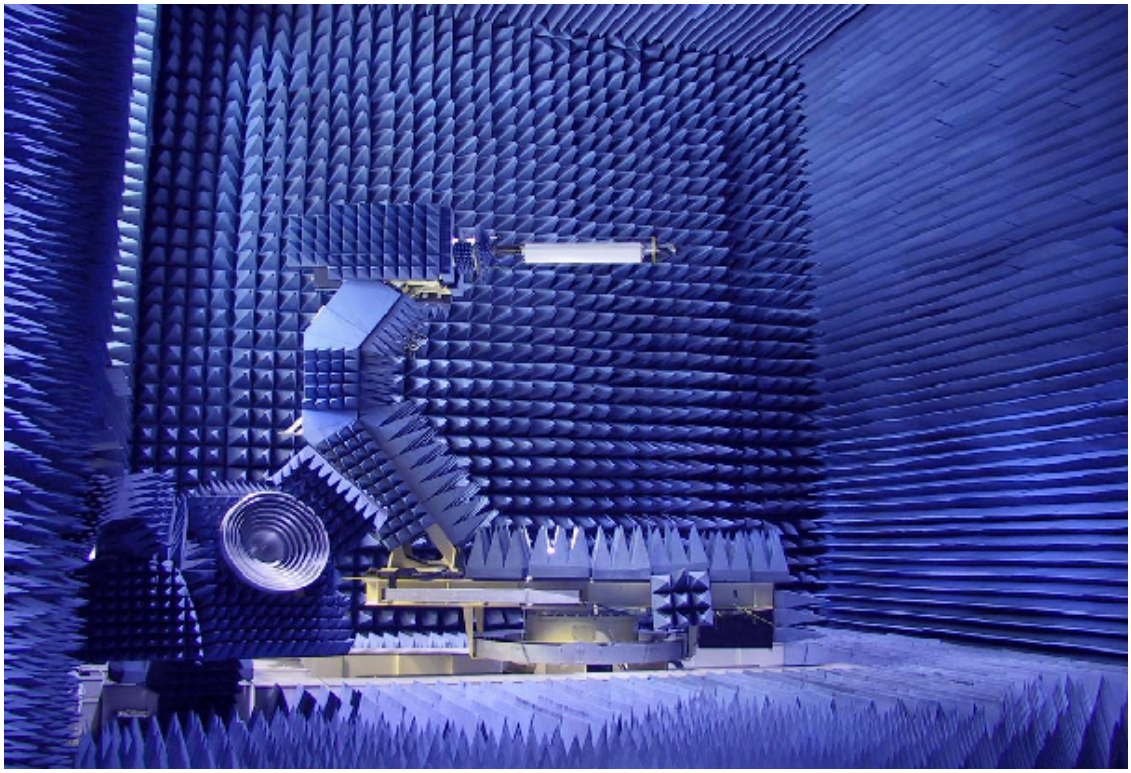
CHALMERS
UNIVERSITY OF TECHNOLOGY



Remote Control of Compact Antenna Test Range for Automated Measurements

Report

Bachelor Thesis in Electrical Engineering

Pontus Andersson
Niklas Görrel

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Remote Control of Compact Antenna Test Range for Automated Measurements
Pontus Andersson
Niklas Görrel
Department of Electrical Engineering
Chalmers University of Technology

Abstract

The present software SAAB Surveillance is using to remote control automated measurements of antennas in the compact arena test range, Mölndal could be improved. The task of this project is to explore if an alternative software could be implemented to make it more user friendly and flexible. To be able to find out what could be improved upon, a research in how the test range functions and how measurements is performed has been conducted and presented. The result is a program containing functions to control axes, switches, polarization and a network analyzer. The program is created in the programming language Matlab.

Sammanfattning

Den nuvarande programvaran SAAB Surveillance använder för att fjärrstyra automatiska mätningar av antenner i antennamätsträckan, medför svårigheter. Projektets syfte är att undersöka om en alternativ programvara skulle kunna implementeras för att göra mätningar mer användarvänliga och flexibla. För att ta reda på vad som kan förbättras har en granskning som omfattar hur mätsträckan fungerar och hur mätningar går till genomförts och presenterats. Resultatet är ett program som innehåller funktioner för att styra axlar, switchar, polarisering och en nätverksanalysator. Programmet skapas med hjälp av programmeringsspråket Matlab.

Acknowledgements

This thesis is a part of the electrical engineering program, 180 HP, at Chalmers University of Technology and comprises 15 HP. The project has been carried out at SAAB Surveillance's compact antenna test range at A15, Mölndal.

We have through out this project been supported by SAAB and its employees. Through times of hardship and success. We are grateful for them being open to cooperate with us. Special thanks should be addressed to Oscar Johansson, our mentor and Olle Grundén at A15 for lots of advise and expertise. We also want to thank Maria Köhler for recruiting us to the project. Finally we want to thank Ashraf Zaman, our examiner for a job well done.

Terminology

ASCII - *American based standard code for information interchange*, a coding system based on characters, including the English alphabet and numbers 0-9.

AUT - *Antenna Under Test*

BCD - *Binary Coded Decimal*

CAN - *Controller Area Network* is a network bus that connect electronic control units together and allows them to communicate.

CATR - *Compact Antenna Test Range*

DUT - *Device Under Test*

GUI - *Graphical User Interface*

IF - *Intermediate Frequency*

LAN - *Local Area Network* is a local network limited to a smaller area which uses a network protocol for communication.

LNA - *Low Noise Amplifier*

LO - *Local Oscillator*

RF - *Radio frequency*

RS-232 - *Recommended Standard-232*, A serial communication method established by *Electronics Industries Association* in the 1960's.

TCP/IP - *Transmission control protocol / internet protocol* is an architecture for communication over different networks.

Contents

List of Figures

1
Introduction

1.1 Background

SAAB Antenna Design and measurements group belong to the Radar Engineering department. The department is responsible for measurement of antennas. Their work includes both testing and verification of antennas, which they do at one of their compact test range facilities named A15, located in Mölndal. The SAAB Group has approximately 17,000 employees and delivers to more than 30 countries and primarily works within the defense industry.

The increasing complexity of modern radar systems also means that test and verification systems need to become more complex. Today's control system for the compact range is controlled by a software called Active Cell, which works for all types of measurements but with the disadvantage that automated measurements cannot be achieved as these require a present person who actively participates. This means that time is wasted, which can be spent on more important purposes. In order to respond to the technical development that has taken place, SAAB has developed external functions that today are controlled via Matlab. Matlab is a development language that is already widespread within the industry. Therefore, it is advantageous if existing software can be replaced with scripts and functions created in this coding environment. The control of measuring objects needs to be integrated with the measurement control system to be able to switch to larger and more efficient automated tests which the current system can not handle. It will be easier and more user friendly if all objects can be controlled in one and the same software. By implementing the project, future upgrades will be more simple to integrate due to that SAAB then has knowledge about the structure of the control program which they did not have before. There are a great possibility that the shortcomings of the existing system could be avoided.

Below are the shortcomings of the existing system identified and summarized with information on what a solution would imply:

- **Time consummation**
  Unnecessary time has to be spent on monitoring measurements.
  -This can be done fully automated which saves valuable time.
- **Difficulties in further development**
  The manufacture cannot provide new functions or updates to the software.
  -SAAB employees will have the competence to upgrade the system by themselves.
- **Complicated user interface**
  Misleading information regarding transmitter and receiver.
  -Since the user interface can be designed in a desired way, then it will be easy to set standards.
- **Multiple software**
  Several software are required to perform a measurement.
  -The system would only consist of one software, Matlab. This will make the measurement fully autonomous.

1.2 Purpose

The objectives of the project includes creating functions to control different sections of the compact range. The most fundamental part is to control the mechanical axes of the compact test range. The axes can move in multiple dimensions and direction X, Y, Azimuth, Elevation and Roll. The movement of the axes and the remaining measurement system is desired to be controlled in the same Matlab script. Additionally, the project includes controlling the different RF, *Radio frequency* paths for the compact test range (Trasmit(TX) and Recive(RX), etc) also using functions created in Matlab. These functions must be standalone so that they can be reused in several measurements for different measuring objects. This requires an understanding of how the compact measurement range operates for various types of measurements and when different system modes are used. It is therefore also a vital part of the project to specify the multiple components and how they interact and how the entire compact range operates.

2

Compact Antenna Test Range

2.1 Introduction

When measuring antennas in a far field, a long distance between the transmitter and the antenna is required since the waves emitted from the transmitter are not uniform plane waves. A compact antenna test range makes it possible to measure antennas at a much shorter distance in an enclosed facility. Thus concerns about weather conditions and leaking confidential information is no longer needed to take in account. The tests are performed in an anechoic chamber, which results in the test being optimal and with high performance.

2.2 Feed Antennas

A feed horn is providing the Device under test, DUT with a signal, or as a receiver when the DUT is transmitting as can be seen in figure 2.2. The feed horns at the test range has the ability to send waves with both horizontal and vertical polarization (more about polarization can be found in the theory chapter). This meanwhile the antennas and standard gain horn can only send in one polarization [1].

2.3 Axes

When performing measurements on antennas located on the control tower, it is desirable to be able to control the tower in a number of dimensions to test all possible cases and all the angles that cover a full sphere the antenna can either send or receive from. This means that the tower in question must be able to move in all planes in order to cover all possible scenarios. The tower can therefore move in the following axes: X, Y, Azimuth, Elevation and Roll. A representation of all the axes is shown in Figure 2.1 below:

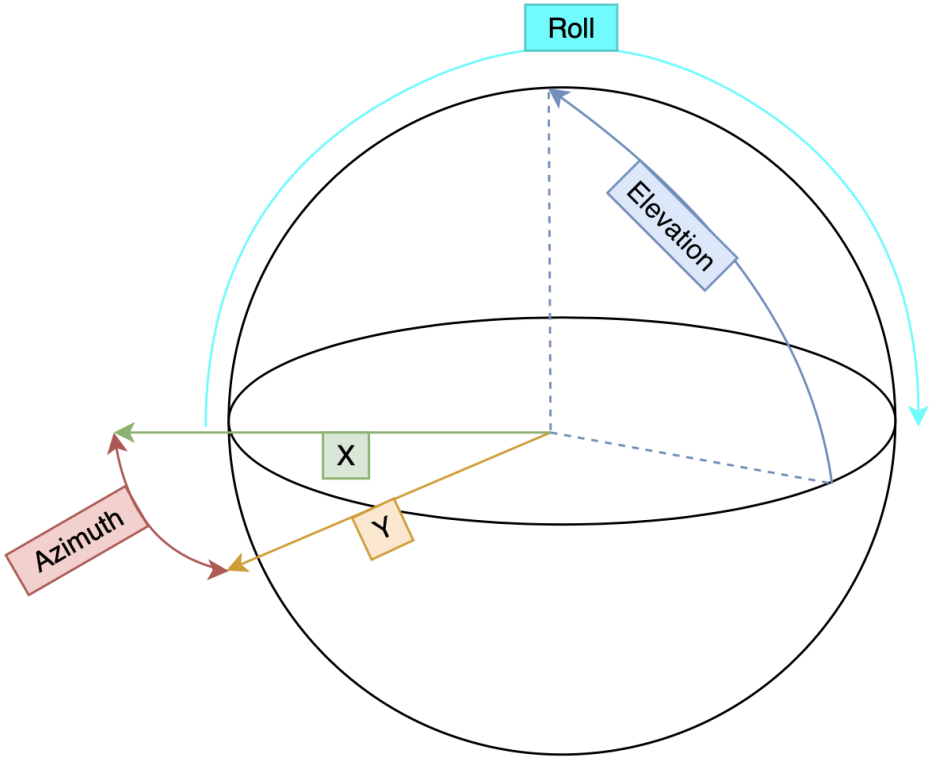


Figure 2.1: Visualization of all space dimensions

2.4 Reflectors

The reflector functions as a large mirror reflecting the output signal, and focuses the signal from the transmitter. When the signal is reflected, the appearance of the signal changes from being waves to becoming uniform plane waves. This means that the unit being tested perceives the signal in the way it would have looked if it had traveled a longer distance. The wave is not perfectly plane, it will still have some curvature. In order to achieve a perfect plane wave, the size of an ideal reflector has to be infinitely large, and since that is not possible, only an approximate planar wave front can be achieved. The most plane portion of the wave is the best for testing, and is known as the "quiet zone". The size of the quiet zone is about 50-60% of the dimensions of the reflector. So, the greater reflector the bigger quiet zone [2]. The reflector is the most critical component in the system. The shape, size as well as the material will indirect affect the results of the measurement.

To get the most efficient and ideal compact range possible, the reflection edge diffraction need to get minimized. This is done by having as good reflectors as possible. The reflector located in the compact range is of the serrated edge type. What makes this reflector special is that it enhances the quiet zone by spreading the diffracted energy in a wider area as the edge angles are constantly changing [3].

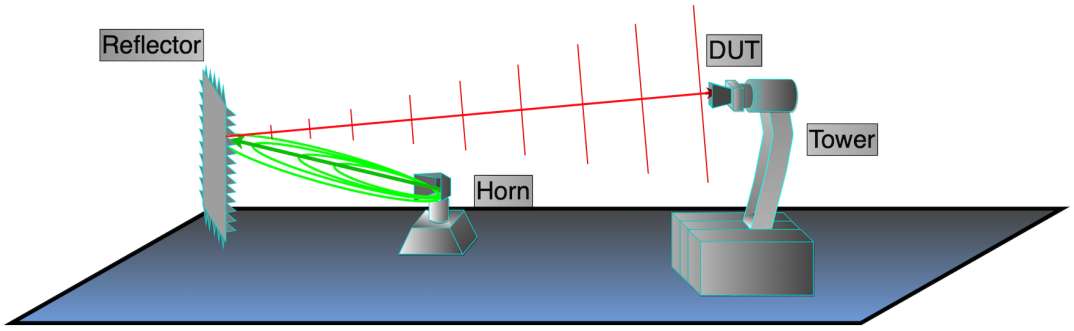


Figure 2.2: Visualization of uniform plane wave

3
System Components

3.1 Antennas

The main function of an antenna is to convert a RF signal from a transmitter to a propagating electromagnetic wave. This electromagnetic wave consist of both electric and magnetic field, where the electric field arises as a result of when a voltage is applied to an antenna. The magnetic field occurs as a current passes through the antenna. The waves have important characteristics such as that they are perpendicular to each other and that they propagate long distances at the speed of light. One of the most important features of all antennas is that there are relationships between amplification, operating frequency and size that cannot be avoided. This is due to that the electromagnetic operation of an antenna requires efficient radiation of a signal, which causes the antenna to have minimal physical dimensions relative to the electric wavelength $\lambda = c/f$ at the operating frequency. This means that the antenna size is reduced as the frequency increases, so antennas designed for microwave frequencies tend to be small and antennas designed for low frequencies are larger [2].

3.2 Electrical components

In this section various electrical components will be addressed.

3.2.1 Filters

Filters are commonly used to sort out or reject specific frequencies in a signal. A filter can be designed in a certain way to achieve a desired outcome, for example if only a higher range of frequencies are sought after a high-pass filter could be applied. If only lower frequencies are sought after a low-pass filter could be applied, etc. There are two main categories of filters, passive and active. The passive filters are constructed using resistors, capacitors and/or inductors. The filters are passive because they only consist of passive components and therefore do not amplify the signal [4]. An active filter on the other hand, can amplify a signal and usually consist of a RC-network in connection to an operational amplifier with a feedback connection.

In figure 3.1, an example of a signal passing through a passive low-pass filter is visualized. The frequencies below the cut-off frequency is not attenuated. It is the

cut-off frequency that decides which frequencies that should be left as they are and which that should be rejected. By choosing component values that corresponds to the desired cut-off frequency the user can design a filter that performs the needed action. The cut-off frequency is equal to the frequency where the signal is reduced by a factor of 0.5 (-3dB).

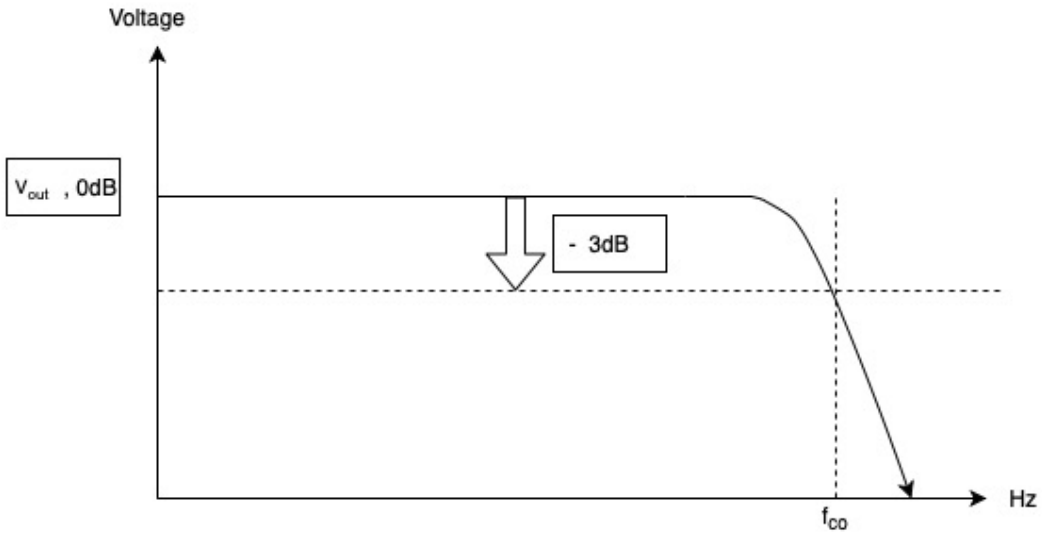


Figure 3.1: Visualization of attenuation of a signal passing through a low-pass filter. Where as the y-axis corresponds to Voltage [v] and x-axis corresponds to frequency[Hz].

3.2.2 Power Amplifiers

Amplifiers are usable tools for changing the gain of a signal. The ideal amplifier emits an output which is directly proportional to the input. The linear amplifier behaves almost as the ideal amplifier, replicates the input with an added gain. This amplifier is commonly used for audio signals. If the signal is of the type radio frequency, RF the amplitude will certainly be variable. The all over power level will then increase after passing a linear amplifier, this is also a common area of use for this amplifier [4]. The amplifiers is divided into different classes depending on their properties. The classes is differentiated using letters A to C.

When constructing an amplifier, a problem that arises is that if a high-noise signal passes through the amplifier, the noise will be amplified as well. There is an direct relation between amplification and noise. A special kind of amplifier called low noise amplifier, LNA is amplifying a signal, but strive to minimize the noise amplification.

3.2.3 Mixers

A mixer is a component that ideally has two inputs and one output. The main task of the mixer is to merge the sum of the two input signals as well as the difference of the two, this will result in two separate signals. This signals are usually sinusodial. The first input of the mixer receives an IF signal and the second input receives a signal from an local oscillator, LO. The LO generates a carrier signal that is mixed with the IF signal inside the mixer, the output signal generated will be both LO + IF and LO-IF [4]. The desired signal is obtained by filtering out unwanted frequencies. Mixers are also used on the receiver side by mixing down the received signal using the same LO frequency from a local oscillator [5].

3.2.4 Radio system

In order to send and receive information in an open space modulated into propagating waves, a radio system is required. This system is utilizing the components described in this chapter to construct a receiver and a transmitter. The two are mirror images of each other in respect to component order. At the transmission side a coded message is modulated into a IF-signal that is filtered and mixed with a carrier signal from the local oscillator. The desired high frequency signal is filtered out, amplified and then sent via the transmitting antenna. At the receiving side an antenna picks up the signal and it passes through a filter to a LNA. The signal is amplified and omit noise from nearby component and external sources. The received signal is then mixed with the same carrier wave from an other LO. Lastly, the original IF-signal is obtained through filtering and amplification. The signal is then demodulated and the message is received [4]. An example of a basic radio system can be seen in figure 3.2.

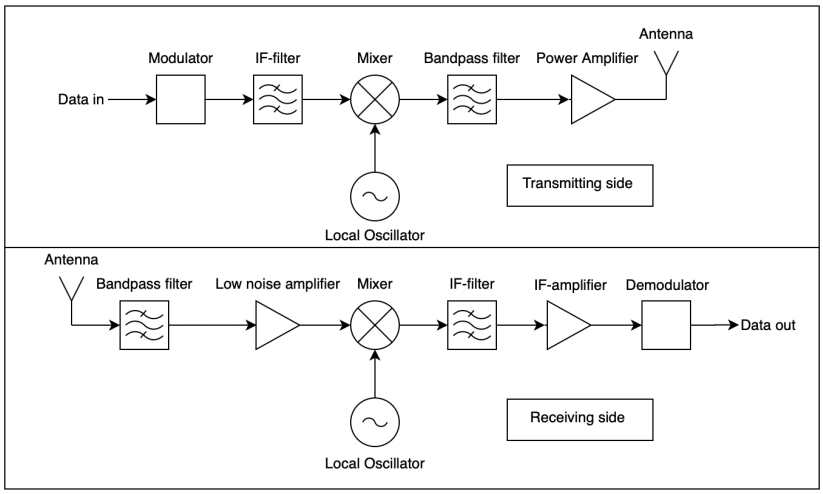


Figure 3.2: Example of a basic radio system.

3.2.5 Directional Coupler

The coupler is a component which can be used to divide a signal into two separate signals. The most common coupler is the 3dB coupler, where the signals will obtain half of the power gain each or in other words a reduction of gain equal to 3dB, hence the name. The component has four ports as can be seen in fig 3.3. The signal enters via the input port and is then divided to the transmission port and the coupled port. The isolated port is necessary due to that a load has to be added, in many cases this load is integrated, hence no port is needed. The load is usually set to 50 ohms which is standard for most cables and connectors [6].

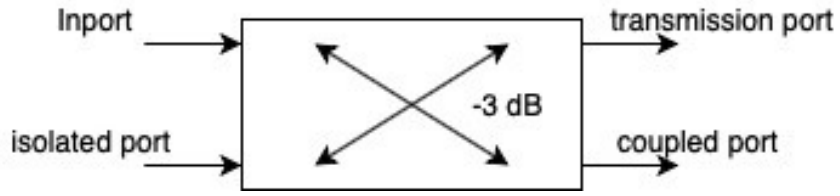


Figure 3.3: The 3dB coupler symbol and port visualisation.

3.3 Network Analyzer

A network analyzer is a tool commonly used when analysing a RF network. The analyzer has many usable features, such as signal generation and tools for analysing. By sweeping a certain bandwidth, the network analyser can do fourier transform analysis and generate s-parameters among many other functions. In the compact range a network analyser is the core of the entire measurement. It is programmed to generate RF-signals alongside with sampling data received or transmitted from the DUT, which is triggered by the controller that operates the servo motors [5].

3.4 Mechanical components

In this section various mechanical components will be addressed.

3.5 Servo controller

A Servo controller is the over all organizer in a closed loop for regulation. By receiving feedback information from the loop, the controller can adjust position, speed, acceleration among other properties with great accuracy. The controllers seen in figure 3.4 are used at the compact range. They are equipped with multi-axes control functions, and they can control up to four axes. In this facility the controllers are responsible for regulating axes and its related properties, polarization and switches [7].

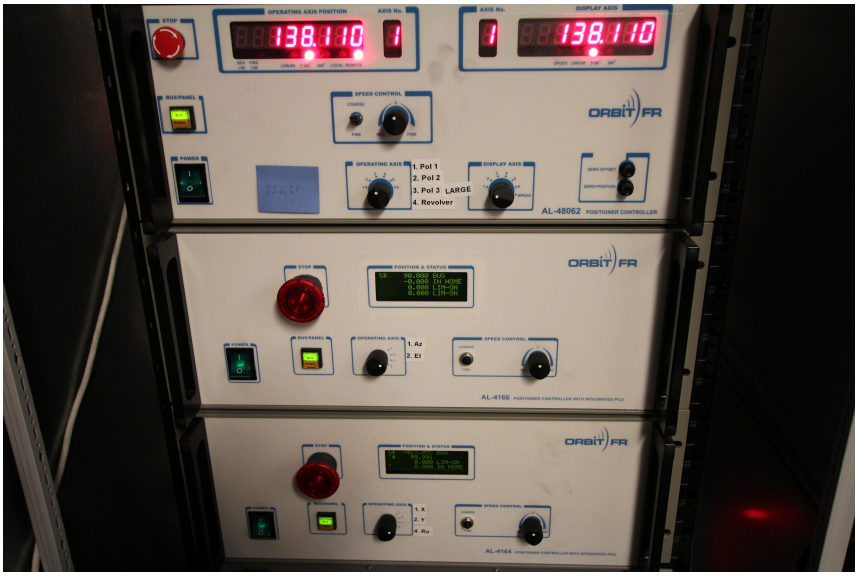


Figure 3.4: Servo Controllers from the manufacturer Orbit.

3.5.1 Encoder

An encoder is a device used to send feedback to the servo controller and can handle multiple applications such as position, direction, speed and counts. The encoder is located in direct contact with the servo motor. There are two different types of encoders: absolute and incremental encoders. The encoders used in the compact antenna test range are of the incremental type. In the test range there are two encoders that are linear and three that are rotating. The linear encoders are used to measure the position of the X and Y axis, in other words the distance. The rotary encoders are used to measure the position of the azimuth, elevation and roll axis which are angels and there for represented in degrees. The linear encoders have a unit factor of 1000, which means that when a new position command is sent, for example, to the X axis, the position value must be multiplied by 1000, which is equivalent to one millimeter in order for the program to interpret the current position. The same principle applies to the remaining three axes, however, with a unit factor that is 18204, equal to one degree since these are of the rotating type [7].

3.5.2 Servo Motor

A servo motor is a motor in an enclosed loop. The choice of motor varies and depends on what the loop regulates. The motors used in the compact range also varies, the Compact antenna test range, CATR supports many different types of electrical motors. A selection of options are presented below [8].

- *DC-motor.* In order for this motor to function, it is important that the current vector is orthogonal to the magnetic field. This must be the case at all time, otherwise the motor will not rotate. The torque of the motor can be adjusted by controlling the applied voltage with a constant load.

- *DC-motor with brush.* This motor is a DC-motor equipped with a brush collector which is also known as commutator. This is a type of electrical switch that rotates. This rotation allows the the current alternate conduction between the rotary and the drive circuit.

- *DC-motor brushless.* This motor has the same alternate conduction properties as the DC-motor, but it is achieved electronically instead.

3.5.3 Servo Loop

In order to move the antenna around the different axes, electrical motors are incorporated to make this possible. Since it is necessary for optimal precision to make the measurements as accurate as possible, servo motors operates in a closed loop. The servo controller receives a position command from the host computer, the controller then send a signal which passes through an amplifier to the motor. The position of the motor is then registered by the encoder which reports back about the position to the controller. The controller then evaluates the position error and then adjust

accordingly until the exact position is achieved. This process is visible in figure 3.5 below [8].

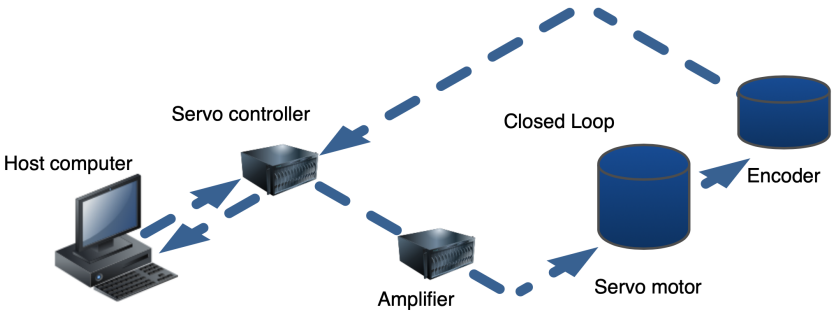


Figure 3.5: Visualization of the closed servo loop.

3.6 Switches

The compact range consist of a number of switches. In order to obtain different RF-paths, the switches is set in a corresponding way to complete the path. The switches are also responsible for selecting which channel, polarization and which port that is to be used. This as well as choosing if the DUT should use the receive or transmission mode. Most of the switches are bipolar which means the they can only be set in two ways. Some switches however, can be set in multiple ways. This is done by using a BCD-code controlled switch. This switch is controlled via a command consisting of two bits, that can either be zero or one. The switches are so called passive, which means that they can not report in which state they are set. Therefore, when the system is to be started the switches is assigned to a default mode and when the path is to be updated, all the switches must be ordered to be set to a state [9].

4

Communication

The communication from the host computer to the controller can be done in three ways. The first way is via serial communication through RS-232, the second option is parallel communication via CAN-bus. The third and last option is via a local area network, LAN with TCP/IP. These three options will be discussed in more detail in this chapter. The syntax is the same regardless of what method of communication that is used. The commands given consist of two capital letters with an additional prefix letter that indicates which axes that shall be targeted. The prefix letter can target either a single axis or a group of axes. For example the prefix letter *"A"* target all axes and prefix letter *"B"* target the first two available axes of the controller. The commands given is divided into two types: command keywords and parameter keywords. The command keywords writes values to the controller by assigning a certain parameter with a value using the allocation sign *"="*, e.g. *"XPS=16383600;"*. In this case *"X"* corresponds to the azimuth axis and *"PS"* is the position parameter. The value assigned is in the form of unit factors which needs to be scaled in order to be interpreted. A unit factor is equal to one count registered by the encoder, this is discussed under the encoder section. In the example above the X-axis is commanded to adjust its position to 90 degrees, where 1 degree is equal to 18204 unit factors, *18204*90=16383600*. The controller also need the terminator sign *";"* or *"<CR>"* to know when the command ends in order to be interpreted. The other type, parameter keywords is a read command to inform which value a parameter contain. This command is given by simply writing the prefix letter and the targeted parameter, e.g *"XPS;"*. If the command is given correctly, or incorrectly the controller will answer. The answer will always start with a *">"* followed by the value of the parameter, e.g. *">16383600"*. If the command is given incorrectly the controller will answer with *"?>"* [10].

4.1 RS-232

RS-232, or Recommended standard 232 is a serial communication method established in 1962 and has since then been extensively used through out the industry. It supports communication in both directions between the transmitter and receiver, this is known as full duplex [11]. The bits sent are sent in a single transmission line in a serial manner. RS-232 allows synchronous communication by transmitting internal clock signals and therefore the need for start and stop bits aren't demanded. However, as the distance the the data has to travel exceeds 15 meters it is recommended to switch to asynchronous mode. With a high quality cable the data can

travel up to 150 meter in asynchronous mode. The communication through RS-232 in this project is ASCII-based. The information encoded in a byte corresponds to a number between 0-255 which in turn translates to a symbol in a ASCII-table.

4.2 CAN bus

The CAN bus is a centralized communication system that allows electronic control units to communicate. It is robust and is extensively used within the car industries since the 2000's. The protocol that the CAN-bus uses consist of frames which in turn contain: start and stop, but also identifier and control bits, among other bits that allow the communication of data to be performed in a correct way. When the CAN bus is chosen as communication medium in the compact range, the host computer will send binary commands to the servo controller [12].

4.3 LAN-TCP/IP

This method of communication is done via LAN, *Local Area Network* and uses a combination of the two protocols TCP, *Transmission Control Protocol* and IP, *Internet Protocol*. The types of protocols are similar to the OSI-model which is a standard model for the structure of communication protocols [13].

The structure is organized and consist of multiple layers. TCP/IP differentiates from the OSI-model in the number of layers. The OSI-model contains seven layers, whereas the TCP/IP only contains four [14]. The four layers are the application layer, transport layer, internet layer and link layer. Here follows a simplified explanation of how it operates; the first layer, the application layer is where the data is received or transmitted between the processes that are communicating. The data is then proceeded to the transport layer which also is a part of the obtaining or transmission of data. Here the data is controlled to make sure it is intact and contains no errors. The data then continues to the internet layer which in turn translates logic addresses to physical addresses and decides which route the data is going to follow through the network. The last layer is the link layer, which is responsible for the data transfer between the units. The physical connection is also included into this layer.

5
Method

SAAB Group in Gothenburg has requested this project as a first look into if their compact antenna test range could be controlled in an alternative way, rather than how it is done currently. The multiple functions, mentioned in the introduction will all involve the use of Matlab in order to be constructed. The version of Matlab available for the the project is Matlab 2019b which is upgraded with a "Data Acquisition Toolbox". The upgrade is necessary for controlling the switches, this will be covered in more detail later in this section. These functions need to output commands and signals which can be interpreted by the control boxes, which in turn manage all motors in the measurement facility. The functions are as follows:

- *Axis control functions*
- *Polarization control function*
- *Switch control function*
- *Frequency control function*
- *Default setting script*
- *Retrieve Error message function*

The control functions will get parameter information from a Graphical User Interface, GUI where the user specifies which mode the measurement should perform (receive or transmit from the DUT, etc), polarization, frequency interval, step sizes and which plane that should be operated. All the functions should be used and accessed from the same script.

The communication from the host computer to the control boxes should preferable be done via LAN, TCP/IP. The interchange of command and parameter information function in full duplex, communication in both directions. This is true except from the configuration of the switches. They are passive and therefore can not signal their current position. The connection can also be done in RS-232 or via CAN bus, more information about these communication types can be found under the "Communication" section. During a measurement, trig signals must be generated in order to signal when the Vector Network Analyzer, VNA should sample. This means that the host computer must have a connection to the VNA as well. An other TCP/IP connection is therefore required. The trig signal is outputted when the axis, polarization and frequency has arrived to their desired position. The confirmation of these three elements restrict the speed of the measurement. The size of this time parameter can be reduced if the program can manage these in a time efficient way. When constructing the program, this will be taken in consideration. The "retrieve

error message function" is constantly active during a measurement, and it has a dominant role to interrupt if a major error has occurred as well as display relevant warnings.

As mentioned, the switches need the Matlab data acquisition toolbox in order to receive commands. This is due to the reason that the switches uses a digital I/O board to allow data interchange. In order to write data to the switches Matlab require this additional software.

The main sources of information is the user manuals to the control boxes. They contain information on how they interpret received commands, how the servo motors operates and information about the hardware. Due to policy from SAAB, all information unfortunately cannot be published. Otherwise, research in various antenna theory literature and discussions with experienced SAAB employees are contributing sources of information.

6
Literature Study

This chapter addresses the challenges and critical aspects that occur in a compact range system.

The compact range is optimized and calibrated before each measurement in order to achieve the most accurate results. The reflector which redirects the transmitted wave, is what has the most significant impact on the outcome of the measurement. The surface material and physical shape of the reflector will determine how the wave will appear after an impact with the reflector. The frequency of the wave shall also be taken in account, the higher frequency the more the surface imperfections will cause ripple. The edges of the reflector is designed to relieve the edge diffraction which otherwise interfere and distort the received wave with ripple in both phase and amplitude. The design of the reflector depend on what frequencies the CATR shall operate in. For higher frequencies the serrated edge design is most common, by tapering the edges gradually the edge diffraction is redirected away from the propagating wave as much as possible and by doing so, destructive and constructive interference with the diffracted waves can be avoided. The most plane portion of a reflected wave, the quiet zone will also be subjected to imperfections. These imperfections will exhibit in the form of phase errors, ripple and variation in amplitude. However, design and optimization of conditions regarding the facility can minimize these issues. First of all, wall reflections can be avoided by using absorbing material which will redirect and absorb unwanted noise which otherwise would decrease the SNR, *signal to noise ratio*. The distance from the source to the quiet zone is constant, however the distance from the source to various points on the reflector varies. During this distance, space attenuation of the emitted wave will directly cause amplitude taper in the quiet zone. The wave emitted from the feed has a spherical appearance and the space attenuation is increasing the bigger radius the sphere has. This can be seen in the following equation: $1/r^2$.

The overall shape of the facility will affect the results of measurements as well as what measurements that is optimal. A rectangular chamber, more common for higher frequencies is simulating free space and increases the volume of the quiet zone as much as possible. The pattern and placement of the feed as well as what frequency the measurements should act upon is important when designing the facility. This is the type of chamber used in A15. A tapered chamber, shaped as a pyramidal horn is more commonly used for lower frequencies. By positioning the feed antenna at the narrowest section of the chamber in a ideal spot, known as apex the specular reflections will occurs close to the antenna will not significantly differ in

phase from the original wave. Therefore, the reflected waves and the original wave constructively interfere and the vectors can be added. For this method to work, the source antenna has to be accurately placed in the apex which at higher frequencies will be more complex to accomplish.
During a measurement, the precision of how the axes moves will affect the results, as well as the time consumption. To be able to receive the wave from all angles possible, the axis has to be able to be controlled with detailed precision. This demands a detailed encoder, which makes it a utmost critical component. The configuration of position will affect both how accurate the positioning is and the speed for each movement. The axes can be set with a precision of 0.0001 degrees which causes a complication which imply an increased time consumption for the time it takes for the axis to adjust to a new value. The precision can be lowered to obtain a faster positioning. These two parameter has to be balanced to fit the desired outcome and adjusted corresponding to which measurement that is performed. Unwanted disturbance in form of leakage from components in the compact range is necessary to be isolated to not interfere with the measurement. To guarantee an optimal CATR performance, it is necessary to overlook the different components and perform maintenance routinely.The trigger signals which orders the network analyzer to sample has to be timed correctly, thus the measurement will consume less time, avoid time offset between samples and reassure the position, frequency and polarization has been set to the correct values. By optimizing these settings, a faster measurement can be possible. All the issues mentioned in this paragraph has not been addressed in the program, such as reflector properties and disturbance since these are physical properties and cannot be affected by a software. However, the precision of the axis movements as well as the trigger signals is incorporated in the program, but can be refined to guarantee optimization in respect to time consumption.

An important aspect that may not be the most obvious but an important one is the human factor. This is because the individuals performing the measurements must adjust the feeding horn as well as calibrating the standard gain horn with a spirit level. The position of these two horns must be set with high precision, to guarantee that both polarisation's are matched. Before each measurement, a verification of the polarisation from the feed horn is done, then when the DUT is calibrated the user ensure that the antenna has the corresponding roll angle. It is also important to direct the antenna strictly towards the reflector to receive the utmost maximum of effect. When a measurement is in progress there exist different potential sources of error, e.g. the cables and connectors have detached due to not been tightened enough with the use of torque wrenches. If the data does not match the expected results, the measurement must be restarted or complemented which result in unnecessary time delay. When engineers perform measurements on various antennas, they should have knowledge about the behavior and function of the antenna being tested. This to be able to verify if the result of a measurement matches the expected outcome. Knowledge about the whole system is also important in order to be able to simplifying the troubleshooting of errors that might occur.

7

Discussion and Conclusion

During the entire project there has been an imminent sense of time pressure due to limited accessibility of the compact range. Since SAAB priorities testing of their antennas and leasing of the facility to external antenna manufacturers there has been only a few days of for testing and verification. This has unfortunately not been enough and the development of the program as well as communication setup has therefore suffered. The limited time that was spent at the compact range unfortunately went to troubleshooting the communication method between host computer and the controllers. The preferred communication method TCP/IP turned out to be difficult to implement due to that Matlab requested an additional toolbox, *instrument control toolbox* which would imply an increased cost that would exceed the set budget. Attempts to circumvent this problem were not successful and hence RS-232 was implemented instead. RS-232 did work out and all axes could now be controlled. Unfortunately this communication method implied the disadvantage that only one controller could be accessed at a time. To fix this, multiple cables will be needed in order to obtain a fully automated measurement. The script functions for controlling polarization and switches were not tested either.

Due to the limited access, the most time of the project has been spent researching the theoretical part of how the controllers operate and how the entire compact range functions. The manuals and literature associated with the controllers are vast and complex. The amount of detail made it difficult to grasp what the essential and fundamental functions of the range is. Thanks to Anders Järnberg, employee at Orbit (the manufacturer of the entire compact range) some fundamental questions could be answered. Discussion with an other SAAB employee, Lars Mellberg who is performing a similar project to ours at another of SAAB's compact ranges also lead to some clarity. His project however is constructed in the program LabVIEW, which according to his progress is more smooth to integrate with the control boxes. In hindsight, LabVIEW may have been the better alternative to achieve the set task.

A prototype of the final script containing the desired functions has been created but has not been able to be put to a test. The program would not be able to perform a non supervised automated measurement in the created Matlab program yet, however it contains the necessary elements for a measurement. To compare the current control software, Active Cell to the new improved Matlab script, there are some obvious statements that can be mentioned. To start with, an improved user interface has been developed with properties which makes it clear for the user which mode the DUT is operating, which were a confusion element with Active Cell.

This would make it easier for users which are not familiar with measurements at the compact range. The Matlab program has the potential to achieve fully automated measurements of the modern more complex antennas which would eliminate the unnecessary need to monitor the measurement. In this regard, the use of Active Cell will no longer be needed. One of the problems with Active Cell was that it was not possible to add new functions that had made it easier to perform a new kind of measurement, which is one of the main reasons why SAAB wants to replace the software. When the matlab program is complete, the use of multiple software will no longer be needed and with the help of matlab it will be easier to write new features that can handle more advanced measurements.

If more time were at disposal, several improvements could be implemented, with exception from the obvious that the current script could be tested and improved accordingly to completion. Mainly, improvements could be made to establish a connection with TCP/IP which would make fully automated measurements possible without the need of additional cables. Elsewise, improvements could be made to the GUI, which would make it easier for the user to precise parameter values and options for a specific measurement. The program could also be made to be more simple to implement on other compact ranges.

As mentioned before, the switches are passive components and can not report their current state. This has the potential to be a future issue, since difficulties might occur when trying to verify if the switches has attained the right state after a command is received. A possible solution to this problem can be to give the command to the switches to be set corresponding to a certain system mode. Then send a signal that passes through the RF-path and investigate if a signal is received on the other end.

8

Appendix: Theory

When a measurement is performed at the CATR, there are certain formulas and theoretical knowledge that needs to be applied in order to achieve a reliable result. The compact range is designed to minimize disturbances to a great extent, but approximation still has to be made in some areas. In the following section, the fundamental theory of how various antennas is measured will be addressed.

8.1 Electric Field

An electric field arises when there is a potential difference between two different conductors. The most common example of this is a capacitor where an electric field occurs between the plates [5]. The strength of the electric field can be calculated by the following formula:

$$E = \frac{q}{4\pi\varepsilon d^2} \tag{8.1}$$

E = Electric field (Ampere/m)
q = (Coulomb, C)
$\varepsilon = (Farad/m)$
$d = Distance\ (m)$

8.2 Magnetic Field

An antenna is a type of electromagnet. A magnetic field occurs around a conductor as current passes through it. To calculate or measure the magnetic field, it is generally assumed that the magnetic field acts as individual lines of force. It is in this way that the magnetic field is represented in most antennas [5]. The magnetic field does not have an important role in the measurements and calculations executed at the compact range part from inducing current. The direction and strength of the magnetic field depends on the size and direction of the current and can be calculated with the following formula:

$$H = \frac{I}{2d\pi} \tag{8.2}$$

H = Magnetic field (Ampere/m)
I = Current (Ampere)

d = Distance (m)

8.3 Far-Field Distance

The far-field distance is defined as the distance it takes for an antenna's emitting of spherical waves to be transformed into plane waves. When calculating the distance, one must know the opening range of the antenna due to that the distance of the far-field depends on the maximum dimensions of the antenna [4]. To calculate the far-field distance, the dimensions of the antenna must first be defined, this is known as the letter D.

$$R = \frac{2D^2}{\lambda} \tag{8.3}$$

R = Far-Field distance (m)
$\lambda = Wavelength\ (m)$
$D = Distance\ (m)$

8.4 Radiation Patterns

All antennas have radiation characteristics that are emitted when the antenna transmits. The radiation properties which are of interest include polarization, phase, radiation intensity, power flow density and field strength. The radiation pattern is defined as a mathematical formula and is represented graphically by space coordinates and when studying the radiation pattern it is desirable to see how the radiation pattern appears in the far-field region [2].

8.5 Friis Transmission Equation

The equation describes how the main parameters relate to each other, such as how the received power relates to transmitted power. But it also describes the gain of an antenna as well as its range and frequency [2].

$$P_D = \frac{P_t}{4\pi R^2} \tag{8.4}$$

$P_D = Isotropic\ Antenna\ (W/m^2)$
$R = Antenna\ separation\ (m)$
$P_t = Transmitted\ power\ (W)$

8.6 Polarization

Polarization is a concept that deals with how the electric and magnetic waves propagate in relation to the earth. As an electromagnetic wave propagates perpendicular to the earth, the wave is considered to be vertically polarized. This is the same in the case where the wave propagates horizontally with the earth, which results in the wave being horizontally polarized.

There are also antennas that have a mix of both and this is called circular polarization, with the result that the magnetic and electric waves rotate as they are sent out from the antenna. With this, it causes two different circular polarization's due to that they can rotate either clockwise or counterclockwise. For the transmission to be as optimal as possible, both the transmitting antenna and the receiving antenna must have the same polarization. However, as the transmission operates over a longer distance, the polarization can change and this has to do with the propagation effects. It should be noted that a vertical or horizontal antenna can receive signals from an antenna having the circular polarization but this results in the signal not being as strong [2].

8.7 Gain

One of the most important aspects of the antenna is how much gain it can produce. There are a number of different techniques for measuring gain, the most common techniques include realized gain measurement and gain transfer measurement. Both of these measurements are based on Friis transmission formula, which can be seen in equation 5.4. What is important to keep in mind is that the two antennas are spaced apart with the distance R and that the far-field criteria is met [2].

8.7.1 Realized Gain Measurement

When two antennas are placed in a free space the following method can be utilized to calculate the unknown gain for an antenna. The wavelength, distance between antennas and the ratio between received power and transmitted power are measured. With the obtained results, a gain can be calculated with Friis transmission equation [2]. However, the equation must be rewritten to a logarithmic scale according to the following equation:

$$(G_{0t}) + (G_{0r}) = 20log_{10}(\frac{4\pi R}{\lambda}) + 10log_{10}(\frac{P_r}{P_t}) \tag{8.5}$$

$G_{0t} = Gain\ of\ the\ transmitting\ antenna\ (dB)$
$G_{0r} = Gain\ of\ the\ receiving\ antenna\ (dB)$
$P_r = Received\ power\ (W)$
$P_t = Transmitted\ power\ (W)$
$R = Antenna\ separation\ (m)$

$\lambda = Wavelength\ (m)$

8.7.2 Gain Transfer Measurement

The Gain transfer method is the most common of the two methods due to that it involves the usage of a compact range instead of a free space, which is more common now a days. The method is based on having an antenna with a known gain (also known as standard gain), which is then used to compare with the antenna that has an unknown gain. This method is divided into two parts. First, a measurement of how much power the test antenna receives when it has a matched load is performed. On the second part, the test antenna is replaced with an antenna with a known gain and the received power is also measured when the load is matched. During the gain transfer measurement the only parameter that is deviant from the two parts is the replacement of antennas, this help minimize potential error sources and makes the measurement more accurate [2].

Despite of which of the two methods, free space or measurement with reflector in a compact range that is chosen, a final equation can be used. The equation below, 6.6 is a reduced form of equation 6.5 that will specify the unknown gain of the test antenna.

$$G_T = G_S + 10log_{10}(\frac{P_T}{P_S}) \tag{8.6}$$

$G_T = Gain\ of\ the\ test\ antenna\ (dB)$
$G_S = Standard\ Gain\ (dB)$
$P_T = Received\ power\ from\ test\ antenna\ (W)$
$P_S = Received\ power\ from\ standard\ gain\ antenna\ (W)$

Bibliography

[1] ORBIT/FR, "Advanced Technology Multi-Axes Servo Controller Hardware Interfaces User's Manual" unpublished, Accessed on: Feb 05, 2020.

[2] Constantine A. Balanis, "Antenna Theory : Analysis and Design", John Wiley Sons, Incorporated, 2016, [Online]
Available: https://ebookcentral.proquest.com/lib/chalmers/reader.action?docID=4205879, Accessed on: May 17, 2020.

[3] Parini, Clive Gregson, Stuart McCormick, John van Rensburg, Daniël Janse. "Theory and Practice of Modern Antenna Range Measurements" Introduction. Institution of Engineering and Technology (2015). [Online]. Available: https://app.knovel.com/hotlink/pdf/id:kt010SO1W4/theory-practice-modern/compact-ra-introduction, Accessed on: May 12, 2020.

[4] David M. Pozar, Book title: Microwave and RF Design of Wireless Systems. 1th ed. New York : Wiley, cop. 2001 ,United States of America, Accessed on: Apr 29, 2020.

[5] Louis E. Frenzel Jr. "Principles of Electronic Communication Systems", McGraw-Hill Education, 2016, United States of America,
[Online] Available: http://e4uhu.com/down/communication%20electronics/Book%204th.pdf, Accessed on: May, 9, 2020.

[6] Wikipedia, "Power dividers and directional couplers," 2020. [Online]. Available: https://en.wikipedia.org/wiki/Power_dividers_and_directional_couplers, Accessed on: Apr, 7, 2020.

[7] Control Robotics Solutions, "SC-AT Family, Advanced Multi-Axes Servo Controller, Script Programming Language and the, Integrated Development Environment, User's Manual" unpublished, Feb 05, 2020.

[8] ORBIT/FR, "AL-4164-4MC-BL, Intergated Controller And Power Control Unit, User Manual" unpublished, Feb 05, 2020.

[9] ORBIT/FR, "MUC9258-4-J-001-F RF Design CDR", unpublished, Feb 05, 2020.

[10] Control Robotics Solutions, “SC-AT Family, Advanced Multi-Axes Servo Controllers, Communication Protocols User’s Manual”, unpublished, Feb 05, 2020.

[11] ARC Electronics, "RS232 Data Interface" 2015, [Online]. Available: https://web.archive.org/web/20151113104839/http://www.arcelect.com/rs232.htm, Accessed on: May 20, 2020.

[12] CSS Electronics "CAN Bus Explained - A Simple Intro", 2020, [Online]. Available: https://www.csselectronics.com/screen/page/simple-intro-to-can-bus/language/en, Accessed on: May 20, 2020.

[13] Wikipedia, "OSI-Modellen", 2020 [Online]. Available: https://sv.wikipedia.or, Accessed on: May 20, 2020.

[14] Wikipedia, "TCP/IP", 2018 [Online]. Available: https://sv.wikipedia.org/wiki/TCP/IP Accessed on: May 20, 2020.

[Titlepage image] "A15 Antenna Measurements" 2007. [Electronic image]. Available: https://saab.com/a15_antenna_measurements, Accessed on: May 09, 2020.

9
Matlab Code

```
function [Done1,Done2,Done3] = Axis_Init(In1)
%AXIS_INIT AL-4164 (X = X-Axis, Y= Y-Axis, W = Roll)
%          AL-4166 (X = Azimuth, Y = Elevation)
% assigning default values for axis operation, the prefix "A" is allowing
% all axis of the controller to be written to at once.

s1 = serial('COM4','BaudRate',115200,'Terminator','CR');
fopen(s1)
Done1 = 0;
Done2 = 0;
Done3 = 0;
for il = 1:numel(In1)
    switch In1{il}.Name
                    case 'Azimuth'
                    fprintf(s1,'XMO=1;'); %Motor On
                    fprintf(s1,'XMM=0;XSM=0;'); %Set Normal Point To Point Motion Mode
                    fprintf(s1,'XAC=500000;'); %Acceleration
                    fprintf(s1,'XDC=500000;'); %Deceleration
                    fprintf(s1,'XDL=1000000;');%Limit Deceleration
                    fprintf(s1,'XWW=0;'); %Profiler Smooth factor
                    fprintf(s1,'XSP=5000;');%Speed
                    case 'Roll'
                    fprintf(s2,'WMO=1;');%Motor On
                    fprintf(s2,'WMM=0;WSM=0;');%Set Normal Point To Point Motion Mode
                    fprintf(s2,'WAC=500000;');%Acceleration
                    fprintf(s2,'WDC=500000;');%Deceleration
                    fprintf(s2,'WDL=1000000;');%Limit Deceleration
                    fprintf(s2,'WWW=0;');%Profiler Smooth factor
                    fprintf(s2,'WSP=5000;');%Speed
                    case 'Elevation'
                    fprintf(s1,'YMO=1;');%Motor On
                    fprintf(s1,'YMM=0;YSM=0;');%Set Normal Point To Point Motion Mode
                    fprintf(s1,'YAC=500000;');%Acceleration
                    fprintf(s1,'YDC=500000;');%Deceleration
                    fprintf(s1,'YDL=1000000;');%Limit Deceleration
                    fprintf(s1,'YWW=0;');%Profiler Smooth factor
                    fprintf(s1,'YSP=5000;');%Speed
                    case 'X'
                    fprintf(s2,'XMO=1;');%Motor On
                    fprintf(s2,'XMM=0;XSM=0;');%Set Normal Point To Point Motion Mode
                    fprintf(s2,'XAC=500000;');%Acceleration
                    fprintf(s2,'XDC=500000;');%Deceleration
                    fprintf(s2,'XDL=1000000;');%Limit Deceleration
                    fprintf(s2,'XWW=0;');%Profiler Smooth factor
                    fprintf(s2,'XSP=5000;');%Speed
                    case 'Y'
                    fprintf(s2,'YMO=1;');%Motor On
                    fprintf(s2,'YMM=0;YSM=0;');%Set Normal Point To Point Motion Mode
                    fprintf(s2,'YAC=500000;');%Acceleration
                    fprintf(s2,'YDC=500000;');%Deceleration
                    fprintf(s2,'YDL=1000000;');%Limit Deceleration
                    fprintf(s2,'YWW=0;');%Profiler Smooth factor
                    fprintf(s2,'YSP=5000;');%Speed
    end
```

Figure 9.1: Matlab code for initialize axis

```
function [Output,Done] = X_Move(inl)
%X_MOVE
%Inserts the next value to be written to the control,
%reads the error and checks that the axis has arrived at its value.
%It has two output signals that control that everything is completed.

Stop = 0; % Initial Value
Done = 0; % Initial Value

fprintf(sl, 'XPE;'); %X = X axis, PE = Position Error
pe=fscanf(sl, '%s');
PE=(str2double(regexp(pe,'\d+\.?\d+','match')))/1000; %Only returns the number

fprintf(sl, 'XPS;'); %X = X axis, PS = Position Status
ps=fscanf(sl, '%s');
PS=str2double(regexp(ps,'\d+\.?\d+','match')); %Only returns the number

if ~(PS == inl)
fprintf(sl,'XAP= %d;',inl); %Starting Motion
Slask = fscanf(sl, '%s');
fprintf(sl,'XBG;'); % Begins Motion
Slask = fscanf(sl, '%s');

while ~((-3<PE) && (PE<3) && (Stop==1))
      fprintf(sl,'XPE;'); % Checking Position error
      pe=fscanf(sl,'%s');
      PE = (str2double(regexp(pe,'\d+\.?\d+','match')))/1000; %Delets all except the number
      fprintf(sl,'XMS[1];'); % Checking Motor fault
      ms=fscanf(sl,'%s');
      Stop = str2double(regexp(ms,'\d+\.?\d+','match'));%Delets all except the number
            if (-5<PE) && (PE<5) %Interval to continue
               fprintf(sl,'XPS;'); %Position Status
               out=fscanf(sl,'%s');
               Output = str2double(regexp(out,'\d+\.?\d+','match'));%Delets all except the number
               Done = 1;
               break
            end
end
end
```

Figure 9.2: Matlab code for controlling X axis

```
function [Output,Done] = Y_Move(inl)
%Y_MOVE
%Inserts the next value to be written to the control,
%reads the error and checks that the axis has arrived at its value.
%It has two output signals that control that everything is completed.

Stop = 0; % Initial Value
Done = 0; % Initial Value

fprintf(sl, 'YPE;'); %Y = Y axis, PE = Position Error
pe=fscanf(sl, '%s');
PE=(str2double(regexp(pe,'\d+\.?\d+','match')))/1000; %Only returns the number

fprintf(sl, 'YPS;'); %Y = Y axis, PS = Position Status
ps=fscanf(sl, '%s');
PS=str2double(regexp(ps,'\d+\.?\d+','match')); %Only returns the number

if ~(PS == inl)
fprintf(sl,'YAP= %d;',inl); %Starting Motion
Slask = fscanf(sl, '%s');
fprintf(sl,'YBG;'); % Begins Motion
Slask = fscanf(sl, '%s');

while ~((-3<PE) && (PE<3) && (Stop==1))
      fprintf(sl,'YPE;'); % Checking Position error
      pe=fscanf(sl,'%s');
      PE = (str2double(regexp(pe,'\d+\.?\d+','match')))/1000; %Delets all except the number
      fprintf(sl,'YMS[1];'); % Checking Motor fault
      ms=fscanf(sl,'%s');
      Stop = str2double(regexp(ms,'\d+\.?\d+','match'));%Delets all except the number
            if (-5<PE) && (PE<5) %Interval to continue
               fprintf(sl,'YPS;'); %Position Status
               out=fscanf(sl,'%s');
               Output = str2double(regexp(out,'\d+\.?\d+','match'));%Delets all except the number
               Done = 1;
               break
            end
end
end
```

Figure 9.3: Matlab code for controlling Y axis

```
function [Output,Done] = Azimuth_Move(in1)
%AZIMUTH_MOVE
%Inserts the next value to be written to the control,
%reads the error and checks that the axis has arrived at its value.
%It has two output signals that control that everything is completed.

Stop = 0; % Initial Value
Done = 0; % Initial Value

fprintf(sl, 'XPE;'); %X = Azimuth axis, PE = Position Error
pe=fscanf(sl, '%s');
PE=(str2double(regexp(pe,'\d+\.?\d+','match')))/18204; %Only returns the number

fprintf(sl, 'XPS;'); %X = Azimuth axis, PS = Position Status
ps=fscanf(sl, '%s');
PS=str2double(regexp(ps,'\d+\.?\d+','match')); %Only returns the number

if ~(PS == in1)
fprintf(sl,'XAP= %d;',in1); %Starting Motion
Slask = fscanf(sl, '%s');
fprintf(sl,'XBG;'); % Begins Motion
Slask = fscanf(sl, '%s');

while ~((-3<PE) && (PE<3) && (Stop==1))
      fprintf(sl,'XPE;'); % Checking Position error
      pe=fscanf(sl,'%s');
      PE = (str2double(regexp(pe,'\d+\.?\d+','match')))/18204; %Delets all except the number
      fprintf(sl,'XMS[1];'); % Checking Motor fault
      ms=fscanf(sl,'%s');
      Stop = str2double(regexp(ms,'\d+\.?\d+','match'));%Delets all except the number
            if (-5<PE) && (PE<5) %Interval to continue
               fprintf(sl,'XPS;'); %Position Status
               out=fscanf(sl,'%s');
               Output = str2double(regexp(out,'\d+\.?\d+','match'));%Delets all except the number
               Done = 1;
               break
            end
end
end
```

Figure 9.4: Matlab code for controlling Azimuth axis

```
function [Output,Done] = Elevation_Move(in1)
%ELEVATION_MOVE
%Inserts the next value to be written to the control,
%reads the error and checks that the axis has arrived at its value.
%It has two output signals that control that everything is completed.

Stop = 0; % Initial Value
Done = 0; % Initial Value

fprintf(s1, 'YPE;'); %Y = Elevation axis, PE = Position Error
pe=fscanf(s1, '%s');
PE=(str2double(regexp(pe,'\d+\.?\d+','match')))/18204; %Only returns the number

fprintf(s1, 'YPS;'); %Y = Elevation axis, PS = Position Status
ps=fscanf(s1, '%s');
PS=str2double(regexp(ps,'\d+\.?\d+','match')); %Only returns the number

if ~(PS == in1)
fprintf(s1,'YAP= %d;',in1); %Starting Motion
Slask = fscanf(s1, '%s');
fprintf(s1,'YBG;'); % Begins Motion
Slask = fscanf(s1, '%s');

while ~((-3<PE) && (PE<3) && (Stop==1))
      fprintf(s1,'YPE;'); % Checking Position error
      pe=fscanf(s1,'%s');
      PE = (str2double(regexp(pe,'\d+\.?\d+','match')))/18204; %Delets all except the number
      fprintf(s1,'YMS[1];'); % Checking Motor fault
      ms=fscanf(s1,'%s');
      Stop = str2double(regexp(ms,'\d+\.?\d+','match'));%Delets all except the number
            if (-5<PE) && (PE<5) %Interval to continue
               fprintf(s1,'YPS;'); %Position Status
               out=fscanf(s1,'%s');
               Output = str2double(regexp(out,'\d+\.?\d+','match'));%Delets all except the number
               Done = 1;
               break
            end
end
end

```

Figure 9.5: Matlab code for controlling Elevation axis

```
function [Output,Done] = Roll_Move(in1)
%Roll_MOVE
%Inserts the next value to be written to the control,
%reads the error and checks that the axis has arrived at its value.
%It has two output signals that control that everything is completed.

Stop = 0; % Initial Value
Done = 0; % Initial Value

fprintf(sl, 'WPE;'); %W = Roll axis, PE = Position Error
pe=fscanf(sl, '%s');
PE=(str2double(regexp(pe,'\d+\.?\d+','match')))/18204; %Only returns the number

fprintf(sl, 'WPS;'); %Y = Roll, PS = Position Status
ps=fscanf(sl, '%s');
PS=str2double(regexp(ps,'\d+\.?\d+','match')); %Only returns the number

if ~(PS == in1)
fprintf(sl,'WAP= %d;',in1); %Starting Motion
Slask = fscanf(sl, '%s');
fprintf(sl,'WBG;'); % Begins Motion
Slask = fscanf(sl, '%s');

while ~((-3<PE) && (PE<3) && (Stop==1))
      fprintf(sl,'WPE;'); % Checking Position error
      pe=fscanf(sl,'%s');
      PE = (str2double(regexp(pe,'\d+\.?\d+','match')))/18204; %Delets all except the number
      fprintf(sl,'WMS[1];'); % Checking Motor fault
      ms=fscanf(sl,'%s');
      Stop = str2double(regexp(ms,'\d+\.?\d+','match'));%Delets all except the number
            if (-5<PE) && (PE<5) %Interval to continue
               fprintf(sl,'WPS;'); %Position Status
               out=fscanf(sl,'%s');
               Output = str2double(regexp(out,'\d+\.?\d+','match'));%Delets all except the number
               Done = 1;
               break
            end
end
end
```

Figure 9.6: Matlab code for controlling Roll axis

```
function [] = End_Motion(Inl)
%END_MOTION Describing the cause for last end of motion
% A describing why the motor was latly disabled
%          AL-4164 (X = X-Axis,  Y = Y-Axis, W = Roll)
%          AL-4166 (X = Azimuth, Y = Elevation)

for il = 1:numel(Inl)
    switch Inl{il}.Name
%--------------------------------------------------------------------------
case 'Azimuth'
     fprintf(sl, 'XEM;');% X = Azimuth Axis, EM = End Motion
     em=fscanf(sl, '%s');
     EM=str2double(regexp(em,'\d+\.?\d+','match')); %Only returns the number
        switch EM
            case '0'
            fprintf('Azimuth Axis, Motor is still active.');
            case '1'
            fprintf('Azimuth Axis, Normal end of motion.');
            case '2'
            fprintf('Azimuth Axis, Forward limit switch (FLS).');
            case '3'
            fprintf('Azimuth Axis, Reverse limit switch (RLS).');
            case '4'
            fprintf('Azimuth Axis, High software limit (PS>HL).');
            case '5'
            fprintf('Azimuth Axis, Low software limit (PS<LL).');
            case '6'
            fprintf('Azimuth Axis, Motor was disabled (Check MF).');
            case '7'
            fprintf('Azimuth Axis, user command (ST or AB).');
            case '8'
            fprintf('Azimuth Axis, Motor off by user (MO=0).');
        end
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
case 'Roll'
      fprintf(sl, 'WEM;'); %W = Roll Axis, EM = End Motion
      em=fscanf(sl, '%s');
      EM=str2double(regexp(em,'\d+\.?\d+','match')); %Only returns the number
        switch EM
            case '0'
            fprintf('Roll Axis, Motor is still active.');
            case '1'
            fprintf('Roll Axis, Normal end of motion.');
            case '2'
            fprintf('Roll Axis, Forward limit switch (FLS).');
            case '3'
            fprintf('Roll Axis, Reverse limit switch (RLS).');
            case '4'
            fprintf('Roll Axis, High software limit (PS>HL).');
            case '5'
            fprintf('Roll Axis, Low software limit (PS<LL).');
            case '6'
            fprintf('Roll Axis, Motor was disabled (Check MF).');
            case '7'
            fprintf('Roll Axis, user command (ST or AB).');
            case '8'
            fprintf('Roll Axis, Motor off by user (MO=0).');
        end
%--------------------------------------------------------------------------
```

```
case 'Elevation'
      fprintf(sl, 'YEM;'); %Y = Elevation Axis, EM = End Motion
      em=fscanf(sl, '%s');
      EM=str2double(regexp(em,'\d+\.?\d+','match')); %Only returns the number
        switch EM
            case '0'
            fprintf('Elevation Axis, Motor is still active.');
            case '1'
            fprintf('Elevation Axis, Normal end of motion.');
            case '2'
            fprintf('Elevation Axis, Forward limit switch (FLS).');
            case '3'
            fprintf('Elevation Axis, Reverse limit switch (RLS).');
            case '4'
            fprintf('Elevation Axis, High software limit (PS>HL).');
            case '5'
            fprintf('Elevation Axis, Low software limit (PS<LL).');
            case '6'
            fprintf('Elevation Axis, Motor was disabled (Check MF).');
            case '7'
            fprintf('Elevation Axis, user command (ST or AB).');
            case '8'
            fprintf('Elevation Axis, Motor off by user (MO=0).');
        end
%-------------------------------------------------------------------------
case 'X'
      fprintf(sl, 'XEM;'); %X = X Axis, EM = End Motion
      em=fscanf(sl, '%s');
      EM=str2double(regexp(em,'\d+\.?\d+','match')); %Only returns the number
        switch EM
            case '0'
            fprintf('X Axis, Motor is still active.');
            case '1'
            fprintf('X Axis, Normal end of motion.');
            case '2'
            fprintf('X Axis, Forward limit switch (FLS).');
            case '3'
            fprintf('X Axis, Reverse limit switch (RLS).');
            case '4'
            fprintf('X Axis, High software limit (PS>HL).');
            case '5'
            fprintf('X Axis, Low software limit (PS<LL).');
            case '6'
            fprintf('X Axis, Motor was disabled (Check MF).');
            case '7'
            fprintf('X Axis, user command (ST or AB).');
            case '8'
            fprintf('X Axis, Motor off by user (MO=0).');
        end
```

```
110 -    case 'Y'
111 -          fprintf(sl, 'YEM;'); %Y = Y Axis, EM = End Motion
112 -          em=fscanf(sl, '%s');
113 -          EM=str2double(regexp(em,'\d+\.?\d+','match')); %Only returns the number
114 -            switch EM
115 -                case '0'
116 -                fprintf('Y Axis, Motor is still active.');
117 -                case '1'
118 -                fprintf('Y Axis, Normal end of motion.');
119 -                case '2'
120 -                fprintf('Y Axis, Forward limit switch (FLS).');
121 -                case '3'
122 -                fprintf('Y Axis, Reverse limit switch (RLS).');
123 -                case '4'
124 -                fprintf('Y Axis, High software limit (PS>HL).');
125 -                case '5'
126 -                fprintf('Y Axis, Low software limit (PS<LL).');
127 -                case '6'
128 -                fprintf('Y Axis, Motor was disabled (Check MF).');
129 -                case '7'
130 -                fprintf('Y Axis, user command (ST or AB).');
131 -                case '8'
132 -                fprintf('Y Axis, Motor off by user (MO=0).');
133 -            end
134 -        end
135 -     end
136
137 - end
138
```

Figure 9.7: Describing the cause for last end of motions.

```
clear
close all

%%
Workflow test

% Discover the NI card.
wbar = waitbar(0,'Discovering devices..');
dev = daq.getDevices;
vend = daq.getVendors;
if ~isnan(dev.ID(1)) && ~isnan(vend.ID(1))
    % Create a session for each channel
    waitbar(0.1,wbar,'Create session..')
    s0 = daq.createSession(vend.ID);
    s1 = daq.createSession(vend.ID);
    s2 = daq.createSession(vend.ID);
    s3 = daq.createSession(vend.ID);
    % Create channels for all the ports
    waitbar(0.2,wbar,'Create channels..')
    P0 = addDigitalChannel(s0,dev.ID,'port0/line7:0','OutputOnly');
    P1 = addDigitalChannel(s1,dev.ID,'port1/line7:0','OutputOnly');
    P2 = addDigitalChannel(s2,dev.ID,'port2/line7:0','OutputOnly');
    P3 = addDigitalChannel(s3,dev.ID,'port3/line4:0','OutputOnly');
    % Create the binary start vector for all the channels
    cValue.P0 = zeros(1,length(P0));
    cValue.P1 = zeros(1,length(P1));
    cValue.P2 = zeros(1,length(P2));
    cValue.P3 = zeros(1,length(P3));
    close(wbar);

    functionname = 'daq_Test';
    f1 = figure(999);clf
    set(999,'tag','daq_Test');
    set(999,'position','default');
    set(999,'name',functionname,'numbertitle','off',...
        'menubar','none')
    temp=get(999,'position');
    temp(4)=temp(4)/1.92;
    set(999,'position',temp);

    % System Mode
    sm_txt = uicontrol(f1,'style','text','units','normalized');
```

```
    sm_txt.Position = [.023 .85 .143 .091];
    sm_txt.String = 'System Mode:';

    dd_SM = uicontrol(f1,'style','popupmenu','units','normalized');
    dd_SM.Position = [.035 .77 .143 .091];
    dd_SM.String = ('TX_05_18','RX_05_18','RX_05_50','Bitcal_TX','Bitcal_RX','Exc_Mixer','Bypass');
    dd_SM.Callback = {@selection,dd_SM};

    % Feed Selection
    fs_txt = uicontrol(f1,'style','text','units','normalized');
    fs_txt.Position = [.172 .85 .215 .091];
    fs_txt.String = ('Feed Selection:');

    dd_FS = uicontrol(f1,'style','popupmenu','units','normalized');
    dd_FS.Position = [.215 .77 .215 .091];
    dd_FS.String = ('Station 1, TXRX','Station 1, TX High Power','Station 2','Station 3');
    dd_FS.Callback = {@selection,dd_FS};

    % Channel Selection
    cs_txt = uicontrol(f1,'style','text','units','normalized');
    cs_txt.Position = [.438 .85 .215 .091];
    cs_txt.String = ('Channel Selection:');

    dd_CS = uicontrol(f1,'style','popupmenu','units','normalized');
    dd_CS.Position = [.465 .77 .155 .091];
    dd_CS.String = ('Channel 1','Channel 2','Channel 3','Channel 4','Channel 5','Channel 6','Channel 7','Channel 8','Channel 9','Channel 10','Channel 11','Channel 12','Channel 13','Channel 14','Channel 15','Cha
    dd_CS.Callback = {@selection,dd_CS};

    % Polarization Selection
    ps_txt = uicontrol(f1,'style','text','units','normalized');
    ps_txt.Position = [.647 .85 .215 .091];
    ps_txt.String = ('Polarization Selection:');

    dd_PS = uicontrol(f1,'style','popupmenu','units','normalized');
    dd_PS.Position = [.660 .77 .143 .091];
    dd_PS.String = ('Horizontal','Vertical');
    dd_PS.Callback = {@selection,dd_PS};

    % Apply settings
    btn_appSet = uicontrol(f1,'style','pushbutton','units','normalized');
    btn_appSet.Position = [.653 .045 .161 .114];
    btn_appSet.String = 'Apply Settings';
```

```
        btn_appSet.Callback = {@applySettings_Fcn,dd_SM,dd_FS,dd_CS,dd_PS,cValue,s0,s1,s2,s3};
        % Cancel
        btn_cancel = uicontrol(f1,'style','pushbutton','units','normalized');
        btn_cancel.Position = [.857 .045 .107 .114];
        btn_cancel.String = 'Cancel';
        btn_cancel.Callback = 'close(f1)';

        % Read Current settings
        btn_appSet = uicontrol(f1,'style','pushbutton','units','normalized');
        btn_appSet.Position = [.035 .045 .200 .114];
        btn_appSet.String = 'Read Current Settings';
        btn_appSet.Callback = {@read_allNIports,cValue,s0,s1,s2,s3};
        %     cValue = Tx_05_18(cValue,s2,s3);
        %     cValue = read_allNIports(cValue,s0,s1,s2,s3);

        uiwait(f1)
        %     cValue = read_NIport(cValue,s1,1);
        % Release the sessions
        release(s0)
        release(s1)
        release(s2)
        release(s3)
    %end
    %%

function cV = Tx_05_18(cV,s2,s3)
x = 3;
dataP2 = [x x 1 x 0 1 x x];
dataP3 = [x x 1 0 1];
cV.P2 = update_bin(cV.P2,dataP2);
cV.P3 = update_bin(cV.P3,dataP3);
outputSingleScan(s2,cV.P2)
outputSingleScan(s3,cV.P3)
end
function cV = Rx_05_18(cV,s2,s3)
x = 3;
dataP2 = [0 1 0 0 0 1 1 0];
dataP3 = [x x 1 1 0];
cV.P2 = update_bin(cV.P2,dataP2);
cV.P3 = update_bin(cV.P3,dataP3);
outputSingleScan(s2,cV.P2)
outputSingleScan(s3,cV.P3)
```

```
127 -   end
128     function cV = Rx_05_50(cV,s2,s3)
129 -   x = 3;
130 -   dataP2 = [0 1 x x 1 0 0 1];
131 -   dataP3 = [x x 1 1 0];
132 -   cV.P2 = update_bin(cV.P2,dataP2);
133 -   cV.P3 = update_bin(cV.P3,dataP3);
134 -   outputSingleScan(s2,cV.P2)
135 -   outputSingleScan(s3,cV.P3)
136 -   end
137     function cV = Bitcal_Tx(cV,s2,s3)
138 -   x = 3;
139 -   dataP2 = [0 1 1 0 0 1 1 0];
140 -   dataP3 = [x x 1 x x];
141 -   cV.P2 = update_bin(cV.P2,dataP2);
142 -   cV.P3 = update_bin(cV.P3,dataP3);
143 -   outputSingleScan(s2,cV.P2)
144 -   outputSingleScan(s3,cV.P3)
145 -   end
146     function cV = Bitcal_Rx(cV,s2,s3)
147 -   x = 3;
148 -   dataP2 = [0 1 0 0 0 1 1 0];
149 -   dataP3 = [x x 1 x x];
150 -   cV.P2 = update_bin(cV.P2,dataP2);
151 -   cV.P3 = update_bin(cV.P3,dataP3);
152 -   outputSingleScan(s2,cV.P2)
153 -   outputSingleScan(s3,cV.P3)
154 -   end
155     function cV = Ext_Mixer(cV,s2,s3)
156 -   x = 3;
157 -   dataP2 = [1 0 x x x x x x];
158 -   dataP3 = [x x 1 0 x];
159 -   cV.P2 = update_bin(cV.P2,dataP2);
160 -   cV.P3 = update_bin(cV.P3,dataP3);
161 -   outputSingleScan(s2,cV.P2)
162 -   outputSingleScan(s3,cV.P3)
163 -   end
164     function cV = Bypass(cV,s2,s3)
165 -   x = 3;
166 -   dataP2 = [x x x x x x x x];
167 -   dataP3 = [x x 1 x x];
168 -   cV.P2 = update_bin(cV.P2,dataP2);
```

```
169 -     cV.P3 = update_bin(cV.P3,dataP3);
170 -     outputSingleScan(s2,cV.P2)
171 -     outputSingleScan(s3,cV.P3)
172 -     end
173       function cV = Feed_S3(cV,s3)
174 -     x = 3;
175 -     dataP3 = [0 0 1 x x];
176 -     cV.P3 = update_bin(cV.P3,dataP3);
177 -     outputSingleScan(s3,cV.P3)
178 -     end
179       function cV = Feed_S2(cV,s3)
180 -     x = 3;
181 -     dataP3 = [0 1 1 x x];
182 -     cV.P3 = update_bin(cV.P3,dataP3);
183 -     outputSingleScan(s3,cV.P3)
184 -     end
185       function cV = Feed_S1_TxRx(cV,s3)
186 -     x = 3;
187 -     dataP3 = [1 0 1 1 x];
188 -     cV.P3 = update_bin(cV.P3,dataP3);
189 -     outputSingleScan(s3,cV.P3)
190 -     end
191       function cV = Feed_S1_TxHP(cV,s3)
192 -     x = 3;
193 -     dataP3 = [1 1 1 0 x];
194 -     cV.P3 = update_bin(cV.P3,dataP3);
195 -     outputSingleScan(s3,cV.P3)
196 -     end
197       function cV = DUT_Channel(cV,s1,ch)
198 -     x = 3;
199 -     binCh = dec2bin(ch-1)-'0';
200 -     dataP1(1:8) = x;
201 -     dataP1 = [dataP1(1:end-length(binCh)) binCh];
202 -     cV.P1 = update_bin(cV.P1,dataP1);
203 -     outputSingleScan(s1,cV.P1)
204 -     end
205
206       function cV = Pol_Horizontal(cV,s0)
207 -     x = 3;
208 -     dataP0 = [x x 0 0 0 x x x]; %this is unknown
209 -     cV.P0 = update_bin(cV.P0,dataP0);
210 -     outputSingleScan(s0,cV.P0)
```

```
212     function cV = Pol_Vertical(cV,s0)
213 -   x = 3;
214 -   dataP0 = [x x 1 1 1 x x x]; %this is unknown
215 -   cV.P0 = update_bin(cV.P0,dataP0);
216 -   outputSingleScan(s0,cV.P0)
217 -   end
218
219     function updated = update_bin(org,new) % Function to update the vector
220 -   updated = org;
221 -   for i=1:length(org)
222 -       if new(i) < 2
223 -           updated(i) = new(i);
224 -       end
225 -   end
226 -   end
227
228     function applySettings_Fcn(src,~,sysMode,FeedS,ChSel,PolSel,cV,s0,s1,s2,s3)
229     % System Mode
230 -   switch sysMode.Value
231 -       case 1
232 -           val = sysMode.Value;
233 -           str = sysMode.String;
234 -           disp(['System Mode: ' str{val}]);
235 -           cV = Tx_05_18(cV,s2,s3);
236 -       case 2
237 -           val = sysMode.Value;
238 -           str = sysMode.String;
239 -           disp(['System Mode: ' str{val}]);
240 -           cV = Rx_05_18(cV,s2,s3);
241 -       case 3
242 -           val = sysMode.Value;
243 -           str = sysMode.String;
244 -           disp(['System Mode: ' str{val}]);
245 -           cV = Rx_05_50(cV,s2,s3);
246 -       case 4
247 -           val = sysMode.Value;
248 -           str = sysMode.String;
249 -           disp(['System Mode: ' str{val}]);
250 -           cV = Bitcal_Tx(cV,s2,s3);
251 -       case 5
252 -           val = sysMode.Value;
253 -           str = sysMode.String;
```

```
254 -           disp(['System Mode: ' str{val}]);
255 -           cV = Bitcal_Rx(cV,s2,s3);
256 -       case 6
257 -           val = sysMode.Value;
258 -           str = sysMode.String;
259 -           disp(['System Mode: ' str{val}]);
260 -           cV = Ext_Mixer(cV,s2,s3);
261 -       case 7
262 -           val = sysMode.Value;
263 -           str = sysMode.String;
264 -           disp(['System Mode: ' str{val}]);
265 -           cV = Bypass(cV,s2,s3);
266 -   end
267     % Feed Selection
268 -   switch FeedS.Value
269 -       case 1
270 -           val = FeedS.Value;
271 -           str = FeedS.String;
272 -           disp(['Feed: ' str{val}]);
273 -           cV = Feed_S1_TxRx(cV,s3);
274 -       case 2
275 -           val = FeedS.Value;
276 -           str = FeedS.String;
277 -           disp(['Feed: ' str{val}]);
278 -           cV = Feed_S1_TxHP(cV,s3);
279 -       case 3
280 -           val = FeedS.Value;
281 -           str = FeedS.String;
282 -           disp(['Feed: ' str{val}]);
283 -           cV = Feed_S2(cV,s3);
284 -       case 4
285 -           val = FeedS.Value;
286 -           str = FeedS.String;
287 -           disp(['Feed: ' str{val}]);
288 -           cV = Feed_S3(cV,s3);
289 -   end
290     % Channel Selection
291 -   switch ChSel.Value
292 -       case 1
293 -           val = ChSel.Value;
294 -           str = ChSel.String;
295 -           disp(['Channel: ' str{val}]);
```

```
296 -            cV = DUT_Channel(cV,sl,1);
297 -        case 2
298 -            val = ChSel.Value;
299 -            str = ChSel.String;
300 -            disp(['Channel: ' str{val}]);
301 -            cV = DUT_Channel(cV,sl,2);
302 -        case 3
303 -            val = ChSel.Value;
304 -            str = ChSel.String;
305 -            disp(['Channel: ' str{val}]);
306 -            cV = DUT_Channel(cV,sl,3);
307 -        case 4
308 -            val = ChSel.Value;
309 -            str = ChSel.String;
310 -            disp(['Channel: ' str{val}]);
311 -            cV = DUT_Channel(cV,sl,4);
312 -        case 5
313 -            val = ChSel.Value;
314 -            str = ChSel.String;
315 -            disp(['Channel: ' str{val}]);
316 -            cV = DUT_Channel(cV,sl,5);
317 -        case 6
318 -            val = ChSel.Value;
319 -            str = ChSel.String;
320 -            disp(['Channel: ' str{val}]);
321 -            cV = DUT_Channel(cV,sl,6);
322 -        case 7
323 -            val = ChSel.Value;
324 -            str = ChSel.String;
325 -            disp(['Channel: ' str{val}]);
326 -            cV = DUT_Channel(cV,sl,7);
327 -        case 8
328 -            val = ChSel.Value;
329 -            str = ChSel.String;
330 -            disp(['Channel: ' str{val}]);
331 -            cV = DUT_Channel(cV,sl,8);
332 -        case 9
333 -            val = ChSel.Value;
334 -            str = ChSel.String;
335 -            disp(['Channel: ' str{val}]);
336 -            cV = DUT_Channel(cV,sl,9);
337 -        case 10
```

```
338 -            val = ChSel.Value;
339 -            str = ChSel.String;
340 -            disp(['Channel: ' str{val}]);
341 -            cV = DUT_Channel(cV,s1,10);
342 -        case 11
343 -            val = ChSel.Value;
344 -            str = ChSel.String;
345 -            disp(['Channel: ' str{val}]);
346 -            cV = DUT_Channel(cV,s1,11);
347 -        case 12
348 -            val = ChSel.Value;
349 -            str = ChSel.String;
350 -            disp(['Channel: ' str{val}]);
351 -            cV = DUT_Channel(cV,s1,12);
352 -        case 13
353 -            val = ChSel.Value;
354 -            str = ChSel.String;
355 -            disp(['Channel: ' str{val}]);
356 -            cV = DUT_Channel(cV,s1,13);
357 -        case 14
358 -            val = ChSel.Value;
359 -            str = ChSel.String;
360 -            disp(['Channel: ' str{val}]);
361 -            cV = DUT_Channel(cV,s1,14);
362 -        case 15
363 -            val = ChSel.Value;
364 -            str = ChSel.String;
365 -            disp(['Channel: ' str{val}]);
366 -            cV = DUT_Channel(cV,s1,15);
367 -        case 16
368 -            val = ChSel.Value;
369 -            str = ChSel.String;
370 -            disp(['Channel: ' str{val}]);
371 -            cV = DUT_Channel(cV,s1,16);
372 -    end
373     % Polarization Selection
374 -    switch PolSel.Value
375 -        case 1
376 -            val = PolSel.Value;
377 -            str = PolSel.String;
378 -            disp(['Polarization: ' str{val}]);
379 -            cV = Pol_Horizontal(cV,s0);
```

```
379 -            cV = Pol_Horizontal(cV,s0);
380 -        case 2
381 -            val = PolSel.Value;
382 -            str = PolSel.String;
383 -            disp(['Polarization: ' str{val}]);
384 -            cV = Pol_Vertical(cV,s0);
385 -    end
386 -    src.UserData = cV;
387 -    disp(src.UserData)
388 -    end
```

Figure 9.8: GUI with callback functions for RF path and polarization.