UFBA
UNIVERSIDADE NOVA

SERVIÇO PÚBLICO FEDERAL

MINISTÉRIO DA EDUCAÇÃO

UNIVERSIDADE FEDERAL DA BAHIA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Volker Kible

An UHF Reflection Coefficient Measurement Block Based on Injection Locking and Quadrature Amplitude Demodulation

Salvador, 2016

Universidade Federal da Bahia

Departamento de Engenharia Elétrica

Programa de Pós-Graduação em Engenharia Elétrica

An UHF Reflection Coefficient Measurement Block Based on Injection Locking and Quadrature Amplitude Demodulation

Volker Kible

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal da Bahia como parte dos requisitos necessários para obtenção do grau de Mestre em Engenharia Elétrica.

Área de Concentração: Processamento de Sinais
Linha de Pesquisa: Microeletrônica em RF

Robson Nunes de Lima
(Orientador)

Salvador, Bahia, Brasil

K46 Kible, Volker

An UHF reflection coefficient measurement block based on injection locking and quadrature amplitude demodulation / Volker Kible. - Salvador, 2016.

170 f. : il. color.

Orientador: Prof. Dr. Robson Nunes de Lima.

Dissertação (mestrado) – Universidade Federal da Bahia, Escola Politécnica, 2016.

1. Circuitos eletrônicos. 2. Rádio. 3. Medidas elétricas. 4. Casamento de Impedância. I. Lima, Robson Nunes de. II. Universidade Federal da Bahia. III. Título.

CDD 621.381 32

An UHF Reflection Coefficient Measurement Block
Based on Injection Locking and Quadrature Amplitude Demodulation

Volker Kible

Dissertação de Mestrado aprovada em 1 de novembro de 2016 pela banca examinadora composta pelos seguintes membros

____
Professor Dr. Robson Nunes de Lima
Orientador – UFBA

____
Professor Dr. Amauri Oliveira
Examinador – UFBA

____
Professor Dr. Edson Pinto Santana
Examinador – UFBA

____
Professor Dr. Fabrício Gerônimo Simões Silva
Examinador externo - IFBA

Acknowledgments

I would like to express my deepest gratitude to my advisor Prof. Dr. Robson Nunes for all his help, counselling and patience.

I want to extend my appreciation to Prof. Dr. Edson Pinto Santana for his help not only with the computers, Prof. Dr. Ana Isabela Araújo Cunha for deepening my understanding of transistors and for her caring and Daniel Prado for his help with producing printed circuit boards.

I also would like to extend my deepest gratitude to my wife Karolinne and to my parents Rose and Dieter, both for their technical and emotional support.

I also want to thank CAPES and CNPq for supporting the work financially.

I finally want to thank everybody who helped directly or indirectly in the progress of this work: the other students, professors and employees of the Laboratório de Concepção de Circuitos Integrados (LCCI-UFBA), the Programa de Pós-Graduação em Engenharia Elétrica (PPGEE-UFBA) and the Departamento de Engenharia Elétrica (DEE-UFBA).

Abstract

Modern telecommunication networks often employ handheld devices, with a trend towards wearable devices. For medical purposes, there are even implanted devices that communicate via a radio frequency interface. All these devices have an antenna in common, whose input impedance varies when parts of the human body or objects made of metal are close. This variation can degrade the performance of the power amplifier of the radio frequency interface. So, it is of high interest to find possibilities to protect radio frequency power amplifiers against such load impedance variation.

One possible strategy to protect the power amplifier is by means of an automatic impedance matching system, which dynamically adjusts the matching network between the power amplifier and the antenna dynamically according to the impedance variation. In order to adjust the matching network correctly, it is important to obtain accurate information about the antenna input impedance. A good way to do this is by measuring the power amplifier's load reflection coefficient.

A number of scientific works have explored various ways of gaining information about the load reflection coefficient using diode power detectors. Diode power detectors generally require linearization techniques, especially when being subjected to strong input power variations as can be expected in the transmission path.

This work explores the use of a combination of injection locking and quadrature amplitude demodulation to overcome this problem. A complete reflection coefficient measurement system was designed and evaluated by means of computer simulations and measurements. These measurements on a printed circuit board setup show the basic function of the system, while at the same time paving the way for a future integrated circuit setup. An average absolute error of 0.044 at 30 dBm and 0.024 at lower power levels (11 dBm, 18 dBm, 24 dBm) was found in the computer simulations. In the measurements, an average absolute error of 0.146 and a maximum absolute error of 0.28 was obtained.

Resumo

Nos sistemas de comunicação modernos, a mobilidade e portabilidade dos dispositivos são uma tendência. Para fins médicos, há ainda dispositivos implantados que se comunicam através de uma interface de radiofrequência (RF). Todos esses dispositivos fazem uso de antenas, cuja impedância de entrada varia com a proximidade de outros objetos, tais como a cabeça do usuário. Esta variação pode ser problemática para o amplificador de potência, o qual é parte fundamental na interface de radiofrequência. Por isso, é de grande interesse para a indústria encontrar possibilidades para a proteção dos amplificadores de potência de RF contra a variação de impedância de carga.

Uma abordagem possível para proteger o amplificador de potência é feita utilizando um sistema automático de adaptação de impedâncias, o qual ajusta dinamicamente a rede de adaptação entre o amplificador de potência e a antena, de acordo com a variação de impedância. Para ajustar a rede de adaptação corretamente, é importante obter informações acerca da impedância de entrada da antena. Uma boa maneira de fazer isso é através da medição do coeficiente de reflexão da carga do amplificador de potência.

Existe na literatura diferentes maneiras de medir componentes do coeficiente de reflexão da carga ou medir o coeficiente total na forma polar usando detectores com diodos. Para esse método, geralmente, faz-se necessário o uso de técnicas de linearização, especialmente quando o sistema está submetido a fortes variações na potência de entrada, como se pode esperar no caminho da transmissão.

Este trabalho explora a utilização de uma combinação da técnica do *injection locking* e da demodulação em quadratura para superar este problema. Um sistema completo de medição do coeficiente de reflexão foi projetado, avaliado por meio de simulações com o *software Keysight ADS* (*Advanced Design System*) e, em seguida, implementado. Medições em uma placa de circuito impresso mostram as funções básica do sistema, enquanto que ao mesmo tempo, abrem caminho para um futuro projeto em circuito integrado. Um erro absoluto médio de 0,044 a 30 dBm e 0,024 em níveis mais baixos de potência (11 dBm, 18 dBm, 24 dBm) foi encontrado nas simulações de computador. Nas medições, um erro absoluto médio de 0,146 e um erro absoluto máximo de 0,28 foi obtido.

Table of Contents

List of Figures

List of Tables

List of Acronyms

A	Attenuation
ADC	Analog to digital converter
ADS	Advanced design system
AIM	Automatic impedance matching
a_x	Ingoing wave at port x
b_x	Outgoing wave at port x
C	Capacitor
C1, C2, C3	Capacitors of variable impedance matching network
C_{BC}	Base-collector-capacitance
CPL	Coupled port
Cx	Quadrature coupling capacitor
D	Directivity
DC	Direct current
E, F, G, H	Proprietary correction values (segmented transmission line)
EPC	Equivalent parallel capacitance
Err	Absolute error
ESL	Equivalent series inductance
ESR	Equivalent series resistance
f_0	Free-running frequency of oscillator
F303	STM32F303 microcontroller (ST Microelectronics)
F3DISCO	STM32F3 discovery board (ST Microelectronics)

F429	STM32F429 microcontroller (ST Microelectronics)
F429DISCO	STM32F429 discovery board (ST Microelectronics)
f_c	Cutoff-frequency
f_{RF}	RF-frequency
g_C	Current gain
GND	Ground
G_P	Power gain
G_T	Transducer power gain
I_out	In-phase component of parasitic phase angle inside Frontend
IC	Integrated Circuit
IDE	Integrated development environment
IN	Input port
IQD	IQ-Demodulation / Quadrature amplitude demodulation
ISM	Industrial, scientific and medical
ISO	Isolated port
j	Imaginary unit
K	Rollett stability factor
LC	Inductor-Capacitor
LCCI	Laboratório de concepção de circuitos integrados
LED	Light emitting diode
LF	Low frequency
LFA	Low frequency amplifier
LO	Local oscillator (input of mixers)
MICS	Medical implant communication service
MOSFET	Metal oxide semiconductor field effect transistor
N	Winding ratio of a transformer
op-amp	Operational amplifier
o_x	Offset measured by ADC x
PA	Power amplifier
PAE	Power added efficiency
P_{AVS}	Power available from source

PCB	Printed circuit board
p_i	Re{V_i}
P_L	Power delivered to load
p_r	Re{V_r}
PWR	Power
Q	Quality factor
Q_out	Quadrature component of parasitic phase angle inside Frontend
QAi	Input impedance matching network
QAo	Output impedance matching network
q_i	Im{V_i}
QILO	Quadrature injection locking oscillator
q_r	Im{V_r}
R	Resistor
r_B	Base resistance
RC	Resistor-Capacitor
RF	Radio frequency
RL	Return loss
RLB	Return loss bridge
SAR	Successive approximation register
S_{ij}	S-Parameter between port j and port i. If i=j, reflection coefficient at this port.
SMT	Surface mounting technology
SRF	Self-resonant frequency
SWR	Standing wave ratio
s_x	Scaling factor for ADC x
t	Time
TRA	Transmitted/Output port
TX	Transmitting
UHF	Ultra high frequency band
V_i	Incident wave
Vi_I, c	In-phase component of incident wave
Vi_Q, d	Quadrature component of incident wave

$V_{inj,p}$	Amplitude of the signal that is injected into the oscillator
VNA	Vector network analyzer
$V_{osc,p}$	Oscillation amplitude of oscillator
V_r	Reflected wave
Vr_I, a	In-phase component of reflected wave
Vr_Q, b	Quadrature component of reflected wave
VSWR	Voltage standing wave ratio
V_T	Thermal voltage (ca. 26 mV)
V_x	Voltage at port x
v_x	Voltage measured by the ADC x
Z_0	Reference impedance
Z_L	Load impedance
Z_S	Source impedance
α	Angle between X-Axis and A-Axis of non-rectangular coordinate system
β	Angle between X-Axis and B-Axis of non-rectangular coordinate system
γ	Angle between X-Axis and C-Axis of non-rectangular coordinate system
Γ_i	Ideal value for measured reflection coefficient (from VNA)
Γ_{in}	Input reflection coefficient
Γ_L	Load reflection coefficient
Γ_{min}	Allowed reflection coefficient error (real value, just magnitude)
Γ_t	Reflection coefficient seen at input of directional coupler
Δ	Rollett stability factor delta value
δ	Angle between X-Axis and D-Axis of non-rectangular coordinate system
θ	Phase deviation due to injection locking
θ_a	Absolute output phase deviation due to injection locking
τ	Time constant
φ	Phase difference between reflected and incident wave
ω	Angular frequency
ω_0	Free-running angular frequency of oscillator
ω_{inj}	Angular frequency of the signal that is injected into the oscillator

1 Introduction

This dissertation explores aspects of automatic impedance matching systems for protecting the radio frequency (RF) power amplifier (PA) against variations of the load impedance. It focuses on the impedance measuring block of such a system. This chapter introduces the RF PA, the problem of load impedance variation and possible solutions to protect the power amplifier, ending with the impedance measuring block of an automatic impedance matching system.

The purpose of a radio frequency power amplifier is generally to amplify the signal in the transmitting (TX) path of a RF communication system before feeding it into the antenna. A typical RF PA comprises of an active element (e.g. a transistor), two bias networks and two fixed impedance matching networks, one for the input (QAi) and another one for the output (QAo) as shown in Figure 1.1.

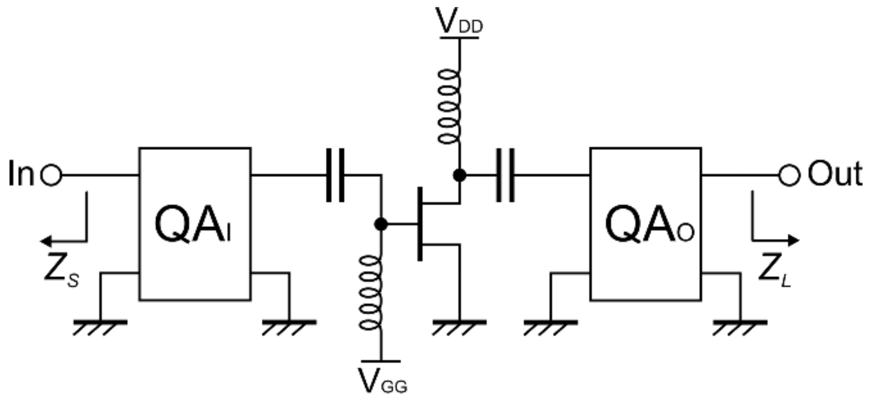


Figure 1.1: A typical Radio Frequency Power Amplifier

The output impedance of the blocks before the PA (such as a driver stage or a filter) and thus the source impedance (Z_S) for the PA is often well defined. The same is not necessarily true for the load impedance (Z_L), that often varies according to effects of objects close to the antenna. Even so, RF PAs are generally designed for a given load impedance, often 50 Ω.

To make an RF PA less sensitive to variations of the load impedance Z_L is of high interest. A mismatch condition between the output of the PA and the load impedance is a common state in portable RF transmission systems, where the antenna appears as a load impedance which is varying with its close surroundings such as the head or hand of the user or the metal of a car or lift cabin [1], [2]. A load impedance mismatch results in power reflected back to the PA as shown in Figure 1.2. This severely degrades the performance of the PA and can, in certain conditions, even destroy it [3].

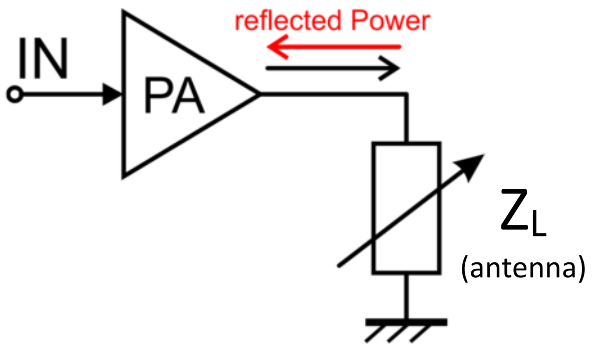


Figure 1.2: Load Impedance Mismatch and Reflected Power

The performance degrading effect of load impedance mismatch on the PA is already known for quite a time, and many scientific works have been done to mitigate it. Earlier works tended to focus on preventing the PA from being destroyed by clamping the output voltage to an allowable value as shown in Figure 1.3 (a) or by reducing the gain of the PA depending on the output voltage or the reflected signal as shown in Figure 1.3 (b) [4], [5]. These approaches don't fully address the performance degradation of the PA induced by the load mismatch. They only remove the problem of over voltage, but not of degradation of efficiency, phase distortion and non-linearity.

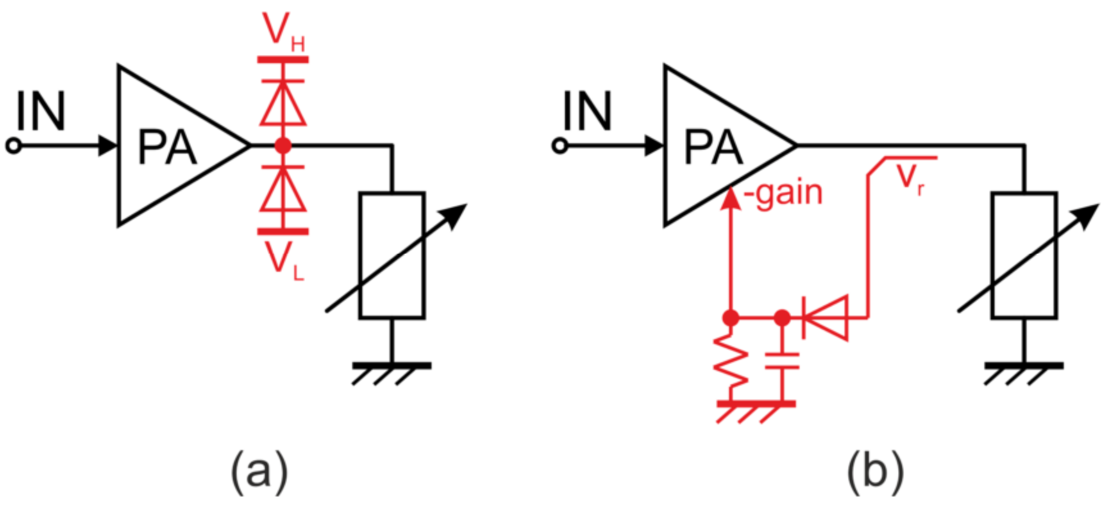


Figure 1.3: PA Protection by (a) Clamping, (b) Gain Adjustment

Other works also include outphasing amplifiers or balanced PA and passive power combining networks comprising transmission line structures such as couplers and hybrids, sometimes together with tuning elements [6]-[9]. Also, distributed active transformer structures have been used [10]-[12]. All these techniques have in common that the susceptibility to load impedance variations can be made lower, but they still do not present the best possible matching condition to the PA output.

Instead, it is possible to construct an Automatic Impedance Matching (AIM) System, using tunable or reconfigurable impedance matching networks composed of transmission lines or inductors and capacitors together with RF switches [3], [13], [14], or tuning elements such as varactors [15]-[17]. Many of the more recent works have explored this method, and some of them have been done at the Universidade Federal da Bahia, developing a general approach to designing such impedance matching networks [3]. This method matches the varying load impedance dynamically and, as such, always presents a minimal mismatch condition to the output of the PA. The higher the number of possible matching states, the lower is the maximum remaining mismatch. This work will explore aspects of an AIM System. As presented in Figure 1.4, an AIM System comprises of three components: The variable impedance matching network (which is controlled), the control block (using measured information about the load impedance) and the measuring block (for obtaining information about the load impedance).

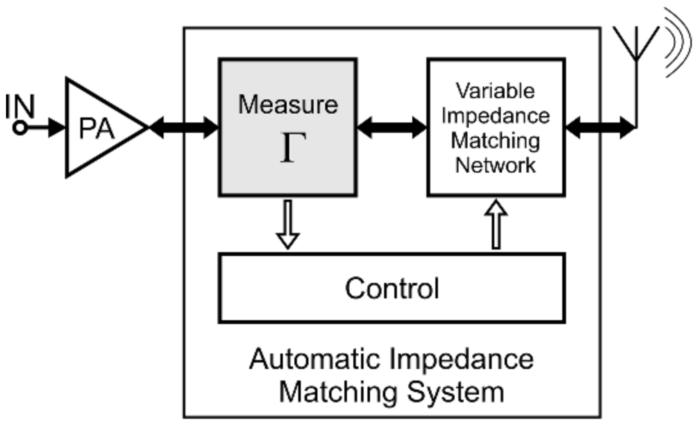

IN
PA
Measure
Γ
Variable Impedance Matching Network
Control
Automatic Impedance Matching System

Figure 1.4: Power Amplifier with Automatic Impedance Matching System

In order to tune or reconfigure the matching/combining circuit, or to change the phase in the outphasing amplifier configuration, it is necessary to acquire information about the actual impedance Z_L, or alternatively about the load reflection coefficient Γ_L. The load reflection coefficient is the quotient of the reflected wave V_r over the incident wave V_i as shown in equation (1.1) [18], or alternatively can be expressed by means of source and load impedance as shown in equation (1.2) [18]. If needed, Z_L can be calculated from Γ_L via equation (1.3) [18], using the known source impedance Z_S.

$$\Gamma_L = \frac{V_r}{V_i} \tag{1.1}$$

$$\Gamma_L = \frac{Z_L - Z_S}{Z_L + Z_S} \tag{1.2}$$

$$Z_L = Z_S \frac{1 + \Gamma_L}{1 - \Gamma_L}. \tag{1.3}$$

Many recent works in the area of automatic impedance matching systems already explored the variable impedance matching network and the search algorithms [14]-[16]. Instead, this work focuses on the less researched reflection coefficient measuring block of the AIM System and in parts on the control block, though an indication for the variable impedance matching block will also be given.

The objective of this work is the development of a new reflection coefficient measurement system, including computer simulations and measurements with a discrete setup. Additional blocks of an automatic impedance matching system are also considered.

This work is divided into five chapters. In chapter 2 a bibliographic review with general considerations about the most common existing techniques for measuring the reflection coefficient is presented. In this chapter, the injection locking phenomenon and its use in a measurement system is also approached.

In chapter 3 the proposed reflection coefficient measurement system is discussed in detail together with simulation results of the functional blocks and of the whole system. The control block and the variable impedance matching network are considered in this chapter as well.

In chapter 4 the measurement setups and results for validating the function of the system are presented.

The final chapter 5 draws conclusions and presents perspectives for future works.

2 RF Impedance Measurement Techniques

In general, the load impedance and the reflection coefficient are complex numbers. If just the magnitude of the reflection coefficient is of interest, ordinary Standing Wave Ratio (*SWR*) or Return Loss (*RL*) meters can be used. Voltage SWR (*VSWR*) and *RL* are real numbers and can be calculated from the magnitude of the reflection coefficient $|\Gamma_L|$ using the following equations (2.1) and (2.2) [18].

$$VSWR = \frac{1 + |\Gamma_L|}{1 - |\Gamma_L|}. \tag{2.1}$$

$$RL = -20 \log |\Gamma_L|. \tag{2.2}$$

In an automatic impedance matching system, however, it is difficult to match the load if only the magnitude of the reflection coefficient is measured and the phase is omitted. Such an approach requires complicated search algorithms to find an optimal matching state. In the following sections, different approaches to measure the reflection coefficient as a complex number will be detailed, making the use of simple matching algorithms possible.

2.1 Return-Loss-Bridge

The schematic of a typical return-loss-bridge (RLB) is given in Figure 2.1 [19]. A RLB allows to measure the reflection coefficient directly if the output is analyzed as a complex number and not just its magnitude [20]. This can be done in polar form or in Cartesian form like discussed in more detail in the sections 2.4 and 2.5. However, the RLB has one big disadvantage: 75% of the power fed into it by the RF source is lost in the resistors, making it impractical at the output branch of an RF power amplifier.

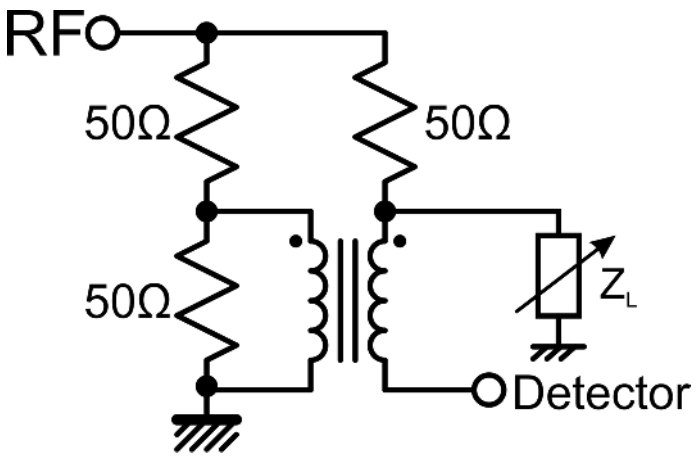


Figure 2.1: Return Loss Bridge (Z_0 = 50 Ω)

2.2 Six-Port and Similar Techniques

Another technique for measuring the reflection coefficient is the six-port reflectometer as proposed in [13]. The phase information is derived by computation from measuring the amplitude (related to the power) at four defined points in an interferometer circuit [21]. This technique is more suitable for use after a PA, as it allows for higher powers to be passed through the measuring circuit. Normally, there are no resistors in the power path. The resulting insertion losses for the six port technique are typically below 1dB in the main path, comparable to couplers. A schematic of the six-port reflectometer technology is shown in Figure 2.2.

The voltage V_x on each port x (x = 1, 2, 3 or 4), and also the amplitude $|V_x|$ that is sampled by the detectors, can be expressed by means of the ingoing wave a_x and outgoing wave b_x, and the reference impedance Z_0 using the following equation (2.3):

$$V_x = \sqrt{Z_0} \cdot (a_x + b_x) \tag{2.3}$$

As a_6 and b_6 are a function of the constant S-Parameters of the six-port-network, the known reflection coefficients of the detectors and the other ingoing and outgoing waves, it is possible to find the desired reflection coefficient at port 6 from V_1, V_2, V_3, and V_4.

Under certain conditions however, the computation algorithm can be very sensitive to small variations in V_1, V_2, V_3, and V_4 and the S-parameters of input 5, what is not desirable under conditions where noise, for example picked up by the antenna from external sources, and modulation artifacts of the PA itself can occur. This technique generally relies on diode power detectors, for which linearization techniques must be applied.

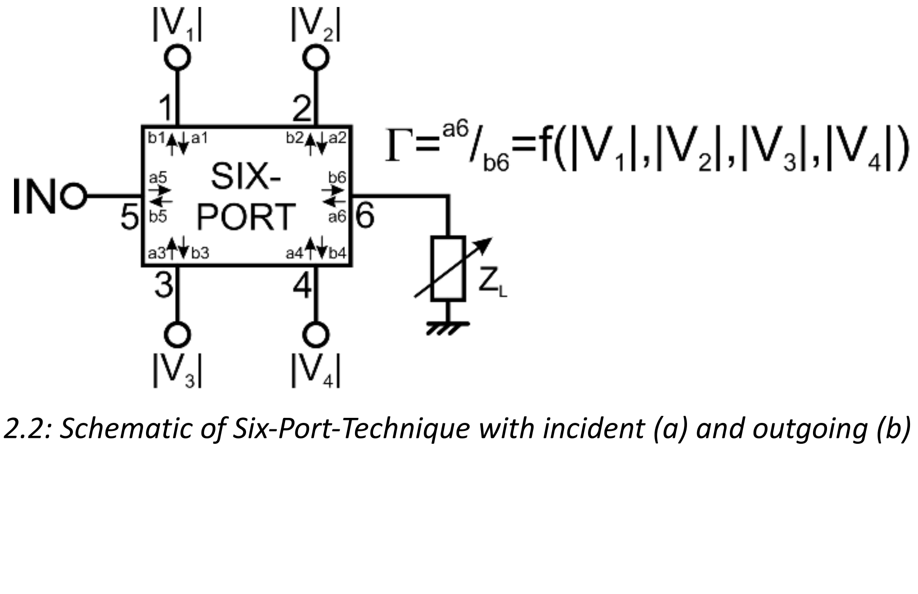


Figure 2.2: Schematic of Six-Port-Technique with incident (a) and outgoing (b) waves.

2.3 Directional Coupler and Circulator

An alternative for the six-port technique to measure the reflection coefficient is using a device to separate the incident signal from the reflected signal. Both directional coupler and circulator allow to separate these two signals, but an RF circulator is normally an expensive and bulky device made with ferrite materials [22] and incompatible with IC technology. However, it allows for additional protection of the PA, as the reflected power is directed to another port than the output of the PA. In [23], active quasi-circulators made of transistors are presented, but in this case the whole quasi-circulator circuit has to be designed for relatively high power levels, has to be protected against load mismatch and reduces the performance of the PA. A schematic on where to measure the incident signal and the reflected signal in a circulator setup is shown in Figure 2.3.

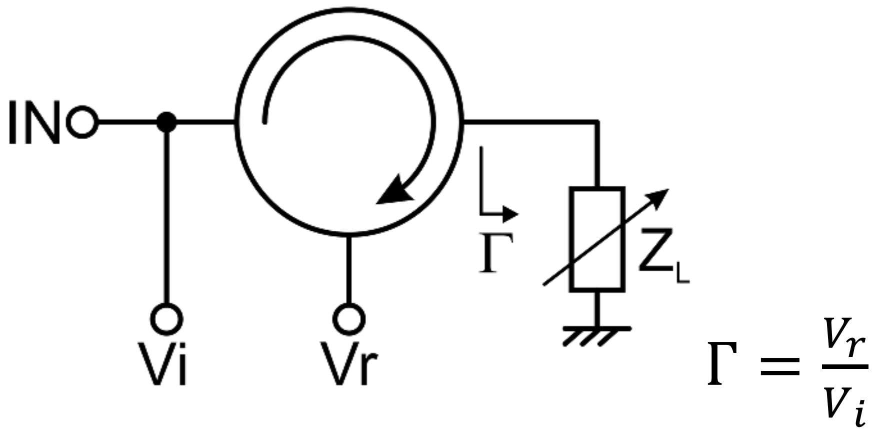


Figure 2.3: Circulator, positions for measuring incident (Vi) and reflected (Vr) wave marked

Directional couplers do not protect the PA from the reflected power. They direct power from the main input port (IN) to the main output port (TRA) and vice-versa, and the attenuation in the main path can be quite low, typically between 0.1dB and 4dB [24]. Even so, this attenuation is the main disadvantage, as it reduces the efficiency of the RF transmission system.

The important property of the directional coupler is that it directs a portion of the incident signal to a third port called “coupled” (CPL) and a portion of the reflected signal to a fourth port, often called “isolated” (ISO), making them available for measurement.

The magnitude of the signals at the CPL and ISO ports is often considerably smaller than the one of the signals at the IN and TRA ports. Commercially available are coupling values between 3dB and 50dB, and for measuring in the power range of interest mostly used are values between 15dB and 20dB [24]. A coupling value of 15dB means that the signal at the CPL port is 15dB lower than the incident signal at the IN port.

Directional couplers are relatively inexpensive devices and can be miniaturized.

Directional couplers for higher frequencies (in the >1 GHz range) are frequently made of parallel transmission lines [25] or slotted hollow wave guides [26]. However, for the 400 MHz frequency range, this leads to large dimensions of the coupler, making it difficult to integrate on a chip. Other approaches have been explored, especially those composed of a single transformer and capacitors [27] (narrow band), capacitors and inductors [28]-[30] or two cross-connected transformers [31] (wide band, schematic shown in Figure 2.4).

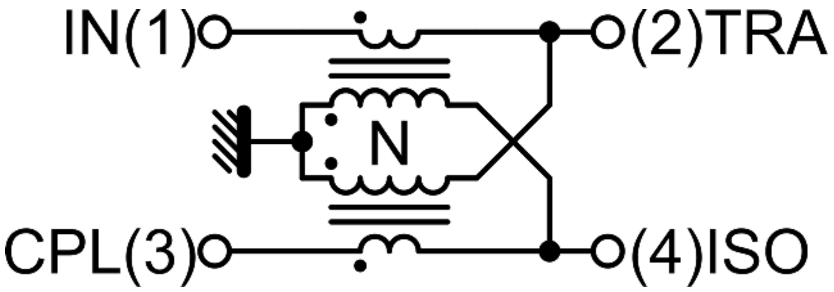


Figure 2.4: Two Transformer Directional Coupler

The coupler made of one transformer and capacitors has already been successfully integrated on a chip [27]. The approach using capacitors and inductors has also been implemented as integrated circuit [28]-[30].

The two-transformer wide band approach has only been applied as discrete coupler with wire wound transformers, but also has some potential for miniaturization.

According to [31], the S-Parameters of a loosely coupled symmetric (winding ratios $N_1 = N_2 = N$) two transformer directional coupler are shown in the following equations (2.4) to (2.8).

$$S_{14} = S_{23} \cong -\frac{1}{2N^3} \approx 0 \tag{2.4}$$

$$S_{11} = -S_{22} = S_{33} = -S_{44} \cong -\frac{1}{2N^2} \approx 0 \tag{2.5}$$

$$S_{12} = S_{34} \cong 1 \tag{2.6}$$

$$S_{13} = -S_{24} \cong -\frac{1}{N} \tag{2.7}$$

$$S_{ij} = S_{ji} \tag{2.8}$$

The directivity D is calculated by equation (2.9):

$$D = 20\log\left|\frac{S_{13}}{S_{23}}\right| = 20\log\left|\frac{-S_{24}}{S_{14}}\right| = 20\log|2N^2| \tag{2.9}$$

The directivity is a number describing how well the coupler separates the incident wave from the reflected wave. It is given as positive value (the higher the better), but also negative values can be found in the literature, equivalent to the inverse of the equation given here. S_{13} denotes the strength of the coupling between IN and CPL, S_{24} is the strength of the coupling between TRA and ISO. These coupling strengths are design parameters of the coupler. S_{14} and S_{23} are the undesired coupling to the other port and should ideally be zero. S_{11}, S_{22}, S_{33} and S_{44} denote the reflection coefficients at the respective ports and should ideally be zero as well. S_{12} is linked to the insertion loss of the coupler and should ideally be one (0dB). The approximate values are the ideal for such a directional coupler, but in reality, not obtainable.

2.4 Polar Reflection Coefficient Measurement

In [32] it is suggested to measure only the phase difference φ between the forward and the reflected signal using amplitude limiters and a Gilbert multiplier cell acting as high frequency phase detector. In addition, they support the opinion that the amplitude ratio is much less important than the phase information and therefore can be omitted. However, in matching circuits with a higher number of possible configurations or tuning steps this does not allow an optimal matching, as the information about the magnitude of the reflection coefficient is lost. This limitation can be overcome by additionally measuring the amplitude of the incident $|V_i|$ and the reflected voltage $|V_r|$ [33]. So in total three analog signals have to be measured as shown in Figure 2.5. In this text, this will be referred to as measuring the reflection coefficient in polar form.

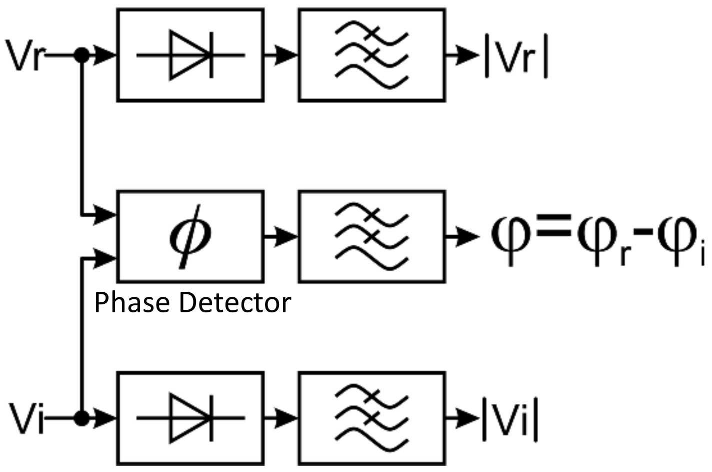


Figure 2.5: Measuring the Reflection Coefficient in Polar Form

The reflection coefficient is calculated from the measured values using the following equation (2.10):

$$\Gamma = \frac{|V_r|}{|V_i|} e^{j \cdot \varphi} \tag{2.10}$$

2.5 Cartesian Reflection Coefficient Measurement

Another approach is the measurement of the reflection coefficient in Cartesian form. This means that the amplitudes of the real and of the imaginary component of the reflection coefficient are measured separately [20]. This can be accomplished by synchronous IQ-demodulation (IQD, also known as quadrature amplitude demodulation) of the reflected signal which is an amplitude detection of the in-phase with the incident signal and 90° out-of-phase components as shown in Figure 2.6. For sinus shaped signals V_{LOI}, V_{LOQ} and V_r, corresponding to the in-phase and quadrature local oscillator signals and the reflected signal, as shown in equation (2.11) the operation of the mixers can be described as in equation (2.12) and (2.13):

$$V_r = A \cdot \cos(\omega t + \varphi)\,,\; V_{LOI} = 2 \cdot \cos(\omega t),\; V_{LOQ} = 2 \cdot \sin(\omega t) \tag{2.11}$$

$$V_{LOI} \cdot V_r = 2 \cdot \cos(\omega t) \cdot A \cdot \cos(\omega t + \varphi) = A \cdot \cos(\varphi) + A \cdot \cos(2\omega t + \varphi) \tag{2.12}$$

$$V_{LOQ} \cdot V_r = 2 \cdot \sin(\omega t) \cdot A \cdot \cos(\omega t + \varphi) = A \cdot \sin(\varphi) + A \cdot \sin(2\omega t + \varphi) \tag{2.13}$$

The double frequency component of the results is removed by a low pass filter, it remains the DC content as shown in equations (2.14) and (2.15):

$$A \cdot \cos(\varphi) = \mathrm{Re}\left\{A \cdot \mathrm{e}^{\mathrm{j}\varphi}\right\} = Vr\_I \tag{2.14}$$

$$A \cdot \sin(\varphi) = \mathrm{Im}\left\{A \cdot \mathrm{e}^{\mathrm{j}\varphi}\right\} = Vr\_Q \tag{2.15}$$

Detection of the incident signal V_i amplitude is also necessary. This can be done synchronously, in other words like in equation (2.14), with $\varphi = 0$. So, there are in total three analog signals to be processed, similar to the polar case.

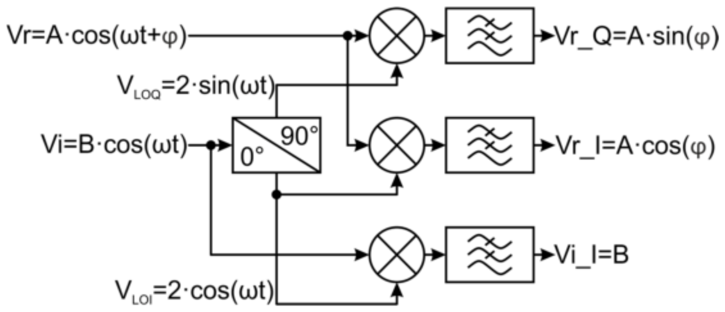


Figure 2.6: Measuring the Reflection Coefficient in Cartesian Form (IQD)

The reflection coefficient is calculated from the measured values using the following equation (2.16):

$$\Gamma = \frac{Vr\_I}{Vi\_I} + \mathrm{j}\frac{Vr\_Q}{Vi\_I} \tag{2.16}$$

However, as will be shown in this work, it is possible to obtain the reflection coefficient in Cartesian form if the IQ-demodulation (IQD) is not synchronous. This means the reference signal for the mixing operation is not in phase with V_i. Just the frequency has to be equal and the phase only has to be constant during one measuring run, so it can be slowly varying. To accomplish this, a fourth analog signal is introduced. Basically, IQD is not only executed for the reflected signal, but also for the incident signal (Figure 2.7). This way information is obtained about the phase between the incident signal and the signal used as reference for the IQD, known as local oscillator (LO). This is crucial when using an injection locking oscillator to generate the quadrature LO signals as proposed here.

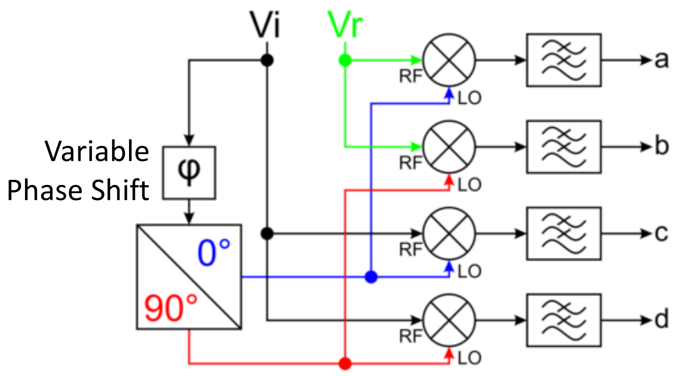


Figure 2.7: Non-Synchronous Cartesian Measurement

The reflection coefficient is calculated from the measured values using the following equation (2.17), that is nothing else than the quotient of two complex numbers V_r = a+jb over V_i = c+jd:

$$\Gamma = \frac{V_r}{V_i} = \frac{ac+bd}{c^2+d^2} + \mathrm{j}\frac{bc-ad}{c^2+d^2} \qquad (2.17)$$

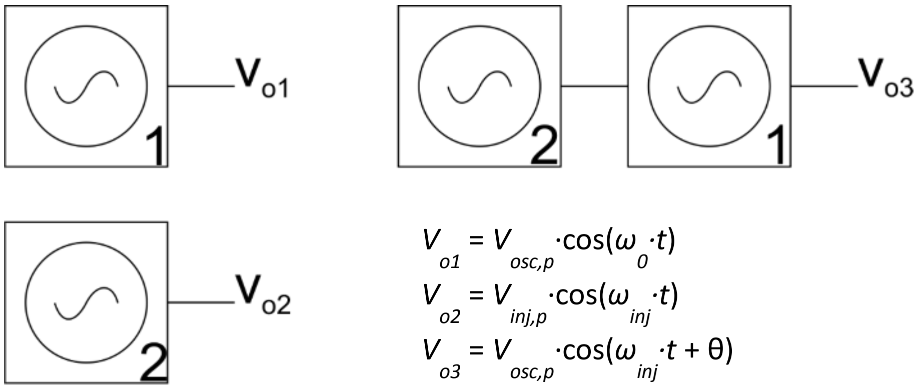
All methods for measuring the reflection coefficient rely on the measurement of amplitude information, what can either be done by synchronous demodulation or by diode power detectors. Synchronous demodulation offers high linearity of the output voltage versus the input amplitude, but introduces a higher level of complexity to the circuit because of the mixers.

Diode detectors are inherently nonlinear, what can be undesirable. Especially in systems with a strong variation in transmission power as can be found in digitally modulated systems, the nonlinearity of the diode power detectors can be a problem. Often, the nonlinearity makes linearization techniques necessary. For example, a logarithm circuit can be used for linearization. Linearization techniques add complexity to the circuit or the control software, so that synchronous demodulation or techniques derived from it become interesting.

This work will explore the use of injection locking to overcome some of the phase accuracy and complexity limitations of quadrature amplitude demodulation.

2.6 Injection Locking

The injection locking phenomenon has already been described for mechanical pendulum clocks by Christiaan Huygens as early as in the 17th century, but it is also the focus of contemporary research. It can be applied to a wide range of physical systems such as electronics and laser optics, whenever there are oscillators coupled in some way. Injection locking means that an oscillator can lock to the frequency of a signal that is injected into the oscillator from an external source. Once it is locked to the external frequency it does not oscillate on its original frequency anymore (Figure 2.8).



$$V_{o1} = V_{osc,p} \cdot \cos(\omega_0 \cdot t)$$
$$V_{o2} = V_{inj,p} \cdot \cos(\omega_{inj} \cdot t)$$
$$V_{o3} = V_{osc,p} \cdot \cos(\omega_{inj} \cdot t + \theta)$$

Figure 2.8: Injection Locking Principle

However, there will be a phase shift between the injected signal and the locked oscillator output signal, depending on the injected amplitude and the difference between the injected signal and the natural frequency of the oscillator. If the injected signal amplitude is too small or the frequency is too far away from the natural frequency of the oscillator (outside the lock range), no injection locking occurs.

The one sided lock range is defined by equation (2.18) [34]:

$$\omega_0 - \omega_{inj} = \frac{\omega_0}{2Q} \cdot \frac{V_{inj,p}}{V_{osc,p}} \cdot \frac{1}{\sqrt{1 - \frac{V_{inj,p}^2}{V_{osc,p}^2}}}, \qquad (2.18)$$

Where ω_0 is the free-running frequency of the oscillator, $V_{osc,p}$ its peak oscillation amplitude, ω_{inj} is the frequency of the injected signal, $V_{inj,p}$ is the peak amplitude of the injected signal and Q is the quality factor of the oscillator.

It is clear that a lower quality factor means a wider lock range. The complete lock range is two times the one sided lock range, symmetrical around ω_0.

The behavior of an oscillator under injection of an unrelated signal is described by the Adler equation (2.19) [34]:

$$\frac{d\theta}{dt} = \omega_0 - \omega_{inj} - \frac{\omega_0}{2Q} \cdot \frac{V_{inj,p}}{V_{osc,p}} \cdot \sin(\theta), \qquad (2.19)$$

When injection locking occurs, equation (2.20) applies:

$$\frac{d\theta}{dt} = 0, \qquad (2.20)$$

From equations (2.19) and (2.20), the phase between the injected and the output signal can be calculated as shown in equation (2.21):

$$\theta = \sin^{-1}\left[(\omega_0 - \omega_{inj}) \cdot \frac{2Q}{\omega_0} \cdot \frac{V_{osc,p}}{V_{inj,p}}\right]. \qquad (2.21)$$

So, the phase can theoretically be used to measure the injected amplitude, once the difference between the free running frequency of the oscillator and the injected frequency is known. However, the usefulness of this relationship between phase and injected amplitude for measuring the amplitude directly is not better than the use of a much simpler diode power detector, because it is nonlinear.

Nevertheless, the technique proposed in this work is to apply injection locking to generate the 0° and 90° LO-signals from the incident signal for the Cartesian method, this way already removing the amplitude information as required for the IQD.

In the following chapter, the design of a complete Cartesian reflection coefficient measurement system based on this technique will be presented.

3 Reflection Coefficient Measurement System

This chapter shows the proposed UHF reflection coefficient measurement block based on injection locking and quadrature amplitude demodulation in detail. It also explores other aspects of an automatic impedance matching system such as the control block and software, and the variable impedance matching network.

The system frequency was chosen to be in the range of 400 MHz (low UHF). This frequency is sufficiently high to get high frequency effects (it allows for extrapolation into the middle and higher UHF range), but also sufficiently low that the wavelength is big enough (in the 0.7 m range) to make it challenging to integrate wave guide structures. Also, antenna mismatch occurs often in the MICS band (Medical Implant Communication Service, 402 to 405 MHz, limited to 25 µW output power to minimize interference), as the devices that operate in this band are frequently worn close to the body or implanted. There is an ISM band (Industrial, Scientific and Medical, 433.050 to 434.790 MHz) in this frequency range as well. This band allows higher output powers and license free operation, but it is only defined for region 1 (Europe, Africa and Middle East).

3.1 Overview

The measurement subsystem features injection locking to generate the quadrature signals from the incident wave signal to enhance the quality of the quadrature phase shift for measuring the reflection coefficient in Cartesian form.

Drawbacks of the injection locking technique such as the amplitude dependent variable phase shift are removed by using one additional mixer and correcting the error mathematically in the control block.

As shown in Figure 3.1, the measurement subsystem consists of the following blocks: Power Amplifier (PA), Directional Coupler (A.), Power Splitters (B.), Preamplifier (C.), Quadrature Injection Locking Oscillator (D.), Mixers and Filters (E.), Low Frequency Amplifiers (F.) and the Microcontroller including Analog-to-Digital-Converters (ADC) and System Processor / Control Block (G.).

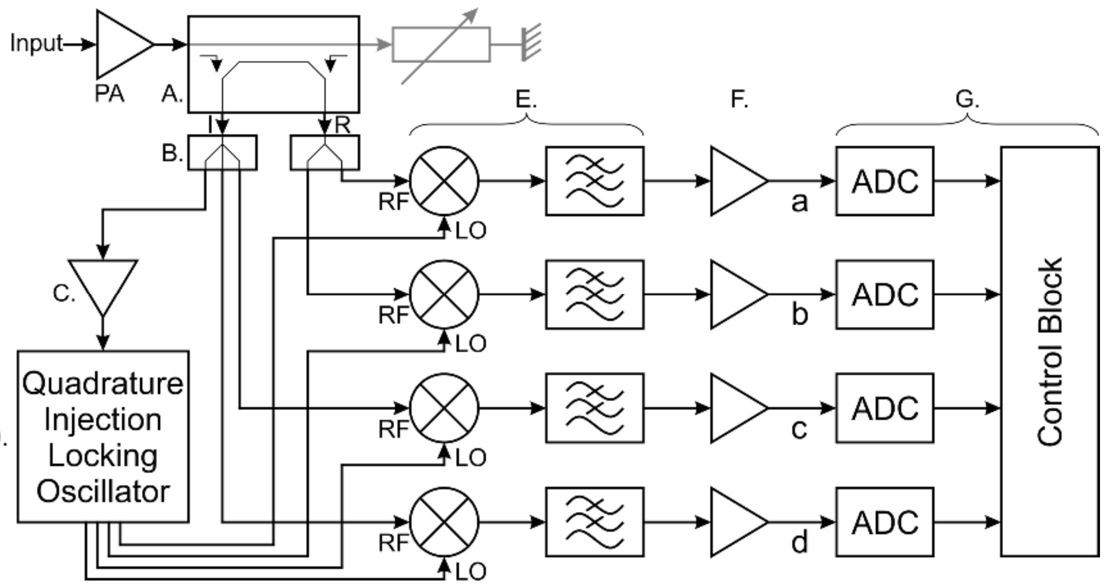


Figure 3.1: Block Diagram of the Designed Solution

The reflection coefficient *Γ* is calculated from the four analog values *a*, *b*, *c* and *d* (Figure 3.1) by equation (2.17) from section 2.5 that is repeated here for convenience:

$$\Gamma = \frac{ac+bd}{c^2+d^2} + \mathrm{j}\frac{bc-ad}{c^2+d^2} \tag{2.17}$$

3.2 MOSFET Power Amplifier

The position of the power amplifier block in the measurement system block diagram is shown in Figure 3.2:

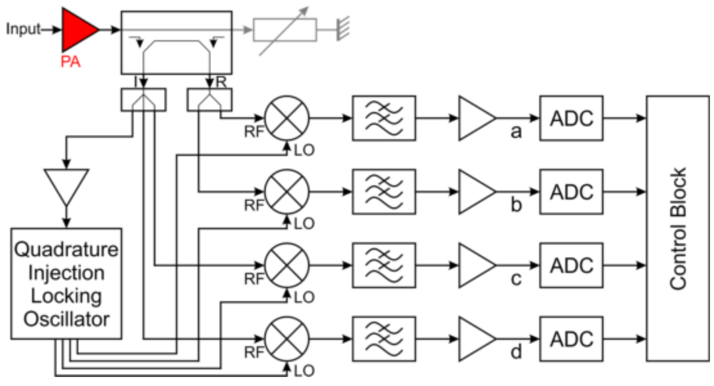


Figure 3.2: Position of Power Amplifier in System

The power amplifier is the block that is being protected against load mismatch by the automatic impedance matching system. It amplifies the RF power from an external signal generator and feeds it into the system (the main input of the directional coupler). The schematic of the used amplifier is shown in Figure 3.3.

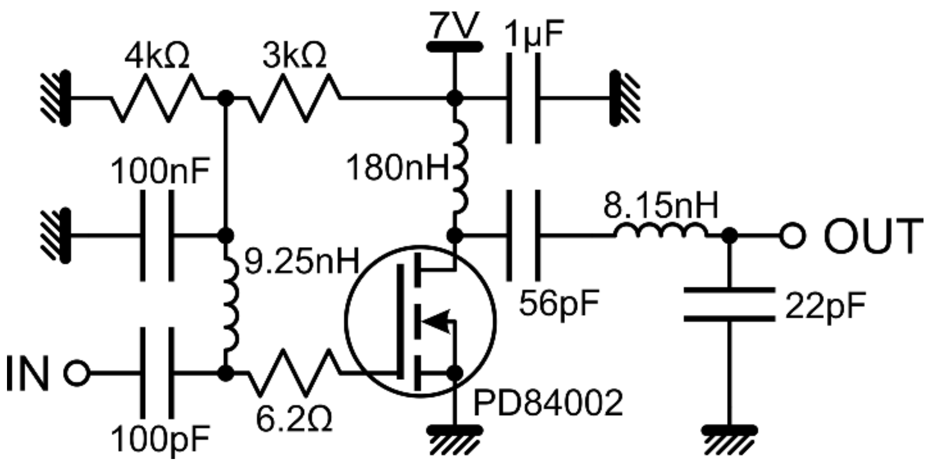


Figure 3.3: Designed Power Amplifier

The power amplifier was designed for an output power of 1 W (30 dBm). The used transistor model is the PD84002 LDMOS transistor [35] from ST Microelectronics that supports up to 2 W output power. Initially operating point simulations were used to determine the bias conditions. Then the parameters of the amplifier such as impedances, output power and 1dB compression point, were optimized in several iteration steps using computer simulations. The impedance matching networks at output and input were manually adjusted in several iterations for maximum output power, using a constant 50 Ω load, because the load pull simulation did not converge (the transistor model did not work with the ADS load pull simulation).

The 1dB compression point found using computer simulations was at 30.2 dBm output power with a gain of 21.2dB and the small signal gain was about 22dB.

The input impedance was optimized to a value close to 50 Ω by simulating the S_{11} of the amplifier and adjusting it to a value close to 0. In the final configuration, a magnitude of S_{11} of 0.003 (about -50dB) was found. The output impedance was not optimized to 50 Ω, but for maximum output power into 50 Ω by adjusting the output impedance matching network in various iterations. In every iteration step, the stability of the circuit was checked using the Rollett stability factor K and the Δ value. The values for the final configuration were K = 1.57 and Δ = 0.49, ensuring that the amplifier is unconditionally stable.

The simulated power delivered to the load (P_L) of the amplifier versus the frequency and the power available from the source P_{AVS} is shown in Figure 3.4.

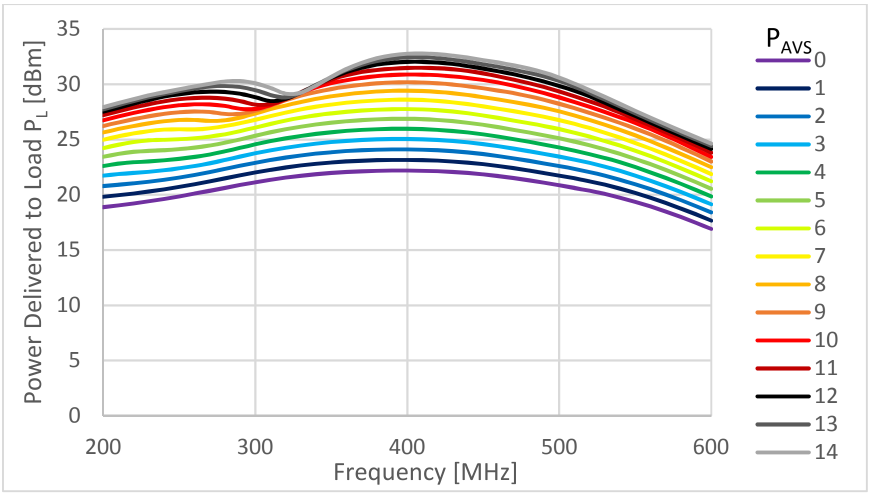


Figure 3.4: Power Delivered to Load P_L vs. Frequency and P_{AVS} (Simulation)

For the condition that source and load reflection coefficients are zero (matched to 50 Ω), but the input reflection coefficient S_{11} and the output reflection coefficient S_{22} of the amplifier are not zero, equations (3.1) to (3.3) are valid. S_{21} is the forward transmission coefficient of the PA. The transducer power gain G_T is shown in equation (3.1), the power gain G_P is calculated from the S-Parameters of the PA as shown in equation (3.3).

$$G_T = \frac{P_L}{P_{AVS}} = \frac{Power\ Delivered\ to\ Load}{Power\ Available\ from\ Source} = |S_{21}|^2 \tag{3.1}$$

$$\frac{P_{AVS}}{P_{IN}} = \frac{Power\ Available\ from\ Source}{Power\ Input\ to\ PA} = \frac{1}{1-|S_{11}|^2} \tag{3.2}$$

$$G_P = \frac{P_L}{P_{IN}} = \frac{Power\ Delivered\ to\ Load}{Power\ Input\ to\ PA} = \frac{|S_{21}|^2}{1-|S_{11}|^2} \tag{3.3}$$

The simulated power gain of the PA versus frequency and P_{AVS} is shown in Figure 3.5.

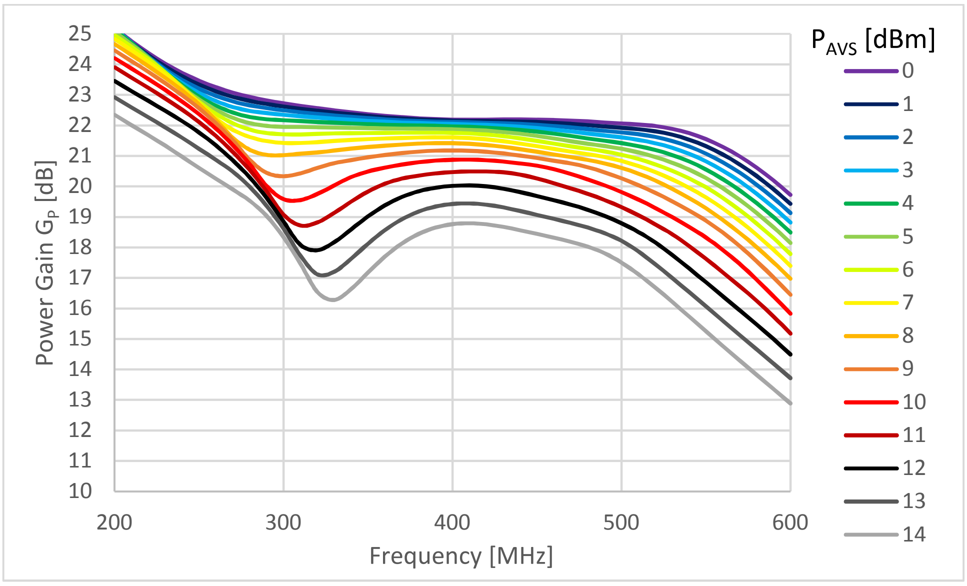


Figure 3.5: Power Gain G_P vs. Frequency and P_{AVS} (Simulation)

In Figure 3.4 is shown that the output power of the PA can be greater than 30 dBm and that it does not change a lot for varying frequency. In Figure 3.5 the gain compression can be observed, the 1dB gain compression point being at about 9 dBm P_{AVS} that results in 30.2 dBm output power.

3.3 Directional Coupler

The position of the directional coupler block in the measurement system block diagram is shown in Figure 3.6:

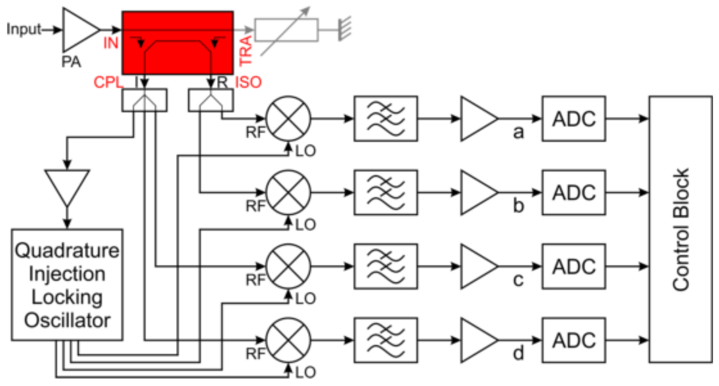


Figure 3.6: Position of Directional Coupler in System

The directional coupler splits a fraction of the total power on its main path (IN to TRA) into incident signal and reflected signal and makes each one available on a separate output (CPL and ISO, respectively). The phase relationship between both is maintained. After the directional coupler, the incident and the reflected measurement signal are fed into the power splitters / attenuators.

A printed circuit board (PCB) setup allows a better access to measuring points inside the circuit than an integrated circuit (IC) setup and it makes fine tuning of the component values possible, so, it was decided to do a PCB setup. For such a setup, there are many directional couplers with various properties available as ready-to-buy components. The Mini-Circuits ADC-15-4 directional coupler [36] was chosen due to its properties (frequency range, attenuation, directivity, package style) and its availability in the LCCI.

Mini-Circuits does not supply an Keysight ADS simulation model for the coupler. Additionally, some simulations such as the transient analysis do not accept a S-Matrix as input. So a model based on the two transformer topology [31] and four resistors to simulate losses was built up. The winding ratio and the values of the resistors were adjusted to match the S-Parameters given in the datasheet [36] using computer simulations.

The resulting schematic is shown in Figure 3.7.

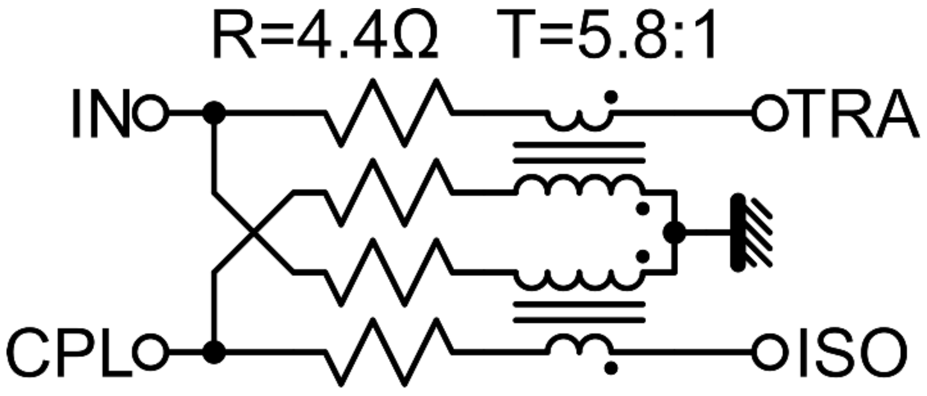


Figure 3.7: ADC-15-4 Simulation Model

After adjusting the resistors and the winding ratio, the simulation model showed an insertion loss of 0.57dB. The datasheet gives between 0.56dB and 0.61dB for the real device. The simulated directivity was 24.4dB compared to the datasheet value between 24.5dB and 25.6dB. The coupling was 15.3dB, equal to the datasheet value. The input and coupled return loss were both 24.1dB, compared to datasheet values of 24.4dB to 26.4dB. The output return loss was with 28.6dB a bit lower than the datasheet values of 31.5dB to 34.5dB. All values are for a frequency of 400 MHz. The small differences in the simulated values compared to the datasheet values are because the exact inner construction of the ADC-15-4 coupler is unknown, but as it is wide band, it was assumed that the two transformers topology is sufficiently accurate. An ideal coupler would have an insertion loss and all reflection coefficients of zero.

3.4 Power Splitters / Attenuators

The position of the power splitters / attenuators blocks in the measurement system block diagram is shown in Figure 3.8:

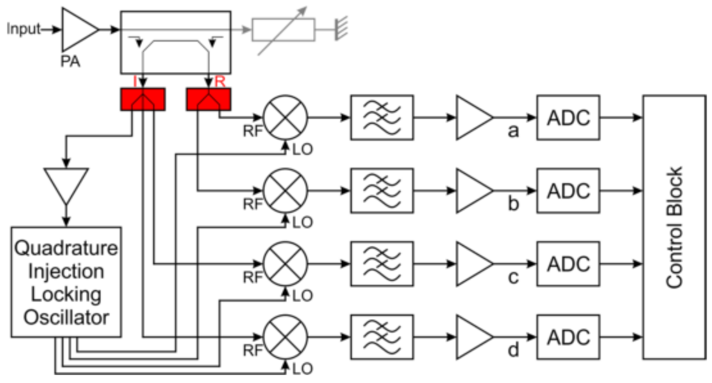


Figure 3.8: Position of Splitters in System

The power splitters are located in the measurement branch of the system, not in the power path. They serve a double purpose: to attenuate the power of the two measurement signals from the coupler to allowable values for the following blocks, and to split and direct the remaining power into five channels (to the input of the preamplifier, as well as to the RF inputs of the four mixers). All inputs and outputs are impedance matched to 50 Ω. As attenuation is necessary anyway, it is practical to use resistive power splitters for this purpose (these also have less influence on the phase of the signals than a Wilkinson divider).

The power splitters / attenuators were calculated using the following equations (3.4) to (3.6) [37]. An attenuation *A* of 6dB means an output amplitude of half the input. An attenuation *A* of 9.54dB results in an output amplitude of one third the input. For one fourth output, two stages of one half were cascaded, for one sixth output, a one half and a one third stage were cascaded. The reference impedance Z_0 equals 50 Ω. The next closest 1% resistor values were chosen.

Y-6dB-Splitter:

$$R_1 = R_2 = R_3 = Z_0/3, \tag{3.4}$$

T-Attenuator:

$$R_{in} = R_{out} = Z_0 \frac{10^{\frac{A}{20}}-1}{10^{\frac{A}{20}}+1}, \tag{3.5}$$

$$R_{foot} = 2 \cdot Z_0 \frac{10^{\frac{A}{20}}}{10^{\frac{A}{10}}-1} \tag{3.6}$$

3.4.1 For Incident Wave

The incident wave splitter directs its input, the incident wave signal V_i, into three channels: The input of the preamplifier for injection locking (having $^1/_6$ of the amplitude of the V_i signal), and the RF inputs of the in-phase and of the quadrature V_i mixers (each having $^1/_4$ of the amplitude of the V_i signal). The values of the resistors are shown in Figure 3.9.

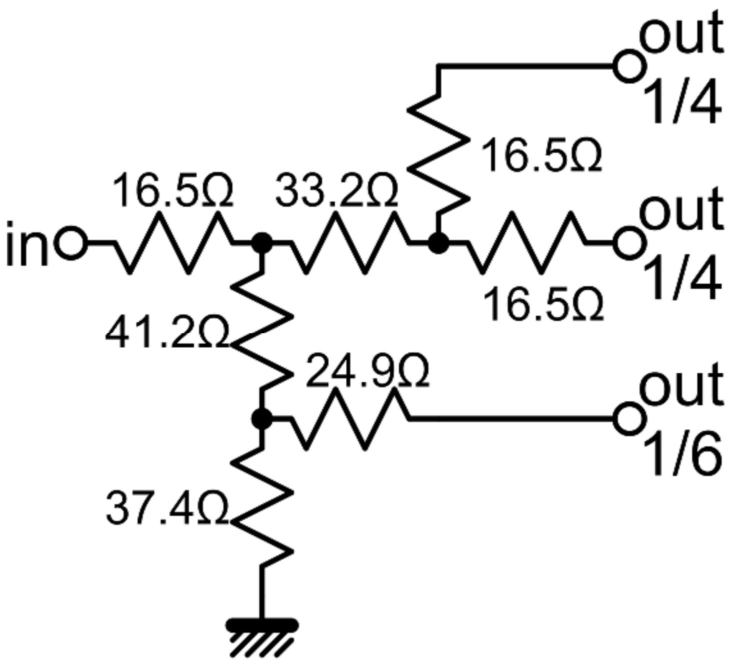


Figure 3.9: Splitter/Attenuator for Incident Wave (in Measuring Path)

3.4.2 For Reflected Wave

The reflected wave splitter directs its input, the reflected wave signal V_r, into the RF inputs of the in-phase and of the quadrature V_r mixers (each having $^1/_4$ of the amplitude of the V_r signal). The values of the resistors are visible in Figure 3.10.

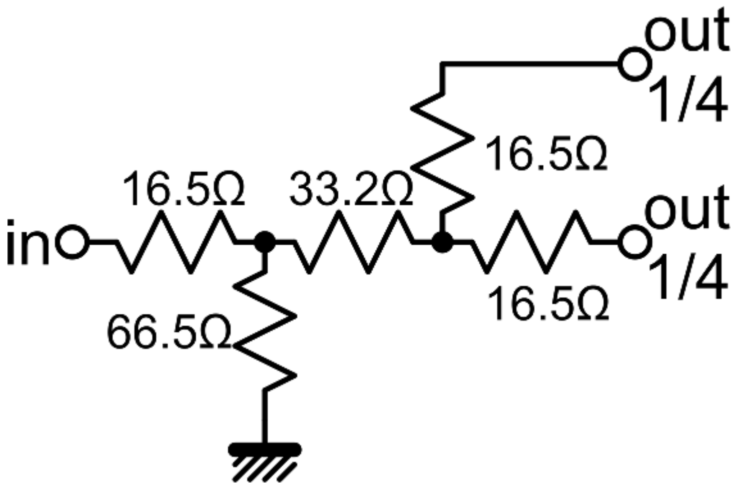


Figure 3.10: Splitter/Attenuator for Reflected Wave (in Measuring Path)

3.5 Preamplifier for Injection Locking

The position of the preamplifier block in the measurement system block diagram is shown in Figure 3.11:

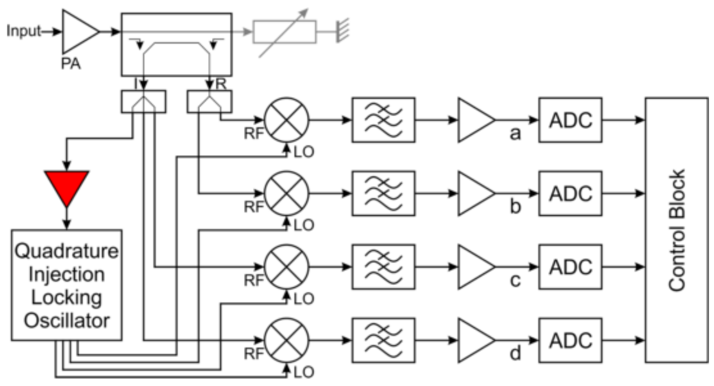


Figure 3.11: Position of Preamplifier in System

The quadrature injection locking oscillator (QILO) was designed as a relatively high power oscillator to minimize the necessary amplification after it, as detailed in section 3.6. The injection signal was fed into the base of the oscillator's transistors that is connected to the collector of the adjacent transistor. As the oscillation amplitude at the injection point is relatively high, the injected signal amplitude also has to be relatively high to have effect. A preamplifier was included, given the direct incident signal after the directional coupler was too weak to achieve locking. To avoid the same design effort that was made for the power amplifier, a ready-made RF amplifier block was used. A balun was added to convert the single ended amplifier output to a differential signal as needed by the oscillator.

As only the voltage amplitude has to be adjusted, this could also be done using a passive network comprising of capacitors and inductors, what would allow a very low power consumption. However, for a laboratory setup, it was decided that the advantage of the RF amplifier block of a low S_{12} was more important, avoiding RF energy flowing backwards and possibly interfering with the measurement of the incident signal V_i. The LO-RF isolation is an important design parameter of the mixers. This means that the back coupling through the mixers is generally low.

3.5.1 RF Amplifier BGA6489

The RF amplifier BGA6489 is a broadband 50 Ω gain block with a maximum output power of 20 dBm. To make use of the maximum dynamic range of the amplifier, its input signal had to be attenuated additionally (this is done in the power splitter). At 8 V supply voltage, only a 39 Ω resistor, a 68 nH inductor and small ceramic capacitors are needed for the amplifier's operation. The schematic is shown in Figure 3.12, more information can be obtained from the datasheet [38]. For the transient simulations, there was also a simulation model constructed for this amplifier, as the model supplied by NXP only includes the S-Parameters at various frequency points. The model also includes clipping for amplitudes greater than 6.75 V. The S-Parameters of the simulation model are compared to the datasheet values in Table 3.1

Table 3.1: S-Parameters of BGA6489 Model at 400 MHz

(Magn./Angle)	S_{11}	S_{12}	S_{21}	S_{22}
Datasheet [38]	0.11 / 21.64°	0.06 / -0.35°	12.31 / 149.28°	0.14 / -46.54°
Simulation Model	0.116 / 7.43°	0	12.28 / 139.46°	0.134 / -100.96°

The schematic for the BGA6489 from the datasheet was adopted.

It is industry standard to combine several capacitors of different values for high frequency blocking. The idea is to reduce the effect of the self-resonant frequency (SRF) of the capacitors. In impedance vs. frequency diagrams of capacitors, the SRF is the point where the impedance is at its minimum. The SRF can be low, for example about 50 MHz for 0805 size 10 nF ceramic X7R surface mounting technology (SMT) capacitors [39] and is generally higher for smaller values of capacitance. So capacitors with smaller values have a lower impedance at higher frequencies than capacitors with higher capacitance. Combining them in parallel can supply a reasonably low impedance in a wider frequency band.

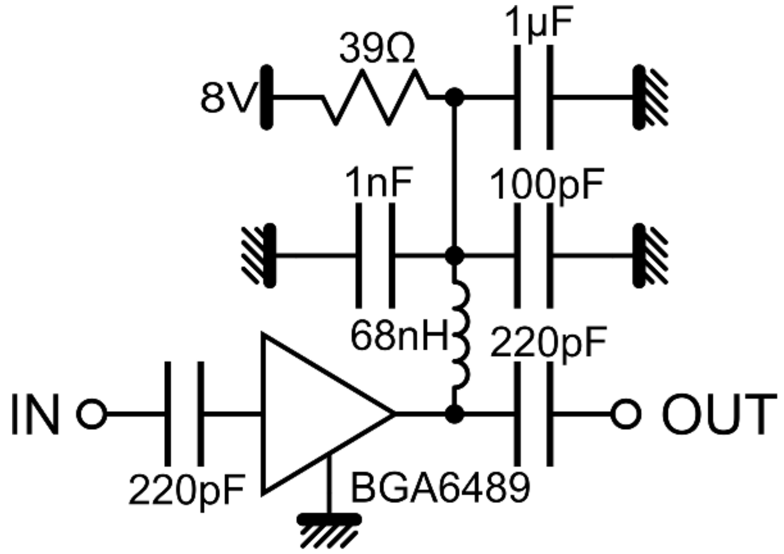


Figure 3.12: BGA6489 and Periphery

3.5.2 Balun and Impedance matching to 75 Ω

For converting the single ended output of the amplifier to a differential signal, a balun was included into the block. As the used balun has 75 Ω impedance and the amplifier 50 Ω, an impedance matching network was included after the amplifier. The matching circuit is shown in Figure 3.13. It was obtained using the Smith-Chart-Utility of Keysight ADS.

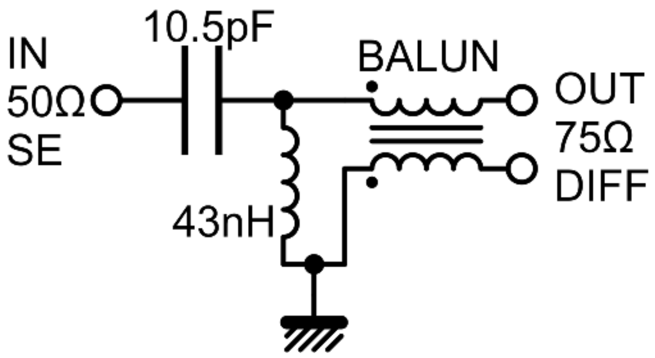


Figure 3.13: Balun and Fixed Impedance Matching Circuit

3.6 Quadrature Injection Locking Oscillator (QILO)

The position of the quadrature injection locking oscillator block in the measurement system block diagram is shown in Figure 3.14:

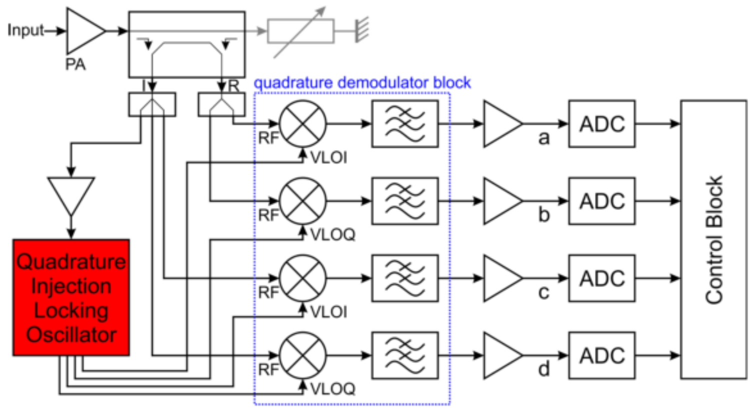

Input
PA
I
R
quadrature demodulator block
RF
VLOI
RF
VLOQ
RF
VLOI
RF
VLOQ
a
b
c
d
ADC
ADC
ADC
ADC
Control Block
Quadrature Injection Locking Oscillator

Figure 3.14: Position of QILO in System

The QILO produces 0° and 90° phase shifted signals from the incident wave signal V_i. The amplitude of the output signals is independent of the input signal amplitude.

The absolute output phase θ_a depends on the input amplitude for any oscillator that is injection locked to an external signal. The relative phase difference between the outputs stays constant. Thus, the oscillator will produce signals with a phase of θ_a+0° and θ_a+90°. As θ_a is generally unknown, additional effort is necessary in the quadrature demodulator block of the system to remove this phase uncertainty error term and recover the reflection coefficient information. This effort can consist of including an additional mixer for the quadrature component of the incident signal. This means there is an in-phase and a quadrature signal for both the incident and the reflected signal. Then the variable phase shift can be removed by applying equation (2.17).

As the output signals of the QILO will be used as local oscillator (LO) signals in the quadrature demodulator block, it was decided to use relatively high oscillation amplitudes in order to limit the necessary amplification of the QILO output signals to a minimum, reducing possible influence on the signal phases.

Nevertheless, to reduce loading of the oscillator core and for impedance matching, output buffers were included in the oscillator design.

Also, to reduce loading the oscillator with low impedance input signals, input buffers with a high impedance output were included. To remove an additional 180° phase uncertainty error term, a single frequency injection locking scheme (shown in Figure 3.15) was used instead of the more common double frequency input at the frequency defining capacitor [40] or at the foot current sources (like it is normally done for injection locked frequency dividers). For a differential oscillator, this makes the application of two input buffers necessary.

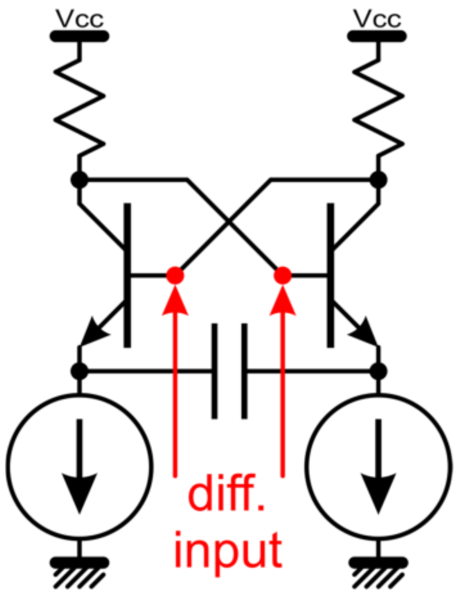


Figure 3.15: Single Frequency Inj. Locking

3.6.1 Input Buffers

The input buffers apply a cascode circuit to achieve a high output impedance. For the same reason, a parallel resonant circuit is used at the output collector of the cascode circuit for biasing. The quality factor of the resonant circuit is defined and made lower than the pure LC quality factor by an additional series resistor to make the circuit reasonably wide band.

In an integrated setup, additional transistors could be used for biasing instead of the resonant circuit. The input is matched to the 37.5 Ω single / 75 Ω differential output impedance of the preceding balun. The circuit including all biasing elements is shown in Figure 3.16. The two input buffers feed their output signal to the bases of the transistors in the oscillator core.

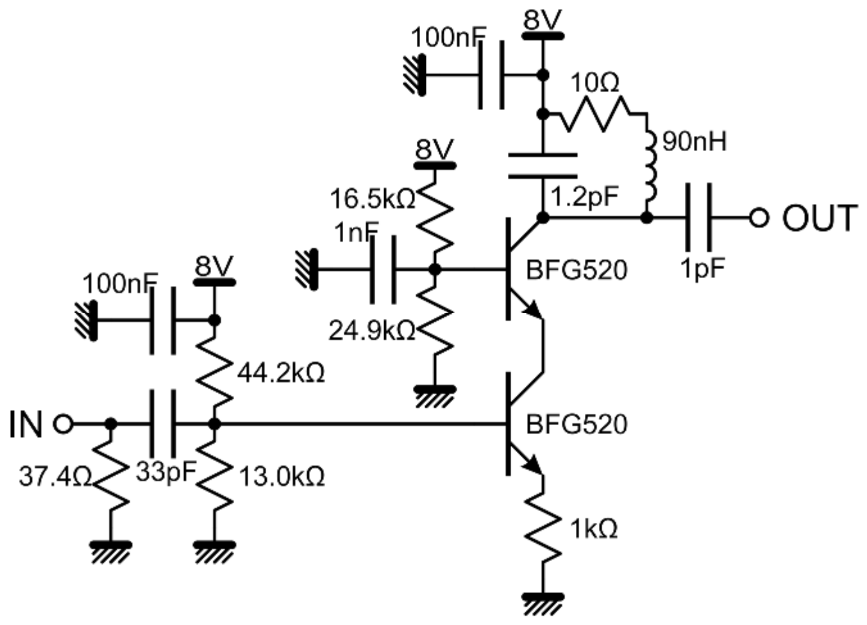


Figure 3.16: One of Two Cascode Input Buffers of the QILO

The frequency response of the input buffers' input and output impedance is shown in Figure 3.17 and Figure 3.18, respectively. As desired, the resonant network at the output generates a high impedance close to 400 MHz. The output spectrum of one input buffer for 1V input amplitude is shown in Figure 3.19.

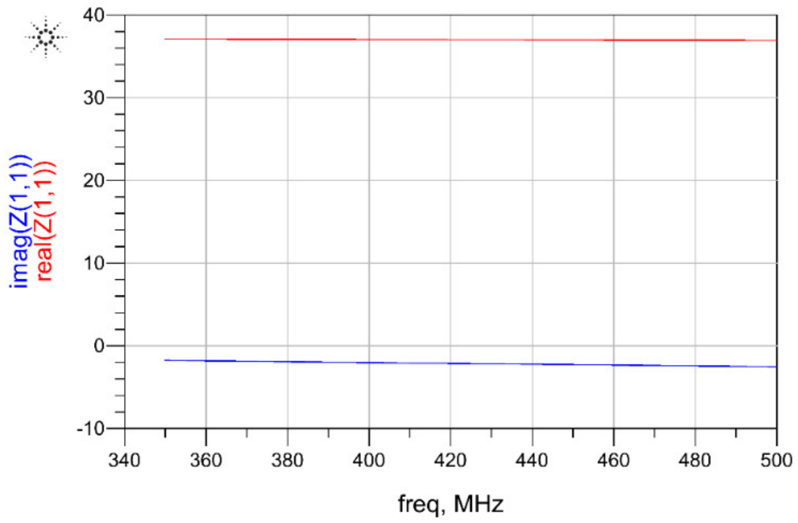


Figure 3.17: Input Impedance of one of the Input Buffers

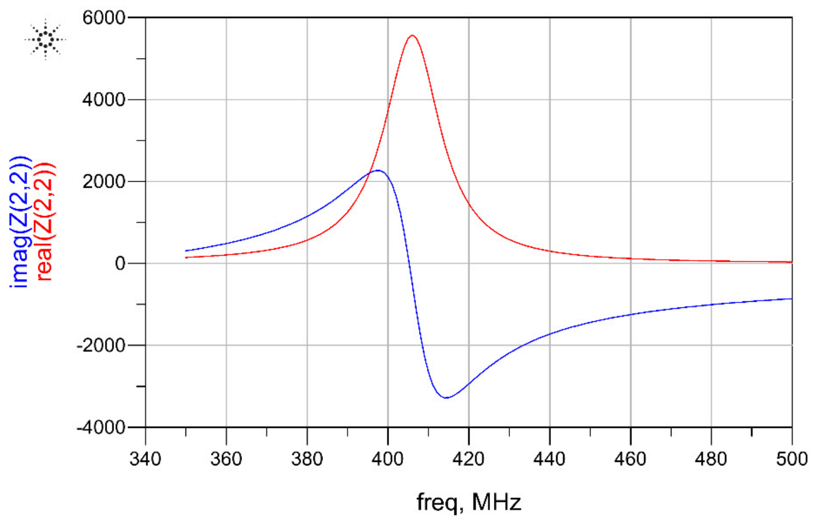


Figure 3.18: Output Impedance of one of the Input Buffers

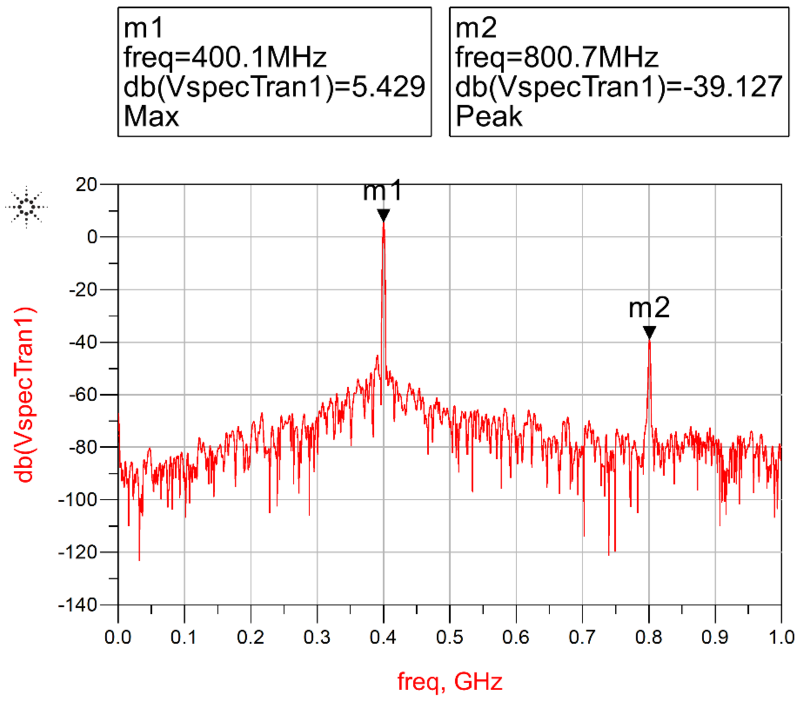


Figure 3.19: Output Spectrum of one of the Input Buffers for 400MHz 1Vp Input

3.6.2 Oscillator Core

A differential RC oscillator topology was chosen instead of an LC topology because RC oscillators generally have a lower quality factor what translates into a wider lock range, see equation (2.18). The basic used topology is shown in Figure 3.20.

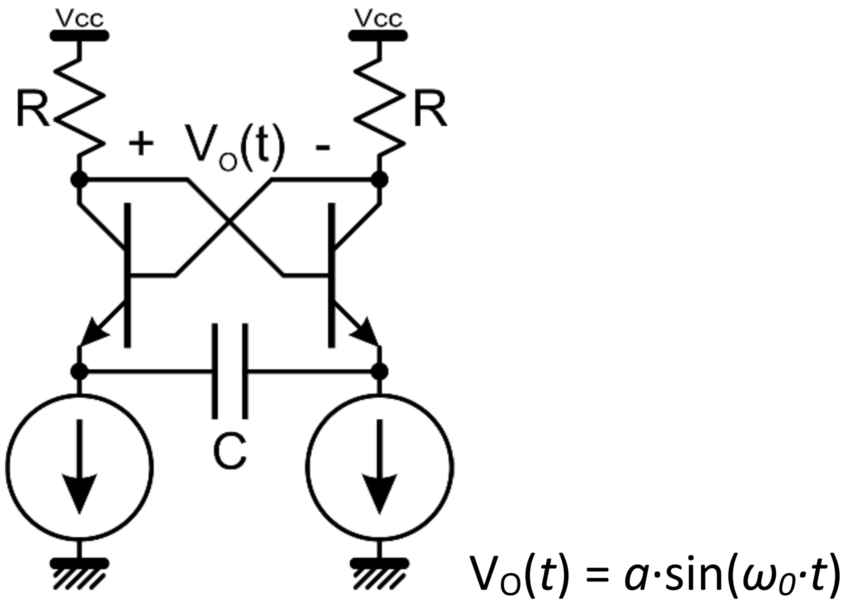


Figure 3.20: Differential RC-Oscillator

In an integrated circuit setup, current sources are used for biasing the circuit. As this is difficult in a PCB setup, a combination of resistors and DC-feeds (inductors) was used. In the chosen differential RC topology, the frequency is mainly defined by two resistors R and one capacitor C.

The open loop small signal transfer function of the oscillator core can be calculated using a simple linear transistor model with the base resistance r_B, the current gain g_c and the base-collector-capacitance C_{BC}, shown in equation (3.7) and in Figure 3.21.

$$H = \frac{i_{out}}{i_{in}} =$$

$$\frac{(2 \cdot R \cdot C_{BC} \cdot (g_c + 1) \cdot s + g_c) \cdot C}{-16 \cdot R \cdot r_B \cdot C \cdot C_{BC}^2 \cdot s^2 + 2 \cdot C_{BC} \cdot (R \cdot C \cdot (3 \cdot g_c - 1) - 4 \cdot R \cdot C_{BC} \cdot (g_c + 1) - 2 \cdot r_B \cdot C) \cdot s + C \cdot g_c - 2 \cdot C_{BC} \cdot (g_c + 1)} \tag{3.7}$$

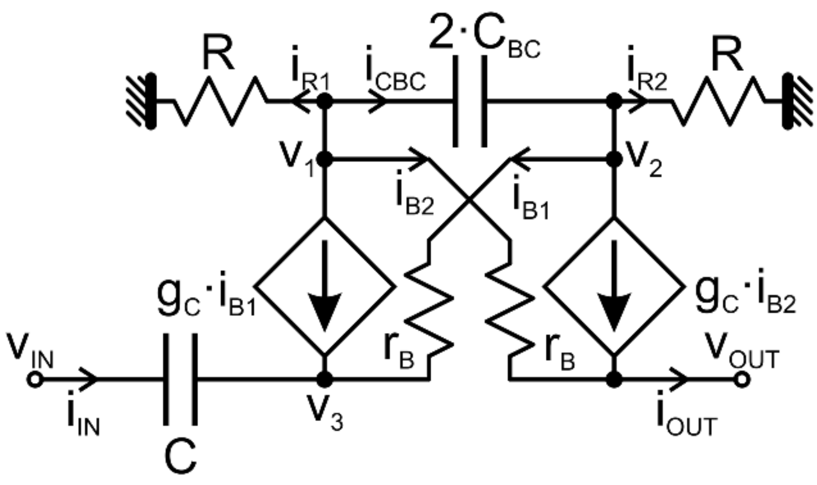


Figure 3.21: Oscillator Model for Small-Signal Open-Loop Transfer Function

Using the Barkhausen stability criterion (Re{*H*}=1 and Im{*H*}=0), the free running oscillation frequency ω_0 can be found, as shown in equation (3.8).

$$\omega_0 = \sqrt{\frac{2\cdot R + r_B}{8\cdot R^2\cdot r_B\cdot C_{BC}\cdot(C - 2\cdot C_{BC})}} \tag{3.8}$$

For the BFG520 transistor, g_C is typically 120 (but can vary between 60 and 250), C_{BC} is 0.3 pF, and r_B can be calculated by equations (3.9) [41]

$$r_B = \frac{g_C\cdot V_T}{I_C} \tag{3.9}$$

Where the thermal voltage V_T is about 26 mV at 25 to 35 °C, and the collector current I_C *is* 20 mA in this application. This leads to equation (3.10).

$$r_B = 156\ \Omega \tag{3.10}$$

The design values are shown in equations (3.11) and (3.12).

$$\omega_0 = 2\cdot\pi\cdot 400\cdot 10^6\ \text{Hz} \tag{3.11}$$

$$R = 200\ \Omega \tag{3.12}$$

With these values, the calculated value for C is shown in equation (3.13).

$$C \approx 6.5 \text{ pF} \tag{3.13}$$

However, the frequency changes considerably when using the topology for quadrature operation. Unluckily, it was not possible to analyze the effect of variation of g_c in the computer simulations.

3.6.3 Quadrature Operation

The simplest way to generate quadrature output signals is by coupling two differential oscillator cores together via coupling elements using one straight pair of connections and one crossed pair as shown in Figure 3.22. In the classical approach, transistors are used as coupling elements. Instead, it was decided to use the capacitor coupling scheme from [42], because it is easier to implement in a discrete setup, and because it promises lower phase noise. However, this raises the free running frequency of the quadrature oscillator with higher used capacitance Cx, as described in [42].

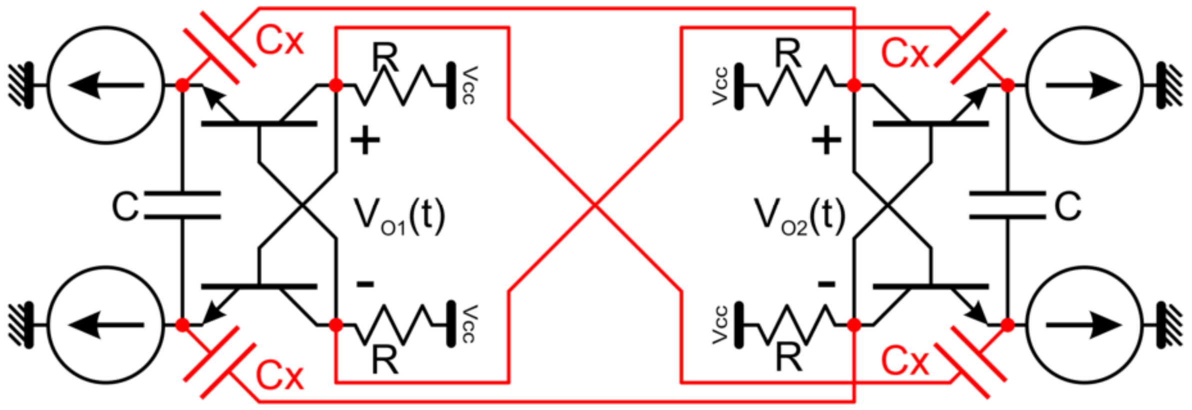


Figure 3.22: Quadrature Connections in Coupled Oscillator

The output signals ideally have the form of equations (3.14) and (3.15):

$$V_{O1}(t) = a \cdot \sin(\omega_0 \cdot t) \tag{3.14}$$

$$V_{O2}(t) = a \cdot \cos(\omega_0 \cdot t) \tag{3.15}$$

3.6.4 Oscillator Including Bias Networks

The designed oscillator, including bias networks, is depicted in Figure 3.23. The output signals of the circuit can be obtained at the collectors of the transistors and are connected to the output buffers via coupling capacitors.

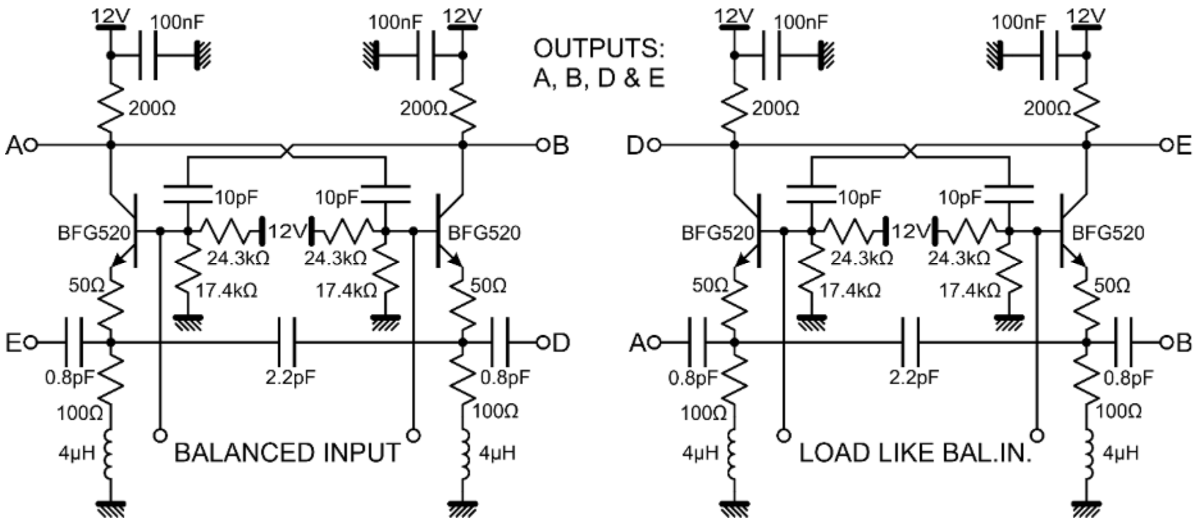


Figure 3.23: QILO without Buffers

3.6.5 Output Buffers

The output buffers should have a high input impedance and a low output impedance. The conventional choice for obtaining this behavior is a common collector circuit. As one stage was not enough to combine a high input impedance with a reasonably high output power, a two stage setup was applied. The complete two stage setup is shown in Figure 3.24, having an output impedance of 300 Ω. This impedance then had to be matched to 50 Ω.

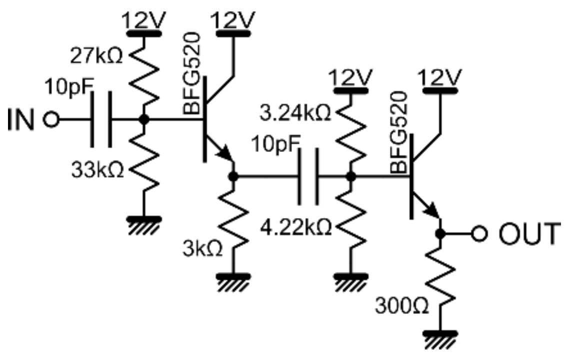


Figure 3.24: Two Stage Common Collector Output Buffer

3.6.6 Impedance Matching to 50 Ω

LC impedance matching circuits were applied to match the 300 Ω output impedance of the output buffers to 50 Ω. A topology that blocks DC and is relatively wide-band was chosen. The schematic is shown in Figure 3.25 and was obtained using the Smith-Chart-Utility of ADS.

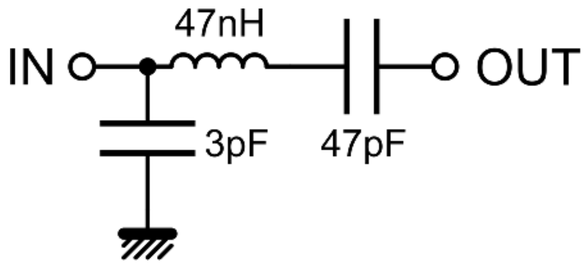


Figure 3.25: Output Buffer Impedance Matching

3.6.7 Simulation Results of Output Buffers

The time-domain voltage signals along the signal path in the output buffers are shown in Figure 3.26. It is visible that the output buffers shape a trapezoid input signal (the raw oscillator core signal contains harmonics, a trapezoid signal is an approximation) to a more sinusoid output signal while also doing the impedance matching. The real input signal will contain less harmonic content than is present in a trapezoid signal and thus will allow even better output signals to be generated. Vbuf1o is the voltage at the output of the first stage, Vout the voltage at the output of the second stage and Vomat after the impedance matching.

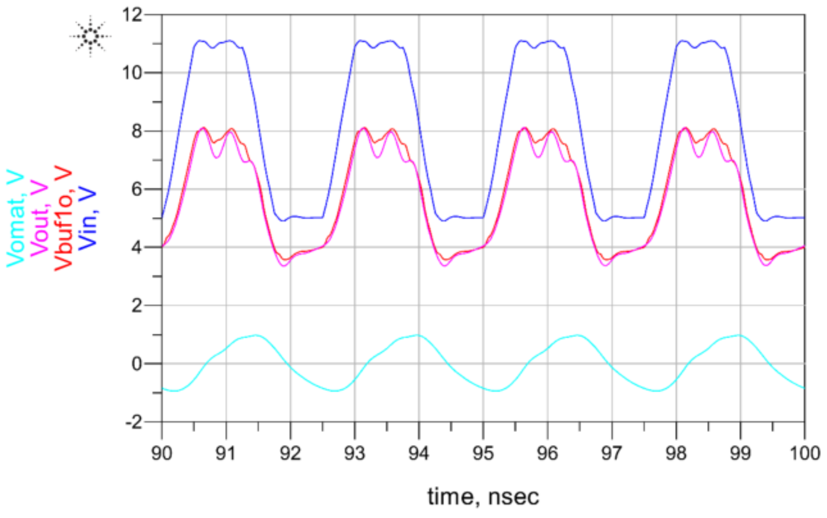


Figure 3.26: Signals before, in and after Output Buffer for Trapezoid Input Signals

3.6.8 Simulated QILO Output Signals after Buffers

In Figure 3.27 and Figure 3.28 the output signals of the whole oscillator with no input at the input buffers are shown. It is visible that they are almost sinusoidal. The exact shape of the signals is not so important as they are only used as local oscillator signals in the mixers, but it is important that the harmonic content does not have an amplitude greater than about 0.3 V, so that the mixers do not produce additional frequencies at their outputs. This is the case. However, the output power must be between 4 dBm (1.0 Vpp at 50 Ω) and 10 dBm (2.0 Vpp at 50 Ω). In fact, all amplitudes are 1.9 Vpp, with maximum error 0.01 V, and all phases are close to multiples of 90°, with maximum error 0.6°, as shown in Table 3.2.

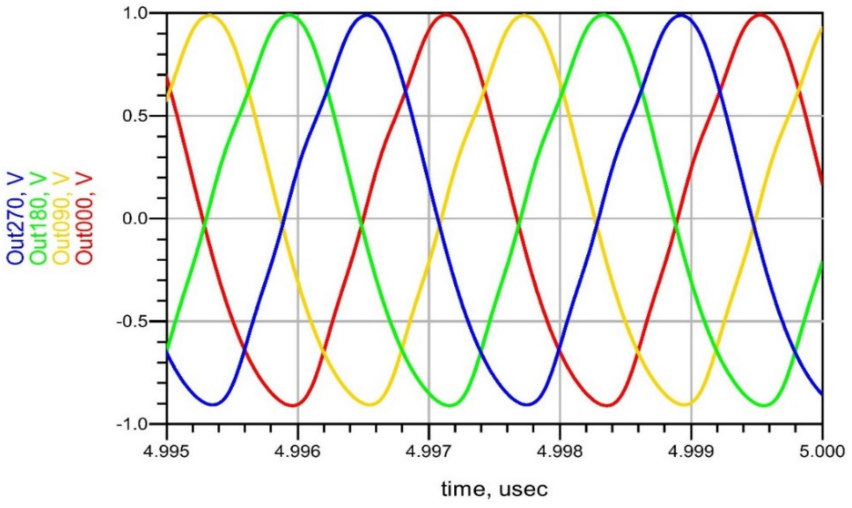


Figure 3.27: Output Signals of QILO

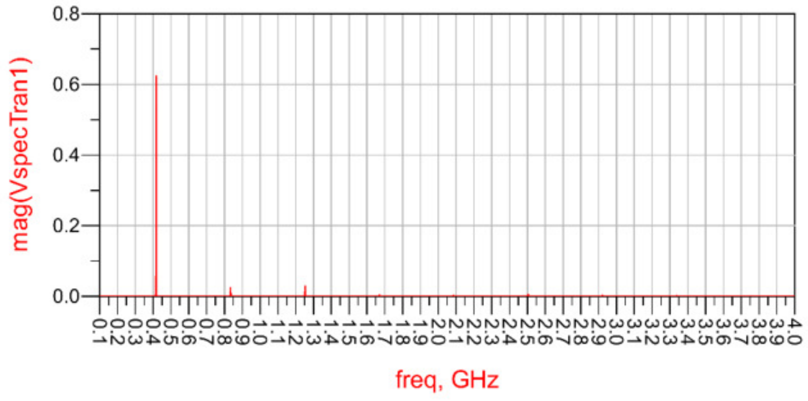


Figure 3.28: Spectrum of Output Signal Out000 (V_{RMS})

Table 3.2: Output Signals of QILO, Amplitude and Phase

Signal	Amplitude P-P (V)	Phase (°)
Out000°	1.90	Reference (0)
Out090°	1.89	89.4
Out180°	1.90	179.9
Out270°	1.89	269.5

3.6.9 Simulations of Oscillator Behavior under Injection Locking

The behavior of the oscillator under injection locking was investigated. For 1, 2, 3, 4 and 5 MHz difference between the injected signal and the free-running frequency $\omega_0=2\cdot\pi\cdot f_0$ of the oscillator, the minimum amplitude for injection locking was found, shown in Figure 3.29. Also, for 1 MHz difference frequency, the input to output phase and output amplitude for output signal Out000° and various input amplitudes was evaluated, shown in Figure 3.30. All data of the computer simulations under injection locking are shown in Table 3.3. As the minimum amplitude for injection locking at 1 MHz difference was 0.4 Vpp (differential) and the maximum input amplitude of the input buffers is about 3.9 Vpp (differential), an input amplitude dynamic range of more than 19dB was achieved. For a change of 13dB of the input amplitude, the output amplitude only changed 0.5%.

The oscillator for these simulations had the following configuration:

C = 1.8 pF, C_x = 0.4 pF, R = 200 Ω. It displayed an f_0 of 406.1 MHz.

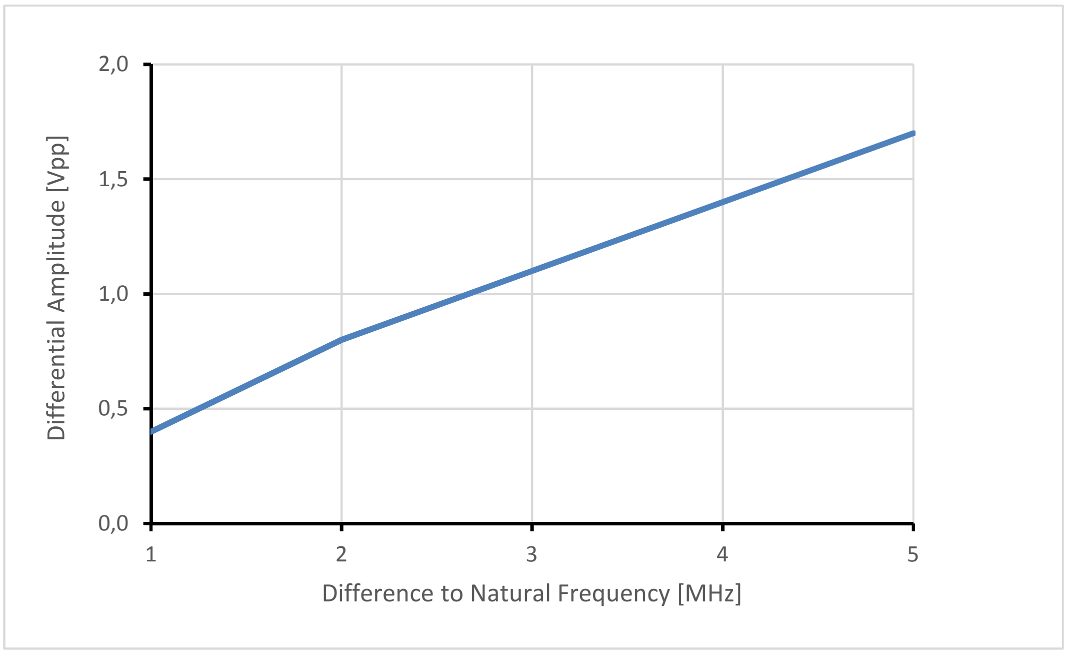


Figure 3.29: Minimum Input Amplitude for Injection Locking

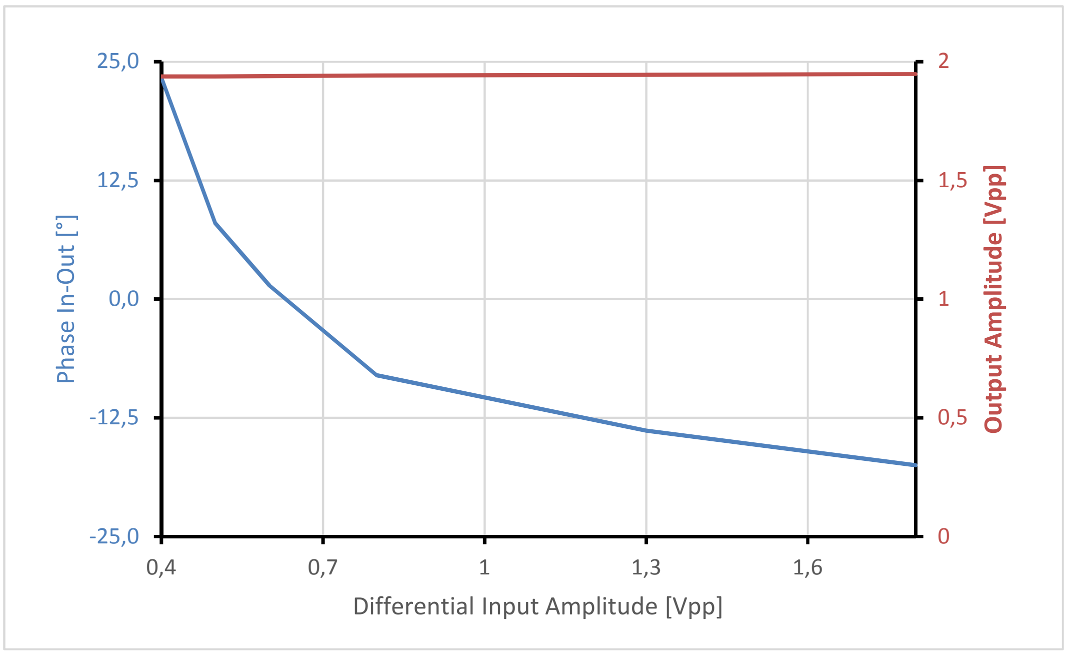


Figure 3.30: Phase and Output Amplitude during Inj. Locking for 1 MHz Difference to f_0

Table 3.3: Behavior under Injection Locking

Inj. Diff. Amplit. (V)	**Inj. Freq. (MHz)**	**Out. Freq. (MHz)**	**Phase (°) / *Comment***	**Output Amplitude Vout (V)**	**Change Vout %**
1.7	401	401	39.0		
1.6	401	404	*No Lock*		
1.4	402	402	36.2		
1.3	402	402.4	*No Lock*		
1.3	403	403	20.3		
1.2	403	403	24.7		
1.1	403	403	33.4		
1.0	403	403.6	*No Lock*		
1.1	404	404	7.3		
1.0	404	404	10.9		
0.9	404	404	18.9		
0.8	404	404	24.7		
0.7	404	404.5	*No Lock*		
1.8	405	405	-17.5	0.974	0.1
1.3	405	405	-13.9	0.973	0.0
0.8	405	405	-8.0	0.971	-0.2
0.6	405	405	1.5	0.970	-0.3
0.5	405	405	8.0	0.969	-0.4
0.4	405	405	23.3	0.969	-0.4
0.3	405	405.3	*No Lock*		

3.7 Quadrature Demodulator: Mixers and Filters

The position of the mixer and filter blocks in the measurement system block diagram is shown in Figure 3.31:

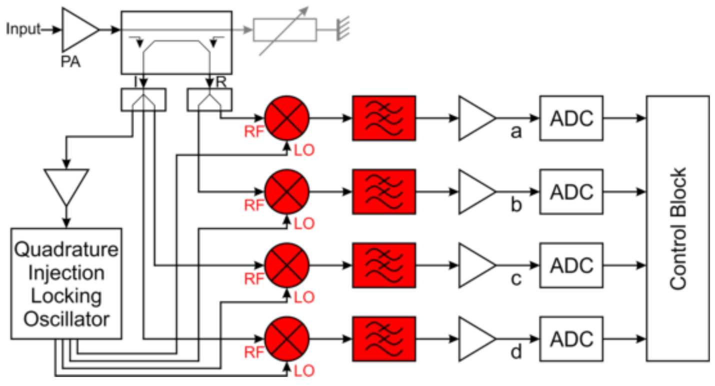


Figure 3.31: Position of Mixers and Filters in System

The purpose of the mixers is to do a frequency conversion of the input signals down to DC. Basically, an active mixer performs a multiplication of its two input signals (RF input and Local Oscillator (LO) input). A passive mixer chops the RF signal in the frequency of the LO signal. Any of both operations leads to the production of new frequencies in the output signal [43]. When applying two frequencies f_1 and f_2 to the mixer inputs, the mixer produces a mix of mainly two frequencies at its output: the sum $f_2 + f_1$ and the difference $f_2 - f_1$ of the input frequencies, shown in Figure 3.32.

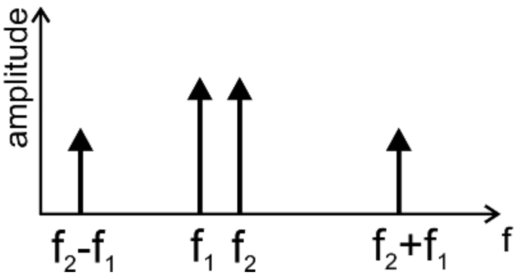


Figure 3.32: Ideal Mixer Operation for Two Input Frequencies

If both frequencies are the same, the difference frequency is zero (the desired DC) and the sum is simply the double frequency. Here the purpose of the low-pass filters becomes obvious: they remove the high frequency content of the output signal and (in the ideal case) just pass the DC content on. This is symbolically depicted in Figure 3.33.

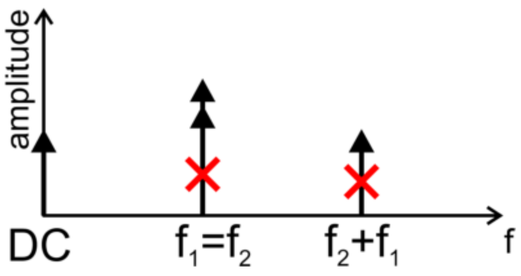


Figure 3.33: Ideal Mixer (black) and Filter (red) Operation for Same Frequency Inputs

It is possible to consider the operation of one mixer with both inputs at the same frequency and one filter together as the operation of a phase and amplitude sensitive synchronous rectifier. If both inputs of the mixer are in-phase, the DC output of the mixer-filter-block is at its maximum, the amplitude of the RF signal times a gain or loss factor (active mixers may have a gain, passive mixers usually have losses). If the two inputs of the mixer are in quadrature, the output of the mixer-filter-block is ideally zero (there can be some offset).

If there are two mixer-filter-blocks, that are both fed with the same RF signal and two different LO signals with the same frequency and a known (ideally 90° / quadrature) phase relationship (they cannot be in-phase), both the relative phase and amplitude of the RF signal can be reconstructed. This is known as quadrature amplitude demodulation and widely used in digital communication frontends.

Here, the incident signal V_i and the reflected signal V_r are both processed using quadrature amplitude demodulation. As the absolute phase of the LO-signals is unknown due to the amplitude dependent variable phase shift introduced by the QILO, but the relative phase between the four LO signals is fixed and well defined, it is necessary to obtain the phase of V_r as well as V_i. This way, in the following calculation of the reflection coefficient (basically the complex division of V_r by V_i), the variable phase shift θ is removed. This becomes clearer in the polar form of the division of the two complex values shown in equation (3.16). It is equal to the division of their magnitudes and the difference of their phases $\varphi_r + \theta$ and $\varphi_i + \theta$.

$$\Gamma = \frac{|V_r|}{|V_i|} e^{j \cdot ((\varphi_r + \theta) - (\varphi_i + \theta))} \tag{3.16}$$

As the variable phase shift is the same for both complex values at any moment in time, given the division is only done for values that were all sampled at the same time, the difference operation removes this part.

3.7.1 Model of Mixer Mini-Circuits ASK-1-KK81

The mixer Mini-Circuits ASK-1-KK81 [44] was chosen because it has a reasonable performance at 400 MHz. It is a double balanced (passive) diode ring mixer. For an integrated setup, both active or passive mixers could be used. Especially the setup in [45] can be interesting for the application.

As most of the computer simulations for the final system were transient simulations, a simulation model for this use had to be developed (Mini Circuits supplies performance data, but no model). According to the diagram in the datasheet and the performance data, the schematic in Figure 3.34 was used with an RF diode model available in Keysight ADS (MBD101) [46]. The impedance matching at the RF input port was necessary to obtain a performance close to the given data.

The model of the mixer has an LO VSWR of 2.62 (datasheet 3.2, 7 dBm LO power) and an RF VSWR of 1.23 (datasheet 1.2) at 400 MHz and -2.3 dBm RF power (that translates to 25 dBm RF power at the coupler input). Between -0.3 dBm and -2.3 dBm the VSWR is better than 1.25.

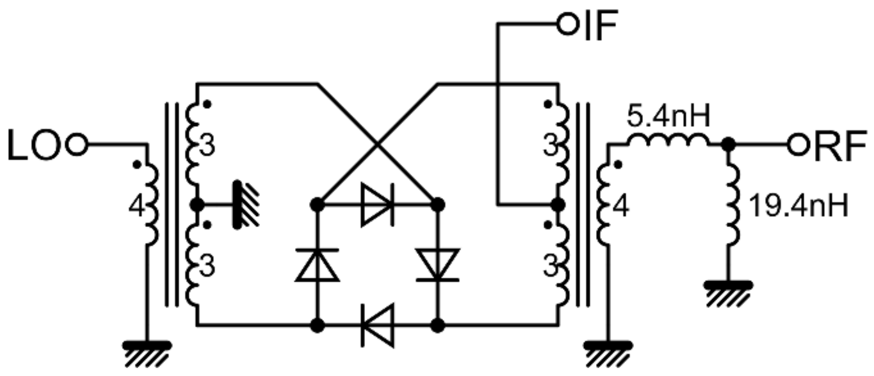


Figure 3.34: Model of Mixer ASK-1-KK81

3.7.2 Filters

Only the signal voltage is of interest at this point, so simple RC filters were used because they are less tolerance dependent than more complex filters and have a reasonable RF performance without making the use of special RF amplifiers necessary. Also, the design complexity was kept to a minimum, what is also reasonable for a proof-of-concept design.

There exists a trade-off between the time constant / delay time of the filter and the attenuation of the RF. The attenuation A (in dB) of a frequency f_{RF} much higher than the cutoff frequency f_c of the low-pass filter is 20dB per decade, shown in equation (3.17). The time constant τ and the cutoff frequency f_c are linked via equation (3.18) [47]. Finally, equation (3.19) describes how the time constant τ and the component values of the filter (resistor R and capacitor C) are connected.

$$A = 20 \cdot \log_{10}\left(\frac{f_{RF}}{f_c}\right) \tag{3.17}$$

$$f_c = \frac{1}{2\pi \cdot \tau} \tag{3.18}$$

$$\tau = R \cdot C \tag{3.19}$$

The attenuation at 400 MHz was chosen to be about 80dB. This value makes sure that the following 12 bit ADC will not convert the RF. Even if the RF amplitude was as high as 0.6 V after the mixer (that is 3 V at the ADC input because of the operation of the LFAs described in the next section), the filter would attenuate this to less than ½ of the resolution of the ADC (the ADC has 2^{12}-1=4095 steps in a 3 V interval). The result is a cutoff frequency of a factor 10^4 lower than the RF to be attenuated. For f_{RF}=400 MHz, this means f_c = 40 kHz. Then τ becomes about 4 µs. To account for component tolerances (mainly of the capacitor) and to achieve an attenuation of a bit more than 80dB, a τ of 5 µs was chosen (so for settling to 0.1%, the time is about 35 µs and f_c becomes about 32 kHz). For a resistor of 50 kΩ, this results in a capacitor of 100 pF. The resulting filter is shown in Figure 3.35.

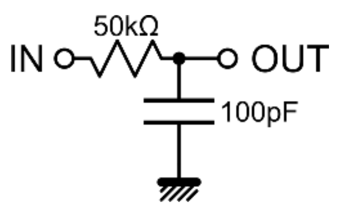


Figure 3.35: Used RC-Low-Pass-Filter

3.8 Low Frequency Amplifiers (LFA)

The position of the low frequency amplifier blocks in the measurement system block diagram is shown in Figure 3.36:

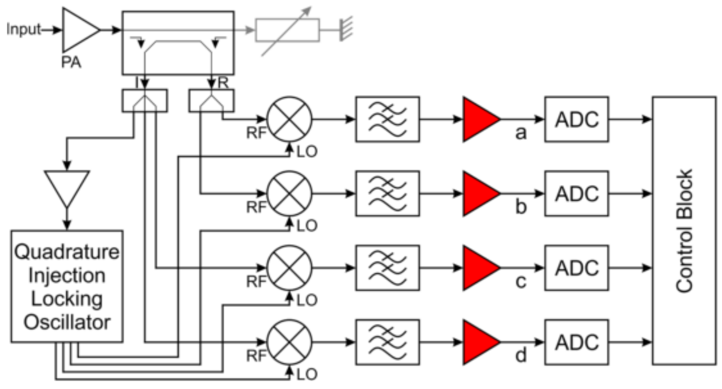


Figure 3.36: Position of Low Frequency Amplifiers in System

The LFAs adjust the signal range as output by the filters to the input range of the ADCs. This way the resolution of the ADC is utilized to a higher extent. Unluckily, they also amplify the offset of the mixers and the noise of the whole system, so the gain cannot be too high. This architecture was chosen, because the mixers do not allow too high RF amplitudes, so some amplification in the end was necessary. The subsystem is to be used with a power amplifier, what means that the signal amplitudes will also not be too low. The LFAs use a symmetric supply of ±1.5 V relative to the RF system ground. However, the ADC ground is connected to the -1.5 V rail, so for the ADC, all signals have an offset of about 1.5 V and the ADC input is limited to 3 V, just as required by the ADC supply voltage. For an integrated circuit however, other techniques for offset adjustment can be realized.

3.8.1 AD8616 Operational Amplifier

The AD8616 is a dual low cost precision operational amplifier (op-amp) from Analog Devices. The chosen small-outline-8-pin (SO-8) package uses the standard pin assignment for two amplifier 8 pin packages, what makes it easy to replace it by another op-amp if needed.

Some of the features of the AD8616 are shown in Table 3.4. Especially the low offset and the low noise are important for the application. More information about this operational amplifier can be found in the datasheet [48].

Table 3.4: Some Features of the AD8616

Rail-to-Rail	Inputs and Outputs
Offset Voltage	65 µV maximum
Input Currents	1 pA
Noise	8 nV/√Hz
Gain-Bandwidth-Product (GBWP)	> 20 MHz
Slew-Rate	12 V/µs
Package	SO-8 (standard pin assignment)

3.8.2 Non-Inverting Amplifier Configuration

The non-inverting configuration was chosen due to its high input impedance in order not to load the RF filter. Additionally, it is easy to reduce the gain for RF to unity by adding another capacitor. So only very few components are needed to complete the ADC input driver as shown in Figure 3.37 for one of four channels.

The 2 kΩ resistor at the input serves to protect the AD8616 from excessive currents at its input when the power supply is switched off (the filter capacitor could discharge over the protection diodes of the AD8616). The gain was chosen to be 5 to make sure there will never be an overdrive condition of the ADC for possible output signals of the mixer. The mixer output signal will always have an absolute value of less than 0.3V. So the Signal seen by the ADC will always be in the interval of 0 V to 3 V (1.5 V ± 1.5 V). In fact, the signals will normally be less than 0.2 V after the mixer, what translates to 1.0 V limit after the LFA, so the interval will be 0.5 V to 2.5 V.

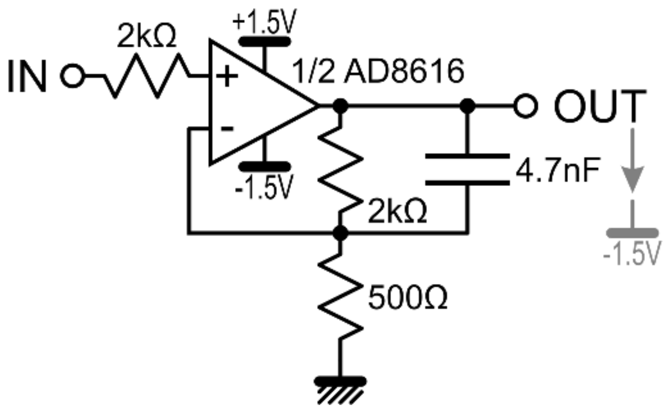


Figure 3.37: Low-Frequency-Amplifier Schematic

3.8.3 Keysight ADS Simulation Model of AD8616

To be able to simulate the whole system including the LFAs, a simplified model of the AD8616 operational amplifier was built using the OpAmpIdeal block from Keysight ADS, some additional capacitors for the inputs and one fixed and two controlled current sources at the power supply pins. The model is shown in Figure 3.38. Together with the gain defining resistors as described before, the model performed as expected with a gain of 5 for DC. Slew rate and offset were not modeled.

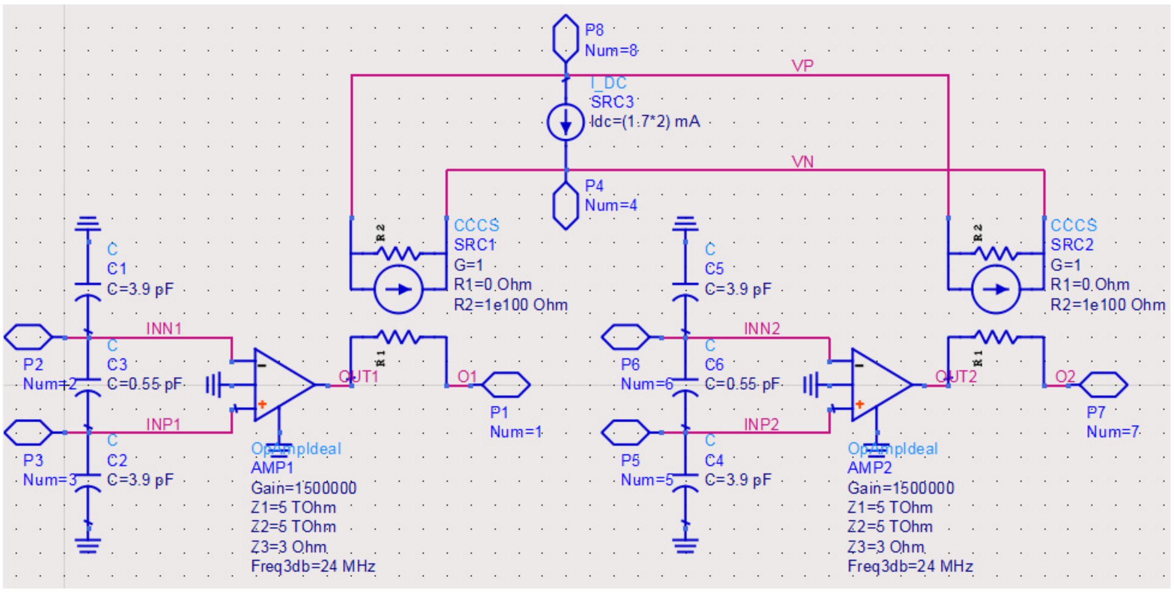


Figure 3.38: Keysight ADS Model of AD8616

3.9 Simulation Results of Complete Analog Measurement Block

The output signals of the complete analog measurement block without ADCs and control block were evaluated using computer simulations. From the output signals, reflection coefficients were calculated and compared to the expected ideal reflection coefficients. Parasitic effects were evaluated and included in the simulations.

3.9.1 Output Signals and Calculated Reflection Coefficients

The data of Table 3.5 was obtained by simulating the whole system with defined load impedances. However, some interconnecting transmission lines that would add phase shift were omitted in this simulation.

As visible in Figure 3.39, the reflection coefficients calculated from these data closely resemble the expected ideal reflection coefficients for that load. The biggest errors were found for magnitude one and argument 45°+n·90° (n = 0 to 3). It could be expected that the system would show less accuracy at magnitudes close to one and high input power, as this is a total mismatch. So a high power signal is reflected and can lead to the production of harmonics that interfere in the mixing operation. However, this condition is improbable in an AIM system, as the impedance is always sufficiently well matched in such a system. It can be observed that the 30 dBm values are in general less accurate. They show an average absolute error of 0.044 while all the lower power levels show an average absolute error of 0.024. It is also observed a remaining phase shift at the 30 dBm values that is not compensated by the system. But these errors can still be tolerated for use in an AIM system with an allowable maximum VSWR of 1.3. Relative errors were not calculated as the smith chart is centered at a reflection coefficient of zero, what would give infinite relative errors for the matched state. As the maximum value for the reflection coefficient is one, the absolute errors are equal to percent full scale relative errors when multiplied by 100.

For this simulation the last setup of the QILO with C = 2.2 pF, Cx = 0.8 pF, R = 200 Ω was used, displaying an f_0 of 417.3 MHz. A system frequency of 416 MHz was chosen for obtaining a reasonable amplitude range for injection locking.

Table 3.5: Simulation Results of Complete Analog Measuring Subsystem

Input	**Load**	**Raw Output Signals**				**Refl.Coef.Calc**		**Refl.Coef.Ideal**		**Error Calc-Ideal**		
Power	***Imped. (Ω)***	***a (V)***	***b (V)***	***c (V)***	***d (V)***	***Re***	***Im***	***Re***	***Im***	***Re***	***Im***	***ABS***
30 dBm	**50**	-0,0147	0,0007	0,9326	-0,1568	-0,0155	-0,0018	0,0	0,0	-0,0155	-0,0018	0,0156
30 dBm	**0**	-0,9186	0,1492	0,9441	-0,1565	-0,9725	-0,0032	-1,0	0,0	0,0275	-0,0032	0,0277
30 dBm	**50M**	0,8934	-0,1468	0,9213	-0,1555	0,9691	0,0042	1,0	0,0	-0,0309	0,0042	0,0312
30 dBm	**-j50**	-0,1972	-0,9120	0,9328	-0,1562	-0,0464	-0,9855	0,0	-1,0	-0,0464	0,0145	0,0486
30 dBm	**+j50**	0,1485	0,9137	0,9328	-0,1564	-0,0048	0,9787	0,0	1,0	-0,0048	-0,0213	0,0219
30 dBm	**10-j20**	-0,5342	-0,4153	0,9384	-0,1567	-0,4820	-0,5230	-0,5	-0,5	0,0180	-0,0230	0,0293
30 dBm	**50+j100**	0,5032	0,4284	0,9270	-0,1559	0,4524	0,5382	0,5	0,5	-0,0476	0,0382	0,0611
30 dBm	**10+j20**	-0,4164	0,5037	0,9384	-0,1565	-0,5188	0,4503	-0,5	0,5	-0,0188	-0,0497	0,0532
30 dBm	**50-j100**	0,3817	-0,5083	0,9270	-0,1562	0,4902	-0,4658	0,5	-0,5	-0,0098	0,0342	0,0356
30 dBm	**350**	0,6904	-0,0846	0,9241	-0,1558	0,7415	0,0335	0,75	0,00	-0,0085	0,0335	0,0345
30 dBm	**7,1429**	-0,7171	0,0882	0,9412	-0,1566	-0,7565	-0,0321	-0,75	0,00	-0,0065	-0,0321	0,0328
30 dBm	**14+j48**	0,0851	0,7094	0,9327	-0,1564	-0,0352	0,7547	0,00	0,75	-0,0352	0,0047	0,0355
30 dBm	**14-j48**	-0,1260	-0,7079	0,9327	-0,1563	-0,0076	-0,7603	0,00	-0,75	-0,0076	-0,0103	0,0128
30 dBm	**100**	0,3090	-0,0198	0,9286	-0,1579	0,3269	0,0343	0,33	0,00	-0,0064	0,0343	0,0349
30 dBm	**40+j30**	0,0139	0,3265	0,9324	-0,1581	-0,0432	0,3429	0,00	0,33	-0,0432	0,0095	0,0442
30 dBm	**25**	-0,3382	0,0218	0,9362	-0,1583	-0,3550	-0,0367	-0,33	0,00	-0,0217	-0,0367	0,0427
30 dBm	**40-j30**	-0,0458	-0,3249	0,9324	-0,1580	0,0097	-0,3468	0,00	-0,33	0,0097	-0,0135	0,0166
30 dBm	**70+j40**	0,2477	0,2306	0,9295	-0,1580	0,2180	0,2851	0,25	0,25	-0,0320	0,0351	0,0475
30 dBm	**70-j40**	0,2045	-0,2572	0,9298	-0,1569	0,2592	-0,2329	0,25	-0,25	0,0092	0,0171	0,0194
30 dBm	**26,9+j15,4**	-0,2355	0,2572	0,9355	-0,1573	-0,2898	0,2262	-0,25	0,25	-0,0398	-0,0238	0,0464
30 dBm	**26,9-j15,4**	-0,2777	-0,2271	0,9355	-0,1570	-0,2491	-0,2846	-0,25	-0,25	0,0009	-0,0346	0,0346
30 dBm	**+j120,71**	0,6906	0,5298	0,9247	-0,1555	0,6327	0,6793	0,71	0,71	-0,0744	-0,0278	0,0795
30 dBm	**-j120,71**	0,4618	-0,6924	0,9244	-0,1574	0,6094	-0,6453	0,71	-0,71	-0,0977	0,0618	0,1156
30 dBm	**+j20,71**	-0,5005	0,6819	0,9406	-0,1578	-0,6358	0,6183	-0,71	0,71	0,0713	-0,0888	0,1139
30 dBm	**-j20,71**	-0,7231	-0,5071	0,9406	-0,1578	-0,6598	-0,6498	-0,71	-0,71	0,0473	0,0573	0,0743
11 dBm	**50**	-0,0008	0,0014	0,0580	-0,0966	-0,0146	0,0000	0,0	0,0	-0,0146	0,0000	0,0146
11 dBm	**0**	-0,0580	0,0941	0,0600	-0,0972	-0,9678	0,0010	-1,0	0,0	0,0322	0,0010	0,0322
11 dBm	**50M**	0,0540	-0,0926	0,0560	-0,0959	0,9646	-0,0006	1,0	0,0	-0,0354	-0,0006	0,0354
11 dBm	**-j50**	-0,0960	-0,0557	0,0581	-0,0965	-0,0154	-0,9849	0,0	-1,0	-0,0154	0,0151	0,0216
11 dBm	**+j50**	0,0927	0,0610	0,0581	-0,0965	-0,0398	0,9845	0,0	1,0	-0,0398	-0,0155	0,0427
11 dBm	**10-j20**	-0,0761	0,0189	0,0590	-0,0969	-0,4919	-0,4862	-0,5	-0,5	0,0081	0,0138	0,0160
11 dBm	**50+j100**	0,0744	-0,0160	0,0570	-0,0962	0,4621	0,4993	0,5	0,5	-0,0379	-0,0007	0,0379
11 dBm	**10+j20**	0,0174	0,0776	0,0590	-0,0969	-0,5041	0,4864	-0,5	0,5	-0,0041	-0,0136	0,0142
11 dBm	**50-j100**	-0,0209	-0,0742	0,0570	-0,0962	0,4755	-0,4994	0,5	-0,5	-0,0245	0,0006	0,0245
11 dBm	**350**	0,0406	-0,0690	0,0565	-0,0961	0,7177	0,0000	0,75	0,00	-0,0323	0,0000	0,0323
11 dBm	**7,1429**	-0,0436	0,0711	0,0595	-0,0970	-0,7322	0,0003	-0,75	0,00	0,0178	0,0003	0,0178
11 dBm	**14+j48**	0,0695	0,0459	0,0581	-0,0965	-0,0315	0,7387	0,00	0,75	-0,0315	-0,0113	0,0334
11 dBm	**14-j48**	-0,0721	-0,0417	0,0581	-0,0965	-0,0123	-0,7389	0,00	-0,75	-0,0123	0,0111	0,0165
11 dBm	**100**	0,0173	-0,0300	0,0559	-0,0972	0,3091	0,0002	0,33	0,00	-0,0242	0,0002	0,0242
11 dBm	**40+j30**	0,0308	0,0206	0,0566	-0,0974	-0,0207	0,3285	0,00	0,33	-0,0207	-0,0049	0,0213
11 dBm	**25**	-0,0192	0,0327	0,0573	-0,0976	-0,3354	-0,0001	-0,33	0,00	-0,0021	-0,0001	0,0021
11 dBm	**40-j30**	-0,0326	-0,0175	0,0566	-0,0974	-0,0114	-0,3285	0,00	-0,33	-0,0114	0,0048	0,0123
11 dBm	**70+j40**	0,0367	-0,0078	0,0561	-0,0972	0,2236	0,2481	0,25	0,25	-0,0264	-0,0019	0,0265
11 dBm	**70-j40**	-0,0112	-0,0363	0,0561	-0,0972	0,2307	-0,2479	0,25	-0,25	-0,0193	0,0021	0,0194
11 dBm	**26,9+j15,4**	0,0085	0,0394	0,0585	-0,0967	-0,2596	0,2447	-0,25	0,25	-0,0096	-0,0053	0,0110
11 dBm	**26,9-j15,4**	-0,0385	0,0101	0,0585	-0,0967	-0,2530	-0,2449	-0,25	-0,25	-0,0030	0,0051	0,0060
11 dBm	**+j120,71**	0,1052	-0,0247	0,0553	-0,0969	0,6593	0,7094	0,71	0,71	-0,0478	0,0022	0,0479
11 dBm	**-j120,71**	-0,0313	-0,1050	0,0553	-0,0969	0,6787	-0,7105	0,71	-0,71	-0,0284	-0,0034	0,0286
11 dBm	**+j20,71**	0,0258	0,1089	0,0582	-0,0978	-0,7066	0,6845	-0,71	0,71	0,0005	-0,0226	0,0226
11 dBm	**-j20,71**	-0,1069	0,0277	0,0582	-0,0978	-0,6896	-0,6832	-0,71	-0,71	0,0175	0,0239	0,0296
18 dBm	**50+j100**	0,1678	0,0370	0,2056	-0,1408	0,4716	0,5028	0,5	0,5	-0,0284	0,0028	0,0285
24 dBm	**50+j100**	0,3105	0,1435	0,4580	-0,1881	0,4699	0,5063	0,5	0,5	-0,0301	0,0063	0,0307

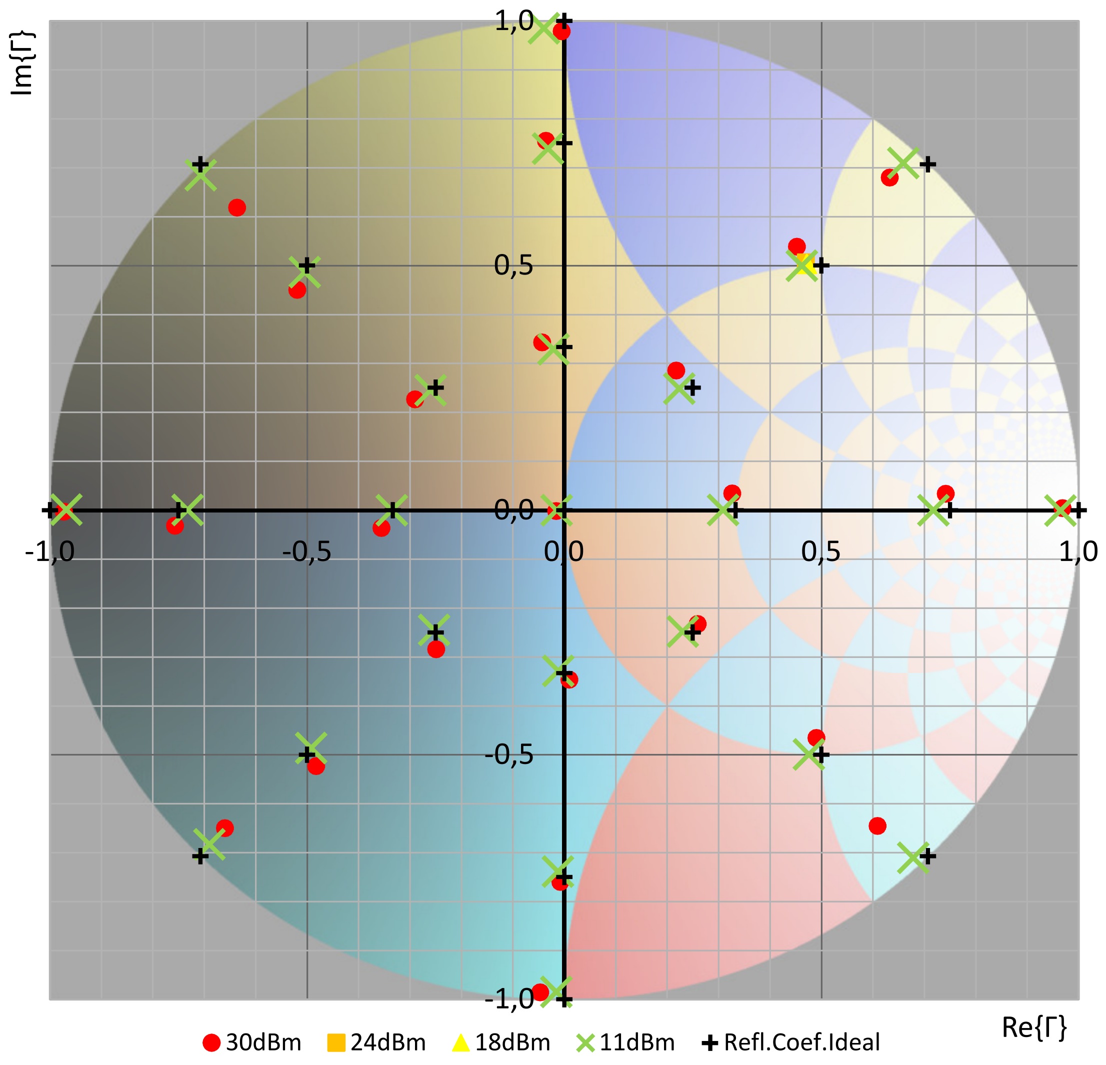


Figure 3.39: Reflection Coefficients as Calculated from Simulation Data

3.9.2 Simulation Time vs. Settling Time

To filter out the RF content of the output signals of the whole system, i.e. to reduce the amplitude of the RF content sufficiently, simple first order filters have to have a relatively big time constant τ. This means that the settling time (about 7τ for an error smaller than 1/1000) also gets big compared to the time period of the RF signal that is being analyzed. The used RF filter had a settling time of about 35 µs compared to a time period of the RF signal of only 0.0025 µs. In other words, the simulation time becomes long and the maximum time step small, what leads to high computation load and long duration of simulations.

The advantage of using a low order filter is the lower sensitivity to component tolerances and parasitics compared to a filter with a steeper roll-off like a higher order Chebychev. So it is expected that a low order RF-filter made from discrete components behaves more like the simulated one.

3.9.3 Parasitics

The effect of most parasitics was evaluated for all analog parts of the system. The majority of analyzed parasitic effects were found to have a small influence. Transmission line effects were modeled and evaluated, but some of the transmission lines were omitted in the complete system performance simulations for the sake of simplicity of the model of the circuit to ensure convergence.

In the QILO, the parasitics simulations included imperfections of the frequency defining components of the oscillator such as equivalent series resistance (ESR), equivalent series inductance (ESL) and equivalent parallel capacitance (EPC), as well as the parasitic capacitance of the tracks in the oscillator and the parasitic inductance of the longer tracks between the two parts of the quadrature oscillator (quadrature connects). Other parasitic track inductances were not directly taken into account. Instead, transmission line models for all the tracks in the PA, the Preamplifier, the QILO including the input- and output buffers and all other RF parts of the circuit were used.

The influence of the ESR of the capacitors (0.2 Ω worst case [49]) and the EPC of the resistors (0.05pF for 0603 package worst case) as well as the track capacitances in the oscillator (about 2.1 pF worst case) were found to have only a small influence on the system (about 2% worst case change in oscillator's free-running frequency f_0), while the ESL of the capacitors in the quadrature interconnects had an influence of about 8.5% on the oscillator's f_0. This ESL value of 220 nH was found by adjusting the self-resonant frequency (SRF) of the capacitor Cx (0.8 pF) to 380 MHz and is probably not very realistic. The SRF of a 100pF 1206 NP0-ceramic capacitor is 380 MHz [50], with a ESL of 1.75 nH, a similar or even lower value of ESL should be expected for the 0.8pF 0603 capacitor, generating an SRF of at least 4.25 GHz and thus a negligible influence on the QILO.

A strong influence of the transmission lines in the quadrature interconnects of the QILO was found. This can make the oscillator run at a completely different frequency, for example a difference in f_0 of 200% or more, depending on the position of the capacitors in the transmission lines and the total length. As it is expected that this effect can be neglected when the system is produced in integrated circuit technology, this effect was not taken into account in the complete system performance simulations. Because it was not possible to remove the transmission lines completely in the PCB layout, their effect was used as a part of the quadrature interconnect mechanism, providing part of the 90° phase shift between the two oscillator cores.

In the Frontend that holds all circuit parts besides the PA, the Preamplifier and the QILO, the main influence of the parasitics was the one of the transmission lines. Especially the transmission line between the unknown impedance input and the directional coupler is critical, as it has various sections with different impedances and lengths. The behavior of this transmission line was simulated and a correction equation was derived. The correction of the measurement values is covered in more detail in section 4.4.3.

Also, the transmission lines between the coupler incident and reflected outputs and the RF-inputs of the mixers were evaluated. They contribute a constant phase shift for constant frequency that is strongly dependent on the properties of the used PCB material, especially its permittivity. However, this phase shift can easily be removed through a calibration process in the software of the system processor, for example by transforming the values measured in a skew coordinate system into an orthogonal Cartesian one as is shown in section 4.4.3. If space on the PCB is not a constraint, it is also possible to remove this effect in the layout by making the transmission lines all same length. Thus, these transmission lines were omitted in the complete system performance simulations. All other transmission lines were modeled as such and used in the simulations.

3.10 Analog to Digital Converters (ADC) and System Processor

The position of the analog to digital converter blocks and the system processor / control block in the measurement system block diagram is shown in Figure 3.40:

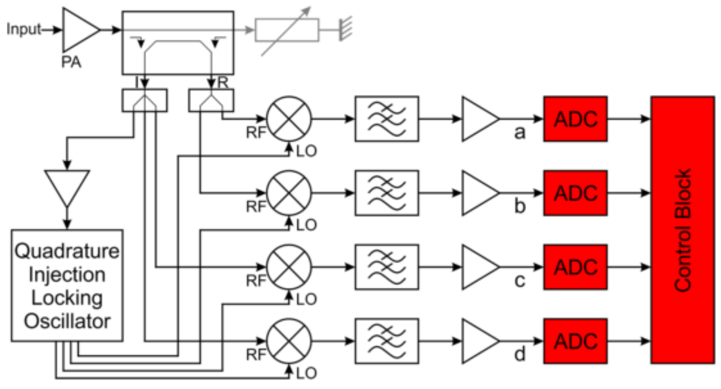


Figure 3.40: Position of ADCs and Processor in System

The ADCs convert the analog low frequency signals from the low frequency amplifier (LFA) blocks to digital signals that can be used by a digital control block, the system processor. The system processor calculates the reflection coefficient from the digital signals and, in a complete automatic impedance matching system, acts on the variable impedance matching network. The ADCs have a direct influence on the control speed of the system, so it is desirable if they have a high conversion speed. On the other hand, the accuracy of the measurement of the reflection coefficient depends on the resolution of the ADC. Higher resolution generally means lower speed. Modern ADCs are frequently integrated into microcontrollers and can reach several million samples per second (MSPS) at a resolution of 12 bit. There are faster and more accurate ADCs available on the market, but it was decided to stay with something that can be integrated into a microcontroller with standard processes.

3.10.1 STM32F429 Microcontroller

The STM32F429 (F429) is a flagship microcontroller of ST Microelectronics [51]. It combines the ARM Cortex M4 Core that includes a floating point unit and DSP functionality with many peripherals, especially three advanced 12-bit 2.4MSPS successive approximation (SAR) analog to digital converters (ADC) and a graphics accelerator enabling the direct use of

LCD-TFT-Displays. It supports a master clock rate of up to 180 MHz. These features make it a good choice for digital signal processing and control, though it is by far not as powerful as a modern personal- or single-board-computer, what would be exaggerated for the application. Especially the three ADCs and the availability and relatively low cost were a good argument for using this device.

3.10.2 Analog Limitations of the STM32F429 Discovery Board

The STM32F429 discovery board (F429DISCO shown in Figure 3.41) connects all general purpose input-outputs of the F429 microcontroller to headers, allowing the user to measure signals or to connect further electronics [52]. However, there is already quite a lot of digital electronics connected to the F429 on the discovery board (especially the display), effectively preventing the use of more than three pins for analog signals. This means that, even though the microcontroller could convert many more channels than three sequentially or alternating, the discovery board does not allow this. As it turned out later in the project, this is a severe limitation. The solution for this problem is the use of another microcontroller and discovery board for the ADC operation as will be detailed in the next section.

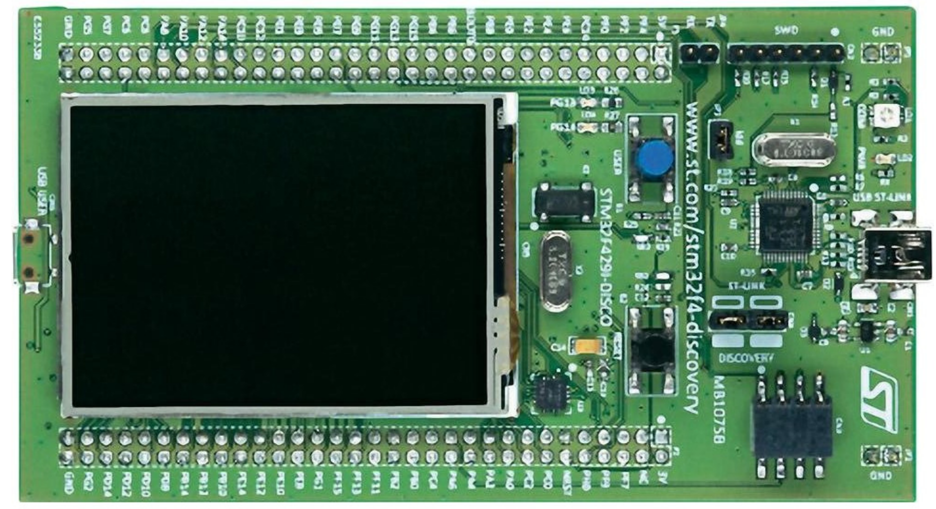


Figure 3.41: STM32F429 Discovery Board (product photograph by ST)

3.10.3 STM32F303 Microcontroller and STM32F3 Discovery Board

The STM32F303 (F303) also incorporates an ARM Cortex M4 Core with floating point unit and DSP functionality, but operates at a lower frequency than the F429 (up to 72 MHz, in this project 64 MHz clock speed were used) [53]. It is part of ST Microelectronics' mainstream/analog microcontroller line and incorporates ultra fast timers, comparators, operational amplifiers and four advanced 12-bit 5 MSPS SAR ADCs. Also, the STM32F3 discovery board [54] (F3DISCO, shown in Figure 3.42) has more free pins than the F429DISCO. This allows more than the necessary four analog signals to be connected. Additionally, it is possible to configure all four ADCs to sample simultaneously. This has the advantage that there is no error in the measurement of the reflection coefficient when it is varying. If the four values were recorded at different times, they would not necessarily represent the same reflection coefficient.

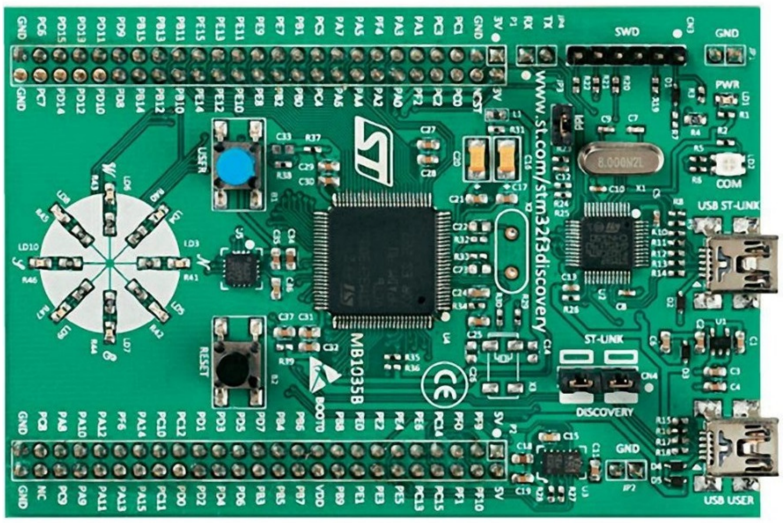


Figure 3.42: STM32F3 Discovery Board (product photograph by ST)

3.10.4 Combination of the two Discovery Boards

It is easily possible to combine the two discovery boards by means of a digital interface. This could be an asynchronous or a synchronous serial interface or a parallel interface. After checking the F429DISCO for free interfaces, it was decided to use a Serial Peripheral Interface (SPI) connection. The asynchronous alternative was ruled out because the F3DISCO's clock source is less accurate, what could lead to bit errors on an asynchronous interface. It would be possible to solder a crystal onto the board, but in factory configuration, this is not populated, and the internal RC oscillator is used.

The main reason for combining the two boards was to have the display of the F429DISCO together with the four ADCs of the F3DISCO. However, there is no need for the display in a possible final system, and the F303 is by far fast enough for all operations that are necessary for impedance matching, so theoretically, the whole digital part of the automatic impedance matching system could be the F303 or even a simpler microcontroller.

3.11 Measuring and Control Software

The measuring and control software is being executed by the system processor and completes the measurement system. Amongst other purposes, it is responsible for the last step in the reflection coefficient measurement, the calculation of the reflection coefficient from the four input values.

3.11.1 First Version using the three ADCs of the STM32F429

In the beginning of the project, it was not clear that the injection locking oscillator would produce a variable phase shift. Especially, a scheme to do injection locking and phase locking in the same oscillator was evaluated [40]. Unluckily it was found that this scheme would be rather complicated to test at 400 MHz with discrete components, and a possible 180° phase uncertainty (because of its inherent double frequency operation) could not be ruled out. However, this scheme would have allowed to do the measurement of the reflection coefficient by measuring only three low frequency analog values.

This was the starting point for the first version of the microcontroller software that was written in the beginning of the project. It already incorporated most of the blocks of the final program besides the measurement value correction. For instance, there were already drivers for AD-converters, timers, output pins and the display, filtering and decimation of the analog values, calculation of the reflection coefficient and analysis and control of the most adequate switching combination to be used for the variable impedance matching network as well as showing the actual state of the process on the display.

The software modules are depicted in Figure 3.43. The Matcher / State Switcher algorithm is based on the matching circles theory described in [3]. It is assumed that it is possible to find the best matching condition based on a geometric distance calculation to the centers of the available matching circles and comparing with the radiuses. A matching state is chosen when the corresponding circle is hit and the last matching circle is left.

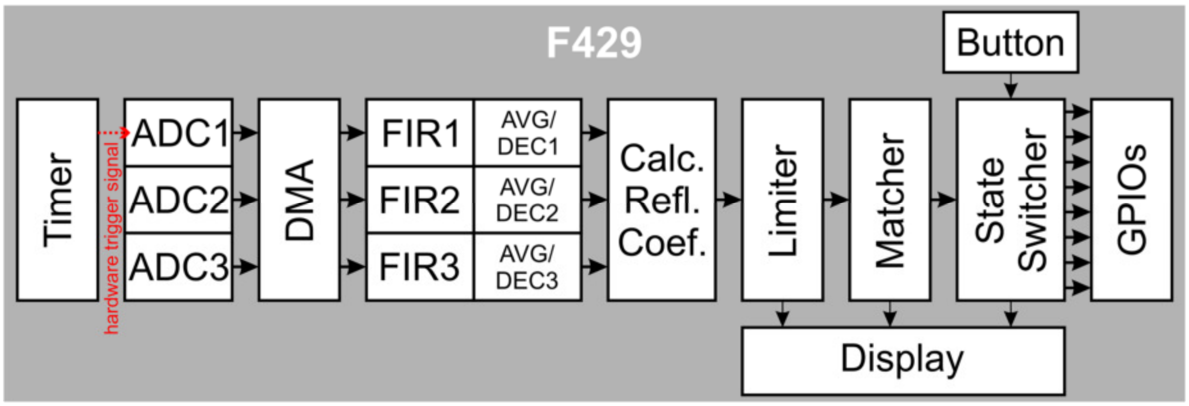


Figure 3.43: First Version of Software (3 ADCs)

As there was no variable impedance matching network available at this point, the state of the switching outputs was shown using LEDs (to make the control action work in an open loop system, it was only executed when a button was pressed). The three input voltages were generated using potentiometers, as there was also no measurement circuit available at this time (as this could produce reflection coefficients with a magnitude greater than one, a Limiter software module had to be included). This hardware setup is shown in Figure 3.44.

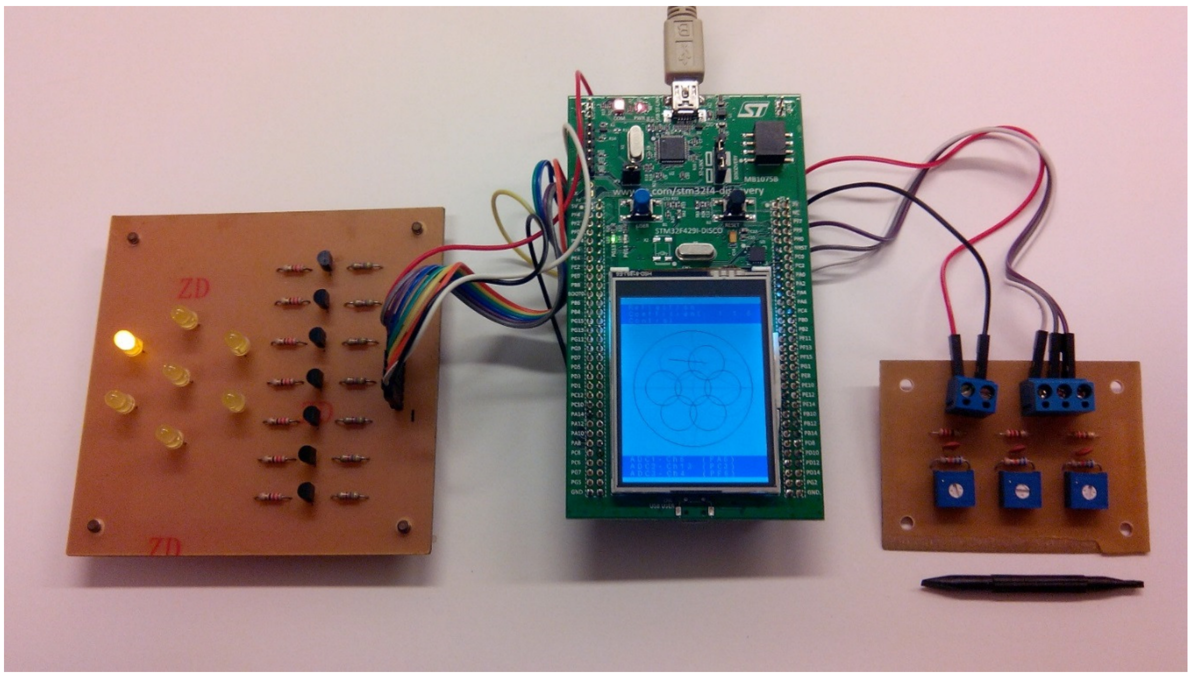


Figure 3.44: Hardware for Testing the First Version of the Software

3.11.2 Second Version using the four ADCs of the STM32F303

When it became clear that there would be a variable phase shift between the input of the system and the outputs of the QILO, a solution was necessary, as this would make the measurement of the reflection coefficient with three LF analog values impossible. This solution consisted of using a fourth mixer and thus generating a fourth LF analog signal that needed to be measured. Using quadrature measurement for the forward as well as for the reflected voltage made it possible to remove the phase shift directly in the calculation of the reflection coefficient. However, as noted before, the STM32F429 has only three ADCs to measure signals simultaneously, and the used discovery-board only allows the connection of three analog signals. So it was necessary to change or modify the platform. To minimize the necessary training time and implementation work, it was decided to keep using the STM32F429 discovery for calculation of the reflection coefficient, variable impedance matching network state control operation and displaying the state and moving the ADC-operation and the filtering to the most similar microcontroller available that has four equal ADCs that can sample simultaneously.

In this case, the STM32F303 microcontroller and the STM32F3 discovery board were chosen. The main changes to the software were to split it in two parts, one for the F429 and one for the F303, add a fourth channel to the ADC, to the filtering and the reflection coefficient calculation routines and add routines to send the filtered values from the F303 to the F429 via the SPI interface. Furthermore, means of correcting the measured values were introduced into the software, including an editing function and saving the correction values to the flash memory of the microcontroller.

The combination of the modules of this software version is shown in Figure 3.45. A photo of the digital part of the measurement subsystem comprising of the two microcontroller discovery boards and a board with LEDs simulating the variable impedance matching network is shown in Figure 3.46. A two screenshots of the F429 display are shown in Figure 3.47. A listing of the program can be found in Appendix A and B. The used integrated development environment (IDE) was EmBlocks (now EmBitz) [55]

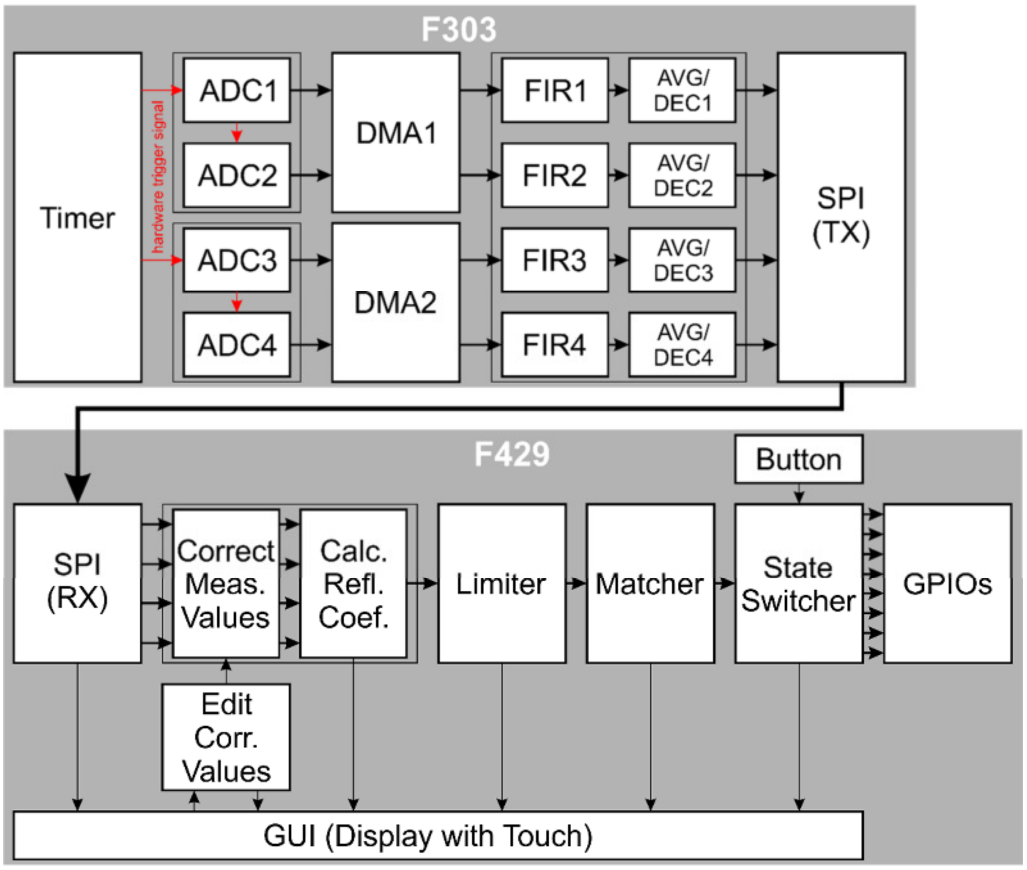


Figure 3.45: Final Version of Software (4 ADCs)

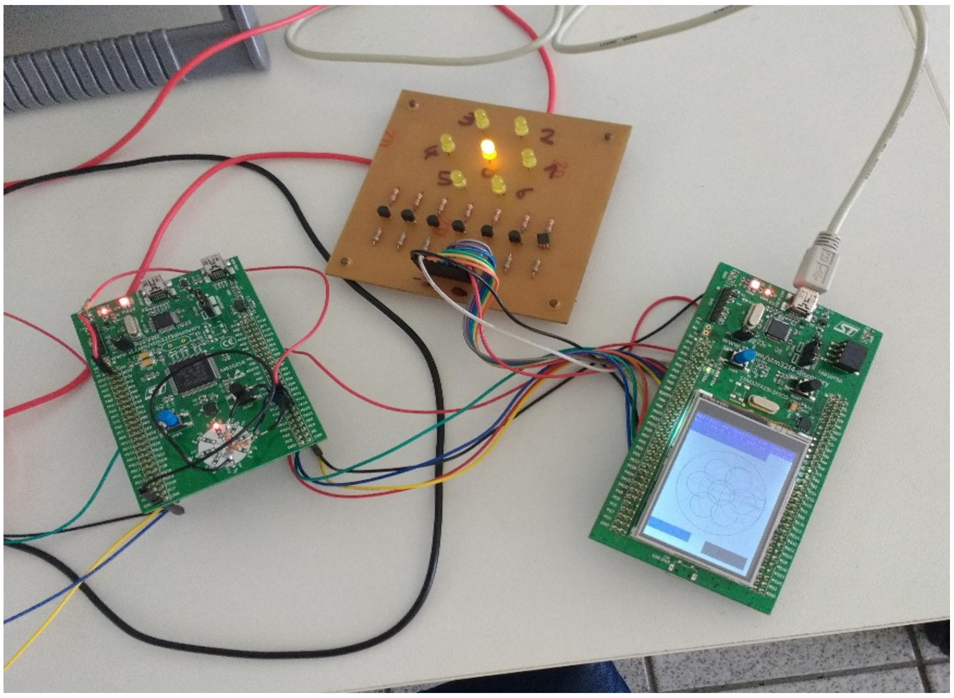
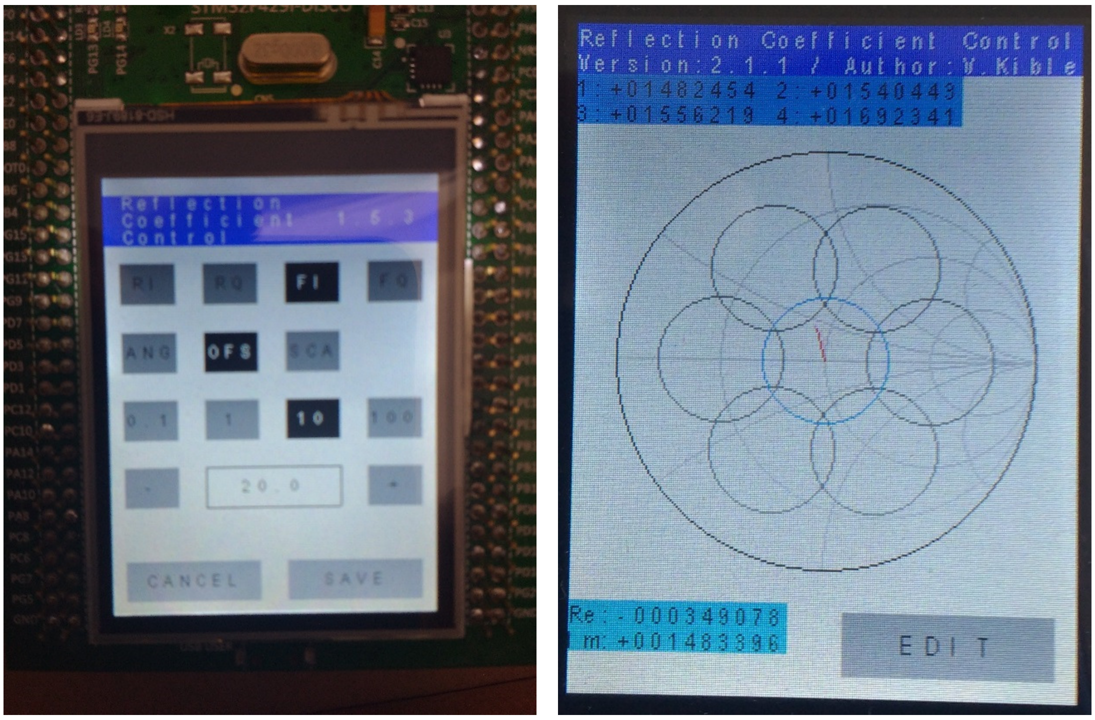


Figure 3.46: Hardware for Final Version of Software



```
Reflection Coefficient Control
Version:2.1.1 / Author:V.Kible
1:+01482454 2:+01540443
3:+01556219 4:+01692341
Re:-000349078
Im:+001483396
```

Figure 3.47: Screenshots of Edit Function (v1.5) and of Main Screen (v2.1)

3.12 Impedance Matching Network

The variable impedance matching network (IMN) is not part of the measurement block, but of an automatic impedance matching system that can be constructed using the measurement block. In Figure 3.1, it would be between the coupler and the load. The IMN needs to be able to match complex impedances above and below 50 Ω, with an imaginary part less than, equal to or above zero. Ideally, all possible reflection coefficients should be matchable. Distributed topologies are not preferred for frequencies as low as 400 MHz, especially when having a solution for integration on an IC in mind. The simplest lumped passive network that theoretically allows for matching all impedances (with a positive, non-zero real part) is a three component T or Pi-Network, with all of its components variable [56]. Two component lumped passive networks can generally only match half the area of the Smith Chart.

As the T-topology has two of its components not connected to ground, making all of its components variable (electronically controllable) is generally more elaborate. Capacitors generally consume less space and have a higher Q-factor and accuracy than inductors considering a future IC implementation. So the preferred solution is the Pi-Network with two capacitors to ground and one inductor between input and output.

The variable inductor can conveniently be transformed into a variable capacitor to ground by using two identical passive impedance inverters, each also a Pi-Network consisting of two constant (for constant frequency) capacitors and one constant inductor, with the impedances of –j·50 Ω and +j·50 Ω, respectively (at 400 MHz, this results in about 8 pF and 20 nH). The result is shown in Figure 3.48.

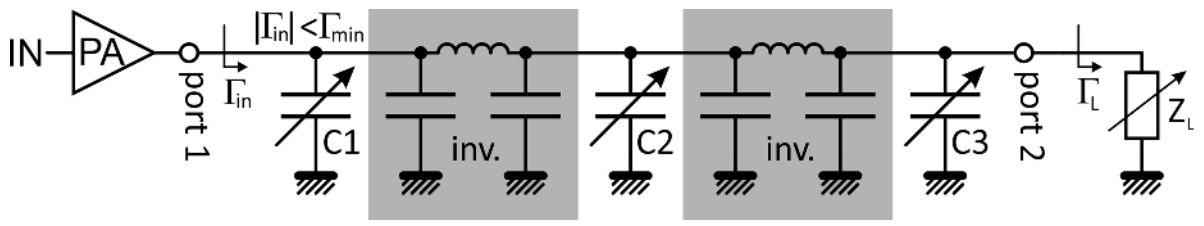


Figure 3.48: Schematic of Variable IMN with Impedance Inverters (inv.)

This topology has already been proposed in [56]. Here, the topology is analyzed numerically according to the matching circles theory proposed in [3] for some combinations of values of the three variable capacitors with the help of a MATLAB script (see Appendix C).

The matching circles depict the area on the Smith chart each configuration of the variable impedance matching network can match with a given allowed error. Different configurations of the matching network will have different circles. Basically, a reflection coefficient in the matching circle (blue) is transformed to a new reflection coefficient close to the center of the smith chart (inside the red circle). This reduces the magnitude of the reflection coefficient to an allowable value ($|\Gamma_{in}| < \Gamma_{min}$). It is assumed that the variable capacitors consist of banks of fixed capacitors with discrete values connected to ideal switches.

In the MATLAB script, the system equation is entered in convenient steps that keep each partial equation small and minimize the probability of errors. In the top part of the script, three vectors for the desired values of the capacitors are defined. The script only analyzes the combination of all three values with index 1, then the combination of all three values with index 2, and so on. This way, full control over which combinations to analyze is given. Also, a proprietary function to display the circles in the smith chart was written. Figure 3.49 shows the output of the script for some combinations of C1, C2 and C3 as an example, demonstrating that the whole smith chart area can be matched. Matching reflection coefficients with a magnitude close to unity is generally difficult, and matching a total mismatch condition is impossible.

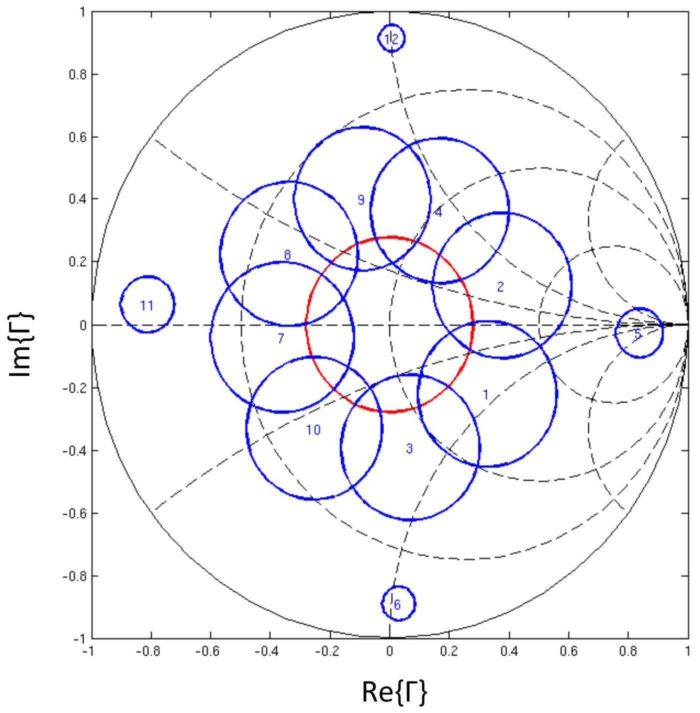


Figure 3.49: Example Matching Circles for an allowed Error of Γ of 0.28

The values of the capacitors C1, C2 and C3 used for calculating and drawing the matching circles shown in Figure 3.49 are shown in Table 3.6:

Table 3.6: Capacitor Values for Matching Circles

Circle	C1 (pF)	C2 (pF)	C3 (pF)
1	1	10	2
2	0.5	10	5
3	5	10	0.5
4	0.5	8	8
5	0.5	28	2
6	25	10	0.5
7	9	4.4	0.5
8	9	2	0.5
9	0.5	4.4	10
10	9	7.2	0.5
11	25	2	2
12	0.5	28	10

The MATLAB script first calculates the output impedance of the matching network in each configuration when the input is terminated with 50 Ω, and from this the (complex) load reflection coefficient *S_{22}* for a 50 Ω load. *S_{22}* and the real valued allowed error *Γ_{min}* are then inserted into the equations (3.20) and (3.21) for the complex center and the real valued radius [3], and the result is handed over to the circle display routine.

$$Center = \frac{S_{22}^{*} \cdot (1-\Gamma_{min})}{1-|S_{22} \cdot \Gamma_{min}|^2} \tag{3.20}$$

$$Radius = \frac{\Gamma_{min} \cdot (1-|S_{22}|^2)}{1-|S_{22} \cdot \Gamma_{min}|^2} \tag{3.21}$$

In the following chapter, measurements on the PCB implementation of the proposed reflection coefficient measurement block and their results are presented. The PCB setup comprising of three PCBs is introduced and the function tests and results of each PCB are reported.

4 Measurements on a Printed Circuit Board Setup

To test the function of the measurement block, a printed circuit board (PCB) setup was designed and manufactured and its performance was evaluated at 400 MHz using the microcontroller boards with dedicated software described in sections 3.10 and 3.11, respectively, and an Agilent E4438C (250 kHz - 3.0 GHz) Signal Generator, an Agilent E5071C (100 kHz - 8.5 GHz) Vector Network Analyzer (VNA), an Agilent 34401A (6.5 Digit) Precision Multimeter and a Maury Microwave Corp. 8045C Coaxial Slide Screw Tuner (0.9 – 12.4 GHz). As the tuner was used outside its range, an additional cable length was necessary to generate phases in a range of more than about 160°. Finally, an Agilent E4404B (9 kHz - 6.7 GHz) Spectrum Analyzer and an Agilent Infinii Vision MSO7104B (1 GHz, 4 GSa/s) Oscilloscope were used to test subsystems such as the QILO.

4.1 Printed Circuit Boards

First, three Printed Circuit Boards (PCB) were designed and manufactured. For the design, the layout functions of Keysight ADS were used, and Gerber data was exported. The Gerber data of the three boards were then sent to the German Würth Electronic PCB manufacturing service. Solder stop mask and silk screen were not used. It was decided to let the boards produce professionally because of the higher quality surface options such as gold, that simplify soldering, and because of the vast number of vias. Vias are contacts between top- and bottom layer by means of a metallized hole. It is recommended to use many vias in parallel for RF designs to reduce parasitic inductance.

Standard FR-4 material was used instead of a specialized RF material in order to reduce costs and to gain higher mechanic stability. RF materials are often slightly flexible, what is not good for the used ceramic capacitors especially and for surface mount devices in general. During layout, it was attempted to reduce track lengths and other parasitic effects where possible. Where it was not possible, transmission lines with defined impedances were used, though it was found that these impedances are not so well defined for standard FR-4 material after all, as the thickness of the material can vary by around 0.1mm and the relative permittivity can vary from about 4.1 to 4.9 (typical value 4.6), as well as the track widths can show a variation in the order of 0.1 mm depending on the used technology. Simulations with Keysight ADS LineCalc show a variation of the impedance in the order of 13%, while the electrical angle varies in the order of 9% for a nominal 50 Ω transmission line on a 1.6 mm thick PCB with the given variation of the PCB parameters.

A photo of the structured top layer of the three boards is shown in Figure 4.1. The Frontend is visible on the upper left of the photo, three copies of the PA can be seen on the lower left and the QILO is shown on the right. Later, the boards were populated with components and characterized. Populating the boards was done in steps, always measuring the basic function of the parts of the circuit once one step was completed, this way making sure the components and blocks of the circuit were connected correctly. Photos of the populated boards are shown in Figure 4.2, Figure 4.3 and Figure 4.4.

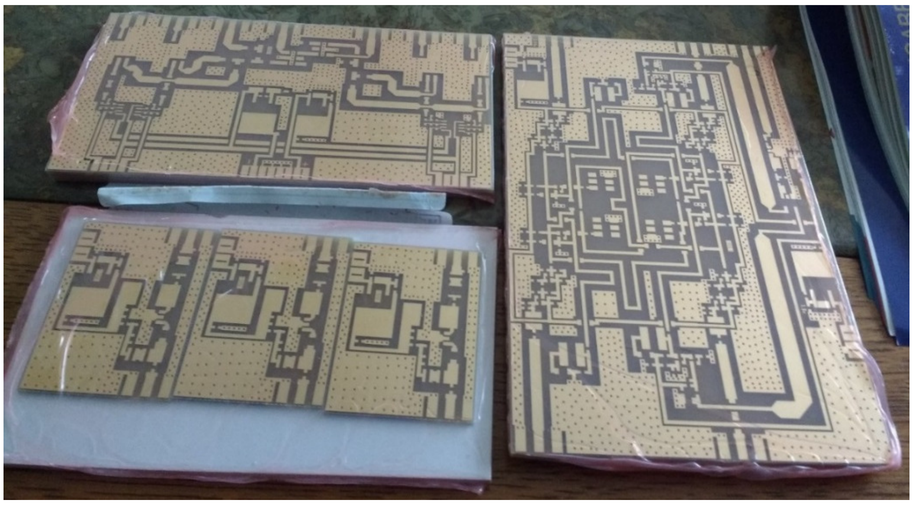


Figure 4.1: The three unpopulated PCBs (three copies each, still wrapped in foil)

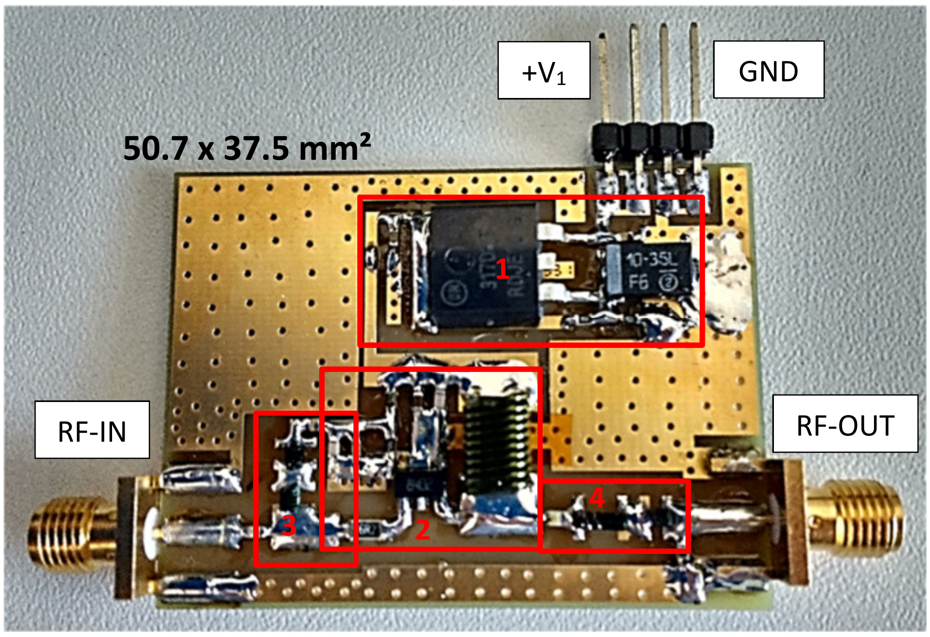


Figure 4.2: Power Amplifier

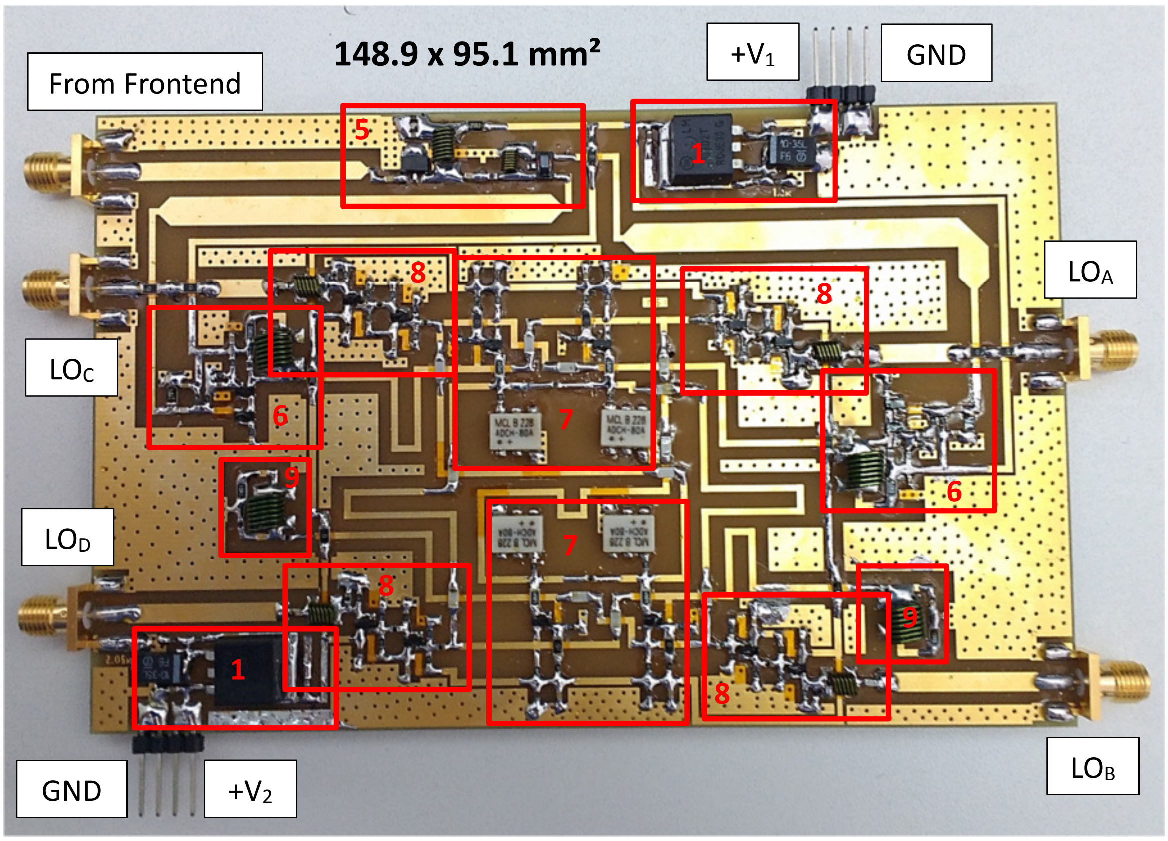


Figure 4.3: Quadrature-Injection-Locking-Oscillator with Voltage Pre-Amp. and Buffers

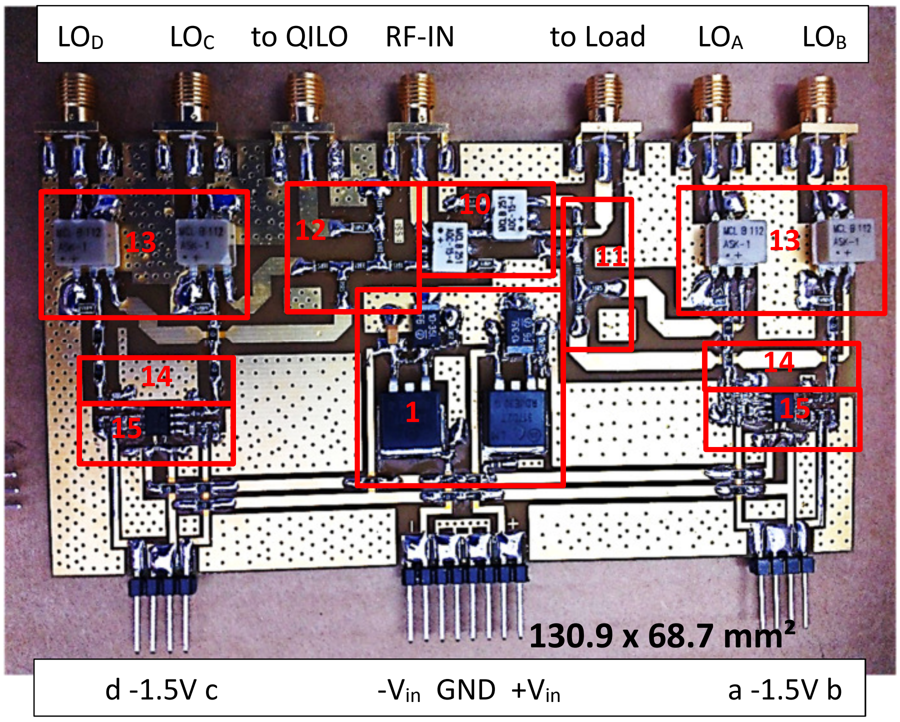


Figure 4.4: Frontend

Legend for numbers in the photographs

1	Voltage regulator
2	Amplifier core with transistor and RF choke as well as biasing networks.
3	Input impedance matching network
4	Output impedance matching network
5	Preamplifier for injection locking
6	Input buffers
7	Oscillator cores
8	Output buffers
9	Dummy load
10	Directional coupler
11	Splitter / Attenuator for V_r
12	Splitter / Attenuator for V_i
13	Mixers
14	Filters
15	Low frequency amplifiers

4.2 Power Amplifier Board

Measurements on the amplifier board show that it functions reasonably well, but the key parameters are almost all less good than they were in the computer simulations, just the power added efficiency was better. This is probably due to the poor quality of the computer simulation model of the power transistor. Though gain, 1dB compression point and efficiency are lower, it was possible to obtain almost 30 dBm output power, though the gain compression was more than 2dB during this operation.

4.2.1 S-Parameters

As shown in Figure 4.6, the real input reflection coefficient of the amplifier as measured with the VNA is not as low as the one obtained in the simulations. Also, shown in Figure 4.7, the gain is lower. Small signal S_{21} at 400 MHz is about 16dB instead of about 22dB. This is probably due to the transistor simulation model, that is not too accurate. For example, the drain current of the model was in the order of 600 mA while the datasheet gives a drain current close to 100 mA for the used bias point of 4 V gate-source voltage. Power amplifiers with discrete transistors are often built based on measurement data, it is unusual that the manufacturer supplies a model for a power MOSFET. Also, imperfections like parasitics and tolerances of the other real components compared to the idealized models of the simulation can be a reason, and finally, the PCB itself has parasitic effects like additional capacitance at the nodes, and inductance and resistance of the tracks, though the latter two should not have too much effect in this design. The measurement setup with the VNA is shown in Figure 4.5.

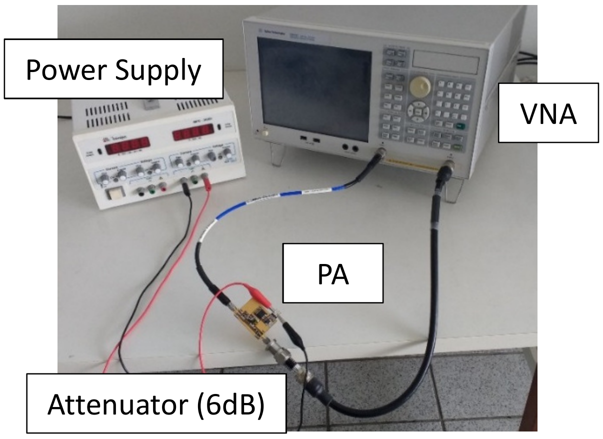


Figure 4.5: Measurement Setup for Obtaining the S-Parameters

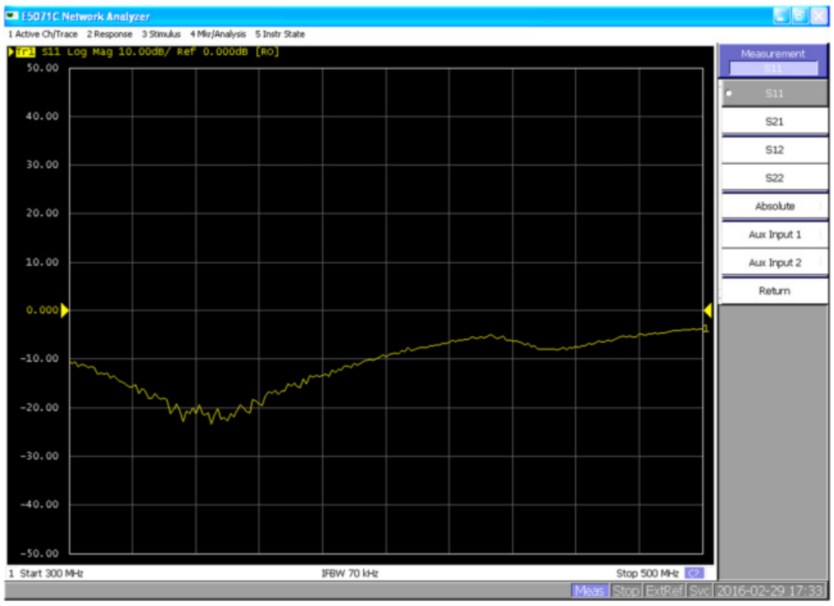


Figure 4.6: PA S_{11}

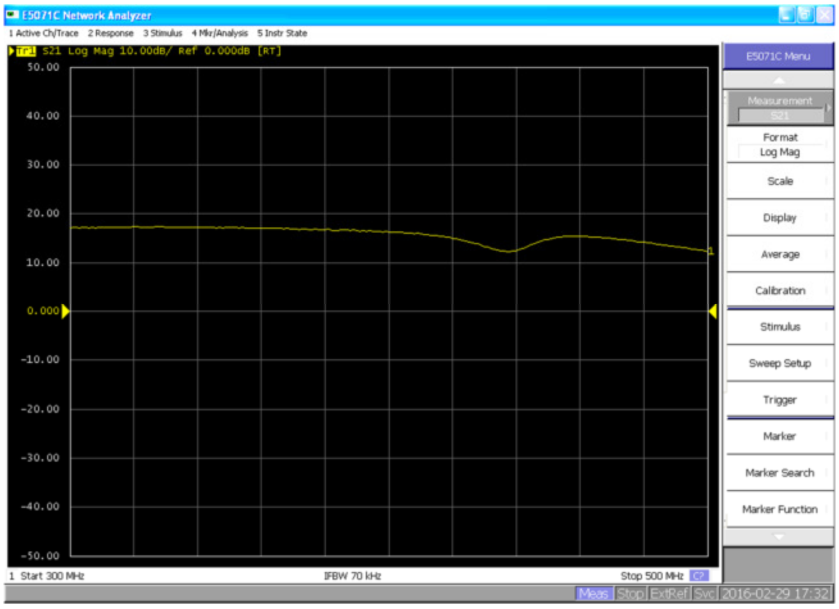


Figure 4.7: PA S_{21}

4.2.2 Output Power and 1dB Gain Compression Point

The 1dB gain compression point at 400 MHz was found to be at 27 dBm output power (12 dBm input power), using the signal generator at the input and the spectrum analyzer at the output of the PA together with an attenuator of 6dB. The input power was raised in steps and the remaining gain was calculated. This 1dB compression point is significantly lower than the value from the computer simulations of about 30.2 dBm output power. This, amongst other reasons already stated before in section 4.2.1, can partly be due to imperfections of the power supply. A voltage regulator with a real output of 6.8 V output was applied, the design supply was 7.0 V. This also leads to a lower bias point of the transistor, as the input bias voltage is derived from the main power supply via a resistive voltage divider as shown in Figure 3.3.

A graph of the measured power delivered to the load P_L versus the RF frequency for a power available from the source P_{AVS} from 0 dBm to 10 dBm is shown in Figure 4.8. The VNA does not allow more output power than 10 dBm. The graph was obtained from the S_{21} measured by the VNA at different power settings, the measurement setup is the same as in Figure 4.5. The power gain G_P versus the RF frequency and P_{AVS}, obtained by equation (3.3) from the S-Parameters measured by the network analyzer is shown in Figure 4.9.

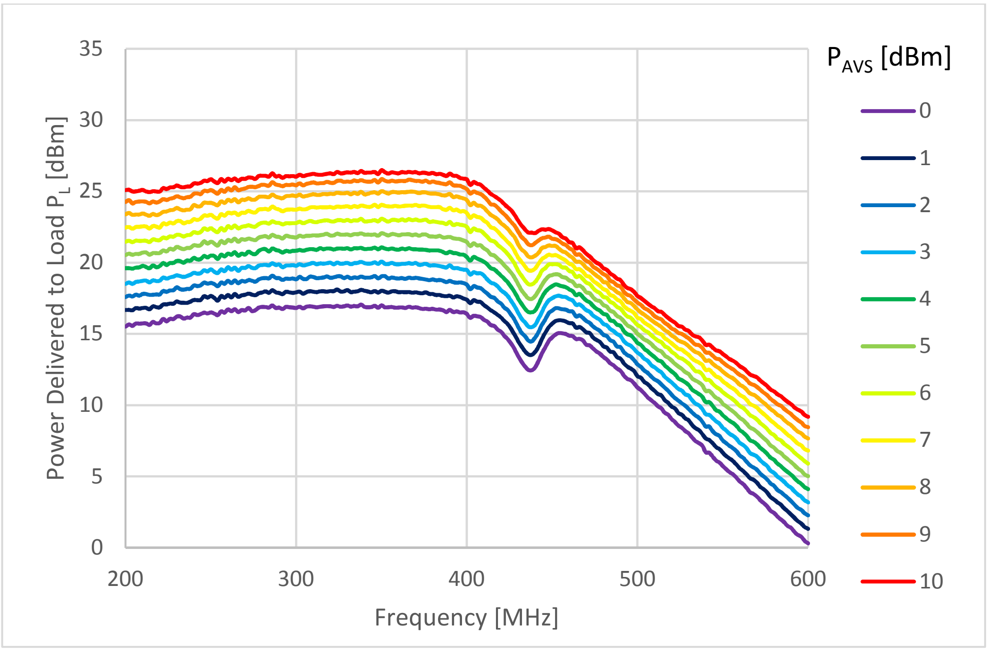


Figure 4.8: Power Delivered to Load P_L vs. Frequency and P_{AVS} (Measured)

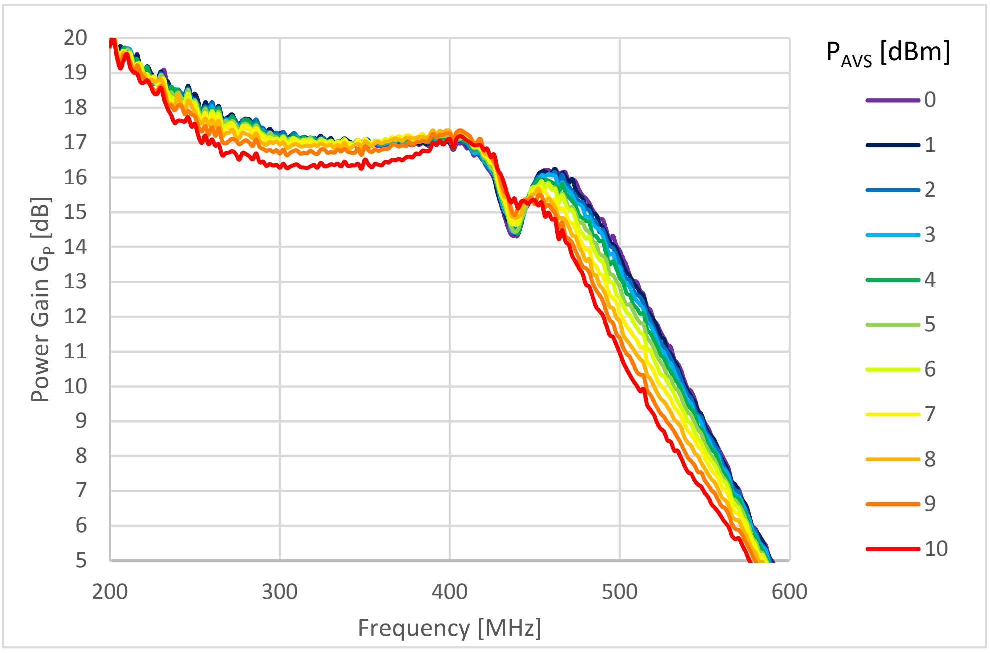


Figure 4.9: Power Gain G_P vs. Frequency and P_{AVS} (Measured)

4.2.3 Power Added Efficiency

Power added efficiency (PAE) [57] at 28 dBm output power (630 mW) is about 56%. The measured supply current was 160 mA at 6.8 V, that is about 1088 mW and input power was 14 dBm = 25 mW. The PAE was calculated using equation (4.1). A comparison between simulation and measurements is shown in Table 4.1.

$$PAE = \frac{P_{out}-P_{in}}{P_{DC}} \cdot 100\% \rightarrow \frac{630mW-25mW}{1088mW} \cdot 100\% \approx 56\% \tag{4.1}$$

Table 4.1: Power Amplifier - Comparison Simulation / Measurements

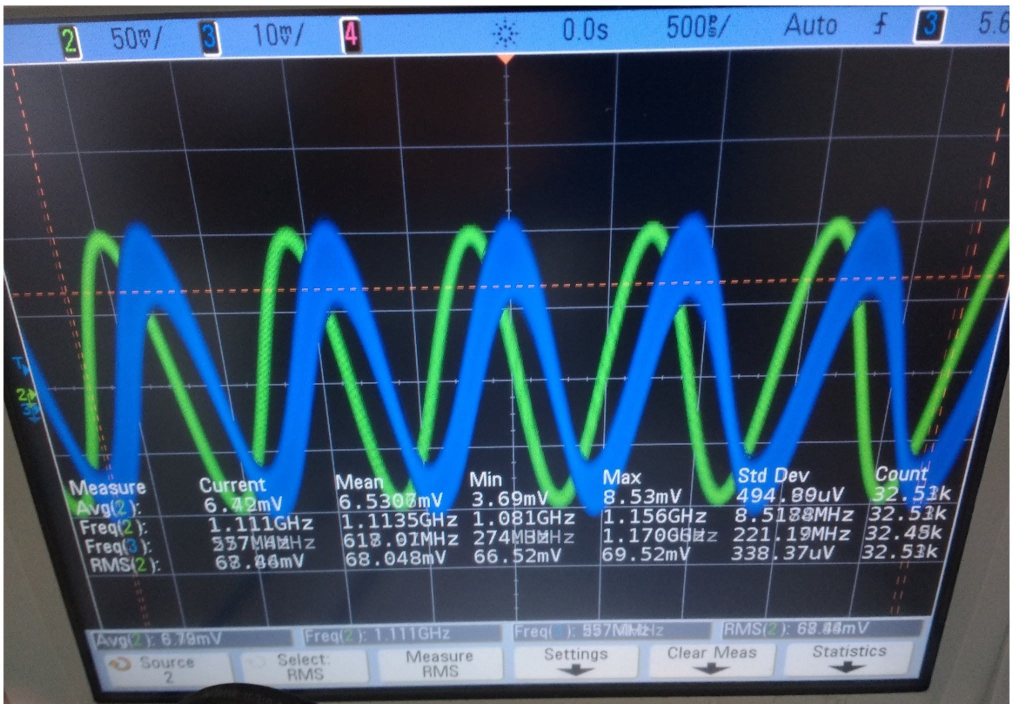
	P_{omax} (dBm)	1dB GCP (dBm)	S_{11} (dB)	S_{21} (dB)	PAE (%)
Simulation	32	30,2	-50	22	37
Measurement	28	27	-10	16	56

4.3 Oscillator Board

While the Frontend and the PA were basically working right away, the same is not true for the QILO. The following section will describe the problems encountered while testing the oscillator board.

4.3.1 First PCB Setup

Due to the parasitics of board and components and the non-ideal isolation of the output buffers, it was not possible to obtain the desired operation of the oscillator core at 400MHz. However, the oscillator worked in quadrature at about 1.1 GHz, but with a strong phase- and amplitude noise and low amplitudes. A photograph of the oscilloscope screen is shown in Figure 4.10 and the measurement setup is shown in Figure 4.11. This was obtained using variable capacitors (variable in the range 1.0 to 2.5 pF) as the main oscillator capacitors, but it was found that the influence of these capacitors on the obtained frequency was negligible.



Measure	Current	Mean	Min	Max	Std Dev	Count
Avg(2):	6.42mV	6.5307mV	3.69mV	8.53mV	494.89uV	32.51k
Freq(2):	1.111GHz	1.1135GHz	1.081GHz	1.156GHz	8.5188MHz	32.51k
Freq(3):	557MHz	618.01MHz	274MHz	1.170GHz	221.19MHz	32.48k
RMS(2):	68.46mV	68.048mV	66.52mV	69.52mV	338.37uV	32.51k

Figure 4.10: Oscilloscope Screen of QILO Output

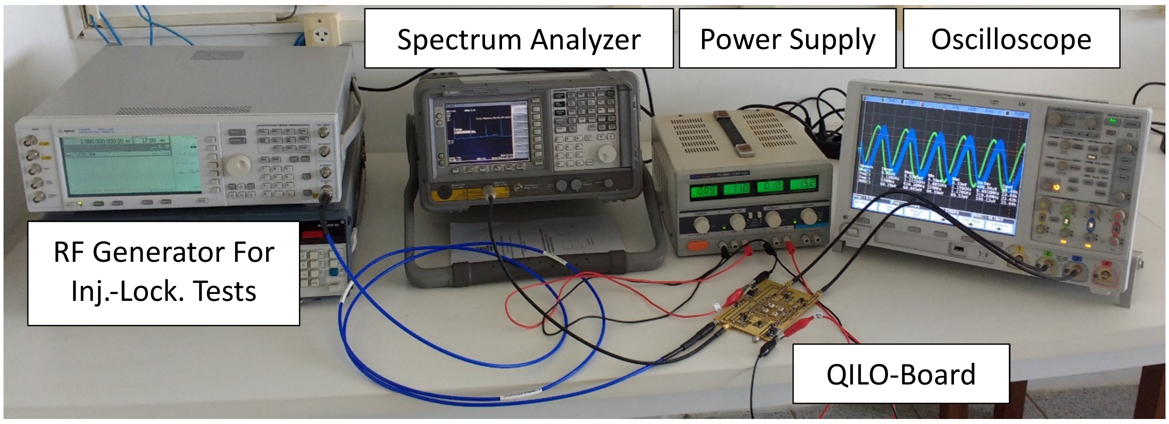


Figure 4.11: Test Setup for Oscillator Measurement

There was a much stronger influence of the oscilloscope on the operation of the circuit, especially if a channel was activated for measurement or inactive. This suggests that the output buffers do not isolate the circuit from the surroundings very well. Also, the transmission lines inside the oscillator, especially the ones for the quadrature connections between the two RC oscillator cores, were found to have a strong effect. Tests with injection locking were successful for frequencies close to the free oscillation frequency, but as this is far away from the desired frequency, no further tests were carried out.

4.3.2 Tombstone Setup of Oscillator

To overcome the problems with the wrong oscillation frequency, a setup using floating wiring and tombstone component placement was adapted. The objective was to minimize parasitic transmission line effects. A photograph of this setup is shown in Figure 4.12. In a first, very rough measurement when both oscillator cores of the quadrature oscillator were still not capacitor coupled (though there may have been parasitic inductive coupling) showed that both oscillators were oscillating. This was found by going close to each oscillator with the probe of the oscilloscope in high impedance mode and observing the RF amplitude. As soon as both cores were coupled, the oscillator did not oscillate anymore.

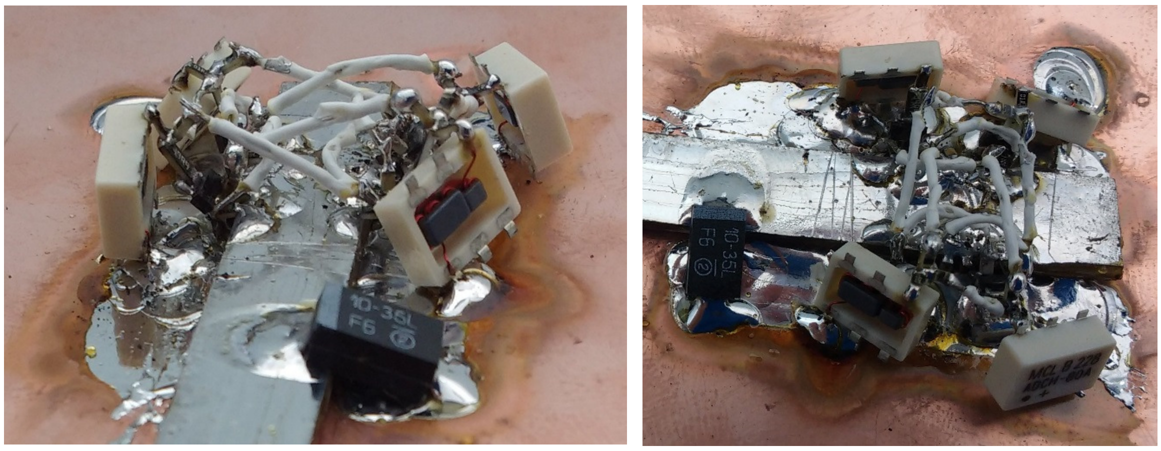


Figure 4.12: Tombstone Oscillator Setup

Though the circuit was assembled very carefully, it is possible that a ceramic capacitor in the center of the oscillator was broken during the wiring operation. Unluckily, there is no simple way to find out about this. If all components are still intact and connected, it is possible that the tolerances of the transistors are the reason why the circuit does not oscillate anymore. The tolerances of the other frequency defining components were analyzed using ADS and found to be tolerable, though there was a strong influence. Also, it is unknown how accurate the model of the used BFG520 transistor is. Finally, the used setup should have less transmission line effects, but there are some parasitic inductors.

4.4 Frontend

The Frontend includes the coupler, the splitters, the mixers, the filter and the LFAs. The basic function of the components was successfully verified.

4.4.1 Phase shift of Signals at the RF-Inputs of the Mixers

Unfortunately, it was not possible to measure the phase shift due to parasitic transmission lines between the power input of the Frontend and the RF inputs of the mixers directly, as the capacitive coupling between the oscilloscope channels in the high impedance mode was too strong. So a more refined measurement scheme was applied using 0° and 90° delay blocks and power splitters at the inputs of the circuit (shown in Figure 4.13), obtaining the in-phase and the quadrature component of the RF signal of each of the four channels. From this, each parasitic phase delay could be calculated. A block diagram of the used technique is shown in Figure 4.14. Two measurements of all four mixer outputs were made, (a) one time with 0° LO input (resulting in I_out), and (b) one time with 90° LO input (resulting in Q_out).

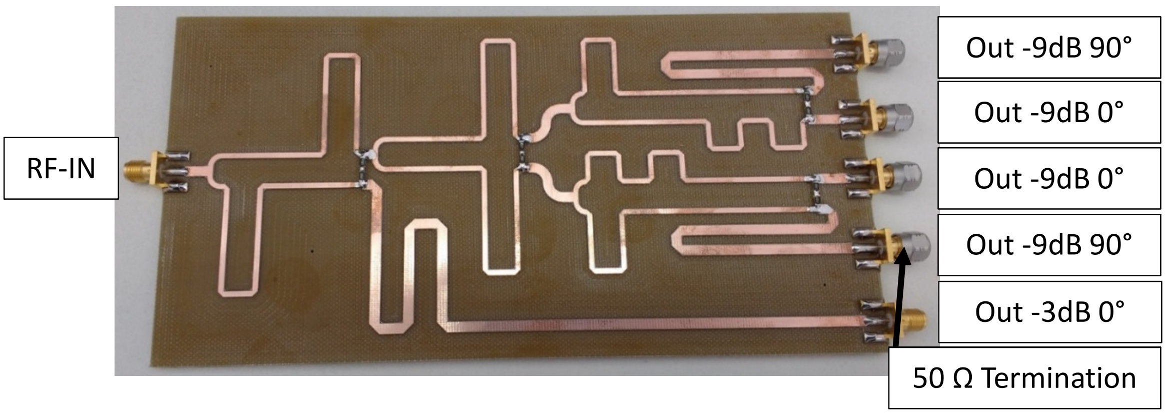


Figure 4.13: PCB Containing Wilkinson Power Splitters and Delay Lines

The results of the phase delay measurements were α ≈ 106°, β ≈ 130°, γ ≈ 74° and δ ≈ 96°, obtained using the ATAN2-function of Excel with each pair of I_out and Q_out values. These values are the total phase shift from the “reflected”-input for α and β, and the “incident”-input for γ and δ until the RF input of the mixers, respectively. For complete correction, about 28° have to be added to the α and β values to transform both values to the same input of the board.

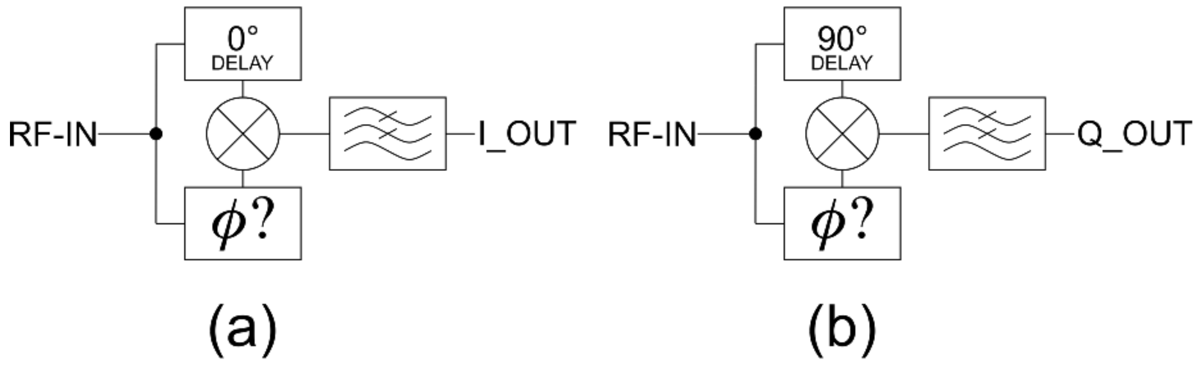


Figure 4.14: Technique to Measure the Parasitic Phase Angles at the Mixers' RF Inputs.

4.4.2 Performance for Input Signals Generated using Splitters and Delay Lines

The delay blocks from the former measurement were used to generate in-phase and quadrature signals from the input RF signal and feed the in-phase signals into two of the LO-inputs and the quadrature signals into the other two. A photograph of this measurement setup is shown in Figure 4.15.

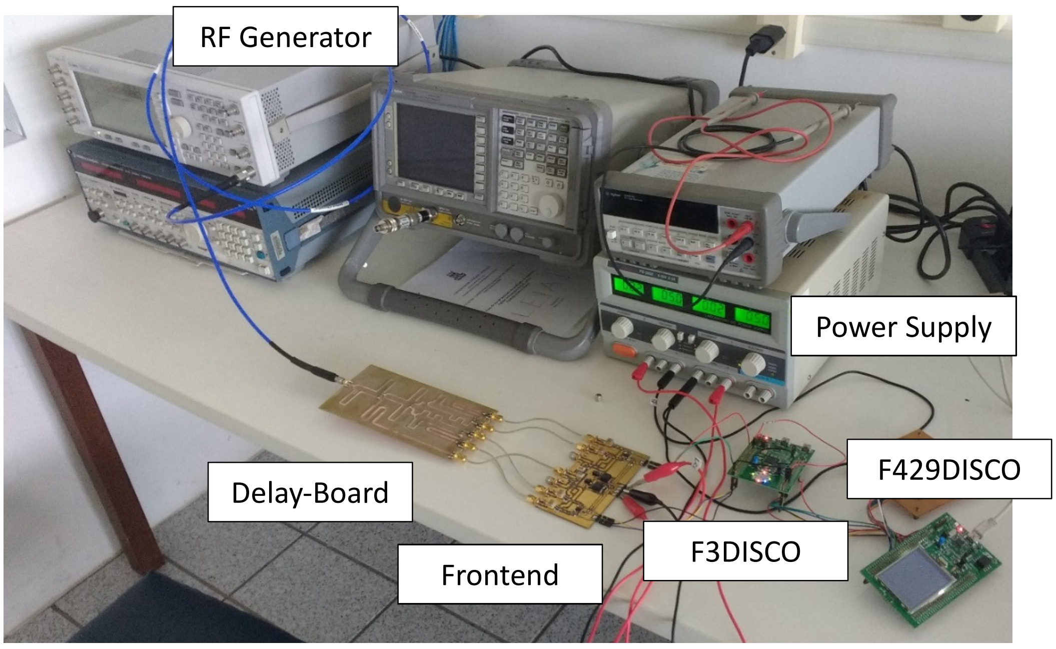


Figure 4.15: Test Setup for Frontend Measurements

Most of the blocks in the block diagram (Figure 4.16) are the same as in the system block diagram in the beginning of chapter 3. The directional coupler (A.), the splitters/attenuators

(B.), the mixers and filters (E.) and the LFA (F.) are all on the Frontend and the ADCs and the Control Block (G.) are based on the microcontroller boards. RF-IN denotes the point where the signal from the generator is fed in. The Wilkinson power splitters and the delay lines (C. and D.) are on the Delay-Board. The 50 Ω resistor is a screwed termination.

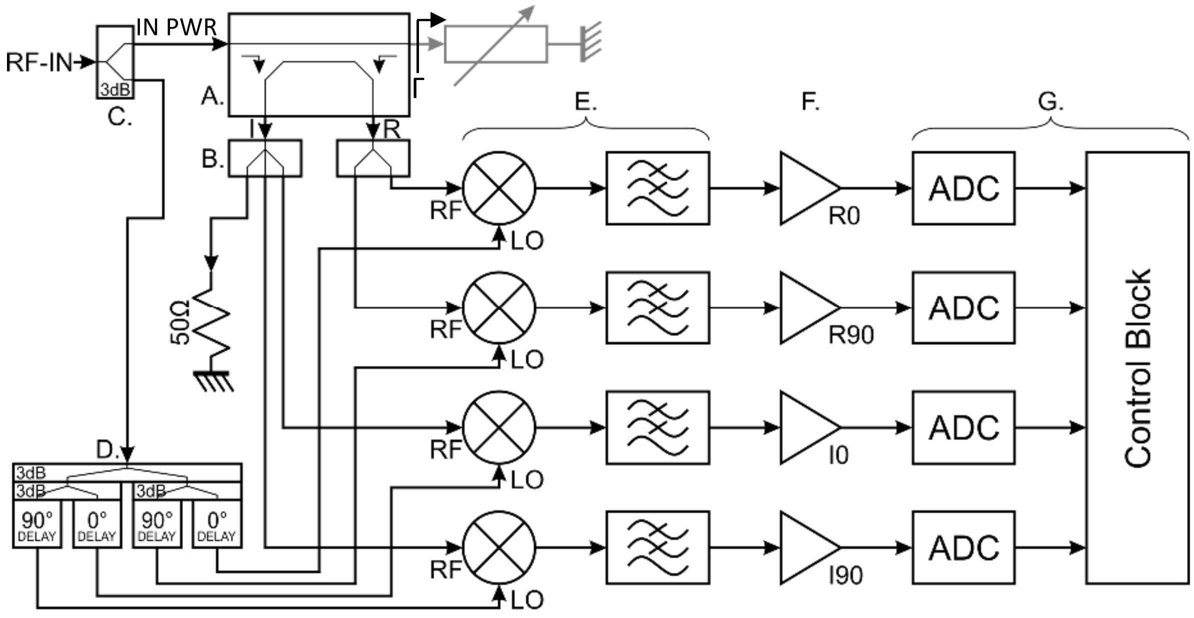


Figure 4.16: Block Diagram of Used Test Setup

The output signals of the measuring block were recorded for the following test conditions (shown in Table 4.2), with IN being the “incident” power (PWR) input of the coupler, and LO the local oscillator inputs of the mixers:

Table 4.2: Conditions for Test of Frontend with Delay-Board

Condition	**IN PWR**	**LO PWR**	**Phase IN-LO**	**Γ**
1	-/-	5 dBm	-/-	-/-
2	-/-	10 dBm	-/-	-/-
3	11 dBm	5 dBm	0°	Table 4.3
4	16 dBm	10 dBm	0°	Table 4.3
5	16 dBm	10 dBm	120°	Table 4.3

The core of the measurement setup is shown in Figure 4.17 for condition 1 and 2, in Figure 4.18 for condition 3 and 4 and Figure 4.19 for condition 5.

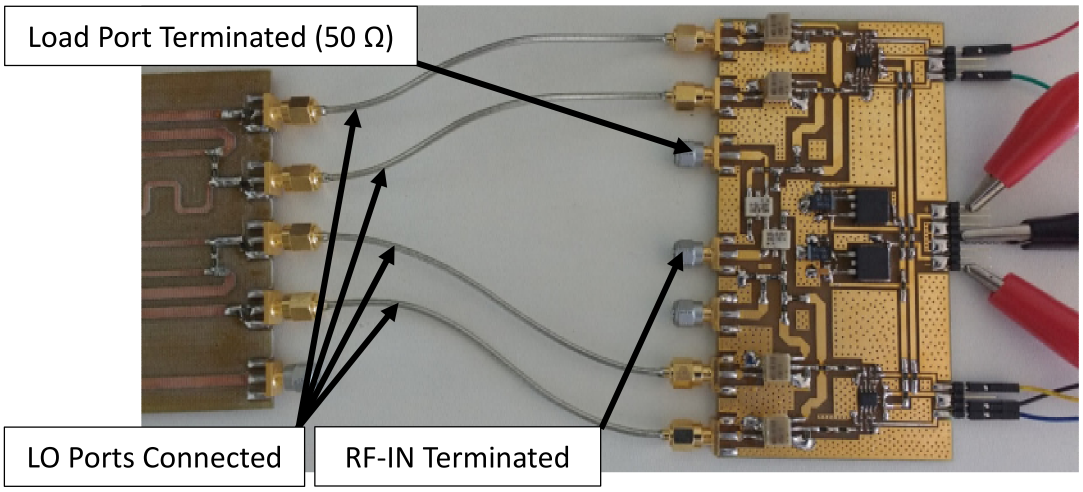


Figure 4.17: Core of Setup for Condition 1 and 2

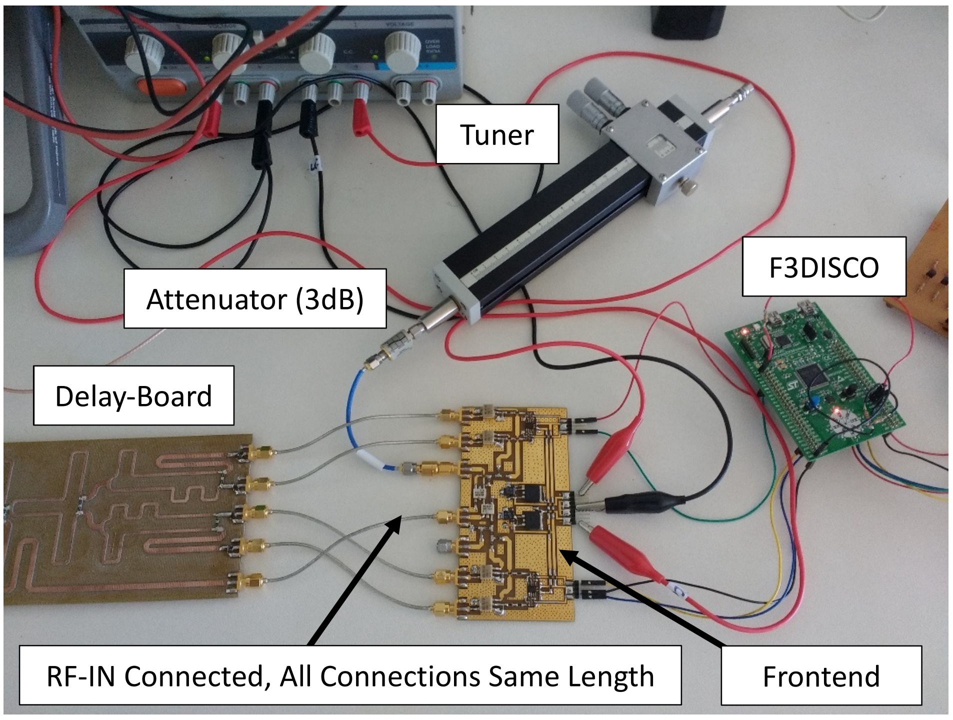


Figure 4.18: Core of Setup for Condition 3 and 4

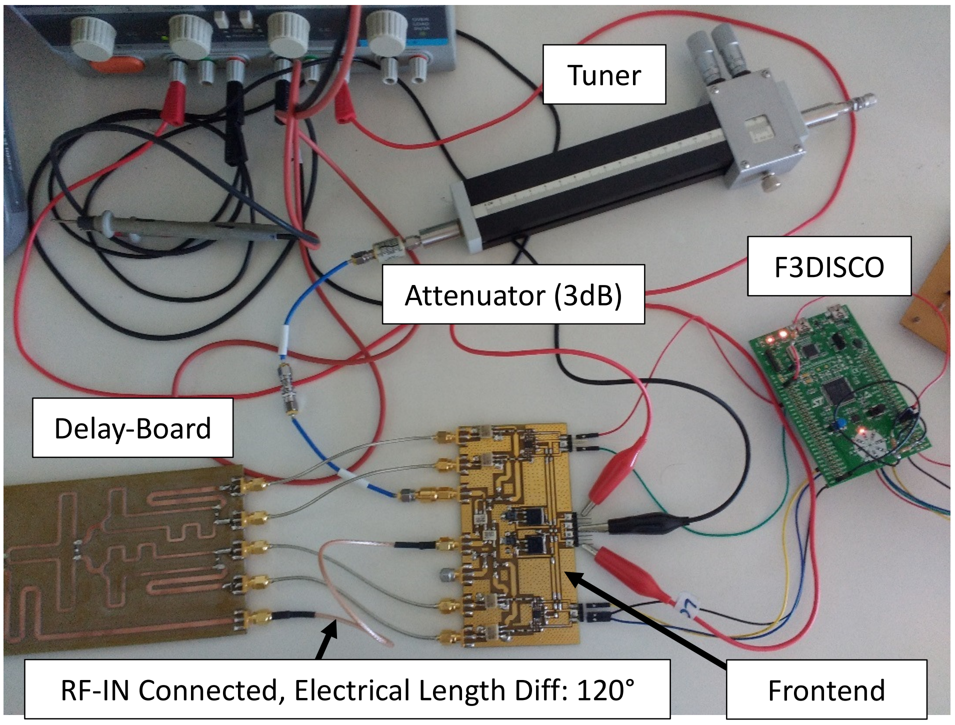


Figure 4.19: Core of Setup for Condition 5

The Reflection Coefficients were adjusted to a value close to the desired value from Table 4.3. with the help of the Coaxial Slide Screw Tuner and the VNA. The Tuner was always terminated with 50 Ω on its far end, and for reproducible handling, only the maximum limit of the reflection coefficient was used. For the magnitude 0.5 reflection coefficients, an additional 3dB attenuator was used between the Tuner and the measuring port of the Device under Test (DUT). Either one cable length was used or two cables were connected in series to obtain phases beyond the reach of the tuner in this frequency range.

Table 4.3: Reflection Coefficients Γ_i for Test of Frontend with Delay-Board

Condition	$\lvert\Gamma_i\rvert$	$\angle\Gamma_i$	**Condition**	$\lvert\Gamma_i\rvert$	$\angle\Gamma_i$
A	0	0°	**L**	0.5	-135°
B	1	-135°	**M**	0.5	-90°
C	1	-90°	**N**	0.5	-45°
D	1	-45°	**O**	0.5	0°
E	1	0°	**P**	0.5	45°
F	1	45°	**R**	0.5	90°
G	1	90°	**S**	0.5	135°
H	1	135°	**T**	0.5	180°
K	1	180°			

The raw measured values are displayed in Table 4.4. After a first correction of the effects of parasitic transmission lines in the setup according to the algorithm in section 4.4.3, the reflection coefficients and errors in Table 4.5 are obtained, also shown in Figure 4.20.

Table 4.4: Raw Measurement Data Obtained Using Delay-Board

$\lvert\Gamma_i\rvert$	$\angle\Gamma_i$ °	**P** dBm	**LO-IN Phase °**	v_a mV	v_b mV	v_c mV	v_d mV
0,928	-135	LO10, IN16	0	1645	1538	1560	1721
0,928	-135	LO10, IN16	120	1538	1692	1282	1462
0,928	-135	LO5, IN11	0	1566	1538	1526	1606
0,95	-89	LO10, IN16	0	1554	1389	1570	1726
0,95	-89	LO10, IN16	120	1682	1701	1286	1472
0,95	-89	LO5, IN11	0	1515	1458	1535	1609
0,93	-45	LO10, IN16	0	1325,5	1334,5	1583,5	1726,5
0,93	-45	LO10, IN16	120	1701	1586	1304	1469
0,93	-45	LO5, IN11	0	1405,5	1414	1545	1612
0	0	LO10, IN16	0	1501	1535	1564	1721
0	0	LO10, IN16	120	1508	1528	1301	1453
0	0	LO5, IN11	0	1490	1526	1531	1607,5
1	0	LO10, IN16	0	1270	1479	1582	1722
1	0	LO10, IN16	120	1595	1415	1321	1460
1	0	LO5, IN11	0	1365	1479	1540	1609
0,922	45	LO10, IN16	0	1398,5	1609	1559	1715
0,922	45	LO10, IN16	120	1451	1361	1318	1440
0,922	45	LO5, IN11	0	1429	1553	1530	1606
0,922	88	LO10, IN16	0	1508	1666	1553	1712
0,922	88	LO10, IN16	120	1365	1400	1309	1437,5
0,922	88	LO5, IN11	0	1490	1588	1525	1604
0,933	135	LO10, IN16	0	1607	1662	1551	1718
0,933	135	LO10, IN16	120	1366	1489	1301	1442,5
0,933	135	LO5, IN11	0	1538	1593	1521	1603
0,94	180	LO10, IN16	0	1647	1623	1547	1715
0,94	180	LO10, IN16	120	1418	1602	1286	1446
0,94	180	LO5, IN11	0	1564	1578	1521	1603
0,45	-135	LO10, IN16	0	1568	1529	1565	1725
0,45	-135	LO10, IN16	120	1529	1610	1294	1459
0,45	-135	LO5, IN11	0	1528	1528	1531,5	1609
0,46	-90	LO10, IN16	0	1522	1465	1570	1725
0,46	-90	LO10, IN16	120	1592	1611	1296,5	1464,5
0,46	-90	LO5, IN11	0	1499	1494	1535,5	1610
0,45	-43	LO10, IN16	0	1431	1455	1474	1725
0,45	-43	LO10, IN16	120	1595	1555	1304	1461
0,45	-43	LO5, IN11	0	1458	1482	1539	1611
0,455	0	LO10, IN16	0	1406	1503	1572	1722
0,455	0	LO10, IN16	120	1551	1485	1312	1458
0,455	0	LO5, IN11	0	1439	1501	1536	1609
0,46	45	LO10, IN16	0	1444	1566	1564	1718
0,46	45	LO10, IN16	120	1482	1448	1311	1448
0,46	45	LO5, IN11	0	1458,5	1538	1532,5	1607
0,46	90	LO10, IN16	0	1507	1607	1561	1718
0,46	90	LO10, IN16	120	1434	1465	1308	1446,5
0,46	90	LO5, IN11	0	1493	1562	1530	1608
0,464	135	LO10, IN16	0	1561	1606	1560	1720
0,464	135	LO10, IN16	120	1433	1508,5	1303	1448
0,464	135	LO5, IN11	0	1519	1564,5	1529	1607
0,45	179	LO10, IN16	0	1581	1581	1557,5	1720
0,45	179	LO10, IN16	120	1468,5	1570	1297	1451
0,45	179	LO5, IN11	0	1533	1555	1529	1608
X	X	LO10, IN0	X	1512	1542,5	1497,5	1534
X	X	LO5, IN0	X	1493,5	1531	1497	1516

The measured values Γ_L are compared to the ideal values Γ_i of reflection coefficient after correction, resulting in the absolute error, shown in equations (4.2) and (4.3):

$$\mathrm{Re}\{Err\} = \mathrm{Re}\{\Gamma_L\} - \mathrm{Re}\{\Gamma_i\} \tag{4.2}$$

$$\mathrm{Im}\{Err\} = \mathrm{Im}\{\Gamma_L\} - \mathrm{Im}\{\Gamma_i\} \tag{4.3}$$

Table 4.5: Corrected Measured Reflection Coefficients and Differences to Ideal Values

$\lvert\Gamma_i\rvert$	$\angle\Gamma_i$ °	P dBm	LO-IN Phase °	Re{Γ_L}	Im{Γ_L}	Re{Err}	Im{Err}	|Err|
0,928	-135	LO10, IN16	0	-0,6055	-0,5395	0,0507	0,1167	0,1273
0,928	-135	LO10, IN16	120	-0,6930	-0,6842	-0,0368	-0,0280	0,0462
0,928	-135	LO5, IN11	0	-0,6974	-0,3496	-0,0412	0,3066	0,3093
0,95	-89	LO10, IN16	0	-0,0232	-0,9283	-0,0398	0,0216	0,0453
0,95	-89	LO10, IN16	120	-0,1023	-1,1512	-0,1189	-0,2013	0,2338
0,95	-89	LO5, IN11	0	-0,0494	-0,8470	-0,0660	0,1029	0,1223
0,93	-45	LO10, IN16	0	0,8534	-0,5712	0,1958	0,0865	0,2141
0,93	-45	LO10, IN16	120	0,5692	-1,0461	-0,0884	-0,3885	0,3985
0,93	-45	LO5, IN11	0	0,8374	-0,7505	0,1798	-0,0929	0,2024
0	0	LO10, IN16	0	0,0346	0,0133	0,0346	0,0133	0,0371
0	0	LO10, IN16	120	0,0663	-0,1695	0,0663	-0,1695	0,1821
0	0	LO5, IN11	0	0,1280	0,0836	0,1280	0,0836	0,1529
1	0	LO10, IN16	0	1,1756	0,2111	0,1756	0,2111	0,2746
1	0	LO10, IN16	120	1,1790	-0,3538	0,1790	-0,3538	0,3965
1	0	LO5, IN11	0	1,3505	-0,0224	0,3505	-0,0224	0,3512
0,922	45	LO10, IN16	0	0,5554	0,7929	-0,0966	0,1409	0,1708
0,922	45	LO10, IN16	120	0,8809	0,5572	0,2290	-0,0948	0,2478
0,922	45	LO5, IN11	0	0,8676	0,7141	0,2156	0,0621	0,2244
0,922	88	LO10, IN16	0	-0,1237	0,8461	-0,1558	-0,0753	0,1731
0,922	88	LO10, IN16	120	0,1013	0,8900	0,0691	-0,0314	0,0759
0,922	88	LO5, IN11	0	0,1149	0,9480	0,0827	0,0266	0,0869
0,933	135	LO10, IN16	0	-0,6763	0,3640	-0,0166	-0,2957	0,2962
0,933	135	LO10, IN16	120	-0,5043	0,5753	0,1555	-0,0844	0,1769
0,933	135	LO5, IN11	0	-0,5701	0,6611	0,0896	0,0014	0,0896
0,94	180	LO10, IN16	0	-0,8028	-0,0961	0,1372	-0,0961	0,1676
0,94	180	LO10, IN16	120	-0,8508	-0,0455	0,0892	-0,0455	0,1001
0,94	180	LO5, IN11	0	-0,8445	0,1839	0,0955	0,1839	0,2072
0,45	-135	LO10, IN16	0	-0,2624	-0,2605	0,0558	0,0577	0,0802
0,45	-135	LO10, IN16	120	-0,3086	-0,4583	0,0096	-0,1401	0,1404
0,45	-135	LO5, IN11	0	-0,2763	-0,1388	0,0419	0,1794	0,1842
0,46	-90	LO10, IN16	0	0,0046	-0,4408	0,0046	0,0192	0,0197
0,46	-90	LO10, IN16	120	-0,0201	-0,6908	-0,0201	-0,2308	0,2317
0,46	-90	LO5, IN11	0	0,0500	-0,3473	0,0500	0,1127	0,1233
0,45	-43	LO10, IN16	0	0,3965	-0,0665	0,0674	0,2404	0,2497
0,45	-43	LO10, IN16	120	0,3161	-0,5903	-0,0130	-0,2834	0,2837
0,45	-43	LO5, IN11	0	0,4416	-0,2704	0,1124	0,0365	0,1182
0,455	0	LO10, IN16	0	0,5131	0,1052	0,0581	0,1052	0,1201
0,455	0	LO10, IN16	120	0,5436	-0,2647	0,0886	-0,2647	0,2791
0,455	0	LO5, IN11	0	0,6577	0,0237	0,2027	0,0237	0,2041
0,46	45	LO10, IN16	0	0,3091	0,3817	-0,0162	0,0565	0,0587
0,46	45	LO10, IN16	120	0,4443	0,1559	0,1190	-0,1693	0,2070
0,46	45	LO5, IN11	0	0,4955	0,3916	0,1703	0,0663	0,1827
0,46	90	LO10, IN16	0	-0,0533	0,4353	-0,0533	-0,0247	0,0588
0,46	90	LO10, IN16	120	0,0700	0,3402	0,0700	-0,1198	0,1388
0,46	90	LO5, IN11	0	0,0899	0,5273	0,0899	0,0673	0,1123
0,464	135	LO10, IN16	0	-0,3371	0,2205	-0,0090	-0,1076	0,1079
0,464	135	LO10, IN16	120	-0,2213	0,2105	0,1068	-0,1176	0,1589
0,464	135	LO5, IN11	0	-0,2277	0,4016	0,1004	0,0735	0,1244
0,45	179	LO10, IN16	0	-0,4027	-0,0218	0,0472	-0,0297	0,0558
0,45	179	LO10, IN16	120	-0,3971	-0,1282	0,0529	-0,1361	0,1460
0,45	179	LO5, IN11	0	-0,3778	0,1656	0,0721	0,1578	0,1735

The magnitude of the absolute Error |Err| is calculated using equation (4.4)

$$|Err| = \sqrt{\mathrm{Re}\{Err\}^2 + \mathrm{Im}\{Err\}^2} \tag{4.4}$$

The same equations (4.2), (4.3) and (4.4) apply to the simulation results in Table 3.5.

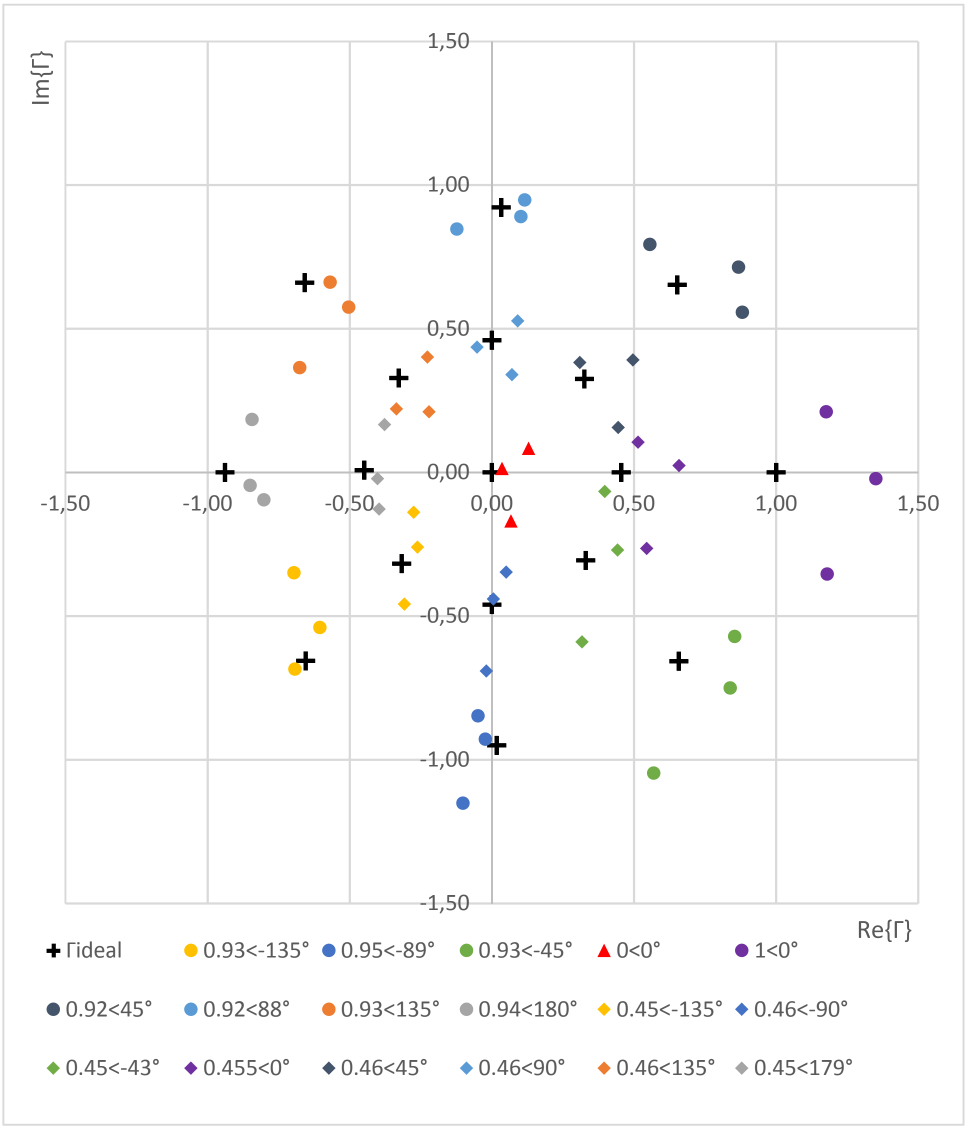


Figure 4.20: Diagram of Measured ReflectionCoefficients

It remains to be said that the measurements with the ADCs of the STM32F303 microcontroller contain noise of about 5 mV_{rms}, what makes it difficult to obtain the values with high accuracy. It was tried to reproduce this noise in simulations (including quantization), but without success. Probably the noise comes from interference of the digital signals of the microcontrollers with the measured analog signals.

Even so it was possible to demonstrate a clear correlation between the measured values and the real reflection coefficient attached to the measurement port after mathematical correction with a system that was not optimized to the last degree and exhibited many parasitic effects. Average absolute errors of 0.146 and maximum errors of 0.28 were reached for the points inside the Smith chart (without the outer limit). In Table 4.6 is shown a comparison of these results with other scientific works that also use discrete setups.

Table 4.6: Comparison of Errors with References

Reference	**Err. max.**	**Err. avg.**
[58] (without outer limit of smith chart)	25%	-/-
[59] (without outer limit of smith chart)	37%	-/-
This work, ***simulation*** 11 dBm (complete area)	4,8%	2,4%
This work, ***simulation*** 30 dBm (complete area)	11,6%	4,4%
This work, measurements (without outer limit of smith chart)	28%	14,6%

The errors in Table 4.6 are given as percent of full scale. As the maximum value of a reflection coefficient is one, these values can be obtained from the absolute errors through multiplication by 100. Reference [58] does not give a number for the error in the text, so it was derived from a Smith chart (like Figure 4.20).

4.4.3 Correction Algorithm

It was found that the mixers have an RF dependent DC offset of less than 10mV, but it is enough to make an offset correction necessary. Together with the offset correction, a scaling correction was provided, using the following equations (4.5) to (4.8). This way the losses in the directional coupler can be taken into account. Here, a, b, c and d are the corrected measured voltages, v_x is the voltage measured by the ADC$_X$, o_x is the offset voltage and s_x is the scaling factor.

$$a = (v_a - o_a) \cdot s_a \tag{4.5}$$

$$b = (v_b - o_b) \cdot s_b \tag{4.6}$$

$$c = (v_c - o_c) \cdot s_c \tag{4.7}$$

$$d = (v_d - o_d) \cdot s_d \tag{4.8}$$

The phase shifts of the transmission lines between the output of the coupler and the RF-inputs of the mixers were corrected using a coordinate system transformation (non-rectangular to Cartesian) as shown in equations (4.9) to (4.12). Here, α, β, γ and δ are the angles between X-Axis of the Cartesian coordinate system and the A-, B-, C- and D-Axis of the non-rectangular coordinate system, respectively. The values a, b, c, d are the intercepts of the measured point on the A-, B-, C- and D-Axis, respectively, shown in Figure 4.21.

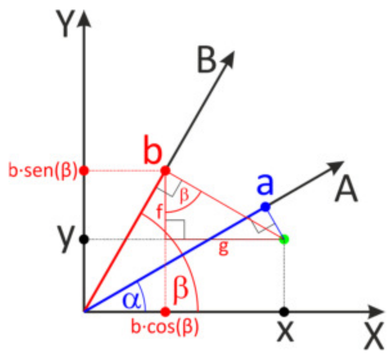


Figure 4.21: Coordinate System Transformation (just shown for Vr, but Vi is the same)

$$p_r = \mathrm{Re}(V_r) = \frac{b \cdot \sin\alpha - a \cdot \sin\beta}{\sin(\alpha - \beta)} \tag{4.9}$$

$$q_r = \mathrm{Im}(V_r) = \frac{a \cdot \cos\beta - b \cdot \cos\alpha}{\sin(\alpha - \beta)} \tag{4.10}$$

$$p_i = \mathrm{Re}(V_i) = \frac{d \cdot \sin\gamma - c \cdot \sin\delta}{\sin(\gamma - \delta)} \tag{4.11}$$

$$q_i = \mathrm{Im}(V_i) = \frac{c \cdot \cos\delta - d \cdot \cos\gamma}{\sin(\gamma - \delta)} \tag{4.12}$$

The reflection coefficient as seen at the input of the directional coupler is then simply the division of the complex voltages V_r by V_i as shown in equations (4.13) to (4.16):

$$\Gamma_t = \frac{V_r}{V_i} = \frac{p_r + \mathrm{j} \cdot q_r}{p_i + \mathrm{j} \cdot q_i} \tag{4.13}$$

$$den = p_i^2 + q_i^2 \tag{4.14}$$

$$\mathrm{Re}\{\Gamma_t\} = \frac{p_r \cdot p_i + q_r \cdot q_i}{den} \tag{4.15}$$

$$\mathrm{Im}\{\Gamma_t\} = \frac{q_r \cdot p_i - p_r \cdot q_i}{den} \tag{4.16}$$

Furthermore, transmission line effects of the connection between “reflected” input and coupler had to be take into account. The correction equation was derived from the impedance transformation equations of the transmission line. Unluckily, this transmission line consists of several segments with different microstrip widths and thus different impedances, what makes the equation rather large. The solution shown in equations (4.17) to (4.27) was found with the help of MATLAB symbolic toolbox as well as with numeric techniques. The values E, F, G and H are proprietary correction values that depend on the impedances and phase shifts of the single segments of the transmission line. As mentioned, they are results of a large equation and are best calculated using the MATLAB script shown in Appendix D.

$$E = 1.8277; \tag{4.17}$$

$$F = 6.99413; \tag{4.18}$$

$$G = 31.6187; \tag{4.19}$$

$$H = 55.1533; \tag{4.20}$$

$$K = F \cdot \mathrm{Re}\{\Gamma_t\} + E \cdot \mathrm{Im}\{\Gamma_t\} - G; \tag{4.21}$$

$$L = E \cdot \mathrm{Re}\{\Gamma_t\} - F \cdot \mathrm{Im}\{\Gamma_t\} + H; \tag{4.22}$$

$$M = H\cdot\mathrm{Re}\{\mathit{\Gamma t}\} - G\cdot\mathrm{Im}\{\mathit{\Gamma t}\} + E; \tag{4.23}$$

$$N = G\cdot\mathrm{Re}\{\mathit{\Gamma t}\} + H\cdot\mathrm{Im}\{\mathit{\Gamma t}\} - F; \tag{4.24}$$

$$P = L^2 + K^2; \tag{4.25}$$

$$\mathrm{Re}\{\Gamma_L\} = \frac{L\cdot M + N\cdot K}{P} \tag{4.26}$$

$$\mathrm{Im}\{\Gamma_L\} = \frac{L\cdot N - M\cdot K}{P} \tag{4.27}$$

After these corrections, the calculated load reflection coefficient was relatively close to the one measured by the VNA (absolute error of less than 0.4, average of less than 0.2, discussed in section 4.4.2). These correction equations have also been implemented on the microcontroller. It is important to note that most of these corrections will not be necessary when integrating the system on an IC, as this will remove most of the transmission line effects.

4.5 Measurements of Complete System

Because of the problems described in sections 4.3.1 and 4.3.2, it was not possible to do measurements with the oscillator as part of the system. So, the tests with the Delay-Board in section 4.4.2 are the only measurements that allow deductions about the operation of the whole system. As not only the reflection coefficient was changed but also the power and the phase of the input signals, these deductions are valid for the expected behavior of a working oscillator. A clear correlation was shown between the mathematically corrected measurement values and the values obtained using the VNA, proving the basic function of the system and opening room for further optimization.

The next chapter draws conclusions for this work and indicates perspectives for future research that can be derived from it.

5 Conclusion

This Dissertation explored aspects of all blocks of an automatic impedance matching system, with a focus on the measuring block.

First, a discrete 30 dBm power amplifier was designed and showed reduced but still reasonable performance in the measurements compared to the computer simulations. An output power of almost 28 dBm was achieved.

Also, the control block and the variable impedance matching network of an automatic impedance matching system were reviewed. The matching circles theory from [3] was introduced to the matching state control algorithm as well as to a three variable component π-network as proposed in [56]. Especially, the latter was analyzed numerically based on matching circles.

Most importantly, a new load reflection coefficient measuring block for use in an automatic impedance matching system was introduced and its operation was demonstrated using computer simulations and measurements.

The simulations demonstrated a good performance for such a relatively simple system, showcasing a maximum absolute error of 0.116 for extreme values of the reflection coefficient and an average absolute error of 0.034. As can be expected in a PCB setup with many parasitics, the measurements did not show a performance as good as the simulations. But, after mathematically correcting some of the parasitics, the maximum absolute error was less than 0.4 and the average absolute error less than 0.2, with the least good values at extremely high impedances. This still shows a clear correlation between the reflection coefficients measured by the designed block and the ones measured by the VNA.

Unlike the simulations, the measurements did not include the Quadrature Injection Locking Oscillator but used a replacement because it was not possible to obtain a regular operation of the oscillator in the desired band, neither in a regular PCB setup nor in a tombstone setup that was built hoping to reduce parasitics. The reason for this are most probably the parasitics (such as transmission line effects), and the tolerances of the components (especially the transistors). These problems with the oscillator should be easier to deal with in an integrated circuit setup. After all, this kind of oscillators is especially suited for integration and its function was already demonstrated [42].

Altogether, for future research it is recommended to produce a version of the presented measuring block as an integrated circuit, as it is expected that it will perform better in this case because of less transmission line effects and tolerance problems in the quadrature injection locking oscillator and in the parts of the circuit that are currently on the Frontend-PCB.

Also, a more thoroughly analysis of the application of matching circles to the control algorithm can be an interesting research topic, as well as the complete design of a three variable component π-network for a given error and number of matching states for a minimum number of capacitors. Finally, various impedance matching network topologies can be analyzed based on the matching circles theory.

References

[1] J. Toftgard and S. Hornsteth, "Effects on portable antennas of the presence of a person," IEEE Transaction on Antennas and Propagation, Vol. 41, No. 6, pp. 739-746, June 1993.

[2] R. A. Sadeghzadeh, M. S. Abrishamian and N. J. McEwan, "Effect of user head on mobile telephone handset antenna impedance and head internal fields distribution," Procedures 24th European Microwave Conference, pp. 603-606, September 1994.

[3] K. Brito and R. Lima, “Impedance network for an automatic impedance matching system,” 2007 Asia-Pacific Microwave Conference, pp. 1-4, December 2007.

[4] A. Scuderil, F. Carrara, A. Castorina, and G. Palmisano “A high performance RF power amplifier with protection against load mismatches,” lEEE MTI-S Digest, pp. 699-702, June 2003.

[5] J. Madic, P. Bretchko, Shuyun Zhang, R. Shumovich, R. McMorrow, "Accurate power control technique for handset PA modules with integrated directional couplers," 2003 IEEE MTT-S International Microwave Symposium Digest, Print ISBN: 0-7803-7695-1, June 2003.

[6] C. Sánchez-Pérez, D. Sardin, M. Roberg, J. de Mingo and Z. Popović, “Tunable outphasing for power amplifier efficiency improvement under load mismatch,” Microwave Symposium Digest (MTT), IEEE MTT-S International, pp. 1-3, June 2012.

[7] Hamhee Jeon, Yunseo Park, Yan-Yu Huang, Jihwan Kim, Kun-Seok Lee, Chang-Ho Lee and J. Stevenson Kenney, "A Triple-Mode Balanced Linear CMOS Power Amplifier Using a Switched-Quadrature Coupler," IEEE Journal of Solid-State Circuits, Vol. 47, No. 9, pp. 2019-2032, September 2012.

[8] Won-Gyu Lim, Seo-Young Park, Wang-Ik Son, Moon-Que Lee, and Jong-Won Yu, "RFID Reader Front-End Having Robust Tx Leakage Canceller for Load Variation," IEEE Transactions on Microwave Theory and Techniques, Vol. 57, No. 5, pp. 1348-1355, May 2009.

[9] Han Lim Lee, Won-Gyu Lim, Kyoung-Sub Oh, and Jong-Won Yu, "24 GHz Balanced Doppler Radar Front-End With Tx Leakage Canceller for Antenna Impedance Variation and Mutual Coupling," IEEE Transactions on Antennas and Propagation, Vol. 59, No. 12, pp. 4497-4504, December 2011.

[10] Younsuk Kim, Jeonghu Han, Dongho Lee, Changkun Park, and Songcheol Hong "A CMOS Power Amplifier for a UHF RFID Reader," Proceedings of Asia-Pacific Microwave Conference 2006.

[11] I. Aoki, S. Kee, D. Rutledge, and A. Hajimiri, "Distributed active transformer-a new power-combining and impedance-transformation technique," IEEE Trans. Microwave Theory Tech., vol.50, pp. 316-331, January 2002.

[12] I. Aoki, "Fully integrated CMOS power amplifier design using the distributed active-transformer architecture," IEEE J. Solid-State Circuits, Vol. 37, No. 3, pp. 371-383, March 2002.

[13] Robson N. De Lima, B. Huyart, E. Bergeault and L. Jallet, "MMIC impedance matching system," IEEE Electronics Letters, Vol. 36, No 16, pp. 1393-1394, August 2000.

[14] Raisa G. Pesel, Sara S. Attar, Raafat R. Mansour, "MEMS-based switched-capacitor banks for impedance matching networks," 2015 European Microwave Conference (EuMC), Electronic ISBN: 978-2-8748-7039-2, September 2015.

[15] Maha Added, Noureddine Boulejfen, "Variable impedance matching network based on varactor diodes," 2015 IEEE 15th Mediterranean Microwave Symposium (MMS), Electronic ISBN: 978-1-4673-7602-0, November 2015.

[16] B. Xiong, K. Hofmann, "Binary search algorithm for adaptive impedance matching network," Electronics Letters Vol.: 52, Issue: 9, PP.: 714-716, April 2016.

[17] L.-Y. Chen, R. Forse, D. Chase, and R. A. York, “Analog tunable matching network using integrated thin-film BST capacitors,” IEEE MTT-S Int. Microw. Symp. Dig., Vol. 1, pp. 261–264, June 2004.

[18] Les Besser, Rowan Gilmore, "Practical RF Circuit Design for Modern Wireless Systems, Volume I. Passive Circuits and Systems," Chapter 2.17, pp. 59-63, ISBN 1-58053-521-6, Artech House, 2003.

[19] G. Harman, "IF Return-Loss Measurements in Microwave Radio Systems," IEEE Transactions on Instrumentation and Measurement, Vol: 18, Issue: 4, pp. 346-352, December 1969.

[20] J. Moritz, D. Lauder and Y. Sun, “Measurement of HF antenna impedances,” IEE Colloquium on Frequency Selection and Management Techniques for HF Communications, pp. 14/1-14/7, March 1999.

[21] Kamil Staszek, Slawomir Gruszczynski, Krzysztof Wincza, "Measurement Accuracy Enhancement in Six-Port Reflectometers," IEEE Microwave and Wireless Components Letters, Vol: 25, Issue: 8, PP.: 553-555, August 2015.

[22] Philips Semiconductors, "Circulators and Isolators, unique passive devices," Application Note AN98035, March 1998.

[23] Geert Carchon and Bart Nauwelaers, "Power and Noise Limitations of Active Circulators," IEEE Transactions on Microwave Theory and Techniques, Vol. 48, No. 2, pp. 316-319, February 2000.

[24] http://www.minicircuits.com/products/Couplers.shtml

[25] Oliver, B.M., “Directional Electromagnetic Couplers,” Proceedings of the IRE, Vol. 42, Issue 11, pp. 1686-1692, November 1954.

[26] Song Wen, Qingyuan Wang, Tao Wu, Yicheng Tan, "Design of a compact 3dB Ka-band directional coupler," International Workshop on Microwave and Millimeter Wave Circuits and System Technology, pp. 1-4, April 2012.

[27] J. Biernacki, D. Czarkowski, “RF transformer as a directional coupler with arbitrary load,” ISCAS '03 Proceedings of the 2003 International Symposium on Circuits and Systems, Vol. 1, pp. I-97 – I-100, May 2003.

[28] A. Cidronali, G. Collodi, M. R. Deshpande, N. El-Zein, H. Goronkin, G. Manes, V. Nair, C. Toccafondi, “A MMIC lumped element directional coupler with arbitrary characteristic impedance and its application,” 30th European Microwave Conference, pp. 1-4, October 2000.

[29] Mina Wahib, A. P. Freundorfer, "A miniaturized lumped element directional coupler with parasitics compensation," IEEE International Symposium on Circuits and Systems 2016, May 2016.

[30] Jonghun Jung, Geunyong Lee, Jong-In Song, "A Lumped-Element Directional Coupler With High Isolation for Mobile RFID Reader," IEEE Microwave and Wireless Components Letters, Vol: 24, Issue: 6, pp.: 382-384, June 2014.

[31] D. Kajfez, “Scattering matrix of a directional coupler with ideal transformers,” Microwaves, Antennas and Propagation, IEE Proceedings Vol. 146, Issue: 4, pp. 295-297 August 1999.

[32] F. Meng, A. van Bezooijen and R. Mahmoudi, “A mismatch detector for adaptive antenna impedance matching,” 36th European Microwave Conference, pp. 1457-1460, September 2006.

[33] S. Kousai, K. Onizuka, T. Yamaguchi, Y. Kuriyama and M. Nagaoka , “Polar antenna impedance detection and tuning for efficiency improvement in a 3G/4G CMOS power amplifier,” ISSCC Dig. Tech Papers, pp. 58-60, February 2014.

[34] B. Razavi, “A study of injection locking and pulling in oscillators,” IEEE Journal of Solid-State Circuits, Vol. 39, Issue: 9, pp. 1415-1424, September 2004.

[35] http://www.st.com/resource/en/datasheet/pd84002.pdf

[36] https://www.minicircuits.com/pdfs/ADC-15-4.pdf

[37] http://www.microwaves101.com/encyclopedias/attenuators

[38] http://www.nxp.com/documents/data_sheet/BGA6489.pdf

[39] http://datasheets.avx.com/X7RDielectric.pdf

[40] M. Raj, A. Emami-Neyestanak, "A wideband injection locking scheme and quadrature phase generation in 65nm CMOS," IEEE Radio Frequency Integrated Circuits Symposium (RFIC), pp. 261-264, June 2013.

[41] Donald O. Pederson, Kartikeya Mayaram, "Analog Integrated Circuits for Communication. Principles, Simulation and Design," Chapter 1.4.1, pp. 5-6, ISBN 978-0-387-68029-3, Springer 2008.

[42] J. Casaleiro, L. Oliveira, and I. Filanovsky, “Low-power and low-area cmos quadrature RC oscillator with capacitive coupling,” IEEE International Symposium on Circuits and Systems (ISCAS), 2012, pp. 1488–1491, May 2012.

[43] Donald O. Pederson, Kartikeya Mayaram, "Analog Integrated Circuits for Communication. Principles, Simulation and Design," Chapter 13.6, pp.: 434-436, ISBN 978-0-387-68029-3, Springer 2008.

[44] https://www.minicircuits.com/pdfs/ASK-1-KK81.pdf

[45] F. Tillman, N. Troedsson, H. Sjland, "A 1.2 volt 1.8GHz CMOS quadrature front-end," Symposium on VLSI Circuits, Digest of Technical Papers, pp. 362-365, June 2004.

[46] http://www.onsemi.com/pub_link/Collateral/MBD101-D.PDF

[47] Andrew Leven, "Telecommunication Circuits and Technology," Chapter 3.2, p. 97, ISBN 0 7506 5045 1, Butterworth-Heinemann 2000.

[48] http://www.analog.com/media/en/technical-documentation/data-sheets/AD8615_8616_8618.pdf

[49] http://datasheets.avx.com/Accu-P.pdf

[50] http://datasheets.avx.com/C0GNP0-Dielectric.pdf

[51] http://www.st.com/resource/en/datasheet/stm32f429zi.pdf

[52] http://www.st.com/resource/en/data_brief/32f429idiscovery.pdf

[53] http://www.st.com/resource/en/datasheet/stm32f303vc.pdf

[54] http://www.st.com/resource/en/data_brief/stm32f3discovery.pdf

[55] http://www.emblocks.org/

[56] Jesús de Mingo, Alfredo Crespo, Antonio Valdovinos, "Input impedance antenna automatic matching system," The 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, Print ISBN: 0-7803-7589-0, September 2002.

[57] Rowan Gilmore, Les Besser, "Practical RF Circuit Design for Modern Wireless Systems, Volume II. Active Circuits and Systems," Chapter 5.1.1, p. 222, ISBN 1-58053-522-4, Artech House 2003.

[58] Dongjiang Qiao, Yu Zhao, Tsaipi Hung, D. Kimball, Mingyuan Li, P. Asbeck, D. Choi, D. Kelly, "Antenna impedance mismatch measurement and correction for adaptive CDMA transceivers, " 2005 IEEE MTT-S International Microwave Symposium Digest, Print ISBN: 0-7803-8845-3, June 2005.

[59] John Buckley, Kevin G. McCarthy, Brendan O'Flynn, Cian O'Mathuna, "The detuning effects of a wrist-worn antenna and design of a custom antenna measurement system," 2010 European Wireless Technology Conference, September 2010.

A. Appendix – Final Program F303

A.1 MAIN.C

A.1.1 Includes and Variables

```
#include "stm32f30x.h"
#include "core_cm4.h"
#include "stm32f30x_conf.h"
#include "arm_math.h"
#include "stm32f30x_it.h"
#include "stm32f3_discovery.h"
#include "firCoeffs.h"

#define BLOCK_SIZE               MAX_ADC_VALS

__IO float32_t f32SPI3txValues[4];
__IO uint8_t uiSPI3rxValues[16];

// FIR-Filter instances
arm_fir_instance_f32 fir1S, fir2S, fir3S, fir4S;
// Declare state buffers of size (numTaps + blockSize - 1)
static float32_t fir1StateF32[BLOCK_SIZE + NUM_TAPS - 1];
static float32_t fir2StateF32[BLOCK_SIZE + NUM_TAPS - 1];
static float32_t fir3StateF32[BLOCK_SIZE + NUM_TAPS - 1];
static float32_t fir4StateF32[BLOCK_SIZE + NUM_TAPS - 1];
// assign denominator for division in average function (also normalization
to Volts)
static float32_t averageDenom = ((float32_t)MAX_ADC_VALS)*4095.0f/3.0f;

static void Timer_Config(void);
static void ADC_Config(void);
static void SPI_Config(void);
static void FilterAndAverageADCvals(float32_t* ADCoutVals);
```

A.1.2 Main Function

```
int main(void)
{
    ADC12ready = false;
    ADC34ready = false;
    SPI3ready  = false;

    isOddRun12 = true;
    isOddRun34 = true;
    adcRecordIndex12 = 0;
    adcRecordIndex34 = 0;

    adc1write = &adc1array1[0];
    adc1read  = &adc1array2[0];
    adc2write = &adc2array1[0];
    adc2read  = &adc2array2[0];
    adc3write = &adc3array1[0];
    adc3read  = &adc3array2[0];
    adc4write = &adc4array1[0];
    adc4read  = &adc4array2[0];

    // Call FIR init function to initialize the instance structure.
    arm_fir_init_f32(&fir1S, NUM_TAPS, (float32_t *)&firCoeffs32[0],
&fir1StateF32[0], (uint32_t)BLOCK_SIZE);
    arm_fir_init_f32(&fir2S, NUM_TAPS, (float32_t *)&firCoeffs32[0],
&fir2StateF32[0], (uint32_t)BLOCK_SIZE);
    arm_fir_init_f32(&fir3S, NUM_TAPS, (float32_t *)&firCoeffs32[0],
&fir3StateF32[0], (uint32_t)BLOCK_SIZE);
      arm_fir_init_f32(&fir4S, NUM_TAPS, (float32_t *)&firCoeffs32[0],
&fir4StateF32[0], (uint32_t)BLOCK_SIZE);

      SystemCoreClockUpdate();

    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_4);

    STM_EVAL_LEDInit(LED3);
    STM_EVAL_LEDInit(LED6);
    STM_EVAL_LEDInit(LED7);
    STM_EVAL_LEDInit(LED9);
    STM_EVAL_LEDInit(LED10);
    STM_EVAL_LEDOff(LED3);
    STM_EVAL_LEDOff(LED6);
    STM_EVAL_LEDOff(LED7);
    STM_EVAL_LEDOff(LED9);
    STM_EVAL_LEDOff(LED10);

    // FPU_Config();
    ADC_Config();
    SPI_Config();
    Timer_Config();

    while(1)
    {
        if (ADC12ready && ADC34ready)
        {
            ADC12ready = false;
            ADC34ready = false;

            STM_EVAL_LEDToggle(LED10);
            FilterAndAverageADCvals(&f32SPI3txValues[0]);
            /*
            f32SPI3txValues[0] = adc1read[MAX_ADC_VALS-1];
            f32SPI3txValues[1] = adc2read[MAX_ADC_VALS-1];
            f32SPI3txValues[2] = adc3read[MAX_ADC_VALS-1];
            f32SPI3txValues[3] = adc4read[MAX_ADC_VALS-1];
            */
            // 1. Enable DMA Rx buffer in the RXDMAEN bit in the SPI_CR2
register, if DMA Rx is used.
            // 2. Enable DMA streams for Tx and Rx in DMA registers, if the
streams are used.
            DMA_Cmd(DMA2_Channel2, ENABLE);
            // 3. Enable DMA Tx buffer in the TXDMAEN bit in the SPI_CR2
register, if DMA Tx is used.
            SPI_I2S_DMACmd(SPI3, SPI_I2S_DMAReq_Tx, ENABLE);
            // 4. Enable the SPI by setting the SPE bit.
            GPIO_ResetBits(GPIOD, GPIO_Pin_0);
            SPI_Cmd(SPI3, ENABLE);
        }

        if (SPI3ready)
        {
            SPI3ready = false;
            // 2. Wait until FTLVL[1:0] = 00 (no more data to transmit).
            while(SPI_GetTransmissionFIFOStatus(SPI3) != 0);
            // 3. Wait until BSY=0 (the last data frame is processed).
            while(SPI_I2S_GetFlagStatus(SPI3, SPI_I2S_FLAG_BSY) != RESET);
            // 4. Disable the SPI (SPE=0).
            GPIO_SetBits(GPIOD, GPIO_Pin_0);
            SPI_Cmd(SPI3, DISABLE);
            // 5. Read data until FRLVL[1:0] = 00 (read all the received
data)
            if (SPI_GetReceptionFIFOStatus(SPI3) != 0)
                uiSPI3rxValues[0] = SPI_ReceiveData8(SPI3);
            // 6. Disable DMA Tx and Rx buffers by clearing the TXDMAEN and
RXDMAEN bits in the
            // SPI_CR2 register, if DMA Tx and/or DMA Rx are used.
            SPI_I2S_DMACmd(SPI3, SPI_I2S_DMAReq_Tx, DISABLE);

            //STM_EVAL_LEDOff(LED10);
        }
    }
}
```

A.1.3 Filtering and Averaging Function

```
static void FilterAndAverageADCvals(float32_t* ADCoutVals)
{
    uint16_t runi;
    float32_t adc1filtered[MAX_ADC_VALS];
    float32_t adc2filtered[MAX_ADC_VALS];
    float32_t adc3filtered[MAX_ADC_VALS];
    float32_t adc4filtered[MAX_ADC_VALS];

    // Call the FIR process function
    arm_fir_f32(&fir1S, adc1read, &adc1filtered[0], (uint32_t)BLOCK_SIZE);
    arm_fir_f32(&fir2S, adc2read, &adc2filtered[0], (uint32_t)BLOCK_SIZE);
    arm_fir_f32(&fir3S, adc3read, &adc3filtered[0], (uint32_t)BLOCK_SIZE);
    arm_fir_f32(&fir4S, adc4read, &adc4filtered[0], (uint32_t)BLOCK_SIZE);

    // sum all values of one block
    ADCoutVals[0] = 0.0f;
    ADCoutVals[1] = 0.0f;
    ADCoutVals[2] = 0.0f;
    ADCoutVals[3] = 0.0f;

    for (runi=0; runi<MAX_ADC_VALS; runi++)
    {
        ADCoutVals[0] += adc1filtered[runi];
        ADCoutVals[1] += adc2filtered[runi];
        ADCoutVals[2] += adc3filtered[runi];
        ADCoutVals[3] += adc4filtered[runi];
    }

    // division for average and normalization to 1V
    ADCoutVals[0] /= averageDenom;
    ADCoutVals[1] /= averageDenom;
    ADCoutVals[2] /= averageDenom;
    ADCoutVals[3] /= averageDenom;
}
```

A.1.4 Timer Configuration Function

```
static void Timer_Config(void)
{
    NVIC_InitTypeDef NVIC_InitStructure;
    TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;

    // System clock configuration
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);
    // Enable TIM4 IRQ Channel
    NVIC_InitStructure.NVIC_IRQChannel = TIM3_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 4;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
    // Time Base configuration
    TIM_TimeBaseStructInit(&TIM_TimeBaseStructure);
    TIM_TimeBaseStructure.TIM_Prescaler = (uint16_t) ((SystemCoreClock) /
100000) - 1; // 100 kHz
    TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
    TIM_TimeBaseStructure.TIM_Period = 199; // 100 kHz / 200 = 500 Hz
refresh
    TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;
    TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);
    // Activate Output of Update-Event by TRGO
    TIM_SelectOutputTrigger(TIM3, TIM_TRGOSource_Update);
    // TIM IT enable
    TIM_ITConfig(TIM3, TIM_IT_Update, ENABLE);
    // TIM4 counter enable (time counter)
    TIM_Cmd(TIM3, ENABLE);
}
```

A.1.5 Analog-to-Digital Converter Configuration Function

```
static void ADC_Config(void)
{
    __IO uint16_t           runVal = 0;
    NVIC_InitTypeDef        NVIC_InitStructure;
    GPIO_InitTypeDef        GPIO_InitStructure;
    DMA_InitTypeDef         DMA_InitStructure;
    ADC_InitTypeDef         ADC_InitStructure;
    ADC_CommonInitTypeDef   ADC_CommonInitStructure;

    // Enable peripheral clocks
    // PC2, PC3, PD11, PD12
    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOC, ENABLE);
    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOD, ENABLE);
    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1, ENABLE);
    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA2, ENABLE);
    RCC_ADCCLKConfig(RCC_ADC12PLLCLK_Div1);
    RCC_ADCCLKConfig(RCC_ADC34PLLCLK_Div1);
    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_ADC12, ENABLE);
    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_ADC34, ENABLE);

    // Interrupt init
    NVIC_InitStructure.NVIC_IRQChannel = DMA1_Channel1_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 2;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
    NVIC_InitStructure.NVIC_IRQChannel = DMA2_Channel5_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 3;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
    NVIC_Init(&NVIC_InitStructure);

    // set all to default
    ADC_DeInit(ADC1);
    ADC_DeInit(ADC2);
    ADC_DeInit(ADC3);
    ADC_DeInit(ADC4);
    GPIO_StructInit(&GPIO_InitStructure);
    DMA_StructInit(&DMA_InitStructure);
    ADC_StructInit(&ADC_InitStructure);

    // Enable voltage regulators
    ADC_VoltageRegulatorCmd(ADC1, ENABLE);
    ADC_VoltageRegulatorCmd(ADC2, ENABLE);
    ADC_VoltageRegulatorCmd(ADC3, ENABLE);
    ADC_VoltageRegulatorCmd(ADC4, ENABLE);
    // wait for startup time to complete
    for (runVal=0; runVal < 350; runVal++);

    // Configure ADC Channel pins as analog input
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AN;
    GPIO_InitStructure.GPIO_Speed= GPIO_Speed_2MHz;
    GPIO_InitStructure.GPIO_OType= GPIO_OType_PP;
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
    // PC2
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;
    GPIO_Init(GPIOC, &GPIO_InitStructure);
    // PC3
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3;
    GPIO_Init(GPIOC, &GPIO_InitStructure);
    // PD11
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11;
    GPIO_Init(GPIOD, &GPIO_InitStructure);
    // PD12
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_12;
    GPIO_Init(GPIOD, &GPIO_InitStructure);

    // DMA1 Channel1 and DMA2_Channel5 configuration
    DMA_InitStructure.DMA_PeripheralBaseAddr =
(uint32_t)(ADC1_2_BASE+0x0C);
    DMA_InitStructure.DMA_MemoryBaseAddr =
(uint32_t)&(uiADC12convValue[0]);
    DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
    DMA_InitStructure.DMA_BufferSize = 1;
    DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
    DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Disable;
    DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_Word;
    DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_Word;
    DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
    DMA_InitStructure.DMA_Priority = DMA_Priority_VeryHigh;
    DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
    DMA_Init(DMA1_Channel1, &DMA_InitStructure);

    DMA_InitStructure.DMA_PeripheralBaseAddr =
(uint32_t)(ADC3_4_BASE+0x0C);
    DMA_InitStructure.DMA_MemoryBaseAddr =
(uint32_t)&(uiADC34convValue[0]);
    DMA_InitStructure.DMA_Priority = DMA_Priority_High;
    DMA_Init(DMA2_Channel5, &DMA_InitStructure);

    DMA_Cmd(DMA1_Channel1, ENABLE);
    DMA_Cmd(DMA2_Channel5, ENABLE);

    // Differential Mode DISABLE
    ADC_SelectDifferentialMode(ADC1, ADC_Channel_8, DISABLE);
    ADC_SelectDifferentialMode(ADC2, ADC_Channel_9, DISABLE);
    ADC_SelectDifferentialMode(ADC3, ADC_Channel_8, DISABLE);
    ADC_SelectDifferentialMode(ADC4, ADC_Channel_9, DISABLE);

      // Set Calibration Mode to Single
      ADC_SelectCalibrationMode(ADC1, ADC_CalibrationMode_Single);
      ADC_SelectCalibrationMode(ADC2, ADC_CalibrationMode_Single);
      ADC_SelectCalibrationMode(ADC3, ADC_CalibrationMode_Single);
      ADC_SelectCalibrationMode(ADC4, ADC_CalibrationMode_Single);

      // Start calibrations and check end
      ADC_StartCalibration(ADC1);
      while(ADC_GetCalibrationStatus(ADC1));
      ADC_StartCalibration(ADC2);
      while(ADC_GetCalibrationStatus(ADC2));
      ADC_StartCalibration(ADC3);
      while(ADC_GetCalibrationStatus(ADC3));
      ADC_StartCalibration(ADC4);
      while(ADC_GetCalibrationStatus(ADC4));

    // wait for startup time to complete
    for (runVal=0; runVal < 35; runVal++);

    // Enable all ADCs
    ADC_Cmd(ADC1, ENABLE);
    ADC_Cmd(ADC2, ENABLE);
    ADC_Cmd(ADC3, ENABLE);
    ADC_Cmd(ADC4, ENABLE);
    while(ADC_GetFlagStatus(ADC1, ADC_FLAG_RDY) == RESET);
    while(ADC_GetFlagStatus(ADC2, ADC_FLAG_RDY) == RESET);
    while(ADC_GetFlagStatus(ADC3, ADC_FLAG_RDY) == RESET);
    while(ADC_GetFlagStatus(ADC4, ADC_FLAG_RDY) == RESET);

    // ADC Common configuration
    ADC_CommonInitStructure.ADC_Mode = ADC_Mode_RegSimul;
    ADC_CommonInitStructure.ADC_Clock = ADC_Clock_SynClkModeDiv1;
    ADC_CommonInitStructure.ADC_DMAAccessMode = ADC_DMAAccessMode_1;
    ADC_CommonInitStructure.ADC_DMAMode = ADC_DMAMode_Circular;
    ADC_CommonInitStructure.ADC_TwoSamplingDelay = 1;
    ADC_CommonInit(ADC1, &ADC_CommonInitStructure);
    ADC_CommonInit(ADC3, &ADC_CommonInitStructure);

    // ADC1 regular configuration
    ADC_InitStructure.ADC_ContinuousConvMode =
ADC_ContinuousConvMode_Disable;
    ADC_InitStructure.ADC_Resolution = ADC_Resolution_12b;
    ADC_InitStructure.ADC_ExternalTrigConvEvent =
ADC_ExternalTrigConvEvent_4;
    ADC_InitStructure.ADC_ExternalTrigEventEdge =
ADC_ExternalTrigEventEdge_RisingEdge;
    ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
    ADC_InitStructure.ADC_OverrunMode = ADC_OverrunMode_Disable;
    ADC_InitStructure.ADC_AutoInjMode = ADC_AutoInjec_Disable;
    ADC_InitStructure.ADC_NbrOfRegChannel = 1;
    ADC_Init(ADC1, &ADC_InitStructure);
    ADC_InitStructure.ADC_ExternalTrigConvEvent =
ADC_ExternalTrigConvEvent_11;
    ADC_Init(ADC3, &ADC_InitStructure);
    ADC_InitStructure.ADC_ExternalTrigConvEvent =
ADC_ExternalTrigConvEvent_0;
    ADC_InitStructure.ADC_ExternalTrigEventEdge =
ADC_ExternalTrigEventEdge_None;
    ADC_Init(ADC2, &ADC_InitStructure);
    ADC_Init(ADC4, &ADC_InitStructure);

    ADC_RegularChannelConfig(ADC1, ADC_Channel_8, 1,
ADC_SampleTime_7Cycles5);
    ADC_RegularChannelConfig(ADC2, ADC_Channel_9, 1,
ADC_SampleTime_7Cycles5);
    ADC_RegularChannelConfig(ADC3, ADC_Channel_8, 1,
ADC_SampleTime_7Cycles5);
    ADC_RegularChannelConfig(ADC4, ADC_Channel_9, 1,
ADC_SampleTime_7Cycles5);

    // Enable ADC1&3 DMA
    ADC_DMACmd(ADC1, ENABLE);
    ADC_DMACmd(ADC3, ENABLE);

    // Enable DMA interrupt
    DMA_ITConfig(DMA1_Channel1, DMA_IT_TC, ENABLE);
    DMA_ITConfig(DMA2_Channel5, DMA_IT_TC, ENABLE);

    ADC_StartConversion(ADC1);
    ADC_StartConversion(ADC3);
}
```

A.1.6 Serial Peripheral Interface Configuration Function

```
void SPI_Config(void)
{
    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOC, ENABLE);
    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOD, ENABLE);
    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA2, ENABLE);
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_SPI3, ENABLE);
    NVIC_InitTypeDef NVIC_InitStructure;
    NVIC_InitStructure.NVIC_IRQChannel = DMA2_Channel2_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 5;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
    GPIO_InitTypeDef GPIO_InitStructure;
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_12 | GPIO_Pin_11 | GPIO_Pin_10;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;      //Als Alternate
Function
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;    //Push-Pull Betrieb
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;    //Pull-Down
Widerstand aktiviert
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz; //50MHz Update-Rate
    GPIO_Init(GPIOC, &GPIO_InitStructure);
    GPIO_PinAFConfig(GPIOC, GPIO_PinSource10, GPIO_AF_6);
    GPIO_PinAFConfig(GPIOC, GPIO_PinSource11, GPIO_AF_6);
    GPIO_PinAFConfig(GPIOC, GPIO_PinSource12, GPIO_AF_6);
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
    GPIO_Init(GPIOD, &GPIO_InitStructure);
    GPIO_SetBits(GPIOD, GPIO_Pin_0);
    DMA_InitTypeDef DMA_InitStructure;
    DMA_InitStructure.DMA_PeripheralBaseAddr = (uint32_t)(SPI3_BASE+0x0C);
    DMA_InitStructure.DMA_MemoryBaseAddr = (uint32_t)&(f32SPI3txValues[0]);
    DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralDST;
    DMA_InitStructure.DMA_BufferSize = 16;
    DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
    DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
    DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_Byte;
    DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_Byte;
    DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
    DMA_InitStructure.DMA_Priority = DMA_Priority_Medium;
    DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
    DMA_Init(DMA2_Channel2, &DMA_InitStructure);
    SPI_InitTypeDef SPI_InitStructure;
    SPI_InitStructure.SPI_Direction = SPI_Direction_2Lines_FullDuplex;
    SPI_InitStructure.SPI_Mode = SPI_Mode_Master;
    SPI_InitStructure.SPI_DataSize = SPI_DataSize_8b;
    SPI_InitStructure.SPI_CPOL = SPI_CPOL_Low;
    SPI_InitStructure.SPI_CPHA = SPI_CPHA_1Edge;
    SPI_InitStructure.SPI_NSS = SPI_NSS_Soft;
    SPI_InitStructure.SPI_BaudRatePrescaler = SPI_BaudRatePrescaler_32;
// Clock is derived from the master, slave does not need to be set.
    SPI_InitStructure.SPI_FirstBit = SPI_FirstBit_MSB;
    SPI_InitStructure.SPI_CRCPolynomial = 7;
    SPI_Init(SPI3, &SPI_InitStructure);

    DMA_ITConfig(DMA2_Channel2, DMA_IT_TC, ENABLE);
}
/*************************END OF FILE****/
```

A.2 STM32F30X_IT.C

A.2.1 Includes and Standard Exception Handlers

```
/* Includes -------------------------------------------------------------*/
#include "stm32f30x_it.h"
#include "arm_math.h"
#include "stm32f30x_conf.h"
#include "stm32f3_discovery.h"

/* Private functions ----------------------------------------------------*/

/*************************************************************************/
/*    Cortex-M4 Processor Exceptions Handlers OMITTED FOR CLARITY!      */
/*************************************************************************/

// [INSERT STANDARD HANDLERS HERE]

/*************************************************************************/
/*            STM32F30x Peripherals Interrupt Handlers                   */
/*************************************************************************/
```

A.2.2 Timer 3 Interrupt Handler

```
void TIM3_IRQHandler(void)
{
  if (TIM_GetITStatus(TIM3, TIM_IT_Update) != RESET)
  {
    TIM_ClearITPendingBit(TIM3, TIM_IT_Update);

    STM_EVAL_LEDToggle(LED3);
  }
}
```

A.2.3 Direct Memory Access 1 Channel 1 Interrupt Handler (ADC12)

```
void DMA1_Channel1_IRQHandler(void)
{
  if (DMA_GetITStatus(DMA1_IT_TC1) != RESET)
  {
    DMA_ClearITPendingBit(DMA1_IT_TC1);

    adc1write[adcRecordIndex12] = (float32_t)uiADC12convValue[0];
    adc2write[adcRecordIndex12] = (float32_t)uiADC12convValue[1];

    adcRecordIndex12++;

    if (adcRecordIndex12 >= MAX_ADC_VALS)
    {
        adcRecordIndex12 = 0;

        if (isOddRun12)
        {
            isOddRun12 = false;
            adc1write = &adc1array2[0];
            adc1read  = &adc1array1[0];
            adc2write = &adc2array2[0];
            adc2read  = &adc2array1[0];
        }
        else // even run
        {
            isOddRun12 = true;
            adc1write = &adc1array1[0];
            adc1read  = &adc1array2[0];
            adc2write = &adc2array1[0];
            adc2read  = &adc2array2[0];
        }

        STM_EVAL_LEDToggle(LED6);
        ADC12ready = true;
    }
  }
}
```

A.2.4 Direct Memory Access 2 Channel 2 Interrupt Handler (SPI TX)

```
void DMA2_Channel2_IRQHandler(void)
{
  if (DMA_GetITStatus(DMA2_IT_TC2) != RESET)
  {
    DMA_ClearITPendingBit(DMA2_IT_TC2);
    // 1. Disable DMA streams for Tx and Rx in the DMA registers, if the
streams are used.
    DMA_Cmd(DMA2_Channel2, DISABLE);

    STM_EVAL_LEDToggle(LED9);

    SPI3ready = true;
  }
}
```

A.2.5 Direct Memory Access 2 Channel 5 Interrupt Handler (ADC34)

```
void DMA2_Channel5_IRQHandler(void)
{
  if (DMA_GetITStatus(DMA2_IT_TC5) != RESET)
  {
    DMA_ClearITPendingBit(DMA2_IT_TC5);

    adc3write[adcRecordIndex34] = (float32_t)uiADC34convValue[0];
    adc4write[adcRecordIndex34] = (float32_t)uiADC34convValue[1];

    adcRecordIndex34++;

    if (adcRecordIndex34 >= MAX_ADC_VALS)
    {
        adcRecordIndex34 = 0;

        if (isOddRun34)
        {
            isOddRun34 = false;
            adc3write = &adc3array2[0];
            adc3read  = &adc3array1[0];
            adc4write = &adc4array2[0];
            adc4read  = &adc4array1[0];
        }
        else // even run
        {
            isOddRun34 = true;
            adc3write = &adc3array1[0];
            adc3read  = &adc3array2[0];
            adc4write = &adc4array1[0];
            adc4read  = &adc4array2[0];
        }

        STM_EVAL_LEDToggle(LED7);
        ADC34ready = true;
    }
  }
}

/************************END OF FILE****/
```

B. Appendix – Final Program F429

B.1 MAIN.C

B.1.1 Includes, Defines, Variables and Function Prototypes

```
/* Includes ------------------------------------------------------------------*/
#include "stm32f4xx.h"
#include "stm32f4xx_conf.h"
#include "system_stm32f4xx.h"
#include <math.h>
#include "arm_math.h"
#include <stdio.h>
#include "stm32f429i_discovery.h"
#include "stm32f429i_discovery_ioe.h"
#include "stm32f429i_discovery_lcd.h"
#include "stm32f4xx_it.h"
#include "MicroGUI.h"
#include "SmithChart.h"

/* Private define ------------------------------------------------------------*/

  #define MESSAGE1       "Reflection Coefficient Control"
  #define MESSAGE2       "Version:2.1.1 / Author:V.Kible"
  #define LINENUM        0x17
  #define FONTSIZE       Font8x12

  #define USE_LCD

/* Private pin and port variables for Matching-LEDs ---------------------*/

uint16_t gpio_pin_match[7] =
{
    GPIO_Pin_4,  //PB4
    GPIO_Pin_7,  //PB7
    GPIO_Pin_11, //PC11
    GPIO_Pin_4,  //PD4
    GPIO_Pin_5,  //PD5
    GPIO_Pin_7,  //PD7
    GPIO_Pin_9   //PG9
};

GPIO_TypeDef* gpio_port_match[7] =
{
    GPIOB, //PB4
    GPIOB, //PB7
    GPIOC, //PC11
    GPIOD, //PD4
    GPIOD, //PD5
    GPIOD, //PD7
    GPIOG  //PG9
};

/* Private variables ---------------------------------------------------*/

__IO bool mainRoutineActivate;
__IO bool displaySPIvalsActivate;
__IO float32_t f32SPI4rxValues[4] = { 0.0f, 0.0f, 0.0f, 0.0f };
__IO float32_t f32SPI4copiedValues[4] = { 0.0f, 0.0f, 0.0f, 0.0f };

/* Private function prototypes -----------------------------------------*/

//static void FilterAndAverageADCvals(float32_t* ADC_re, float32_t* ADC_im,
float32_t* ADC_forw);
static void CalculateReflCoef(vec2D_t* reflCoefRaw);
static void LimitReflCoef(vec2D_t* reflCoefRaw, vec2D_t* reflCoefLim);
static void Matcher(vec2D_t* reflCoefLim);
static void StateSwitcher(void);
static void Timer_Config(void);
static void Pin_Config(void);
static void SPI_Config(void);
static void Init_MatchStates(void);

#ifdef USE_LCD
static void Display_Init(void);
#endif /* USE_LCD */

/* Private functions ---------------------------------------------------*/
```

B.1.2 Main Function

```
int main(void)
{
    vec2D_t   rawReflCoef, limReflCoef;

    mainRoutineActivate = false;
    displaySPIvalsActivate = false;
    activeState  = 0;
    desiredState = 0;

    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_4);

    Init_MatchStates();
    Init_SmithArrays();

    STM_EVAL_LEDInit(LED3);
    STM_EVAL_LEDInit(LED4);
    STM_EVAL_LEDOn(LED3);
    STM_EVAL_LEDOff(LED4);
    STM_EVAL_PBInit(BUTTON_USER, BUTTON_MODE_GPIO);

    Pin_Config();
    SPI_Config();
    Timer_Config();

    #ifdef USE_LCD
    Display_Init();
    IOE_Config();
    GUI_Init();
    #endif /* USE_LCD */

    while(1)
    {
        if (mainRoutineActivate)
        {
            mainRoutineActivate = false;

            STM_EVAL_LEDToggle(LED3); // heartbeat

            CalculateReflCoef(&rawReflCoef);

            LimitReflCoef(&rawReflCoef, &limReflCoef);

            Matcher(&limReflCoef);

            #ifdef USE_LCD
            GUI_Refresh(&rawReflCoef, &limReflCoef);
            #endif /* USE_LCD */

            StateSwitcher();
        }
    }
}
```

B.1.3 Reflection Coefficient Calculation and Correction Function

```
static void CalculateReflCoef(vec2D_t* reflCoefRaw)
{
    float32_t aRaw, bRaw, cRaw, dRaw, aCor, bCor, cCor, dCor, denom;

    if ((sinA_B == 0.0f) || (sinC_D == 0.0f))
    {
        reflCoefRaw->x = 0.0f;
        reflCoefRaw->y = 0.0f;
        return;
    }

    aRaw = (f32SPI4copiedValues[0]-aOffset)*aScale;
    bRaw = (f32SPI4copiedValues[1]-bOffset)*bScale;
    cRaw = (f32SPI4copiedValues[2]-cOffset)*cScale;
    dRaw = (f32SPI4copiedValues[3]-dOffset)*dScale;

    aCor = (bRaw*sinA-aRaw*sinB)/sinA_B; // Re{R}
    bCor = (aRaw*cosB-bRaw*cosA)/sinA_B; // Im{R}
    cCor = (dRaw*sinC-cRaw*sinD)/sinC_D; // Re{F}
    dCor = (cRaw*cosD-dRaw*cosC)/sinC_D; // Im{F}
    denom = cCor*cCor + dCor*dCor;

    if (denom == 0.0f)
    {
        reflCoefRaw->x = 0.0f;
        reflCoefRaw->y = 0.0f;
        return;
    }

    reflCoefRaw->x = (aCor*cCor + bCor*dCor)/denom;
    reflCoefRaw->y = (bCor*cCor - aCor*dCor)/denom;
}
```

B.1.4 Limiter Function

```
static void LimitReflCoef(vec2D_t* reflCoefRaw, vec2D_t* reflCoefLim)
{
    vec2D_t solution1, solution2, startPoint, endPoint;
    float32_t m, a, b, k, c, d;
    float32_t squaredDistance1, squaredDistance2, dist1x, dist1y, dist2x,
dist2y;

    startPoint.x = matchState[activeState].re;
    startPoint.y = matchState[activeState].im;

    endPoint.x = startPoint.x + reflCoefRaw->x;
    endPoint.y = startPoint.y + reflCoefRaw->y;

    if ( (startPoint.x != endPoint.x) || (startPoint.y != endPoint.y) )
    {
        if (endPoint.x * endPoint.x + endPoint.y * endPoint.y > 1.0f)
        { // outside unit circle
            if (startPoint.x == endPoint.x)
            { // vertical line
                d = 1.0f - startPoint.x * startPoint.x;
                if (d < 0.0f)
                { // dummy solution, no real solution
                    solution1.x = 0.0f;
                    solution1.y = 0.0f;
                    solution2.x = 0.0f;
                    solution2.y = 0.0f;
                }
                else
                { // calculate the intersections with the unit circle
                    arm_sqrt_f32(d, &k);
                    solution1.x = startPoint.x;
                    solution1.y = k;
                    solution2.x = startPoint.x;
                    solution2.y = -k;
                }
            }
            else // line is not vertical, m is not infinite
            {
                m = (endPoint.y - startPoint.y) / (endPoint.x -
startPoint.x); // slope
                a = 1.0f + m * m;
                b = 2.0f * m * (startPoint.y - m * startPoint.x);
                k = (m * startPoint.x - startPoint.y);
                c = k * k - 1.0f;
                d = b * b - 4.0f * a * c; // discriminant

                if (d < 0.0f)
                { // dummy solution, no real solution
                    solution1.x = 0.0f;
                    solution1.y = 0.0f;
                    solution2.x = 0.0f;
                    solution2.y = 0.0f;
                }
                else
                { // calculate the intersections with the unit circle
                    arm_sqrt_f32(d, &k);

                    solution1.x = (-b + k) / (2.0f * a);
                    solution1.y = m * (solution1.x - startPoint.x) +
startPoint.y;

                    solution2.x = (-b - k) / (2.0f * a);
                    solution2.y = m * (solution2.x - startPoint.x) +
startPoint.y;
                }
            }

            // the desired solution is the one that is closer to the
endPoint, so calculate a distance measure
            dist1x = solution1.x - endPoint.x;
            dist1y = solution1.y - endPoint.y;
            dist2x = solution2.x - endPoint.x;
            dist2y = solution2.y - endPoint.y;
            squaredDistance1 = dist1x * dist1x + dist1y * dist1y;
            squaredDistance2 = dist2x * dist2x + dist2y * dist2y;

            // compare the two distance measures to decide what solution to
use
            if (squaredDistance1 < squaredDistance2)
            {
                reflCoefLim->x = solution1.x;
                reflCoefLim->y = solution1.y;
            }
            else
            {
                reflCoefLim->x = solution2.x;
                reflCoefLim->y = solution2.y;
            }
        }
        else // within unit circle
        {
            reflCoefLim->x = endPoint.x;
            reflCoefLim->y = endPoint.y;
        }
    }
    else // startPoint = endPoint, perfectly matched condition
    {
        reflCoefLim->x = endPoint.x;
        reflCoefLim->y = endPoint.y;
    }
}
```

B.1.5 Matcher Function

```
static void Matcher(vec2D_t* reflCoefLim)
{
    uint8_t runVal;
    uint8_t minNumber = 0;
    float32_t minDistSq = 1000000.0f;
    float32_t distanceSq;

    float32_t distx = matchState[activeState].re - reflCoefLim->x;
    float32_t disty = matchState[activeState].im - reflCoefLim->y;

    float32_t activeDistSq = distx * distx + disty * disty;

    // check only if outside active matching state
    if ( activeDistSq >
(matchState[activeState].rad)*(matchState[activeState].rad) )
    {
        // find new minimum distance
        for (runVal = 0; runVal < NUM_MATCH_STATE; runVal++)
        {
            distx = matchState[runVal].re - reflCoefLim->x;
            disty = matchState[runVal].im - reflCoefLim->y;

            distanceSq = distx * distx + disty * disty;
            if (distanceSq < minDistSq)
            {
                minDistSq = distanceSq;
                minNumber = runVal;
            }
        }

        // second check, to prevent the system from oscillating in the not
matchable area
        if ( (activeDistSq - minDistSq) > 0.1f)
                desiredState = minNumber;
        else    desiredState = activeState;
    }
    else desiredState = activeState;
}
```

B.1.6 State Switcher Function

```
static void StateSwitcher(void)
{
    static bool buttonStateOld = false;
    bool buttonState = STM_EVAL_PBGetState(BUTTON_USER);
    uint8_t run;

    // if button pressed (only rising edge)
    if ( (buttonState == true) && (buttonStateOld == false) )
    {
        STM_EVAL_LEDOn(LED4);

        // do control action
        activeState = desiredState;

        // set outputs accordingly
        for (run = 0; run < NUM_MATCH_STATE; run++)
        {
            GPIO_WriteBit(gpio_port_match[run], gpio_pin_match[run], (
(matchState[activeState].out_pin)&(1 << run) ) >> run );
        }
    }
    else STM_EVAL_LEDOff(LED4);

    buttonStateOld = buttonState;
}
```

B.1.7 Timer 4 Configuration Function

```
static void Timer_Config(void)
{
    NVIC_InitTypeDef NVIC_InitStructure;
    TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;

    /* System clock configuration */
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4, ENABLE);
    /* Enable TIM4 IRQ Channel */
    NVIC_InitStructure.NVIC_IRQChannel = TIM4_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 4;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
    /* Time Base configuration */
    TIM_TimeBaseStructInit(&TIM_TimeBaseStructure);
    TIM_TimeBaseStructure.TIM_Prescaler = (uint16_t) ((SystemCoreClock/2) /
100000) - 1; // 100 kHz
    TIM_TimeBaseStructure.TIM_ClockDivision = 0;
    TIM_TimeBaseStructure.TIM_Period = 19999; // 100 kHz / 20000 = 5 Hz ->
5 Hz refresh
    TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
    TIM_TimeBaseInit(TIM4, &TIM_TimeBaseStructure);
    /* TIM IT enable */
    TIM_ITConfig(TIM4, TIM_IT_Update, ENABLE);
    /* TIM4 counter enable (time counter) */
    TIM_Cmd(TIM4, ENABLE);
}
```

B.1.8 Output Pin Configuration Function

```
static void Pin_Config(void)
{
    uint8_t run;
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOB, ENABLE);
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC, ENABLE);
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOG, ENABLE);

    GPIO_InitTypeDef GPIO_InitStructure;
    GPIO_StructInit(&GPIO_InitStructure);
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;      //Als Ausgang
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;     //Push-Pull Betrieb
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_DOWN;  //Kein Pull-Up oder
Pull-Down Widerstand aktiviert
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz; //50MHz Update-Rate
    for (run = 0; run < NUM_MATCH_STATE; run++)
    {
        GPIO_InitStructure.GPIO_Pin = gpio_pin_match[run];
        GPIO_Init(gpio_port_match[run], &GPIO_InitStructure);
        if (run == 0)   GPIO_SetBits(gpio_port_match[run],
gpio_pin_match[run]);
        else            GPIO_ResetBits(gpio_port_match[run],
gpio_pin_match[run]);
    }
}
```

B.1.9 Serial Peripheral Interface Configuration Function

```
void SPI_Config(void)
{
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOE, ENABLE);
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_DMA2, ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_SPI4, ENABLE);

    NVIC_InitTypeDef NVIC_InitStructure;
    NVIC_InitStructure.NVIC_IRQChannel = DMA2_Stream0_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 2;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);

    GPIO_InitTypeDef GPIO_InitStructure;
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2 | GPIO_Pin_4 | GPIO_Pin_5 |
GPIO_Pin_6;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;      //Als Alternate
Function
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;    //Push-Pull Betrieb
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;    //Pull-Down
Widerstand aktiviert
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz; //50MHz Update-Rate
    GPIO_Init(GPIOE, &GPIO_InitStructure);
    GPIO_PinAFConfig(GPIOE, GPIO_PinSource2, GPIO_AF_SPI4);
    GPIO_PinAFConfig(GPIOE, GPIO_PinSource4, GPIO_AF_SPI4);
    GPIO_PinAFConfig(GPIOE, GPIO_PinSource5, GPIO_AF_SPI4);
    GPIO_PinAFConfig(GPIOE, GPIO_PinSource6, GPIO_AF_SPI4);

    DMA_InitTypeDef DMA_InitStructure;
    DMA_InitStructure.DMA_Channel = DMA_Channel_4;
    DMA_InitStructure.DMA_PeripheralBaseAddr = (uint32_t)(SPI4_BASE+0x0C);
    DMA_InitStructure.DMA_Memory0BaseAddr =
(uint32_t)&(f32SPI4rxValues[0]);
    DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralToMemory;
    DMA_InitStructure.DMA_BufferSize = 16;
    DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
    DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
    DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_Byte;
    DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_Byte;
    DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
    DMA_InitStructure.DMA_Priority = DMA_Priority_VeryHigh;
    DMA_InitStructure.DMA_FIFOMode = DMA_FIFOMode_Disable;
    DMA_InitStructure.DMA_FIFOThreshold = DMA_FIFOThreshold_HalfFull;
    DMA_InitStructure.DMA_MemoryBurst = DMA_MemoryBurst_Single;
    DMA_InitStructure.DMA_PeripheralBurst = DMA_PeripheralBurst_Single;
    DMA_Init(DMA2_Stream0, &DMA_InitStructure);
    DMA_ITConfig(DMA2_Stream0, DMA_IT_TC, ENABLE);

    SPI_InitTypeDef SPI_InitStructure;
    SPI_InitStructure.SPI_Direction = SPI_Direction_2Lines_FullDuplex;
    SPI_InitStructure.SPI_Mode = SPI_Mode_Slave;
    SPI_InitStructure.SPI_DataSize = SPI_DataSize_8b;
    SPI_InitStructure.SPI_CPOL = SPI_CPOL_Low;
    SPI_InitStructure.SPI_CPHA = SPI_CPHA_1Edge;
    SPI_InitStructure.SPI_NSS = SPI_NSS_Hard;
    SPI_InitStructure.SPI_BaudRatePrescaler = SPI_BaudRatePrescaler_32;
// Clock is derived from the master, slave does not need to be set.
    SPI_InitStructure.SPI_FirstBit = SPI_FirstBit_MSB;
    SPI_InitStructure.SPI_CRCPolynomial = 7;
    SPI_Init(SPI4, &SPI_InitStructure);

    DMA_Cmd(DMA2_Stream0, ENABLE);

    SPI_I2S_DMACmd(SPI4, SPI_I2S_DMAReq_Rx, ENABLE);

    SPI_Cmd(SPI4, ENABLE);
}
```

B.1.10 Matching States Initialization Function

```
static void Init_MatchStates(void)
{
    uint8_t run;

        matchState[0].re      = 0.0f;
        matchState[0].im      = 0.0f;
        matchState[0].rad     = 0.3f;
        matchState[0].out_pin = 1;

    for (run=1; run<NUM_MATCH_STATE; run++)
    {
        matchState[run].re      = 0.5f * arm_cos_f32( 2*PI*((float32_t)run-
1.0f)/((float32_t)NUM_MATCH_STATE-1.0f) );
        matchState[run].im      = 0.5f * arm_sin_f32( 2*PI*((float32_t)run-
1.0f)/((float32_t)NUM_MATCH_STATE-1.0f) );
        matchState[run].rad     = 0.3f;
        matchState[run].out_pin = 1 << run;
    }
}
```

B.1.11 Display Initialization Function

```
#ifdef USE_LCD
static void Display_Init(void)
{
  //uint8_t aTextBuffer[50];
  //sFONT*  pActiveFont;

  /* Initialize the LCD */
  LCD_Init();
  LCD_LayerInit();

  /* Eable the LTDC */
  LTDC_Cmd(ENABLE);

  /* Set LCD Background Layer  */
  LCD_SetLayer(LCD_BACKGROUND_LAYER);

  /* Clear the Background Layer */
  LCD_Clear(LCD_COLOR_WHITE);

  /* Configure the transparency for background */
  LCD_SetTransparency(0);

  /* Set LCD Foreground Layer  */
  LCD_SetLayer(LCD_FOREGROUND_LAYER);

  /* Configure the transparency for foreground */
  LCD_SetTransparency(200);

  /* Clear the Foreground Layer */
  LCD_Clear(LCD_COLOR_WHITE);

  /* Set the LCD Back Color and Text Color*/
  LCD_SetBackColor(LCD_COLOR_BLUE);
  LCD_SetTextColor(LCD_COLOR_WHITE);

    /* Set the LCD Text size */
  LCD_SetFont(&FONTSIZE);

  LCD_DisplayStringLine(LCD_LINE_0, (uint8_t*)MESSAGE1);
  LCD_DisplayStringLine(LCD_LINE_1, (uint8_t*)MESSAGE2);

  /* Set the LCD Back Color and Text Color*/
  LCD_SetBackColor(LCD_COLOR_WHITE);
  LCD_SetTextColor(LCD_COLOR_BLUE);
}
#endif /* USE_LCD */

// [ASSERT FUNCTION OMITTED FOR CLARITY]

/* END OF FILE ********************************************/
```

B.2 STM32F4XX_IT.C

B.2.1 Includes, Variables, Standard Exception Handlers

```
/* Includes ------------------------------------------------------------*/
#include "arm_math.h"
#include "stm32f4xx_it.h"
#include "stm32f4xx_conf.h"
#include "stm32f429i_discovery.h"

/* Private variables ---------------------------------------------------*/
extern __IO bool mainRoutineActivate;
extern __IO bool displaySPIvalsActivate;
extern __IO float32_t f32SPI4rxValues[4];
extern __IO float32_t f32SPI4copiedValues[4];

/**********************************************************************/
/*        Cortex-M4 Processor Exceptions Handlers OMITTED FOR CLARITY!    */
/**********************************************************************/

// [INSERT STANDARD HANDLERS HERE]

/**********************************************************************/
/*             STM32F4xx Peripherals Interrupt Handlers               */
/**********************************************************************/
```

B.2.2 Timer 4 Interrupt Handler

```
void TIM4_IRQHandler(void)
{
  if (TIM_GetITStatus(TIM4, TIM_IT_Update) != RESET)
  {
    TIM_ClearITPendingBit(TIM4, TIM_IT_Update);

    mainRoutineActivate = true;
  }
}
```

B.2.3 Direct Memory Access 2 Stream 0 Interrupt Handler (SPI RX)

```
void DMA2_Stream0_IRQHandler(void)
{
  if (DMA_GetITStatus(DMA2_Stream0, DMA_IT_TCIF0) != RESET)
  {
    DMA_ClearITPendingBit(DMA2_Stream0, DMA_IT_TCIF0);

    f32SPI4copiedValues[0]=f32SPI4rxValues[0];
    f32SPI4copiedValues[1]=f32SPI4rxValues[1];
    f32SPI4copiedValues[2]=f32SPI4rxValues[2];
    f32SPI4copiedValues[3]=f32SPI4rxValues[3];

    displaySPIvalsActivate = true;
  }
}
/*************END OF FILE*************/
```

B.3 MICROGUI.C

B.3.1 Includes, Defines, Types and Variables

```
/* Includes ----------------------------------------------------------*/
#include "MicroGUI.h"
#include <stdio.h>
#include "stm32f429i_discovery.h"
#include "stm32f429i_discovery_ioe.h"
#include "stm32f429i_discovery_lcd.h"
#include "SmithChart.h"
#include <math.h>
#include "arm_math.h"

#define ABS(X)  ((X) > 0 ? (X) : -(X))

typedef enum BOOL { false, true } bool;

extern __IO bool displaySPIvalsActivate;
extern __IO float32_t f32SPI4copiedValues[4];

TP_STATE* TouchState;
uint16_t  OldTouchDetected;
FLASH_Status flashResult;
uint16_t radioButtonSignal, radioButtonValue, radioButtonFactor,
windowNumber;

// Memory Region In Flash
__attribute__((__section__(".user_data"))) const int32_t userConfig[15];

uint8_t infoTextError[] = "Error writing!";
uint8_t infoTextStart[] = "Write Flash?";
uint8_t* pInfoText = infoTextStart;
int32_t configLimitPos[15] = { 1800, 9999, 99999, 1800, 9999, 99999, 1800,
9999, 99999, 1800, 9999, 99999, 0, 0, 0 };
int32_t configLimitNeg[15] = {-1800,-9999,     1,-1800,-9999,     1,-1800,-
9999,     1,-1800,-9999,     1, 0, 0, 0 };

GUI_button_t button[] =
{ /* { Caption, Xpos, Ypos, Width, Height } */
    {"EDIT", 130, 280, 100,  30 },
    {"RI",    10,  60,  40,  30 },
    {"RQ",    70,  60,  40,  30 },
    {"FI",   130,  60,  40,  30 },
    {"FQ",   190,  60,  40,  30 },
    {"ANG",   10, 110,  40,  30 },
    {"OFS",   70, 110,  40,  30 },
    {"SCA",  130, 110,  40,  30 },
    {"0.1",   10, 160,  40,  30 },
    {"1",     70, 160,  40,  30 },
    {"10",   130, 160,  40,  30 },
    {"100",  190, 160,  40,  30 },
    {"-",     10, 210,  40,  30 },
    {"+",    190, 210,  40,  30 },
    {"CANCEL",  10, 280, 100,  30 },
    {"SAVE",   130, 280, 100,  30 },
    {"CANCEL",  10, 160, 100,  30 },
    {"SAVE",   130, 160, 100,  30 }
};
```

B.3.2 Function to Deserialize Configuration Values

```
void GUI_DeserializeConfigVals(void)
{
    // angle mismatch in 0.1 degrees
    // copyConfig[0] // RI ANG A --- sinA, cosA
    cosA = arm_cos_f32( PI*((float32_t)copyConfig[0])/1800.0f );
    sinA = arm_sin_f32( PI*((float32_t)copyConfig[0])/1800.0f );
    // copyConfig[3] // RQ ANG B --- sinB, cosB, sinA_B
    cosB = arm_cos_f32( PI*((float32_t)copyConfig[3])/1800.0f );
    sinB = arm_sin_f32( PI*((float32_t)copyConfig[3])/1800.0f );
    sinA_B = arm_sin_f32( PI*((float32_t)(copyConfig[0]-
copyConfig[3]))/1800.0f );
    // copyConfig[6] // FI ANG C --- sinC, cosC
    cosC = arm_cos_f32( PI*((float32_t)copyConfig[6])/1800.0f );
    sinC = arm_sin_f32( PI*((float32_t)copyConfig[6])/1800.0f );
    // copyConfig[9] // FQ ANG D --- sinD, cosD, sinC_D
    cosD = arm_cos_f32( PI*((float32_t)copyConfig[9])/1800.0f );
    sinD = arm_sin_f32( PI*((float32_t)copyConfig[9])/1800.0f );
    sinC_D = arm_sin_f32( PI*((float32_t)(copyConfig[6]-
copyConfig[9]))/1800.0f );

    // offsets in 0.1mV int32 to 1V float32
    aOffset = ((float32_t)(copyConfig[1]+15000))/10000.0f;  // RI OFS
    bOffset = ((float32_t)(copyConfig[4]+15000))/10000.0f;  // RQ OFS
    cOffset = ((float32_t)(copyConfig[7]+15000))/10000.0f;  // FI OFS
    dOffset = ((float32_t)(copyConfig[10]+15000))/10000.0f; // FQ OFS

    // scaling mismatch in 0.1% int32 to part of one
    aScale = ((float32_t)copyConfig[2])/1000.0f;  // RI SCA aScale
    bScale = ((float32_t)copyConfig[5])/1000.0f;  // RQ SCA bScale
    cScale = ((float32_t)copyConfig[8])/1000.0f;  // FI SCA cScale
    dScale = ((float32_t)copyConfig[11])/1000.0f; // FQ SCA dScale

    // (0=RI 1=RQ 2=FI 3=FQ)*3 + (0=ANG 1=OFS 2=SCA)
}
```

B.3.3 Function to Check Buttons for Touch Event

```
uint16_t GUI_CheckTouchButtons(uint16_t firstButtonNum, uint16_t
lastButtonNum)
{
    uint16_t run = firstButtonNum-1;
    bool buttonTriggered = false;

    while ( (run < lastButtonNum) && (!buttonTriggered) )
    {
        buttonTriggered = ( (TouchState->X >= button[run].Xpos) &&
(TouchState->X <= (button[run].Xpos + button[run].Width ) ) &&
                            (TouchState->Y >= button[run].Ypos) &&
(TouchState->Y <= (button[run].Ypos + button[run].Height) ) );
        run++;
    }

    if (buttonTriggered) return run;
    else                 return 0;
}
```

B.3.4 Draw Buttons Function

```
void GUI_DrawButtons(uint16_t firstButtonNum, uint16_t lastButtonNum,
uint16_t relativeActiveButtonNum)
{
    uint16_t run = firstButtonNum-1;
    uint16_t absActiveButton = run + relativeActiveButtonNum;
    while (run < lastButtonNum)
    {
        if (run == absActiveButton)
        {

            LCD_SetTextColor(GUI_BACK_ACTIVE);
            LCD_DrawFullRect(button[run].Xpos, button[run].Ypos,
button[run].Width, button[run].Height);
            LCD_SetBackColor(GUI_BACK_ACTIVE);
            LCD_SetTextColor(GUI_TEXT_ACTIVE);
        }
        else
        {
            LCD_SetTextColor(GUI_BACK_INACTV);
            LCD_DrawFullRect(button[run].Xpos, button[run].Ypos,
button[run].Width, button[run].Height);
            LCD_SetBackColor(GUI_BACK_INACTV);
            LCD_SetTextColor(GUI_TEXT_INACTV);
        }
        LCD_DisplayStringCenteredXY(button[run].Xpos + button[run].Width/2,
button[run].Ypos + button[run].Height/2, button[run].Caption);

        run++;
    }
}
```

B.3.5 Draw Value Box Function

```
void GUI_DrawValueBox(void)
{
    uint8_t aTextBuffer[8];
    uint8_t posArr = radioButtonSignal*3 + radioButtonValue;

    LCD_SetBackColor(LCD_COLOR_WHITE);
    LCD_SetTextColor(LCD_COLOR_BLACK);

    LCD_DrawRect(70, 210, 30, 100);

    sprintf((char*)aTextBuffer, "%ld.%ld", copyConfig[posArr]/10,
ABS(copyConfig[posArr])%10 );
    LCD_DisplayStringCenteredXY(120, 225, aTextBuffer);
}
```

B.3.6 Function to Program Flash Memory

```
bool GUI_ProgramFlash(void)
{
    uint8_t run;

    copyConfig[14]++; // for test

    FLASH_Unlock();
    FLASH_ClearFlag(FLASH_FLAG_EOP | FLASH_FLAG_OPERR | FLASH_FLAG_WRPERR |
FLASH_FLAG_PGAERR | FLASH_FLAG_PGSERR);
    flashResult = FLASH_EraseSector(FLASH_Sector_23, VoltageRange_3);

    run = 0;
    while ( (run < 15) && (flashResult == FLASH_COMPLETE) )
    {
        flashResult = FLASH_ProgramWord((uint32_t)(&(userConfig[run])),
copyConfig[run]);
        run++;
    }

    FLASH_Lock();

    return (flashResult == FLASH_COMPLETE);
}
```

B.3.7 GUI Refresh Function

```
void GUI_Refresh(vec2D_t* rawReCo, vec2D_t* reflCoef)
{
    OldTouchDetected = TouchState->TouchDetected;
    TouchState = IOE_TP_GetState();

    if ( (!TouchState->TouchDetected) && OldTouchDetected )
    { // execute whenever touch is released
        switch (windowNumber)
        {
            case 2:
                GUI_CheckWindow2();
                break;
            case 3:
                GUI_CheckWindow3();
                break;
            default:
                GUI_CheckWindow1();
                break; // 1
        }

        // clear screen area
        LCD_SetBackColor(LCD_COLOR_WHITE);
        LCD_SetTextColor(LCD_COLOR_WHITE);
        LCD_DrawFullRect(8, 58, 224, 254);

        // Window Number can change in ButtonCheck Routines
        switch (windowNumber)
        {
            case 2:
                GUI_RedrawWindow2();
                break;
            case 3:
                GUI_RedrawWindow3();
                break;
            default:
                GUI_RedrawWindow1();
                break; // 1
        }
    }

    if (windowNumber == 1)
    {
        GUI_RedrawReflectionCoefficientChart(reflCoef);
        GUI_RedrawSPIvals(rawReCo);
    }
}
```

B.3.8 Function to Check Window 1 for Touch

```
void GUI_CheckWindow1(void)
{
    switch ( GUI_CheckTouchButtons(1, 1) )
    {
        case 1:
            // button 1 pressed, go to window 2, preset values
            radioButtonSignal = 0;
            radioButtonValue = 0;
            radioButtonFactor = 1;
            windowNumber = 2;
            break;
        default:
            // no button, do nothing
            break;
    }
}
```

B.3.9 Function to Check Window 2 for Touch

```
void GUI_CheckWindow2(void)
{
    uint8_t posArr = radioButtonSignal*3 + radioButtonValue;
    int32_t edFact = 1;
    uint8_t run;
    for (run = 0; run < radioButtonFactor; run++) edFact *= 10;

    switch ( GUI_CheckTouchButtons(2, 16) )
    {
        case 2:
            radioButtonSignal = 0;
            break;
        case 3:
            radioButtonSignal = 1;
            break;
        case 4:
            radioButtonSignal = 2;
            break;
        case 5:
            radioButtonSignal = 3;
            break;
        case 6:
            radioButtonValue = 0;
            break;
        case 7:
            radioButtonValue = 1;
            break;
        case 8:
            radioButtonValue = 2;
            break;
        case 9:
            radioButtonFactor = 0;
            break;
        case 10:
            radioButtonFactor = 1;
            break;
        case 11:
            radioButtonFactor = 2;
            break;
        case 12:
            radioButtonFactor = 3;
            break;
        case 13:
            copyConfig[posArr] -= edFact;
            if (copyConfig[posArr] < configLimitNeg[posArr])
                copyConfig[posArr] = configLimitNeg[posArr];
            break;
        case 14:
            copyConfig[posArr] += edFact;
            if (copyConfig[posArr] > configLimitPos[posArr])
                copyConfig[posArr] = configLimitPos[posArr];
            break;
        case 15:
            for (run = 0; run < 15; run++)
            {
                copyConfig[run] = userConfig[run];
            }
            windowNumber = 1;
            break;
        case 16:
            pInfoText = infoTextStart;
            windowNumber = 3;
            break;
        default:
            // no button, do nothing
            break;
    }
}
```

B.3.10 Function to Check Window 3 for Touch

```
void GUI_CheckWindow3(void)
{
    switch ( GUI_CheckTouchButtons(17, 18) )
    {
        case 17:
            windowNumber = 2;
            break;
        case 18:
            if ( GUI_ProgramFlash() )
            {
                GUI_DeserializeConfigVals();
                windowNumber = 1;
            }
            else pInfoText = infoTextError;
            break;
        default:
            // no button, do nothing
            break;
    }
}
```

B.3.11 Function to Redraw Window 1

```
void GUI_RedrawWindow1(void)
{
    LCD_SetFont(&Font12x12);
    GUI_DrawButtons(1, 1, 100);
}
```

B.3.12 Function to Redraw Window 2

```
void GUI_RedrawWindow2(void)
{
    LCD_SetFont(&Font8x12);
    GUI_DrawButtons(2,   5, radioButtonSignal);
    GUI_DrawButtons(6,   8, radioButtonValue);
    GUI_DrawButtons(9,  12, radioButtonFactor);
    LCD_SetFont(&Font16x24);
    GUI_DrawButtons(13, 14, 100);
    GUI_DrawValueBox();
    LCD_SetFont(&Font12x12);
    GUI_DrawButtons(15, 16, 100);
}
```

B.3.13 Function to Redraw Window 3

```
void GUI_RedrawWindow3(void)
{
    uint8_t aTextBuffer[20];

    LCD_SetFont(&Font16x24);
    LCD_SetBackColor(LCD_COLOR_WHITE);
    LCD_SetTextColor(LCD_COLOR_RED);
    LCD_DisplayStringCenteredXY(120, 105, pInfoText);

    LCD_SetFont(&Font12x12);
    LCD_SetTextColor(LCD_COLOR_BLACK);
    sprintf((char*)aTextBuffer, "WR-OPs:%ld", copyConfig[14]);
    LCD_DisplayStringCenteredXY(120, 245, aTextBuffer);
    GUI_DrawButtons(17, 18, 100);
}
```

B.3.14 Function to Redraw the Values that were Received from SPI

```
void GUI_RedrawSPIvals(vec2D_t* rawReflCoef)
{
    uint8_t aTextBuffer[20];
    LCD_SetFont(&Font8x12);
    LCD_SetBackColor(LCD_COLOR_BLUE2);
    LCD_SetTextColor(LCD_COLOR_BLACK);
    sprintf((char*)aTextBuffer, "1:%+09ld 2:%+09ld",
(int32_t)(1000000.0f*f32SPI4copiedValues[0]),
(int32_t)(1000000.0f*f32SPI4copiedValues[1]));
    LCD_DisplayStringLine(LCD_LINE_2, aTextBuffer);
    sprintf((char*)aTextBuffer, "3:%+09ld 4:%+09ld",
(int32_t)(1000000.0f*f32SPI4copiedValues[2]),
(int32_t)(1000000.0f*f32SPI4copiedValues[3]));
    LCD_DisplayStringLine(LCD_LINE_3, aTextBuffer);
    sprintf((char*)aTextBuffer, "Re:%+09ld",
(int32_t)(1000000.0f*rawReflCoef->x));
    LCD_DisplayStringLine(LCD_LINE_23, aTextBuffer);
    sprintf((char*)aTextBuffer, "Im:%+09ld",
(int32_t)(1000000.0f*rawReflCoef->y));
    LCD_DisplayStringLine(LCD_LINE_24, aTextBuffer);
}
```

B.3.15 Function to Redraw the Reflection Coefficient Chart

```
void GUI_RedrawReflectionCoefficientChart(vec2D_t* reflCoefLim)
{
    uint8_t run;

    // Clear drawing area
    LCD_SetBackColor(LCD_COLOR_WHITE);
    LCD_SetTextColor(LCD_COLOR_WHITE);
    LCD_DrawFullRect(18, 58, 204, 204);

    // Draw Smith diagram
    LCD_SetTextColor(ASSEMBLE_RGB(0xD0, 0xD0, 0xD0));
    LCD_DrawQuarterCircle(220, 60, 100, 2);
    LCD_DrawQuarterCircle(220, 260, 100, 3);
    LCD_DrawCircle(170, 160, 50);
    LCD_DrawUniLine(20, 160, 220, 160);
    DrawSmithCircles();

    // Draw outer circle
    LCD_SetTextColor(LCD_COLOR_BLACK);
    LCD_DrawCircle(120, 160, 100);

    // Draw matching circles in GRAY
    LCD_SetTextColor(LCD_COLOR_GREY);
    for (run=0; run<7; run++)
    {
        if ( (run != activeState) && (run != desiredState) )
        {
            LCD_DrawCircle( (uint16_t)(120.0f+100.0f*(matchState[run].re)),
                            (uint16_t)(160.0f-100.0f*(matchState[run].im)),
                            (uint16_t)(       100.0f*(matchState[run].rad))
);
        }
    }
    // Draw active circle
    LCD_SetTextColor(LCD_COLOR_BLACK);
    LCD_DrawCircle( (uint16_t)(120.0f+100.0f*(matchState[activeState].re)),
                    (uint16_t)(160.0f-100.0f*(matchState[activeState].im)),
                    (uint16_t)(       100.0f*(matchState[activeState].rad))
);

    // Draw desired circle
    LCD_SetTextColor(LCD_COLOR_BLUE2);
    LCD_DrawCircle(
(uint16_t)(120.0f+100.0f*(matchState[desiredState].re)),
                    (uint16_t)(160.0f-
100.0f*(matchState[desiredState].im)),
                    (uint16_t)(
100.0f*(matchState[desiredState].rad)) );

    // Draw line in RED
    LCD_SetTextColor(LCD_COLOR_RED);
    LCD_DrawUniLine((uint16_t)(120.0f+100.0f*(matchState[activeState].re)),
                    (uint16_t)(160.0f-100.0f*(matchState[activeState].im)),
                    (uint16_t)(120.0f+100.0f*(reflCoefLim->x)),
                    (uint16_t)(160.0f-100.0f*(reflCoefLim->y)) );
}
```

B.3.16 GUI Initialization Function

```
void GUI_Init(void)
{
    uint8_t run;

    windowNumber = 1;

    for (run = 0; run < 15; run++)
    {
        copyConfig[run] = userConfig[run];
    }

    GUI_DeserializeConfigVals();

    GUI_DrawButtons(1, 1, 100);
}
/***** END OF FILE ****************************************************/
```

B.4 SMITHCHART.C

B.4.1 Includes, Defines and Variables

```
#include <math.h>
#include "arm_math.h"
#include "stm32f429i_discovery.h"
#include "stm32f429i_discovery_lcd.h"
#include "SmithChart.h"

#define NUM_PTS 11

uint16_t pts1xx[NUM_PTS+1];
uint16_t pts1yp[NUM_PTS+1];
uint16_t pts1ym[NUM_PTS+1];
uint16_t pts2xx[NUM_PTS+1];
uint16_t pts2yp[NUM_PTS+1];
uint16_t pts2ym[NUM_PTS+1];
```

B.4.2 Function to Convert a Normalized Impedance to Reflection Coefficient

```
void ZtoReflCoef(float32_t Zre, float32_t Zim, float32_t* RCre, float32_t*
RCim)
{
    float32_t Zre1 = Zre + 1.0f;
    float32_t nenner = Zre1*Zre1 + Zim*Zim;
    *RCre = (Zre*Zre + Zim*Zim - 1.0f)/nenner;
    *RCim = 2.0f*Zim/nenner;
}
```

B.4.3 Function to Draw a Circle with Constant Real Part

```
void DrawCircleReal(float32_t ZreCirc)
{
    float32_t RCreCirc, dummy, circCenter, circRadius;

    ZtoReflCoef(ZreCirc, 0.0f, &RCreCirc, &dummy);
    circCenter = (RCreCirc + 1.0f) / 2.0f;
    circRadius = 1.0f - circCenter;

    LCD_DrawCircle( (uint16_t)(120.0f+100.0f*circCenter),
                    (uint16_t)(160.0f),
                    (uint16_t)(       100.0f*circRadius) );
}
```

B.4.4 Function to Draw a Circle with Constant Imaginary Part

```
void DrawCircleImag(float32_t ZimCirc)
{
    float32_t ZreCirc = 0.0f;
    float32_t RCreNew, RCimNew, RCreOld, RCimOld;

    ZtoReflCoef(ZreCirc, ZimCirc, &RCreOld, &RCimOld);

    ZreCirc = 0.1f;
    for(ZreCirc = 0.1f; ZreCirc < 50.0f; ZreCirc *= 2.0f)
    {
        ZtoReflCoef(ZreCirc, ZimCirc, &RCreNew, &RCimNew);

        LCD_DrawUniLine((uint16_t)(120.0f+100.0f*RCreOld),
                        (uint16_t)(160.0f-100.0f*RCimOld),
                        (uint16_t)(120.0f+100.0f*RCreNew),
                        (uint16_t)(160.0f-100.0f*RCimNew) );
        LCD_DrawUniLine((uint16_t)(120.0f+100.0f*RCreOld),
                        (uint16_t)(160.0f+100.0f*RCimOld),
                        (uint16_t)(120.0f+100.0f*RCreNew),
                        (uint16_t)(160.0f+100.0f*RCimNew) );

        RCreOld = RCreNew;
        RCimOld = RCimNew;
    }

}
```

B.4.5 Function to Fill Arrays with Values for Fast Redraw of Smith Chart

```
void Init_SmithArrays(void)
{
    float32_t G, reZ, reRC, imRC;
    uint16_t run;
    for(run = 0; run < NUM_PTS; run++)
    {
        G = 2.0f*((float32_t)run)/((float32_t)NUM_PTS)-1.0f;
        reZ = (1+G)/(1-G);
        ZtoReflCoef(reZ, 0.33333333f, &reRC, &imRC);
        pts1xx[run] = (uint16_t)(120.0f+100.0f*reRC);
        pts1yp[run] = (uint16_t)(160.0f-100.0f*imRC);
        pts1ym[run] = (uint16_t)(160.0f+100.0f*imRC);
        ZtoReflCoef(reZ, 3.0f, &reRC, &imRC);
        pts2xx[run] = (uint16_t)(120.0f+100.0f*reRC);
        pts2yp[run] = (uint16_t)(160.0f-100.0f*imRC);
        pts2ym[run] = (uint16_t)(160.0f+100.0f*imRC);
    }
    pts1xx[NUM_PTS] = 220;
    pts1yp[NUM_PTS] = 160;
    pts1ym[NUM_PTS] = 160;
    pts2xx[NUM_PTS] = 220;
    pts2yp[NUM_PTS] = 160;
    pts2ym[NUM_PTS] = 160;
}
```

B.4.6 Smith Circles Drawing Function (Based on Arrays)

```
void DrawSmithCircles(void)
{
    uint16_t run, run1;
    for(run = 0; run < NUM_PTS; run++)
    {
        run1 = run + 1;
        LCD_DrawUniLine(pts1xx[run], pts1yp[run], pts1xx[run1],
pts1yp[run1]);
```

```
        LCD_DrawUniLine(pts1xx[run], pts1ym[run], pts1xx[run1],
pts1ym[run1]);
        LCD_DrawUniLine(pts2xx[run], pts2yp[run], pts2xx[run1],
pts2yp[run1]);
        LCD_DrawUniLine(pts2xx[run], pts2ym[run], pts2xx[run1],
pts2ym[run1]);
    }
    LCD_DrawCircle(145, 160, 75);
    LCD_DrawCircle(195, 160, 25);
}
```

B.4.7 Alternative Smith Circles Drawing Function

```
void DrawSmithCirclesFull(void)
{
    uint8_t run;
    float32_t circValues[2] = { 0.333f, 3.0f };

    for(run=0; run < 2; run++)
    {
        DrawCircleReal(circValues[run]);
        DrawCircleImag(circValues[run]);
    }
}
/***** END OF FILE ***********************************/
```

C. Appendix – Matching Circles MATLAB Script

C.1 Matching Network Script

```
% Main definitions
f    = 400e6; % Hz

Gmin = 0.28; % Absolute value

Z0   = 50;

C1   = [1.00e-12;  0.50e-12;  5.00e-12;  0.50e-12;  0.50e-12;  25.0e-12;...
        9.00e-12;  9.00e-12;  0.50e-12;  9.00e-12;  25.0e-12;  0.50e-12];
        % in F (close to PA)
C2   = [10.0e-12;  10.0e-12;  10.0e-12;  8.00e-12;  28.0e-12;  10.0e-12;...
        4.40e-12;  2.00e-12;  4.40e-12;  7.20e-12;  2.00e-12;  28.0e-12];
        % in F (replacing L, middle of the network)
C3   = [2.00e-12;  5.00e-12;  0.50e-12;  8.00e-12;  2.00e-12;  0.50e-12;...
        0.50e-12;  0.50e-12;  10.0e-12;  0.50e-12;  2.00e-12;  10.0e-12];
        % in F (close to load)

% values for circuit calculations
w    = 2.*pi.*f;
Li   = Z0./w        % H
Ci   = 1./(Z0.*w)   % F
Xli  = j.*w.*Li;    % Ohm
Yci  = j.*w.*Ci;    % S (1/Ohm)
Yc1  = j.*w.*C1;
Yc2  = j.*w.*C2;
Yc3  = j.*w.*C3;

% circuit equations
Yp1  = Yc1 + 1/Z0 + Yci;
Zp1  = 1./Yp1;
Zs1  = Zp1 + Xli;
Ys1  = 1./Zs1;
Yp2  = Yc2 + Ys1 + 2.*Yci;
Zp2  = 1./Yp2;
Zs2  = Zp2 + Xli;
```

```
Ys2  = 1./Zs2;
Ypo  = Yc3 + Ys2 + Yci;
Zout = 1./Ypo;

% reflection coefficient S22
S22  = (Zout - Z0) ./ (Zout + Z0);
S22c = conj(S22);

format long;

% circles
denominator = 1 - abs(Gmin.*S22).^2;
cent = S22c.*(1 - Gmin.^2)./denominator
rad  = Gmin.*(1 - abs(S22).^2)./denominator

FDrawCircles( cent, rad, Gmin );

%%% END OF FILE %%%
```

C.2 FDrawCircles Function

```
function [ ] = FDrawCircles( center, radius, minGamma )
% FDrawCircles draws cicles and smith chart %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % Input values error check %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    [m,n] = size(center);
    [o,p] = size(radius);

    if ( (n ~= 1) || (p ~= 1) || (m ~= o) )
        error('Both inputs must be vectors and of same size!')
    end

    if ( sum(imag(radius)) ~= 0.0 )
        error('Radius must be pure real vector or value!')
    end

    % Prepare chart (draw border of smith chart and add labels) %%%%%%%%%%%
    alpha = linspace(0,2*pi,50);
    x = cos(alpha);
    y = sin(alpha);
    hand = figure;
    set(hand,'Position',[100,100,700,700]);
    plot(x, y,'k');
    xlabel('Re(\Gamma)','fontsize',16);
    ylabel('Im(\Gamma)','fontsize',16);
    hold on;

    % Draw circle of minimum reflection coefficient %%%%%%%%%%%%%%%%%%%%%%%
    plot(x.*minGamma, y.*minGamma,'r','LineWidth',2);

    % Draw circles with constant real part %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    plot(0.5 + x./2, y./2,'--k');         % z = 1   --> G = 0
    plot(0.25 + x.*0.75, y.*0.75,'--k'); % z = 1/3 --> G = - 1/2
    plot(0.75 + x.*0.25, y.*0.25,'--k'); % z = 3   --> G = + 1/2

    % Draw circles (and line) with constant imaginary part %%%%%%%%%%%%%%%%
    y = 1 - sin(alpha./4);
    x = 1 - cos(alpha./4);
    plot(x, y,'--k', x, -y, '--k'); % the two circles with Im(z)=1 & -1

    G = linspace(-1,0.999,50);      % equal distribution in smith chart
    beta  = (1+G)./(1-G);           % transform to impedance plane
    gamma = (beta + j./3 - 1)./(beta + j./3 + 1); % transform back to smith
    x = real(gamma);
    y = imag(gamma);
    plot(x, y,'--k', x, -y, '--k'); % the two circles with Im(z)=1/3 & -1/3

    gamma = (beta + j.*3 - 1)./(beta + j.*3 + 1);
    x = real(gamma);
    y = imag(gamma);
    plot(x, y,'--k', x, -y, '--k'); % the two circles with Im(z)=3 & -3

    plot([-1 1],[0 0],'--k');       % the line with Im(z)=0
    % Draw matching circles %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    sinAlpha = sin(alpha);
    cosAlpha = cos(alpha);
    for run = 1:m
        text(real(center(run,1)),imag(center(run,1)),int2str(run),...
            'HorizontalAlignment','center','Color','b');
        x = real(center(run,1)) + radius(run,1)*cosAlpha;
        y = imag(center(run,1)) + radius(run,1)*sinAlpha;
        plot(x, y,'b','LineWidth',2);
    end

    hold off;
end

%%% END OF FILE %%%
```

D. Appendix –MATLAB Script for Correction of TML

```
clear all;
diary('ResultsDiary.txt')

syms Z1 Z2 Z3 Z4 Z5 Z6 Z7 GL
syms Z01 Z02 Z03 Z04 Z05 Z06
syms tE1 tE2 tE3 tE4 tE5 tE6
syms GtRe GtIm GlRe GlIm

assume(Z01,'real')
assume(Z02,'real')
assume(Z03,'real')
assume(Z04,'real')
assume(Z05,'real')
assume(Z06,'real')

assumeAlso(Z01 >= 0)
assumeAlso(Z02 >= 0)
assumeAlso(Z03 >= 0)
assumeAlso(Z04 >= 0)
assumeAlso(Z05 >= 0)
assumeAlso(Z06 >= 0)

assume(tE1,'real')
assume(tE2,'real')
assume(tE3,'real')
assume(tE4,'real')
assume(tE5,'real')
assume(tE6,'real')

assume(GtRe,'real')
assume(GtIm,'real')
assume(GlRe,'real')
assume(GlIm,'real')

%assumeAlso(1.0 >= GtRe >= -1.0)
%assumeAlso(1.0 >= GtIm >= -1.0)
```

```
% SYSTEM EQUATIONS
A = GL == (Z1 - sym(50))/(Z1 + sym(50));
B = Z2 == Z01*(Z1+1i*Z01*tE1)/(Z01+1i*Z1*tE1);
C = Z3 == Z02*(Z2+1i*Z02*tE2)/(Z02+1i*Z2*tE2);
D = Z4 == Z03*(Z3+1i*Z03*tE3)/(Z03+1i*Z3*tE3);
E = Z5 == Z04*(Z4+1i*Z04*tE4)/(Z04+1i*Z4*tE4);
F = Z6 == Z05*(Z5+1i*Z05*tE5)/(Z05+1i*Z5*tE5);
G = Z7 == Z06*(Z6+1i*Z06*tE6)/(Z06+1i*Z6*tE6);
H = GtRe + 1i*GtIm == (Z7 - sym(50))/(Z7 + sym(50));

% SYMBOLIC SOLUTION
[SGL,SZ1,SZ2,SZ3,SZ4,SZ5,SZ6,SZ7] = solve(A,B,C,D,E,F,G,H, ...
  GL, Z1, Z2, Z3, Z4, Z5, Z6, Z7);
% alphabetical order of outputs!
% Symbolic Solution is very long...
% Equation of interest is SGL (Solution Gamma Load)

% NUMERIC SOLUTION
VarNames  = ...
[Z01,   Z02,   Z03,   Z04,   Z05,   Z06,   tE1, ...
tE2,        tE3,        tE4,        tE5,        tE6]
% VarValues from Simulation, tEx is tangent of Ex
VarValues = ...[50.000,53.863,51.018,62.794,85.083,83.377,0.087488664, ...
0.098755749,0.102387497,0.013614409,0.007592328,0.188482015]

% Substitute numeric values into Solution Gamma Load
subGL = subs(SGL, VarNames, VarValues);

% Real and Imaginary Part
reSubGL = real(subGL)
imSubGL = imag(subGL)

% Floating Point Solution with 6 Digits
fpGLre = vpa(reSubGL,6)
fpGLim = vpa(imSubGL,6)

% Print in better readable form
pretty(fpGLre)
%{
             12                  12                  11
#3 (5.51533 10   GtRe - 3.16187 10   GtIm + 1.8277 10  )
-------------------------------------------------------- +
                           #1

          12                  12                  11
(5.51533 10   GtIm + 3.16187 10   GtRe - 6.99413 10  ) #2
---------------------------------------------------------
                           #1

where

          2     2
   #1 == #3  + #2

                 11                  11                  12
   #2 == 1.8277 10   GtIm + 6.99413 10   GtRe - 3.16187 10

                 11                  11                  12
   #3 == 1.8277 10   GtRe - 6.99413 10   GtIm + 5.51533 10

%}
```

```
pretty(fpGLim)
%{
              12                    12                    11
#3 (5.51533 10   GtIm + 3.16187 10   GtRe - 6.99413 10  )
----------------------------------------------------------
                            #1

                12                    12                   11
     (5.51533 10   GtRe - 3.16187 10   GtIm + 1.8277 10  ) #2
   - ---------------------------------------------------------
                                  #1

where

         2     2
   #1 == #3  + #2

                  11                    11                    12
   #2 == 1.8277 10   GtIm + 6.99413 10   GtRe - 3.16187 10

                  11                    11                    12
   #3 == 1.8277 10   GtRe - 6.99413 10   GtIm + 5.51533 10

%}

% The factor of 10^11 can be removed because it appears both in
% numerator and denominator in both factors (divide num and denom
% by (10^11)^2). Then same numbers can be replaced by same names.

%%% END OF FILE %%%
```