Universität der Bundeswehr München

Fakultät für Informatik
Institut für Technische Informatik

{JONATHAN LUSKY

jonathan.lusky@unibw.de
www.jlus.de

Master Thesis

Master of Science, Computer Science

Design and Programming of the flying electronic systems for the REXUS HERMESS experiment

by Jonathan Lusky
(Matrikel-Nr: 1173942)

Neubiberg, July 28, 2021

Supervised by the chair of "Embedded Systems / Computers in Technical Systems"

First Examiner:	Prof. Klaus Buchenrieder Ph.D.
Second Examiner:	Prof. Dr. Wolfgang Hommel
Supervisor:	Dr. Heike Rolfs


REXUS HERMESS
UNIVERSITY OF THE GERMAN ARMED FORCES MUNICH


REXUS/BEXUS
Student Experiment Programme


DLR
Deutsches Zentrum
für Luft- und Raumfahrt
German Aerospace Center


Rymdstyrelsen
Swedish National Space Agency


esa


SSC


ZARM

Contents

List of Figures

List of Listings

List of Abbreviations

ADC	analog to digital converter	17
AHB	advanced high-performance bus	39
AMBA	Advanced Microcontroller Bus Architecture	6
APB3	advanced peripheral bus 3	6
API	application programming interface	49
BEXUS	Ballon EXperiment for University Students	12
CCC	clock conditioning circuit	48
CRC	cyclic redundancy check	53
CS	chip select	42
D	design requirement	22
DAPI	Data- and Programming Interface	26
DC	direct current	15
DLR	German Aerospace Center	12
ECSS	European Cooperation for Space Standardization	21
EEPROM	electrically erasable, programmable read only memory	49
EMI	electromagnetic interference	23
ESA	European Space Agency	12
F	functional requirement	21
FIC	fabric interface controller	39
FIIC	fabric interface interrupt controller	39
FPGA	field programmable gate array	26
FSR	full scale range	18
GPIO	general porpuse input / output	39
GSS	ground station software	7
GUI	graphical user interface	53
HAL	hardware abstraction layer	49
HBK	Hottinger, Brüel & Kjær	16
HBM	Hottnger Baldwin Messtechnik GmbH	16
HDL	hardware description language	6
HERMESS	Hull applied ElectroResistive Measurement of Structural Strains	6
IC	integrated circuit	28
ISR	interrupt service routine	49

IDE	integrated development environment	38
I/O	input / output	29
JTAG	Joint Test Action Group	57
LED	light emitting diode	31
LSB	least significant bit	18
LO	Lift-Off	15
MISO	master in / slave out	42
MORABA	Mobile Rocket Base	12
MOSI	master out / slave in	42
MSS	microcontroller subsystem	26
NVIC	nested vector interrupt controller	39
O	operational requirement	24
P	performance requirement	22
PCB	printed circuit board	28
PCM	pulse code modulation	15
PGA	programmable gain amplifier	18
RAM	random access memory	49
REXUS	Rocket EXperiment for University Students	12
RTD	resistive temperature detector	6
RXBX	REXUS / BEXUS	12
RXSM	REXUS service module	14
SED	Student Experiment Documentation	14
SF2	SmartFusion2	28
SGR	strain gauge rosette	16
SNR	signal-to-noise-ratio	19
SNSA	Swedish National Space Agency	12
SoC	System on Chip	27
SODS	Start / Stop of Data Storage	15
SOE	Start / Stop of Experiment	15
SPI	serial peripheral interface	29
SPU	Signal Processing Unit	6
SSC	Swedish Space Corporation	12
STAMP	Strain and Temperature Applied Measurement Point	6
TM	telemetry	15
UART	universal asynchronous receiver transmitter	15
WD	Watchdog	48
ZARM	Applied Space Technology and Microgravity	12

1 | Introduction

Sounding rockets for high-altitude research are subjected to a harsh environment during take-off, re-entry and landing. Employing high safety margins in structural designs helps mitigating failures leading to damages or even the loss of a vehicle and therefore improve operational safety. To help increase the effiency of sounding rockets, the Hull applied ElectroResistive Measurement of Structural Strains (HERMESS) experiment aims to provide a system for gathering accurate in-flight data on loads occuring in the rockets structures. These data can be used to help understanding the acting forces and therefore enable verification and enhancement of existing simulation models. This improved understanding may lead to new structural designs for sounding rockets to increase payload mass to overall mass ratio resulting in lower costs per mass unit. [15]

Expensive and effortful launches of the experiment are necessary in order to gain enough data to generate meaningful insights. These are usually not managable by a single university but needs cooperative actions with other parties involved with space technology and science. The REXUS / BEXUS (RXBX) programme is such a partner and makes this work possible.

"The Rocket EXperiment for University Students (REXUS)/Ballon EXperiment for University Students (BEXUS) (RXBX) programme is realised under a bilateral Agency Agreement between the German Aerospace Center (DLR) and the Swedish National Space Agency (SNSA). The Swedish share of the payload has been made available to students from other European countries through the collaboration with the European Space Agency (ESA). Experts from DLR, Swedish Space Corporation (SSC), Applied Space Technology and Microgravity (ZARM) and ESA provide technical support to the student teams throughout the project. EuroLaunch, the cooperation between the Esrange Space Center of SSC and the Mobile Rocket Base (MORABA) of DLR, is responsible for the campaign management and operations of the launch vehicles." [23]

In 2019 the HERMESS experiment successfully qualified for participation in the REXUS progamme for a launch on the Improved Orion "RX29" together with four more experiments contributed by other student teams. The launch campaign is expected to be executed in 2022 from the Esrange Space Center in Sweden. Currently 30 students from the Universität der Bundeswehr München (university of the federal armed forces of Germany in Munich) participate in one of five technical groups inside

the HERMESS team working on signal generation and processing, software, calibrations, testing, mechanical hardware and outreach. [15]

The goal of this thesis is to implement and verify a system stack enabling measurement of strains and temperature on the outer hull surface of a sounding rocket during flight. This is to be done by utilizing the sensorplatform described in section 2.2. Furthermore, the measurements shall be stored and made available in- and post-flight. For approval by the RXBX review board, all components must comply with the requirements listed in section 3.1. It was determined, that a new signal acquisition and processing approach with a novel concept was necessary to address the shortcomings of previous implementation attempts characterized in section 3.2. Generally, this thesis shall provide a base for all flying electronic systems on which further work can be done.

This document first describes some basics necessary for a better understanding of the design choices made throughout the thesis. Next up is a summary of the various requirements imposed by other either the RXBX board or other HERMESS team members working on the mechanical components or mathematical aspects. The concept to be implemented outlining what exact subsystems need implementation to fulfill the overal goal is described after that. The main part, the implementation, is described in chapter 4 and not only highlights certain aspects that needed special considerations, but also provides review-of-design verifications for most of the requirements listed before. The evaluation chapter covers how the developed parts were or will be verified by formal testing in the future.

2 | Basics

Some basics helping with understanding the main chapters 3: "Requirements and Concept" and 4 "Implementation" are provided here. Sections 2.1 and 2.2 contains background knowledge about REXUS or HERMESS specifics based on internal documentation and efforts made prior to this thesis.

Especially useful resources for further reading are the HERMESS Student Experiment Documentation (SED) in its current version [15] and the REXUS user manual [8].

2.1 REXUS vehicle and mission profile

The REXUS sounding rocket is the primary target vehicle for the HERMESS experiment. All related mechanical, elecrical and electronical interfaces must therefore meet the specifications provided via the REXUS User Manual [8]. In addition, the experiments design must perform in scope of the mission profile dictating estimated flight metrics, timed events and launch and recovery procedures.

Typically a single stage rocket with a hull diameter of 356 mm and height between 5 m and 6 m is used as the REXUS vehicle. An Improved Orion solid fuel rocket motor, capable of transporting approximately 95 kg of payload mass to an altitude between 80 km and 90 km after around 140 s with a burning duration of only 26 s, will be used for the "RX29" mission, in which HERMESS participates alongside four other student teams. Because the rocket is spin stabilized at 4 Hz exceeding the limits of other experiments, a Yo-Yo de-spin system reducing the spin rate to a maximum of 0.08 Hz will be triggered 65 s after lift-off, a second before motor separation. [8, 22]

The payload consists of a recovery module mounted directly above the rocket motor, the REXUS service module (RXSM) and five experiments. The HERMESS experiment will be located closest to the predicted position experiencing the highest flight loads, which is in between the RXSM and the recovery module. [15, 8, 22]

When the rocket descended to 4.6 km above sea level after reaching apogee, a parachute will be deployed in two stages via a drogue chute by the recovery mod-

ule. This system is designed to reduce the speed for impact with ground to around 8 m/s. The GPS coordinates will be transmitted via an IRIDIUM satellite communication link, to help the airborn recovery team locating the payload. Additionally, the coordinates will be transmitted via the telemetry stream and an autonomous homing beacon transmitter is included in the recovery system.

The primary electronical interface for all experiments is the RXSM. Via a dedicated 15 pin D-Sub connector for every experiment, it provides 28V battery power, an up- and downlink called telecommand and telemetry (TM) respectively, battery charging capability, and three binary signaling lines. Furthermore, it contains the processor boards monitoring "housekeeping", flight and environmental parameteres and controlling most on-board systems like the motor ignition unit, seperation unit and Yo-Yo system.

The experiments are provided with a normally 28 V 1 A direct current (DC) power supply, when connected to the RXSM. On ground an umbilical cable connects an external regulated supply unit to the on-board power consumers. This cable will be disconnected with lift-off, causing the RXSM to switch over to an internal battery. Every experiments power supply is protected with two fuses, whereas the first protects against high short-term overcurrents, and the other one protects against low long-term overcurrents. Those fuses may be remotely reset before launch. The power can be switched on and off before lift-off manually or automatically based on a timeline and is usually cut 600 s into the flight, before impact.

Data transfers from the experiments to their respective ground stations can be realized using the existing pulse code modulated (PCM) or umbilical telemetry (TM) link between the RXSM and a ground transceiving system. For that, experimenteers are responsible for formatting, failure recognition and correction of their TM stream. To avoid bandwidth overload and buffer overflows, it is recommended to no exceed 30 kbit/s and send packages not greater than 64 B. A universal asynchronous receiver transmitter (UART) powered RS-422 interface is used to connect the experiments with the TM subsystem of the RXSM. On ground, a RS-232 or a science-net port is provided to interface with the experiments ground stations.

Three unidirectional, binary signaling lines between the RXSM and an experiment can be used to implement basic control features. Those lines are named Start / Stop of Data Storage (SODS), Start / Stop of Experiment (SOE) and Lift-Off (LO). They are implemented as open collector outputs against the power supply ground, with a low impedance being an active signal. SODS and SOE are unique for every experiment and are timeline programmable or may be asserted manually. Their usage is therefore not restricted to the functionalities implied by their names. LO however is a single output connected to all experiments and will be activated once the umbilical cable is removed from the RXSM upon lift-off. [8]

2.2 Sensorplatform

With the HERMESS experiment module the loads on the rocket structure and the temperatures occuring during flight are supposed to be measured. The sensors for that porpuse are grouped in six sets of each two biaxial dual strain gauge rosettes (SGRs) and a single temperature sensor, also called a resistive temperature detector (RTD). A single set of these three vertically stacked sensors with an adjacent soldering terminal for cable relief are defined as a Strain and Temperature Applied Measurement Point (STAMP) and is shown in figure 2.1. The six STAMPs are glued to the inside of the outer hull structure with a seperation of approximately 60° with slight deviations due to known non-uniformities like hatches or drilling holes. This results in a total of 12 SGRs and 6 RTDs for the sensor platform to be acquired. [15]

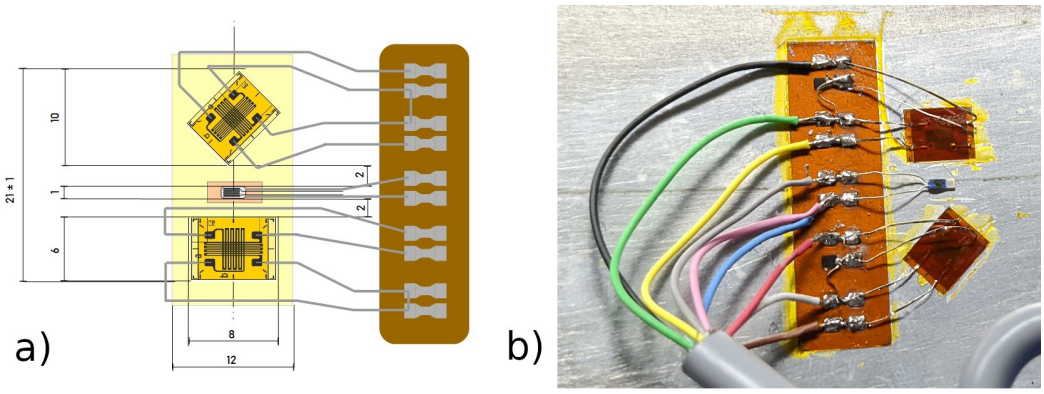


Figure 2.1: A single STAMP with its three sensors and the soldering terminal. a) overview [15] b) soldered STAMP

Both types of sensors are electroresistive, thus changing their electric resistance in a known relation to the physical property of interest. The following equations describe these relationships under the assumption for linearity within the measuring range for SGRs and RTDs.

The relative change of resistance and the absolute resistance for SGRs are expressed by equations 2.1 and 2.2 respectively, where
- R the resistance of a single strain gauge is,
- k the constant of proportionality or gauge factor is,
- ΔR the change of resistance is,
- R_0 the resistance of a single strain gauge in an unstrained state is and
- ϵ the strain is. [11]

"1-XY93-3/120 DMS" by Hottinger, Brüel & Kjær (HBK) (renamed in 2020 from Hottnger Baldwin Messtechnik GmbH (HBM)) are being utilized for the STAMPs having a gauge factor k of approximately 2 and a base resistance R_0 of $120\,\Omega$. ([15], [10])

$$\frac{\Delta R}{R_0} = k \times \epsilon \tag{2.1}$$

$$R = R_0 \cdot (1 + k \times \epsilon) \tag{2.2}$$

Furthermore, the absolute resistance of an RTD can be calculated using equation 2.3, where
- R the resistance of the RTD is,
- ΔR the resistance change per +1 K temperature change is,
- R_0 the resistance of the RTD at 0 °C is and
- T the temperature in °C is. [16]

The device selected is an "M 310 A PT100" by Heraeus. As it is typical for PT100 RTDs, it has an R_0 of 100 Ω and a ΔR of 0.385 Ω/K. ([15], [12])

$$R = R_0 + (\Delta R \cdot T) \tag{2.3}$$

It was determined that a halfbridge configuration for the strain gauge rosettes is required to mitigate the effects of thermal expansion. Using this setup the similar thermal effects on both orthogonally stacked strain gauges cancel each other out. [7]

2.3 Differential Delta-Sigma analog to digital conversion

Delta-Sigma or Sigma-Delta analog to digital converters (ADCs) with $N > 1$ output bits are cheaper but also more difficult to understand than classic successive-approximaion ADCs. Because of reasons described in chapter 4.1, the implementation of temperature and strain measurements rely on Delta-Sigma ADCs, to which may be referred as just "ADCs" hereinafter. This section further includes descriptions of features which are not part of the modulator performing the actual conversion, but are commonly included in integrated circuits housing such ADCs.

Both measurement setups used in this thesis are differential, meaning that the voltage U_{IN} between the two input voltages U_{INP} and U_{INN} is to be measured. In fully differential setups, the sum of U_{INP} and U_{INN} is approximately constant, thus the common mode voltage U_{CM} as defined in equation 2.4 defines a level, to which the relative input signals sum is 0. Another possible mode of operation is pseudo-differentially, where both input voltages may float independently from each other,

resulting in U_{CM} not being constant. [16]

$$\begin{aligned} U_{IN} &= U_{INP} - U_{INN} \\ U_{CM} &= \frac{1}{2}(U_{INP} + U_{INN}) \end{aligned} \tag{2.4}$$

Figure 2.2 and following enumeration describe typical Delta-Sigma ADC components.

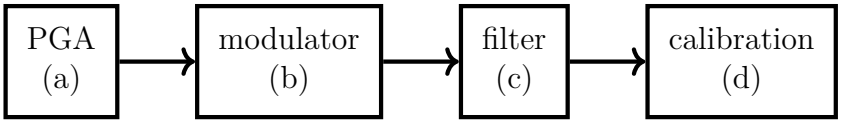


Figure 2.2: Typical signal flow in a Delta-Sigma ADC

(a) A **programmable gain amplifier (PGA)** is used to amplify the analog input signal to a differential voltage more suitable for measurement relative to the selected reference voltage U_{REF}. As the name suggests, the amplification factor $G \in \{2^x | x \in \mathbb{N} \wedge 0 \leq x < 8\}$ can be set programmatically via an internal variable resistor and does not require specific external hardware configurations. This allows modification of the input range across which the measurements can be performed, the full scale range (FSR). Modifying the FSR consequently changes the minimally quantizable input voltage difference, also described as least significant bit (LSB). Equation 2.5 shows the influence of PGAs on converter outputs. [16]

$$\begin{aligned} FSR &= \pm U_{REF} \,/\, G \\ LSB &= FSR \,/\, 2^N \end{aligned} \tag{2.5}$$

To prevent nonlinear behaviour of the PGA, its amplification factor must meet device specific requirements. The TI ADS114x series used in this thesis prohibits the PGAs output to swing closer than 100 mV to the analog power supply voltages U_{AP} and U_{AN}. Therefore the requirements expressed by equation 2.6 must be considered when selecting G. [16]

$$\begin{aligned} U_{CM(MIN)} &\geq U_{AN} + 0.1V + \frac{G \times U_{IN(MAX)}}{2} \\ U_{CM(MAX)} &\leq U_{AP} - 0.1V - \frac{G \times U_{IN(MAX)}}{2} \end{aligned} \tag{2.6}$$

(b) The base of a Delta-Sigma ADC is a feedback loop, called the **modulator**, consisting of at least three operation amplifiers in different configurations, as shown in figure 2.3. This count increases with the ADC order defining the number of

consecutive difference amplifiers and integrators. Contrary to most other architectures, this type of converter has three different samplerates, namingly the signal frequency f_s, with which the input signal is being sampled one bit at a time, the output frequency f_o defining the time between 1-bit quantizations, and the data frequency f_d which is the rate determining the distance between two consecutive data ready events in continuous mode, triggered whenever a measurement has been completed. Therefore usually only f_d is of interest for a system designer, when using a commercial delta-sigma ADC. [2]

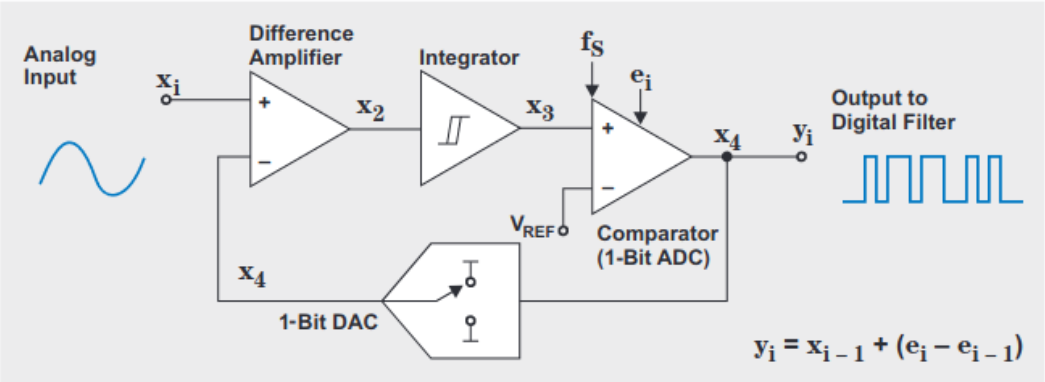


Figure 2.3: First-order Delta-Sigma modulator [2]

The positive input of the difference amplifier, described as "Analog Input" in figure 2.3, is connected to the PGA output. The binary representation of the previous 1-bit conversion, expressed through either the positive or negative reference potential is connected to the negative input. The integrator is designed to delay the signal propagation for $1/f_s$ thus implementing a weak voltage hold mechanism to prevent erroneous readings by the subsequent latched 1-Bit-ADC. The density of positive signals at its output within the time of one f_o clock cycle is proportional to the analog input voltage in relation to the reference voltage. [2]

This design allows for two techniques improving signal-to-noise-ratio (SNR) known as oversampling and noise shaping. Oversampling is achived by having $f_s > f_o$, and therefore spreading the constant quantization noise with random magnitudes in the frequency domain, introduced by the conversion of a continuous signal to a discrete one, over a wider frequency range. Noise shaping aims to distribute the quantization noise towards frequencies higher than $f_o/2$ and is integral to this ADC design. Because the introduced error e_i at f_s clock cycle i is being fed back into the sampling sequence, the subsequent quantization iterations will in average level the introduced errors for lower frequencies out and shift them towards higher frequencies. [18]

(c) A **digital filter** is required to attenuate the high frequency quantization error from the 1-bit data stream being outputted by the modulator. Typically a SINC

averaging lowpass filter is used for that porpuse. Another porpuse of the filter is the reduction of the output data rate by discarding intermediate samples not required to fulfill the Nyquist theorem based on the desired data frequency f_d. [3]

(d) Some ADCs such as the one being used for this thesis include digital **calibration** measures to mitigate the effects of differential offset, meaning a non 0 V input voltage in an "unstressed" system, and gain errors in the signal path. In both cases simple digital arithmetic operations with previously recorded values as second operands are performed on the original output data before they are being made available to the measurement consumer. [16]

The offset calibration mechanism subtracts a value stored in a dedicated offset calibration register from the unaltered ADC reading. This register can be set by either running a system offset calibration, where both internal and external offset for a system at "unstressed" conditions is being measured, a self offset calibration, where only the internal offset is being recorded or by manually writing to the register directly. [16]

To set the full-scale calibration register used for system gain calibration by multiplying its value with the offset calibrated reading it is recommended to apply the maximally desired input values and issue the system gain calibration command. However it is possible to write the register content directly. [16]

3 | Requirements and Concept

The requirements that need to be fulfilled by the HERMESS experiment as well as the concept to be implemented in scope of this thesis are desribed in this chapter. The system for measurement of strains and temperatures must be compatible with prerequisites outlined in the RXSM user manual [8] and SED guidelines [21].

Even though the system developed is a piece of space technology in scope of major european agencies, only a selected subset of European Cooperation for Space Standardization (ECSS) standards have to be met. Critical subsystems like for example pyrotechnics, free falling units or batteries are therefore either vetted especially diligent by an experts committee during the reviews or outsourced to verified systems like the RXSM. No such especially critical parts are in use for this experiment leaving a greater degree of freedom for conceptualizing and implementation.

Approval for the change of concept was obtained in the beginning of April 2021 from a responsible ZARM supervisor.

3.1 Requirements

Following tables list the requirements worked out for the HERMESS SED. All items are categorized into one of the following categories in compliance with ECSS-E-ST-10-06C "Space engineering Technical requirements specification" [25] with modifications imposed by the RXBX SED guidelines [21] to account for the limited scope of a high-altitude research mission. The columns "Sec" of these tables indicate the sections, in which the requirement is mentioned for fulfillment. Further explanations and rationales for all items can be found in the SED [15] or the REXUS user manual [8].

Table 3.1 lists functional requirements (F) defining the systems functionalities to conform mission needs. These items are referenced to with "FX", where X denotes the respective incremental requirement ID.

ID	Description
F1	The experiment shall measure strain signals at several locations

ID	Description
F2	The experiment shall measure temperature at several locations

Table 3.1: Functional requirements for the HERMESS experiment from [15]

In context of RXBX, performance requirements (P) define verifiable absolute measures for ranges, resolutions and configurations of systems fulfilling the needs of functional requirements. Those for the HERMESS experiment are listed in table 3.2. These items are referenced to with "PX", where X denotes the respective incremental requirement ID.

ID	Sec	Description
P1	4.1.2	The range for the strain gauge measurements shall be up to 2 % relative strain
P2	4.1.2	The accuracy for the strain gauge measurements shall be 0.001 % relative strain
P3	4.3	The sample rate for the strain gauge measurements shall be minimum 1 kHz
P4	4.1.1	The range for the temperature measurements shall be −40 °C to +200 °C
P5	4.1.1	The accuracy for the temperature measurements shall be ±1 K
P6	4.3	The sample rate for the temperature measurements shall be minimum 10 Hz
P7	3.2	The experiment shall measure 6 × strain signals due to longitudinal loading and 6 × strain signals due to shear loading
P8	3.2	The experiment shall measure temperature at 6 positions

Table 3.2: Performance requirements for the HERMESS experiment from [15]

In context of RXBX, design requirements (D) include environmental, physical, interface, mission, configuration and as the name suggests, design requirements as defined in ECSS-E-ST-10-06C and are listed in table 3.3. These items are referenced to with "DX", where X denotes the respective incremental requirement ID.

ID	Sec	Description
D1	4.1	The experiment shall withstand the temperature profile of the REXUS rocket
D2	4.1	The experiment shall withstand the pressure profile of the REXUS rocket
D3	4.1	The experiment shall withstand the vibration profile of the REXUS rocket

ID	Sec	Description
D4	4.1	The experiment shall withstand the shock profile of the REXUS rocket
D5	4.1	The experiment shall properly function under the influence of expectable electromagnetic interference
D6	5	The mass of the experiment module must not exceed 7.5 kg
D7	3.2	The dimensions of the experiment shall comply with the 120 mm rocket ring module
D8	4.1	The mean power consumption of the experiment shall not exceed 28 W
D9	4.1	The peak power consumption of the experiment shall not exceed 80 W
D10	4.1	The input voltage range of the experiment shall include 24 V to 36 V
D11	4.3	The telemetry data throughput shall not exceed 30 kbit/s
D12	3.1	The experiment shall comply with the request of radio silence

Table 3.3: Design requirements for the HERMESS experiment from [15]

For further elaboration, some deductions are made for use in this thesis based on simplifications of requirements D1 - D5 and D12.

1. Based on D1 and given that the storage temperature is not expected to exceed the limits measured by previous flights, shall the experiment be operational in the temperature range between −40 °C and +80 °C.

2. The pressure profile mentioned in D2 is not expected to have variations from the surrounding air pressure during flight. Therefore the experiment shall be operational in the pressure range between 10 mbar and 1 bar.

3. During launch a typical acceleration of 20 g acts on the longitudinal axis of the Improved Orion. However during re-entry decelerations of 20 g may occur along all axes, resulting in the derivation of D4 stating, that the experiment shall be operational with 20 g acceleration forces in all axes.

4. The best practices in the REXUS user manual dictate that data and power cables shall be twisted and housings as well as adjacent connectors shall have a shielding of 20 dB to reduce electromagnetic interference (EMI) influence and may be sufficient to fulfill D5.

5. Radio silence as required by D12 makes it mandatory to implement fail-safe mechanisms to prevent any current flow inside the experiment, once radio silence or payload off is commanded. Since this experiment does not use power sources

other than the RXSM and no large capacitors, no special precaution must be taken. [8]

Operational requirements (O) define the operation, control and reaction triggering events of functional units. Only one item was identified for the HERMESS experiment and is listed in table 3.4. This item is referenced to with "O1".

ID	Sec	Description
O1	3.2	The experiment shall operate fully autonomous after launch

Table 3.4: Operational requirements for the HERMESS experiment from [15]

Because of limited bandwidth for the up- and downlink capability during flight, O1 must be satisfied. This experiment therefore requires a storage capacity for the duration from shortly before lift-off until power off approximately 600 seconds into the flight, before impact. Additionally the system must react to time programmable events triggered by the RXSM via three control lines. Those may be used to initiate data storage shortly prior to lift-off and safely shutdown before the power cut. [8]

3.2 Concept

This concept describes the components of the HERMESS experiment and their interactions with each other based on the given requirements. Experience with previous SPU concepts ([13], [14]) heavily utilizing computational redundancy led to the realization, that these systems with a theoretically increased failure tolerance, pose many hard to overcome difficulties during implementation. For example, to implement true redundancy with the ability for every parallel branch to handle undesired behaviour differently, every branch should utilize unique chipsets and software. Since some interfaces like the RXSM connection cannot be provided redundantly, it is not expected, that a highly redundant system offers enough advantages for the failure tolerance to justify the required resource consumption in terms of development time and payload mass. The SPU concept proposed here aims therefore to be slim and not exceed the requirements, if doing so introduces additional complexity.

Figure 3.1 depicts the slim concept proposed for this thesis. The experiment system uses a standardized 14″ wide, 4 mm thick aluminium cylinder with radial-axial joints on top and bottom for compliance with D7. To the cylinders will be referred as "module rings" hereinafter. They form the outer hull of the rocket, with exception to the motor and nosecone section. For a typical REXUS rocket, the payload section is made up of four to six stacked module rings and a nosecone, each containing an experiment or flight hardware.

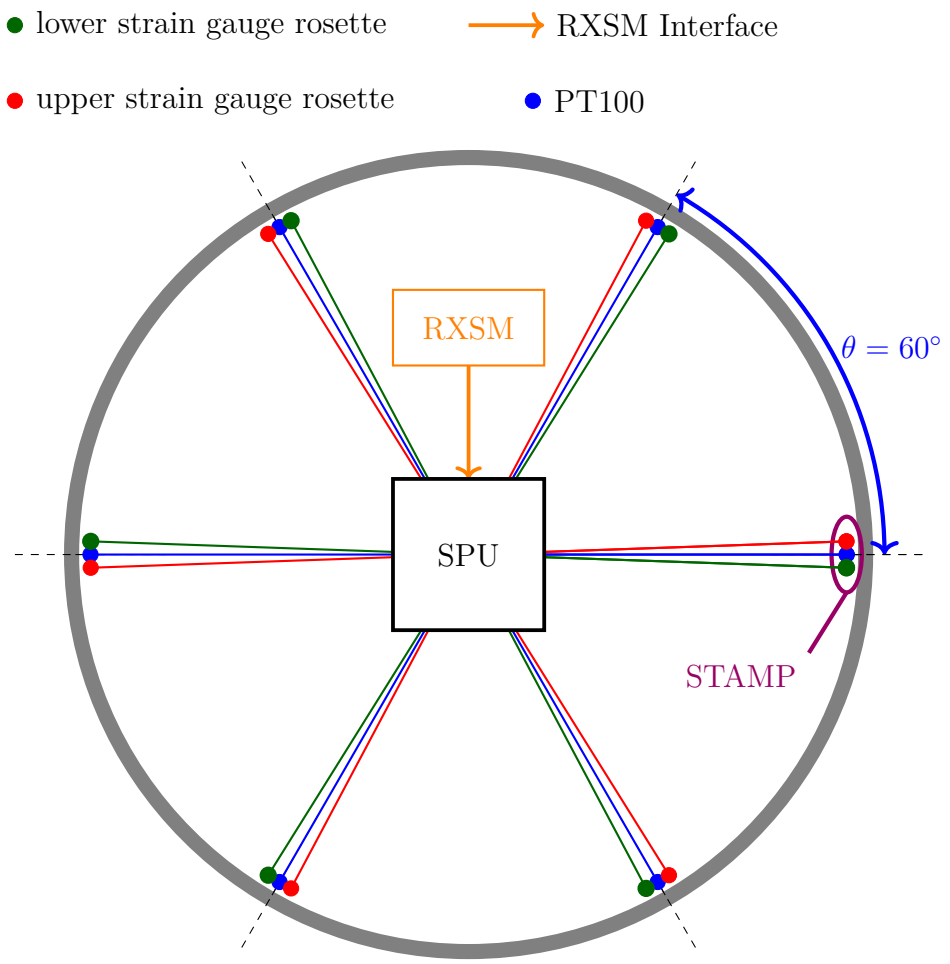


Figure 3.1: HERMESS experiment base components in flight configuration, based on [15]

The sensorplatform as described in section 2.2 consists of six approximately evenly distributed "Strain and Temperature Applied Measurement Point" (STAMPs) glued to the inside surface of the module ring. With the upper strain gauge rosettes (SGRs) aligned for longitudinal loading, and the lower SGRs for shear loading, is P7 satisfied. Furthermore, P8 is fulfilled with each STAMP having a PT100 resistive temperature detector (RTD).

To limit system complexity, all electrically active components are encompassed by the Signal Processing Unit (SPU). This concludes, that the STAMPs and intermediate components only feature passive elements and must be connected to the SPU. This concept proposes six multi-wire cables, each connecting a single STAMP with the SPU.

The SPU is primarily responsible for performing measurements on the connected STAMPs, storing these data on internal memory, and providing communication links for data and status retrieval as well as for configuration, controlling and programming

porpuses. In flight or flight simulation setup, the RXSM or an appropriate simulator is connected to the SPU providing it with telemetry (TM) capabilities and the three signaling lines as described in section 2.1. Otherwise the Data- and Programming Interface (DAPI), a USB-interface for use on ground, may be used. In any case however, the power must be supplied via the RXSM connector, eliminating the need for a dual input power distribution system.

A dual-SPU-concept proposed for the current SED version [15] features two almost identical interconnected SPUs, with each having three STAMP connections and an STM32 based microcontroller. Programming efforts yielded the fact, that this architecture posed timing issues caused by the need to handle a high frequency of interrupts triggered by the SGR-ADCs in an almost serial matter. This effect would have worsened, when doubling the number of SGR-ADCs for a single-SPU-concept as proposed for this thesis. Therefore, to parallel time consuming operations, a Smartfusion2 SoC comprising an field programmable gate array (FPGA) and a Cortex-M3 microcontroller is used as the main processing chip.

The FPGA portion is used as shown in figure 3.2 for standard bus operations like reading out ADC values and aggregating those into data packages for storage. Furthermore it is capable of basic error detection and recovery of non-responsive or asynchronous SGR-ADCs.

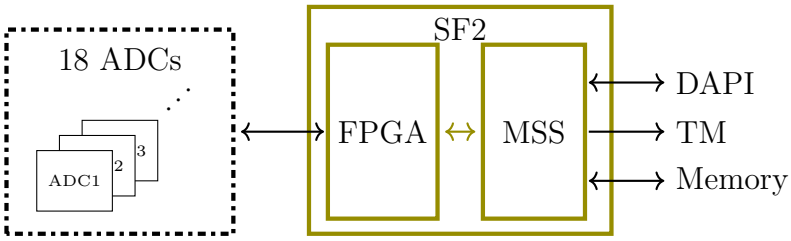


Figure 3.2: Interactions of SF2 interactions with ADCs, memory and interfaces

The microcontroller subsystem (MSS) is programmed to handle all other communications. Namely this includes the TM link, the three signaling lines provided through the RXSM, the DAPI and the SPU internal flash storage. For every ADC reading cycle it is interrupted and provided with a data package, which does not require further processing prior to permanent storage on flash for later post-measurement retrieval.

The limited TM downstream bandwidth of 30 kBit/s falls short of the expected data acquisition volume, thus only a fraction of the gathered data can be transmitted in flight. The TM link is primarily used for status information like performance stats, signaling events or failures.

As described in section 2.1, three programmable signaling lines called LO, SODS and SOE can be utilized for basic experiment control. For the HERMESS experiment, those are used to satisfy the O1 requirement. Is either SODS or SOE active for longer than one second, the SPU will start saving measurement values to the internal storage

for a preconfigured minimal timespan. This is not true, when either signal is active on SPU startup, in which case, data storage is continued without the one second delay. Data storage will be arrested, once both signals were inactive for at least one second. Any activity on the LO signal line will only be logged.

The DAPI is the most sophisticated electronic interface of this experiment. It connects a computer to the SPU for programming of the System on Chip (SoC), configuring the experiments parameters, performing and managing ADC calibrations, and finally governing storage features like read-outs and clearings. On the computer side, all actions are either performed using third party softwares or the HERMESS ground station software.

4 | Implementation

This chapter depicts the implementation of the concept as described in section 3.2 and is divided into four sections for that porpuse. Each section corresponds to its own technological level, starting with the hardware layout, through the SoC FPGA and MSS programming, ending with the development of the HERMESS ground station software. They mostly contain detailed descriptions of their respective final architectures and their driving key design considerations.

Most unsuccessful or more accurately "intermediate" implementations will not be mentioned as part of this thesis, even though they contributed significantly towards the insights required for the current state of the implementation.

4.1 Hardware layout

The electronic hardware of a Signal Processing Unit is only a single assembled printed circuit board (PCB). A PCB basically only interconnects the various integrated circuits (ICs), sockets and basic electrical elements in a useful way utilizing thin copper wires etched on one or more layers of otherwise non conductive panels. For this project, it was designed in three stages using *Autodesks Eagle* software. First, a schematic was created connecting the components with each other. Then the PCB outline was drawn by applying outer barriers, placing holes and defining special areas, where for example copper may not be put. Finally, the components were virtually placed and airwires routed until all connections are made. This design was then sent for manufacturing to an external fabricator.

As outlined in section 3.2 above, the main components of an SPU are a SmartFusion2 (SF2) SoC, a number of ADCs and a flash storage. Figure 4.1 provides a more detailed overview of the components and their connections. It is apparent, that the SF2 is the central device for all digital systems on the SPU.

Six units labeled "ST1" through "ST6", consisting of each two ADCs for acquiring measurements of the strain gauge rosettes (SGRs) and one ADC for measuring the temperatures via an resistive temperature detector (RTD) are connected to their

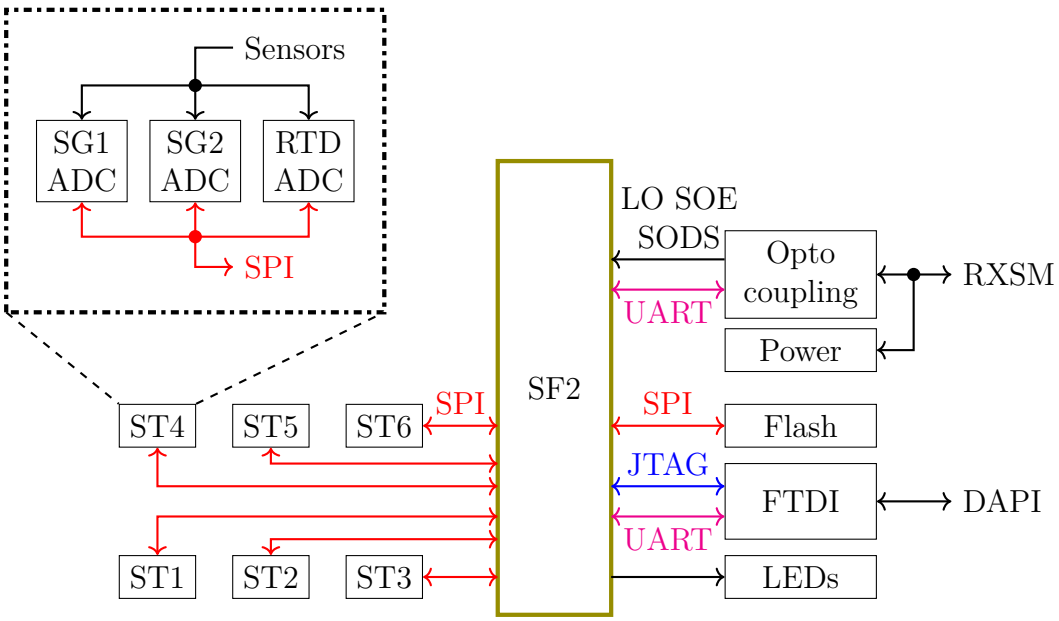


Figure 4.1: Overview of components inside the SPU

respective STAMPs outside the SPU. This concludes, that there are in total 18 dedicated ADCs present on an SPU. Internally, each unit is connected via a seperate serial peripheral interface (SPI) bus to the SF2. This allows for parallel input / output (I/O) operations among the units, countering the negative effects of long I/O phases, which, as briefly explained in section 3.2, rendered a previous dual-SPU-design unfeasable.

Prior to designing the first iteration of the SPU PCB schematics and layout, a prototype was build as a proof of concept. As observable in figure 4.2 depicting the prototype board, its main component is a *Trenz SMF2000* evaluation board for the SmartFusion2 SoC. Three ADCs with their respective circuitry are connected to the evaluation board. Two of them are set up to measure strains from each a strain gauge rosette (SGR) and the remaining ADC is set up to measure temperature via a connectable PT-100 resistive temperature detector (RTD). Together they form the measurement circuitry for a single STAMP. Not shown in figure 4.2 is a breakout board with the memory chip connected via a flat ribbon cable. All major IC families used for the prototype were later utilized for the actual SPU hardware. This approach allowed for programming efforts to start very early with results, that may be used with only slight alterations later on for the productive SPU. It furthermore helped identifying and correcting design flaws in an accelerated manner, because no time-consuming PCB remanufacturing was necessary.

Some special considerations were to be made in regards for the design requirements listed in section 3.1 during the process of schematic generation. To cope with D1, only components rated for industrial use typically operational in the required temperature range were to be used. Because the PCB surfaces will directly encounter the severely

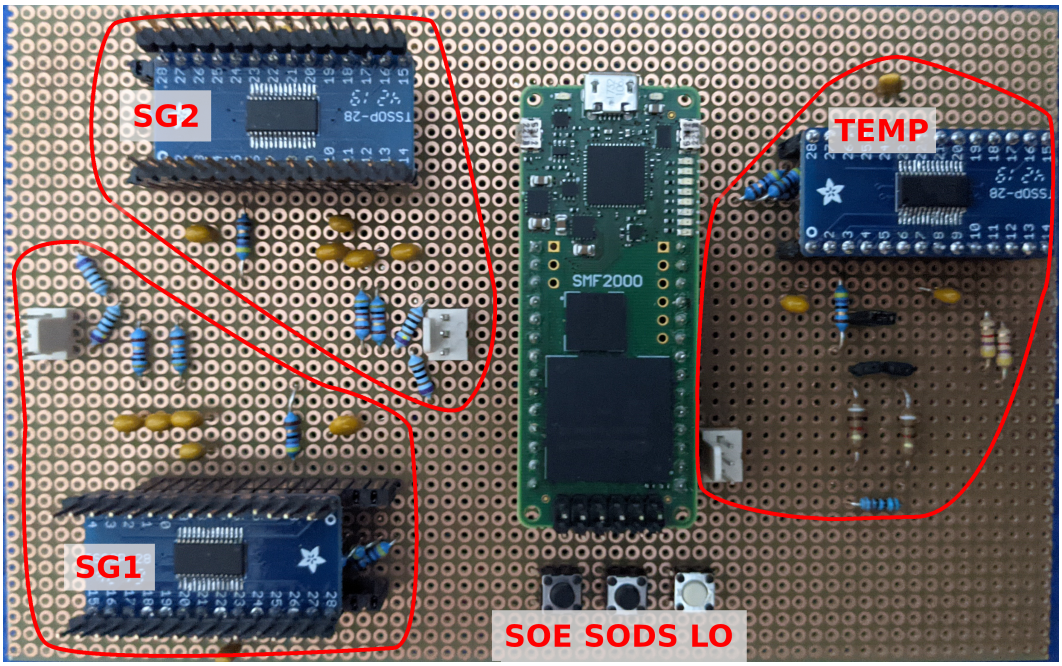


Figure 4.2: Prototype board for reading out a single STAMP

limited atmospheric pressure, no electrolytic capacitors, except the tantalum based, may be used in order to satisfy D2. D3 and D4 dictate the use of only vibration and mechanically shock resistant connector systems and clock generators, as typical quartz oscillators would be out of their operational limits.

The *M2S010-TQG144I* from the SmartFusion2 (SF2) SoC product family was chosen as the main processing unit. Accomodating approximately 10,000 logic elements, its FPGA is the largest one of the product family still having a 144 pin count thin quad flat pack rectangular package (TQ144) for easier assembly in comparison to ball grid arrays. To not be forced to invest in expensive software licenses, it was important to chose a SoC with only up to 25,000 logic elements [19]. The selected IC also satisfies the requirements for use in the industrial temperature range. It is RoHS compliant and comes with a lead-free package, which was the only available option at the time of purchase. Coincidentally, the SF2 embedded in the evaluation board is an *M2S010* as well [9], even further reducing necessary efforts when porting software initially build for the prototype to the productive hardware. Integrated, programmable pull-up and pull-down resistors in the I/O banks help reduce the amount of surface mounted components and therefore design complexity. [20]

The electrical power is provided by the RXSM through a D-Sub15 connector. An isolating *Traco THN 10-2411WIR* DC/DC converter with an input voltage from 9 V to 36 V directly connected to the power supply, outputs 5 V for further conversion. This converter with its large input voltage range satisfies D10. Because it is the only power

consumer or distributor directly connected to the RXSM, its maximum output current of 2 A at 5 V can be utilized to satisfy the requirements D8 and D9 regarding the mean and maximum power consumption of 28 W and 80 W respectively. This is possible, because the converter can not provide more than $2\,\mathrm{A} \times 5\,\mathrm{V} = 10\,\mathrm{W}$ for a time longer than 250 µs [1]. Two linear voltage regulators each provide a 3.3 V power line, with one of them powering only the SF2 internal logic, and the other one supplying all remaining ICs. In the first PCB iteration, four out of eight light emitting diodes (LEDs) indicate power availability on the various voltage lines. The others are connected to the SF2 and are therefore either controllable by the FPGA or programmable via firmware executed by the microcontroller subsystem (MSS).

According to the REXUS user manual [8], not only the three signaling lines shall be opto-isolated, but the full duplex telemetry / telecommand interface as well. The latter are connected via a *TI MAX488* driver to the interface connector.

No opto-coupling is required for the Data- and Programming Interface (DAPI) micro USB port, thus it is omitted. A *FTDI 2232D* IC provides easy to implement USB client functionality, as no low level host drivers need to be developed. This particular chip requires a connected non volatile memory unit to store its configuration containing identification numbers, modes for the output pins, driver modes and more. These are necessary to be visible simultaneously as a full duplex UART interface and an embedded SF2 programmer and debugger. This configuration can be programmed by any host running a FTDI issued, freely available software. Fortunately, the evaluation board used in the prototype utilized a similar FTDI chip with compatible configuration attributes, allowing for a readout and reuse for the SPU.

The schematics for the first iteration of the PCB are attached in appendix A. Unfortunately, the used software did not allow to combine hierarchical design with design blocks to limit repitions in the schematic representation introduced by six functionally equivalent STAMP units.

After finishing the schematics, the next step in creating the PCB layout was to define the outlines, namely the dimensions and mounting holes. Instead of placing the components first and then defining the outline, it was perceived faster to just fix it to a specific value once in the beginning. A base for a good guess provided the discarded dual-SPU-PCB layout only containing half of the required ADCs. Doubling the surface area should therefore provide enough space for all ADCs and tolerance, because subsystems like the ones for processing and power supply would remain similar in size. Thus, the edge lenghts were multiplied with $\sqrt{2}$ and rounded up resulting in $160\,\mathrm{mm} \times 130\,\mathrm{mm}$, which still ensures enough space around the SPU inside the HERMESS experiment module. Three metric M5 holes for a case mounting were placed evenly distributed on each side, whereas every edge shared the two holes located at the corners with adjacent edges.

Because no high pin density packages were used, there was no need for a PCB with

more than four layers. In Eagle, polygons named after signals in combination with the ratsnest command may be used to copperize planes, connecting to all pads with the same signal name and excluding all others. On the top and bottom layer this was used for the 3.3 V supply. One of the inner layers is a dedicated ground plane. Appendix B shows an overview of the PCB layout. Digital design files are linked in appendix C.

Figure 4.3 depicts a fully assembled SPU in its first iteration of this design. First, solderpaste was applied via a stencil to the top side, so that the components could be placed manually and soldered using a reflow oven. Another stencil was used for applying solderpaste to the bottom side. Because a second reflow cycle with the PCB being upside-down would have released the already mounted components, the bottom side had to be soldered manually under a microscope. Lastly, through-hole elements were attached.

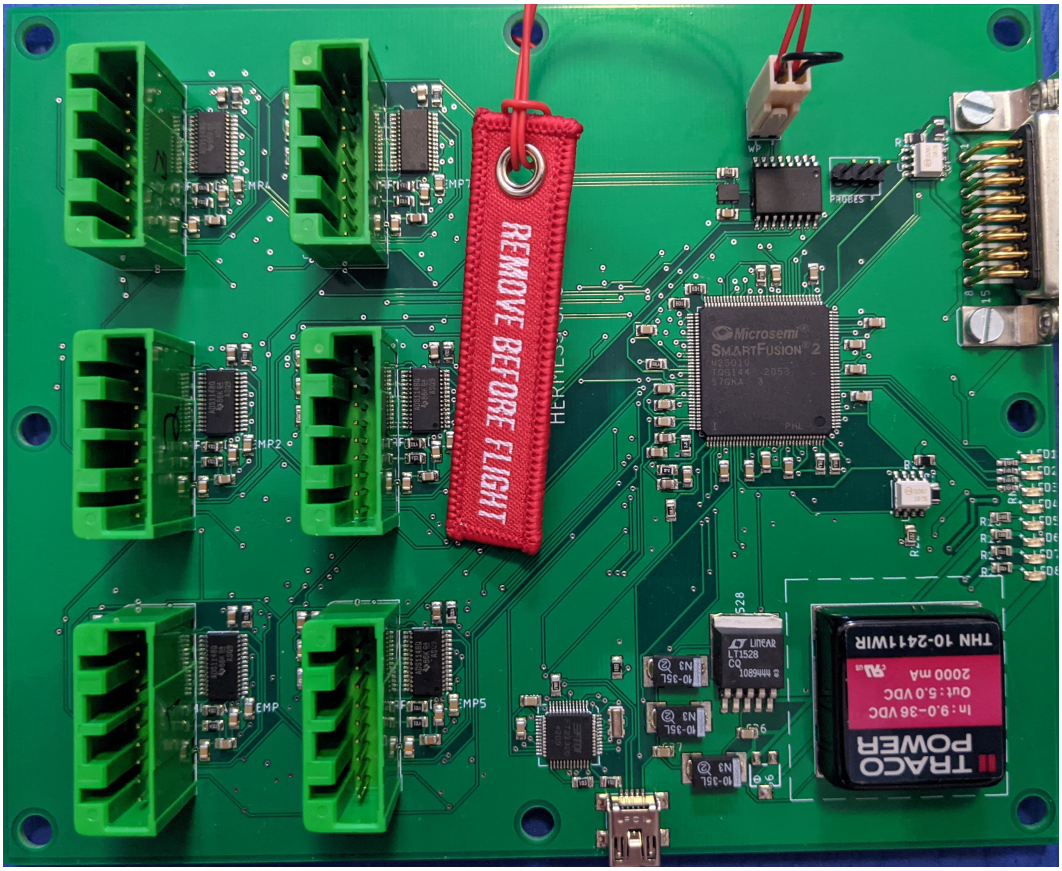


Figure 4.3: Assembled Signal Processing Unit with "REMOVE BEFORE FLIGHT" flash write protection

Protection against electromagnetic interference (EMI) according to D5 is supposed to be achieved by differential RC-filters with 3 dB attenuation at 2 kHz, shielded cabling and ferrite beads on the sensor end and the PCB. A full custom metal casing with

venting holes is expected to further help EMI resilience. Proper testing to verify this assumption however is still pending and is scheduled to take place during an upcoming review [8].

4.1.1 Temperature measurement

The utilized Delta-Sigma analog to digital converter (ADC), a *TI ADS1148*, provides recommendations for ratiometric three wire RTD measurements in its datasheet which were followed for the six temperature measuremets setups. This technique allows for readings with high accuracy and reduced influence by lead wire resistance. [16]

For ratiometric measurements, the value of interest val is derived by measuring the voltage drop U_{RTD} of the sensor R_{RTD} itself relative to a reference voltage U_{ref} along a resistor R_{ref}. This resistor is connected on one end to the resistive sensor and the other end to system ground. A constant current I_{const} is applied to the sensors lead, not connected to R_{ref} but the ADC. Equation 4.1 shows how this setup is independent of the value and therefore accuracy of I_{const}, but is only sensitive to resistances. [16]

$$\begin{aligned} val_{rm1} &= U_{RTD}/U_{ref} \\ &= (R_{RTD} \times I_{const}) \ / \ (R_{ref} \times I_{const}) \\ &= R_{RTD}/R_{ref} \end{aligned} \tag{4.1}$$

Equation 4.2 further details the measurement by taking the wire resistances introdocued by cabling connecting the SPU with a RTD into account. Figure 4.4 shows the setup for this equation. All three wires leading to R_{RTD} feature a similar wire resistance R_l and a second constant current source with I_{const} feeding the other RTD - ADC connection was introduced. Considering Kirchhoff's second law, the voltage drop along this connection must be attributed negatively. Note, how the lead wire resistance between R_{RTD} and R_{ref} has no influence in the output value. [16]

$$\begin{aligned} val_{rm2} &= \frac{(R_{RTD} + R_l - R_l) \times I_{const}}{R_{ref} \times 2 \times I_{const}} \\ &= \frac{R_{RTD}}{2 \times R_{ref}} \end{aligned} \tag{4.2}$$

This setup, except for added differential filters at the ADC inputs, is used for RTD measurements on the SPU and can be evaluated in detail by reviewing the bottom left sections on sheets three through eight in appendix A.

Two programmable matched excitation current sources are integrated into the used ADC IC and may be set to currents in $\{50, 100, 250, 500, 750, 1000, 1500\} \times \mu A$. Generally it is recommended to select a current as high as possible, while being low enough to

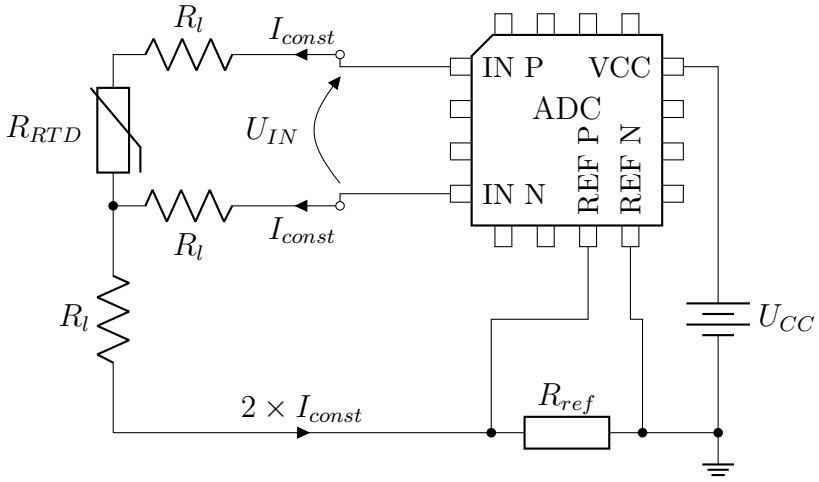


Figure 4.4: Basic principle of ratiometric measurements used for RTD sensors

cause only neglectable RTD self heating. For the selected PT-100 sensor, self heating is specified with 0.4 K/mW [12]. Equation 4.3 shows, how a worst case self heating when measuring with maximum excitation current and maximum temperature of 200 °C would still be within the ±1 K resolution requirement P5.

$$\begin{aligned} deviation &= self\,heating \times R_{RTD\_200\,°C} \times I_{const}^2 \\ &= \frac{0.4\,\mathrm{K}}{\mathrm{mW}} \times (100\,\Omega + (\frac{0.385\,\Omega}{\mathrm{K}} \times 200\,°\mathrm{C})) \times 1.5\,\mathrm{mA}^2 \\ &\approx \frac{0.4\,\mathrm{K}}{\mathrm{mW}} \times 0.4\,\mathrm{mW} \\ &= 0.16\,\mathrm{K} \end{aligned} \tag{4.3}$$

However, to not unnecessarily degrade the accuracy, an I_{const} of 1 mA was selected after verifying its feasibility on the prototype. This decreased the expected maximum self heating to only about 0.07 K at a dissipation power of 177 µW. The low required samplerate of 10 Hz in contrast to the high sampling speed allows for current cut-offs in between readings, further reducing heat generation on the sensor side at the cost of increased software complexity. Ultimately, this scheme was not implemented, because the downsides were estimated to outweigh the advantages.

It is possible to programmatically map most analog I/O pins to the ADC internals, enabling a technique called chopping, where the two current source outputs are swapped between each two consecutive measurements [16]. This should help reduce an inaccuracy introduced through the assumption of equal constant current supplies for equation 4.2. Here, another theoretical worst case analysis shows, that chopping is not necessary to stay within the resolution requirement. A typical absolute mismatch between the sources is specified with ±0.2 %. Absolute errors of up to 6 % and

temperature drifts are not relevant for the measurement, as both supplies are effected evenly. Because variances in wire resistances induced by for example soldered connectors or slightly different lengths are not within this range of accuracy, optimization with chopping would be inappropriate for this setup.

It is recommended to select a R_{ref} for the reference voltage V_{ref} to be near mid supply. As most ICs on the SPU, the ADCs are powered with $U_{CC} = 3.3\,\mathrm{V}$. Equation 4.4 shows how a value of $820\,\Omega$ was selected for the reference resistor resulting in $V_{ref} = 1.65\,\mathrm{V}$. The ADC manufacturer recommends using a device with $0.01\,\%$ tolerance, because any variances directly influence the measurement result as shown in equation 4.2.

$$\begin{aligned} R_{ref} \times 2 \times I_{const} &= U_{CC} \;/\; 2 \\ \Rightarrow R_{ref} &= \frac{3.3\,\mathrm{V}}{2 \times 2 \times 1\,\mathrm{mA}} \\ &= 820\,\Omega \end{aligned} \tag{4.4}$$

An appropriate programmable gain amplifier (PGA) setting must be found to meet the performance requirements P4 and P5 dictating a measurement range $-40\,^{\circ}\mathrm{C}$ to $+200\,^{\circ}\mathrm{C}$ and resolution of $\pm 1\,\mathrm{K}$ respectively. It was deemed, that a higher resolution would be more advantagous than a wider input range, thus the gain setting will be derived using equations 2.3 and 2.5 matching the full scale range (FSR) to expectable values of U_{RTD}. Equation 4.5 shows, that an optimal PGA setting would be 9.32.

$$\begin{aligned} G &= U_{ref} \;/\; U_{RTD(max)} \\ &= 1.65\,\mathrm{V} \;/\; 100\,\Omega + (200\,\mathrm{K} \times 0.385\frac{\Omega}{\mathrm{K}}) \times 1\,\mathrm{mA} \\ &\approx 9.32 \end{aligned} \tag{4.5}$$

To not potentially force U_{IN} outside its allowable range, an equal or the next greater available PGA setting should be selected. Thus, a value of 16 is chosen after verification of the requirements expressed by equation 2.6.

The measurement is performed pseudo differentially with a 16-bit biploar ADC in a guaranteed positive range. Therefore only 15 bit of actual measurement data are in use. Equation 4.6 shows the temperature translation function $ttf(out)$ for converting the output code *out* to its represented temperature in °C.

$$
\begin{aligned}
out = 2^{15} \times G \times val_{rm2} &= \frac{2^{14} \times G \times R_{RTD}}{R_{REF}} \\
\Rightarrow R_{RTD} &= \frac{R_{REF} \times out}{2^{14} \times G} \\
\Rightarrow ttf(out) &= \left(\frac{R_{REF} \times out}{2^{14} \times G} - R_0 \right) \Big/ \Delta R \\
&= \left(\frac{820\,\Omega \times out}{2^{14} \times 16} - 100\,\Omega \right) \Big/ 0.385\,\frac{\Omega}{\mathrm{K}}
\end{aligned}
\tag{4.6}
$$

4.1.2 Strain measurement

Domjahn [7] determined, that a half bridge configuration as depicted in figure 4.5 shall be used for the strain gauge rosette (SGR) measuring the strains on the outer hull strucuture. A major advantage is the temperature compensation, as both strain gauges are affected similarly due to their positional proximity while still providing meaningful output.

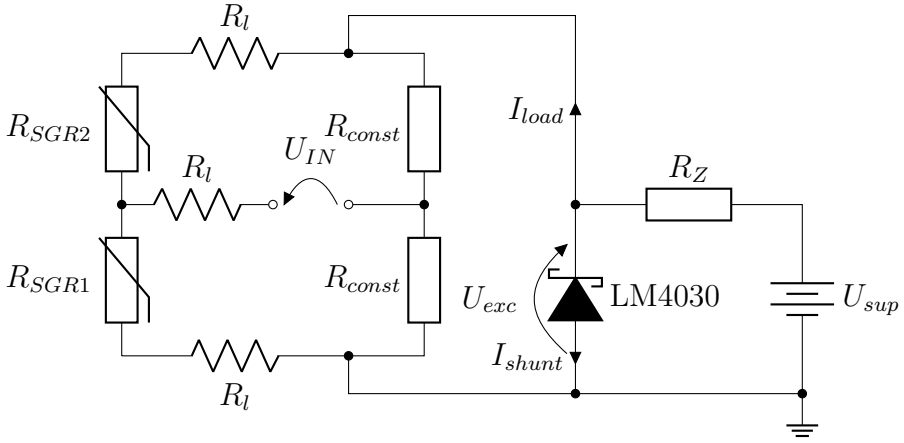


Figure 4.5: Wheatstone halfbridge used for measuring the strain gauge rosettes

Equation 4.7 shows how the voltage at the measurement point is only dependend on the relative strains ϵ_1 and ϵ_2 using the expected SGR characteristic $k = 2$.

$$
\begin{aligned}
V_{IN} &= U_{exc} \times \left(\frac{R_{SGR1}}{R_{SGR1} + R_{SGR2}} - \frac{R_{const}}{2 \times R_{const}} \right) \\
&= U_{exc} \times \left(\frac{1 + 2\epsilon_1}{2 + 2\epsilon_1 + 2\epsilon_2} - \frac{1}{2} \right)
\end{aligned}
\tag{4.7}
$$

There are different common types of connections for halfbridge setups, which are distingsuishable by the numbers of wires between sensors and measurement circuitry.

Essentially, the more wires are used, the smaller the resulting influence of lead wire resistances R_l is. In this case however, the measurement circuitry does not need to compensate for errors having the same influence during ground and flight operations. This is because the Skopinsky method used for calibration and evaluation eliminates these errors, as the values are defined by observations made while applying known loads [24]. Therefore a minimal setup with only three wires was chosen.

Any value of R_{const} is acceptable, as long as the components used have low tolerances. In this instance, 1 kΩ 0.01 % resistors are used. The maximum and minimum values for R_{SGR} are calculated in equation 4.8 according to requirement P1.

$$\begin{aligned} R_{SGR\_min} &= R_0 \times (1 + (k \times \epsilon_{min})) = 120\,\Omega \times (1 + (2 \times -0.02)) = 115.2\,\Omega \\ R_{SGR\_max} &= R_0 \times (1 + (k \times \epsilon_{max})) = 120\,\Omega \times (1 + (2 \times 0.02)) = 124.8\,\Omega \end{aligned} \tag{4.8}$$

An excitation voltage of $U_{exc} = 4.096\,\mathrm{V}$ provided by a single high precision schottky diode for all SGR bridges was chosen. This avoids any fluctuations leading to a common mode voltage $U_{CM} \neq 4.096\,\mathrm{V}/2 = 2.048\,\mathrm{V}$ and, as shown by equation 4.7 above, unreliable U_{IN} voltages. To power this diode, an already existing $U_{sup} = 5\,\mathrm{V}$ can be utilized. Considering $120\,\mu\mathrm{A} \leq I_{shunt} \leq 30\,\mathrm{mA}$ and an exaggerated range of possible load currents I_{load}, equation 4.9 shows the selection and verification for R_Z according to the LM4030 datasheet [17].

$$\begin{aligned} I_{load} &= \#\text{SGR-bridges} \times U_{exc} \times \left(\frac{1}{R_{SGR1} + R_{SGR2}} + \frac{1}{2 \times R_{const}}\right) \\ I_{load\_max} &= 12 \times 4.096\,\mathrm{V} \times \left(\frac{1}{2 \times 115.2\,\Omega} + \frac{1}{2\,\mathrm{k}\Omega}\right) \approx 238\,\mathrm{mA} \\ R_Z &= \frac{U_{sup} - U_{exc}}{120\,\mu\mathrm{A} + I_{load\_max}} = \frac{5\,\mathrm{V} - 4.096\,\mathrm{V}}{120\,\mu\mathrm{A} + 238\,\mathrm{mA}} \approx 3.8\,\Omega \end{aligned} \tag{4.9}$$

Ensuring the LM4030 is not starved, a standard resistor with a lower than calculated value of $R_Z = 3.65\,\Omega$ is selected. Furthermore, the current I_{shunt} must not exceed 30 mA continously. Equation 4.10 shows, that this is not the case.

$$\begin{aligned} I_{load\_min} &= 12 \times 4.096\,\mathrm{V} \times \left(\frac{1}{2 \times 124.8\,\Omega} + \frac{1}{2\,\mathrm{k}\Omega}\right) \approx 221.5\,\mathrm{mA} \\ I_{shunt\_max} &= \frac{U_{sup} - U_{exc}}{R_Z} - I_{load\_min} \approx 24.7\,\mathrm{mA} \end{aligned} \tag{4.10}$$

Looking back at the ADC side, a suitable U_{REF} and PGA setting G had to be found. The former is simply chosen to be the ADCs internal reference voltage of 2.048 V. For the latter, one must first determine the expected measurement range as in equation 4.11 using the ±2 % relative strain requirement defined by P1.

$$\begin{aligned} U_{IN\_max} &= -U_{IN\_min} \\ &= U_{exc} \times \left(\frac{R_{SGR\_max}}{R_{SGR\_min} + R_{SGR\_max}} - \frac{1}{2} \right) \\ &= 20\,\text{mV} \end{aligned} \tag{4.11}$$

This results in a full scale range (FSR) of at least ±20 mV. The closest possible compliant PGA setting is 64 defining the FSR to be ±32 mV. The ADC datasheet describes the effective number of bits at the given samplerate and PGA setting to be 13 [16]. Hence, the least significant bit (LSB) resolution for verification of performance requirement P2 is 3.9 µV. Comparing this to an untuned half bridge, where one SGR is 0.001 % deformed, yields a positive result, as U_{IN} would be approximately 20.5 µV and is therefore well above the LSB accuracy.

Lastly, the PGA must be confirmed to not swing closer than 100 mV to the analog power supply set at 3.3 V. Measurements using a wheatstone bridge are fully differential, meaning the common mode voltage U_{CM} is constant at $U_{exc}/2 = 2.048\,\text{V}$, far exceeding the allowable range of 0.74 V to 2.56 V derived from equation 2.6.

In total twelve ADCs measuring each one SGR shall be synchronized to produce an easier to evalute dataset. The internal clock signal generators however have an operating range of 3.89 MHz to 4.3 MHz, which would lead in an extreme case to an unacceptable offset of 1 ms in just 410 ms. To resolve this, a shock hardened oscillator common to all SGR ADCs is used as external clock.

4.2 FPGA user logic

The main processing unit onboard the SPU is a SmartFusion2 (SF2) System on Chip (SoC) by Microchip, former Microsemi. It contains a field programmable gate array (FPGA), also called fabric, and a Cortex-M3 microcontroller subsystem (MSS) with ARMv7-M architecture [6]. This section focuses on the user logic loaded onto the FPGA.

The affiliated Libero SoC integrated development environment (IDE) enables a user to generate fabric layouts either visually utilizing so called SmartDesigns or text based using a hardware description language (HDL). In fact, all SmartDesigns are automatically converted into HDL resources prior to synthesis. For this project, a hybdrid solution using both techniques was chosen, where the primary components are developed using a HDL, namely VHDL, and subsequently interconnected by a SmartDesign root component. The IDE contains various tools enabling and enhancing workflows for simulations, timing and power analysis and programming the IC itself with numerous FPGA typical intermediate steps.

Several options are available to digitally connect an FPGA component with the MSS. Most basically, a set of microcontroller peripherals like UART-, SPI- or general porpuse input / output (GPIO)-interfaces can be routed to the fabric. While this suffices for simple setups, it is unsuitable for data centric medium to high traffic use cases. For these, the fabric interface controller (FIC) offers bridging functions from the MSS embedded advanced high-performance bus (AHB) to an AMBA interface, enabling access of FPGA components within the MSS address range. For the SF2, AMBA connections can be an AHB-lite or APB3. Latter are used exclusively for custom MSS to fabric bus interconnections, as they are superior in terms of required implemention effort and power consumption for applications not dependend on pipelined fabric operations. [6]

Interrupts are transported bidirectional via the fabric interface interrupt controller (FIIC) residing parallel to other MSS peripherals between it and the FPGA. For each direction, 16 individually configurable, level-sensitive interrupt ports are available. In addition, 27 MSS internal interrupts fixed to specific events can be routed to the fabric as well. Features like changing the control's status and priorization are accessable through a nested vector interrupt controller (NVIC). [6]

The FIC routes master and slave APB3 interfaces to the fabric. When using a master port, the FPGA component must drive address and control lines. A slave component however must drive a ready signal. The MSS shall be in control of the address bus for the instances outlined in this thesis, leaving fabric components to be slaves. [6]

An APB3 compliant slave must be clocked in sync and reset with the MSS via the provided signals *PCLK* and *PRESETN*. Two additional lines for slave state control named *PSEL* and *PENABLE* jointly indicate, if there is currently a bus access to a specific peripheral client. This spares an evaluation of all address bits at component level, but only the ones used for user defined, internal accesses. With APB3 and the single core SF2, a bus access blocks any further firmware execution, until the *PREADY* line is asserted by the correspondig slave. A write enable signal *PWRITE* denotes, whether the data bus of varibale width $w_{data} \in \{\{8, 16, 32\} \times bits\}$ is master driven or shall be set by the slave upon asserting *PREADY*. Finally but unused by the custom blocks developed for the HERMESS experiment, an accessed slave may optionally trigger an interrupt for exception handling through use of the dedicated *PSLVERR* signal. The bus connections are illustrated in figure 4.6 for easier comprehension. [6]

The address bus *PADDR* generally has a total width of 32 bit with up to 12 bit of the numerically least significant portion usable for APB3 slave internal addressing. Yet unknown as to why, the IDE automatically and inevitably assigns two disjoint but functionally seemingly identical address ranges to custom instances. As noted above, the data bus may either be 8 bit, 16 bit or 32 bit wide. To consider this, three corresponding sets of assembly commands are available for firmware development. The address resolution however is in any case bytewise, resulting in hard faults when executing

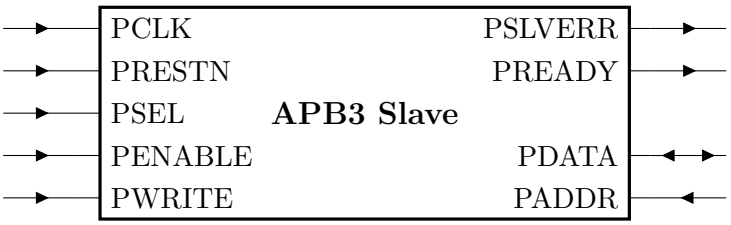


Figure 4.6: Bus connections of an AMBA APB3 slave

N bit access commands on any decimal address A_{10} for which $(A_{10} \times 8) \mod N \neq 0$ true is.

Figure 4.7 shows a simplified version of the root components visual representation. A main APB3 bus is connecting the MSS with six *STAMP* units each having their own outbound SPI link to sets of three ADCs as previously illustrated in figure 4.1. The *STAMP* components automatically gather available data from attached ADCs and provide a 64 bits wide data bus to the memory unit for further processing. They also enable communication between the MSS and an ADC, which is crucial for configuration and calibration porpuses. The *STAMP* components are covered more detailed in section 4.2.1.

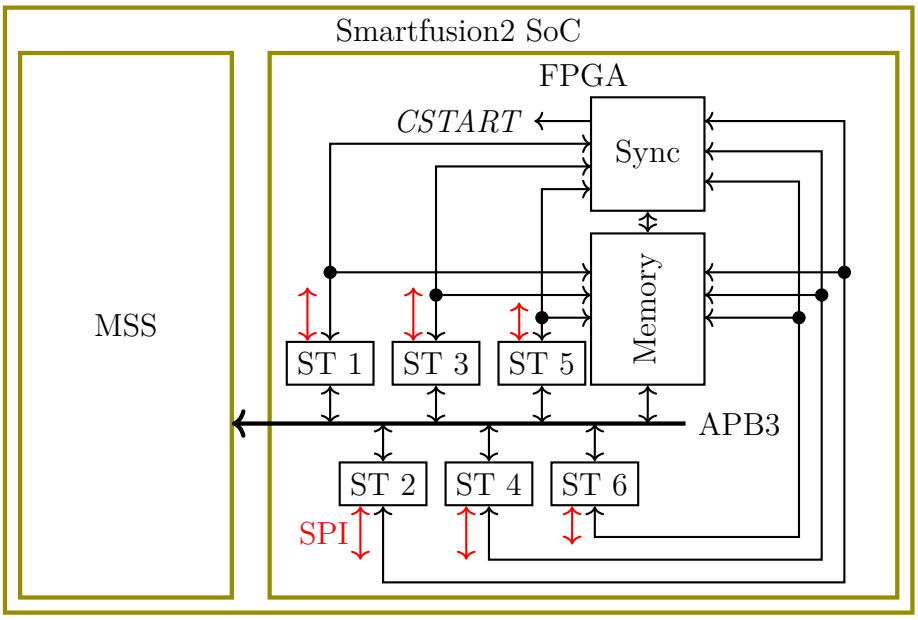


Figure 4.7: Overview of HDL components used in user logic fabric of Smartfusion2. "ST" as abbreviation for STAMP.

The memory and sync blocks are actually implemented in a single HDL component *MemSync*, sparing the construction of another APB3 slave and measures for keeping their states alligned. The memory logic continously collects all acquired data and

builds a package with uniform size and structure, suitable for unaltered storage. After such a package was built, an interrupt is triggered signalling its availability and need to read it out. For historical reasons leading back to concerns about unmatched ADC clock frequencies before a common clock generator was utilized, a synchronization logic detects asynchronous ADC events and, when a ruleset is satiesfied, triggers a resynchrounisation using the *CSTART* signal. Section 4.2.2 provides more information about the design and inner workings of this combined component.

After power up or a full reset, all components are in an idle state waiting for the MSS to command them into action. Only after a mandatory sequence of configurations trips the FPGA into autonomous operation. The only truly necessary action to be performed by the microcontroller is to read out the available data from the *MemSync* unit and write it onto memory. Not doing so would lead to a certain loss of gathered data whilst the acquisition continues unfettered.

Table 4.1 lists all instances of custom APB3 slaves with their assigned address ranges and processor interrupt signals. Note, that only the first address range is used by the firmware. Coincidentally, all components trigger an interrupt, once a dataframe or package is available for acquisition.

Component	**Address ranges**	**Interrupts**
MemSync	`0x5000.0000 - 0x5000.0FFF` `0x3000.0000 - 0x3000.0FFF`	Data Available: `F2M[0]` Hard reset: SF2 RESETN
STAMP1	`0x5000.1000 - 0x5000.1FFF` `0x3000.1000 - 0x3000.1FFF`	Data Available: `F2M[1]`
STAMP2	`0x5000.2000 - 0x5000.2FFF` `0x3000.2000 - 0x3000.2FFF`	Data Available: `F2M[2]`
STAMP3	`0x5000.3000 - 0x5000.3FFF` `0x3000.3000 - 0x3000.3FFF`	Data Available: `F2M[3]`
STAMP4	`0x5000.4000 - 0x5000.4FFF` `0x3000.4000 - 0x3000.4FFF`	Data Available: `F2M[4]`
STAMP5	`0x5000.5000 - 0x5000.5FFF` `0x3000.5000 - 0x3000.5FFF`	Data Available: `F2M[5]`
STAMP6	`0x5000.6000 - 0x5000.6FFF` `0x3000.6000 - 0x3000.6FFF`	Data Available: `F2M[6]`

Table 4.1: Memory and interrupt list for custom APB3 slaves

4.2.1 STAMP

The six FPGA *STAMP* components are instances of the eponymous VHDL component and are connected via a single dedicated SPI interface per instance to all three ADCs of a physical STAMP. These are specifically one for RTD and two for SGR measurements.

A STAMP component shall provide means of communication between its ADCs and the MSS. After proper configuration it shall also continously automatically read out attached ADCs and build uniform 64 bits data packages for further processing by the *MemSync* block. Finalization of such a package is indicated by assertion of a *data available* signal connected to an MSS interrupt port and a *MemSync* input.

Connected are the ADCs via shared *master in / slave out (MISO)*, *master out / slave in (MOSI)* and *serial clock* traces. The *CSTART* signal mentioned in section 4.2 above is shared by all ADCs and is thus not further discussed in this section. Additionaly to the shared lines, each device has two dedicated, negated signals connected to its *STAMP* component with the first one being an SPI *chip select (CS)* line and the other one a unidirectional *data ready* output. The latter must be configured prior to usage as such and is then asserted whenever the ADC finished an operation like a conversion or calibration until an SPI access occurs.

The used ADCs are configurable using the SPI link and lose their state during reset or power cycling. Thus, when running these in non-default configuration, a reprogramming is necessary after each power up. This is generally achived by writing to the relevant registers with the exception of toggling data continuous mode, for which two dedicated commands are available. For example, the strain gauge rosette (SGR) measuring ADCs are set to a PGA value of 64, a data output rate of 1000 Hz and their respective calibration frames. Whilst the configuration is always controlled by the MSS, data collection can be automatically performed by the FPGA *STAMP* component. Once a falling edge on *data ready* is detected, a `NOP` command is issued signaling the ADC to write its 16 bit data word to *MISO*. This signal is then captured and parallelized in the *STAMP* unit.

Compared to the ADCs running on a 4 MHz clock, the FPGA clocks in at a relatively high 50 MHz. Therefore a number of non-trivial timing requirements requiring explicit implemetation must be considered when communicating with an ADC. Various timers utilizing countdown mechanisms were integrated into the STAMP component and a prescaler applied to the embedded SPI master blocks.

Each STAMP has two synchronized SGR ADCs attached which should trigger the *data ready* signal simultaneously. Any unusual offsets are recorded and eventually after exceeding a configurable threshold assert the *sync request* signal to the *MemSync* unit. Because the strain gauge rosettes are inherently temperature compensated, a loss of thermal readings is considered to have a low impact on evaluability. RTD ADCs are therefore not involved when issuing a resynchronization request.

Figure 4.8 shows the layout of the FPGA *STAMP* component 32 bit configuration register. Except for the last three bits containing a unique, HDL generic STAMP identifier *ID*, all fields are read- and write-accessable. Bits 29 downto 24 contain the *async threshold*, after which the *sync request* signal is set. For that, the offset clock cycles divided by a HDL configurable prescaler are compared against *async threshold*.

The 30th bit *C* enables or disables the *continuous mode*. Lastly, the *R* bit resets the entire component including all internal states and stored values, implicating, that this set bit itself will also be reset. As shown, most places are actually reserved for possible future use or are rather unused. This only comes with the neglectable penalty of more necessary FPGA logic units but not speed or similar downsides, as the MSS is capable of 32 bits transfer operations.

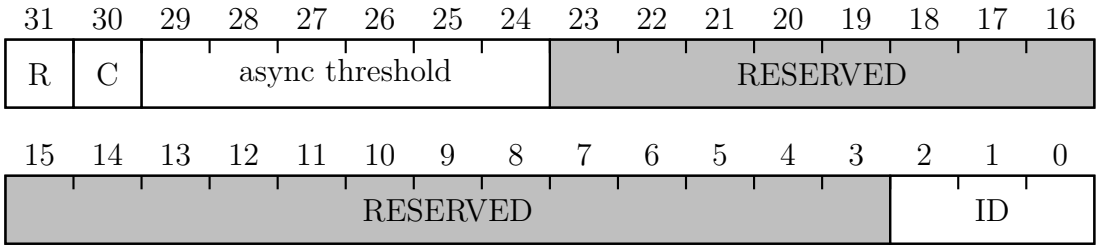


Figure 4.8: Register map for *STAMP* components configuration registers

The 64 bit wide output data frame must either be read via two seperate APB3 accesses or an FPGA internal evaluation module like the *MemSync* block. Internally it is stored in a register which is summarized by figure 4.9. The first 48 bit contain the last readings of all three attached 16 bit ADCs. The last 16 bit form the status register containing information about the data validity and STAMP identity. All three ADC states are represented by each two flags. The first flag bits, N1 through N3 for *new value* is set to `1`, if between the last status register reset and the reading of it, the represented ADC was read out at least once. This may help identifying unresponsive ADCs. The second set of flag bits, O1 through O3 for *overwritten value* is set to `1`, if the status register reset occured too late, and the respective ADC was read out at least twice. The flag labeled RR is only a shortcut for *request resync* and is described above. 6 signed integer encoded bits counting the *async cycles* to be compared against *async threshold* are also stored in the output frame for later reallignment. The *ID* is equal to the one set in the configuration register.

STAMP units provide a set of combinable commands executable through accesses to specific APB3 addresses. The address bus connection is 12 bits wide with the four least signifant bits being unused. Those should be `0` for every access to prevent MSS hard faults resulting from illegal address resolutions for certain assembly instructions. Following list describes the *STAMP*. `X` denotes its combinability with other commands filling the gap at this place.

1. `1XXX XXXX 0000` **Atomic operation**: This command modifier prevents a STAMP unit running in continuous mode to automatically read another ADC value into its internal storage until the subsequent command was fully executed. As a consequence, it is important to keep the time between asserting this modifier and the follow-up command short. This feature is mainly used to effectively freeze the data values while reading them out using multiple commands.

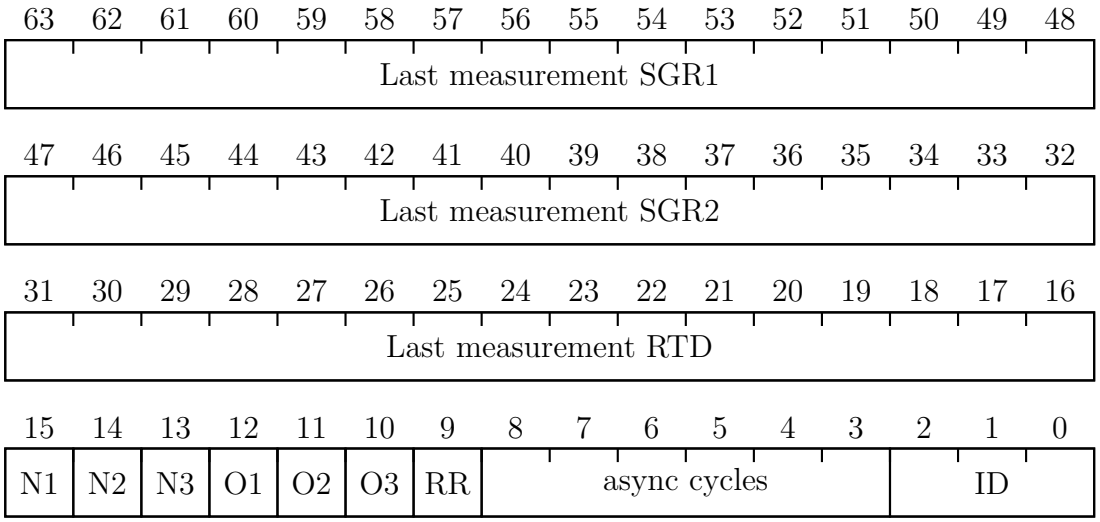


Figure 4.9: Register map for *STAMP* components output data frame

2. `X1XX XXXX 0000` **Reset status register after finishing operation**: This resets only the internal status register but not the entire component. For example, this modifier is useful when reading and resetting the status register in a single command.

3. `XX00 XXX1 0000` **ADC for SGR1**: Allows directly writing to the ADC capturing axial and tangential deformations usign the first SGR. Any parallel MISO activities are recorded and can be read out by a seperate command. Note, how it is possible to write out commands to multiple ADCs at the same time. Doing so is only advisable, when assertig commands not driving the MISO port.

4. `XX00 XX1X 0000` **ADC for SGR2**: Acts similar to the previous command, but with the SGR ADC capturing shear components.

5. `XX00 X1XX 0000` **ADC for PT-100**: Acts similar to the previous command, but with the RTD ADC.

6. `XX00 1XXX 0000` **Polling request**: Control to the MSS is only given back after the *data ready* lines return to their active state after issuing the instructed command via any of the three previously ADC access functions. Otherwise those commands are executed in parallel. This is especially useful when performing actions like startups or calibrations, where time requirements are hard to estimate.

7. `XX00 0000 0000` **NOP**: Does nothing but waste clock cycles and is only implemented for testing porpuses.

8. `XX00 1000 0000` **Last ADC reading**: Returns the last received *MISO* stream. Even though the address masks may suggest it, this command is not combinable

with any ADC access function or the polling request modifier.

9. `XX01 0000 0000` **SGR1 and SGR2 readout**: Outputs the first 32 bit of the data frame containing the last automatic readings of both SGRs ADCs.

10. `XX01 1000 0000` **RTD and status register readout**: Outputs the last 32 bit of the data frame containing the last automatic readings of the SGRs ADC and the status register.

11. `XX10 0000 0000` **Access configuration register**: Accessing the read and write permitted configuration register.

12. `XX11 1000 0000` **Access dummy register**: Accessing the read and write permitted 32 bit wide dummy register. This is only used for debugging porpuses.

4.2.2 MemSync

The *MemSync* unit fulfills two tasks combined in a single HDL component. First it shall collect the dataframes from all six *STAMP* units and build a data package suitable for storage without further processing. It shall also provide appropriate resynchronization and recovery functionalities for asynchronous or failed ADCs.

Data packages for storage are simply generated by joining the six lastly available dataframes from all *STAMP* together and prepending the metadata register value and a separation pattern. Figure 4.10 depicts a bytewise register map for a single data package. The first byte contains just plain zeroes to positively identify the start of a new data package, because all bits are set to one as the memories initial state after erasure. Next comes a 32 bits wide timestamp indicating the first *STAMP* data available event with a resolution of 100 µs starting with startup of the FPGA giving a timeframe of just below 5 d, far exceeding the expected operating conditions. Subsequent data available events are timed relative to the timestamp and stored with each a single byte. The FS bits are made up of 6 places indicating failed *STAMPs*, a single bit which is set, if a gathering iteration by the MSS was missed and a last bit, if a resynchronization was triggered. Following to that, the six dataframes are appended.

The data package is latched until it was marked read out using a command modifier with exception of the FS byte that may flip the bit indicating a missed data acquisition. For a full readout, fifteen four byte read operations must occur. APB3 access is not permitted, while a data package is in process of assembly, as this would introduce timing errors for determining the data available events offsets. Following list describes the addresses for these readouts. An `X` in the address marks compatibility with other commands filling the gap.

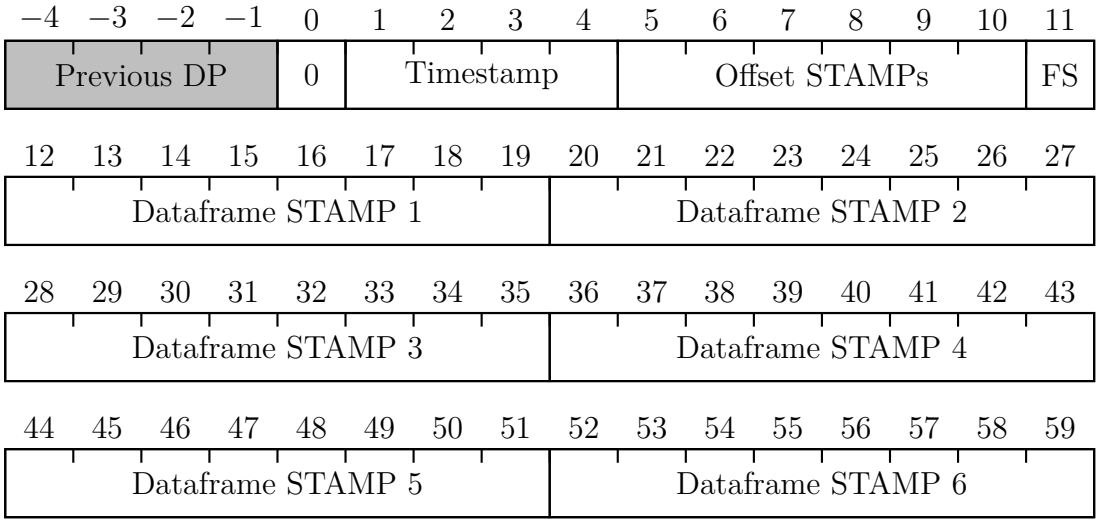


Figure 4.10: Bytewise register map for a single data package

1. `100X XXXX 0000` **Allow new data acquisition after finishing this operation**: This command modifier frees the latched data package for new incoming dataframes.

2. `X000 AAAA 0000` **Access output data package**: The address bits `AAAA` with binary values between 0 and 14 are used to access the output data package in increments of four bytes. An access outside the described address range leads to undefined behavior.

3. `X001 0000 0000` **Access configuration register**: Accessing the read and write permitted configuration register.

The configuration register is 32 bit wide where only the three least significant bits are used. From bit 0 to 2, the configuration options are component reset, enable resynchronization and enable hard reset. The component reset is the same as with the *STAMP* unit. All values are initialized with a value if `0`. The latter two values should only be set to `1`, once all *STAMP* components are fully configured for continuous mode.

Resynchronization can improve post-flight data evaluability, when it was executed after it was detected that some ADCs are not running in unison with the others. There are two levels available for such detection: On the lower level, the *STAMP* components themselves assert the *sync request* signal, whenever their two attached ADCs exceed a configurable time delay between two matching data ready events. The higher level detection is realized by measuring the time offset between the first and any subsequent *data available* signal assertions coming from the *STAMP* units. A resynchronization mechanism is provided through a single *CSTART* trace shared by all ADCs. Is this signal pulled low, the current conversion is interrupted immediately and a new one

initated once the voltage is pulled high again. Such resynchronization introduces an ADC downtime of maximally the time required for a single conversion at the selected samplerate plus a neglectable *CSTART* reaction delay time. Because a the data loss caused by resynchronization is minimal, the requirements for triggering and limiting them are relatively low. A resynchronization is considered worth it, once one or more *STAMP* components request it or the higher level detection oversteps a threshold of 300 µs. To prevent infinite resync loops, a delay between two consecutive resynchronizations must exceed 1 s, the equivalent of 2000 measurements at a samplerate of 2 kHz. This feature can be toggled via the firmware.

A full system restart may be needed in order to recover unexpectedly failed ADCs. Is an ADC not asserting its *data ready* signal in regular intervals, its responsible HDL *STAMP* component will not produce any further data packages leaving both affiliated ADCs and the component itself to be considered failed. It is therefore insignificant, which and how many elements of a STAMP have failed to be counted as a single failure. After not responding for 5 ms, the equivalend of the time needed for 5 to 10 measurements depending on the selected samplerate, the STAMP with all components is marked as failed and will not be waited upon in further iterations. For simplicity reasons and the empirically low probability of such failures, the mechanism to cope with them is to initiate a full system restart through a FPGA hard reset signal. It was determined an appropriate balance between the downtime of at least 2 s and its use is only given if four or more failures occured, rendering a meaningful evaluation with the Scopinsky method impossible.

The deterministic finite state machine for executing these activities is illustrated by the state diagram of the underlying Mealy machine in figure 4.11. A legend for the edges is given in table 4.2.

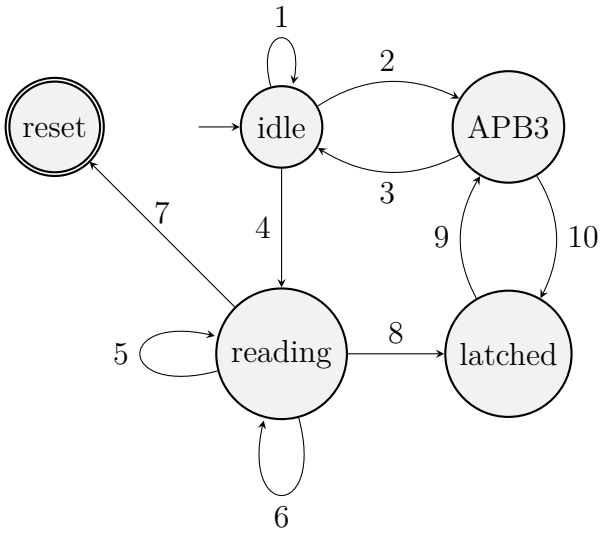


Figure 4.11: Mealy machine for the *MemSync* HDL component

Edge	Input	Output
1	Resynchronization enabled ∧ (Last time in reading state ≥ 300 µs ∨ # of request resyncs > 0)	Trigger resynchronization
2	APB3 access	None
3	APB3 access finished ∧ (Came from idle state ∨ Allow new data acquisition set)	None
4	∃ *data available* signal	Copy *STAMP* dataframe
5	∃ new *data available* signal	Copy *STAMP* dataframe
6	Time in state ≥ 5 ms ∧ # unfinished or failed *STAMPs* ≤ 3	Mark failed *STAMPs*
7	Time in state ≥ 5 ms ∧ # unfinished or failed *STAMPs* > 3	Trigger hard reset
8	∄ unfinished and not failed *STAMP*	*data available* interrupt
9	APB3 access	None
10	APB3 access finished ∧ Came from latched state ∧ Allow new data acquisition not set	None

Table 4.2: Legend for the edges in figure 4.11

4.3 Firmware

The SmartFusion2 (SF2) houses an embedded Cortex-M3 microcontroller subsystem (MSS). Most peripherals such as UART, SPI and GPIO units may be enabled and configured using the System Builder integrated into the Libero SoC IDE. Also configurable is the clock source which was selected to be the default 100 MHz output of the 50 MHz RC oscillator driven clock conditioning circuit (CCC). Programming was done using an Eclipse IDE derivate named SoftConsole in the C and assembly programming languages.

The hardware Watchdog (WD) mechnism prevents long term MSS malfunctions, in which the program is not reaching specific places in its sequence, that should have been reached in usually regular intervals. A possible cause for such behavior could be an unexpected deadlock or infinite loop. The mechanism works by decrementing a timer counter and if it is not resetted by a program instruction before reaching `0`, it will reset the entire SF2 SoC with a software readable flag indicating the startup occured after a WD timeout. The firmware shall refresh the WD timer counter in the main loop mostly being idle and waiting for interrupts. It is clocked with the 50 MHz RC oscillator, thus a refresh value of `0x2FA.F080` is selected to have a reset after 1 s

of not refreshing.

A volatile random access memory (RAM) for primary storage is included as well and may be extended by external ICs or FPGA logic. The internal storage is sufficient for the experiments firmware and therefore no extensions are used. Hex files containing any contents can be written onto the onboard nonvolatile electrically erasable, programmable read only memory (EEPROM) and is used for this setup as the program instructions as well as the configuration storage unit. For that porpuse, the EEPROM is fractioned into 125 KiB read only memory for the program and 320 B read and write access memory for the configuration.

For this project, there are two different types of drivers available. First there are the Libero SoC exported public drivers and then there are the custom ones developed for the attached APB3 components described in section 4.2. The latter are usually based on a small set of assembly instructions or hardware abstraction layer (HAL) functions and expand these into component specific C application programming interfaces (APIs) to allow for higher level programming. The STAMP API uses struct instances for emulating an object oriented paradigm in the otherwise procedural C programming language.

The nature of APB3 operations based on address offsets and multi-porpuse registers deems the frequent usage of structs like the one in listing 4.1 quite convenient for high level programming. It often requires however explicit functions for converting the struct into a data format for applying to the bus and vice versa. The option of a bitfield wrapped by a union having a data bus wide unsigned integer as second member was explored but later discarded, because the allignment of C bitfields is compiler dependent and therefore would have required additional efforts for ensuring a correct compilation.

The MSS features two integrated 32 bit timers in either one-shot or periodic configuration counting down from a predefined value until reaching zero and when enabled forcing a program sequence jump to the associated interrupt vector. Both system timers may be joined together to a single 64 bit one. Only the first timer is used for utility functions like delaying the next command or periodically jump to a function. Listing 4.2 contains the implementation of a delay function for delays up 2^{32} ms $\approx$ 50 d with 1 ms LSB resolution using the first timer `TIM1`. This example shall provide insight on how most SF2 public driver APIs and interrupts are to be used. Because no prescaler is available, `TIM1` is operated with the system clock at 100 MHz. An interrupt service routine (ISR) decrements a static counter until it eventually reaches `0`, at which point the timer is stopped and the function returns.

The pseudo code in listing 4.3 summarizes the firmware program sequence and is further elaborated in the following paragraphs.

After MSS startup, a series of initializations are executed to enable the use of needed peripherals. The DAPI UART port for instance is set to a baudrate of 19 200 with one stop bit, no parity bits and eight bit words according to the specifications

```
1  typedef struct stamp_status {
2      // indicates, if the ADC read a new value since last
          acquisition
3      uint8_t dms1NewVal : 1;
4      uint8_t dms2NewVal : 1;
5      uint8_t tempNewVal : 1;
6      // indicates, if a an ADC reading was overwritten, since
          the last acquisition
7      uint8_t dms1OverwrittenVal : 1;
8      uint8_t dms2OverwrittenVal : 1;
9      uint8_t tempOverwrittenVal : 1;
10     // indicates, if this stamp component currently requests a
          resync
11     uint8_t requestResync : 1;
12     // the number of clock cycles one dms adc was ahead of the
          other one
13     uint8_t asyncCycles : 6;
14     // the generic HDL identifier of this component
15     uint8_t stampId : 3;
16 } stamp_status_t;
17 uint32_t APB_STAMP_generateStatusBitfield (stamp_status_t *
      status);
18 void APB_STAMP_generateStatusStruct (uint32_t bits,
      stamp_status_t *status);
```

Listing 4.1: Examplatory struct definition using a bitfield

referenced in appendix C. Most peripherals however only require the invocation of a single initialization function with no or only trivial attributes. These will only be configured once they are needed, as it is the case illustrated with the delay function in listing 4.2. This procedure may initiate a single component multiple times, but it ensures a correct state for every use case. A special initialization is performed for the heartbeat LED, where the `TIM1` ISR is setup to toggle the LED with 1 Hz.

If the startup occured after a WD reset, will this event now be transmitted via the telemetry (TM) link. To log this event on the flash memory together with the data packages assembled by the *MemSync* unit, the `wd_bit` is set to the non-default value of `1` and will later be appended to the first page of data packages. How these pages are structured will be explained in the paragraph describing how *MemSync* interrupts are handled below.

Next up is a special EEPROM partition containing the SPUs configuration read out and converted into a globally available struct. These values are then used for setting up the *STAMP* and *MemSync* HDL components via APB3. Here, the order of execution

```
1  static volatile uint32_t delayCompleted = 0;
2
3  void Timer1_IRQHandler () { // ASB weak defined vector
4      delayCompleted--;
5      MSS_TIM1_clear_irq();
6  }
7
8  void delay (uint32_t ms) {
9      delayCompleted = ms;
10     MSS_TIM1_init(MSS_TIMER_PERIODIC_MODE);
11     MSS_TIM1_load_immediate(100000); // 1ms @100MHz
12     MSS_TIM1_enable_irq();
13     MSS_TIM1_start();
14     while (delayCompleted) {
15         MSS_WD_reload();
16     }
17     MSS_TIM1_stop();
18 }
```

Listing 4.2: Delay utility function

is important, because as mentioned in section 4.2.2 above, false system resets may be triggered, if the *MemSync* unit is enabled before all *STAMPs* are in continuous mode. *STAMP* components must be configured on two levels. First the ADCs themselves need to be set up to use the samplerates conforming P3 and P6, the calculated PGA settings, the correct reference voltage sources and further basic operational preferences. After that may the *STAMPs* unit itself be configured for continuous operation.

If startup calibrations are selected, will those be executed at this point. Even though both, full-scale- and offset-calibration types, are available options, really only the latter seems to be useful at startup. Prerecorded calibration data loaded into ADC registers during the previous step are overwritten for the duration of this power cycle.

A current page-wise location pointer is established next by iterating the external flash memories initial bytes of each page, until an empty page was found. This pointer is then used for assigning writing locations to available data packages. This causes a restart not to automatically overwrite previously recorded measurements but also a possible space shortage, if the memory was not erased prior to flight. A single data package has a size of 60 B and a page is 512 B large and programmable at once due to a buffer of equal size. The firmware therefore collects $\lfloor 512\,\mathrm{B}/60\,\mathrm{B} \rfloor = 8$ data packages to store them at once with an appended MSS status byte containing amongst others the `wd_bit`. The flight phase with powered experiments lasts approximately 600 s. Equation 4.12 shows how the available memory space of 2×64 MiB [5] is sufficient to start data acquisition at a maximum samplerate of 2 kHz 10 min prior to launch and

```
initialize mss components
initialize periodic heartbed led
If restart after watchdog reset Then
    wd_bit := 1
    tm_print("WD␣Triggered")
End If
config := load_configuration()
configure_stamps(config[stamp])
configure_memsync(config[memsync])

If config[calibrate] is set Then
    calibrate stamps
End If

location := 0
While eeprom[location] is not empty Do
    location = location + 1 page
End While

Infinite Loop Do
    If config[recording_conditions] are met Then
        handle memsync interrupts
        periodically send tm frames
    Else If
        handle dapi interrupts
    End If
    refresh watchdog
End Infinite Loop
```

Listing 4.3: Pseudo code describing the SPU firmware

still maintain the mandatory 25 % margin [26], even when effectively wasting (512 B mod 60 B) − 1 B = 31 B per page.

$$\begin{gathered}\#\text{pages available} = (2 \times 64\,\text{MiB} \times 8\,\frac{\text{bit}}{\text{B}})/512\,\text{bit} = 2 \times 10^6 \\ \#\text{pages required} = (2000\,\text{s}^{-1} \times 1200\,\text{s} \times 125\,\%)/8 = 375 \times 10^3 \\ \Rightarrow \#\text{pages available} \gg \#\text{pages required}\end{gathered} \tag{4.12}$$

Another concern to be verified is the necessary writing speed of 250 pages per second. The datasheet [4] specifies typical program rate to be 1.5 MiB/s far exceeding the rate of 125 KiB/s at which the data packages accrue. The same applies to the SPI periphery clocking at a maximum of 50 MHz and therefore enabling transmissions with

theoretically up to $\approx 6\,\mathrm{MiB/s}$.

Data recording is only possible, if the physical IC write protection with the "REMOVE BEFORE FLIGHT" tag is removed from the SPU. The loaded configuration dictates a minimum recording time after the conditions for starting the acquisition are met. These are defined through a subset of the three control lines LO, SOE and SODS, in which any of them must signal an active state. Another subset is configured for halting the acquisition after the minimum recording time exceeded, if all of them are in an inactive state.

DAPI interrupts may occur anytime and are disabled during data acquisition to prevent them from taking possibly otherwise needed processing time. Via this interface, the ground station software (GSS) is able to not only push new configurations but also to immediately trigger, read and write ADC calibrations, read all written flash memory pages and erase them prior to the next data capture. All communications are GSS initiated through a UART interface and are either C-string based or a collection of framed raw bytes.

Telemetry datagrams are periodically send via the RXSM to provide some real-time information about the experiments state during flight. Here, the principle of a slim design is applied by simply transmitting 50 page contents with 481 B each per second. This corresponds to every 50th or 25th written page at a configured samplerate of 2 kHz or 1 kHz respectively. For error correction, a cyclic redundancy check (CRC) checksum with a yet to be determined polynomial is appended to each page. This setup requires a TM bandwidth of $\approx 4000\,\mathrm{kbit/s}$ below the design requirement item D11 limiting the bandwidt to 30 kbit/s. This part of the firmware is not implemented or even properly specified yet and may be subject to change in the future.

4.4 Ground station software

REXUS experimenteer groups are encouraged to collect live data from their experiments shortly before lift-off and during flight using the provided telemetry (TM) link. For that, either a serial or a science net port may be utilized on ground. Furthermore, the HERMESS experiment requires pre-flight configuration and post-flight data acquisition via the Data- and Programming Interface (DAPI).

A custom ground station software (GSS) was developed as a multiporpuse solution for all interfacing needs with the experiments SPU. Its main features include organizing the generated files in project catalogs, enabling handling of configuration and ADC calibration data, visualizing the measurements, and fully integrating other features described in the DAPI and TM specifications. This software is not fully developed yet as of completion of this thesis thus has to be finished prior to launch. Specifically, only local components such as the graphical user interface (GUI) and file handling can be

considered finished to satisfaction.

The GSS is mainly written in Kotlin, a programming language compiled for execution by the Java virtual machine. It combines the advantages of Javas highly developed environment with a more modern and convenient set of syntax rules. Kotlin is in fact mostly compatible with existing Java code, so that wrapper frameworks exist for the most commonly used ones. Dependencies and build configuration are maintained by Maven.

TornadoFx, a Kotlin wrapper for JavaFx, is used in conjunction with the modelview-view-model pattern for developing the GUI. Here, an intermediate instance between background model and GUI view, the modelview, acts as the input validator output selector. For some more general aspects such as application and connection management, a controller in the sense of the model-view-controller pattern was introduced.

Figure 4.12 shows the general layout of the HERMESS GSS. Three menus are located in the top area for filesystem, DAPI and TM related tasks. The main area on the right is a tabview, where at least one tab is open in any active state. An accordion to the left with hidable categories displays an open project with its related files.

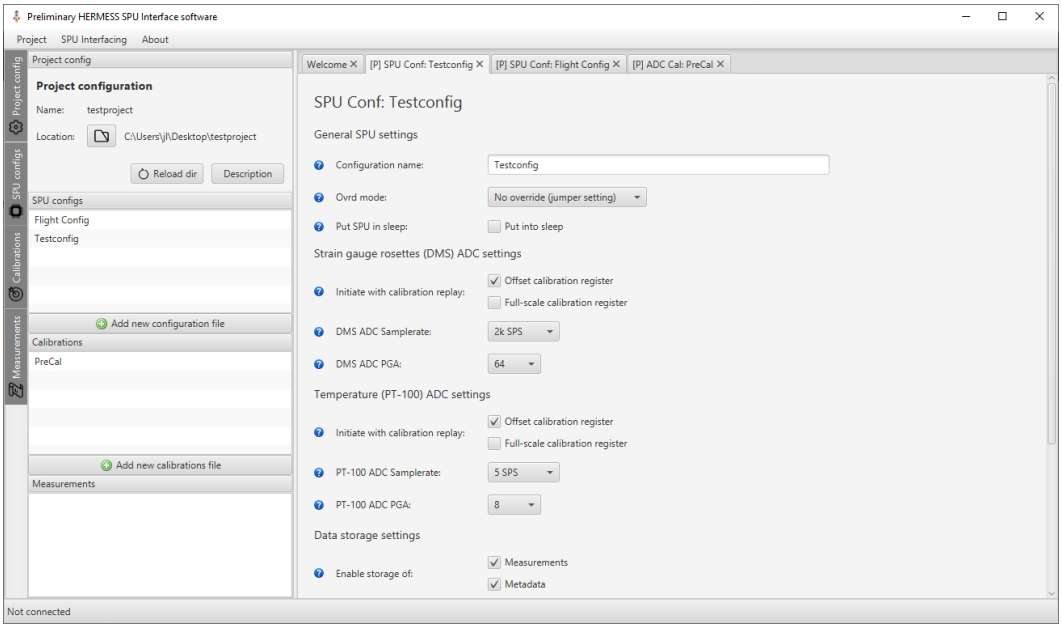


Figure 4.12: Screenshot of the HERMESS ground station software (GSS)

All tabs with the sole exception of the welcome tab, are interactive and host storable content. These instances can either be linked to an open project or be independent. The linked state is remarked alongside a possible marker for unsaved content in the tab title. Any project operations like saving, reloading and closing are automatically applied to all linked tabs. Not linked ones however cannot be saved sparing additional development efforts necessary for single-file data handling. They can however still be

used regularly, as long as no filesystem actions are involved and therefore enable users to test intermediate configurations. A one way converion to a project linked tab is the intended workflow for file creation, meaning all clean tabs are generally not linked.

A measure to increase overview and practically neglectably limiting users is, that every unique tab may only have a single instance open at any time. Tabs are unique, when their tab type-filename-pair is unique. Also, most options feature hoverable tooltips giving hints for their usage. This is especially necessary, because it cannot be guaranteed, that any deeply involved party is present for all operations.

Project files are stored in a directory named after the project itsef with different file extensions for simple document type identification. Therefore the program only reads all files within a directory and considers them to be part of the same project. Contents are formatted as a JSON string, because this is the recommended data storage format with the Kotlin programming languge. Following list describes all file extensions supported by the GSS.

.herpro A file containing only the stylable description string for the project. The project title itself is derived from the filename which must match the containing directories name.

.herconf The file containing all configuration items to be programmed via the DAPI into the SPUs EEPROM. These include but are not limited to PGA and sampler-ate settings or requirements on when to start recording. Multiple configurations may exist within a single project allowing the separation of test, calibration and flight settings.

.hercal The calibration data of the onboard ADCs. These are not to be confused with the calibration data generated in the first phase of the Scopinsky method for determining loads from strains. Normally, these data are gathered by first putting the HERMESS experiment in the required configuration, then trigger the appropriate ADC calibration via the GSS and subsequently download these data via the DAPI. Even though it is not recommended, these values can be manually altered and reuploaded.

.hermeas A file containg the readouts of measurement data recovered from the on-board flash unit processed into ordered groups of datasets per unique timestamp.

5 | Evaluation

Prior to this thesis, a verification matrix according to ECSS-E-HB-10-02A "Verification guidelines" was generated in a group effort. It assigns all requirements listed in section 3.1 to one or more of the four fundamental verification methods test, inspection, analysis and review-of-design. Nine tests are scheduled in the verification plan for this year 2021 covering a total of 23 requirements. Tests for the others are unpractical and are therefore covered by other means. [15]

Except for D6 limiting the module mass to 7.5 kg, all items were verified by review-of-design in this thesis. A proper average and maximum power consumption analysis must be performed ensuring not only safety of the vehicle platform, but also reliable operation of the HERMESS experiment itself. The review made in section 4.1 only conforms D8 and D9 without simultaneous regard for functional requirements F1 and F2.

The ECSS compliant risk register for the HERMESS experiment does not contain any items with a Propability × Severity score above "low" [15]. A pending reevaluation due to changes of the SPU design even elimantes item MS-10 having a major severity in case of a faulty primary/secondary SPU mode detection. The other items can only be addressed with further testing.

Unfortunately, further test progressions with the assembled SPU shown in figure 4.3 are not possible until a short believed to be on the 3.3 V supply net was identified and resolved. A new PCB iteration must be manufactured regardless of the nature of the short mentioned. That is because a list of in parts necessary design changes accumulated while assembling and working with the SPU.

For this project, large FPGA portions were debugged through trial and error or a technology called SmartDebug, with which any signal can be routed to one of two external probe ports without reprogramming. For the larger components however it proved advantageous to invest the efforts for development of a testbench usable with the ModelSim software, which comes shipped with the FPGA IDE, to perform pre- or post-synthesization simulations. A compilable language for bus functional models significantly helps with creating testbenches where bus operations shall be simulated by generating HDL files describing the formally correct access procedures.

The HDL *MemSync* unit was initially designed to also directly perform the SPI communications with the flash memory unit and is because of lack of time not yet satisfactorily alligned with the descriptions made in section 4.2.2. Doing so effectively means a full component redesign to eliminate all legacy segments.

Although only mechanically passive components are used, require some of them special attention for wear management. This concerns especially both plug systems for connecting the STAMPs with first housing and SPU. For example, an automotive grade crimped PCB to multiwire connector specifies the number of reliable matings to be just 10. The costly nature of these plugs renders it unpractical to use intermediate connectors during development and testing. A simple but restraining solution is chosen to strictly limit all mating actions. Another known component subject to wear are the memory units. The SPI flash memory unit experiencing the most traffic for example guarantees up to 10 000 write / erase cycles per cell. That equals the number of freshly started data acquisitions, because no load distribution algorithms are utilized.

The firmware was developed without any sort of formal test suites like unit tests running in a simulated environment because of its low complexity. Given an experienced developer, is the softwares behavior easily comprehensible and step by step verifiable using the SoftConsole integrated Joint Test Action Group (JTAG) debugger. Assertions at several places shall highlight any unexpected conditions occuring during non-productive times such as development, testing and calibrations.

While all aspects of implementation need further addressing, the advances made for this thesis formed a usable base making a timely completion of the flying electronic systems possible. The ground station software (GSS) will be taken over by another team member to finalize its implementation with primary focus on the DAPI integrations.

6 | Conclusion

The overall goal of developing the hard- and software for the signal processing tasks of the HERMESS experiment was partially fulfilled. A usable and from the ground up verifiable base for a finally flight-worthy Signal Processing Unit (SPU) was built alongside a functional prototype acting as firmware testbench and proof of concept. It therefore significantly advances the progress towards a unique functional module for measuring structural loads of sounding rockets.

The new concept is designed to be minimal and reduces overheads accumulated in previous building attempts. The FPGA design houses components simultanously and in real time acquiring the readings from dedicated ADCs. These components are also linked via an APB3 to the microcontroller subsystem (MSS) for data transfer to the firmware, where the packages are distributed to the telemetry (TM) and flash memory unit. Post-flight can this storage then be read out via the USB Data- and Programming Interface (DAPI). In addition to the flying electronic systems, major parts of the ground station software (GSS) for interfacing with the DAPI and TM were developed.

Design issues detected during assembly of the SPU such as a wrongly interfaced REXUS service module (RXSM) connector or false labeling make a second PCB iteration inevitable for future work. The not yet located short mentioned in chapter 5 however must be resolved prior to manufacture of any further SPUs to address possibly yet undetected design shortcomings. Not explicitly covered by this thesis are the finished and yet to be performed soldering jobs on the cables connecting the STAMPs with the SPU. This process takes a lot of time and sufficiently experienced craftsmen due to the accuracy required and is planned to be done in August 2021.

The prospect is that, once a fully functional SPU with compatible ground station software (GSS) is available and formally tested, load calibrations for the Skopinsky method are performed and the module is eventuelly launched in 2022.

There were many lessons learned while working on the project. One of the most precious ones is propably going to be the glimpse into how working on pieces of technologies in domains with special verification requirements is supposed to be conducted.

BIBLIOGRAPHY

[1] Traco Electronic AG. *Datasheet DC/DC Railway Converter - THN 10WIR Series, 10 Watt*. 2020.

[2] Bonnie Baker. "How delta-sigma ADCs work, Part 1." In: *Analog Applications Journal* (2011). URL: https://www.ti.com/lit/an/slyt423a/slyt423a.pdf.

[3] Bonnie Baker. "How delta-sigma ADCs work, Part 2." In: *Analog Applications Journal* (2011). URL: https://www.ti.com/lit/an/slyt423a/slyt423a.pdf.

[4] Cypress Semiconductor Corporation. *S25FL512S SPI Flash datasheet*. 2019.

[5] Cypress Semiconductor Corporation. *S70FL01GS SPI Flash datasheet*. 2019.

[6] Microsemi Corporation. *UG0331 - User Guide - SmartFusion2 Microcontroller Subsystem*. 2018.

[7] Karl Domjahn. *Master / Bachelor thesis: Design of a Flight Load Measurement System for Sounding Rockets*. The University of Queensland, 2016.

[8] Eurolaunch. *REXUS User Manual*. 2018.

[9] Trenz Electronic GmbH. *TEM0001 - SFM2000*. 2017. URL: https://wiki.trenz-electronic.de/display/PD/TEM0001+-+SFM2000.

[10] HBK (former HBM). *Datenblatt Serie Y Dehnungsmessstreifen (DMS)*.

[11] HBK (former HBM). *The Wheatstone Bridge Circuit*. URL: https://www.hbm.com/en/7163/wheatstone-bridge-circuit/.

[12] Heraeus. *Datasheet Platinum temperature sensor in thin-film technology M 310*.

[13] REXUS Team HERMESS. *HERMESS Student Experiment Documentation, REXUS29, V1*. 2020.

[14] REXUS Team HERMESS. *HERMESS Student Experiment Documentation, REXUS29, V1.1*. 2020.

[15] REXUS Team HERMESS. *HERMESS Student Experiment Documentation, REXUS29, V4*. 2020.

[16] Texas Instruments Inc. *ADS114x 16-Bit, 2-kSPS, Analog-to-Digital Converters With Programmable Gain Amplifier (PGA) for Sensor Measurement*. 2016.

[17] Texas Instruments Inc. *Datasheet LM4030 SOT-23 Ultra-High Precision Shunt Voltage Reference*. 2013.

[18] Inc. Maxim Integrated Products. *Sigma-Delta ADCs*. 2003. URL: https://www.maximintegrated.com/en/design/technical-documents/tutorials/1/1870.html.

[19] Microchip. *FPGA Design Tools - Licensing*. URL: https://www.microsemi.com/product-directory/design-resources/1711-licensing.

[20] Microchip. *SmartFusion2 SoC - Product Tables*. URL: https://www.microsemi.com/product-directory/soc-fpgas/1692-smartfusion2#product-tables.

[21] REXUS/BEXUS Organisers. *Guidelines for Student Experiment Documentation; SED Guidelines*. 2018.

[22] REXUS/BEXUS Organisers. *RX29 CAM ACCOMODATION v3-0 15Jun20*. 2020.

[23] REXUS BEXUS programme organizer. *Outreach guidelines within the REXUS/BEXUS programme (v.1.3)*. 2020.

[24] T. H. Skopinski, William S. Aiken Jr., and Wilber B. Huston. *Technical Note 2993 - Calibration of Strain-Gage Installations in aircraft structures for the measurement of flight loads*. Langley Aeronautical Laboratory, 1953.

[25] European Cooperation For Space Standardization. *ECSS-E-ST-10-06C Space engineering Technical requirements specification*. 2009.

[26] European Cooperation For Space Standardization. *ECSS-E-ST-40C Space Engineering Software*. 2009.

7 | Declaration of authorship

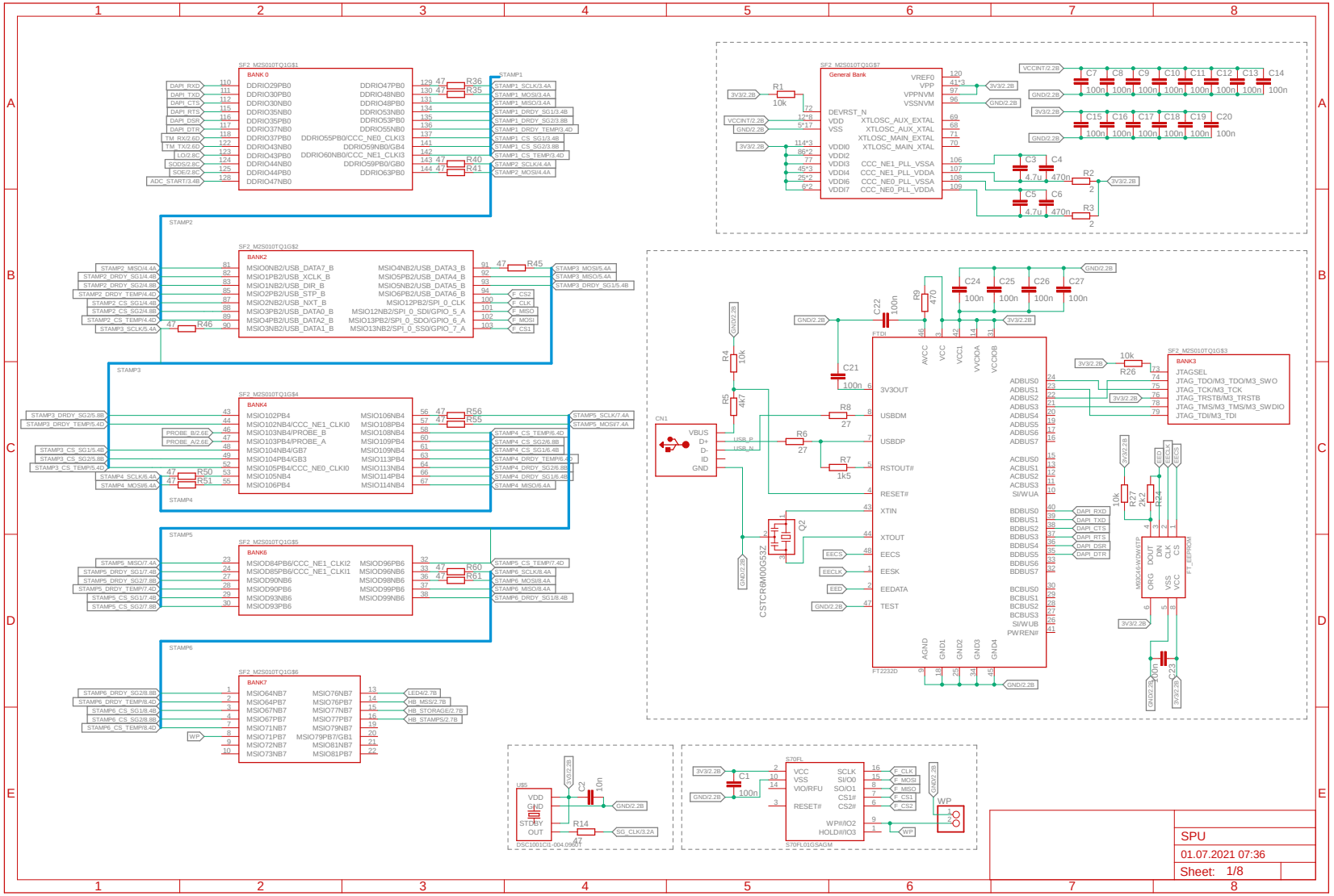
Hiermit versichere ich, die vorliegende Arbeit selbständig und ohne fremde Hilfe verfasst, die Zitate ordnungsgemäß gekennzeichnet und keine anderen als die im Literatur/Schriftenverzeichnis angegebenen Quellen und Hilfsmittel benutzt zu haben.

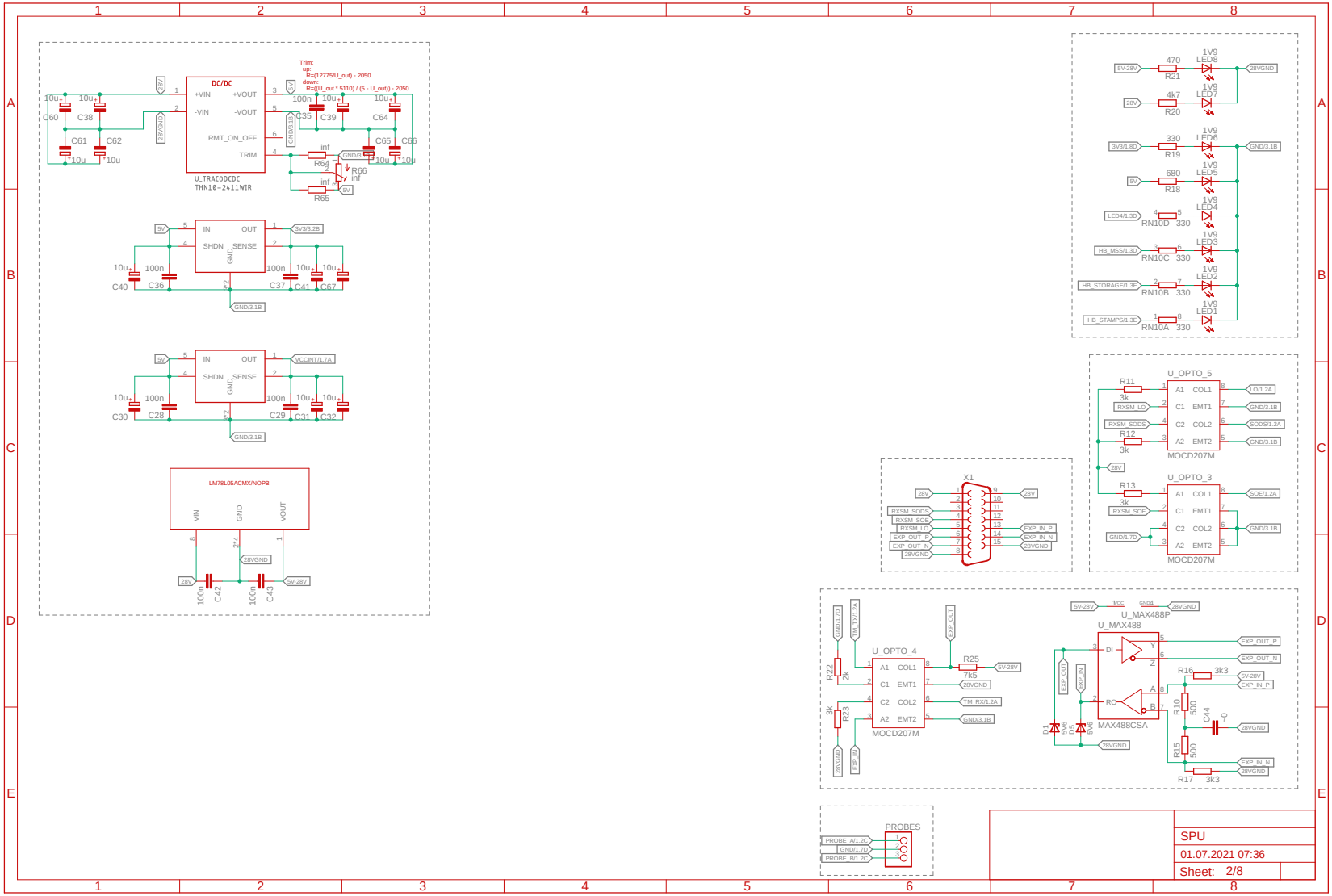
I hereby assure to have written the present work independently and without external help, properly marked the quotes and not used any other sources and aids specified in the bibliography.

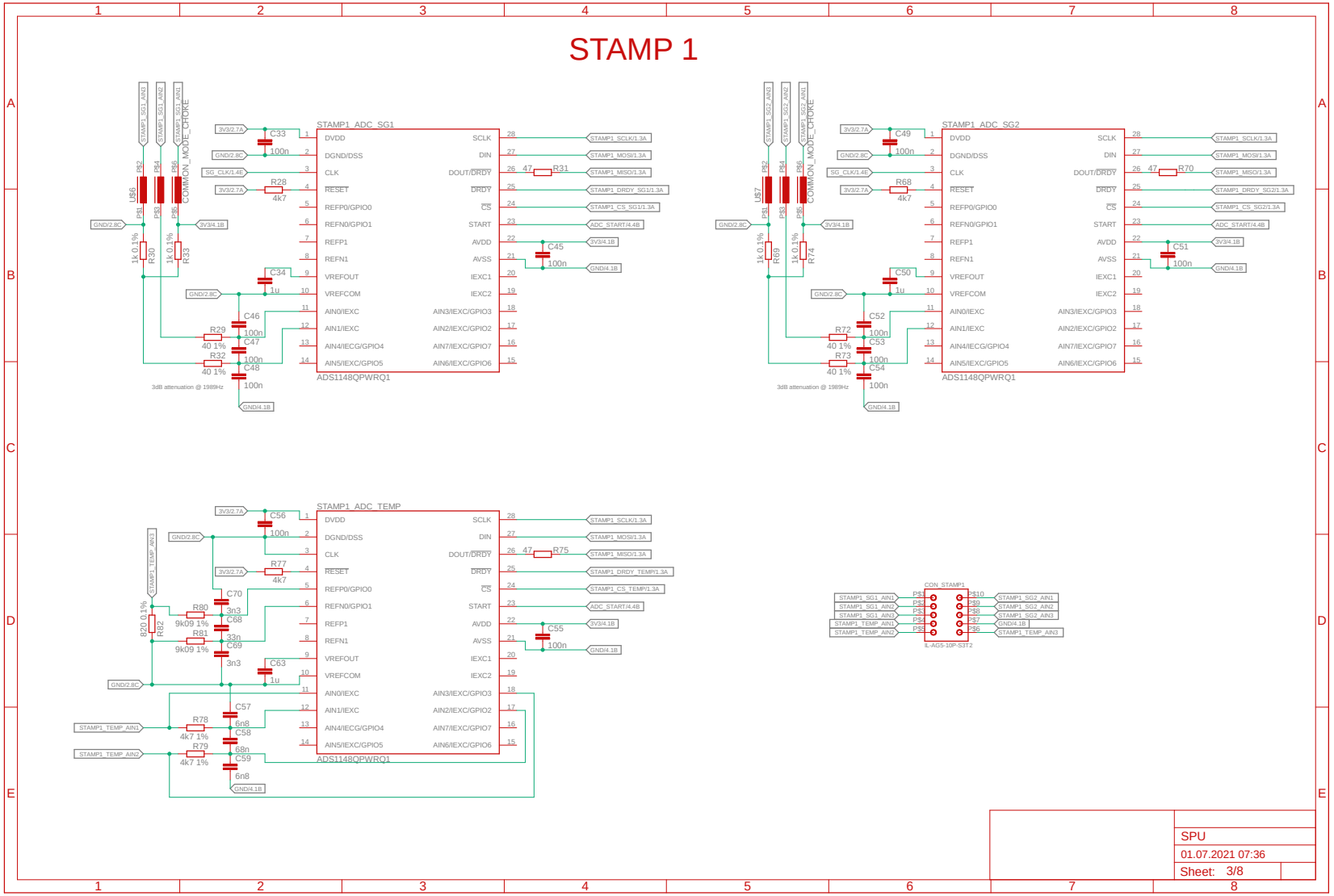
Jonathan Lusky

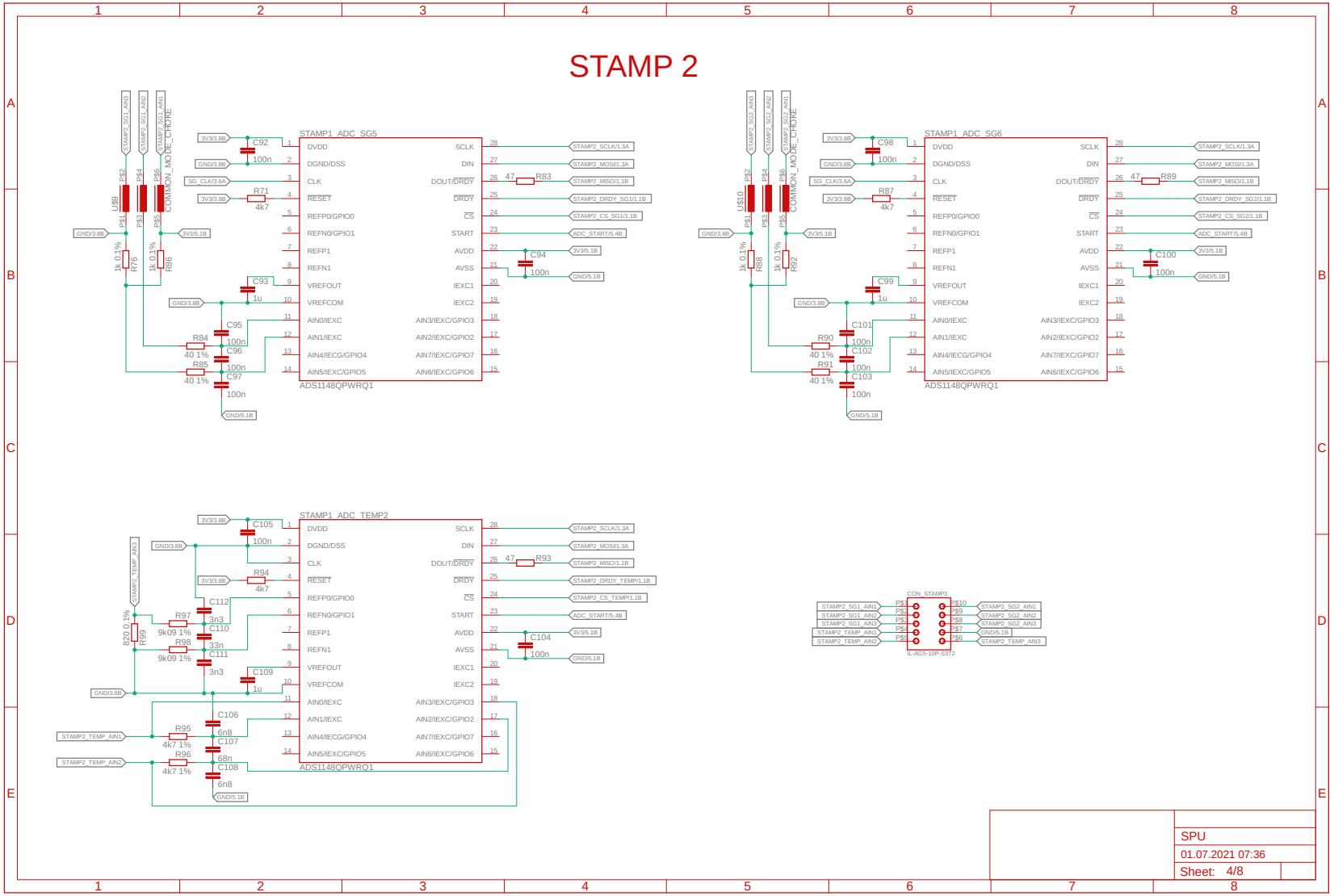
8 | APPENDICES

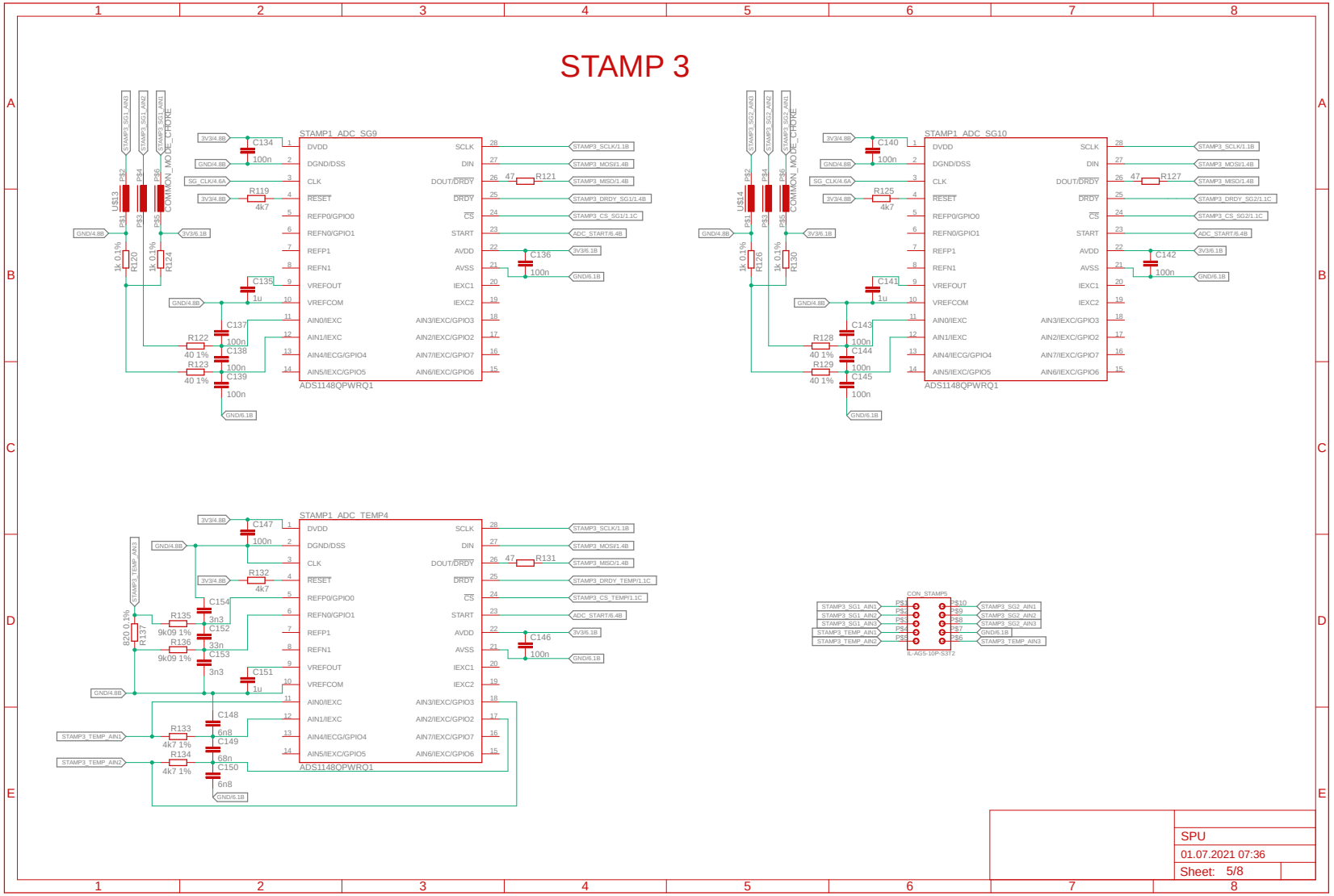
A SPU schematics

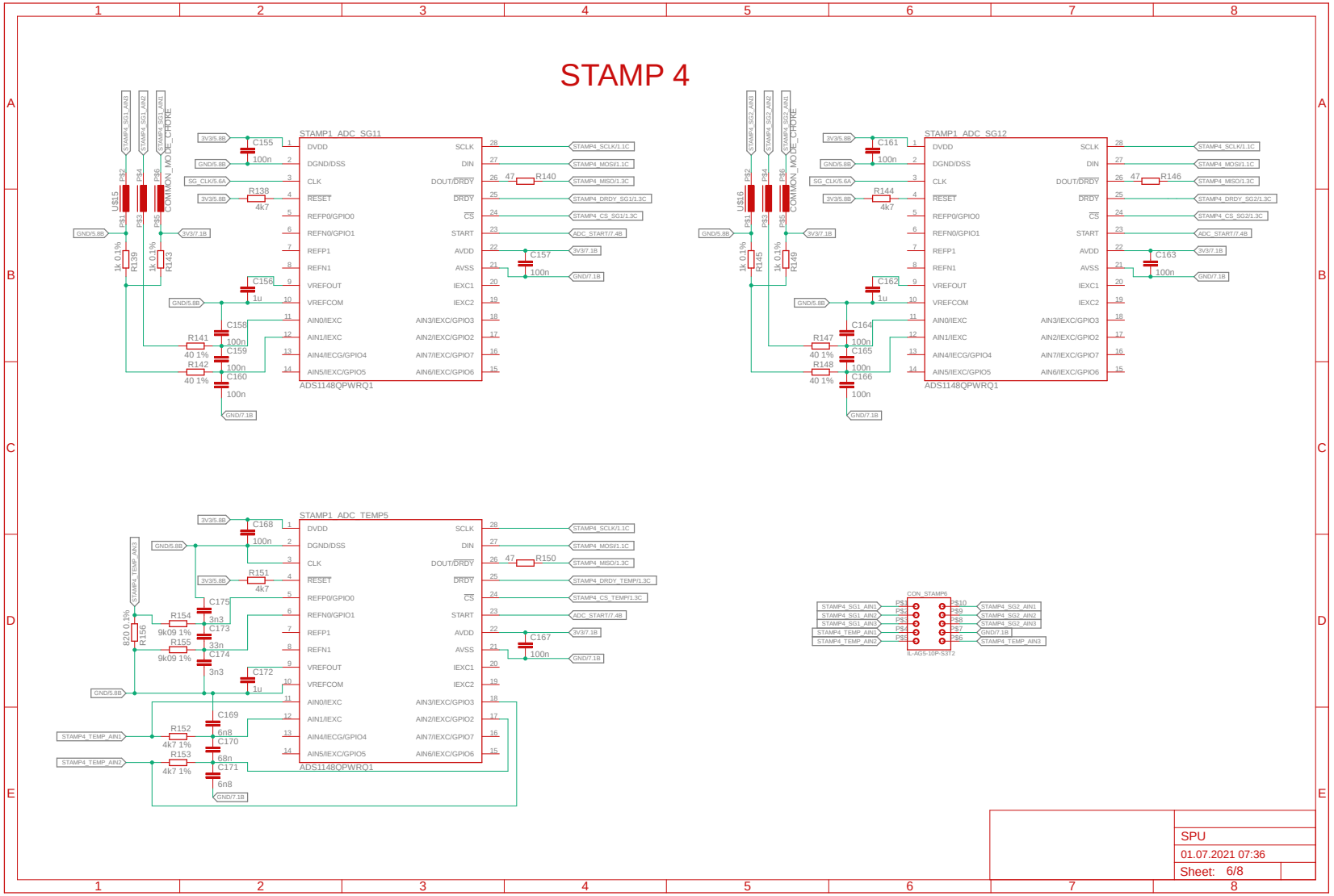

SPU
01.07.2021 07:36
Sheet: 1/8
STAMP1
STAMP2
STAMP3
STAMP4
STAMP5
STAMP6
FTDI
FT2232D
S70FL
SF2_M2S010T1G$1
SF2_M2S010T1G$2
SF2_M2S010T1G$3
SF2_M2S010T1G$4
SF2_M2S010T1G$5
SF2_M2S010T1G$6
SF2_M2S010T1G$7
BANK0
BANK2
BANK3
BANK4
BANK6
BANK7
General Bank
CN1
FT_EEPROM
CSTCR6M00G53Z
S70FL01GSAGM
DSC1001CI1-004.0000T

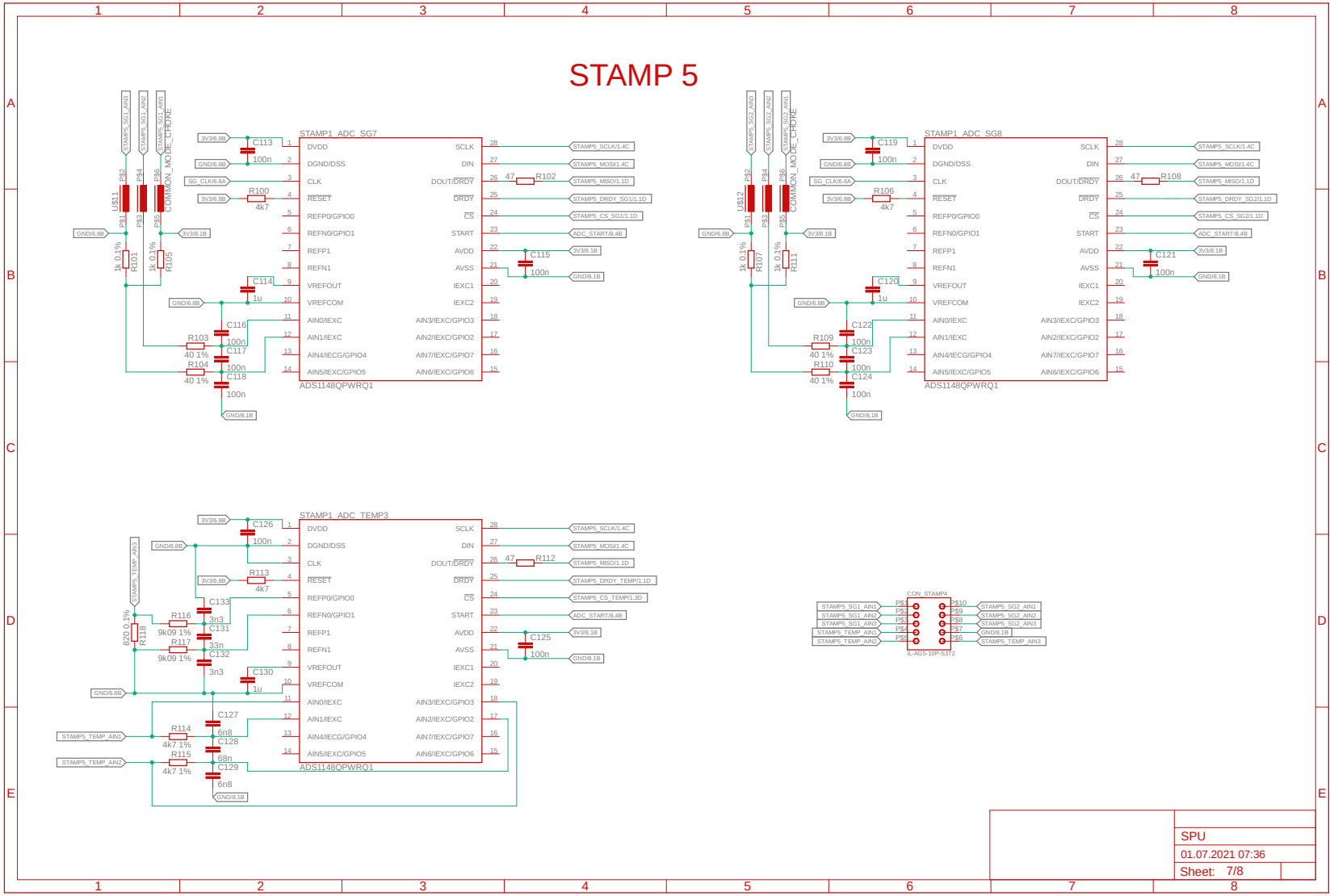

DC/DC
+VIN
+VOUT
-VIN
-VOUT
RMT_ON_OFF
TRIM
V_TRACODCDC
THN10-2411WIR
LM78L05ACMX/NOPB
VIN
GND
VOUT
U_MAX488P
U_MAX488
MAX488CSA
U_OPTO_4
U_OPTO_5
U_OPTO_3
MOCD207M
X1
PROBES
SPU
01.07.2021 07:36
Sheet: 2/8

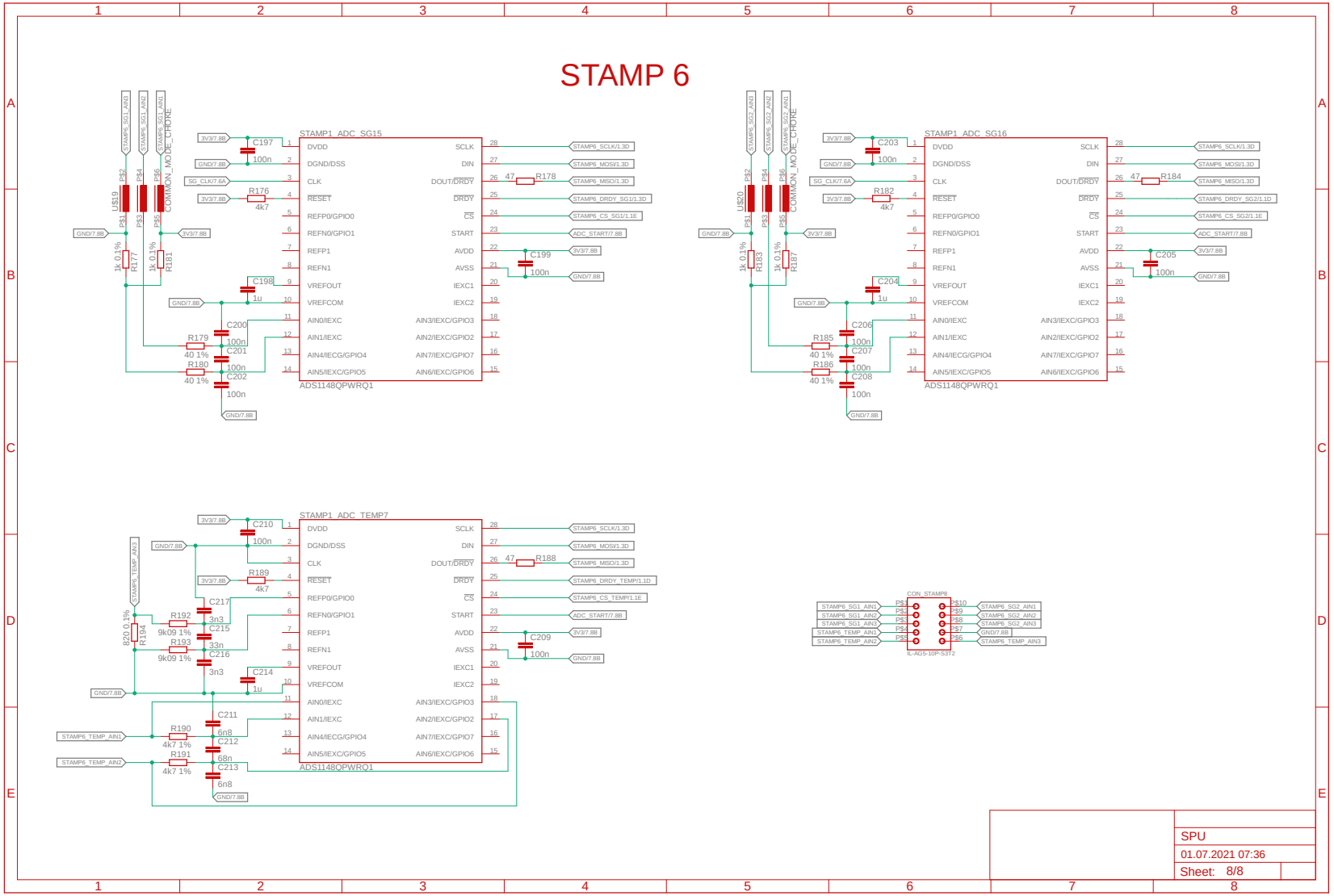

STAMP 1
STAMP1_ADC_SG1
STAMP1_ADC_SG2
STAMP1_ADC_TEMP
ADS1148QPWRQ1
COMMON_MODE_CHOKE
CON_STAMP1
IL-AG5-10P-S3T2
3dB attenuation @ 1989Hz
SPU
01.07.2021 07:36
Sheet: 3/8


STAMP 2
STAMP1_ADC_SG5
STAMP1_ADC_SG6
STAMP1_ADC_TEMP2
ADS1148QPWRQ1
DVDD
DGND/DSS
CLK
RESET
REFP0/GPIO0
REFN0/GPIO1
REFP1
REFN1
VREFOUT
VREFCOM
AIN0/IEXC
AIN1/IEXC
AIN5/IEXC/GPIO5
SCLK
DIN
DOUT/DRDY
DRDY
CS
START
AVDD
AVSS
IEXC1
IEXC2
AIN3/IEXC/GPIO3
AIN2/IEXC/GPIO2
AIN7/IEXC/GPIO7
AIN6/IEXC/GPIO6
AIN4/IECG/GPIO4
CON_STAMP3
IL-AGS-10P-S3T2
SPU
01.07.2021 07:36
Sheet: 4/8

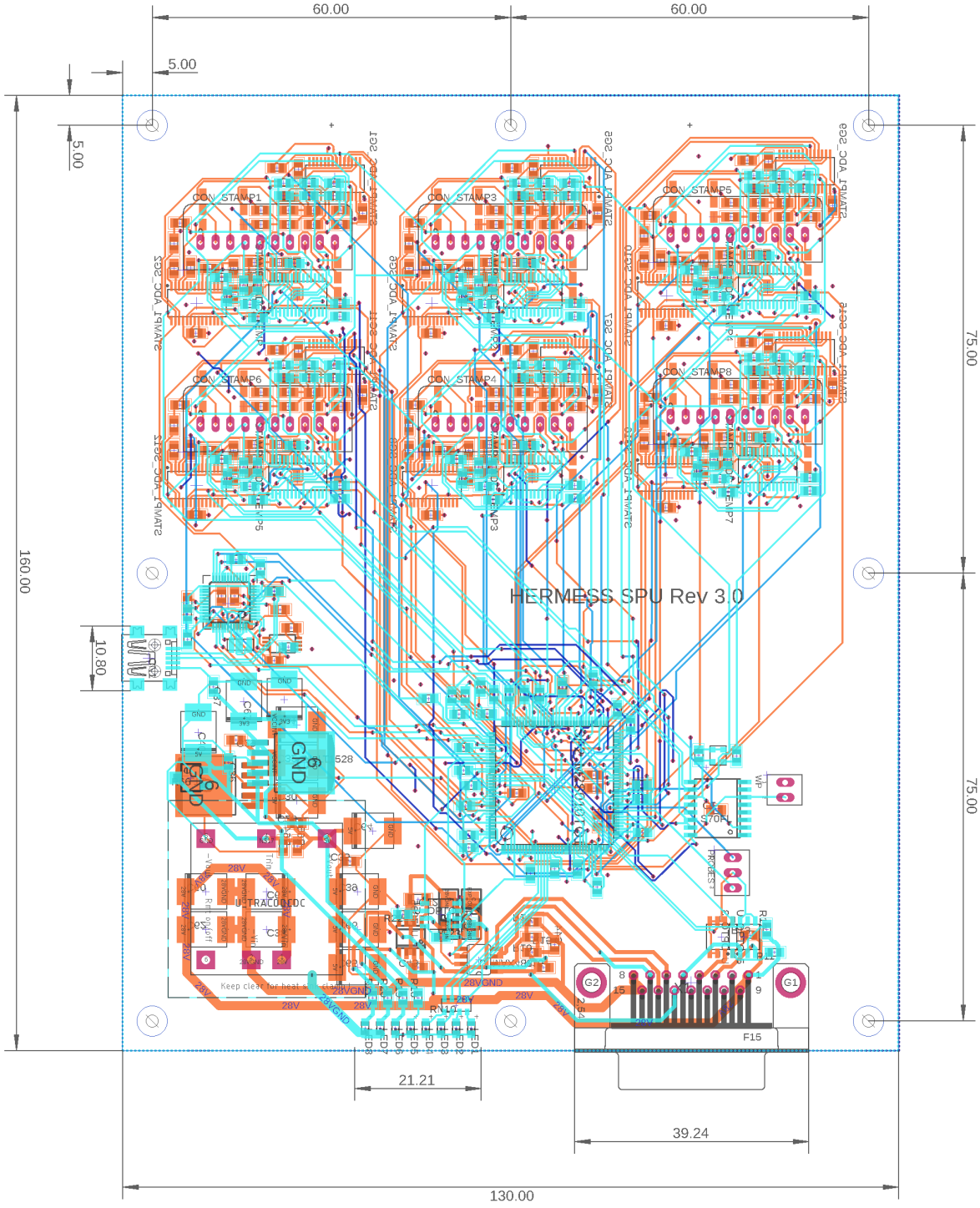

STAMP 3
STAMP1_ADC_SG9
STAMP1_ADC_SG10
STAMP1_ADC_TEMP4
ADS1148QPWRQ1
COMMON_MODE_CHDRE
CON_STAMP5
IL-AGS-10P-S3T2
SPU
01.07.2021 07:36
Sheet: 5/8


STAMP 4
STAMP1_ADC_SG11
STAMP1_ADC_SG12
STAMP1_ADC_TEMP5
ADS1148QPWRQ1
DVDD
DGND/DSS
CLK
RESET
REFP0/GPIO0
REFN0/GPIO1
REFP1
REFN1
VREFOUT
VREFCOM
AIN0/IEXC
AIN1/IEXC
AIN4/IECG/GPIO4
AIN5/IEXC/GPIO5
SCLK
DIN
DOUT/DRDY
DRDY
CS
START
AVDD
AVSS
IEXC1
IEXC2
AIN3/IEXC/GPIO3
AIN2/IEXC/GPIO2
AIN7/IEXC/GPIO7
AIN6/IEXC/GPIO6
COMMON_MODE_CHOKE
CON_STAMP6
SPU
01.07.2021 07:36
Sheet: 6/8


STAMP 5
STAMP1_ADC_SG7
STAMP1_ADC_SG8
STAMP1_ADC_TEMP3
ADS1148QPWRQ1
CON_STAMP4
IL-AGS-10P-S3T2
SPU
01.07.2021 07:36
Sheet: 7/8


STAMP 6
STAMP1_ADC_SG15
STAMP1_ADC_SG16
STAMP1_ADC_TEMP7
ADS1148QPWRQ1
DVDD
DGND/DSS
CLK
RESET
REFP0/GPIO0
REFN0/GPIO1
REFP1
REFN1
VREFOUT
VREFCOM
AIN0/IEXC
AIN1/IEXC
AIN4/IECG/GPIO4
AIN5/IEXC/GPIO5
SCLK
DIN
DOUT/DRDY
DRDY
CS
START
AVDD
AVSS
IEXC1
IEXC2
AIN3/IEXC/GPIO3
AIN2/IEXC/GPIO2
AIN7/IEXC/GPIO7
AIN6/IEXC/GPIO6
COMMON_MODE_CHOKE
CON_STAMP8
SPU
01.07.2021 07:36
Sheet: 8/8

B SPU PCB layout



C Web resources

Many resources like specifications, source codes and design files developed for this thesis were made publicly available.

Hardware Layout: A git repository primarily containing the SPU hardware design files compatible with *Autodesks Eagle* software. It furthermore includes all used non managed libraries and the configuration file for the used FTDI IC.
https://github.com/HERMESSSignalProcessingSoftware/HardwareLayout

Software and FPGA design: A git repository whose branch "sm2" contains all *Libero SoC* project files for the FPGA design including simulation files. Unfortunately every resynthesizing with subsequent commiting produces many changes in the repositories history, because no minimally working set of files could be determined to exclude all others from versioning. The directory "mssSrc" contains the firmware files.
https://github.com/HERMESSSignalProcessingSoftware/MicroController

Groundstation software: A git repository containing the kotlin / maven source code files alongside instructions on how to compile and execute them.
https://github.com/HERMESSSignalProcessingSoftware/GroundStation

Versioned specifications: A set of versioned interface specifications. Specifically, this git repository contains a specification for the DAPI and another one for the TM stream.
https://github.com/HERMESSSignalProcessingSoftware/VersionedSpecs