Alexander Knoll, BSc

Design Process and Verification for a Multichannel Electrode Switching Unit

Master's Thesis

to achieve the university degree of
Dipl. Ing.

submitted to
Graz University of Technology

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

Institute of Electronics

Graz, October 2018

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master‘s thesis.

Date

Signature

Abstract

The goal of this thesis is to develop an automated test device, which verifies the functionality of a unit of a product. Deviations to the specification and function should be identified and recorded in a report. The design process is in consideration of the current standards.

For the hardware design, the schematic and the printed circuit board were designed. For the software, the interface controlling the device was programmed.

Furthermore, the test cases were defined, to verify functionality and compliance with the specification. Finally, an automated test report was generated, documenting the results, which can be archived for quality management.

Successful diagnosis of various errors proved the functionality of the test device. The changeable test adapter allows the test device to be used for similar units with different connectors.

Keywords: multichannel switching unit, design process, verification, functional test, test adapter, test report

Kurzfassung

Das Ziel dieser Arbeit war es, ein automatisiertes Testgerät zu entwickeln, um die Funktionalität einer Komponente eines Produktes zu verifizieren. Unterschiede zur Spezifikation und Funktion sollten erkannt und ein Bericht erstellt werden. Bei diesem Designprozess müssen die entsprechenden Normen berücksichtigt werden.

Im Hardware-Design wurden der Schaltplan und die Leiterplatte entworfen. Im Software-Design wurde die Programmoberfläche zur Steuerung des Gerätes programmiert.

Weiters wurden die Testfälle definiert, um die Funktion und die Einhaltung der Spezifikation zu überprüfen. Abschließend wird ein automatisierter Testbericht erstellt, um die Ergebnisse zu dokumentieren, der für das Qualitätsmanagement archiviert werden kann.

Die erfolgreiche Identifizierung verschiedener Fehler hat die Funktionsfähigkeit des Testgeräts bestätigt. Durch den wechselbaren Testadapter kann das Testgerät auch für ähnliche Komponenten mit unterschiedlichen Konnektoren verwendet werden.

Schlüsselwörter: Mehrkanal Schaltmatrix, Design Prozess, Verifikation, Funktionstest, Test Adapter, Testbericht

Contents

List of Figures

List of Tables

Abbreviations

Abbreviation	**Meaning**
EEG	Electroencephalography
PC	Personal Computer
GND	Ground
GUI	Graphical User Interface
LED	Light Emitting Diode
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
USB	Universal Serial Bus
UART	Universal Asynchronous Receiver Transmitter
FTDI	Future Technology Devices International Ltd.
PCB	Printed Circuit Board
RTFM	Real-time Functional Mapping
ECS	Electrical Cortical Stimulation
FES	Functional Electrical Stimulaton
QMS	Quality Management System
DHF	Design History File
ADC	Analog to Digital Converter
API	Application Programming Interface
LDO	Low-dropout Regulator
Opamp	Operational Amplifer
DAC	Digital to Analog Converter
DUT	Device Under Test
HAL	Hardware Abstraction Layer
IC	Integrated Circuit
FPC	Flex-print Cable
PWM	Pulse Width Modulation
SPI	Serial Peripheral Interface
SAR	Successive Approximation
CAD	Computer-aided Design
ACK	Acknowledge Byte
I2C	Inter-Integrated Circuit
XML	Extensible Markup Language

Symbols

Symbols	**Meaning**
A	Ampere
m	milli
M	Mega
G	Giga
Hz	Hertz
k	kilo
p	pico
s	seconds
V	Volt
W	Watt
μ	micro
Ω	Ohm

1. Introduction

1.1. Motivation

Medical grade devices must pass several tests during design and production to verify compliance to specific standards. Such verification tests and if necessary product validations are required to finally get a product certification. To reduce the needed time for testing and re-testing, automated testing devices are beneficial to ensure the quality of products. As black box testing cannot always be applied to verify certain functionality of a device, typically several functional units of a device are identified during design which can then be tested in an easier and better way.

1.2. Goal

The goal of this master thesis is firstly to investigate the applicable standards, guidelines and design process for a medical device and secondly to design and construct a testing device to support specific unit testing of the medical device. For that the state of the art will be shown. Then the developed test device will be described in detail. It is responsible to verify a unit of a medical device, in this case one of the 16 modules. This is necessary to save time and money during the production process and also to verify the design easily. Also, a section with the results and an outlook visualize the possibilities.

1.3. Electrical stimulation

Electrical stimulation can be used to stimulate the brain with certain electrical current pulses. With an electrical stimulator and two electrodes this stimulation signal is applied to the brain. The current is selectable between 50 μA and 15 mA and the frequency from 1-500 Hz with a phase duration up to 1 ms. For safety reasons the charge density is calculated and a warning is shown in the graphical user interface (GUI) if a defined limit is exceeded.

These pulses have the following parameters (see figure 1.1).

- Used waveform (steady and alternating rectangular)
- Frequency
- Amplitude of the current
- Pulse width
- Duration of the stimulation

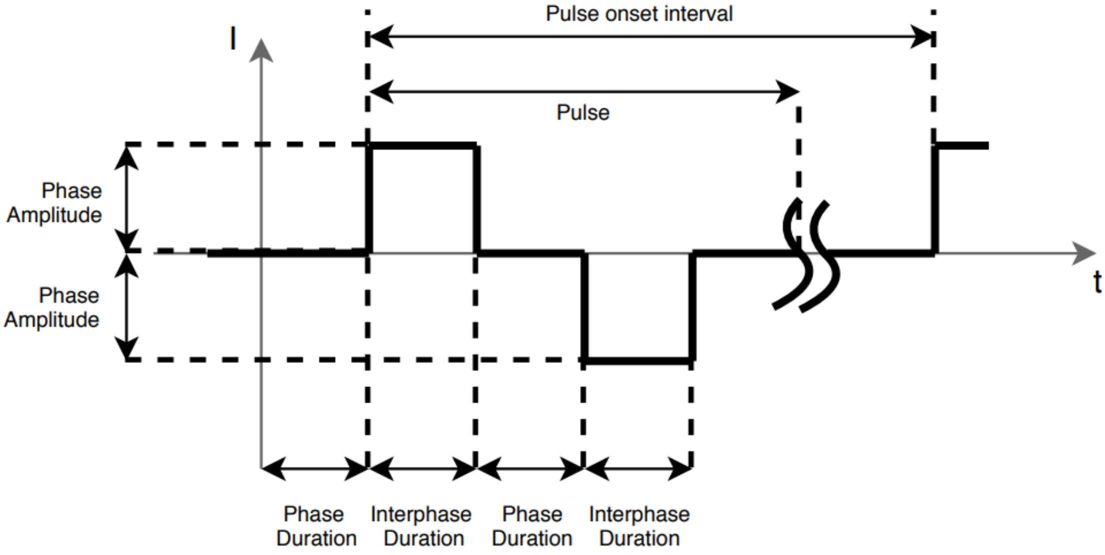


Figure 1.1.: Various durations in one stimulus

1.4. Brain mapping

One usecase of electrical stimulation is brain mapping. The electroencephalography (EEG) is recorded and with two electrodes, anode and cathode, an electrical stimulation is applied. After the stimulation is finished the response of the potentials are an indication to recognize the function of this area in the brain. The most important areas are the motor, sensory, language-related, visual and auditory functions.
Generally, the areas can be described as

- Motoric (*"Relating to muscular movement."*[23])
- Perception (*"The ability to see, hear, or become aware of something through the senses."*[24])
- Cognition (*"The mental action or process of acquiring knowledge and understanding through thought, experience, and the senses."*[25])
- Memory (*"The faculty by which the mind stores and remembers information."*[22])

The localization of these areas are necessary at brain surgery, when the patient has a brain tumor which should be removed without damaging the important brain areas. Also, for treatment of some diseases, like epilepsy this is used. Electrical stimulation is also used for therapy, often as part of the presurgical part to identify the functional brain regions.

1.5. Multichannel switching unit [26]

Epilepsy is a brain disorder which produces steady seizures. A seizure is an abnormal neuronal activity in the brain. That can affect the different brain areas, like described in chapter 1.4. Mostly these seizures are started without a special cause. With anti-epileptic drugs these seizures can be minimized. Also, it is sometimes possible to remove a brain region in a surgical therapy to stop some kinds of seizures.

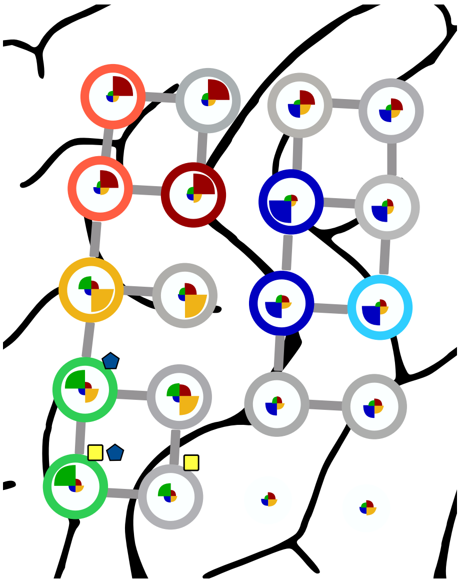


Figure 1.2.: *"Data display in the main application for a 20-channel electrode grid. The pie-charts display the results of four real-time functional mapping (RTFM) tasks in different colors. The outer rings display the color-coded categories of an electrical cortical stimulation (ECS) mapping. The grey lines in between the electrodes represent stimulation events."* [26]

A big advantage especially in therapy of epilepsy is, to combine ECS with RTFM and a so called multichannel switching unit. Due to the combination of these systems the risk of a seizure is reduced, because the necessary stimulations and time spent for the whole process is reduced.

The hardware components for the combined system are a biosignal amplifier, a cortical stimulator and the multichannel switching unit (see figure 1.3). The devices are controlled via USB. The electrodes are connected to the headbox. Per default all channels are set to measuring, so they are routed to the amplifier. With the amplifier up to 256 channels can be measured and the digitized data is sent to the recording computer software. With the software of the switching unit electrodes can be chosen to be the anode or cathode for a cortical stimulation. So if a trigger of the stimulation is set to the switching unit, it connects these electrodes to the outputs of the stimulator. A possible current up to 15 mA with a frequency from 1 to 500 Hz is sent to the electrodes as a rectangular pulse (see chapter 1.3).

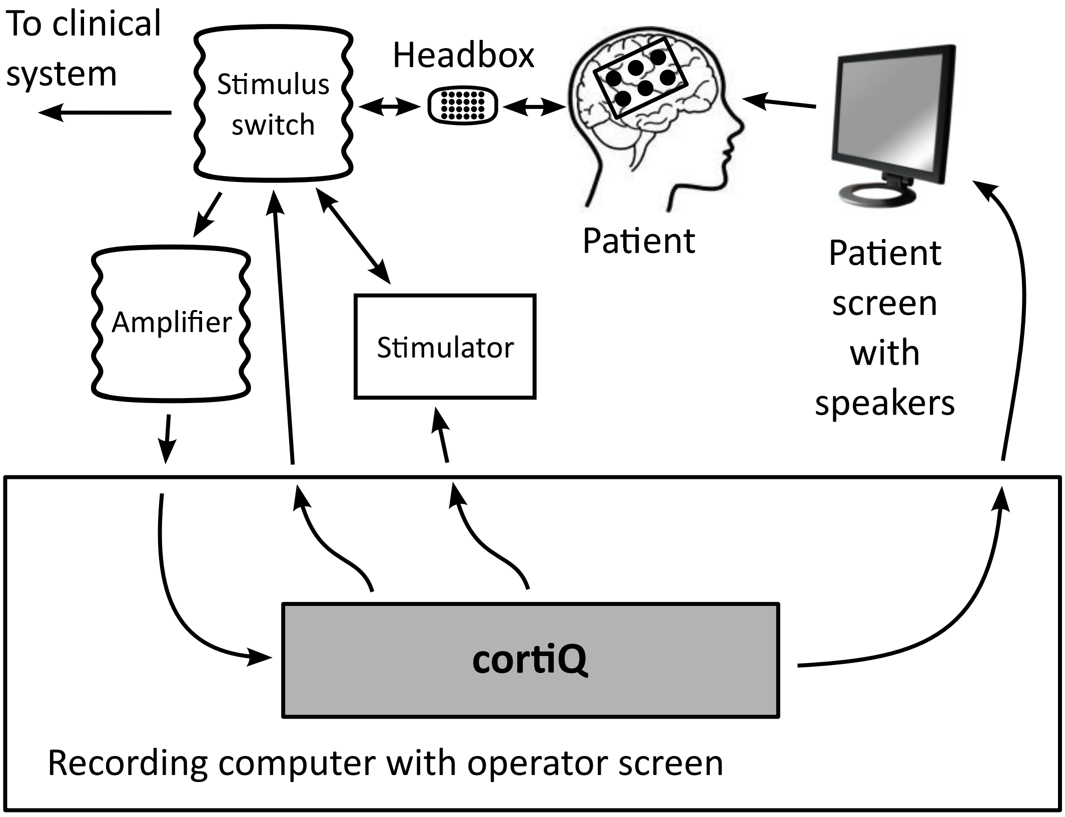


Figure 1.3.: Architecture of the combined system. The used devices are a biosignal amplifier, a cortical stimulator and a multichannel switching unit. The electrodes are connected via the headbox. [26]

1.5.1. Device under test (DUT)

In this section the DUT is described in principal since it is important to verify the test device functions. The DUT has 16 channels and each channel is in principal an interconnection of switches and a buffer between the electrode (input) and the amplifier (output) with the possibilities to switch a stimulator anode (STIM-) or cathode (STIM+) to any electrode and switches the buffer away.

1.5.1.1. Block Diagram

In figure 1.4 the basic block diagram of a module is shown. It consists of a microcontroller and 16 channels. To each channel the cathode and anode of the electrical stimulator is wired. Each channel is wired to a fixed electrode connector and has a buffer from the electrode to the amplifier connection. The outputs to the amplifier channel are also connected to two 8-bit multiplexers (Amp_1-8 to MUX_1 and Amp_9-16 to MUX_2) which is then a 16:2 multiplexer (AUDIO_O_1 and AUDIO_O_2).

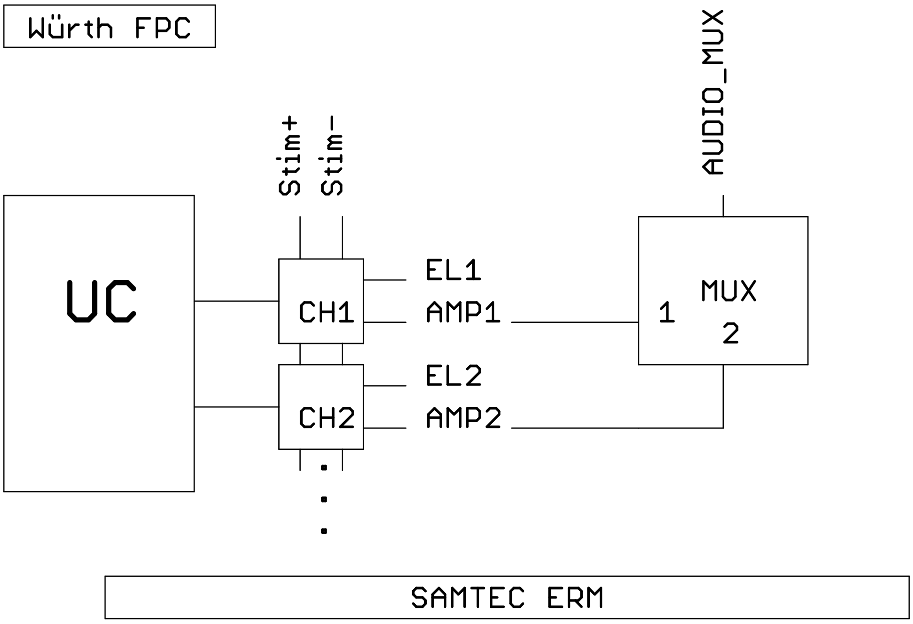


Figure 1.4.: Block diagram of a module

1.5.1.2. Switch control

In figure 1.5 the switch interconnection of one channel is shown. A_SW and B_SW are switching the stimulator output electrodes to the electrode, when C1_SW, also called safety switch, is closed. When C1_SW is closed, C2_SW will be opened to interrupt the connection to the buffer, to ensure it is not being damaged during the stimulation. If C2_SW and D_SW are closed, the electrode has ground (GND) connection and therefore it is the reference potential for the measurements. All switches are controlled by the microcontroller on the module, with the digital outputs A, B, C and D corresponding to the respective switches.

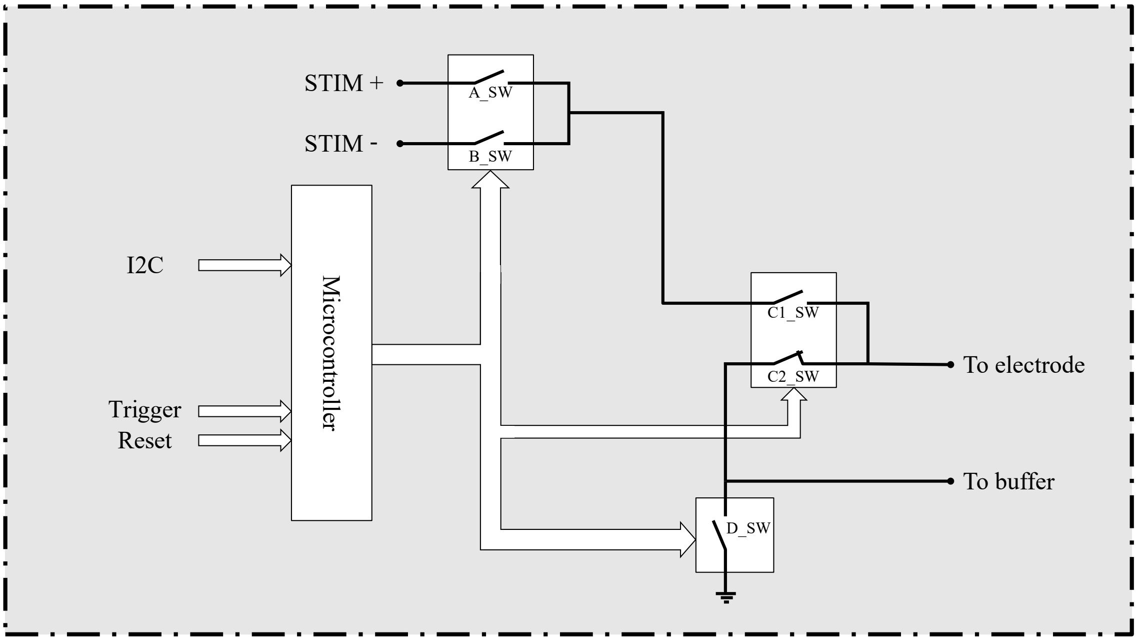


Figure 1.5.: Block diagram of one channel

The microcontroller on the module (uC_Module) is only for switching the switches and is connected to the microcontroller via inter-integrated circuit (I2C). If the trigger input has a rising edge the outputs will be switched like configured before via I2C. When the reset input line goes to HIGH, all outputs are set to LOW (see figure 1.6)
Module microcontroller: PIC24FJ128GA310

If the digital output C is HIGH, C1_SW, D_SW are closed and C2_SW is opened. If the digital output A or B is HIGH, switch A_SW or B_SW is closed. If all outputs are LOW, the channel is in measuring mode. If A and C are HIGH the electrode is configured as STIM+, with B and C HIGH the electrode is configured as STIM- and with D HIGH as GND reference electrode.

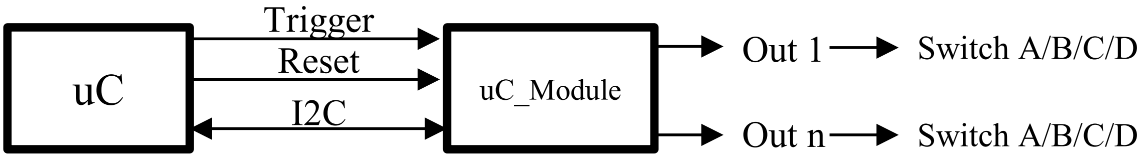


Figure 1.6.: Switch control block diagram

For the switch test it is important that C1_SW and C2_SW can be switched independently. In figure 1.7 the ST and HI switches are shown, which are used for individually switching C1_SW and C2_SW. If ST is LOW and C is HIGH, only C2_SW is switched (used for safety switch off). If HI is LOW and C is HIGH, only C1_SW is switched (used for the selftest). If HI and ST both are HIGH and C is HIGH, both switches are switched (see table 1.1).

Table 1.1.: Truth table c-switch

C	**HI**	**ST**	**C2**	**C1**
0	X	X	closed	open
1	0	0	closed	open
1	0	1	closed	closed
1	1	0	open	open
1	1	1	open	closed

In the following table the channel functions in dependents of A, B, C and D are described.

Table 1.2.: Channel configuration

A	B	C (1 ->C1 closed C2 open) (0 ->C1 open C2 closed) (HI & ST ->1)	D	Function
1	0	1	0	Stim+
0	1	1	0	Stim-
0	0	0	1	GND
0	0	0	0	Measure (default)

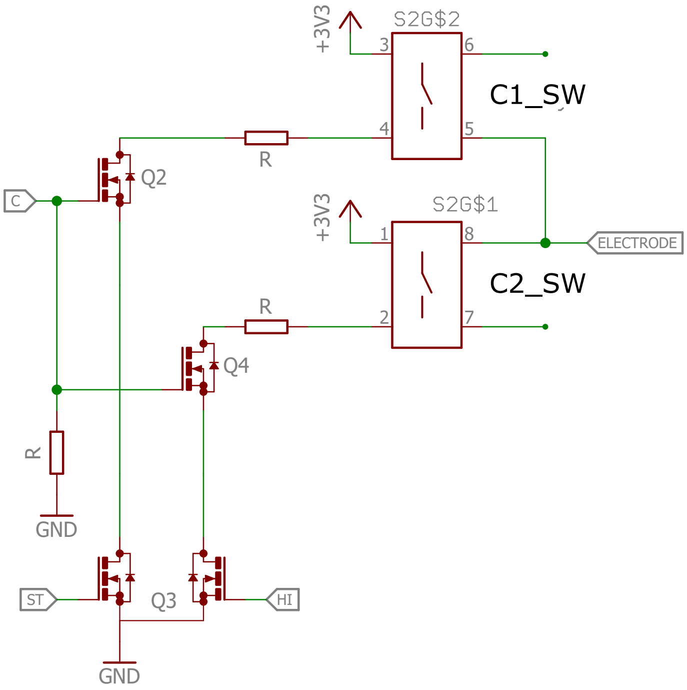


Figure 1.7.: Schematic for switching C1_SW and C2_SW independently

1.5.1.3. Audio 16:2 Multiplexing

The 16 amplifier channels of the module are multiplexed to two channels (AUDIO_O_1 and AUDIO_O_2) with two 8:1 analog multiplexers, so a 16:2 multiplexer is built.

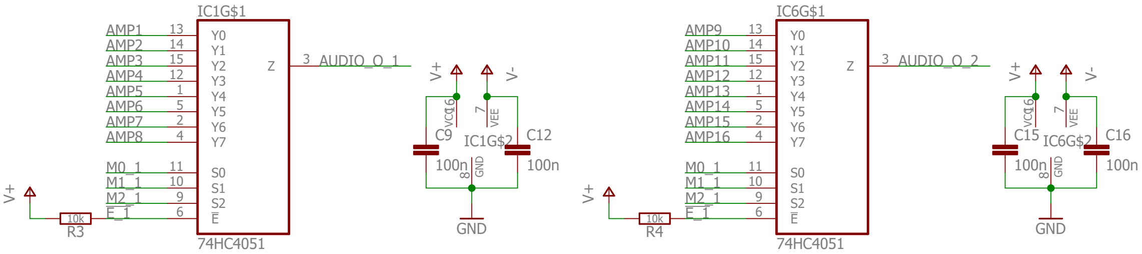


Figure 1.8.: Schematic for audio multiplexing

1.6. Quality management systems according to EN ISO 13485 [17]

In this standard the quality management system (QMS) is described which is required for a company that manufactures medical devices and wants to sell them. If a company is certified with this standard, it complies to the european QMS legal requirements. For the international compliance, especially for USA also the FDA 21 CFR part 820 has to be fulfilled (see chapter 1.7).

The essential chapters are:

- Quality management system
- Management responsibility
- Resource management
- Product realization
- Measurement, analysis and improvement

Quality management system

In this chapter general and documentation requirements for a QMS are described.

Management responsibility

The top management is responsible to review the organization's QMS in planned and documented intervals, to ensure the continuity of effectiveness, suitability and adequacy.

Resource management

The organization shall document the processes needed for the training of employees and the requirements needed, to achieve conformity to the product requirements, for the work environment. Further the process equipment and other infrastructure must be appropriate.

Product realization

Every process of the product realization must be planned, developed and documented. Also, the risk management is part of the product realization, which record shall be maintained. The whole design process requirements are described in this chapter. These are similar to the CF21 Part 820 Subpart C (more in chapter 1.7)

Measurement, analysis and improvement

This part is to define the important points for the evaluation, like monitoring and measurement, control of a nonconforming product, analyzing of data and improvements.

1.7. Design controls according to 21 CFR part 820.30 [8]

For the design of medical devices it is very important to make sure that the design process is a controlled process with the following steps. In the CFR 21 part 820.30, just like in the EN ISO 13485 the following steps are required to establish a company's conformity with these standards. Each organization shall establish and maintain procedures to control the design and development process to meet the product specification.

Design and development planning

Each manufacturer shall establish and maintain plans to identify and describe the interface with different groups and activities, that are resulting in a design input. These plans should be reviewed and approved.

Design input

Procedures shall be established to ensure that the design requirements relating to a medical product are appropriate and the intended use is met, including the user and patient needs. The design inputs shall be reviewed and approved with date and signature of the responsible persons.

Design output

The design outputs shall meet the design input requirements and provide information for purchasing and production. A process shall be defined to document the output in such a way that the conformance to the input can be verified. Records of the results of the review shall be maintained and approved with date and signature of the responsible persons.

Design review

At defined stages the systematic review of the design and development shall be performed, to evaluate the ability of the results to meet the specification and user needs and to identify necessary actions. The results of the review, including the identification of the design, the participants of the review and the date should be documented in the design history file (DHF).

Design verification

Verification shall confirm that the output meets the design input requirements. The results, including identification of the design process, methods, the date and the employee performing the verification shall be also documented in the DHF.

Design validation

The validation is done under defined operation conditions on representative products like initial production units or batches, to define if the user needs are fulfilled. This step should include methods and acceptance criteria. This is also where the software validation takes place and the risk analysis should be verified to be appropriate. The results are also stored in the DHF.

Design transfer

The process to translate the design into production specification shall be established by the organization.

Design changes

Each design change must be done according to a defined process, which includes the identification, documentation, verification, review and approval of the changes before they are implemented.

Design history file

The DHF shall be maintained and established for each device and should contain or reference the records, which are necessary to show that the device was developed conforming to this part.

1.8. Verification of a module from a multichannel switching unit

1.8.1. Basic concept

There are two important verifications, firstly the test after the design to verify the function of the schematic and secondly the tests after the production of each device. The design verification is possible to be done by hand, but for the production tests an automated test device is necessary to save time and money, because every device must be tested, to ensure proper operation. Hence the focus of the test cases is on the production tests, a testing system is built which checks the functionality of the device. A detailed test is made with a PC which is creating an automated test report afterwards which can be printed and is additionally saved as ".pdf" file, with the option to commit it to a Subversion repository.

1.8.2. Description of the test cases

In the following sections the different test cases are described. They are separated into 3 parts.

- Switches
- Crosstalk
- Signal

Test A - Switches

This test is done to ensure that all switches are working as intended. It checks each switch, if it is able to switch on and off. For that a voltage is applied and the switches are closed in a row and pull the potential to ground. Then each switch opens and closes again and so the microcontroller which is connected to the top, shows a falling / rising edge each time, when the digital value at the corresponding pin is read.

Test B - Crosstalk

This test is optional and therefore not described in detail. In principal a sinusoidal signal with 10-20 Hz and in the range of +-200 mV is applied at one channel and all channel outputs are measured. Then only at the applied channel the signal is measured. At all other channels the correlation coefficients must not exceed a defined limit. This test is repeated for all channels.

Test C - Signal

For testing the buffer and audio multiplexer, a sinusoidal signal is applied at the input and the output is measured. This happens for each channel from 1 to 16. For the comparison the correlation coefficients are used. If they are in a defined range the test is passed. Since the measured signal with the module is a bipolar signal, it must be ensured that the test signal is also bipolar, in the range of +-200 mV and 10-20 Hz.

2. Methods

2.1. Hardware

In this chapter the hardware design is described in detail and the used parts and schematics are explained.

2.1.1. Concept

Due to the information from chapter 1.8 and the given guidelines a test device was built with the following requirements:

- It should be an automated testing system which is connected and powered via USB.
- There should be a switch to reset the test device without the need to unplug the cable.
- For the connection of the module a test adapter has to be built due to the mating cycles limitation of the used connector.
- The PC-software should be programmed in *Python*.
- The firmware should be programmed in *C*.
- A test report should be generated.
- With the test results it should be possible to determine if the module is working like expected.

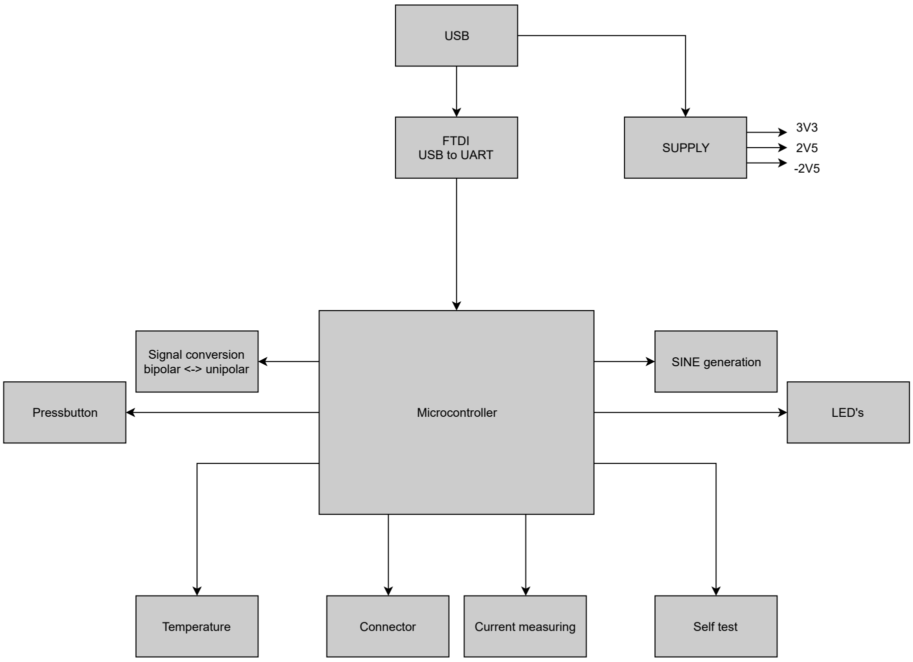


Figure 2.1.: The control unit of the circuit is a microcontroller. For the communication a USB to universal asynchronous receiver transmitter (UART) converter from Future Technology Devices International Ltd. (FTDI) is used. The different parts are called units.

Figure 2.1 is showing the single units and their connection to the microcontroller. The entire device is controlled by this microcontroller. The communication with the PC is done via USB and a USB to UART converter. The device is supplied via USB and then converted to the necessary voltages. Light emitting diodes (LEDs) are used to indicate the device status, and a push button can be used for an optional test mode without a PC. The test circuit can be connected to the device under test (DUT) with some solid state relays and metal-oxide-semiconductor field-effect transistors (MOSFET). A digital to analog converter (DAC) is used for the sine wave generation, with a look up table for the values. The measurement is done via a bipolar to unipolar converter circuit and the built in analog to digital converter (ADC) of the microcontroller. It is also possible to measure the temperature in the test device using a temperature sensor connected to an ADC channel. The individual parts are described in detail in the following sections.

2.1.2. Design and development

2.1.2.1. Supply

The device is powered via 5 V from USB, converted to +-3.3 V and +- 2.5 V with a low-dropout regulator (LDO). The 3.3 V are used for the microcontroller and the DAC. The +-2.5 V are used to supply the operational amplifiers (opamp), since the signal range of the module is also +-2.5 V. The voltage regulators are two adjustable TPS7A4501 for +3.3 V and +2.5 V, an LM2663 for -3.3 V and a TPS72301 for -2.5 V. The -3.3V are only used to get the -2.5 V. The schematic is shown in figure 2.2.

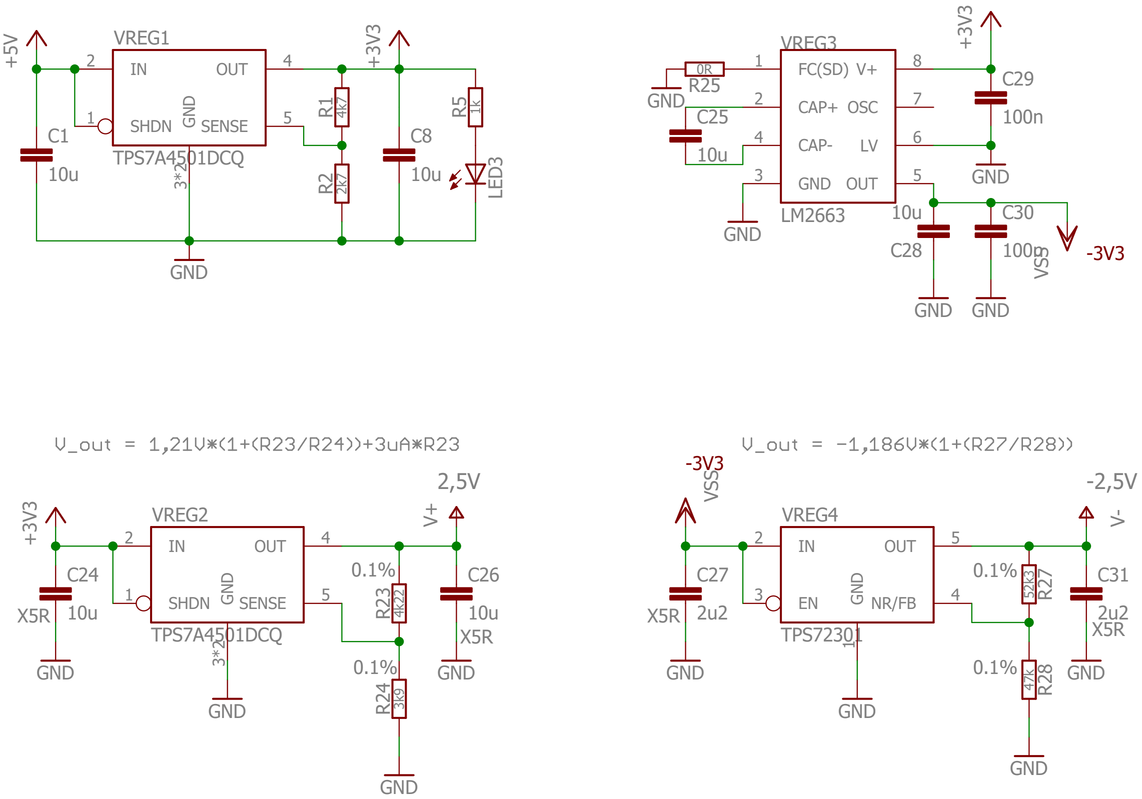


Figure 2.2.: Power is supplied via the 5 V from USB and then converted to +-3.3 V and +-2.5 V with low-dropout regulators

2.1.2.2. Communication [21]

For communication a USB to UART converter chip from Future Technology Devices International (FTDI) is used. It can communicate with an application programming interface (API) provided by the manufacturer. So it is easy to establish communication with the microcontroller, by the software, especially since it converts the USB to UART and the used microcontroller has a built in UART module. The block diagram is shown in figure 2.4. A more detailed description can be found in the datasheet.

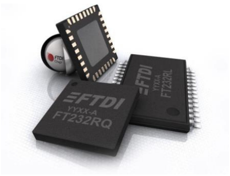


Figure 2.3.: FT232R chip from Future Technology Devices International Ltd. [21]

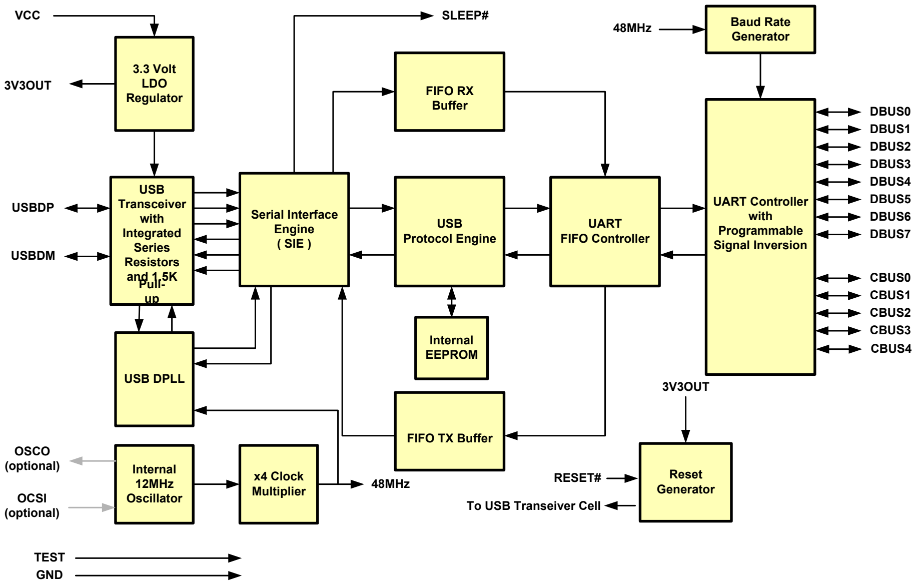


Figure 2.4.: The block diagram of the FT232R chip from Future Technology Devices International Ltd. [21]

2.1.2.3. Microcontroller[12]

The microcontroller was selected due to the good experiences made in the company. Therefore, it was predefined for this test device. A hardware abstraction layer (HAL) for the firmware and a development board for testing single parts already exists.

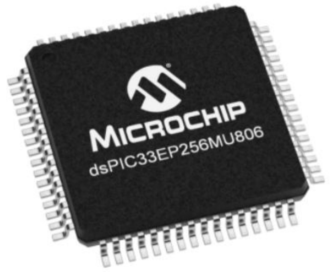
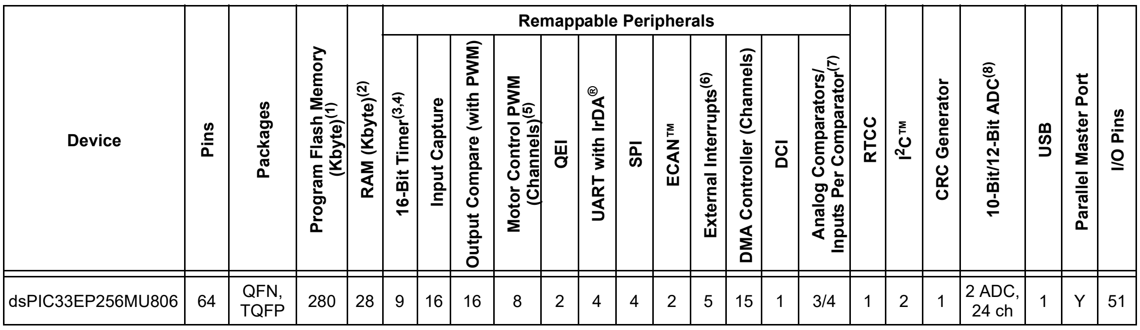


Figure 2.5.: The microcontroller is a 16-bit dsPIC33EP256MU806 from Microchip. [12]

The dsPIC33EP256MU806 from Microchip has several functions that are used. In figure 2.6 the peripherals and in figure 2.7 the pinning of the microcontroller is shown. Some of them like I2C and the analog input pins are fixed, others like SPI or external interrupts are remappable to several pins.



Device	Pins	Packages	Program Flash Memory (Kbyte)(1)	RAM (Kbyte)(2)	Remappable Peripherals												RTCC	I^2C™	CRC Generator	10-Bit/12-Bit ADC(8)	USB	Parallel Master Port	I/O Pins
					16-Bit Timer(3,4)	Input Capture	Output Compare (with PWM)	Motor Control PWM (Channels)(5)	QEI	UART with IrDA®	SPI	ECAN™	External Interrupts(6)	DMA Controller (Channels)	DCI	Analog Comparators/ Inputs Per Comparator(7)							
dsPIC33EP256MU806	64	QFN, TQFP	280	28	9	16	16	8	2	4	4	2	5	15	1	3/4	1	2	1	2 ADC, 24 ch	1	Y	51

Figure 2.6.: The peripherals of the microcontroller [12]

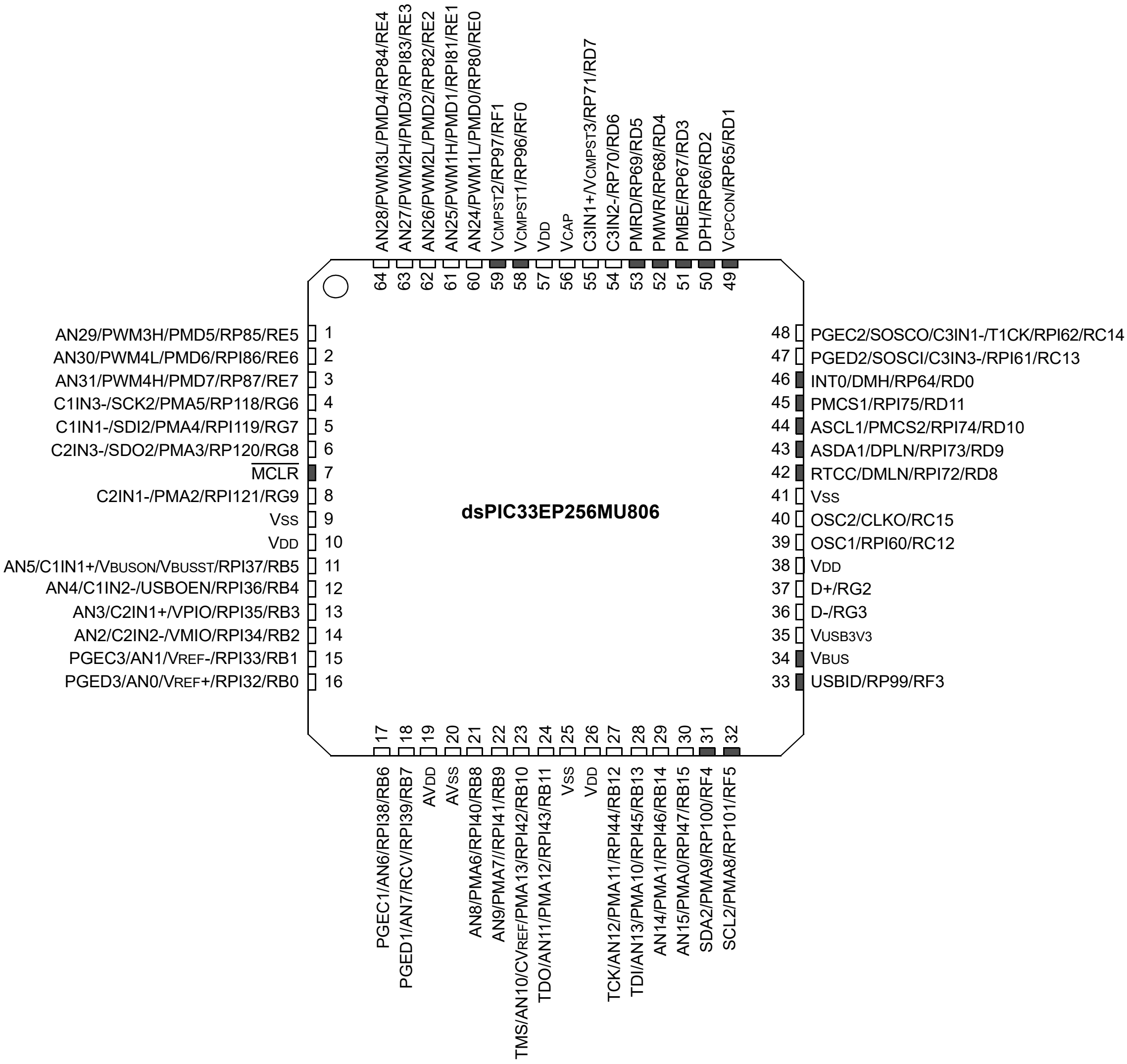

AN28/PWM3L/PMD4/RP84/RE4 64
AN27/PWM2H/PMD3/RPI83/RE3 63
AN26/PWM2L/PMD2/RP82/RE2 62
AN25/PWM1H/PMD1/RPI81/RE1 61
AN24/PWM1L/PMD0/RP80/RE0 60
VCMPST2/RP97/RF1 59
VCMPST1/RP96/RF0 58
VDD 57
VCAP 56
C3IN1+/VCMPST3/RP71/RD7 55
C3IN2-/RP70/RD6 54
PMRD/RP69/RD5 53
PMWR/RP68/RD4 52
PMBE/RP67/RD3 51
DPH/RP66/RD2 50
VCPCON/RP65/RD1 49
AN29/PWM3H/PMD5/RP85/RE5 1
AN30/PWM4L/PMD6/RPI86/RE6 2
AN31/PWM4H/PMD7/RP87/RE7 3
C1IN3-/SCK2/PMA5/RP118/RG6 4
C1IN1-/SDI2/PMA4/RPI119/RG7 5
C2IN3-/SDO2/PMA3/RP120/RG8 6
MCLR 7
C2IN1-/PMA2/RPI121/RG9 8
VSS 9
VDD 10
AN5/C1IN1+/VBUSON/VBUSST/RPI37/RB5 11
AN4/C1IN2-/USBOEN/RPI36/RB4 12
AN3/C2IN1+/VPIO/RPI35/RB3 13
AN2/C2IN2-/VMIO/RPI34/RB2 14
PGEC3/AN1/VREF-/RPI33/RB1 15
PGED3/AN0/VREF+/RPI32/RB0 16
dsPIC33EP256MU806
48 PGEC2/SOSCO/C3IN1-/T1CK/RPI62/RC14
47 PGED2/SOSCI/C3IN3-/RPI61/RC13
46 INT0/DMH/RP64/RD0
45 PMCS1/RPI75/RD11
44 ASCL1/PMCS2/RPI74/RD10
43 ASDA1/DPLN/RPI73/RD9
42 RTCC/DMLN/RPI72/RD8
41 VSS
40 OSC2/CLKO/RC15
39 OSC1/RPI60/RC12
38 VDD
37 D+/RG2
36 D-/RG3
35 VUSB3V3
34 VBUS
33 USBID/RP99/RF3
17 PGEC1/AN6/RPI38/RB6
18 PGED1/AN7/RCV/RPI39/RB7
19 AVDD
20 AVSS
21 AN8/PMA6/RPI40/RB8
22 AN9/PMA7//RPI41/RB9
23 TMS/AN10/CVREF/PMA13/RPI42/RB10
24 TDO/AN11/PMA12/RPI43/RB11
25 VSS
26 VDD
27 TCK/AN12/PMA11/RPI44/RB12
28 TDI/AN13/PMA10/RPI45/RB13
29 AN14/PMA1/RPI46/RB14
30 AN15/PMA0/RPI47/RB15
31 SDA2/PMA9/RP100/RF4
32 SCL2/PMA8/RP101/RF5

Figure 2.7.: The pinning of the microcontroller [12]

The microcontroller is programmable via a Microchip MPLAB ICD3 or a PICkit 3 which is using the PGD, PGC and MCLR pins as well as VCC and GND (see figure 2.8 and 2.9).

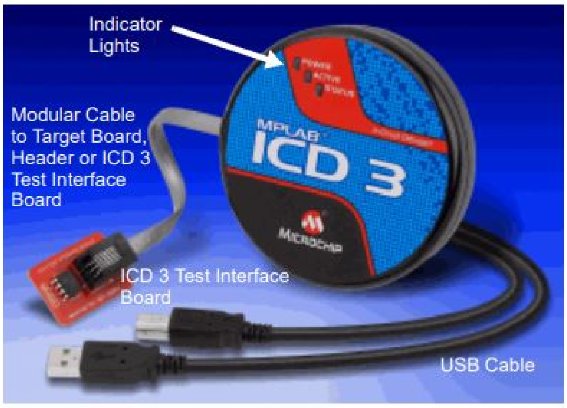

Indicator Lights
Modular Cable to Target Board, Header or ICD 3 Test Interface Board
MPLAB ICD 3
MICROCHIP
ICD 3 Test Interface Board
USB Cable

Figure 2.8.: The ICD3 is an in-circuit debugger controlled by the MPLAB X IDE software. It allows the user to debug his application with his own hardware, with hardware and software breakpoints and monitor the internal registers. It is also used to program the device. [14]

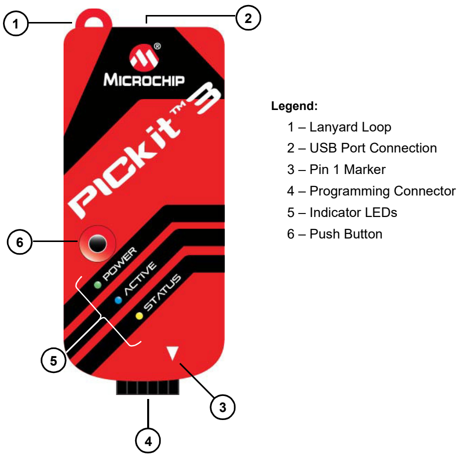


Figure 2.9.: The PICkit 3 is also an in-circuit debugger/programmer system that can be used for debugging in a similar way to the ICD 3, but it is a low-cost and simple version with less functionality. [15]

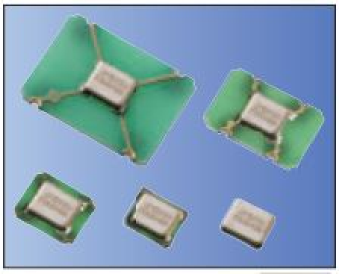


Figure 2.10.: For the clock of the microcontroller, a 4 MHz oscillator from Kyocera was used. [18]

2.1.2.4. Switch test unit

There are 3 resistors connected to two different voltages, which can be switched to the stimulation channel. One is leading to a current flow of $-48\ \mu A$ (1), one to $7\ \mu A$ (2) and one to $+40\ \mu A$ (3), which can be added to (2) (see figure 2.11). $I_{Measure}$ is measured with an ADC channel of the microcontroller. When the measured voltage exceeds a limit, which is equivalent to a current through R, greater than $35\ \mu A$ in positive and negative direction, a flag is set. This is done by the firmware. The circuit for the current measurement is explained in detail in the next section.

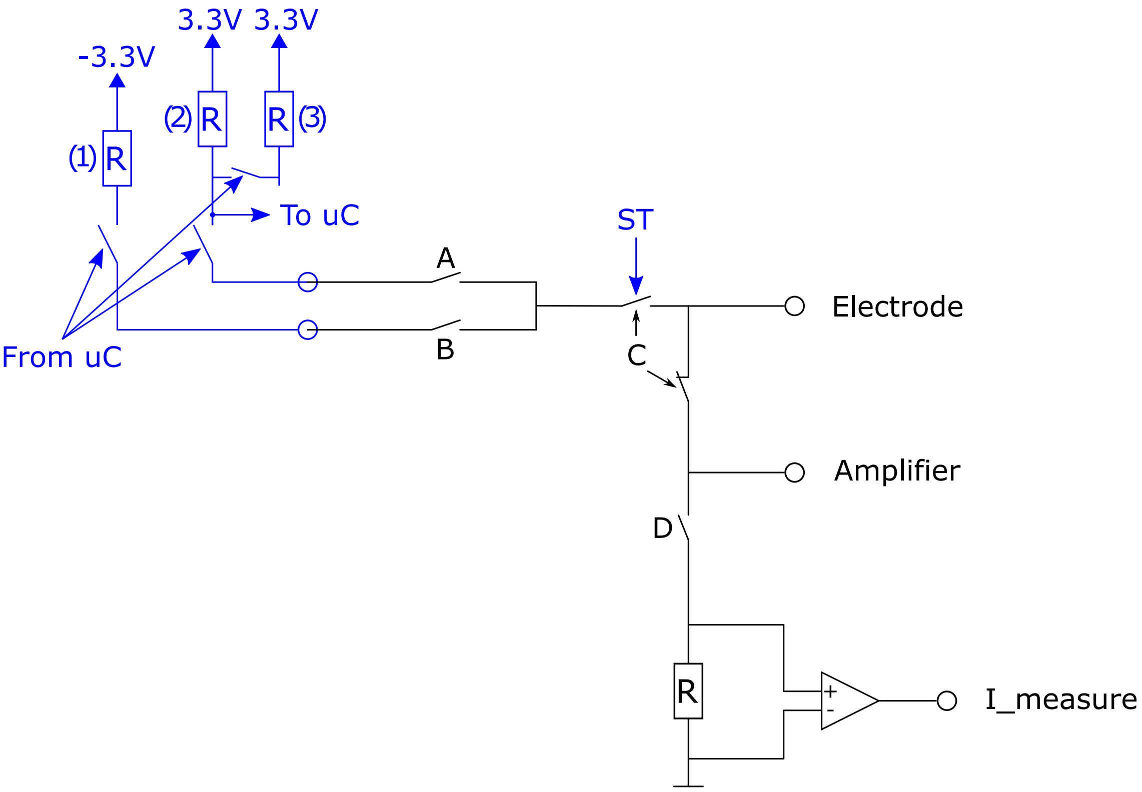


Figure 2.11.: The test circuit is shown in blue. The switch interconnection of the module and the current measurement is black.

Stage I, II and III describe the single stages during switch test of a single channel which is repeated for every channel.

Stage I) Check switch functionality A, C1, C2 and D

- (2) is switched to STIM+ channel
- A, C1, C2 and D are closed
- The microcontroller input detects a digital low
- Then every switch in the row is opened and closed one by one
- Every time a switch is opened or closed the microcontroller recognizes an edge and thus checks the switch functionality

Stage II) Check overcurrent detection, while positive current flow

- (2) is switched to STIM+ channel.
- A, C1, C2 and D are closed
- The microcontroller input detects a digital low
- (3) is switched
- The analog microcontroller input $I_{Measure}$ detects if the current is as expected, otherwise the test is failed.

Stage III) Check overcurrent detection and switch B, while negative current flow

- (1) is switched to STIM- channel.
- C1, C2 and D closed
- Measured current via $I_{Measure}$ should be within the limits.
- B is closed
- If the microcontroller analog input $I_{Measure}$ measures a current greater than the defined limit, the test is passed.

2.1.2.5. Current measurement circuit

In this section the calculation of the circuit and the main function is described. Figure 2.12 shows the schematic and the equations below the calculation of the current.

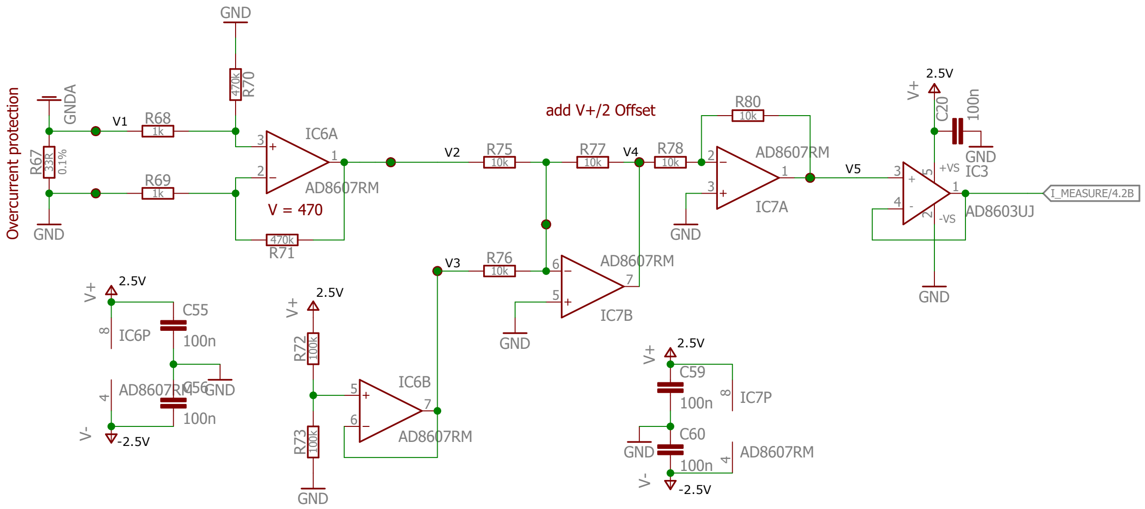


Figure 2.12.: Schematic of the current measurement circuit

$$V_1 = R_{shunt} \cdot I = 33\ \Omega * 35\ \mu A = 1,16\ mV \tag{2.1}$$

$$V_2 = U_1 \cdot V \quad with \quad V = \frac{470\ k\Omega}{1\ k\Omega} = 470 => V_2 = 1,16\ mV \cdot 470 = 543\ mV \tag{2.2}$$

$$V_3 = \frac{V_+}{2} = 1,25\ V \tag{2.3}$$

$$V_4 = -(V_2 + V_3) = -(543\ mV + 1,25\ V) = -1,793\ V \tag{2.4}$$

$$I_{Measure} = V_5 = -V_4 = 1,793\ V \tag{2.5}$$

2.1.2.5.1. Simulation:

The simulated circuit is shown in figure 2.13.
The measured points are marked as V1 to V5.

The simulated voltage are:

- $V_1 = 1,155\ mV$
- $V_2 = 534,856\ mV$
- $V_3 = 1,25\ V$
- $V_4 = -1,785\ V$
- $V_5 = 1,785\ V$
- $V_{Out} = 1,785\ V$

The simulated voltages have only a small deviation from the calculated ones, due to the operational amplifier's tolerances and variance which is taken into account during the simulation.

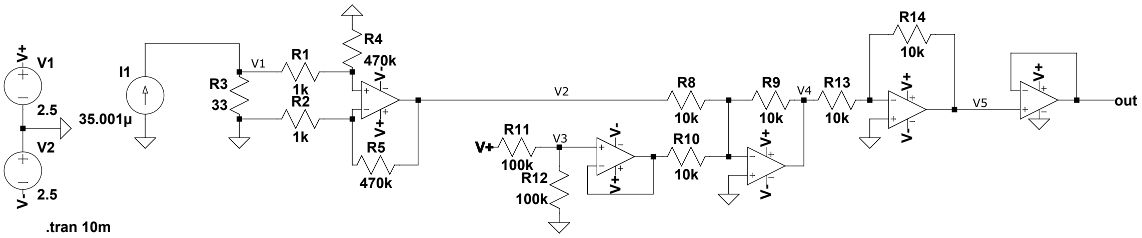


Figure 2.13.: Current measurement circuit for the simulation drawn in LTSpice

2.1.2.6. Signal test unit

There are two ways implemented in the hardware to generate a sine wave. The first variant is to use a pulse width modulation (PWM), so that a sine can be generated with a filter and opamp circuit. The second way is to use a DAC to generate a stepped sine, which is then filtered. Since the DAC output and PWM are unipolar between 0 and 3.3 V, the signal must be scaled to obtain a bipolar signal in the desired range. Both variants are described in the following sections and in the test device both are usable. For the used test device, the DAC variant was chosen due to the good experience already gained in the company at other test devices and due to the higher accuracy.

2.1.2.6.1. PWM

The PWM signal is filtered through a lowpass, scaled to +-2.5 V and then divided to get a sinusoidal signal in a +-200 mV range. Figure 2.14 shows the simulated circuit. The required PWM signal is shown in figure 2.15, the simulated voltages in figure 2.16 and the final circuit in figure 2.17.

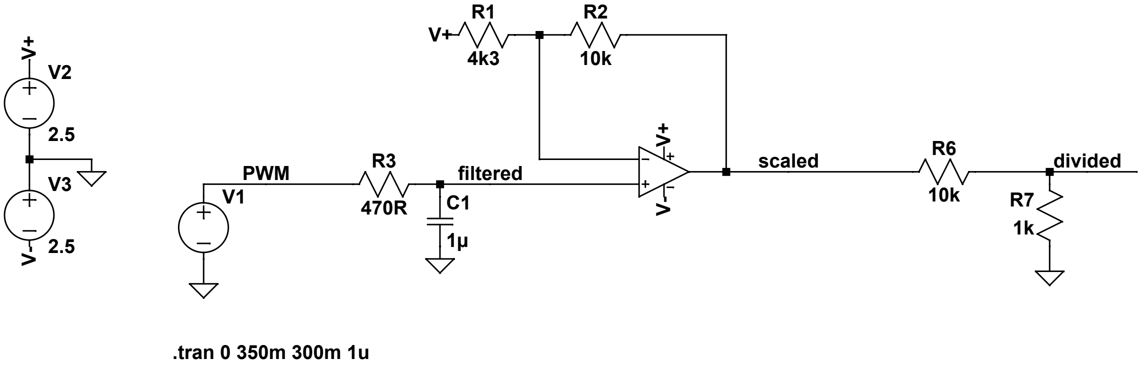


Figure 2.14.: PWM circuit drawn in LTSpice

The modulated PWM signal is applied in the simulation with piecewise linear (PWL) functions for voltage and current sources, which are used to generate custom waveforms through a series of straight line segments. The PWL list of the voltage source looks like this:

```
repeat forever
0 0 1u 3.3 74.9u 3.3 75u 0 99.9u 0 100u 3.3 192.9u 3.3 193u
0 199.9u 0 200u 3.3 300u 3.3 392.9u 3.3 393u 0 399.9u 0 400u
3.3 474.9u 3.3 475u 0 499.9u 0 500u 3.3 549.9u 3.3 550u 0
599.9u 0 600u 3.3 624.9u 3.3 625u 0 699.9u 0 700u 3.3 706.9u
3.3 707u 0 899.9u 0 900u 3.3 906.9u 3.3 907u 0 999.9u 0 1000u
3.3 1024.9u 3.3 1025u 0 1099.9u 0 1100u 3.3 1149.9u 3.3 1150u
0 1199.9u 0 1200u 3.3 1274.9u 3.3 1275u 0
endrepeat
```

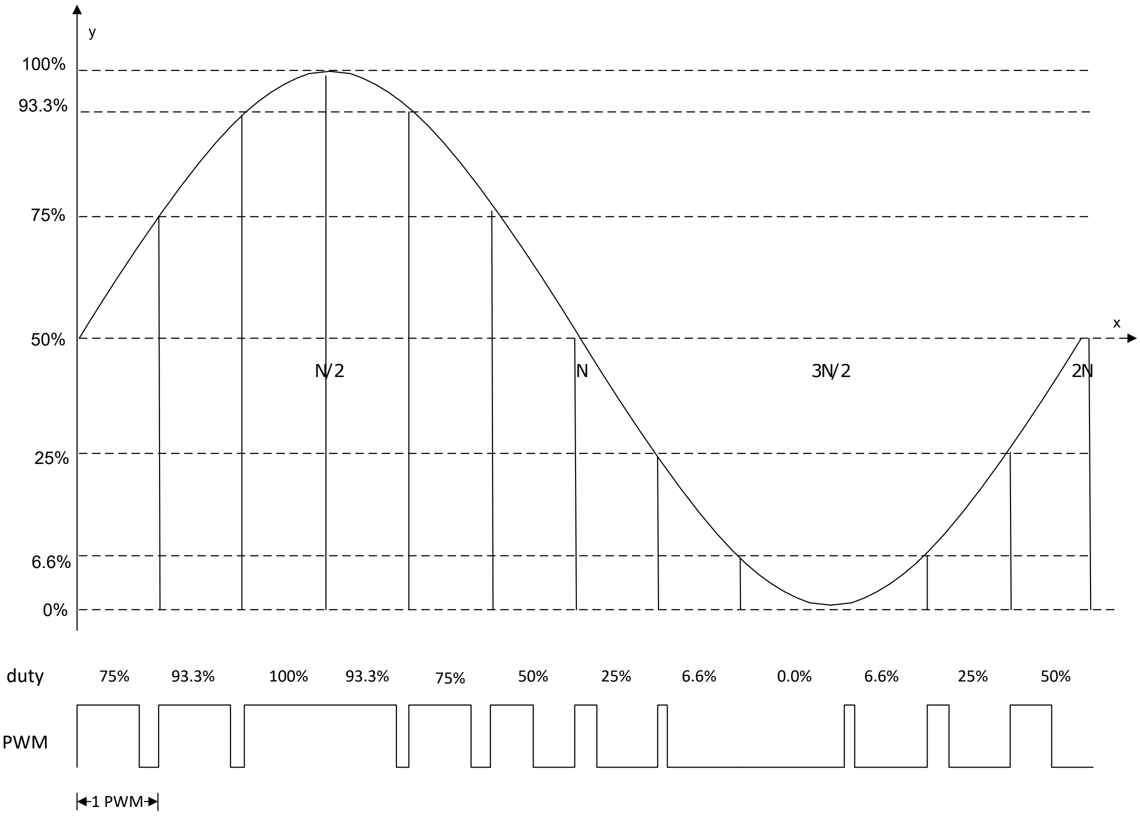


Figure 2.15.: Sample points of a sine wave with the corresponding PWM signal. [16]

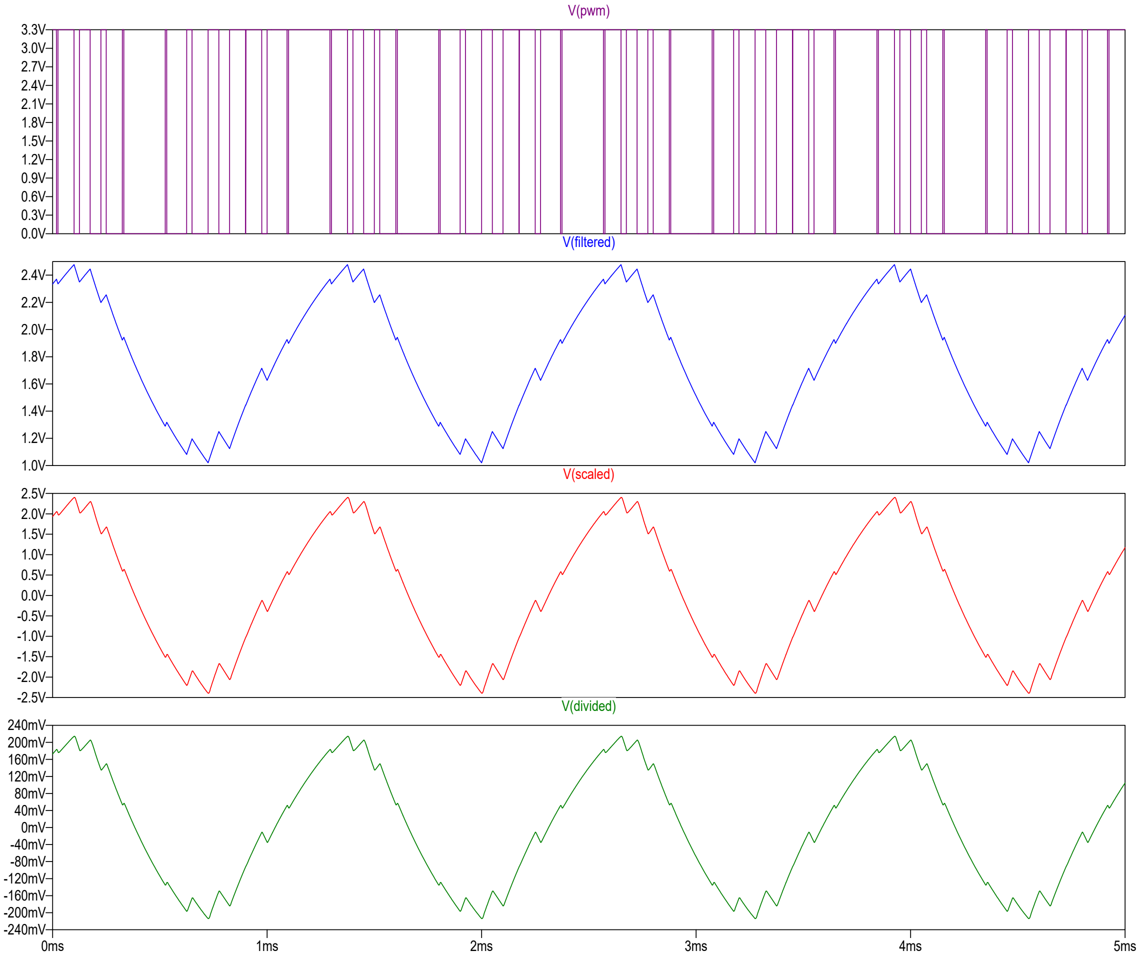


Figure 2.16.: Simulation of the PWM circuit. V(pwm) is the PWM signal from the microcontroller. V(filtered) is after the low-pass filter. It is a sine with a maximum at 2.5 V to match the input range of the opamp. V(scaled) is after the opamp circuit which is scaling the signal to the supply range of +-2.5 V. V(divided) is after the voltage divider to get a sinusoidal signal in the range of +-200 mV.

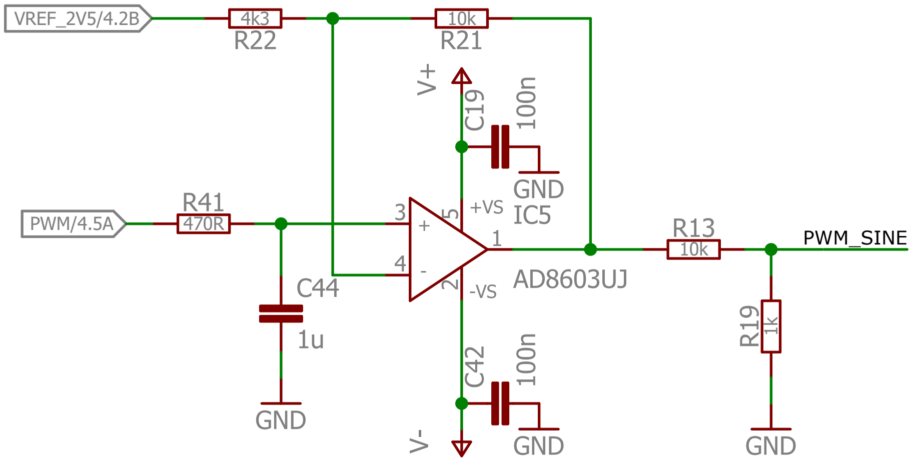


Figure 2.17.: Schematic of the final PWM circuit.

2.1.2.6.2. DAC

For the DAC variant an AD5621 from Analog Devices is used. It is a single 12-bit, buffered voltage output DAC that is operating via a serial peripheral interface (SPI). It was chosen since it is also being used in the company and therefore the know-how exists and it is easy to get the part. In figure 2.18 the schematic is shown. The DAC generates an output voltage adjustable over SPI from the microcontroller and then, because it is in the range from 0 to 3.3 V, it is scaled to +-2.5 V by the following opamp circuit. After that it is filtered with a low-pass to smoothen the signal and finally get a sine at the output. With the voltage at the output of the DAC, the range of the sine at the output of the circuit, can be adjusted via software. Figure 2.19 shows the simulated circuit, drawn in LTSpice and figure 2.20 the simulated voltages at the different points.

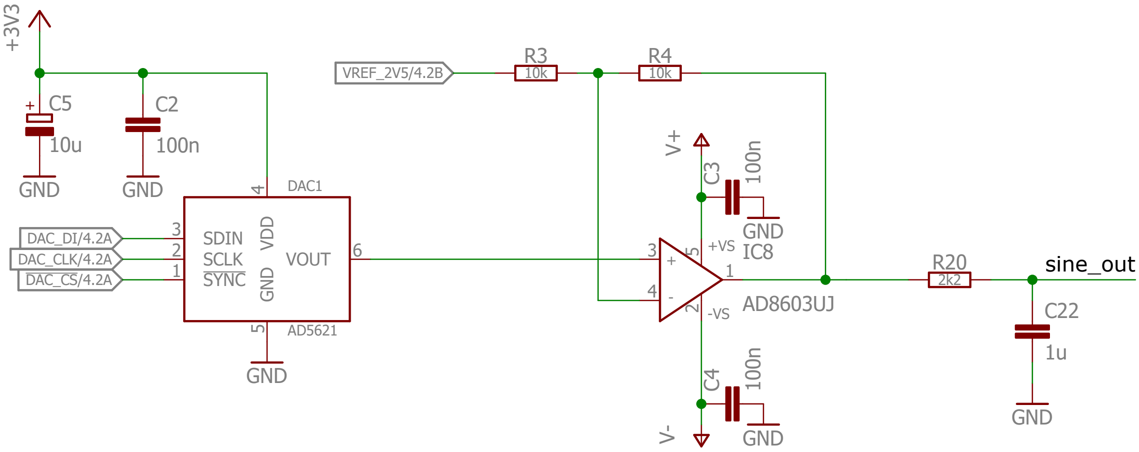


Figure 2.18.: Schematic of the DAC circuit. The DAC is connected to SPI 1 of the microcontroller. The signal is converted to a bipolar signal and filtered with a low-pass.

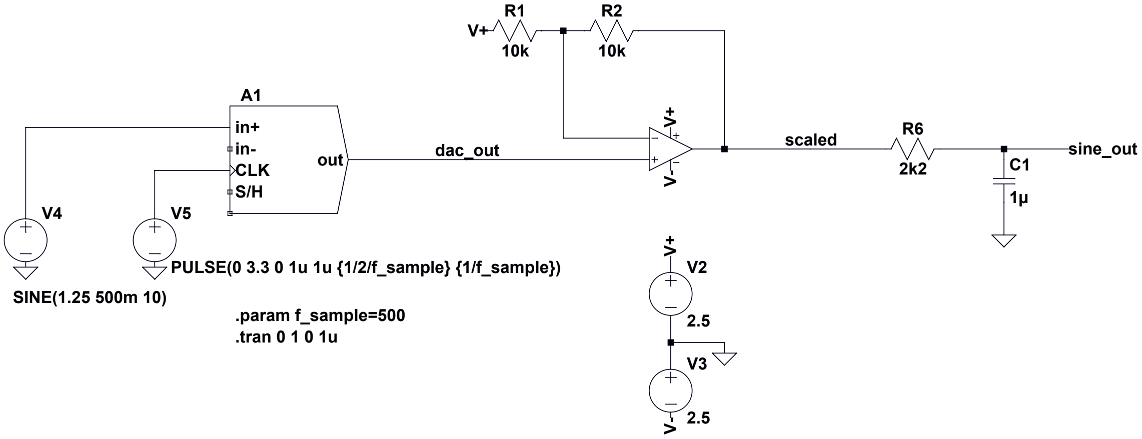


Figure 2.19.: DAC circuit drawn in LTSpice. The DAC output is a stepped sinusoidal signal with a frequency of 10 Hz, an offset of 1.25 V and an amplitude of +-0.5 V. It is sampled with a frequency of 500 Hz

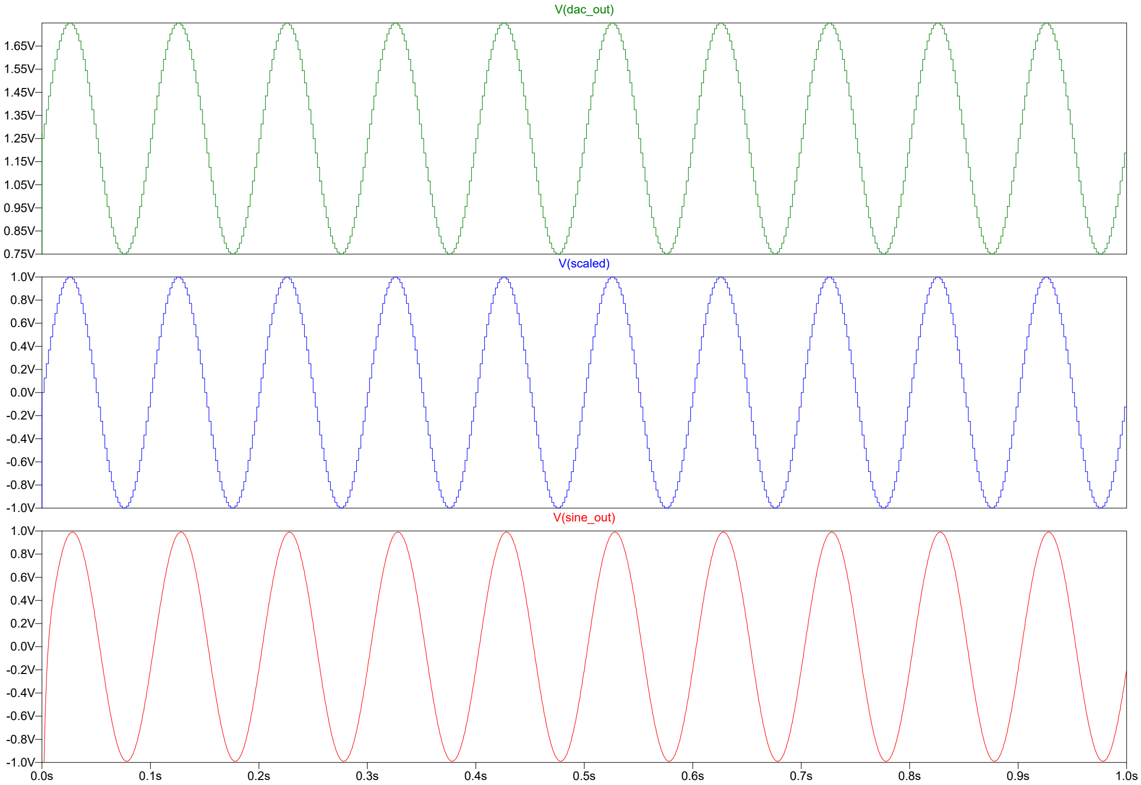


Figure 2.20.: Simulation of the DAC circuit. V(dac_out) is the output of the DAC. The single steps are visible, the sine is a 10Hz sine with an amplitude of +-0.5 V and an offset of 1.25 V. V(scaled) is after the opamp circuit, which scales the DAC output signal to +-2.5 V and also double the amplitude of the signal. So it is in the range of +-1 V in this example. Later in the test device the range will be +-200 mV, so the DAC output voltage is +-100 mV with an offset of 1.25 V

In figure 2.21 and figure 2.22 below, the waveform of the signal measured with the oscilloscope is shown.

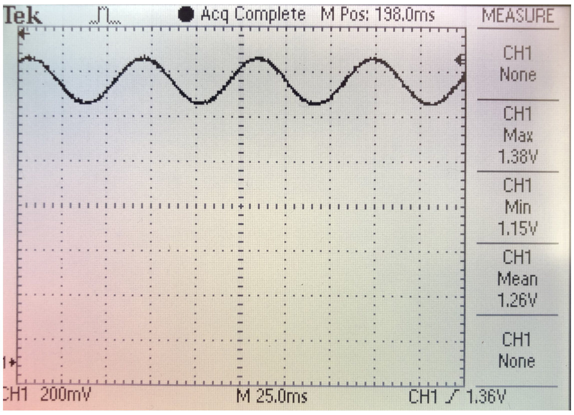


Figure 2.21.: The voltage measured at the output of the DAC has an offset of 1.26 V and an amplitude of +-100 mV. The frequency is approximately 15 Hz.

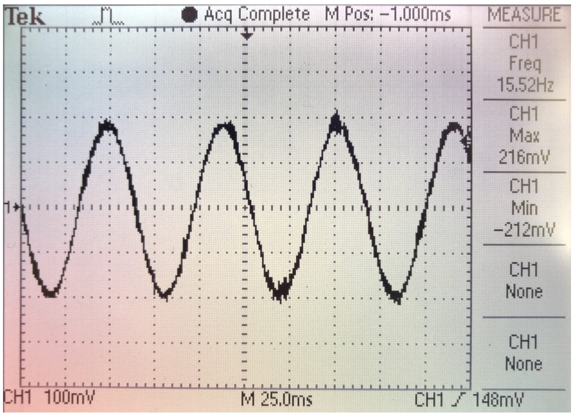


Figure 2.22.: After the opamp circuit the signal is a sine with double the amplitude to +-200 mV and also the same frequency but with no offset so it is now a bipolar sinusoidal signal.

2.1.2.7. Unipolar to bipolar converter

The test signal is a bipolar sinusoidal signal in the range of +-200 mV. So an offset has to be added to the signal to measure it with the ADC. With the circuit used the signal is shifted by 1.25 V, so it is in the range 1.05 V to 1.45 V and therefore able to be measured without damaging the ADC input. In the software the offset could be removed, if the original signal is needed. In figure 2.23 the final schematic is shown. In this example SINE_OUT is the sinusoidal signal from figure 2.22 with +-200 mV. The signal is then shifted by an offset of 1.25 V (see equation 2.6 and figure 2.24).

$$Offset = VREF\_2V5 \cdot \frac{R38}{R37 + R38} = 2.5V \cdot \frac{100k\Omega}{100k\Omega + 100k\Omega} = 1.25V \qquad (2.6)$$

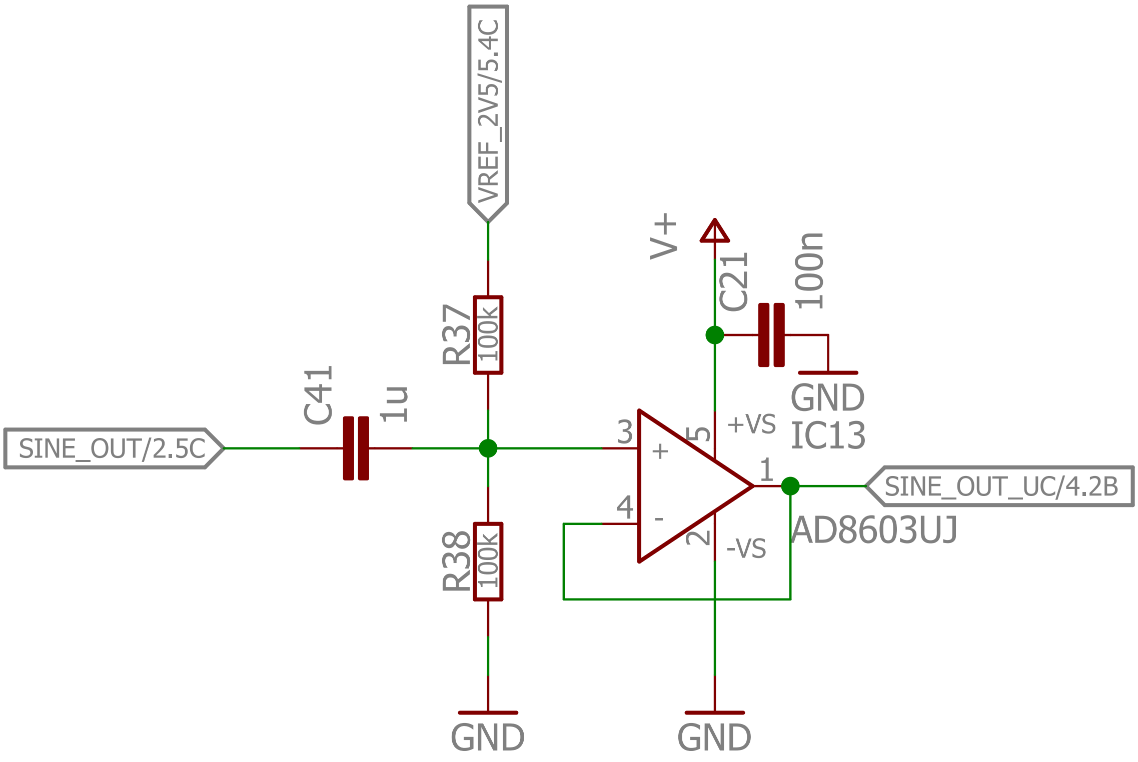


Figure 2.23.: Schematic of the unipolar to bipolar converter

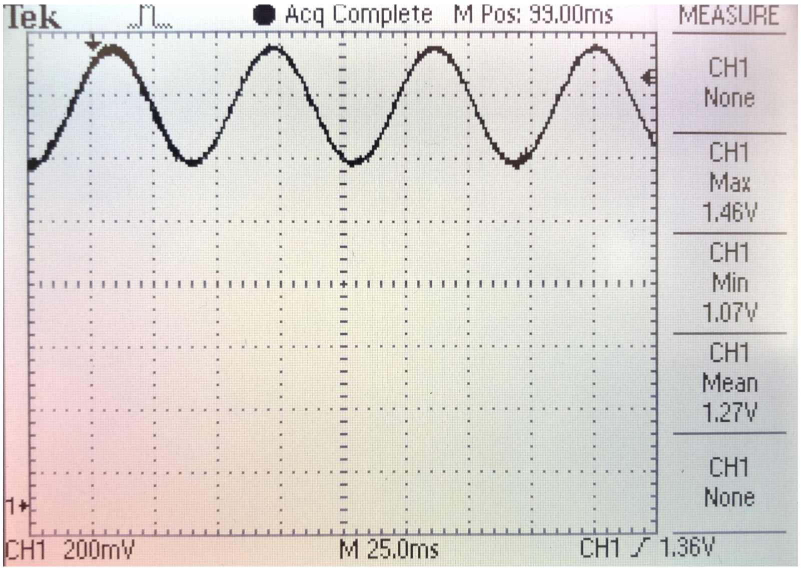


Figure 2.24.: The shifted signal after the convertion is unipolar and has an offset of 1.27 V, with an amplitude of +-200 mV.

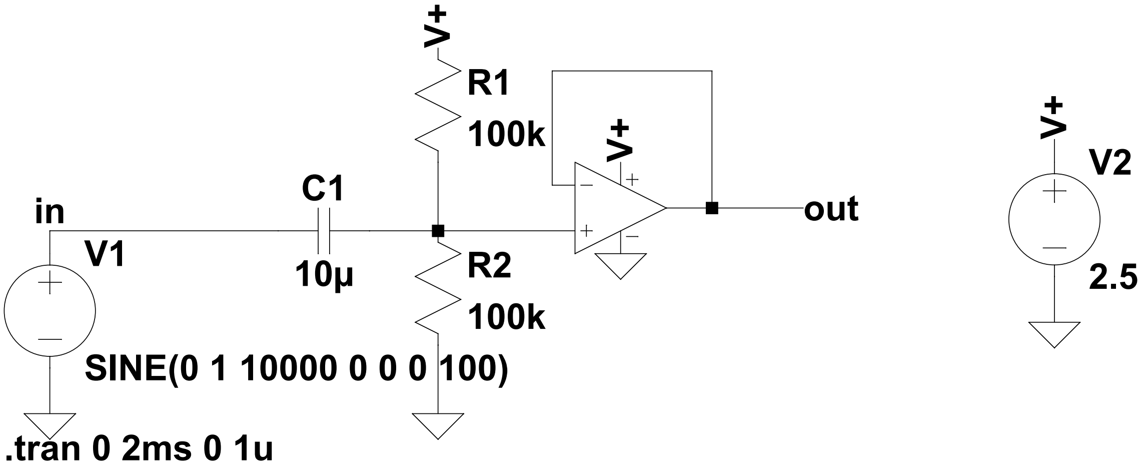


Figure 2.25.: The simulated circuit drawn in LTSpice

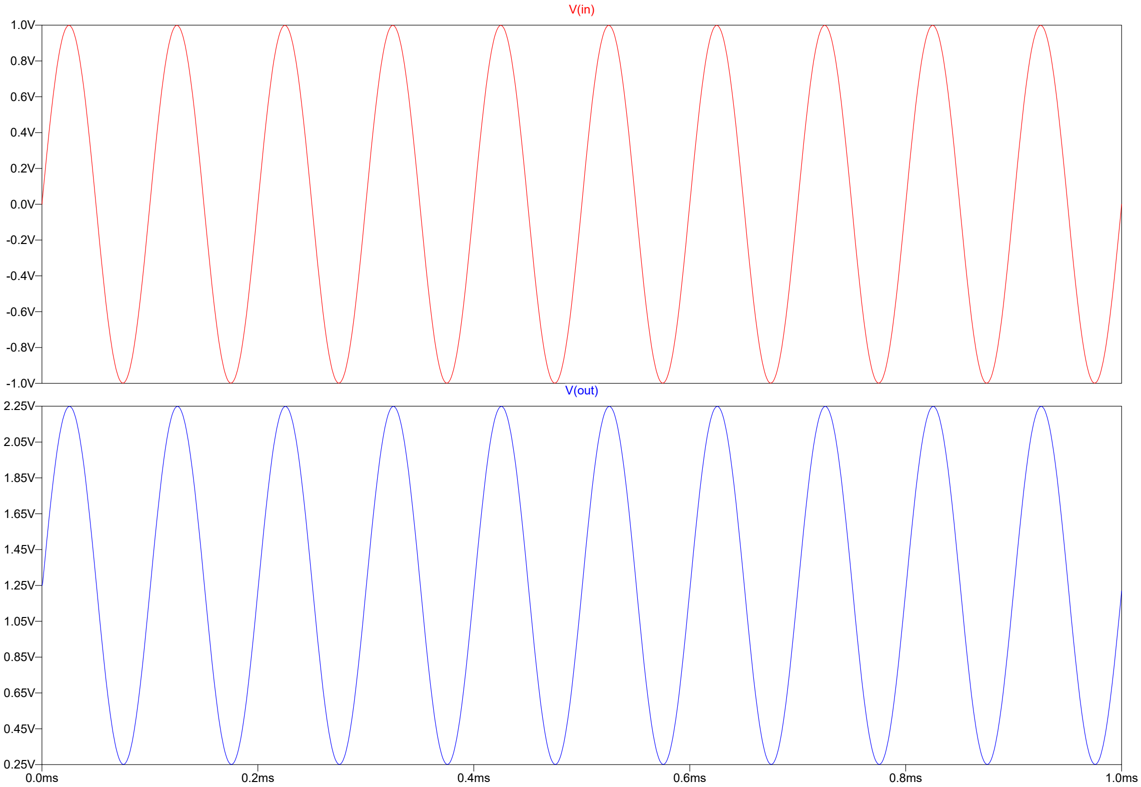
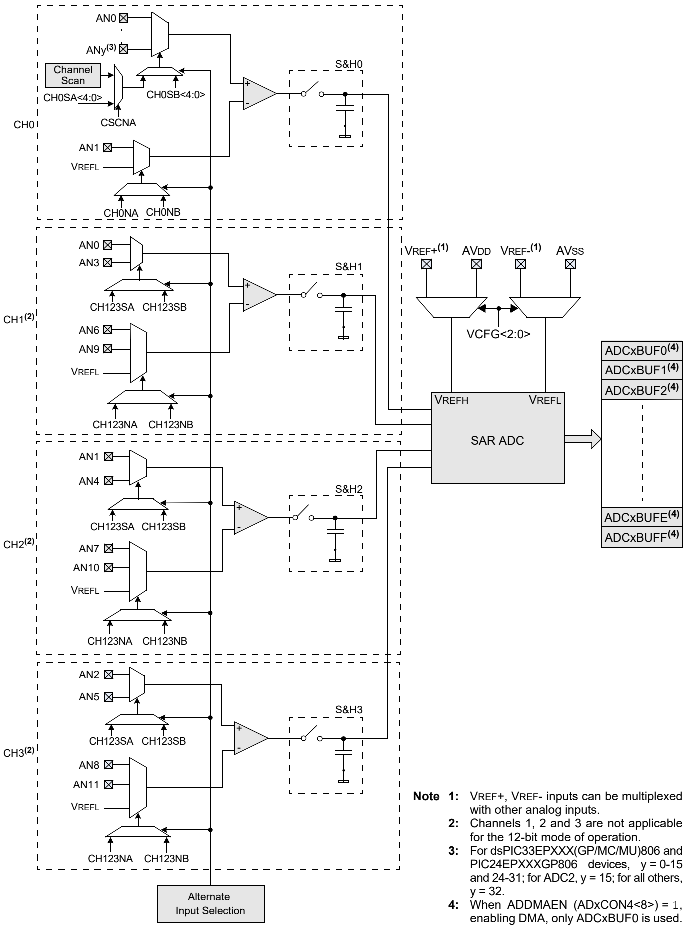


Figure 2.26.: The input voltage (V(in)) in the simulation is a bipolar signal with an amplitude of +-1 V and a frequency of 10 kHz. After the conversion it is unipolar, with the same amplitude, but with an offset of 1.25 V.

2.1.2.8. Analog to digital converter (ADC)

To measure the signals the internal ADC of the microcontroller is used. Via cross correlation the similarity of the input and the output signal is arbitrated, to see if the signal is going through the circuit without being changed. The result is that the buffer is working in the intended way. The internal ADC of the microcontroller dsPIC33EP256MU806 is a 10-/12-bit successive approximation (SAR) converter with a conversion speed up to 1.1 MS/s and up to 32 analog input pins. It uses the supply voltage of the microcontroller as a voltage reference, but it can also use an external voltage reference. For the test device a 2.5 V voltage reference is used, because the maximum signal range of the module is in the same range. In figure 2.27 the block diagram is shown.



Note 1: VREF+, VREF- inputs can be multiplexed with other analog inputs.

2: Channels 1, 2 and 3 are not applicable for the 12-bit mode of operation.

3: For dsPIC33EPXXX(GP/MC/MU)806 and PIC24EPXXXGP806 devices, y = 0-15 and 24-31; for ADC2, y = 15; for all others, y = 32.

4: When ADDMAEN (ADxCON4<8>) = 1, enabling DMA, only ADCxBUF0 is used.

Figure 2.27.: dsPIC33EP256MU806 ADC block diagram [12]

2.1.2.9. Cross correlation [29]

The correlation coefficient of two signals is a measure for similarity of those two. The correlation coefficient is defined as

$$\rho(A,B) = \frac{1}{N-1} \cdot \sum_{i=1}^{N} (\frac{A_i - \mu_A}{\sigma_A})(\frac{B_i - \mu_B)}{\sigma_B} \tag{2.7}$$

where μ_A is the mean value and σ_A the standard deviation of A and vice versa for B. An other representation is in terms of the covariance of A and B.

$$\rho(A,B) = \frac{cov(A,B)}{\sigma_A \cdot \sigma_B} \tag{2.8}$$

The correlation coefficient matrix of two signals is defined as

$$R = \begin{pmatrix} \rho(A,A) & \rho(A,B) \\ \rho(B,A) & \rho(B,B) \end{pmatrix} \tag{2.9}$$

Since $\rho(A,A) = 1$ and $\rho(B,B) = 1$, because they are always directly correlated with themselves, the matrix can be written as

$$R = \begin{pmatrix} 1 & \rho(A,B) \\ \rho(B,A) & 1 \end{pmatrix} \tag{2.10}$$

2.1.2.10. Temperature measuring [13]

For the temperature acquisition a Microchip MCP9700 is used. It is a low-power linear active thermistor IC with an accuracy of +-2 °C. The temperature is calculated with the following equation.

$$Temp = \frac{V_{out} - 500mV}{10} \tag{2.11}$$

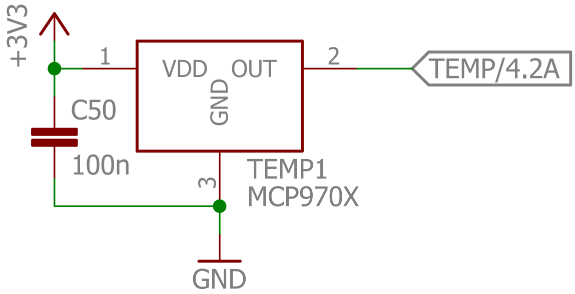


Figure 2.28.: Temperature measuring with MCP9700 from Microchip

2.1.2.11. Connector

The connector used for the test adapter is predefined as it must mate with the connector of the module. Since the module connector is the ERM8-025-01-L-D-EM2-TR from Samtec, for the test adapter the connector ERF8-025-05.0-L-DV-L from Samtec was chosen (see figure 2.29).

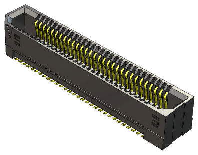


Figure 2.29.: 0.80 mm Edge Rate® Rugged High-Speed Socket Strip from Samtec [27]

2.1.2.12. Housing

Figure 2.30 shows the raw housing and figure 2.31 shows the dimensions of the test device. This enclosure has been chosen, because it is ideal in size and also used for an other product in the company. The front and the back panel are not included and hand-made with acrylic glass. The cut-out for the LEDs, the switch and the USB cable are made with a laser-cutter which are designed with a computer-aided design (CAD) tool.

For the printed circuit board (PCB) the dimensions of the housing were imported to the software *EAGLE (Autodesk, Inc., San Rafael, United States of America (USA))* and the PCB outline was adjusted to it. The finished PCB was ordered from the manufacturer *Eurocircuits (Eurocircuits GmbH, Kettenhausen, Germany)*.

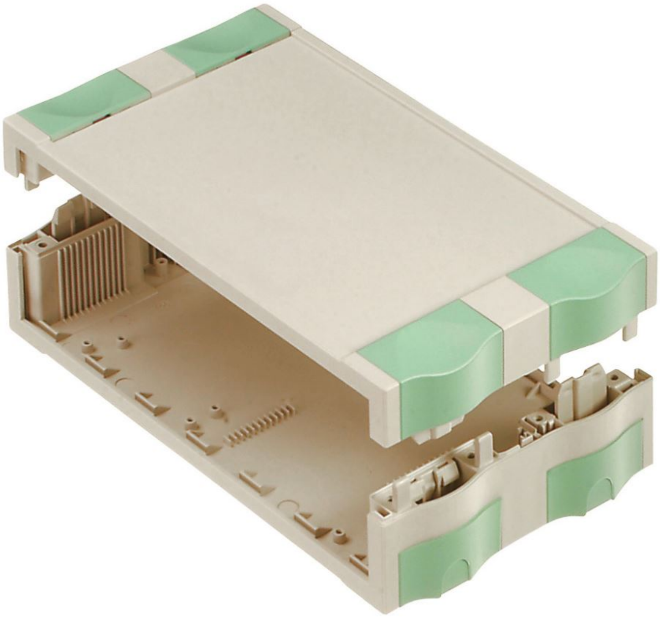


Figure 2.30.: Raw Bopla Botego 51412 enclosure [7]

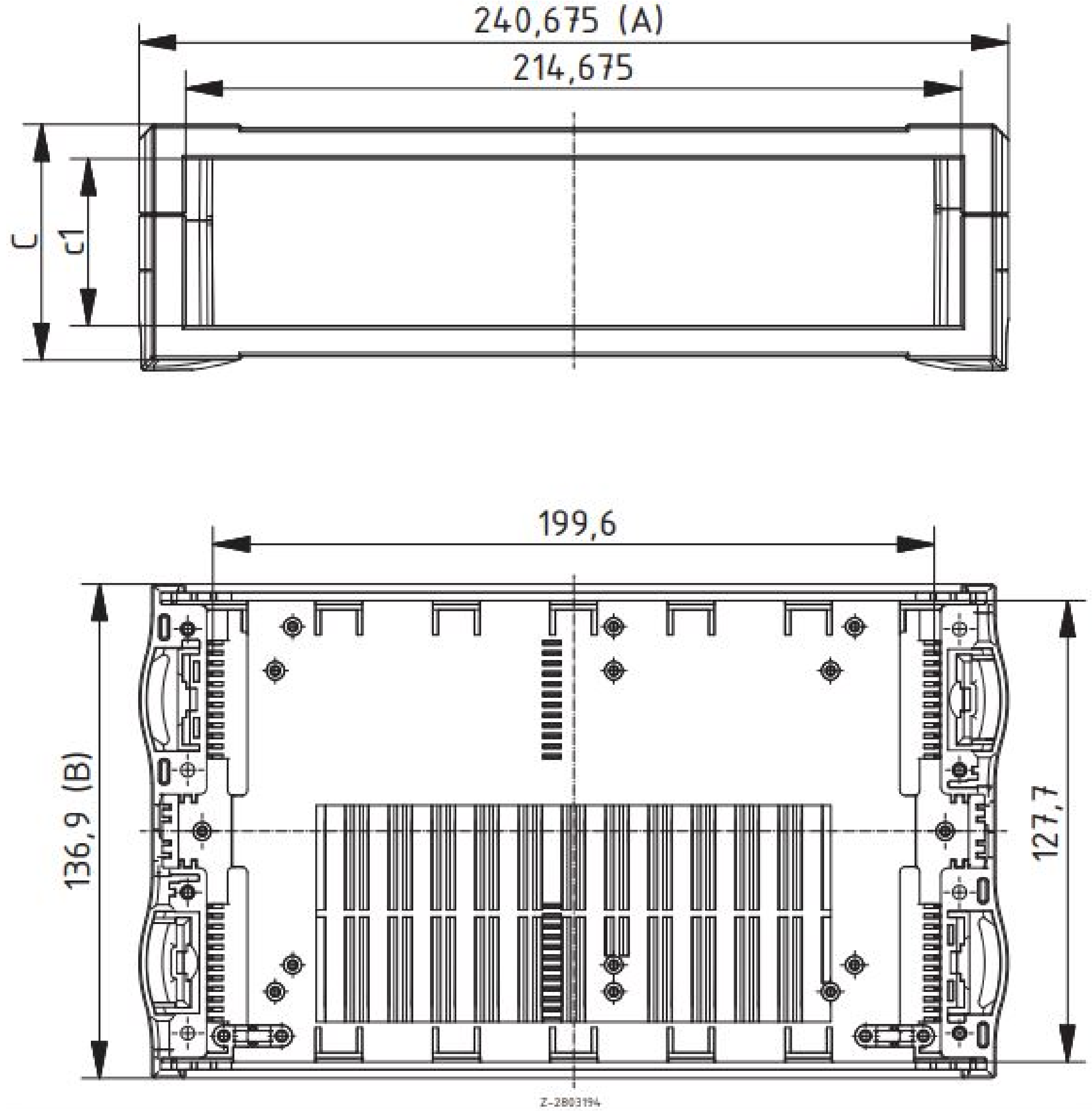


Figure 2.31.: Dimensions of the Bopla Botego 51412 enclosure [7]

2.2. Software

In this chapter the firmware, PC software, graphical user interface (GUI) and the communication concept between microcontroller and PC are described as the parts of the software.

2.2.1. Concept

In this block diagram, the interaction of the software units is shown. The software is running on a PC and communicates with the firmware on the microcontroller. The microcontroller is responsible for controlling and measuring the DUT. The software is controlled by the GUI, which is a part of the software and programmed in Python. The firmware is programmed in C. The report is created with a LATEX template and then converted to a .pdf file via pdflatex.

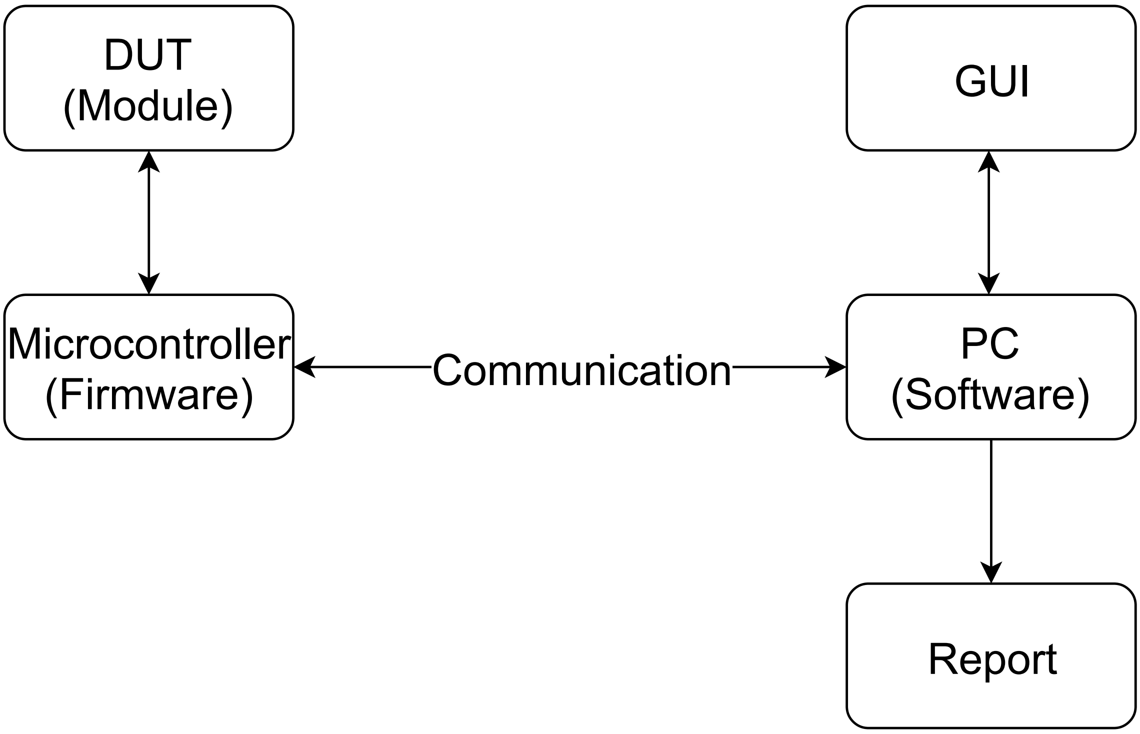


Figure 2.32.: Block diagram of the software concept

The communication is done via UART and an FTDI chip which converts the USB signal to a UART signal. For that the related application programming interface (API) for Python is used. This API uses the D2XX driver of the chip manufacturer. [19] The structure is like in figure 2.33. The first byte is the command, followed by a value which is also one byte long. After the two bytes are received an acknowledge byte (ACK) is sent, to confirm that the communication was successful.

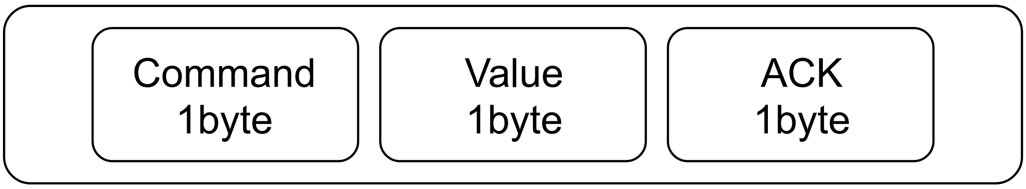


Figure 2.33.: Data package structure for the communication between the PC and the microcontroller

2.2.2. Firmware

In this chapter the firmware with its functions and structure is described in more detail.

2.2.3. Concept

The concept of the firmware in a basic block diagram is represented in figure 2.34. The firmware is written in the programming language C. The core part is the main function. To access the hardware functions a hardware abstraction layer (HAL) is used. It provides the necessary functions to access the microcontroller modules like SPI, input/output pins, the ADC and I2C. The HAL was provided by the company, since it has been used in many other projects with the same microcontroller. The main function also mediates between the PC and the microcontroller and is responsible for the communication and the execution of the corresponding test procedures, when receiving the respective commands.

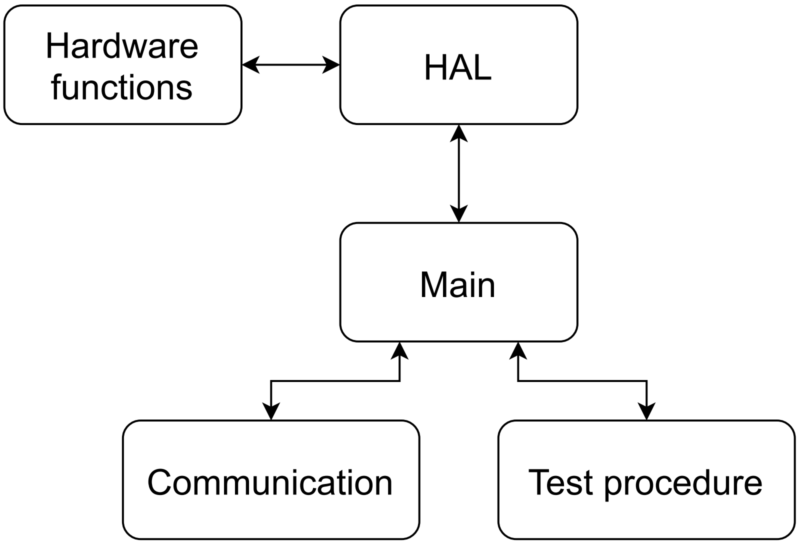


Figure 2.34.: Block diagram of the firmware

2.2.3.1. Functions

Following, the single test procedures of the firmware and later the interaction between the main program and the functions, according to the received commands are described.

2.2.3.1.1. Test A: Switches

The function for "Test A: Switches" is named *startSwitchTest()*. When this function is called, a struct for the communication via I2C is initialized. A data buffer with two bytes length is used. Then the test procedure is performed for each channel, like described in chapter 2.1.2.4 (see appendix A.1 for code examples).

2.2.3.1.2. Test B: Crosstalk

The function for "Test B: Crosstalk" will be named *startCrosstalkTest()* which is not implemented yet, because it has not been specified by the company yet.

2.2.3.1.3. Test C: Signal

The function for "Test C: Signal" is named *startSignalTest()*. When the function is called, a struct for the communication via I2C is initialized. For the data a buffer with two byte in size is used. The test procedure is performed for each channel one after another. In the test procedure, first the test will be activated, then the channel configuration is sent to the module. Finally the function *sendSineWave(sineLUT64hex, 10)* is called. For this function a parameter in form of a look up table is given and the duration in means of periods. The table is generated with a tool from Daycounter, Inc. [4]. It consists of 64 values that are transmitted to the DAC one after another, resulting in a stepped sine wave at the DAC output. After transmitting one value, the ADC channels are read in and then the next value is transmitted. This happens in a for-loop till all values have been sent. This loop will then be repeated, for the number of periods given as the parameter to the function. The time of 1 ms to wait between the values was determined empirically, to get a sine between 10 and 20 Hz. The read values of the ADC are saved into the prepared result array, which is sent to the PC via UART making the data available to the software for generating the report and further analyzing (see appendix A.1 for some code examples).

2.2.3.2. The main program

The main program is reponsible for the communication and the pertinent functions are called with the required parameters. The microcontroller boots into the *main()* function after powering up and returns to this function after completing a test. The flow chart is plotted in figure 2.35. After initializing the HAL and the relvant variables the program enters a while loop and waits for an interrupt, from the UART RX module of the controller. If data are received via UART, the *uart_rx_flag* is set, and the data are saved to the *uart_buffer[]*. *uart_buffer[0]* is the command and *uart_buffer[1]* the value. Then the respective function to the received command is called with an if - else if - else statement.

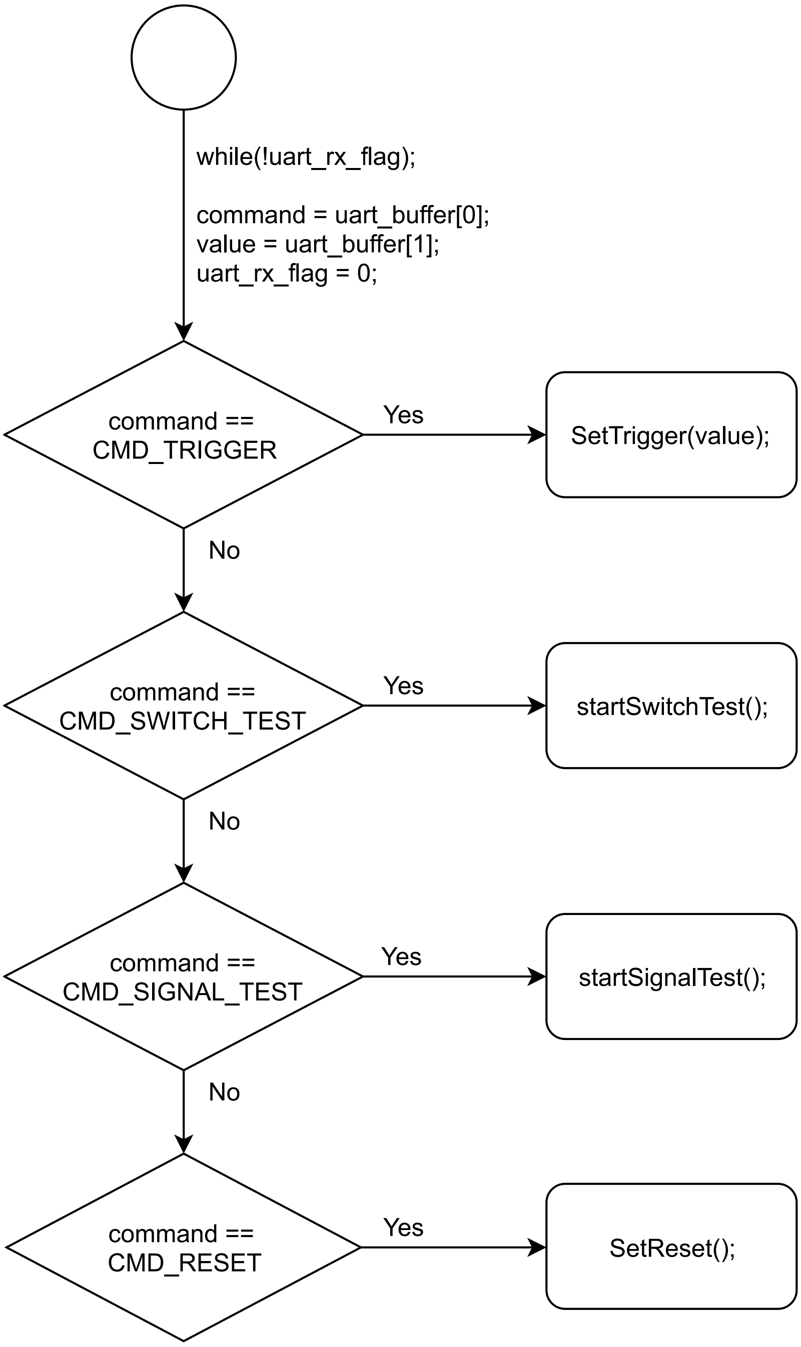


Figure 2.35.: Flow chart of the main loop

2.2.4. PC Software

In this chapter the API, GUI, the single functions and usability of the PC software are explained in more detail. The software is programmed in Python, an open source development environment with many modules and packages available for different applications. With the package PyQt5, which wraps the C++ Qt library [2], a GUI can be created. The syntax is easy to learn, if the user has experience in other programming languages and software development. Python also has the possibility of object-orientated programming, which is an other advantage.

2.2.5. Concept

The concept of the software is shown in the following block diagram (figure 2.36).

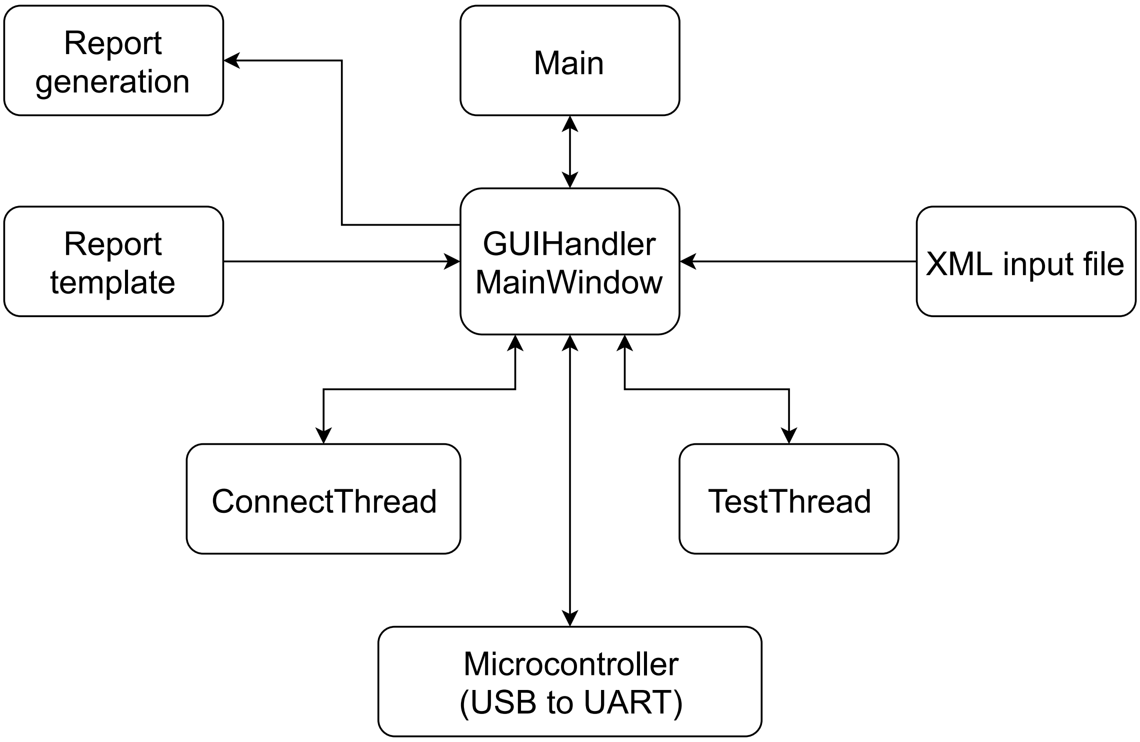


Figure 2.36.: Block diagram of the PC software

- **main**

  The program always starts and ends in the main function. It generates the log-file, and an object of the main window that is then opened. The class for this window is located in a separate file. With the following command the .py file is compiled from a .ui file, which is created with the Qt Designer software. This software is included in the PyQt5 package, which is described later.

  ```
  C:/path.../pyuic5 -x C:/path.../gui.ui -o C:/path.../gui.py
  ```

  The created file is imported into the GUIHandler.

- **GUIHandler - MainWindow**

  The GUI was designed with the Qt Designer software and then converted to a Python file which then is included into the GUIHandler. With this class an object is created:

  ```python
  from PyQt5 import QtCore, QtWidgets
  from GUI.gui import Ui_MainWindow

  class MainWindow(QtWidgets.QMainWindow, Ui_MainWindow):
     def __init__(self, parent=None):
        super(MainWindow, self).__init__(parent)
        self.setupUi(self)

  def main():
     app = QtWidgets.QApplication(sys.argv)
     window = MainWindow()
     window.show()
     sys.exit(app.exec_())
  ```

- **Microcontroller**
  The microcontroller receives the software commands sent by the GUIHandler.

- **XML input file**
  The extensible markup language (XML) file, named *TestConfiguration.xml*, is used as configuration input at every start of the program. In this file, the important information for the report and the GUI are specified, like the list of the test engineers, the information about the manufacturer and the limit for the tests.

```xml
<Tables>
   <ManufacturerInfo>
      <Manufacturer Name="NAME" />
      <Street Name="STREET" />
      <City Name="CITY" />
   </ManufacturerInfo>
  <Engineer>
      <TestEngineer Name="User1" />
      <TestEngineer Name="User2" />
      <TestEngineer Name="User3" />
  </Engineer>
  <Tests>
    <Device Name="A">
       <A>
         <AdapterInfo Name="Serial" Value="XX-01234" />
         <AdapterInfo Name="Version" Value="0.1" />
       </A>
    </Device>
    <Device Name="T">
       <T>
         <TestStep Limit="-" Meas="Switches" Name="A" />
       </T>
    </Device>
  </Tests>
</Tables>
```

- **ConnectThread**
  For multithreading, the class QThread is used. It is imported in the GUI-Handler via the PyQt5 package. After opening the GUI, the *ConnectThread* is started and the software searches for the test device. If it is found, an object for the microcontroller is created, to access the API functions. Next the UART interface is configured. The baud rate is set to 38400 Bd, the data characteristics to 8-bit with no stop bit and no parity bit and the timeouts for read and write to 1 s.

```python
    uc = ftdi.open(dev_index) # open uc
    self.uc.setBaudRate(38400) # set baud rate
    self.uc.setDataCharacteristics(8, 0, 0) # 8 bit no parity / stop bit
    self.uc.setTimeouts(1000, 1000) # 1s timeout read/write
    self.DevOpened = True # device opened flag
```

- **TestThread**
  The *TestThread* is executed when the user presses the start button in the GUI. For each test, a seperate function is called. The output is written in the *cmdOutput QListWidget* in the GUI. For each test, the start and stop time and the duration time, plus the total duration time is recorded. After the tests are done, a result dictionary is created, which is then handed over to the function for generating the report. The calculation of the cross-correlation coefficients is also done in the *TestThread*.
  Example Python code for the calculation of the correlation coefficients:

```python
import numpy as np
def calculateCorrcoef(self, data1, data2):
    # Test state initial value
    TestC_state = "PASS"
    # initialize arrays for calculation and results
    # Result[0]: channel number
    # Result[1]: correlation coefficient
    # Result[2]: OK or NOK
    # Result[3]: if NOK -> channel number
    Result = [[], [], [], []]
    R = np.zeros((16, 1))
    C = np.zeros((2, 2))
    # get limit
    limit = self.xml_test[2][2]
    # calculate coefficients for each channel and append results to the Result array
    for msr_idx in range(0, 16):
        Result[0].append(str(msr_idx + 1))
        C = np.corrcoef(data1[msr_idx, :], data2[msr_idx, :])
        R[msr_idx, 0] = "%.4f" % C[0, 1]
        if C[0, 1] > float(limit):
            Result[1].append(str(R[msr_idx, 0]))
            Result[2].append("OK")
        else:
            Result[1].append(str(R[msr_idx, 0]))
            Result[2].append("NOK")
            Result[3].append(str(msr_idx + 1))
            TestC_state = "FAIL"
    return Result, TestC_state
```

- **Report template and generation**
  The report is generated with a LATEX template and after the tests are done, the software copies it to the root folder. The .tex file is generated with the package jinja2 and the result dictionary as data input. The variables in the template are replaced with the values of the keys, named like the variables of the dictionary. The latex command *pdflatex* is used afterwards, to convert the .tex to a .pdf file. Finally, all created files are moved in a new folder, named after the serial number and date plus time. The logfile and the created plots are moved there as well. For the copy and move operation the package *shutil* is used.

2.2.6. Packages

The most commonly used packages and their function are described in the next sections. To install a package, for example ftd2xx, the following command is used [9]:

```
pip install ftd2xx
```

This is only possible if the path to the installed Python is stored in the environmental variables. At the installation of Python it can be done automatically, by checking the checkbox.

2.2.6.1. ftd2xx

The package ftd2xx is a python wrapper of the D2XX dll from FTDI. It is used to get access to the D2XX function described in the D2XX programmer's guide of FTDI [20].
Call_ft and write function of ftd2xx.py:

```python
def call_ft(function, *args):
    """Call an FTDI function and check the status. Raise exception on error"""
    status = function(*args)
    if status != _ft.FT_OK:
        raise DeviceError(status)
def write(self, data):
    """Send the data to the device. Data must be a string representing the
    bytes to be sent"""
    w = _ft.DWORD()
    call_ft(_ft.FT_Write, self.handle, data, len(data), c.byref(w))
    return w.value
```

2.2.6.2. PyQt5

With PyQt5 the whole GUI is created. The multithreading is programmed with *QThread*, which is also a part of this package. Essentially PyQt5 wraps the C++ Qt library for usage in Python. For more information look at the PyQt [3] and Qt [2] documentation.

2.2.6.3. NumPy [5]

NumPy is used for the calculation of the correlation coefficients and for handling multidimensional arrays. With this package, it is much easier to get the results, to handle big matrices and to perform high-level-mathematical calculations. For more information look at the documentation [5].

2.2.6.4. Matplotlib

With the package matplotlib the plots are created and saved as .png files. For more information refer to the matplotlib [6] user's guide.
Example code for plotting a simple graph and save them as .png file:

```
import matplotlib.pyplot as plt
plt.figure(fig_num, figsize=(16, 9), dpi=100)
plt.plot(data, label="label")
plt.xlabel("xlabel")
plt.ylabel("ylabel")
plt.legend(loc='upper right')
plt.title("Title")
plt.show()
plt.savefig(directory + save_name + ".png",
   bbox_inches='tight', dpi=200)
```

2.2.6.5. PyInstaller

For easier use an executable file .exe is created, which bundles the whole Python project, including all dependencies and libraries. The command for generating the file is:

```
pyinstaller -w --onefile C:/path.../main.py
```

With the option *-w* the console is hidden and with - - *onefile* only a single .exe is created. The generated .exe file is then located in the folder "dist", in the root folder of the .py file [28].

2.2.7. The program

Figure 2.37 explains the principal sequence of the software from start to end. The program is started, the .xml file is read and the GUI is opened. Then the test device communication is initialized automatically. After pressing the START button, input for the serial and test engineer will be required and once these are entered, the test cases are executed as selected. When the tests are finished, the result dictionary is created and used as input for the test report generation. After the report is generated, it is saved and opened. Now the program is finished and is waiting for pressing the START button again.

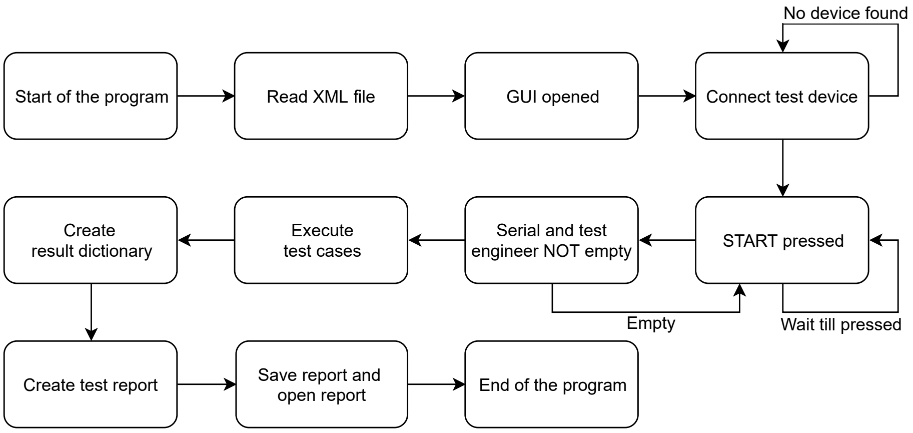


Figure 2.37.: Flow chart of the test program

The communication to the firmware is shown as a flow chart in figure 2.38. At the begin of the communication the command (1 byte) is sent to the FTDI chip, which translates the USB signal to an UART signal. This signal is transmitted to the microcontroller, followed by the value (1 byte). After that the software reads the queue, of the receive buffer of the FTDI chip, while it is empty. If a byte is received and it is the acknowledge byte (ACK), the communication was successful. If not, the error "ACK not received" is displayed.

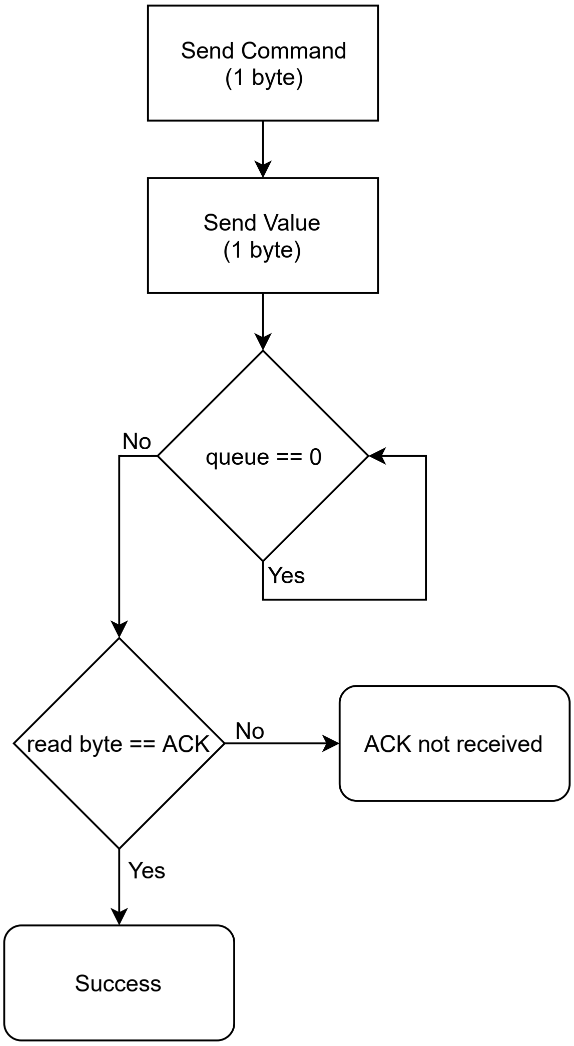


Figure 2.38.: Flow chart of the communication from the software to the firmware

3. Results

3.1. Hardware

The schematic and the layout were created with the software *EAGLE (Autodesk, Inc., San Rafael, United States of America (USA))*. The housing dimensions are imported into Eagle and the PCB outline was adjusted to it. The finished PCB was ordered from the manufacturer *Eurocircuits (Eurocircuits GmbH, Kettenhausen, Germany)*.

3.1.1. Module test adapter

The test adapter with a module is shown in figure 3.1. It can be soldered in two ways. One way is with a rectangular pin header like shown in the picture, and the other way is with a vertical pin header. The used connector for the module has limited mating cycles, so the adapter must be replaced every time the limit is reached.

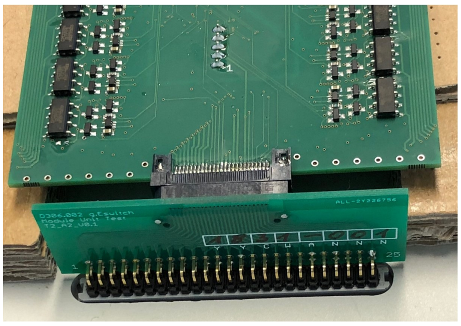


Figure 3.1.: The testadapter and a module

3.1.2. The printed circuit board (PCB)

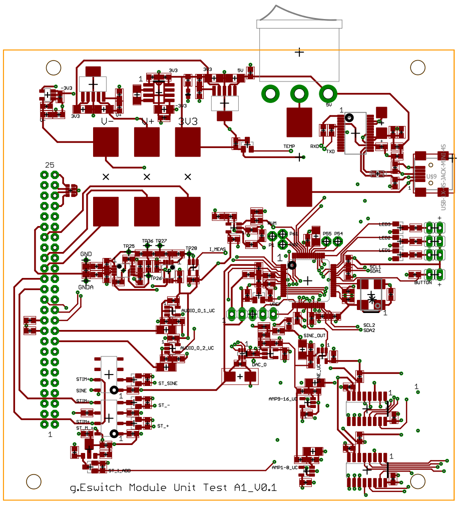


Figure 3.2.: The top side of the PCB

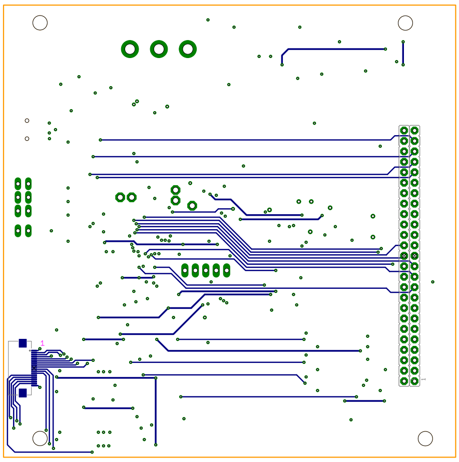


Figure 3.3.: The bottom side of the PCB

In figure 3.4 the manufactured and assembled PCB is shown. The units of the schematic are marked with the red rectangles. In the upper area of the PCB the supply, communication and the fuses are placed. In the central region the microcontroller with the current measurement circuit and the LED connectors are located. Below the central region the bipolar to unipolar converter and the sine generation is located. In the lower right area the multiplexer for audio is on TOP, and on BOT the flex-print-cable (FPC) connector is placed. Marked with (6) is the entire switching part that connects the various signals to test the functions of the module.

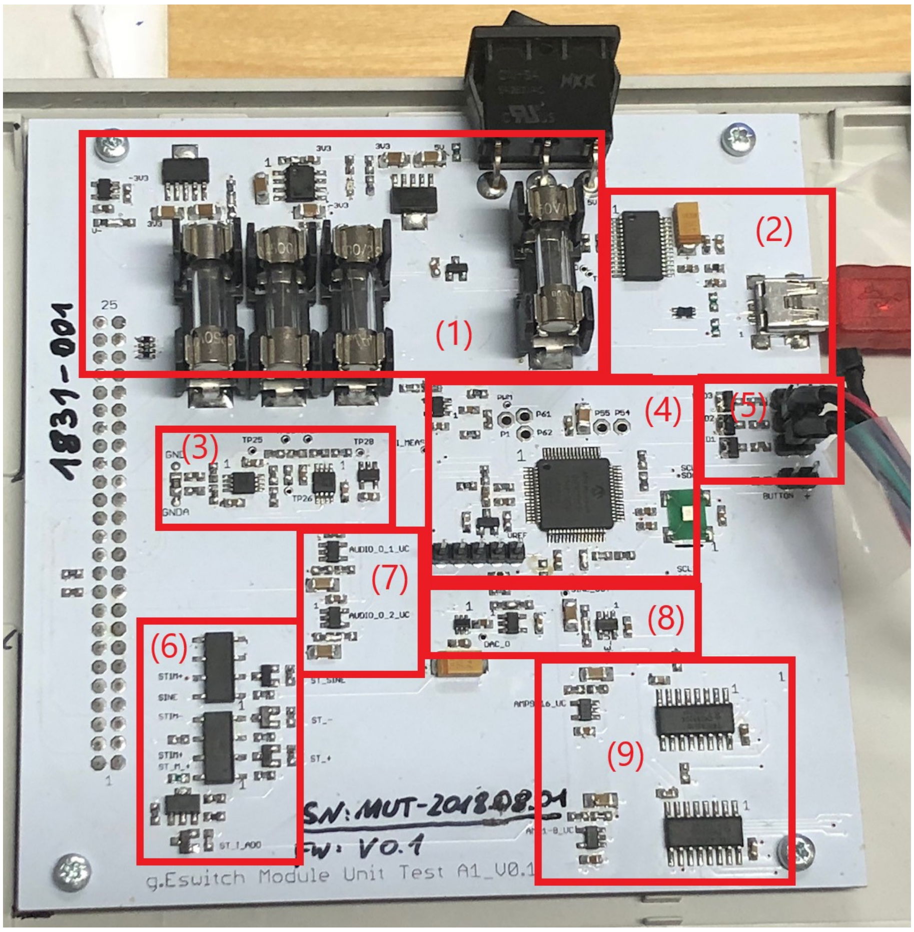
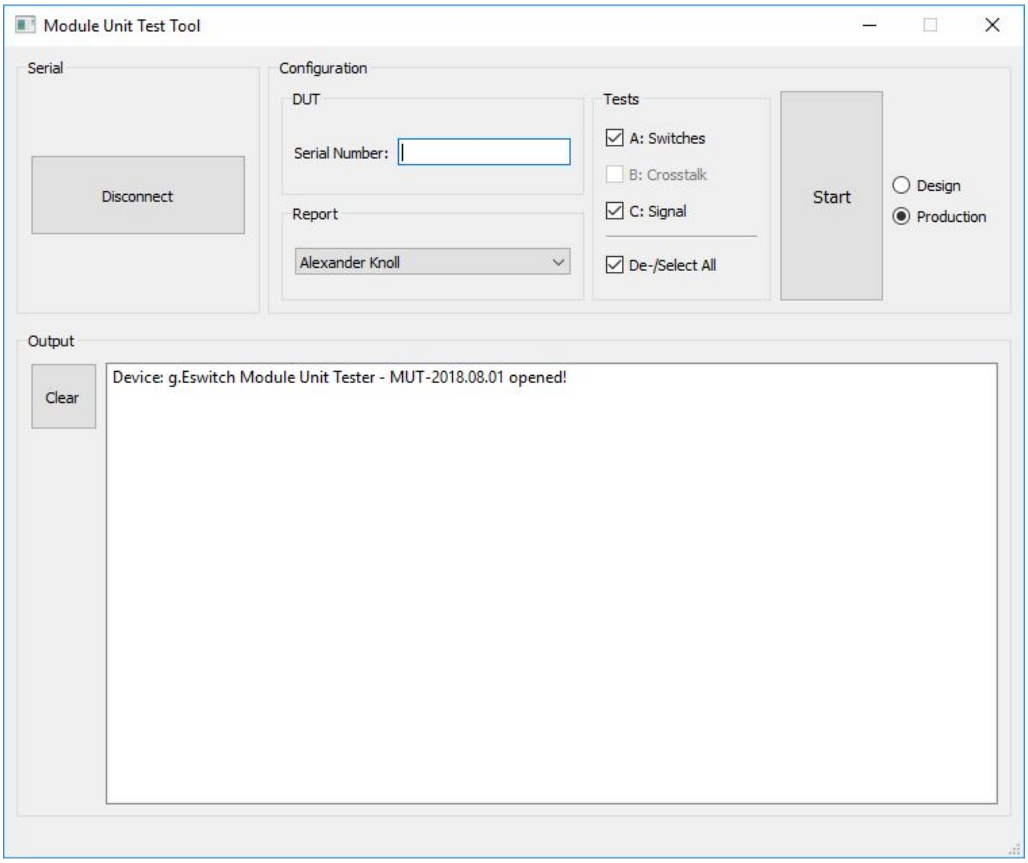


Figure 3.4.: The manufactured PCB: (1) Supply, (2) USB to UART Converter, (3) Current measuring, (4) Microcontroller, (5) LED's, (6) Selftest switching, (7) Bipolar to unipolar converter for ADC measurement, (8) Sinus wave generation, (9) Multiplexing unit for amplifier signal

3.2. Software

3.2.1. Graphical user interface (GUI)

In the following figure 3.5 the GUI is shown. It connects automatically to the test device when it is switched on. If no module is found, or if an error occurrs, it can be connected via the *"Connect/Disconnect"* button manually. For the report the test engineer can be selected with a drop-down list. It is mandatory to enter the serial number of the DUT. With check boxes the tests, which should be done, can be chosen. Test B is not available at the moment and a placeholder for a future test case which is not specified at the moment. When pressing the start button, the tests are done and the output is shown at the text box at the bottom.



Serial Number:

Report: Alexander Knoll

☑ A: Switches

□ B: Crosstalk

☑ C: Signal

☑ De-/Select All

○ Design

● Production

```
Device: g.Eswitch Module Unit Tester - MUT-2018.08.01 opened!
```

Figure 3.5.: The GUI for the module unit tester to verify a module

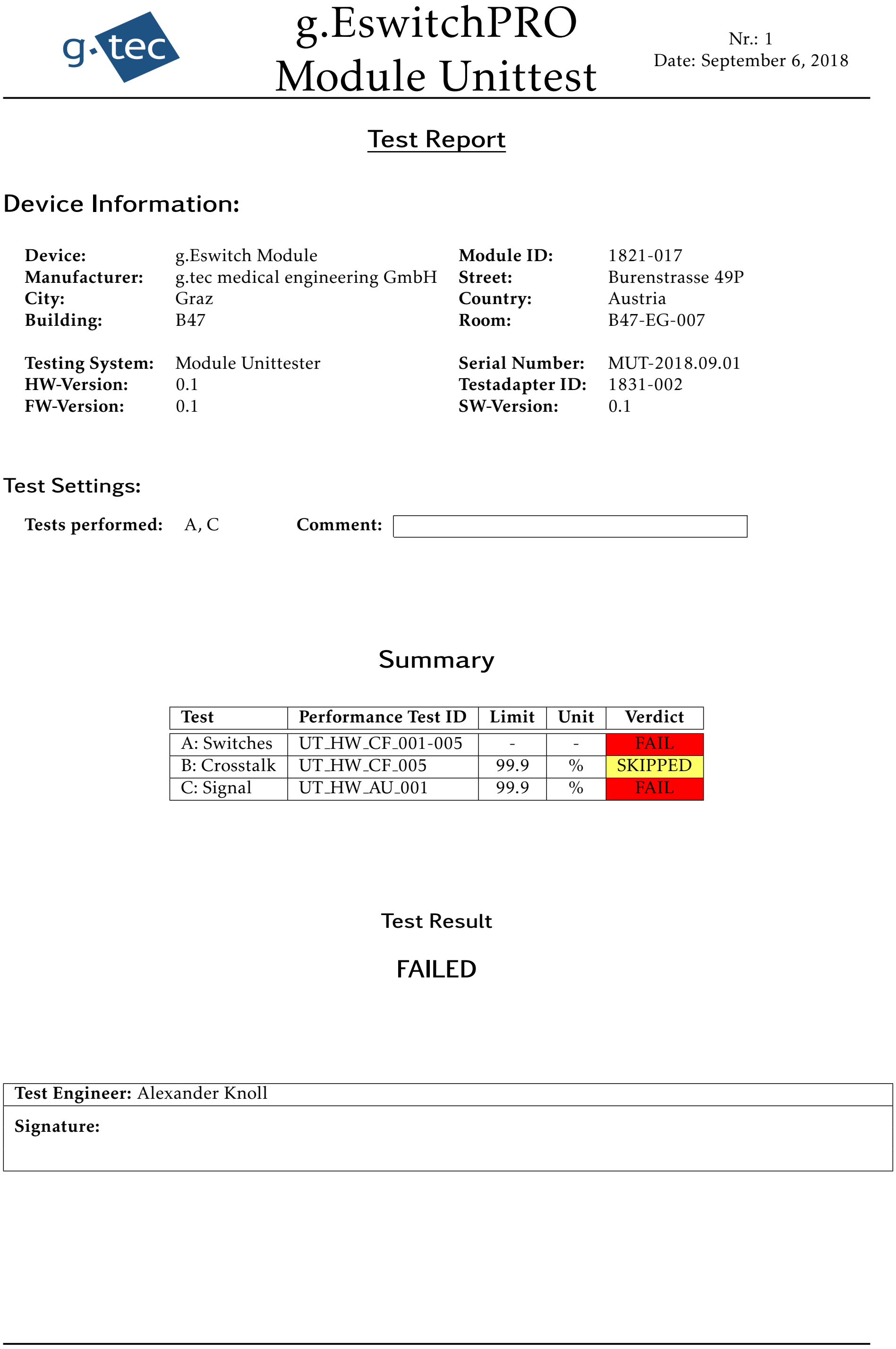
3.3. Test results and measurements

For each module a report is generated as a .pdf file. The first page is printed out and stored in a folder (see figure 3.6 for the layout of the first page). It shows an overview for the completed tests.

The top of the page shows a continuous number for the total number of tests done, with the current test adapter. After exceeding a defined number the test adapter must be replaced, due to the limited mating cycles of the connector. This page also shows the device information, which is imported for verification and the test settings. In the summary the tests are marked as failed, passed or skipped. If test A and C, which are necessary, are passed and not skipped, the final test result is passed. If not, it is failed like in the example. At the bottom the test engineer is displayed, who needs to sign the sheet and store it.

The further pages only contain information about the problem and why the test failed. This information is necessary for repairing the module. For more information, see the results in the further subsections. This report was created with a module where the channel 15 and 16 are not working due to the desoldering of a switch.

With the test result below the damaged switches of channel 15 and 16 can be identified, and after changing them the test passes.



g.EswitchPRO
Module Unittest

Nr.: 1
Date: September 6, 2018

Test Report

Device Information:

Device:	g.Eswitch Module	**Module ID:**	1821-017
Manufacturer:	g.tec medical engineering GmbH	**Street:**	Burenstrasse 49P
City:	Graz	**Country:**	Austria
Building:	B47	**Room:**	B47-EG-007
Testing System:	Module Unittester	**Serial Number:**	MUT-2018.09.01
HW-Version:	0.1	**Testadapter ID:**	1831-002
FW-Version:	0.1	**SW-Version:**	0.1

Test Settings:

Tests performed: A, C **Comment:** □

Summary

Test	Performance Test ID	Limit	Unit	Verdict
A: Switches	UT_HW_CF_001-005	-	-	FAIL
B: Crosstalk	UT_HW_CF_005	99.9	%	SKIPPED
C: Signal	UT_HW_AU_001	99.9	%	FAIL

Test Result

FAILED

Test Engineer: Alexander Knoll

Signature:

Figure 3.6.: The first page of the test report

3.3.1. Test A: Switches

Below the results of the first test are shown. Channel 15 and 16 are not ok, as expected. The code and count help the test engineer to determine, what is wrong with the channel and makes it easier to repair the module.

TEST A: Switches

Bad channels: **15, 16**

Table 3.1.: The appendix for "Test A: Switches". The status shows if the switches of the channel are working as intended. Additionally the latest error code is shown and also the amount of errors verifying issues with the module.

Channel	Status	Code	Count
1	OK	-	-
2	OK	-	-
3	OK	-	-
4	OK	-	-
5	OK	-	-
6	OK	-	-
7	OK	-	-
8	OK	-	-
9	OK	-	-
10	OK	-	-
11	OK	-	-
12	OK	-	-
13	OK	-	-
14	OK	-	-
15	NOK	15	7
16	NOK	15	7

3.3.2. Test C: Signal

In this section the results of the signal test are shown. The cross correlation is calculated after the measurement of the input and output, and is shown in table 3.2. All values greater than 0.999 which means 99.9% correlation are green and passed. The two ”sabotaged” channels are worse in correlation, so they failed as expected. Additionally, the measured values are plotted to give the test engineer more data for analysis.

Bad channels: **15, 16**

Table 3.2.: The appendix for ”Test C: Signals”. The cross correlation coefficients are shown in the table and the bad channels are marked red.

Channels	Correlation Coefficients							
1-8	0.9994	0.9995	0.9994	0.9992	0.9993	0.9994	0.9994	0.9993
9-16	0.9994	0.9994	0.9995	0.9994	0.9994	0.9996	0.5073	0.4171

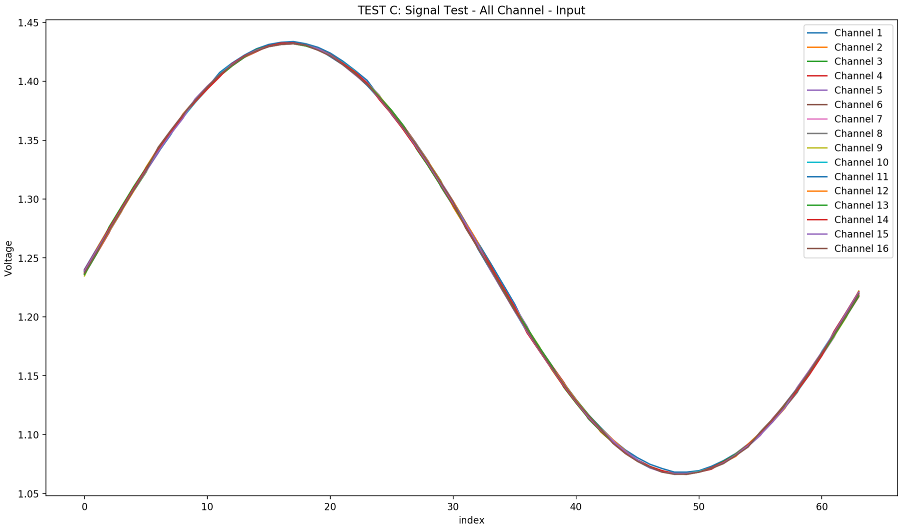


Figure 3.7.: Here all signals, measured at the ADC of the microcontroller, are plotted. These are the direct output of the sine generation circuit and therefore all channels should be identically.

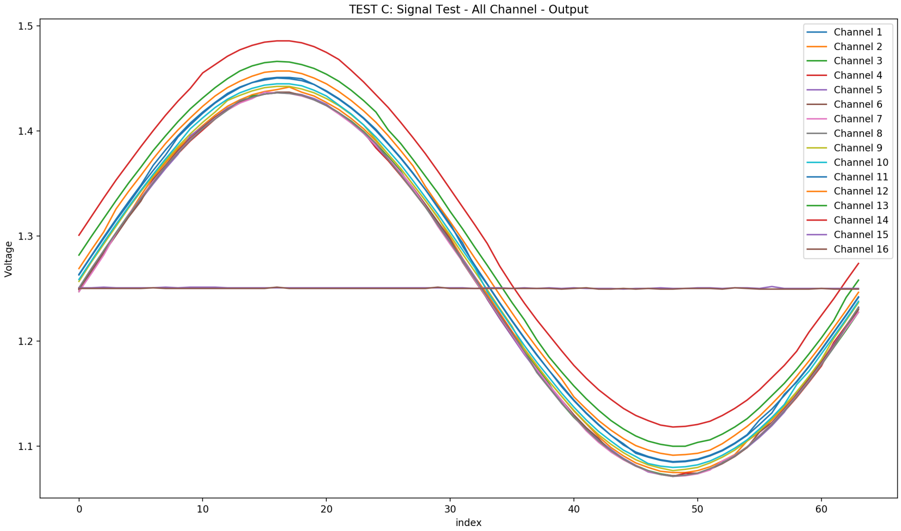


Figure 3.8.: This is the plotted output signal of the DUT measured at the microcontroller ADC on the test device. Because of the tolerances of the parts and the buffer these signals have an offset, but should be identically in waveform. Since channel 15 and 16 are not working these outputs are zero which is already recognized at the correlations coefficients.

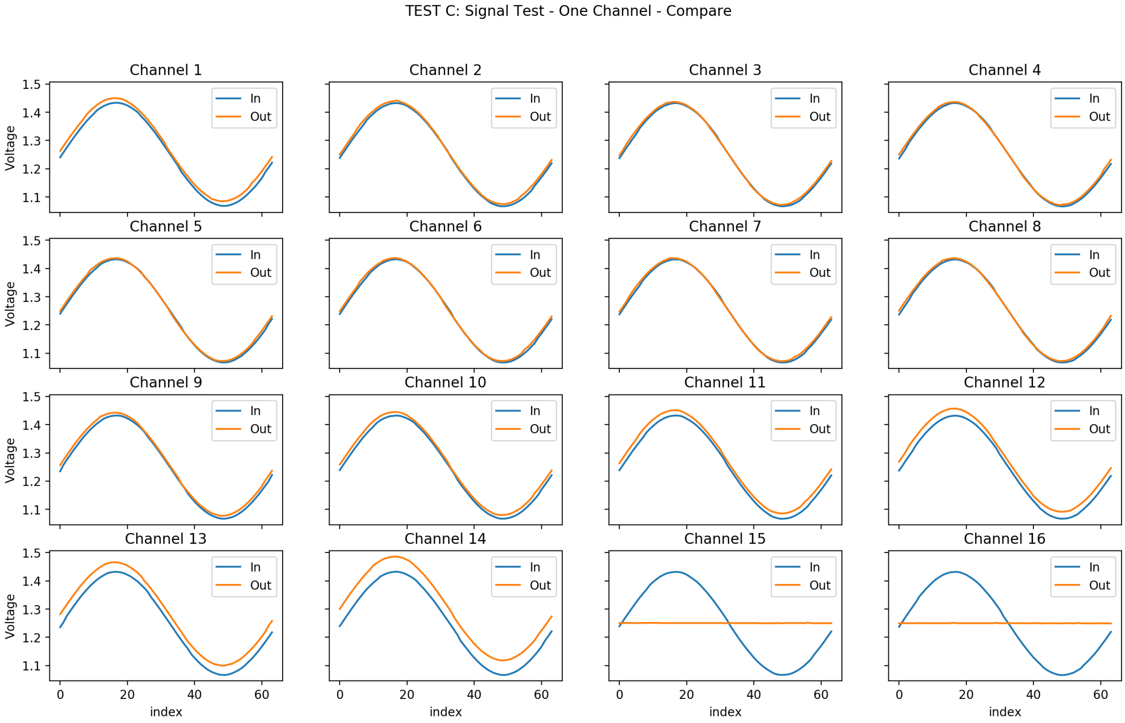


Figure 3.9.: Here all 16 channels are plotted separately with input and output in the same plot. Also, in this plot the not working channels are visible.

3.3.3. The LOG file

For each DUT, in addition to the report, a log file is written and stored in the report folder. This file is used for logging the output while running the software and testing a module.

Here is an example of this file:

```
 INFO: ————————————————————-
INFO: Datum: 06-09-2018 15-56-22
INFO:
INFO: Device: g.Eswitch Module Unit Tester - MUT-2018.08.01 opened!
INFO: Module Serial: 1821-017
INFO: Testadapter ID: MUT-2018.09.01
INFO: Testcount: 1
INFO: Engineer: Alexander Knoll
INFO: ————————————————————-
INFO: Test A: Switches started...
INFO: SUCCESS Write to uC S, 0
INFO: CH 1: OK
INFO: CH 2: OK
INFO: CH 3: OK
INFO: CH 4: OK
INFO: CH 5: OK
INFO: CH 6: OK
INFO: CH 7: OK
INFO: CH 8: OK
INFO: CH 9: OK
INFO: CH 10: OK
INFO: CH 11: OK
INFO: CH 12: OK
INFO: CH 13: OK
```

```
INFO: CH 14: OK
INFO: CH 15: NOK - CODE: 15 Count: 7
INFO: CH 16: NOK - CODE: 15 Count: 7
INFO: NOK: 2
INFO: OK: 14
INFO: Test A (Switches) finished...
INFO: Time needed: 0:00:12
INFO: ————————————————————-
INFO: Test C: Signal started...
INFO: SUCCESS Write to uC A, 0
INFO: Start testing channels...
INFO: Channel 1
INFO: Channel 2
INFO: Channel 3
INFO: Channel 4
INFO: Channel 5
INFO: Channel 6
INFO: Channel 7
INFO: Channel 8
INFO: Channel 9
INFO: Channel 10
INFO: Channel 11
INFO: Channel 12
INFO: Channel 13
INFO: Channel 14
INFO: Channel 15
INFO: Channel 16
INFO: creating Plot...
INFO: template/figs/testc_input.png saved
INFO: creating Plot...
INFO: template/figs/testc_output.png saved
INFO: creating Plot...
INFO: template/figs/testc_compare.png saved
INFO: calculating correlation coefficients...
```

```
INFO: Time needed: 0:00:16
INFO: ——————————————————————-
INFO: Creating report...
INFO: Report created...
INFO: Report data moved...
INFO: ——————————————————————-
INFO: Time needed total: 0:00:35
INFO: ——————————————————————-
INFO: All tests done!
INFO: Resetting uC...
INFO: Resetting uC...
INFO: SUCCESS Write to uC r, 0
INFO: ——————————————————————-
```

4. Discussion

4.1. Hardware

The hardware of the test device meets the requirements specified. There are no big issues and the functionality has been verified in a documented way by the company, according to the standards. The mechanical parts, like the test adapter stability when plugging and unplugging, should be improved. A front and back-panel has to be built and a label for the test device has to be created. Also, a nice feature would be, if the test device recognizes a plugged-in module and starts the selected test. The cycle counter could then be incremented for each mating cycle automatically. This would give the possibility to do faster batch processing.

4.2. Software

The software works as intended. The test cases and the limits are defined in the specification document of the company. It is possible that it will be changed in the future, since they are too tolerant or restrictive for the tests. This is easily done, by altering the .xml input file. Generally, it would be nice, if the test data are saved in a future version of the test device. Also, the user should be able to cancel the test procedure. With the final module it will also be possible to read out the hardware and firmware version to use it in the test software. At the moment, the test count is also saved in the .xml file, which should be only an input file in the future and therefore the test count should be saved in an extra file. A menu at the top of the GUI with a button to change the settings in the

.xml file would ensure that it will not be damaged if parts are deleted or typed incorrectly. Another advantage of using a menu is that information about the plugged-in module and the test device can also be displayed without having to create a report.

4.2.1. Test cases

"Test A: Switches" is working as intended and the results are verified as correct. For "Test C: Signal" the correlation coefficient is a good method to verify the result, to get an idea, if it is working in principal. For a more precise statement, it is necessary to look at the amplitude and the offset as well.

4.2.2. Report

In the report, a short description of the test would make it easier to understand what is done. If an error occurred, an error description should be implemented to make it easier to diagnose the module and to quicker decide what to do with it. An example plot "How it should look" also increases the speed of finding a damaged channel or function. In batch processing, an overview page of the tests performed and an appendix with the failed reports would mean that the test engineer only has to sign one document and not one for each module, saving paper and protecting the environment.

4.3. Conclusion

The goal of creating a test-device to verify a unit of a product within the design process was fulfilled, and the created device is used for quality assurance and to save time and money compared to manual testing. It is possible to identify different errors and to do an inspection of ordered modules of the product upon receipt. For each device a report is generated. An example is appended to the appendix A. It is also possible to do an individual test case after a device has been repaired to see if it passes the test.

On a final note, it can be said that the device created will have an important role in the design process and will be also used in future possibly with slight modifications as described in the section above.

4.4. Outlook

The built test device is an important part of the product design process to ensure the quality and functions as intended. In the future also a production test has to be built in order to test the function of each device after production, and to minimize the error rate at roll-out.

Bibliography

[1] Inc Analog Devices. Datasheet: 12-bit nanodac with spi, September 2018. http://www.analog.com/media/en/technical-documentation/data-sheets/AD5601_5611_5621.pdf.

[2] The Qt Company. Qt documentation, September 2018. http://doc.qt.io/qt-5/.

[3] Riverbank Computing. Pyqt5 reference guide, September 2018. http://pyqt.sourceforge.net/Docs/PyQt5/.

[4] Inc. Daycounter. Sine look up table generator calculator, September 2018. https://daycounter.com/Calculators/Sine-Generator-Calculator.phtml.

[5] SciPy developers. Numpy and scipy documentation, September 2018. https://docs.scipy.org/doc/.

[6] The Matplotlib development team. Matplotlib user's guide, September 2018. https://matplotlib.org/users/index.html.

[7] Bopla enclosures. Botego bo514 enclosures, September 2018. https://www.bopla.de/en/enclosure-technology/product/botego/botego-bo-514-l-enclosures/bo-51412.html.

[8] FDA, Center for Devices and Radiological Health. 21 CFR Part 820 (Quality System Regulation). 2018.

[9] Python Software Foundation. Pip install reference guide, September 2018. https://pip.pypa.io/en/stable/reference/pip_install/.

[10] Python Software Foundation. Python, September 2018. https://www.python.org/.

[11] Eurociruits GmbH, September 2018. https://www.eurocircuits.com/.

[12] Microchip Technology Inc. dspic33ep256mu806, September 2018. https://www.microchip.com/wwwproducts/en/dsPIC33EP256MU806.

[13] Microchip Technology Inc. Low-power linear active thermistor ics, September 2018. http://ww1.microchip.com/downloads/en/DeviceDoc/20001942G.pdf.

[14] Microchip Technology Inc. Mplab® icd 3 in-circuit debugger user's guide, September 2018. http://ww1.microchip.com/downloads/en/DeviceDoc/50002081B.pdf.

[15] Microchip Technology Inc. Pickit™ 3 in-circuit debugger/programmer user's guide, September 2018. http://ww1.microchip.com/downloads/en/DeviceDoc/52116A.pdf.

[16] Texas Instruments Incorporated. Sine wave generation using pwm, September 2018. http://www.ti.com/lit/an/spna217/spna217.pdf.

[17] International Organisation for Standardization. EN ISO 13485:2016 Medical devices - Quality management systems - Requirements for regulatory purposes. 2016.

[18] Kyocera. Clock oscillators surface mount type, September 2018. https://global.kyocera.com/prdct/electro/product/pdf/clock_k_e.pdf.

[19] Future Technology Devices International Ltd. D2xx direct drivers, September 2018. https://www.ftdichip.com/Drivers/D2XX.htm.

[20] Future Technology Devices International Ltd. D2xx programmer's guide, September 2018. https://www.ftdichip.com/Support/Documents/ProgramGuides/D2XX_Programmer's_Guide(FT_000071).pdf.

[21] Future Technology Devices International Ltd. Ft232r usb uart ic, September 2018. https://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf.

[22] oxforddictionaries. Definition of memory in english:, July 2018. https://en.oxforddictionaries.com/definition/memory.

[23] oxforddictionaries. Definition of motoric in english:, July 2018. https://en.oxforddictionaries.com/definition/motoric.

[24] oxforddictionaries. Definition of perception in english:, July 2018. https://en.oxforddictionaries.com/definition/perception.

[25] oxforddictionaries. Definition of perception in english:, July 2018. https://en.oxforddictionaries.com/definition/cognition.

[26] Prueckl et al. Passive functional mapping guides electrical cortical stimulation for efficient determination of eloquent cortex in epilepsy patients. 2017.

[27] Samtec. 0.80 mm edge rate® rugged high-speed socket strip, September 2018. https://www.samtec.com/products/erf8.

[28] PyInstaller Development Team. Pyinstaller quickstart, September 2018. https://www.pyinstaller.org/.

[29] Inc. The MathWorks. Correlation coefficients, September 2018. https://de.mathworks.com/help/matlab/ref/corrcoef.html.

A. Appendix

A.1. Firmware Code

Example code for the ”Test A: Switches” test function:

```
//-----------------------------------------------------------------------------
/// Testprocedure for the switch test
///
/// \param module_adress:
///      module_adress=0b01010000: address of the connected module
///
/// \return none
//
void startSwitchTest(uint16_t module_address)
{
   // initialize a buffer for i2c data
   uint8_t i2c_buffer_TestA[2] = {0, 0};
   // initialize transfer_TestA as predefined struct
   i2cTransfer_t transfer_TestA;

    //prepare i2c transfer data struct
   transfer_TestA.data = i2c_buffer_TestA;
   transfer_TestA.size = 2;
   transfer_TestA.dest_addr = module_address;
   transfer_TestA.i2c_name = 1;

   while(channel <= MAX_CHANNEL)
   {
      ... // Testprocedure
   }
}
```

Example code for the "Test C: Signal" test function:

```
//----------------------------------------------------------------------------
/// Testprocedure for the signal test
///
/// \param module_adress: 0b01010000 = address of the connected module
///
/// \return none
//
void startAudioTest(uint16_t module_address, uint16_t channel)
{
 uint8_t idx = 1;
 // initialize a buffer for i2c data
 uint8_t i2c_buffer_TestC[2] = {0, 0};
 // initialize transfer_TestC as predefined struct
 i2cTransfer_t transfer_TestC;

 //prepare i2c transfer data struct
 transfer_TestC.data = i2c_buffer_TestC;
 transfer_TestC.size = 2;
 transfer_TestC.dest_addr = module_address;
 transfer_TestC.i2c_name = 1;

 for(idx = 1; idx <= MAX_CHANNEL; idx++)
 {
  //activate the test circuit
  gpioSetPin(ST_SINE);
  gpioClearPin(UC_HI);
  // switch channel of module multiplexer to connect the
    channel to the ADC of the microcontroller
  configChannel(AUDIO, idx);
  //activate multiplexer
  gpioClearPin(M_E);
  // prepare data for the transfer - ST_D: switches STIM+ to
    the buffer of the respective channel
  i2c_buffer_TestC[0] = ST_D;
  i2c_buffer_TestC[1] = idx;
  // send data to module via I2C
  i2cWriteBuffer(&transfer_TestC);
  // send sinewave via DAC and read values via ADC -> send
    data to PC software
  sendSineWave(sineLUT64hex, 10);
 }
}
```

The used look up table was generated with a tool, from the website of Daycounter, Inc. [4].

```
const uint16_t sineLUT64hex[] = {
  0x1f4,0x225,0x256,0x285,0x2b3,0x2e0,0x30a,0x331,
  0x356,0x377,0x394,0x3ad,0x3c2,0x3d2,0x3de,0x3e6,
  0x3e8,0x3e6,0x3de,0x3d2,0x3c2,0x3ad,0x394,0x377,
  0x356,0x331,0x30a,0x2e0,0x2b3,0x285,0x256,0x225,
  0x1f4,0x1c3,0x192,0x163,0x135,0x108,0xde,0xb7,
  0x92,0x71,0x54,0x3b,0x26,0x16,0xa,0x2,
  0x0,0x2,0xa,0x16,0x26,0x3b,0x54,0x71,
  0x92,0xb7,0xde,0x108,0x135,0x163,0x192,0x1c3,
};
```

Example code for the sine generation and reading the values with the ADC:

```
//---------------------------------------------------------------------------
/// Testprocedure for the signal test
///
/// \param sineLUT: look up table for the sine
/// \param duration: number of periods
///
/// \return none
//
void sendSineWave(const uint16_t* sineLUT, uint16_t duration)
{
   uint16_t idx = 0;
   uint16_t counter = 0;
   //prepare array for 5 adc channels and the 64 bit data arrays for storing the values
   uint16_t adc_values[5] = {0};
   uint16_t sine_values[64] = {0};
   uint16_t audio_values_1[64] = {0};
   uint16_t audio_values_2[64] = {0};

   while(counter < duration)
   {
    for (idx = 0; idx <= 63; idx++)
    {
      gpioClearPin(DAC_CS);
      spiDataWriteWait(1, sineLUT[idx] + 5700);
      gpioSetPin(DAC_CS);

      readADC(adc_values);
      sine_values[idx] = adc_values[3];
      audio_values_1[idx] = adc_values[1];
      audio_values_2[idx] = adc_values[2];
      // wait for 1 ms - this is used for getting a sine wave with approximately 15 Hz
      wait(1);
    }
```

```
      counter++;
    }
    // pull chip select(CS) to LOW and send data to SPI with a
       offset, then CS is set to HIGH again
    gpioClearPin(DAC_CS);
    spiDataWriteWait(1, sineLUT[0] + 5700);
    gpioSetPin(DAC_CS);
    // send data arrays to the PC software via UART
    uart1SendUINT16Array(sine_values, 64);
    uart1SendUINT16Array(audio_values_1, 64);
    uart1SendUINT16Array(audio_values_2, 64);
}
```

A.2. Automatically created test report

The test report is shown on the next pages.

g.EswitchPRO
Module Unittest

Nr.: 1
Date: September 6, 2018

Test Report

Device Information:

Device:	g.Eswitch Module	**Module ID:**	1821-017
Manufacturer:	g.tec medical engineering GmbH	**Street:**	Burenstrasse 49P
City:	Graz	**Country:**	Austria
Building:	B47	**Room:**	B47-EG-007
Testing System:	Module Unittester	**Serial Number:**	MUT-2018.09.01
HW-Version:	0.1	**Testadapter ID:**	1831-002
FW-Version:	0.1	**SW-Version:**	0.1

Test Settings:

Tests performed: A, C **Comment:**

Summary

Test	Performance Test ID	Limit	Unit	Verdict
A: Switches	UT_HW_CF_001-005	-	-	FAIL
B: Crosstalk	UT_HW_CF_005	99.9	%	SKIPPED
C: Signal	UT_HW_AU_001	99.9	%	FAIL

Test Result

FAILED

Test Engineer: Alexander Knoll
Signature:

TEST A: Switches

Bad channels: **15, 16**

Channel	Status	Code	Count
1	OK	-	-
2	OK	-	-
3	OK	-	-
4	OK	-	-
5	OK	-	-
6	OK	-	-
7	OK	-	-
8	OK	-	-
9	OK	-	-
10	OK	-	-
11	OK	-	-
12	OK	-	-
13	OK	-	-
14	OK	-	-
15	NOK	15	7
16	NOK	15	7

TEST C: Signal

Bad channels: **15, 16**

Channels	Correlation Coefficients							
1-8	0.9994	0.9995	0.9994	0.9992	0.9993	0.9994	0.9994	0.9993
9-16	0.9994	0.9994	0.9995	0.9994	0.9994	0.9996	0.5073	0.4171

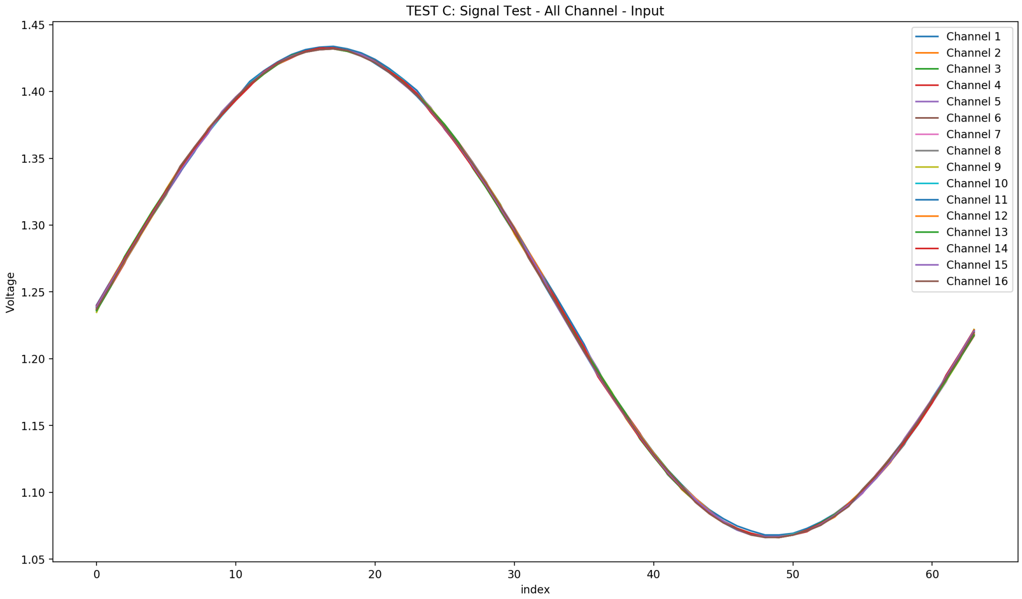


Figure 1: All Channel Input

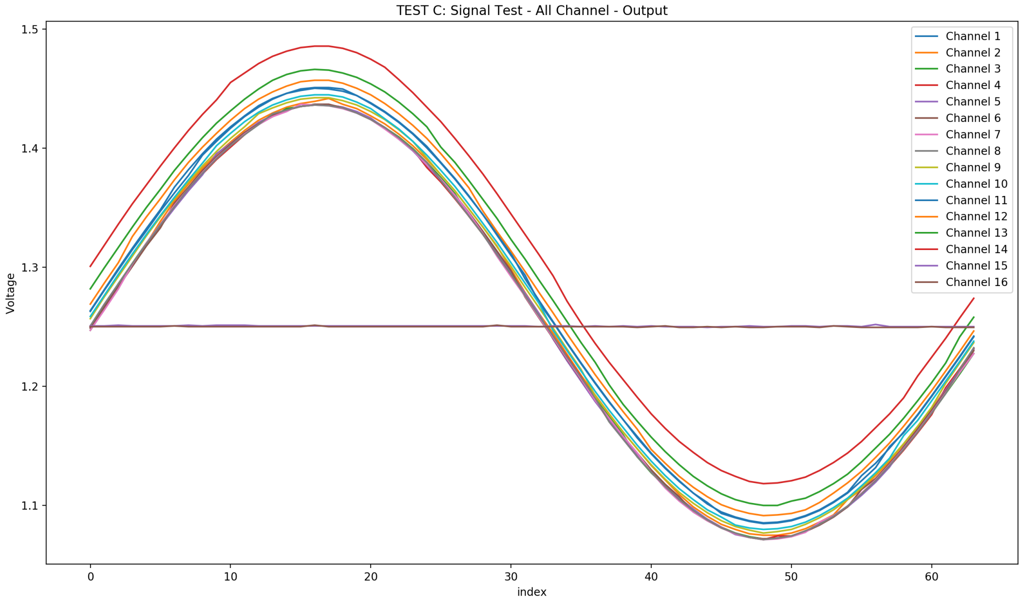


Figure 2: All Channel Output

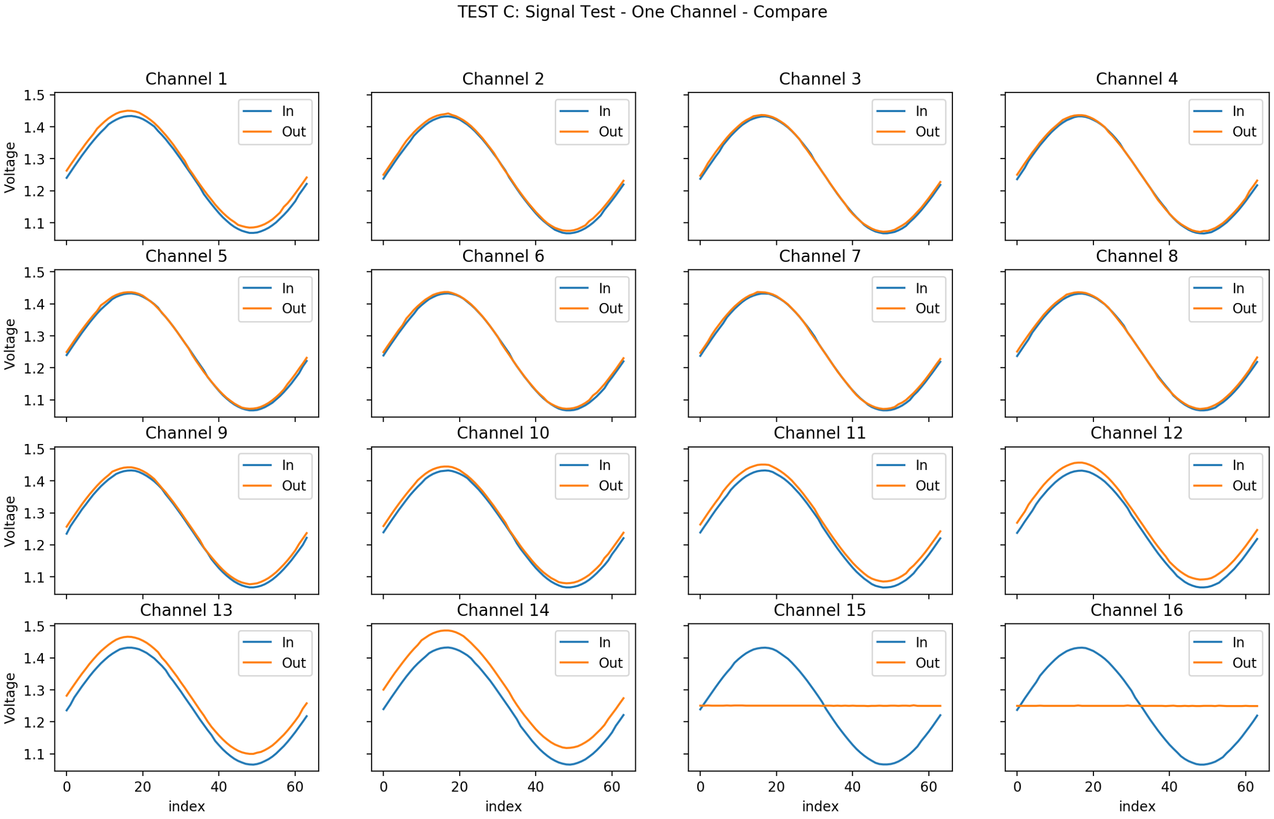


Figure 3: All Channel Compare