.13: Translators 12 V to 3.3 V

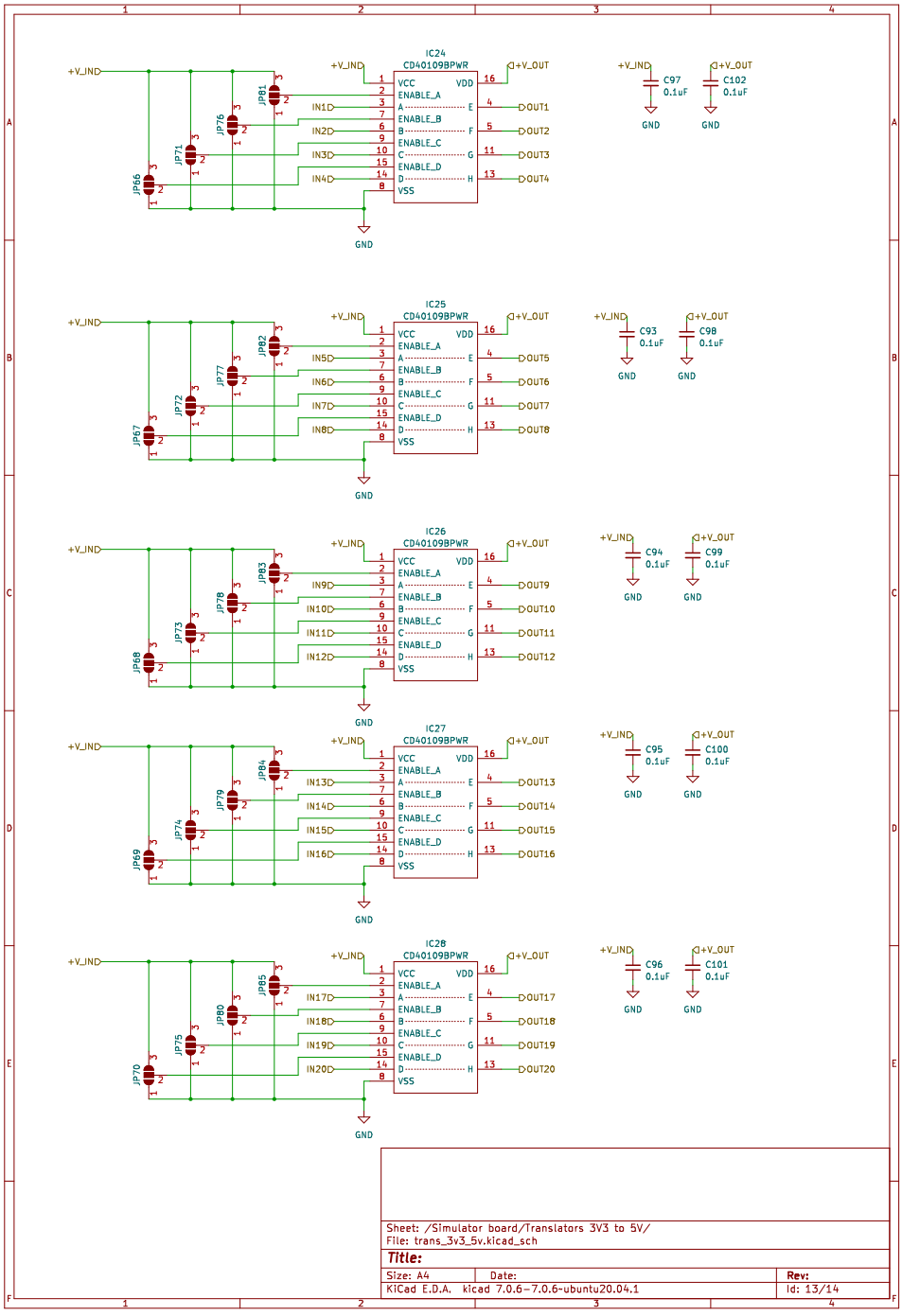


Figure C.14: Translators 3.3 V to 5 V