Michael Moitzi, BSc

Noise Sensitivity Measurement on Smart Card Devices

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Electrical Engineering

submitted to

Graz University of Technology

Supervisor

Ass.Prof. Dipl.-Ing. Dr.techn. Peter Söser

Institute of Electronics (IFE)

In Cooperation with NXP Semiconductors Austria GmbH
Supervisor: Dipl.-Ing.(FH) Ambros Weissenbacher

Graz, May 2019

Abstract

Stability of smart card devices is constantly challenged. With smaller semiconductor manufacturing processes the influence of noise and crosstalk on semiconductor devices increases. Additionally security measures implemented on the secure device are increasing the sensitivity to environmental influences. In order to cover investigations of sensitivity to noise of smart card devices, a suitable environment is needed.
The goal of this master thesis is to develop an external hardware solution, which adds the possibility to perform noise sensitivity measurement of smart card devices to the existing automatic test equipment. In order to define the stimuli requirements an evaluation phase is needed. Within this phase, all necessary signal parameters should be specified. After that an external hardware should be designed, which is able to generate signals specified in the evaluation phase. Additionally this external hardware should be integrated into the existing test environment. Therefore a multi-point measurement has to be integrated on the tester. This measurement is used to identify a reset of the smart card device. Finally this external hardware should be able to add the option to perform noise sensitivity measurements in production environment. This enables the possibility to analyse a high quantity of smart card devices.

Kurzfassung

Immer kleiner werdende Halbleiter-Fertigungsprozesse, neue Fertigungstechnologien und erhöhte Sicherheitsmaßnahmen beeinflussen die Stabilität von Smart Card Devices in deren Anwendungen. Um diese Sensibilität, von bereits als gut deklarierten Teilen, schon in der Produktionsumgebung erfassen zu können, wird eine geeignete Umgebung benötigt.
Das Ziel dieser Masterarbeit ist es, eine geeignete Hardware zu entwickeln, welche die Untersuchung von Noise Sensitivity von Smart Card Produkten am bestehenden Tester erlaubt. Um die Hardware näher zu spezifizieren, soll das Resetverhalten eines Smart Card Devices im Rahmen einer Evaluierungsphase untersucht werden. Innerhalb der Evaluierungsphase sollen Parameter gefunden werden, welche einen Reset auslösen können. Mit Hilfe dieser Parameter soll nun eine externe Hardware entwickelt werden, welche in der Lage ist geeignete Signale zu erzeugen. Um einen möglichen Reset des Devices erkennbar zu machen, soll am vorhandenen Testsystem eine Mehrpunkt-Strommessung implementiert werden. Das gesamte Setup soll dann in einer Produktionsumgebung Devices in größeren Stückzahlen überprüfen.

Acknowledgment

I want to thank Dipl.-Ing.(FH) Ambros Weissenbacher from NXP Semiconductors Austria and Ass.Prof. Dipl.-Ing, Dr.techn. Peter Söser from the Institute of Electronics (IFE) at Technical University Graz for their continious support during the work on this master's thesis. Without your helping answers to my questions and your fitting solutions to my challenges this thesis would not have been possible. I also want to thank my family for their permanent support during my education. For me it is easy to say that without them gaining an academic degree would not have been possible.

Contents

List of Figures

List of Tables

Chapter 1

Motivation

Increasing fields of applications raise the need for smart card devices to adapt to multiple possible working conditions. These secure systems are designed to operate with different supply voltage levels. Cost-saving strategies and power requirements are driving this semiconductor area. Advanced semiconductor manufacturing processes are used to fulfil these requirements. Upside of enhanced technologies is that the overall area per device decreases. The number of devices per wafer is in inverse relation to cost per die. Downside of this continuous development is that smaller dies are more sensitive to noise and crosstalk.

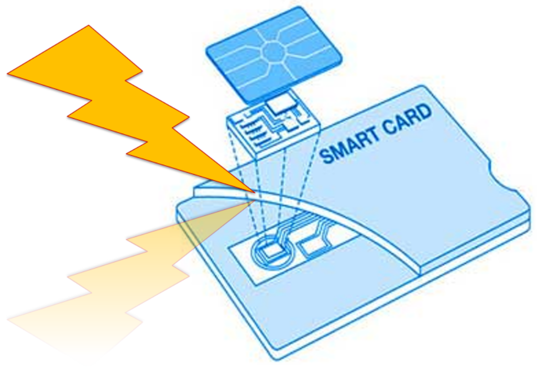


Figure 1.1: Illustration of electrically stressed smart card device.

In addition to more sensitivity to noise and crosstalk caused by smaller chip area smart card devices are under constant strain of attacks. These attacks are trying to cause malfunctions of smart card devices. They either try to gain information on software level or try to attack the device on hardware level. An attack on physical level might be a manipulation of device input or output pin. Equally the goal of any attack is to acquire secret information about the cryptography of the secure device. Leakage of secret data, like keys, would cause the total opposite of the basic use case, insecurity in application. To continue providing trust-worthy secure devices measures are being applied to increase the robustness against attacks. These countermeasures against attacks are mostly sensors implemented and software running on the silicon. To get more information about the stability of smart card devices during attacks different laboratory setups are made. Due to low quantity through-put and high amount of resources needed to perform these tests an appropriate alternative needs to be found.

Goal of this master's thesis is to develop an external hardware which adds the possibility to perform noise sensitivity measurements on smart card devices with the use of automatic test equipment. Multi-point current measurement should be implemented to identify a reset of the device under test. To understand the devices behaviour when a reset occurs, the reset behaviour of such devices has to be determined during an evaluation phase. Within this stage the definition of the stimuli is specified. To make a reset observable a multi-point current measurement has to be implemented on the tester. Subsequently the external hardware needs to be integrated within the existing tester. To allow investigations, the external hardware should easily be added into the existing production environment. This means that the overall size of the hardware is limited.

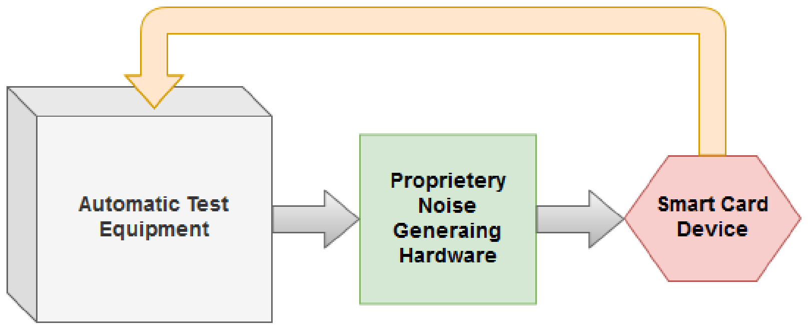


Figure 1.2: Overall system concept of noise sensitivity measurement of smart card devices.

Beginning with chapter 2 all specific theory needed to accomplish the functional specification will be covered. There will be more information regarding smart card devices, their specifications, countermeasures against security attacks, the tester and the theoretical approach on solving performance requirements.

With the knowledge of chapter 2 it was possible to create several potential solutions. Advantages and disadvantages of these possible solutions will be discussed in chapter 3.

In chapter 4 the best potential solutions were combined and integrated into a fully functional unit. Within this chapter the system and all the necessary implementations and modifications will be described.

Ending with chapter 5 a conclusion and outlook of the project will be given. Possible improvements or further use cases will be pointed out in this last chapter.

Chapter 2

Theory

Smart cards are complex systems running multiple applications on different versions of silicon. For that reason it is very important to understand the behaviour of smart card devices and their properties as well as standardisations. Within this chapter smart card devices, their properties and testing will be covered as well as all additional theory necessary to fulfil the requirements of this master's thesis.

2.1 What is a Smart Card Device

A smart card is known to be a credit card sized plastic card. Usually a smart card contains a microprocessor. This processing unit is capable of running a variety of cryptographic software applications. A so called smart card reader is used to enable communication and secure transactions with the smart card device. This reader also powers the cryptographic device. Most of present smart cards use a contact interface, while others use a contactless interface. Some smart cards are using both interfaces. These cryptographic devices are mainly used to authentication and secure transactions like access authorisation systems or banking solutions. To grant secure application the embedded cryptographic processor is able to run different cryptographic algorithms. These algorithms are devided in symmetrical and asymetrical algorithms. On the one hand Data Encryption Standard (DES) and Advanced Encryption Standard (AES) are examples for symmetrical algorithms. On the other hand Rivest, Shamir und Adleman (RSA) is an example for an asymmetrical algorithm. But even with this high amount of security, special protocols and cryptographic measures implemented on smart card devices, they are not fully trusted as a secure source of information. This is caused by the fact that secret data might leak through side-channels, where power analysis attacks are able to collect and analyse this data. More details about side-channel attacks and countermeasures against such attacks will be covered in chapter 2.5. [MDS02] [LWP$^+$08] [MAK15] [MA10]

2.1.1 ISO7816

The International Organization for Standardization (ISO) has defined the standard ISO7816. Within this standardization smart card dimensions and their interfaces, commands and protocols are defined. Most important for this thesis is the definition of contacts and pinning of a smart card device, which is covered within ISO7816-3. In part 2 of this ISO-standard dimensions and location of the contacts on the plastic card are covered. Such parts as mechanical stress and robustness against X-rays, UV-light and temperature are covered within part 1 of the standard. Since this thesis focuses on electrical robustness, ISO7816-3 will be covered in more detail. While ISO7816 focuses on operating smart cards via the contact interface, there is an addition to this standard. ISO14443 focuses on contactless proximity objects, meaning that this standardisation defines contactless RF communication for smart card devices. [Int11] [Int07] [Int06] [RE02]

2.1.2 Voltage Classes

Power requirements and cost-saving strategies are the main driving factor in the smart card industry. To fulfil these requirements, enhanced semiconductor processes are used. Enhancing semiconductor processes result in smaller supply voltages, which create the need for multiple possible voltage classes. These voltage classes are defined in ISO7816-3. Currently there are three voltage classes standardised within ISO7816. More details can be seen in table 2.1. It is clearly stated, that energy consumption of smart card devices is decreasing with an decrease of supply voltage. [RE02] [Int06]

Voltage class	Voltage	Maximum current
Class A	$5V$ ($\pm 10\%$)	$60mA$
Class B	$3V$ ($\pm 10\%$)	$50mA$
Class C	$1.8V$ ($\pm 10\%$)	$30mA$

Table 2.1: Voltage classes defined in ISO7816-3

After powering a smart card device, it transmits a so called answer to reset (ATR) response to the reader. This response contains several informations like maximum frequency, maximum current and supported voltage classes. Therefore it is possible to alter the supply voltage to match the smart cards preferred voltage class. This makes it possible to operate modern smart card devices with older readers and vice versa. [Int06]

2.1.3 Smart Card Packages

Most important for a smart card device is its package. This enclosure connects the fragile silicon die to the outer world. Therefore it defines available interfaces to the device. There are several different packages available. Packages can be differentiated in SIM card packages and non SIM card packages. An example for a non SIM card package is the SOT500 package, which is used for contactless communication. This means, that this package has just two pins available, which are connecting the antenna to the device. Specification of contactless communication can be found in ISO14443. On the other hand SIM card packages are typically available with 6, 8 or even 10 pins. These modules are either able of interfacing via a contact interface or a combination of contact and contactless interface. Smart cards combining these two interfaces are called dual-interface devices. Compared to a smart card using a contact interface, dual-interface devices are equipped with two additional pins. These two pads are used to connect an antenna to the device. In table 2.2 the ISO7816 pin assignment of smart card devices is shown. As already stated SIM card modules have a different pin count. While ISO7816-3 defines 8 pins for contacting purpose, smart card device manufactures are varying this number. Pins $AUX1$ and $AUX2$ are provided for additional product specific interfaces, like USB or I^{2}C. If additional interfaces are not necessary, it is common to exclude pins $AUX1$ and $AUX2$ resulting in a smaller package size. To provide a contactless interface, two additional pins on the backside of the package are implemented. Smart card modules are typically handled in 35mm reels by reel-to-reel handlers. [RE02] [Int07] [Int06]

VCC		GND
RST		SPU
CLK		I/O
AUX1		AUX2

Table 2.2: Electrical assignment of smart card contacts according to ISO7816

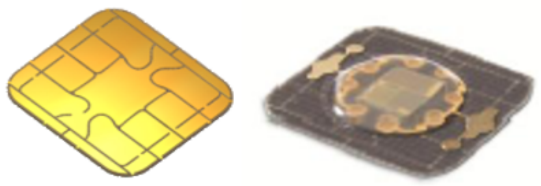


Figure 2.1: Dual-interface package of a smart card device.

Pin Name	Definition
VCC (C1)	Supply Power Contact
RST (C2)	Reset Contact
CLK (C3)	Clock Contact
GND (C5)	Ground Contact
SPU (C6)	Standard or Proprietary Use Contact
I/O (C7)	Input/Output Contact
AUX1 (C4)	Auxiliary 1
AUX2 (C8)	Auxiliary 2

Table 2.3: Electrical assignment of smart card contacts according to ISO7816

2.1.4 Equivalent Circuit Diagram

For simulation purposes covered in chapter 4 an equivalent circuit diagram of a smart card device is necessary. In figure 2.2 a simplification of a smart card device in respect to V_{DD} pin is illustrated. This model is used to simulate the behaviour of the generated signal when applied to a smart card device. In table 2.4 parameters for this model are given. Care must be taken when using this model in simulation. Since this is just an approximation of impedance of the device, parametric values of the equivalent circuit might vary. Furthermore smart cards impedance depends on several conditions, like current operating state, running cryptographic algorithms and voltage supply.

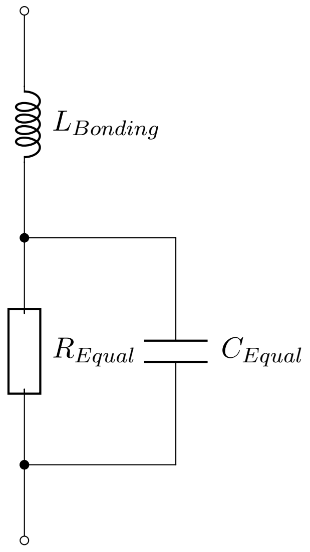


Figure 2.2: Equivalent circuit of a common smart card device in respect to V_{DD} pin.

Name	Value
R_{Equal}	$1k\Omega$
C_{Equal}	$200pF$
$L_{Bonding}$	$10nH$

Table 2.4: Equivalent circuit diagram parameters

2.2 Tester and Tools

Automatic test equipment (ATE) is becoming more and more important in the semiconductor industry. With the need of high-quantity and high-quality devices, the importance of testing semiconductor products increases. For that reason there are various possible instrumentations of ATE systems on the market. Each of these configurations have their advantages and disadvantages, resulting in very specific areas of application. Overall there are two big fields of semiconductor testing. On one side there is wafer test, which covers high-quantity testing with high multi-site numbers. In general wafer test focuses on finding defects on silicon. On the other side there is package test. In package test mostly applicative and functional tests are performed. For the reason of increased crosstalk, stressing devices on wafer level is not possible. Therefore noise sensitivity measurement is performed at package level, because a packaged device is closer to the final application of the smart card device. For that reason the package tester will be described in more detail.

2.2.1 Smart Card Tester

The smart card tester used at NXP semiconductors is the SPEA CT1000. The CT1000 series is designed to test smart cards (contact, contactless and combined) and any kind of HF and UHF product. Protocol based functionality of this tester is highlighted. Parametric and functional requirements for measurements of contact (ISO7816) and contactless (LF, HF and UHF) devices can easily be achieved with these highly configurable and scalable testers. A PC controls the system. On Software side the tester is currently powered by AtosCT system software that provides all necessary functions to program, execute and debug all kinds of tests. The SPEA CT1000 tester is organised in several racks. In these racks it is possible to install different instrument slots like for LF/HF, UHF, ISO7816 and memory blocks. Each cluster consists of 5 slots. [SPEb]

2.2.2 Reel-to-Reel Handler

As already broached in section 2.1.3, smart card devices are handled with reel-to-reel handlers. The arrangement of packaged dies on a 35mm reel is mostly depending on the package used. With typical smart card packages a 35mm reel provides two rows of devices. In NXP production environment a product with such a package would be tested with a multi-site number of 24. This means that with 24 devices tested in parallel, multi-site efficiency increases. With decreasing package size, the number of rows per reel raises. This is the case with smaller contactless packages, where three or even four rows on a 35mm reel are possible. Depending on probe card and tester used, the multi-site number increases correspondingly.

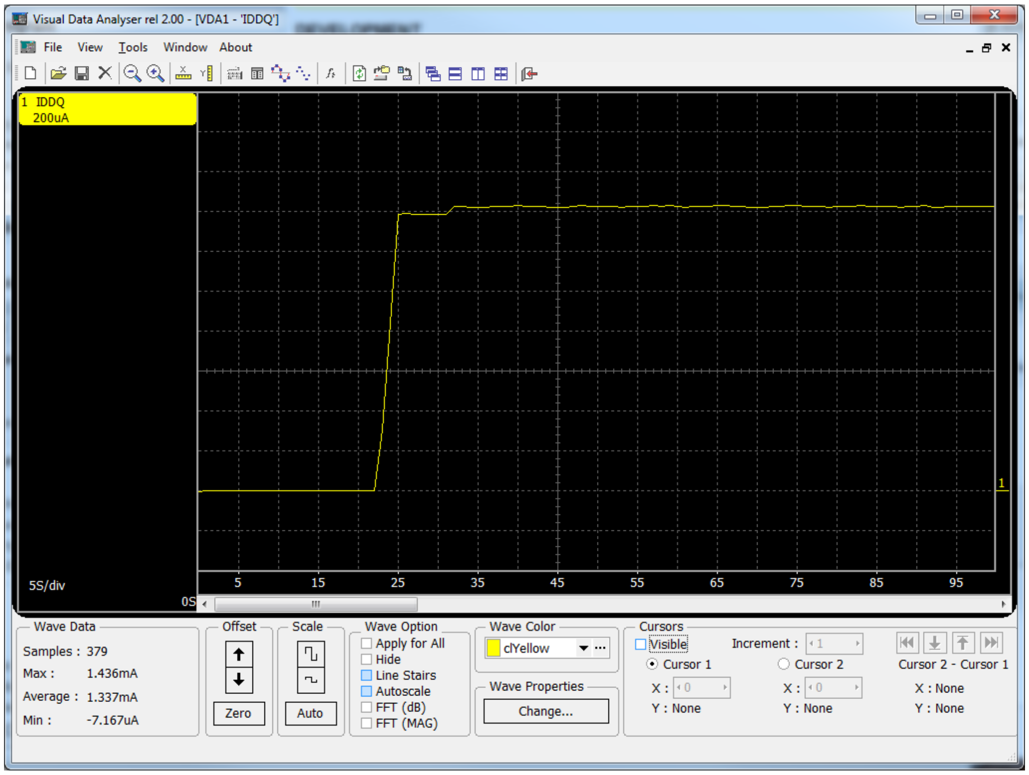
2.2.3 Test Plan Editor

In order to setup a test, a so called test plan is used. This test plan contains all test steps with all necessary data to perform these tests. To modify and visualise test plans, the Test Plan Editor (TPE) is used. This NXP specific VBA based tool is benefiting from Microsoft Excels advantages. The TPE allows to view test plans clearly. It also has several features implemented, which enable fast and easy data modification and data comparison within a single test plan or multiple test plans. The Test Plan Editor is used to modify parameters which are necessary to perform noise sensitivity measurements.

2.2.4 Visual Data Analyzer

The Visual Data Analyzer (VDA) is a tool provided by tester supplier SPEA, which is mainly used for debugging purposes. This software package allows to visualise waveforms which are acquired by one or more instruments of the CT1000 tester. The Visual Data

Analyzer is used to view results of the implemented multi-point current measurement. This visualisation allows to evaluate if the current consumption of the smart card device indicates a reset or if the product behaves abnormally while performing noise sensitivity tests. An example of a general current trace can be seen in figure 2.3



Samples : 379
Max : 1.436mA
Average : 1.337mA
Min : -7.167uA

Wave Option: ☐ Apply for All, ☐ Hide, ☐ Line Stairs, ☐ Autoscale, ☐ FFT (dB), ☐ FFT (MAG)

Wave Color: clYellow

Cursors: ☐ Visible; [✓ Cursor 1] Cursor 2; Increment : 1

Cursor 1 — X : 0, Y : None
Cursor 2 — X : 0, Y : None
Cursor 2 - Cursor 1 — X : None, Y : None

Figure 2.3: Visual Data Analyzer tool from SPEA software.

2.3 Reset Behaviour

There are multiple causes for a smart card device to reset. Therefore the following sections will focus more on differences and causes of reset.

2.3.1 Hard Reset

The most prominent reset type is a so called hard reset. When a hard reset occurs, the device is completely removed from all power supplies. Due to the fact that a hard reset implies the removal of the power supply, a hard reset is also called cold reset. [Int06]

2.3.2 Soft Reset

A soft reset is mainly caused by software conditions. Sensors implemented on silicon can trigger a soft reset of the device. This can be an increase of temperature, which would cause the device to overheat. In that case the device would not restart unless temperature is within operating range again. Another reason for a soft reset can be an identified attack onto the device. If the operating system recognises an attack, either by causing a software fault or by triggering a sensor, then the device would automatically provoke a soft reset. Since the voltage supply of the smart card device stays at operating level, compared to a hard reset a soft reset requires less time to restart. A soft reset is also known as warm reset. [Int06]

2.3.3 Security Reset

In case of an attack to a secure device, a soft reset can be caused. Depending on the intensity and impact of the attack, a security reset might lock some part of the implemented software or disconnects sensitive hardware, like memory. In either way, a security reset is logged within the smart card device. Security resets are defined by firmware and operating system. Typically the maximum count of security resets is limited. As soon as the maximum number of security resets is reached, the device gets locked. The difference between a security reset and a soft reset is, that the implemented firmware or operating system has detected an attack to the device. Therefore it can be said, that a security reset is a stricter soft reset.

2.4 Side-Channel Attacks

Side-channel attacks (SCA) use physical information which is observable from outside of the system. Goal of a side-channel attack is to break the security of a cryptographic device. This additional physical information might be the power consumption, timing behaviour, electromagnetic radiation, fault or sound emission. Within this section definition and examples of side-channel attacks will be discussed. [MAK15]

2.4.1 Definition

Side-channel attacks are categorized in two ways. An implemented attack can either be **passive** or **active**. In addition to these two categories it is also possible to differentiate the **invasiveness** of an attack.

While attacking a secure system with a **passive attack**, the cryptographic device is operated within its specification. The secret information is retrieved by the chips physical properties, like execution time and power consumption. In the second category, an **active attack**, the device is strained to malfunctioning. This stress can be generated by manipulating its I/O pins or modifying its operating environment to make the cryptographic device behave abnormally. This abnormality might reveal information about the secret key of the device to an attacker. [POM07]

The invasiveness of an attack can be differentiated into invasive, semi-invasive and non-invasive attacks. All of these attacks can either be active or passive. [MOP07]

Invasive attacks require expensive equipment like a focused ion beam (FIB). It is easy to say that these are the strongest kind of attacks which can be mounted on a cryptographic device. An attacker might cause a malfunction while manipulating the devices internal wiring, which could retrieve secret information. Due to the high expenses for such high-technology equipment there are just few publications regarding invasive attacks on smart card devices. [POM07] [MOP07]

Semi-invasive attacks require to remove the die from its package. In contradiction to invasive attacks there is no direct electrical contact made with the device itself. This means that the passivation layer is not removed or damaged. Malfunction of the cryptographic device might be induced by light. Compared to invasive attacks the equipment for semi-invasive attacks is not that expensive. [POM07]

Non-invasive attacks do not manipulate the device permanently. The system is attacked as it is and additionally no evidence of an attack will be left behind. To cause malfunctions of the device, directly accessible interfaces are manipulated. This might be I/O pins. Due to the relatively inexpensive equipment needed to perform a non-invasive attack, this kind of attack is posed as a serious practical threat to the security of all kinds of cryptographic devices, like smart card devices. [MOP07]

In regards of this master's thesis topic, a noise generating hardware should be designed, which allows to perform electrical stress in form of active, non-invasive attacks on smart card devices.

2.4.2 Power Analysis Attacks

To gain more details about why side-channel attacks are a threat to smart card devices, two specific power analysis attacks (PAA) will be explained briefly. As the name implies these attacks try to gather information about the secret key of the cryptographic system by analysing the devices power consumption. Due to the external power supply of smart cards, power consumption is a potential source of side-channel information. Since cryptographic operations within the secure microprocessor drains a lot of power, there is a direct relation between power consumption of the smart card device and the cryptographic operation. This power consumption can easily be measured by placing a resistance between devices ground and common ground. Attacking a smart card device by observing its power consumption can be performed by simple power analysis (SPA) attacks or differential power analysis (DPA) attacks. [MDS02]

Simple Power Analysis

By performing a simple power analysis (SPA) attack the secret key of a cryptographic device can be revealed on a basis of a few power traces of the secure system. This means that it might be possible to retrieve the key with only a single power trace given. Success of this attack assumes that the secret key has a significant impact on the power consumption of the device under attack. [POM07] [MOP07]

Differential Power Analysis

In comparison to SPA attacks, differential power analysis (DPA) attacks use statistical methods and digital processing techniques. For a successful attack a large number of power traces are being analysed. With this procedure it is possible to reduce the influence of electrical noise. Likewise the difference signals are strengthened. This makes it easier to correlate the power consumption to the secret key. Publications on this topic are warning about the power of DPA attacks. It is said that with this analysis technique it is possible to monitor the actions of a single transistor within a smart card device. [MTS11] [MDS02]

2.4.3 Why are Side-Channel Attacks possible

Advanced semiconductor-manufacturing processes are mostly based on complementary metal-oxide-semiconductor (CMOS) technology. In a CMOS circuit most of the power is dissipated when the input signal of the CMOS transistors is changed. Therefore dynamic power consumption is higher compared to static power dissipation. As the gate voltage V_{Gate} changes from GND to V_{DD} the transistors T_1 and T_2 are both conducting for a short period. While both transistors are conductive, a current will flow from V_{DD} to ground. This current will also discharge the capacitor C_{Load}. In contradiction to the GND to V_{DD} transition, the capacitor C_{Load} will be charged when the gate voltage V_{Gate} switches from V_{DD} to GND. This current difference can be seen on the devices power consumption. Either way information is leaked which can be used to analyse the cryptography of the secure device. The total amount of current being drawn is directly related to the number of gates attached to the load capacitance C_{Load}. [MDS02] [MAK15]

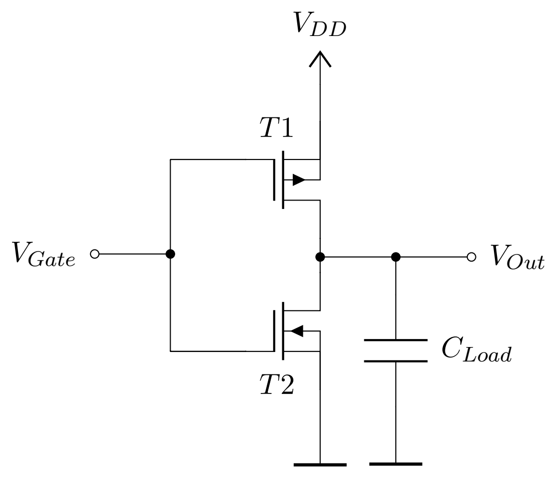


Figure 2.4: Illustration of a CMOS transistor.

2.5 Countermeasures Against Attacks on Smart Card Devices

The goal of countermeasures against side-channel attacks is to eliminate the current to data dependency seen at the power consumption of cryptographic devices. In this chapter some examples of software and hardware countermeasures used to minimise the current to data dependency will be analysed. [MG08]

2.5.1 Software Countermeasures

The use of software countermeasures has advantages regarding its flexibility of their implementation within already existing secure systems. These Implementations can easily be developed and modified if needed. The possibility of a successful power analysis attack increases with the constant repetition of targeted operations in time. Software countermeasures make it possible to randomly shift operations in time. Therefore random delays between operation steps can be implemented. Additionally dummy calculations can be executed to decrease the current to data dependency even further. With this displacement a successful statistical analysis of devices power consumption is made more difficult. [MDS02]

Another possibility to decrease the dependency of cryptographic processes and power consumption is to use data masking or using a special session key. Either way this concept is realised by using a random mask or key. This mask is generated every time a cryptographic algorithm is executed. Afterwards the data will be XORed with the data mask, which hides sensitive information of secret data. When reprocessing the XORed data a reversible operation is used to recover the native data. With this software countermeasure data is masked and therefore current to data dependency decreases. This statement only stays valid as long as the mask or key stays secret. [MDS02] [MOP07]

2.5.2 Hardware Countermeasures

Hardware countermeasures against side-channel attacks can be implemented in two ways. Either by trying to minimise the power to data dependency, or by observing if the smart card is forced to malfunctioning.
The aim on one side is to hide secret informations by the use of circuit countermeasures. This means that hardware circuits are designed in a special way, that the cryptographic operations do not relate to the power dissipation of the device. This means, that the power consumption of the device is being randomised. This can be achieved by switching the outputs independently from their input values. Another approach is to enhance the power management of the device so that it consumes constant energy in each cryptographic cycle. [MTS11] [MG08] [MM16]
Another approach to decrease the data to current dependency by hardware countermeasures is to monitor if the device is being operated within its specified operating range. This means that the smart card device is being operated normally without any external strain which could cause malfunctions. Manipulating the devices input or output pins, attempt malfunctions by inducing light or changing the operating temperature might be a performed side-channel attack. [MDS02] [RE02]

Circuit Level Countermeasures

Hardware countermeasures on circuit level are very effective. However this effectiveness comes with a cost. They require special design effort of internal structures of data processing blocks. Increasing costs, large area occupation and high-power dissipation are in contradiction to the effectiveness of these special hardware countermeasures. Examples for circuit level countermeasures are noise generation and noise injection hardware, power signal filtering and special circuit design methodologies. These countermeasures cannot reduce the dependency of power consumption to cryptographic processes to zero. This implies that power analysis attacks are still possible, but they would require larger numbers of power traces to reveal secret data of the device. [MG08] [MDS02] [BST16] [MA10]

Passivation Monitoring

In the end of the manufacturing process of a smart card device, the so called passivation layer is processed. This additional layer has no electrical purpose. The intention of adding this additional layer onto the silicon is to prevent the die from oxidation. For that reason the passivation layer ensures that the smart card device is functioning over an increased lifetime. Another purpose of the passivation layer is to avoid the possibility to allow electrical contact directly. Attackers might want to manipulate the device to cause malfunction. In order to make electrical contact to the semiconductor the passivation layer needs to be removed. With the use of chemical processes this removal is possible. For the use of controlling the condition of the passivation layer, passivation monitoring is implemented on every smart card device. A sensor circuit is observing the condition of the passivation layer by measuring resistance or capacitance. If this sensor structure identifies a damage within the passivation layer, the smart card devices software might cause an interrupt or the whole chip is shut-down and locked. With this structure implemented, direct analysis of the smart cards electrical behaviour is avoided. [RE02]

Voltage Monitoring

Voltage monitoring is implemented on every smart cards microcontroller. Its purpose is to ensure a defined shut-down of the chip, whenever the voltage is outside of the defined boundaries. By operating a microcontroller outside of its specified voltage range, malfunctions might cause the processor to perform an error in program count. This might lead to an undefined software state, which makes the device vulnerable. For the reason of increased security, voltage sensors are constantly monitoring the supply voltage of the device. For a smart card device supporting voltage classes A and B, this means that the microprocessor is forced to shut down if the voltage drops below $2.3V$ or is above $6.3V$. The specified operating voltages for class A and class B according to the definition in ISO7816 are $3.3V$ and $5V$. In addition to monitoring the supply voltage, there are special power-on recognition circuits implemented, which define the smart card devices power-on condition regardless of the reset signal. [RE02]

Frequency Monitoring

Clock signal generation is defined by the reader, therefore operating speed of the microcontroller is determined from outside of the secure smart card device. This makes it theoretically possible to perform each program step per single clock signal. If an attacker would like to investigate the current consumption of the smart card device while performing cryptographic operations, this condition eases the analysis of each clock cycle. Furthermore this means that each program step can be analysed in detail. To avoid such external driven behaviour, sensors are implemented which are constantly monitoring devices clock frequency. If the clock frequency drops below a certain value, the microprocessor is shut down. Same applies if the clock signal is above the upper limit. The limits are depending on the devices maximum and minimum clock frequency. Since these parameters are product depended, a general statement about specifying limits cannot be made. [RE02]

Temperature Monitoring

The use of temperature sensors in smart card devices is controversial. For the reason that a temporary increase of operating temperature does not harm the function of a smart card device, it is not stated to pose an attack. Therefore temperature monitoring is not fully accepted as a countermeasure against attacks. Temperature monitoring is mainly used to guarantee devices operating behaviour over a bigger temperature range. If the device is tried to be operated outside the defined region, the temperature sensor will force the device to shut down. This ensures that the microprocessors software can operate without causing faults and errors in program counter. [RE02]

2.5.3 V_{DD} Noise and Power Analysis Attacks

Noise is an omnipresent influence on every electric circuit. For that reason there is a possibility that this electrical influence causes threats to the secure device. Consequently some countermeasures against attacks on smart card devices might get triggered, even if there was no attack at all. During a performed power analysis attack there are multiple different types of noise and noise sources present. These potential types of noise are external, intrinsic, quantisation, sampling and algorithmic noise. While performing a power analysis attack, external noise is generated by external sources, like measurement equipment. The source of intrinsic noise on the other hand, is generated in the semiconductor itself. Reason for this type of noise is the random movement of charge carriers. Since every smart card has analog-to-digital converters implemented on silicon, quantization noise is also present on a smart card. By discrete time digitalisation of the power signal, sampling noise is introduced into the system. Finally algorithmic noise is also present within a smart card device. The cause of this type of noise is due to the different processed data bytes by the smart card. [MDS02] [YWCZ14]

With the knowledge of types of noise present on smart card devices it is also possible to use them as a countermeasure against side-channel attacks. The idea behind this method is to decrease the signal-to-noise ratio (SNR). With a low SNR the chances of a successful performed PAA are decreasing. [DMN$^+$14] [YWCZ14] [YK17]

2.6 State of the Art Noise Sensitivity Measurement

Noise sensitivity of smart card devices is currently mainly inspected in laboratory environment. These measurements require costly and precise equipment. For that reason appropriate laboratory infrastructure is highly recommended. In this section a possibility of such measurement setup will be described briefly.

2.6.1 Setup

A sine-wave generator, triggered by a timing delay unit, produces a noise signal. The setup can be seen in figure 2.6. This noise signal is stressing a smart card device under test (DUT), which is supplied from a reader. A switching matrix performs necessary signal level separation and adds the noise level onto devices pins. This switching matrix and smart card reader can be seen in figure 2.7. Whenever the noise signal causes the DUT to reset, the smart card device will send an ATR response to the reader. In order to visualise the voltage levels and signals at DUTs ISO7816 pins, an oscilloscope is used. The overall block diagram of this setup is illustrated in figure 2.5.

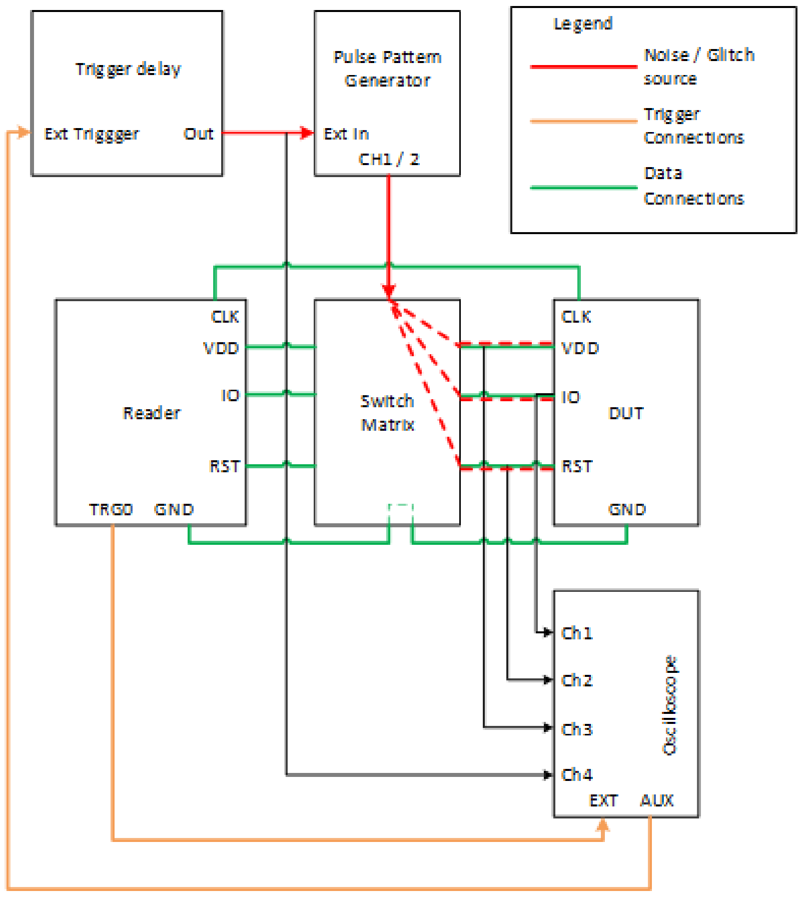


Figure 2.5: Block diagram of the laboratory setup.

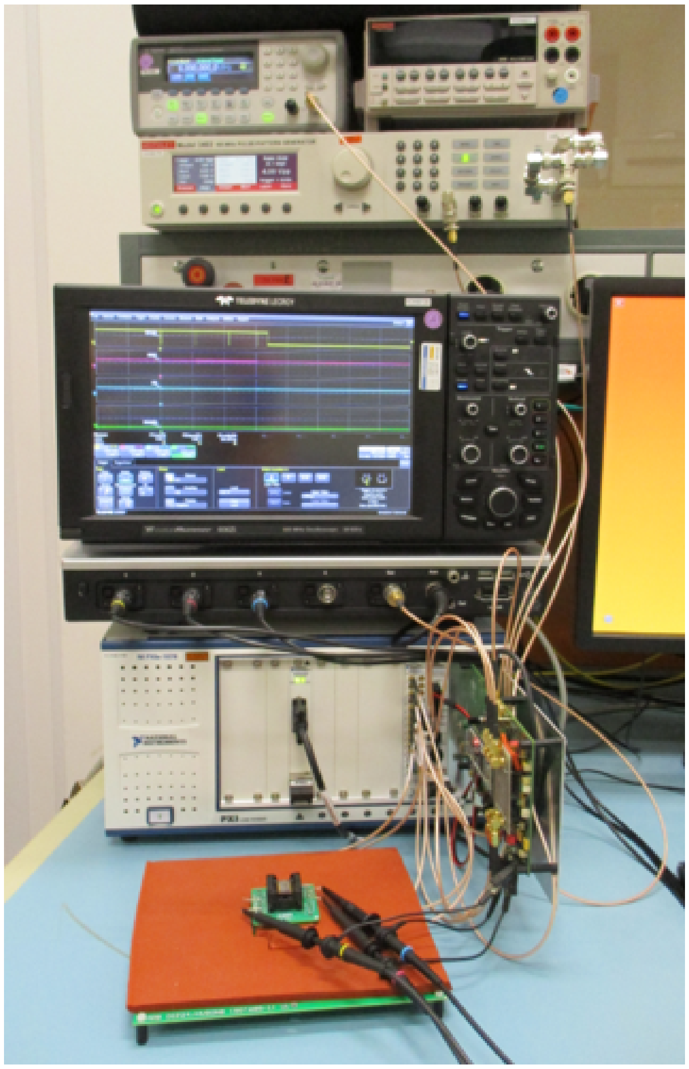


Figure 2.6: State of the art noise sensitivity measurement performed in NXP laboratory.

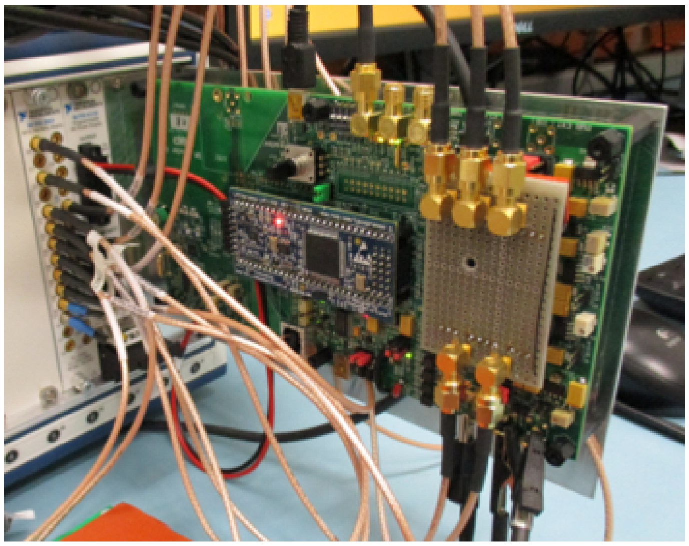


Figure 2.7: Signal switching matrix (left) and smart card reader (right).

2.6.2 Disadvantages

This state of the art setup to inspect sensitivity of smart card devices to noise and to instabilities in voltage supply is very accurate and delivers results in high resolution. In contrary to this positive properties, high test time and low through-put are disadvantages. Performing these measurements consumes several hours for few devices. In respect to the stated attributes this measurement approach is not suitable for high-quantity testing.

2.6.3 Conclusion

For the reason that the results of this measurement are treated confidential, the results of this measurement cannot be published. However a conclusion can be given. The conclusion of this measurement setup is, that smart card devices are very sensitive to voltage variation within their start-up phase. It could also be proofed that these security products are more stable as soon as they are in idle state. Another observation was that glitches on devices input and output ports, as well as the V_{DD} line, have more impact on causing resets of the device as pulses have. The results of these measurements were the foundation of any further investigation within this master's thesis.

2.6.4 Verification

To get a first feeling of the reset behaviour of smart card devices and to verify the results of previous measurement, a simple setup with a function generator was used. A block diagram describing the measurement can be seen in figure 2.8. The signal generating device was programmed to output custom pulses. These pulses were defined as short as possible, resulting in a pulse length of approximately $80ns$. The arbitrary function is shown in figure 2.9. The reason why this approach was not successful is mainly influenced by two conditions. Firstly the period time of the function is too short. This results in too many pulses within the specified waiting time. It is specified that the signals period time should be in the range of $500\mu s$ to $1ms$. With the arbitrary function signal it is not possible to increase the period time from approximately $350ns$ to the specified range. The period time is fixed due to the fact that the number of variable points within a custom function is limited. For that reason it gets quite difficult to observe a reset of the device easily. Secondly the signal length is to narrow. For that reason not enough energy is put onto the corresponding pin to trigger an internal countermeasure which would cause the smart card device to reset. Since the period time of the arbitrary function generator is basically fixed, it does not make a lot of sense to increase the pulse width. More details about electrical parameters of the setup using a function generator is covered in table 2.5. Stressing the smart card device during the start-up phase increases the impact of the forced signal. It was also observable that the signal is not suitable to cause a reset of the device when it is already powered up and in idle state. Therefore the planned hardware needs more flexibility regarding timing and signal parameters. [Agi02]

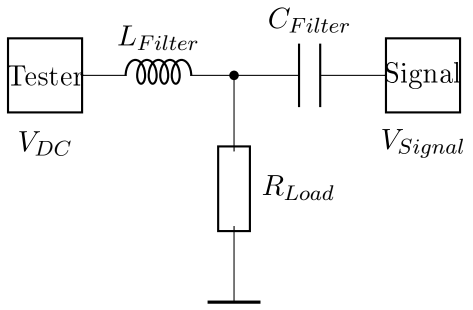


Figure 2.8: Block diagram of verification measurement.

Parameter	Value
V_{DC}	$5V$
V_{Signal}	$5V$ @ f=$2.5MHz$
C_{Filter}	$330pF$
L_{Filter}	$32mH$
R_{Load}	$1k\Omega$

Table 2.5: Settings of verification setup using a function generator

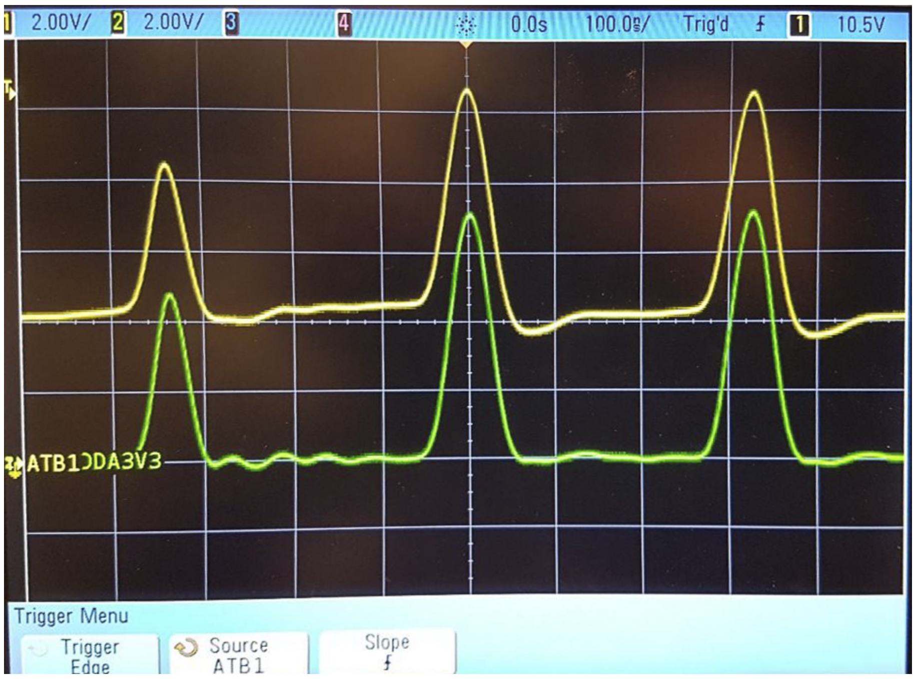


Figure 2.9: Pulse generated by function generator (green: pulse, yellow: 5V DC and pulse).

2.7 Pattern Controlled VDD Glitch

To verify previous achieved measurement results, a smart card device was stressed with glitches on its V_{DD} pin. Therefore the SPEA CT1000 tester was used to generate a pattern. This pattern made it possible to adjust the length of the glitch and to control the low-level of V_{DD}.

2.7.1 Glitch Control

Using the VectorVIEW tool provided by the tester software environment AtosCT, a pattern was generated. This pattern generation allowed to control the V_{DD} level of the smart card device. Within this tool it is possible to define low-level of V_{DD} and duration of this glitch. By defining a specific low-level and varying the duration of this signal, it was possible to observe devices reset behaviour in regard of glitches on supply voltage. The definition of the pattern can be seen in figure 2.10. In this VectorVIEW window it can be seen, that duration of the glitch is defined to be $20ns$. This value is the shortest possible duration. Furthermore the low-level of the signal is defined to be 0.5V. By applying so called service instructions, it is possible to stretch the length of a glitch. These special commands allow to loop the definition of the current pattern step. Therefore multiples of $20ns$ are possible values for the duration of the glitch. In order to give the smart card device a proper start-up time, V_{DD} is set to high-level for multiple milliseconds before performing the glitch. This high-level is defined to be 5V which means that the smart card device was operated in voltage class A.

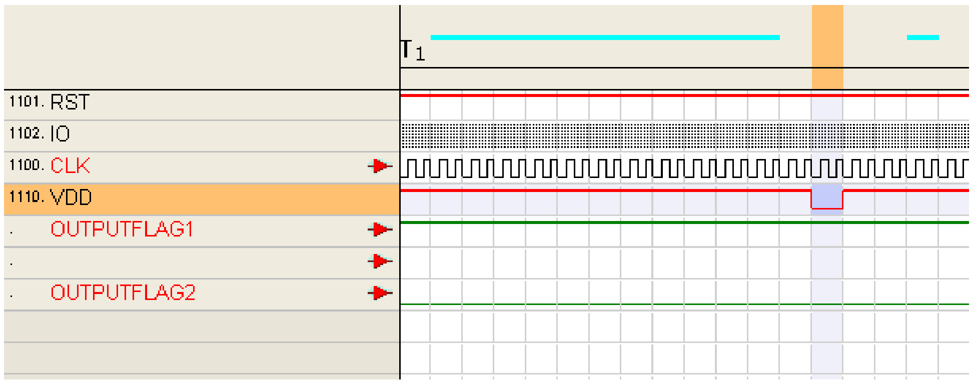


Figure 2.10: Settings of pattern controlled V_{DD} glitch performed on smart card device.

2.7.2 Setup

The digital board of the smart card tester was supplying smart cards V_{DD}. This made it possible to control V_{DD} properties via pattern. Duration and low-level of this glitch could easily be adjusted using service instructions. To supply DUT with V_{DD} generated by digital board of the tester, two adapter PCBs where necessary. These adapters can be seen in figure 2.11. In addition to these two custom printed circuit boards, another adapter PCB was used to ensure connectivity with a smart card device. This product was assembled in a standard ISO7816 package. The adapter in the middle, connected to testers analog $JANL03$ interface, provides access to pattern controlled V_{DD}. The adapter on the right connects DUT with testers digital $JCH01$ interface. Additionally the PCB on the left allows easy electrical connection to the smart card device itself.

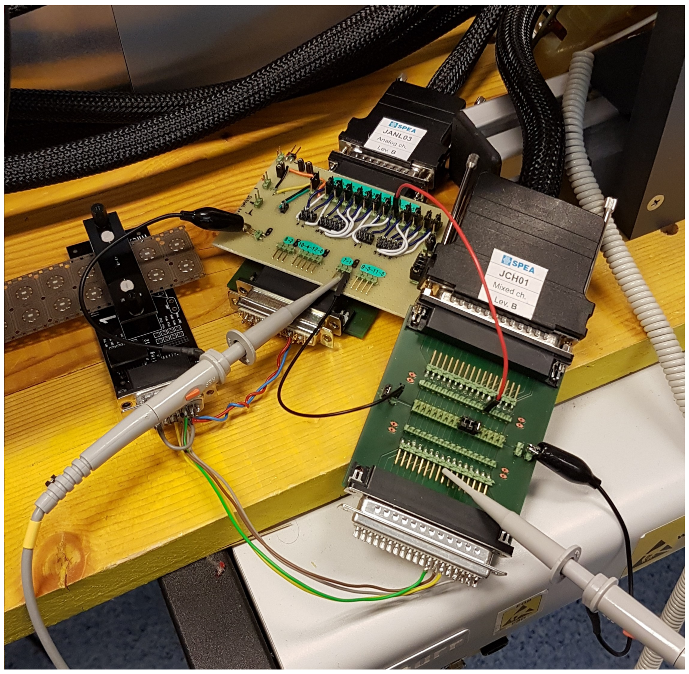


Figure 2.11: Setup of pattern controlled V_{DD} glitch performed on smart card device in ISO7816 package.

2.7.3 Results

Observations of pattern controlled V_{DD} glitch measurement have shown, that the reset behaviour is depending on the combination of V_{DD} low-level and length of the glitch. The effect of this relationship is visualised in figure 2.12. This graph indicates the length of a V_{DD} glitch at a certain low-level, where the smart card device starts to reset.

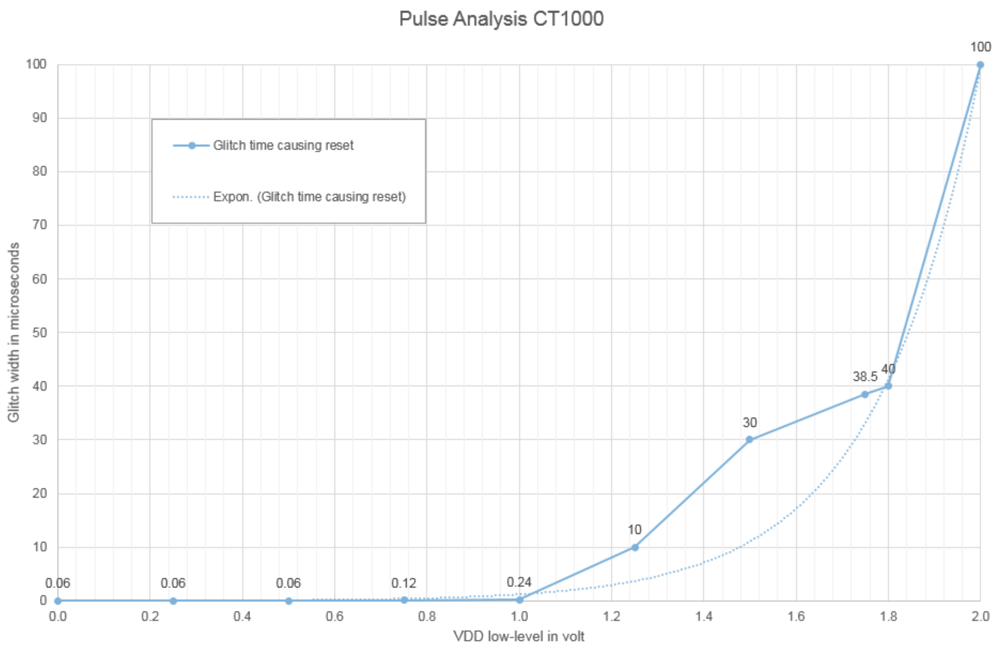


Figure 2.12: Sensitivity to glitches on V_{DD} pin in respect to signals low-level and duration of the glitch.

2.8 Influences and Effects of Measurement Equipment

The influence of measurement devices while performing noise sensitivity measurements could be observed during the measurement described in section 2.7. It is observable that the influence of an oscilloscope for example has big impact on the result of the performed measurement. The capacitance of the scope and its probe is sufficient to power the smart card device during a short lasting glitch on power supply pin. After disconnecting the oscilloscope from DUT, a reset could be seen by observing the current consumption of the device. Additionally to these capacitive influences, another effect could be observed. Due to a non-ideal measurement setup with long traces and additional wires from various laboratory equipment, the influence of inductive parasitics could be observed. This unwanted influence is causing ringing and overshoot on the high-frequency signal. Therefore it can be said that a non-ideal measurement setup is causing a decrease in signal quality.

2.9 Signal Description

With results from measurement covered in section 2.7, the definition of the signal could be realised. These parameters, like signal amplitude and length of the signal, are essential for stressing devices. The goal of the signal specification is to define parameters, so resets can be caused. In order to enable the design of an appropriate hardware to generate corresponding signals, these specifications are a first estimation. To make hardware design straight forward it was also defined to use a simple square wave signal form.

2.9.1 Parameter of Pulse Signal

To stress the device with pulses a square wave signal is generated. This wave form is then added to the DC voltage level of the corresponding pin. In respect to the definition of parameters like amplitude and length, values can be seen in table 2.6. It has to be pointed out that amplitude values are adding up to the already existing voltage level of the stressed pin. To visualise this glitch signal figure 2.13 is added.

	Minimum	Maximum
Amplitude	$1.5V$	$5V$
Length	$50ns$	$20\mu s$
Period	$500\mu s$	$1ms$

Table 2.6: Definition of pulse parameters

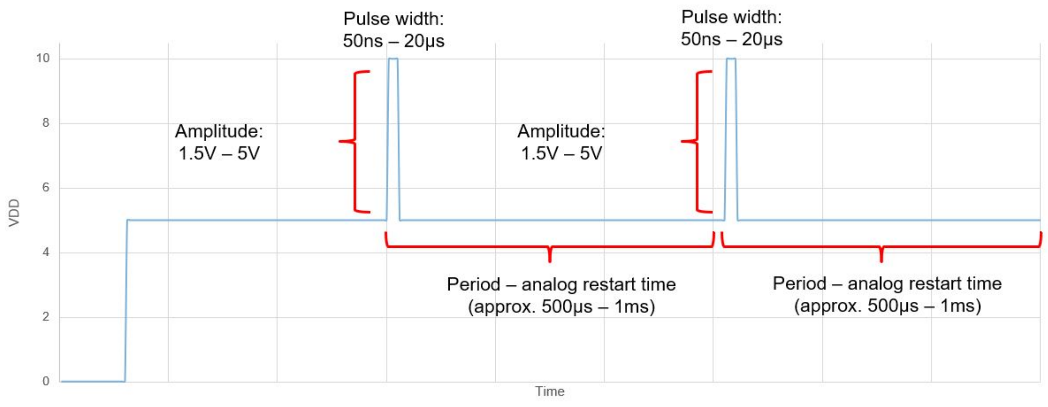


Figure 2.13: Definition of pulse parameters visualised in a graph.

2.9.2 Parameter of Glitch Signal

The definition of signal parameters are shown in table 2.7. Amplitude in this context means low-level of the already existing voltage level of the stressed pin. In figure 2.14 the definition of glitch signal is visualised with all necessary parameters.

	Minimum	Maximum
Amplitude (low-level)	$1.5V$	$5V$
Length	$50ns$	$20\mu s$
Period	$500\mu s$	$1ms$

Table 2.7: Definition of glitch parameters

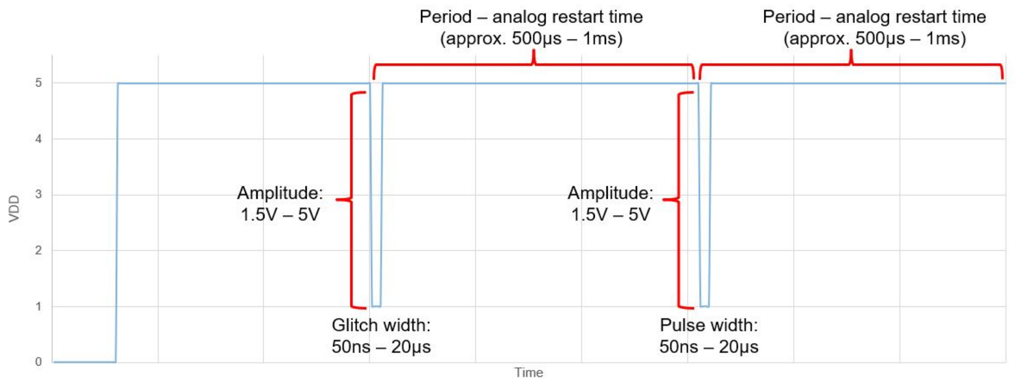


Figure 2.14: Definition of glitch parameters visualised in a graph.

2.10 Direct Power Injection

While testers DC level should not be influenced by the noise sensitivity measurement and signal generation should not be affected by testers DC level, an appropriate filter design is necessary. For this reason a simplified version of direct power injection is used to add the signal to testers DC voltage. On one hand a capacitor is used to block DC levels. On the other hand an inductance blocks any high-frequency signals. This solution ensures that no HF signal is seen by the tester and no DC level is influencing the signal generation circuit. This means that, on the junction of capacitor and inductance DC level plus pulse or glitch is measurable. Therefore the signal fully sinks into DUT. [Gun06]

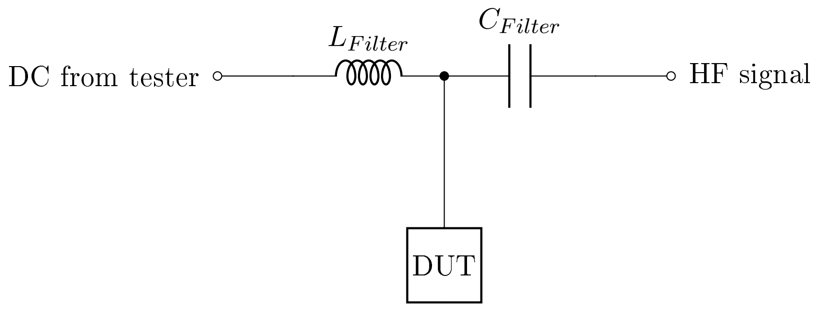


Figure 2.15: Simplified version of direct power injection to add signal to DC level.

2.11 Half-Bridge Signal Generation

To provide fast and flexible signal generation, a half-bridge circuit is used. This special electronic circuitry, which is simplified illustrated in figure 2.16, has the purpose of transforming the microcontrollers output into a variable signal. To ensure proper switching times, a gate driver IC is used to translate its input signal into high-current gate driving voltages. With this additional component it is possible to increase the switching speed of the half-bridge. Furthermore, the gate driving IC also translates microcontrollers output signal into two gate driving voltages. One of these signals is inverting, while the other signal is non-inverting. For more details on the half-bridge circuit, two NMOS are used. One transistor is responsible for switching high-side of the power supply to the signals output. The other transistor is connecting the output of the half-bridge to GND. Therefore the output of the half-bridge circuit is always defined. More details about generating pules and glitches with this half-bridge circuit will be given within this section.

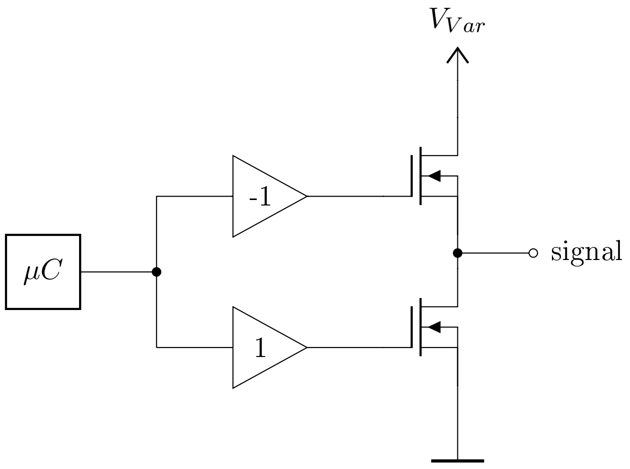


Figure 2.16: Simplified half-bridge design for signal generation.

2.11.1 Generating Pulse

Generating pulse signals on the output of the half-bridge circuit, as shown in figure 2.17, is relatively simple. The bridge voltage V_{Var} is set to the desired amplitude of the pulse signal. Whenever a low-level is applied to the gate driving IC, the high-side transistor $T1$ will be switched off, while the low-side FET $T2$ will be switched on. On the other hand, when applying a high-level to the gate drivers input, the low-side transistor $T2$ will be switched off, while the high-side switch $T1$ is turned on. By programming the microcontrollers output so that it generates a very fast transition from low to high and back from high to low again, a pulse signal will be generated at half-bridges output. In order to allow fast switching times, an additional gate driver IC is used as illustrated in figure 2.16. In case of pulse generation the half-bridge circuit works as a level converter. This means that the microcontrollers output signal is converted to the level of V_{Var}. This can either be bigger or smaller than the output voltage of microcontrollers I/O-port.

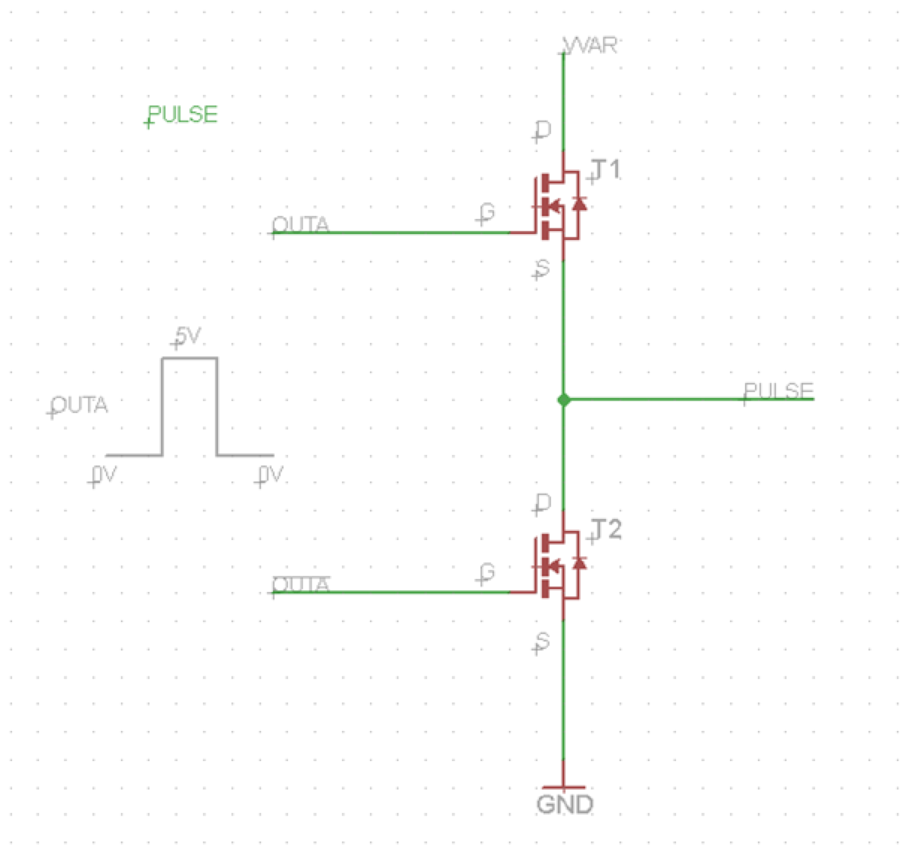


Figure 2.17: Half-bridge circuit used to generate pulse signal.

2.11.2 Generating Glitch

Within this section two possible solutions are described how to generate a glitch signal with the use of half-bridges circuits. On one side a negative voltage can be switched by transistors to generate a glitch. On the other side it is also possible to invert the microcontrollers output signal. More detail about glitch signal generation will be given in the following subsections.

Inverted Gate Input Signal

With an inverted gate driver input signal the half-bridge circuit is able to generate a glitch signal. This approach is similar to generating a pulse signal discussed in the previous section, except the inverted input signal. By applying the desired DC level on V_{Var} glitches can be generated directly. The bridge-voltage V_{Var} is then directly related to the low-level of the glitch, when added to a DC-level. Advantage of this approach is, that there is no need for additional hardware. If pulses and glitches need to be generated, the same circuitry can be used just by changing the input signal. Disadvantage on the other hand is, that this approach is not as flexible as the solution, which will be covered in the next subsection.

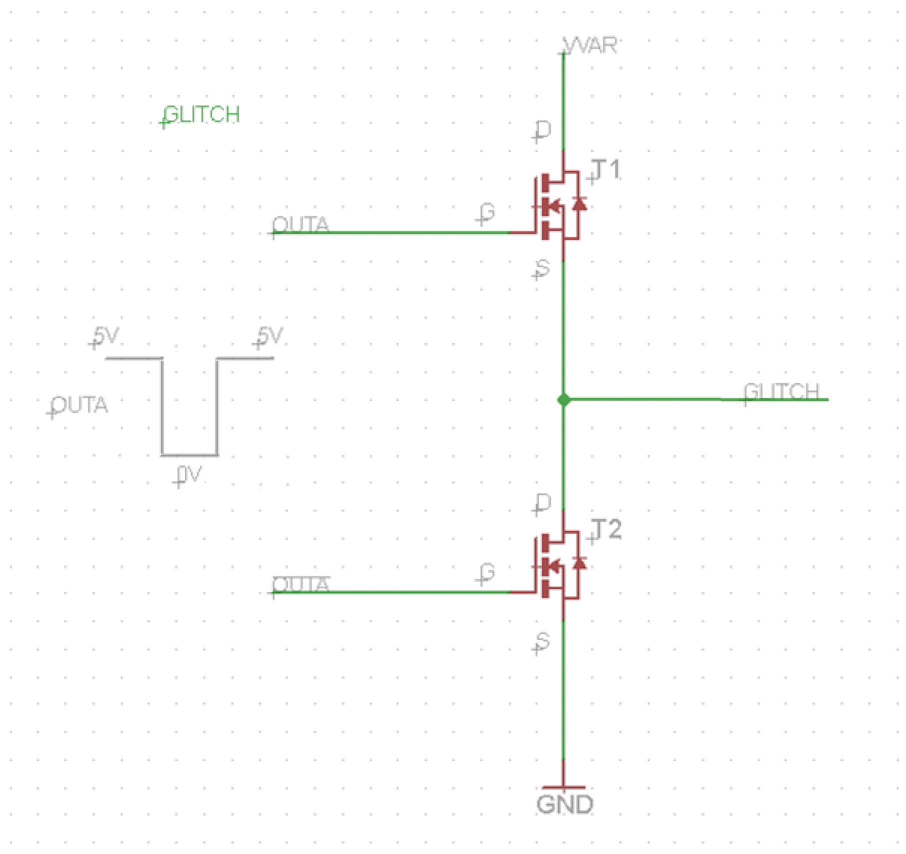


Figure 2.18: Half-bridge circuit used to generate glitch signal with inverted input signal.

Negative Bridge Voltage

By applying a negative voltage to a half-bridge circuit it is possible to generate pulses with negative amplitude. Overlaying this negative pulses to an existing DC-level would create glitches with a low-level of DC-level minus signal amplitude. With this approach it is also possible to generate glitches with a negative low-level in respect to GND. Downside of this approach is the increased amount of circuitry. To drive this half-bridge properly, three independent voltage levels are necessary where two of them must be negative in respect to GND. One of them is used to provide $-V_{Var}$ and the other one is used to provide gate drivers low-level. This low-level is necessary to allow proper switching of the MOS transistors. The value of negative gate driving voltage is $-5V$ while the positive gate driving voltage is $3.3V$. The signal description can be seen in figure 2.20. Another disadvantage is, that care needs to be taken when switching this half-bridge circuit seen in figure 2.19. By applying an unsuitable gate voltage, the MOSFETs are in danger of thermal destruction.

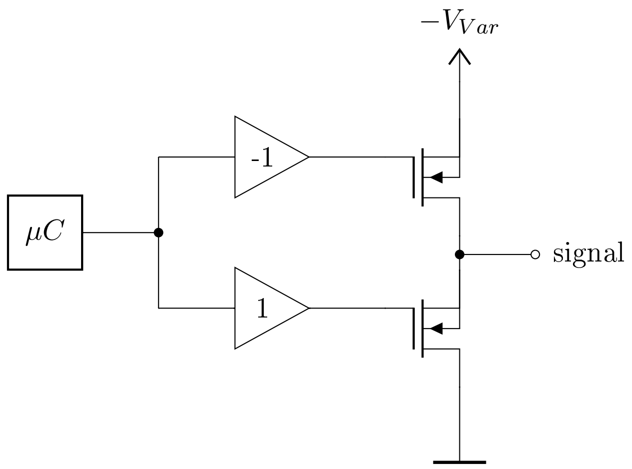


Figure 2.19: Simplified half-bridge design with negative output signal.

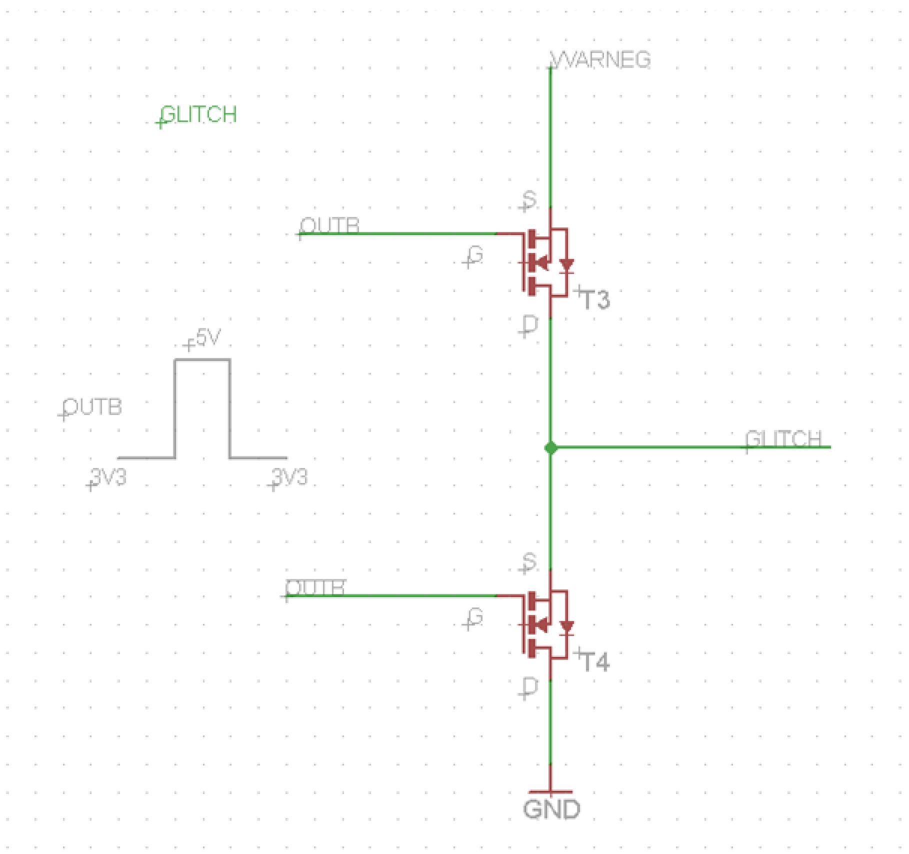


Figure 2.20: Half-bridge circuit used to generate glitch signal with negative amplitude.

2.12 Voltage Regulator with Negative Voltage Output

For the reason that every electrical circuit is in need of a power supply, voltage generating circuits are necessary. In order to achieve the desired voltage supply, multiple different regulators are available in various topologies. In contrary to the large amount of voltage regulators available for positive supply levels is the lack of integrated circuits allowing to easily implement negative supply voltages on PCB-level. For implementation of negative voltages in respect to GND special circuit design needs to be implemented. Within this section two possible solutions for generating negative voltages on PCB-level are discussed. [Now12] [ST07]

2.12.1 Inverting Charge Pump

Charge pumps are commonly used for voltage supplies with low current requirements. If these circuits are used in inverting topology, negative voltages can easily be generated. By using two switches and two capacitors an inverting charge pump can be implemented. An inverting charge pump can be seen in figure 2.21. In the first step, $S1$ is connected to the input voltage, while $S2$ is connected to GND. Within this operating step, the capacitor $C1$ is charged to the input voltage. After the capacitor $C1$ is charged, the switch $S1$ connects GND to the capacitor while $S2$ connects $C1$ with $C2$. With this transition, the voltage of $C1$ gets inverted. This means that the capacitance $C2$ gets charged to inverted level of the input voltage. In the end, the voltage of $C2$ is seen at the output terminal of the inverting charge pump. With an adjustable voltage at the input of this inverting charge pump circuit, negative voltages can be realised on PCB level. As already described, the current driving capability is very limited on these charge pump circuits. Therefore load resistance is the determining factor of output ripple of the generated negative voltage. [ST07] [Now12]

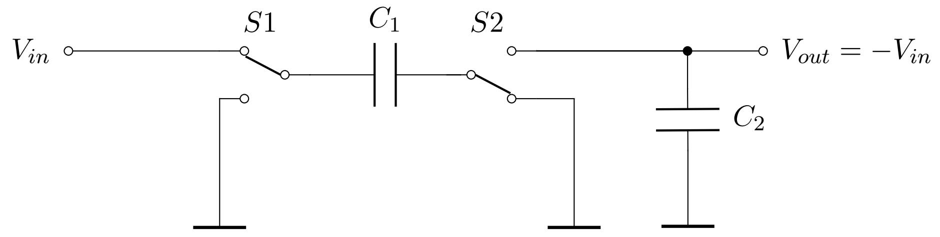


Figure 2.21: Charge Pump to generate negative voltage.

2.12.2 Buck-Boost Voltage Inverter

Negative voltages can also be generated by the use of regulators in buck-boost topology. These special ICs are also known as flyback converters. In order to supply negative voltages, the ground pin of the voltage regulator is connected to $-V_{out}$. For providing reference, the output capacitance is reversed polarity. This means that the positive lead of the capacitor is connected to ground. Furthermore the output voltage can be adjusted by selecting different resistance values for the voltage divider represented by $R1$ and $R2$. In order to allow voltage changes on the fly, an additional potentiometer is connected in parallell to

$R2$. With that implemented, the use of an adjustable negative voltage source is possible. Advantage of a buck-boost voltage inverter is its high-current driving capability. For that reason the use of a buck-boost converter to generate negative voltages is preferred. [ST07] [HKP+10] [Now12]

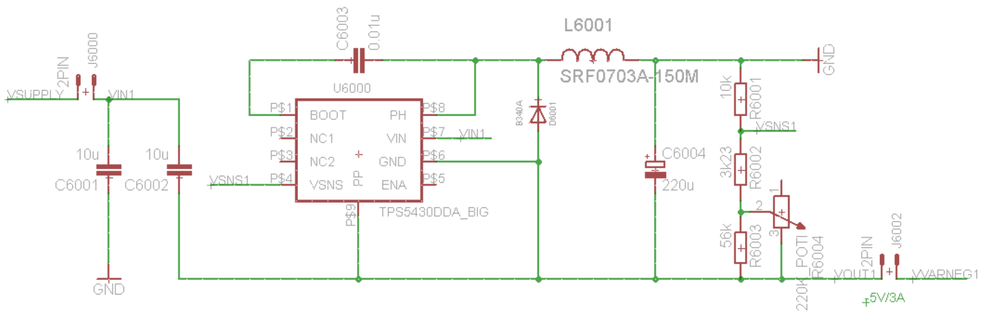


Figure 2.22: Voltage regulator in buck-boost topology used to generate negative output.

2.13 Serial Peripheral Interface

To ensure correct communication between signal generating hardware and tester, a special interface is required. Due to the simple fact that the tester already has serial peripheral interface integrated, serial peripheral interface (SPI) is used to allow communication between CT1000 tester and noise generating printed circuit board.

2.13.1 Specification

With synchronous serial interfaces it is possible to transmit individual bits consecutively. Timing is determined by a clock signal. Serial peripheral interface is a common used interface for communication between microprocessors and other devices. An example of transmitted data is visualised in figure 2.23.

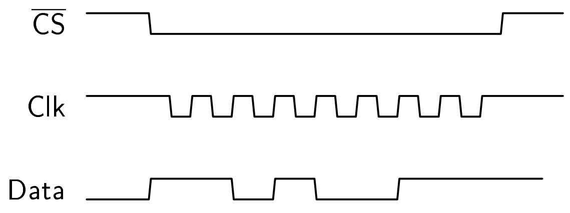


Figure 2.23: 8 bits of data transmitted via SPI.

Serial peripheral interface basically consists of two wires, a data line and a clock line. To make bidirectional communication possible, a second data line is introduced. The master device is always responsible for providing a clock signal. While clocking data from master to slave, it is also possible that the slave sends data in sync. Furthermore, when communicating with more than one slave, an additional chip select line is used. By selecting the corresponding chip select line the master can transmit data to a specific slave.

A description of SPI pins can be found in table 2.8. Within serial peripheral interface data format is not standardised. This data format can be set up in several ways. For example the definition of clock signal in idle state can either be low or high. This parameter is called clock polarity. Another parameter is called clock phase. This parameter determines if data is read on even or odd clock pulses. Size of transmitted data is freely selectable. Overall transmitted data is mostly limited by receiving capabilities of the slave device. The communication shown in figure 2.23 for example has following SPI parameters. Clock polarity is set to 1 which means that idle state of CLK is HI. Clock phase is equal to 0 which means that falling edge of clock indicates data valid. To sum up some advantages of this synchronous interface, low wire count and modest data rates make easy integration onto printed circuit possible. [HKP+10]

$Data$	Data
CLK	Clock
$\overline{CS}$	Chip Select
$MOSI$	Master Out Slave In
$MISO$	Master In Slave Out

Table 2.8: Definition of SPI pins

2.14 ATE Multi-Point Current Measurement

In order to allow accurate analysis of smart card devices reset behaviour and sensitivity to noise, the SPEA CT1000 tester is used to monitor the current consumption of the device. With the available tester multi-point measurement is possible. First of all, the built-in tester hardware characterises measurement capabilities of the overall system. Secondly, timing properties are defined at the tester. Lastly, the programmed signals need to be connected to the noise generating printed circuit board. Within this section, the possibilities of the SPEA CT1000 smart card tester are being discussed.

2.14.1 Analog Measurement

With the capabilities of SPEA CT1000 tester, it is possible to implement multi-point current measurement. In order to sample analog values at defined points in time, special requirements need to be considered. Within this section analog and digital possibilities of the tester are described, as well as information needed to ensure correct timing of the performed measurement.

Analog Driver/Sensor (ADS)

An analog measurement on SPEA CT1000 is performed using the “Analog Driver Sensor” (ADS) instrument family. This unit is able to force and measure voltages and currents per pin. Each of these drivers providing voltages up to $\pm 10V$ and currents up to $100mA$ on four quadrants, operating both as generator and load. [SPEb]

Digital Driver/Sensor (DDS)

In contrary to the analog driver the "Digital Driver Sensor" (DDS) instrument family is designed to generate digital pattern up to 80MHz. It also manages the logic for comparison and the acquisition memory of the signal which is sensed by the digital sensor board. [SPEb]

Analog Phases

To synchronise the analog instrument of the tester, timing signals for analog tests are generated. With this special timing capability, it is possible to allow accurate synchronisation between measurement and signal switching, as well as synchronisation between tester and noise generating PCB. The available signals are described in section 2.14.2 in more detail.

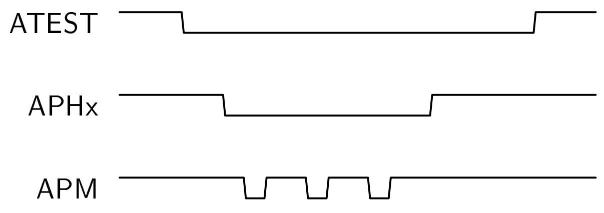


Figure 2.24: Example of Phases and Strobes within an analog measurement test.

To synchronise the source section and the measurement section of the analog instrument of the tester, four phases APH1 to APH4 are used. Additionally, the strobe is used to synchronise the measurement section only. All Phases and the strobe are completely independent among them. Analog tests are only executed if APH1 is programmed and enabled. Therefore, this makes APH1 the master phase. Furthermore, ATEST is a signal, which is active for the complete test time. It is also important to say, that ATEST is also depending on other outputs. All outputs synchronised with APH1 or APH2 can be formatted in three different ways, in direct pulse mode, in inverted pulse mode or in continuous mode. In direct pulse mode (`D_PULSE`) the programmed voltage or current is provided to the output pin only during on-time $T2$ of the analog phase. In inverted pulse mode (`I_PULSE`) the voltage or current, which is provided by the source section, is only provided to the output pin during on-delay time $T1$ and off-time $T3$ of the analog phase. Therefore `I_Pulse` mode is inverted to `D_Pulse` mode. In contradiction to the mentioned two modes is the continuous mode (`CONT`). Here the programmed voltage or current is provided as soon as the instrument has been connected and enabled. Furthermore this means, that during `CONT` mode outputs are timing independent. [SPEb]

Strobes

Strobes (APM) are fully independent to analog phases APH1 to APH4. These strobes simply determine the point in time, when an analog measurement needs to be executed. Furthermore this describes, when a measurement value will be acquired. Only limitation, besides timing, is that all strobes need to be placed while ATEST is active. More details about timing capabilities of strobes are defined in the following section 2.14.2. [SPEb]

2.14.2 Timing

In order to ensure proper timing, it is very important to use phases and strobes correctly. The phases and the strobes determine the main properties of a performed measurement. More details about timing capabilities of the system are illustrated within the following tables.

Analog Phases

Analog phases (APHx) are programmatically defined with the following instruction set:

$$AnlAccPhaseSet(PhaseNr, T1, T2, T3, Cycles); \tag{2.1}$$

The parameters of this instruction set provide information of the phase used, as well as timing properties. Firstly there is $PhaseNr$, which defines the analog phase. The parameter $T1$ specifies the on-delay, while $T2$ defines the on-time of the analog phase. Parameter $T3$ defines off-time of the analog phase. Lastly $Cycles$ specifies the amount of iterations. Definition of the parameters described within this section can be seen in table 2.9 and 2.10. [SPEb]

T1, T2, T3	Value
Minimum length	$500ns$
Maximum length	$2147.483648s$
Step	$500ns$

Table 2.9: Timing definition of analog phases (APH)

Cycle	Value
Minimum	1
Maximum	65535

Table 2.10: Cycle definition of analog phases (APH)

Strobes

Strobes are used to define the timing of the measurement, as well as the number of acquired values. Therefore Strobes (APM) are programmatically defined with the help of the following instruction set:

$$AnlAccStrobeSet(T1, T2, Cycles); \tag{2.2}$$

Similar to the definition of parameters while using analog phases, is the use of strobes. Firstly parameter $T1$ defines the delay, before the first strobe will be performed. Secondly $T2$ specifies the spacing before the next strobe will be performed. Lastly $Cycles$ defines the number of total strobes generated by the tester. This count is equivalent to the number of samples acquired by the tester. Definition of the parameters described within this section can be seen in table 2.11, 2.12 and 2.13. [SPEb]

T1	Value
Minimum length	$500ns$
Maximum length	$2147.483648s$
Step	$500ns$

Table 2.11: Timing definition T1 of strobe (APM)

T2	Value
Minimum length	$1.5\mu s$
Maximum length	$2147.483648s$
Step	$500ns$

Table 2.12: Timing definition T2 of strobe (APM)

Cycle	Value
Minimum	1
Maximum	65535

Table 2.13: Cycle definition of strobe (APM)

2.14.3 Synchronization with Printed Circuit Board

To ensure accurate alignment of CT1000 tester and noise generating printed circuit board, an interface is necessary. To make timing signals available at a tester interface it is possible to connect the timing signals to specific outputs. To create an association between the outputs and the synchrobus lines, the following instruction set is used.

$$AnlTimingSyncSet(Setmode, Syncref, Mode); \tag{2.3}$$

With this function it is possible to change the connection of the synchrobus lines and the output as desired. Furthermore this allows to output the timing signal specified via instruction sets, described in section 2.14.2, on a custom interface. For that reason it is possible to align tester and noise generating PCB. In order to understand these connections in more detail, default connections to the synchrobus are shown in table 2.14. Furthermore parameters of the instruction set are discussed specifically. Firstly, parameter $Setmode$ specifies if the corresponding signal is defined as input or output. Secondly, $Syncref$ defines the synchronisation line reference. Lastly, if $Setmode$ is set to $INPSYNC$ $Mode$

identifies the synchronisation mode itself. Otherwise this parameter gets ignored. Further informations about possible settings of this instruction set are given in table 2.15. [SPEb]

Name	Description	Default
ATEST	Test time	SBUS13
APH1	Phase 1	SBUS14
APH2	Phase 2	SBUS15
APH3	Phase 3	Not associated
APH4	Phase 4	Not associated
APM	Strobe	SBUS16

Table 2.14: Default connections of phases and strobe to synchrobuses.

Parameter	Value	Description
Setmode	INPSYNC OUTSYNC	Defines if the signal is output or input
Syncref if Setmode = INPSYNC	SBUS1 to SBUS7 SBUS10 to SBUS 12 S50Hz SINTERFACE SNONE	Identifies the synchrobus lines or other lines as the start of analog timing
Syncref if Setmode = OUTSYNC	SBUS13 SBUS14 SBUS15 SBUS16	Identifies the line of the synchrobus
Mode if Setmode = INPSYNC	REDGE FEDGE	Defines if timing is started on rising or falling edge of signal

Table 2.15: Parameters of instruction set AnlTimingSyncSet.

Chapter 3

Potential Solutions

Within this chapter possible solutions to fulfil the requirements of this thesis are being discussed. Defining the overall system frame as well as more specific hardware designs, like voltage supply, signal generation and sweep of parameters are covered within this chapter.

3.1 Design including FPGA

With pulses and glitches in the range of a few nano seconds it is very important to select the right hardware, which is able to generate those high-speed signals. For that reason the first idea of a hardware design included an FPGA for high-speed signal generation.

3.1.1 Block Diagram

A microcontroller communicates with the tester and receives all necessary data to setup signal parameters for noise sensitivity measurement. Therefore voltage levels need to be adjusted and signal parameters and form need to be generated. For high-speed signal generation within nano second range, FPGAs are the first choice. Idea of this system design is, that a microcontroller defines signals parameters and form. An FPGA connected to a R2R resistor ladder then generates the signal. This signal is then amplified to the corresponding voltage level defined by the tester. With a switching matrix this custom signal is then applied to DUT.

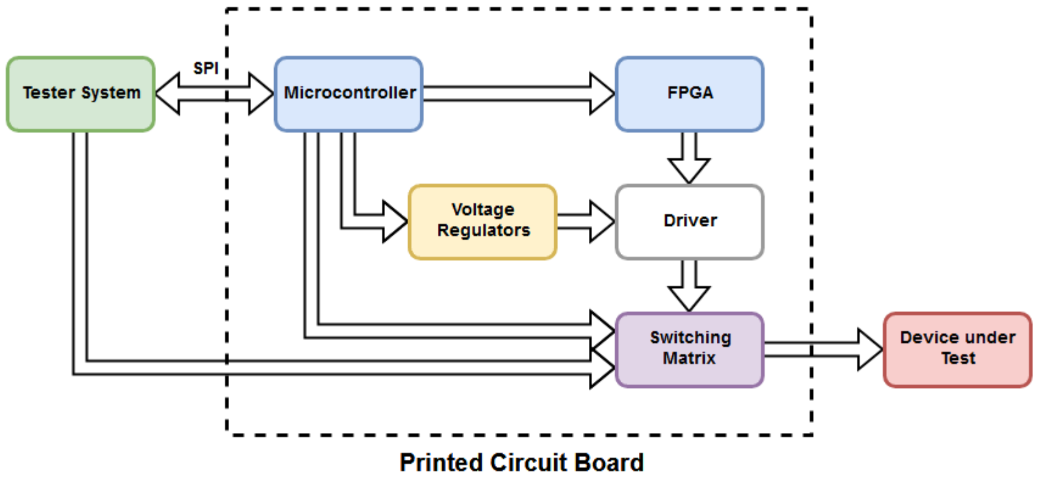


Figure 3.1: Systems block diagram with FPGA as signal generating unit.

3.1.2 Advantages and Disadvantages

Flexibility of this solution is a big advantage. With an FPGA it is possible to generate multiple signal forms, like square waves, triangle waves and slopes. With that feature, investigations about sensitivity to special wave forms are achievable. Also output switching speed adds to the positive. Clock-to-output times below $3ns$ are realistic. Disadvantage on the other side is the higher complexity of hardware design, which also results in bigger area of the PCB. This would also make it more difficult to integrate the external hardware on probe-card level. [XIL18]

3.2 Design with MCU

A design with the absence of an FPGA decreases the overall complexity of the PCB. After investigating output speed of the STM32F072RBT6 microcontroller, it can be said, that the output switching speed capability is fast enough in respect to the timing requirements. With a maximum rise and fall time of $5ns$, the STM32F072xB family switches their outputs fast enough to fulfil the requirements. For that reason and for decreased PCB area, the design including an FPGA became obsolete. [ST 17]

3.2.1 Block Diagram

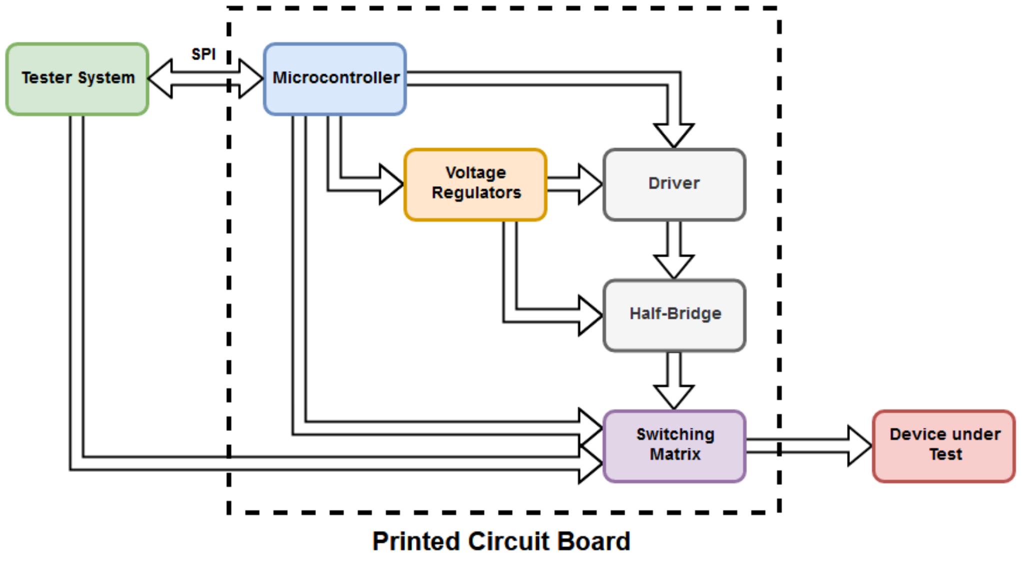


Figure 3.2: Systems block diagram with Microcontroller as signal generating unit.

3.2.2 Advantages and Disadvantages

Biggest upside of a design with just a microprocessor instead of an additional FPGA is overall decreased complexity of the system. With only an MCU as signal generation device the area of PCB can drastically be decreased. In addition to this advantage is that the software necessary to control the FPGA as well as the MCU needs less attention. Only disadvantage of this solution is the limited flexibility regarding signal forms. With the absence of an FPGA, special wave forms are very restricted.

3.3 Power Supply of Printed Circuit Board

A stable power supply increases accuracy of the overall signal generating hardware. Within this section general ways of supplying the printed circuit board with power are being discussed, as well as transient and constant current supply.

3.3.1 External Power Supply

Using an external power supply to provide a proper voltage level to the noise generating PCB is an obvious solution. Flexibility within the debugging phase of the hardware design and independence from testers environment are advantages. Furthermore power consumption can easily be monitored using an external power supply. Regarding implementation within production environment, an additional attached device is impractical and causes more effort to set up the noise sensitivity measurement environment.

3.3.2 Using Testers Power Supply

The CT1000 smart card tester provides multiple service cables. This service interface also includes several power supplies which can be used by external peripherals on service cable $JS04$. The user power supply is specified to deliver a voltage level of either $15V$ or $5V$. The maximum output current is defined to be $1A$. Therefore a maximum power of $15W$ is available on this interface. For implementation within production environment, an integration into testers power supplies would definitely benefit the overall flexibility and integrability to the tester. [SPEa] [SPE16]

3.3.3 Transient and Constant Current Supply

Under normal conditions, the tester supplies DUT with all necessary voltage levels. While performing noise sensitivity measurements on DUT, additional power gets injected into pin under test. This power is provided by voltage regulators and decoupling capacitors of the external hardware. While constant current supply is delivered by the tester environment, transient current is provided by the noise generating PCB.

3.4 Half-Bridge for Signal Generation

To amplify the signal to the desired voltage level, a half-bridge circuit is used. Within this section reasons for using this kind of electrical circuit and its properties are discussed.

3.4.1 Why using Half-Bridge Design

The reason why a half-bridge design is used to amplify the signal is, because its implementation and hardware complexity is quite simple. Two NMOS transistors are used to define the voltage level on the output pin of the half-bridge stage. By simply modifying the bridge voltage, the high-level output voltage of the half-bridge is changed. This makes adjustments of the signals amplitude very easy. Regarding filter components, like the capacitance used to block DC from pushing into the PCB, it is also necessary to discharge the capacitance, when a low-level needs to be applied. Therefore the NMOS in GND path creates a low-resistance to GND, which discharges the capacitor.

3.4.2 Switching Positive Voltages

Switching positive voltages with a half-bridge is quite simple. There are basically just few things which need to be considered. It is necessary to drive the high-side switch within the half-bridge circuit with a gate voltage V_G , which is higher than the bridge voltage V_{DD}. Otherwise the signal does not get fully amplified by the bridge voltage, because the NMOS is not fully switched on. In addition it is necessary to take into account, that the signal integrity is improving, when the output is switched as slow as possible. If the NMOS transistors are switching with their maximum speed, it is most likely, that this will cause unwanted effects to the signal. This results most definitely in higher chance of ringing and overshoot on the signal. For that reason speed in the signal generation path is restricted in two ways. Firstly, additional resistors are used to decrease the overall charging current of the gate capacitance. Secondly, the gate driving voltage is adjusted according to the half-bridge voltage. With these two modifications it is possible to increase the quality of the signal.

3.4.3 Switching Negative Voltages

With a half-bridge design, as discussed in section 2.11.2, it is also possible to generate signals with negative amplitudes in respect to ground. However switching these negative voltages raise the need to pay more attention on hardware design. As soon as the voltages V_{GS} and V_{DS} are outside the operating limits, the devices are in danger of thermal destruction. Furthermore gate driving capabilities are very important for correct switching behaviour. If the gate drive is too small, the MOSFET is in risk of thermal overload due to high drain to source resistance. Furthermore freewheeling currents through the internal body diode of the NMOS transistor can cause thermal destruction of the device. This may happen, when the slow reverse recovery time of the internal body diode is interfering with the on-switching time of the opposing NMOS within the half-bridge circuit. [Maj16]

Additional Problem

In addition to the previously discussed difficulties certain challenges apply to the circuit design when using negative voltages. Since the adjustment of the signal amplitude should be fully automated, digital potentiometer with SPI are used. Unfortunately operating these potentiometers with negative voltages at their terminals is not possible, because the negative signals are outside of the specified maximum values of the terminals. [Tex16b]

3.4.4 Inverted Gate Signal for Glitch Generation

The output signal of microcontrollers can also be inverted to generate glitches. Consequently MCUs output is switched from high-level to low-level and back to high-level again. This approach makes it very easy to generate glitches, because there is no additional hardware necessary. Therefore overall footprint of the signal generating circuit decreases enormously. Another advantage is the simple implementation by software. Disadvantage of this approach is that it is not possible to reach a low-level below $1.5V$ when the device is operated in voltage class A. Another point which needs to be considered is that the initial state of the capacitor used to block the DC level, has to be changed, when switching from pulse signal to glitch signal. In case of such a change the capacitance needs to be charged to the corresponding initial amplitude value of the signal.

3.4.5 Simulation

In order to simulate the signal generation, an LTSpice simulation was created. With NMOS transistors model integrated within the SPICE software, the half-bridge circuit was simulated. The schematic of this simulation can be seen in figure 3.3, where a pulse is generated. The simulation result can be seen in figure 3.4.

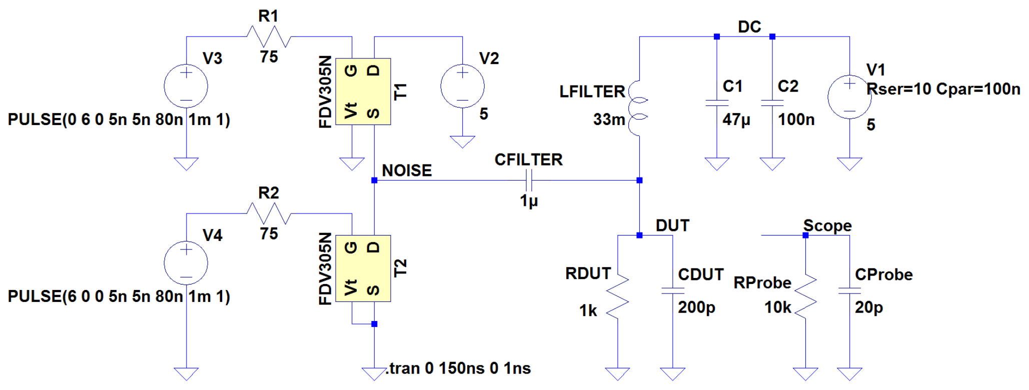


Figure 3.3: SPICE schematic of pulse generating circuit.

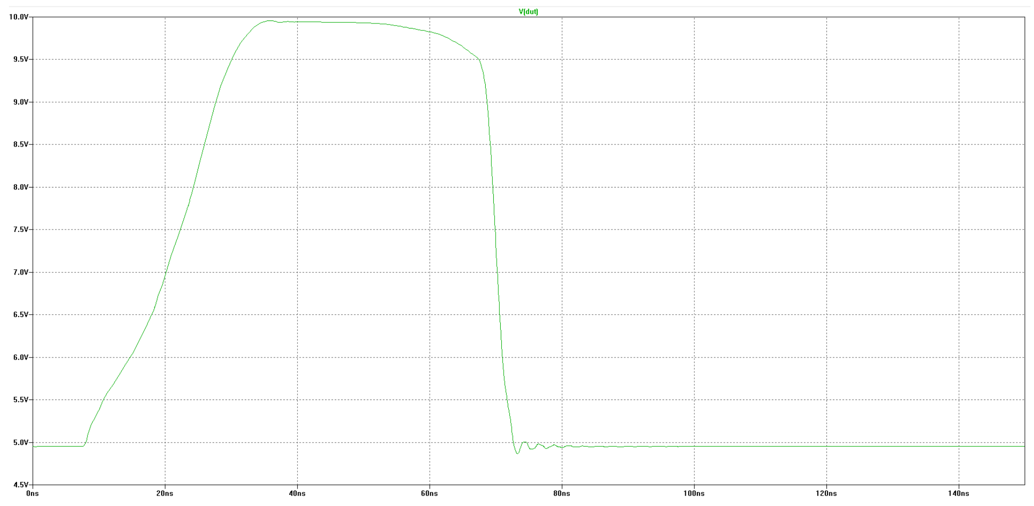


Figure 3.4: SPICE simulation result of pulse generating circuit.

To simulate influences caused by parasitics, additional $10nH$ of inductance were added into each current trace. The schematic used for simulation with parasitics is visualised in figure 3.5. With this minor adjustment, massive effects can be seen at the signal generated by the half-bridge circuit. The signal with influences from inductive parasitics can be seen in figure 3.6. With that simulation result in mind, it is very important to think about possible parasitics on PCB level and how to avoid respectively reduce them.

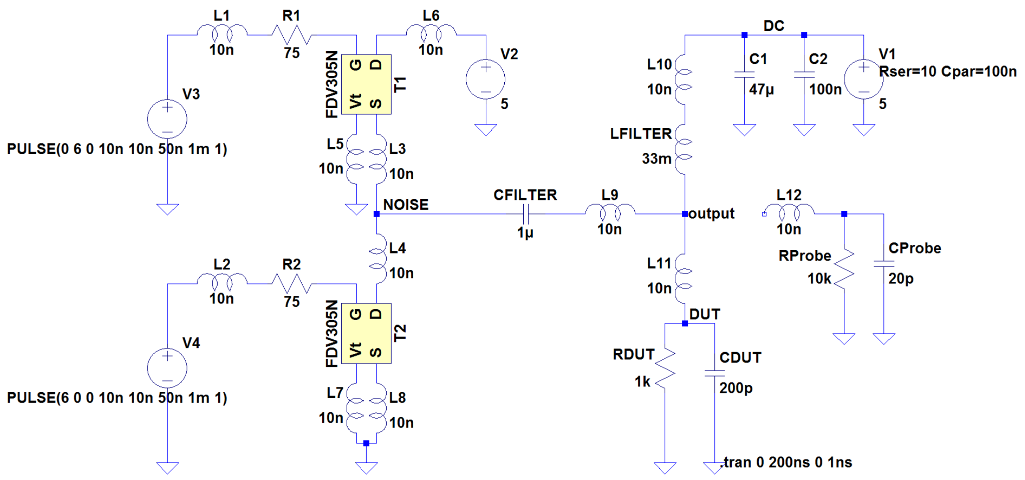


Figure 3.5: SPICE schematic of pulse generating circuit with inductive parasitics.

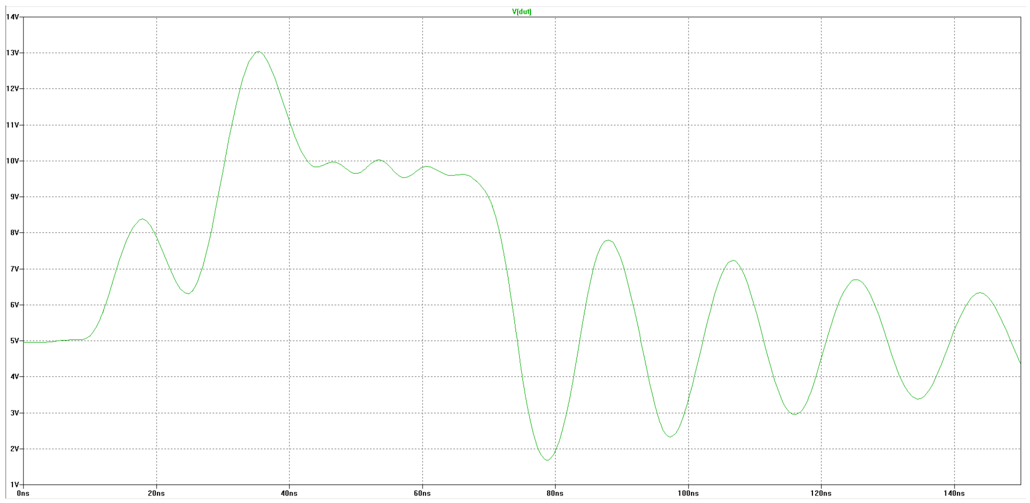


Figure 3.6: SPICE simulation result of pulse generating circuit with inductive parasitics.

3.4.6 Control Half-Bridge Switching with Gate Driver

With the additional use of a gate driver IC it is possible to improve the switching properties of a MOSFET. Typically a gate driver decreases the switching time. For the simple reason that the gate driver provides a high-current gate signal, the gate capacitance of the FET can be charged or discharged more quickly. But faster switching speed mostly comes with side effects. Ringing, over- and undershoot might be an effect caused by the influence of too fast switching transitions. This effect can be seen in figure 3.7. Within this figure violet represents the voltage at the load resistance and green is the output of the half-bridge circuit before the filtering capacitor. It is easy to see, that over- and undershoot is present. Additionally ringing can be seen in the signal. The settings used in the measurement visualised in figure 3.7 are pictured in the table 3.1.

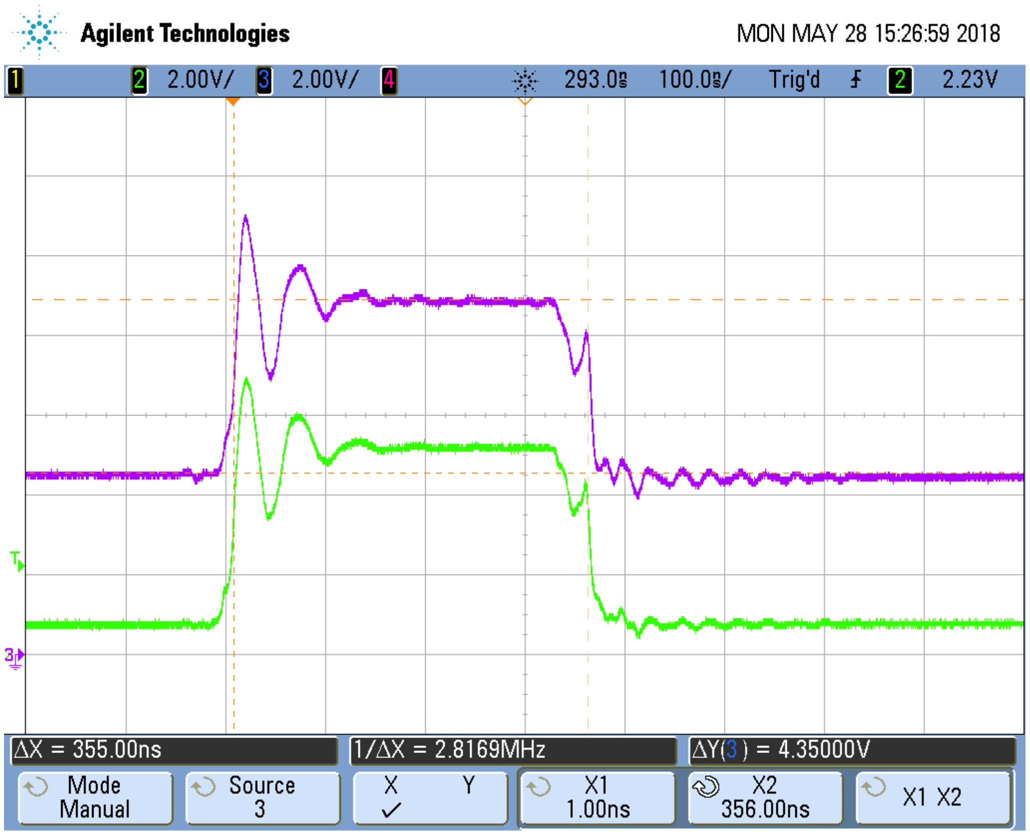


Figure 3.7: Half-bridge controlled with gate driver ($V_{Drive} = 10V$).

Name	Parameter	Value
Half-bridge voltage	V_{Bridge}	$5V$
DC level	V_{DC}	$5V$
Gate driver voltage	V_{Driver}	$10V$
Load resistance	R_{Load}	$1k\Omega$
Filter capacitance	C_{Filter}	$22nF$
Filter inductance	L_{Filter}	$33mH$

Table 3.1: Settings and parameters for gate driver investigation

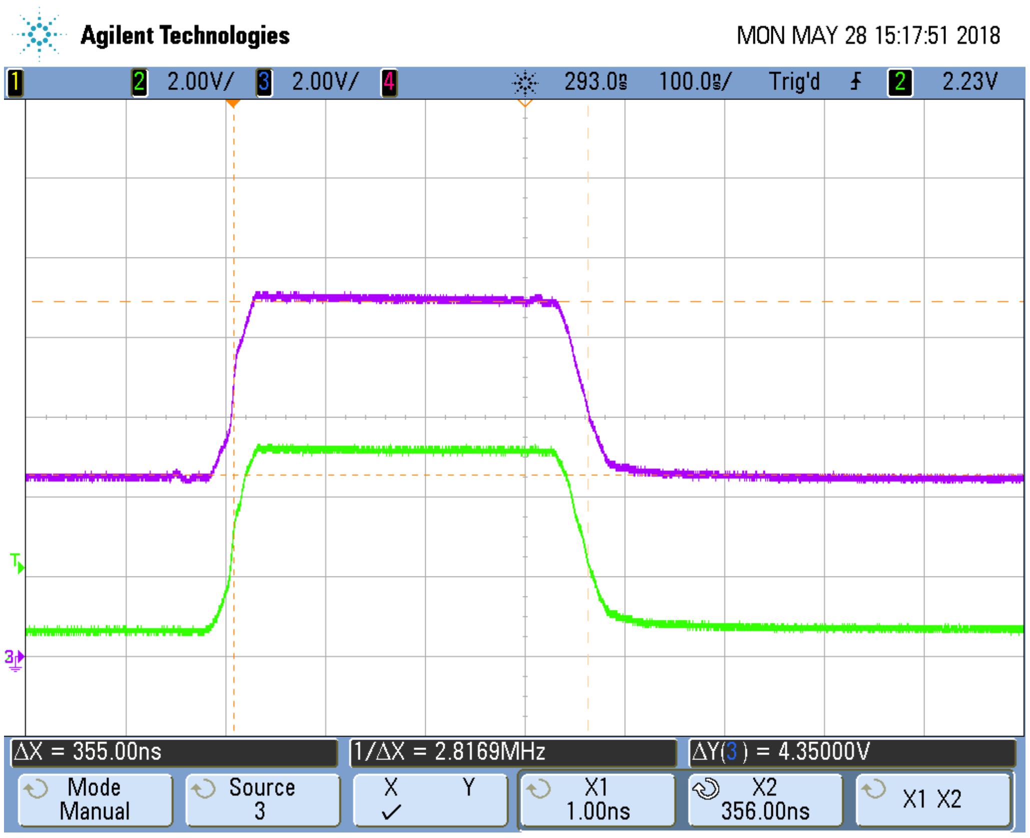


Figure 3.8: Half-bridge controlled with gate driver ($V_{Drive} = 6V$).

Name	Parameter	Value
Half-bridge voltage	V_{Bridge}	$5V$
DC level	V_{DC}	$5V$
Gate driver voltage	V_{Driver}	$6V$
Load resistance	R_{Load}	$1k\Omega$
Filter capacitance	C_{Filter}	$22nF$
Filter inductance	L_{Filter}	$33mH$

Table 3.2: Settings and parameters for gate driver investigation with adjusted driver voltage

In order to reduce these side effects caused by too fast switching of the transistors, a simple approach is used. By reducing the gate driving voltage V_{Driver}, the signal quality increases. Figure 3.8 illustrates, that the effects caused by fast switching have decreased tremendously.

3.5 Switching Matrix

To ensure that the signal is properly transmitted to the DUT, a switching matrix is needed. This switching matrix is controlled via microcontroller. The MCU defines which pin is stressed by the external hardware. Therefore a specific output of the MCU is set. This output signal causes a bipolar transistor to turn on. The switched on bipolar transistor causes a relay to switch its contacts from high-impedance to conducting.

3.5.1 PhotoMOS Relays

For the reason of decreased PCB area, the footprint of the used relays should be as small as possible. For that reason photoMOS relays are the first choice. Disadvantage of photoMOS relays is the parasitic capacitance. This capacitance leads to signal leakage even if the photoMOS is switched off. As a result of this property, photoMOS relays are not suitable of switching high-frequency signals. [Pan14]

Simulation

To investigate the signal switching behaviour of photoMOS relays, the signal generating circuit was modelled and the off-capacitance implemented. This circuit is visualised in figure 3.9. Then the signal seen by DUT was simulated, seen in figure 3.10 and figure 3.11.

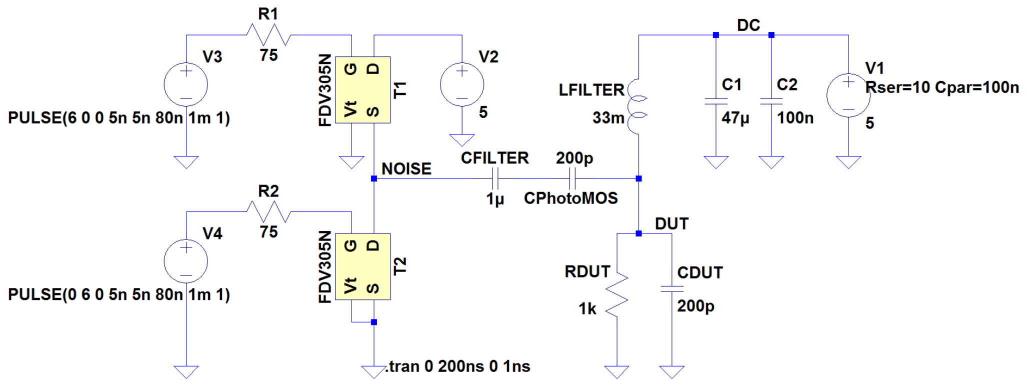


Figure 3.9: Pulse generating circuit used to simulate a switched-off photoMOS relay.

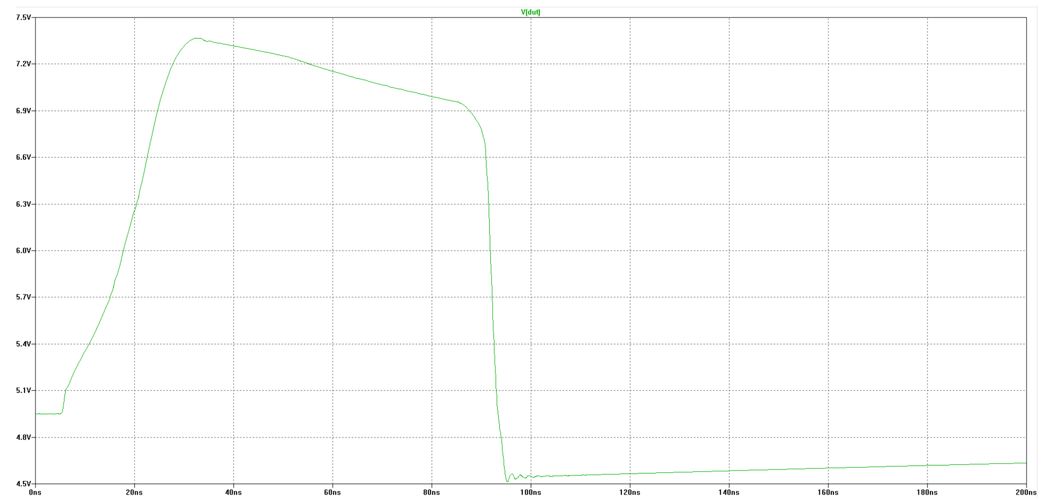


Figure 3.10: Pulse signal at device under test with photoMOS switched-off.

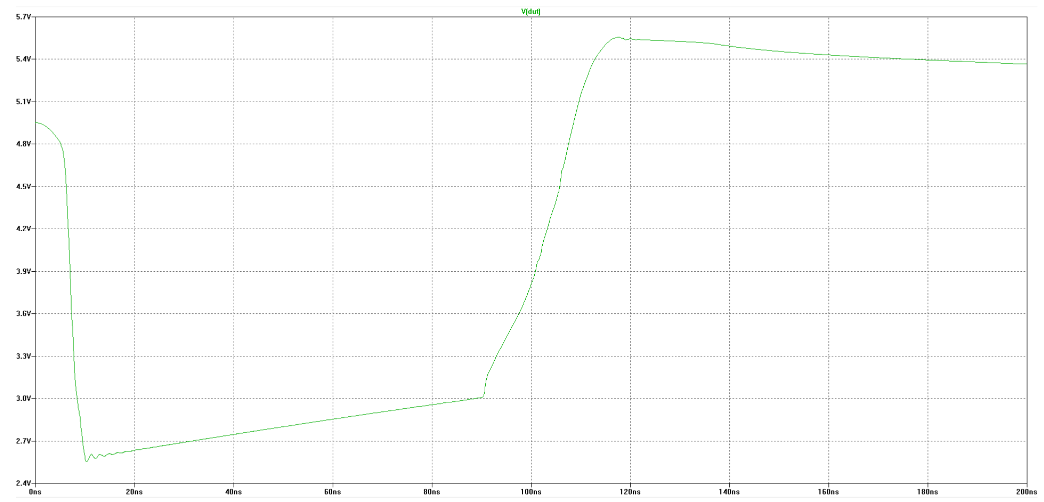


Figure 3.11: Glitch signal at device under test with photoMOS switched-off.

3.5.2 Reed Relays

Low on-resistance, small capacitance and high isolation are promising properties. Therefore reed relays are suitable switching components for this application. There are very small packages available for reed relays. These devices have footprints below $4mm^2$ which is perfectly suitable for a small PCB design. [Pic]

3.6 Voltage Sweep

In order to analyse smart card devices noise sensitivity in more detail, it should be possible to change signal parameters like length and amplitude over time. Since width of the signal can easily be adjusted by modifying MCUs software, alterations of amplitude need additional hardware effort. A voltage regulator with adjustable output voltage needs to be implemented in the final PCB design to add the possibility to sweep the voltage.

Parameter	Value
Minimum Voltage	$1.5V$
Maximum Voltage	$5V$
Steps	16
Resolution	$22mV$

Table 3.3: Voltage regulators output voltage specification

3.6.1 Debug Version

For debugging purposes a synchronous step-down voltage regulator was used. To make the regulators output voltage adjustable, a simple parallel circuit of a fixed resistor and a potentiometer in the feedback path was used. This modification makes it possible to adjust the output voltage by changing the potentiometers resistance.

3.6.2 Digital Potentiometer

After verifying that the signal generated by the printed circuit board can cause resets on smart card devices, the manual approach with a potentiometer needs to be replaced by a fully automated solution. For that reason a digital potentiometer with integrated SPI connectivity was used. This resistance of the digital component is set by sending a bit string from the MCU via SPI.

3.6.3 Disadvantage

Although this setup allows fast and easy adjustment of the voltage regulators output, it causes unintended effects within the signal path. Whenever the output voltage is increased, this increase occurs as an additional pulse on DUT side. Therefore the output voltage of the step-down regulator needs to adjust slower. By adding a resistance into the power path, the capacitors are charged more slowly, which influences the effect seen at device under test. By increasing the charging time of the capacitors, amplitude of the unindented decreases drastically.

3.6.4 Simulation

The simulation of the behaviour described in section 3.6.3 is illustrated in figure 3.12. In figure 3.13 it can be seen, that a fast changing voltage at *noise* input of the capacitor causes big effects seen by DUT. By increasing the time, which is needed to reach the next voltage level, the magnitude of the influence seen at *output* is decreased. This effect is illustrated in figure 3.14.

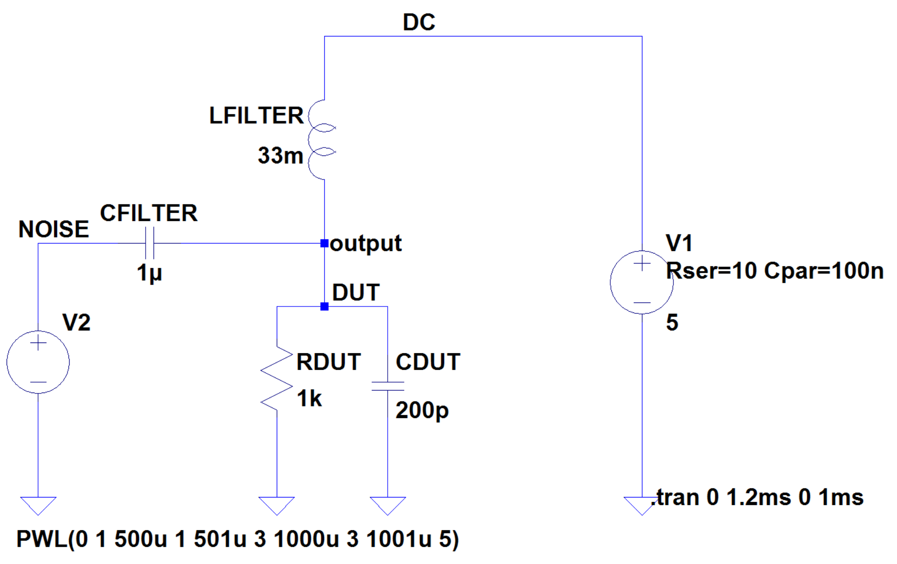


Figure 3.12: Circuit of voltage sweep simulation.

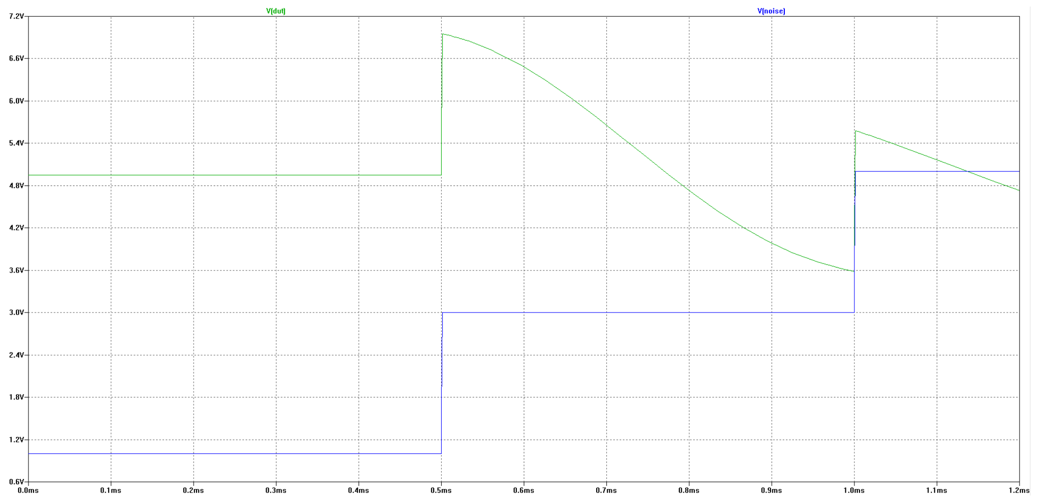


Figure 3.13: Simulation of voltage sweep without precaution (green: V_{DUT}, blue: V_{Sweep}).

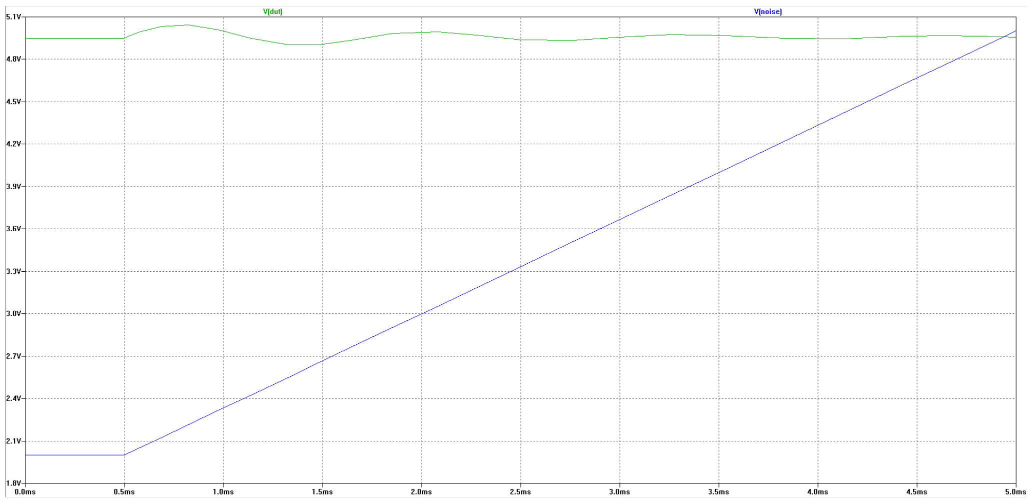


Figure 3.14: Simulation of voltage sweep with precaution (green: V_{DUT}, blue: V_{Sweep}).

3.7 Beta Status Printed Circuit Board

After using a printed circuit board in debug version to verify the PCB design, an optimisation of the hardware is needed. Therefore special attention is paid to decrease the footprint of the PCB, while increasing the overall functionality. Additionally special care was taken, to increase the signal integrity of glitch and pulse signal. Furthermore additional circuitry was designed to increase the flexibility and stability of the noise generating external hardware. Within this section special modifications of the hardware design are being discussed.

3.7.1 Layout Modifications

By modifying the layout of the external hardware it is possible to increase the quality of the generated signal. For that reason it is very important to consider special effort, when working on the layout of the printed circuit board. To gain more stability in voltage supply, traces from voltage regulators are chosen to be as wide as possible. This measure decreases the resistance of the trace, which causes less voltage ripple seen at the current sinking circuits. Furthermore high-speed switching paths are kept as short as possible to provide the best HF signal possible. Additionally HF paths are routed within an inner layer of the PCB. As a result of this, HF signals generated by half-bridge circuit are better shielded against interferences caused by electro-magnetic emission.

3.7.2 Component Choice

In addition to the measures on physical layout level described in the previous section, component choice is important as well. The right choice of basic components, like resistors and capacitors, might have a big impact on the overall performance of the noise generating printed circuit board. Therefore special effort is needed, when selecting the best component types.

Gate Resistors

As already stated in section 3.7.1, HF signal paths need special attention. In addition to the traces themselves, all components within these paths are influencing the overall signal quality. For that reason thin-film type resistors are used in high-frequency signal paths. This type of resistor has several benefits. In respect to their low parasitics, thin-film resistors are the right choice to use in HF circuits. With the use of thin-film type resistors, capacitance and inductance, compared to thick-film type resistors, is reduced. Additionally to this characteristic thin-film resistors are more accurate in respect to their parametric value. Furthermore this type of resistance is causing less noise than comparable technologies. Therefore, they are the first choice for high-precision applications. In respect to the stated characteristics, thin-film type resistors support enhanced signal properties.

Filter Capacitance

For the same reasons as choosing the correct gate resistor type, capacitors need to be well chosen. Therefore, the capacitor used to block DC signals from the noise generating circuit, should have enhanced properties in respect to parasitics and tolerances. For that reason a MLCC type capacitor is used. These multilayer ceramic chip capacitors stand out with low parasitics, like ESR and ESL. This means, that equivalent series inductance and resistance of this capacitor type is very low. For that reason, MLCC type capacitors are well suited for HF applications.

3.7.3 Discharging Circuit

In order to increase flexibility of the noise generating hardware, a discharging circuit is implemented. With a fully automated voltage sweep functionality, the need of this discharging circuit arises. For the simple reason, that adjustments of the signal voltage happen within few microseconds, decoupling capacitors need to be discharged. Decoupling capacitors need to be discharged to the initial voltage level while using the voltage sweep. Therefore this additional discharging circuitry is implemented into the printed circuit design. In order to discharge the capacitors within the setup phase of the measurement the value of the discharging resistors needs to be selected carefully.

3.7.4 Firmware Update Interface

In order to provide the possibility to update microcontrollers firmware, an update interface is implemented. This additional circuit allows to update software running on the system. Furthermore this creates the potential to integrate new functions or to adapt the system to new specifications. With this firmware update interface in place the noise generating printed circuit board gains more flexibility and changeability.

Chapter 4

System Description

In this chapter the system used to provoke and observe resets on smart card devices is described. Overall system interaction and necessary settings as well as connectivity to the external hardware and analog measurement of current consumption is discussed within this chapter. Furthermore the noise generating printed circuit board and software running on the microcontroller are characterised.

4.1 Overall System Interaction

Overall system interaction is illustrated in figure 4.1. The difference of this setup, compared to the laboratory measurement described in section 2.6 is, that it indicates a reset via a change in current consumption. The laboratory setup on the other hand indicates a reset with a received ATR response.

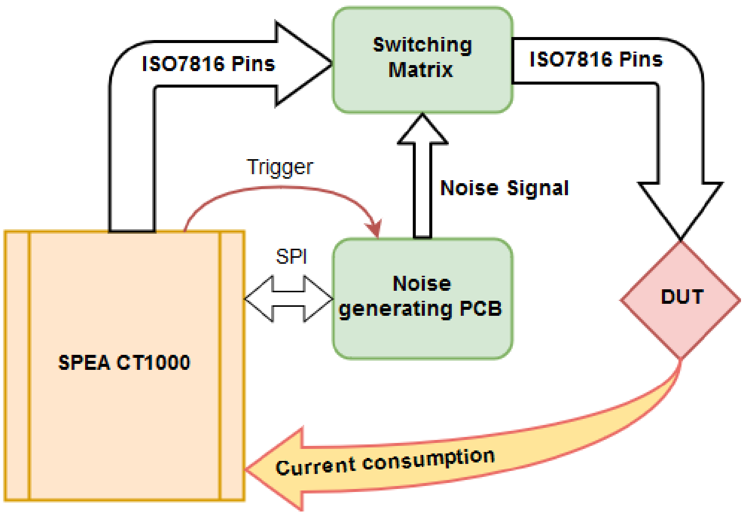


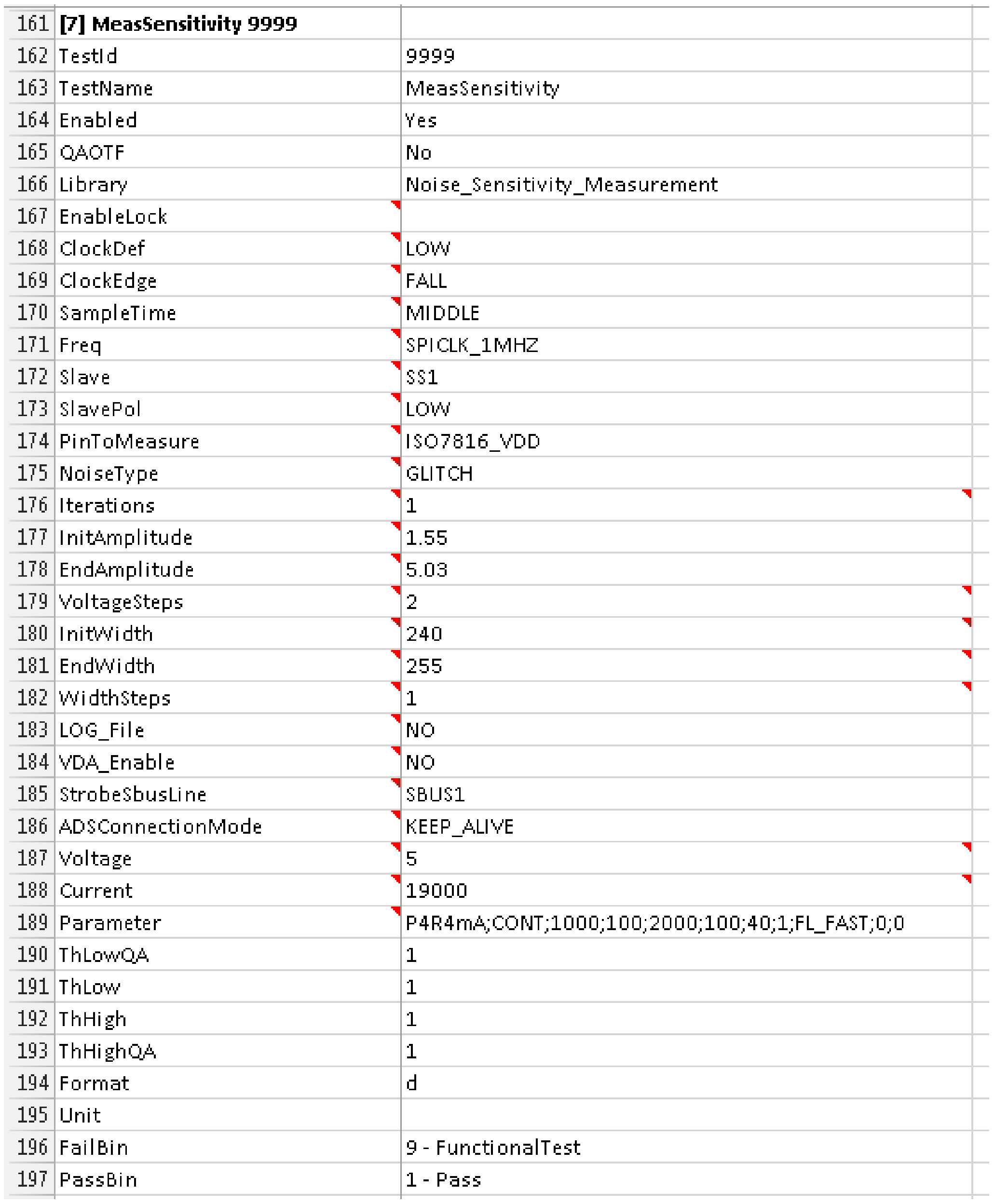
Figure 4.1: Overall system interaction.

4.2 Tester Integration

Within this section the integration of the external hardware into the test program will be described in more detail. The used test model to modify parameters for noise sensitivity measurement will be described. Furthermore details about modifications within the test program itself will be given.

4.2.1 Test Model

Within the newly created test model *Noise Sensitivity Measurement*, all parameters necessary to perform noise sensitivity measurement are defined. Furthermore these parameters are editable within the test plan. Here all data, which is transferred onto the signal generating PCB via SPI, can be adjusted to the desired values. The parameters within the test model *Noise Sensitivity Measurement* can be seen in figure 4.2.



161	**[7] MeasSensitivity 9999**	
162	TestId	9999
163	TestName	MeasSensitivity
164	Enabled	Yes
165	QAOTF	No
166	Library	Noise_Sensitivity_Measurement
167	EnableLock	
168	ClockDef	LOW
169	ClockEdge	FALL
170	SampleTime	MIDDLE
171	Freq	SPICLK_1MHZ
172	Slave	SS1
173	SlavePol	LOW
174	PinToMeasure	ISO7816_VDD
175	NoiseType	GLITCH
176	Iterations	1
177	InitAmplitude	1.55
178	EndAmplitude	5.03
179	VoltageSteps	2
180	InitWidth	240
181	EndWidth	255
182	WidthSteps	1
183	LOG_File	NO
184	VDA_Enable	NO
185	StrobeSbusLine	SBUS1
186	ADSConnectionMode	KEEP_ALIVE
187	Voltage	5
188	Current	19000
189	Parameter	P4R4mA;CONT;1000;100;2000;100;40;1;FL_FAST;0;0
190	ThLowQA	1
191	ThLow	1
192	ThHigh	1
193	ThHighQA	1
194	Format	d
195	Unit	
196	FailBin	9 - FunctionalTest
197	PassBin	1 - Pass

Figure 4.2: Test model *Noise Sensitivity Measurement* shown in test plan editor.

Parameters

Since the parameters used to define the noise sensitivity measurement on smart card devices, have multiple possible values, a short explanation is necessary. Further description of the parameters and possible values can be seen in table 4.1.

Parameter	Possible Values	Description
ClockDef	HIGH, LOW	Clock definition (SPI)
ClockEdge	RISE, FALL	Clock edge (SPI)
SampleTime	MIDDLE, END	Sample time (SPI)
Freq	SPICLK_1MHZ, SPICLK_2MHZ, SPICLK_5MHZ, SPICLK_10MHZ	Clock frequency (SPI)
Slave	SS1, SS2, SS3, SS4	Slave select (SPI)
SlavePol	LOW, HIGH	Slave polarity (SPI)
PinToMeasure	ISO7816_VDD, ISO7816_RST, ISO7816_IO, ISO7816_CLK	Defines ISO7816 pin to measure
NoiseType	PULSE, GLITCH	Defines noise signal type
Iterations	1 to 15	Iterations of measured point
InitAmplitude	5.03, 4.75, 4.55, 4.31, 4.12, 3.85, 3.61, 3.42, 3.15, 2.91, 2.72, 2.45, 2.25, 2.03, 1.75, 1.55	Initial amplitude value in volt
EndAmplitude	5.03, 4.75, 4.55, 4.31, 4.12, 3.85, 3.61, 3.42, 3.15, 2.91, 2.72, 2.45, 2.25, 2.03, 1.75, 1.55	End amplitude value in volt
VoltageSteps	0 to 15	Steps from initial to end value
InitWidth	0 to 255	Initial Width in counter value
EndWidth	0 to 255	End Width in counter value
WidthSteps	0 to 15	Steps from initial to end value
LOG_File	YES, NO	Enables log-file
VDA_Enable	YES, NO	Enables VDA-tool
StrobeSbusLine	SBUS1, SBUS2, SBUS3, SBUS4, SBUS5, SBUS6, SBUS7, SBUS8, SBUS9, SBUS10	Defines strobe output to SBUS
AdsConnectionMode	ADS, ADS_HF, ADS_FORCE_V, ADS_HS, ADS_HOT_GND, HOT_GND, ADS_COLD_GND, COLD_GND, KEEP_ALIVE, NONE	Analog driver connection mode
Voltage	-8 to 8	Devices V_{DD} level in volt
Current	-500 to 100000	Current limit in microampere
Parameter	*multiple*	Timing and sample parameters

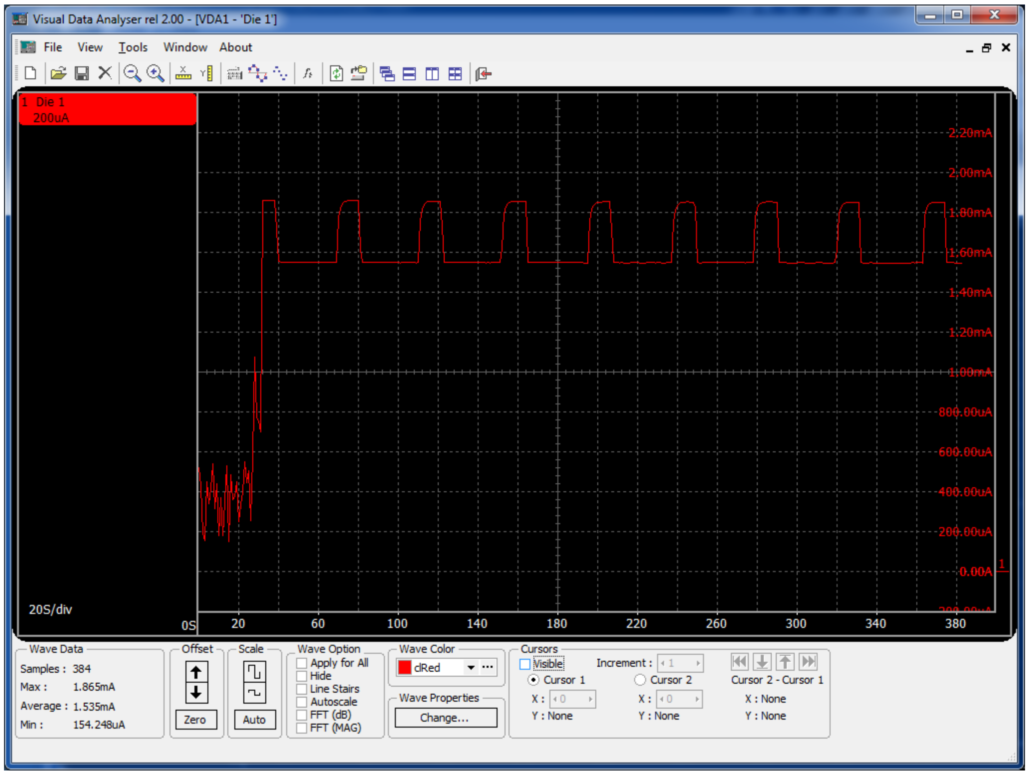
Table 4.1: Parameters of test model *Noise Sensitivity Measurement*.

Logging

The parameter *LOG_File* enables or disables the functionality to create log-files. Within these log-files all measurement data is stored as plaintext. These files are used to record current traces in text form. In order to visualise current traces these files can be opened by the Visual Data Analyzer (VDA) tool. This makes investigations of current consumption at a later point in time possible.

VDA functionality

The parameter *VDA_enable* is used to activate and deactivate the VDA functionality. With this parameter enabled, a window on the tester PC pops up, where the current trace of the latest touchdown is visualised after performing noise sensitivity measurement. Such a visualised VDA current trace can bee seen in figure 4.3.



Wave Data
Samples : 384
Max : 1.865mA
Average : 1.535mA
Min : 154.248uA

Wave Color: dRed

Cursor 1 X : 0 Y : None
Cursor 2 X : 0 Y : None
Cursor 2 - Cursor 1 X : None Y : None

Figure 4.3: Current trace visualised by VDA tool.

4.2.2 Test Program

In order to allow proper measurement of smart card devices current consumption, modifications within the test program are necessary. For the reason of alignment between tester and printed circuit board, special timing and trigger methods are very important. Additionally communication between these two separate systems needs to be guaranteed. Finally, measurement of devices current consumption needs to be in place. More details about these alterations are being discussed within this section.

Triggering Printed Circuit Board

For the reason of alignment of CT1000 tester and noise generating printed circuit board, special triggering is implemented within the test program. Therefore an analog signal generates a rising edge. This signal is connected via an output of the digital driver board (DDS) to the printed circuit board. Whenever a rising edge is detected by the noise generating printed circuit board, the next step in the measurement flow will be triggered. Furthermore, there are three dominant trigger events, of whom some may be repeated several times. Firstly, there is the main trigger. This rising edge notifies the PCB to switch into communication mode. Whenever the main trigger rises, the noise generating printed circuit board is ready to send and receive data via SPI. Secondly, there is a trigger, which indicates additional SPI commands. Whenever this signal rises, the PCB is ready to send and receive additional bytes via serial peripheral interface until an EOC command is received. The third trigger is then covering the measurement itself. Whenever this trigger is applied to the PCB, a single measurement will be performed. This means, that n triggers are necessary to successfully perform n iterations of the measurement. With these triggers precise timing of stressing signal and measurement is possible.

SPI Communication

In order to allow proper setup of the measurement, SPI communication is integrated in the test program. With this interface, parameters defined within the test plan, are packed into bytes and sent to the noise generating printed circuit board. These parameters can be seen in figure 4.2. Starting with a handshake byte, SPI communication is started. After this byte there are several bytes which transfer all parameters defined within the test plan. To terminate communication, a so called end of communication (EOC) command is sent. More detail about the procedure of SPI communication is covered in section 4.6.1.

Multi-Point Current Measurement

Most important modification within the test program is the implemented multi-point current measurement. With this analog measurement it is possible to investigate noise sensitivity of smart card devices by observing the current consumption of the device. For that reason, phases and strobes of the analog instrument (ADS) are aligned with the implemented triggers. Therefore it is possible to investigate the current consumption of the smart card device directly after the stressing signal was applied. With this setup it is currently possible to acquire 32 consecutive analog values. With an acquisition time of $40\mu s$ per sample, this means that current consumption is measured for $1.28ms$. As indicated in section 2.9, the period time of the signal should be between $500\mu s$ and $1ms$. Since this is the time a smart card device approximately needs to fully restart, the implemented measurement covers this whole period.

4.3 Printed Circuit Board to DUT Interface

In order to ensure connection from the noise generating PCB to the device under test, special additional adapters at the tester to probe card interface are necessary. These adapter boards allow to contact each pin and consequently all channels of the tester interface. Thus the signal can be injected into the device under test. The discussed adapter printed circuit boards are seen in figure 4.4 and figure 4.5. To define the affiliation from adapter PCB to tester interface, table 4.2 and table 4.3 are determined.

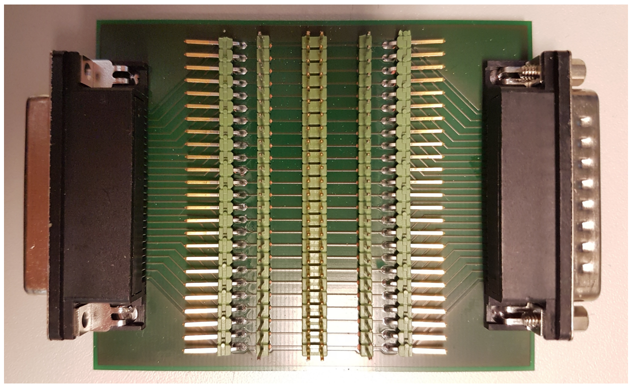


Figure 4.4: Adapter printed circuit board to connect JANL interface of the tester.

Smart Card interface	JANL01 - JANL08

Table 4.2: Supported interfaces with 25 pin JANL adapter PCB.

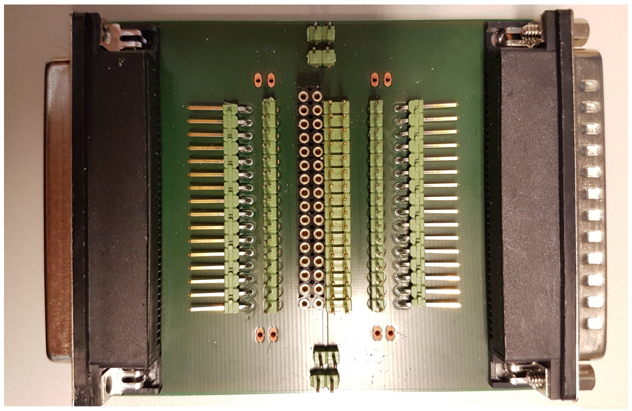


Figure 4.5: Adapter printed circuit board to connect JCH interface of the tester.

Smart Card interface	JCH01 - JCH08

Table 4.3: Supported interfaces with 37 pin JCH adapter PCB.

4.4 Noise Generating Printed Circuit Board

For signal generation a printed circuit board is designed. Dimensions of the latest revision of the noise generating printed circuit board are 101mm x 81mm. The small overall area of the PCB allows easy and comfortable integration within the existing testing environment. In order to supply the device with power, two interfaces are available. On one side there is a DC socket. With this socket it is possible to use a standard power supply. On the other side there are two banana sockets available, which can be used to connect the PCB to a laboratory power supply. Furthermore this 4 layer PCB consists of multiple different blocks, which will be described within the following section. Since the newest revision of the noise generating printed circuit board is not fully verified at this point of time, all following verifications were made with a predecessor PCB. All schematic and layout files of revision C can be found in appendix A.

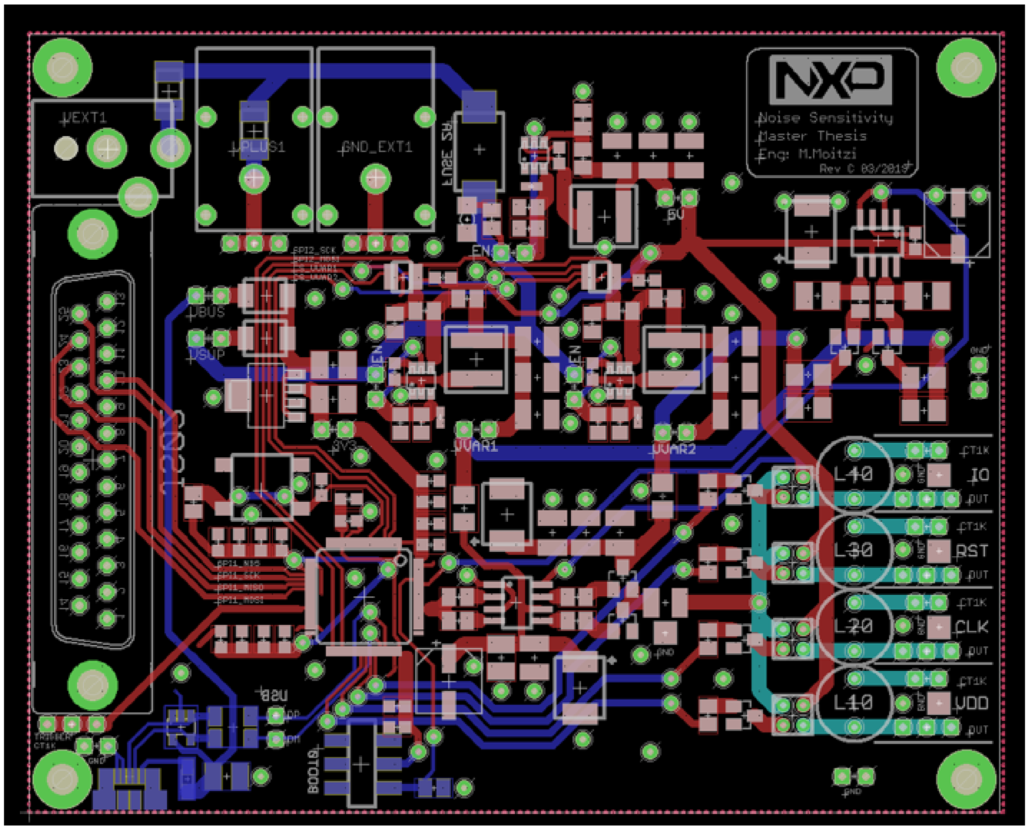


Figure 4.6: Layout of revision C of printed circuit board.

4.4.1 Voltage Supplies

To supply power to all blocks on the printed circuit board, several voltage supply circuits are implemented on the PCB. Firstly, a 3.3V LDO regulator is implemented. This low-dropout voltage regulator is responsible for supplying the microcontroller. To enable firmware updates without powering the whole PCB, the LDO can also be powered via USB. This LDO can be seen in figure 4.7. [Tex15]

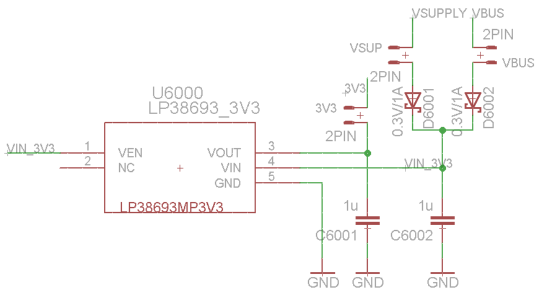


Figure 4.7: Low-dropout regulator used to power the microcontroller.

Secondly there is a synchronous step-down voltage regulator, which is illustrated in figure 4.8. This voltage regulator supplies the gate driving ICs with the appropriate voltage level. For that reason the output voltage of the TPS565208 step-down converter is adjustable. This is accomplished by defining the default output voltage V_{Var1} of the regulator to $6V$. With an additional resistor and an parallel connected digital potentiometer, the voltage V_{Var1} is adjustable from $2.5V$ to $6V$ with a resolution of $22mV$. This digital potentiometer can be seen in figure 4.10. The calculation of the output voltage resistors is described in the following equations.

$$V_{OUT} = 0.760 \cdot (1 + \frac{R_1}{R2}) \tag{4.1}$$

With the selection of $R_2 = 8.2k\Omega$ and $V_{OUT} = 6V$, R_1 results in $56.5k\Omega$. Since this value is not present in any E series of preferred numbers, $R_1 = 56.2k$ was selected. The inductance L_1 was selected in respect to the data sheet. Furthermore any additional components were selected considering the data sheet. In order to make this voltage regulator adjustable, additional circuitry is needed. For this reason the value of R_2 plus extra resistance should cause the voltage regulator to supply a lower voltage at the output. For that reason, the output voltage was assumed to be $2.5V$. With $R_1 = 56.2k\Omega$ and $V_{OUT} = 2.5V$, the resistance is defined to be $R_2 = 24.5k\Omega$. With an $8.2k\Omega$ resistance already in place, an additional resistance is needed, which varies from zero to $16.3k\Omega$. For that reason, two resistors are connected in parallel. With a digital $100k\Omega$ potentiometer selected, the parallel resistance necessary to provide $R_3 = 16.3k\Omega$ is calculated to be $R_{Par} = 19.5k\Omega$. In respect to the E series of preferred numbers, $R_{Par} = 19.6k\Omega$ was selected. An overview of all selected values can be seen in table 4.4

$$R_3 = \frac{R_{Poti} \cdot R_{par}}{R_{Poti} + R_{par}} \tag{4.2}$$

$$R_{Par} = \frac{R_{Poti} \cdot R_3}{R_{Poti} - R_3} \tag{4.3}$$

Resistor	Value
R_1	$56.2k\Omega$
R_2	$8.2k\Omega$
R_{Par}	$19.6k\Omega$
R_{Pot}	$100k\Omega$

Table 4.4: Selected parametric values for voltage regulator supplying gate driver IC.

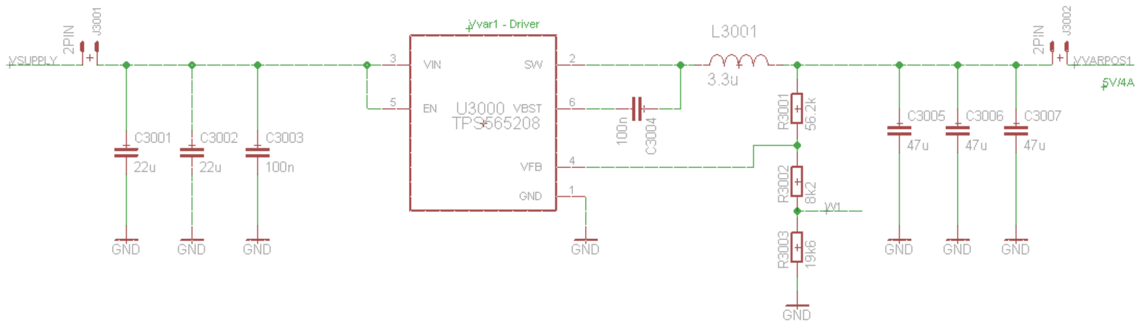


Figure 4.8: Adjustable voltage supply for gate driver IC.

Next there is an additional synchronous TPS565208 step-down voltage regulator, which is basically the same as the regulator supplying the gate driver. This circuit can be seen in figure 4.9. The difference compared to the voltage regulator supplying V_{Var1} is that the maximum voltage level of V_{Var2} is defined to be $5V$, while the minimum voltage is specified to be $1.5V$. For that reason, the values in table 4.5 for resistances were calculated.

Resistor	Value
R_1	$56.2k\Omega$
R_2	$10k\Omega$
R_{Par}	$80.6k\Omega$
R_{Pot}	$100k\Omega$

Table 4.5: Selected parametric values for voltage regulator supplying half-bridge voltage.

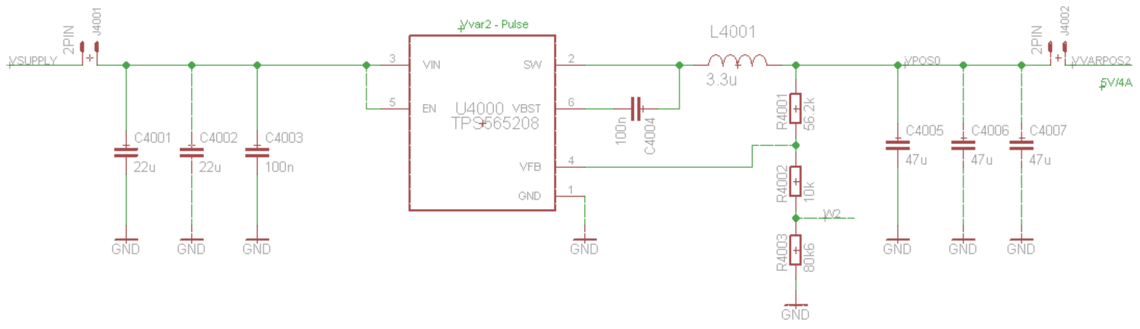


Figure 4.9: Adjustable voltage supply for half-bridge voltage.

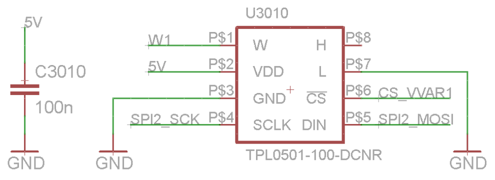


Figure 4.10: $100k\Omega$ digital potentiometer used for voltage adjustments.

Finally there is another TPS565208 voltage regulator with fixed output of $5V$ to supply the reed relays used in the switching matrix. This circuit is basically dimensioned like the voltage regulator which is supplying V_{Var2}. Therefore the calculations can be realised like in formula 4.1. This $5V$ supply is illustrated in figure 4.11. [Tex18]

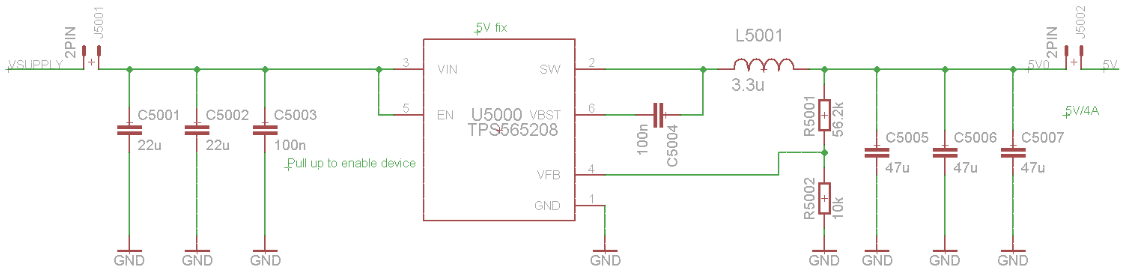


Figure 4.11: Step-down voltage regulator to supply the switching matrix.

Reducing Speed of Charging Capacitors

Fast adjustments of half-bridge voltage causes unwanted spikes on the corresponding pin. For that reason, charging of decoupling capacitors is slowed down by using additional resistors. With the equation to charge a capacitor, the resistance required is calculated.

$$V_C = V_0 \cdot (1 - e^{-\frac{t}{R \cdot C}}) \tag{4.4}$$

With $C = 150\mu F$, $t = 5ms$, $V_0 = 5V$ and $V_C = 1.5V$ selected, the additional resistance $R = 93.5\Omega$ is calculated by the following formula. The resistance value selected is $R = 93.1\Omega$.

$$R = -\frac{t}{C \cdot \ln(1 - \frac{V_C}{V_0})} \tag{4.5}$$

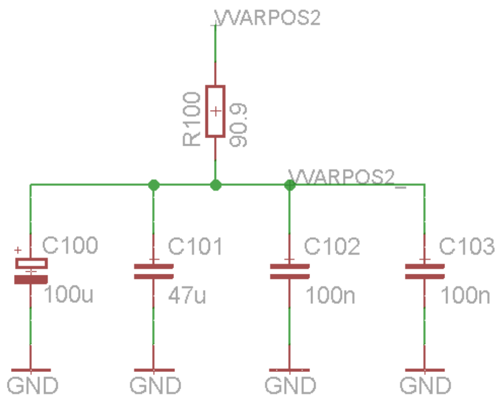


Figure 4.12: Additional resistance used to reduce the charging speed of capacitors.

4.4.2 Discharging Circuit

By allowing to sweep different voltage levels, a discharging circuit is required to discharge all capacitors to the initial voltage level. This is necessary to provide initial voltage level at the beginning of the next measurement. For that reason, resistors are connected to decoupling capacitors of the voltage supply. Whenever a voltage sweep is finished, the microcontroller enables discharging of the capacitors. Therefore a resistance is connected to the capacitors and switched with a MOSFET. In order to make the discharge time as slow as possible, the chosen resistance needs to be calculated carefully. For that reason, the formula to discharge a capacitance is used to calculate the correct resistive value. For this calculation the worst case was assumed, which means to discharge the capacitors from $5V$ to $1.5V$.

$$V_C = V_0 \cdot e^{-\frac{t}{R \cdot C}} \tag{4.6}$$

With $t = 70ms$, $C = 290\mu F$, $V_0 = 5V$ and $V_C = 1.5V$ the resistance necessary to discharge the capacitors in the specified time is $R = 200\Omega$. In order to increase maximum power dissipation of this discharging resistance, two 402Ω resistors were selected and connected in parallel.

$$R = -\frac{t}{C \cdot \ln \frac{V_C}{V_0}} \tag{4.7}$$

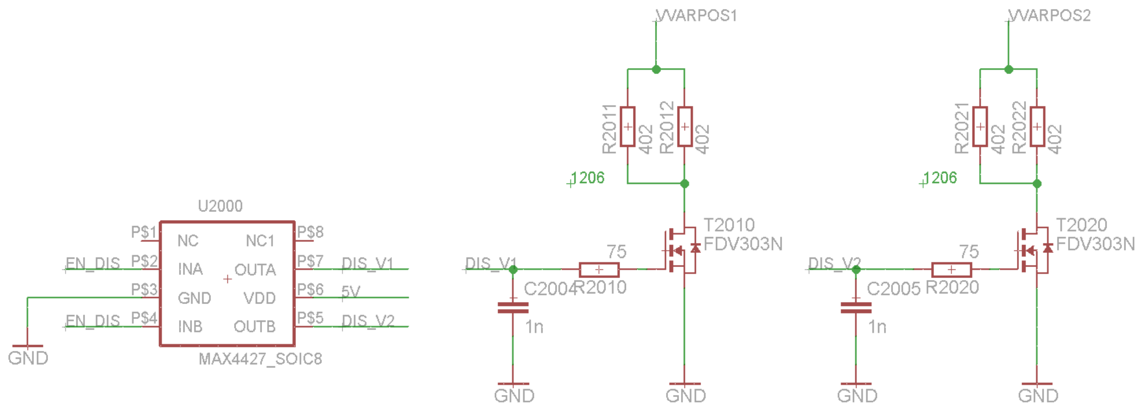


Figure 4.13: Circuit to discharge capacitors to initial voltage level.

4.4.3 Microcontroller

a microcontroller is used for signal generation and communication with the smart card tester. The STM32F072RBT6 is capable of running different interfaces, like USB and SPI. These interfaces are used to allow connectivity to the outer world. The main purpose of the MCU is communication with the tester and signal generation. Additionally the microcontroller is responsible for setting up the voltage levels of all necessary voltage regulators. Specific decoupling capacitors are used to provide a stable supply to the microcontroller. For restarting purposes of the MCU a reset button is implemented within the design. Additionally there is a switch, which allows to activate the so called boot-loader mode of the microcontroller. With this mode activated it is possible to upgrade the software running on the MCU by downloading firmware via the USB interface. [ST 17]

4.4.4 Firmware Update Interface

In order to allow software and firmware updates, an USB interface is implemented on the noise generating printed circuit board. With the use of this additional interface, microcontrollers software can easily be updated. Therefore a resistor is needed to pull the $BOOT0$ pin to ground. With a switch it is possible to connect a 510Ω pull-down resistor to MCU $BOOT0$ pin. In order to download new software, the microprocessor needs to be reset then. With the resistance connected to $BOOT0$ pin, the microcontroller boots into the boot-loader mode. Within this mode it is possible to update firmware and software. To run new software updates the boot-loader switch needs to be switched off again. After a reset of the microcontroller the MCU is running its new software.

4.4.5 Serial Periphal Interface

To allow communication of the smart card tester with the noise generating printed circuit board, a D-sub connector is placed on the PCB. At this connection the $JS02$ interface of the CT1000 tester is plugged in. Therefore it is possible to send all necessary parameters for the measurement from the tester to the noise generating printed circuit board via SPI. This interface is then directly connected to the microcontrollers corresponding serial peripheral interface pins. Additionally to this interface, there is another serial peripheral interface integrated on the microcontroller. With the additional SPI pins the digital potentiometers are connected to the microcontroller. Therefore, resistance can be automatically adjusted by the MCU.

4.4.6 Signal Generating Half-Bridge

Additional hardware is necessary to amplify microcontrollers output signal. For that reason a half-bridge circuit is implemented in PCB design. To allow fast switching times, an LM5110 gate driving IC is used. This module supplies the high current needed to charge the gate capacitance of the FDV303N NMOS transistor. To reduce the speed of the signal generating circuit, additional resistors are placed in the current trace. To adjust the switching speed even further, the supply voltage of the gate driving IC is modified to tune the shape of the signal. Another characteristic which needs to be considered is the parasitic inductance in resistor components. Standard thick-film resistors tend to have relatively high parasitic inductance. For the reason of reduced parasitics, thin-film resistors are used in signal paths. The whole half-bridge circuit including the gate driver IC is illustrated in figure 4.14. [Tex16a] [ON 17]

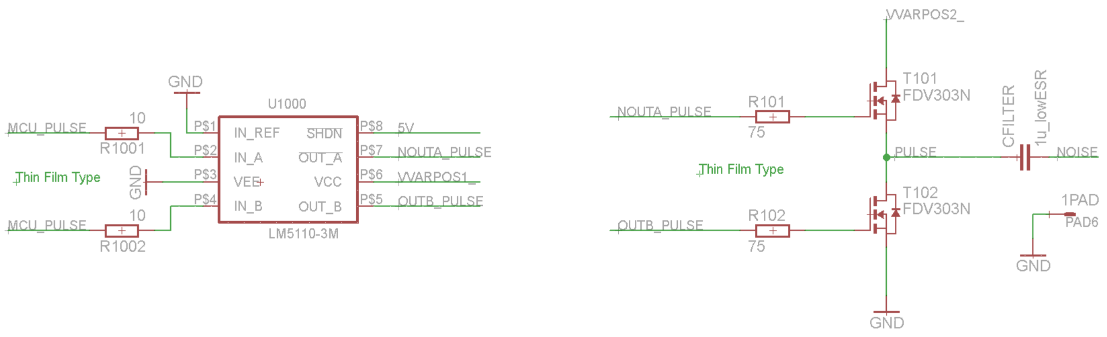


Figure 4.14: Gate driver IC and half-bridge circuit.

4.4.7 Direct Power Injection

A proper filter design is vital to guarantee the capability of providing a suitable signal to the device under test. Therefore the parametric calculations of filtering components need special attention. To ensure proper blocking of DC level and HF signal, impedance of the corresponding component should be around the value of load resistance times ten at the operating frequency. [Gun06]

$$X_{L/C} = 10 \cdot R_{Load} \tag{4.8}$$

With a load resistance of approximately $R_{Load} = 1k\Omega$, the impedance of the filtering components should reach at least $10k\Omega$ within the specified frequency range. With calculations using the following equations values of capacitor and inductor were found.

$$L = \frac{X_L}{2 \cdot \pi \cdot f} \tag{4.9}$$

$$C = \frac{1}{2 \cdot \pi \cdot f \cdot X_C} \tag{4.10}$$

With $C = 1\mu F$ and $L = 33mH$ selected, the following impedance behaviour of the filtering circuit is given:

$$X_L \geq 10k\Omega \quad for\ f \geq 50kHz \qquad with\ L = 33mH \tag{4.11}$$

$$X_C \geq 10k\Omega \quad for\ f \leq 16Hz \qquad with\ C = 1\mu F \tag{4.12}$$

With the calculated values from this section, it is possible to provide the desired signal levels at device under test. Nevertheless, the assumption about equivalent resistance of a smart card device is just an approximation. With this calculation it is possible to specify the approximate behaviour of the filtering components.

4.4.8 Switching Matrix

As already discussed in section 3.5, reed relays are used to connect the signal to the corresponding ISO7816 pin. This circuitry can be seen in figure 4.15. Additional transistors are used to activate these electrical switches. These bipolar BC817 transistors allow to activate reed relays with microcontrollers output. Therefore a resistance is used to define the operating point of this bipolar transistor. The resistance is calculated in the following formula, with $V_{BE} = 1.2V$, $I_B = 1mA$ and $V_{MCU} = 3.3V$. [Mul13]

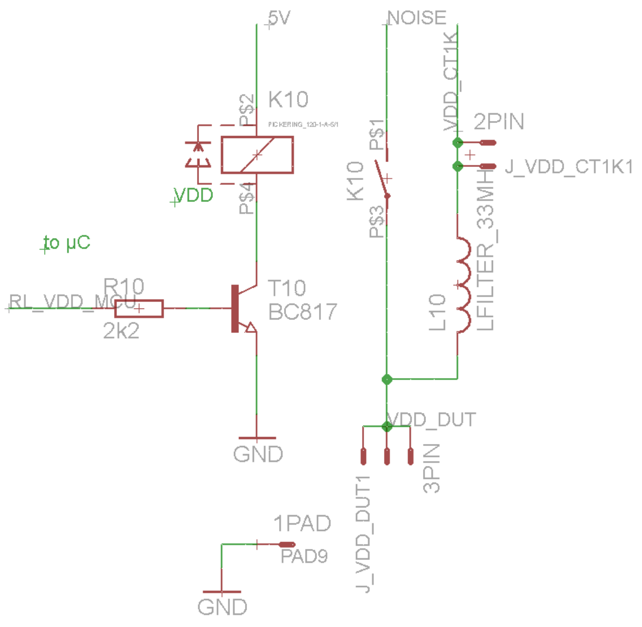


Figure 4.15: One pin of the switching matrix.

$$R_V = \frac{V_{MCU} - V_{BE}}{I_{BE}} \tag{4.13}$$

When using this formula, R_V is calculated to be 2100Ω. In respect to the E series of preferred numbers, $R_V = 2.2k\Omega$ was selected.

4.4.9 Layout

For increased signal quality and stability, some layout design guidelines need to be considered. Paths with high switching frequency need to be placed carefully. These traces should be as short as possible. Additionally high-frequency traces are routed in inner layers of the PCB. The inner layer is used to provide additional shielding against electromagnetic interference. With GND filled on all the other layers, the high-frequency signal is shielded against influences of electromagnetic emission.

4.5 Microcontroller Software Implementation

Software running on the microcontroller handles several different tasks. First of all, SPI communication is implemented. Within this part of the software, all necessary data to perform noise sensitivity measurements are loaded into the microcontroller. After receiving all commands, the data is broken down into each parameter. After that, the system is initialised and ready to perform measurements. In figure 4.16 this flow is illustrated. Additionally more detail will be given within this section.

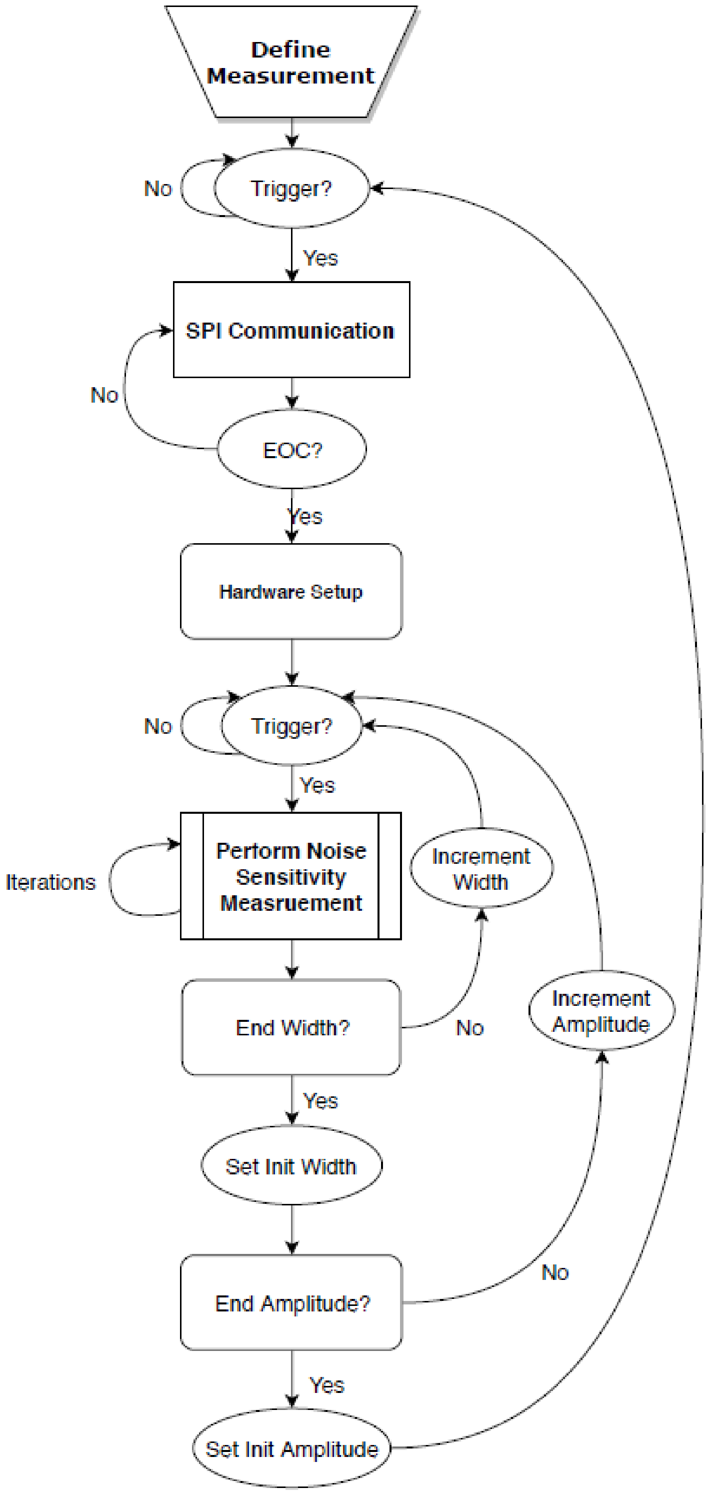


Figure 4.16: Flowchart of the overall software system interaction.

4.5.1 Communication with Tester

Serial peripheral interface is used to allow data exchange from SPEA CT1000 tester to the external noise generating hardware. In order to provide controlled communication, a special transmission protocol is used. This defined communication flow is illustrated in figure 4.17. First of all, the microcontroller waits for a trigger, which indicates that SPI transmission will begin shortly. This trigger is followed by a so called handshake byte. With this command the tester is checking if the noise generating printed circuit board is connected. After a successful handshake, the first command is sent. This byte contains information about signal type, ISO7816 pin and initial amplitude of the signal. Subsequently the second command is being transmitted. This command defines the initial signal width. The third command contains information about number of iterations and signals end amplitude. Then the end signal width is determined with command number four. After that the fifth command defines number of amplitude steps and width steps between initial and end values. Finally the communication is terminated with a so called end of communication (EOC) command. With this executed protocol, all necessary parameters for performing noise sensitivity measurements on smart card devices are transferred to the printed circuit board. More detailed information about the byte internal structure and data arrangement is covered in table 4.6.

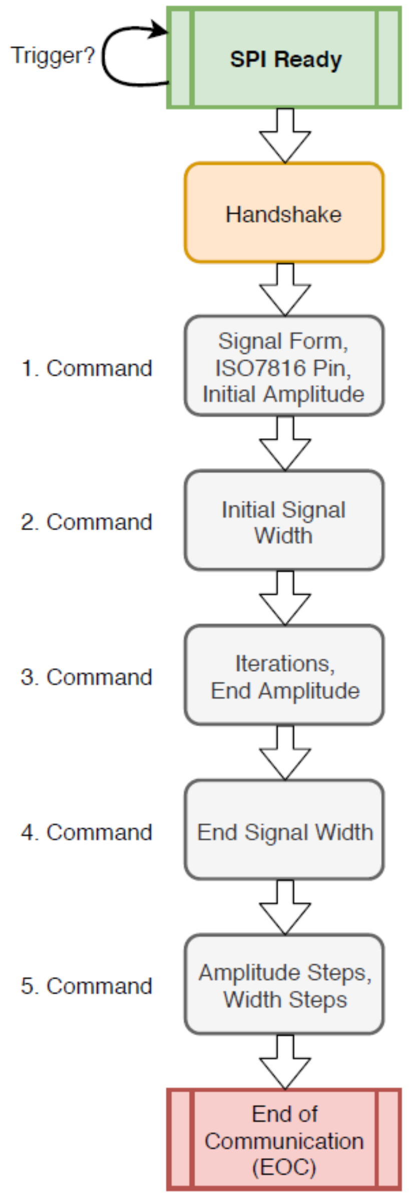

Trigger?
SPI Ready
Handshake
1. Command
Signal Form, ISO7816 Pin, Initial Amplitude
2. Command
Initial Signal Width
3. Command
Iterations, End Amplitude
4. Command
End Signal Width
5. Command
Amplitude Steps, Width Steps
End of Communication (EOC)

Figure 4.17: Flowchart of SPI communication.

4.5.2 Setup Noise Sensitivity Measurement

Data arrangement of the received bytes is defined in table 4.6. After splitting up all received bytes, the data is categorised into parameters. This happens with bit shift and masking operations. As soon as parameters are extracted from the data, the setup of the measurement is executed. This means, that the signal form is defined, as well as the pin, which should be stressed. Additionally initial and end voltage levels, as well as initial and end width are defined within the software. The number of iterations and steps between initial and end value are specified as well. With all parameters defined within microcontrollers software, the printed circuit board is ready to perform noise sensitivity measurements.

Command	Data received	Data send	Bit description
Handshake	0x70	0xBA	
1. Command	7 0 S P P P A A A A MSB LSB	Handshake byte 0x70	S ... signal type P ... ISO7816 pin A ... initial amplitude
2. Command	7 0 w w w w w w w w MSB LSB	1. Command byte	W ... initial width
3. Command	7 0 N N N N A A A A MSB LSB	2. Command byte	N ... iterations A ... end Amplitude
4. Command	7 0 w w w w w w w w MSB LSB	3. Command byte	W ... end width
5. Command	7 0 A A A A w w w w MSB LSB	4. Command byte	A ... amplitude steps W ... width steps
End of communication (EOC)	0x8F	5. Command byte	

Table 4.6: Data arrangement during communication via serial peripheral interface from PCB-side.

4.5.3 Signal Generating Code

After all necessary parameters are extracted and defined within the code the PCB is waiting for a trigger to start the measurement. As soon as the tester delivers the measurement trigger, the printed circuit board generates the first pulse or glitch. After that, an additional trigger will generate an additional pulse or glitch. Furthermore, voltage sweep and width sweep are implemented within this part of the software. The code running on the microcontroller can be found in appendix C.

Increasing Minimum Pulse Width

In order to allow generation of fast pulses shorter than $100ns$, special register commands are used. With the standard libraries I/O set/reset command, such fast pulses and glitches would not be possible. By the use of standard library commands, the minimum signal width is around $400ns$. With the use of register commands, the minimum width of the signal could be reduced to under 70ns. Using register commands instead of HAL Library GPIO Output obviously improves fast signal generation. [ST 17]

4.6 Design Verification

Finally, the external hardware design needs to be verified. Therefore communication with the tester needs to be reviewed. Additionally signal generation needs to be checked. Signal parameters like amplitude and width need to be specified by a measurement. Lastly, noise sensitivity measurement needs to be performed.

4.6.1 Communication with Tester

An example of a SPI protocol transmission is illustrated in figure 4.18. In yellow the trigger, which indicates the begin of SPI communication, is seen, This signal is generated by the CT1000 tester and is connected to microcontrollers I/O-port. In violet, the signal at the MOSI pin can be seen. This pin indicates communication flow from the tester to the noise generating printed circuit board. Lastly, the green trace illustrates the MISO pin. Here the answer from the microcontroller can be seen.

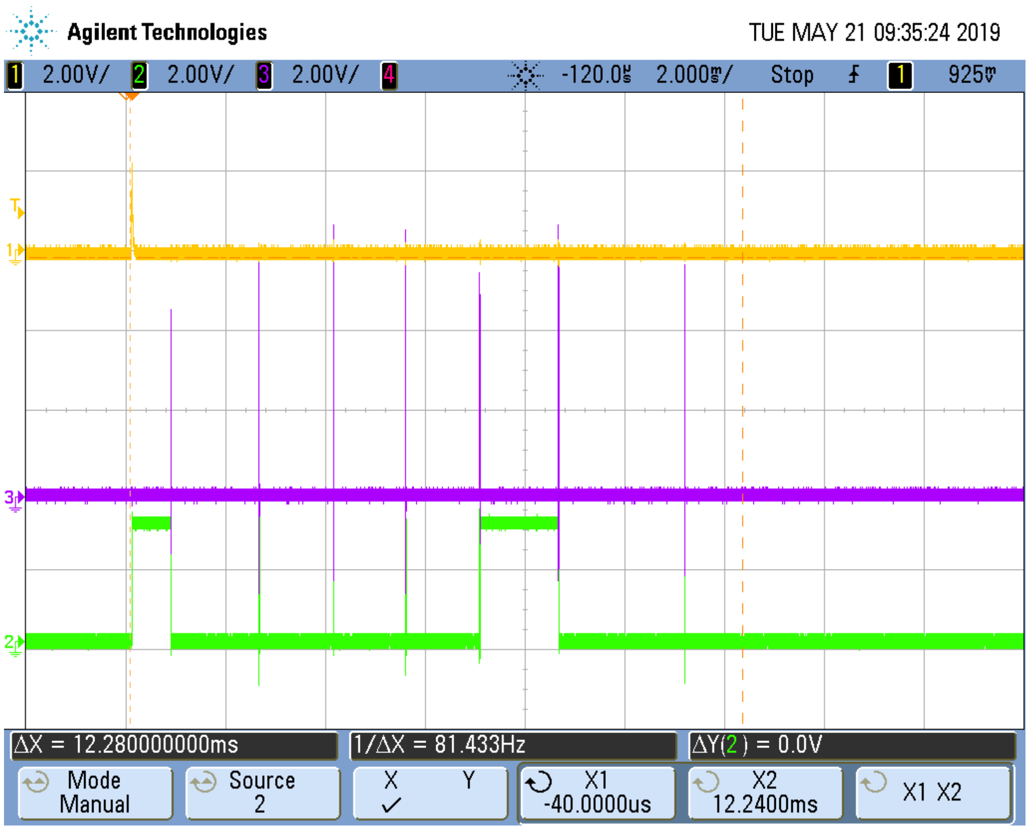


Figure 4.18: Example of SPI communication.

4.6.2 Pulse and Glitch Generation

Pulse and glitch signals are already specified within section 2.9. To verify the generated signals pulses and glitches were generated and measured. For that reason, pulse and glitch signals are investigated in respect to their switching capabilities.

Pulse Signal

On-time of the fastest possible pulse is approximately $65ns$, as indicated in oscilloscope capture illustrated in figure 4.19. In reference to the signal definition discussed in section 2.9, the minimum signal width could not entirely be fulfilled. For the reason that smart card devices tend to be more sensitive to signals with longer on-times, the increased minimum width is accepted. Further on, side effects of fast switching can be seen in figure 4.19. Over- and undershooting is the most dominant effect. The amplitude of these spikes is approximately $2V$. Further effects are considered negligible.

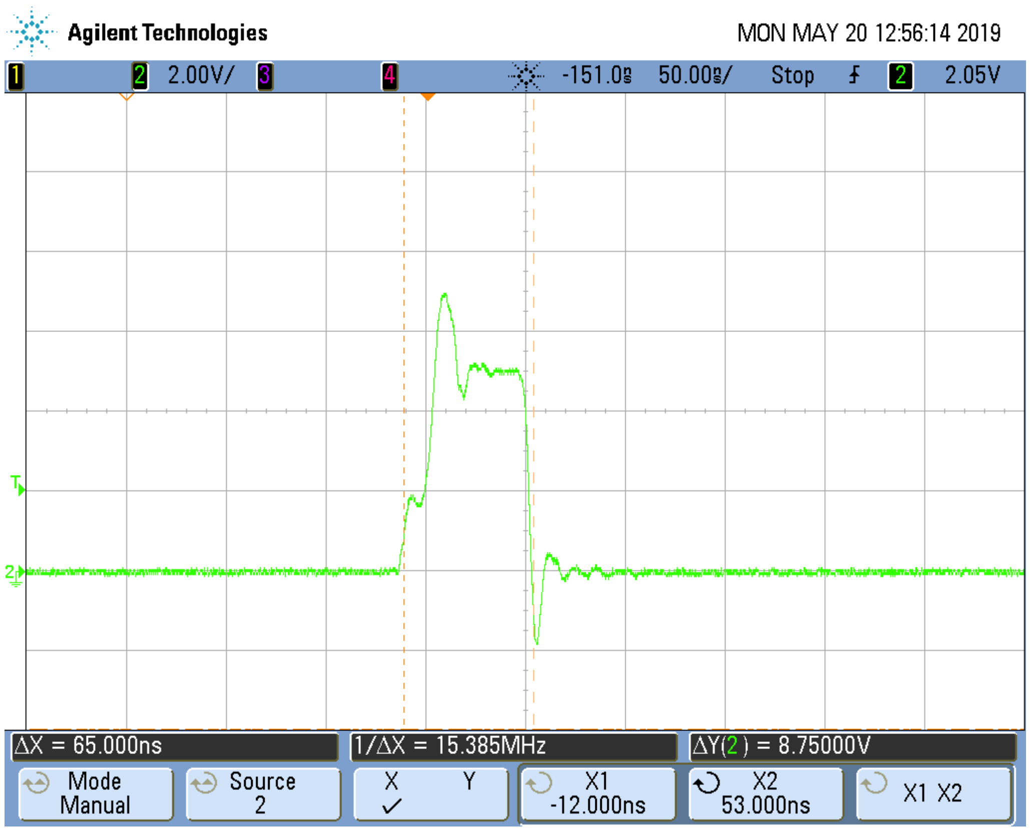
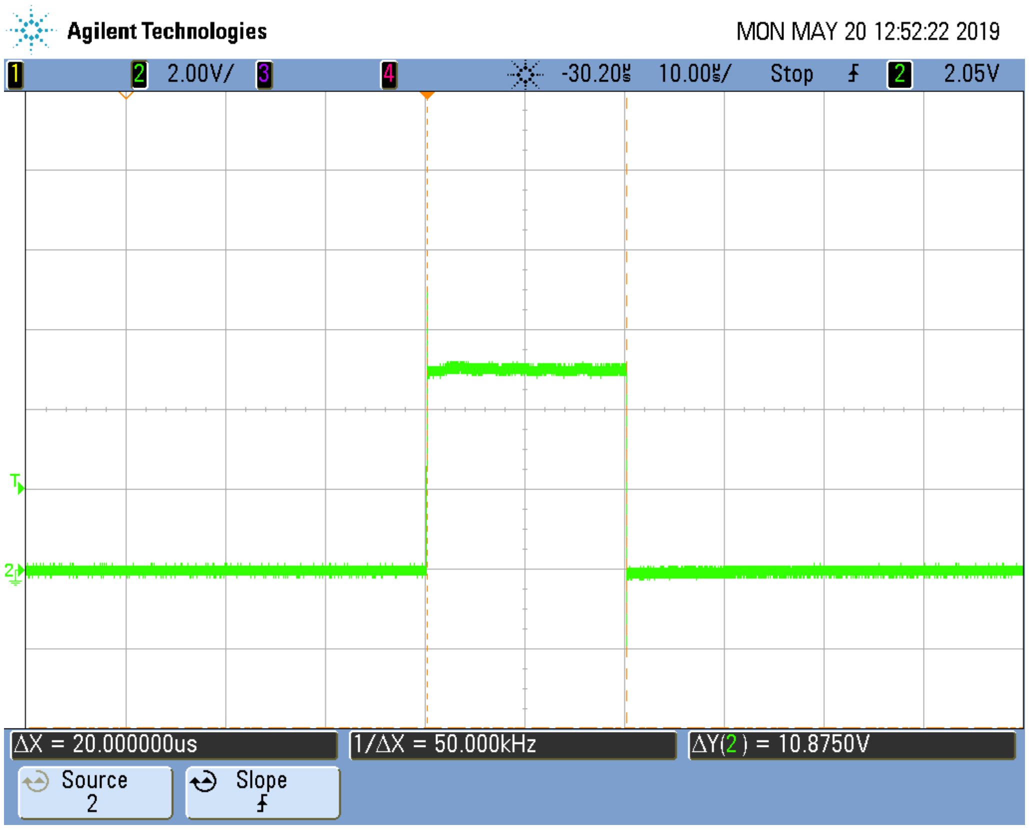


Figure 4.19: Fastest possible pulse signal.

Since the on-time of pulses is not restricted by hardware it can be further increased by software if needed. Side effects of fast switching are also observable when stressing the device with longer pulses. A slow pulse is illustrated in figure 4.20.

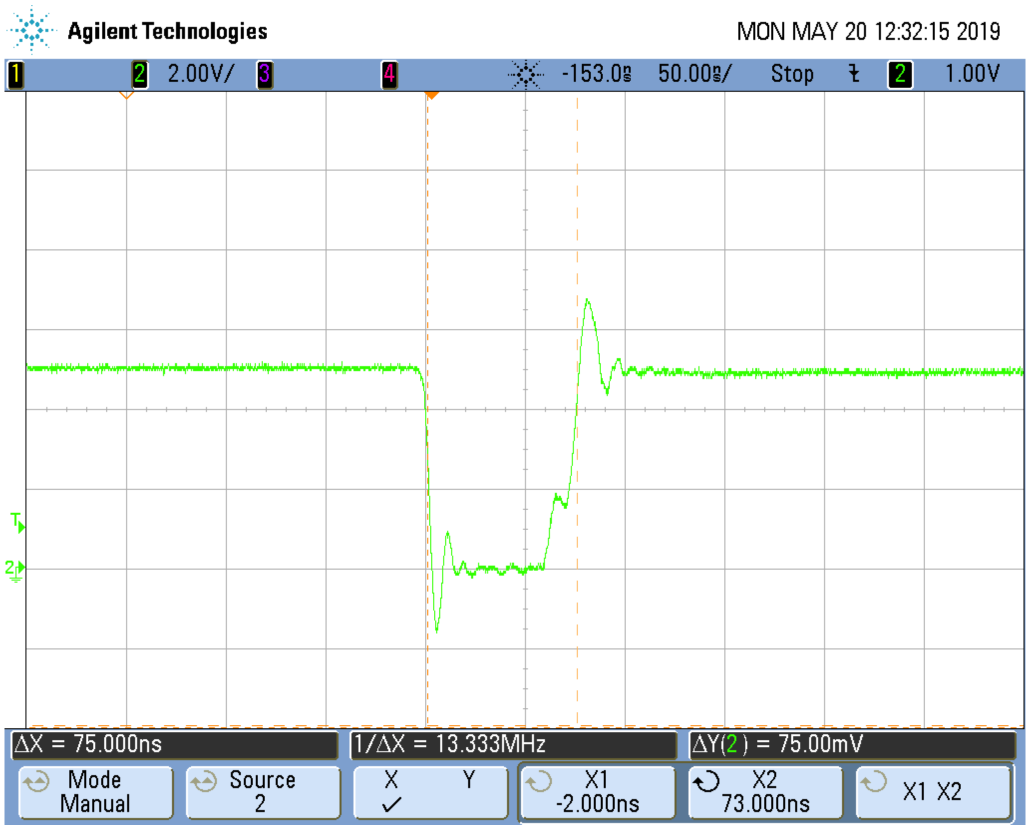


ΔX = 20.000000us
1/ΔX = 50.000kHz
ΔY(2) = 10.8750V

Figure 4.20: Slowest possible pulse signal.

Glitch Signal

Compared with a pulse, the fastest possible glitch is approximately 10ns longer. This can be seen in figure 4.21. This minimum glitch time is accepted for the same reasons as with pulses. Effects caused by fast switching are also present when generating glitches.



ΔX = 75.000ns
1/ΔX = 13.333MHz
ΔY(2) = 75.00mV
X1 -2.000ns
X2 73.000ns

Figure 4.21: Fastest possible glitch signal.

The maximum width of the glitch signal is just limited by software and can be easily adjusted. Additionally signal properties are the same as with the use of long pulse signals.

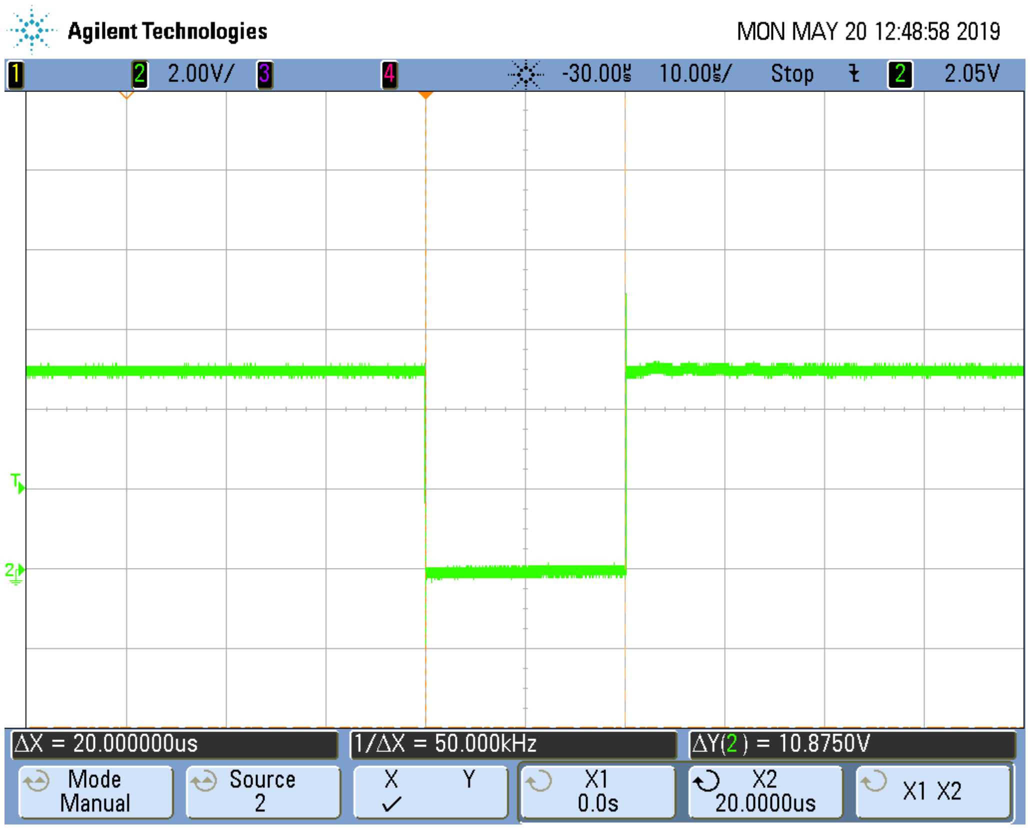


Figure 4.22: Slowest possible glitch signal.

4.6.3 Noise Sensitivity Measurement on Smart Card Device

Specification of the generated signal, as well as timing information can be seen in figure 4.23. A single stressing signal is generated after a trigger is received from the tester. With this approach it is possible to align timing of tester with the external noise generating hardware. This behaviour can be seen in figure 4.25 where a measurement with fixed amplitude is performed. The yellow trace indicates the trigger signal from the tester, while the green trace visualises a pulse signal with $5V$ amplitude. An example of a noise sensitivity measurement with voltage sweep and width sweep is illustrated in figure 4.24. The yellow trace shows a performed glitch signal, while the green trace visualises the adjustment of signals amplitude. Starting at the left, signal with initial amplitude and initial width is output. While maintaining constant amplitude, the width gets increased, until end width is reached. After this condition, width is reset to initial value, while signals amplitude increases to the next value. Finally, the rightmost signal is the signal with maximum amplitude and maximum width. At the end of each measurement, the capacitors get discharged to the initial value of the amplitude. This behaviour can be seen in figure 4.24.

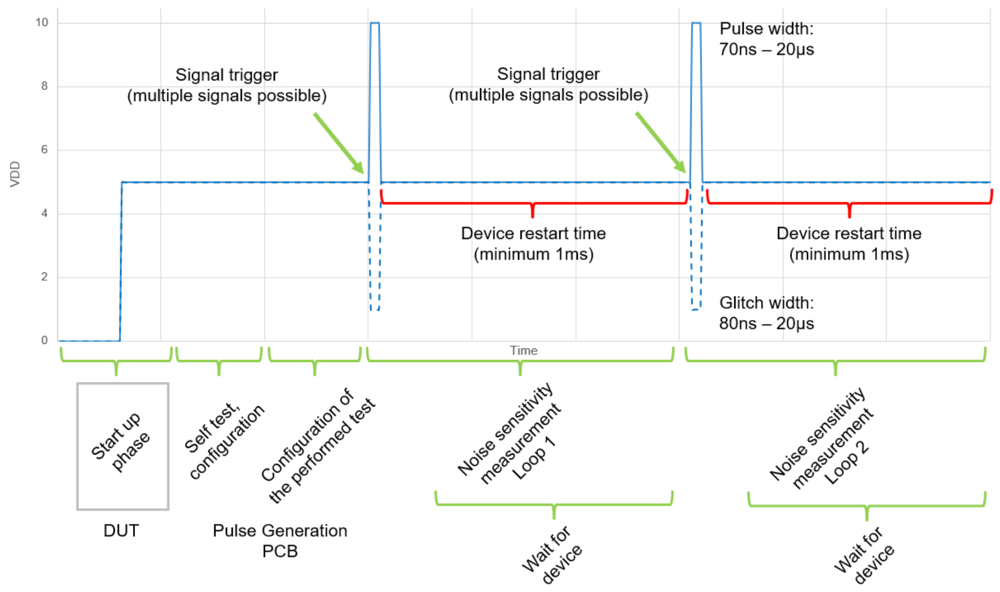


Figure 4.23: Signal description for pulse and glitch.

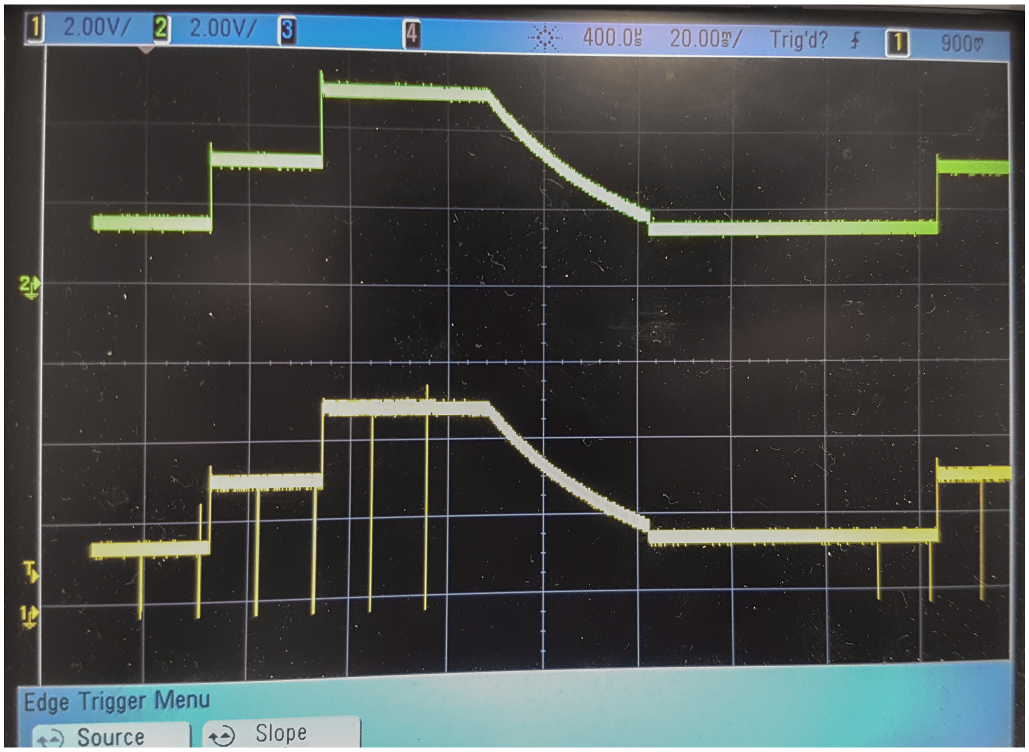


Figure 4.24: Noise sensitivity measurement with voltage sweep.

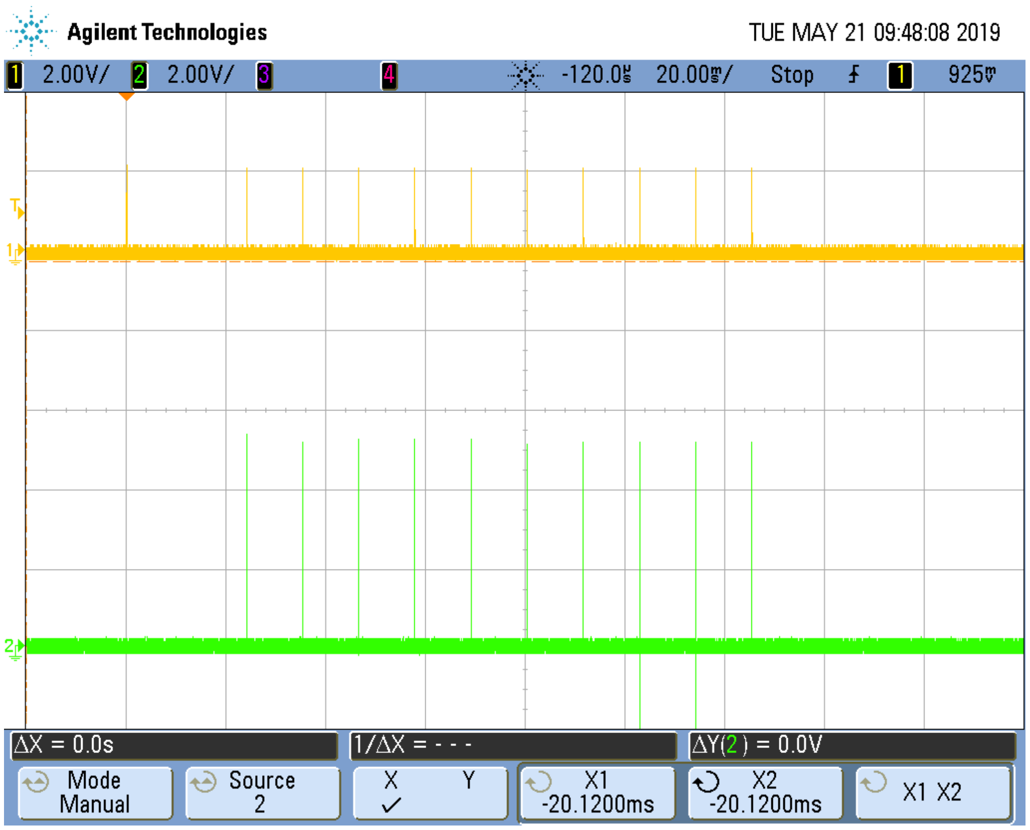


Figure 4.25: Noise sensitivity measurement with fixed amplitude.

Chapter 5

Conclusion and Outlook

Finalising this master's thesis a conclusion and outlook will be given. Therefore the latest status will be discussed, followed by further optimization and implementations to improve the overall setup for noise sensitivity measurement on smart card devices.

5.1 Latest Status

With the noise generating printed circuit board, which is described in this thesis, investigations regarding noise sensitivity of smart card devices can be accomplished. The overall setup is capable of measuring multiple pins on a single die. Fully automated sweeps of signal parameters like amplitude and length are possible. This features allow to investigate the behaviour of smart card devices, when under the strain of noise on ISO7816 pins. The design has also proven to provoke resets on smart card devices, so further investigation on robustness against noise on these pins are the consequence.

5.2 Multi-Site Implementation

To allow measurements on multiple devices in parallel, multi-site implementation is the next logical step. Therefore the switching matrix needs to be extended. With external hardware capable of generating noise signals to perform noise sensitivity measurement on multiple devices in parallel, investigation of robustness against stressing signals on ISO7816 pins would become simpler. Consequently test time would decrease, while comparability between devices on the same touch-down would increase. Furthermore the flexibility of noise generating printed circuit board would improve.

5.3 Improve Interface to Device Under Test

With increased hardware necessary to allow multi-site implementation, a solution needs to be found to avoid a decrease of flexibility of the overall system. For that reason it is very important to keep the printed circuit board as small as possible, while obtaining minimal wire length on signal traces. Furthermore, a suitable solution for the improvement of the interface to DUT needs to be found. With increasing count of components a special additional PCB would be convenient. This printed circuit board would implement the switching matrix in multi-site. This solution would also allow to design custom PCBs, which are perfectly fitting into each probe card to tester interface. Unless wire length on signal traces is kept at a minimum, this approach would allow easy and flexible integration into the existing production environment.

5.4 Increase Amplitude Range

Currently the amplitude of the pulse signal is defined to be $5V$ above DC level. For glitch signals the low-level is specified to be $1.5V$. An extended amplitude range would cause improved investigation methods. This means, that pulses greater than $5V$ should be possible. For glitch signal on the other hand this would mean, that negative glitches in respect to ground should be possible. To achieve these requirements, more effort needs to be spend on hardware design.

5.5 Multi-Signal Generation

In an upcoming revision of the noise generating printed circuit board more focus can be spent on multiple possible signal forms. In addition to the current setup, other different signal shapes would increase the possibilities to investigate the robustness against noise on ISO7816 pins. Examples for additional wave forms would be rectangular, slopes, triangle or sine waves. Furthermore a decrease of minimum signal length can be discussed. An possible implementation of multiple signal forms would be an integration of an FPGA. This additional IC could be used to generate multiple possible wave forms with increased speed.

5.6 Support other Products

Noise sensitivity is not just a topic at smart card devices. ICs with other applications may have comparable properties. Therefore a system supporting multi-platform products would be convenient. Additionally, other product portfolios are very interesting to perform investigations on noise sensitivity. With the increasing demand of IoT products in the market, the next reasonable step is to also perform noise sensitivity measurements on IoT devices. With such a setup it is possible to investigate a bigger variety of product portfolios to their robustness against environmental influences.

Glossary

V_{DD} Positive supply voltage.. 6, 10, 16, 18–21, 40, 50

ADS Analog Driver/Sensor. 32

AES Advanced Encryption Standard. 3

APH Analog Phases on SPEA Tester. 33

APM Strobe on SPEA Tester. 33

ATE Automatic Test Equipment. 6

ATEST Period when Test is Active. 33

AtosCT SPEA Tester Software Framework.. 7, 18

ATR Answer to Reset. 4, 14

CMOS Complementary Metal-Oxide-Semiconductor. 10

D-sub Common type of electrical connector with characteristic D-shaped metal shield.. 54

DC Direct Current. 24, 25, 27, 28, 40, 41, 47, 60

DDS Digital Driver/Sensor. 19, 33

DES Data Encryption Standard. 3

DPA Differential Power Analysis. 10

DUT Device Under Test. 14, 19, 21, 24, 25, 37, 40–42, 44

ESL Equivalent Serial Inductance. 47

ESR Equivalent Serial Resistance. 47

FET Field Effect Transistor. 26, 28, 41

FIB Focused Ion Beam. 9

FPGA Field Programmable Gate Array. 37–39, 60

GND Ground. 10, 25, 28, 30, 40

HF High-Frequency. 7, 24, 25, 46, 47

I/O Input-Output. 9, 26

I^{2}C Inter Integrated Circuit. 5

IC Integrated Circuit. 25, 26, 30, 41, 53, 54, 60

IoT Internet of Things. 60

ISO International Organization for Standardization. 4

ISO14443 ISO Standard defining Contactless Communication. 4

ISO7816 ISO Standard defining Smart Cards. 4, 5, 7, 12, 14, 19, 20, 50, 59, 60

LDO Low-Dropout. 53

LF Low-Frequency. 7

MCU Microcontroller Unit. 39, 41, 43, 44, 53, 54

MLCC Multilayer Ceramic Chip Capacitor. 47

MOS Metal-Oxide-Semiconductor. 28, 41, 42

multi-site Number of devices tested in parallel.. ii, iii, 7, 59, 60

NMOS n-Channel Metal-Oxide-Semiconductor. 25, 40, 41, 54

PAA Power Analysis Attack. 9, 11, 13

PC Personal Computer. 7, 51

PCB Printed Circuit Board. 19, 30, 33, 35, 38–40, 42–44, 46, 48, 52–55, 60

RF Radio-Frequency. 4

RSA Rivest, Shamir und Adleman. 3

SCA Side-Channel Attack. 8, 9, 11

SNR Signal-To-Noise Ratio. 13

SPA Simple Power Analysis. 10

SPI Serial Peripheral Interface. 31, 32, 41, 44, 48, 53, 54

TPE Test Plan Editor. 7

UHF Ultra-High-Frequency. 7

USB Universal Serial Bus. 5, 53, 54

UV Ultra-Violet. 4

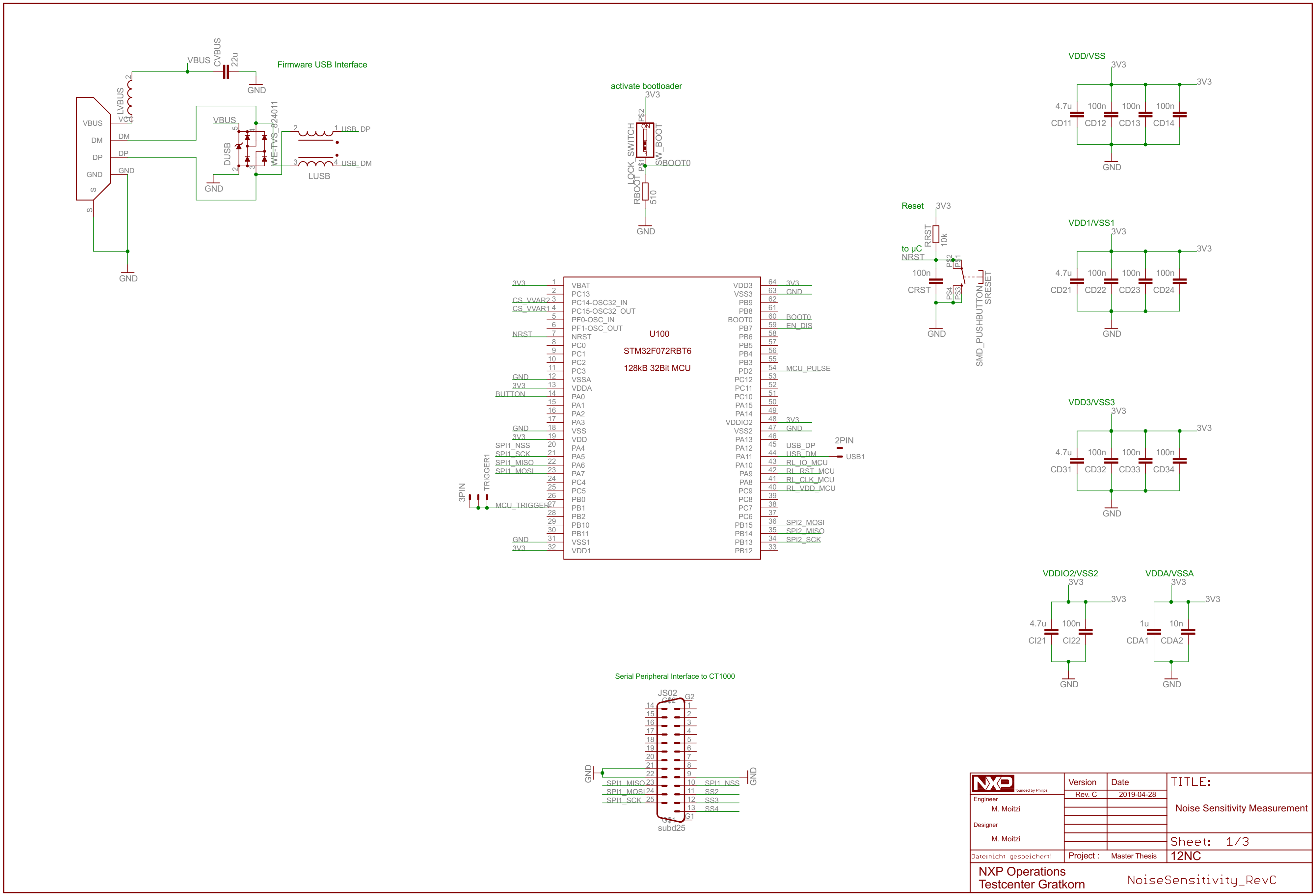
VBA Visual Basic for Applications. 7

VDA Visual Data Analyzer. 7, 51

VectorVIEW SPEA Tool allowing Pattern control.. 18

Appendix A

EAGLE Schematic

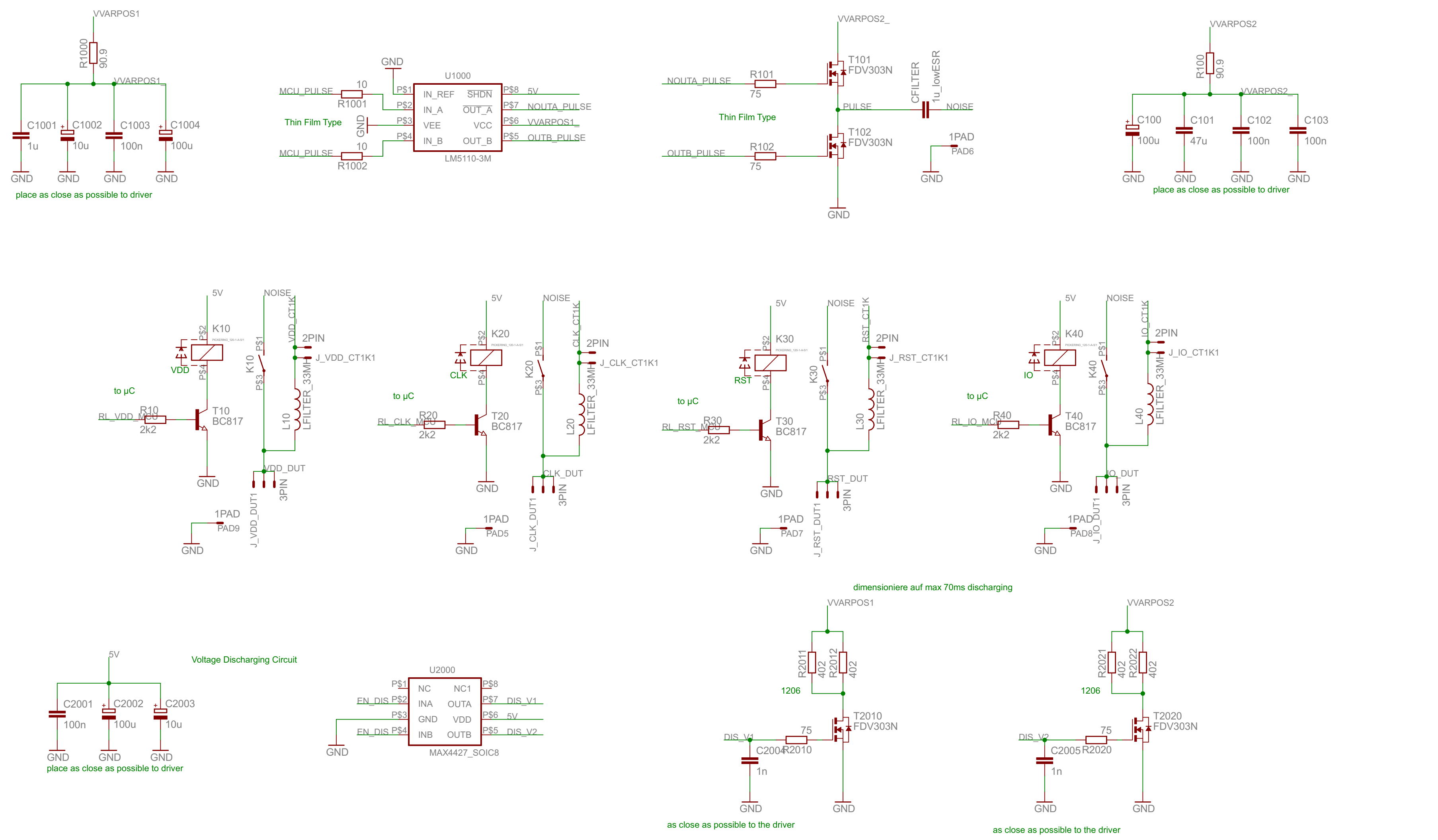


NXP founded by Philips

Engineer	Version	Date	TITLE:
M. Moitzi	Rev. C	2019-04-28	Noise Sensitivity Measurement
Designer			Sheet: 1/3
M. Moitzi			
Date:nicht gespeichert!	Project :	Master Thesis	12NC

NXP Operations
Testcenter Gratkorn

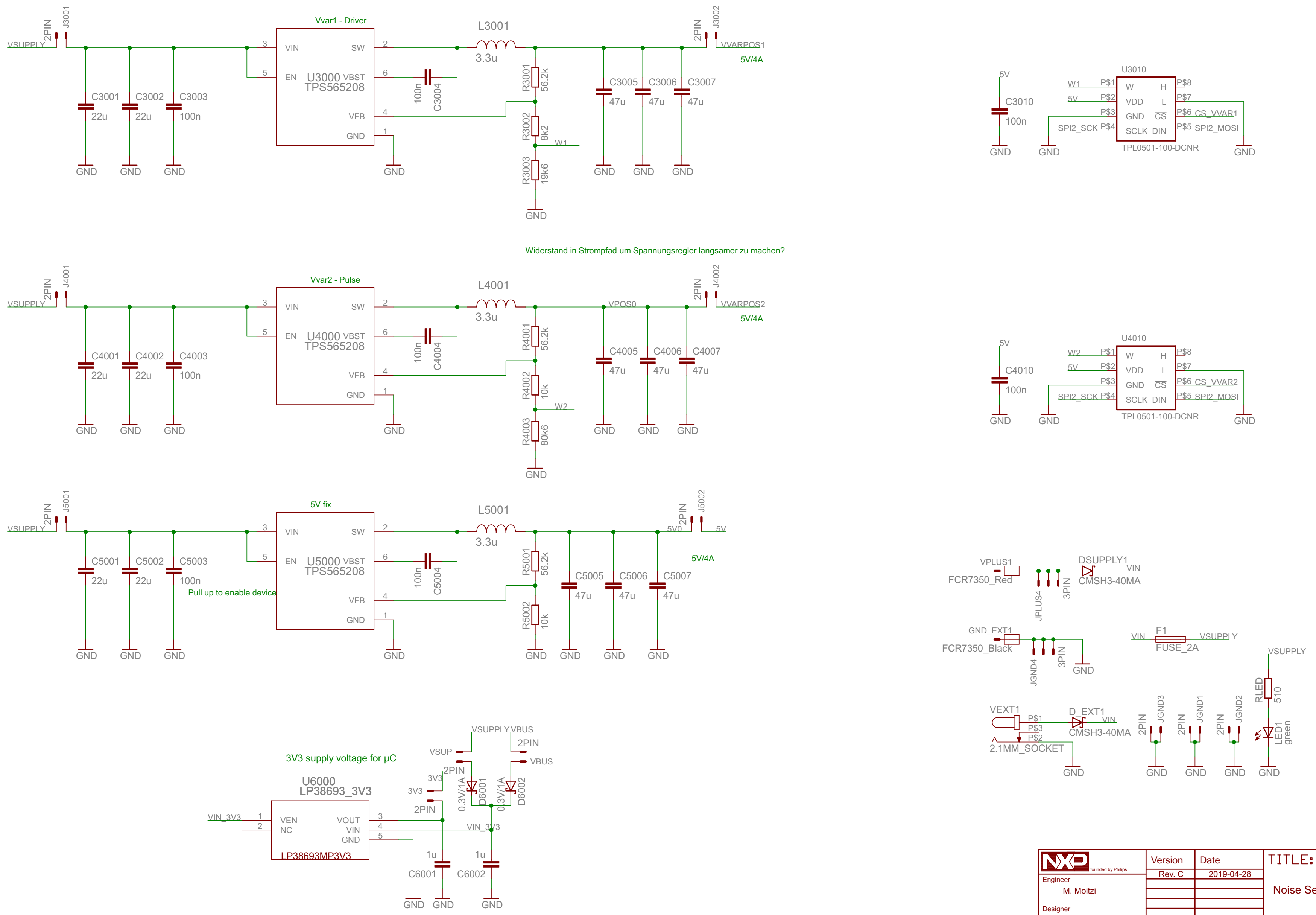
NoiseSensitivity_RevC



	Version	Date	TITLE:
NXP founded by Philips	Rev. C	2019-04-28	Noise Sensitivity Measurement
Engineer M. Moitzi			
Designer M. Moitzi			Sheet: 2/3
Date:nicht gespeichert!	Project :	Master Thesis	12NC

NXP Operations
Testcenter Gratkorn

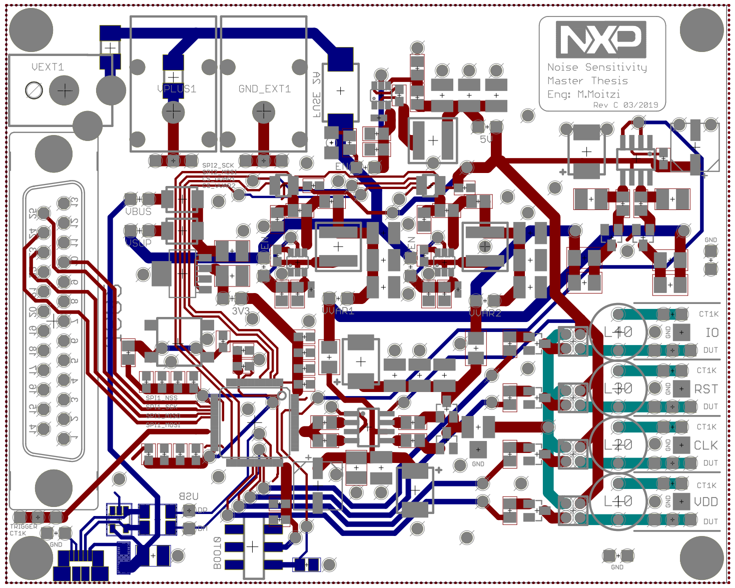
NoiseSensitivity_RevC



NXP	Version	Date	TITLE:
Engineer M. Moitzi	Rev. C	2019-04-28	Noise Sensitivity Measurement
Designer M. Moitzi			Sheet: 3/3
Date:nicht gespeichert!	Project :	Master Thesis	12NC
NXP Operations Testcenter Gratkorn			NoiseSensitivity_RevC

Appendix B

EAGLE Layout


NXP
Noise Sensitivity
Master Thesis
Eng: M.Moitzi
Rev C 03/2019
VEXT1
VPLUS1
GND_EXT1
5V
VBUS
3V3
VVAR1
VVAR2
SPI2_SCK
SPI1_NSS
CT1K
IO
RST
CLK
VDD
OUT
GND
L40
L30
L20
L10
TRIGGER
USB

Appendix C

Microcontroller Software

```
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  ** This notice applies to any and all portions of this file
  * that are not between comment pairs USER CODE BEGIN and
  * USER CODE END. Other portions of this file, whether
  * inserted by the user or by software development tools
  * are owned by their respective copyright owners.
  *
  * COPYRIGHT(c) 2019 STMicroelectronics
  *
  * This microcontroller software for the STM32F072RBT6 was generated using
  * STMicroelectronics software CubeMX. With the help of this tool it is quite
  * easy to set up all necessary interfaces and pins.
  *
  ******************************************************************************
  */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "stm32f0xx_hal.h"

/* USER CODE BEGIN Includes */
#include "time.h"
/* USER CODE END Includes */

/* Private variables ---------------------------------------------------------*/
I2C_HandleTypeDef hi2c2;

SPI_HandleTypeDef hspi1;
SPI_HandleTypeDef hspi2;

TSC_HandleTypeDef htsc;

PCD_HandleTypeDef hpcd_USB_FS;

/* USER CODE BEGIN PV */
/* Private variables ---------------------------------------------------------*/

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_I2C2_Init(void);
static void MX_TSC_Init(void);
static void MX_USB_PCD_Init(void);
static void MX_SPI1_Init(void);
static void MX_SPI2_Init(void);

/* USER CODE BEGIN PFP */
/* Private function prototypes -----------------------------------------------*/

/* USER CODE END PFP */

/* USER CODE BEGIN 0 */
uint8_t spiRxBuf = 0x00;
uint8_t iRecData = 0x00;
uint8_t spiTxBuf = 0x00;
uint8_t oldValue = 0x00;
uint8_t newValue = 0x00;

//bit flags
unsigned char cPulseMask = 128;     //0b10000000
unsigned char cPinMask = 112;       //0b01110000
unsigned char cAmplMask = 15;       //0b00001111
unsigned char cLoopMask = 240;      //0b11110000

//ISO7816 Pin compare constants
#define PIN_RST             16
#define PIN_CLK             32
#define PIN_IO              48
#define PIN_GND             64
#define PIN_VDD             80

//SPI command bytes
uint8_t EOC = 143;          //0x8F
uint8_t HSANSBYTE = 186;    //0xBA
uint8_t HANDSHAKE = 112;    //0x70
#define SPITIMEOUT          5000    //in milli seconds
#define VOLTAGE_SET_TIME    1       //in milli seconds
#define DUMMYBYTE           255     //0xFF

//definitions
#define BUFFERSIZE          16      //size of receiving buffer
#define MAXWIDTHCNTPULSE    180     //max value of counter to 20us pulse (160)
#define MAXWIDTHCNTGLITCH   138     //max value of counter to 20us pulse (138)
#define MAXBYTEVALUE        255     //MAXBYTEVALUE value transferable by SPI
#define OFFDELAY            1       //delay after pulse/glitch in ms (HAL_DELAY)
#define RESETDELAY          10000   //delay before reset (HAL_DELAY)
#define MINVOLTAGE          1000    //minimum voltage in mV
#define MAXVOLTAGE          5000    //maximum voltage in mV
```

```
/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  *
  * @retval None
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration----------------------------------------------------------*/

  /* Reset of all peripherals,Initializes the Flash interface and the Systick.*/
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_I2C2_Init();
  MX_TSC_Init();
  MX_USB_PCD_Init();
  MX_SPI1_Init();
  MX_SPI2_Init();

  /* USER CODE BEGIN 2 */
  //set SPI CS pins to high
  HAL_GPIO_WritePin (GPIOC,GPIO_PIN_14,GPIO_PIN_SET);
  HAL_GPIO_WritePin (GPIOC,GPIO_PIN_15,GPIO_PIN_SET);
  //set discharging relais to off
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_12, GPIO_PIN_RESET);
  /* USER CODE END 2 */

  /* Infinite loop */
  while (1)
  {

  /* USER CODE BEGIN 3 */
    int iBitsCompare = 0;
    int iTransLoop = 0;
    int iComStat = 0;
    int iHandshake = 0;
    double dWidthStart = 0;
    double dWidthStop = 0;
```

```
int iWidthCnt = 0;
int iWidthStep = 0;
int iWidthStepCnt = 0;
int iWidth = 0;
int iMeasLoops = 0;
int iAmplStart = 0;
int iAmplStop = 0;
int iAmplStep = 0;
int iAmplStepCnt = 0;
int iAmplCnt = 0;
int iAmplCntSweepEnd = 0;
int iLoopCnt = 0;
int iPGFlag = 0;
int iCnt = 0;
uint8_t iRecBuffer[BUFFERSIZE] = {0x00};
uint8_t iDriverControl[16] = {28,24,20,17,14,12,10,8,7,6,5,4,3,2,1,0};
uint8_t iDriverControlSend[16] = {0};
uint8_t iVoltageControl[16] = {250,136,90,63,47,36,28,22,17,14,10,8,5,3,2,0};
uint8_t iVoltageControlSend[16] = {0};

//first initialization
spiTxBuf = HSANSBYTE; //0xBA
spiRxBuf = DUMMYBYTE; //0xFF

//delete existing data in iRecBuffer array
for (iCnt = 0; iCnt < BUFFERSIZE ; iCnt++)
  iRecBuffer[iCnt] = 0x00;

//wait for trigger - SPI communication will start immediately
while(!HAL_GPIO_ReadPin (GPIOC, GPIO_PIN_13));

do
{
  switch(HAL_SPI_TransmitReceive(&hspi1, &spiTxBuf, &spiRxBuf,
                                 sizeof(spiTxBuf) / sizeof(*(&spiTxBuf)),
                                 SPITIMEOUT))
  {
    case HAL_OK:
      if (iHandshake == 0)
      {
        if (spiRxBuf != (HANDSHAKE & 0xFF))
        {
          spiTxBuf = HSANSBYTE;
          iHandshake = 0;
        }
        else
        {
          spiTxBuf = spiRxBuf;
          iRecBuffer[iTransLoop] = spiRxBuf;
          iTransLoop++;
          iHandshake++;
        }
      }
      else
      {
        spiTxBuf = spiRxBuf;
```

```
          iRecBuffer[iTransLoop] = spiRxBuf;
          iTransLoop++;
          iHandshake++;
        }
        break;

      case HAL_TIMEOUT:
        spiTxBuf = HSANSBYTE;
        iHandshake = 0;
        iTransLoop = 0;
        break;

      case HAL_ERROR:
        spiTxBuf = HSANSBYTE;
        iHandshake = 0;
        iTransLoop = 0;
        break;

      case HAL_BUSY:
        break;
    }
  } while(spiRxBuf != (EOC & 0xFF));

  iTransLoop = 0;

  //Error checker
  //-> check for handshake byte on index 0 of receive buffer
  if (iRecBuffer[iTransLoop] == 0x70)
  {
    iTransLoop++;
    do
    {
      switch(iTransLoop)
      {
        //first command - pulse/glitch, pins and initial voltage level
        case 1:
          //Pulse or Glitch
          if (iRecBuffer[iTransLoop] & cPulseMask)
          {
            //Set Relay to Pulse
            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_12, GPIO_PIN_SET);
            //set initial voltage level of half-bridge to low-level
            HAL_GPIO_WritePin(GPIOD, GPIO_PIN_2, GPIO_PIN_SET);
            iPGFlag = 1;
          }
          else
          {
            //Set Relay to Glitch
            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_12, GPIO_PIN_RESET);
            //set initial voltage level of half-bridge to high-level
            HAL_GPIO_WritePin(GPIOD, GPIO_PIN_2, GPIO_PIN_RESET);
            iPGFlag = 0;
          }

          //Get ISO 7816 Pin to measure
          iBitsCompare = iRecBuffer[iTransLoop] & cPinMask;
    switch(iBitsCompare)
    {
      case PIN_VDD:
        //Set VDD relay
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_SET);    //VDD
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, GPIO_PIN_RESET);  //RST
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, GPIO_PIN_RESET); //CLK
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_15, GPIO_PIN_RESET); //IO
        break;
      case PIN_RST:
        //Set RST relay
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_RESET);  //VDD
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, GPIO_PIN_SET);    //RST
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, GPIO_PIN_RESET); //CLK
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_15, GPIO_PIN_RESET); //IO
        break;
      case PIN_CLK:
        //Set CLK relay
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_RESET);  //VDD
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, GPIO_PIN_RESET);  //RST
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, GPIO_PIN_SET);   //CLK
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_15, GPIO_PIN_RESET); //IO
        break;
      case PIN_IO:
        //Set IO relay
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_RESET);  //VDD
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, GPIO_PIN_RESET);  //RST
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, GPIO_PIN_RESET); //CLK
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_15, GPIO_PIN_SET);   //IO
        break;
      case PIN_GND:
        //Set GND relay
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_RESET);  //VDD
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, GPIO_PIN_RESET);  //RST
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, GPIO_PIN_RESET); //CLK
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_15, GPIO_PIN_RESET); //IO
        break;
    }

    //Get initial voltage level (1.5V to 5V amplitude)
    iBitsCompare = iRecBuffer[iTransLoop] & cAmplMask;
    iAmplStart = iBitsCompare;
    break;

  //second command - initial signal width
  case 2:
    if (iPGFlag > 0)
    {
      dWidthStart = (double)MAXWIDTHCNTPULSE / (double)MAXBYTEVALUE
        * (double)iRecBuffer[iTransLoop];
      dWidthStart++; //otherwise 0x00 would be 0
    }
    else
    {
      dWidthStart = (double)MAXWIDTHCNTGLITCH / (double)MAXBYTEVALUE
        * (double)iRecBuffer[iTransLoop];
```

```
          dWidthStart++; //otherwise 0x00 would be 0
        }
        break;

      //third command - iterations, end amplitude
      case 3:
        //cLoopMask = 240;      //0b11110000
        iBitsCompare = iRecBuffer[iTransLoop] & cLoopMask;
        iMeasLoops = iBitsCompare >> 4; //get number of loops
        iBitsCompare = iRecBuffer[iTransLoop] & cAmplMask;
        iAmplStop = iBitsCompare;
        break;

      //fourth command - end signal width
      case 4:
        if (iPGFlag)
          dWidthStop = (double)MAXWIDTHCNTPULSE / (double)MAXBYTEVALUE
          * (double)iRecBuffer[iTransLoop];
        else
          dWidthStop = (double)MAXWIDTHCNTGLITCH / (double)MAXBYTEVALUE
          * (double)iRecBuffer[iTransLoop];
        break;

      //fith command - amplitude steps, width steps
      case 5:
        //cLoopMask = 240;      //0b11110000
        iBitsCompare = iRecBuffer[iTransLoop] & cLoopMask;
        iAmplStepCnt = iBitsCompare >> 4; //get number of loops
        if (iAmplStepCnt == 0)
          iAmplStepCnt = 1;
        iAmplStep = (iAmplStop - iAmplStart) / iAmplStepCnt;

        //cAmplMask = 15;       //0b00001111
        iBitsCompare = iRecBuffer[iTransLoop] & cAmplMask;
        iWidthStepCnt = iBitsCompare; //get number of loops
        iWidthStep = (dWidthStop - dWidthStart) / iWidthStepCnt;
        break;
    }
    iTransLoop++;
  } while (iRecBuffer[iTransLoop] != EOC);
}
else
{
  //handshake byte not on index 0 -> ERROR
  HAL_GPIO_WritePin (GPIOC,GPIO_PIN_6,GPIO_PIN_SET);
  HAL_GPIO_WritePin (GPIOC,GPIO_PIN_7,GPIO_PIN_SET);
  HAL_GPIO_WritePin (GPIOC,GPIO_PIN_8,GPIO_PIN_SET);
  HAL_GPIO_WritePin (GPIOC,GPIO_PIN_9,GPIO_PIN_SET);
  while(1); //wait for reset
}

//poll for trigger
//-> pulse generating + sweeping

//steps explained:
//iAmplStepCnt = 0    just init value
```

```
//iAmplStepCnt = 1    init value and end value
//iAmplStepCnt = 2    init value, end value and 1 additional step in between
// ...

//define voltage levels of half-bridge and driver
if(iAmplStepCnt == 0)
{
  iVoltageControlSend[0] = iVoltageControl[iAmplStart];
  iDriverControlSend[0] = iDriverControl[iAmplStart];
  iAmplCntSweepEnd = 0;
}
else if(iAmplStepCnt == 1)
{
  iVoltageControlSend[0] = iVoltageControl[iAmplStart];
  iDriverControlSend[0] = iDriverControl[iAmplStart];
  iVoltageControlSend[1] = iVoltageControl[iAmplStop];
  iDriverControlSend[1] = iDriverControl[iAmplStop];
  iAmplCntSweepEnd = 1;
}
else if(iAmplStepCnt >= 2)
{
  iVoltageControlSend[0] = iVoltageControl[iAmplStart];
  iDriverControlSend[0] = iDriverControl[iAmplStart];
  iVoltageControlSend[iAmplStepCnt] = iVoltageControl[iAmplStop];
  iDriverControlSend[iAmplStepCnt] = iDriverControl[iAmplStop];
  iAmplCntSweepEnd = 1;

  for(iCnt = 1 ; iCnt < iAmplStepCnt ; iCnt++)
  {
    iVoltageControlSend[iCnt] = iVoltageControl[iAmplStart + (iAmplStep * iCnt)];
    iDriverControlSend[iCnt] = iDriverControl[iAmplStart + (iAmplStep * iCnt)];
    iAmplCntSweepEnd++;
  }
}

for(iAmplCnt = 0 ; iAmplCnt <= iAmplCntSweepEnd ; iAmplCnt++)
{
  //Amplitude sweep
  //Send SPI data to digital resistor to set voltage level

  //Driver control - Vvarpos1 CS = PC14
  GPIOC->BRR=(1<<14);
  HAL_SPI_Transmit(&hspi2, &iDriverControlSend[iAmplCnt], 1, HAL_MAX_DELAY);
  GPIOC->BSRR=(1<<14);

  //Voltage control - Vvarpos2 CS = PC15
  GPIOC->BRR=(1<<15);
  HAL_SPI_Transmit(&hspi2, &iVoltageControlSend[iAmplCnt], 1, HAL_MAX_DELAY);
  GPIOC->BSRR=(1<<15);

  HAL_Delay(VOLTAGE_SET_TIME);

  for(iWidthCnt = (int)dWidthStart ; iWidthCnt < (int)dWidthStop ;
      iWidthCnt += iWidthStep)
  {
    //loops per pulse/glitch width
```

```
        for(iLoopCnt = 0 ; iLoopCnt < iMeasLoops ; iLoopCnt++)
        {
          //wait for trigger
          while(!HAL_GPIO_ReadPin (GPIOC, GPIO_PIN_13));

          if (iPGFlag)
          {
            //Pulse
            if(iWidthCnt == 1)
            {
              //fastest pulse
              GPIOD->BRR=(1<<2);
              __NOP();
              GPIOD->BSRR=(1<<2);
            }
            else
            {
              GPIOD->BRR=(1<<2);
              for(iWidth = 0; iWidth < (iWidthCnt-11); iWidth++)
              {
                __NOP();
              }
              GPIOD->BSRR=(1<<2);
            }
          }
          else
          {
            //Glitch
            if(iWidthCnt == 1)
            {
              //fastest glitch
              GPIOD->BSRR=(1<<2);
              __NOP();
              GPIOD->BRR=(1<<2);
            }
            else
            {
              GPIOD->BSRR=(1<<2);
              for(iWidth = 0; iWidth < iWidthCnt; iWidth++)
              {
                __NOP();
              }
              GPIOD->BRR=(1<<2);
            }
          }
          HAL_Delay(OFFDELAY);
        }
      }
    }

    //set voltages to initial state
    //Driver control
    //Vvarpos1 CS = PC14
    GPIOC->BRR=(1<<14);
    HAL_SPI_Transmit(&hspi2, &iDriverControlSend[0], 1, HAL_MAX_DELAY);
    GPIOC->BSRR=(1<<14);
    //voltage control
    //Vvarpos2 CS = PC15
    GPIOC->BRR=(1<<15);
    HAL_SPI_Transmit(&hspi2, &iVoltageControlSend[0], 1, HAL_MAX_DELAY);
    GPIOC->BSRR=(1<<15);

    //discharge capacitors for next measurement
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_12, GPIO_PIN_RESET);
    HAL_Delay(100);
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_12, GPIO_PIN_SET);

    //_______PINNING___________________________________________________________
    //
    //MCU_PULSE ... PD2
    //MCU_TRIGGER ... PB1
    //
    //RL_IO_MCU ... PA10
    //RL_RST_MCU ... PA9
    //RL_CLK_MCU ... PA8
    //RL_VDD_MCU ... PC9
    //
    //LED5 ... PC9
    //LED4 ... PC8
    //LED3 ... PC6
    //LED6 ... PC7
    //
    //BUTTON1 ... PA0
    //
    //SPI1_MOSI ... PA7
    //SPI1_MISO ... PA6
    //SPI1_SCK ... PA5
    //SPI1_NSS ... PA4
    //
    //SPI2_MOSI ... PB15
    //SPI2_MISO ... PB14
    //SPI2_SCK ... PB13
    //CS_VVAR1 ... PA15
    //CS_VVAR2 ... PA14
    //
    //EN_DIS ... PB7
    //
    //USB_DP ... PA12
    //USB_DM ... PA11
    //________________________________________________________________________
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct;
  RCC_ClkInitTypeDef RCC_ClkInitStruct;
  RCC_PeriphCLKInitTypeDef PeriphClkInit;

    /**Initializes the CPU, AHB and APB busses clocks
    */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI|RCC_OSCILLATORTYPE_HSI48;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSI48State = RCC_HSI48_ON;
  RCC_OscInitStruct.HSICalibrationValue = 16;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL6;
  RCC_OscInitStruct.PLL.PREDIV = RCC_PREDIV_DIV1;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

    /**Initializes the CPU, AHB and APB busses clocks
    */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_1) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

  PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_USB;
  PeriphClkInit.UsbClockSelection = RCC_USBCLKSOURCE_HSI48;

  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

    /**Configure the Systick interrupt time
    */
  HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq()/1000);

    /**Configure the Systick
    */
  HAL_SYSTICK_CLKSourceConfig(SYSTICK_CLKSOURCE_HCLK);

  /* SysTick_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(SysTick_IRQn, 0, 0);
}

/* I2C2 init function */
static void MX_I2C2_Init(void)
{
  hi2c2.Instance = I2C2;
  hi2c2.Init.Timing = 0x20303E5D;
  hi2c2.Init.OwnAddress1 = 0;
  hi2c2.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
  hi2c2.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
  hi2c2.Init.OwnAddress2 = 0;
  hi2c2.Init.OwnAddress2Masks = I2C_OA2_NOMASK;
  hi2c2.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
  hi2c2.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
  if (HAL_I2C_Init(&hi2c2) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

    /**Configure Analogue filter
    */
  if (HAL_I2CEx_ConfigAnalogFilter(&hi2c2, I2C_ANALOGFILTER_ENABLE) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

    /**Configure Digital filter
    */
  if (HAL_I2CEx_ConfigDigitalFilter(&hi2c2, 0) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }
}

/* SPI1 init function */
static void MX_SPI1_Init(void)
{
  /* SPI1 parameter configuration*/
  hspi1.Instance = SPI1;
  hspi1.Init.Mode = SPI_MODE_SLAVE;
  hspi1.Init.Direction = SPI_DIRECTION_2LINES;
  hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
  hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
  hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
  hspi1.Init.NSS = SPI_NSS_HARD_INPUT;
  hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
  hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
  hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
  hspi1.Init.CRCPolynomial = 7;
  hspi1.Init.CRCLength = SPI_CRC_LENGTH_DATASIZE;
  hspi1.Init.NSSPMode = SPI_NSS_PULSE_DISABLE;
  if (HAL_SPI_Init(&hspi1) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }
}

/* SPI2 init function */
static void MX_SPI2_Init(void)
{
  /* SPI2 parameter configuration*/
  hspi2.Instance = SPI2;
  hspi2.Init.Mode = SPI_MODE_MASTER;
  hspi2.Init.Direction = SPI_DIRECTION_2LINES;
  hspi2.Init.DataSize = SPI_DATASIZE_8BIT;
  hspi2.Init.CLKPolarity = SPI_POLARITY_LOW;
  hspi2.Init.CLKPhase = SPI_PHASE_1EDGE;
  hspi2.Init.NSS = SPI_NSS_SOFT;
  hspi2.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_4;
  hspi2.Init.FirstBit = SPI_FIRSTBIT_MSB;
  hspi2.Init.TIMode = SPI_TIMODE_DISABLE;
  hspi2.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
  hspi2.Init.CRCPolynomial = 7;
  hspi2.Init.CRCLength = SPI_CRC_LENGTH_DATASIZE;
  hspi2.Init.NSSPMode = SPI_NSS_PULSE_ENABLE;
  if (HAL_SPI_Init(&hspi2) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }
}

/* TSC init function */
static void MX_TSC_Init(void)
{
    /**Configure the TSC peripheral
    */
  htsc.Instance = TSC;
  htsc.Init.CTPulseHighLength = TSC_CTPH_2CYCLES;
  htsc.Init.CTPulseLowLength = TSC_CTPL_2CYCLES;
  htsc.Init.SpreadSpectrum = DISABLE;
  htsc.Init.SpreadSpectrumDeviation = 1;
  htsc.Init.SpreadSpectrumPrescaler = TSC_SS_PRESC_DIV1;
  htsc.Init.PulseGeneratorPrescaler = TSC_PG_PRESC_DIV4;
  htsc.Init.MaxCountValue = TSC_MCV_8191;
  htsc.Init.IODefaultMode = TSC_IODEF_OUT_PP_LOW;
  htsc.Init.SynchroPinPolarity = TSC_SYNC_POLARITY_FALLING;
  htsc.Init.AcquisitionMode = TSC_ACQ_MODE_NORMAL;
  htsc.Init.MaxCountInterrupt = DISABLE;
  htsc.Init.ChannelIOs = TSC_GROUP1_IO4|TSC_GROUP2_IO4|TSC_GROUP3_IO3;
  htsc.Init.ShieldIOs = 0;
  htsc.Init.SamplingIOs = TSC_GROUP1_IO3|TSC_GROUP2_IO3|TSC_GROUP3_IO2;
  if (HAL_TSC_Init(&htsc) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }
}

/* USB init function */
static void MX_USB_PCD_Init(void)
{
  hpcd_USB_FS.Instance = USB;
  hpcd_USB_FS.Init.dev_endpoints = 8;
  hpcd_USB_FS.Init.speed = PCD_SPEED_FULL;
  hpcd_USB_FS.Init.ep0_mps = DEP0CTL_MPS_64;
  hpcd_USB_FS.Init.phy_itface = PCD_PHY_EMBEDDED;
  hpcd_USB_FS.Init.low_power_enable = DISABLE;
  hpcd_USB_FS.Init.lpm_enable = DISABLE;
  hpcd_USB_FS.Init.battery_charging_enable = DISABLE;
  if (HAL_PCD_Init(&hpcd_USB_FS) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }
}

/** Configure pins as
        * Analog
        * Input
        * Output
        * EVENT_OUT
        * EXTI
*/
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct;

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_14|GPIO_PIN_15|NCS_MEMS_SPI_Pin|EXT_RESET_Pin
                          |LD3_Pin|LD6_Pin|LD4_Pin|LD5_Pin
                          |GPIO_PIN_10|GPIO_PIN_12, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_12|GPIO_PIN_3|GPIO_PIN_7, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8|GPIO_PIN_9|GPIO_PIN_10|GPIO_PIN_15, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOD, GPIO_PIN_2, GPIO_PIN_RESET);

  /*Configure GPIO pins : PC13 PC4 */
  GPIO_InitStruct.Pin = GPIO_PIN_13|GPIO_PIN_4;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  /*Configure GPIO pins : PC14 PC15 NCS_MEMS_SPI_Pin EXT_RESET_Pin
                          LD3_Pin LD6_Pin LD5_Pin PC10
                          PC12 */
  GPIO_InitStruct.Pin = GPIO_PIN_14|GPIO_PIN_15|NCS_MEMS_SPI_Pin|EXT_RESET_Pin
                          |LD3_Pin|LD6_Pin|LD5_Pin|GPIO_PIN_10
                          |GPIO_PIN_12;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  /*Configure GPIO pins : MEMS_INT1_Pin MEMS_INT2_Pin */
  GPIO_InitStruct.Pin = MEMS_INT1_Pin|MEMS_INT2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_EVT_RISING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  /*Configure GPIO pin : B1_Pin */
  GPIO_InitStruct.Pin = B1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_EVT_RISING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : PB12 PB3 PB7 */
  GPIO_InitStruct.Pin = GPIO_PIN_12|GPIO_PIN_3|GPIO_PIN_7;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /*Configure GPIO pin : LD4_Pin */
  GPIO_InitStruct.Pin = LD4_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
  HAL_GPIO_Init(LD4_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : PA8 */
  GPIO_InitStruct.Pin = GPIO_PIN_8;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pins : PA9 PA10 PA15 */
  GPIO_InitStruct.Pin = GPIO_PIN_9|GPIO_PIN_10|GPIO_PIN_15;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pin : PD2 */
  GPIO_InitStruct.Pin = GPIO_PIN_2;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

  /*Configure GPIO pin : PB6 */
  GPIO_InitStruct.Pin = GPIO_PIN_6;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
}

/**
  * @brief  This function is executed in case of error occurrence.
  * @param  file: The file name as string.
  * @param  line: The line in file as a number.
  * @retval None
  */
void _Error_Handler(char *file, int line)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */
  while(1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t* file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

Bibliography

[Agi02] Agilent Technologies Inc. *Agilent 33120A 15MHz Function/Arbitrary Waveform Generator*, 03 2002. Version 6.

[BST16] Davide Bellizia, Giuseppe Scotti, and Alessandro Trifiletti. On-Chip Analog Current Equalizer as a Countermeasure Against Side-Channel Attacks in CMOS Nanometer Technology. available at http://ieeexplore.ieee.org/document/7529737/, 2016.

[DMN^{+}14] Debayan Das, Shovan Maity, Saad Bin Nasir, Santosh Ghosh, Arijit Raychowdhury, and Shreyas Sen. High Efficiency Power Side-Channel Attack Immunity using Noise Injection in Attenuated Signature Domain. available at http://ieeexplore.ieee.org/document/7951799/, 2014.

[Gun06] Gunter Winkler and Bernd Deutschmann. Störfestigkeit integrierter Schaltungen gegenüber hochfrequenten leitungsgeführten Störgrößen. 4. EMV Fachtagung TU Graz, 2006.

[HKP^{+}10] Harald Hartl, Edwin Krasser, Wolfgang Pribyl, Peter Söser, and Gunter Winkler. *Elektronische Schaltungstechnik - Mit Beispielen in PSpice*, volume 3. Pearson Studium, 2010.

[Int06] International Organization for Standardization. ISO7816-3 - Identification Cards - Integrated Circuit Cards - Part 3: Cards with contacts — Electrical interface and transmission protocols, 2006.

[Int07] International Organization for Standardization. ISO7816-2 - Identification Cards - Integrated Circuit Cards - Part 2: Cards with contacts — Dimensions and location of the contacts, 2007.

[Int11] International Organization for Standardization. ISO7816-1 - Identification Cards - Integrated Circuit Cards - Part 1: Cards with contacts — Physical characteristics, 2011.

[LWP^{+}08] Huiyun Li, Keke Wu, Bo Peng, Yiwei Zhang, Xinjian Zheng, and Fengqi Yu. Enhanced Correlation Power Analysis Attack on Smart Card. available at http://ieeexplore.ieee.org/document/4709305/, 2008.

[MA10] Hanan Mahmoud and Khaled Alghathbar. Novel Algorithmic Countermeasures for Differential Power Analysis Attacks on Smart Cards. available at http://ieeexplore.ieee.org/document/5604079/, 2010.

[Maj16] Majeed Ahmad and Lee Teschler . How and when MOSFETs blow up. available at https://www.powerelectronictips.com/how-and-when-mosfets-blow-up/, 2016.

[MAK15] Hridoy Jyoti Mahanta, Abul Kalam Azad, and Ajoy Kumar Khan. Power Analysis Attack: A Vulnerability to Smart Card Security. available at http://ieeexplore.ieee.org/document/7058206/, 2015.

[MDS02] Thomas S. Messerges, Ezzat A. Dabbish, and Robert H. Sloan. Examining Smart-Card Security under the Thread of Power Analysis Attacks. available at http://ieeexplore.ieee.org/document/1004593/, 2002.

[MG08] Radu Muresan and Stefano Gregori. Protection Circuit against Differential Power Analysis Attacks for Smart Cards. available at http://ieeexplore.ieee.org/document/4585359/, 2008.

[MM16] Matthew Mayhew and Radu Muresan. Implementation of a decoupling based power analysis attack countermeasure. available at http://ieeexplore.ieee.org/document/7742705/, 2016.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks – Revealing the Secrets of Smart Cards*. Springer Science+Business Media, LLC, 2007.

[MTS11] Cancio Monteiro, Yasuhiro Takahashi, and Toshikazu Sekine. Resistance Against Power Analysis Attacks on Adiabatic Dynamic and Adiabatic Differential Logics for Smart Card. available at http://ieeexplore.ieee.org/document/6146067/, 2011.

[Mul13] Multicomp. *NPN General Purpose Amplifier*, 04 2013. Rev. 1.

[Now12] Rich Nowakowski. Techniques For Implementing A Positive And Negative Output Voltage For Industrial And Medical Equipment. available at http://www.ti.com/lit/ml/szzn001/szzn001.pdf, 2012.

[ON 17] ON Semiconductor. *FDV303N Digital FET N-Channel*, 10 2017. Rev. 4.

[Pan14] Panasonic. *CT1000 Family - Identification IC Tester Data Sheet*, 05 2014.

[Pic] Pickering. *Single Pole 4mm^2 Reed Relays*.

[POM07] Thomas Popp, Elisabeth Oswald, and Stefan Mangard. Power Analysis Attacks and Countermeasures. available at http://ieeexplore.ieee.org/document/4397178/, 2007.

[RE02] Wolfgang Rankl and Wolfgang Effing. *Handbuch der Chipkarten – Aufbau – Funktionsweise – Einsatz von Smart Cards*, volume 10. Carl Hanser Verlag München Wien, 2002.

[SPEa] SPEA SpA. *CT1000 Family System Interface*. Version 7.

[SPEb] SPEA SpA. *CT1000 Family System Technical Info.* Version 7.

[SPE16] SPEA SpA. *CT1000 Family - Identification IC Tester Data Sheet*, 07 2016. Version 11.

[ST 17] ST Microelectronics. *STM32F072x8 STM32F072xB*, 01 2017. Rev. 5.

[ST07] Christoph Schenk and Ulrich Tietze. *Electronic Circuits*, volume 2. Springer, 2007.

[Tex15] Texas Instruments. *LP3869x/-Q1 500-mA Low-Dropout CMOS Linear Regulators Stable With Ceramic Output Capacitors*, 12 2015.

[Tex16a] Texas Instruments. *LM5110 Dual 5-A Compound Gate Driver With Negative Output Voltage Capability*, 09 2016.

[Tex16b] Texas Instruments. *TPL0501 256-Taps Single-Channel Digital Potentiometer With SPI Interface*, 05 2016.

[Tex18] Texas Instruments. *TPS565208 Synchronous Step-Down Voltage Regulator*, 06 2018.

[XIL18] XILINX. *Spartan-3E FPGA Family Data Sheet*, 12 2018.

[YK17] Weize Yu and Selcuk Köse. Implications of Noise Insertion Mechanisms of Different Countermeasures Against Side-Channel Attacks. available at http://ieeexplore.ieee.org/document/8050635/, 2017.

[YWCZ14] Jianlei Yang, Chenguang Wang, Yici Cai, and Qiang Zhou. Power Supply Noise Aware Evaluation Framework for Side Channel Attacks and Countermeasures. available at http://ieeexplore.ieee.org/document/7082770/, 2014.