

A low cost transceiver for single wire sensor control

Eric Kooistra

Netherlands Foundation for Research in Astronomy (ASTRON),
Oude Hoogeveensedijk 4, 7990 AA Dwingeloo, The Netherlands
E-mail: kooistra@astron.nl

Abstract—The High Band Antenna (HBA) in the Low Frequency Array (LOFAR) radio telescope includes a phased array of 16 antenna elements. Via selectable delay lines 16 input signals are combined into one radio frequency (RF) signal beam that is transported via a coaxial cable to a receiver unit. To control the delay settings of the HBA, a new half duplex, master to multiple slaves communication protocol and a transceiver were developed. The control signaling is done via the same coax that carries the RF signal. Not counting the ground return path such a communication link is called a single wire or one-wire communication link. This paper fully describes the single wire control protocol and transceiver design, it highlights their features and it explains why existing single wire communication schemes were not used.

Index Terms— Single wire communication, sensor control, Manchester decoding, half duplex

I. INTRODUCTION

THE LOFAR telescope uses phased antenna arrays to form an aperture synthesis telescope for receiving radio signals in the frequency band of 30 MHz – 240 MHz [1-3]. The LOFAR radio telescope is developed by ASTRON in the Netherlands. In total LOFAR is planned to have 77 phased array antenna stations and a first antenna station is currently being evaluated in the field. Each antenna station contains 192 receiver units (RCU) to sample the radio frequency (RF) signals from 96 dual polarization antennas. The receiver units are placed in a central cabinet where also further digital signal processing is done on the RF signals. The distance between an antenna and receiver is about 100 m. Two antennas per receiver are used to cover the frequency band and only one of them can be operational at a time. The Low Band Antenna (LBA) covers 30 MHz – 80 MHz and the High Band Antenna (HBA) covers 120 MHz – 240 MHz. The LBA is a single dipole antenna, whereas the HBA antenna consists of a tile of 4 by 4 dipole antennas and an analogue beam former [12]. The analogue beam former consists of selectable delay lines in each dual polarization frontend and two summatoms. The HBA requires control to select the delay lines. The delay setting for each antenna dipole fits in one byte and has to be updated every second to keep the HBA beam pointed at a source. With 16 dipoles, 2 polarizations and 8 delay control bits the HBA

control task is to write in total 256 bits every second. For HBA monitoring a read access is also required, a half duplex link suffices for that. Hence for LOFAR a control bit rate of about 1 kbps is sufficient, for other purposes like e.g. fast HBA testing a higher bit rate of e.g. 10 kbps is desirable.

After an orientation on existing control link solutions the subsequent sections describe the new single wire communication protocol and transceiver design as used for the HBA control in LOFAR [13].

II. ORIENTATION

A. Existing techniques

Several existing communication techniques were considered for the HBA control. Using a wireless radio link was rejected because this might cause interference in the LOFAR band or in other astronomical bands in case the HBA is located near an astronomical site, e.g. near the Westerbork Radio Synthesis Telescope. Using RS485 was rejected because of the cost and complexity of the additional wiring in an LOFAR antenna station. Hence the preferred scheme is to perform the HBA control via the same coaxial cable that carries the RF signal.

Again a radio link via the coax is possible using radio transceivers in the industrial, scientific and medical (ISM) band or in the short range devices (SRD) band, however this was rejected for cost. Using the SensorPath one-wire bus [4] was rejected because it can only address up to 7 slave devices. Using the 1-Wire® serial bus [5] was rejected because of its fixed bit rate (standard 15.3 kbps) which makes it less flexible in case it is necessary to exchange bit rate for robustness given the 100 m communication distance for the HBA control. Using power line modems (PLM) was rejected because they are too specifically built around the presence of the 50 Hz power signal. Based on lowest cost and the fact that a low bit rate of about 1 kbps suffices for the HBA control a microcontroller (< 1 €) solution similar to those used in DiSEqC™ [6] seems the most appropriate. Using a microcontroller also allows integrating the other input/output (I/O) interfaces that the HBA control needs.

DiSEqC™ is the Digital Satellite Equipment Control

standard. It defines DC (direct current) power supply and fixed 0.66 kbps control signaling over coaxial cable using amplitude shift keying (ASK) of a 22 kHz tone. DiSeqCTM assumes an impedance of 15 Ω at the master side. The received signal is detected over this impedance by both master and slaves. In practice the control signal is generated over a parallel RLC circuit that acts as a frequency dependent pull-up tuned to 22 kHz. The R is 15 Ω . The coil passes the DC and e.g. L = 220 μ H, C = 220 nF. Apart from control, also the DC for the HBA needs to be supplied via the coax. The HBA accepts 48 V and takes about 1 A. At 1 Ampere a coil of 200 μ H becomes quite big (> 1 cm³) and costly (> 1 €). Increasing the tone frequency would help somewhat, but it can not be made too high when the transceiver should be implemented using a low cost microcontroller.

B. HBA context

Given that the HBA is a dual polarization antenna it has two RF output signals that are passed on via two coaxial cables. This allows supplying DC via one coax and using the other coax for HBA control as in Fig. 1. The DC is passed on at 48 V to the HBA. A DC/DC converter at the HBA then distributes the DC at a lower level again via coax cables to each frontend. Separating DC from control avoids the problems with the large inductance as in DiSeqCTM. Furthermore it allows using rail-to-rail (ground to supply) signaling at baseband for the control, making it very robust.

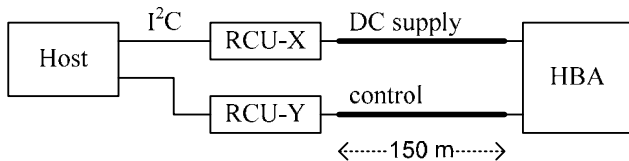


Fig. 1. Control path between host and HBA

The HBA control is extended into the HBA to each of the 16 dual polarization frontends. This is possible because the summator in the HBA forms a short circuit to the low frequency control signal. This results in a point to multi-point (p2mp) HBA control link between a client device in the RCU and 16 server devices in the HBA as shown in Fig. 2. The advantage of this p2mp link is that it avoids dedicated control wiring inside the HBA.

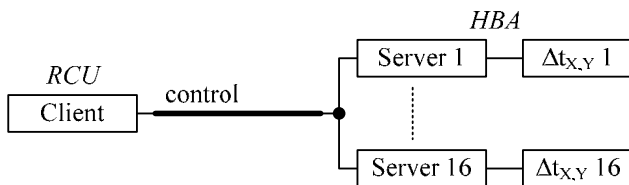


Fig. 2. Point to multi-point control link

The control device at the RCU is the client, because it issues requests. The control devices at the HBA are the servers. Note that contrary to typical client-server networks here there are multiple servers and only one client. The host uses I²C [7, 8] to communicate with the client device, because

in the LOFAR antenna stations I²C is used whenever a chip to chip serial bus is needed. Hence the client device on the RCU is an I²C slave for the host and it is the master for the HBA control link. In this way the client device hides the HBA control physical layer from the host.

C. Messaging protocol

Examples of various bus specific messaging protocols can be found in the specification of e.g. wireless transceivers and DiSeqCTM. MODBUS [9] is a well known bus independent messaging protocol. It was not investigated further whether MODBUS could have been used. Instead for the HBA control link a simple proprietary messaging protocol was developed to allow implementation in a low cost microcontroller with little memory. The messaging protocol should support a broadcast message. By using a broadcast message all delay line selections in the HBA can be set at once, so that the HBA beam will point in the new direction nearly instantaneously. Without such a broadcast message the HBA beam would vary in an undesirable way during the time that the delay line selections are set one by one. A unicast message may be used to read back data from an antenna element.

III. TRANSCIVER MEDIUM ACCESS LAYER

A. Message flow

The link concept is that of a master to multiple slaves. The client is the master and the servers are the slaves. A communication is always started by the master, a slave can only respond. The server response has to start within some bit times after the client request. For the HBA control the host can set the delays for all servers via a broadcast request as in Fig. 3. Each server extracts from the payload its own data using its server address as index. The broadcast access is not responded by the servers. The host may verify the delay settings by reading them back one by one from each server using a unicast request as in Fig. 4. A valid unicast request message is always responded by the server.

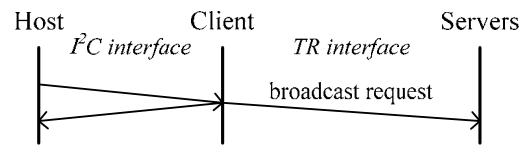


Fig. 3. Broadcast message flow

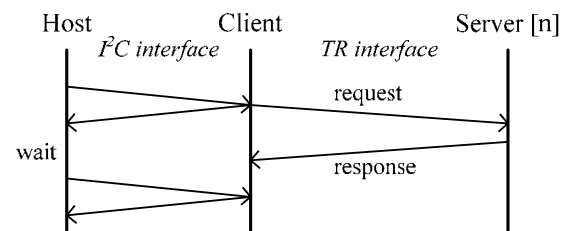


Fig. 4. Unicast message flow

B. User data definition

The unit user data format is an octet (8 bits). The data is transmitted, most significant bit first. The user data consists of a 1 octet address followed by a payload of 0 or more octets as shown in Fig. 5.



Fig. 5. User data definition

The broadcast address is defined as 0. The slave addresses are 1 to 126. In a master request message the most significant address bit is always 0 and for the response message the slave uses its own address but with the most significant bit set to 1. Hence when the most significant address bit is 1, then the other slaves can quit further reception. Address 127 is reserved as production address. Address value 128 can not occur, because there is no response to the broadcast message. Table 1 summarizes the HBA control link address definition.

Address	Description
0	Broadcast
1-16	HBA delay element servers
17-126	Spare
127	Production server address
128	Not used
129-255	Response address of server 1-127

Table 1. HBA control link address definition

Server address 127 is reserved as production server address. The operational server address can be programmed via the transceiver interface by writing the server address register. The server address register must be in non-volatile memory in order that after a subsequent power down - power up cycle the operational server address is retained [15]. The server still uses address 255 to acknowledge the original write server address request. For subsequent requests it will use its operational server address. This scheme allows the same software image to be used for all servers, i.e. all produced with server address 127. The new server address must be in the valid range of 1 to 127. Furthermore the server address can only be changed using a unicast server access. Otherwise the new server address is ignored.

IV. TRANSCEIVER APPLICATION LAYER

This section defines the payload formats for the client request and the server response. The payload fits in the user data shown in Fig. 5.

A. Client request payload

The server knows that it is a unicast payload or a broadcast payload, based on the user data address.

The unicast (uc) request payload structure is kept generic in order that the client does not require knowledge of the server

functionality. Thanks to the payload length field the receiver directly knows the payload length without having to derive it from the information in the other fields. This leads to the request payload defined in Table 2. The size is defined in octets. The payload length is the number of (nof) octets in the payload including the payload length field itself.

The write data can be 0 or more bytes, dependent on the server function. The register definition is dependent on the server application. The register identification (ID) field identifies the addressed server register. Table 3 lists the supported server functions.

Offset	Field	Size	Comment
0	Payload length	1	Nof octets in payload
1	Function	1	Server function
2	Register	1	Server register ID
3	Data	0..max	Write data

Table 2. Definition of the client request unicast payload

Function	Description
0	Reserved
1	Reserved
2	Set byte
3	Get byte
4	Set word (= 2 bytes)
5	Get word (= 2 bytes)
6-127	Reserved
128-255	Server application dependent function codes

Table 3. Definition of the server functions

Table 4 defines the broadcast (bc) request payload. The 'first server' and 'last server' fields contain the address range of all servers participating in the broadcast. The payload contains the data for all servers. Using its own address as index each server can extract its data from the payload.

Offset	Field	Size	Comment
0	Payload length	1	Nof octets in payload
1	Function	1	Server function
2	Register	1	Server register ID
3	First server	1	First server address participating in the bc
4	Last server	1	Last server address participating in the bc
5	Data first server	0..max	Write data first server
	Data next server	0..max	Write data next server

	Data last server	0..max	Write data last server

Table 4. Definition of the client request broadcast payload

B. Server response payload

Table 5 defines the response message payload. The client can use the payload length field to check against the expected

payload length. For a read request function the response payload will contain the read data. For a write request function the response is merely an acknowledge and the response payload will contain only the payload length field with value 1.

Offset	Field	Size	Comment
0	Payload length	1	Nof octets in payload
1	Data	0..max	Read data dependent on request function

Table 5. Definition of the server response payload

V. CLIENT REGISTERS

The host accesses the client registers via the I²C interface. An I²C write access consists of a command (cmd) byte followed by the data bytes to write. An I²C read access consists of a command byte after which the client will provide the read data bytes. Table 6 lists the I²C command registers that are supported by the HBA control client. More registers may be defined, e.g. a register to control a LED, a register to set the speed of the HBA control link.

Cmd	Register	Size	Description
0	REQUEST	38	W: Write and issue request R: Read back request
1	RESPONSE	4	W: Overwrite the response R: Read response

Table 6. Client I²C command registers

The size of the request register fits a broadcast write request of 2 data bytes for 16 servers. The size of the response register fits a unicast read request of 2 data bytes. The read back request register command can be used to verify the I²C interface and the overwrite response register command can be useful to clarify that the client has received a fresh response.

A. Request register

Table 7 and Table 8 define the request register for a unicast request and for a broadcast request. The distinction between a unicast or a broadcast request is made based on the server address field being > 0 or = 0, respectively.

Offset	Field	Size	Comment
0	Server address	1	Server address > 0
1	Payload length	1	Idem as in Table 2
2	Function	1	Idem as in Table 2
3	Register	1	Idem as in Table 2
4	Data	0..max	Idem as in Table 2

Table 7. Definition of the client request register for a unicast server access

Offset	Field	Size	Comment
0	Broadcast address	1	Server address = 0
1	Payload length	1	Idem as in Table 4
2	Function	1	Idem as in Table 4
3	Register	1	Idem as in Table 4
4	First server	1	Idem as in Table 4
5	Last server	1	Idem as in Table 4
6	Data first server	0..max	Idem as in Table 4
	Data next server	0..max	
	
	Data last server	0..max	

Table 8. Definition of the client request register for a broadcast server access

For a broadcast request the same function and register are used for each server, only the contents of data can differ per server. The size of the data field depends on the cast type (unicast or broadcast) and on the function. A new request register I²C write access will be ignored when the client is already busy processing a request.

B. Response register

Table 9 defines the response register for a unicast server access. For a broadcast access the servers do not respond and the response register is then not affected.

Offset	Field	Size	Comment
0	Status	1	Response status
1	Nof bytes	1	Nof bytes including this byte
2	Data	0..max	Read data from the server

Table 9. Definition of the client response register

The host knows what kind of response data to expect given the preceding request. For the functions set byte and set word the response nof bytes field will be 1. For the functions get byte and get word the response nof bytes field will be 2 respectively 3. The response status field contains the server response address if the request went OK, otherwise in case something went wrong the response status contains 0.

VI. SERVER REGISTERS

The server registers are accessed via the transceiver interface. Table 10 defines the server registers that are used to control the delay elements in a HBA frontend unit.

Register ID	Register name	Size	Description
0	X delay	1	X pol. delay setting
1	Y delay	1	Y pol. delay setting
255	Server address	1	Server address

Table 10. Frontend unit server registers

A. Delay registers

The X delay and Y delay registers can be accessed individually using the ‘set byte’ or ‘get byte’ function. By accessing the X delay register using the ‘set word’ or ‘get word’ function it is possible to access both the X delay and the Y delay register in one request.

B. Server address register

The server address register indicates the server address in non-volatile memory (e.g. EEPROM). This register allows changing the production address of a server.

VII. TRANSCEIVER PHYSICAL LAYER

The transceiver principle design is shown in Fig. 6. The DC supply to the HBA will be carried by the other coax, so the control signaling can be done at baseband. The design fits both the client as well as the server, except in principle only the client device requires a pull up resistor.

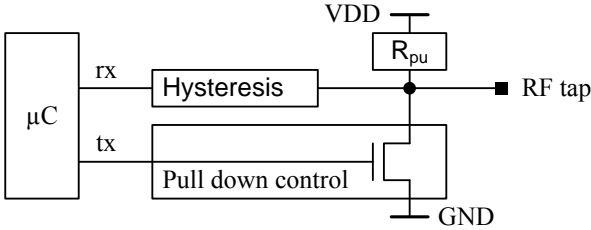


Fig. 6. Transceiver principle design

Using a pull up resistance R_{pu} of 500 Ω and assuming the total load on the signal line is 20 nF (10 nF for the coax and 10 nF for the connected devices) yields a RC time of $\tau=10 \mu\text{s}$. For robust communication the rise time should be about $t_r = 3\tau$. To avoid transmission line effects with the lengthy coax (disturbing reflections) there needs to be slow rate control on the pull down. A suitable slow rate is 0.5 V/ μs , so given a 5 V supply voltage this yields a fall time of $t_f = 10 \mu\text{s}$. Fig. 7 shows the shape of the control signaling. Adding t_f and t_r implies that reliable communication at a bit rate of 25 kbps is feasible. The signal to noise (SNR) ratio is abundant, hence bit errors due to additive white Gaussian noise (AWGN) will not occur. Errors may occur due to e.g. spikes or software faults, but avoiding these lies outside the scope of the transceiver principle design.

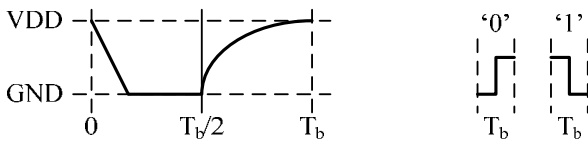


Fig. 7. Signal shape and bit definition

Allowing a poor timing accuracy of worst case $\pm 20\%$ in each device implies that each bit must be self-timed. Therefore the bits are transmitted using Manchester encoding, whereby bit ‘0’ becomes “01” and bit ‘1’ becomes “10” as in Fig. 7. The nominal duty-cycle is 50% and should be accurate within

$\pm 5\%$. Note that Manchester encoding is equivalent to Binary Phase Shift Keying (BPSK) whereby the carrier frequency equals the bit rate.

The slow rate controlled pull down and the hysteresis in the receiver together provide a reliable detection of the falling edge. Therefore the bit detection can be based on measuring the time between falling edges. Given the Manchester encoding the measured times can be T_b , $1.5T_b$ or $2T_b$. The detected bit depends on the previous bit and the measured interval time. The detection thresholds should be set at $1.25T_b$ and $1.75T_b$. Table 11 defines the bit detection scheme. Note that an interval time of $2T_b$ after a 0 bit can not occur. A simulation reveals that for random data the probabilities for T_b , $1.5T_b$ and $2T_b$ are 1/2, 1/3 and 1/6 respectively [13]. A similar scheme is discussed in [10], but the decoding is done slightly different there.

Previous bit	Interval time [T_b]	Detected bit(s)
‘0’	1	‘0’
‘0’	1.5	“01”
‘0’	2	‘X’
‘1’	1	‘1’
‘1’	1.5	‘0’
‘1’	2	“01”

Table 11. Bit detection scheme

A known preamble sequence is used to define the start of the signaling, so that after that the received bits can be uniquely decoded from the measured interval times. To ensure that the last data bit can be detected with a falling edge a trailer bit is added. Fig. 8 shows the preamble sequence and trailer bit. The user data (so the address and payload of Fig. 5) is projected by a 2 byte Cyclic Redundancy Check (CRC). The CRC-16 polynomial is $X^{16} + X^{15} + X^2 + 1$. If the calculated CRC does not match the received CRC, then the received user data must be discarded.

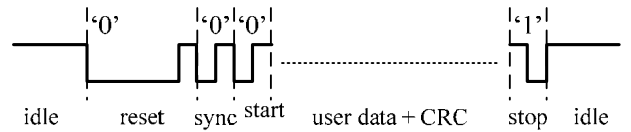


Fig. 8. Packet preamble and trailer definition

During idle the line is high (pull up). The falling edge of the reset pulse wakes up the servers if they are not already active. The extended low part of the reset pulse uniquely defines the beginning of a new message. The reset pulse is defined as $t_{low} \geq 2.5T_b$ and $t_{high} \geq 0.5T_b$. The ‘0’ reset bit, ‘0’ sync bit and ‘0’ start bit together ensure that the first measured time interval occurs during the sync bit and yields T_b . The receiver can use this measured T_b to adjust its thresholds for decoding the subsequent time intervals. After the start bit the user data follows. The ‘1’ stop bit at the end ensures a last falling edge before going back to idle, this is necessary to always be able to decode the last bit of the CRC.

The client may set the bit rate per request message, the slave will respond with the same speed. Lowering the bit rate will give a more robust link or allow longer distance, increasing the bit rate will give a shorter communication time.

VIII. TRANSCEIVER DESIGN

The transceiver hardware design is built around a PIC® microcontroller (µC) [11]. Fig. 9 shows the schematic of the transceiver [14].

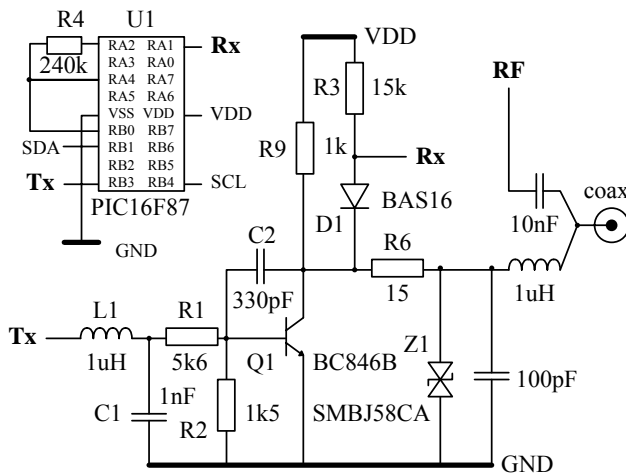


Fig. 9. Transceiver schematic

The design uses the internal comparator of the µC to detect the received (Rx) signal. Resistor R4 and two internal resistors in the µC provide hysteresis. The I²C slave function in the µC is only used in the client. Both client and server can go into sleep mode when the control link is not used to minimize electromagnetic interference (EMI). An I²C access will wake up the client and a falling edge on the Rx input will wake up the server. A timer module is used for measuring the time between falling edges and for the transmit (Tx) bit time generation. The µC runs on its internal 4 MHz oscillator which has an accuracy of ±10%, avoiding the need for an external crystal. Low pass filter L1, C1 blocks noise from the µC from causing interference on the RF signal. R1 is a bias resistor for transistor Q1. R2 with R1 ensures that Q1 gets cutoff when Tx is low. C1 forms the feedback circuit for the falling edge slew rate control. The comparator input will have a low impedance when the µC is not powered, therefore diode D1 ensures that a server does not hold the Tx low in case it has a power failure. Diode D1 also protects the circuit if by mistake the 48 V supply coax cable would be connected to the server control, see Fig. 1. Transistor Q1 can also withstand 48 V on its collector. The digital signal is fed to the coax via an LC combination that decouples the RF signal from the digital section. The transient voltage suppressor (TVS) Z1 is placed for over voltage protection. Its capacitive load is 400 pF, so together with the decoupling capacitor of 100 pF this results in a total load per device of 0.5 nF. Resonances that travel

along the cable are damped by R6. R6 is low enough to not raise the zero level of the signal too much. Resistor R9 is the pull up resistor. Due to D1 it was necessary to use a distributed pull-up rather than a single pull-up at the client. The approach taken is that R9 at the client provides charging of the coax cable and R3 at each server takes care of the additional capacity that this server adds. Hence R9 at the client is 1k and R3 at each server is 15k, resulting in a total R_{pu} of about 500Ω.

The software implementation [15] of the client transceiver uses the I²C slave module of the µC and is implemented on an 18 pins PIC16F87. The server transceiver instead implements a 28 pins PIC16F882 to have more I/O pins for setting the delay line switches on the frontend. The server software implementation [15] fits in 1.5 kWord of program memory (flash), allowing it to use the smallest, so cheapest, µC in the PIC16F88* range.

IX. CONCLUSION

The presented HBA control link provides control signaling via the same coaxial cable that carries RF. The DC supply is provided via other means. This allows for a low cost, robust transceiver design based on a microcontroller. The adjustable bit rate to exchange speed for distance is a new feature compared to existing single wire control techniques. Other key aspects of the link are: sleep mode support to minimize EMI, programmable server address, able to run on an inaccurate clock thanks to Manchester encoding, and the use of a broadcast message to synchronously control multiple devices. Tests to verify a guaranteed reliable operation of the HBA control link are ongoing, but the control link has already proven to be a proper solution for LOFAR and could also fit other single wire (sensor) control applications.

REFERENCES

- [1] LOFAR. Available: <http://www.lofar.nl>
- [2] A. W. Gunst, M. J. Bentum, "Signal Processing Aspects of the Low Frequency Array", IEEE International Conference on Signal Processing and Communications, 24-27 November 2007, Dubai, United Arab Emirates.
- [3] A. W. Gunst, G. W. Kant, "Signal transport and processing at the LOFAR Remote Stations", URSI, October 2005, New Delhi, India.
- [4] SensorPath™. Available: <http://www.national.com>
- [5] 1-Wire®. Available: <http://www.maxim-ic.com>
- [6] *DiSEqTM Bus Functional Specification Version 4.2*, 1998, Feb. 25, Eutelsat.
- [7] I²C. Available: <http://www.nxp.com>
- [8] SMBus. Available: <http://www.smbus.org>
- [9] *MODBUS Application Protocol Specification V1.1b*. Available: <http://www.modbus.org>
- [10] L. Olmedo, M. Alves, "Using the XGATE for Manchester Decoding", AN3015, Rev. 0, 10/2005. Available: <http://www.freescale.com>
- [11] PIC® microcontroller. Available: <http://www.microchip.com>
- [12] G. W. Kant, A. W. Gunst, M. Drost, "High Band Antenna Architectural Design Document", LOFAR-ASTRON-ADD-016, v1.0, 2007, Jan 16.
- [13] E. Kooistra, "HBA Control Design Description", LOFAR-ASTRON-MEM-175, v2.0, 2007, March 23.
- [14] P. H. Riemers, "A control line for LOFAR-HBA", ASTRON.
- [15] A. van Duin, "HBA control firmware design description", ASTRON.