
Digital Antenna Control System for the Ground Station Raisting 1

Philipp Berthold

Bachelor Thesis



Institute for Communications and Navigation

Prof. Dr. Christoph Günther

Supervised by

Dipl.-Phys. Sebastian Knogl

Munich, June 2012

Abstract

In the course of the involvement in the European Student Moon Orbiter mission offered by the European Space Agency, the Student Satellite Initiative Munich has undertaken the task of supplying an appropriate ground station. After some inquiries the satellite antenna *Raisting 1* located in Raisting near Weilheim, could be obtained for this purpose. The Student Satellite Initiative Munich has reconditioned the mechanical structure and control electronics of the antenna which already had been built in 1964. They additionally enhanced the Radio-Frequency sub-system in order to meet the mission requirements. For allowing an operation of the antenna at the planned mission control center located 30 km away in Munich a fully remotely controllable antenna control system had to be provided.

This thesis describes the development and implementation of a new digital antenna controller. It is called Antenna Control Interface. During the design the main objectives have been to realize a safe and reliable operation protecting the antenna's structure and existing analog control electronics. First task was to determine suitable interfaces to the existing motion electronics. Based on the interfaces a modular concept for the system has been designed and the single modules interacting with the motion electronics have been developed. The communication between the modules has been defined and a Main Processing Unit has been set up which manages all modules and provides all control algorithms. The Main Processing Unit can be accessed remotely by a client software developed for this purpose. This client software is intended to be run at the mission control center of the European Student Moon Orbiter mission.

The installed Antenna Control System has been tested and first remote control activities outside the original antenna control room could be performed. The system provides all necessary control modes like manual remote steering and NASA/NORAD Two Line Elements Format file tracking. As soon as the Antenna System Interface and Antenna Monitoring System, which control and monitor all peripheral subsystems of the antenna, are ready *Raisting 1* can be entirely remotely controlled.

Contents

1	Introduction	8
2	Original Antenna Electronics	10
2.1	Antenna steering range	10
2.2	Original Modes of Operation	12
2.3	Resolvers	13
2.4	Cam Disc Sensors	14
2.5	Internal Antenna Steering Signal	15
3	Concept of the Antenna Control Interface	19
3.1	Main Processing Unit	19
3.2	Interface modules	21
3.3	Bus	21
3.3.1	Power Supply	21
3.3.2	Controller Area Network	22
3.3.3	Hardware Protection System Signals	22
3.3.4	Individual Signals	22
3.4	Antenna Protection System	23
3.4.1	Level One: Main Processing Unit	23
3.4.2	Level Two: Module Software	23
3.4.3	Level Three: Module Hardware	24
3.5	System Interaction	24
4	Interface Modules	26
4.1	Common Setup	26
4.1.1	Task	26
4.1.2	Hardware	26
4.1.2.1	Galvanic isolation	26
4.1.2.2	Fuse	27
4.1.2.3	On-board Power Supplies	27
4.1.2.4	Uninterruptible Power Supply	28
4.1.2.5	Hardware Protection System	28
4.1.2.6	Hardware Protection System – status information lights .	31
4.1.2.7	Microcontrolling Unit	31
4.1.2.8	CAN node	32
4.1.2.9	Connectivity	32

4.1.3	Software	32
4.1.3.1	Timer	33
4.1.3.2	Runlevel	33
4.1.3.3	Strategy	34
4.1.3.4	Software Protection System	35
4.1.3.5	CAN	36
4.2	Interface Specific Setup	40
4.2.1	Resolver Input Module	40
4.2.1.1	Task	40
4.2.1.2	Hardware	44
4.2.1.3	Software	44
4.2.1.4	Tests and Performance	46
4.2.2	Cam Disc Input Module	47
4.2.2.1	Task	47
4.2.2.2	Hardware	47
4.2.2.3	Software	47
4.2.2.4	Tests and Performance	48
4.2.3	Velocity Output Module	48
4.2.3.1	Task	48
4.2.3.2	Hardware	49
4.2.3.3	Software	51
4.2.3.4	Tests and Performance	52
5	Supporting Hardware	54
5.1	Hub	54
5.1.1	Task	54
5.1.2	Hardware	54
5.2	Bus adapter for the Main Processing Unit	54
5.2.1	Task	54
5.2.2	Hardware	55
5.2.3	Software	55
5.3	Bus terminator	56
5.3.1	Hardware	57
5.4	Programmer	57
5.4.1	Software	57
5.5	Debugger	57
5.5.1	Hardware	58
5.5.2	Software	58
6	Main Processing Unit	60
6.1	CAN Interface	61
6.2	Intelligence	61
6.3	Kinetics	61
6.3.1	Position	62

6.3.2	Velocity	66
6.3.3	Acceleration	66
6.4	Control Loop	66
6.4.1	Selectable Input Sources	67
6.4.1.1	Mass	69
6.4.1.2	Goto	69
6.4.1.3	Velocity	69
6.4.1.4	Trackmem	69
6.4.1.5	TLE	70
6.4.1.6	Autotracking	70
6.4.1.7	Matlab Interface	70
6.4.1.8	Extern	70
6.4.1.9	Calibration	70
6.4.1.10	Direct	71
6.4.2	Multiplexer	71
6.4.3	Trajectory Generator	71
6.4.4	Controllers	72
6.4.4.1	Position controller	72
6.4.4.2	Tracking controller	72
6.4.4.3	Velocity controller	73
6.4.5	Signal conversion	73
6.4.6	Omega Filter	73
6.4.7	Modules	73
6.5	Network Interface	73
6.6	Logging Functionalities	76
7	Client Software	77
7.1	Concept	77
7.2	Functionalities	79
8	Tests and Performance	80
8.1	Estimation of Positioning Errors	80
8.1.1	Resolver Signal Attenuation	80
8.1.2	Resolver nonlinearity	83
8.1.3	Processing latency	83
8.1.4	Mechanical accuracy	85
8.1.5	Overall positioning error	85
8.2	System workload	85
9	Conclusion	86
9.1	Trivia	86

1 Introduction

The European Student Moon Orbiter (ESMO) is an educational satellite mission offered by the European Space Agency (ESA). The mission is performed as a cooperation of several university teams all over Europe with each being responsible for a specific sub-system. The Student Satellite Initiative Munich (SSIMUC), a group of students interested in satellite communication at the Technische Universität München (TUM), has proposed to supply the ground segment. As ground station, the satellite antenna *Raisting 1* could be obtained. It is located at Raisting in the local district of Weilheim/Schongau 30 km south of Munich. The antenna was built in 1964 by order of the German Federal Post Office. It served as node for the communication link between America and Europe. The mechanical structure has been constructed by M.A.N. and weights 300 tons (see also Figure 1.2). Its reflector has a diameter of 25 m and is designed as a Cassegrain sub-reflector system. The electronic control system built by Siemens is completely based on analog technology. A particularity of the antenna facility is its protection by a pressurized Radar Dome (radome) which can be seen in Figure 1.1. Due to this cover, the antenna stays well preserved until now although it was set out of operation already in 1985. Due to its importance as a symbol of the early ages of satellite communications the facility was declared as a landmark in 1999.



Figure 1.1: View on the radome of *Raisting 1*.

In the course of the adaption of the antenna for the ESMO mission, the main tasks of the SSIMUC are to maintain the antenna's functionality and enhance it with suitable digital Radio-Frequency (RF) and control systems.

The ESMO mission demands a fully remote operation of the antenna at the mission control center which will be located in Munich without on-site human supervision at the antenna facility. This requires an autonomous digital controller which has to offer a very high safety and robust operation. After the existing control electronics had been investigated and suitable interfaces for a digital control loop had been figured out, a prototype version of a simple digital controller had been developed. This had been the 1st version of the Antenna Control Interface (ACIv1) [1] and served as a proof-of-concept in order to confirm the applicability of the selected interfaces. Based on these results, during the work of this thesis an elaborated concept for the digital controller has been designed and finally realized by developing a self-contained, fully functional system¹.

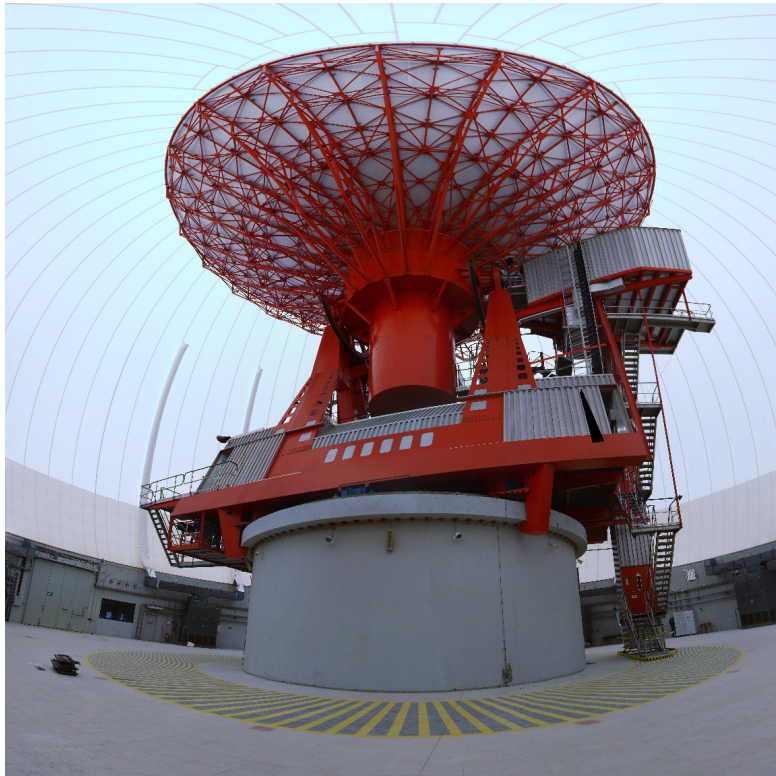


Figure 1.2: The antenna inside its radome.

¹This is the second version of the Antenna Control Interface (ACI) and therefore called ACIv2. In this thesis, this acronym is abbreviated to ACI due to clarity.

2 Original Antenna Electronics

This chapter outlines some basic information about the electronics of the original antenna control system. In the documentation [1] of the ACIv1 additional background information about the whole motion control system can be found.

The first step of the development of the ACI was the investigation of the existing motion control system to figure out practical electronic interfaces for attaching digital hardware. This investigation only based on the existing hardware would have been impossible. Only due to the availability of the original documentation, supplied by Siemens, and the noble support of the former chief-engineer of the antenna, Hans Lopau, the investigation has been successful.

In the course of this thesis, only the existing motion control system – called *Reglergestell* – has been examined (see Figure 2.1). This system consists of a rack and several plugged-in cards. Each card undertakes a specific task. Because the cards can easily be plugged in or out, it comes in handy to figure out cards with suitable signals for the interfaces of the ACI and thus replace these cards with electric identical function which are enhanced with all necessary ACI interface hardware. This approach is also in harmony with the monumental protection of the antenna: no permanent modifications must be done. Beside the *Reglergestell*, there is another system called *Verriegelungsgestell* (see also Figure 2.1). This system controls and monitors all peripheral functions of the antenna. It has to control the power supplies of the control electronics, the oil pumps and the elevation bolting, for example. For complete remote control of the antenna also the *Verriegelungsgestell* has to be enhanced with a digital interface (which consists of a monitoring system and an antenna system interface). The development of such a system is currently planned for upcoming winter term. As soon as this system is functional, all peripheral (such as electric supply controls) and drive systems – except of the RF parts – of the antenna will be fully remotely controllable via the Internet.

A graphical overview of the original control loop is given in Figure 2.8.

2.1 Antenna steering range

The antenna possesses two degrees of freedom, realized by the azimuth and elevation axis. The elevation axis can rotate about 115° and can be directed to the horizon (0°) and to the zenith (90°).

The azimuth axis rotates about 760° which allows tracking satellites independently from their starting point of visibility by choosing the right 360° space. The center of the slew range (0°) is oriented to the geographical east. This definition of the slew range is hereafter called the native coordinate system. Note that the position displays of the



Figure 2.1: The two motion control racks of *Raisting 1*. The left rack is the *Verriegelungs-gestell*, the right rack is the *Reglergestell*.

antenna are related to the north and only count from 0° (north) to 360° (north again). In this thesis all positions are referred to the native coordinate system. Table 2.1 gives a overview about the kinetic boundary conditions.

AXIS	POSITION	VELOCITY	ACCELERATION
Azimuth	$-380^\circ \dots +380^\circ$	$-1.5^\circ/\text{s} \dots +1.5^\circ/\text{s}$	$-0.5^\circ/\text{s}^2 \dots +0.5^\circ/\text{s}^2$
Elevation	$-1^\circ \dots +110^\circ$	$-1.5^\circ/\text{s} \dots +1.5^\circ/\text{s}$	$-0.5^\circ/\text{s}^2 \dots +0.5^\circ/\text{s}^2$

Table 2.1: Kinetic boundary conditions of *Raisting 1*

2.2 Original Modes of Operation

As soon as the *Verriegelungsgestell* has started all subsystems, the *Reglergestell* is able to move the antenna. As shown in Figure 2.2, the *Reglergestell* allows four different kinds of operation:



Figure 2.2: Mode selection panel of the *Reglergestell*. The *ENF* button is a German abbreviation for *Eigennachführung* and selects the auto-tracking functionality.

PR The Prozessrechner (PR) mode was used to control the antenna via an external signal source. This mode has to be selected (by the Antenna System Interface in future and by hand currently) when the ACI should control the movement of the antenna¹. The PR mode is only selectable if the *PR bereit* (external processor is ready) signal is received from the antenna motion system. As long as the PR mode is actually selected the *PR aktiv* (PR is active) signal will be emitted.

Hand The antenna is manually controllable by positioning wheels integrated in the *Reglergestell*. Small built-in engines can drive the positioning wheels and therefore the antenna. Hidden velocity potentiometers determine the speed of the engines.

Standby The default mode of operation: this control mode outputs no velocity and is engaged when the antenna boots up or an error has occurred (in this case, the velocity of the antenna will be slowed down and halted).

ENF The former antenna electronics also had an auto-tracking function, abbreviated by *ENF*. As soon as a satellite had been locked after a manual search, the auto-tracking function was able to follow this satellite.

¹However, the ACI does not use the original signal source but another analog interface. This is not accessible in one of the other three modes of operation, because the antenna's monitoring function denies a foreign signal there.

After the mode of operation has been selected, each axis of the antenna is controlled by an own controller. Both controllers are similar but use different control settings. They are always kept synchronized by the *Verriegelungsgestell*; that means if one axis detects an error, always both axis will be stopped. The systems behind the two axis are always symmetrical if not even unified. This principle has been continued in the Antenna Control Interface: as far as possible, the control systems are combined and axis specific subsystems are at least symmetrical.

2.3 Resolvers

To determine the position of the antenna, each axis is equipped with two resolvers. The first resolver – called coarse resolver – is directly mounted on the axis with conversion ratio 1:1 and returns the angle of the axis. The second resolver – called fine resolver – is connected with the axis with a rotation ratio of 72:1. That means that the coarse resolver is only used to determine in which 5°-section the axis is arranged. Hence, the final precision depends only on the accuracy of the fine resolver. Each resolver angle position is biased with a specific offset against the native coordinate system – this specific offset has to be measured employing a system calibration. A resolver consists of an inner coil which is mounted on the rotateable axis and two sensor coils which are attached perpendicular to each other surrounding the inner coil (see Figure 2.3).

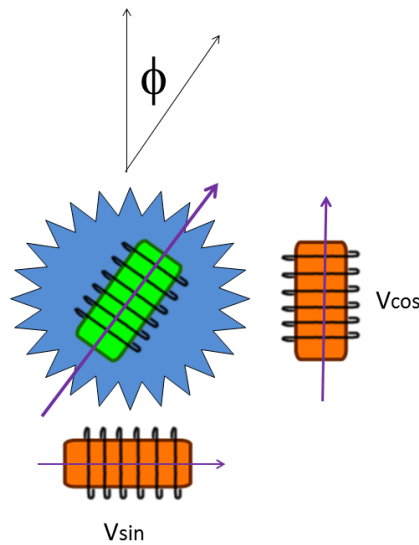


Figure 2.3: Theoretic assembly of a resolver.

The inner coil is supplied with a sinusoidal current which inducts, depending on the angle of the axis, voltages on the two outer coils. In *Raisting 1* a central 400 Hz sine generator is used as reference for all sinusoidal signals. It provides a voltage of

$$V_{\text{ref}}(t) = 36.8 \text{ V} \cdot \sin(2\pi \cdot 400 \text{ Hz} \cdot t), \quad (2.1)$$

which is also routed to the inner coils of all resolvers. The induction voltage on both outer coils can be described by

$$V_{\cos}(t) = c \cdot V_{\text{ref}}(t) \cdot \cos(\phi) \quad (2.2)$$

$$V_{\sin}(t) = c \cdot V_{\text{ref}}(t) \cdot \sin(\phi) \quad (2.3)$$

with c representing the induction attenuation. Hereby it is assumed that the attenuations on both coils are identical. High-quality resolvers hold this in a high degree, nevertheless in chapter 8.1.1 the case with different attenuations is analyzed.

With knowledge about $V_{\sin}(t)$ and $V_{\cos}(t)$ ², it is possible to determine the angle ϕ :

$$\frac{V_{\sin}(t)}{V_{\cos}(t)} = \frac{c \cdot V_{\text{ref}}(t) \cdot \sin(\phi)}{c \cdot V_{\text{ref}}(t) \cdot \cos(\phi)} = \frac{\sin(\phi)}{\cos(\phi)} = \tan(\phi) \quad (2.4)$$

$$\rightarrow \phi = \text{atan2}(V_{\sin}(t), V_{\cos}(t)) \quad (2.5)$$

This calculation is independent of $V_{\text{ref}}(t)$ and therefore $V_{\text{ref}}(t)$ has not to be detected.

$V_{\sin}(t)$ and $V_{\cos}(t)$ of the two resolvers are routed to the respective axis controller of the *Reglergestell* and first attenuated by approx. 77% and then processed. For the attenuation the Dämpfungsglied 1 (D1) is used. This card receives the signals of both resolvers, attenuates the signals by using simple but accurate resistive voltage dividers and delivers both attenuated signals as output signals. The D1 card is shown in Figure 2.4.

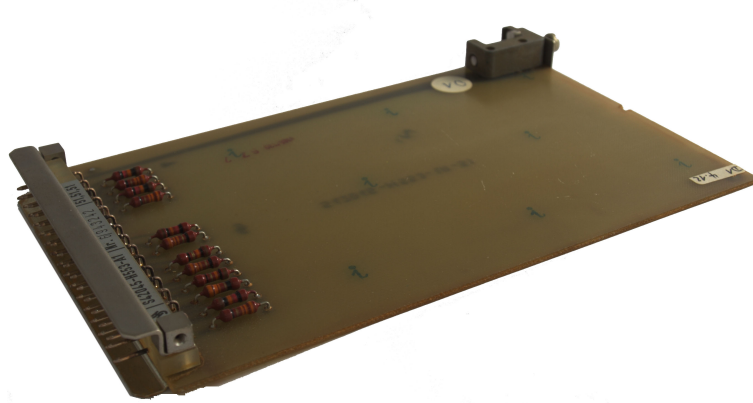


Figure 2.4: Original resolver attenuation card D1 with voltage dividers.

2.4 Cam Disc Sensors

The coarse resolvers capture a full axis rotation of 360°. This is sufficient for determining the position of the elevation axis. The azimuth axis position area, however, starts at

²The knowledge about the ratio $V_{\sin}(t)/V_{\cos}(t)$ alone is not sufficient: for four-quadrant angle inverse calculation the knowledge about the sign of each voltage is obligatory.

-380° and ends up with $+380^\circ$. The resolver output alone therefore is not sufficient to decide about the absolute position. Although a resolution of 360° is sufficient to focus a target on every direction, at least 720° are necessary to set the right starting point for tracking some trajectories without rewinding the antenna between.

The original analog control system only treated a region of rotation of 360° – the user had to determine and set the correct 360° space himself. The ACI, however, should be able to decide the 360° space automatically.

The antenna is provided with cam disc sensors to determine if the azimuth direction is near the end of the slewing range which can also be used to decide the correct 360° space. Due to its circuitry the only way to read out the signals from the cam disc sensor would be to cut off the connection between the cam disc sensor and the *Verriegelungsgestell* and create an interface. It would have to read out the cam disc and forward the switch states on the one hand to the *Verriegelungsgestell* and on the other hand to the ACI. Fortunately an identical copy of the cam disc sensor exists which is not used and not connected to another system. Therefore the ACI uses this sensor.

The cam disc sensor offers, amongst other, two essential switches: N4 and N5. The N4 switch is opened in the angle region from -380° to -178° and closed from -178° to $+380^\circ$. The N5 switch is opened in the angle region from $+178^\circ$ to $+380^\circ$ and closed from -380° to $+178^\circ$. Both switches split up the slewing range into three parts with each being smaller than 360° and consequently give full information about the 360° region the antenna is positioned in. Figure 2.5 outlines the information offered by the cam disc sensor, Figure 2.6 shows the cam disc sensor of the azimuth axis.

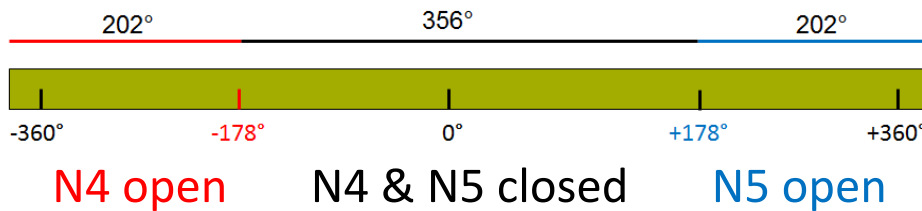


Figure 2.5: The switches N4 and N5 of the azimuth cam disc sensor divide the slewing range in three parts with each being fully ascertainable by the resolvers.

2.5 Internal Antenna Steering Signal

In order to setup the control loop, a suitable control variable had to be determined. In the consisting control electronics a nominal (angular) velocity signal is generated by the existing position controller and compared with the actual antenna velocity value by the Soll-Ist-Vergleich 3 (SIV3) card (shown in Figure 2.7). The functionality of the SIV3 card is outlined in Figure 2.8). The result of this comparison – the control error – is then forwarded to other subsystems limiting the signals and generating a current which

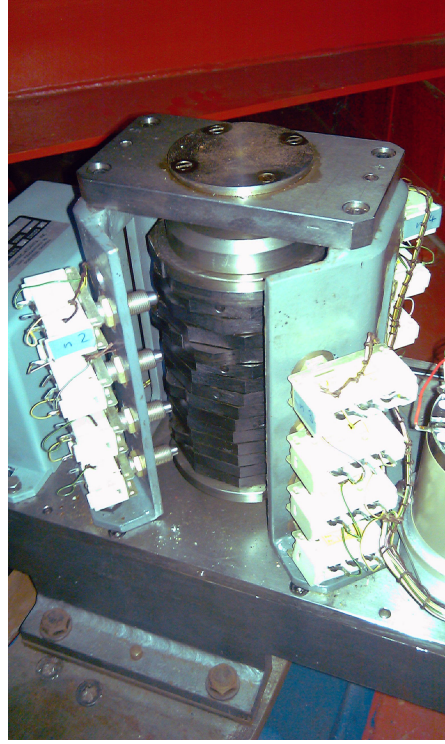


Figure 2.6: Opened azimuth cam disc switch.

finally controls the power electronics.

This nominal velocity signal can be used as a control variable for the ACI control loop. According to the circuit documentation the nominal velocity signal is a simple 400 Hz amplitude modulated voltage signal with a linear voltage - velocity conversion of 1 V corresponding to 1 °/s. Thus, the signal can be expressed as function of the desired angular velocity $\omega(t)$ by:

$$V_{\text{out}}(t) = \frac{\omega(t)}{1^\circ/\text{s}} \cdot \sin(2\pi \cdot 400 \text{ Hz} \cdot t) \cdot 1 \text{ V} \quad (2.6)$$

or, in a more circuit-related description:

$$V_{\text{out}}(t) = \frac{\omega(t)}{1^\circ/\text{s}} \cdot \frac{V_{\text{ref}}(t)}{36.8 \text{ V}} \cdot 1 \text{ V} \quad (2.7)$$

It should be mentioned here that at some point in the existing control system the further circuits demodulate the resulting signal (referencing to the reference signal $V_{\text{ref}}(t)$) and afterwards handle it as a direct current signal. An intervention in a later stage of the processing chain is more complicated and dangerous because all limiting systems and antenna-specific control filters would be bypassed.

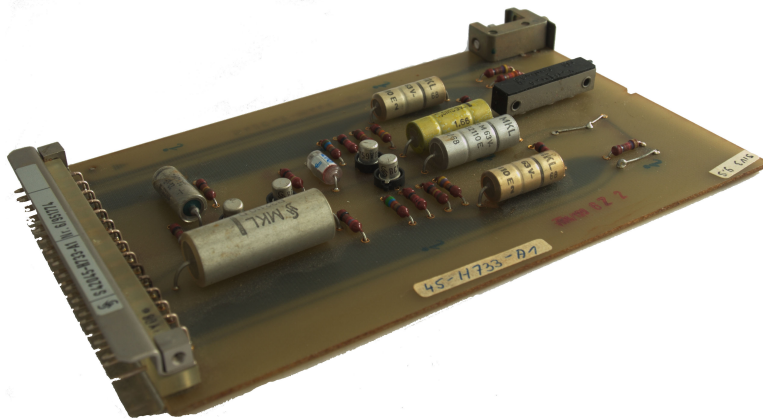


Figure 2.7: Original SIV3 card which served for calculation of the velocity error.

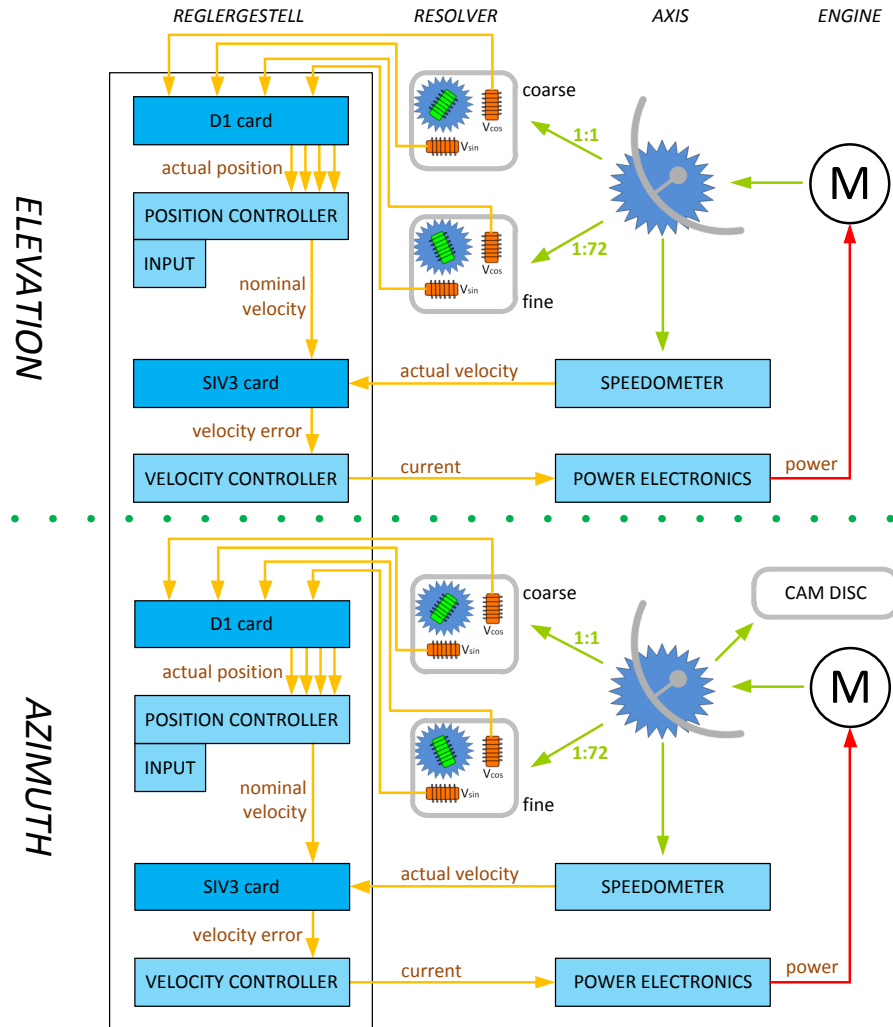


Figure 2.8: Simplified overview of the original antenna control loop. The resolvers are directly attached to the specific axis and measure the actual position. The position controller compare the actual position with the nominal position generated from the input block. The input blocks symbolize the different control input modes described in chapter 2.2. After the comparison, the position controllers output a nominal velocity. These velocities are compared with the actual velocity which is measured using the speedometers, independently from the resolvers. The velocity controllers finally output a specified current that controls the power electronics. The power electronics supply the engines which drive the antenna.

3 Concept of the Antenna Control Interface

This chapter introduces the basic features and ideas of the concept and also the requirements on the digital control system. The implementation is described in detail in the following chapters.

The specified signals can be tapped on different locations in the *Reglergestell* (or respectively on the mounted cards) or – regarding the cam disc sensor – next to the antenna’s axis. To take the mutual interference of the two axis into account and to guarantee an efficient control system, a high-performance Main Processing Unit (MPU) has been chosen which controls both axis at the same time. To connect the MPU to the tap points modularized interfaces are used because each interface needs its own hardware-related circuits and processing systems in order to handle time-critical real-time tasks and drive complex interfaces (for example, a 16bit Analog-Digital Converter (ADC) cannot be connected to the high-level MPU directly). This concept of one powerful MPU and different interfaces/modules connected by a data bus offers several considerable advantages. One advantage is that the interaction with the antenna gets split up – the modules handle the analog and time-critical parts of the problem and reduce it to an easier, conditioned (and digital) one which will be efficiently processed by the MPU. Another advantage is the expandability and flexibility which comes with a modularized concept: because all data interfaces are defined, each module can be replaced, modified or even redesigned separately without touching the rest of the system. Besides, a modularized system is easier to manage because each subsystem can be developed, tested and, if broken, replaced separately.

Figure 3.1 gives an overview about the adapted control loop.

3.1 Main Processing Unit

The MPU is the processing unit which contains the main logic and all control algorithms. It offers a network interface for several local, network or Internet clients which can connect to the MPU. There is no maximum number of read-only clients, that means every client can read out the status of the antenna (the only exception is the remote emergency stop of the antenna). However, only one single client is allowed to control the antenna. The MPU software contains a user right management system which allows an user with the appropriate rights to override another active user and decrease its privileges to read-only. This is useful if a remote client hangs up or tries to hijack the antenna control.

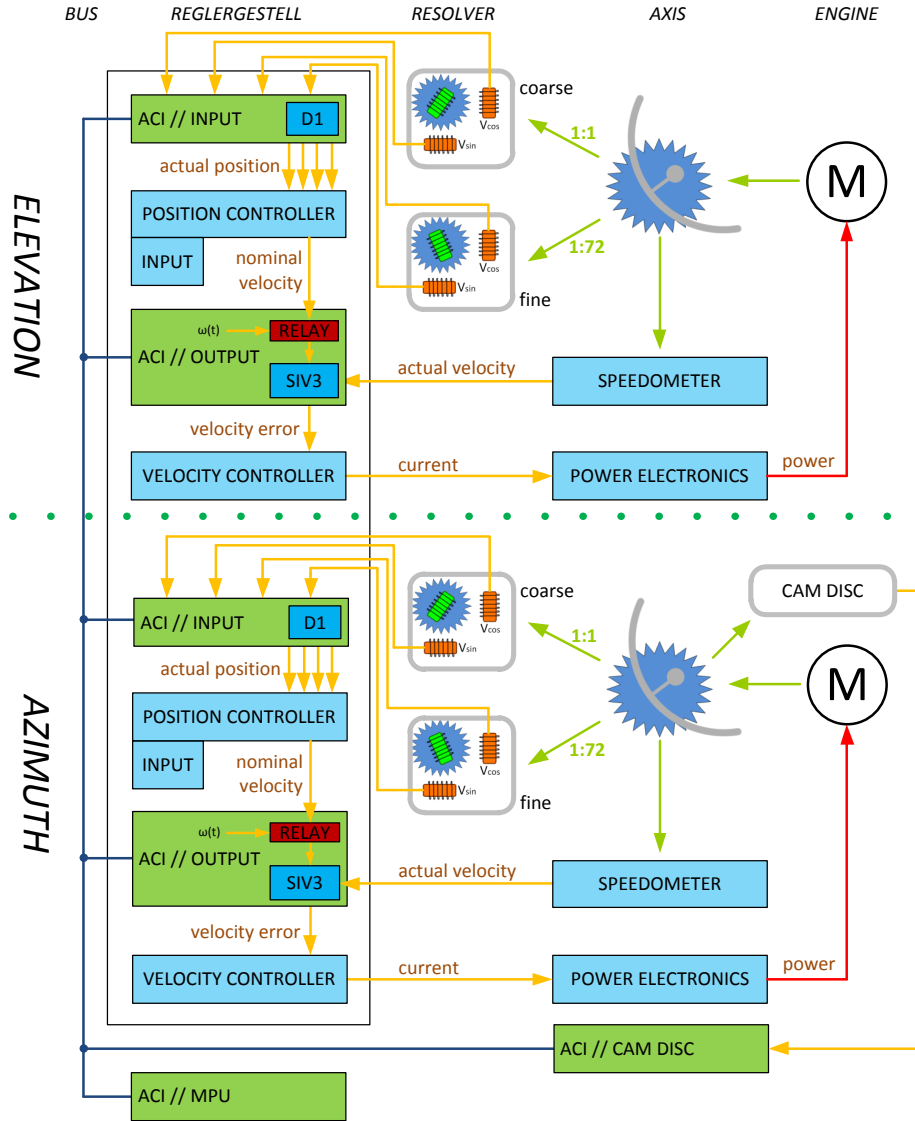


Figure 3.1: Simplified overview of the adapted antenna control loop. The original control loop as shown in Figure 2.8 is enhanced with digital interfaces. The interfaces are realized as so-called modules (symbolized by green boxes) which replace the existing cards in the control rack *Reglergestell*. The original functionality of these replaced cards is also implemented on the modules (replacements). The output modules use a relay to switch between the nominal velocity signal from the position controller and the nominal velocity signal which is generated by the modules. This relay is switched according to the control mode (see chapter 2.2) – if the PR mode is selected, the relay forwards the generated nominal velocity signal. A bus connects all modules and the Main Processing Unit of the ACI.

3.2 Interface modules

The interface modules have the task to enhance all necessary analog antenna signals with well-defined digital interfaces. Those interfaces are accessible by the MPU using a bus which is explained in chapter 3.3. Some interface modules receive the analog signal data from the antenna. And of course some other interface modules generate analog data signals and send those signals to the existing electronics of the antenna. In this matter, these modules could be called "technology age" adapters. Each module has its own software intelligence. It monitors for example permanently the validity of the MPU's commands and can act autonomously if the MPU seems to be broken or goes offline (eg. if the MPU stops sending velocity data the velocity output modules must automatically slow down the antenna smoothly).

In the actual configuration five modules are used:

input module An input module replaces the D1 card and samples the resolver signal data for position read-out. Each axis is equipped with an input module.

output module An output module replaces the SIV3 card and generates the nominal velocity signal. Each axis is equipped with an output module.

cam disc module The cam disc module determines the 360° space of the azimuth axis. Only this axis is equipped with a cam disc module.

3.3 Bus

This chapter enumerates the functionalities of the ACI bus.

First, all modules and the MPU have to communicate via a data bus. Beside the data bus, for an easy and uncomplicated setup all further necessary signals are routed via the same cable, including analog reference signals and supply lines. Consequently, each module needs only one single cable plugged in beside the connection to the antenna. Small outline CAT.5e-specified cables with suitable RJ45-jacks are used for the bus connectivity due to its very good transmission behavior. To summarize, standard network cables can be employed to connect the modules. For safety issues, the pin setting is chosen in a kind that a bus cable would not damage a network device if plugged in there by mistake. For additional safety, the color of the bus cables was chosen to be red to get the bus cables distinguishable from common network cables (which are blue).

3.3.1 Power Supply

One pair of wires of the bus cable is used for the power supply. Unlike the ACIv1 which needed six different external power supplies, this version is only supplied with one main voltage of 12 Volts. This allows the modules to generate their own supply voltages according to their needs and requirements for their power supplies like voltage level, accuracy and noise. For example, the output module derives ten different supply voltages from the main supply.

According to the CAT.5e specification¹, a single pair of wire is able to deliver 48 Volts with a current of 175mA. Each module is secured with a 150 mA fuse to avoid overloading of the cable. Since the CAT.6 standard release Power over Ethernet (PoE) is introduced which allows an even higher power transport.

3.3.2 Controller Area Network

Another function of the bus is, of course, the data transfer itself. The data transfer bus must handle multi-node, bidirectional communication. Furthermore it should offer high data rates with real-time capability and come with mechanisms ensuring safe data transports. The Controller Area Network (CAN)² Bus from Bosch is the most suitable bus for this application. It has been developed for rough but critical environments like car systems where disturbances must not affect the safety of the car. Furthermore it uses a packet-based arbitrary transmission which allows to set a priority tag to more important data packets – they will be sent earlier.

With its Cyclic Redundancy Check (CRC) checksum verification and differential transmission, the CAN bus fulfills all requirements to be used as data bus for the ACI. Due to its widespread applications there already exist fully developed hardware chips which can be directly connected to micro processor units using the Serial Protocol Interface (SPI) bus. Furthermore a Linux kernel driver for CAN (called SocketCAN³ developed by Volkswagen) is available what makes the CAN bus a perfect choice. CAN also needs only one pair of wires.

3.3.3 Hardware Protection System Signals

In addition to the data bus a simple hardware protection bus is routed via the main bus cable. The signal of the hardware protection bus is able to slow down the antenna in the very unlikely case that the data bus and at least one micro processing unit fail at the same time. This bus is consciously independent from all processing units and driven by pure logic circuits. The Hardware Protection System, again, needs two wires (one to signalize the error and another to release the error).

3.3.4 Individual Signals

The tasks of the last two wires of the 8-wired CAT.5e-cables are not uniquely defined: some modules need some specific, individual signals which can also be routed via the bus cable. The bus hub offers the opportunity to route these signals in the desired matter. In the current configuration there are five different individual signals:

- *PR bereit* signal
- *PR aktiv* signal

¹see IEEE 802.3af for more information

²A concise description about CAN and its concept can be found at [2]

³More information on SocketCAN can be found at the official development web page <http://developer.berlios.de/projects/socketcan/>

- 400 Hz Reference signal for the azimuth axis, normed to 20 Vpp
- 400 Hz Reference signal for the elevation axis, normed to 20 Vpp
- Power Control⁴

3.4 Antenna Protection System

One of the most important aspects of the development of the Antenna Control Interface is to protect the antenna. It is determined to be controlled remotely via the Internet. If a critical error occurs the system must be able to deal with it autonomously – there are no human supervisors to intervene. The protection systems are not only designed to guarantee material safety but also operational safety as long as this does not decrease the material safety.

There are multiple possible error scenarios: for example, the MPU control program might output wrong velocity data, a new module on the bus uses wrong data commands, the whole data bus might go offline, the program on the micro controlling unit on the several modules might be mis-programmed and also the hardware circuits and data converters on the modules may fail – nevertheless it must be avoided at any costs that wrong velocity data (which require to high velocities or invalid accelerations, for example) is sent to the antenna.

The protection system is divided in three different levels.

3.4.1 Level One: Main Processing Unit

The MPU is likely the most probable error source because all future works on the control loops and trajectory inputs are implemented here. Although there are several limiting filters, only the last filter in this system is part of the protection system. Therefore, it is called "omega filter" and is implemented right prior the velocity data send-out functions. If the omega filter detects invalid velocity data, it tries to limit the velocity and acceleration values so that the tracking can continue. It is the only part of the protection system which does not stop the antenna immediately when invalid data occurs.

3.4.2 Level Two: Module Software

After processing the velocity data by the omega filter and sending it to the modules via the bus, the microcontrolling units on the modules analyze the velocity data again. If the data does not seem to be reasonable it must be assumed that the whole MPU malfunctions (including the omega filter) and the Software Protection System (SPS) (see Section 4.1.3.4) slows down the antenna autonomously. However, there are also some other error reasons that will trigger the SPS like forbidden commands in some states. Every by a forbidden command affected module will send a so-called "softstop" CAN message to all other modules on the ACI bus. The modules will immediately slow down

⁴This signal is emitted by the MPU when the modules should be supplied with power.

and stop the antenna and block further commands. Each softstop broadcast message contains information about the softstop initiator and the error reason which will be logged by the MPU.

The modules' protection systems are released when the MPU restarts the operation (in fact, it can re-engage the active runlevel as explained later).

3.4.3 Level Three: Module Hardware

The last protection system, called Hardware Protection System (HPS) (see Section 4.1.2.5), is implemented directly in hardware. It acts if the microcontrolling unit or some important hardware components malfunction. Regarding the output modules, for example the generation of the 400 Hz velocity output signal is monitored.

All sensors and signals signalize with a logic '1' that the system is ok, and with a logic '0' that an error occurred. The idea behind is the following: if a signal line is interrupted, several pull-down resistors care that the incoming signal is a logic '0' and avoid that a broken connection is able to transmit an ok-state by mistake. Always a current has to flow to be interpreted as logic '1' and therefore as ok-state, increasing the safety.

If the HPS of one module is triggered, it is assumed that the microcontrolling unit or an important hardware component is broken. Although it would be possible to configure the CAN transceivers in a way that they send a stop broadcast to all other modules when a specific input pin is triggered, the HPS uses its own bus for safety reasons (if the microcontrolling unit fails the CAN node can also have a malfunction). This bus is used to tell the HPS of the other modules that an error occurred. The bus completely bypasses all processing circuits.

As soon as the HPS detects an error or another module sends an error signal using the HPS bus the output modules slow down the antenna via a fixed hardware system. Because the complete ACI can no longer be considered as to be correctly functioning when an HPS error occurs, the HPS can only be reset by an external system like the Antenna Monitoring System.

Besides the active protection the HPS offers, every circuit is designed in a way that invalid and possibly harming signals can not reach the antenna – clamps, resistors and limited supply voltages are used for that purpose. A further precaution regarding the output velocity cards is the direct connection between the *PR aktiv* signal which the antenna control emits if all systems are ready for an external velocity signal and the relays on the output modules which switch between the original velocity signal from the existing electronics of the antenna and the on-board generated velocity signal. That means if the antenna detects any error, it will stop emitting the *PR aktiv* signal and the relays on the output boards will interrupt the on-board generated velocity signal.

3.5 System Interaction

Figure 3.2 gives a summarized overview of the interaction of the ACI.

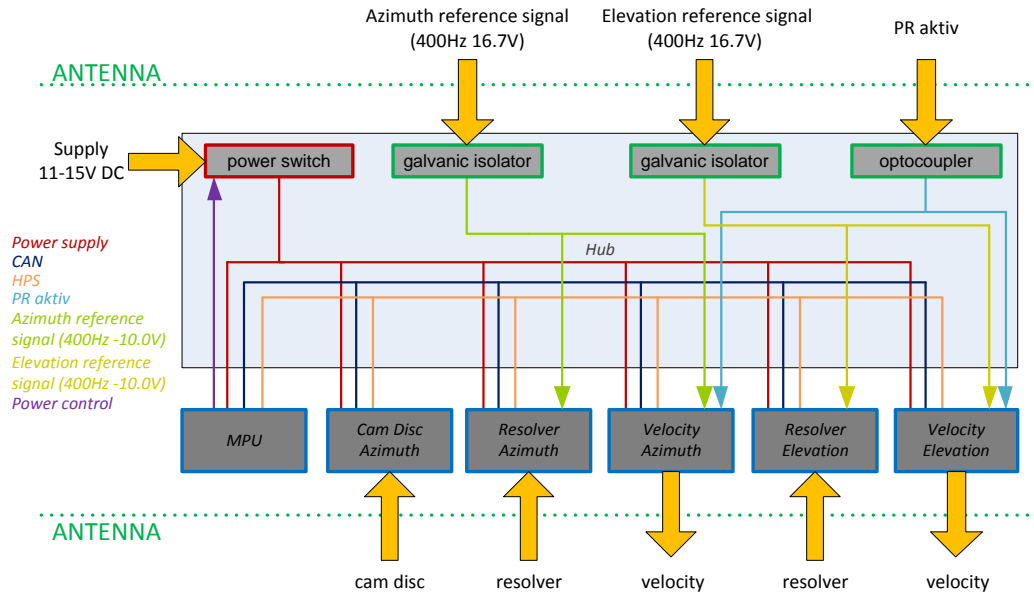


Figure 3.2: Interrelation of the Antenna Control Interface bus. Each interface module is connected to a hub which allocates all necessary signals in the bus. A mosfet on the hub can supply all modules with power. This mosfet can be triggered by the MPU. All modules and the MPU are connected amongst themselves with the CAN data bus. Also the HPS lines are routed to each module and the MPU. The hub offers the possibility to route the individual signals in the desired manner. In the actual configuration the 400 Hz reference signal of each axis is relayed to the modules which interact with the axis and need the reference signal. However, the reference signal is galvanic decoupled first before it is connected to the ACI bus due to safety issues. The output modules require the *PR aktiv* signal which is first galvanic decoupled using an opto-coupler and then routed to both modules.

4 Interface Modules

This chapter outlines the design of the interface modules and the implementation of the above mentioned features.

4.1 Common Setup

4.1.1 Task

The common task of all modules is to serve as an intermediary interface between the MPU and the antenna. Every module has to come with reliable mechanisms ensuring the protection of the antenna.

4.1.2 Hardware

This chapter enumerates the common hardware features. All components are placed only on the top side of the board to save mask costs if the components are mounted professionally by a company.

4.1.2.1 Galvanic isolation

As one of the safety precautions the module logic and the antenna interface (called "antenna side") is galvanic isolated from the bus (called "bus side") – if the bus is connected by mistake to voltages up to 1000V DC, it will not harm the antenna (but the antenna is slowed down very abrupt). For a full galvanic isolation between the bus side and the antenna side, the power supplies, the CAN bus and also all further signals have to be isolated.

For this purpose a novel CAN transceiver, the ISO1050 from Texas Instruments, is used. It combines a level converter, a galvanic isolator and a bus driver in one single component. All digital signals are converted using a digital isolator chip from Analog Devices. The most complex conversion concerns the analog 400 Hz reference signals. Because this signal can not be generated on-board using phase and amplitude information from a measurement (the 400 Hz reference signal is altering slightly but has to be relayed exactly because the electronics of the antenna uses it as reference to extrapolate the amplitude information) the analog signal itself has to be transmitted isolated from bus to antenna side. This is possible with the IC ISO122 from Texas Instruments which can transmit variable analog signals below 20 kHz. Additionally, according to its datasheet a noise filter is needed to eliminate noise on special frequencies. However, this IC and the required components for the filter are quite expensive and at least six pieces are needed for the whole system, resulting in costs higher than 300 Euro.

After direct contact with the manufacturer Texas Instruments sponsored generously eight pieces.

4.1.2.2 Fuse

To prevent any damage on the modules if the main power supply is wrongly connected, each module is equipped with its own fuse circuitry shown in Figure 4.1. It protects the module against voltage reversal and over-voltage. The fuse circuitry consists of a fuse, a diode and a zener diode. The fuse and the diode are connected in series; the diode defines the flow direction. The diode only allows a current flow if there is no voltage reversal.

Parallel to both components a zener diode in anti-flow direction is mounted. It will only conduct current if a zener-specific voltage is reached. This zener diode is used to protect against over-voltage – that means if the supply voltage reaches 16 Volt or more, the zener diode will conduct too much current and the fuse will trip. The on-board power consumers work in a voltage range between 9 Volt and 18 Volt with a nominal voltage of 12 Volt. The zener diode limits the upper voltage to 15 Volt. Below 15 Volt, the zener diode will not be active and a normal operation is possible.

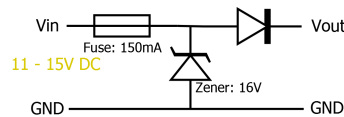


Figure 4.1: Power protection circuit.

4.1.2.3 On-board Power Supplies

The ACI uses a decentralized power supply. Each module has to derive its supply voltages from the 12 V main power supply delivered by the ACI bus. With exception of the cam disc module ten different voltages are used on each module, listed as follows:

- bus-side: 12V (the ACI bus supply)
- bus-side: 5V for the HPS, status lights and CAN driver
- bus-side: ± 15 V for the galvanic isolation of the 400 Hz reference signal
- antenna-side: 5V for logic-components
- antenna-side: 3.3V for the processing unit, derived from the above 5V
- antenna-side: ± 5 V for the analog circuitry
- antenna-side: ± 15 V for the galvanic isolation of the 400 Hz reference

For the generation of the voltages low-noise and galvanic isolated DC-converters (TM-series from manufacturer Traco) are utilized. The 5V for the logic-components on the antenna side is separately generated from the $\pm 5V$ analog supply because the digital parts produce high noise while the analog part should be kept in a low-noise environment for better accuracy. In addition the power supply performs better when symmetrically loaded. All DC-converters are supplied from the ACI bus main power supply.

4.1.2.4 Uninterruptible Power Supply

Originally it was planned to equip each module with an own Uninterruptible Power Supply (UPS) to be able to slow down the antenna smoothly if the main power supply fails. This feature has been dropped because it makes the whole system more complex (a special loading circuit is needed) and decreases the safety. Besides, the incoming 400 Hz reference signal is dependent from the main power supply – without the reference signal it is not possible to generate the velocity signal.

If the power supply breaks in the current configuration, the velocity signal immediately decreases to zero – leading to an immediate but harmless antenna stop. The UPS is obligatory when the current velocity output concept is replaced by a system that directly controls the antenna's power electronic.

4.1.2.5 Hardware Protection System

This chapter outlines the functionality of the HPS. Figure 4.2 gives an illustration about the functional principle.

The main function of the HPS is to monitor the hardware systems, check them for errors and – if an error occurs – slow down the antenna and keep it halted (stop-state) until the error is solved *and* the HPS is released. If the user tries to release the HPS while the error is not solved, the HPS should not release the hardware circuits. The HPS generally consists of two parts: on the one hand, sensors monitoring the circuitry and check their health and actors which slow down the antenna if needed (in case of the output module). This part is different on each module. On the other hand, the HPS exists also of a logic that communicates with the other HPSs of the other modules, reports errors to them or receives errors and provides an interface to the sensors and actors. The HPS is designed for a very robust and reliable operation. After dealing with some different ideas and concepts for an HPS, the following concept has been considered as the best.

First, the HPS uses two wires of the ACI bus: one so-called "error" line and a "release" line. The error line is used to detect an error-occurrence. The release line is used to release the HPS from the stop-state. According to the signal philosophy mentioned in chapter 3.4.3, a logic '1' means that the system is ok and a logic '0' means that the HPS should go in stop-state concerning the error signal. A logic '1' means that the HPS should release, if possible, and a logic '0' means that the HPS should stay in the current state concerning the reset line. The actual state of the error line is directly passed to the actors. The error line is pulled up (logic '1') by a resistor located in the ACI hub and can be pulled to ground (logic '0') by a mosfet which is part of the HPS.

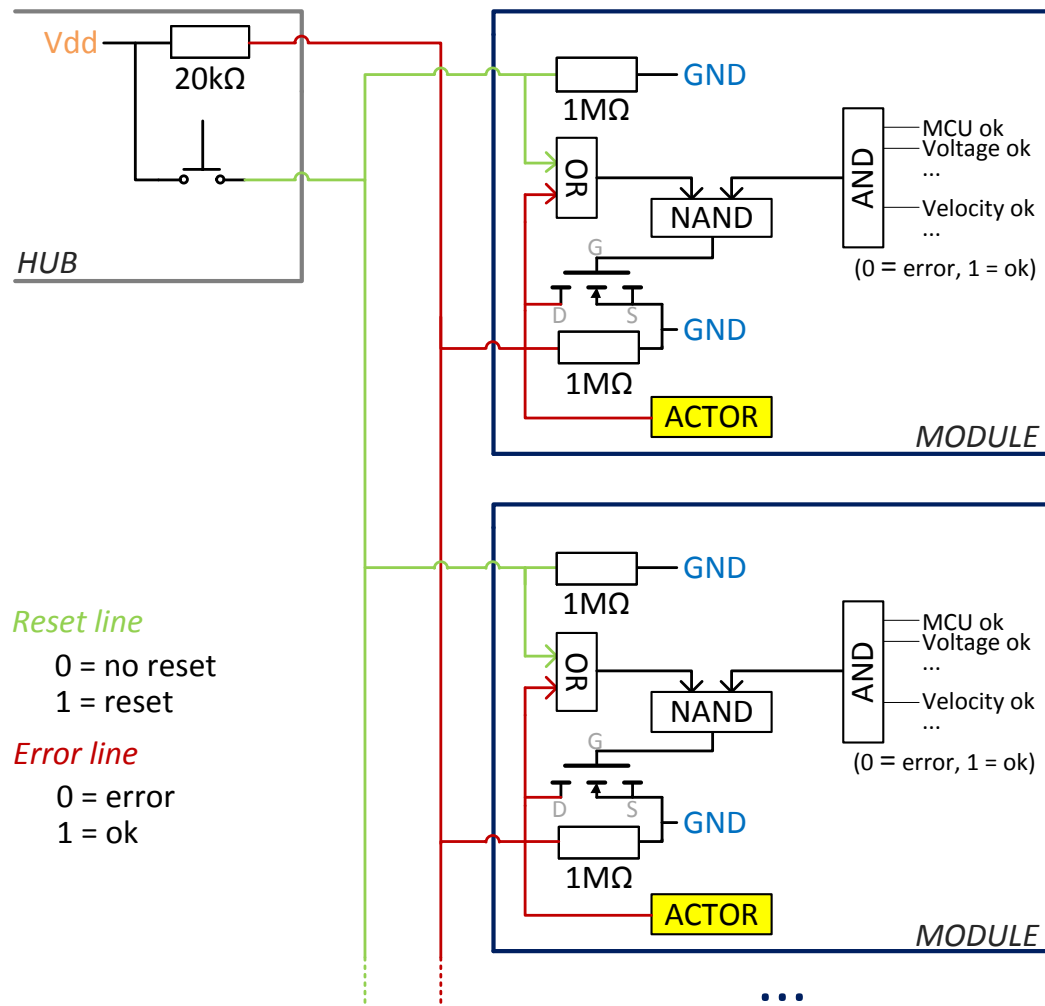


Figure 4.2: Hardware protection system circuitry. All modules are connected to the HPS bus. The hub pulls up the error line and can be used to manually release the HPS. As soon as the Antenna Monitoring Interface is ready, it is also able to pull up the error line. The MPU is connected to the HPS bus as well and contains the same HPS circuit as the modules. The HPS of each module can trigger an actor. In the actual configuration only the output modules are equipped with an HPS actor. The "OR" gate decides if the HPS should trigger by an external request. The "AND" gate collects all possible errors, and the "NAND" gate decides if the mosfet should trigger. All HPS lines are equipped with pull-down resistors that guarantee a safe state even if any line is interrupted.

This mosfet is activated if

1. the local sensors detect an error *or*
2. the error line emits the error signal *and* the release line is not active. That means that on an incoming error signal every HPS on the bus itself emits an error and intensifies the signal.

In other words, the mosfet on each HPS does not emit the error signal (that means does not pull the error line to ground) if the local sensors do not detect an error and the error line is already emitting the ok-signal or is emitting the stop-signal while the release-line is triggered.

The following scenario should illustrate the functionality of the HPS. In this scenario, five modules called A ... E are connected to the bus.

1. In the initial situation all modules are assumed to be healthy. The HPS is not triggered, nor is the release line.
2. The sensors of the HPS of module A detect a too high acceleration on its nominal velocity output signal. The first condition for triggering the mosfet is met, the mosfet pulls the error signal to ground and overwrites therefore the logic '1' generated by the pull-up resistor on the ACI hub.
3. After the error line is pulled to zero, two things happen. On the one hand, the actors of each HPS (including module A) are engaged, the antenna is slowed down immediately. On the other hand, the second condition for triggering the mosfet *on every module* is met, now the error line is pulled to ground by every HPS.
4. Assuming that the sensors of the HPS of module A still detect an error, but the user, however, tries to release the HPS by drawing the release line to logic '1'. The consequence is that – on every module – the second condition for triggering the mosfet is no longer met. All modules, except of module A with its still active condition one, are no longer emitting the error signal. But because the actors are only dependent from the actual state of the error line and module A is still drawing it to logic '0', the actors of all the HPSs stay in stop-state.
5. The user stops trying to release the error using the release line. The second condition for triggering the mosfets on all HPS are met again, all mosfets are triggered.
6. Imagine that now the error of module A could have been solved. As soon as this happens, the first condition of triggering the mosfet of the HPS of module A, which had been met until now, is released. But because the second condition is fulfilled on every HPS, it stays in stop-state.
7. The user tries again to release the error. The second condition is no longer met, and every mosfet stops pulling the error line to ground. The pull-up resistor on the ACI hub pulls a logic '1' on the error line. All actors allow control operation again.

8. The user stops triggering the release line. Because the second condition is still no longer fulfilled (before, the release line was logic '1', now, the error line is logic '1'), all HPS stay in ok-state until the next error occurs.

Another scenario might be that the user triggers the release line until the error gets finally solved. In this case permanently only one mosfet (the mosfet of module A) pulls the error line to logic '0'. Although this is possible without any safety problem, the procedure in the upper scenario is even more safe and therefore recommended.

Because the HPS also monitors the power supplies, the HPS always triggers when the main power supply is switched on (the power supplies do not reach their target voltage immediately). That means that the HPS has to be released each time the ACI system is powered on (in the final configuration with a working monitoring system, this should be done automatically). Some measurements have shown that the HPS, thanks to its few simple components, is triggered even if the error exists only for a very short time (300 ns or less). However, it has been acting very robust since the first tests and *never* emitted an error by mistake. This is also due to the noise-robust circuit monitoring sensors.

The Main Controlling Unit (MCU) is able to read out the actual state of the HPS and can also trigger it manually.

4.1.2.6 Hardware Protection System – status information lights

In contrast to the SPS, the HPS has no opportunity to log the error. This is not a big issue because the HPS is considered to be triggered very rare or hardly never. If the HPS is triggered anyway, a traffic light similar status display is used to tell the user about the available error information (see also Figure 4.3):

green The green status Light Emitting Diode (LED) is on when the module is supplied with power and the fuse circuit is not triggered.

yellow The flashing yellow LED signalizes that the release line is drawn to logic '1' and therefore currently triggered. If the release line is triggered and the yellow LED is not flashing the release line is interrupted.

red The red LED signalizes that the on-board mosfet of the HPS is currently triggered and therefore the error line is drawn to zero. If the error line emits the error signal and the release line is triggered at the same time, the conditions for triggering the mosfet (and therefore an illuminated red LED) is only met if there is an existing local error (first condition). This can be used to determine which module emits the error: if the red LED of a module is on although the release line is triggered (the yellow LED is flashing), its local HPS sensors currently detect an error.

4.1.2.7 Microcontrolling Unit

Each module is equipped with a high-performance microcontrolling unit (MCU). It mainly implements the SPS, the CAN gateway, the driver for the specific analog hardware and an intelligence subsystem (the "brain" of the module).

The microcontroller atxmega128a1 from Atmel, which has been used for ACIv1, has shown a very robust performance and is also used as the module processing unit. For a high timing accuracy the MCU is equipped with an external 10 MHz oscillator. Major parts of the former ACIv1 firmware could be reused.

4.1.2.8 CAN node

The CAN connection consists of two parts: the CAN controller, which is directly connected to the MCU and controls the second part, the already-mentioned galvanic isolated CAN driver. The MCP2515 from Microchip is used as CAN controller for its relatively easy control and its compatibility not only with the MCUs but also with the MPU (using the SocketCAN driver). It is also the most common CAN controller and therefore a lot of documentation is available. Some tests have affirmed its reliable performance with respect to the data transmission.

4.1.2.9 Connectivity

Each module is provided with several jacks and switches (see Figure 4.3). All are accessible from the front, allowing their use even if the module is mounted in the rack:

RJ45 jack The RJ45 jack is used to connect the module with the ACI bus.

JTAG header In order to program the MCUs the Joint Test Action Group (JTAG) header offers a programming interface for In-System Programming (ISP). For an easy and fast workflow a flashing tool was written for the ACI which is based on the popular AVRdude¹ driver. To avoid that the wrong firmware is transferred to a module (eg. input module firmware instead of output module firmware) the tool explicitly tells the user what firmware is designated to be flashed. Chapter 5.4 gives a description.

Debug header Every new firmware needs to be tested. For this matter a debug interface is very useful. This debug interface uses the popular UART interface. An adapter from this header to USB including a debug tool has also been developed in this thesis and is described in chapter 5.5. The firmware contains the same software debug interfaces as the original ACIv1 firmware.

Reset button A reset button is placed on every module to restart the MCU. This has to be done after each firmware flash. The reset button can also be utilized to test the safety systems.

4.1.3 Software

The ACI module firmware is based on the ACIv1 firmware which was already optimized for performance. The firmware has been extended with the following sub-functions.

¹More information can be found under <http://www.nongnu.org/avrdude/>

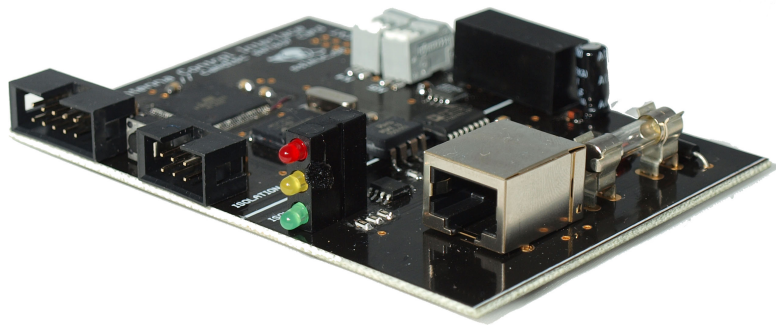


Figure 4.3: The common jacks of the interface modules (here the cam disc module): from left to right, the JTAG header, the reset button (partly hidden), the debug header, the HPS status LEDs and the ACI bus jack.

4.1.3.1 Timer

Several functions have to be executed regularly with a specific time interval. With ACIv1, the functions had to use own MCU-internal hardware timer and interrupts. With the new firmware a timer subsystem has been created offering a very easy and reliably timing interface to every subsystem: if a subsystem needs a function to be called with a specific time interval, it can tell the timing subsystem its function pointer and the desired time interval. The timing subsystem itself uses a hardware timer to measure the time intervals and starts the execution of the registered functions according to the selected time intervals. It is also possible to change the settings or clear a timer.

4.1.3.2 Runlevel

Depending on the actual state each module has to emit an error on a specific signal or may not emit an error if a specific signal gets lost. For example when the system is not actively controlling the antenna the connection to the MPU is not required. However, if the ACI is actively controlling the antenna, the SPS has to check permanently the validity of the data and also has to check if the MPU is accessible. To distinguish these states, runlevels have been introduced. Those runlevels are implemented on the module MCUs and also on the MPU. The runlevels of all modules and the MPU have always to be in the same state. Each runlevel needs some specific conditions to be engaged. Additional, in order to engage a runlevel, it has also to meet all conditions of the lower runlevels.

The tasks of all ACI subsystems are dependent from the actual runlevel. All the tasks of all lower runlevels have also to be executed.

RUNLEVEL	NAME	CONDITIONS
0	ACI shut down	No conditions have to be met.
1	ACI server online	The MPU software has been started successfully.
2	ACI modules online	The main power supply is switched on.
3	ACI ready	All modules needed for active operation are online, the SPS and the HPS are released.
4	Antenna online	400 Hz reference signals and position data are available.
5	Antenna ready	Antenna emits <i>PR aktiv</i> signal, all relays triggered.
6	ACI preparing	All systems are ready, no errors are detected.
7	ACI active	Runlevel 6 has been active for more than 5 s.

Table 4.1: The runlevels and the specific conditions.

RUNLEVEL	TASKS
0	No tasks.
1	ACI server program allows clients to connect.
2	The status of all available modules is frequently checked by the MPU.
3	Input modules start sending resolver and cam disc data, if available.
4	No tasks.
5	No tasks.
6	MCUs start "runlevel 6 to 7 delay" counter.
7	MPU starts to send velocity data continuously.

Table 4.2: The runlevels and the specific tasks.

4.1.3.3 Strategy

The strategy tag is another attribute. It is also synchronized between the modules and the MPU. The strategy tag defines the actual operational policy as explained in table 4.3.

If the debug operation strategy is selected, the debug features are accessible. For example, this operation allows the modification of some control parameters during an active tracking. In this way, the function of the controllers can be optimized. All protection systems still protect the antenna against a misconfiguration.

The critical operation can be used during very critical tracking actions, where a high operational reliability is very important. The material safety is decreased slightly. However, if actually an error occurs in this strategy, the ACI auto-restarts the whole system immediately and tracks again autonomously without any user intervention.

STRATEGY	DESCRIPTION
0	normal operation
1	debug operation
2	(reserved)
3	critical operation

Table 4.3: Currently supported debug modes.

4.1.3.4 Software Protection System

The MCU also contains the whole logic of the SPS. The SPS of all modules are based on some SPS basic functions.

First, the SPS checks every incoming runlevel engage request. If all runlevel conditions for this runlevel are fulfilled the desired runlevel will be engaged. If not all runlevel conditions are fulfilled, a softstop broadcast message will be sent and the runlevel is not engaged. It should be noticed that it is only possible to increase the runlevel by one, that means it is not possible to pass over one certain runlevel. A request to pass over a runlevel will lead to a softstop error. However, it is possible to decrease the actual runlevel by more than one.

If a softstop message is received by an SPS it adapts the runlevel: if the runlevel is higher than 5, it is decreased to 5. If the runlevel is equal to 5 or lower, the runlevel will not be changed. The background is that the runlevels 0 to 5 are initialization runlevels: each runlevel corresponds to a state of the starting system. The runlevels 6 and 7 are operational runlevels – in both runlevels all conditions for an active motion operation have to be met. As soon as one or more conditions are no longer fulfilled in these two runlevels, the runlevel is immediately set down to 5. In runlevel 5 all functionalities of the modules are still provided and the system can be restarted quickly. It is the task of the MPU to gather more information about the error and decide the further steps to restart the system.

Starting from runlevel 2, the MPU starts to request the status from all modules which are accessible in the bus – the MPU gathers information about the modules on the bus and permanently checks their availability. The modules use these requests to determine if the MPU is online. If there has not been any individual module status request for more than 100 ms, a module assumes that the MPU is offline and emits a softstop broadcast. The status requests are therefore also called *heartbeat*.

The SPS is also responsible for the verification of some critical hardware components like the external oscillator.

As later explained the SPS of each module also monitors all external run conditions (like the 400 Hz reference signal) and triggers a softstop if one condition is no longer met in runlevels 6 and 7.

The actual implementation of all SPS defines overall eleven conditions which, as mentioned before, are – together with the sender module ID – attached to each softstop broadcast.

CODE	NAME	DESCRIPTION
1	heartbeat	The MPU has not sent a status requests for 100 ms.
2	speedbeat	The MPU has not sent a velocity request for 20 ms.
3	velocity	The MPU has requested a too high velocity.
4	acceleration	The MPU has requested a velocity which would lead to a too high acceleration.
5	forbidden request	A request has been received which requires a higher runlevel.
6	runlevel overstep	A runlevel has been missed out.
7	runlevel 6 condition	Not all conditions for the requested runlevel 6 are met.
8	runlevel 7 condition	Not all conditions for the requested runlevel 7 are met.
9	external conditions	The external run conditions are no longer met.
10	HPS	The HPS has been triggered.
11	SPS	The SPS has been triggered (broken component detected).

Table 4.4: SPS softstop trigger circumstances.

The external conditions are module-specific and outlined in Table 4.5. A detailed explanation is given in the description of the modules.

MODULE	DESCRIPTION
resolver module	The 400 Hz reference signal is assumed to be lost.
cam disc module	The cam disc switches are not plugged in.
velocity module	The <i>PR aktiv</i> signal is no longer given.

Table 4.5: External signals which are monitored by the SPS.

4.1.3.5 CAN

This chapter outlines the CAN model designed for the ACI. The performance and reliability of the CAN bus are strongly dependent on the CAN model. In order to optimize the model, diverse properties have to be regarded. Due to clarity only the most critical properties are described in this context.

The firmware has been extended by a CAN driver to control the CAN controller chip and a CAN message gateway. The CAN driver implements all necessary functions to configure the MCP2515 CAN controller and to interact with it. Some basic functions of the CAN driver have been inspired by a CAN workshop [3] offered by Fabian Greif. The CAN driver has been extended with an interrupt-controlled and ring-buffer enhanced interface. This enables an effective high-speed and low-latency communication. The

CAN driver is also equipped with CAN ID² hardware masks. The MCP2515 offers the opportunity to setup some criteria the CAN message identifiers (IDs) have to fulfill to be transmitted to the MCU, saving processing power (for example, the output modules do not have to process the resolver data and therefore the CAN messages containing the resolver data may be masked out by the CAN controller on the output modules³).

The ACI CAN bus works with 11bit-identifiers, offering $2^{11} = 2048$ different CAN IDs, and a bus speed of 1 MBit/s. Both the CAN message priority and the hardware masks are dependent on the CAN IDs. Therefore much effort was expended on the outline of the CAN ID table, according to the following datums:

- For the CAN ID arbitration, a logic '1' is the recessive bit and a logic '0' the dominant bit. The arbitration starts with the most significant bit (MSB) of the CAN ID, so a lower ID number represents a higher priority.
- The MCP2515 supports only two hardware masks.
- There are different kinds of CAN IDs used by the ACI bus: specific data messages for the communication between MPU and a specific module, and broadcast messages which can be emitted by every CAN node (modules or MPU) and has to be processed by every other CAN node on the bus.

The 11-bit CAN ID field is split into five parts in descending priority:

1. separate priority field (2 bits)
2. task identifier (4 bits)
3. broadcast tag (1 bit)
4. module identifier (only parsed if the message is no broadcast, 3 bit)
5. axis identifier (only parsed if the message is no broadcast, 1 bit)

The first part, the priority field, defines the priority of the message and therefore must be located in the beginning of the message. Although the other parts are already sorted and optimized for an arbitration-focused design, this separate priority field can be used to explicitly control the priority and therefore override (increase) the inherent message priority. The priority field is not analyzed by the module MCUs because the arbitration is processed by the CAN controllers.

The second part defines the task of the current message. By reference to the task, the priority of the message can be specified since task field is located directly after the priority field.

The interpretation of the data fields of each task can be found as comment in the specific source code. Usually, the modules automatically emit new data when available.

²Below other, a CAN message contains two parts: a CAN ID tag which defines the type ("label") of the message, and a data field which contains the binary payload data.

³Note that the CAN bus is a broadcast-only protocol; there is no recipient-tag in the CAN messages and therefore all CAN nodes have to decide themselves if the CAN message is allotted to them.

DECIMAL	BINARY	PRIORITY
0	00	critical
1	01	high
2	10	middle
3	11	low (standard priority)

Table 4.6: CAN message priority field.

DECIMAL	BINARY	TASK
2	0010	set velocity
4	0100	resolver data
5	0101	cam disc data
7	0111	set runlevel and strategy
8	1000	module status
9	1001	module extended status
10	1010	external information 1
11	1011	external information 2
12	1100	debug interface: Master in, Slave out (MISO)
13	1101	debug interface: Master out, Slave in (MOSI)
14	1110	broadcast ping

Table 4.7: CAN message task field.

By setting the desired CAN ID and the CAN remote transmission request (RTR) bit the MPU can force the module emitting the requested CAN message. This is used for the heartbeats (requesting module status information), for example.

The external information messages are module-specific and contain individual data, like the amplitude and period length of the 400 Hz reference signal in case of the resolver input modules. For a direct debug gateway via the MPU it was initially planned to route a terminal connection via the CAN bus instead of only the UART debug header, offering the opportunity to debug the modules via the Internet. This function is not realized in the actual configuration because using the external information CAN messages all available debug information is already accessible by the MPU. The tasks "set runlevel and strategy" and "broadcast ping" are only emitted by the MPU and sent as broadcasts. The broadcast ping asks all available modules on the bus to send out their status at the same time – the MPU can therefore figure out which modules are online. The broadcast ping does not replace the heartbeats which request the status of a module seriatim. All other tasks are intended to be associated to a certain module.

Although the task implies already a broadcast message, a separate part is used for the identification of a broadcast message due to two reasons: on the one hand, the task list can be separated between module affiliated messages and broadcast messages, offering a total amount of 16 associated messages and 16 broadcast messages. On the other hand

the hardware filters of the CAN controller have to distinguish whether the incoming message is either a broadcast message or a message which is explicitly intended for the local module. Because the hardware filters are spare, a separate explicit broadcast field makes the recognition of a broadcast message more easily and the use of a hardware filter can be avoided here.

DECIMAL	BINARY	BROADCAST
0	0	broadcast message
1	1	associated message

Table 4.8: CAN message broadcast field.

If the message is an associated message, the last two parts are used to determine the addressed module. The first part defines the kind of module, the second part defines the axis.

DECIMAL	BINARY	MODULES
1	001	velocity output module
4	100	cam disc input module
5	101	resolver input module

Table 4.9: CAN message module type field.

DECIMAL	BINARY	AXIS
0	0	azimuth
1	1	elevation

Table 4.10: CAN message axis field.

If the message is a broadcast message, these two parts may be ignored. The bit order of the axis field is selected according to their priorities. The azimuth axis is intended to be more critical because it possesses more mass and is moving in a human accessible area.

If the hardware filters are triggered on a just transmitted CAN message, the CAN controller signalizes the MCU a new message using the interrupt line. The MCU immediately pauses its actual work and reads the CAN message into its ring buffer in order to avoid losing messages. Then the MCU continues its actual work.

On a certain point in its main loop (which is executed every 40 μ s) the ring buffer is read out and each message is parsed by the CAN gateway and then routed to the software intelligence of the MCU, where it is checked by the SPS before being processed.

In order to trigger one of the both protection systems a specific CAN message can be used. The SPS always sends out this message when it is triggered, the HPS does not use

the CAN bus for broadcasting the stop request (but the hardware error line). However, the HPS can be triggered via the CAN bus manually.

CAN ID	DESCRIPTION
00 0000 0 000 0	hardstop broadcast: all modules trigger their HPS.
00 0000 0 000 1	softstop broadcast: every SPS on the bus is triggered.

Table 4.11: CAN stop request messages.

CAN ID (BINARY)	PURPOSE
11 1110 0 111 1 [RTR]	emit broadcast ping
11 0010 1 001 1	send velocity command to velocity output module elevation

Table 4.12: Two examples to illustrate the composition of the message fields.

4.2 Interface Specific Setup

4.2.1 Resolver Input Module

4.2.1.1 Task

The resolver input module samples the analog resolver signals. It replaces the D1 card mounted in the *Reglergestell* as shown in Figure 4.4. The D1 card consists of few voltage dividers, realized with ordinary resistors. Each ingoing signal is divided to approx. 23%. Four of the voltage dividers process the raw resolver data which has to be sampled: Sine and cosine of the coarse and the fine resolver. The resolvers are fed with the 400 Hz reference signal and output a signal with a maximum amplitude of approx. 18 V. The maximum amplitude for the sine signal is reached at the resolver angle $\phi = \pm 90^\circ$ and hence for the cosine signal of $\phi = 0^\circ$. This signal is attenuated to a maximum amplitude of approx. 4.1 V by the D1 circuit and routed to the control electronics.

To determine the angle of the resolver the atan2 -function is used with the sine and cosine voltage signals as parameters. It should be mentioned again that the atan2 -function relies on the ratio of the sine and cosine voltage signals, not on their actual amplitude. In order to get an accurate voltage signal measurement three approaches are employed:

- A high Signal-to-Noise Ratio (SNR) is desirable to minimize influence of the noise. Therefore the resolver signals are sampled when the 400 Hz reference signal $V_{\text{ref}}(t)$ is near its maximum amplitude, resulting that also the in-phase resolver signals are in the region of their actual (position-dependent) maximum amplitude.
- Because the measurement accuracy depends on the ratio of the signals, a latency between the sine and the cosine measurement would directly cause an inaccuracy.

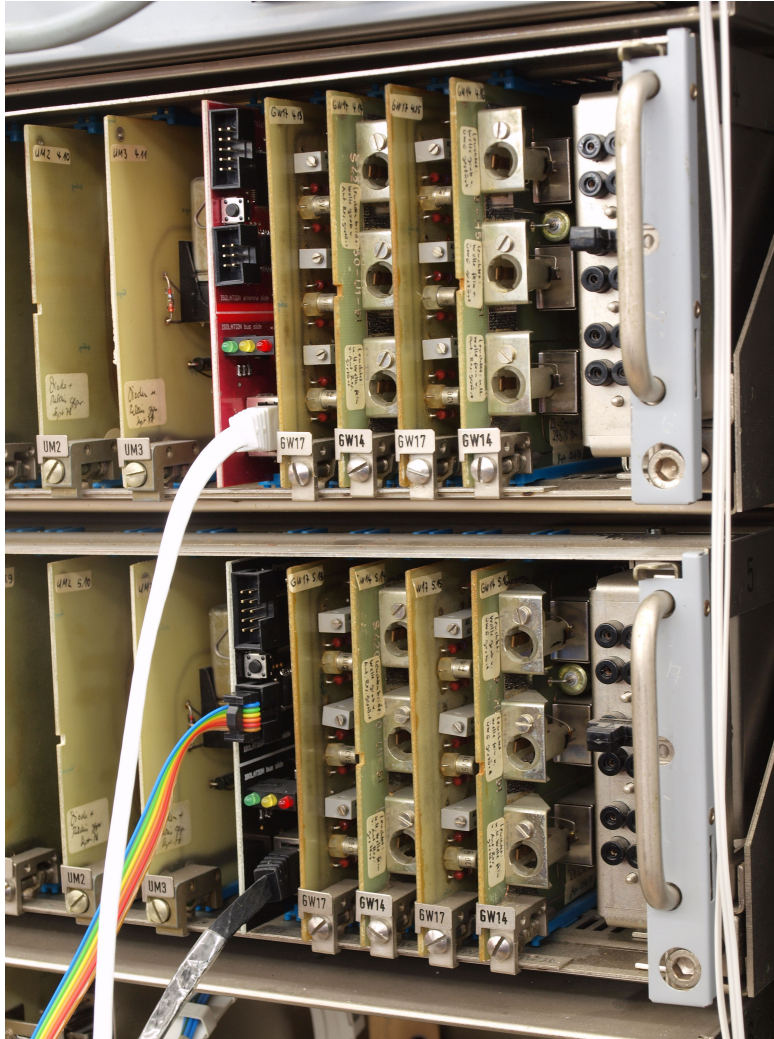


Figure 4.4: Resolver input modules mounted in the *Reglergestell*. The upper module replaces the D1 card of the elevation axis, the lower module replaces the D1 card of the azimuth axis. The modules are connected via the ACI bus, and the lower card is currently debugged using the debugging jack. In this photo, the *Reglergestell* is opened. In the actual configuration (see Figure 2.1) it is closed and the bus cables are routed to the back of the racks where the ACI hub is installed.

Hence, a multi-channel analog-digital-converter with sample-and-hold circuits is used. When the analog-digital-converter starts a measurement, the sample-and-hold circuits are triggered at the same time. The circuits will measure the actual voltage signal of their individual channels and store them analog to a, for a high accuracy optimized, buffer. Then the analog-digital-converter core immediately samples all the sample-and-hold buffers sequentially and stores the digitalized values in value-stable digital registers. These registers can be accessed then at any time.

- To improve the measurement accuracy, multiple samples are collected sequentially and then averaged, resulting in an improved accuracy of the signal.

To determine the point of time when the resolver signal reaches its maximum amplitude the resolver signal additionally has to be analyzed by the resolver input module. The analog-digital-converter uses five channels: the four resolver signals and the reference signal.

To guarantee that the measurement sequence is actually triggered in the region of the maximum amplitude an adaption of the so-called early-late filter has been designed: In addition to the averaged measurement samples, some samples before and after are grabbed by the MCU. These additional samples are used to figure out the correct start point of the measurement sequence: their $V_{\text{ref}}(t)$ channels are averaged and afterwards compared, resulting in the voltages V_{before} and V_{after} . If the averaged reference signal value of the measurements before and after the main measurement are identical, the center of all measurements has been placed on the middle of the reference signal amplitude. If the values are not identically, the difference can be used to calculate a new measurement starting point with the formula:

$$V_{\text{trigger}} = V_{\text{oldTrigger}} + \frac{V_{\text{after}} - V_{\text{before}}}{2} \quad (4.1)$$

A voltage comparison with $V_{\text{ref}}(t)$ and V_{trigger} is employed. If V_{trigger} is higher than $V_{\text{ref}}(t)$, a measurement (consisting of the main measurement and the two additional measurements before and after the main measurement) is triggered. The default, starting value of V_{trigger} is the half of the usual amplitude height. To make this filter more robust against signal disturbances, the adaption of V_{trigger} is slowed down by modifying the formula to:

$$V_{\text{trigger}} = V_{\text{oldTrigger}} + \frac{V_{\text{after}} - V_{\text{before}}}{4} \quad (4.2)$$

That means that the positioning error of the measurement trigger voltage is not immediately placed correctly and the transient effect needs few measurements, resulting in an insignificant adaption time up to $10 - 20 \mu\text{s}$. Note that also measurements which are not settled in region of the maximum reference signal amplitude deliver reliable positioning data. Figure 4.5 gives an example of the adaption procedure.

In the current configuration, a measurement triggers only on the positive amplitude of the 400 Hz reference signal, resulting in 400 measurements a second. Due to the high-performance components (analog-digital-converter, module MCU, CAN bus) it would

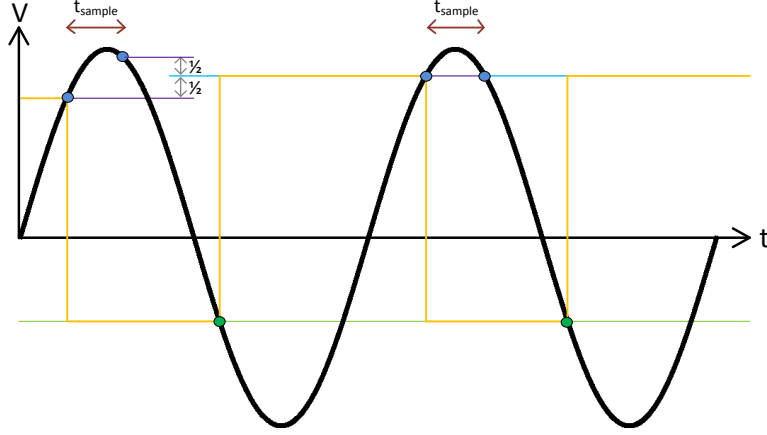


Figure 4.5: Adaption procedure of this approach. To illustrate this example, see formula 4.1. The blue level symbolizes the positive trigger level V_{trigger} , the green level symbolizes the fixed negative trigger level. The orange level indicates which of both trigger levels is currently active. The two blue dots of each period indicate V_{before} and V_{after} , and the mean value of both levels is used as the next positive trigger level. Using this formula, the trigger level will be adapted now and the two reference measurements should be equal. Therefore, the positive trigger level will no longer change. In reality, the trigger level alters slightly caused by the noise.

also be possible to trigger an additional complete measurement on the negative maximum amplitude, resulting in 800 resolver position updates of one axis a second.

If there is a high signal disturbance, the V_{trigger} may be set higher than the amplitude of the reference signal and therefore, no future measurement will be taken. To avoid this, a safety timer monitors the time since the last measurement. If it determines a time significantly higher than the 2.5 ms period length of the reference signal, V_{trigger} will be reset to a lower basic startup voltage which is definitely lower than the reference signal amplitude and the counter is reset to zero. If still no measurements are triggered, the reference signal is assumed to be offline. If this happens in an active runlevel (6 or 7), a softstop is triggered. However, if the measurement could be relaunched after resetting the V_{trigger} , a debug counter is increased by one. This counter can be accessed by the MPU for monitoring issues. Starting with an offline reference signal, the same methods are used: if only one measurement is triggered, a debug counter is increased. If two and more measurements are triggered sequentially, the reference and resolver signals are assumed to be online.

4.2.1.2 Hardware

The reliability and accurateness of the measurements directly depends on the performance of the analog-digital-converter. The analog-digital-converter AD7606-6 by Analog Devices, which has also been successfully used in ACiv1, is utilized. It provides the following important features:

- true bipolar analog inputs (Positive and negative voltages can be measured.)
- six channels: four are used for the resolver signals (coarse sine, coarse cosine, fine sine, fine cosine) and one for the reference signal. One channel is not used.
- sample-and-hold circuits
- selectable input range of $\pm 5V$ or $\pm 10V$
- 16-bit resolution
- up to 200,000 samples a second on all channels, resulting in 30,000 effective samples a second.
- parallel bus for high-speed read-out
- input channels are secured with clamps and two $1\text{ M}\Omega$ resistors

Thanks to clamps, the AD7606 can safely be connected to the resolver signals, without the need for decoupling operational amplifiers in between. The operational amplifiers would disturb the signal quality significantly in relation to the accurateness of the AD7606. The ingoing resolver signals on the D1 card cannot be measured by the AD7606 because of its insufficient maximum range of $\pm 10V$. Therefore, the attenuated signals are measured; the AD7606 is configured to use the input range of $\pm 5V$ which is sufficient here. For the generation of the trigger voltages two Digital-Analog Converters are employed which are controlled by the MCU. One Digital-Analog Converter (DAC) is used for the positive trigger voltage, the other for the negative trigger voltage. Although it would be possible to use one DAC and generate the positive and negative trigger voltage from its output, the second DAC offers a higher flexibility and also the opportunity to implement different read-out strategies. The trigger voltages are compared with the reference voltage using a comparator. The outputs of these comparators are used as interrupt sources for the module MCU.

4.2.1.3 Software

When the MCU starts up, it sets both DACs to half of the voltage of the usual reference signal amplitude. The "positive" interrupt (triggered by the comparator analyzing the positive amplitude) will be activated then. If the interrupt is triggered, a measurement consisting of sixteen samples is grabbed. The middle eight samples are used for the measurement, the four first and the four last samples are averaged (resulting in V_{before} and V_{after}) and used for the calculation for the next trigger voltage as described. After

the eight main samples are averaged and prepared for transport, they are immediately transferred to the MPU via the CAN bus. Figure 4.6 outlines this approach.

Theoretically, in adapted state, after the last measurement the reference signal voltage

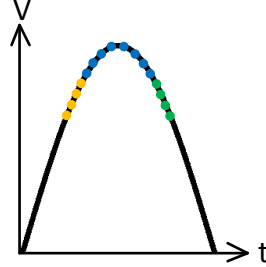


Figure 4.6: Averaging of the resolver data samples. This figure is not true to scale. Each dot symbolizes a sample consisting of five channels (sub-samples): the reference signal and the four resolver signals (coarse sine, coarse cosine, fine sine, fine cosine). The yellow, left four samples, are averaged to compute V_{before} – only the reference signal channel is considered here. After the left four samples, the main measurement starts consisting of eight samples. They are symbolized with blue dots. The resolver data channels are therefore averaged; the reference signal channel is ignored. Finally, four further samples are performed and averaged to compute V_{after} . The resolver channels are again ignored.

should be just below the trigger voltage due to symmetrical reasons. However, noise can cause a false trigger on the positive interrupt several times before the reference signal voltage is below the actual trigger voltage. For this reason, the positive interrupt is deactivated directly after it is triggered and the negative interrupt is activated. If the reference signal voltage is negative and lower than the constant negative trigger voltage of the second DAC, the negative interrupt is triggered. The MCU will deactivate it immediately and enable the positive interrupt again. Using this principle, it is possible to use comparator outputs as interrupts what usually is quite difficult due to noise. It should be clarified here, that the fifth channel of the analog-digital-converter, sampling the reference signal, is only used to calculate the positive trigger voltage (which will be additionally sent to the MPU for debugging and monitoring issues). The four channels connected to the resolver signals are used to determine their values and not used to calculate the positive trigger voltage (this would be possible, but because of their position-dependent amplitude more complex and less robust).

Runlevel 4 is only enabled if the reference signal is online. If the reference signal is spontaneously assumed to be offline while the system is in runlevel 6 or 7, a softstop will be engaged.

4.2.1.4 Tests and Performance

Tests showed that this concept provides robust results: even during raw conditions with multiple changing reference signal amplitudes, high noise and provoked DAC bus errors (that means, too high trigger voltages were sent to the DAC) the system at any time was always able to detect the state of the reference signal and to re-swing-up the trigger voltages. Although the safety timer was only implemented as safety precaution, it has been found that this timer is obligatory in *Raisting 1* due to rare but high disturbances on the signals (which only appear when high loads are switched by the *Verriegelungsgestell*). Even if the trigger voltage has been set too high, the module restarts the resolver measurements quickly. Since the initial operation in the antenna never problems have occurred.

The noise, the offset, the accurateness and the linearity of the ADC have been measured. It offered a good performance and seems to fulfill its specification – more accurate measurements have not been possible with the available measurement equipment. At least, the noise could be determined. It is below $800\text{ }\mu\text{V}$. The input module is shown in Figure 4.7.

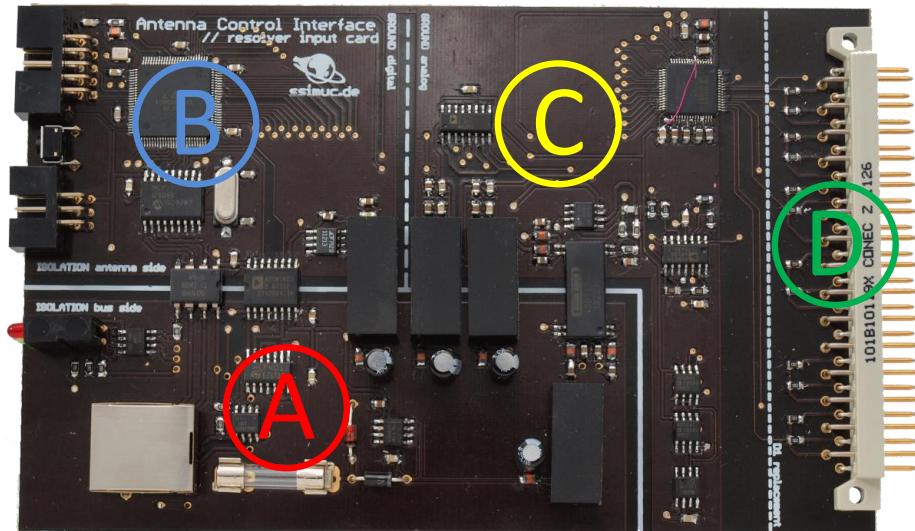


Figure 4.7: First prototype of an input module. The on-board printed white lines divide the module into four main parts. The power electronics and the HPS logic on the bus-side is labeled with A. All further parts are on the antenna-side: B symbolizes the logic and digital part, including the MPU and the CAN controller. The next part, C, contains all analog, noise-sensitive circuits including the ADC. D labels the D1 replacement.

4.2.2 Cam Disc Input Module

4.2.2.1 Task

For determining the complete azimuth position the cam disc sensor has to be read out. The cam disc switches N4 and N5 are inverted; that means that the switches close the connected circuit when they are not triggered and interrupt the circuit when they are triggered (see 4.13). Here the same principle is used as already applied for the HPS: the default signal is always defined as logic '1' (or: current flows) so that a logic '0' (no current on the line) signalizes an error or the circuit has been damaged. In other words, this concept can be used to distinguish whether the cam disc sensor is plugged out (the connected circuits to N4 and N5 are opened, and would be therefore triggered, what is not possible) or both switches are not triggered (the connected circuits are closed, what means that the antenna is arranged between $0^\circ \pm 178^\circ$). If a non-inverted switch was used, it would not be possible to determine whether the cam disc is plugged out or both switches are not triggered because in both cases the connected circuits would be opened. Each switch shows a slight hysteresis.

N4	N5	CONSEQUENCE
open	open	(not connected)
open	close	$-380^\circ \leq \phi \leq -178^\circ$
close	close	$-178^\circ \leq \phi \leq +178^\circ$
close	open	$+178^\circ \leq \phi \leq +380^\circ$

Table 4.13: Logic table of cam disc switches N4 and N5.

4.2.2.2 Hardware

The hardware of the cam disc module follows a simple setup: the module offers two cam disc jacks for N4 and N5. The jacks come with pull-down-resistors, so that an unplugged cam disc sensor can be detected correctly.

Because the cam disc sensor offers several other signals, which might be interesting for possible future extensions, there is an additional extension jack. For example, the original resolver signals can be tapped there. This jack provides a power supply, eight digital inputs, an Universal Asynchronous Receiver Transmitter (UART), an Inter-Integrated Circuit (I²C) bus and an SPI bus for possible add-ons.

4.2.2.3 Software

The module sends the current cam disc state during start-up and after each state alterations (push). The cam disc signals are debounced by software. Additionally the current cam disc state is accessible for the MPU via the CAN bus (pull). The SPS emits a softstop broadcast if N4 and N5 are opened at the same time in runlevels 6 and 7.

4.2.2.4 Tests and Performance

The cam disc module performs as expected and without any problems. The module is shown in Figure 4.8.

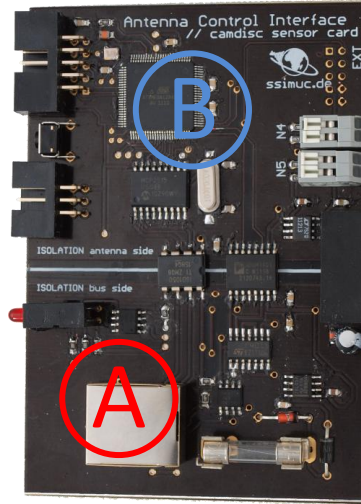


Figure 4.8: The cam disc module. The on-board printed white lines divide the module into two main parts. The power electronics and the HPS logic on the bus-side is labeled with *A*. The other part, *B*, is on the antenna-side. It contains all further circuits, including the MPU, the two cam disc switch jacks and the extension header.

4.2.3 Velocity Output Module

4.2.3.1 Task

The velocity output module receives velocity signals from the MPU, generates a corresponding analog signal and sends it to the existing antenna control electronics. To avoid damage to the antenna caused by harmful output signals (caused by too high velocity or acceleration) or broken components, the safety systems have to check constantly the velocity signal and the resulting acceleration on software (SPS) and hardware (HPS) level.

The output modules replace the SIV3 cards and therefore also have to duplicate the original SIV3 circuitry. As mentioned in the interface description in chapter 3.4.3, in order to switch between the original velocity signal and the generated velocity signal a relay is used. The output of this relay is fed to the SIV3 circuitry.

See chapter 2.5 for the definition of the velocity output signal.

4.2.3.2 Hardware

This chapter describes the generation of the nominal velocity signal step by step. Figure 4.9 illustrates this approach.

For the amplitude generation of the nominal velocity signal a 16-bit DAC is employed which is connected to the MCU. This DAC is enhanced with an external high-precision voltage source which is configured to output 2.5 V. The DAC is therefore able to output a voltage in the range of 0V to 2.5V:

$$V_{\text{DAC}}(t) = \frac{LSB(t)}{65536} \cdot 2.5 \text{ V} \quad (4.3)$$

The later circuits are designed in a way that this voltage already defines the final amplitude of the final 400 Hz voltage signal. The output of the DAC is fed to an inverter which generates a negative voltage signal. The original, positive voltage signal and the negative voltage signal are connected as inputs to an analog switch (realized as integrated circuit). The switch can therefore output the positive or negative DAC voltage signal and is used for setting the sign of the velocity and thus the direction $dir(t)$ the axis of the antenna should be moving in:

$$V_{\text{switch,direction}}(t) = \text{sgn}(dir(t)) \cdot V_{\text{DAC}}(t) \quad (4.4)$$

This signal is again fed to another analog switch as input; the other input is constantly connected to 0 V. This analog switch is an HPS actor. By default the switch routes the velocity signal to the output. If the HPS is triggered, the switch however outputs 0 V, setting the nominal velocity to 0 °/s:

$$V_{\text{HPS,actor}}(t) = \begin{cases} V_{\text{switch,direction}}(t) & \text{if HPS is released} \\ 0 \text{ V} & \text{if HPS is triggered} \end{cases} \quad (4.5)$$

In the following it is assumed that the HPS is released for clarity issues. $V_{\text{HPS,actor}}(t)$ is routed to a slope limiter. The slope limiter limits the first derivative of the velocity signal, in this example the acceleration of the antenna. In principle it consists of a (variable) resistor, a capacitor and an operational amplifier. The input voltage is connected to the resistor and afterwards to the capacitor⁴. If the input voltage changes, the voltage on the capacitor is decoupled by the resistor and limits the change of the input voltage. The slope limiter behaves according to the charging characteristic of a capacitance. The variable resistor is configured in a way that the maximal acceleration of 0.5 °/s is never exceeded. The voltage on the capacitor cannot be directly burdened and is thus decoupled by the operational amplifier in unity gain buffer mode (voltage follower). In normal operation the derivative of the velocity signal is widely identical and the slope limiter does not alter the signal. The slope limiter is therefore not regarded in the following equations of this chapter.

⁴This is the simplified basic structure of a slope limiter. In fact, the slope limiter is also extended by a bidirectional diode clipping network. The functional principle and more information about the slope limiter can be found in the ACIv1 documentation [1, p. 27] or in a technical article [4].

The resulting voltage signal is then routed to an analog multiplier. It multiplies the signal with the galvanic isolated 20 Vpp 400 Hz reference signal and then divides it by fixed 10.0 V, what finally means that the ingoing voltage signal is multiplied by the 400 Hz reference signal normed to 1 V. The resulting signal is the amplitude-modulated output signal:

$$V_{\text{out}}(t) = \frac{V_{\text{ref},20\text{Vpp}}(t)}{10.0 \text{ V}} \cdot V_{\text{HPS,actor}}(t) \quad (4.6)$$

Before it is fed to the relay, it is decoupled by another unity gain buffer. Although the HPS avoids too high voltages on the velocity output signal this unity gain buffer is supplied with a lower supply voltage as last safety barrier. If the voltage of the incoming velocity signal is too high, the operational amplifier is not able to pass this voltage and limits it to its supply voltage. The output voltage in adapted state can be summarized to:

$$V_{\text{out}}(t) = \frac{V_{\text{ref},20\text{Vpp}}(t)}{10.0 \text{ V}} \cdot \text{sgn}(\text{dir}(t)) \cdot \frac{LSB(t)}{65536} \cdot 2.5 \text{ V} \quad (4.7)$$

Using formula 2.7, the nominal velocity signal can be given as function of $LSB(t)$ and $\text{dir}(t)$:

$$\omega(t) = \frac{1^\circ/\text{s}}{1 \text{ V}} \cdot \frac{V_{\text{ref},20\text{Vpp}}(t)}{10.0 \text{ V}} \cdot \text{sgn}(\text{dir}(t)) \cdot \frac{LSB(t)}{65536} \cdot 2.5 \text{ V} \quad (4.8)$$

Beside the HPS actor, the second analog switch, the HPS monitors the velocity and the acceleration information of the velocity signal. The resulting voltage, which is emitted by the last unity gain buffer, is monitored using comparators. The comparators compare the voltage with on-board generated maximum positive and negative reference voltages. If the voltage of the velocity signal is higher than the maximum positive reference voltage or lower than the maximum negative voltage, the HPS is triggered. The reference voltages can be set using a potentiometer after the reference voltages have been determined and calibrated. Normally the potentiometers could be set too high by the user and therefore make the sensors of the HPS non-effective. To avoid this, a fixed upper and lower voltage limit for the comparator reference signal is additionally generated on the module. That results that it is not possible to set a too high reference voltage which would already harm the antenna (This is additionally ensured by the limited supply voltages of the last unity gain buffer.). However, the reference voltage has to be set properly using the potentiometer to avoid that the *Amplitudenbeschränker* (amplitude limiter) of the later antenna control circuitry is triggered.

To gather the acceleration information of the velocity signal, the slope limiter is used. The slope limiter is fed with the nominal velocity signal and outputs the actual velocity signal. The voltage difference of both signals, the resulting nominal acceleration, is calculated analog and once again compared with maximum positive and negative acceleration reference voltages which are generated the same way as the maximum velocity reference voltages. If the acceleration is higher than the permitted acceleration, the HPS is triggered. Note that the actor of the HPS does not harm the antenna: although the nominal speed of the antenna is immediately tied to $0^\circ/\text{s}$ by the HPS actor, the resulting velocity is always smoothly slowed down by the downstream slope limiter. The same

applies to an invalid nominal value due to too high acceleration: the acceleration will be measured and checked first before the slope limiter ties its output value to the input, nominal value.

By specification the relay is controlled by the *PR aktiv* signal directly. However, in worst case the control line of the relay could be broken and high-frequently switched between logic '0' and '1', resulting in quite dangerous velocity output signal which shuttles between the velocity signal and the original velocity signal (which is tied to 0 V in PR mode). Additionally, this behavior cannot be monitored by the HPS because it is behind its area of influence. To avoid this risk, a resistor capacitor (RC) filter is placed before the control line of the relay. This RC filter only allows the change of the control line signal every two seconds. If a fault causes the (pulled-down) control line being floating, the RC filter delays the triggering. One single trigger will not harm the antenna, and the MPU will detect a nominal-actual velocity mismatch and shut down the system. The card is shown in Figure 4.10.

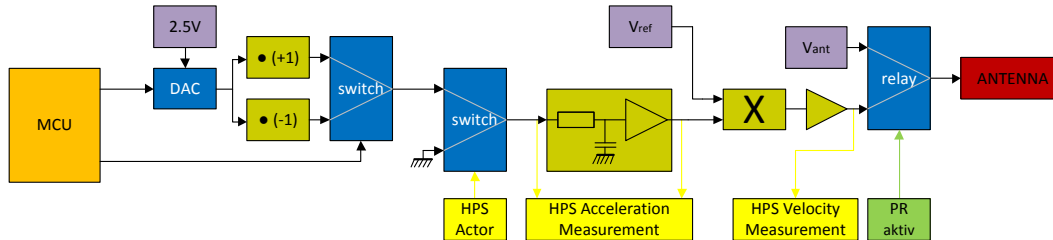


Figure 4.9: Simplified velocity signal generation circuit. The DAC outputs a voltage between 0 V and 2.5 V respectively to the commands of the MCU. This signal is inverted; a switch is used to select the positive or negative DAC voltage to determine the velocity sign. This voltage is finally multiplied with the reference signal. A slope limiter is employed for safety issues. The HPS measures the acceleration and the velocity of the output signal – if the signal exceeds pre-defined thresholds, the HPS triggers and the actor decreases the output signal to 0 V. To switch between the generated nominal velocity signal and the original nominal velocity signal (generated by the existing position controller of the antenna electronics), a relay is applied.

4.2.3.3 Software

The SPS will deny incoming velocity commands in all runlevels lower than 7 using a softstop broadcast. In runlevel 7, however, the MPU has to send a velocity request to the module at least every 20ms in order to not trigger the SPS. Referring to the term heartbeat a so-called *speedbeat* softstop broadcast will be sent if the last velocity request is longer than 20ms ago.

The SPS will monitor all incoming velocity requests in terms of velocity and acceleration.

There are hard-coded maximum negative and positive velocity values which must not be exceeded. The monitoring of the acceleration is complex because it is not possible to differentiate exactly on a 8-bit MCU on the basis of the (variable) time since the last velocity request. Therefore the following procedure has been designed: every time a velocity request is received, its alteration since the last velocity request is calculated and added to a counter. The value of the counter is decreased by the value which corresponds to the maximum acceleration every 20ms. If the counter anyhow reaches a maximum acceleration reference value, a too high acceleration has been monitored and a softstop broadcast will be sent. This concept offers the opportunity to send a velocity request every 20ms which will be monitored correctly, but it is also possible to send velocity requests (emitted by a more frequently executed control loop) in much smaller time intervals with lower acceleration between the single requests. In both cases the resulting acceleration counter value after 20ms is equal.

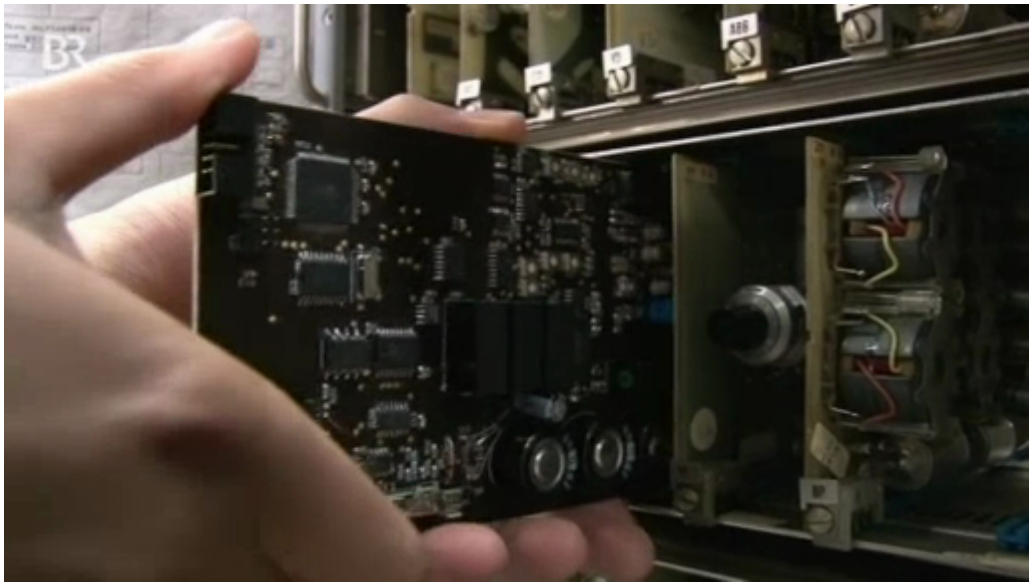


Figure 4.10: The output module prototype is plugged in in the *Reglergestell* as shown in the television broadcast *Abendschau* of the television channel *Bayerischer Rundfunk* (BR). The broadcast is supplied as podcast [5].

4.2.3.4 Tests and Performance

During the first tests, the HPS has been triggered while switching the direction of the velocity although the velocity signal voltage is always set to 0V before the direction is switched. Measurements have shown that during the transition time of the first analog switch (245 ns according to its datasheet), the output signal had been floating and therefore triggered the HPS. This leads to the conclusion that the HPS is able to react faster than 245 ns. This bug has been quickly neutralized by coupling the output of the

analog switch with a very small conductivity which does not alter the velocity signal. The whole HPS had been tested under laboratory conditions and finally in operational use. If the HPS is triggered even on full speed, the antenna slows down quickly but smoothly. It is softer than the existing antenna stop which is triggered by the *Verriegelungsgestell* when an error occurs. In fact, the velocity controllers (PI-controller) of the antenna harmonizes with the characteristic line of a discharging capacitor (D controlled system) which is used in the slope limiter, resulting in a smooth slow down without a jerk. The first prototype is shown in Figure 4.11.

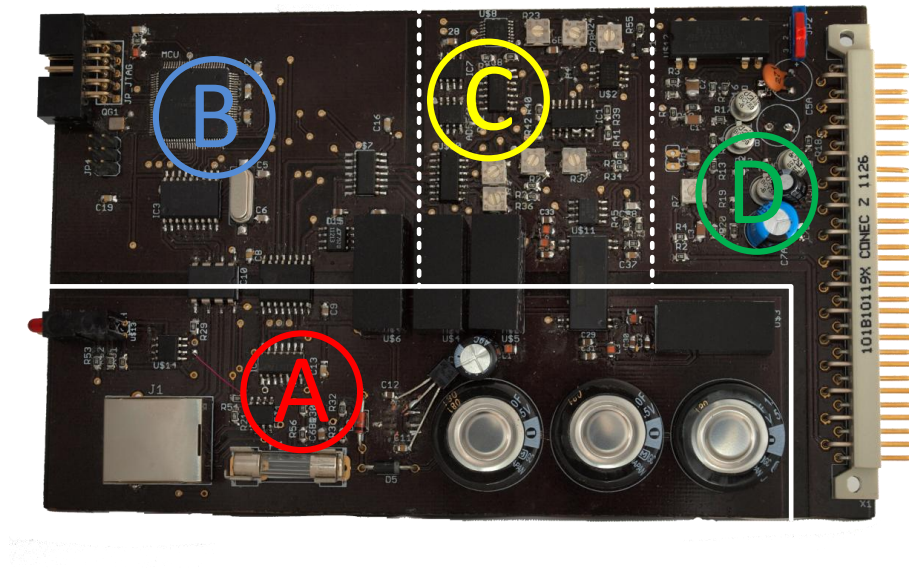


Figure 4.11: First prototype of an output module. This is the first module prototype at all. The white lines divide the module into four main parts. The power electronics and the HPS logic on the bus-side is labeled with A. All other parts are on the antenna-side. B symbolizes the logic and digital part, including the MPU and the CAN controller. The next part, C, contains all analog, noise-sensitive circuits like the nominal velocity generation and the HPS sensors and actors. D labels the D1 replacement with the additional relay. Note that part A also consists of a capacity array originally used for the UPS. This feature is dropped in the actual versions because of the reasons mentioned in chapter 4.1.2.4.

5 Supporting Hardware

5.1 Hub

5.1.1 Task

The ACI hub is used to provide the connection between all modules and the MPU. It contains basic components like the main power supply including fuse protection, the HPS pull-up resistor and the CAN termination resistor. Beside the six fixed signals of the ACI bus, the hub manages the routing of the specific individual signals to the modules. All necessary individual signals are prepared on the ACI hub: the 400 Hz reference signals for the two axis are connected to the hub. After the two signals are galvanic isolated, they are routed to the specific resolver input and velocity output modules. The *PR aktiv* signal is also connected directly to the hub where it is decoupled and galvanic isolated using an opto-coupler. The signal is afterwards routed to the two output modules. The MPU also uses two individual, outgoing signals. The first signal triggers the hub to power the main supply and therefore switch on all connected modules. The second signal triggers the hub that the MPU is ready to control the antenna; this signal is afterwards galvanic isolated using an external relay. It is fed to the antenna to activate the *PR bereit* (external velocity input is ready) signal. The PR mode is then selectable by the user or the Antenna System Interface. In addition, the hub is also equipped with some switches for debugging issues. The hub is shown in Figure 5.1.

5.1.2 Hardware

In some cases there are several input sources for a specific signal. For example, the *PR aktiv* signal can be set by the antenna and by the debugging switches, OR logic gates are used to combine them. Unused inputs of the logic gates are pulled to ground using pull-down resistors.

5.2 Bus adapter for the Main Processing Unit

5.2.1 Task

In order to connect the MPU to the ACI bus, a suitable adapter (shown in Figure 5.2) has been designed. It is equipped with a full HPS, a driver for the two individual signals and a CAN bus controller. Here, the HPS does not utilize sensors and actors, but the MPU can trigger the HPS directly and also read out its state just like every MCU. The CAN controller is connected via the SPI interface which allows a low-latency communication with the MPU.

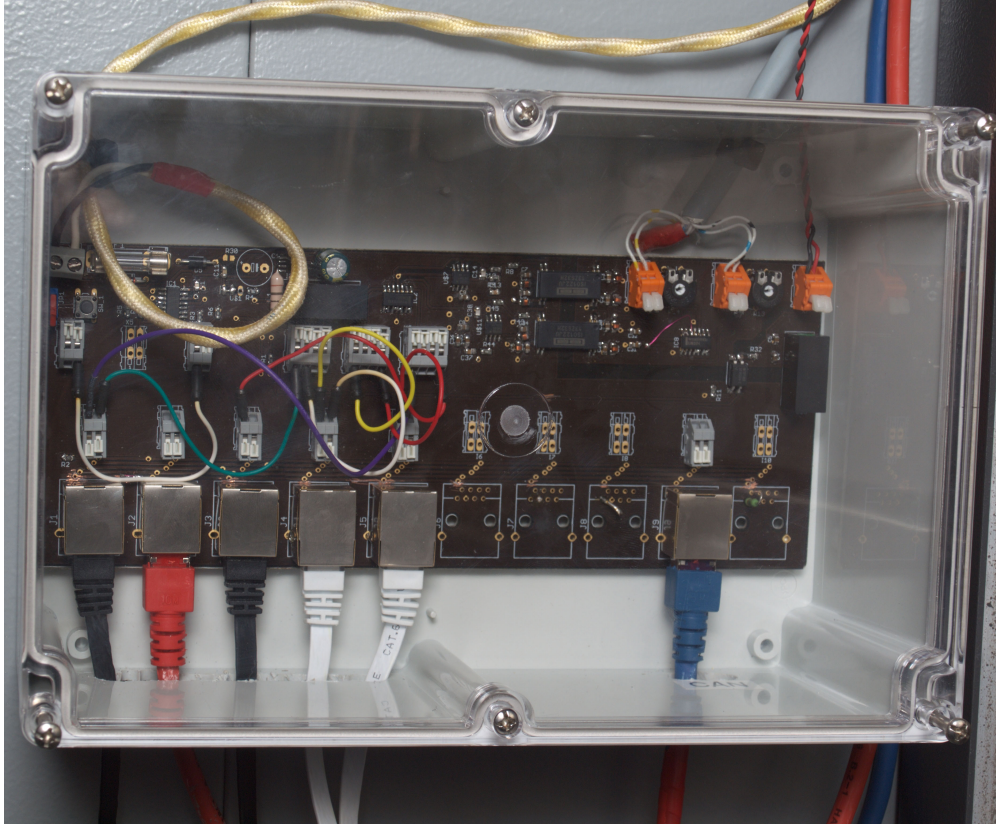


Figure 5.1: The hub of the ACI system. It allocates and routes all signals to the modules and to the MPU.

5.2.2 Hardware

Because the MPU depends on different logic levels than the complete ACI bus, all interfaces are equipped with level converters. To obtain the galvanic isolation of the whole ACI bus, the adapter decouples all interfaces between the ACI hub and the MPU.

5.2.3 Software

The Linux kernel has to support the ACI bus, so it has to be enhanced with appropriate drivers. The individual signals can be controlled with common General Purpose Input/Output (GPIO) drivers. For the support of the CAN bus, a suitable SPI driver and the SocketCAN driver had to be implemented. The CAN bus is accessible in the Linux Userspace as socket and can be used similar to a network interface.

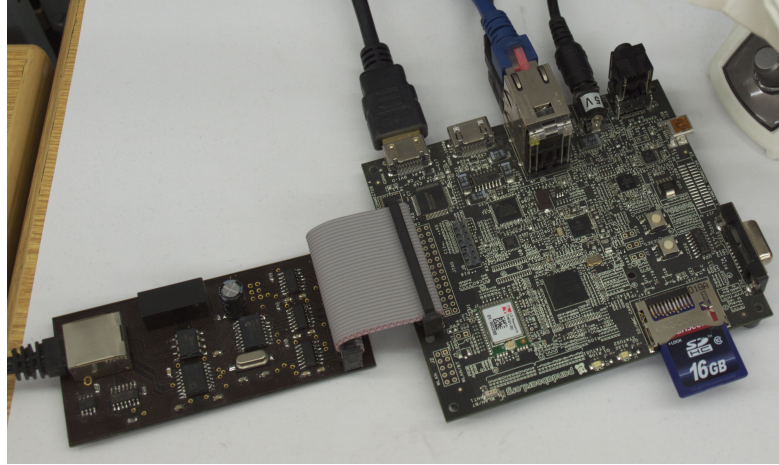


Figure 5.2: CAN adapter (*left*) for the MPU. Both are connected by a ribbon cable.

5.3 Bus terminator

The CAN bus which is routed in the ACI bus has to be terminated on both sides of the bus as illustrated in Figure 5.3. Hence, two terminating boards have been designed which can easily be plugged before the first and after the last node of the bus. If a new module is added to the bus acting as a new end node, the terminator board can be quickly attached to the new end node. This principle of using terminator boards instead of a jumper on each module which enables the terminator resistor has the advantage that at no time more or less than two terminator resistors are enabled and hence this protects the system against a misconfiguration of the CAN bus.

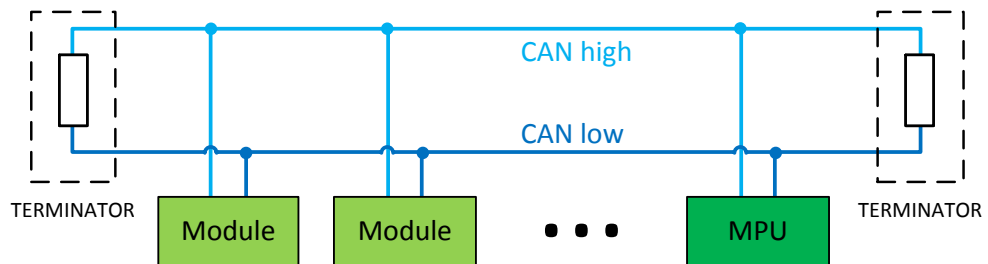


Figure 5.3: CAN terminator. The resistors are located on the bus terminator boards and have to be placed before the first and after the last node of the bus for best performance.

5.3.1 Hardware

The terminator board consists of two RJ45 jacks (bus input and output to the desired node) and a simple 120Ω resistor. The cable between the output and the bus node should be short, so that the terminators are also physically placed at the end of both sides of the bus.

5.4 Programmer

Every MCU on a module can be programmed with a JTAG programmer which can be plugged in the standardized JTAG jacks. The AVR Dragon¹ has been used as programmer.

5.4.1 Software

Usually the AVR Dragon is accessed by the AVR Studio². However, an own much slimmer programmer tool has been written (see screenshot shown in Figure 5.4). The tool is optimized for the programming of the ACI modules and therefore offers a faster and much more effective workflow. The user just has to select the module he wants to flash, and the programmer tool automatically gathers the desired module information and binary files.

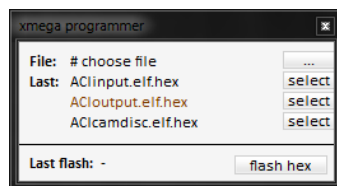


Figure 5.4: Programmer tool. It offers a simple and safe programming procedure.

5.5 Debugger

For debugging issues every module is enhanced with a debug jack. The debug jack offers an UART interface which can be used for opening a terminal session. This debug opportunity is completely independent from the ACI bus; the module only has to be supplied with power. Therefore the debug feature can also be used to test the CAN interface.

Using the debug interface several debug variables can be read out which are not accessible by the ACI bus and it is also possible to control the on-board hardware directly. For example, the DAC on the output module can be controlled by the user applying the terminal to bypass the SPS and hence to test the HPS. In order to get all debug

¹The AVR Dragon is offered by the manufacturer Atmel who also sells the MCUs.

²This programming tool is freeware and also supplied by Atmel.

functionalities, in the beginning of the terminal session the debug flag has to be enabled. Because the debug concept is similar to the one used in ACIv1, its documentation gives more details about it.

To connect the debug jack with a computer, an USB-UART adapter has to be used. In the context of this thesis, a suitable adapter board has been designed. The tool does not use the proprietary AVR Studio programmer driver but the open-source AVRdude driver which is also supported under Linux.

5.5.1 Hardware

One of the most common UART-USB converter chips is the FT232RL by FTDI. This chip has been also used for the debugger tool board illustrated in Figure 5.5 which is optimized for debugging the ACI modules (small outline, enhanced with a cable which is long enough to debug modules even if they are mounted in the racks).

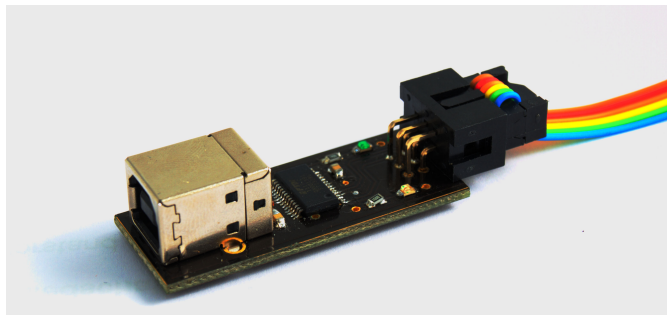


Figure 5.5: Debugging interface. It connects the modules to a personal computer.

5.5.2 Software

The terminal software (see screenshot in Figure 5.6) has been adopted from the one which has been developed and used for the ACIv1. However, the software has been improved and can handle several terminal session, allowing to debug multiple modules at the same time (advantageous for CAN message debugging).

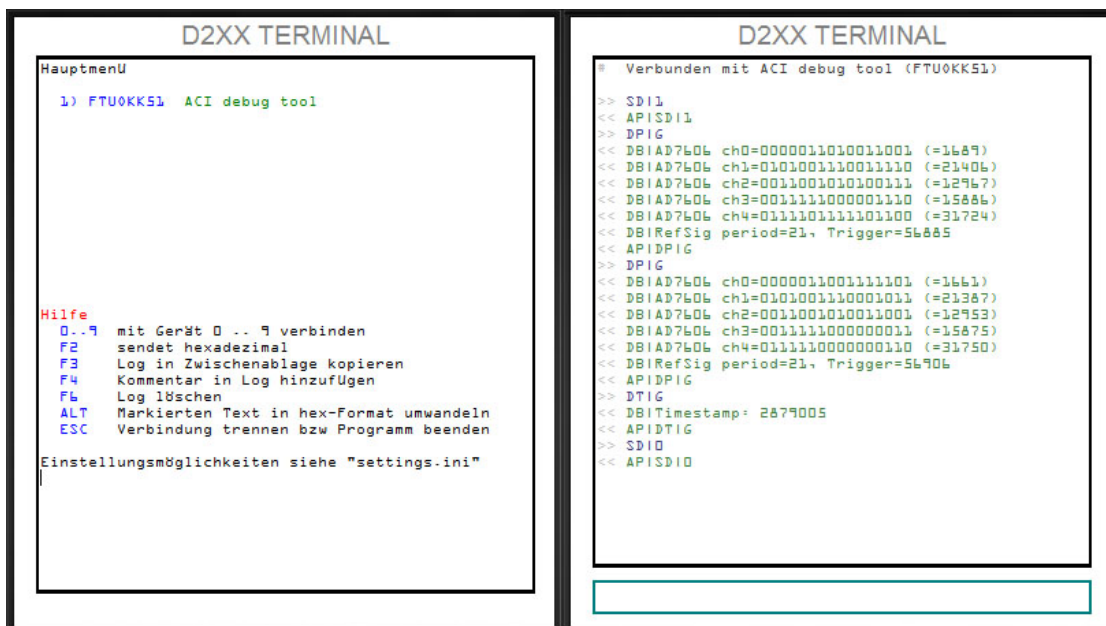


Figure 5.6: Debugging tool. On the left, the main menu is shown. On the right, an active debugging session is shown. In this example the ADC of the input module is debugged.

6 Main Processing Unit

There are several requirements for the heart of the ACI system, the MPU. First, it must be suitable for an extension which connects the MPU to ACI's data bus. Second, it should offer good performance regarding the advanced control algorithms to steer the antenna – a Floating Point Unit (FPU) is obligatory. Linux support, a network interface, solid state components for a reliable operation and cost-efficiency are further important demands on the MPU.

After some investigations it was decided to use the so-called *Pandaboard*¹, a basic single-board computer by Texas Instruments. Figure 6.1 illustrates the *Pandaboard*. It was built with similar hardware to the future upcoming high-end smartphones and offers a dual-core 1.0 GHz ARM7 processor and a (hardware-accelerated) video output. The latter is a nice-to-have feature allowing to connect a monitor, displaying status information on the antenna's status, directly to the MPU. In addition the *Pandaboard* offers an SPI interface which could be used to enhance it with a data bus node. It is currently available for about 160\$.

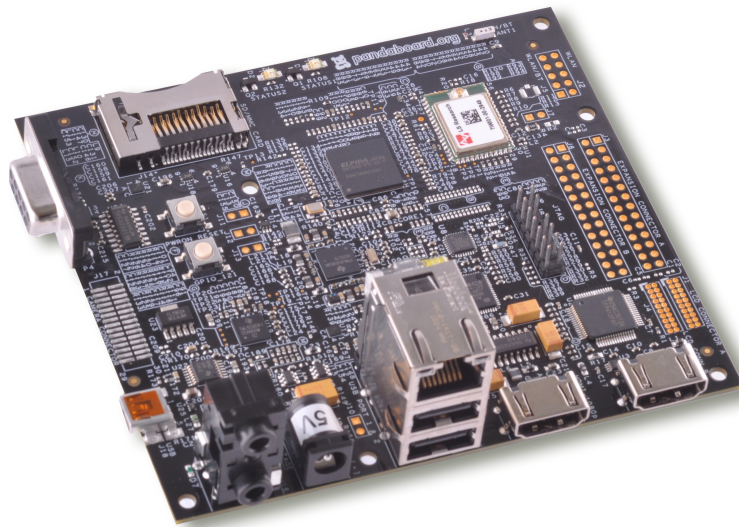


Figure 6.1: Because of its excellent performance, the *Pandaboard* is applied as MPU.

¹More information on the *Pandaboard* can be found on pandaboard.org.

As operating system Gentoo Linux has been chosen for its slinness, performance and configurability which makes it a good choice for an embedded system. The MPU processes the ACI server software. The concept of the server software is described in the following.

6.1 CAN Interface

The server software implements an interface to the CAN socket which is provided by SocketCAN. Every incoming and outgoing packet is handled by this interface: incoming packets are firstly parsed and than forwarded to the server software Intelligence in an edited form (C++ structures with all necessary information). The Intelligence allocates the incoming packets to the specific subsystems which process the message. If packets have to be sent out, all necessary information is put in the same form and routed to the CAN interface which converts and transmits them.

6.2 Intelligence

The so-called Intelligence is the "brain" or core of the server software. It conducts all subsystems like the control loop, includes the whole first-level protection system (see chapter 3.4.1) and manages the modules. Its task is to gather all available information from the modules, check their status, calculate runlevel conditions and finally, conducts the runlevel of the whole system to the user-defined nominal value as far as possible. If an error occurs, the Intelligence analyzes it and decides if the system has to be stopped or can be restarted autonomously. Due to its complexity, not all features can be described here, but are documented in the source code. The Intelligence is optimized for a reliable and autonomous operation. In principle, the user just has to request a task (eg. track a specific satellite), and the complete management of starting up/down and monitoring the system is handled by the Intelligence. If the nominal runlevel cannot be engaged, the Intelligence is able to give detailed information on that issue. The Intelligence also knows how to handle unexpected situations like unsynchronized runlevels of some modules and inconsistent external signals. It also monitors the actual velocity of the antenna and compares it with the nominal velocity which is output by the velocity output cards; a too high difference will lead to a softstop. Every anomaly is reported to the logging subsystem, even if it could be solved without canceling the actual tracking operation. Thanks to the reasoned concept the Intelligence is – despite its complexity – clearly structured.

6.3 Kinetics

Starting with runlevel 3, all input (resolver and cam disc) modules start sending raw position data. This raw data is combined and used to calculate the kinetic state of the antenna. If the data is checked for validity and all kinetic information can be calculated, runlevels 4 and higher are allowed.

6.3.1 Position

First of all, the position of the both axis is determined. The Intelligence routes all incoming position information to the kinetic calculation subsystem. Ideally, the position evaluation is straight forward: the angles of the coarse and fine resolver have to be calculated using the atan2-function and afterwards have to be transformed to the specific axis coordinate system (each resolver has an individual offset which has to be figured out). Thereafter it is just necessary to use the coarse resolver angle to determine the 5°-section where the fine resolver angle is settled. Afterwards the fine resolver angle has to be transformed to the size of the section (that means, reduce its 360°-range to the 5°-range by dividing it by 72) and added to the starting point of the just determined 5°-section. However, this approach is not possible due two main reasons:

- The coarse and fine resolver are not exactly consistent and the resolver signals are noisy. Therefore a wrong angle could be calculated, applying the mentioned procedure. The following example (also revealed in Figure 6.2) illustrates this behavior. Assuming that the coarse resolver is just before the end of a 5°-section

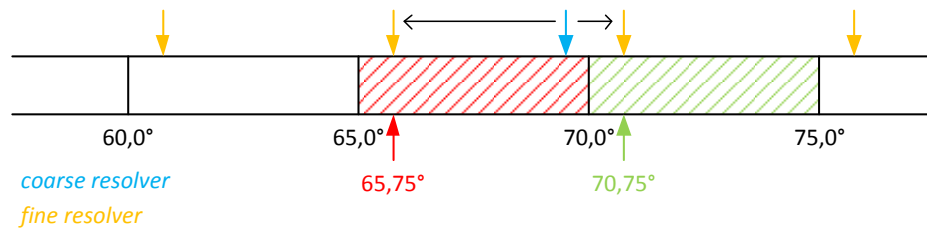


Figure 6.2: Determination of the angle of an axis. The coarse resolver points to a single specific angle, here symbolized with the blue arrow. The fine resolver points to a specific angle in a 5°-section and therefore to 72 possible angles of the complete 360°-range. If only the coarse resolver is considered to determine the 5°-section and noise occurs, the position can be falsely computed (see red arrow). A more intelligent approach is to include the angle of the fine resolver in the determination of the 5°-section. This approach determines the correct 5°-section and therefore angle of axis (see green arrow).

but the fine resolver is shortly after a new 5°-section, the position is mis-calculated by the size of a complete section (5°). This is because the mentioned procedure always assumes that the coarse resolver points out the correct 5°-section. Thus, it is necessary to determine the section by analyzing the difference between the coarse resolver angle and the two surrounding possible angles given by the fine resolver angle. Theoretically, coarse resolver errors up to 2.5° can be neutralized as outlined in Figure 6.3.

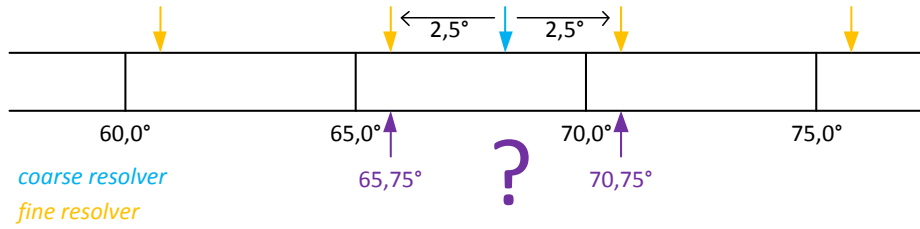


Figure 6.3: Maximal recognizable error of coarse resolver. As soon as the position mismatch between the coarse resolver and the fine resolver exceeds 2.5° , a proper position determination is no longer possible without separate measures.

- Measurements have shown that both coarse resolvers are significantly non-linear: near certain angular regions their error can be up to 6° . Additionally, the signals of the coarse resolver signals are noisy up to an error of 1° . By applying the intelligent approach to determine the angle of a axis, the noise of the coarse resolver can be neutralized (because the noise is lower than the maximum allowed error of 2.5°). The nonlinearity, however, is too high for a correct determination of the 5° -section. Fortunately the fine resolvers do not exhibit these disturbances: the corresponding noise is lower than 0.05° in relation to their full 360° range. When divided by 72 (and therefore related to the final calculated axis position noise), it is lower than 0.001° (most often even lower than 0.0005°). These measurements have been taken at several positions when the antenna was standing still. Exact measurements, also concerning the non-linearity of the fine resolvers, can be performed when the RF parts of the antenna are ready and can be used to calibrate the position of the antenna by focusing satellites with known positions. In the actual state the values can only be compared to the analog position displays of the antenna which are highly assumed to have an offset (azimuth: about 0.5°) according to some measurements of the Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR).

Regarding all issues, the final position can be determined by selecting the nearest correspondent final resolver angle at the coarse resolver angle. The following approach has been developed for this approach:

1. The incoming input signal data from the cam disc and resolver modules are validated by the SPS.
2. The angles of the coarse and fine resolvers are computed using the atan2-function on the respective sine and cosine signals.
3. The 5° -section, to which the coarse resolver points, is determined. Because of the nonlinearity of the coarse resolver, a look-up table is implemented for this purpose. The output value of the look-up table (adjusted coarse resolver angle

given in amount of 5°-sections) is already related to the native coordinate system. Assuming an example, the atan2-function computes an angle of 65°. This angle is processed by the look-up table and it returns the adjusted amount of 5°-sections, here 2.8 sections. This corresponds to $2.8 \text{ sections} * 5^\circ/\text{section} = 14^\circ$. Summarized, the look-up table neutralizes the nonlinearity of the input angle of 65°, subtracts the offset of the resolver (transforming the angle to the native coordinate system) and finally outputs the adjusted angle as amount of 5°-sections in relation to the zero point. The look-up table has been created by performing resolver calibrations. The coarse resolver of the elevation axis has been measured every 5°, the coarse resolver of the azimuth axis has been measured basically every 15°, but near severe nonlinearities local 5°-measurements additionally have been performed.

4. Beside of the position of coarse resolver, also the angle of the fine resolver has to be transformed to the native coordinate system. This is done by simply subtracting the offset of the fine resolver. At this point, the zero point of the native coordinate system, the coarse resolver position (that means 0 sections) and the angle of the adapted fine resolver position match.
5. The resulting, adapted fine angle is now considered as an angle in the range of 0° to 360°. Because 0° of this angle matches an integer of the sections, 360° corresponds to the respective next section integer. Figure 6.4 illustrates this view.

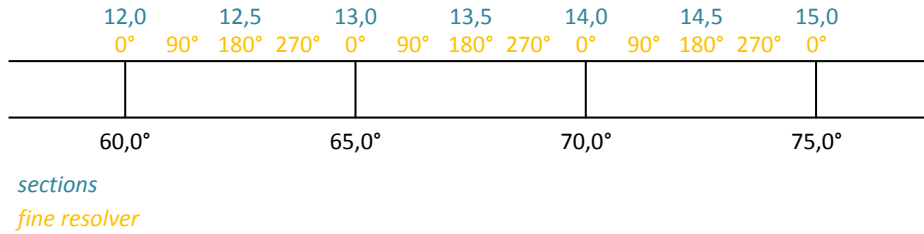


Figure 6.4: Relationship between sections and the fine resolver. Integers of the sections matches to zero points of the fine angle. The fine resolver angle is considered as an angle between 0° and 360°.

6. As preparation for the next step the output value of the look-up table, the amount of 5°-sections, is split in two parts: the integer and the rest of this value. The integer of the output value, in the above mentioned example 2, is called frame. The rest, here 0.8, is called sub-frame.

7. This step determines the correct frame using the section information of the coarse resolver and the fine resolver angle. This is done with the following pseudo-algorithm:

```

    frame ← integer(section)
    subframe ← floor(section)
    if subframe > 0.75 ∧ anglefine < 90° then
        frame ← frame + 1
    end if
    if subframe < 0.25 ∧ anglefine > 270° then
        frame ← frame - 1
    end if
    return frame

```

8. Now the frame can be combined with the fine angle to get the final position:

$$position = frame \cdot 5^\circ / section + \frac{angle_{fine}}{72} \quad (6.1)$$

In case of elevation this position is final and returned by the position calculating algorithm.

9. The coarse azimuth resolver reveals another anomaly beside its noise and non-linearity: near a certain angle the resolver angle leaps about 3° spontaneously. This anomaly cannot be neutralized with the current angle calculation algorithm because it exceeds the maximal tolerant error threshold of 2.5° and therefore causes 5° leaps. A workaround has been implemented; in the certain angle range the angle is interpolated using the fine resolver and the angle of the coarse resolver is ignored. This procedure requires that the system is starting outside of this range. If this is not the case (for example during an autonomous tracking), the coarse resolver is used for determining the actual position after all and used as interpolation starting point. If the resolver is outside of the critical range and it reveals that the position is wrong due to a false starting point, the position is corrected automatically (that is because outside of this range the normal calculation procedure is used). If the antenna is again in the critical range, the system is able to determine the actual antenna position correctly using the interpolation without any problems. Summarized, the resulting value is treated with a position-dependent anti-flicker filter in this step.

10. The last step combines the information about the axis position (considered in a range between -180° and $+180^\circ$) and the cam disc state to get the extended range of the azimuth axis from -380° to $+380^\circ$. The following pseudo-algorithm realizes this approach:

```

if  $N4 = triggered \wedge position > -20^\circ$  then
     $position \leftarrow position - 360^\circ$ 
end if
if  $N5 = triggered \wedge position < +20^\circ$  then
     $position \leftarrow position + 360^\circ$ 
end if
return  $position$ 

```

The resulting angle is final and will be returned.

The final accuracy depends on the fine resolver. According to the passed measurements, the calculated angle is very stable and does not require a filter for stabilization. Note that therefore the only position averaging is done by the resolver input modules when the positive amplitude of a sine period is sampled eight times.

6.3.2 Velocity

As far as the position calculation is performed properly, the velocity can be determined as derivative of the position. To avoid quantization noise, a moving average filter is used.

6.3.3 Acceleration

As soon as the velocity can be calculated and the moving average filter is fully operable, the acceleration can be determined as derivative of the velocity. Once again a moving average filter is applied for the acceleration data. If this moving average filter is fully operable and all other values can be calculated, the kinetic data, among the position data, is available and runlevel 4 is allowed. If a critical problem occurs in an active runlevel (6 or 7), a softstop will be triggered by the SPS.

6.4 Control Loop

In runlevel 7, the control loop is executed directly after the kinetic calculation. In the current configuration the control loop execution frequency is 100 Hz. Like the kinetic calculation algorithm, both axis are processed in the control loop at the same time. Under the condition that the resolver modules also sample negative amplitudes, the whole system is able to execute the control loop on every new measurement, resulting in a control loop execution frequency of 800 Hz ($f_{\text{ref}} \cdot 2$). The CAN bus will be loaded by around 30 % in this case which corresponds to 300 kBit/s. The control loop has been designed in a dynamic way, that means that it does not afford that the execution frequency and the time intervals between two executions have to be constant. The time interval since the last control loop execution is measured every time for the processing

of the derivations, controllers, filters and safety systems only few changes would have to be performed to change the actual control loop execution frequency. However, a control loop execution frequency of 100 Hz is already fast (and precise) enough for a sluggish controlled system.

The control loop consists of several functional blocks, outlined in Figure 6.5. The input of the control loop is realized using so-called source blocks. A source block is a subsystem which generates the nominal trajectory of the antenna. A multiplexer allows to switch between all available source blocks. Because the source blocks may contain invalid trajectories (that means, the trajectory provides invalid kinetic demands to position, velocity or acceleration) and the multiplexer may switch spontaneously, the output of the multiplexer is routed to a trajectory generator which filters the input data and generates a valid trajectory. This valid trajectory is routed to an appropriate controller. The controller compares the input trajectory with the actual kinetic values of the antenna and outputs the nominal velocity. If the velocity is given in degrees per second, the velocity will be calculated to its respective voltage using the signal conversion block. The voltage signal is then routed to the omega filter block. The omega filter block again checks if the output signal fulfills all kinetic conditions. Directly after the omega filter the voltage signals are sent to the velocity output modules via the CAN bus. Each block is outlined in the following.

6.4.1 Selectable Input Sources

This source concept has been developed in a way which allows adding or changing a source block very efficiently. Further projects, like autotracking, can insert their control input data here. Every source has read-only access to the actual kinetic values if the source trajectory depends on it. If the source blocks are requested to output the actual nominal trajectory, they have to send several information. First of all, they have to provide their status. The status describes if the source is still ready or the trajectory ends. This can happen, for example, when the satellite, which is tracked by the NASA/NORAD Two Line Elements Format (TLE) source block, is no longer visible or a source detected an error. This parameter is interpreted and handled by the multiplexer. Another return value outputs the trajectory with any desired unit. That means the type of output (like position request or velocity request) can be selected optionally. This allows the source blocks to use different controllers. In the current implementation, the following output commands are supported:

Stop This return command requests the control loop to slow down and stop the specific axis. This is the only output command which is not attached with any value and does not request an own controller.

Raw [mV] A return command for debug purposes. The source can set directly the desired output voltage, all controllers are bypassed. This type can also be used to test new experimental controllers which are implemented in the source block and offers direct influence on the voltage signal output. The omega filter, however, is still active and cannot be circumvented consciously.

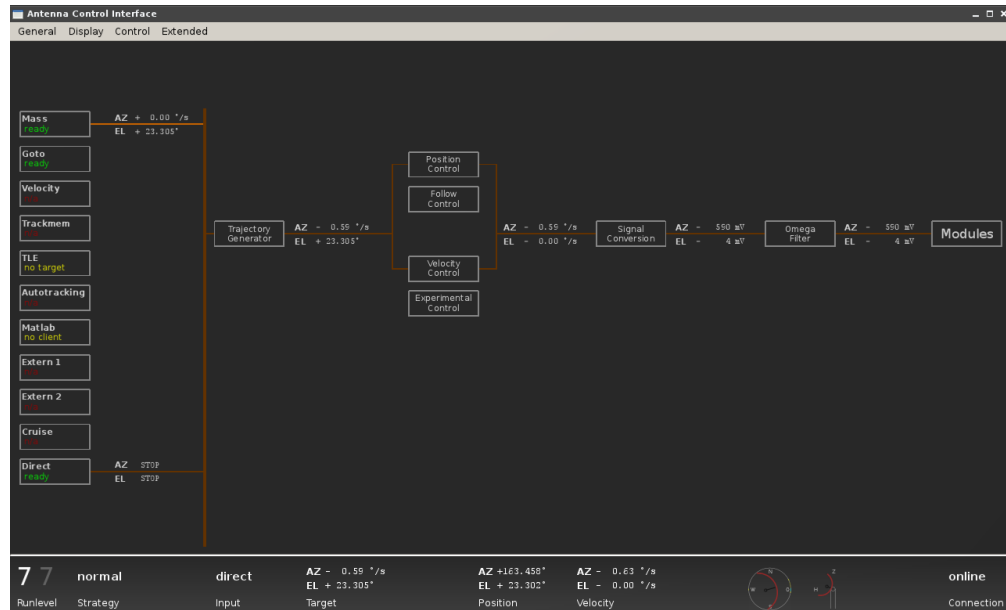


Figure 6.5: Control Loop Overview. This is a screenshot from the client software which is equipped with a graphical user interface and displays the actual state of the control loop in real-time. The control loop is processed from left to right: firstly, the source blocks output the nominal trajectory. The multiplexer selects the desired source block and relays the trajectory to the trajectory generator. The adjusted trajectory is afterwards passed to the matching controller. The output of the controllers is filtered with the omega filter before the velocity commands are sent to the modules.

Position [°] If this return unit is selected, the control loop uses a position controller to move the axis to the desired position.

Track [°] The track unit is similar to the position unit. However, it tells the control loop that the target is moving and therefore a tracking controller is used which uses an integral term for small position errors.

Velocity [°/s] This return unit demands the control loop to interpret the value as desired velocity and thus a speed controller shall be used on this output.

Note that the values have to be returned twice, once for each axis. This means that the source may use separate controllers for each axis.

If the user steers the antenna manually using the client software, it is possible to enable a safety timer. If it is enabled, every fixed time, eg. five seconds, a new command has to be received. If the connection to the client software is interrupted this timer will order the stop command and the antenna is slowed down. However, automatic tracking operations are not dependent from the connection to the client software due to operational safety issues. Thus, if the connection is lost in a tracking or another autonomous operation,

the ACI server software still performs its tasks. This is no safety loss because the ACI will stop the antenna if it reaches its position boundaries.

6.4.1.1 Mass

The so-called mass source is the default source. The only task it has, is to slow down the antenna and stop it. Its name is derived from the fact that the antenna control electronics interpret 0 V as velocity of 0 °/s. This source also holds the actual position: If the actual axis velocity is higher than a pre-defined threshold, the mass source requests the velocity controller to set a velocity of 0 °/s. If the antenna is slowed down and the actual velocity is below the small threshold, the mass controller stores the actual position and now commands the position controller to hold this position. This avoids that noise causing a slight movement of the antenna.

This source outlines the advantage that both axis can use different controllers: one axis might be moving slower already and the position controller can be engaged earlier.

6.4.1.2 Goto

This source uses the position controllers to move the antenna to a user-defined position. Not both axis target positions have to be set up; if one axis target is disabled the source just requests the stop for this axis. Hence, an axis can be moved alone. For example, the user can use the Goto source to move the antenna to specified coordinates like 160 ° in azimuth and 25 ° in elevation.

6.4.1.3 Velocity

This source is very similar to the Goto source, but requests a user-specified velocity. Again, it is possible to move a single axis. This source can be used to move the antenna manually and supports permanently altering target values, for example in case a joystick is used to set the target velocity values.

6.4.1.4 Trackmem

ACIv1 used a tracking memory (trackmem) as interface to the TLE interpretation software Trackman. Trackman sent out position data every second which has been stored and interpolated by the tracking memory. For compatibility issues, it was formerly planned to implement trackmem in the second version of the ACI too. However, in context of the Phoenix project² a TLE interpreter has been developed which can be implemented directly as source block and offers therefore much higher accuracy, performance, functionality and also reliability (no external computer and network connection are required). Nevertheless the trackmem source block is kept reserved.

²More information on the phoenix project by SSIMUC can be found under ssimuc.de/content/pmwiki.php?n=Main.Phoenix

6.4.1.5 TLE

The TLE source block implements and controls the Phoenix TLE interpreter – its functionality is outlined detailed in the Phoenix documentation. This source block also enhances the TLE interpreter with an azimuth trajectory start point search: the azimuth angle output of the TLE interpreter is arranged between 0° and 360° while the azimuth axis supports a range of -380° to $+380^\circ$. The start point search analyzes the whole upcoming trajectory, calculates its minimal and maximum azimuth positions and determines which start points can be selected without violating the azimuth range. If more than one start points can be chosen, the trajectory search takes the nearest start point.

Beside TLE tracking, this block also supports the tracking of the moon, the sun, stars and Earth-Centered Inertial (ECI) coordinates.

6.4.1.6 Autotracking

Another source block is reserved for the autotracking feature. In context of another project an autotracking feature is being developed which is able to determine satellite target positions by evaluating the incoming electromagnetic waves. Due the computing power of the MPU, the autotracking feature could be implemented directly in this source block. Alternatively, it is possible to enhance this source block with a network interface which is accessible by the autotracking processing unit.

6.4.1.7 Matlab Interface

One source block also offers a Matlab interface. New experimental trajectory sources can be implemented in Matlab and used with this source. As interface a network connection is provided which allows Matlab being run on the MPU or on an external computer in the local network. Because every source block can also read out the kinetic data, it is also possible to implement the whole control loop in Matlab. To bypass the existing controllers the raw velocity value has to be returned. The latency due to the network connection and Matlab routines will produce a minimal control error. Using the kinetic data information it is also possible to monitor and log all control-related values in Matlab. The Matlab source block has not to be active for this purpose.

6.4.1.8 Extern

This source block implements the same network interface as used in the Matlab source block for compatibility issues. The external source can be used to read in external trajectories, eg. supplied by third parties which will control the antenna prospectively. The external source block is available twice, called "Extern 1" and "Extern 2".

6.4.1.9 Calibration

If the antenna has to be (auto-) calibrated, the calibration algorithms can be implemented in this source block.

6.4.1.10 Direct

For testing and debugging issues the direct source block is available: it is very similar to the velocity source block but returns the raw voltage signal and bypasses therefore all controllers.

6.4.2 Multiplexer

The multiplexer selects the source whose trajectory shall be routed to the controllers. The user can set the desired source. However, if the source triggers that it is no longer active, the mass source will be selected automatically. The multiplexer only selects one single source, it is not supported to select different sources for each axis consciously. If a source demands to stop one axis (returns the stop unit), the multiplexer pipes the mass source output of the specific axis to the controllers. This is the only case where more than one source block can be active at the same time³.

6.4.3 Trajectory Generator

The trajectory generator receives the desired trajectory from the multiplexer and generates a nominal trajectory which fulfills all kinetic requirements on the trajectory and therefore can be tracked by the antenna. Although the later used omega filter does a similar job, this trajectory generator has to be used to avoid spuriously increasing I-terms of the controllers. The principle behind the trajectory generator is:

- do that, what the user wants, as fast and precisely as possible
- but without harming the antenna

This means that the framework conditions of the antenna must not be violated. For smooth operations it is possible to decrease the maximum acceleration. The trajectory works command type dependable and supports position, velocity and raw signals. If a position controller shall be used, the position is limited to the range of the specific axis. If raw signals are processed they will be limited later by the omega filter, but if the axis is near its end the raw signals will be limited already here. The more complex limitation is the velocity limitation, because the position range must not be violated, neither the maximum velocity nor the maximum acceleration. For example, the maximum velocity of the azimuth axis is in principle set to $+1.5^\circ/\text{s}$, but it is decelerated near the higher end of the slewing range. It is $0^\circ/\text{s}$ at the position of $+375^\circ$, that means it is not possible to move the antenna beyond this angle. If anyhow the position of the azimuth axis gets beyond this value, the maximum velocity is negative what means that the antenna will automatically drive backwards until it reaches again $+375^\circ$. The limiter was realized by calculating maximum and minimum velocity values for each of the three aspects for the

³If it would be necessary that both axis are controlled by different sources, it is possible to extend the system by a new source which pipes the trajectory data from two other sources, merges it and outputs it to the multiplexer.

actual control loop acceleration. The intersection is calculated and compared with the desired trajectory; if the desired trajectory exceeds or underruns the possible velocity intersection it is limited to its nearest possible values. If no intersection could be built, a compromise is made with the highest priority assigned to the maximum and minimum velocity.

This is a complete new approach in comparison to ACIv1 where all input sources have been combined in a single TLE tracking controller which had to carry about valid start-up trajectories by itself – the sources of this version of the ACI may just output their desired trajectories and the trajectory generator will move the antenna to the current trajectory point if it is distant with maximum allowed velocity (using maximum allowed acceleration).

This approach limits the actual kinetics (position, velocity, acceleration). Formerly, the trajectory generator was planned as 5th degree polynomial generator which can also limit jerks. This approach has been deprecated because the controllers have to harmonize with the trajectory generator and a 5th degree polynomial does not output a smoother trajectory than the actual trajectory generator and leads to additional significant problems. Due to the antenna electronics the output trajectory will be distorted, no matter which of both generators is used. If eventually the antenna power electronics is controlled directly, it makes sense to discuss an implementation of a polynomial generator.

6.4.4 Controllers

The nominal trajectory is fed to the controllers here. The mode of the trajectory decides which controller is selected. Note that the controllers are optimized for a safe and very smooth operation in development phase. The control parameters can be adjusted when the position calibration is finished.

6.4.4.1 Position controller

The position controller receives the nominal trajectory and the actual position as inputs. It determines the error and returns a velocity as output. The position controller is realized as standardized P-term controller and is used for goto-commands. Although this controller can be used for tracking, a P-term controller will produce a slight control error for moving targets (about 0.05° in the actual parametrization when tracking a Glonass satellite).

6.4.4.2 Tracking controller

The tracking controller is based on the position controller but activates an I-term when the position error is small, offering a lower control error. Because the velocity control electronic of the antenna already uses an I-term, this I-term has to be small-weighted to avoid interferences.

6.4.4.3 Velocity controller

The velocity controller is used if the source outputs nominal velocity data. The controller compares the nominal velocity data with the actual velocity and outputs a new velocity which is assumed to produce the nominal velocity.

6.4.5 Signal conversion

After the controllers output the new velocity data, it is processed in the signal conversion block. The velocity data will be converted to a velocity voltage signal. As mentioned before, the characteristic curve between velocity and voltage signal is designed to be linear according to the documentation. This could actually be confirmed in test measurements and therefore, a linear conversion has been implemented. As soon as exact position calibrations can be done with the help of a stable and constant external signal source, it is recommended to measure and review the characteristic curve exactly. If the sources output already velocity raw data, this block is bypassed.

6.4.6 Omega Filter

The omega filter is the last filter which is processed on the MPU and is only fed with (raw) velocity voltage signals. These signals are evaluated – similar to the velocity limiter of the trajectory generator – respectively to the position, velocity and acceleration. The omega filter is parametrized to limit the velocity stronger than the SPS on the module MCUs to avoid that scarce invalid velocity requests can pass the omega filter but not the SPS and therefore cause a softstop.

6.4.7 Modules

After all, the output of the omega filter is sent directly out on the CAN bus and processed by the modules. The execution of the control loop is completed.

6.5 Network Interface

The whole functionality of the MPU is accessible and controllable via a network interface, enabling a remote control of the antenna. It is possible to activate functions and set parameters, and it is also possible to read out status and debug variables. These variables are checked for changes every 20 ms – if a change occurred, the new value is sent out to all network clients. In the current configuration about 220 variables are monitored. To offer an easy and flexible functionality to monitor these variables and set functions a suitable interface has been developed. To use this interface, each subsystem, like the Intelligence or the kinetic calculator, has to register at the interface and tell its ID (abbreviation of the subsystem). The interface will then allocate a so-called netnode to the subsystem. A netnode provides the whole network system. If the subsystem wants a variable to be automatically checked for changes and be accessible (read-only) via the network, it can use a single simple function offered by the netnode:

```
appendUpdateAttribute(shortNameOfVariable,pointerToVariable);
```

This function determines automatically the type of the variable and adds it to an update list. Every 20 ms the interface checks the update list items for changes (using a suitable buffer variable for each item) and sends it out to all network clients if actually a change occurred. In the actual configuration, however, the timer for checking for changes is set to 500 ms for debugging issues (using a console, updates which are sent every 500 ms are better human-readable).

This approach takes the advantage for a very fast workflow and makes debugging easier. Normally, the programmer would have to implement a monitor function and buffer variables for each individual variable. Every time an update shall be sent out to the clients, the value of each altered variable is attached with the subsystem ID and the name of the variable. Figure 6.6 illustrates an exemplary update procedure.

A similar approach is used to set variables and functions from the network. A request has to consist of the subsystem ID, the name of the variable or function and finally a value. The interface can route the request to the specific subsystem using the ID and call the request event handler. The request event handler processes the request and returns the success of the execution or, if the execution was not successful, an error code. This feedback is now routed back to the network client.

The network interface supports several clients, but only one client may control the antenna. Thus, a user management system has been implemented with the following user right levels:

Inactive The user is connected to the antenna but is not able to set or receive control data.

Spectator The user will be informed about all changes. However, he will not be able to set control data.

Operator The user will receive all updates and can access all control functions and parameters. Only one user can obtain this user right level.

Administrator In case an operator controls the antenna but its software freezes or the control of the antenna should be passed on to the administrator, he or she can degrade the actual operator to a spectator and gain antenna control by himself or herself.

If a new client connects (that means, a user gains an user right level higher than "inactive"), the network interface will send all available information to this client. After that, the client will get incremental updates as explained.

Furthermore, each subsystem is able to emit event messages. This event messages may occur spontaneously and are also sent to all network clients. An example is the occurrence of a softstop: the event will be attached with all available detailed information and then emitted.

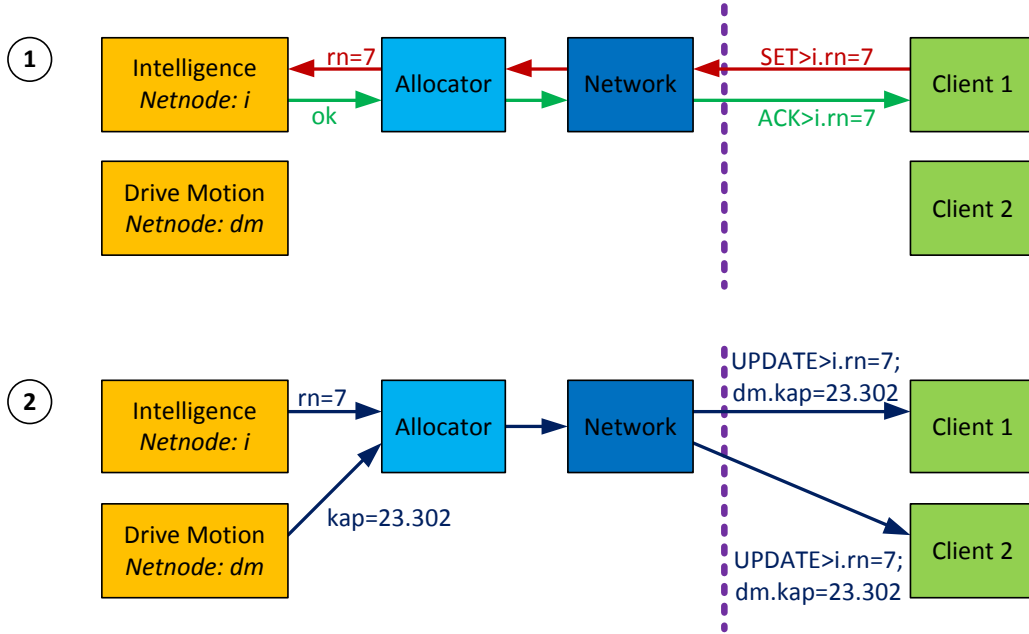
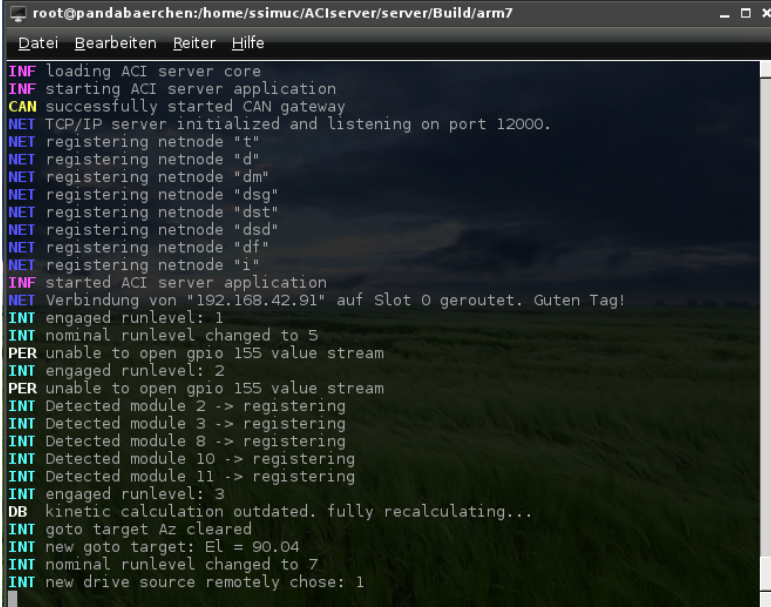


Figure 6.6: Server Networking. This example shows how a client request is processed and the update routines are broadcasting new values. (1) First, a client with appropriate rights requests to change the nominal runlevel to 7. Because the Intelligence manages the runlevels, this request is attached with the ID of the Intelligence (*i*). This request is sent to the server software via the network in the request format. The network subsystem routes this request to the so-called Allocator which relays the request to the Intelligence. The Intelligence processes this command and returns it if the command could be processed successfully or otherwise the error code. This return value is sent back via the Allocator and the Network. In this example, the runlevel could be changed successfully. (2) On the next update event, all subsystems are requested to emit new changes. Due to clarity only two values are changed: the nominal runlevel and the position of the azimuth axis. These values are sent to all clients in the update format.

6.6 Logging Functionalities

The server software comes with some logging functionalities. All variables, which are registered at the network interface, and all events can be logged. The software also outputs additional information about the actual status, events and important incoming requests to the console (see Figure 6.7). Also these messages can be logged.



```
root@pandabaerchen:/home/ssimuc/ACIserver/server/Build/arm7
Datei Bearbeiten Beiter Hilfe
INF loading ACI server core
INF starting ACI server application
CAN successfully started CAN gateway
NET TCP/IP server initialized and listening on port 12000.
NET registering netnode "t"
NET registering netnode "d"
NET registering netnode "dm"
NET registering netnode "dsg"
NET registering netnode "dst"
NET registering netnode "dsd"
NET registering netnode "df"
NET registering netnode "i"
INF started ACI server application
NET Verbindung von "192.168.42.91" auf Slot 0 geroutet. Guten Tag!
INT engaged runlevel: 1
INT nominal runlevel changed to 5
PER unable to open gpio 155 value stream
INT engaged runlevel: 2
PER unable to open gpio 155 value stream
INT Detected module 2 -> registering
INT Detected module 3 -> registering
INT Detected module 8 -> registering
INT Detected module 10 -> registering
INT Detected module 11 -> registering
INT engaged runlevel: 3
DB kinetic calculation outdated. fully recalculating...
INT goto target Az cleared
INT new goto target: El = 90.04
INT nominal runlevel changed to 7
INT new drive source remotely chose: 1
```

Figure 6.7: Console output of the server software. A typical starting-up procedure was performed; the corresponding output messages are shown in this console.

7 Client Software

7.1 Concept

The client software is the counterpiece of the server software. It receives all status variables and events and displays them graphically but also comes with all necessary control panels to access all functions of the server software. It can be run on the MPU locally or on the client machine, eg. a Notebook, a workstation in a control center or on a Handheld PC, etc.

The client software is based on the popular Qt¹ multi-platform framework. Software which is based on Qt can be compiled for Microsoft Windows, Linux, MacOS and also for some mobile phone platforms like Windows Mobile, Symbian or Android. This allows the ACI client software to be run under several operating systems and offers a high flexibility².

The client software supports external input devices such as joysticks to drive the velocity of the antenna manually. If the client software is compiled for a mobile phone, the integrated accelerometer may be used for this purpose.

Basically, the client software exists of one main window and several widgets. Each widget controls specific functions or displays specific information. The widgets are sizable and can be displayed in the main window or opened as single windows; offering the possibility that the widgets can be arranged and layouted by the user and also displayed on different monitors – thus, it is possible to layout an individual status screen on an external monitor for supervision or operational use. Because a new widget is opened as new instance, it is possible to open the same widget several times (to be shown on several monitors or mounted in several parent widgets). This approach affords an information manager. When a new widget is created, it is registered in a subscription list (and, of course, unregistered when it is closed). Every widget on the subscription list will be fed with information updates which are sent from the server software as illustrated in Figure 7.1. Because a newly opened widget has no initial information data available, the manger will send all buffered initial data to the new widget. This initial data is cached on base of the whole already received data from the server software (initial data and incremental updates). The manager and widget-concept is optimized for a simple apposition of further widgets which may be developed in future works.

¹Please see qt.nokia.com for more information.

²If the client software should be compiled for a mobile phone, little platform-dependent changes have to be performed

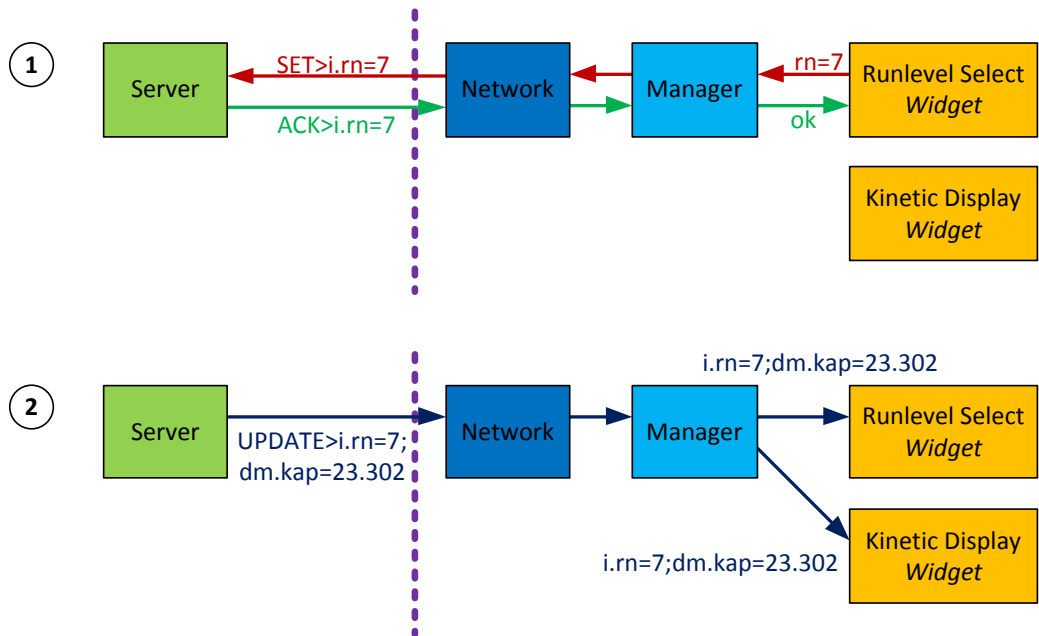


Figure 7.1: Client Networking. This example outlines the networking of the widgets on client side. There are some widgets offering control functionality and some widgets offering pure displaying functionality. (1) If the user changes the runlevel, he or she can use the Runlevel Select widget. As soon as the user has chosen the desired runlevel, the widget sends the request to the Manager which generates the request format. The Network subsystem transmits the command to the server software and receives the return value. This return value is relayed back to the widget. (2) When the server software emits an update sequence, the Manager receives the new values and relays them to all opened widget instances. The widgets can now process these data and display it graphically, for example.

7.2 Functionalities

Figure 7.2 exposes some main features of the client software.

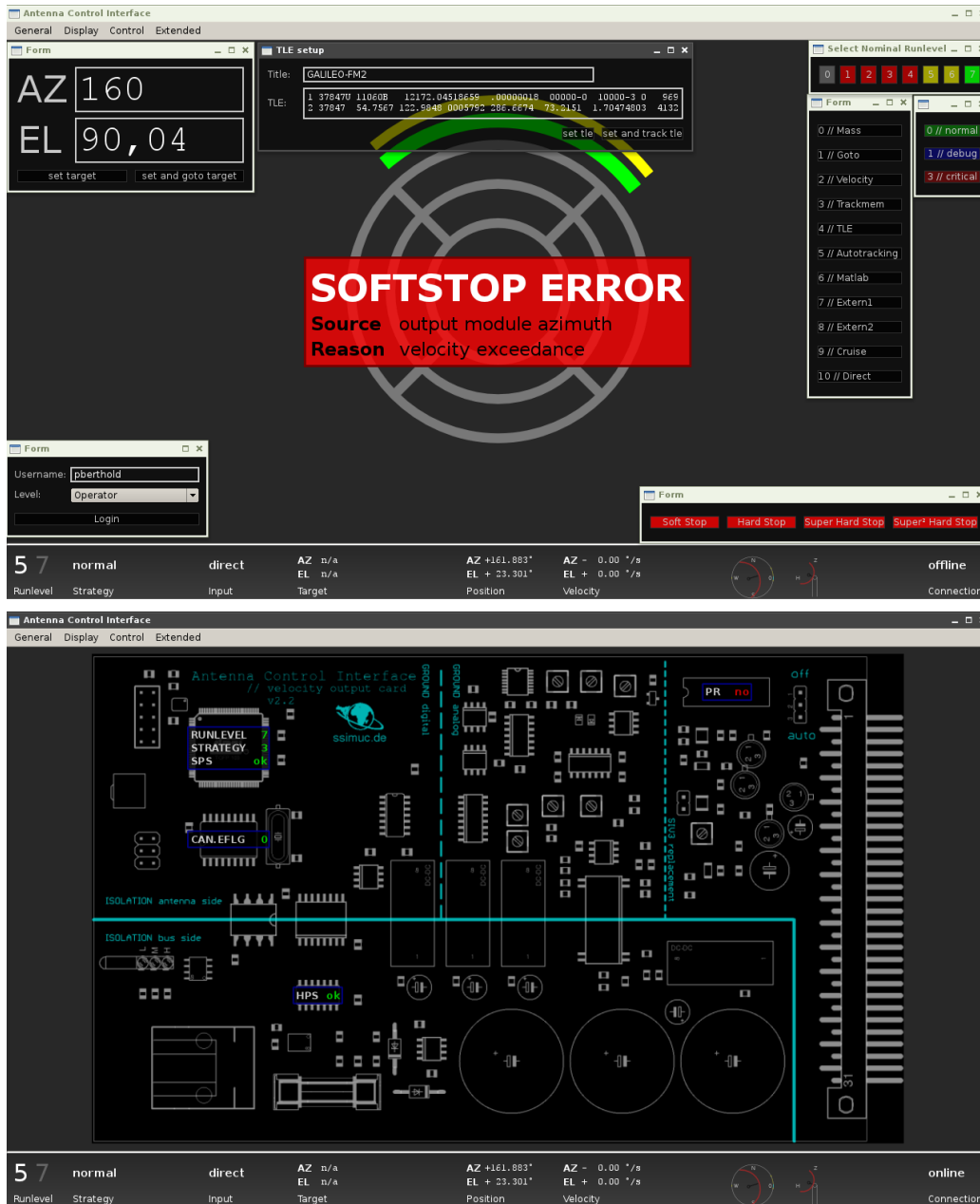


Figure 7.2: Client software. The upper screenshot shows some of the control widgets and a softstop event message. The lower screenshot shows the debugging functionality of the client software; here the conditions of the critical components of the output module are displayed.

8 Tests and Performance

In this chapter the testing of the ACI system is described in detail and the performance and capabilities of all position-related sub-systems is presented.

8.1 Estimation of Positioning Errors

8.1.1 Resolver Signal Attenuation

The resolver attenuation on the $V_{\sin}(t)$ and $V_{\cos}(t)$ output signals is assumed to be identical in the previous chapters. In fact, there are two reasons which actually may cause a different attenuation:

- Different induction attenuation directly caused by the resolvers
- Imprecise voltage dividers on the D1 card

The first point is completely reliant on the quality and accuracy of the resolver and not under influence of the ACI. The second point depends on the accuracy of the voltage divider resistors. Assuming that the resistors are ideal, Figure 8.1 and Equation 8.1 outline the function of a voltage divider.

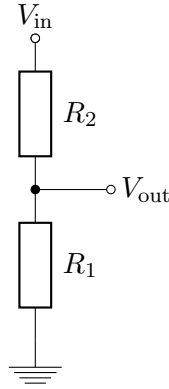


Figure 8.1: Voltage divider as used on the D1 card.

$$V_{out} = V_{in} \cdot \frac{R_1}{R_1 + R_2} \quad (8.1)$$

The resistors on the original D1 card are labeled to offer an accuracy of 5%. Actually, their individual values differ highly from their nominal value. However, the accuracy of

a voltage divider is only dependent on the accuracy of the ratio of the resistor values. A measurement has shown that the ratio of the resistor values matches the nominal attenuation ratio exactly. This suggests the assumption that the resistors have been chosen from a wide variety and selected for exact attenuation ratios. This also leads to the implication that the resolvers have not to be loaded with an exactly defined resistance¹.

Nowadays resistors with an accuracy of 0.1% are available and therefore chosen for the replacement of the D1 functionality implemented in the resolver input modules. These voltage dividers may distort the resolver attenuation ratio slightly and therefore cause a position read-out error. To consider worst-case conditions, each voltage divider is distorted maximally if both deployed, prospective professionally mounted resistors $R_1 < R_2$ have the greatest possible deviation from their nominal value in different manner:

$$R'_1 = R_1 \cdot (1 + e) \wedge R'_2 = R_2 \cdot (1 - e) \quad (8.2)$$

with e representing the maximal deviation. In case of 0.1%-resistors, e amounts to ± 0.001 . Each voltage divider attenuates one individual resolver signal, for example $V_{\sin}(t)$. For a better comprehensibility $V_{\sin}(t)$ is assumed as falsely attenuated in the following example as described in equation 8.2. To obtain the maximal positing error, the corresponding resolver signal $V_{\cos}(t)$ has to be distorted in inversed matter as shown in Figure 8.2.

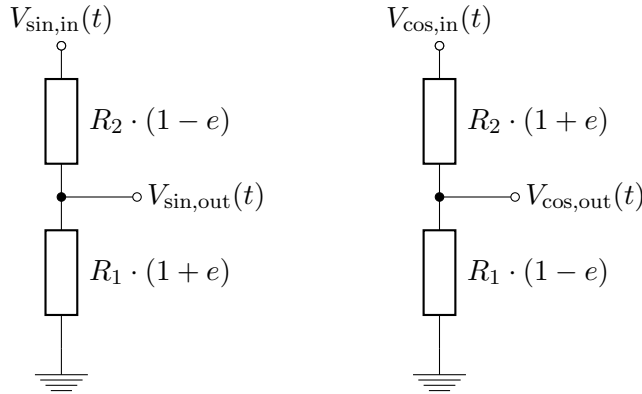


Figure 8.2: Worst case attenuation ratio error caused by inaccurate resistors of the voltage dividers. Each resolver (coarse, fine) of both axis (azimuth, elevation) is provided with such a pair of voltage dividers. The D1 card also attenuates two additional, for the ACI needless, signals what is now done by the D1 card replacements on the resolver input modules.

¹This is the reason why it was decided to connect the input of the ADCs of the resolver input modules directly to the outputs of the voltage dividers. Besides, the ADCs have a very high, symmetrical input impedance and therefore do not distort the resolver signals.

The final attenuation ratio can be determined:

$$V_{\sin, \text{out}} = V_{\sin, \text{in}} \cdot \frac{R_1 \cdot (1 + e)}{R_1 \cdot (1 + e) + R_2 \cdot (1 - e)} \quad (8.3)$$

$$V_{\cos, \text{out}} = V_{\cos, \text{in}} \cdot \frac{R_1 \cdot (1 - e)}{R_1 \cdot (1 - e) + R_2 \cdot (1 + e)} \quad (8.4)$$

$$\rightarrow \frac{V_{\sin, \text{out}}}{V_{\cos, \text{out}}} = \frac{V_{\sin, \text{in}}}{V_{\cos, \text{in}}} \cdot \left(1 + \frac{R_2 \cdot 4e}{R_2(e - 1)^2 - R_1(e - 1)(e + 1)} \right) \quad (8.5)$$

With knowledge about the resistor values ($R_1 = 10k\Omega$, $R_2 = 33k\Omega$), this formula can be simplified to

$$\frac{V_{\sin, \text{out}}}{V_{\cos, \text{out}}} = \frac{V_{\sin, \text{in}}}{V_{\cos, \text{in}}} \cdot 1.0031 \quad (8.6)$$

The final worst-case attenuation ratio error amounts therefore to 1.0031. The influence on the positioning error is dependent on the angle of the resolver and plotted in Figure 8.3.

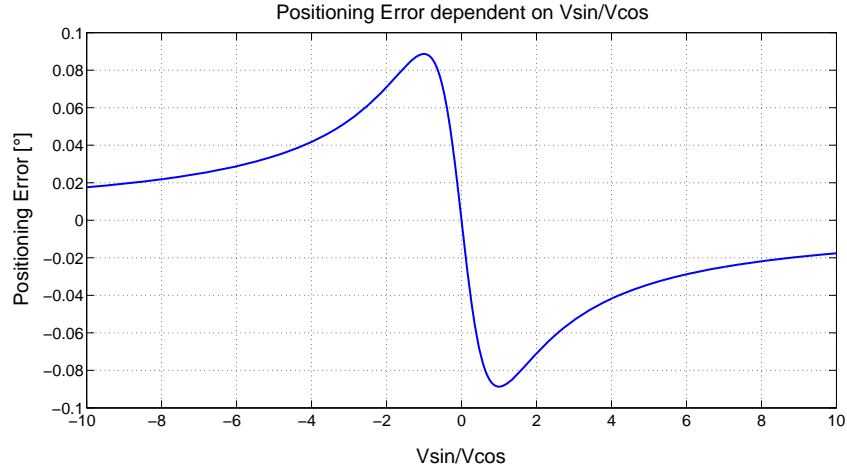


Figure 8.3: The relative positioning error as function of the ratio $V_{\sin}(t)/V_{\cos}(t)$ caused by the attenuation ratio error e . This chart is computed by comparing $\text{atan}(V_{\sin}(t)/V_{\cos}(t))$ with $\text{atan}(1.0031 \cdot V_{\sin}(t)/V_{\cos}(t))$

The maximal error is reached when the resolver angle ϕ is 45° ($V_{\sin}(t)/V_{\cos}(t) = 1$) and amounts to 0.088° . Due the look-up table, the read-out of the coarse resolver is not affected to attenuation ratio errors. The read-out of the fine resolver is affected, and the final worst-case positioning error – in relation to the native coordinate system – can be computed by:

$$p_{\text{error,attenuation}} = \frac{0.088^\circ}{72} = 0.0012^\circ \quad (8.7)$$

This error is identical if the attenuation error of the voltage dividers for $V_{\sin}(t)$ and $V_{\cos}(t)$ are switched.

Both outlined possible resolver signal attenuation ratio errors can be neutralized by a calibration. Usually, it is sufficient to measure the resolver angle at a specific axis angle to determine its offset. If another measurement can be performed at a second angle, it is possible to determine the signal attenuation ratio error and consider it in the position calculation algorithm processed by the MPU.

8.1.2 Resolver nonlinearity

Chapter 6.3.1 broached the subject of resolvers showing a nonlinear behavior. The coarse resolvers are highly affected by the nonlinearity, but the look-up table eliminates this problem. The nonlinearity of the fine resolvers can not be determined exactly without extensive measurements performed including the use of RF equipment. However, based on the prior measurements the positioning error in consequence of the nonlinearity can be estimated. In contrast to the coarse resolvers, the fine resolvers have shown a high accuracy. The signals of the fine resolvers never leaped and the noise is insignificant. Because all observed nonlinear phenomenas could be limited to a dimension of 0.05° , the accuracy of the fine resolvers is assumed to be lower than 0.10° at this point. Nevertheless it is strongly recommended to perform additional measurements to confirm this assumption and determine the exact accuracy. Transformed to the native coordinate system, this error is computed to be lower than

$$p_{\text{error,nonlinearity}} = \frac{0.10^\circ}{72} = 0.0014^\circ \quad (8.8)$$

8.1.3 Processing latency

A further, direct effect on the positioning error is the contouring error. It exists a timespan between the measurement of the resolver signals and the processing of these signals on the MPU. This timespan can be split up in three main parts:

- Averaging time on resolver input module
- CAN data transfer time
- MPU CAN driver latency

The first point concerns the time latency during the time, the resolver input modules grab all samples for averaging. When the average is computed, some time has passed since the middle of the sampling time. This time is

$$t_{\text{latency,average}} = 72 \mu\text{s} \quad (8.9)$$

When the resolver sample data is prepared to be sent out, it is transferred by the CAN bus. The transfer time on the CAN bus is dependent on packet collision probabilities and therefore the CAN bus packet modeling. This is one reason why much effort was put in a structured and priority based CAN identifier setup as described in chapter 4.1.3.5. The worst-case latency calculation [6], which is always dependent on the used

CAN modeling, is quite complex and therefore only outlined in this context. The mean latency for the transmission of the resolver data packet can be computed to 130 μs . This value applies in about 97% of all transmissions. The improbable worst-case latency is 262 μs . This value is used for the allover worst-case transfer latency:

$$t_{\text{latency,transfer}} = 262 \mu\text{s} \quad (8.10)$$

As soon as the packets are sent out on the bus, the CAN controller of the MPU receives the packet. The SocketCAN driver relays it to the server software after a short latency. By performing benchmarks the dimension of the latency could be determined: the latency is always between 400 μs and 800 μs . Thus, the worst-case latency is assumed to be

$$t_{\text{latency,driver}} = 800 \mu\text{s} \quad (8.11)$$

As soon as the resolver data packet is received by the server software the packet will be evaluated and processed immediately.

The overall latency can be summed up:

$$t_{\text{latency}} = t_{\text{latency,average}} + t_{\text{latency,transfer}} + t_{\text{latency,driver}} = 1134 \mu\text{s} \quad (8.12)$$

The latency causes a velocity-dependent positioning error. The angular velocity is typified by $\omega(t)$. The latency can be computed by

$$p_{\text{error,latency}} = \omega(t) \cdot t_{\text{latency}} \quad (8.13)$$

Figure 8.4 shows the plot of this formula.

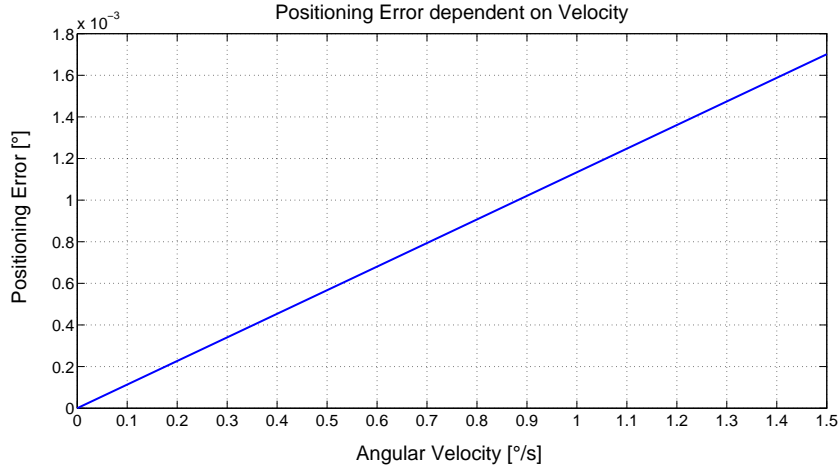


Figure 8.4: This chart outlines the positioning error which is caused by the contouring error. The error is plotted as function of the angular velocity $\omega(t)$.

Assuming that a GPS satellite (velocity as seen from the ground station: 0.0083°/s) is tracked, the contouring error is $9.45^\circ \cdot 10^{-6}$. To consider worst-case conditions, the positioning error of a tracking with full speed can be estimated to:

$$p_{\text{error,latency}} = 1.5^\circ/\text{s} \cdot 1134 \mu\text{s} = 0.0017^\circ \quad (8.14)$$

8.1.4 Mechanical accuracy

Beside negative effects of the electronics, also the mechanical structure of the antenna has to be considered for an overall position error computation. In the datasheets from M.A.N., the manufacturer of the antenna, the accuracy of the mechanical steering mechanism is quoted to be lower than

$$p_{\text{error,mechanic}} = 0.003^\circ \quad (8.15)$$

8.1.5 Overall positioning error

To compute the overall positioning error, the four types of positioning errors are summed up. The calculation is done in table 8.1:

ERROR TYPE	DEPENDENT ON	POSITIONING ERROR
resolver attenuation	resolver angle	0.0012°
resolver nonlinearity	(indeterminate)	0.0014°
processing latency	angular velocity of axis	0.0017°
mechanics	(static)	0.003°
overall positioning error		0.0073°

Table 8.1: Overview of the main positioning error types.

Thus, the overall positioning error in worst-case conditions is computed to be 0.0073°. Because many parts of positioning error types are deterministic (for example, the mean CAN bus transfer time and the latency error are known), it is possible to implement an algorithm in the server software which considers biases in the position calculation and minimizes the positioning error.

The final accuracy can be determined as soon as exact measurements are performed. Until then, it is possible that yet unknown error types are present and the resulting maximal positioning error is conservatively estimated to be

$$p_{\text{error}} \leq 0.01^\circ \quad (8.16)$$

8.2 System workload

The workloads of the critical parts of the ACI have been measured in full operation (tracking in runlevel 7) to figure out the performance of the system and free resources for future enhancements. In the actual configuration the CAN bus is loaded up to 16 %. If the resolver measurement is performed twice a period, its load is computed to be approx. 29 %. The first core of the MPU has a load of 28 %. The second core is underemployed and can be used for outsourcing heavy algorithms in future extensions.

9 Conclusion

After finishing this thesis, the ACI system is ready for a reliable and robust antenna steering control. This was achieved by several steps: First, all important signals of the analog control electronics had to be figured out and suitable interfaces had to be determined. Based on the results of ACIv1 a modularized concept was designed which provides high flexibility and safety. According to the concept, the exposed interfaces are implemented on modules which replace existing cards in the control racks. The functionality of the replaced cards is copied to the new interface modules and thus the monumental protection is not violated. A bus connects all interface modules to the MPU which manages and controls them. The MPU is enhanced with suitable bus drivers and also implements the motion controllers. The MPU is accessible via a network interface. An appropriate client software was written that interacts with the MPU and gives the user full control about the new digital controller. This work focused on the reliable framework of the ACI system, and several parts are optimized for easy, safe and uncomplicated enhancements of future add-ons.

As soon as the development of the Antenna System Interface (ASI) is finished, it is possible to use both systems to remotely control the whole antenna facility. Both systems are enhanced with the Antenna Monitoring System (AMS) which monitors both systems, the existing motion controlling system and also the human accessible areas. This system will provide the compulsory safety to control the antenna without human supervision.

9.1 Trivia

DESCRIPTION	VALUE
Codelines of the resolver input module (MCU)	3864
Codelines of the cam disc input module (MCU)	3101
Codelines of the velocity output module (MCU)	3172
Codelines of the server software (MPU)	11186
Codelines of the client software	9592
Amount of SPS conditions which are monitored each second	1800
Maximum number of clients	12

Acronyms

ACI	Antenna Control Interface
ACIv1	1st version of the Antenna Control Interface
ADC	Analog-Digital Converter
AMS	Antenna Monitoring System
ASI	Antenna System Interface
CAN	Controller Area Network
CRC	Cyclic Redundancy Check
D1	Dämpfungsglied 1
DAC	Digital-Analog Converter
DLR	Deutsches Zentrum für Luft- und Raumfahrt e.V.
ECI	Earth-Centered Inertial
ESA	European Space Agency
ESMO	European Student Moon Orbiter
FPU	Floating Point Unit
GPIO	General Purpose Input/Output
HPS	Hardware Protection System
I²C	Inter-Integrated Circuit
ISP	In-System Programming
JTAG	Joint Test Action Group
LED	Light Emitting Diode
MCU	Main Controlling Unit
MPU	Main Processing Unit
PoE	Power over Ethernet

PR Prozessrechner
radome Radar Dome
RF Radio-Frequency
SIV3 Soll-Ist-Vergleich 3
SNR Signal-to-Noise Ratio
SPI Serial Protocol Interface
SPS Software Protection System
SSIMUC Student Satellite Initiative Munich
TLE NASA/NORAD Two Line Elements Format
TUM Technische Universität München
UART Universal Asynchronous Receiver Transmitter
UPS Uninterruptible Power Supply

List of Figures

1.1	View on the radome of <i>Raisting 1</i>	8
1.2	The antenna inside its radome	9
2.1	The two motion control racks of <i>Raisting 1</i>	11
2.2	Mode selection panel of the <i>Reglergestell</i>	12
2.3	Theoretic assembly of a resolver	13
2.4	Original resolver attenuation card D1	14
2.5	Divided azimuth slewing range according to cam disc state	15
2.6	Opened azimuth cam disc switch	16
2.7	Original SIV3 card	17
2.8	Simplified overview of the original antenna control loop	18
3.1	Simplified overview of the adapted antenna control loop	20
3.2	Interrelation of the Antenna Control Interface bus	25
4.1	Power protection circuit	27
4.2	Hardware protection system circuitry	29
4.3	The common jacks of the interface modules	33
4.4	Resolver input modules mounted in the <i>Reglergestell</i>	41
4.5	Adaption procedure of the trigger voltage	43
4.6	Averaging of the resolver data samples	45
4.7	Input module	46
4.8	Cam disc module	48
4.9	Simplified velocity signal generation circuit	51
4.10	Output module as shown in the television broadcast <i>Abendschau</i>	52
4.11	Output module	53
5.1	The hub of the ACI	55
5.2	CAN adapter for the MCU	56
5.3	CAN terminator	56
5.4	Programmer tool	57
5.5	Debugging interface	58
5.6	Debugging tool	59
6.1	Pandaboard	60
6.2	Determination of the angle of an axis	62
6.3	Maximal recognizable error of coarse resolver	63

6.4	Relationship between sections and the fine resolver	64
6.5	Control Loop Overview	68
6.6	Server Networking	75
6.7	Console output of the server software	76
7.1	Client Networking	78
7.2	Client software	79
8.1	Voltage divider as used on the D1 card	80
8.2	Worst case attenuation ratio error	81
8.3	Relative positioning error	82
8.4	Contouring error due to latency	84

List of Tables

2.1	Kinetic boundary conditions of <i>Raisting 1</i>	11
4.1	The runlevels and the specific conditions	34
4.2	The runlevels and the specific tasks	34
4.3	Currently supported debug modes	35
4.4	SPS softstop trigger circumstances	36
4.5	External signals which are monitored by the SPS	36
4.6	CAN message priority field	38
4.7	CAN message task field	38
4.8	CAN message broadcast field	39
4.9	CAN message module type field	39
4.10	CAN message axis field	39
4.11	CAN stop request messages	40
4.12	Two example message compositions	40
4.13	Logic table of cam disc switches N4 and N5	47
8.1	Overview of the main positioning error types	85

Bibliography

- [1] Philipp Berthold, Stefan Diewald, and Lars Kreutz. Antenna Control Interface. Technical report, March 2011.
- [2] Me-Systeme. CAN-Bus Grundlagen.
- [3] Fabian Greif. Ansteuerung des MCP2515 (Tutorial), July 2007.
- [4] electronic design. Special low-pass filter limits slope, May 1998.
- [5] Bayerischer Rundfunk. Abendschau - Live aus Raisting, October 2011.
- [6] K. Tindell, A. Burns, and A. Wellings. Calculating Controller Area Network (CAN) Message Response Times. *Control Engineering Practice*, 3, 1995.