Manuel Kubicka, BSc

Development of a new control software for the satellite ground station at the Lustbühel observatory

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Space Sciences and Earth from Space

submitted to

Graz University of Technology

Supervisor

Univ.-Prof. Dipl.-Ing. Dr.techn. Otto Koudelka

Name of the institute

Institut für Kommunikationsnetze und Satellitenkommunikation

Graz, April 2016

AFFIDAVIT

Abstract

The satellite ground station Graz Lustbühel is part of the Lustbühel observatory and serves as a medium sized station for satellite communications in the Ku frequency band. It has mostly been used for communication with geostationary satellites and is currently dormant due to a failure of a control PC required to operate the station. The focus of this thesis is to reactivate and modernize the station by developing a completely new operating and control software that will replace the existing software *LEOS* (Lustbühel Earth station Operation System). Additionally, a new control PC provides the platform for the newly developed software while the current hardware that is required to operate the satellite station should be kept as it is. The challenge was to modernize the station and to keep most of the old and expensive hardware at the same time. The solution was found in using National Instruments LabVIEW development environment to create the new control software and coupled the software with Ethernet based converter cards that provide an interface between control PC and old station hardware. During the development and testing process parts of the station's missing documentation and malfunctioned hardware have been restored. After the Ethernet-based IP network that now connects the station's hardware components was set up, the newly developed software, *LEOSv2*, could be tested and now provides full control of all relevant components that are required to operate the satellite station. The development of a new operating and control software in combination with the implementation of a new control PC and an Ethernet-based solution for controlling the station's hardware carries a lot of potential for future extensions and enhancements of the station. Such enhancements are, for example, an easy implementation of new hardware components like temperature sensors or webcams for remote operation of the station and an easy extension of the software functionality like including a satellite tracking algorithm.

Table of contents

List of tables

1 Introduction

The satellite station Graz-Lustbühel is part of the Lustbühel observatory which was built in 1974 [1]. The observatory contains several departments of Graz University of Technology and Karl-Franzens-University Graz. Its main components are the optical telescope for astronomical observations, the satellite ranging station for satellite geodesy and a satellite ground station for satellite communications. The Lustbühel satellite station, also built in 1974, is a dedicated satellite communications station operating within the Ku-Band. It is located in a separate building near the main observatory building.

The Lustbühel satellite ground station was previously used for communication with geostationary satellites. For running operations a dedicated software, the **L**ustbühel **E**arth station **O**perating **S**ystem, short LEOS, was designed by Rupert Temmel [2]. LEOS is a DOS-based software, distributed on two PCs, that provided a user interface for controlling the satellite station's functions, such as repositioning the antenna, routing and monitoring of the transmit and receive signals and controlling related hardware. The station was operational for a long time and during that time a part of the station's hardware, the up-converter, was replaced by a newer model. This newer model was incompatible with the existing LEOS software and a separate PC was required to control the new up-converter, leading to an unwanted distribution of the stations control functions to several PCs. Eventually one of the main control PCs failed, leaving the operation of the station dormant for several years.

This theses is dedicated to reactivate the Lustbühel satellite station by the means of understanding the existing configuration, replacing faulty hardware and developing a new operating and control software for the whole station. The gaol is to keep the existing and still functional hardware, regardless of age, and replace the old control PCs and the existing control software, LEOS, by modern, state of the art systems. In order to achieve that goal one first has to understand what different kinds of hardware components are used in the station, how they are linked together, how they are connected to the LEOS control PCs and how they are controlled by the LEOS software. There are several different hardware components: waveguide switches and coaxial switches to route the transmit and receive high frequency signals, a stepper motor controller that provides functions regarding positioning of the antenna, angle reference counters for azimuth and elevation that are used to calibrate the antenna controller, a high power amplifier for the transmitting signal, a power meter to monitor the transmit power level and an up-/down-converter that controls transmit and receive frequencies. All these hardware components are connected to the LEOS control PCs via different interfaces. An RS-232 interface is used for the angle counters and the up-converter. The antenna controller, the power meter and some of the waveguide switches use an IEEE interface and the rest of the waveguide switches as well as the coaxial switches use a Digital-I/O interface. For each of those interfaces a dedicated interface card was built into the LEOS control PCs and therefore allowed control of the hardware components with the LEOS software. As previously mentioned the existing hardware of the station should be retained, so if the LEOS control PCs are replaced, the old, built-in interface cards need to be replaced as well. Until now there where two control PCs, one located in the ground station building and one located near the antenna. Each of those PCs had several different interface cards to communicate with the connected hardware components. This configuration led to two options for replacing the interface cards and PCs: Option one was to replace the two old PCs with two new ones and buy new interface cards for each PC. Option two was to eliminate one of the control PCs and decouple the interface cards from the PC. Option two has a number of advantages and therefore was the option of choice for the reactivation of the Lustbühel satellite station.

Even if the interface cards are decoupled from the PC hey still need to communicate with it. The solution for this are Ethernet based interface cards that embed the interface data into Ethernet and use an IP-based network to communicate with the control PC which leads to a number of advantages: Firstly, the interface cards are now decoupled from the control PC, so the number of available interface cards is not limited by the interface slots of the PC. Secondly, by connecting the interface cards and the control PC via an Ethernet network, the interface cards can by physically located anywhere within the station and are no longer bound to the location of the PC. Thirdly, the Ethernet network allows an easy extension by adding more interface cards to the network and also allows easy replacement of existing ones. This also means that only one PC can now operate all the necessary interface converters which in return reduces the complexity of the control software so there are no special requirements for the control PC, which now could be any off-the-shelf PC or laptop.

Up to this point we focused on the station from a hardware point of few, so let’s now examine the software requirements. As already stated, the old LEOS software is DOS-based and specifically designed for the old hardware configuration and the old interface cards. By replacing the control PCs by modern devices a new software for operating the station is required and a major part of this theses is dedicated to the development of a completely new operating and control software for the Lustbühel satellite station. In honour of Rupert Temmel's work the acronym LEOS was kept and the new software is named Lustbühel Earth station Operating System Version 2, short **LEOSv2**. As the new operating and control software for the station LEOSv2 was designed as a flexible and extendable control software and was developed using the contemporary development environment LabVIEW from National Instruments. So why was LabVIEW chosen in favour of other development environments? Firstly, developing software in LabVIEW incorporates the creation of a user interface in the development process (see [3], [4] and [5]) which is an advantage to some purely text based programming languages [6]. Another advantage is that LabVIEW provides many predefined functions that make it very easy to communicate with external hardware like the Ethernet based interface cards that.

The newly developed LEOSv2 combines a modern, intuitive user interface with the familiarity of the previous LEOS software. Some of the visual aspects of the old user interface have been kept in order to provide a familiar environment while this environment was extended by some helpful features that allow the user to get a better overview of the overall state of the station. The LEOSv2 user interface provides all necessary information at a glance and separates the control of different components by grouping them in a tab-based manner, so only functions that are logically linked together are visible at a time. LEOSv2 integrates all hardware functionality that is necessary to use the Lustbühel satellite station for satellite communications [7]:

- Controlling of antenna pointing with the antenna controller
- Calibration of the antenna controller with the help of angle counters
- Waveguide switches to route the high frequency signals
- Coaxial switches to route intermediate frequency signals
- Switching the high power amplifier
- Monitoring the transmitted signal strength
- Controlling and monitor the up-converter

The LOESv2 user interface can be controlled by mouse and also by modern touch screens and the constant development of LabVIEW ensures compatibility of the LEOSv2 software with future PCs and operating systems.

Testing the software at the station revealed that a part of the station's hardware that has not yet been mentioned also required some adaption before the new software, control PC and interface card could be used properly. Some of the hardware components, namely waveguide switches, coaxial switches and the high power amplifier are not connected directly to their corresponding Digital-I/O interface cards. Instead they are connected to the so called LEOS control box (LEOS box), which is a hardware box that was created by Joanneum Research. The main purpose of the LEOS box is to convert the Digital-I/O voltages from the interface cards into the required voltages that are necessary to operate the waveguide switches, coaxial switches and the high power amplifier. The plans for the internal circuitry of the LEOS box have been lost, therefore a part of this thesis is the documentation and adaption of the LEOS box to the current requirements. More details about the LEOS box can be found in chapter 5.1.

After the LEOS box was adapted to the new interface cards LEOSv2 could be tested successfully, including all previously available functions of the old software and of course also the new features of the LEOSv2 software that include control of the up-converter. Keeping in mind potential future enhancements of the station, LEOSv2 can easily be adapted to new hardware components or other requirements by enhancing its user interface. A possible future use of the satiation could be remote operation via windows remote desktop or Virtual Network Computing (VNC, [8]). For that purpose the installation of a webcam might be considered to optically monitor the position of the antenna. This webcam then could also be integrated into the LEOSv2 software. Another enhancement would be the implementation of a tracking algorithm for satellite tracking.

The following chapters will introduce the Lustbühel satellite station and its components in detail for both, the old setup and the new setup that has been implemented in the course of this thesis. The hardware components of the station and their interconnections are depicted for in schematic views for better orientation. After the station is explained the software is examined in detail, starting with an introduction of the Lab VIEW development environment and an explanation of the used concepts and terminology. The LEOSv2 software is then introduced step-by-step with the help of screenshots of the LabVIEW graphical code. The LEOS box and hardware that is connected to the LEOS box is described afterwards, followed by a summary of the hardware setup and commands that are required to communicate with a specific hardware component.

2 Station overview

The following chapter gives an overview over the Lustübhel satellite station by showing the physical location of the different components of the station followed by an explanation of those components.

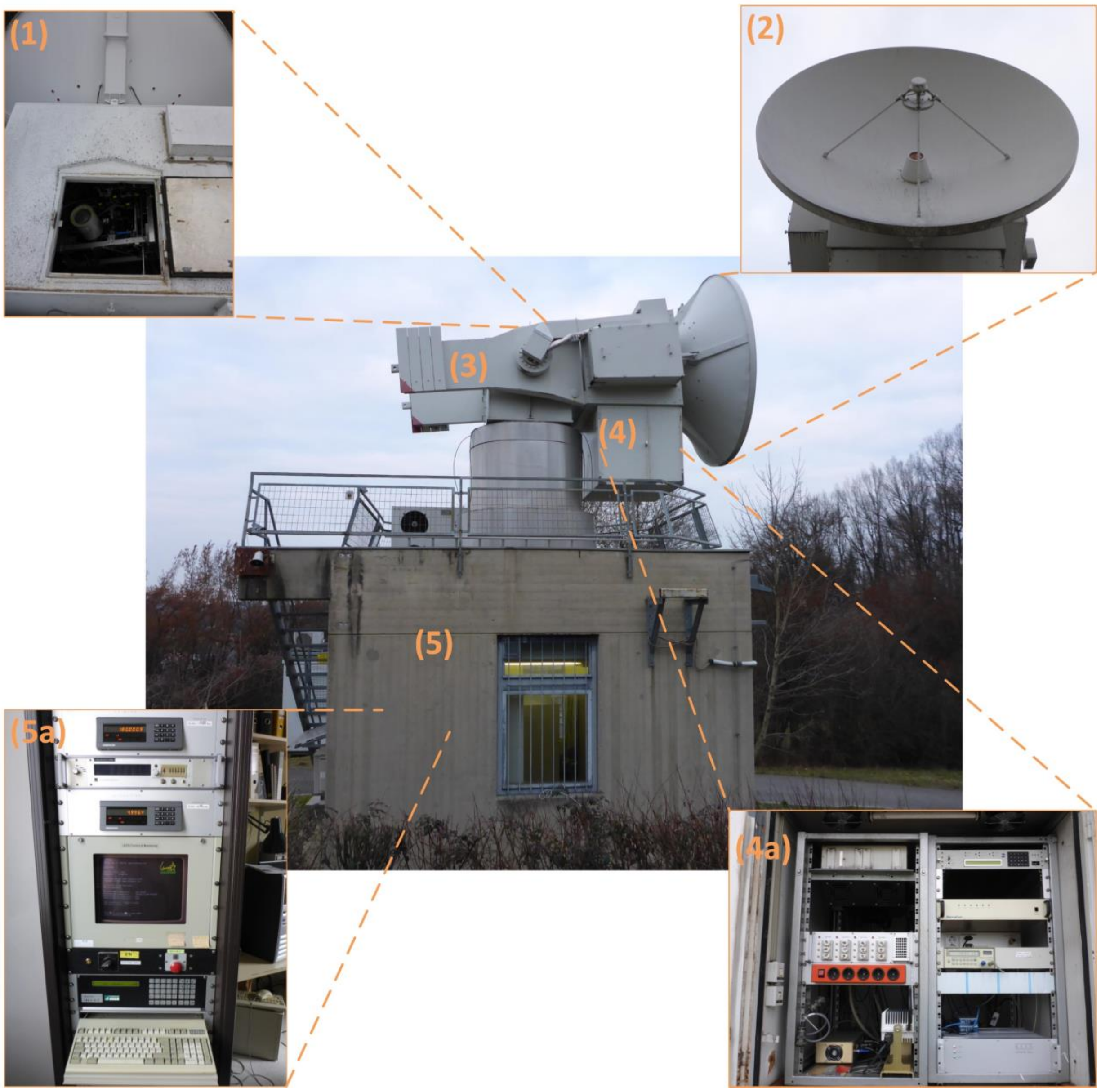


Figure 2-1 Ground station building of the Lustbühel satellite station. (1) Shows the open hatch on top of the antenna pedestal, providing access to the waveguide housing. (2) Antenna and Cassegrain feed. (3) Antenna pedestal, holding the antenna and the antenna hardware box. (4) Antenna hardware box that provides space for hardware and cables. (5) Ground station building. (5a) Ground station control rack.

Figure 2-1 shows the station building with the antenna pedestal on top, carrying the antenna and a hardware box. On top of the antenna pedestal there is a housings for the high frequency components. The labelled elements in Figure 2-1 are:

(1) Waveguide housing
(2) Antenna
(3) Antenna pedestal
(4) Antenna hardware box
    a. Left side of the box, showing two 19’’ racks with mounted hardware.
(5) Ground station building
    a. LEOS rack, located inside the building.

The **waveguide housing** is located in the antenna pedestal, directly behind the antenna. It can be accessed through a hatch on top of the antenna pedestal, as shown in Figure 2-1 (1). It contains the interconnected waveguides, waveguide switches and coaxial switches that build transmit and receive signal paths for Ku band transmission. The signals can be routed to two orthogonal linear polarized planes and the orientation of these polarized planes can be rotated manually by a crank that rotates a metal frame in which the high frequency components are mounted. The components inside the waveguide housing that are software controlled are the waveguide switches and the coaxial switches, either by a Digital-I/O interface or an IEEE interface.

The **antenna** is positioned at the front of the antenna pedestal, with the feed directly in front of the waveguide housing. The 3m dish has a Cassegrain feed, an efficiency of about 50% and an antenna gain of about 50 dB at 14 GHz.

The **antenna pedestal** is mechanically steerable 360° in azimuth and 180° in elevation, if not limited by the steering hardware [9]. As mentioned above, there is a hatch at its top, providing access to the antenna housing. The antenna hardware box was subsequently installed below the antenna pedestal and behind the antenna after the station was finished. Positioning of the antenna pedestal is controlled by a stepper motor controller from the company Phytron. The model is: Phytron IXE-alpha-C-T [9]. This antenna controller is software controlled via an IEEE interface.

The **antenna hardware box** provides a temperature controlled housing for diverse hardware. It can be accessed from both sides, providing two 19'' racks on its left side (as seen from behind the antenna) and access to the back of the racks plus some additional space on its right side. In the old configuration one of the two LEOS control PCs was located in the antenna box. This PC has now been removed, and its interface cards have been replaced by the Ethernet based solution. There is now one Moxa RS232-Ethernet converter, one National Instruments IEEE-Ethernet converter and two National Instruments Digital-I/O – Ethernet converters that are connected to the LEOS box, a box that connects the waveguide and coaxial switches to the converters and provides them with power supply. Software controlled components also located in the antenna hardware box are an up-converter from the company SierraCom, consisting of an indoor unit (IDU) and an outdoor unit (ODU), a power meter from the company Hewlet Packard (model HP-437B, [10]) and the high power amplifier (HPA).

The **ground station building** contains hardware for controlling and monitoring the satellite station. It provides a workplace to operate the old LEOS software and the new LEOSv2 software. The LEOS rack, shown in Figure 2-1 – (5a), is located within the ground station building. This rack contains the Heidenhain angle counters (model ND281, [11], the Phytron IXA-alpha antenna controller and a power switch for operating those components plus an emergency stop to abort movement of the antenna / antenna pedestal. The old LEOS control PC is located in the rack and still partially operational but it is disconnected from the angle counters and the antenna controller. Those two are now connected to the Ethernet converters. The ground station contains one Moxa RS232-Ethernet converter (for the angle counters) and one National Instruments IEEE-Ethernet converter (for the antenna controller). Those converters are connected via a switched network to the converters in the antenna hardware box and to the LEOSv2 control PC.

The following chapters provide a bridge between the old station setup and what has been done within this theses in order to make clear what is new and how it has been adapted. Starting with an overview of the station, the relevant components to operate the station are described in Chapter 3. The physical location of those components and their interconnection will help to understand the stations setup and how it is implemented in LEOSv2. The software itself is then introduced in Chapter 4 after a short introduction of the development environment LabVIEW and the software conventions.

Chapter 4.4.10 explains the LEOS box and the circuitry and pin / connector layouts of the components that are connected to the LEOS box. This chapter introduces the station hardware from a physical point of view, while Chapter 6 sums up the protocols for communication with the hardware.

3 Station Description

The following chapter serves as an introduction to the hardware components of the Lustbühel satellite station. It is important to get an overview of all the relevant components and how they are connected. To start with, a general overview of the station’s old hardware setup is shown and the connections, as well as the interfaces between the components are listed. To compare the station’s old hardware setup with the new one, both setups are shown in a hardware- and connection diagram (see Sections 3.2 and 3.3). These diagrams include a schematic overview of the whole station, providing the network setup of the station, the connections between the relevant components, the hardware interfaces, the physical connectors and the signal flow paths.

The high frequency (HF) paths and components of the station, including the antenna, waveguides, waveguide switches, coaxial switches, high power amplifier (HPA) as well as the up- and down converters are shown and described in block diagrams later in this chapter.

Breaking up the stations components, we can refer to the following groups, which then will be described in detail:

- Station control software and hardware (LEOS and LEOS control PCs, PC for up converter)
- Intermediate hardware (LEOS control box)
- Antenna and hardware for antenna control and monitoring (antenna controller and angle reference)
- Physical high frequency path (waveguides, waveguide switches, coaxial switches)
- Operational high frequency path (up- and down converter, HPA, hot loads, cold loads, filter, low noise amplifiers)

3.1 The stations component groups

This chapter provides an overview of the different component groups of the Lustbühel satellite station. Each group can be divided into several sup-components, for example the station's control software, the control PCs for this software and of course all hardware that is required to operate the station.

3.1.1 Station control software and hardware

The Lustbühel Earth station Operating System, LEOS (for a short introduction see chapter 4.1) was originally written in the programming language Pascal and is a control software for the station's hardware, distributed on two DOS computers. These two PCs are equipped with interface cards, providing the necessary RS232, IEEE-488 and Digital-I/O communication between PC and the station's hardware components. One PC is located in the station and will be further referred to as LEOS station PC, the other one is mounted in the antenna box and is therefore referred to as LEOS box PC. The LEOS station PC is running the main part of the software and controls all the hardware located within the ground station building, the antenna controller, angle counters and downlink oscillator. It also runs the LEOS user interface which can be operated on a 19'' rack-mounted terminal in the building. The LEOS box PC runs the second part of the LEOS software and is also connected to the LEOS control box, controlling all the hardware located within the waveguide housing and the antenna box (waveguide switches, coaxial switches, HPA, up converter, power meter). The two PCs are interconnected by a coaxial based network interface. Additionally, a third desktop PC (running Windows 98) was required to control the new SierraCom up-converter which replaced the former Marconi up-converter. Operating the up converter is therefore decoupled from the original LEOS software.

The within this master's project newly created LEOSv2 software not only includes control of all the hardware, it was developed from scratch in a modern, seminal way, using National Instruments LabView software development environment. Three major advantages arise with the new LEOSv2 software: Firstly, the old DOS machines can be replaced by a modern, state of the art PC running the latest operating system. Secondly, the two old DOS PCs are replaced by one single PC and the interface cards for communication with the stations hardware components are decoupled from the control PCs. This decoupling allows the use of a single "off the shelf" PC or laptop to run the LEOSv2 software instead of two expensive PCs that require industrial standard. All the interface cards are replaced by a converter that transforms the respective physical interface, using the Ethernet standard. A simple switched network allows for communication between the Ethernet converters and the new LEOSv2 PC. This is a sustainable solution that allows easy replacement in case of a hardware failure, it is easily expandable because the addition of new converters is not limited by the amount of interface slots in the PC and since the Ethernet standard is not proprietary, Ethernet based hardware will be available for a long time. Thirdly, another advantage arises, using a modern PC for operating the LEOSv2 software is that now remote access and control of the station is made possible by the use of virtual private network (VPN) technology.

3.1.2 Intermediate Hardware

The term intermediate hardware is chosen for the LEOS control box. This box acts as an intermediate step between the Ethernet converters and the actual hardware components of the station, providing power supply and / or drive logic for the following components: the HPA, the waveguide switches and the coaxial switches (and the Marconi up-converter in the very first setup). The LEOS control box was developed in collaboration with Joanneum Research Graz and its original configuration has been extended, by adding pull-up resistors to some of the Digital-I/O ports (see Chapter 5.1. Within the course of this work, the internal circuitry of the LEOS control box (which had been lost) has been documented anew.

For the HPA the box provides 230V power supply that can be turned on or off using a Digital-I/O port. The waveguide switches connected to the LEOS box consist of one 3-way switch and five 2-way switches. Waveguide switches are provided with 28V and 5V power supply. For switching the 3-way switch a special circuit board is needed. The 2-way switches do not require any special circuit for switching, therefore the Digital-I/O ports from the Ethernet converter are directly routed to the switches within the LEOS box. For the coaxial switches a higher voltage is required, so the LEOS box provides relays to transform the 5V TTL voltage into the required 28V switching voltage. For the indicators of both, the waveguide switches and the coaxial switches, TTL voltage is provided by the switches with the help of the above mentioned pull-up resistors, so the indicator pins of all switches are directly routed to the digital I/O pins of the Ethernet converter.

3.1.3 Antenna and hardware for antenna control and monitoring

The antenna is a 3 meter dish antenna with a Cassegrain feed and a gain of roughly 50 dB at 14 GHz, derived by the following formula (see [12]):

$$G = 10 * \log\left(\eta\left(\frac{\pi\,\mathrm{D}}{\lambda}\right)^2\right)$$

G is the antenna gain, η is the antenna efficiency which is roughly 50% as determined by measurements [13], D is the antenna diameter and λ the wavelength. The antenna is mounted on top of the ground station building, directly in front of the high frequency waveguides that are located in the antenna housing behind the dish. The antenna housing contains the waveguide switches and coaxial switches to route the transmit and receive signals and can be operated electrically. Adjustment of the angle of the polarization planes can only be performed manually. To do so, one must climb on top of the antenna and enter the antenna housing from above. A more detailed description of the HF path can be found in Chapter 3.1.4.

The orientation of the antenna is controlled by the Phytron IXE-alpha antenna controller. The IXE-alpha is a programmable stepper motor controller providing drive and readout for azimuth and elevation. Positioning of the antenna can either be done by using the IXE-alpha's input panel or by sending commands to the antenna controller over its IEEE interface. Programming the IXE-alpha allows for limiting the angular range in azimuth and elevation in order to stay within the physical limitations of the antenna pedestal.

In addition to the IXE-alpha antenna controller, two high precision angle measurement devices produced by the company Heidenhain are used as an angular reference for the antenna controller. These angle references can be used to monitor the antenna controllers angular readings and, if needed, to calibrate the antenna controller.

3.1.4 Physical high frequency path

The physical high frequency path includes the waveguides, the waveguide switches, coaxial lines and coaxial switches. The station has an 11 GHz path and a 12 GHz path that both can be accessed from the horizontal or vertical polarization path, resulting in a total of 4 signal paths. There is also an additional path for local testing using the test translator. The block diagram in Figure 3-1 shows a schematic of the physical high frequency path.

There is a total of nine waveguide switches, one of which is a 3-way switch, which will be further referred to as Switch 1 (SW1), the others are 2-way switches (SW2-9). The switches SW1 to SW6 are controlled by Digital-I/O logic and SW7 to SW9 have an IEEE interface. SW1 to SW6 are connected to the LEOS control box, which provides power supply and drive logic. SW7 to SW9 are directly connected to the IEEE - Ethernet converter located in the antenna box.

The two 4-way coaxial switches, further referred to as COAX1 and COAX3 are used to route the received signal or the test translator from one of the four previously mentioned signal paths to the down converter. A 2-way coaxial switch, COAX2, is used for switching the test translator frequency. All the coaxial switches are connected to the LEOS control box (see chapter 5.1) that provides power supply and drive logic.

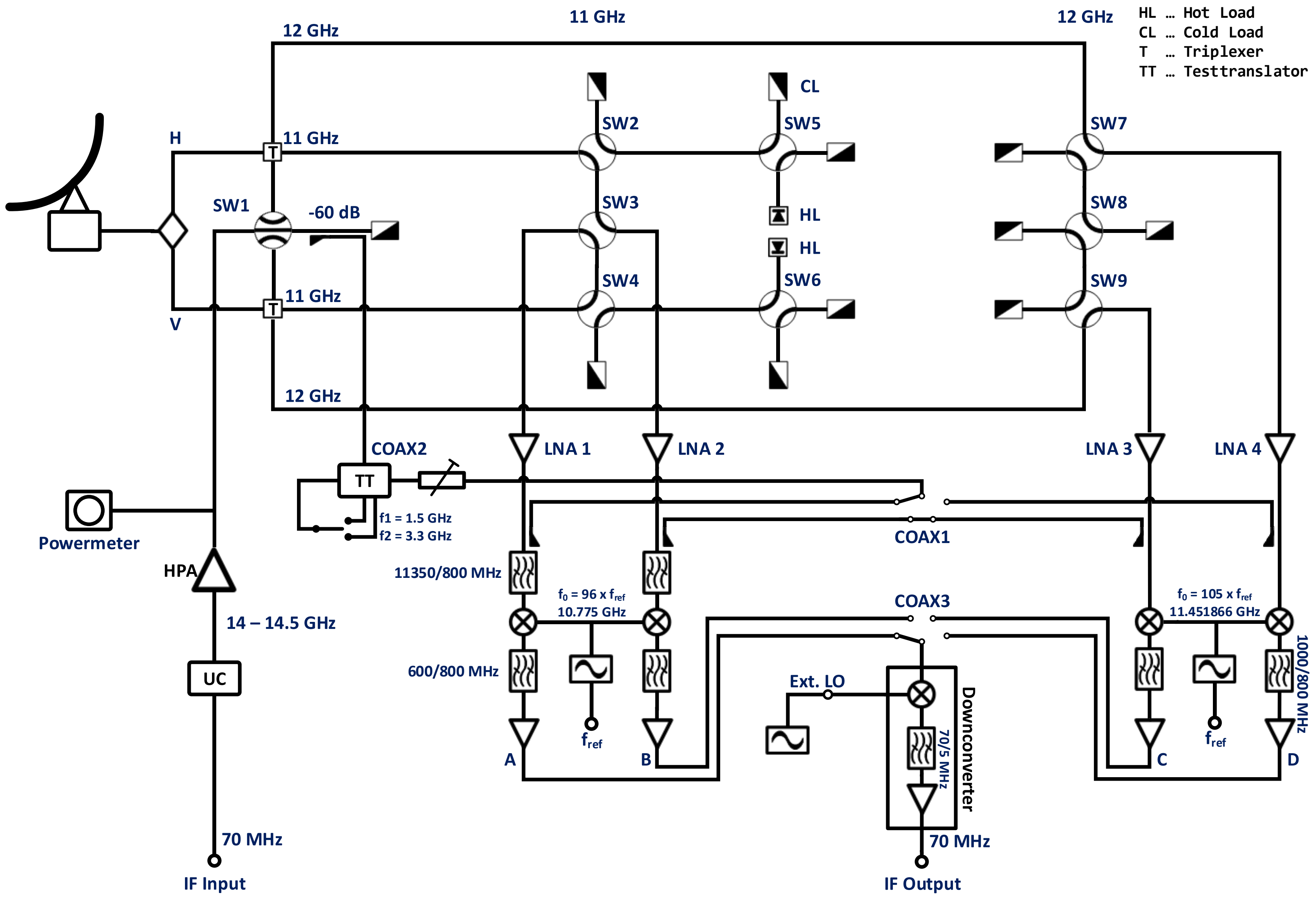


Figure 3-1 Block diagram of the physical high frequency signal path.

3.1.5 Operational high frequency path

Beside the physical signal path the sum of electronic components like amplifiers, filters, converters and terminators is hereby referred to as operational high frequency path. Some of these components like the LNAs, filters and terminators (hot loads and cold loads) are built into the signal path and are not controlled by the LEOSv2 software. Software controlled components are: The HPA, power meter, downlink oscillator and the up converter. The HPA is switched by a relay in the LEOS box that is controlled via a port of the digital-I/O to Ethernet converter in the antenna box. The power meter readout is done over its IEEE interface using the IEEE - Ethernet converter in the antenna box.

The new SierraCom up-converter replaced the old Marconi up-converter which was connected to the LEOS control box. The SierraCom up-converter is controlled using a serial RS232 interface and, in the old configuration, was connected to the Win 98 PC in the station building over a RS232 – RS485 – RS232 converter path to overcome the long distance. In the new setup, the SierraCom up converter is connected to a Moxa RS232 to Ethernet converter that is located in the antenna box. Software control for the up converter is now fully implemented in the new LEOSv2 software and doesn’t require an additional PC any more.

3.2 The old setup

Previously the station was controlled by the two LEOS control PCs, one of which is located directly in the station building, the second one is located in the antenna box and was directly connected to the LEOS hardware control box. The two PCs were connected via a coaxial cable and had built-in interface cards for R-S232, IEEE and Digital-I/O interfaces. The LEOS software was written in Pascal and running on a DOS-based computer. It connected the two control PCs and managed all the hardware communication for the station’s components. The software did not include control for the new SierraCom up-converter because this was added to the station after the LEOS software was finished. To fully operate the station one had to boot 2 separate PCs, the LEOS control PC and a separate machine running Windows 98, where the up-converter terminal was installed. Table 1 gives an overview of the relevant components that were controlled by the LEOS software and the Windows 98 PC:

Table 1 Station hardware, interfaces and connections for the old setup.

Hardware	Interfaces	Interface card	Connected to
Antenna controller	IEEE	PCL-848	PC (station)
Angle counters	RS-232	-	PC (station)
Downlink oscillator	IEEE	PCL-848	PC (station)
Power meter	IEEE	PCL-848	PC (box)
High power amplifier	-	-	LEOS control box
Waveguide switch 1-6	Digital-I/O	PCL-720	LEOS control box via PC (box)
Waveguide swtich 7-9	IEEE	PCL-848	PC box
COAX switch 1-3	Digital-I/O	PCL-720	LEOS control box via PC (box)
Up-converter old	Digital-I/O	PA200	LEOS control box via PC (box)
Up-converter new	RS-232	-	Win 98 PC

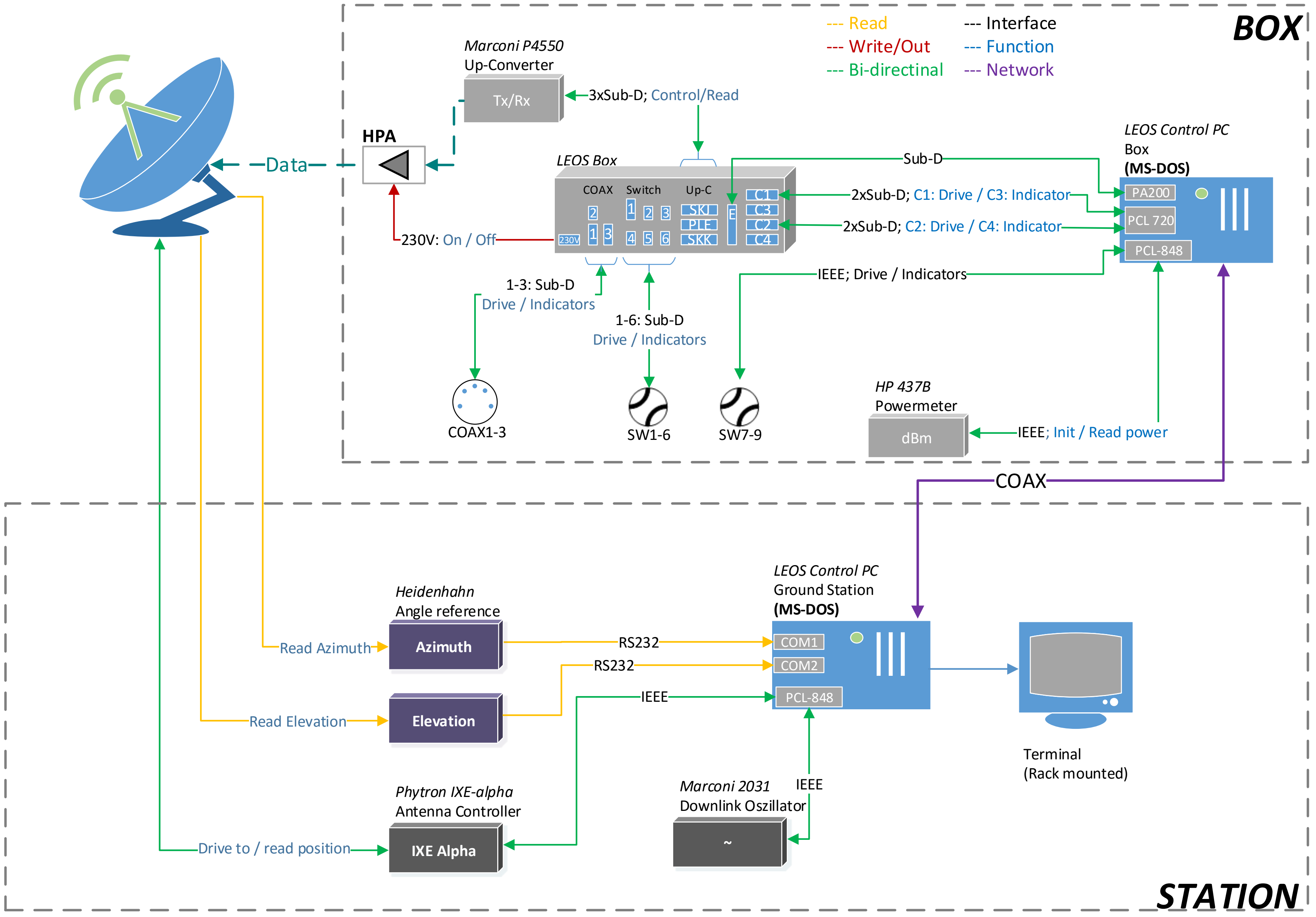


Figure 3-2 Satellite station components and old network setup.

3.3 The new setup

The current LEOS PCs are very old (20+ years) and the box PC has already stopped working, so a substantial part of the hardware components cannot be controlled by software any more. Also, getting spare parts for such old PCs becomes increasingly difficult. Adding new hardware to the PCs is limited by the amount of interface slots a PC provides, so a new solution had to be found. It was desirable to reduce the software complexity by eliminating the second control PC and combine all the software logic in one single programme. Further the new control PC was decoupled from the hardware interface cards. This was made possible by the use of Ethernet based converters for the corresponding physical interfaces (Moxa Ethernet to RS-232 [14], National Instruments Ethernet to IEEE [15] and National Instruments Ethernet to Digital-I/O [16]). An advantage of using Ethernet converters is the flexible network setup, e.g. easy addition or removal of converters to the switched network. Additionally the control PC can now easily be replaced and does not have any special requirements like, for example, the need of industrial standard. Any commercially available PC or laptop can be used. Table 2 shows the hardware and its corresponding interface and interface cards.

Table 2 Hardware and corresponding interfaces and interface cards for the new setup.

Hardware	Interface	Interface card
Antenna controller	IEEE	NI GPIB-ENET/1000 (station)
Angle counters	RS-232	Moxa NPort 5410 (station)
Power meter	IEEE	NI GPIB-ENET/1000 (box)
High power amplifier	Digtal-I/O	LEOS control unit via NI 9403 [16] (box)
Waveguide switch 1-6	Digtal-I/O	LEOS control unit via NI 9403 (box)
Waveguide swtich 7-9	IEEE	NI GPIB-ENET/1000 (box)
COAX switch 1-3	Digtal-I/O	LEOS control unit via NI 9403 (box)
Up-converter new	RS-232	Moxa NPort 5410 (box)

The Ethernet converters are connected via a very simple switched network using the private IPv4 Class-B net 169.254.0.0 to 169.254.255.255 with the Subnet mask 255.255.0.0.

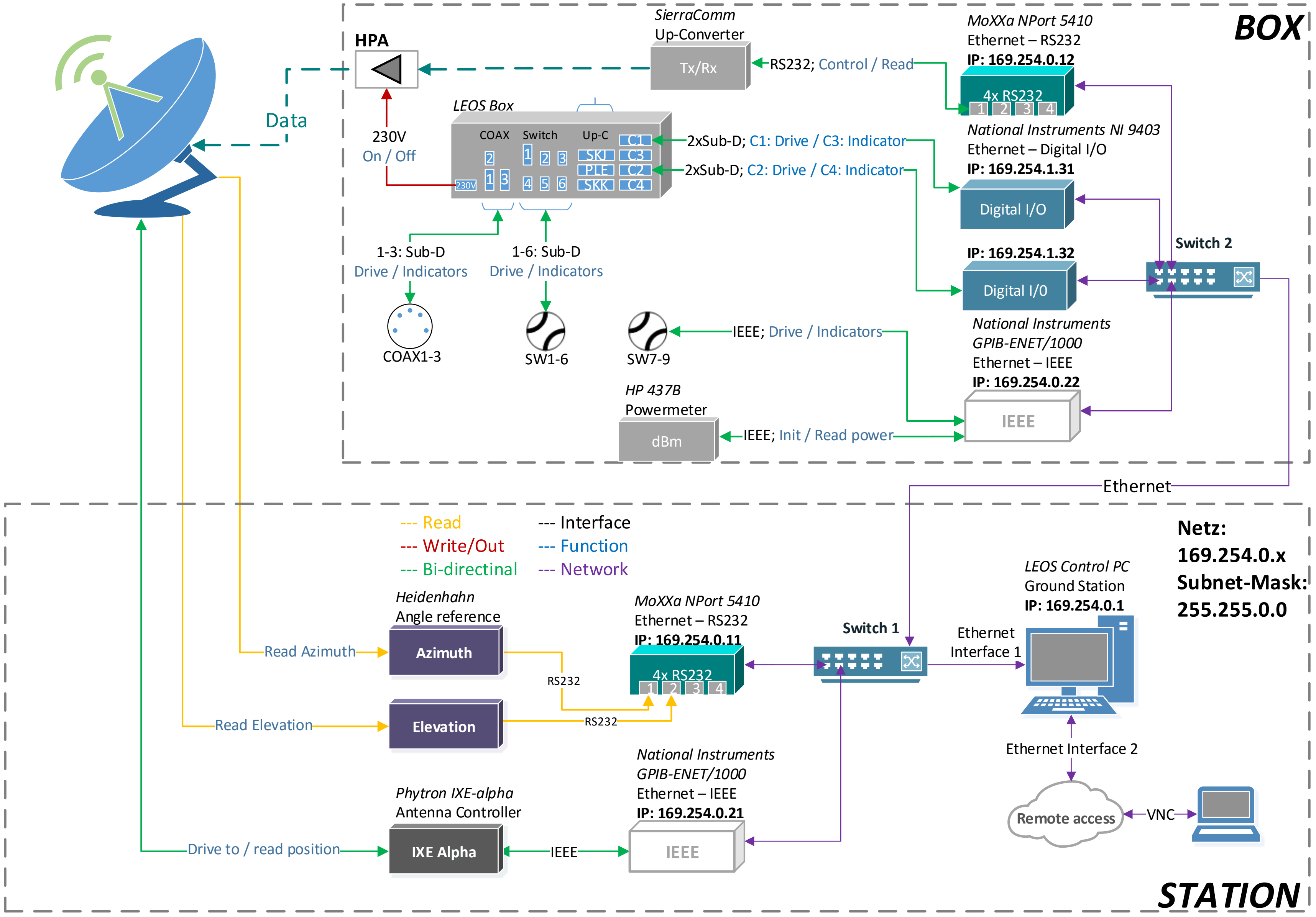


Figure 3-4 Satellite station components and new network setup.

4 Software

This chapter gives an overview of the Lustbühel satellite station's control software. Operating the station requires a detailed knowledge of a substantial amount of different commands for each of the station's hardware components. Many of the components are physically separated, therefore operating the station would be impractical without a comprehensive control- and operating software. So the purpose of the control software is the creation of an easy, user-friendly and intuitive user interface to control and monitor the station's hardware. The new LEOS software, LEOSv2, which has been developed in the course of this thesis, was created from scratch and is intended to replace the older LEOS version. It will extend the functionality of the old software, whilst maintaining some of the familiar features of the old Graphical User Interface (GUI). LEOSv2 was created using National Instruments **Lab**oratory **V**irtual **I**nstrumentation **E**ngineering **W**orkbench, short LabVIEW. It is intended as a sustainable platform to create an operating- and control software that can be run on any modern PC or laptop. LEOSv2 is designed to be flexible and easily extendible for new hardware and future applications.

Starting with a brief introduction of the old LEOS software, we will then continue with the new software requirements and give an introduction to LabVIEW. After introducing the concepts and terminology of LabVIEW, the LEOSv2 software will be described in detail, including the new GUI, the software's components, the programming concepts, software naming conventions and the operation and application of LEOSv2.

4.1 LEOS software old

The original LEOS software, created by Rupert Temmel [2], was designed with the programming language Pascal. It is operated on two DOS based PCs. The LEOS software is the control and operating system that provides a GUI and manages the interface converter cards that are connected to the station's hardware. Since two PCs were necessary for the operation, the LEOS software components have been distributed in packages for the PC that is located in the ground station building (Station-PC) and the second PC, located in the antenna box (Box-PC). Communication between those two PCs was handled by the software's network routines. Figure 4-1 shows the LEOS main GUI which depicts a block diagram of the station's physical high frequency path (see Chapter 3.1.4 and Figure 3-1).

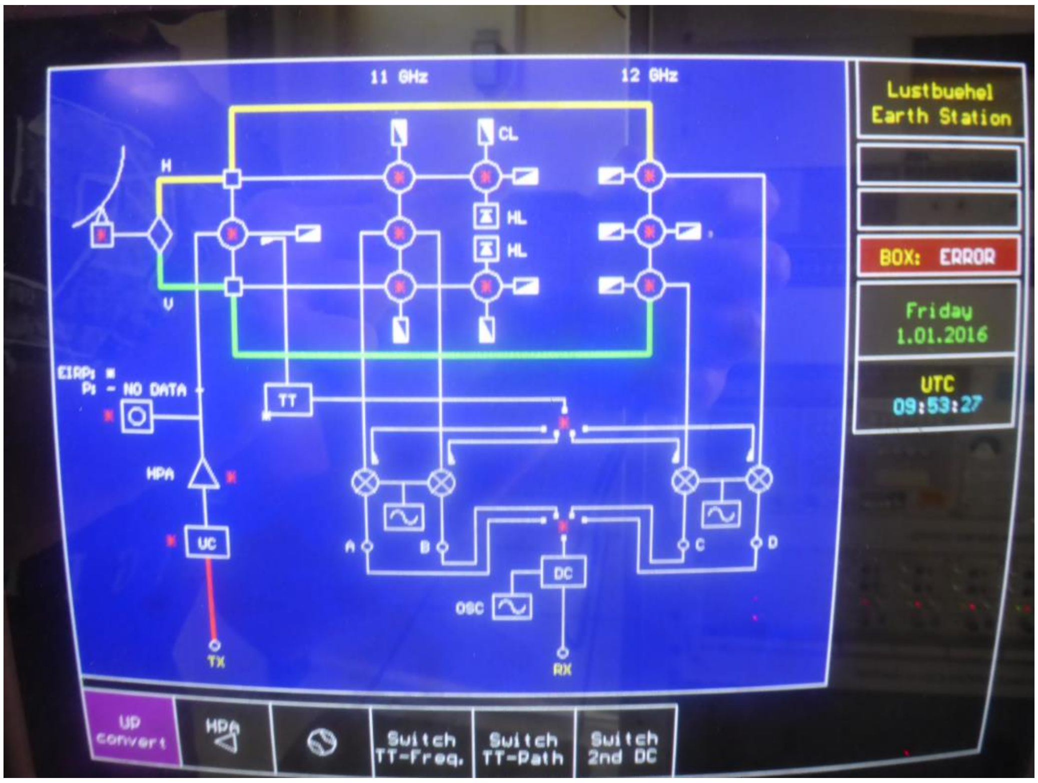


Figure 4-1 Graphical user interface of the LEOS software.

In order to maintain the already familiar software handling, the main layout of the old LEOS software has been adopted and is expanded by some useful features and functionality in the new LEOSv2.

4.2 Development Environment

LEOSv2 was built in a visual programming language, provided by National Instruments LabVIEW [17]. LabVIEW is a development environment with the underlying visual programming language named "G" (see [3], [5] and [18]). G itself is based on dataflow programming, meaning that different functional nodes are connected by "wires" in the graphical LabVIEW block diagram. Elements and wires in the block diagram can be interpreted as the LabVIEW source code.

Programmes and subroutines in LabVIEW are generally referred to as virtual instruments (VIs). Each VI consists of three different components. These components are called block diagram, front panel and connector panel. The block diagram, contains the graphical source code, represented by nodes, loops, structures and their interconnections. The front panel represents the user interface, displaying nodes from the block diagram as controllable objects (**controls**) or displays (**indicators**). A control is an input that can be operated by the user, while the indicators are outputs that indicate or display status and results based on programme logic or user input. This concept of the front panel

allows an easy development of a user interface, because the user interface is inherently included into the development process. The connector panel allows a VI to be represented within another VI, the so called calling VI. A virtual instrument can be executed on its own or it can be built into another VI. When using a VI as a function or subroutine (Sub-VI) it is required to assign controls or indicators to the **connector panel** of this VI. Elements linked to the connector panel serve as input or output for this Sub-VI and can therefore be imagined as the parameters and return values of text based programming languages.
If a large amount of inputs or outputs need to be transferred from a VI, the use of a **cluster** is recommended. A cluster is a collection of nodes that are then represented as a single element within the block diagram so it doesn't get cramped with too many nodes.

LabVIEW provides routines for nearly any commercially available hardware and has simple built-in mechanisms for hardware-software communication. The inherent creation of a graphical user interface during the development process allows a simple and fast creation of complex GUIs. A lot of graphical elements that are commonly used (for example: switches, buttons, LEDs,..) are included in the development environment and can be directly used for GUI building.

4.2.1 LabVIEW nomenclature

Here you find a summary of the terms that are later used in the software description. The explanations here are intended as a simple definition of the terms. For a more complete definition, the LabVIEW help [4] can be used. The following terms are commonly used, when working with LabVIEW and a good description of the different terms can also be found in [3]:

Virtual instrument (VI): A programme -or subroutine consisting of a front end, a block diagram and the connector panel. If a VI is used as a Sub-VI within another VI, the connector panel needs to be linked to controls and / or inputs.

Sub-VI: A Sub-VI is a VI that is called within the block diagram of another VI. The purpose of this is clarity and structuring of the graphical code, much like writing subroutines in text-based programming languages.

Calling VI: If a VI uses a Sub-VI within its block diagram, it is called the calling VI.

Front panel: A VI’s front panel is a graphical user interface that contains controls and indicators. It may also contain other, non-functional elements to structure and design the user interface.

Block diagram: Here the graphical source code is created. Controls, indicators or Sub-VIs appear as nodes on the block diagram. Nodes are interconnected using wires. The arrangement of nodes and wires determines the signal flow and therefore the programme logic.

Connector panel: The connector panel serves as a VI’s interface for input and output. Controls and indicators can be assigned to a connector on the connector panel. Each assigned connector can then be attached to a wire in the block diagram.

Control: A control is data source and is either a user controlled element on the front end (e.g. a switch, a numerical input, a slider,…) or a data input of some other source.

Indicator: To display results in various ways on the front panel, indicators are used. This can be, for example, LEDs or numerical displays.

Node: An element on the block diagram that is connected by a wire. This can either be controls, indicators or other VIs.

Cluster: A collection of controls or indicators that appear individually on the front end but are grouped together in the block diagram to avoid the need of an individual wire for each element in the cluster. A cluster also appears as a node in the block diagram.

Type definition: A customized control that only has a block diagram. Changes done in a type definition are automatically updated everywhere in the programme, where this type definition is used.

Enumeration constant (enum): An enum constant is a list of text elements, where a number is assigned to each element. Enum constants are often used to save the states for a state machine and therefore are usually saved as a type definition.

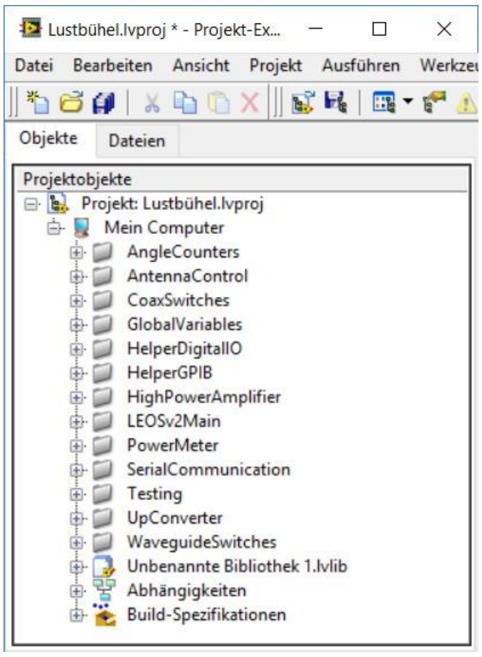
Case structure: A case structure executes code, depending on which case is active on its input. A simple form of a case structure with two cases can be compared to an if-statement in text based programming language.

State machine [18]**:** A state machine in LabVIEW can be realized by iterating through the different cases case structure in a loop by using different states that are stored in an enum constant [19]. Usually this enum constant is defined as a type definition. Starting with the initial state, each state stored within the enum constant is executed and if the last state is reached, the loop is stopped.

VISA-resource: VISA is a standard-I/O-API for programming of devices. A VISA-resource holds a reference to device, for example the name of the Digital-I/O lines of a Digital-I/O device or the name of an IEEE device.

4.3 Project overview

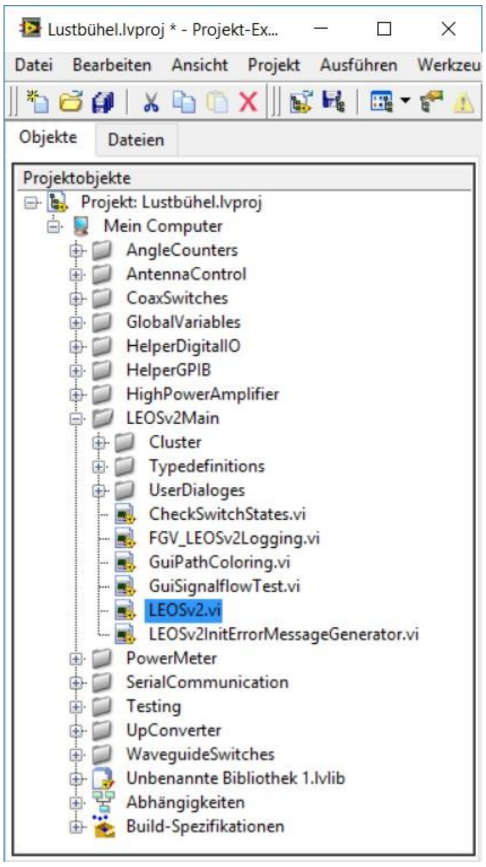
This chapter provides a short overview of the LabVIEW project layout. LabVIEW allows the creation of virtual folders to organize the VIs of a project. While the files for the LEOSv2 project are located within a single folder on the hard disk for easy file handling, they are collected in groups within a virtual folder in the LabVIEW environment. The virtual folders are listed alphabetically and contain VIs that are functionally linked together. Figure 4-2 shows a project overview with a listing of the main virtual folders, most of which directly correspond to a certain hardware component. Each virtual folder is extended by subfolders to further group VIs that serve as type definitions, clusters, user dialogs or custom made controls and indicators, that are required within this project.
There is a dedicated virtual folder for every hardware component that is controlled by the software. The dedicated hardware folders are: *AngleCounters*, *AntennaControl, CoaxSwitches, HighPowerAmplifier, PowerMeter, UpConverter* and *WaveguideSwitches*. Additional folders contain components of the user interface, VIs that are used by more than one hardware group, VIs for global variables and VIs for testing. The folder *LEOSv2Main* contains the main graphical user interface (GUI) and its dependent components. The folder *HelperDigitalIO* contains routines for working with the National Instruments Digital-I/O interfaces. The folder *HelperGPIB* contains routines for working with the National Instruments IEEE interfaces. For serial communication on COM-ports, VIs from the folder *SerialCommunications* are used. General VIs for testing and debugging can be found in the folder *Testing* and the folder *GlobalVariables* self-explanatoryly contains all global variables.



```
Projektobjekte
Projekt: Lustbühel.lvproj
  Mein Computer
    AngleCounters
    AntennaControl
    CoaxSwitches
    GlobalVariables
    HelperDigitalIO
    HelperGPIB
    HighPowerAmplifier
    LEOSv2Main
    PowerMeter
    SerialCommunication
    Testing
    UpConverter
    WaveguideSwitches
    Unbenannte Bibliothek 1.lvlib
    Abhängigkeiten
    Build-Spezifikationen
```

Figure 4-2 Project overview showing the virtual folders.

Figure 4-3 shows the extended *LEOSv2Main* folder as an example. Functional VIs are listed directly, additional types of VIs are grouped within separate sub folders.



```
Projektobjekte
Projekt: Lustbühel.lvproj
  Mein Computer
    AngleCounters
    AntennaControl
    CoaxSwitches
    GlobalVariables
    HelperDigitalIO
    HelperGPIB
    HighPowerAmplifier
    LEOSv2Main
      Cluster
      Typedefinitions
      UserDialoges
      CheckSwitchStates.vi
      FGV_LEOSv2Logging.vi
      GuiPathColoring.vi
      GuiSignalflowTest.vi
      LEOSv2.vi
      LEOSv2InitErrorMessageGenerator.vi
    PowerMeter
    SerialCommunication
    Testing
    UpConverter
    WaveguideSwitches
    Unbenannte Bibliothek 1.lvlib
    Abhängigkeiten
    Build-Spezifikationen
```

Figure 4-3 Project overview with extended folder “LEOSv2Main”, showing the grouped VIs and sub folders.

4.3.1 Naming conventions

For easier reading and clarity, a certain naming convention for virtual folders, VIs and variables will now be introduced. Within this project several different naming conventions are used for VIs, folders, type definitions and variables:

- **AllUpperCase:** Each word starts with an upper case letter and there are no spaces in between words. This is used for virtual folders and functional VIs.
- **firstLowerThenUpperCase:** The first word starts with an lower case letter, every following word with an upper case letter, again with no spaces in between words. This is used for type definitions. The first word of a type definition also indicates the purpose of the type definition. Here are two examples that are used in the LEOSv2 project for naming type definitions: the name of the type definition **statesWaveguideSw2way** means that this is used to store the states of the state machine for switching a 2-way waveguide switch, while the type definition **positionsWaveguideSw2wayDrive** stores the possible positions to which a 2-way waveguide switch can be switched.

- **lower case with space**: All lower case words are separated by a space. This is used for each listed element within an enum constant.
- **lower_case_with_underscore:** All lower case words are separated by an underscore. This is used for local variables, controls and indicators. Global variables use the additional tag "**global_**" in front of the variable name. An exception from this scheme are abbreviations for the waveguide or coaxial switches.
- **ALL UPPER CASE WITH SPACE**: All upper case words, separated by a space. This is used for the naming of Sub-VIs.

Aside from this defined naming scheme some other rules and concepts of naming have been applied within the LabVIEW project in order to keep the code readable and understandable. All type definitions use the name under which they are stored as display text in the block diagram. An enum constant that is stored as a type definition displays either its full name or is labelled with "next step" if used within a state machine. Controls and indicators are named according to their function. Most of the VIs block diagram symbols are labelled with a short descriptive text, some of them include a descriptive graphic. Additionally, each VI is documented with a help file that can be viewed with

LabVIEW's context help function. Within the block diagram, there are comments to further improve understanding of the code.

4.4 Software description

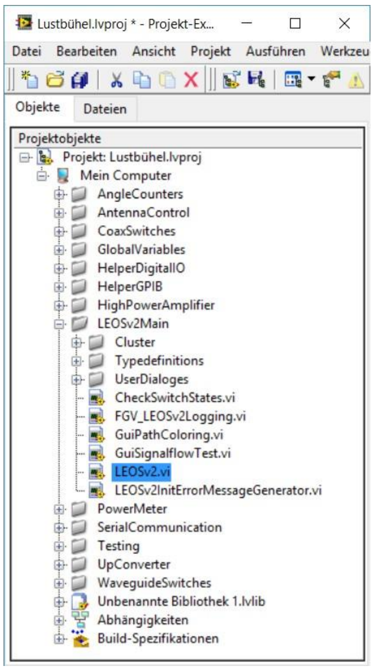
This chapter provides a more detailed description of the software functionality and concepts which the LEOSv2 software is based upon. Starting with the LEOSv2 main programme and graphical user interface (GUI), this chapter then continues step by step with an explanation of all functionality that is provided by the user interface. The user interface's functionality includes control of the following previously mentioned hardware components of the satellite station: angle counters, antenna controller, high power amplifier, up-converter, waveguide switches and coaxial switches. Each of those component's LabVIEW code is divided into several Sub-VIS that can be found in the corresponding project folders, as explained in Chapter 4.3.

The description of the different software components (e.g. the Sub-VIs for each hardware component) starts with showing the hardware associated user interface within the LEOSv2 GUI. Then the corresponding section of the GUI is explained in the GUI block diagram. After that the functionality of the Sub-VIs is explained in detail by showing the Sub-VI block diagrams followed by an explanation of the most important functions. In other words, the LabVIEW code is explained from top to bottom in the sense, that the upper layer code is explained first, and the functional depth of the code is illustrated by going through the underlying Sub-VIs. Note that this chapter only explains the most relevant concepts of the software and does not show every single detail of the code.

For easier orientation all variable names, VI names or other text passages that appear in the screenshots of the LabVIEW code in this chapter are printed as bold text. Additionally the name of a (Sub)-VI block diagram symbol is printed in bold and upper case letters. In the Figure caption of a (Sub-)VI this (Sub-)VI is always mentioned with its full project file name and the text of the LabVIEW block diagram symbol is added in parentheses.

4.4.1 LEOSv2 main

The LEOSv2 main programme or main VI is the executable of this project. It can be found in the project'S virtual folder (Figure 4-4) **LEOSv2Main** and the VI is simply called **LEOSv2**.



Lustbühel.lvproj * - Projekt-Ex...

Datei Bearbeiten Ansicht Projekt Ausführen Werkzeu

Objekte Dateien

```
Projektobjekte
Projekt: Lustbühel.lvproj
  Mein Computer
    AngleCounters
    AntennaControl
    CoaxSwitches
    GlobalVariables
    HelperDigitalIO
    HelperGPIB
    HighPowerAmplifier
    LEOSv2Main
      Cluster
      Typedefinitions
      UserDialoges
      CheckSwitchStates.vi
      FGV_LEOSv2Logging.vi
      GuiPathColoring.vi
      GuiSignalflowTest.vi
      LEOSv2.vi
      LEOSv2InitErrorMessageGenerator.vi
    PowerMeter
    SerialCommunication
    Testing
    UpConverter
    WaveguideSwitches
    Unbenannte Bibliothek 1.lvlib
    Abhängigkeiten
    Build-Spezifikationen
```

Figure 4-4 LEOSv2 project overview, showing the LEOSv2 main VI.

The front panel of the LEOSv2 VI contains the graphical user interface, GUI. The GUI provides all necessary functions to operate and monitor the satellite stations hardware. The GUI is divided into different areas that will be described in this chapter. Those areas contain controls and indicators for hardware operation and status information about station hardware and the Ethernet based converters.

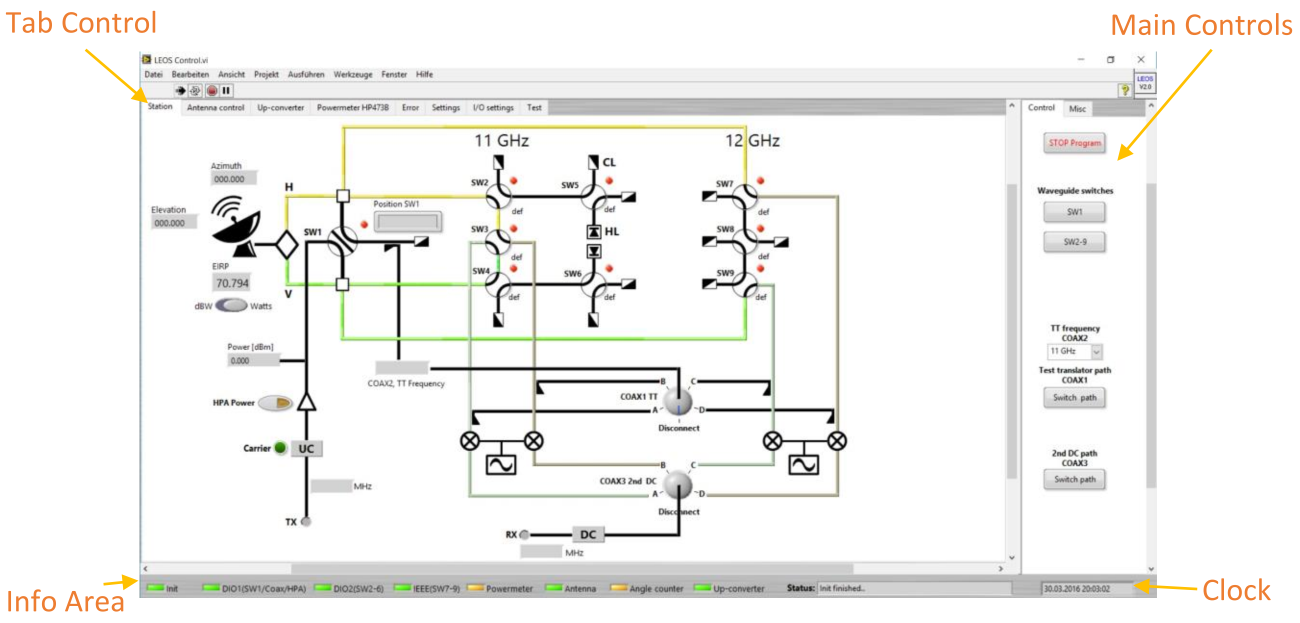


Figure 4-5 LEOSv2 graphical user interface, showing the different GUI areas and the active "Station" tab.

Figure 4-5 shows the LEOSv2 GUI with markings for the different areas. At the top left the **Tab Control** contains several tabs, each of which carries individual controls and indicators that are linked to a hardware component of the station. The example in Figure 4-5 shows the active tab **Station** that shows an overview of the physical high frequency path including status information for the antenna, up-converter frequencies and power level for the transmitting signal. The tab control area is the main area within the GUI and can be adjusted in size by dragging its borders with the mouse. It provides horizontal and vertical scrollbars to move the view around on smaller monitors. The different tabs of the tab control are explained in the following chapters.

At the right of the tab control the next area is called **Main Controls**. This is a separated area containing the controls for changing the configuration of the physical high frequency path and here the programme can be stopped. This area always stays visible, so if the software is started on a smaller monitor, the controls can always be accessed without the need of scrolling through the GUI.

At the bottom right of Figure 4-5 a clock provides time and date, should the software be run in full screen mode. The next important area is the **Info Area**. This area provides information about current events at the **Status** display and a number of LEDs that indicate the status of the software and also the status of the Ethernet based hardware converters for the Digital-I/O, IEEE and RS-232 interfaces. Table 3 gives an overview over the Info area status LEDs, their function and the associated Ethernet converters, interface addresses and connected hardware components.

The **Init** LED at the left in the Info Area is only **green**, if every step of the software initialization (see chapter 4.4.1.4) was accomplished successfully. If any error during the initialization occurred, this LED will show the colour **yellow**, meaning that one or more of the hardware components encountered an error during the initialization process. If the LED is yellow, the software can still be operated but potentially with limited functionality.

All other status LEDs appear only **green** if the Ethernet converter that is associated to a certain LED is functional and all components that are connected to this Ethernet converter also function properly. If any connected hardware encounters an error, either during initialization or during running operation, the corresponding status LED appears **orange**. In this case, the user is warned with an error message, and the corresponding error indicator on the GUIs **Error** tab (see chapter 4.4.1.3) shows details about the Error.

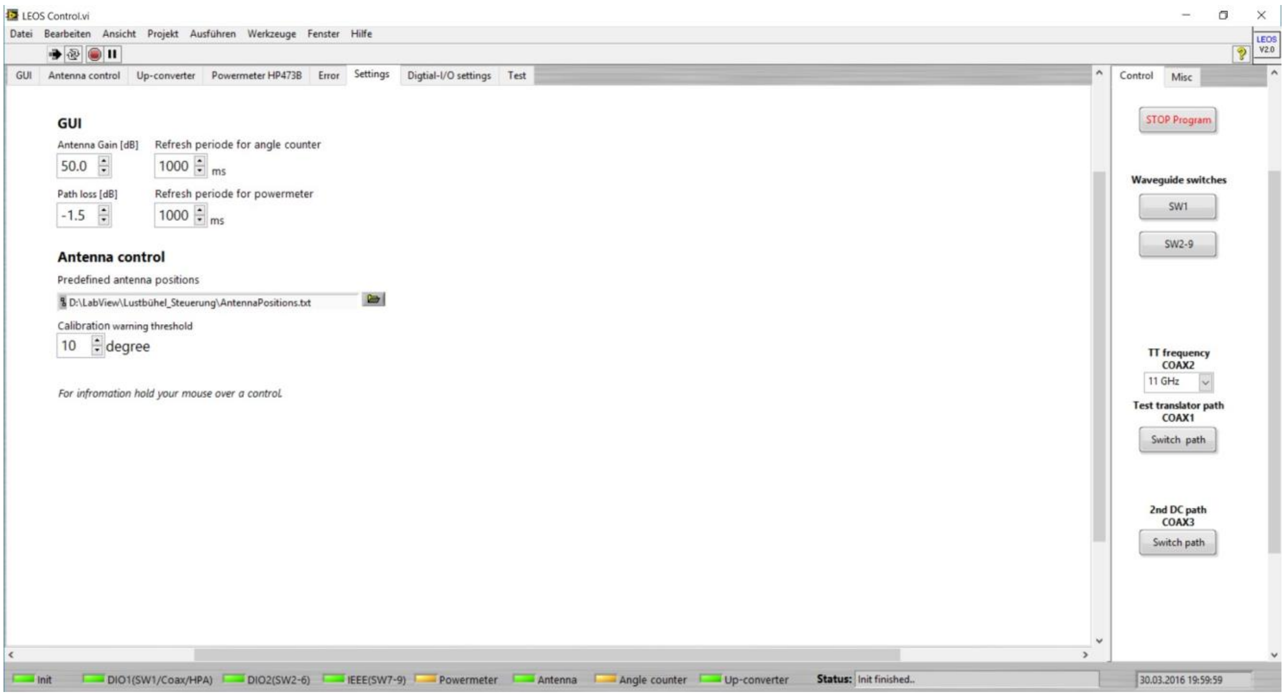
Table 3 LEOSv2 Info area status LED overview.

LED name	Meaning	Interface address	OK colour	Error colour
Init	Cumulative status for software initialization process	N/A	Green	Yellow
DIO1(SW1/Coax/HPA)	Indicator for the first NI 9403 Digital-I/O-Ethernet converter and the connected hardware: waveguide SW1, coaxial switches 1-3 and HPA	IP: 169.254.1.31	Green	Orange
DIO2(SW2-6)	Indicator for the first NI 9403 Digital-I/O - Ethernet converter and the connected hardware: waveguide SW2-6	IP: 169.254.1.32	Green	Orange
IEEE(SW7-9)	Indicator for the waveguide switches SW7-9, connected to the NI GPIB-ENET/1000 IEEE - Ethernet converter, located in the antenna box	IP: 169.254.0.22 IEEE address: SW7: “B” SW8: “A” SW9: “A”	Green	Orange
Power meter	Indicator for the Power meter, connected to the NI GPIB-ENET/1000 IEEE - Ethernet converter, located in the antenna box	IP: 169.254.0.22 IEEE address: N/A	Green	Orange
Antenna	Indicator for the antenna controller, connected to the NI GPIB-ENET/1000 IEEE - Ethernet converter, located in the station	IP: 169.254.0.21 IEEE address: N/A	Green	Orange
Angle counter	Indicator for the angle counters, connected to the Moxa NPort 5410 RS-232 – Ethernet converter, located in the station	IP: 169.254.1.11	Green	Orange
Up-converter	Indicator for the up-converter, connected to the Moxa NPort 5410 RS-232 – Ethernet converter, located in the antenna box	IP: 169.254.1.12	Green	Orange

The GUI tab control contains some tabs that are hardware specific, e.g. or the up-converter, the antenna controller and the power meter, and some tabs for general functions. The next few Figures will show the general function tabs, while the hardware specific tabs will be shown in the corresponding chapters.

4.4.1.1 Settings

Figure 4-6 shows the **Settings** tab, that contains some general settings for the LEOSv2 software.



GUI

Antenna Gain [dB] 50.0

Refresh periode for angle counter 1000 ms

Path loss [dB] -1.5

Refresh periode for powermeter 1000 ms

Antenna control

Predefined antenna positions D:\LabView\Lustbühel_Steuerung\AntennaPositions.txt

Calibration warning threshold 10 degree

For infromation hold your mouse over a control.

TT frequency COAX2 11 GHz

Test translator path COAX1

2nd DC path COAX3

Status: Init finished..

30.03.2016 19:59:59

Figure 4-6 LEOSv2 GUI, showing the Settings tab.

The provided settings are:

Antenna Gain: This setting allows to specify the antennas gain in dB, which is then used for EIRP calculation.

Path Loss: Sets the path loss for EIRP calculation in dB.

Refresh period for angle counter: Adjustment of the interval for reading the angle counters. The possible range is from 100 ms to 10 s.

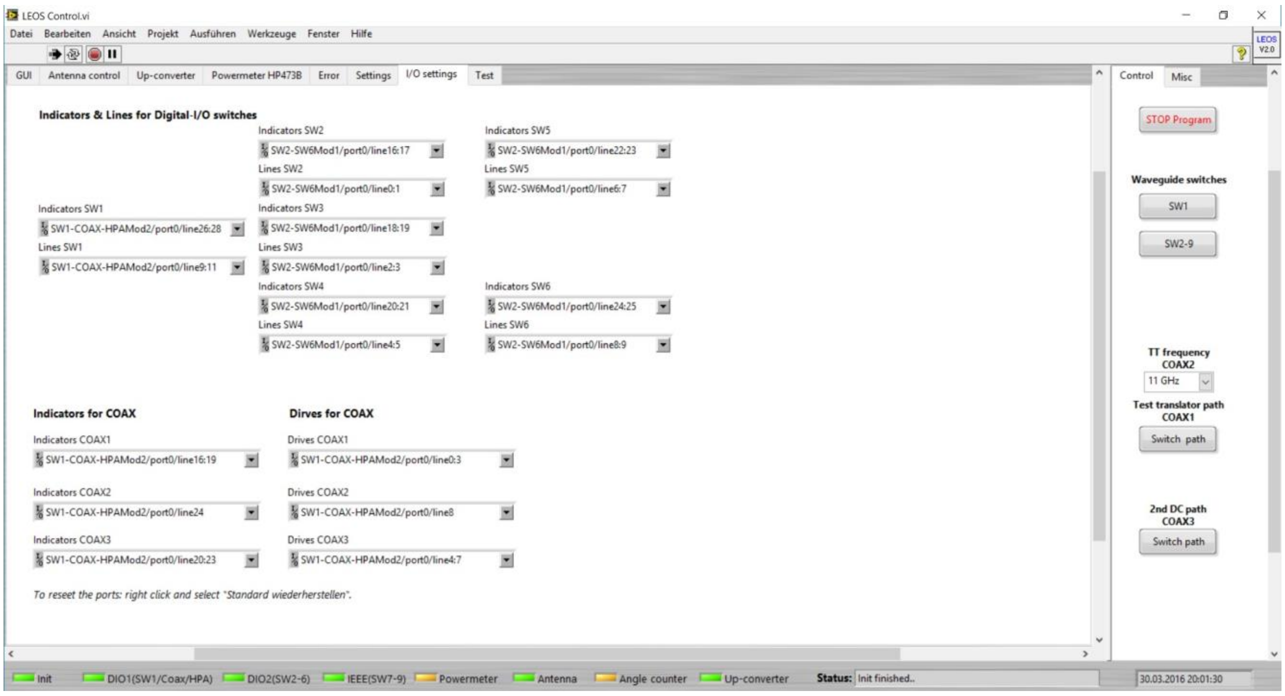
Refresh period for power meter: Adjustment of the interval for reading the power meter. The possible range is from 100 ms to 10s.

Predefined antenna positions: Path to the text file that contains a list of predefined antenna positons. This is described in detail in 4.4.4.2.

Calibration warning threshold: Sets a threshold for calibrating azimuth and elevation of the antenna controller. If one of those angles differs by more than the specified threshold from the readings of the angle counters, the user gets a warning.

4.4.1.2 I/O settings

Figure 4-7 shows the tab **I/O settings** that contains the controls to select the Digital-I/O ports for the waveguide switches and the coaxial switches on the corresponding National Instruments Digital-I/O – Ethernet converters.



Indicators & Lines for Digital-I/O switches

Indicators SW1: SW1-COAX-HPAMod2/port0/line26:28
Lines SW1: SW1-COAX-HPAMod2/port0/line9:11
Indicators SW2: SW2-SW6Mod1/port0/line16:17
Lines SW2: SW2-SW6Mod1/port0/line0:1
Indicators SW3: SW2-SW6Mod1/port0/line18:19
Lines SW3: SW2-SW6Mod1/port0/line2:3
Indicators SW4: SW2-SW6Mod1/port0/line20:21
Lines SW4: SW2-SW6Mod1/port0/line4:5
Indicators SW5: SW2-SW6Mod1/port0/line22:23
Lines SW5: SW2-SW6Mod1/port0/line6:7
Indicators SW6: SW2-SW6Mod1/port0/line24:25
Lines SW6: SW2-SW6Mod1/port0/line8:9

Indicators for COAX

Indicators COAX1: SW1-COAX-HPAMod2/port0/line16:19
Indicators COAX2: SW1-COAX-HPAMod2/port0/line24
Indicators COAX3: SW1-COAX-HPAMod2/port0/line20:23

Dirves for COAX

Drives COAX1: SW1-COAX-HPAMod2/port0/line0:3
Drives COAX2: SW1-COAX-HPAMod2/port0/line8
Drives COAX3: SW1-COAX-HPAMod2/port0/line4:7

To reseet the ports: right click and select "Standard wiederherstellen".

TT frequency COAX2: 11 GHz
Test translator path COAX1
2nd DC path COAX3

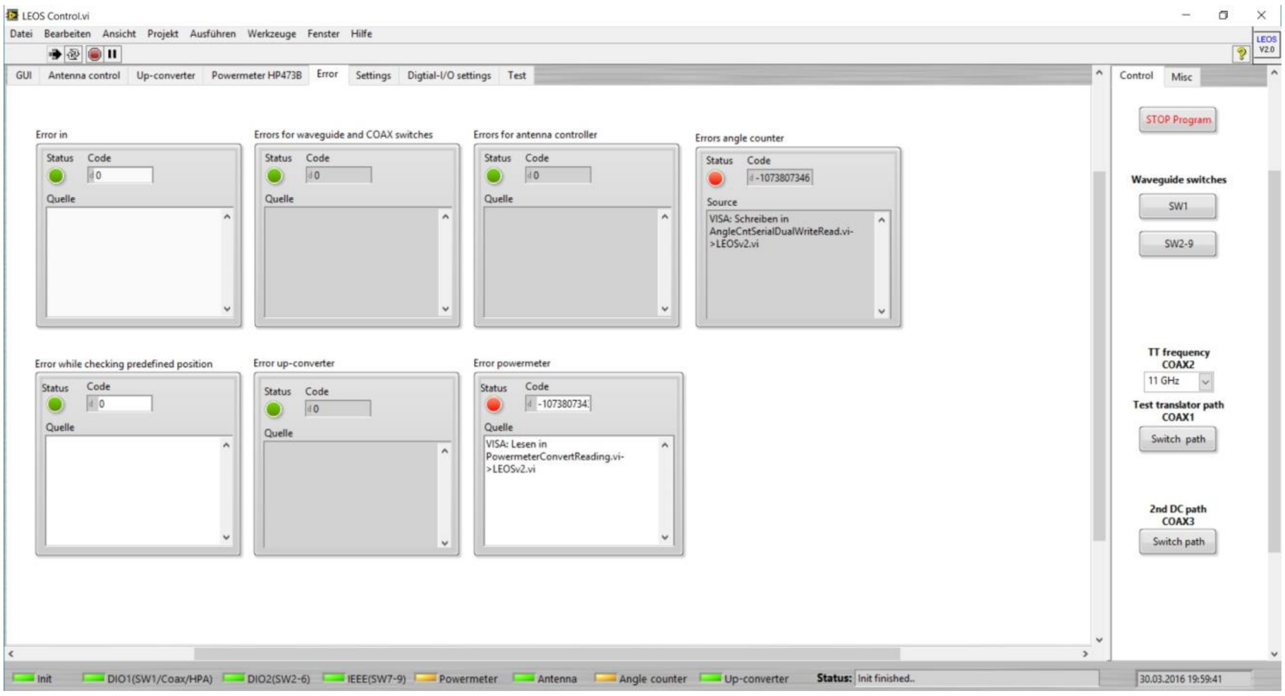
Status: Init finished..
30.03.2016 20:01:30

Figure 4-7 LEOSv2 GUI, showing the I/O settings tab.

The current configuration of the DIO ports is set as default and can be restored by right clicking on a control and selecting the option "Standard wiederherstellen".

4.4.1.3 Errors

Figure 4-8 shows the Error tab, containing a collection of the LEOSv2 error indicators. Each error indicator is assigned to one or more hardware components and is labelled accordingly.



Error in: Status ●, Code 0, Quelle (empty)

Errors for waveguide and COAX switches: Status ●, Code 0, Quelle (empty)

Errors for antenna controller: Status ●, Code 0, Quelle (empty)

Errors angle counter: Status ●, Code -1073807346, Source: VISA: Schreiben in AngleCntSerialDualWriteRead.vi->LEOSv2.vi

Error while checking predefined position: Status ●, Code 0, Quelle (empty)

Error up-converter: Status ●, Code 0, Quelle (empty)

Error powermeter: Status ●, Code -107380734, Quelle: VISA: Lesen in PowermeterConvertReading.vi->LEOSv2.vi

Status: Init finished..

30.03.2016 19:59:41

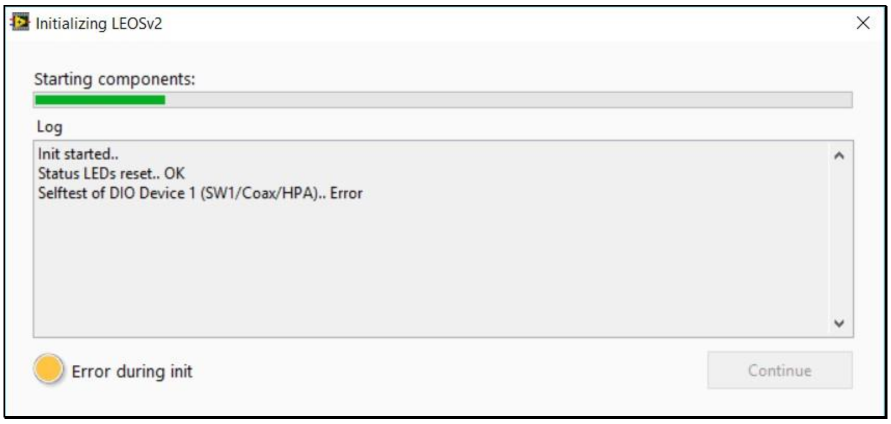
Figure 4-8 LEOSv2 GUI, showing the Error tab.

The error control **Error in** allows to set a custom error and test the software initialization process, however this may be only useful for debugging purpose and is of limited use for running operation. All other error indicators show the most recent error that occurred at the corresponding hardware. Any hardware error that is shown on this tab is also always displayed as a user dialog beforehand, so this tab serves the purpose of collecting all errors and allows the user to look up the most recent error, if the error dialog has been closed already.

4.4.1.4 Initialization

By starting LEOSv2, the software undergoes an initialization routine, resetting the GUI and checking the Ethernet converters and the connected hardware. By checking all software controlled hardware components right at the start of LEOSv2, it is made sure that the hardware works properly and that the user is notified in case of an error.

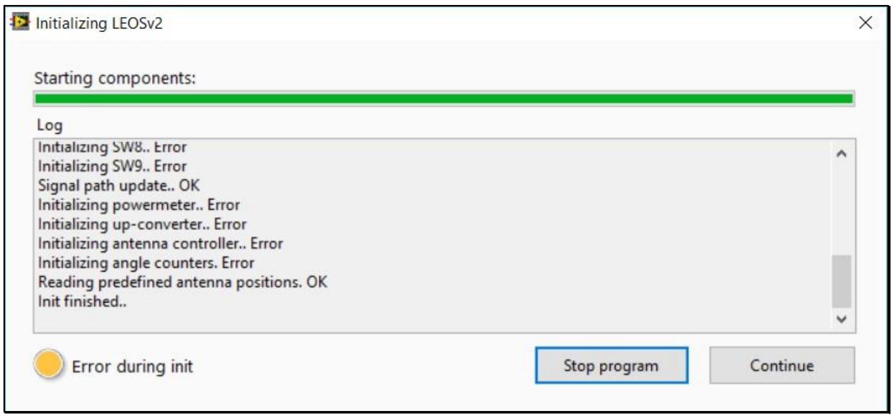
The initialization routine opens a user dialog that allows the user to keep track of the initialization progress and blocks the LEOSv2 GUI until initialization is complete. Figure 4-9 shows the initialization user dialog. The green bar on top of the dialog shows the current progress and the window **Log** shows the current initialization step and the status of this step, e.g. if the step was completed successfully (**OK**) or unsuccessfully (**Error**). Any error during the initialization routine is indicated on the LED **Error during init** at the bottom left of the dialog.



```
Init started..
Status LEDs reset.. OK
Selftest of DIO Device 1 (SW1/Coax/HPA).. Error
```

Figure 4-9 LEOSv2 initialization dialog. The initialization routine is still in progress.

Once the initialization routine finished, the button **Continue** is enabled, allowing the user to close the dialog and switch to the LEOSv2 GUI. If an error occurred during initialization, a second button **Stop program** appears, as shown in Figure 4-10.



```
Initializing SW8.. Error
Initializing SW9.. Error
Signal path update.. OK
Initializing powermeter.. Error
Initializing up-converter.. Error
Initializing antenna controller.. Error
Initializing angle counters. Error
Reading predefined antenna positions. OK
Init finished..
```

Figure 4-10 LEOSv2 initialization dialog. The initialization routine has finished with an error.

In the case an error occurred during the initialization routine, the user is provided with two options. Option one is to close the software and option two is to continue with errors. By pressing **Continue** the software is still operational, although limited to control of hardware that did not encounter an error. Allowing the software to be operational in the error case is useful if any hardware is currently not functional, if hardware is being replaced or in any other case the user still wants to control parts of the station with the LEOSv2 software.
By continuing with errors the LEDs in the GUI Info area (as described in chapter 4.4.1, Figure 4-5) are coloured accordingly. The **Init** LED is coloured yellow, meaning that one or more errors occurred during initialization and the hardware LEDs are coloured orange if the associated hardware encountered an error. Figure 4-11 shows an example, where LEOSv2 has been started without any connected hardware, leading to the case that all status LEDs show the error case.

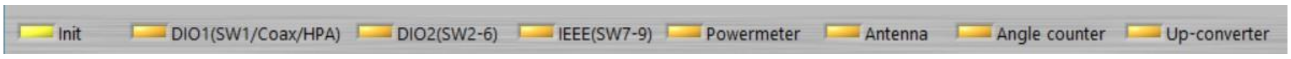


Figure 4-11 LEOSv2 GUI Info area status LEDs in the case of errors during initialization.

4.4.2 LEOSv2 main block diagram

After the LEOSv2 GUI let’s now have a look at the underlying LabVIEW block diagram. LabVIEW unfortunately has no integrated zoom function and the LEOSv2 main block diagram doesn’t fit on a single computer screen. So if you open the project on a single screen, you only see parts of the whole. To still get an overview of the complete block diagram, Figure 4-12

Figure 4-12 Schematic of the LEOSv2 block diagram. The areas and loops contain descriptions of their contained functions.

shows a schematic view of the LEOSv2 main block diagram.

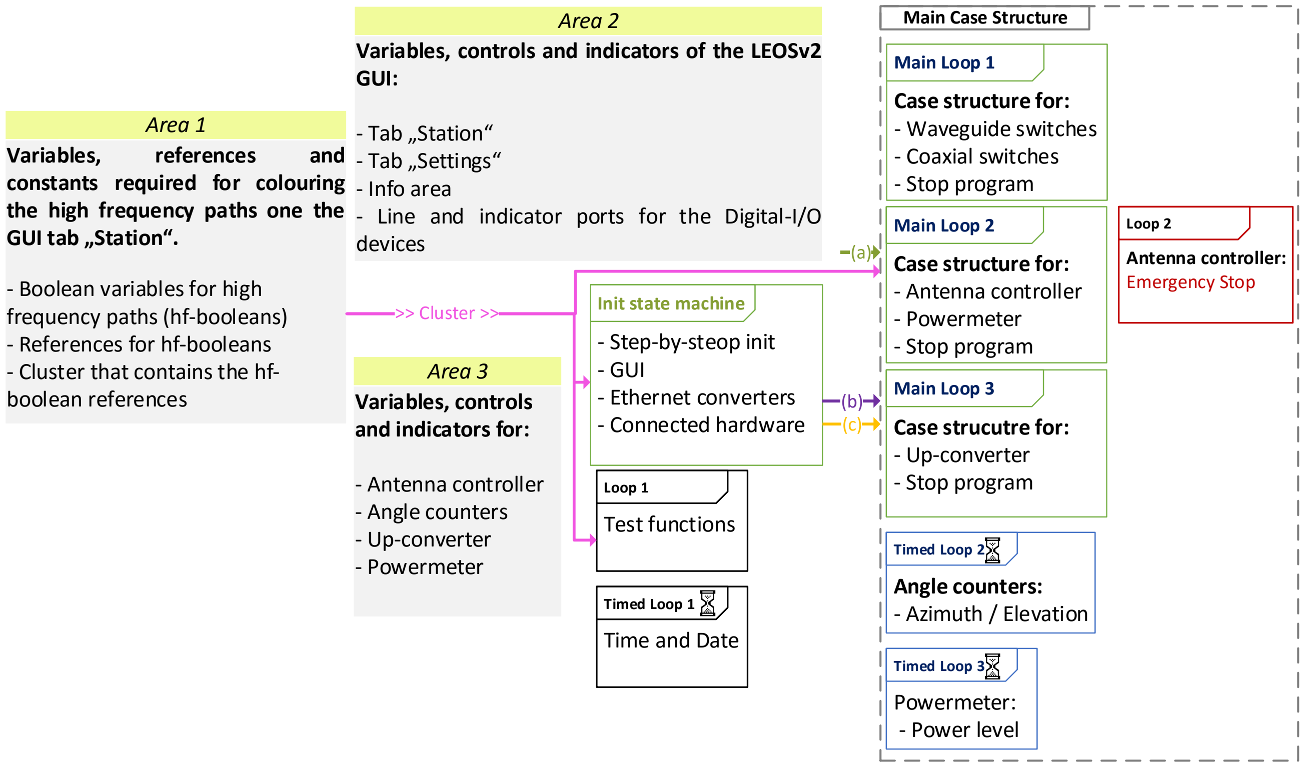


Figure 4-12 Schematic of the LEOSv2 block diagram. The areas and loops contain descriptions of their contained functions.

The different blocks are arranged the same way as they are arranged in LabVIEW and for a better understanding of how the whole block diagram looks like, all the blocks from Figure 4-12 are also marked in Figure 4-13, which shows a LabVIEW screenshot of the actual LEOSv2 block diagram.

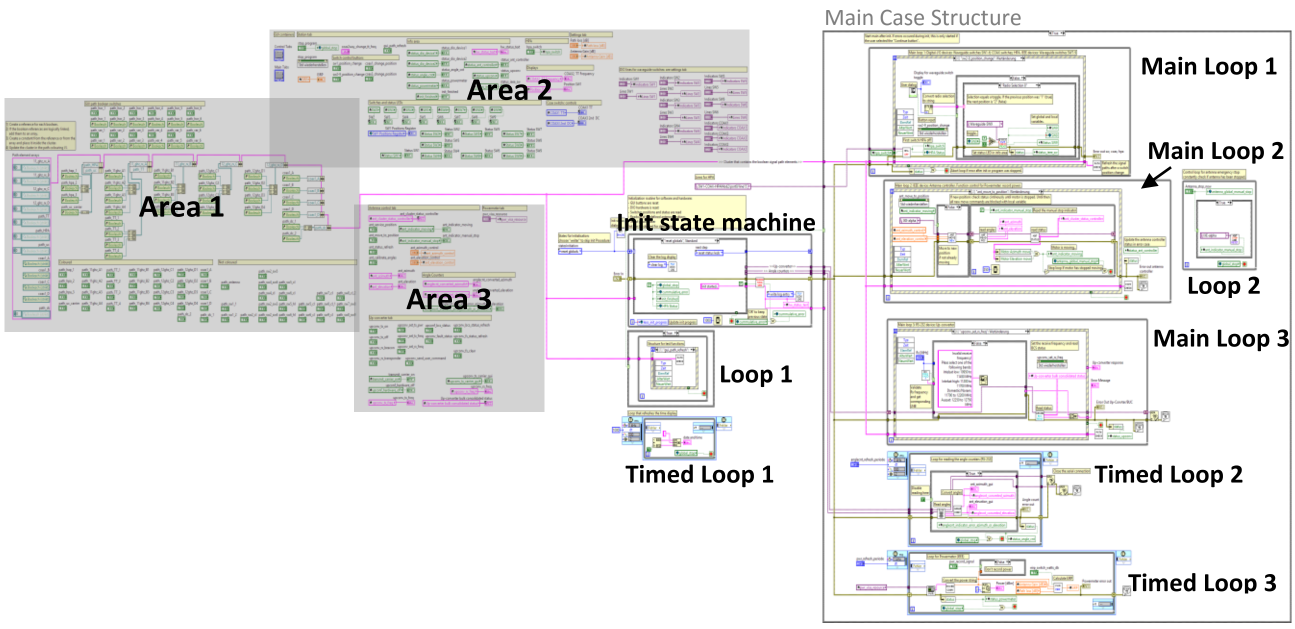


Figure 4-13 LEOSv2 block diagram overview, showing the different areas contain variables, controls and indicators. The loops control the software functionality.

The different areas are divided in smaller sub areas and labelled with short descriptive texts that describe where the variables, controls and indicators of that area belong to.

Area 1 solely contains Boolean variables and references to those variables that are used for colouring the active signal path on the GUI's **Station** tab. A description of how the signal path colouring works can be found in 4.4.10. The pink line emerging from this area contains a cluster that collects all relevant references that are used for colouring the active signal path. This cluster is passed on the the Init state machine, the Main Case Structure and to Loop1.

Area 2 contains variables, controls and indicators that are used in the LEOSv2 GUI on the **Station** tab, the **Settings** tab and the **Info area**. It also contains local variables that store the Digital-I/O ports that are used for controlling the waveguide switches and the coaxial switches.

Area 3 holds everything dedicated to a certain hardware component: variables, controls and indicators for the **Antenna controller** tab and the **Up-converter** tab, as well as for the angle counters and the power meter.

The **Init state machine**, as the name says, is a state machine handling the initialization routine. There are two arrows, **(b)** and **(c)**, that pass on resources from the initialization routine to the **Main Case Structure**. The arrow (b) represents VISA resources for the RS-232 interfaces that are required for the Up-converter and the Angle counters. Arrow (c) represents the error forwarding, and allows to pass on errors from the initialization routine to the main loops.

Loop 1 is only kept for testing and debugging and of minor importance for running operation.

Timed Loop 1 periodically refreshes the clock in the GUI.

The **Main Case Structure** is started after the initialization routine is finished. It provides two cases:

- Case "True": initialization has finished, either with or without errors and the user continued the programme. In this case all the loops within this case structure are started.
- Case "False": initialization has finished with errors and the user stopped the programme (see chapter 4.4.1.4). In this case the global variable **global_stop** is set to **True** and the programme stops.

Main Loop 1 contains an event structure that handles every event which forces a change of the configuration of the physical high frequency path, e.g. a change in position of a waveguide switch or a coaxial switch. The corresponding controls that trigger an event in this case structure are the buttons located on the **Main Controls** area (see Figure 4-5).

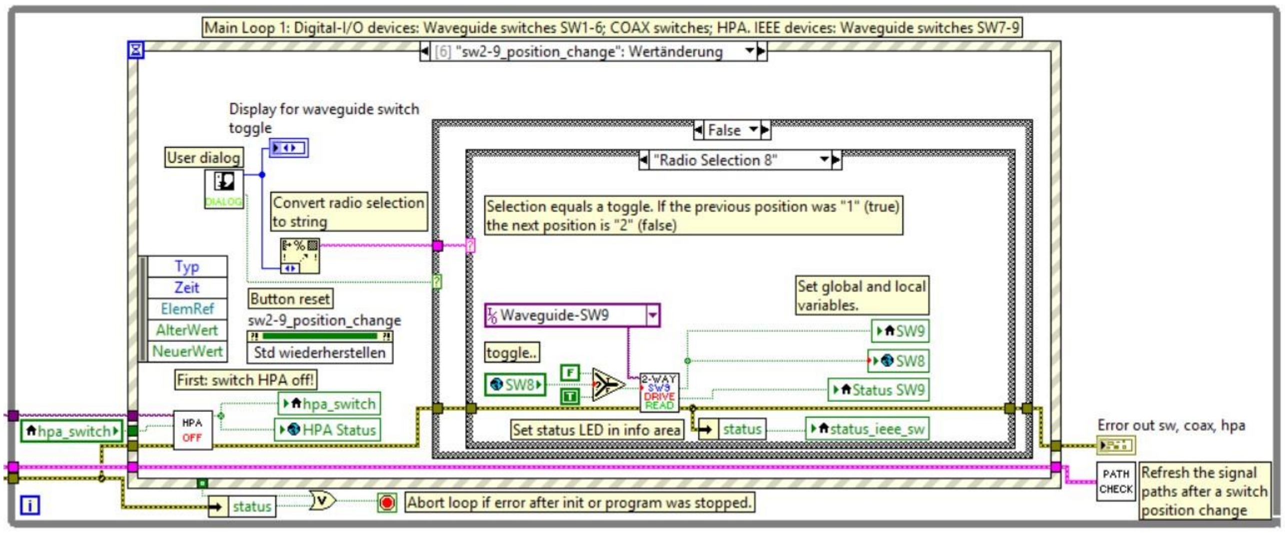

Main Loop 1: Digital-I/O devices: Waveguide switches SW1-6; COAX switches; HPA. IEEE devices: Waveguide switches SW7-9
[6] "sw2-9_position_change": Wertänderung
Display for waveguide switch toggle
User dialog
DIALOG
Convert radio selection to string
Typ
Zeit
ElemRef
AlterWert
NeuerWert
Button reset
sw2-9_position_change
Std wiederherstellen
First: switch HPA off!
hpa_switch
HPA Status
HPA OFF
False
"Radio Selection 8"
Selection equals a toggle. If the previous position was "1" (true) the next position is "2" (false)
Waveguide-SW9
toggle..
SW8
F
T
2-WAY SW9 DRIVE READ
Set global and local variables.
SW9
SW8
Status SW9
Set status LED in info area
status
status_ieee_sw
Error out sw, coax, hpa
PATH CHECK
Refresh the signal paths after a switch position change
status
Abort loop if error after init or program was stopped.

Figure 4-14 LEOSv2 main block diagram, showing the Main Loop 1.

Figure 4-14 shows the **Main Loop 1** at the example case **sw2-9_position_change**. Here we only focus only on two important aspects of Main Loop 1. First, at the lower left, you find the VI **HPA OFF**. In every case, where the configuration of the physical high frequency path is changed, the HPA is first switched off for safety. The second aspect is the VI **PATH CHECK** at the lower right. This VI checks the physical high frequency path after each event case and updates the colouring of the active signal path.

Main Loop 2 also contains a case structure, handling events that regard the antenna controller and the power meter. The important aspects of Main Loop 2 are the update of the status LED for the antenna controller (see Info Area in Figure 4-5), which is updated after each event, in case of an error. This LED is associated to the local variable **status_ant_controller** (see right in Figure 4-15, outside the case structure).

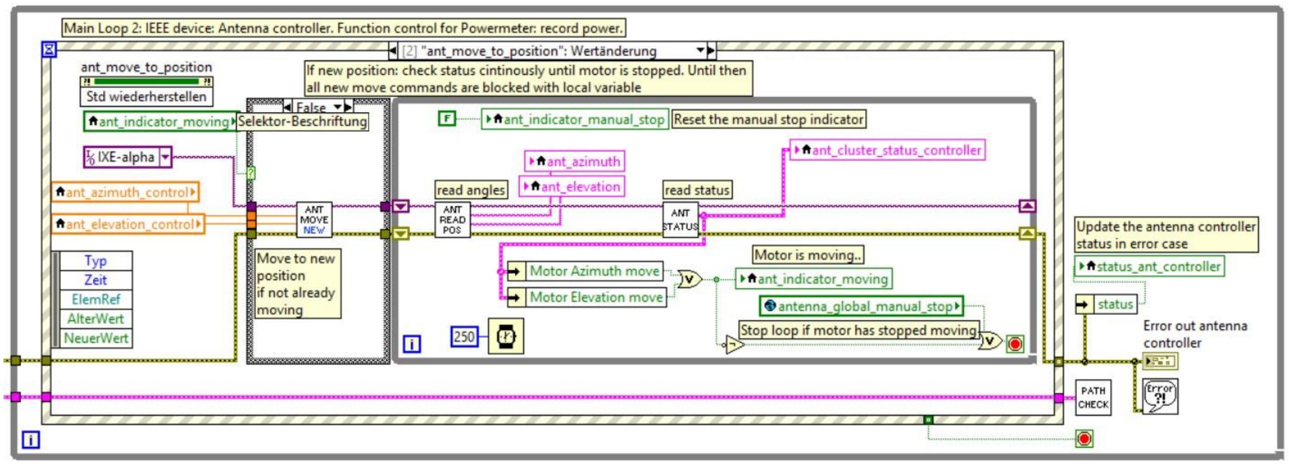

Main Loop 2: IEEE device: Antenna controller. Function control for Powermeter: record power.
[2] "ant_move_to_position": Wertänderung
If new position: check status cintinously until motor is stopped. Until then all new move commands are blocked with local variable
ant_move_to_position
Std wiederherstellen
ant_indicator_moving
Selektor-Beschriftung
False
IXE-alpha
ant_azimuth_control
ant_elevation_control
Typ
Zeit
ElemRef
AlterWert
NeuerWert
ANT MOVE NEW
Move to new position if not already moving
ant_indicator_manual_stop
Reset the manual stop indicator
ant_cluster_status_controller
ant_azimuth
ant_elevation
read angles
ANT READ POS
read status
ANT STATUS
Motor is moving..
Motor Azimuth move
Motor Elevation move
ant_indicator_moving
antenna_global_manual_stop
Stop loop if motor has stopped moving
250
Update the antenna controller status in error case
status_ant_controller
status
Error out antenna controller
PATH CHECK

Figure 4-15 LEOSv2 main block diagram, showing the Main Loop 2.

This is now a good place to introduce **Loop 2**, because it is also related to the antenna controller. This loop is dedicated to the emergency stop button **STOP** on the Antenna controller tab, shown in Figure 4-16.

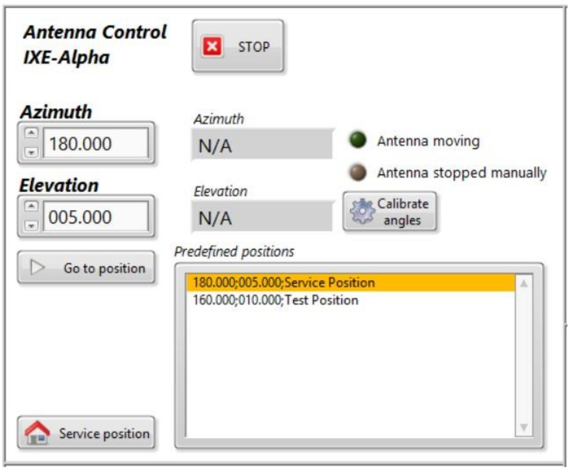
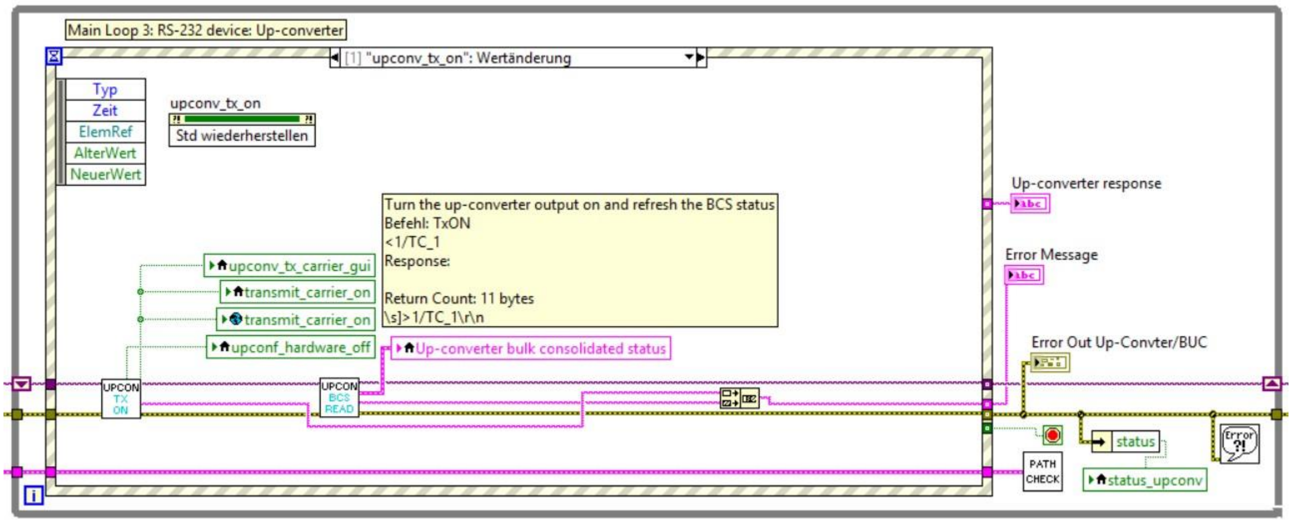


Figure 4-16 LEOSv2 GUI, showing a section of the tab "Antenna control" and the STOP button on top.

If this button is pressed, the LED **Antenna stopped manually** flashes up, indicating that the antenna was stopped during movement and therefore might be not in the desired position.

Main Loop 3 contains a case structure dedicated to handle events regarding the up-converter. Figure 4-17 shows the example case **upconv_tx_on** and at the right, outside the case structure, the up-converter status LED (see GUI Info Area in Figure 4-5) is updated in the event of an error. The associated local variable is called **status_upconv** (lower right)**.**

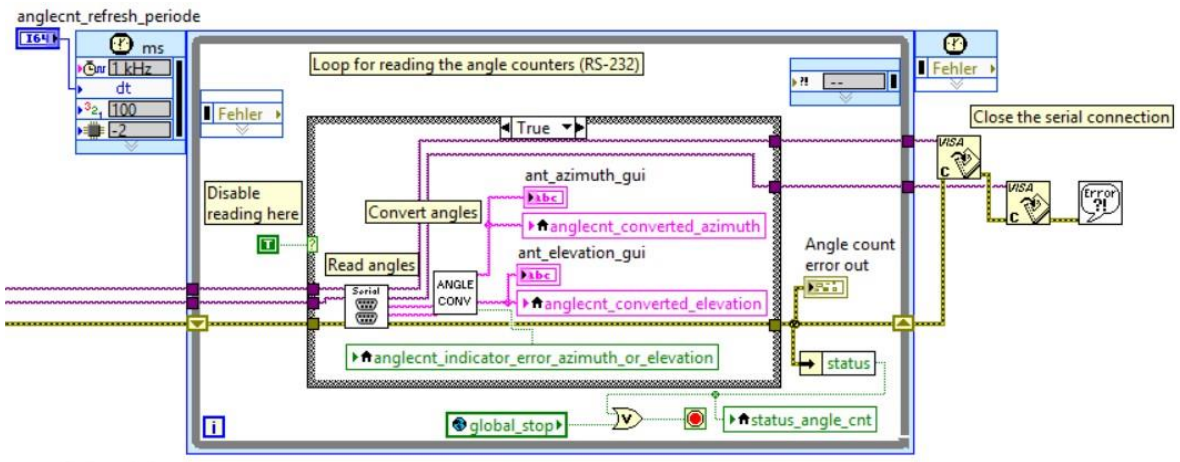

Turn the up-converter output on and refresh the BCS status
Befehl: TxON
<1/TC_1
Response:

Return Count: 11 bytes
\s]>1/TC_1\r\n

Figure 4-17 LEOSv2 main block diagram, showing the Main Loop 3.

For more information about what happens in the main loops please see the following chapters which provided detailed descriptions of the hardware related functions that are handled in each main loop.

Timed Loop 2 periodically reads the angle counters and updates the corresponding GUI status LED that is associated with the local variable **status_angle_cnt**, shown in Figure 4-18 to the lower right.



anglecnt_refresh_periode
Loop for reading the angle counters (RS-232)
Close the serial connection
Disable reading here
Convert angles
Read angles
ant_azimuth_gui
anglecnt_converted_azimuth
ant_elevation_gui
anglecnt_converted_elevation
Angle count error out
anglecnt_indicator_error_azimuth_or_elevation
status
global_stop
status_angle_cnt
Fehler
True

Figure 4-18 LEOSv2 main block diagram, showing the Timed Loop 2.

The functions with in Timed Loop 2 are described in detail in chapter 4.4.3.

Timed Loop 2 periodically reads the power meter and updates the corresponding GUI status LED that is associated with the local variable **status_powermeter**, shown in Figure 4-19 in the lower center.

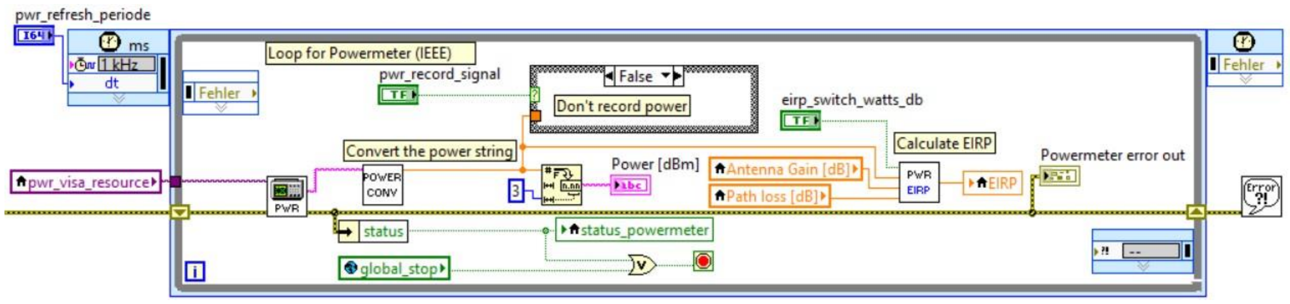
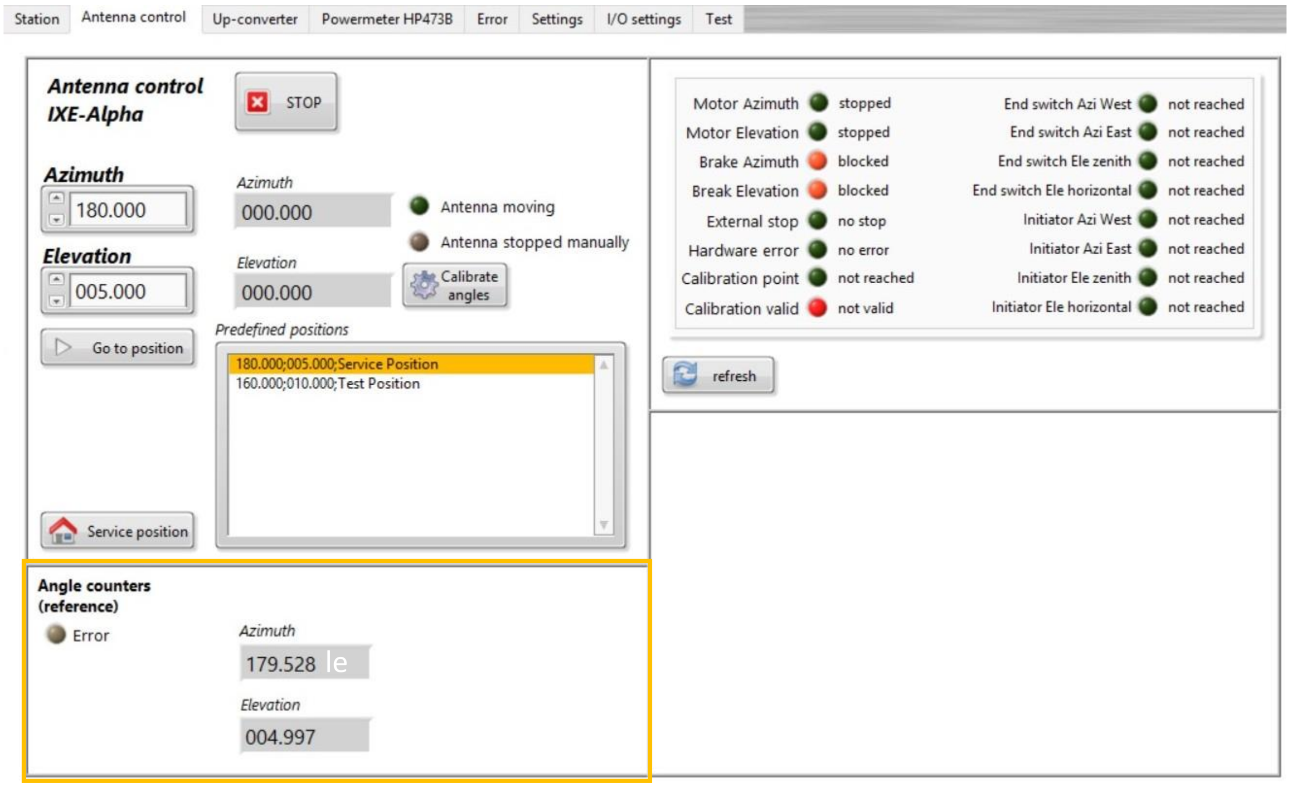


Figure 4-19 main block diagram, showing the Timed Loop 3.

The functions with in Timed Loop 1 are described in detail in chapter 4.4.7.

4.4.3 Angle counters

The Heidenhain angle counters are used as reference angles for azimuth and elevation for the Phytron IXE-alpha antenna controller. The readings of the angle controller can be used to calibrate the antenna controller. The user interface for the angle counters is integrated in the tab **Antenna controller** in the main GUI which is shown in Figure 4-20. The orange frame surrounds the section for the angle counters and shows the reference angles for azimuth and elevation. Errors during communication with the angle counter are indicated by the **Error** LED.


Station | Antenna control | Up-converter | Powermeter HP473B | Error | Settings | I/O settings | Test

Antenna control IXE-Alpha

STOP

Azimuth 180.000

Elevation 005.000

Go to position

Azimuth 000.000

Elevation 000.000

Antenna moving

Antenna stopped manually

Calibrate angles

Predefined positions

180.000;005.000;Service Position
160.000;010.000;Test Position

Service position

Motor Azimuth	stopped	End switch Azi West	not reached
Motor Elevation	stopped	End switch Azi East	not reached
Brake Azimuth	blocked	End switch Ele zenith	not reached
Break Elevation	blocked	End switch Ele horizontal	not reached
External stop	no stop	Initiator Azi West	not reached
Hardware error	no error	Initiator Azi East	not reached
Calibration point	not reached	Initiator Ele zenith	not reached
Calibration valid	not valid	Initiator Ele horizontal	not reached

refresh

Angle counters (reference)

Error

Azimuth 179.528

Elevation 004.997

Figure 4-20 LEOSv2 GUI: Front panel of the Antenna control tab. The controls and indicators for the angle counters are highlighted.

The angle counters are polled with the help of the **Timed Loop 2** that can be found in the LEOSv2 main block diagram (see previous chapter).

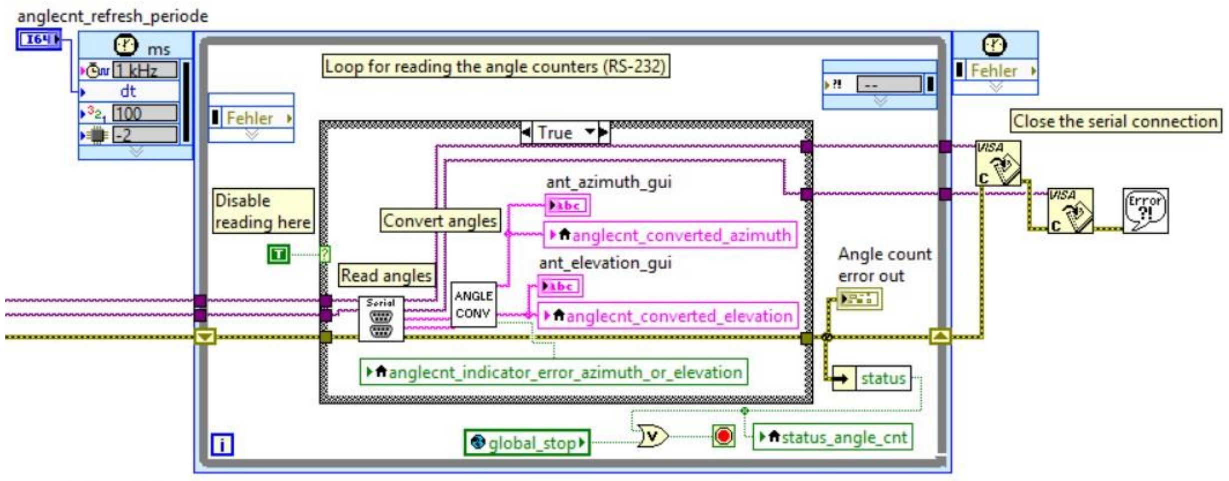


Figure 4-21 Timed loop with in the block diagram of the GUI for reading the angle counters.

Starting from the left, the COM ports for both angle references are opened. The Sub-VI **ANGLE SERIAL INIT** then configures the serial interface with the following settings:

Baud: 9600

Data bits: 7

Parity: even

Stop bits: 2

Flow control: none

After configuring the serial ports, the loop for reading the angles is started. The readout interval refreshing the antenna's azimuth and elevation positions can be controlled by the control **anlecnt_refresh_periode.** Reading the angles is done with the Sub-VI **Serial.** This VI uses a state machine for writing a command for two serial interfaces and then reads the response from the tow serial interfaces one after the other.

The command for reading the angles from the angle counters is the ASCII command **Start Of Text (STX)**. The hexadecimal representation of **STX** is **0x02**. To send the hexadecimal number 0x02 to a serial interface in LabVIEW, a string constant has been used and configured to display hexadecimal numbers. This can be configured in the string constant's context menu by selecting the option "Hexadezimalanzeige".

The angle counters response to a **STX** command is its current angle reading in a 17 byte long format. Starting with a '+' sign, the angle reading is preceded and followed by spaces and is enclosed with a carriage return and a line feed (\r\n). Two example readings are now shown:

Azimuth example: +\s\s179.528.6\s\s\s\r\n

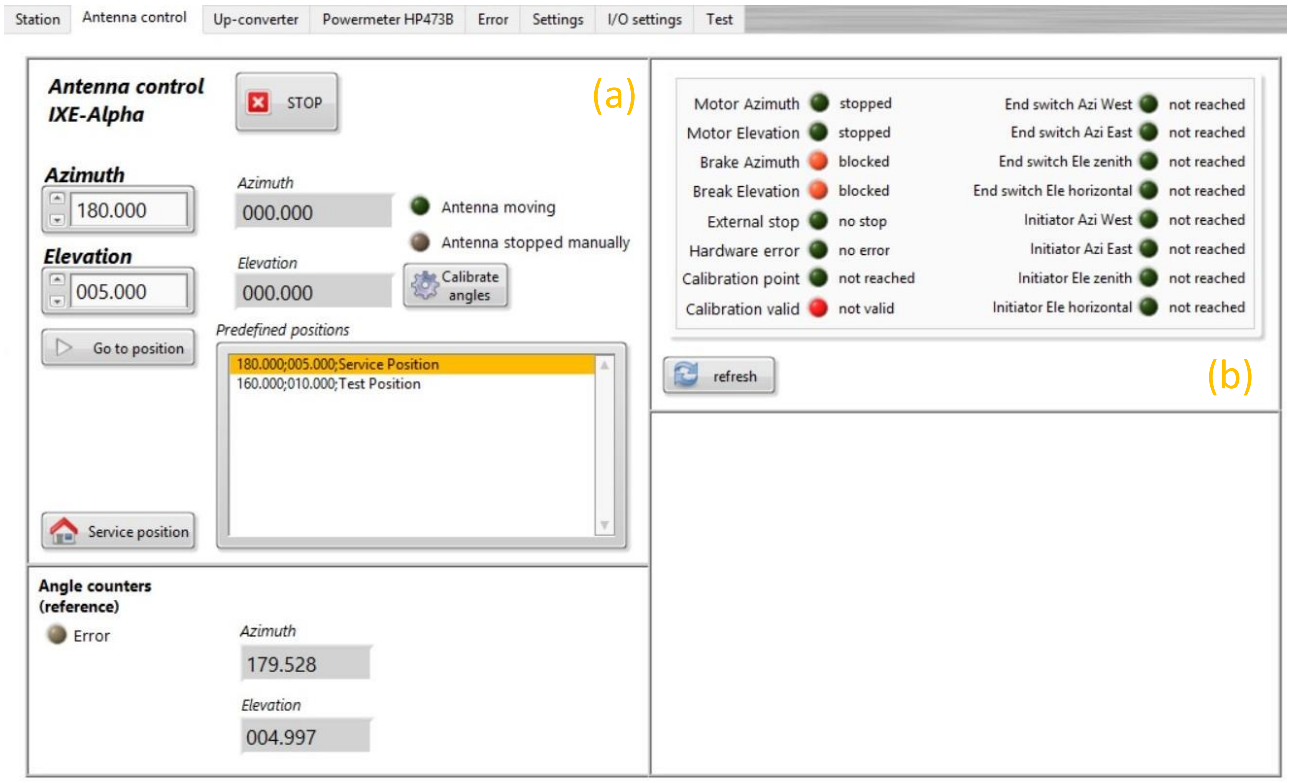
Elevation example: +\s\s\s\s4.997.5\s\s\s\r\n

The angle is returned as a 7-9 byte long number and is followed by 4 spaces (\s). The preceding spaces vary from 2-4, depending on the size of the angle and always fill the whole serial response to a total length of 17 bytes.

The Sub-VI **ANLGE CONV** checks if the response from the angle counters is valid. To do so, first the length of the serial response is checked. If the length is 17 bytes, and the first byte is a '+' sign, the angles are extracted, converted to strings and returned to the indicators **anglecnt_converted_azimuth** and **anglecnt_converted_elevation**. If the LEOSv2 software is stopped, the VISA-Resources for the serial ports are released.

4.4.4 Antenna controller

The Phytron IXE-alpha antenna controller controls the antenna movement in azimuth and elevation. It reads the antenna's current position and provides various status flags regarding antenna movement, position, end positions and calibration. The user interface for the antenna controller can be found in the main GUI at the tab **Antenna control**. Figure 4-22 shows the main functions of the antenna controller.



Antenna control IXE-Alpha

Azimuth (set)	180.000
Elevation (set)	005.000
Azimuth	000.000
Elevation	000.000
●	Antenna moving
●	Antenna stopped manually

Predefined positions
180.000;005.000;Service Position
160.000;010.000;Test Position

Angle counters (reference)
● Error

Azimuth	179.528
Elevation	004.997

Motor Azimuth ●	stopped	End switch Azi West ●	not reached
Motor Elevation ●	stopped	End switch Azi East ●	not reached
Brake Azimuth ●	blocked	End switch Ele zenith ●	not reached
Break Elevation ●	blocked	End switch Ele horizontal ●	not reached
External stop ●	no stop	Initiator Azi West ●	not reached
Hardware error ●	no error	Initiator Azi East ●	not reached
Calibration point ●	not reached	Initiator Ele zenith ●	not reached
Calibration valid ●	not valid	Initiator Ele horizontal ●	not reached

Figure 4-22 LEOSv2 Antenna control tab. (a) Functions for antenna repositioning, angle readings, calibration button, indicators for antenna movement, predefined antenna positions and the emergency stop button. (b) Antenna status indicators.

Figure 4-22a shows the controls and indicators for the antenna controller. At the left, the numerical control elements **Azimuth** and **Elevation** allow the selection of a new antenna position. To move the antenna to the selected position, the button **Go to positon** will trigger a user dialog, where the new position has to be confirmed. **Predefined positions** can be selected by double clicking one of the listed entries.

While the antenna is moving, the angles and the status indicators are periodically updated and the LED **Antenna moving** turns green. The movement of the antenna can be stopped at any time by pressing the emergency stop button **STOP**. This will cause an immediate halt of the antenna and cancels all further commands. If the stop button is pressed, the LED **Antenna stopped manually** will be turned on until a new move command is sent.

Figure 4-22b shows the antenna controller status indicators. These indicators are refreshed automatically during antenna movement and can be refreshed manually using the **Refresh** button below the indicators. The purpose of each status indicator is listed in Table 4.

Table 4 Overview of the antenna controller status indicators.

Indicator	***LED On***	***LED Off***
Motor Azimuth	Motor is moving in azimuth	No movement in azimuth
Motor Elevation	Motor is moving in elevation	No movement in elevation
Break Azimuth	Azimuth movement is blocked	Azimuth free to move
Break Elevation	Elevation movement is blocked	Elevation free to move
External stop	Movement has been stopped manually, not within the software (External emergency halt button)	No external stop
Hardware error	An hardware error occurred	No error
Calibration point	Antenna has reached the calibration point	Antenna not at calibration point
Calibration valid	The current antenna calibration is valid	The current antenna calibration is not valid
End switch Azi West	Endpoint in azimuth west direction has been reached	Azimuth west endpoint not reached
End switch Azi East	Endpoint in azimuth east direction has been reached	Azimuth east endpoint not reached
End switch Ele zenith	Endpoint in elevation zenith direction has been reached	Elevation zenith endpoint not reached
End switch Ele horizontal	Endpoint in elevation horizontal direction has been reached	Elevation horizontal endpoint not reached
Initiator Azi West	Initiator in azimuth west has been reached	Azimuth west initiator not reached
Initiator Azi East	Initiator in azimuth east has been reached	Azimuth east initiator not reached

Initiator Ele zenith	Initiator in elevation zenith has been reached	Elevation zenith initiator not reached
Initiator Ele horizontal	Initiator in elevation horizontal has been reached	Elevation horizontal initiator not reached.

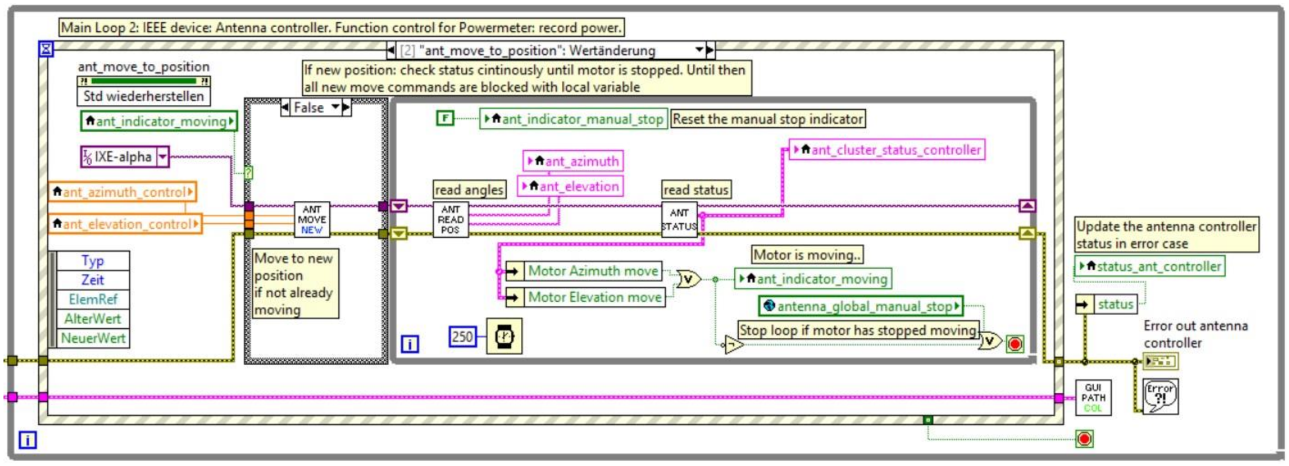
User interactions on the Antenna control tab are handled in an event structure within the **Main Loop 2** of the GUI block diagram. The possible user actions are:

- Moving the antenna to a new position
- Moving the antenna to a predefined positions
- Moving the antenna to the service position
- Refreshing the antenna controller status indicators
- Calibrating azimuth and elevation

4.4.4.1 Moving the antenna to a new position

To move the antenna to a new position, the event case **ant_move_to_position** is triggered.

Figure 4-23 shows **Main Loop 2**, a loop that is part of the LEOSv2 main block diagram and contains the event structure that handles user inputs for the antenna controller, the HPA and the power meter. The figure shows the case for pressing the button **Go to position** on the Antenna control tab in the GUI. The event is labelled **ant_move_to_position** and is used for moving the antenna to a new azimuth and elevation, whilst periodically checking the current readings of the antenna controller status flags and the angle readings.



Main Loop 2: IEEE device: Antenna controller. Function control for Powermeter: record power.
[2] "ant_move_to_position": Wertänderung
ant_move_to_position
Std wiederherstellen
ant_indicator_moving
IXE-alpha
ant_azimuth_control
ant_elevation_control
Typ
Zeit
ElemRef
AlterWert
NeuerWert
If new position: check status cintinously until motor is stopped. Until then all new move commands are blocked with local variable
False
ANT MOVE NEW
Move to new position if not already moving
ant_indicator_manual_stop
Reset the manual stop indicator
ant_azimuth
ant_elevation
read angles
ANT READ POS
read status
ANT STATUS
ant_cluster_status_controller
Motor is moving..
Motor Azimuth move
Motor Elevation move
ant_indicator_moving
antenna_global_manual_stop
Stop loop if motor has stopped moving
250
Update the antenna controller status in error case
status_ant_controller
status
Error out antenna controller
GUI PATH

Figure 4-23 Event case for moving the antenna to a new position. The loop containing the event structure is part of the main LEOSv2 block diagram.

Within the event case, starting from the left, a case structure checks if the antenna isn't already moving. If the antenna isn't moving, a new move command is sent to the antenna controller, using the controls **ant_azimuth_control** and **ant_elevation_control** as inputs for the Sub-VI **ANT MOVE NEW**. After the move command is sent a while loop is started and periodically reads the antenna controller status and the antenna controller azimuth and elevation angles with the help of the Sub-VIs **ANT STATUS** and **ANT READ ANGLES**. The loop is stopped if movement in azimuth and movement in elevation has stopped.

Before the move command is sent to the antenna controller a user dialog is shown that allows the user to check the new azimuth and elevation angles before moving the antenna. Figure 4-24 shows the block diagram of the Sub-VI **ANT MOVE NEW**, which creates the user dialog.

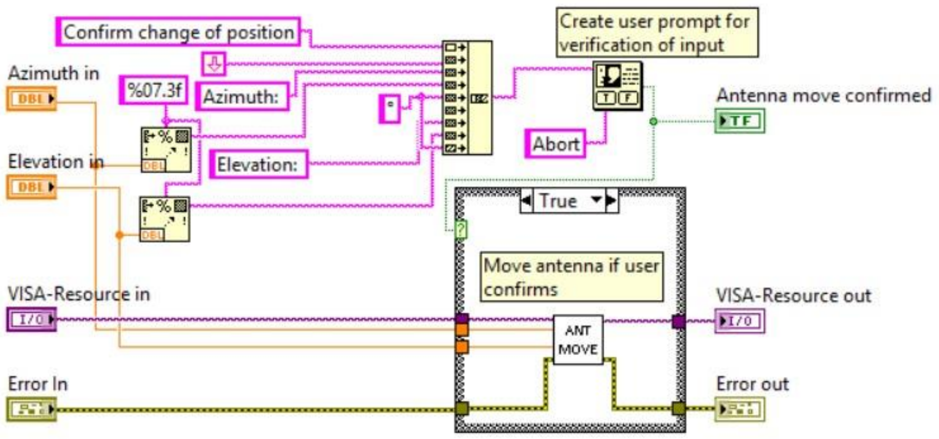


Figure 4-24 Block diagram of the VI AntennaMoveToPosition (ANT MOVE NEW). This VI creates a user dialog before the new positions are sent to the antenna controller.

If the user confirms the new angles, a move command is sent to the antenna controller. A move command actually consists of a series of sub commands that are handled in the Sub-VI **ANT MOVE** (Figure 4-25).

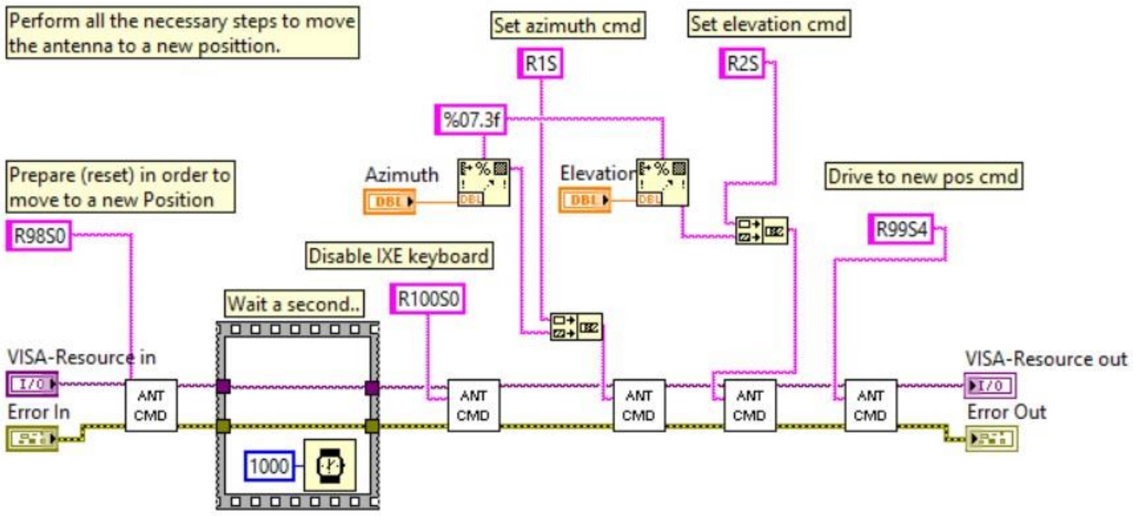


Figure 4-25 Block diagram of the VI AntMove (ANT MOVE). This VI handles the necessary commands for moving the antenna to a new position.

The sequence of commands starts with a reset to prepare the controller for a new move command. The reset command is the string '**R98S0**'. After a 1000 ms delay, the antenna controller's hardware keyboard is disabled with the command '**R100S0**'. Now azimuth and elevation can be set with the string '**R1Sxxx.xxx**' where xxx.xxx is the azimuth in degrees. The elevation command uses the prefix R2S and the full command is '**R2Syyy.yyy**' with yyy.yyy being the elevation in degrees. After the new values for azimuth and elevation have been set, the command '**R99S4**' confirms the new angles and starts the antenna movement.

Each command for the antenna controller is handled within the Sub-VI **ANT CMD**. Its block diagram is shown in Figure 4-26.

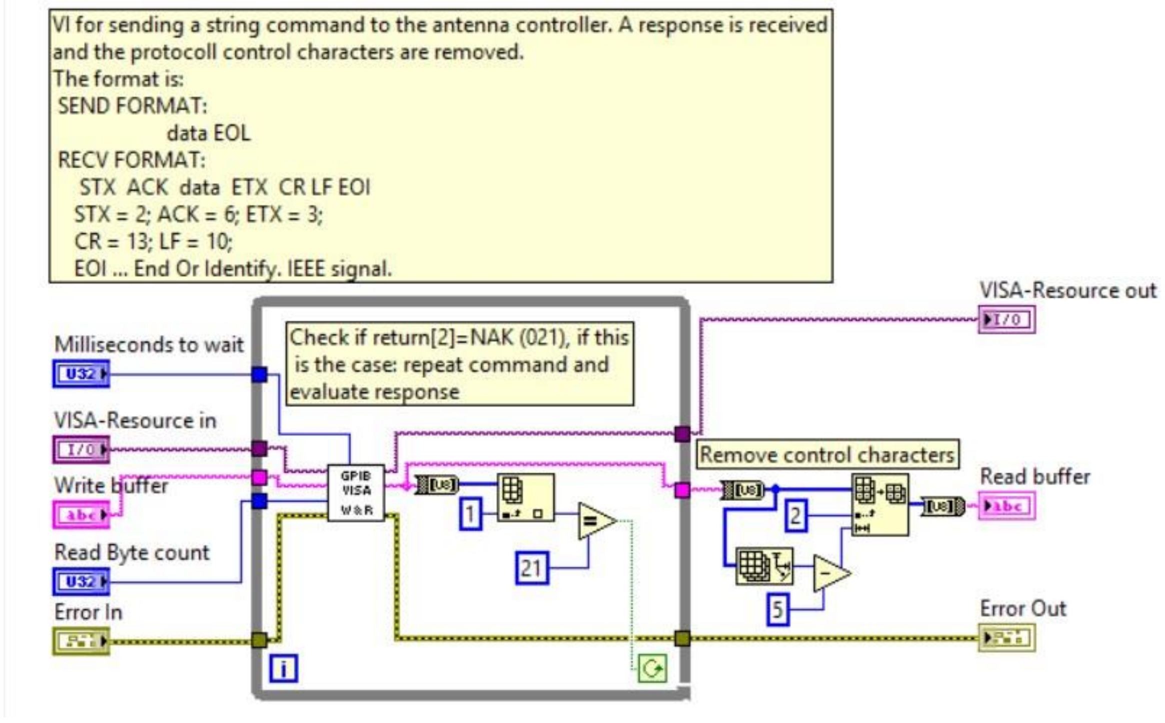

VI for sending a string command to the antenna controller. A response is received and the protocoll control characters are removed.
The format is:
SEND FORMAT:
data EOL
RECV FORMAT:
STX ACK data ETX CR LF EOI
STX = 2; ACK = 6; ETX = 3;
CR = 13; LF = 10;
EOI ... End Or Identify. IEEE signal.
Milliseconds to wait
VISA-Resource in
Write buffer
Read Byte count
Error In
Check if return[2]=NAK (021), if this is the case: repeat command and evaluate response
GPIB VISA W&R
1
21
Remove control characters
2
5
VISA-Resource out
Read buffer
Error Out

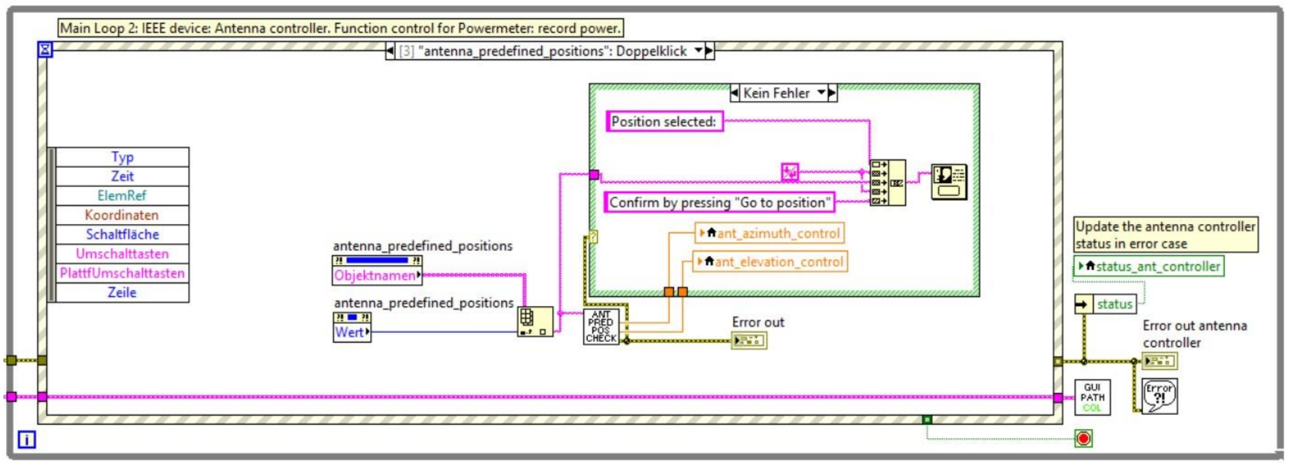
Figure 4-26 Block diagram for the VI AntCommand (ANT CMD).

Communication with the antenna controller is ended with an “acknowledged” (ACK) or “not acknowledged” (NAK). If a command is not acknowledged, the command needs to be repeated until the antenna controller responds with an ACK (see while loop in Figure 4-26). If an ACK is received, the control characters are removed from the response string and the raw response is returned at the **Read buffer**.

The Sub-VI **GPIB VISA W&R** handles the actual communication with the antenna controller over a VISA-Resource, using a state machine to write a command to the antenna controller and afterwards reads the corresponding response.

4.4.4.2 *Moving the antenna to a predefined position*

Using a predefined position for antenna movement doesn't differ much from entering a new position manually via the azimuth and elevation controls. By double clicking an entry in the predefined positions list (see Figure 4-22a) the event case **antenna_predefined_positions** is triggered the predefined position is set at the azimuth and elevation controls. Figure 4-27 depicts the event case.



Main Loop 2: IEEE device: Antenna controller. Function control for Powermeter: record power.
[3] "antenna_predefined_positions": Doppelklick
Typ
Zeit
ElemRef
Koordinaten
Schaltfläche
Umschalttasten
PlattfUmschalttasten
Zeile
antenna_predefined_positions
Objektnamen
antenna_predefined_positions
Wert
Kein Fehler
Position selected:
Confirm by pressing "Go to position"
ant_azimuth_control
ant_elevation_control
ANT PRED POS CHECK
Error out
Update the antenna controller status in error case
status_ant_controller
status
Error out antenna controller
GUI PATH COL

Figure 4-27 Event case for moving the antenna to a predefined position.

First the predefined positions are checked for a correct format of the entries. If the format is correct, the user gets a message that the positions are valid and that the antenna can now be moved by pressing the **Go to position** button (see Figure 4-22a).

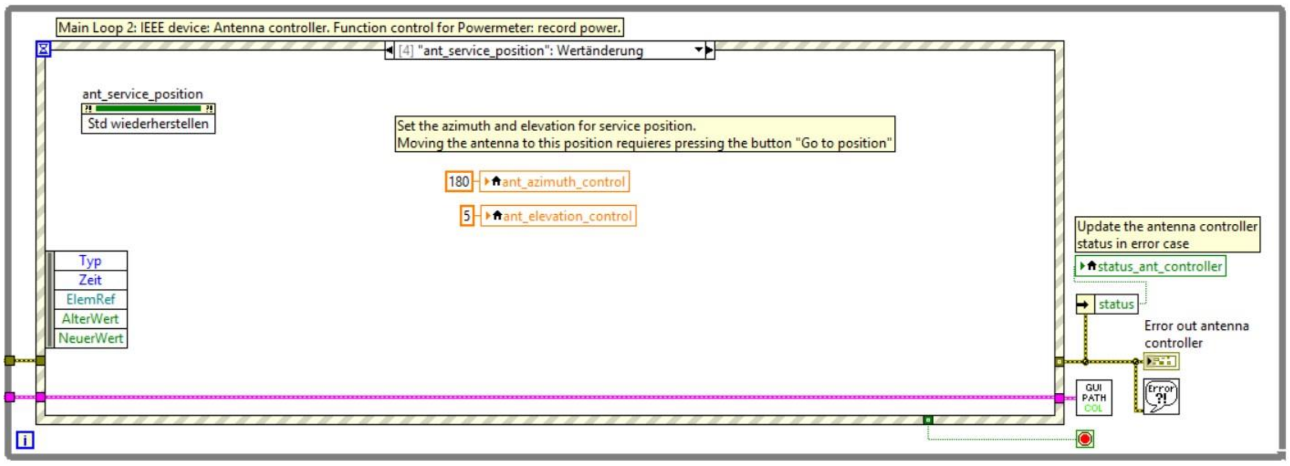
The predefined positions are stored in a text file in the format **xxx.xxx;yyy.yyy;<description>\r\n**. The azimuth is represented by xxx.xxx in degrees and separated by a semicolon from the elevation yyy.yyy. After the angles a description for the position can be entered. Entries are separated by a carriabe return. Below there is an example of the positions text file, that has been used for creating the entries in the predefined positions list in Figure 4-22a:

180.00;005.000;Service Position<cr>

160.000;010.000;Test Position<cr>

4.4.4.3 Moving the antenna to the service position

The antenna's service position is set to 180 degree in azimuth and 5 degree in elevation. This position allows for best access to the hardware box mounted on the antenna. Figure 4-28 shows the event case **ant_service_position**. This event is triggered by pressing the button **Service position** (see Figure 4-22a) and simply sets the local variables for the azimuth and elevation controls **ant_azimuth_control** and **ant_elevation_control** to the required values. Alternatively the service position entry of the predefined positions list can be used.

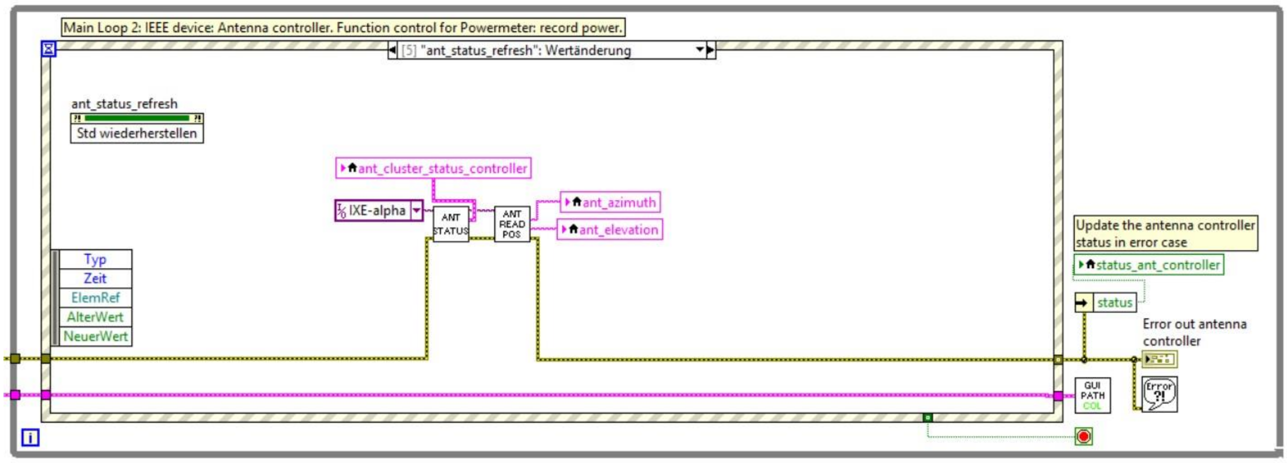


Main Loop 2: IEEE device: Antenna controller. Function control for Powermeter: record power.
[4] "ant_service_position": Wertänderung
ant_service_position
Std wiederherstellen
Set the azimuth and elevation for service position.
Moving the antenna to this position requieres pressing the button "Go to position"
180
ant_azimuth_control
5
ant_elevation_control
Typ
Zeit
ElemRef
AlterWert
NeuerWert
Update the antenna controller status in error case
status_ant_controller
status
Error out antenna controller
GUI PATH COL
Error ?!

Figure 4-28 Event case for moving the antenna to the service position.

4.4.4.4 Refreshing the antenna controller status indicators manually

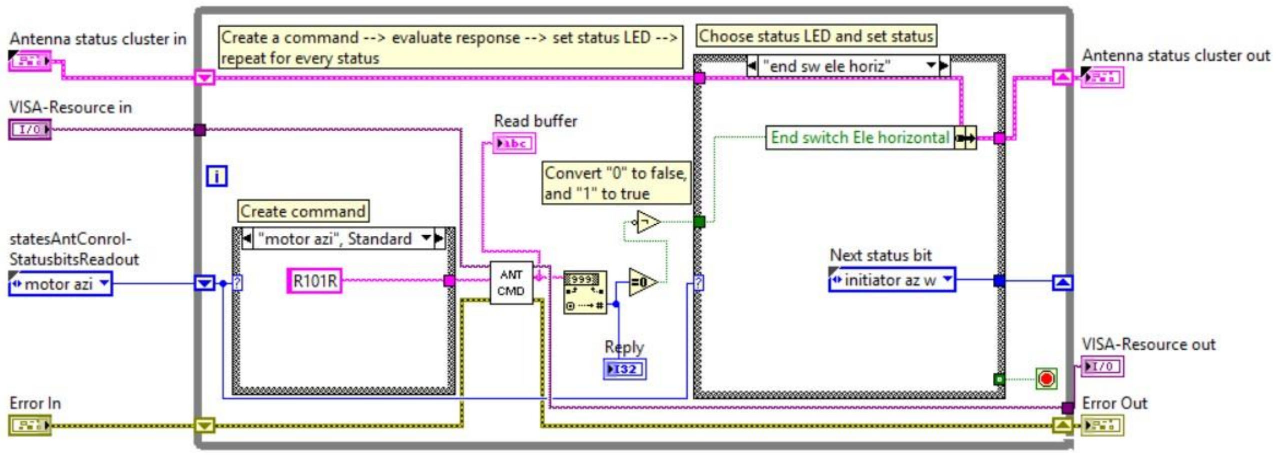
Figure 4-29 shows the event case for refreshing the antenna controller status indicators manually by pressing the **Refresh** button (see Figure 4-22b). First the status indicators are updated, afterwards the angles from the antenna controller are refreshed.



Main Loop 2: IEEE device: Antenna controller. Function control for Powermeter: record power.
[5] "ant_status_refresh": Wertänderung
ant_status_refresh
Std wiederherstellen
ant_cluster_status_controller
IXE-alpha
ANT STATUS
ANT READ POS
ant_azimuth
ant_elevation
Typ
Zeit
ElemRef
AlterWert
NeuerWert
Update the antenna controller status in error case
status_ant_controller
status
Error out antenna controller
GUI PATH

Figure 4-29 Event case for refreshing the antenna controller status indicators.

The status indicators are collected in the cluster **ant_cluster_status_controller**. The status indicators are read one after another in the Sub-VI **ANT STATUS**, which is shown in Figure 4-30.



Antenna status cluster in
VISA-Resource in
statesAntConrol-StatusbitsReadout
motor azi
Error In
Create a command --> evaluate response --> set status LED --> repeat for every status
Create command
"motor azi", Standard
R101R
Read buffer
ANT CMD
Convert "0" to false, and "1" to true
Reply
Choose status LED and set status
"end sw ele horiz"
End switch Ele horizontal
Next status bit
initiator az w
Antenna status cluster out
VISA-Resource out
Error Out

Figure 4-30 Block diagram of the VI AntStatus (ANT STATUS).

The VI for reading the antenna status indicators is a state machine that sends a command for each status indicator and evaluates the response from the antenna controller by converting a string ‘0’ to a logical false and a string ‘1’ to a logical true. Then the according status LED in the status cluster is set (see Figure 4-22b).

It has to be noted that every status indicator has to be read sequentially, and therefore this operation takes relatively long (a few seconds).

Figure 4-31 shows the block diagram of the Sub-VI **ANT READ POS**, where the readout of azimuth and elevation of the antenna controller is handled.

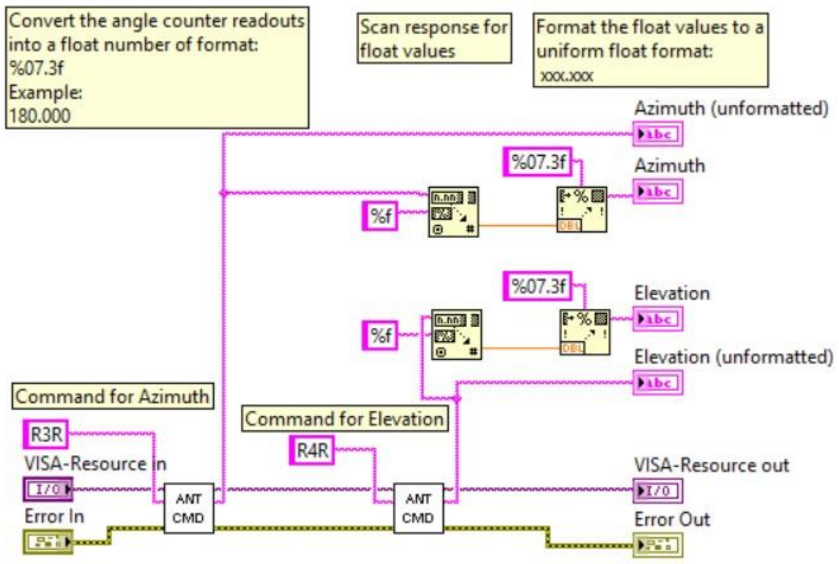
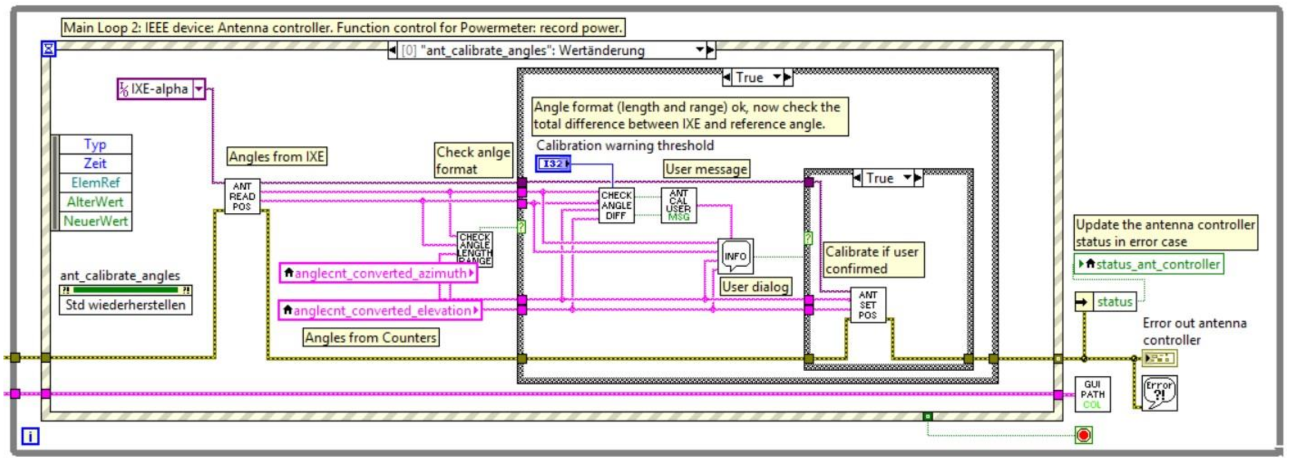


Figure 4-31 Block diagram of the VI AntReadPosition (ANT READ POS).

The command for reading the azimuth position is **R3R** and the elevation command is **R4R**. The response from the antenna controller is a string and built-in LabVIEW routines are used to filter the angles and format them to a 7 digit floating point number of format **nnn.nnn**.

4.4.4.5 *Calibrating azimuth and elevation*

The angle readings from the antenna controller are relative readings from a calibration point and require a new calibration from time to time. For this purpose the Heidenhain angle counter readings can be used as reference and can be applied to the antenna controller. Figure 4-32 shows the event case **ant_calibrate_angles** that is triggered by pressing the **Calibrate angle** button (see Figure 4-22a).



Main Loop 2: IEEE device: Antenna controller. Function control for Powermeter: record power.
[0] "ant_calibrate_angles": Wertänderung
IXE-alpha
Typ
Zeit
ElemRef
AlterWert
NeuerWert
ant_calibrate_angles
Std wiederherstellen
Angles from IXE
ANT READ POS
Check anlge format
CHECK ANGLE LENGTH RANGE
anglecnt_converted_azimuth
anglecnt_converted_elevation
Angles from Counters
True
Angle format (length and range) ok, now check the total difference between IXE and reference angle.
Calibration warning threshold
I32
User message
CHECK ANGLE DIFF
ANT CAL USER MSG
INFO
User dialog
True
Calibrate if user confirmed
ANT SET POS
Update the antenna controller status in error case
status_ant_controller
status
Error out antenna controller
GUI PATH COL
Error ?!

Figure 4-32 Event case for calibrating azimuth and elevation of the antenna controller.

Before calibrating, the angles from the controller are refreshed with the use of the Sub-VIs **ANT READ POS**. The angles from the angle counters are updated continuously and don't require dedicated refreshing. After all angles are up to date, the Sub-VI **CHECK ANGLE LENGTH RANGE** checks, if the format of the angles is correct. After that the Sub-VI **CHECK ANGLE DIFF** performs another check to see, if the deviation between the angle readings from the controller and the angle counter differ by more than a certain threshold. If this threshold is exceeded, the user is warned but may continue with the calibration process. This additional check is for security reasons, so a calibration with wrong values is made unlikely. The threshold at which this warning is shown can be set with the control **Calibration warning threshold** on the tab **Settings** in the GUI.

After all checks have been performed, a user dialog is opened that sums up the old angles and the calibrated angles and shows any warning that might occur during the calibration process. This dialog allows the user to either confirm the calibration parameters or abort the calibration process.

4.4.5 Coaxial switches

The physical high frequency path contains one 2-way coaxial switch and two 4-way coaxial switches. The 2-way switch **COAX2** is used for changing the frequency path for the test translator between 11 GHz and 12 GHz. The 4-wax coaxial switches **COAX1** and **COAX3** are used for routing the received signal to one of the four signal paths A, B, C or D.

Figure 4-33 shows the LEOSv2 GUI where a block diagram of the physical high frequency path is depicted. The orange highlighted area in the left, labelled **COAX2, TT frequency** shows the display for the test translator frequency, which is controlled by COAX2. In the middle, the path switches **COAX1 TT** and **COAX3 2nd DC** are marked and on the right there are the controls for changing the coaxial switch positions and the test translator frequency.

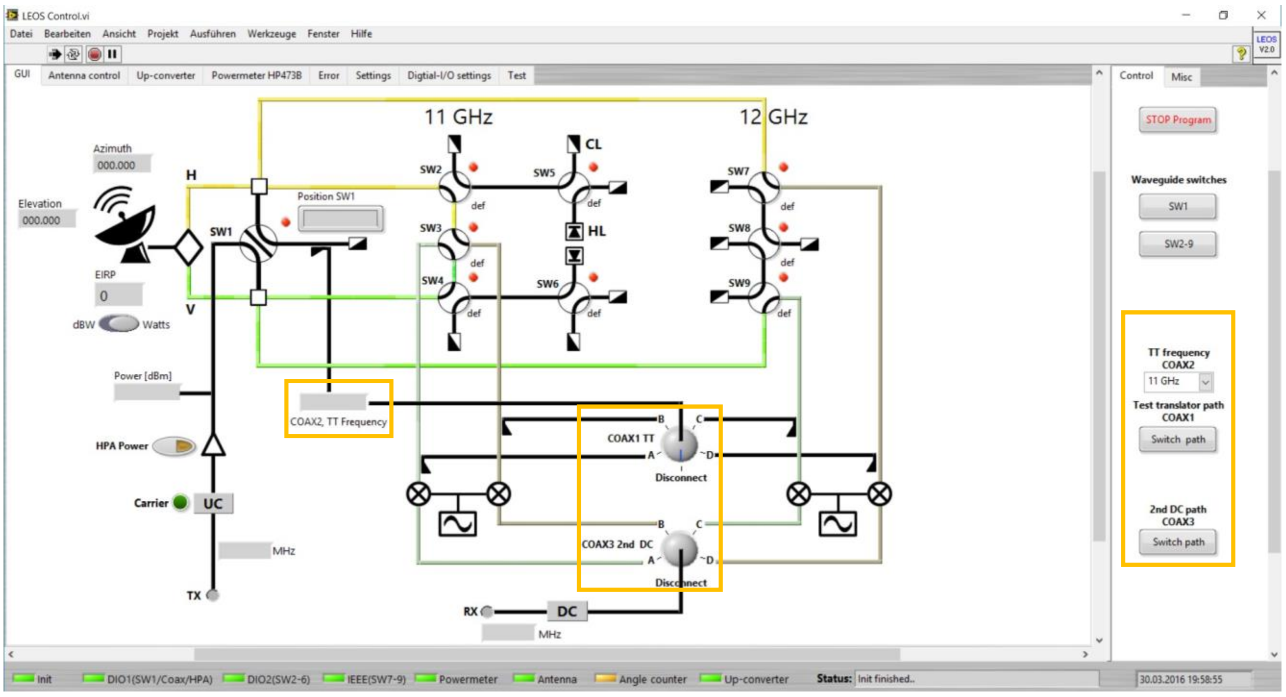
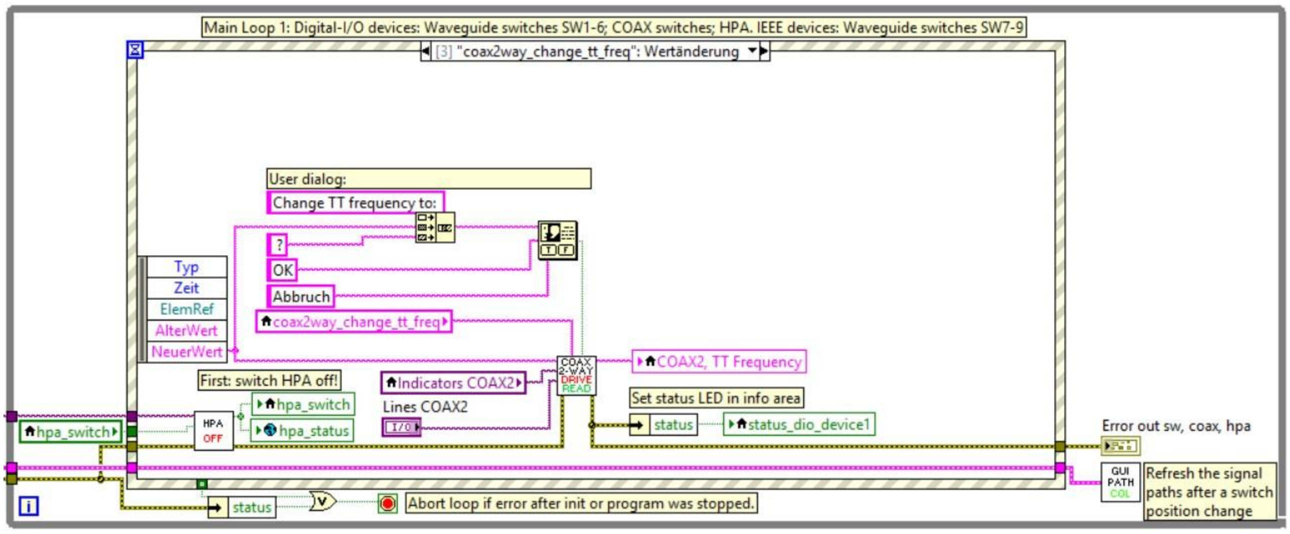


Figure 4-33 LEOSv2 GUI. The COAX switches and controls are highlighted orange.

The button **Switch TT** controls the switch COAX1 and allows routing of the test translator signal to one of the four signal paths A, B, C or D. The dropdown field **TT frequency change** controls the switch COAX2 and the button **2nd DC Path** controls the switch COAX3, allowing the received signal to be routed to path A, B, C or D. Pressing any of the above controls, causes the HPA to switch off for safety reasons. The HPA then has to be turned back on manually. Switching the coaxial switches is done via the Digital-I/O converters. Changing the position requires a constant signal at the corresponding drive ports.

4.4.5.1 2-wax coaxial switch

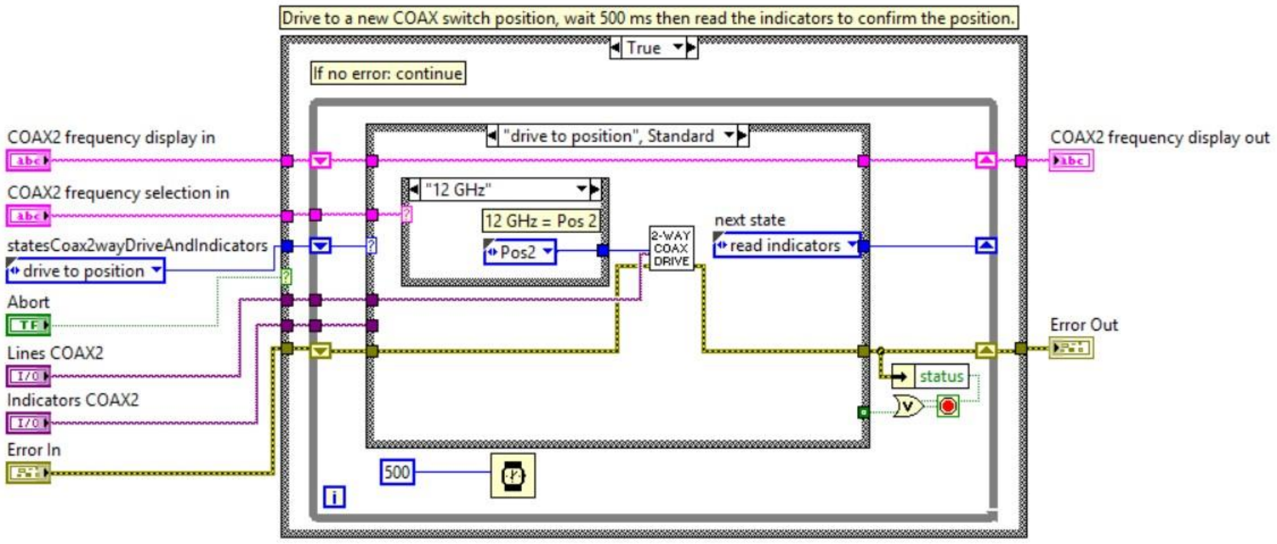
Selecting a new frequency at the control **TT frequency change** in the GUI (see Figure 4-33) triggers the event case **coax2way_change_tt_freq**. This case is shown in Figure 4-34.



Main Loop 1: Digital-I/O devices: Waveguide switches SW1-6; COAX switches; HPA. IEEE devices: Waveguide switches SW7-9
[3] "coax2way_change_tt_freq": Wertänderung
User dialog:
Change TT frequency to:
?
OK
Abbruch
coax2way_change_tt_freq
Typ
Zeit
ElemRef
AlterWert
NeuerWert
First: switch HPA off!
Indicators COAX2
hpa_switch
hpa_status
Lines COAX2
HPA OFF
COAX 2-WAY DRIVE READ
COAX2, TT Frequency
Set status LED in info area
status
status_dio_device1
Error out sw, coax, hpa
GUI PATH COL
Refresh the signal paths after a switch position change
status
Abort loop if error after init or program was stopped.

Figure 4-34 Event case for switching the test translator frequency.

As mentioned before, using any of the available coaxial switch controls in the GUI automatically switches off the HPA. This is done by the Sub-VI **HPA OFF** at the lower left. After that, a user dialog informs the user about the selected frequency. If confirmed, the Sub-VI **COAX 2-WAY DRIVE READ** toggles the 2-way coaxial switch COAX2. Figure 4-35 shows the block diagram of this VI.



Drive to a new COAX switch position, wait 500 ms then read the indicators to confirm the position.
True
If no error: continue
"drive to position", Standard
"12 GHz"
12 GHz = Pos 2
Pos2
2-WAY COAX DRIVE
next state
read indicators
COAX2 frequency display in
COAX2 frequency selection in
statesCoax2wayDriveAndIndicators
drive to position
Abort
Lines COAX2
Indicators COAX2
Error In
COAX2 frequency display out
Error Out
status
500

Figure 4-35 Block diagram of the VI Coax2wayDriveAndRead (COAX 2-WAY DRIVE READ).

The outer case structure is used to skip the process if the user aborted the frequency change. However, if the user did confirm the change, the while loop and the case structure form a state machine. First the selected frequency is passed on to the Sub-VI **2-WAY COAX DRIVE**, then the position is confirmed, by reading the indicator ports of the 2-way coaxial switch, using the Sub-VI **COAX 2-WAY POS** (not depicted in the picture).

Coaxial switches are controlled by the National Instruments Digital-I/O converters. LabVIEW provides routines for opening communication on Digital-I/O ports in their DAQmx VIs. Figure 4-36 shows the block diagram for the VI **COAX 2-WAY DRIVE** where a DAQmx resource is opened and configured for digital output.

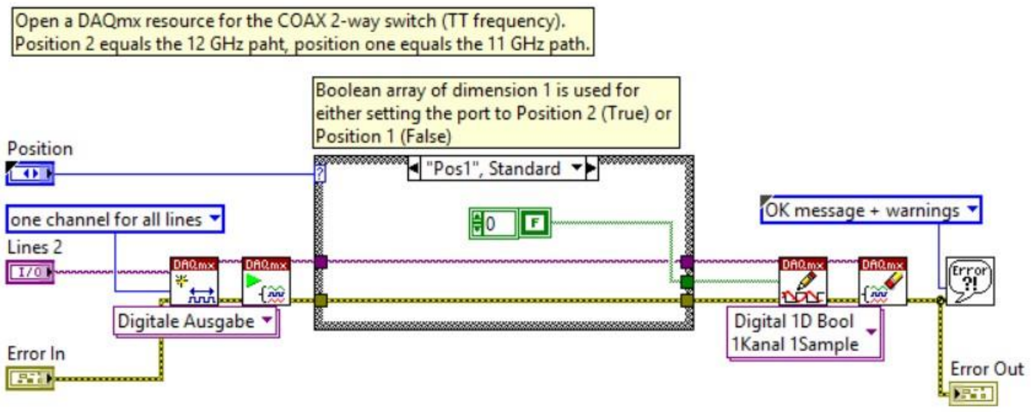

Open a DAQmx resource for the COAX 2-way switch (TT frequency).
Position 2 equals the 12 GHz paht, position one equals the 11 GHz path.
Boolean array of dimension 1 is used for either setting the port to Position 2 (True) or Position 1 (False)
Position
"Pos1", Standard
one channel for all lines
Lines 2
OK message + warnings
Digitale Ausgabe
Digital 1D Bool 1Kanal 1Sample
Error In
Error Out

Figure 4-36 Block diagram of the VI Coax2wayDrive (COAX 2-WAY DRIVE).

Digital-I/O ports can be controlled by sending Boolean arrays to the converters. The length of the Boolean array has to be equal to the number of ports that should be controlled. A Boolean **True** sets the switches the port on and Boolean **False** switches the port off. After sending data to the Digital-I/O converter, the DAQmx resource is closed again.

Reading data from the Digital-I/O converters is similar to writing data to them, as shown in Figure 4-37.

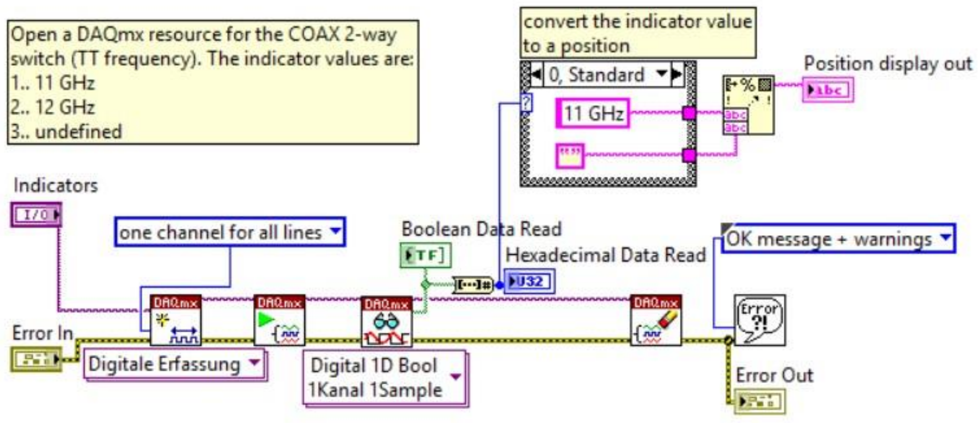

Open a DAQmx resource for the COAX 2-way switch (TT frequency). The indicator values are:
1.. 11 GHz
2.. 12 GHz
3.. undefined
convert the indicator value to a position
0, Standard
11 GHz
Position display out
Indicators
one channel for all lines
Boolean Data Read
Hexadecimal Data Read
OK message + warnings
Error In
Digitale Erfassung
Digital 1D Bool 1Kanal 1Sample
Error Out

Figure 4-37 Block diagram of the VI Coax2wayIndicator (COAX 2-WAY POS).

Again, a DAQmx resource is opened and configured for digital data acquisition. Then data is read and the DAQmx resource is closed again. The readings from the Digital-I/O ports are evaluated in the case structure, converting the Boolean response from the converters into a string that represents the position of COAX2 and therefore the selected frequency for the test translator.

4.4.5.2 4-way coaxial switch

The two 4-way coaxial switches COAX1 and COAX3 are used to route a high frequency signal to one of the four signal paths A, B, C or D. Additionally, these coaxial switches can also be in position **Disconnect**, meaning that none of the four paths is selected.

The GUI control **Switch TT** (see Figure 4-33) triggers the event case **coax3_change_position**. This case is shown in Figure 4-38.

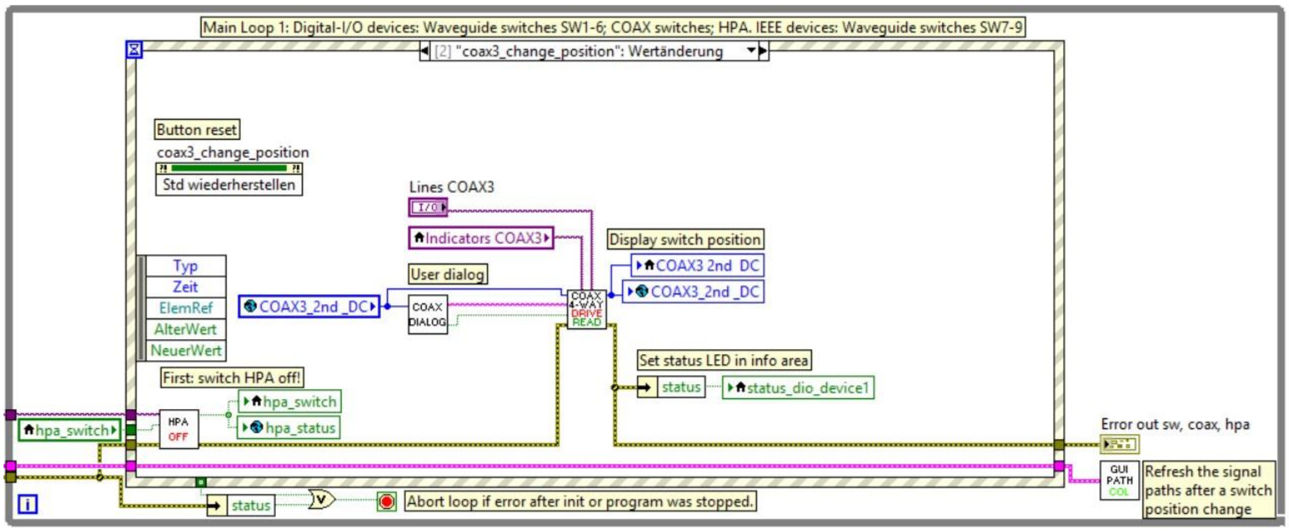


Figure 4-38 Event case for switching the 4-way coaxial switch COAX3.

After switching of the HPA, a user dialog is opened that depicts a 4-way coaxial switch and allows to choose a position. If the user selects an option, the Sub-VI **COAX 4-WAY DRIVE READ** (Figure 4-39) handles the procedure of changing the coax position. First a new position is set, afterwards the position is confirmed by reading the indicator ports of the coaxial switch.

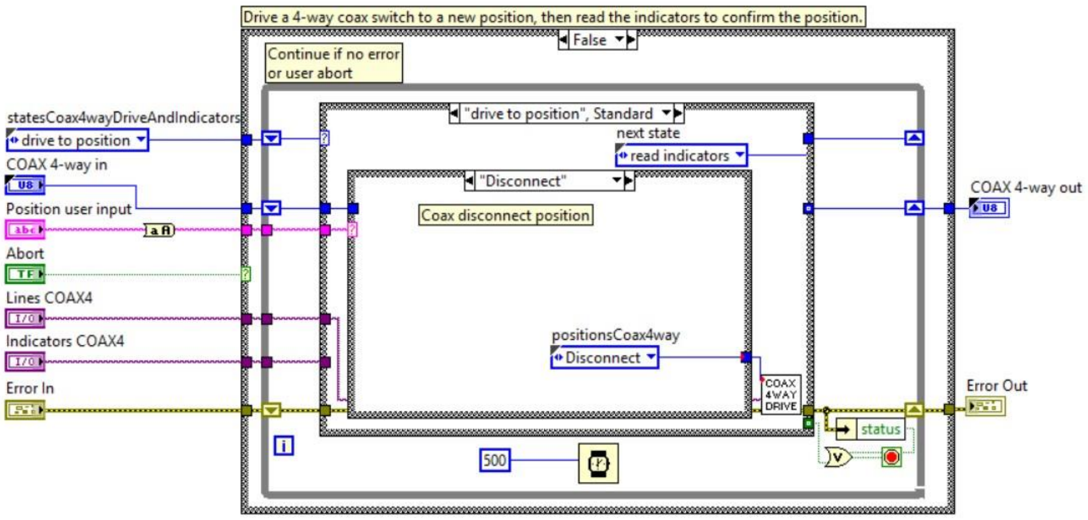
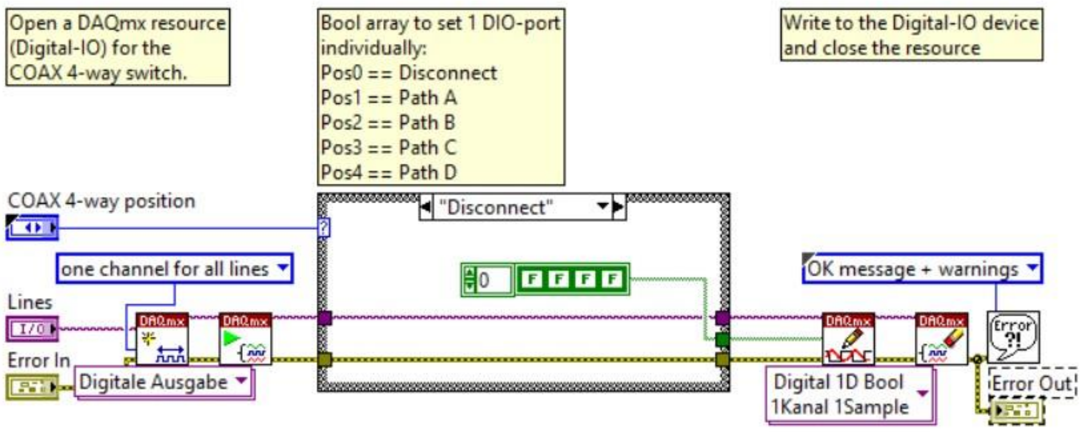


Figure 4-39 Block diagram for the VI Coax4wayDriveAndRead (COAX 4-WAY DRIVE READ).

The Sub-VI **COAX 4-WAY DRIVE** (Figure 4-40) handles communication with the Digital-I/O converters similar to the way the 2-way coaxial switch is handled. The difference is that instead of using a Boolean array of length 1, the four drive ports of the switches COAX1 and COAX3 are controlled at once. Each Boolean value of the array corresponds to one drive port for the switches. Setting all of the Boolean values to false will cause the switch to disconnect. Setting either of the Boolean values to true will cause the switch to change its position.



Open a DAQmx resource (Digital-IO) for the COAX 4-way switch.

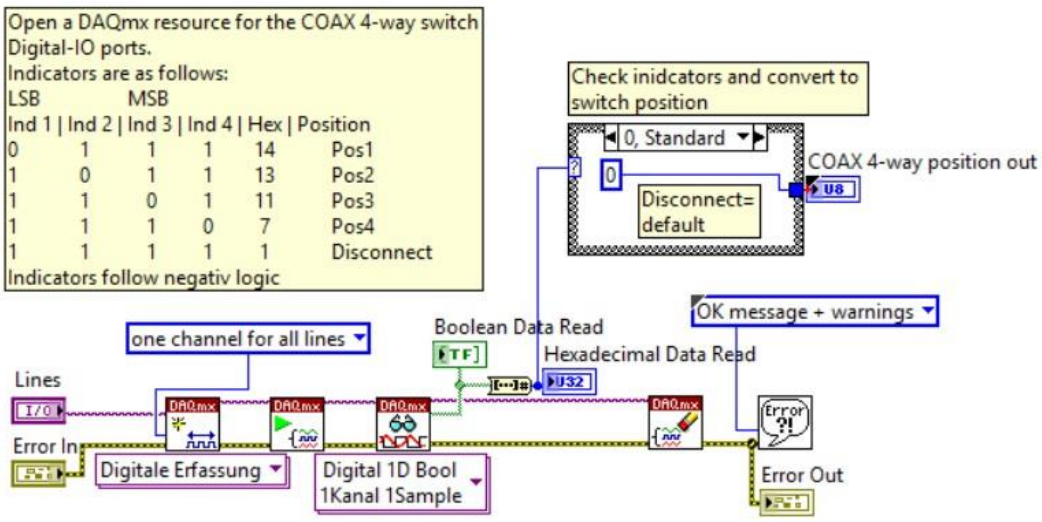
Bool array to set 1 DIO-port individually:
Pos0 == Disconnect
Pos1 == Path A
Pos2 == Path B
Pos3 == Path C
Pos4 == Path D

Write to the Digital-IO device and close the resource

Figure 4-40 Block diagram for the VI Coax4wayDrive (COAX 4-WAY DRIVE).

Like there are four ports for driving a coax switch to a new position, there are four indicator ports to read the current position. The indicators follow negative logic, meaning that an indicator is active if it has the logical state **low** (False).

The reading of the indicators of a 4-way switch returns a Boolean array of length 4. This array is converted to a hexadecimal number, as shown in Figure 4-41.



Open a DAQmx resource for the COAX 4-way switch Digital-IO ports.
Indicators are as follows:
LSB MSB

Ind 1	Ind 2	Ind 3	Ind 4	Hex	Position
0	1	1	1	14	Pos1
1	0	1	1	13	Pos2
1	1	0	1	11	Pos3
1	1	1	0	7	Pos4
1	1	1	1	1	Disconnect

Indicators follow negativ logic

Check inidcators and convert to switch position

Figure 4-41 Block diagram for the VI Coax4wayIndicators (COAX 4-WAY POS).

The case structure in the figure evaluates the hexadecimal values and returns the corresponding positon of the switch.

4.4.6 High power amplifier

The high power amplifier (HPA) is controlled by a National Instruments Digital-I/O converter. It can either be switched on or off. Figure 4-42 shows the block diagram of the physical high frequency path and the location of the HPA control **HPA Power** is highlighted orange.

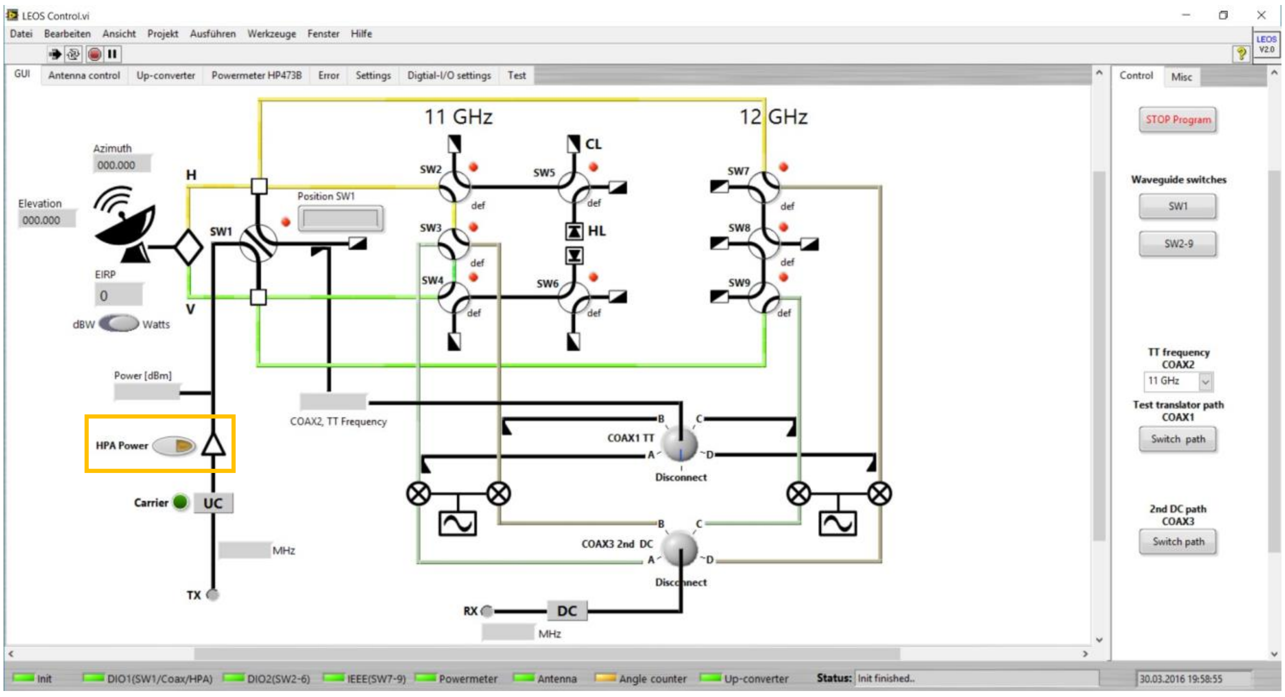
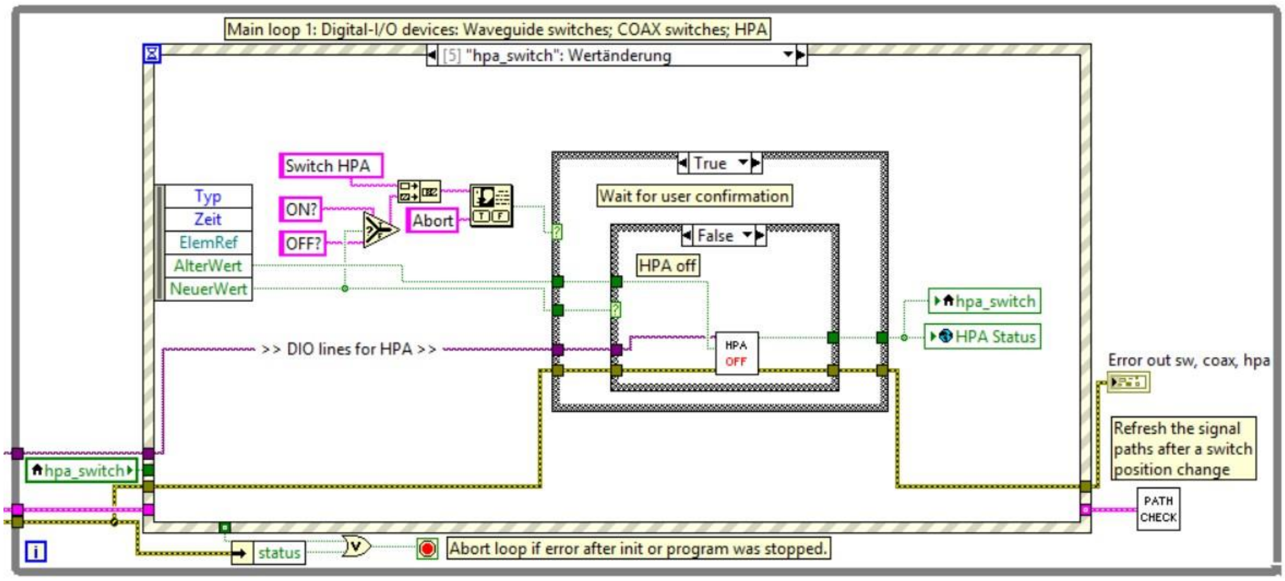


Figure 4-42 LEOSv2 GUI. The HPA LED and control button are highlighted orange.

To switch the HPA simply click on the control **HPA Power**. This control includes a LED that indicates if the HPA is currently on or off. If the HPA control is clicked the event **hpa_switch** in the **Main Loop 1** is triggered. Figure 4-43 shows Main Loop 1 in the LEOSv2 main block diagram.



Main loop 1: Digital-I/O devices: Waveguide switches; COAX switches; HPA
[5] "hpa_switch": Wertänderung
Typ
Zeit
ElemRef
AlterWert
NeuerWert
Switch HPA
ON?
OFF?
Abort
True
Wait for user confirmation
False
HPA off
HPA OFF
>> DIO lines for HPA >>
hpa_switch
HPA Status
Error out sw, coax, hpa
Refresh the signal paths after a switch position change
PATH CHECK
status
Abort loop if error after init or program was stopped.

Figure 4-43 Event case for switching the HPA on or off.

First, a dialog lets the user confirm the switching of the HPA. If the user confirms, the Sub-VIs **HPA OFF** and **HPA ON** handle the switching. As an example, the Sub-VI **HPA OFF** is shown in Figure 4-44, for the case that no error occurred during Digital-I/O communication.

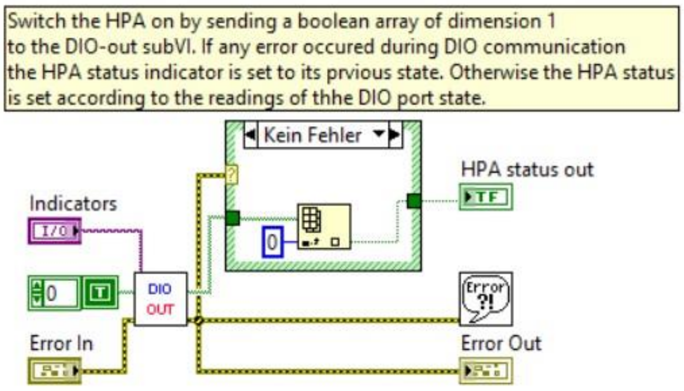

Switch the HPA on by sending a boolean array of dimension 1
to the DIO-out subVI. If any error occured during DIO communication
the HPA status indicator is set to its prvious state. Otherwise the HPA status
is set according to the readings of thhe DIO port state.

Figure 4-44 Blockdiagram for the VI HpaOn (HPA ON), case: no error.

Here the **HPA status out** indicator is set, according to the state of the corresponding Digital-I/O port. Figure 4-45 shows what happens, if an error occurred whilst setting the Digital-I/O port. It is assumed, that the port wasn't changed and therefore keeps its current state, meaning that the **HPA status out** also stays unchanged by keeping its current state **HPA status in**.

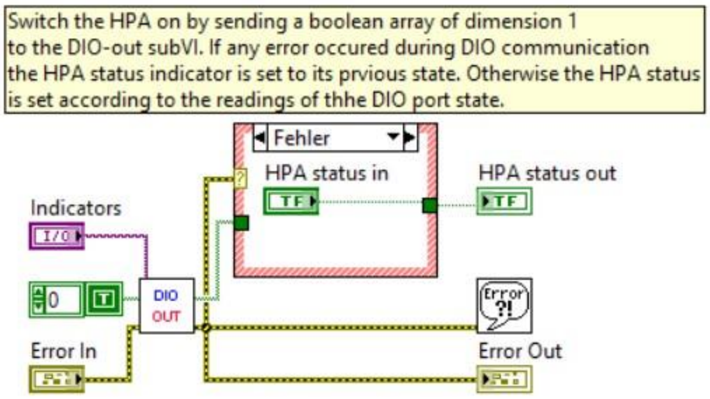

Switch the HPA on by sending a boolean array of dimension 1
to the DIO-out subVI. If any error occured during DIO communication
the HPA status indicator is set to its prvious state. Otherwise the HPA status
is set according to the readings of thhe DIO port state.

Figure 4-45 Blockdiagram for the VI HpaOff (HPA OFF), case: error.

With the help of the Sub-VI **DIO OUT** (Figure 4-46) a Boolean array with a length of 1 is sent to the Digital-I/O port to which the HPA is connected. If the Boolean array is set to **True**, the HPA is switched on, if set to **False**, the HPA is switched off.

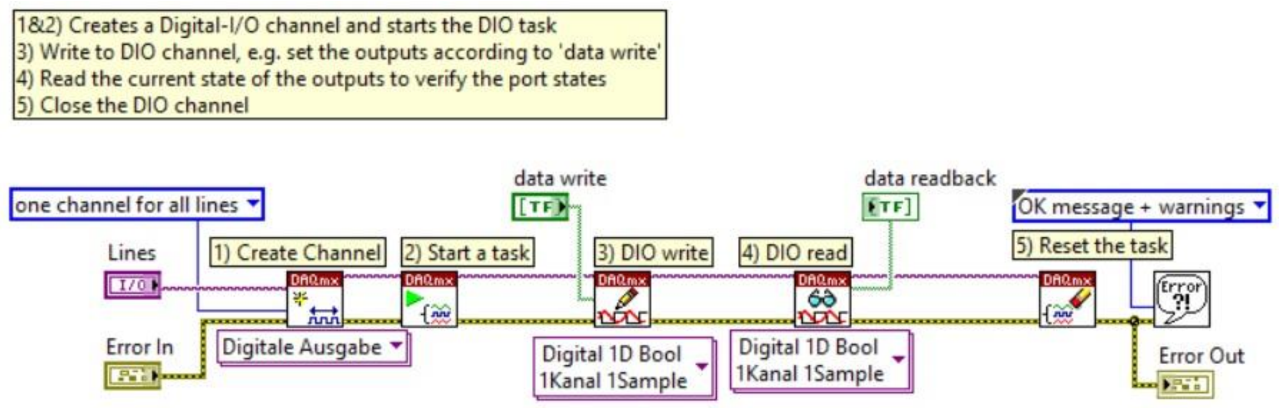

1&2) Creates a Digital-I/O channel and starts the DIO task
3) Write to DIO channel, e.g. set the outputs according to 'data write'
4) Read the current state of the outputs to verify the port states
5) Close the DIO channel
one channel for all lines
data write
data readback
OK message + warnings
Lines
1) Create Channel
2) Start a task
3) DIO write
4) DIO read
5) Reset the task
Error In
Digitale Ausgabe
Digital 1D Bool 1Kanal 1Sample
Digital 1D Bool 1Kanal 1Sample
Error Out

Figure 4-46 Block diagram for the VI DioOutput (DIO OUT).

The numbered elements in Figure 4-46 serve the following purpose:

1) Creating a DIO channel and configure the channel by setting the corresponding DIO lines and the operating mode one **channel for all lines**. This setting allows several DIO lines to be operated by a single channel. A DIO line represents a physical DIO port.
2) Starting the DIO task switches the DIO channel to operating mode and allows for input or output operations.
3) Writing data to the DIO channel in order to set the DIO output port(s) to **high** or **low**.
4) Reading back from the DIO ports to check the current state of the port(s).
5) Resetting and closing the DIO channel and release the used resources.

Errors during the DIO operation are then displayed as a user dialog using LabVIEWs “Einfacher Fehlerbehandler”.

4.4.7 Power meter

The power meter, once initialized, requires no user interaction. The current readings are shown in the marked area in Figure 4-47.

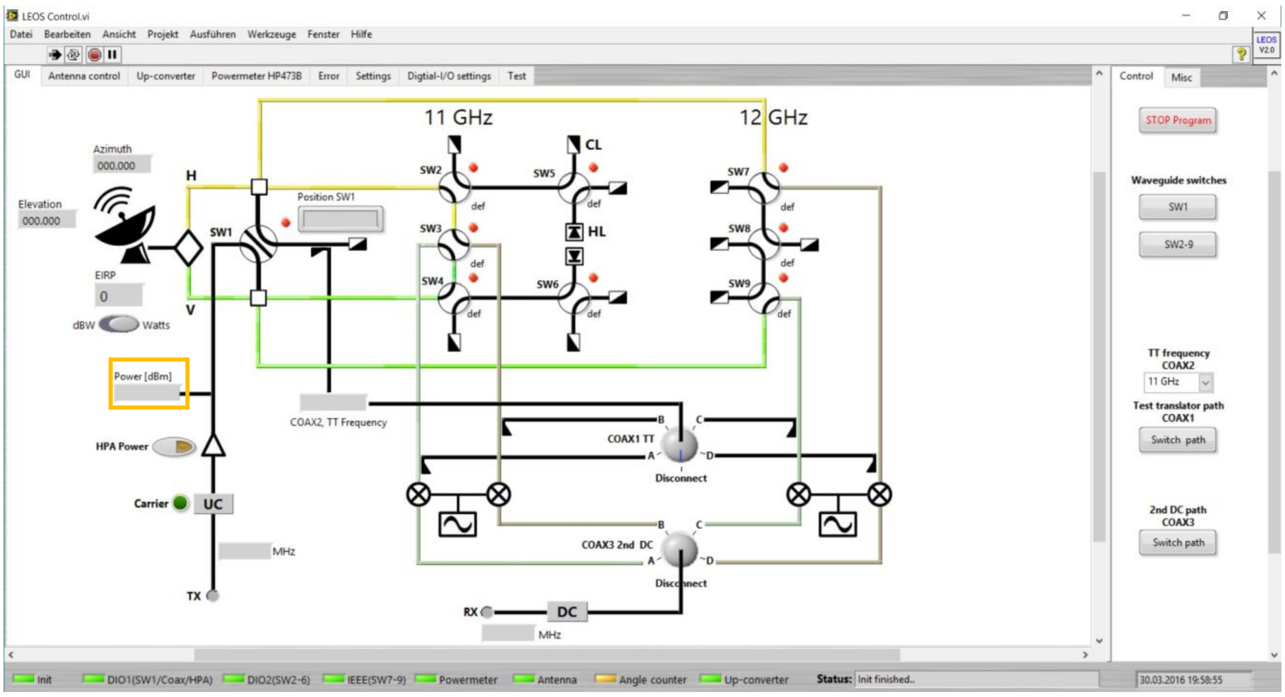


Figure 4-47 LEOSv2 GUI. The power meter display is highlighted orange

For refreshing the power meter readings, a timed loop called **Timed Loop 3** is used, that is directly placed in the main LEOSv2 block diagram. In order to adjust the refresh period, the control **Refresh periode for power meter** on the GUI's settings tab can be used.

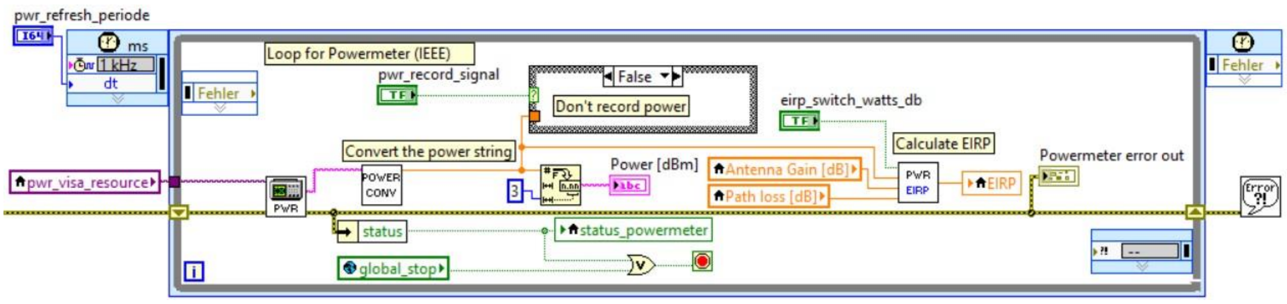


Figure 4-48 Timed loop in the block diagram of the main GUI. Here the power meter readings are refreshed periodically.

Figure 4-48 shows the timed loop for power meter readout. The power level provided by the power meter is a string of the format **+x.xxxxE+yy** where x.xxxx is the power level and yy is the exponent for the power level. The Sub-VI **POWER CONV** converts the power readings to a 3 digit decimal number, as shown in Figure 4-49.

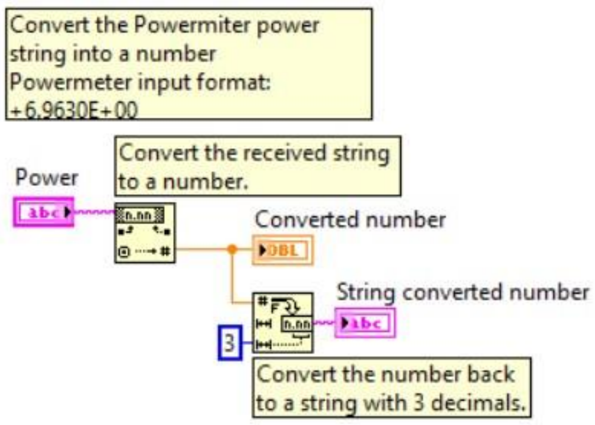
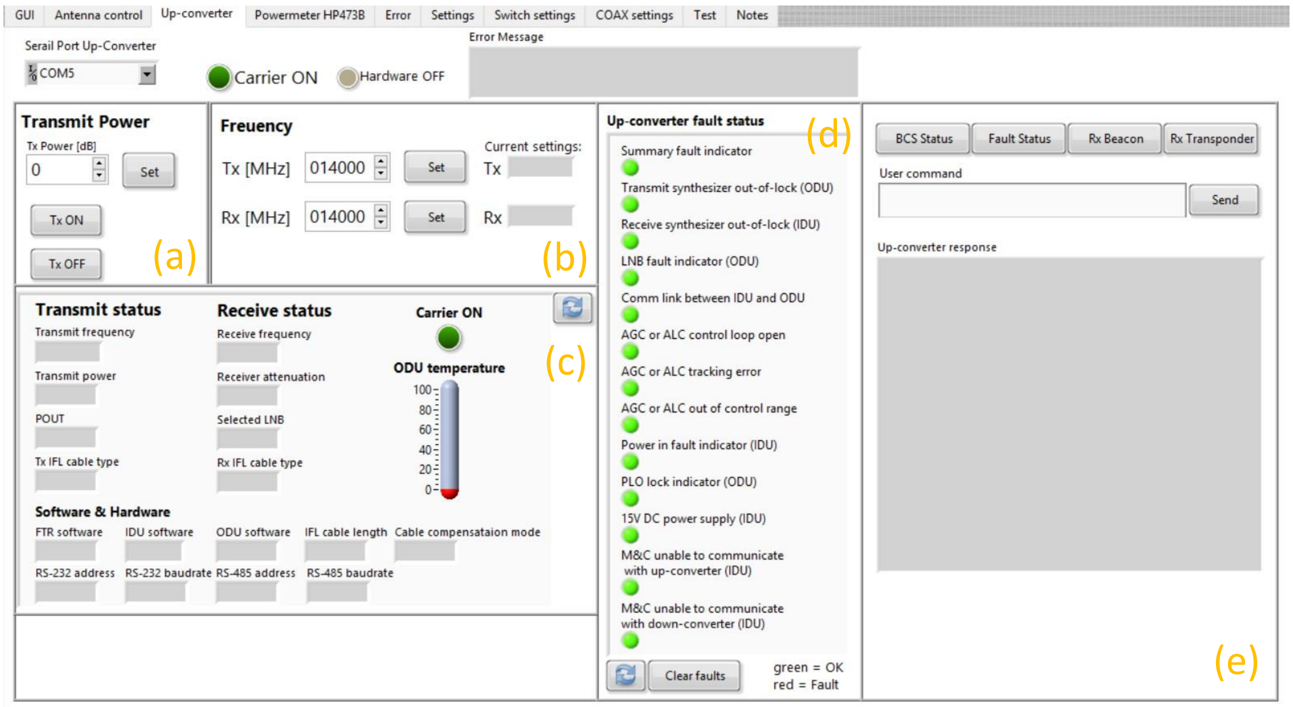

Convert the Powermiter power string into a number Powermeter input format: +6.9630E+00
Power
Convert the received string to a number.
Converted number
String converted number
3
Convert the number back to a string with 3 decimals.

Figure 4-49 Block diagram for the VI PowermeterConvertReading (POWER CONV).

First, the string representation of the current power reading is converted to a double number, then this number is converted back to a decimal string with 3 digits.

4.4.8 Up-converter

The up-converter functions are now included in the LEOSv2 software. The **Up-converter** tab of the main GUI is shown in Figure 4-50 and it contains a full implementation of all status and fault indicators [20] and the most important functions of the up-converter.



Serail Port Up-Converter: COM5

Tx Power [dB]: 0

Tx [MHz]: 014000

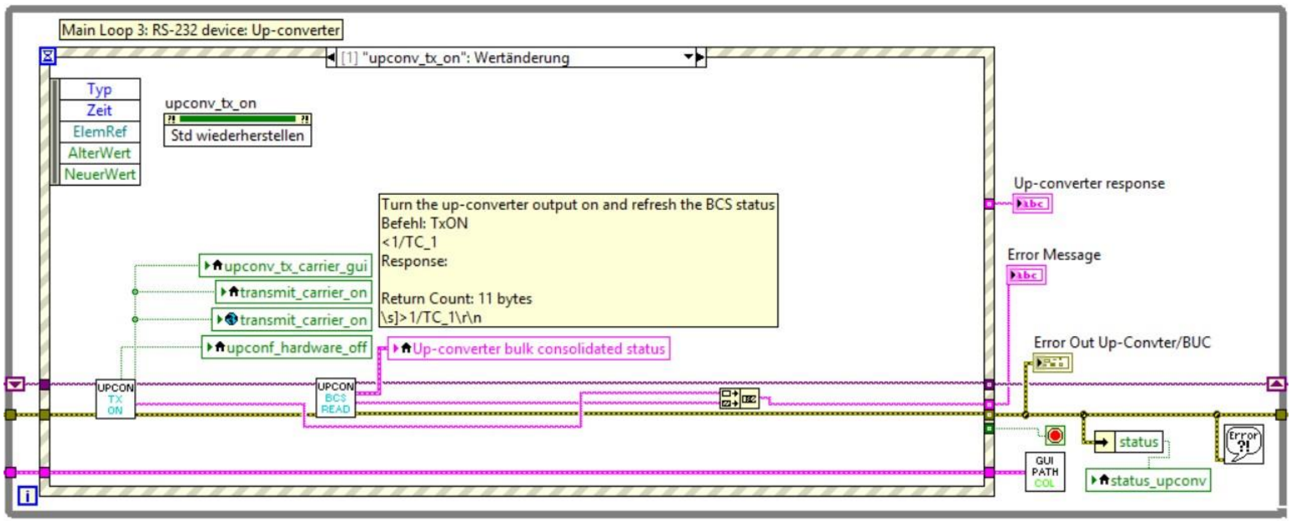
Rx [MHz]: 014000

Figure 4-50 LEOSv2 GUI: front panel of the up-converter tab. (a) Controls for setting the transmit power and switching the transmitter on and off. (b) Controls for changing transmit and receive frequency. The current frequencies are also shown. (c) Status displays for the Bulk Consolidated Status (BCS). (d) Fault status (FS) indicators. ((e) Area for using selected predefined commands or direct serial communication with the up-converter (user commands).

The buttons in Figure 4-50a are used for controlling the transmit power and for switching the transmitter on. The transmit power can be **Set** within 0-20 dB in 0.1dB steps [20]. The buttons **Tx ON** and **Tx OFF** switch the transmitter on or off. The current transmitter state is indicated by the LED **Carrier ON** at the top of the panel. In Figure 4-50b the controls and indicators for transmit and receive frequencies are collected. Before a new frequency is sent to the up-converter, a check is performed, validating the frequency setting and thus informing the user about the validity. Figure 4-50c shows the Bulk Consolidated Status (**BCS**) of the up-converter. The BCS is a collection of all available information about the up-converter's configuration and setup and is realized as a cluster in the software. The BCS is refreshed each time the user changes the transmit power, frequency or state. It may also be refreshed manually by using the refresh symbol in the panel. Figure 4-50d contains the fault status (FS) LEDs of the up-converter, which are also collected within a cluster. If a fault is detected, the corresponding LED will turn red. A description of the faults can be found in in the up-converter handbook [20]. Figure 4-50e contains some predefined functions that have been already implemented in the old, console based up-converter software. These features are kept for convenience. **Rx Beacon** sets the receive frequency directly to 12501 MHz and **Rx Transponder** sets it to 12646 MHz.

The up-converter features an extensive list of configuration options, some of which are hardly used and therefore not implemented directly into the GUI. However, there is the option of directly sending text-based commands to the up-converter and thus also use non-implemented settings. Commands can be entered in the field **User command** and then sent to the up-converter with the **Send** button. The field **Up-converter response** displays the corresponding response to the sent command.

Any user action on the **Up-converter** tab will trigger an event in the event structure of the **Main Loop 3** in the block diagram of the GUI. As an example, the case **upconv_tx_on** is shown in Figure 4-51.



Turn the up-converter output on and refresh the BCS status
Befehl: TxON
<1/TC_1
Response:

Return Count: 11 bytes
\s]>1/TC_1\r\n

Figure 4-51 Event case for switching the up-converter transmitter on.

After each change in the up-converter configuration, the BCS cluster is refreshed. Querying the BCS information from the up-converter triggers a response that contains a string listing of all status flags. The individual status flags need to be filtered from the list. This is done in the Sub-VI **UPCON BCS READ**, described in Chapter 4.4.8.6.

4.4.8.1 The up-converter command format

Before describing the software functions for the up-converter in detail, an explanation of the command format is appropriate. Every command sent to the up-converter triggers a response containing the sent command plus the current setting for this specific command or a list of different several settings.

The format of a command that is sent to the up-converter is:

<addr/CMD_[value]<cr>

The response from the up-converter is:

>addr/CMD[_value]<cr>

The sign '<' always means, that the command is sent to the up-converter, '>' means this is a response from the up-converter. The tag **addr** is the current IEEE address of the up-converter and is simply set to '**1**' [22]. The tag **CMD** is the name or abbreviation of a certain command. A listing of available commands can be found in chapter 6.2. The squared brackets around the **value** tag mean that this tag is optional and depends on the command. If a value is added to a command it has to be added without the brackets. Each command is ended with a carriage return **<cr>.** The response of the up-converter has a similar format but starts with a '>' and the underscore may also be optional. If a command triggers a response with more than one **CMD** tags, the format is as follows:

>addr/CMD1_value<cr>

CMD2_value<cr>

.

.

CMDn_value<cr>

The beginning of the response is the same, but here the **underscore** and a **value** are always added. After the first returned CMD, each following CMD is printed without the leading **>addr/** tag. The different CMDs are separated by a carriage return.

An example command and response would be:

Command: <1/TC_1\r\n (switch the transmit carrier on)

Response: >1/TC_1\r\n (transmit carrier is switched on)

4.4.8.2 Switching the transmitter on or off

A command to the up-converter is always followed by a response. In Figure 4-52 the block diagram of the Sub-VI **UPCON TX ON** is shown. In the left, the command is sent, using the Sub-VI **SERIAL W & R**, which also reads the response afterwards.

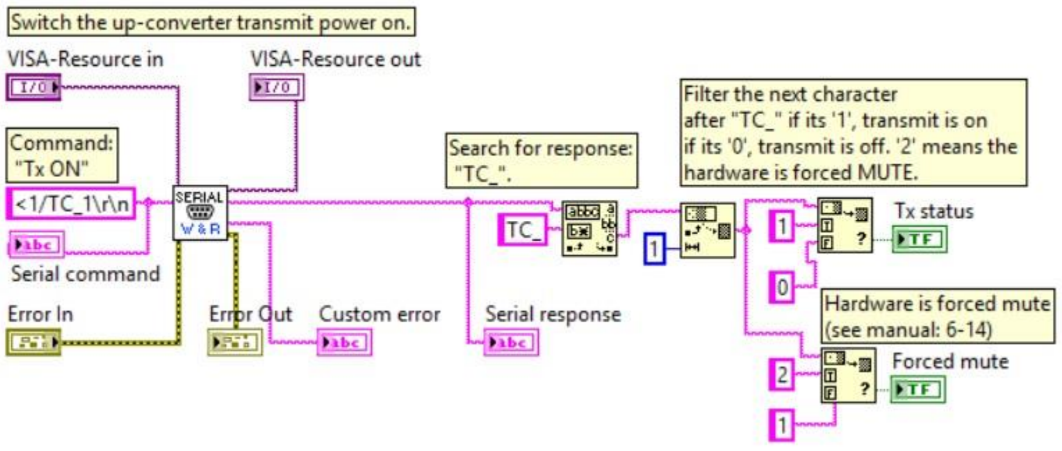

Switch the up-converter transmit power on.
VISA-Resource in
VISA-Resource out
Command: "Tx ON"
<1/TC_1\r\n
Serial command
Error In
SERIAL W & R
Error Out
Custom error
Search for response: "TC_".
TC_
Serial response
Filter the next character after "TC_" if its '1', transmit is on if its '0', transmit is off. '2' means the hardware is forced MUTE.
Tx status
Hardware is forced mute (see manual: 6-14)
Forced mute

Figure 4-52 Block diagram for the VI UpConvTxOn (UPCON TX ON).

The command for switching the transmitter on or off is '**TC_n**', where n is a **0** to switch it off or a **1** to switch the transmitter on. The response for the command 'TC_' is then analysed (see the pink line that comes out of the Sub-VI SERIAL W & R). If the response is '**TC_0**', the transmitter is off. If the response is '**TC_1**', the transmitter is on and if the response is '**TC_2**', then the hardware is forced mute. Forced mute usually means that the BUC is switched off.

4.4.8.3 Transmit power level

All VIs concerning the up-converter are very similar. Figure 4-53 shows the Sub-VI **UPCON TX POWER**. The command for setting the power level is 'TP_nn.n', where nn.n is the power level in dB.

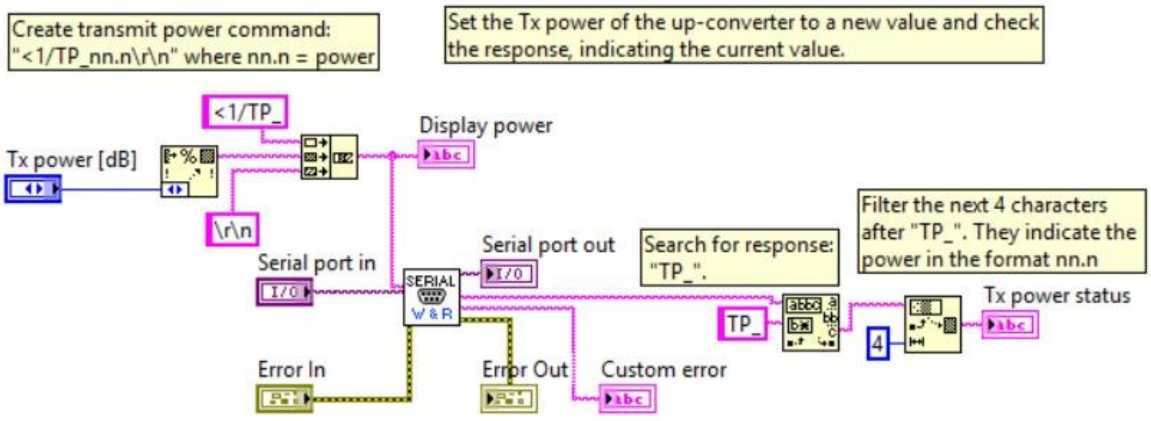

Create transmit power command: "<1/TP_nn.n\r\n" where nn.n = power
Set the Tx power of the up-converter to a new value and check the response, indicating the current value.
<1/TP_
Tx power [dB]
Display power
\r\n
Serial port in
Serial port out
Search for response: "TP_".
Filter the next 4 characters after "TP_". They indicate the power in the format nn.n
Tx power status
TP_
SERIAL W & R
Error In
Error Out
Custom error

Figure 4-53 Block diagram of the VI UpConvTxPower (UPCON TX POWER).

To the left, the numerical control **Tx power [dB]** (see Figure 4-50a) is converted into a string and afterward the full Tx power command **<TP_nn.n\r\n** is formed. The response from the up-converter is again filtered and the current power level is extracted.

4.4.8.4 Transmit frequency

Setting a new transmit frequency in the GUI with the **Set** button (see Figure 4-50b) triggers the event case shown in Figure 4-54. Before the transmit frequency can be set, it is checked whether or not the selected frequency is valid.

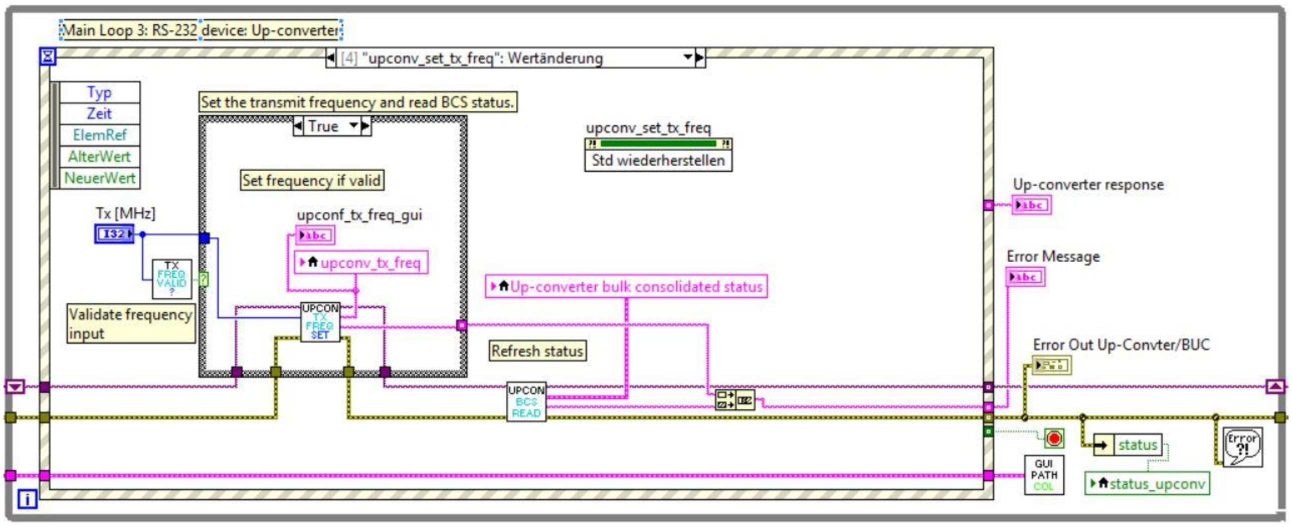


Figure 4-54 Event case for changing the up-converter transmit frequency.

The case structures **True** case sends the new frequency to the up-converter, if the new frequency is validated by the Sub-VI **TX FREQ VALID**, which is shown in Figure 4-55.

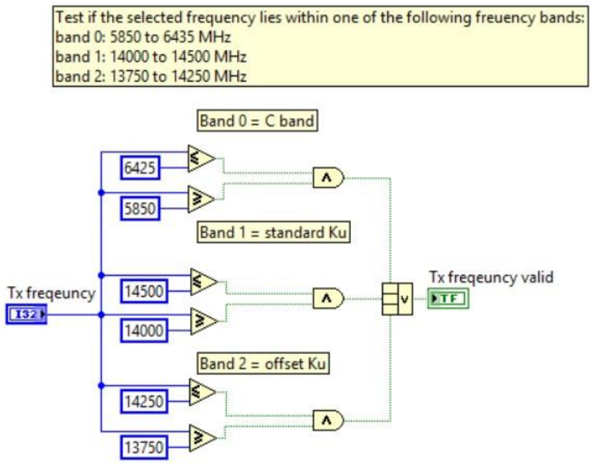


Figure 4-55 Block diagram for the VI UpConvTxFrequencyValidation (TX FREQ VALID).

The available frequency ranges are: 5850 – 6425 MHz (C band), 14000 – 14500 MHz (standard Ku) and 13750 – 14250 MHz (offset Ku).

The command ‘**TXF_nnnnnn**’ is used for setting a frequency and checking the current frequency. The frequency format is a 6 digit number **nnnnnn**. Starting from the left in Figure 4-56, the Sub-VI **TX FREQ SET**, the numerical control **Tx frequency [MHz]** is converted to a string and then used for the TXF command. The up-converters response then is displayed at the indicator **Tx frequency**.

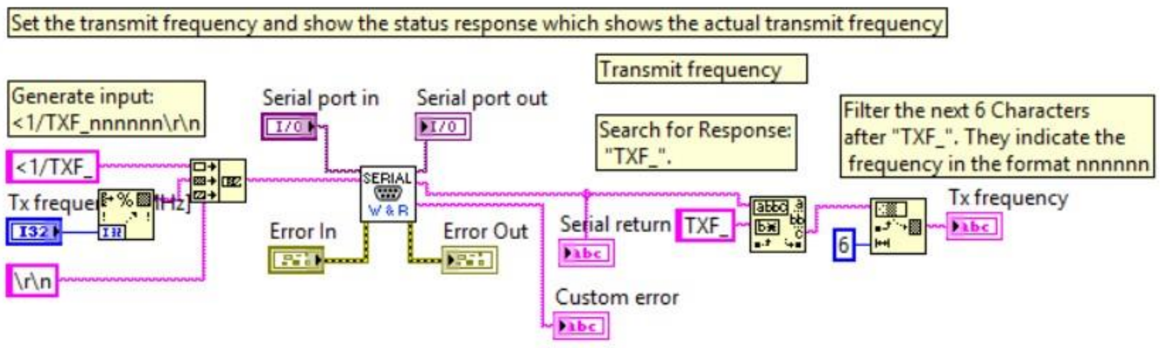

Set the transmit frequency and show the status response which shows the actual transmit frequency
Generate input: <1/TXF_nnnnnn\r\n
<1/TXF_
Tx freque [MHz]
\r\n
Serial port in
Serial port out
SERIAL W & R
Error In
Error Out
Transmit frequency
Search for Response: "TXF_".
Serial return
TXF_
Custom error
Filter the next 6 Characters after "TXF_". They indicate the frequency in the format nnnnnn
6
Tx frequency

Figure 4-56 Block diagram for the VI UpConvTxFreq (TX FREQ SET).

4.4.8.5 Receive frequency

Setting a new transmit receive frequency in the GUI with the **Set** button (see Figure 4-50b) triggers the event case shown in Figure 4-57. Before the receive frequency can be set, two checks are performed. First the new frequency is validated, second the correct LNB for the frequency range needs to be set. Changing the receive LNB is done automatically by the LEOSv2 software and does not require user action.

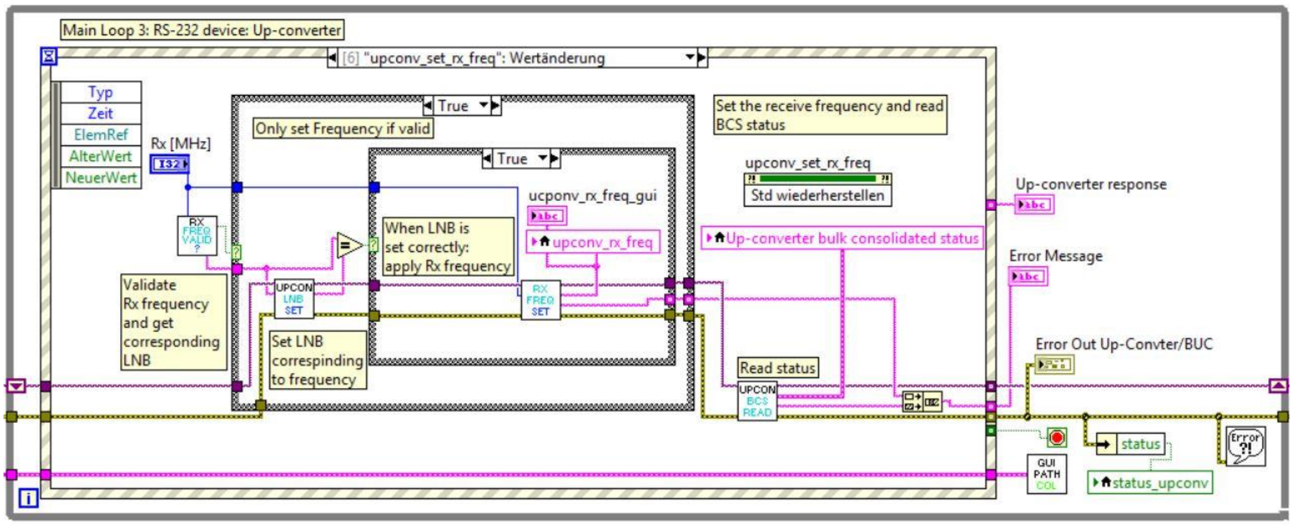

Main Loop 3: RS-232 device: Up-converter
[6] "upconv_set_rx_freq": Wertänderung
Typ
Zeit
ElemRef
AlterWert
NeuerWert
Rx [MHz]
Only set Frequency if valid
True
True
Set the receive frequency and read BCS status
upconv_set_rx_freq
Std wiederherstellen
ucponv_rx_freq_gui
upconv_rx_freq
When LNB is set correctly: apply Rx frequency
Validate Rx frequency and get corresponding LNB
Set LNB correspinding to frequency
Up-converter bulk consolidated status
Read status
Up-converter response
Error Message
Error Out Up-Convter/BUC
status
status_upconv

Figure 4-57 Event case for changing the up-converter receive frequency.

If the frequency is valid, the first case structure is entered, then the LNB is set according to the frequency. Comparing the LNB number (see the '=' function above) from the Sub-VIs **RX FREQ VALID** (Figure 4-58) and **UPCON LNB SET** (Figure 4-59) checks, if the LNB switching was successful. Only then, the new receive frequency is set within the inner case structure, using the Sub-VI **UPCON RX FREQ SET** (Figure 4-60).

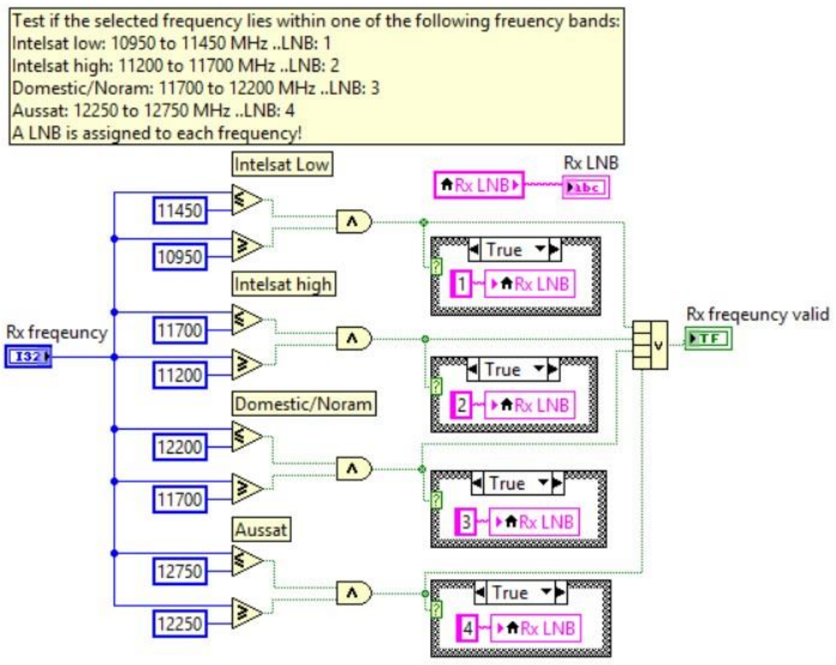

Test if the selected frequency lies within one of the following freuency bands:
Intelsat low: 10950 to 11450 MHz ..LNB: 1
Intelsat high: 11200 to 11700 MHz ..LNB: 2
Domestic/Noram: 11700 to 12200 MHz ..LNB: 3
Aussat: 12250 to 12750 MHz ..LNB: 4
A LNB is assigned to each frequency!
Intelsat Low
Intelsat high
Domestic/Noram
Aussat
Rx freqeuncy
Rx LNB
Rx freqeuncy valid
True

Figure 4-58 Block diagram of the VI UpConvRxFrequencyValidation (RX FREQ VALID).

Figure 4-59 shows the command for setting the LNB is ‘**LNB_n**’, where **n** is the LNB number.

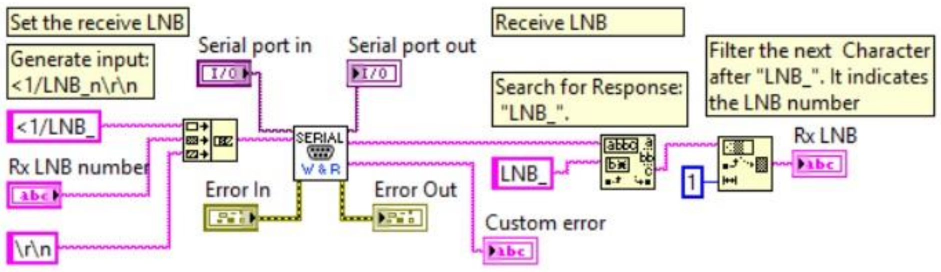

Set the receive LNB
Generate input: <1/LNB_n\r\n
<1/LNB_
Rx LNB number
\r\n
Serial port in
Serial port out
Error In
Error Out
Receive LNB
Search for Response: "LNB_".
LNB_
Custom error
Filter the next Character after "LNB_". It indicates the LNB number
Rx LNB

Figure 4-59 Block diagram of the VI UpConvRxLNB.

Figure 4-60 shows the command for setting the transmit frequency is ‘**RXF_nnnnnn**’, with **nnnnnn** representing the desired frequency.

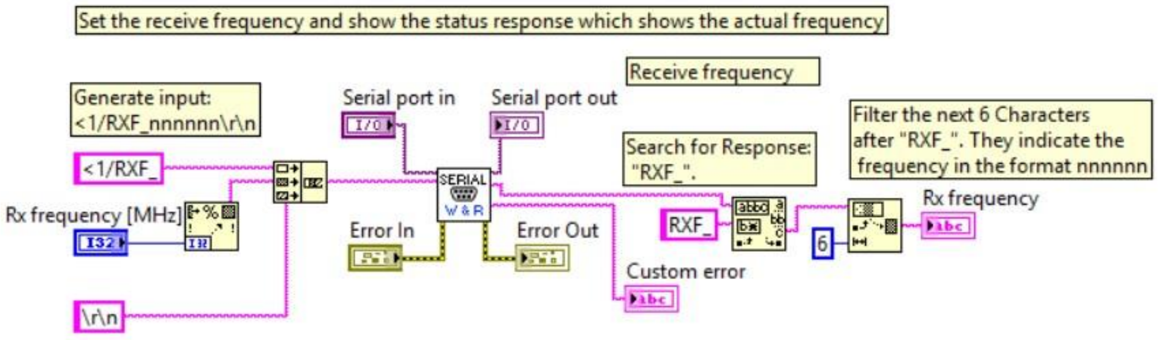

Set the receive frequency and show the status response which shows the actual frequency
Generate input: <1/RXF_nnnnnn\r\n
<1/RXF_
Rx frequency [MHz]
\r\n
Serial port in
Serial port out
Error In
Error Out
Receive frequency
Search for Response: "RXF_".
RXF_
Custom error
Filter the next 6 Characters after "RXF_". They indicate the frequency in the format nnnnnn
Rx frequency

Figure 4-60 Block diagram of the VI UpConvRxFreq (RX FREQ SET).

The numerical control **Rx frequency [MHz]** and the indicator **Rx frequency** can be found in Figure 4-50b.

4.4.8.6 Bulk Consolidated Status

Filtering information from the BCS has been achieved by creating a number of filters, each designed to filter exactly one of the status flags. Moving each filter into a dedicated Sub-VI makes the individual filters reusable and helps to create a better overview of the block diagram of the VI. Figure 4-61 shows the VI **UPCON BCS READ** with the input **Serial response** as a string, containing the BCS information from the up-converter. This string is split and directed into the individual filters that are stacked atop each other and deliver the information to the BCS cluster **Up-converter BCS status out**. This cluster is then displayed in the GUI (see Figure 4-50c).

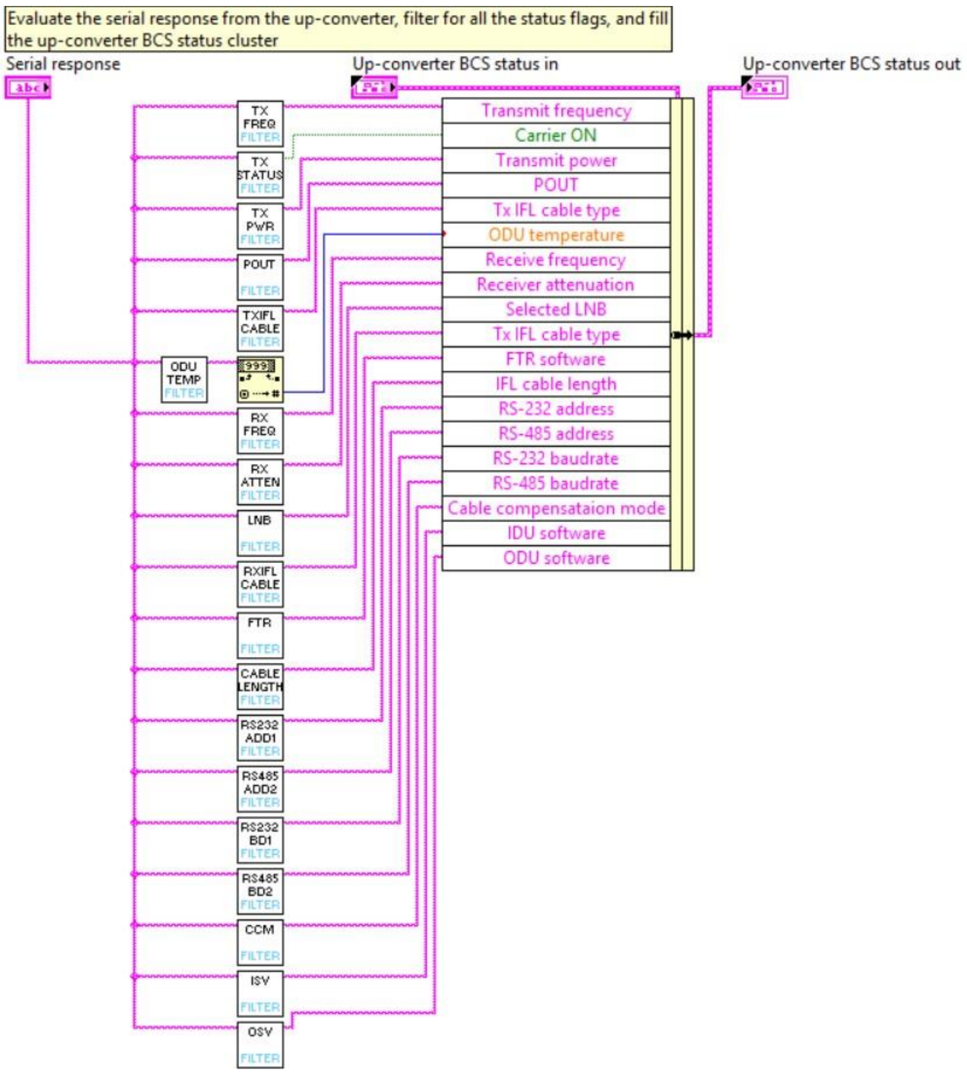
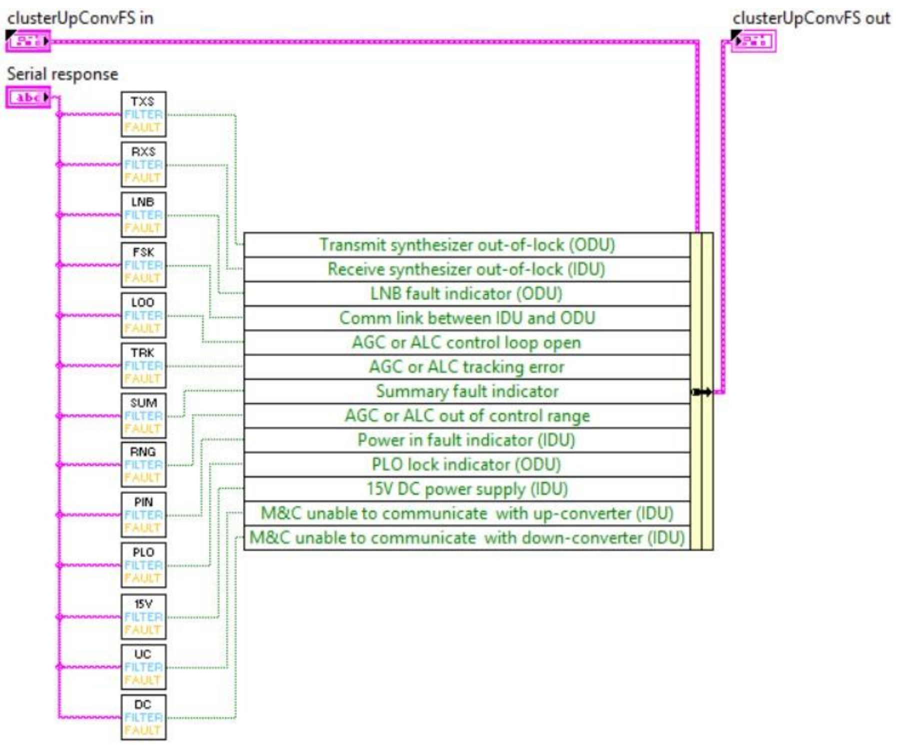


Figure 4-61 Block diagram of the VI UpConvBCSStatus (BCS STATUS).

4.4.8.7 Fault status

The fault status (FS) cluster, as shown in Figure 4-50d, is a collection of LEDs that show if any fault occurred during operation of the up-converter. Reading the faults follows the same principle that is used reading the BCS in the previous chapter. Sending the command '**FS_**' to the up-converter returns a series of commands, each indicating the status of one fault. Figure 4-62 shows the block diagram of the Sub-VI **FS STATUS**.



clusterUpConvFS in
clusterUpConvFS out
Serial response
TXS FILTER FAULT
RXS FILTER FAULT
LNB FILTER FAULT
FSK FILTER FAULT
LOO FILTER FAULT
TRK FILTER FAULT
SUM FILTER FAULT
RNG FILTER FAULT
PIN FILTER FAULT
PLO FILTER FAULT
15V FILTER FAULT
UC FILTER FAULT
DC FILTER FAULT
Transmit synthesizer out-of-lock (ODU)
Receive synthesizer out-of-lock (IDU)
LNB fault indicator (ODU)
Comm link between IDU and ODU
AGC or ALC control loop open
AGC or ALC tracking error
Summary fault indicator
AGC or ALC out of control range
Power in fault indicator (IDU)
PLO lock indicator (ODU)
15V DC power supply (IDU)
M&C unable to communicate with up-converter (IDU)
M&C unable to communicate with down-converter (IDU)

Figure 4-62 Block diagram of the VI UpConvFSStatus (FS STATUS).

Here the **Serial response** from the up-converter is run through a stack of filters, each filtering one of the fault status flags and then passes the fault flag on to the FS cluster **clusterUpConvFS out**.

4.4.9 Waveguide switches

The receive signal of the satellite station can be routed to either one of four different receive paths A, B, C and D or it can be routed to a hot load or a cold load. In order to route the signal, the physical signal path is changed with the use of waveguide switches. Figure 4-63 shows the schematic of the station’s physical high frequency path, where the waveguide switches are located. There are nine waveguide switches, marked with **SW1 – SW9**. SW1 is highlighted in the orange frame at the left, SW2-9 can be found in the centre orange box and the controls for changing the position of the waveguide switches are highlighted on the right-hand side panel.

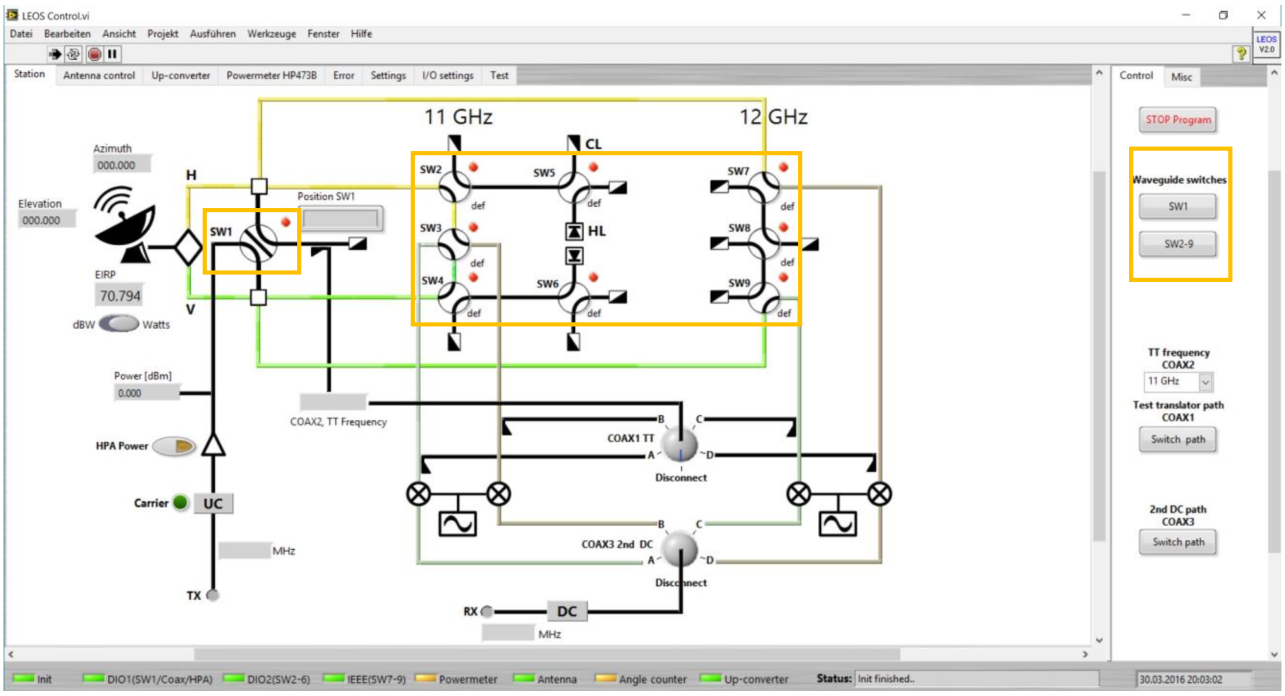


Figure 4-63 LEOSv2 GUI: block diagram of the physical high frequency path. Waveguide switches and corresponding controls are marekd orange.

The built in switches are either controlled by the National Instruments Digital-I/O or IEEE converters. Switches 1 to 6 are Digital-I/O controlled and SW1 can be switched between three different positions while SW2 - SW6 are 2-way switches. The IEEE switches SW7-SW9 are also have two possible positions. Each of the switches in the GUI is accompanied by a status LED positioned to its upper right that indicates if communication with the switch via the converters is available. Changing switch positions with the controls at the right opens a dialog so that the user can check his choice. Only after confirming the dialog, the switch position is changed. If a position is changed, the HPA is switched off to avoid the risk of damaging hardware components.

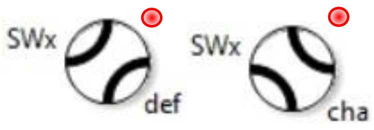
4.4.9.1 Digital-I/O switches

There are two types of Digital-I/O switches: one 3-way switch, SW1, and five 2-way switches, SW2-SW6. Each switch position has an associated drive port and an indicator port. This means that SW1 requires 3 drive ports and 3 indicator ports and therefore 6 Digital-I/O ports. The switches SW2-SW6 have 2 drive ports and 2 indicator ports, therefore require a total of 4 Digital-I/O ports. Changing the position of the Digital-I/O switches requires an impulse of roughly **100 ms** at the drive ports. A continuous signal on the drive ports may destroy the switches!

The indicators of the switches present the switches' position in negative logic, meaning that an indicator (and therefore the corresponding switch position) is active when it is at a **low** level.

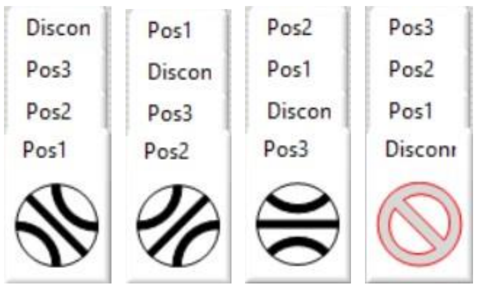
Creating the GUI elements for the switches has been done by equipping available LabVIEW controls with the images that represent the switch position.

For the **2-way switches** SW2 - SW6 a Boolean control has been used as a basis. Then the Boolean control's front panel image has been replaced with one of the two switch positions:



The text to the lower right of the switch is either **def**, standing for default (True) or **cha**, standing for changed (False). The status LED to the top right indicates if the switch is available

The **3-way switch** SW1 is a bit more complicated to create, since Boolean switches naturally allow for only two different states. For SW1 a tabbed pane has been used and each tab contains one possible switch position.



Instead of a status LED, the 4th tab contains the state **Disconnected**, meaning that the switch is not available.

In order to create a nice GUI design, showing the tabs of the tabbed pane can be disabled by right clicking the tabbed pane in LabVIEW and unchecking the show tab's option in the menu "Visible objects" (German LabVIEW version: Menu "Sichtbare Objekte" -> "Registerkarten"). Doing so still leaves the main pane visible as a square around the switch symbol, so the main pane can also be made invisible by using a transparent “colour” with the help of the LabVIEW Toolbox. The active tab of a tabbed pane can be controlled by integer numbers. Sending a number **1** to the pane switches it to **position 1**, number **2** switches to **position 2, 3** to **position 3** and **4** to position **Disconnected**.

4.4.9.1.1 Waveguide switch 1 (SW1)

The waveguide switch SW1 routes the transmit signal to either one of three possible signal paths: horizontally polarized, vertically polarized or test translator path. Pressing the **SW1** button in the GUI (see Figure 4-63 to the right) triggers the event case **sw1_position_change**, shown in Figure 4-64. This event case allows changing the position of SW1 and therefore a re-route of the transmit signal.

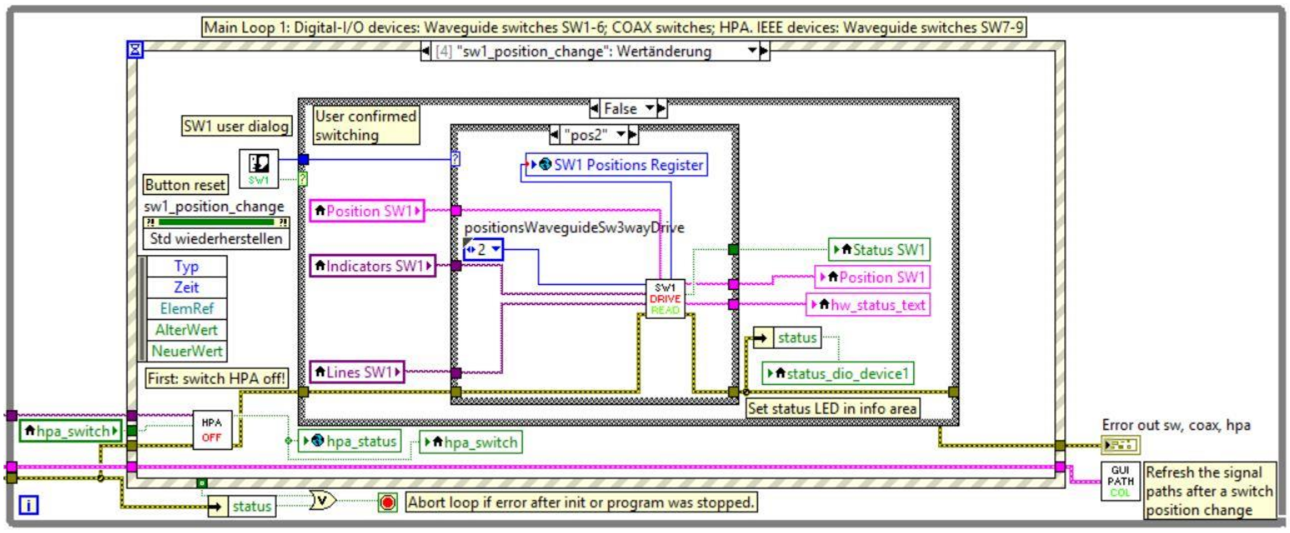


Figure 4-64 Event case for switching the 3-way waveguide switch SW1.

Starting from the left within the event structure, the HPA is switched off first (Sub-VI **HPA OFF**). Labelled under **Button reset**, the state of the SW1 GUI button is restored to default. Figure 4-65 shows the user dialog for changing the position of SW1 (label **SW1 user dialog** on the top left).

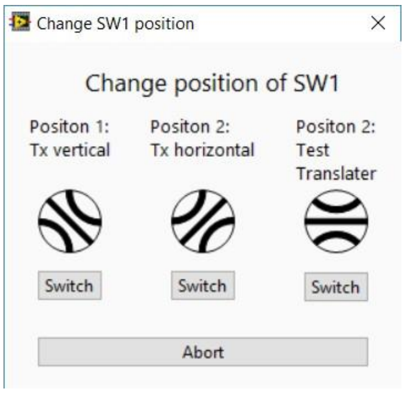


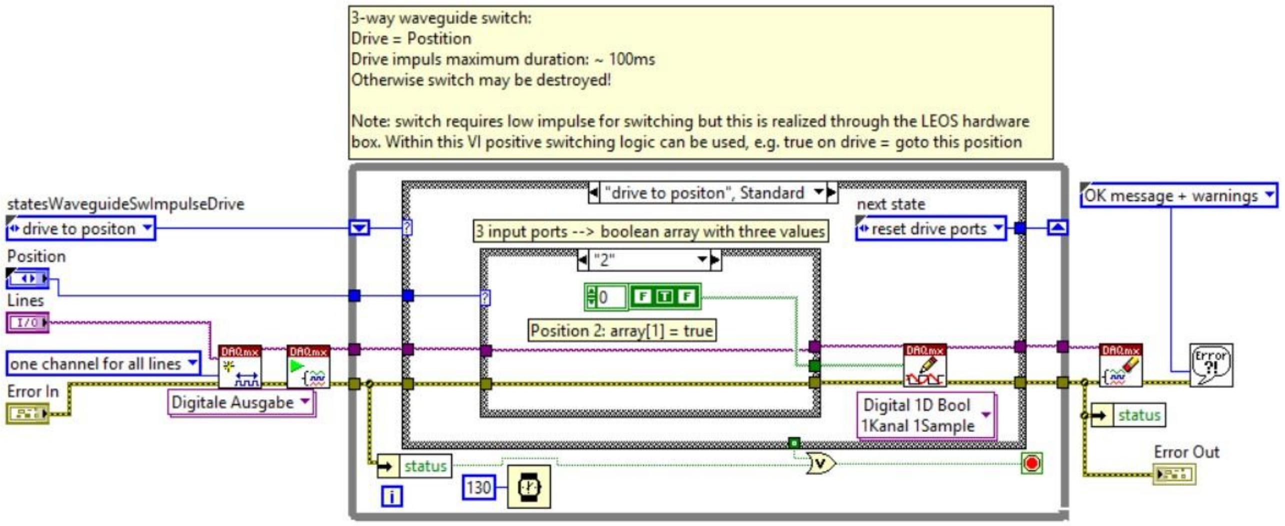
Figure 4-65 User dialog for changing the position of SW1.

This dialog shows the three possible switch positions for SW1 and offers a short description to each position. If the user confirms the position change by pressing one of the **Switch** buttons, the outer case structure is entered. In this case structure the position for SW1, as well as its indicators and lines (drives) are set.

The inner case structure sends the position selected by the user to the Sub-VI **SW1 DRIVE READ**. This Sub-VI contains a state machine that drives the switch to the new position and confirms the new position afterwards, by reading the SW1 indicators.

Note: *The drive ports for SW1 only require a short impulse of ~100 ms for changing the switche's position. A permanently active drive port may damage the switche's electronics!*
A drive port of the switch actually requires a low pulse. The LEOS hardware box inverts the positive Digital-I/O signals, so that within the LEOSv2 software all drive ports can be controlled with positive pulses.

To drive to a new position, the Sub-VI **SW1 DRIVE** (see Figure 4-66) is used.


3-way waveguide switch:
Drive = Postition
Drive impuls maximum duration: ~ 100ms
Otherwise switch may be destroyed!

Note: switch requires low impulse for switching but this is realized through the LEOS hardware box. Within this VI positive switching logic can be used, e.g. true on drive = goto this position

statesWaveguideSwImpulseDrive
drive to positon
Position
Lines
one channel for all lines
Error In
DAQmx
Digitale Ausgabe
"drive to positon", Standard
3 input ports --> boolean array with three values
"2"
0
F T F
Position 2: array[1] = true
next state
reset drive ports
Digital 1D Bool 1Kanal 1Sample
status
130
OK message + warnings
status
Error Out

Figure 4-66 Block diagram of the VI WaveguideSW3wayDrive (SW1 DRIVE), showing the state machine case "drive to position".

This Sub-VI again contains a state machine that first sends the new position to the switches drive ports and resets the drive ports after 130 ms. Starting from the left in Figure 4-66, the DAQmx resource for SW1 is opened, enabling communication on the Digital-I/O ports of the National Instruments Digital-I/O converters. The while loop and the outer case structure represent the state machine, and the inner case structure is used to change the switche's position by sending a Boolean array of length 3 to the 3 drive ports of SW1. Here the example case for switching SW1 to position 2 is shown. This happens in the state machine case **drive to position**. After this case is complete, the next case **reset drive ports** is started. Figure 4-67 shows the case **reset drive ports**. Here the DAQmx drive ports for SW1 are all set to **false**, in order to reset any active drive port.

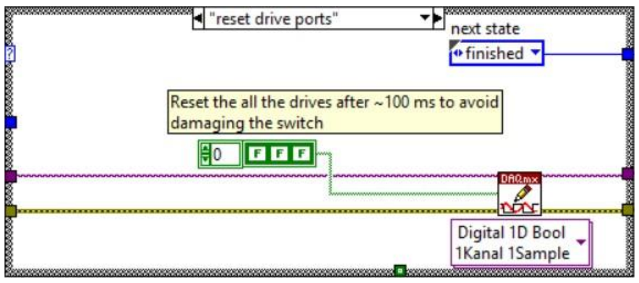
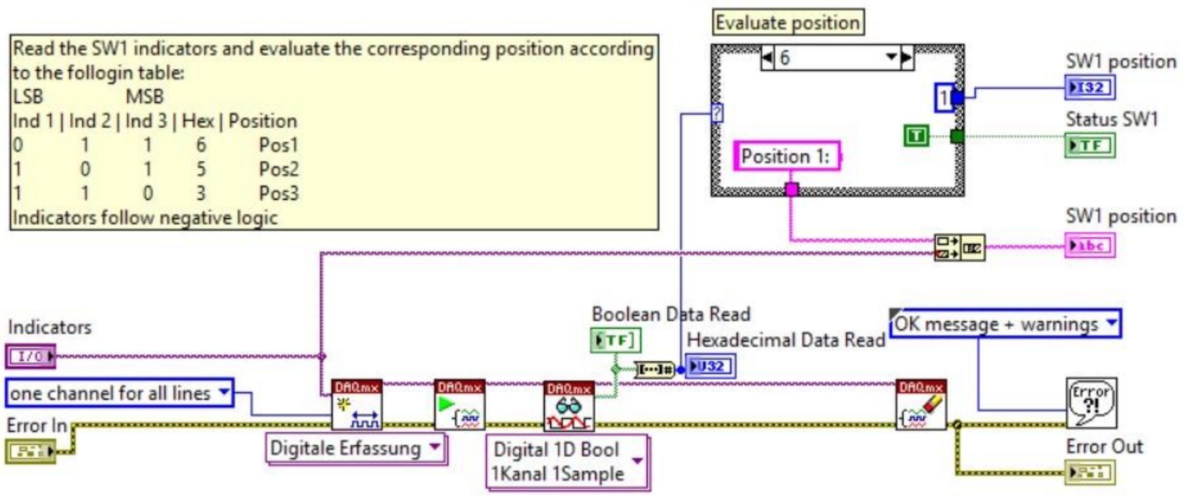

Reset the all the drives after ~100 ms to avoid damaging the switch

Figure 4-67 VI WaveguideSW3wayDrive, state machine case "reset drive ports".

After the reset of the drive ports, the state machine stops and the DAQmx resource is closed.

Reading the positions for SW1 requires evaluation of the switches' indicator ports. There are three indicator ports, one for each position. Figure 4-68 shows the Sub-VI **SW1 POS** where a DAQmx resource is opened and the Digital-I/O ports that correspond to the SW1 indicators are returned as a Boolean array.


Read the SW1 indicators and evaluate the corresponding position according to the follogin table:

LSB		MSB		
Ind 1	Ind 2	Ind 3	Hex	Position
0	1	1	6	Pos1
1	0	1	5	Pos2
1	1	0	3	Pos3

Indicators follow negative logic

Figure 4-68 Block diagram of the VI Waveguide SW3wayIndicators (SW1 POS).

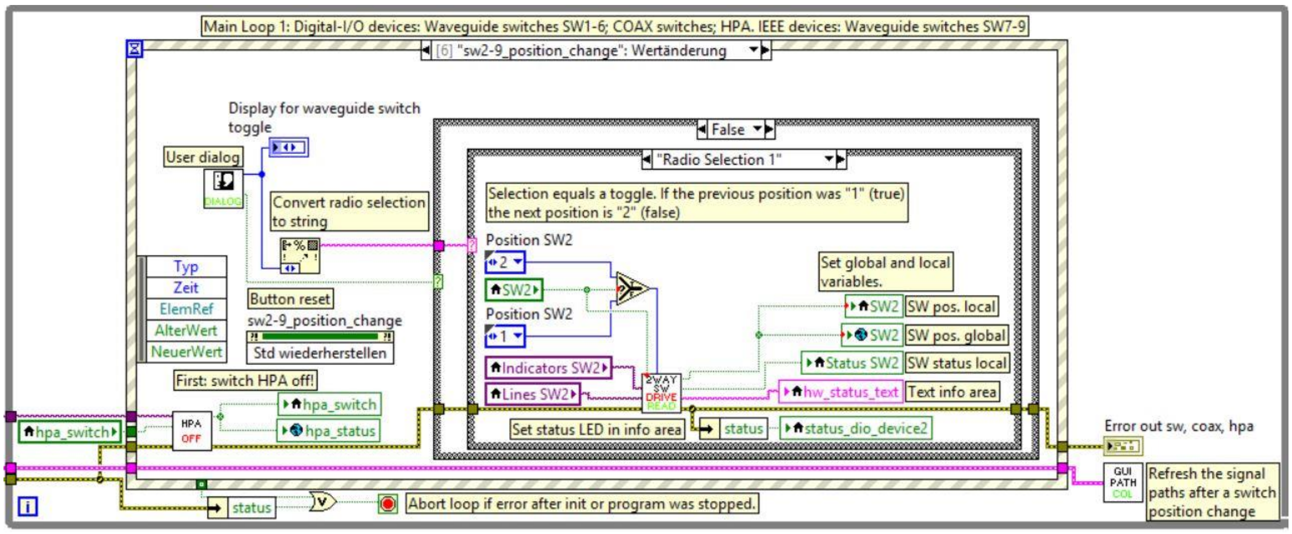
The Boolean array contains the indicator states in negative logic, meaning that a Boolean **false** marks an active indicator. By converting the 3 bit Boolean array into a number, the position of the switch can be regained. Table 5 shows the possible configuration of the SW1 indicator ports, the corresponding hexadecimal number, the associated position of the switch and the transmit signal path for each position.

Table 5 Waveguide switch SW1 indicators, positions and corresponding signal paths.

LSB		*MSB*			
Ind 1	**Ind 2**	**Ind 3**	**Hex**	**Position**	**Signal path**
0	1	1	6	Position 1	Tx vertical
1	0	1	5	Position 2	Tx horizontal
1	1	0	3	Position 3	Tx test translator

4.4.9.1.2 Waveguide switches 2-way

The 2-way waveguide switches SW2 – SW9 are used for routing the received signal to one of the four signal paths A, B, C or D or to terminate the signal in a hot load or a cold load. The switches SW2 – SW6 are Digital-I/O switches and SW7 – SW9 are IEEE switches. Changing the position of any of those switches is handled within the same event structure, though communication with the IEEE switches, of course, uses different VIs than those used for the Digital-I/O switches. The IEEE switches will be explained in the next chapter. Each of the switches SW2 – SW9 can be toggled between one of its two possible positions by pressing the **SW2 - SW9** button in the GUI (see Figure 4-63 in the right), which triggers the event case **sw2-9_position_change**, shown in Figure 4-69.



Main Loop 1: Digital-I/O devices: Waveguide switches SW1-6; COAX switches; HPA. IEEE devices: Waveguide switches SW7-9
[6] "sw2-9_position_change": Wertänderung
Display for waveguide switch toggle
User dialog
DIALOG
Convert radio selection to string
Typ
Zeit
ElemRef
AlterWert
NeuerWert
Button reset
sw2-9_position_change
Std wiederherstellen
First: switch HPA off!
hpa_switch
hpa_status
HPA OFF
False
"Radio Selection 1"
Selection equals a toggle. If the previous position was "1" (true) the next position is "2" (false)
Position SW2
2
SW2
Position SW2
1
Indicators SW2
Lines SW2
2WAY SW DRIVE READ
Set global and local variables.
SW2
SW pos. local
SW2
SW pos. global
Status SW2
SW status local
hw_status_text
Text info area
Set status LED in info area
status
status_dio_device2
Error out sw, coax, hpa
GUI PATH COL
Refresh the signal paths after a switch position change
status
Abort loop if error after init or program was stopped.

Figure 4-69 Event case for switching a 2-way Digital-I/O waveguide switch.

Starting from the left within the event structure, the HPA is switched off first (Sub-VI **HPA OFF**). Labelled under **Button reset**, the state of the SW1 GUI button is restored to default. On the top left a **User dialog** is created, as shown in Figure 4-70.

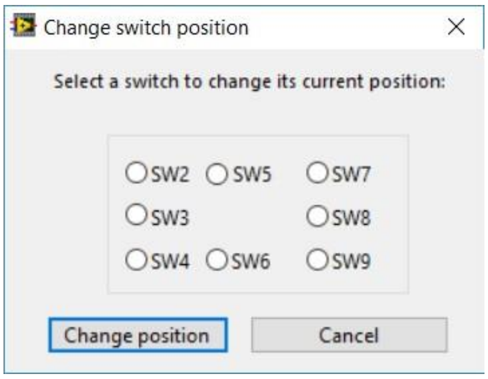


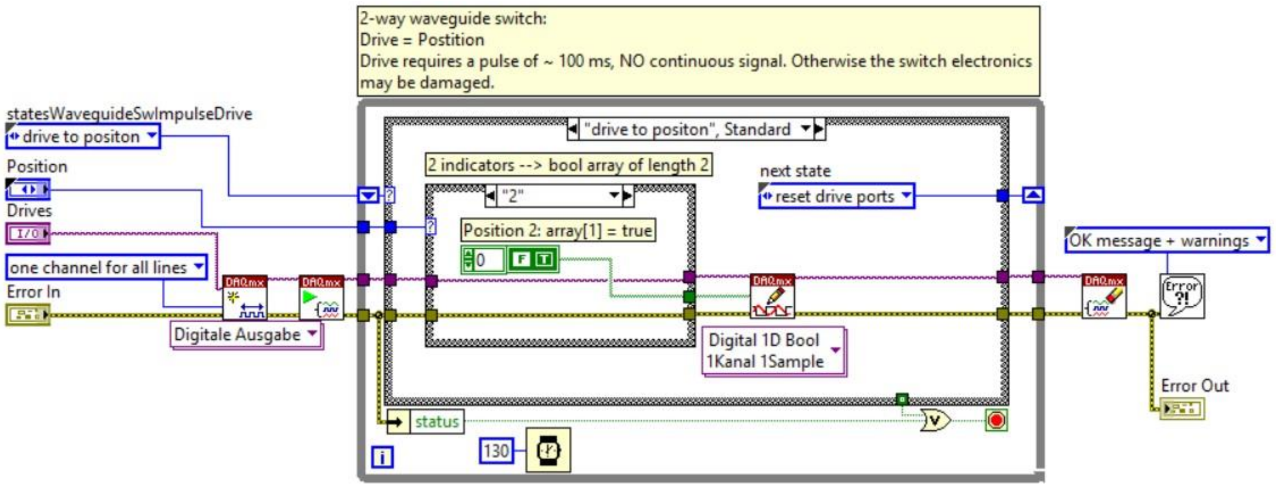
Figure 4-70 User dialog for changing the position of a 2-way Digital-I/O waveguide switch.

The layout of the radio buttons for the switches in the dialog is similar to the block diagram in the GUI, so that the desired switch can easily be found.

By pressing the button **Change position** the outer case structure in Figure 4-69 is entered. The inner case structure evaluates the selected switch in the dialog and toggles this switch. Digital-I/O lines (drives) and indicator ports are set and the Sub-VI **2-WAY SW DRIVE RAED** (not shown in a figure) is called. This Sub-VI sets a new switch position and then confirms the switches position within a state machine with the help of the Sub-VIs **SW 2-WAY DRIVE** and **SW 2-WAY POS**.

Note: *The drive ports for SW2 – SW6 only require a short impulse of ~100 ms for changing the switche's position. A permanently active drive port may damage the switches electronics!*

Figure 4-71 shows the Sub-VI **SW 2-WAY DRIVE**, where the switch is driven to a new position by a pulse of 130 ms.


2-way waveguide switch:
Drive = Postition
Drive requires a pulse of ~ 100 ms, NO continuous signal. Otherwise the switch electronics may be damaged.

2 indicators --> bool array of length 2

Position 2: array[1] = true

Figure 4-71 Block diagram of the VI WaveguideSW2wayDrive (SW 2-WAY DRIVE), showing the state machine case "drive to position".

Starting from the left, a DAQmx resource for the selected switch is opened, enabling communication on the Digital-I/O ports of the National Instruments Digital-I/O converters. The loop and the outer case structure form a state machine. In the state **drive to position** a Boolean array of length 2 is sent to the two drive ports of a 2-way switch. After 130 ms the next case **reset drive ports** (see Figure 4-72) switches the active Digital-I/O port back to the **low** state.

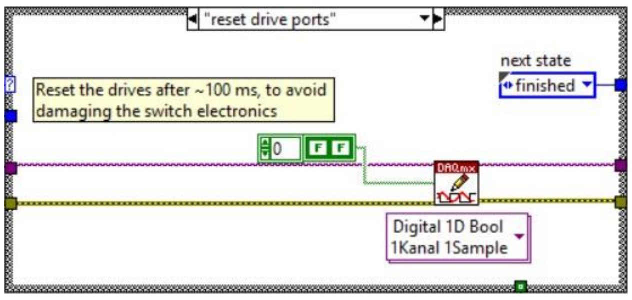
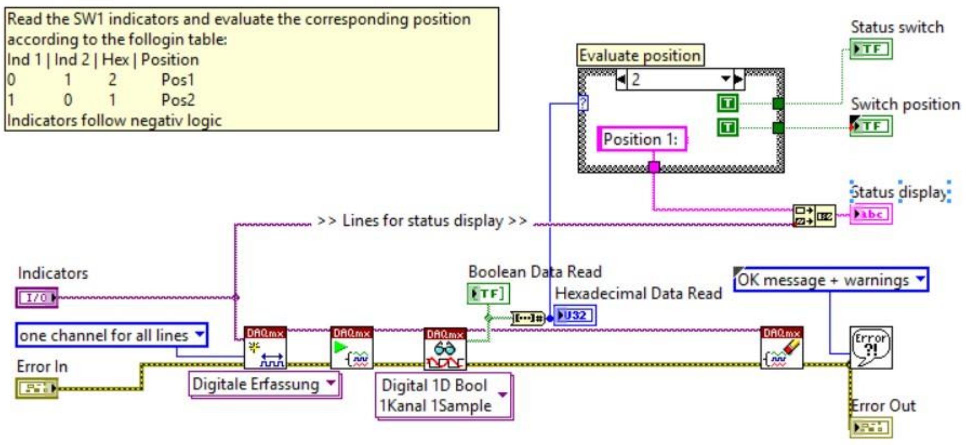

Reset the drives after ~100 ms, to avoid damaging the switch electronics

Figure 4-72 VI WaveguideSW2wayDrive, state machine case "reset drive ports".

After resetting of the drive ports, the state machine stops and the DAQmx resource is closed.

Reading the positions for the 2-way waveguide switches requires evaluation of the switches' indicator ports. There are two indicator ports, one for each position. Figure 4-73 shows the Sub-VI **SW 2-WAY POS** where a DAQmx resource is opened and the Digital-I/O ports that correspond to the 2-way switch indicators are returned as a Boolean array.



Read the SW1 indicators and evaluate the corresponding position according to the follogin table:

Ind 1	Ind 2	Hex	Position
0	1	2	Pos1
1	0	1	Pos2

Indicators follow negativ logic

Figure 4-73 Block diagram of the VI Waveguide SW2wayIndicators (SW 2-WAY POS).

The Boolean array contains the indicator states in negative logic, meaning that a Boolean **false** marks an active indicator. By converting the 2 bit Boolean array into a number, the position of the switch can be regained. Table 6 shows the possible configuration of the 2-way switch indicator ports, the corresponding hexadecimal number and the associated position of the switch.

Table 6 Indicators and corresponding positions for 2-way waveguide switches.

LSB	*MSB*		
Ind 1	**Ind 2**	**Hex**	**Position**
0	1	2	Position 1
1	0	1	Position 2
0	0	0	N/A

If both indicator ports return '**0**', the position is unknown and the status LED **Status switch** is set to **false** (red).

4.4.9.2 IEEE switches

Like the 2-way Digital-I/O switches the 2-way IEEE switches are used to route the received signal to one of the four signal paths A, B, C or D. The LEOSv2 GUI representation of the IEEE switches does not differ from the Digital-I/O switches, however, changing the position and reading the position of those switches differs. Figure 4-74 shows the event **case sw2-9_position_change** and the selected case in the inner case structure is **Radio Selection 6** which is used to change the position of IEEE switch **7**. Radio selections in LabVIEW start with the number 0, therefore there is an offset of 1 between the radio selection and the switch number.

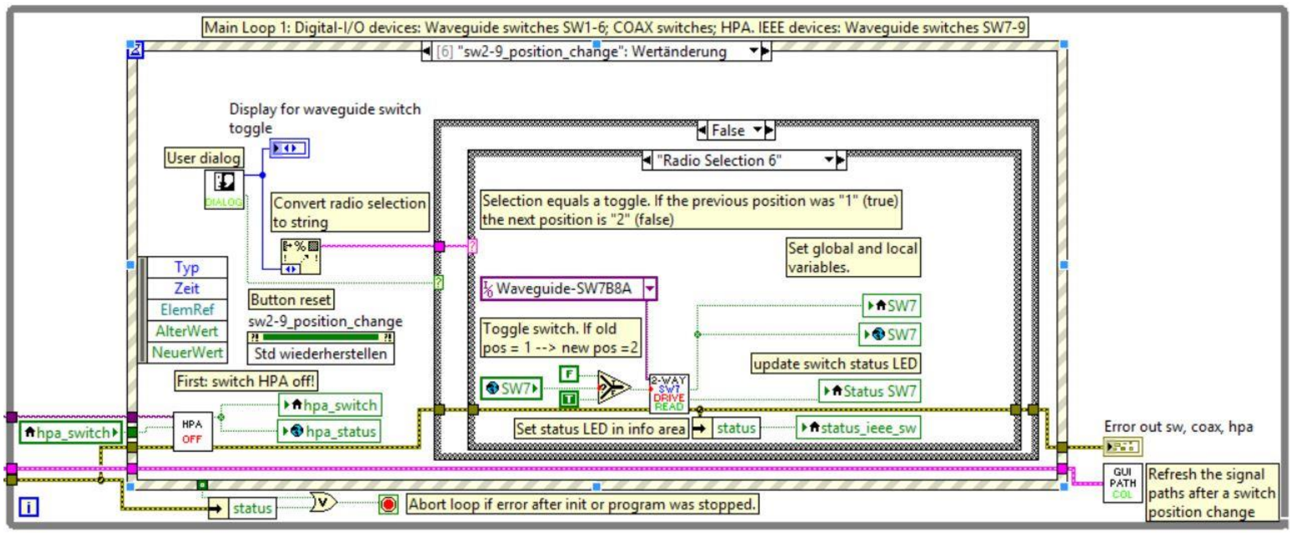

Main Loop 1: Digital-I/O devices: Waveguide switches SW1-6; COAX switches; HPA. IEEE devices: Waveguide switches SW7-9
[6] "sw2-9_position_change": Wertänderung
Display for waveguide switch toggle
User dialog
Convert radio selection to string
Typ
Zeit
ElemRef
AlterWert
NeuerWert
Button reset
sw2-9_position_change
Std wiederherstellen
First: switch HPA off!
hpa_switch
hpa_status
hpa_switch
False
"Radio Selection 6"
Selection equals a toggle. If the previous position was "1" (true) the next position is "2" (false)
Set global and local variables.
Waveguide-SW7B8A
Toggle switch. If old pos = 1 --> new pos =2
SW7
SW7
SW7
update switch status LED
Status SW7
2-WAY SW7 DRIVE READ
Set status LED in info area
status
status_ieee_sw
Error out sw, coax, hpa
Refresh the signal paths after a switch position change
status
Abort loop if error after init or program was stopped.

Figure 4-74 Event case for switching a 2-way IEEE waveguide switch.

The event case is very similar, as described in the previous chapter 4.4.9.1.2. By selecting either SW7, SW8 or SW9 in the user dialog (Figure 4-75) and confirming the selection with **Change positions** the selected switch will be toggled in position with the help of the Sub-VI **2-WAY SWx DRIVE READ** (SWx stands for the selected switch).

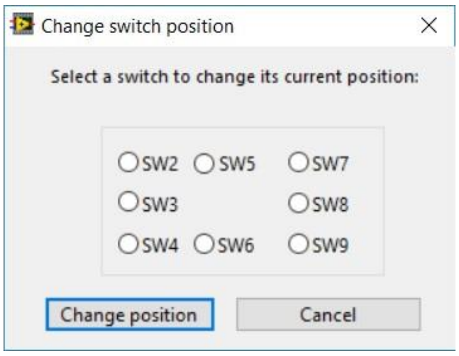


Figure 4-75 User dialog for changing the position of a 2-way IEEE waveguide switch.

As an example the Sub-VI **2-WAY SW7 DRIVE READ** contains a state machine that first sets the switch to a new position and checks the position afterwards using the Sub-VIs **SW7 SET POS** and **SW7 GET POS.**

Changing the position of an IEEE switch uses a very simple command, namely the IEEE address of the switch plus a 1 digit number that defines the position in the format **<address><position>**. The following Table 7 shows the commands that are used for the IEEE switches:

Switch	Address	Position string (drive / read)	Position of switch
7	B	1	Position 1
		3	Position 2
8	A	1	Position 1
		3	Position 2
9	A	1	Position 1
		3	Position 2

Table 7 Commands and addresses for 2-way IEEE waveguide switches.

An example command for setting switch SW7 to position 1 would simply be the string "**B1"**. Figure 4-76 shows the Sub-VI **SW7 SET POS** where the command **B1** is sent to a VISA-Resource that communicates with the National Instruments IEEE converter (located in the antenna box) that controls the IEEE switches.

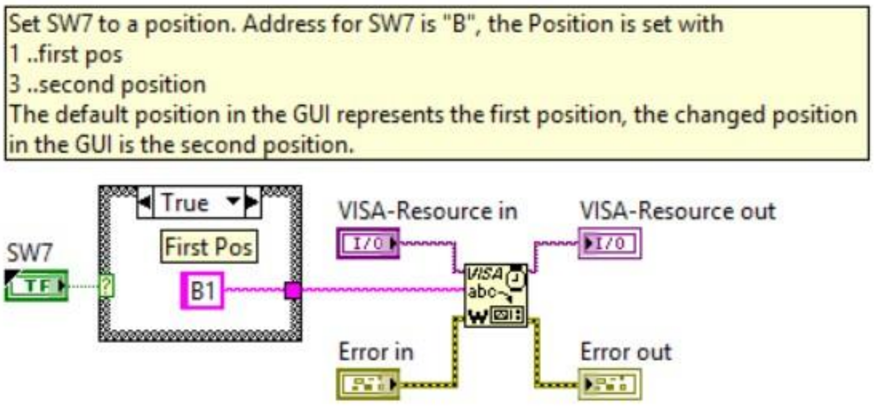

Set SW7 to a position. Address for SW7 is "B", the Position is set with
1 ..first pos
3 ..second position
The default position in the GUI represents the first position, the changed position in the GUI is the second position.

Figure 4-76 Block diagram of the VI WaveguideSw7SetPos (SW7 SET POS).

The VIs for changing the position of SW8 and SW9 follow the same principle and are not shown in this chapter. Reading the position of an IEEE switch just returns the position of the switch without its address. The VISA-Resource for switches SW7 and SW8 always returns the position of both switches at once in the format **<position SW8>,<position SW7>**. So the first byte that is returned belongs to SW8, the second byte is a comma and the third byte belongs to SW7. Figure 4-77 shows the Sub-VI **SW7 GET POS**, that reads the position for IEEE switch 7.

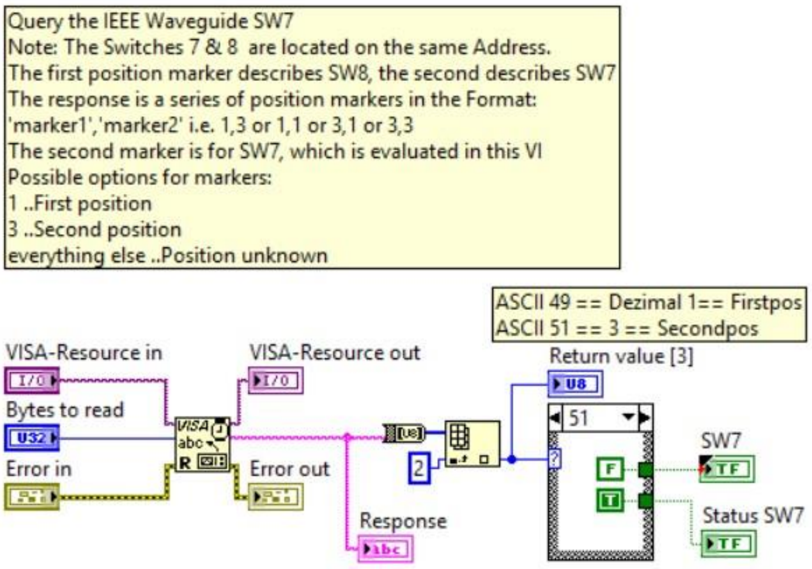

Query the IEEE Waveguide SW7
Note: The Switches 7 & 8 are located on the same Address.
The first position marker describes SW8, the second describes SW7
The response is a series of position markers in the Format:
'marker1','marker2' i.e. 1,3 or 1,1 or 3,1 or 3,3
The second marker is for SW7, which is evaluated in this VI
Possible options for markers:
1 ..First position
3 ..Second position
everything else ..Position unknown
ASCII 49 == Dezimal 1== Firstpos
ASCII 51 == 3 == Secondpos
VISA-Resource in
Bytes to read
Error in
VISA-Resource out
Error out
Response
Return value [3]
51
SW7
Status SW7

Figure 4-77 Block diagram of the VI WaveguideSw7GetPos (SW7 GET POS).

The response from the IEEE converter is a string that is then converted into a character array. The third Byte of the character array contains the position number of SW7. This position is represented as ASCII code, meaning that an ASCII value of **49** equals a 1, or **position 1** and the ASCII value **51** equals 3, or **position 2**. The same principle is used for SW8, only here the first Byte of the response from the IEEE converter is evaluated. IEEE switch SW9 returns a different string regarding its current position (see Figure 4-78). The first character is a carriage return (\r\n) and the second character contains the position, again either the ASCII value 49 for **position 1** or ASCII 51 for **position 2**.

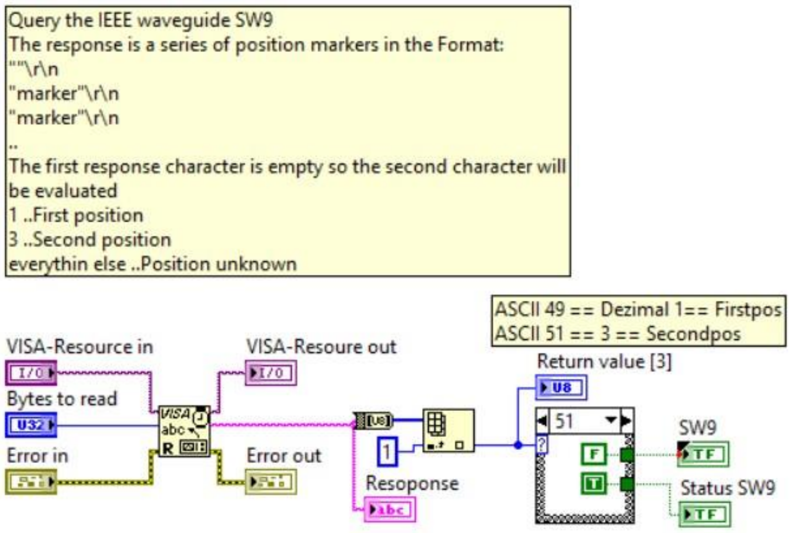

Query the IEEE waveguide SW9
The response is a series of position markers in the Format:
""\r\n
"marker"\r\n
"marker"\r\n
..
The first response character is empty so the second character will be evaluated
1 ..First position
3 ..Second position
everythin else ..Position unknown
ASCII 49 == Dezimal 1== Firstpos
ASCII 51 == 3 == Secondpos
VISA-Resource in
Bytes to read
Error in
VISA-Resoure out
Error out
Resoponse
Return value [3]
51
SW9
Status SW9

Figure 4-78 Block diagram of the VI WaveguideSw9GetPos (SW9 GET POS).

4.4.10 High frequency path colouring

The stations physical high frequency path contains a transmit path, four receive paths, A, B, C and D and a test translator path. To make it visually more obvious which path is currently active, the paths are coloured accordingly. The state of a path and therefore its colour depends on the position of the waveguide switches SW2-9 and the coaxial switches COAX1 and COAX3. COAX2 only switches the test translator frequency and does not influence the path colouring. Figure 4-79 shows the horizontally polarized paths coloured in yellow and the vertically polarized paths in green. The darker shaded colours indicate that this signal path is routed between the waveguide switches and the coaxial switch **COAX3 2nd DC** but the coaxial switch is disconnected, meaning that the signal is not passed on to the down-converter.

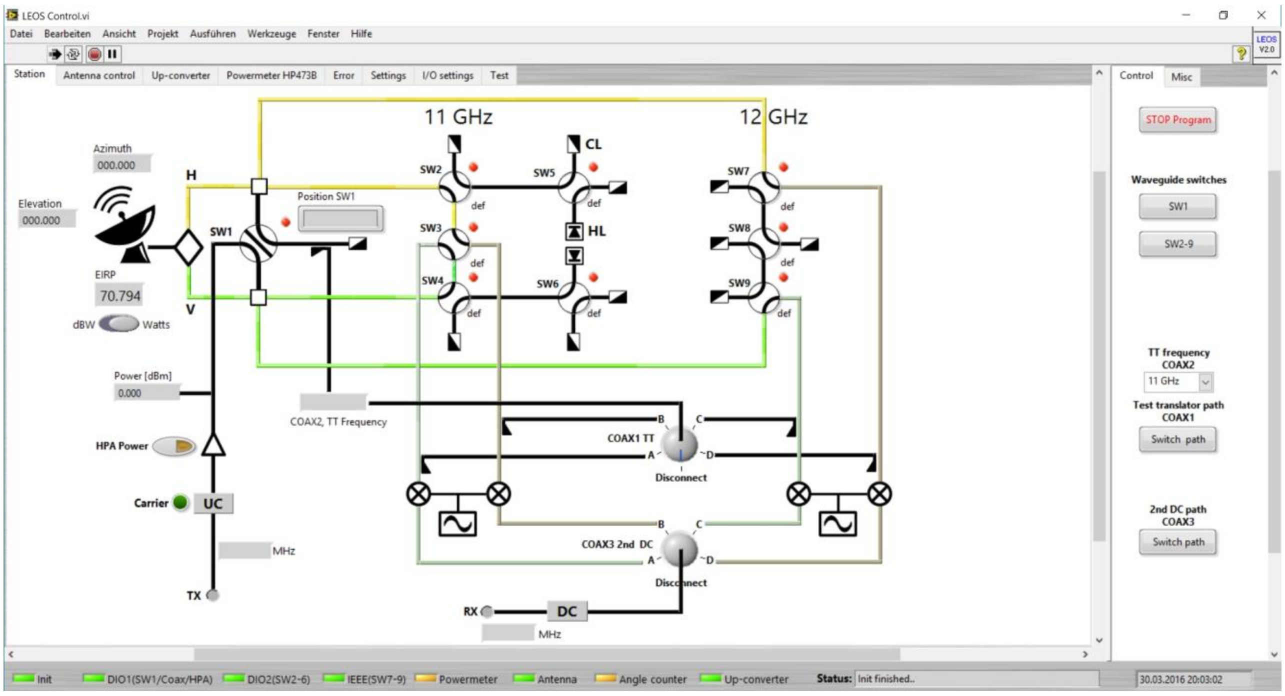


Figure 4-79 LEOSv2 GUI, showing the physical high frequency path.

If now COAX3 is switched to either position A, B, C or D, the corresponding signal path will show a bright colour, meaning that this path is now active. Note that the signal below COAX3, from COAX3 to the label **RX** is always displayed in green if active, regardless of the polarization.

As for the transmitting path, shown at the left-hand side in Figure 4-79, the line between the label **TX** and the HPA will be coloured in green, if the up-converter transmitter is switched on. Further, if the HPA is switched on, the line between the HPA and SW1 will also appear green. If up-converter transmitter and HPA are switched on and SW1 is in its test translator position, the whole path from the label **TX** to **COAX1 TT** appears in green.

Let's now look at the logic behind the path colouring by examining the corresponding block diagrams. First it is important to mention that all the (coloured) lines in Figure 4-79 are Boolean indicators. Those indicators are grouped together in the LEOSv2 main block diagram, as shown in Figure 4-80.

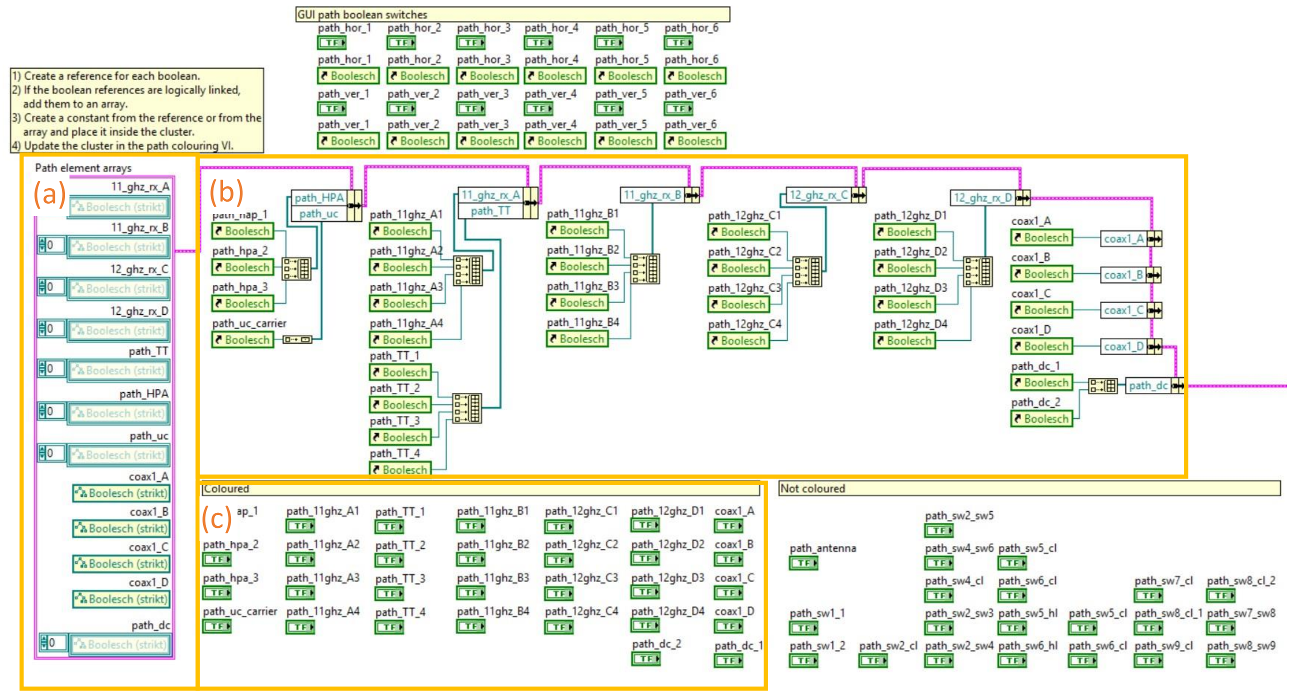

1) Create a reference for each boolean.
2) If the boolean references are logically linked, add them to an array.
3) Create a constant from the reference or from the array and place it inside the cluster.
4) Update the cluster in the path colouring VI.

Figure 4-80 LEOSv2 main block diagram, showing the area for the GUI path colouring. Panel (a) shows the cluster that contains a constant for each Boolean reference or group of references. Panel (b) contains the Boolean references that are grouped and added to the cluster. Panel (c) shows Boolean variables that are currently coloured in the GUI.

In order to change the colour of a Boolean in LabVIEW, a reference to this Boolean needs to be created. Figure 4-80c shows the Boolean variables that are used for the coloured signal paths in the GUI. Figure 4-80b contains a reference for each Boolean in Figure 4-80c and groups references that are logically linked in arrays. Those arrays are then added to the cluster constant, shown in Figure 4-80a, that contains a collection of all references. All unmarked elements in Figure 4-80 are part of the GUI signal path but are not coloured.

4.4.10.1 Colouring the receive path

The cluster constant containing the Boolean references is then passed on to the VI **GuiPathColouring**. This VI checks the current state of the signal paths and applies the according colour. A colour is chosen based on the state of the signal path. There are five available colours for the five cases:

Table 8 Available colours for the signal paths.

Path state	**Colour**
Terminated	Black
Vertically available	Dim green
Vertically active	Bright green
Horizontally available	Dim yellow
Horizontally active	Bright yellow

The path state “available” in Table 8 means that a receive signal path is routed all the way to switch COAX3, but COAX3 is disconnected. Once COAX3 is switched to a signal path, the state of this path becomes “active” because the signal is now routed to the down-converter. The state “terminated” means that the signal is ends in a hot load or a cold load. Table 9 gives an overview over the receive signal paths, the corresponding frequency and the possible polarizations for each path.

Table 9 Receive signal paths, frequencies and polarizations.

Signal path	**Frequency [GHz]**	**Polarization**
A	11	H, V
B	11	H, V
C	12	V
D	12	H

Now that we know the possible configurations of signal paths, frequencies, polarizations and colours we can look at the VI **GuiPathColouring**, which checks the active signal path and applies the corresponding colour to the Boolean elements in the GUI. Figure 4-81 shows the block diagram of this VI and the case **11GHz-A**. The outer loop and the case structure form a state machine that checks the state of all available signal paths.

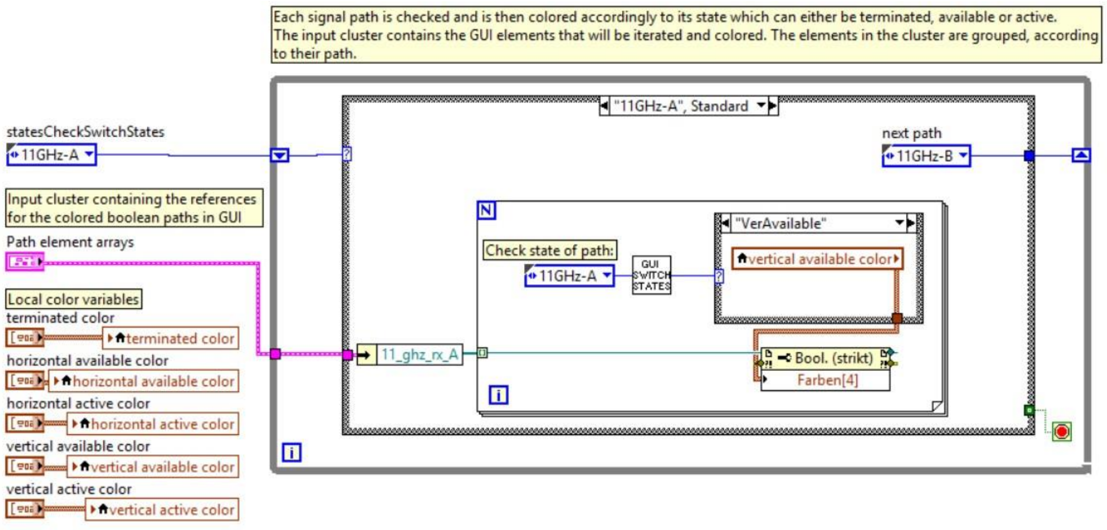

Each signal path is checked and is then colored accordingly to its state which can either be terminated, available or active.
The input cluster contains the GUI elements that will be iterated and colored. The elements in the cluster are grouped, according to their path.
statesCheckSwitchStates
11GHz-A
"11GHz-A", Standard
next path
11GHz-B
Input cluster containing the references for the colored boolean paths in GUI
Path element arrays
Check state of path:
11GHz-A
GUI SWITCH STATES
"VerAvailable"
vertical available color
Local color variables
terminated color
terminated color
11_ghz_rx_A
Bool. (strikt)
Farben[4]
horizontal available color
horizontal available color
horizontal active color
horizontal active color
vertical available color
vertical available color
vertical active color
vertical active color

Figure 4-81 Block diagram of the VI GuiPathColouring (GUI PATH COL).

Remember that the input cluster **Path element arrays** (see Figure 4-81) contains references of the Boolean path elements in the GUI. Each state of the state machine picks out the corresponding references of the cluster and applies the appropriate colour within the for-loop. The appropriate colour is chosen according to the state of the corresponding signal path with the help of the VI **GUI SWITCH STATES**. This VI checks the current positions of the waveguide switches and determines the signal path that emerges by their configuration. Figure 4-82 shows the block diagram of this VI. As an example we will now examine the depicted case **VerAvailable** (inner case structure) for the signal path **11GHz-A** (outer case structure).

In order for the signal path **A** to be considered **vertically available** the following conditions are required: SW3 and SW4 need to be in their default positions and COAX3 cannot be in positon A. If these conditions hold, the state of this signal path has been determined and the loop is aborted. If these conditions don't hold, the signal path 11GHz-A is considered terminated and the inner case structure switches to the next case, until the correct state for this signal path is found.

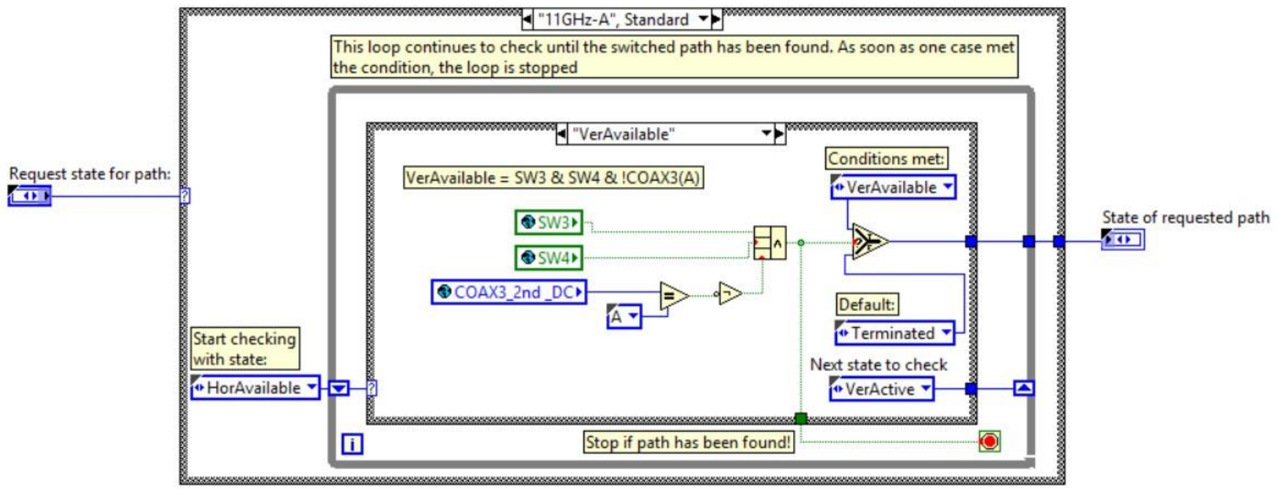

"11GHz-A", Standard
This loop continues to check until the switched path has been found. As soon as one case met the condition, the loop is stopped
Request state for path:
"VerAvailable"
VerAvailable = SW3 & SW4 & !COAX3(A)
Conditions met:
VerAvailable
SW3
SW4
COAX3_2nd_DC
A
Default:
Terminated
State of requested path
Start checking with state:
HorAvailable
Next state to check
VerActive
Stop if path has been found!

Figure 4-82 Block diagram of the VI GuiSwitchStateLogic (GUI SWITCH STATES).

4.4.10.2 Colouring the transmit path

Colouring the transmit path is more simple than colouring the receive path because less conditions are to be met. Here the VI **GuiPathColouring** (see Figure 4-81) is used again. As already mentioned, the transmit path colouring depends on the state of the up-converter transmitter and the state of the HPA. If both are on, the transmit path is coloured green and if SW1 is in test translator position, the whole path to COAX1 appears green.

4.4.10.3 Colouring logic

Table 10 provides an overview over the logical conditions, e.g. the waveguide and coaxial switch positions that are required for the colouring of a certain signal path. The columns **Path** and **State** contain the same labels as they are used in the state machine of the VI GuiPathColouring. The column **Condition** shows the global variables on which the current state of a path depends (switch positions are noted in round brackets). The notation of the column **Condition** follows the typical conventions from modern programming languages [21] with ‘&&’ as a logical AND, ‘||’ as a logical OR and ‘!’ as logical negation.

Table 10 Overview of the logical conditions for colouring the signal paths.

Path	State	Condition	Colour
11GHz-A	Terminated	!SW2 && !SW3 && !SW4	Black
	HorAvailable	SW2 && !SW3 && !COAX3(A)	Dim yellow
	HorActive	SW2 && !SW3 && COAX3(A)	Yellow
	VerAvailable	SW3 && SW4 && !COAX3(A)	Dim green
	VerActive	SW3 && SW4 && COAX3(A)	Green
11GHz-B	Terminated	!SW3 && !SW3 && !SW4	Black
	HorAvailable	SW3 && SW3 && !COAX3(B)	Dim yellow
	HorActive	SW3 && SW3 && !COAX3(B)	Yellow
	VerAvailable	!SW3 && SW4 && !COAX3(B)	Dim green
	VerActive	!SW3 && SW4 && COAX3(B)	Green
12GHz-C	Terminated	!SW9	Black
	VerAvailable	SW9 && !COAX3(C)	Dim green
	VerActive	SW9 && COAX3(C)	Green
12GHz-D	Terminated	!SW7	Black
	HorAvailable	SW7 && !COAX3(D)	Dim yellow
	HorActive	SW7 && COAX3(D)	Yellow
uc carrier	Off	!transmit_carrier_on	Black
	On	transmit_carrier_on	Green
hpa status	Off	!hpa_status	Black
	On	hpa_status	Green
tt active	Active	hpa_status && transmit_carrier_on && SW1(3)	Green
	Inactive	In any other case	Black
tt path A	Active	COAX1(A) && SW1(3)	Green
	Inactive	In any other case	Black
tt path B	Active	COAX1(B) && SW1(3)	Green
	Inactive	In any other case	Black
tt path C	Active	COAX1(C) && SW1(3)	Green
	Inactive	In any other case	Black
tt path D	Active	COAX1(D) && SW1(3)	Green
	Inactive	In any other case	Black
2nd dc path	Active	!COAX3(Disconnect)	Green
	Inactive	In any other case	Black

5 Hardware

An important feature and also requirement of the new LEOSv2 software is its compatibility with the existing station hardware. Many of the station's hardware components can be connected directly to either a serial interface, an IEEE interface or to a Digital-I/O interface. Those components that require any additional arrangements are connected to the LEOS box, providing any intermediate steps necessary to control them via LabVIEW and the interface cards. The waveguide switches, the coaxial switches and the high power amplifier are connected to the LEOS box and their requirements are explained within this chapter.

5.1 LEOS Box

To understand the functionality of the physical high frequency path and its underlying components, the existing LEOS hardware box had to be studied and adapted to the new requirements. Firstly, the LEOS box inner circuitry required new documentation, because plans and layouts were only partially available. Secondly, the connector cables between control PC and LEOS box had to be redesigned to fit to the new connector layout. Thirdly, a breadboard was built to provide a 5V supply to the waveguide- and coaxial switch indicator pins over pull-up resistors.

Figure 5-1 shows a schematic of the internal circuitry of the LEOS box, as it is currently used. For easy orientation, all the Sub-D connectors are arranged the same way, as they are found on the backside of the LEOS box. Starting from the bottom, there are four Sub-D connectors labelled as **C1, C2, C3** and **C4**, all of which have 25 pins. Those are the connectors that lead to the National Instruments Digital-I/O interface card and are either dedicated outputs (Drives: C1 and C3) or inputs (Indicators: C2 and C4). The four connectors were originally connected to four ports of the old control PCs Digital-I/O interface card. However, the new setup is based on two 37-pin (32 channel) Digital-I/O interface cards. For this purpose, new connector cables were built, to remap the connectors C1 to C4 to two 37-pin connectors. Schematics of the new connector cables can be found in Figure 5-2 and Figure 5-3. Note that only 20 of the 25 available pins of C1 to C4 are actually connected inside the LEOS box and not all of these 20 pins are used, therefore two 32 channel Digital-I/O interface cards provide enough ports.

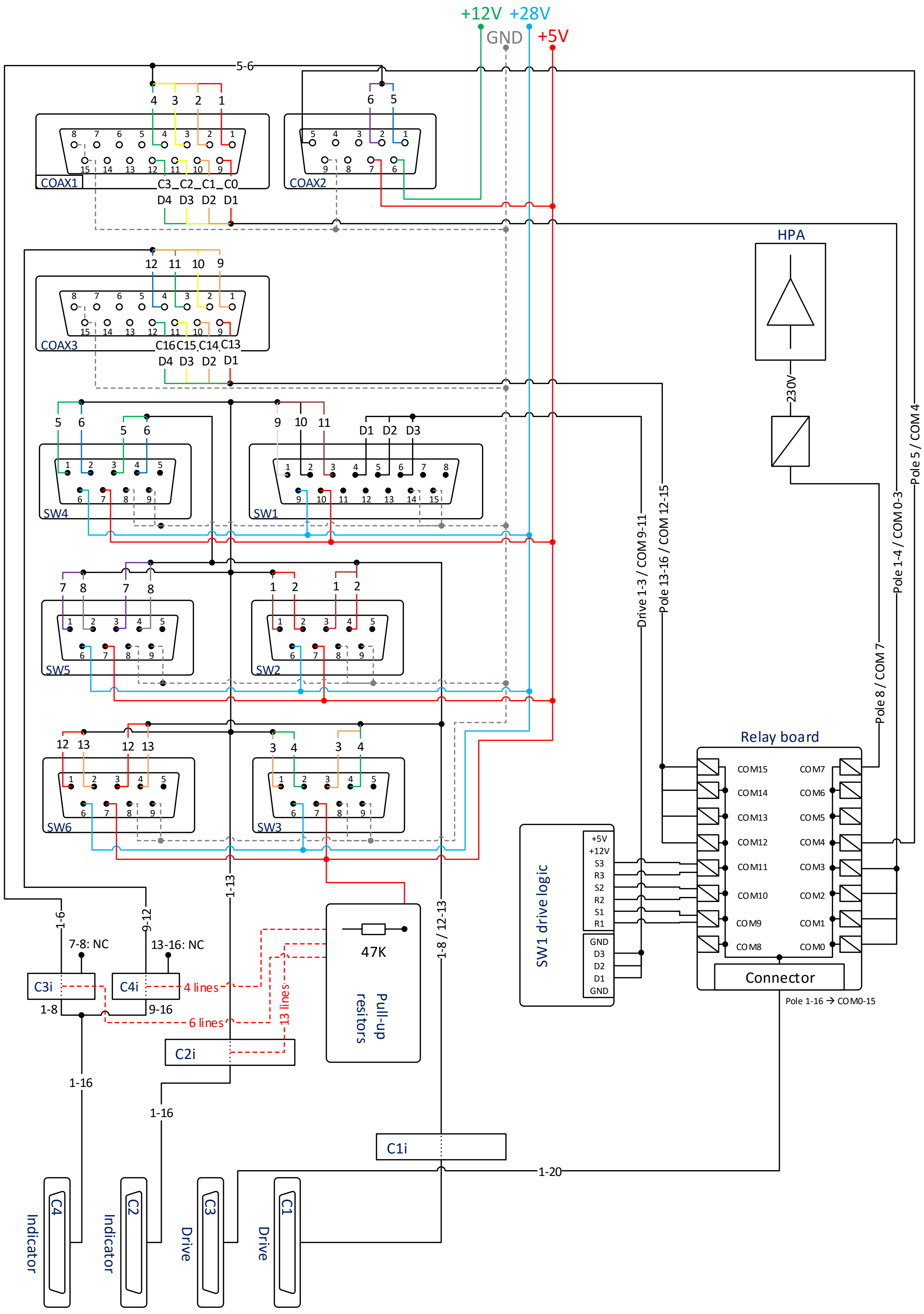


Figure 5-1 Schematic of the LEOS box internal circuitry and connectors.

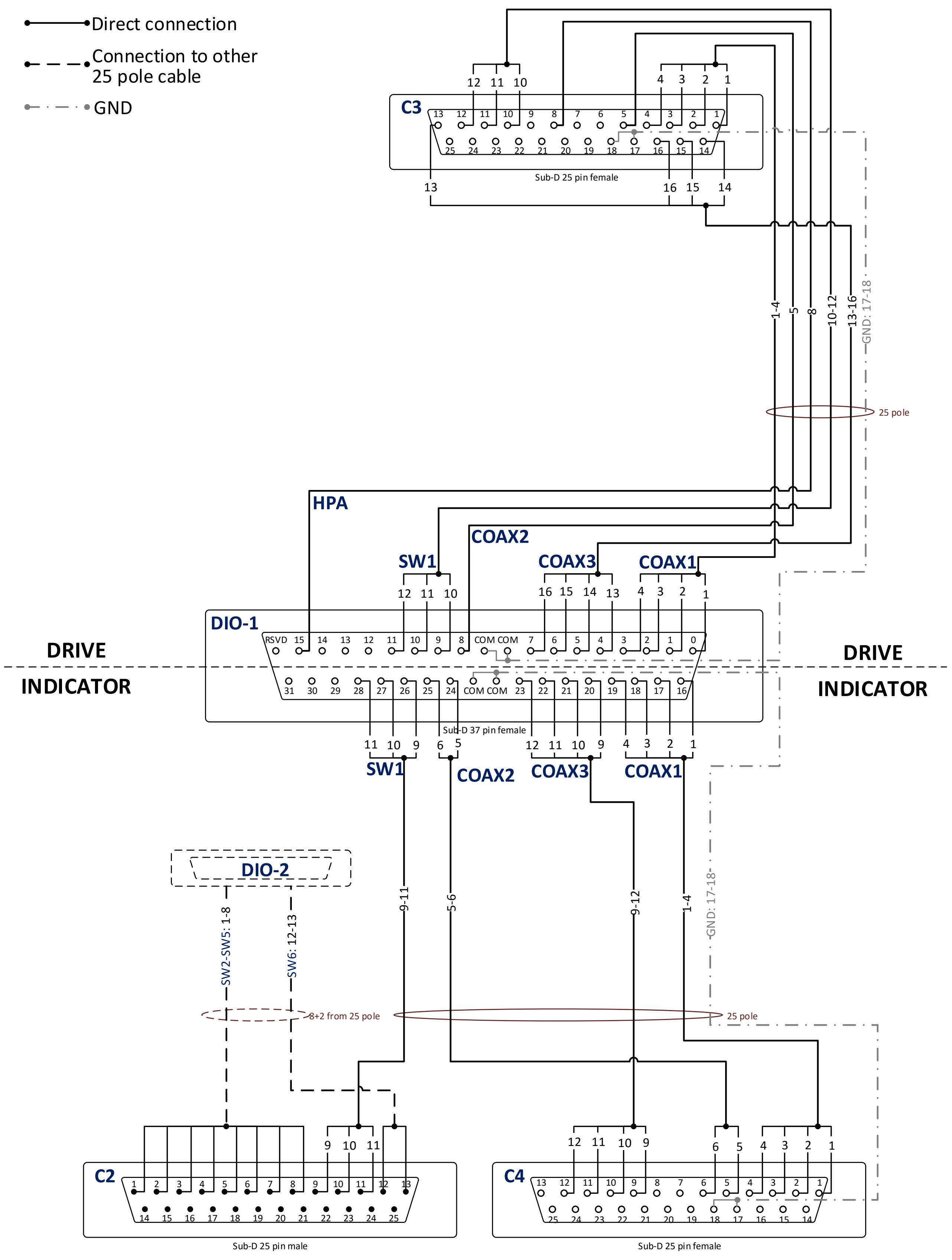


Figure 5-2 Wiring of the first connector cable: LEOS box to Digital-I/O interface card.

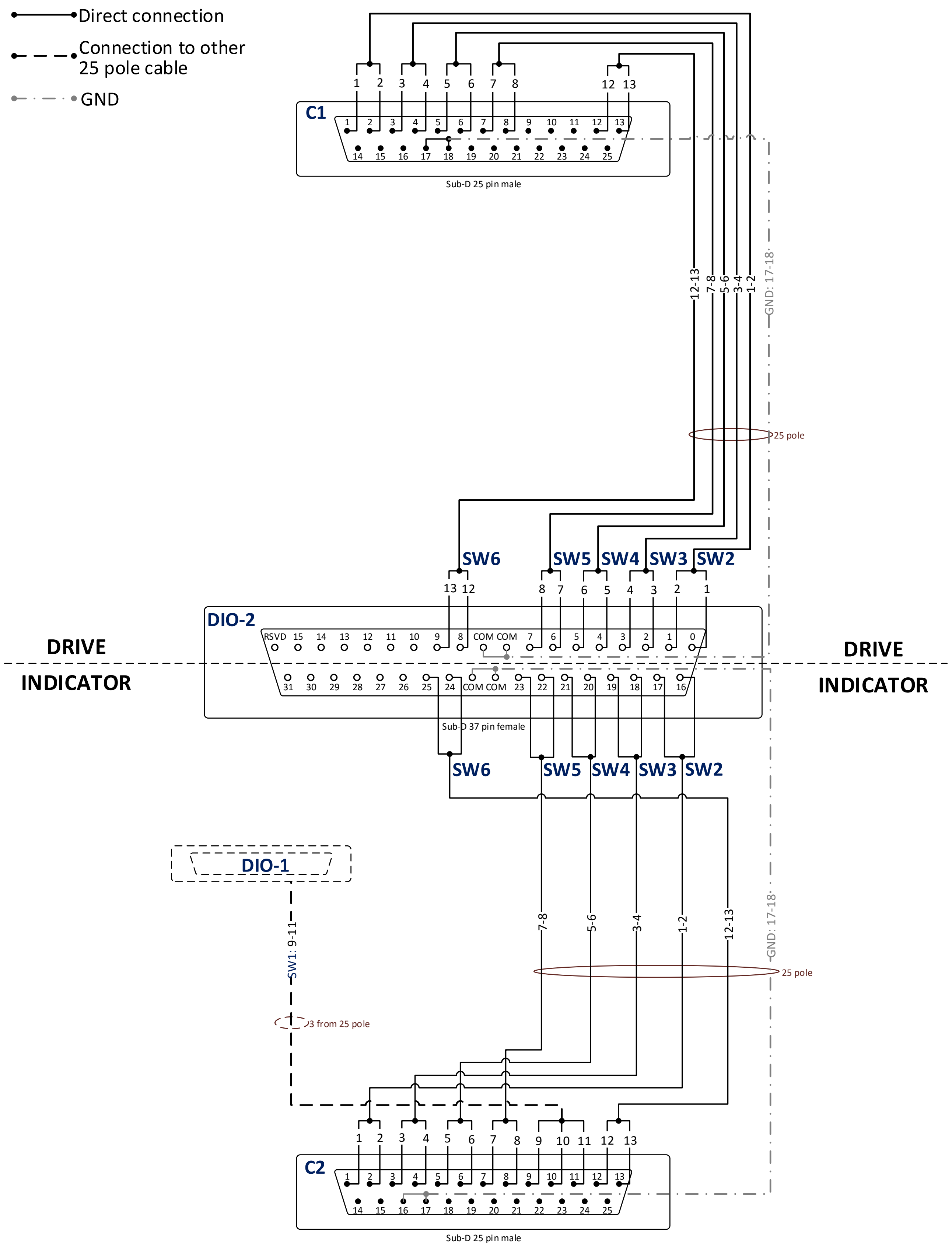


Figure 5-3 Wiring of the second connector cable: LEOS box to Digital-I/O interface card.

5.1.1 Digital-IO pull-up board

The indicators for the waveguide switches and the coaxial switches are open connectors by default. Only if an indicator is active, it is pulled to GND. The old Digital-I/O interface card PCL-720 had included internal pull-ups, but the National Instruments Digital-I/O board requires external pull-up resistors. For this purpose, a simple breadboard was built, using 47 kΩ pull-up resistors connected to the 5V power supply. The indicator ports of the waveguide switches SW1-SW6 and the coaxial switches COAX1-COAX3 are now provided with 5V, using the pull-up board. The board is connected to the LEOS box internal connectors C2i, C3i and C4i, as depicted in the LEOS box schematic layout in Figure 5-1. Figure 5-4 shows the pull-up board schematic layout:

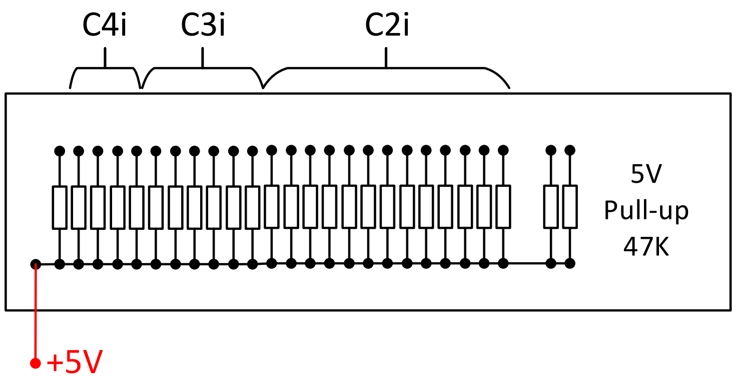


Figure 5-4 Pull-up board inside the LEOS box, providing 5V for the waveguide switch and coaxial switch indicators.

5.2 Digital-I/O waveguide switches

The Digital-I/O waveguide switches SW1 – SW6 are connected to the LEOS box which provides power supply and pull-up resistors for the switches indicator ports and routes the switches drive and indicator ports to the National Instruments Digital-I/O – Ethernet converters.The drive ports of the 3 position switch SW1 should only be active one at a time. If two or more of the drive ports are active simultaneously, the switch could be destroyed. To avoid that, an electronics board inside the LEOS box makes sure that only one drive port can be active at a time. Additionally, the drive ports of all Digital-I/O switches only require a short pulse of ~100 ms to change the switches' position.

The next two pages show a summary for the waveguide switches that was created with the help of the existing LEOS hardware documentation (see [22]) and was extended by some features. The summary contains the pin layouts, pin labels and colours of the wire that connects each switch to the LEOS box, including the pin layout of the Sub-D connectors at the LEOS box. For SW1 the possible switch positions are shown and below the positions you find the configuration of the wheels that are attached to the switch. These wheels can be used to manually change the position of the switch. The two position switches SW2 – SW9 can also be moved manually by turning the corresponding switch motor.

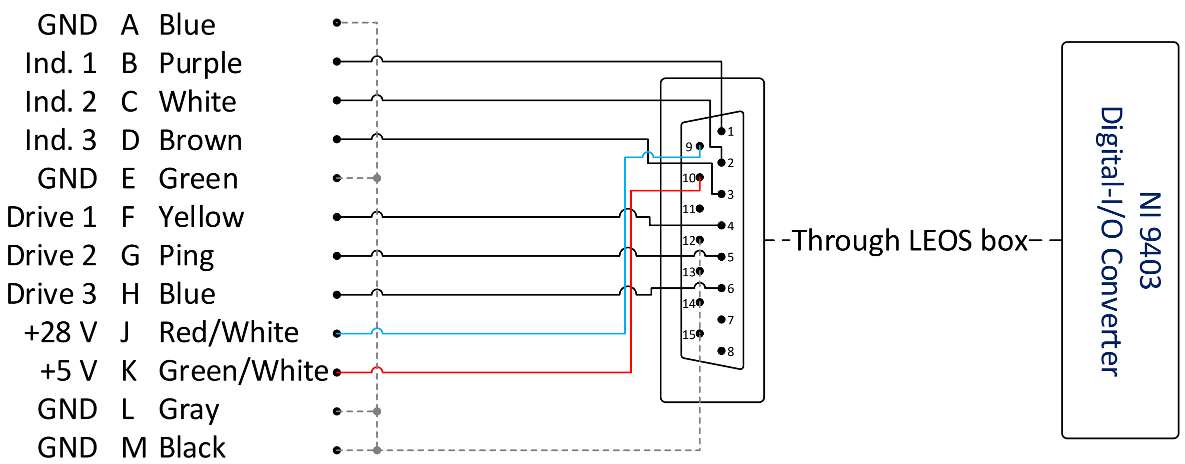
At the bottom of the summary a signal graph is shown, describing the digital signals of the drive ports and the corresponding indicator ports. All switches indicator ports are pulled to GND when the corresponding positon is active.

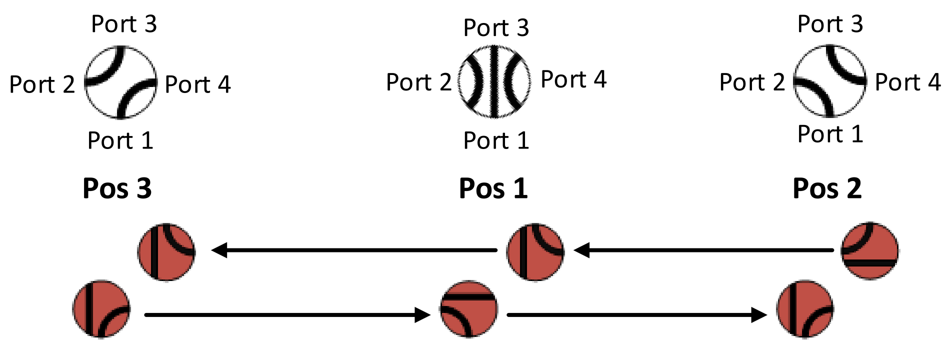
5.2.1 3-way switch

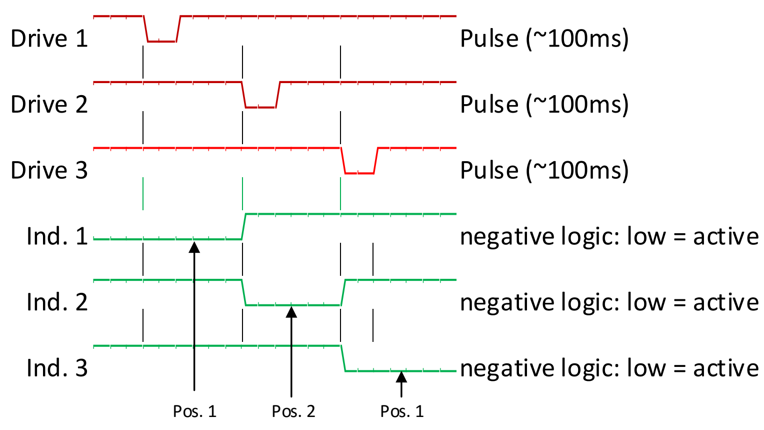
Waveguide switch
3 positions

Interface: Digital-I/O
Connected to: National Instruments Ethernet to Digital-I/O converter (NI9403)

Drives: ~100 ms pulse
Indicators: active low: pull-up required







Drives: active low
LEOS box inverts drives from Digital-I/O card, so Drives can be controlled with positive pulses.

LEOS box also allows only one active Drive port at a time to avoid damaging the switch.

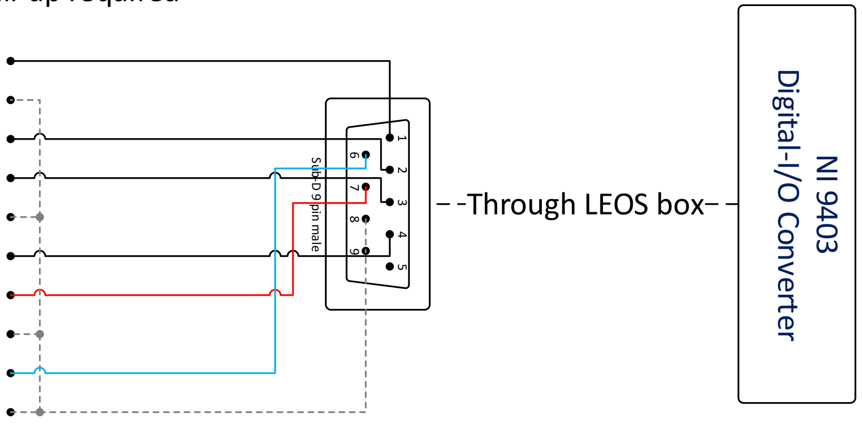
5.2.2 2-way switch

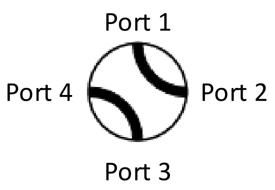
Waveguide switch
2 positions

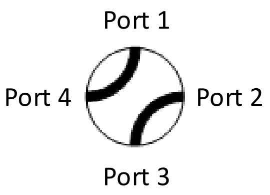
Interface: Digital-I/O
Connected to: National Instruments Ethernet to Digital-I/O converter (NI9403)

Drives: ~100 ms pulse
Indicators: active low: pull-up required

Ind. 1	A	Blue
Ind. Comm	B	Red
Ind. 2	C	White
Pos. 1 Drive	D	Brown
GND (5V)	E	Black
Pos. 2 Drive	F	Green
+5 V	G	Gray
GND(28V)	H	Ping
+28 V	J	Purple
GND	K	Yellow

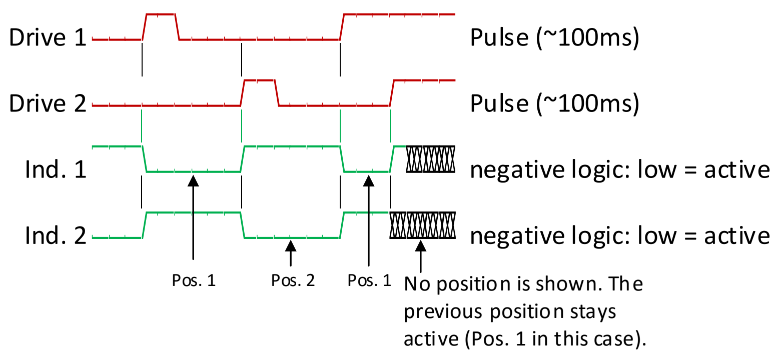






Pos 1: A & B connected
C open

Pos 1: B & C connected
A open


Drive 1
Pulse (~100ms)
Drive 2
Pulse (~100ms)
Ind. 1
negative logic: low = active
Ind. 2
negative logic: low = active
Pos. 1
Pos. 2
Pos. 1
No position is shown. The previous position stays active (Pos. 1 in this case).

5.3 Coaxial switches

The Digital-I/O coaxial switches are connected to the LEOS box which provides power supply and sufficient voltage for the switches' drive ports. The four position switches have and additional positons "disconnected", if neither of the four positons is active. To switch to one of the four positons a constant voltage of 28V is required at the corresponding drive port. The 28V are provided through a relay board inside the LEOS box that converts the Digital-I/O signal to 28V. The indicator ports of the 4 position switches are routed directly through the LEOS box to the Digital-I/O interfaces.

The two positions switch requires +5V and +12V as power supply, which is also provided by the LEOS box. For changing its switch position a TTL signal is required on the drive port. This TTL signal is also provided by the relay board in the LEOS box. This switch has two indicator ports, but only indicator one is used for determining the switch position.

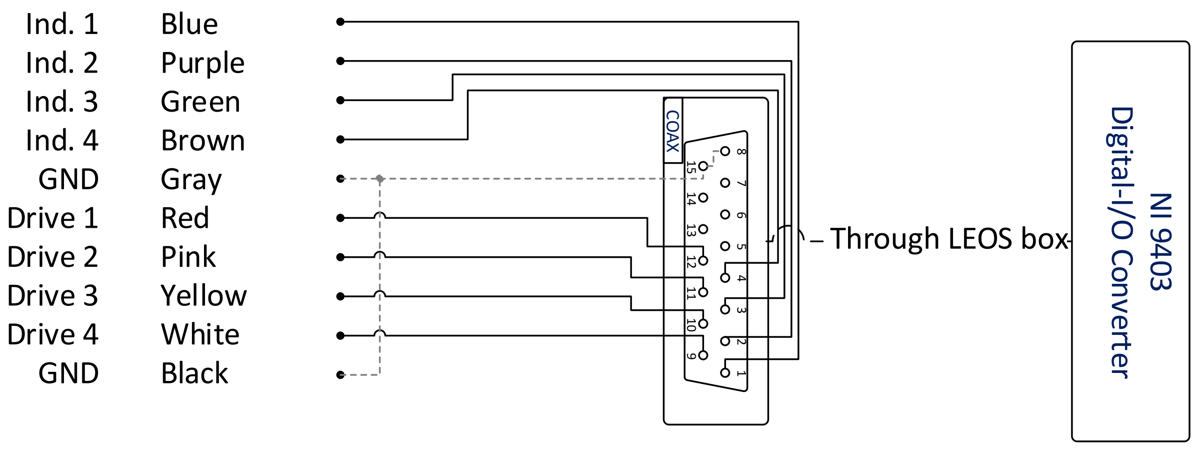
The next two pages show a summary for the coaxial switches that was created with the help of the existing LEOS hardware documentation (see [22]) and was extended by some features. The summary shows the switches' pin layouts, connector diagrams and signal graphs explaining the states of the indicator ports and the drive ports.

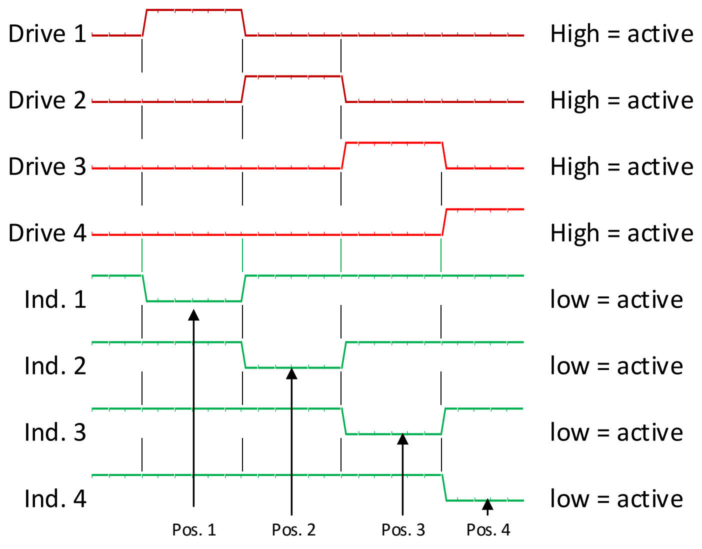
5.3.1 4-way switch

Coaxial switch
4 positions

Interface: Digital-I/O
Connected to: National Instruments Ethernet to Digital-I/O converter (NI9403)

Drives: Constant voltage, 28V provided by LEOS box relay board
Indicators: active low: pull-up required





Drives: active high
Switch requires permanent 28V to keep its positon.
28V is provided via the relay board in the LEOS box.

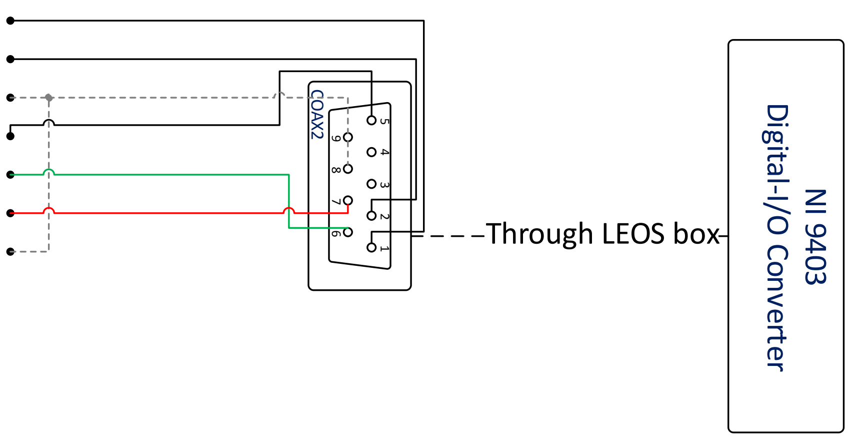
5.3.2 2-way switch

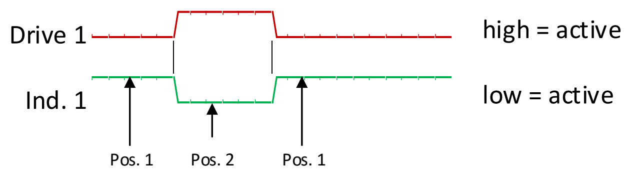
Coaxial switch
2 positions

Interface: Digital-I/O
Connected to: National Instruments Ethernet to Digital-I/O converter (NI9403)

Drives: Constant voltage, 28V provided by LEOS box relay board
Indicators: active low: pull-up required

Ind. 1	Green
Ind. 2	White
Ind. C (GND)	Gray
TTL (Drive)	Blue
+12V	Pink
+5V	Red
GND	Black





Drives: active high
Switch requires permanent 28V to keep its positon.
28V is provided via the relay board in the LEOS box.

6 Hardware setup and commands

This chapter sums up the hardware configuration for devices that are connected to a RS-232 interface and IEEE devices. The angle counters and the up-converter use a RS-232 interface and the interface setup is listed here. IEEE devices like the antenna controller and the waveguide switches SW7, SW8 and SW9 don't require any special setup, only an IEEE address in some cases. This chapter lists the command formats and response formats for each device and shows some examples.

6.1 Angle counters

Serial settings:

Baud: 9600

Data bits: 7

Parity: even

Stop bits: 2

Flow control: none

The angle counters follow the principle of command and response, meaning after each command the respond with the angle readings. To read the angles, send the command **Start Of Text (STX)**. STX has the corresponding ASCII value of 0x02. To send a hex number in LabVIEW use a text field with hexadecimal representation.

The response contains 17 bytes:

Azimuth: +\s\s179.528.6\s\s\s\r\n

Elevation: +\s\s\s\s4.997.5\s\s\s\r\n

Note that the actual angle in the response varies in length from 7-9 bytes (see example above) but the total response always is filled with leading spaces to a total length of 17 bytes.

6.2 Up-converter

Serial settings:

Baud: 9600

Data bits: 7

Parity: even

Stop bits: 2

Flow control: Xon/Xoff

The command format for the up-converter is as follows:

<addr/CMD_[value]<cr>

A command always starts with an "**<**" followed by the integer address "**1**" and a slash. The command **CMD** follows after the slash and the **value** is separated by an underscore from the command. Each command is terminated with a carriage return. The up-converter always responds to a command with the setting associate to that command. Also an address is required for the up-converter. The response form the up-converter has a similar format, starting with an "**>**":

>addr/CMD[_value]<cr>

Some commands trigger responses with a list of parameters in the following format.

>addr/CMD1_value<cr>

CMD2_value<cr>

.

.

CMDn_value<cr>

Example 1:

Command and response for switching the transmit carrier on:

```
Command: <1/TC_1\r\n
Response: >1/TC_1\r\n
```

Example 2:

Requesting the bulk consolidated status (BCS) returns multiple commands:

```
Command: <1/BCS_\r\n
Response: >1/BCS_\r
          TXF_14250\r
          TC_0\r
          TP_13.0\r
          POUT_7.0\r
          TMP_65\r
          TCBT_1\
          rRXF_12646\r
          RA_33.0\r
          LNB_4\r
          RCBT_1\r
          FTR_0\r
          CBL_50\r
          ADD1_1\
          rADD2_1\r
          BD1_9600\r
          BD2_2400\r
          CCM_1\r
          ISV_1.05\r
          OSV_1.04\r\n
```

Table 11 shows all commands for the up-converter that have been directly implemented in the LEOSv2 GUI. For a more detailed description and a list of all available commands please see the up-converter handbook [20], chapter 6.

Table 11 Implemented commands for the up-converter.

Command	Response	Description
<addr/TC_n<cr>	>addr/TC_n	Switch the transmit carrier (TC) on or off
<addr/TP_nn.n<cr>	>addr/TP_nn.n	Set the transmit power (TP)
<addr/TXF_nnnnnn<cr>	>addr/TXF_nnnnnn	Set transmit frequency (TXF)
<addr/RXF_nnnnnn<cr>	>addr/RXF_nnnnnn	Set receiveer frequency (RXF)
<addr/BCS_<cr>	>addr/BCS_ ..and a list of commands	Request bulk consolidated status (BCS) The BCS contains the complete configuration of the up-converter
<addr/FS_<cr>	>addr/FS_ ..and a list of commands	Request the fault status (FS). The FS contains a list of all faults since the last time all faults have been cleared
<add/CSF_<cr>	>addr/CSF_	Clears stored faults (CSF)

6.3 Antenna controller

The Phytron IXE-alpha antenna controller doesn't require any spcial setup or IEEE address. Commands are for controlling or reading data from the antenna controller are sent directly in a string format. The command format is

"cmd"

The response is:

STX ACK data ETX CR LF EOI

The characters surrounding the **data** can be found in an ASCII table and are summed up in Table 12:

Table 12 Antenna control response format control characters.

Short text	Long text	ASCII value
STX	Start Of Text	002
ETX	End Of Text	003
ACK	Acknowledge	006
LF	Line Feed	010
CR	Carriage Return	013

EOI stands for End Or Identify and is an IEEE specific command, used to indicate the last byte of the message [22].

Example 1:

Command for reading the azimuth:

R3R

Response from the antenna controller:

STX ACK **180.000** ETX CR LF EOI

➔ The current azimuth is 180.000°.

Example 2:

Command for checking, if the antenna is moving in azimuthal direction:

R101R

Response:

STX ACK **1** ETX CR LF EOI

➔ The **1** indicates that the antenna is moving in azimuthal direction.

Table 13 lists all commands that are implemented in the LEOSv2 software. Some of the commands are for control only, some trigger a response from the antenna controller.

Table 13 Commands and responses for the Antenna controller

Command	Response (data)	Description
Read angles		
R3R	7 byte floating point number: nnn.nnn	Read the azimuth
R4R	7 byte floating point number: nnn.nnn	Read the elevation
General commands		
R100S0	-	Disable the IXE keyboard
R98S0	-	Reset and prepare for a new command
Calibration		
R3S	-	Set a new azimuth
R4S	-	Set a new elevation
R99S5	-	Apply the new azimuth and elevation
Move the antenna		
R1S	-	Set a new azimuth
R2S	-	Set a new elevation
R99S4	-	Move the antenna to the new azimuth and elevation
Antenna controller status bits		
R101R	0 ..halt / 1 ..moving	Motor azimuth: halt / moving
R102R	0 ..halt / 1 ..moving	Motor elevation: halt / moving
R103R	0 ..blocked / 1 ..free	Break azimuth: blocked / free
R104R	0 ..blocked / 1 ..free	Break elevation: blocked / free
R114R	0 ..no halt / 1 ..halt	External stop: halt / no halt
R115R	0 ..no error / 1 ..error	Hardware: error / no error
R116R	0 ..not reached / 1 ..reached	Calibration point: not reached / reached
R117R	0 ..valid / 1 ..not valid	Calibration: valid / not valid
R105R	0 ..off / 1 ..on	End switch azimuth: off / on
R106R	0 ..off / 1 ..on	End switch elevation: off / on
R107R	0 ..off / 1 ..on	End switch zenith: off / on
R108R	0 ..off / 1 ..on	End switch elevation horizontal: off / on
R110R	0 ..off / 1 ..on	Initiator azimuth west: off / on
R111R	0 ..off / 1 ..on	Initiator azimuth east: off / on
R112R	0 ..off / 1 ..on	Initiator elevation zenith: off / on
R113R	0 ..off / 1 ..on	Initiator elevation horizontal: off / on

6.4 IEEE waveguide switches

There three IEEE waveguide switches, SW7, SW8 and SW9. SW7 and SW8 are listed as a single IEEE device within LabVIEW and are distinguished by their corresponding IEEE address. The address format for these switches is a simple character, “A” or “B”. Table 14 shows the switch number, the associated addresses. The column “Positions string” shows the string that needs to be sent to the switch to change its positon. The same string also indicates the current position when the switch is read only. The LabVIEW IEEE resource is the name of the name of the VISA resource that is associated to the IEEE switches in LabVIEW.

Switch	Address	Position string (set / read)	Position of switch	LabVIEW IEEE resource
7	B	1	Position 1	Waveguide-SW7B8A
		3	Position 2	
8	A	1	Position 1	Waveguide-SW7B8A
		3	Position 2	
9	A	1	Position 1	Waveguide-SW9
		3	Position 2	

Table 14 Commands, addresses and LabVIEW resources for 2-way IEEE waveguide switches.

The positions of a switch can be changed by sending the following command to the switch:

<Address><Position string>

Example:

To set SW7 to position 1, send the string “**B1**” to the corresponding IEEE interface.

To set SW8 to position 1, set the string “**B2**”.

Since SW7 and SW8 are on the same IEEE resource, performing a read operation on that resource always returns the position of both switches in the format of two numbers separated by a comma:

<number1>,<number2>

Number 1 is associated to the position of SW8 and number 2 is associated to SW7

Example:

1,1 ..SW8 position 1, SW7 position 1

1,3 ..SW8 position 1, SW7 position 2

3,1 ..SW8 position 2, SW7 position 1

Setting the positon of SW9 follows the same scheme and the response here only contains one number, either **1** (position 1) or **3** (position 2).

7 Summary

With the documentation of the hardware and software that is used to control the satellite station Graz Lustbühel future operators are provided with a useful tool to quickly understand the station, its components, the software and how all these things are related. This theses contains all the necessary information that was required to develop the new operation and control software, LEOSv2, that replaces the previous control software. With LEOSv2 it was made possible to modernize the station and replace the old and obsolete control PCs by state-of-the-art computers. By decoupling the new control PC from all hardware interfaces it was also made possible to decentralise the RS-232, IEEE and Digital-I/O interface cards that provide an interface to the station's hardware components. This leads to the advantage that all existing hardware components could be kept and thus a lot of cost for new hardware could be avoided.

The LEOSv2 user interface provides a number of new features: an extensive initialization routine, informing the user right at the start of the software about possible problems with the station's hardware, a familiar overview of the station that has been extended by some useful information to see all relevant parameters at a glance, a grouping of functions dedicated to a certain hardware component and full implementation of the new up-converter. The user interface will also help people to quickly understand and operate the satellite station.

Since the existing documentation of the station is very old and therefore only available in printed or hand drawn form, parts of the existing documentation were reproduced in digital form and added to this theses. Missing parts of the documentation were restored within the course of this thesis. This includes a complete schematic of the LEOS control box and the pin layouts of the hardware components that are connected to the LEOS control box. Additionally, schematics of the whole station including all hardware components and their interconnection have been created and will prove very useful for people that operate the station in the future.

For future operations the new Ethernet-based configuration of the Lustbühel satellite station, consisting of the new control PC, the new hardware interface cards and the new LEOSv2 software, provides a flexible platform on which further applications and enhancements of the station can be based upon. This could include remote operation of the satellite station, thus eliminating the need to travel to the station in person. Any additional hardware that might be required for remote operation, for example a webcam to monitor the movement of the antenna, can now easily be added to the existing Ethernet network and can quickly be included to LEOSv2. The software can be extended to provide control for additional hardware components, and software enhancements like a satellite tracking algorithm can be implemented and visualized within the LEOS GUI.

8 References

[1] H. Haupt, 1976. [Online]. Available: http://physik.uni-graz.at/de/igam/forschen/mess-stationen/observatorium-lustbuehel/geschichte-des-observatoriums/. [Accessed 3 April 2016].

[2] R. Temmel, *Documentation folder "LEOS SW" at the Lustbühel satellite station..*

[3] H.-P. Halvorsen, “Introduction to LabVIEW,” March 2014. [Online]. Available: http://home.hit.no/~hansha/documents/labview/training/Introduction%20to%20LabVIEW/Introduction%20to%20LabVIEW.pdf. [Accessed 7 April 2016].

[4] “LabVIEW 2012 help,” Natinal Instruments, June 2012. [Online]. Available: http://zone.ni.com/reference/en-XX/help/371361J-01/. [Accessed 7 April 2016].

[5] “LabVIEW Wiki,” Wikipedia, March 2016. [Online]. Available: https://en.wikipedia.org/wiki/LabVIEW. [Accessed 2 April 2016].

[6] “PWCT Features - Visual Programming,” Programming Without Coding Technology, [Online]. Available: http://doublesvsoop.sourceforge.net/pwcthelp/features/visualprogramming.htm. [Accessed 3 April 2016].

[7] G. M. a. M. Bousquet, Satellite Communications Systems: Systems, Techniques and Technology, John Wiley & Sons; Auflage: 5, 2009.

[8] “VNC Frequently Asked Questions,” University of Cambridge, [Online]. Available: http://www.cl.cam.ac.uk/research/dtg/attarchive/vnc/faq.html. [Accessed 4 April 2016].

[9] U.-P. D.-I. D. O. Koudelka, *Personal conversation regarding the antenna efficiency of the satellite station graz Lustbühel,* 2016.

[10] Phytron, Manual für Schrittmotorsteuerung IXE alpha-C.

[11] H. Packard, HP 437B Power Meter Operating Manual.

[12] Heidenhain, Arbeiten mit der Meßwertanzeige ND281.

[13] H. Lobensommer, Handbuch der modernen Funktechnik. Prinzipien, Technik, Systeme und praktische Anwendungen., Franzis Verlag GmbH, 1999.

[14] “NPort 5410/5430/5450,” Moxa, [Online]. Available: http://de.moxa.com/product/NPort_5410.htm. [Accessed 7 April 2016].

[15] “NI GPIB-ENET/1000,” National Instruments, [Online]. Available: http://sine.ni.com/nips/cds/view/p/lang/de/nid/209211. [Accessed 03 April 2016].

[16] National Instruments, Messsystem NI 9403 . [Online]. Available: http://sine.ni.com/nips/cds/view/p/lang/de/nid/209906. [Accessed 03 April 2016].

[17] “Systemdesingsoftware LabVIEW,” National Instruments, [Online]. Available: http://www.ni.com/labview/d/. [Accessed 6 April 2016].

[18] “LabVIEW QuickStart Guide,” May 1997. [Online]. Available: http://www.ni.com/pdf/manuals/321527a.pdf. [Accessed 8 April 2016].

[19] D. R. Wright, “NC State University,” [Online]. Available: http://www4.ncsu.edu/~drwrigh3/docs/courses/csc216/fsm-notes.pdf. [Accessed 07 April 2016].

[20] “Tutorial: State Machines,” National Instruments, October 2015. [Online]. Available: http://www.ni.com/tutorial/7595/en/. [Accessed 07 April 2016].

[21] SierraCom, 3100 Series VSAT Ku-Band Operator's Manual, 1998.

[22] R. Temmel, LEOS hardware documentation folder "LEOS HW"., Satellite station Graz Lustbühel.

[23] “Logischer Operator,” Wikipedia, [Online]. Available: https://de.wikipedia.org/wiki/Logischer_Operator. [Accessed 7 April 2016].

[24] N. Instruments, National Instruments, [Online]. Available: http://www.hit.bme.hu/~papay/edu/GPIB/tutor.htm. [Accessed 5 April 2016].

Appendix A - List of figures