Chapter 4

ADES internal digital communication system design

Chapter 4 will start off by procuring the correct hardware for the ADES system as well as the communication system. Here after the main focus will shift toward the development of the ADES digital communication system that will be implemented between the internal functional units (from now on referred to as the ADES digiComm protocol). This chapter will be divided into the different OSI layers that will be implemented.

4.1 Introduction

The ADES digital internal communication protocol (digiComm) will be tailored to meet the particular needs of an AMB system. The objective of the communication system will be to exchange data between the selected digital controller and the various specified functional units. The main goal will be to design a protocol to utilize a selected link between the main controller system, the power amplifiers, the ISensorboard, the motor drive unit and the power conditioning unit. This developed protocol needs to improve the reliability and robustness of the system, , reduce the cost, improve noise immunity and incorporate methods to ensure expandability and flexibility.

In this chapter the hardware selection according to the specified architecture will be discussed first. Then the focus will shift towards the development of a protocol for the internal communication signals.

4.2 Hardware

The hardware selected to meet the specifications of the architecture specified in Section 3.3 will be discussed in this section. All the hardware modules will be commercial off the shelf (COTS) items. This will reduce risk, cost and development time.

4.2.1 Master node (PMC module)

Communication control will be implemented on FPGAs. The FPGA on the main controller will be the master in the internal communication system and the FPGAs on the other functional units will be the slaves. The selected master node (PMC module) is illustrated in Figure 4-1.

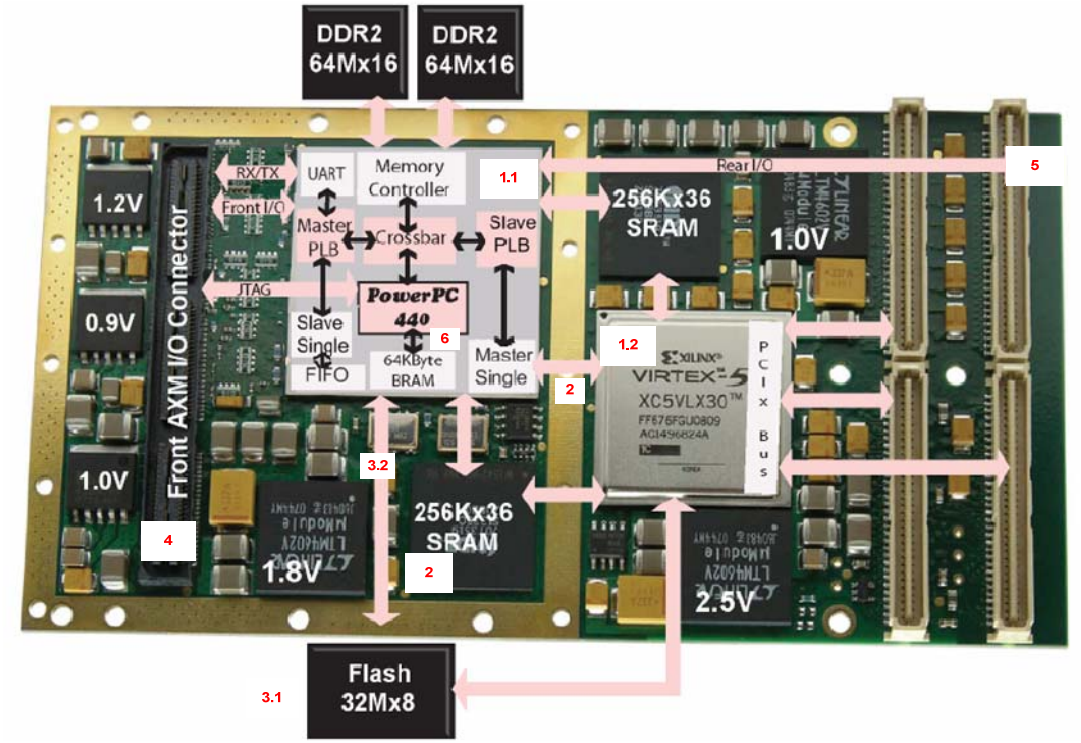


Figure 4-1: General overview of the master node.

The master node consists of:

1. Two Virtex 5 FPGAs. One FPGA (1.1) will be used for communication with the internal functional units through the front I/O and for the control algorithm of the ADES. The other Virtex 5 FPGA (1.2) is used only to establish communication between the main controller and the SBC by means of a PCI-X bus.

2. A local bus interface is used to connect the second smaller Virtex 5 FPGA that handles the PCI-X interface (2).

3. Various memory modules. All the programs will be situated on the Flash memory (3.1). During startup the block memory (BRAM) downloads the boot loop program from the flash memory into the BRAM (3.2). The boot loop program then configures the FPGA.

4. A front input/ output (I/O) connector (4) which will communicate with the various internal functional units. A digital I/O mezzanine board was selected to slot into the front I/O connector. This module is illustrated in Figure 4-2.

5. A rear I/O interface (5) that can be used to communicate by means of LVDS with I/O units.

6. A PowerPC interface (6). The PowerPC interface will be utilized to obtain data from the PCI bus and perform critical control functions including PID control.

4.2.2 AXM-D03 digital mezzanine module

The AXM-D03 I/O module consists of 16 bi-directional CMOS transceiver and 22 bi-directional differential signals. The FPGA is buffered by using RS 485 transceivers on the differential channels. These drivers are able to achieve a data transfer rate of up to 20 Mbps.

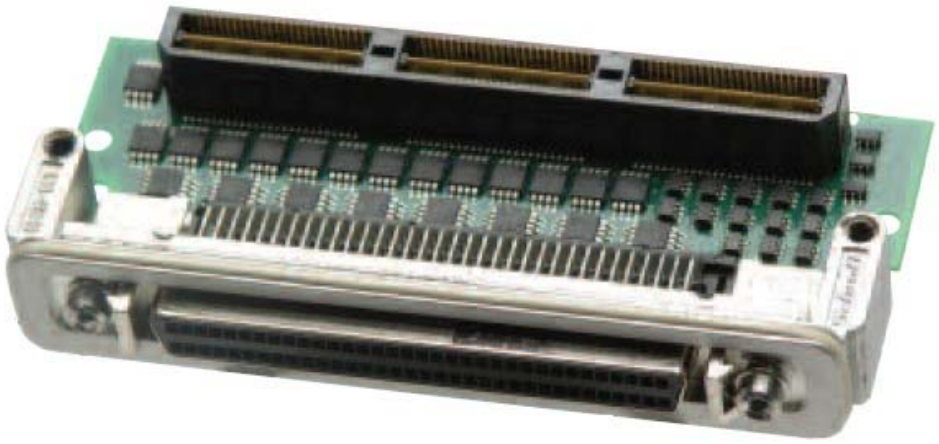


Figure 4-2: AXM-D03 mezzanine module.

4.2.3 SCSI cable

The AXM-D03 module is accessed via a 68 SCSI front panel connector. A 2 meter, round, 68 conductor, shielded cable with a male SCSI-3 connector on both ends and 34 twisted pairs will be

used to connect the AXM-D03 module to the termination panel. The mentioned cable is shown in Figure 4-3. This cable is not terminated.

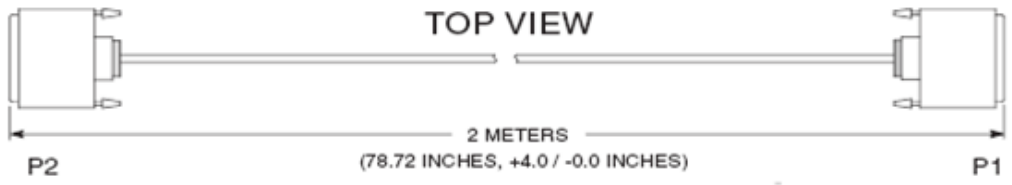


Figure 4-3: SCSI cable with connector

4.2.4 Termination panel

The SCSI cable with the male SCSI-3 connectors will connect to a termination panel as shown Figure 4-4 consisting of a DIN-rail mountable panel which provides 68 screw terminals for universal field I/O termination. The 68 screw terminals will be used to interface with the slave modules and to terminate the transmission lines on both ends.

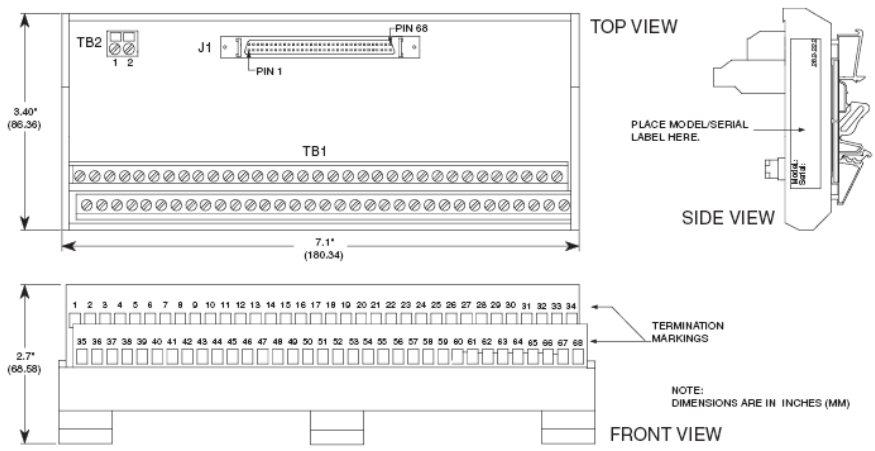


Figure 4-4: Termination panel

4.2.5 SBC (cPCI module)

The selected SBC ***shall*** meet the following requirements for the communication system:

1. Adhere to the cPCI industrial standard.
2. Consist of at least two PMC slots one for the controller and another for the Profibus card.
3. Consist of two Ethernet ports one for the remote access port and another for the maintenance port.

The module that satisfied these constraints was the PP410/03x SBC supplied by Concurrent Technologies Inc. Figure 4-5 illustrates the selected module.

Figure 4-5: Single Board Computer (SBC)

4.2.6 Profibus option

The selected Profibus card ***shall*** meet the following requirements:

1. Support the Profibus-DP standard.
2. Consist of a PMC interface.
3. Reach a baud rate of 12 Mbps.

The PMC253 module supplied from Kontron Inc adhered to these specifications. Figure 4-6 illustrates the selected module.

Figure 4-6: Profibus card

4.3 Protocol functioning

The communication system as illustrated in Figure 4-7 shows a point-to-point structure for each of the internal communication interfaces as identified in chapter 3. The ISensorboard (FU 1.2), power amplifiers (FU 1.3), motor drive unit (FU 1.4) will interface with the main controller via a selected transmission medium. This transmission medium will route back to a selected I/O module (FU 1.1.5) which will be situated on a PMC FPGA card (FU 1.1.3). Communication will be a half-duplex setup between all the internal functional units, except for the units CMOS digital lines.

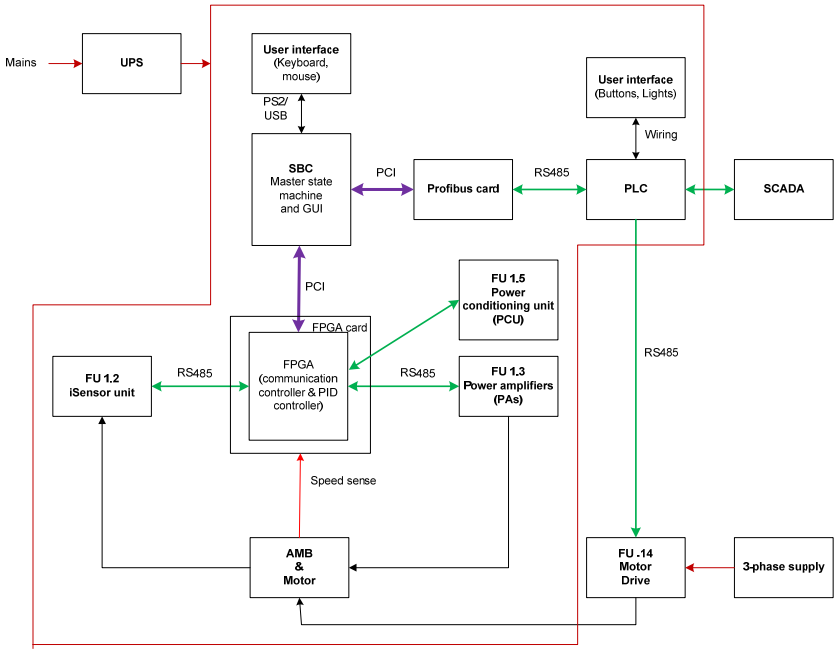


Figure 4-7: The communication structure of the system.

The internal communication structure will be classified as a master-slave structure, where the Virtex 5 FPGA is the master and the power amplifiers, sensor driver unit, the motor drive unit and SBC are the various slaves. The master will be responsible for:

- The master clock which will enable the various functional units.
- Obtaining the correct data from the various functional units.
- Distributing data to the correct functional units.

A master clock will trigger each of the units in the data flow path. The master clock will be provided on one of the digital channels available on the AXM-D03 module. One communication cycle of the ADES should be completed within a 20 kHz cycle which corresponds to 50 μs. In the

passive state each of the functional units will wait to be enabled by the master clock (sync) signal. The internal control cycle communication data flow path will be discussed in the next section.

4.4 Internal communication data flow path

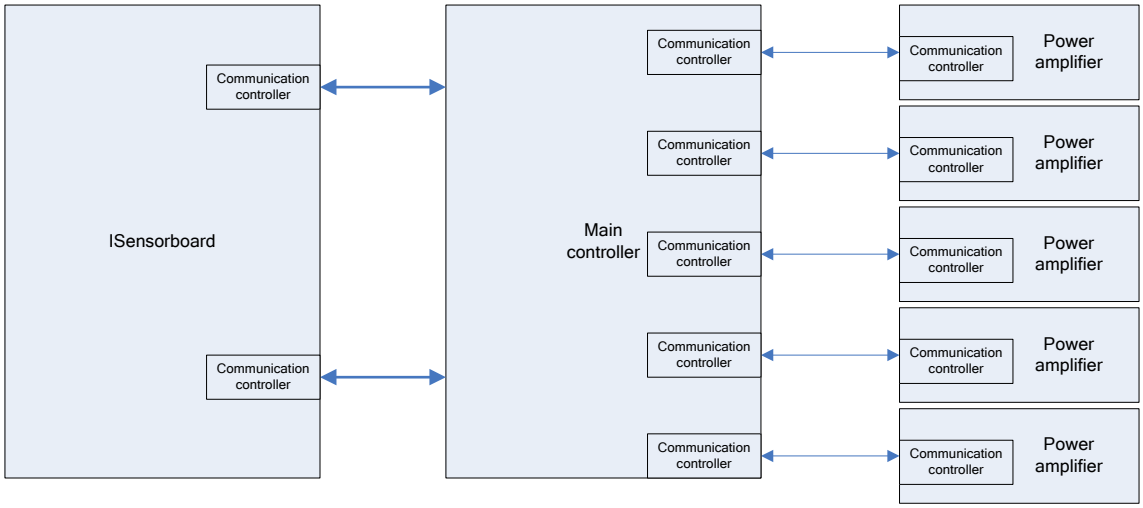


Figure 4-8: Communication data path

The internal communication data flow of the ADES will start on the ISensorboard. The ISensorboard will transmit two X, Y and Z position values to the main controller. These values will be used by the main controller to generate five current reference values that will be transmitted to five power amplifier boards. Each of the power amplifier boards consist of two power amplifiers. After each of the power amplifier boards received the reference current value, two true current values (one from each power amplifier) will be transmitted back to the main controller. All the internal communication of the ADES will be handled by the communication controllers.

4.5 Communication timing

In higher level communication systems, the timing requirements are in general not that strict, however when developing lower level communication timing systems, staying within timing requirements becomes crucial. The last very important aspect of the internal communication protocol that needs to be discussed is the communication timing diagram of the system.

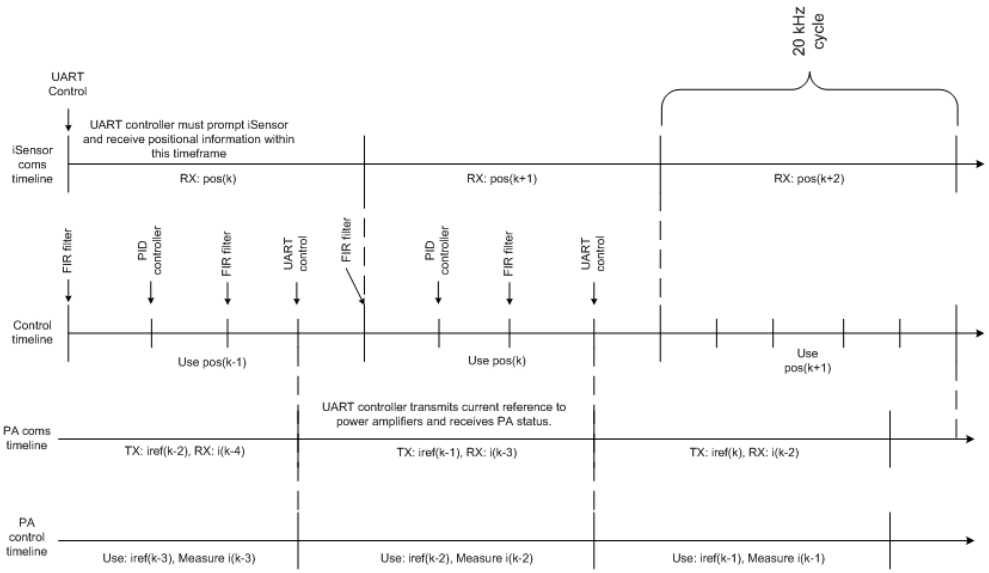


Figure 4-9: System timing

The timelines that influence the control of the ADES starts off at exactly the same time. The ISensorboard communication (iSensor coms) timeline begins by transmitting the position values, ***pos(k)***. The control timeline starts off at the same time, but because, ***pos(k)***, is not available at this stage the previous position value , ***pos(k-1)*** , is used to calculate the current reference value. When examining the power amplifier communication (PA coms) timeline, it is noted that the main controller, due to the delay of the position values and the reference value, transmits, ***iref(k-2)*** , which is delayed by two control cycles.

Another timeline that also starts at exactly the same time as the other timelines is the power amplifier control (PA control) timeline. Due to the previous delays, the power amplifier control timeline must use, ***iref(k-3)*** , implying that the total control cycle is delayed by three 20 kHz cycles. This is the maximum tolerable delay thus it is crucial that each of the communication cycles must not exceed the 20 kHz requirement.

4.6 Protocol layers

The ADES digiComm protocol will not consist of all the ISO/OSI layers as discussed in Chapter 2. The layers that will be implemented in the ADES digiComm protocol is the physical layer and the data link layer.

4.6.1 Physical layer

"This layer is concerned with the physical layer of the communications link – mechanical and electrical – and provides the means to transmit bits across a continuous communications path. The protocol for this layer sets details such as cable size, loss and frequency characteristics, connector types, pin arrangements, voltage levels and transmission coding." [22]

4.6.2 Electrical specifications

As previously mentioned and motivated in Chapter 3 the ADES digiComm protocol will employ the RS 485 standard in the physical layer. The RS 485 is a complete electrical specification which specifies driver-output and receiver input characteristics [49].

4.6.3 Mode

The RS 485 standard will only implement point-to-point connections instead of multi-point bus architecture. The reason for this is to exclude the possibility of a single point of failure in the system as well as increased data transfer speed. The RS 485 standard is used to communicate bi-directional as a half-duplex bus.

4.6.4 Cable selection

The RS 485 standard only specifies that a "Balanced Interconnection Media" and that "paired cables with metallic conductors should be employed" [50], omitting to specify a specific cabling solution. Various cabling solutions exist which adhere to these specifications. Examples of these cables are twisted pair, flat cable and ribbon cable. The question however remains which cabling solution must be employed in the ADES digiComm system.

One of the key requirements of the ADES is robustness, thus the need exists to employ a cabling solution that increases noise immunity immensely. Since flat and ribbon cables are very susceptible to differential noise that can corrupt the data, they will not be used as interconnection media in the ADES digiComm communication system.

Several companies manufacture cables especially to be used for the RS 485 standard. One of these companies is, Belden Inc. The cables that they designed and recommend for RS 485 standard are twisted pair cables. One of the advantages of twisted pair cables is that the twists keep the impedance constant over the length and in a case where noise is coupled into the medium, it is expected to be equal on both of the conductors – aiding in reducing noise susceptibility [50]. This results in obtaining a more robust and noise immune communication system.

In an effort to decrease the degree of noise susceptibility, various application guides were studied thoroughly. The universal suggestion was to use shielding to minimize the differential noise from coupling into the twisted media [51].

Thus aside from being able to transfer data at the required rate, the selected cable must also have the following attributes:

1. The cable must be shielded to reduce noise from coupling into the system.
2. The cable's shield must be connected to the digital ground. The grounding policy will be discussed in Section 4.6.5 to motivate this statement.

The cable that satisfies all the attributes is the BLDN9841 as shown in Figure 4-10.The BLDN9841 cable has the following specification: "one twisted pair 24 AWG with an aluminium-polyester shield and an overall tinned copper braid shield" [50]. In this particular cable the shield is directly connected to the ground [50].

Figure 4-10: Interconnection media - BLDN9841

4.6.5 Grounding

Grounding is also a vital part in any RS 485 network. As already mentioned in Chapter 2, RS 485 is designed to operate with a ground potential difference of +/- 7 V. Although this potential difference can easily be maintained during normal operation, during fault conditions for example lightning strike, it can result in damage or even failure of one or more devices connected on the RS 485 network. An excellent way to keep the ground potential within the limits is by running a third wire – which usually is the shield around a twisted pair. This creates a dedicated return path for the RS 485 network, reducing the noise coupled into the system due to current leaking from large equipment and ESD. Furthermore, as discussed in Section 4.6.4 the shielding will also aid in reducing noise coupled into the transmission line from outside sources.

The grounding policy for the communication system is shown in Figure 4-11. Each functional unit has its own isolated power supply. This allows each functional unit to be referenced to a single earth- star connection. The shielded twisted pair cables will also only be grounded at the main

controller side, to avoid ground loops. When the shield is connected at both functional unit ends there will be an additional path connecting the two grounds which will cause ground loops.

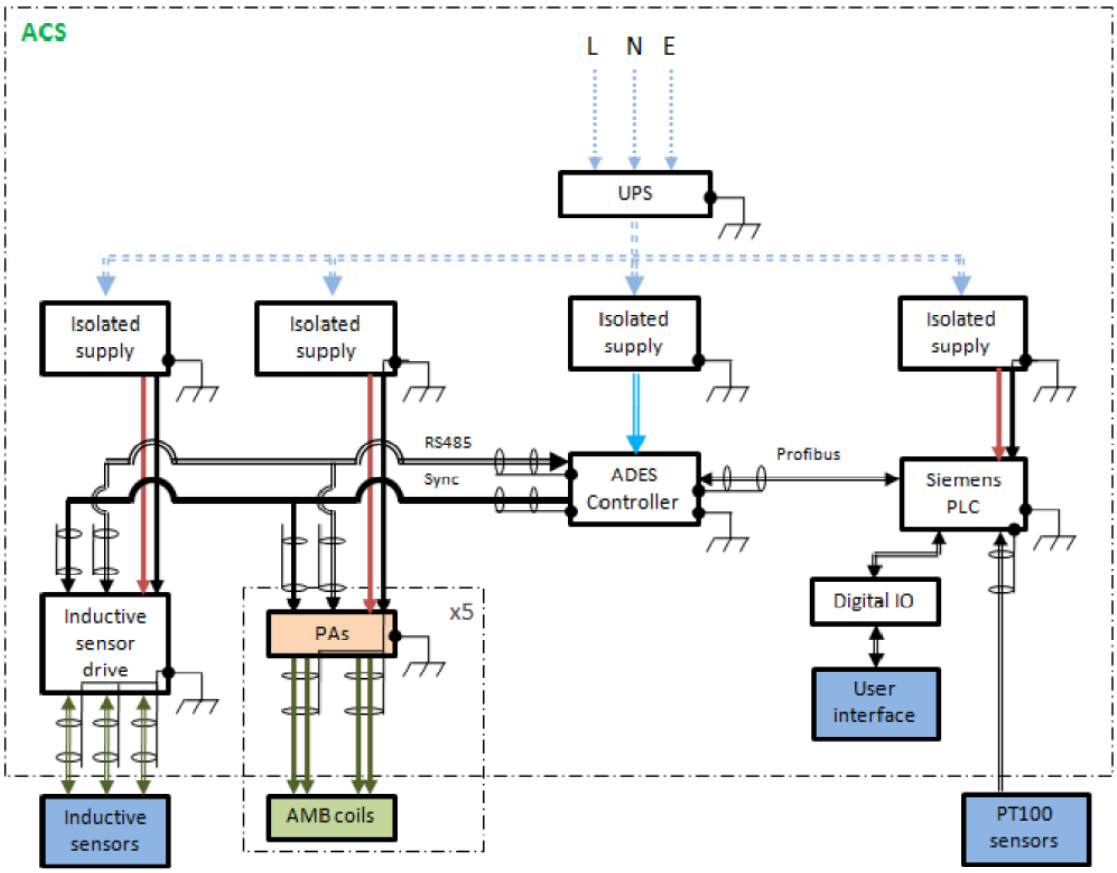


Figure 4-11: Grounding policy [52]

4.6.6 Drivers

4.6.6.1 Main controller

The RS 485 drivers that will be used on the main controller can reach a maximum data transfer rate of up to 20 Mbps. These drivers will be used to translate between RS 485 differential signalling to an LVTTL compatible voltage level. This particular part is supplied by Acromag Inc.

4.6.6.2 Power amplifiers, ISensorboard

The drivers used on the power amplifiers and the ISensorboard are also half-duplex transceivers designed for RS 485. This driver chip is supplied by Texas Instruments (part number -

SN65HVD3082E) and able to reach data transfer rates of up to 20 Mbps. For more information about the driver refer to Appendix B.2.

4.6.6.3 Motor drive

During the first stage of the ADES development, a motor drive was selected that adheres to the communication specifications as listed in Chapter 3. The specifications included a RS 485 communication interface capable of data transfer rates up to 1 kbps. A motor drive that met these specifications was obtained from Siemens Inc. However this drive communicates by means of an established USS (**U**niver**S**al **s**erial interface) protocol. During the second stage of the ADES project the motor drive will be developed as well, and the ADES digiComm protocol will be used to interface with the motor drive.

4.6.7 Termination options

It is important to know that termination becomes crucial when operating at high data transfer rates and over long cables. The ADES digiComm system operates at relatively high data transfer rates for RS 485 systems. Various termination options exist that can be used when implementing RS 485. The first option is no termination, which at this stage has already been ruled out, due to the high data transfer rates. The next option is to connect a single resistor across the conductor pair at each end. The resistor value needs to match the characteristic impedance of the cable. When cables are terminated in this way no reflection will occur and signal reliability is excellent [49]. The termination resistor was selected as 120 Ω, because the nominal characteristic impedance of the cable is 120 Ω [53].

4.6.8 Biasing

Another technique that needs to be discussed for RS 485 networks is fail-safe biasing. When applications implement asynchronous communication, fail-safe biasing becomes crucial. For example UARTs constantly check for a start bit which could either be a low or a high state. When data is not being communicated over the lines, the lines will most likely remain in a high or low state unless it is forced into an idle state by an active driver. In RS 485 systems this effect can become troublesome, because once there is no active drivers on the bus, the bus can become undetermined. This can result in possible false indications of start bits because when the RS 485 network is in tristate (idle) all the drivers are in receiver mode. When implementing fail safe biasing a known state is provided when there are no active drivers on the line preventing this effect.

Failsafe biasing is implemented by connecting a pull-up resistor to line B and a pull-down resistor to line A as shown in Figure 4-12.

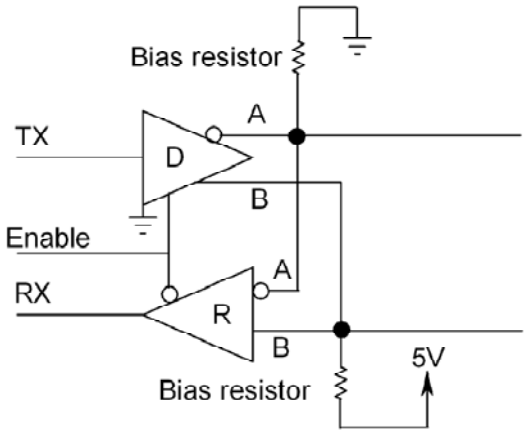


Figure 4-12: Failsafe biasing schematic [12]

The value of the biasing resistors is determined by the following procedure; consider each node on the bus to have an impedance of 12 kΩ. If there are two nodes on the bus (point-to-point) then the equivalent impedance is 6 kΩ, keeping in mind that the nodes are in parallel.

The termination resistor is selected as 120 Ω. Two of these resistors are in parallel thus the total resistance is 60 Ω. When the nodes and the termination resistors are placed in parallel the total resistance equals 59,4 Ω on the bus. Thus it can be derived that the termination resistors are dominant in the calculations and from here on the total bus resistance will be listed as 60 Ω.

The receiver sensitivity is 200 mV therefore the current required for the bus is;

$$\frac{200x10^{-3}}{60} = 3.33 \text{ mA}$$

The biasing is done from a 5V supply and the current required is 3.33 mA. Thus the ***total*** resistance required to deliver this current is;

$$\frac{5}{3.33x10^{-3}} = 1502\,\Omega$$

If the termination resistor subtracted from the total resistance, 1442 Ω is obtained. The biasing resistor is thus selected as 721 Ω. The final biasing and termination circuit is illustrated in Figure 4-13.

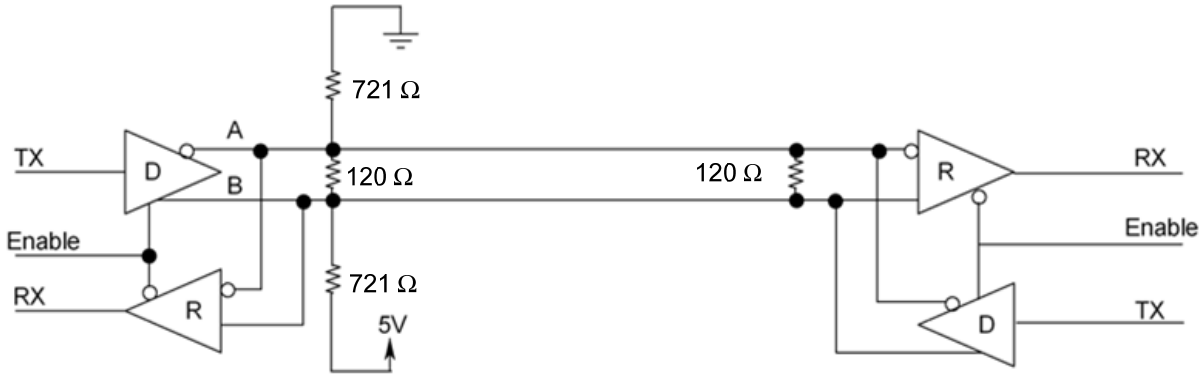


Figure 4-13: Fail-safe biasing circuit [12]

4.6.9 Isolation options

The drivers on the main controller and the other functional units are not isolated. Isolation will not be necessary for this application, keeping in mind that each of the functional units as well as the main controller are situated in the same controller box and not great distances apart or on different plants. The main controller and interfacing functional units are thus referenced to a single earth star connection as explained in Section 4.6.5.

4.6.10 Connector types and pin connections

The different connector types used in the communication sub-system of the ADES for each function units are given in Table 4-1.

Table 4-1: Connector types

Functional unit	**Connector**
Power amplifiers	Db-9
Motor drive	Db-9
ISensorboard	Db-15
Main Controller	SCSI-3

4.6.11 Encoding

Another aspect that needs to be defined in the physical layer is, whether encoding is going to be implemented and if so, which method to employ. Well-known encoding methods include Manchester encoding, 4B5B encoding and 8B10B encoding [54] .

Not every communication system implements encoding methods. Some of these communication systems are, ARCNET, Profibus, Serial Real-Time Communication (SERCOS) and Controller Area Network (CAN)[54]. These protocols address unique industrial requirements that pass data at lower data transfer rates (< 20 Mbps) over shorter connection distances (< 100 m). This is similar to the ADES digiComm protocol specifications .Therefore the ADES digiComm protocol will transmit and received un-encoded serial data that remain in their logic 1 or logic 0 states for the determined time intervals. This method is referred to as Non-Return-to-Zero (NRZ) representation of binary data [54].

4.7 Data link layer

"This layer ensures reliable communication over the physical layer. The protocol determines the structure of data i.e. the frame or packet size, and deals with aspects such as flow control, error detection and error recovery." [22]

First of all it is important to remember that the data link layer provides the network layer with connection between two nodes. The main details that need to be specified in the data link layer are [55];

1. Framing –To recognise the different parts of the information [55].
2. Medium access – For communication control when more than one node is connected on one bus [55].
3. Error control – For example implementing parity or cyclic redundancy check (CRC) to determine if erroneous data was transmitted or received. However this layer does not determine what to do when data is corrupt it is usually left for the upper layer to deal with corrupt data [55].

4.7.1 Framing

The ADES digiComm protocol is a universal serial protocol. This communication protocol defines a technique to access the different functional units in the ADES via a serial bus. The ADES digiComm protocol works by means of a master-slave principle. Each slave is connected to the master via a point-to-point communication link. The slaves are addressed by the master and can only transmit data when enabled by the master. Furthermore, no data transfer can commence between slaves.

Two different frames exist in the data link layer. These frames are discussed in the next sections.

4.7.1.1 Data Frame

The data frame will be used to carry data from the transmitters to the receivers. The data frame consist of 4 bit fields, a start of frame bit, data field bits, error field bit and end of frame bit field. Figure 4-14 illustrates the data frame. The start bit will be one bit, the data bits will be 16 bits, the parity will be one bit and the stop bit will be one bit. Even parity will be implemented to provide the first line of defence against possible data corruption.

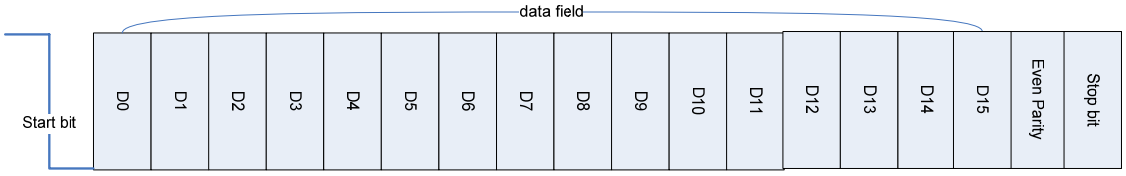


Figure 4-14: Data frame

The data frame and the error frame looks almost similar to the well known specified universal asynchronous receiver transmitter (UART) frame, with only one difference, the UART data field is 8 bits, however the data field of the ADES digiComm protocol will be 16 bits.

4.7.1.2 Error Frame

The error frame as shown in Figure 4-15 will be used to transmit a Cyclic Redundancy Check (CRC) to aid in more advanced error detection on the bus. The error frame is similar to the data frame; the only difference is that it carries the calculated CRC in the data field generated from all the previous data frames. How the particular CRC is calculated will be discussed in Section 4.9.5.

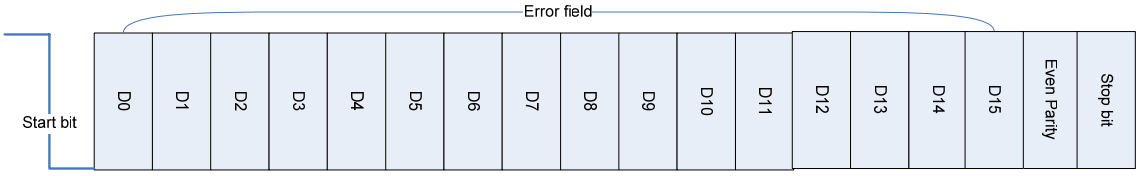


Figure 4-15: Error frame

4.7.1.3 Frame combination

The frames will work as follows; the data values will be transmitted in data frames. The transmitted data values will be used to calculate the CRC and then the CRC will be transmitted in

the error frame as shown in Figure 4-16. The amount of data frames depend on the functional unit. The last frame is the error frame that consists of the CRC,

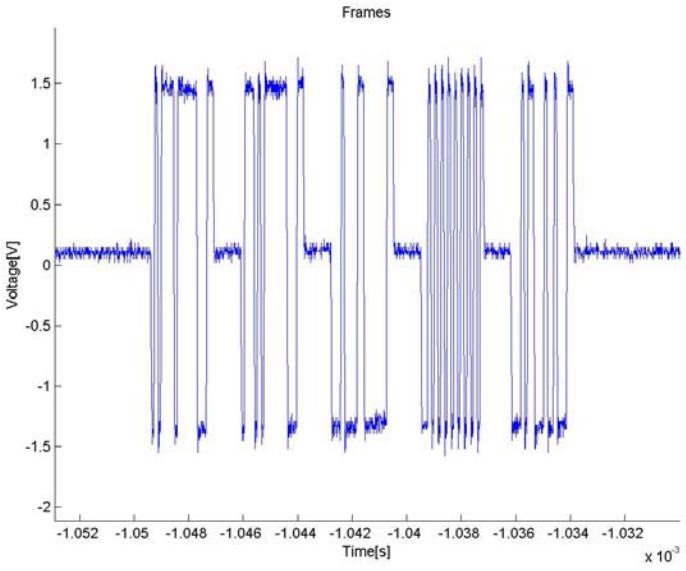


Figure 4-16: Frame description

4.7.2 Medium access control

A half-duplex, point-to-point connection is considered as shared media, because two nodes use the same transmission line. For applications that implement half-duplex, point to point connections; the media access control defined is very simple. The frames are placed on the media by one node and removed by the other node. Addressing will not be used considering that each of the nodes only has one destination and by adding addressing only unnecessary overhead will be added.

Data collision is prevented by controlling the direction pin of each of the drivers. When one node is transmitting the direction pin of the driver is set to high and the receiving node's direction pin is set low. This prohibits transmitting and receiving during the same time interval.

4.7.3 Error detection

Two error detection methods will be used in the ADES digiComm protocol. In the next section these error detection methods will be discussed thoroughly. The implementations however will be discussed in Section 4.9.5.

4.7.3.1 Method 1

Even parity will be implemented as the first line of defence against possible data corruption. Even parity is calculated by counting the number of ones in the message. If the number of ones is even

then a 0 is appended to the message. If the number of ones is not even a 1 is appended to the message. This method is able of detecting only one error. Mathematically a parity bit is merely the modulo 2 sum of the various bits and can be represented by (4-1) [55]:

$$p = m_1 + \cdots + m_n \tag{4.1}$$

where p represents the parity bit, m represent the data bits and n represent the number of data bits.

4.7.3.2 Method 2

In the ADES digiComm protocol Cyclic Redundancy Checks (CRCs) will also be used. By using this error detection method it will be possible to determine whether the data transmitted or received were corrupted. The question however remains why CRCs will also be implemented in the ADES digiComm protocol? A possibility always exist that a frame can be transmitted, but the calculated error detection function can still yield the correct frame check sequence (FCS). For example when the function used to detect errors only adds or subtracts frames to obtain the FCS. However when implementing CRCs the probability of this happening is reduced immensely and almost eliminated [56]. The basic idea of CRC will be discussed further in Section 4.9.5.1.

4.7.4 Error correction

Forward error correction (FEC) requires that more redundancy is added to the transmitted frames. The added symbols are used by the receiver to recover the correct data. Implementing FEC is complex and requires additional overhead which will lower the data transfer speed. At this stage error correction will not be implemented in the ADES digiComm protocol.

4.8 Protocol Implementation

The units that will be of utmost importance for the internal communications on the master node are shown in Figure 4-17.

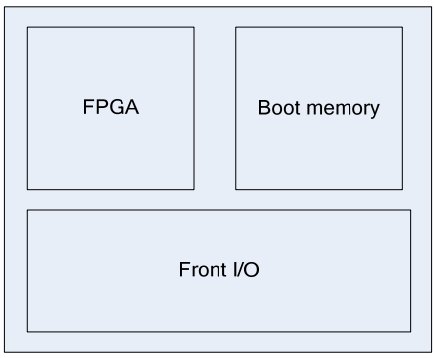


Figure 4-17: Important units in the functional architecture.

The different functional blocks that will be implemented on the FPGA of the master node are;

1. Dual port ram (DPR)
2. UART controller
3. UART transmit
4. UART receive
5. FIFO receive
6. FIFO transmit
7. Communication controller
8. CRC controller
9. UART top-level

Figure 4-18 illustrates how these functional blocks connect on the master node when interfacing with a **single** power amplifier. It is important to know that the UART receive, UART transmit, FIFO receive, FIFO transmit and the CRC controller will remain the same for all functional units. However the Communication controller and UART controller will differ depending on the interface.

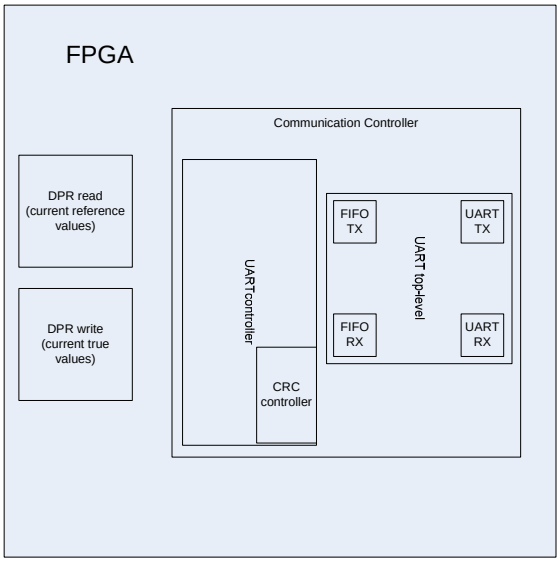


Figure 4-18: Functional architecture of master node

4.9 Basic functional blocks

Communication control will be done on FPGAs. This will result in faster performance [57]. Furthermore the digital communication system will be designed by means of state machines and described by using VHDL. In this section the detailed design of the basic functional blocks will commence.

4.9.1 Design of UART receiver entity

The UART receiver will make use of oversampling and a predetermined pattern as tabulated in Table 4-2. These two techniques are required, because no clocking information is available with the transmitter signal. Oversampling will be used to determine the midpoint of each received data bit. This is needed, because the receiver does not know the exact time when the start bit will be received. Thus the ideal will be to sample the incoming data bit not on the first falling or rising edge of a bit, but at the midpoint of each bit [58].

Table 4-2: Transmission parameters

Transmission parameters specification	**Specification**
Number of start bits	1
Number of data bits	16, LSB first
Number of parity bits	1
Stop bits	1
Start bit default	'0' (low)
Stop bit default	'1' (high)
Parity type	even
Over-sampling ratio	16

The oversampling technique in conjunction with the predetermined pattern works as follows and is illustrated in Figure 4-19.

1. Select the oversampling rate, S.
2. Determine the number of data bits to be received, N.
3. Determine the number of stop bits to be received, M.
4. Wait for the receiver to detect a falling edge, which indicates the start bit.
5. Create an internal counter to count to the S/2, now the midpoint of the start bit has been reached.

6. Clear the counter to 0 and resume.
7. Increment the counter with each clock cycle until the counter reaches 15.
8. When the counter reaches S, the midpoint of the incoming data bit has been determined.
9. Shift the value into a register.
10. Clear the counter to 0 and resume.
11. Repeat steps 8, 9 and 10 until all the data bits were received.
12. Repeat steps 8, 9 and 10 for the parity bit.
13. Repeat step 8, 9 and 10 for the stop bits.

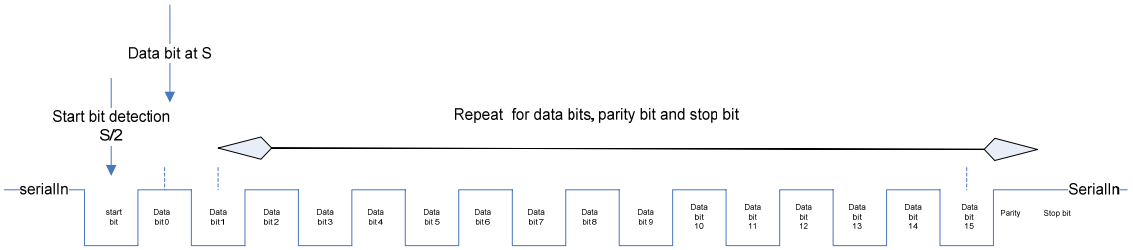


Figure 4-19: Graphical representation of the oversampling technique

The oversampling rate will be selected at 16 times the baud rate, the oversampling rate was selected high to ensure proper detection of start and stop bits as well aid in utilizing the midpoint of the each bit period to guarantee proper framing of the data. This method is not advisable for systems that require ***very high*** data transfer rates, the reason being that very high internal clocks must be generated on the processors. However the ADES employ FPGAs capable of generating system clocks of 133 MHz and the bit rate required is significantly lower [58]. To view the state machines refer to Appendix C.1.

4.9.2 Design of UART transmit entity

Now the focus shifts toward the design of the UART transmit component. The UART transmitter will make use of the same predetermined pattern as specified in Table 4-2. This component can be seen as a shift register, which shifts out data bit by bit at a selected bit rate. The UART transmitter module will use the same baud rate as the UART receiver module implying that an internal counter will be necessary to ensure that the UART transmit module only shifts out one bit every 16 enable ticks. This must be done to compensate for the oversampling on the UART receiver end. To view the state machine refer to Appendix C.2.

4.9.3 First In First Out (FIFO)

Now that the UART transmit and- receive components have been designed and discussed, the focus shifts toward developing a component with the following abilities:

1. The ability to signal the UART transmitter and - receiver when new data is available [58].
2. The ability to avoid re-transmission of the same data numerous times [58].
3. The ability to provide a buffered space between the UART controller and the UART transmitter and – receiver [58].
4. The ability to ensure that data is always available to be accessed by the UART and the UART controller.

For this particular use, a FIFO (first-in-first-out) buffer – which is an elastic storage unit – will be implemented [58]. The conceptual diagram of a FIFO buffer is shown in Figure 4-20.

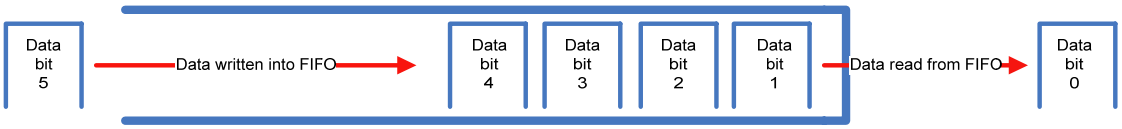


Figure 4-20: The FIFO concept

This component works by means of two control signals, a read and write signal. If the write signal is asserted, the UART will write into the FIFO. When the read signal is asserted the first input of the FIFO is "removed", and now the next item becomes available.

4.9.4 Dual Port Ram (DPR)

Another very important component that must be discussed is the DPR. This component will be used as the memory interface between the communication controllers and the other system controllers. An illustration of this component is shown in Figure 4-21.

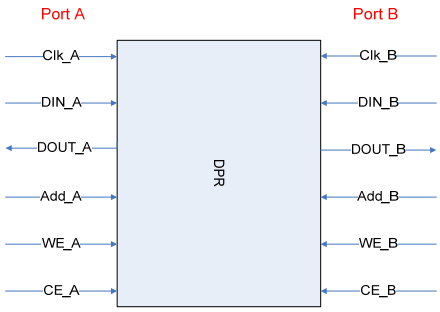


Figure 4-21: DPR component entity

The main functionality of this component is that it has the ability to write data into the RAM and read data from the RAM via two ports. Data can be written into the DPR by setting chip enable (CE_B) and write enable (WE_B) high, selecting the memory space (Add_B) where the data must be written into and writing the data into DIN_B. Data can be read out of the DPR by setting chip enable (CE_B) high and selecting the memory space (Add_B) where that data must be read from.

The data will then be written out on DOUT_B. This works precisely the same for port A. Both the ports can be accessed at the same time; however the same memory space cannot be accessed at the same time.

4.9.5 CRC function

4.9.5.1 The basic idea

The main aim of this error detection technique is to determine if a message that was transmitted was corrupted due to a noisy channel. This is done by constructing a Frame Check Sequence (FCS) and transmitted the frame check sequence after the data has been transmitted. After the messages and the FCS value have been received the receiver uses the data values received to also calculate the frame check sequence. If these two frames do not correlate the assumption is made that the message was corrupted.

The basic idea behind CRC algorithms are as follows:

1. The message as a whole is treated as a binary number.
2. This binary number is then divided by another fixed binary number and the remainder after division is referred to as the frame check sequence or checksum.
3. This checksum or frame check sequence is then transmitted. In the case where the sender and the receiver FCS do not match a negative acknowledgment signal can be sent by the receiver, indicating faulty transmission and prompting re-transmission.

4.9.5.2 Polynomial and CRC arithmetic

Before commencing to the implementation of the CRC algorithm in VHDL two key concepts has to be discussed. The first concept is a term used numerous times when referring to CRC algorithms, that term is **polynomial**. A given CRC algorithm will always work together with a particular polynomial. What is a polynomial? The **message** to be communicated is referred to as the "divided", the **generator polynomial** is referred to as the "divisor" and finally the **CRC** is referred to as the "remainder". In CRC algorithms these three values are not considered to be positive integers, but polynomials with binary coefficients. For example, a decimal value 19 is 13 in hex and 10011 in binary and corresponds to the polynomial:

$$x^4 + x^1 + 1 \tag{4.2}$$

The second concept that must be noted is that CRC arithmetic is equivalent to XOR operations at various shifting offsets. Finally the CRC arithmetic operation can be summarized as follows:

1. Choose a generator polynomial *G* of width *W*.
2. Append *W* zeros to the original message *M*.
3. Divide *M* by *G* by using CRC arithmetic.
4. Send the CRC value in the error frame

4.9.5.2.1 Selecting a polynomial

The next question that arises is what CRC polynomial to select for the ADES digiComm protocol? The ADES digiComm protocol is developed for an embedded system network. For embedded networks the property of interest is the hamming distance (HD) and the burst error detection. The Hamming distance can be defined as: *"The minimum number of bit inversions that must be injected into a message to create an error that is undetectable by that message's CRC-based FCS."* [59] For example when a CRC has a HD=7 then when a combination of 1, 2,3,4,5 or 6 inversions do exist in a message; these errors will be detected. However when a combination of 7 or more bit inversions have occurred a possibility exists that the errors will be undetected by the CRC. Another property of interest is the burst error detection potential. A burst can be defined as a block starting with an error and finishing with an error, whose transitional bits can contain errors or no errors [56]. In this section CRC codes will only be considered if the burst error detection potential is equal to the CRC polynomial length.

The customary technique followed to select a CRC polynomial is to use already commonly used CRCs. However some commonly used CRC provide less significant error detection capabilities than others. Furthermore it is important to know that when selecting a good CRC polynomial, ***it does not only involve the size of the CRC but also the size of the data transmitted in the specified data frame (excluding the CRC).*** For example, when selecting the CCITT-16 bit polynomial which is; $x^{16} + x^{12} + x^5 + 1$ it is generally assumed that this polynomial will provide better error detection capabilities than a smaller polynomial. This however is not the case. The CCITT-16 bit CRC can detect any possibility of three bit errors or fewer at a message length of 48 bits. However with a HD=4, 84 of all possible 4-bit errors go undetected. In comparison the CAN CRC polynomial is one bit smaller and can detect 5 randomly distributed errors for message lengths up to 112 bits [59][60].

Therefore a CRC polynomial will be selected that can achieve the maximum HD for the specified data frame length. After careful consideration and studying of the tables in Appendix B.3 it was decided to use the CAN 15 bit CRC polynomial for the following reasons. Firstly the CRC performance is nearly optimal for transmissions of 64 bit frames. Secondly the polynomial provides a HD= 6 for data lengths up to 112 bits (the most data that will be transmitted at a time will be 80 bits), and lastly it is a familiar and standard CRC polynomial [59] .

4.9.5.2.2 Implementing CRC algorithm in VHDL

VHDL is a hardware descriptive language meaning that a hardware circuit needs to be developed to generate the correct CRC checksum. This will require reducing the polynomial division process to its fundamentals. For the purpose of discussing the detailed design, the selected CAN 15 CRC will be used. The polynomial is given in (4.3):

$$G = x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1 \tag{4.3}$$

This corresponds to binary 1100010110011001.

This polynomial division will employ a shift register which will be specified according to the selected generator polynomial *G*. Figure 4-22 is a graphical representation of the hardware implementation of generating a CRC.

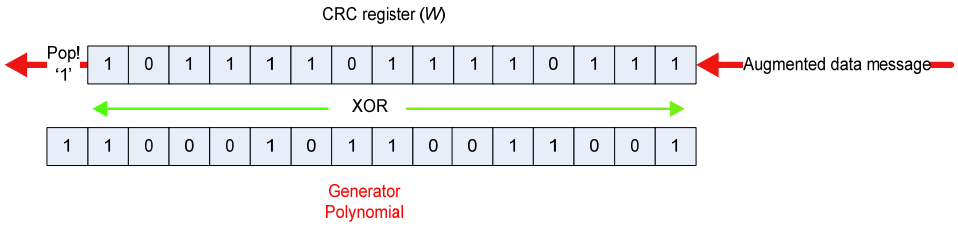


Figure 4-22: Graphical representation of the CRC hardware implementation

The hardware division process can be described and coded as follows:

1. Instantiate a CRC register of length *W* (*W* equals the degree of *G*).
2. Initializing the CRC register to all zeros.
3. Feed the augmented message (message with the appended zeros) through the division register.
4. When a '1' pops out of the register, XOR the CRC register with *G*. When studying Figure 4-22 it is seen that the highest order bit of the generator polynomial is excluded. The reason being that the highest order bit of the generator polynomial is always a '1' and when the '1' which has popped out and the '1' of the highest order bit of *G* is XORed this will always result is a '0' as can be seen from the XOR truth table shown in Table 4-3.

Table 4-3: XOR truth table

P	Q	P XOR Q
0	0	0
0	1	1
1	0	1
1	1	0

These steps can be implemented in hardware by using a Linear Feedback Shift Register (LFSR). A linear feedback shift register consists of D-flip-flops and XOR gates. In Figure 4-23 a linear feedback shift register for the CRC-15 is shown.

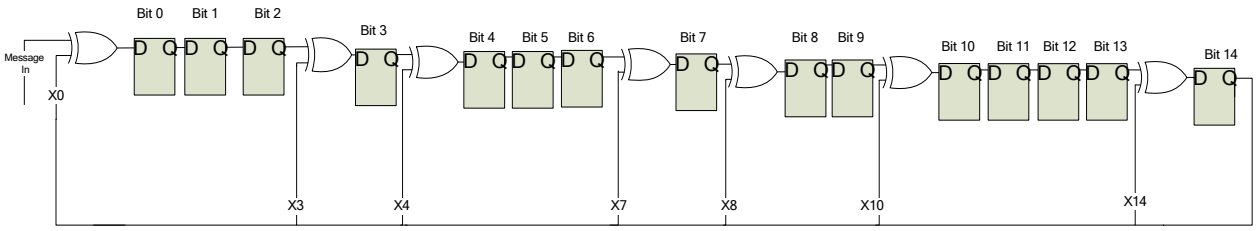


Figure 4-23: CRC 15 polynomial division circuit

As shown in Figure 4-23 the shift registers equal the degree of the generator polynomial, G. The XOR gates create a feedback for the LSFR acting as a tap controller for the generator polynomial. After the entire message, M has been shifted out the final bits in the shift registers are the remainder [61][62].

For the ADES digiComm protocol all the data values transmitted in the data frames will be concatenated to form the message M. The message M will be supplied to the CRC controller to be calculated. Once the CRC controller is finished the CRC will be transmitted in the error frame. On the receiver side, the message M will be calculated by concatenating all the data frames. Once all the data frames have been received, M is supplied to the CRC controller, where after the calculated CRC is compared with the CRC received in the error frame.

The VHDL code will be written by using D-flip-flops shift registers and XOR gates. The assumption can therefore be made that the VHDL code written to calculate the CRC will be synthesized on the FPGA as the circuit shown in Figure 4-23.

4.10 Slave nodes

The slave nodes that require communication controllers for the first version of the ADES are;

1. Power amplifiers
2. ISensorboard
3. Main controller

The functional blocks that are crucial for the ADES digiComm system to function are shown in Figure 4-24. In the previous section the focus was primarily on the basic functional blocks, now the focus will shift toward the design of the ***communication controllers*** and the ***UART controllers*** keeping in mind that these controllers differ for each functional unit.

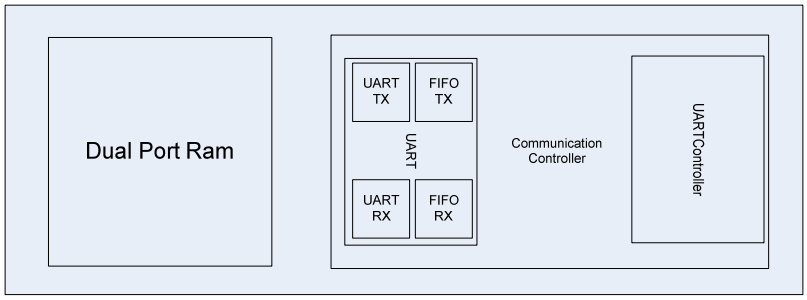


Figure 4-24: Functional architecture of the slave nodes.

4.11 Communication control

Communication control will be implemented on FPGAs. This will result in faster performance [57]. The design of the communication control modules will be done by means of state machines and coded by using VHDL.

The procedure that will be followed to discuss the communication controllers will be as follows:

1. First the interconnection between the master and the slave module will be discussed by providing a functional architecture.
2. Secondly a brief overview of the functionality of each of the blocks present in the functional architecture will be discussed.
3. Thirdly the state machines implemented on each of the communication controllers will be described.

4.12 ISensorboard and main controller interconnection

The first interface that will be discussed is between the ISensorboard and the main controller. As already specified, the ISensorboard is the slave and the Main controller is the master and these two units are connected by two RS 485 shielded twisted pair cables. The various functional blocks used to implement communication control between these units are illustrated in Figure 4-25. In this section the functionality of each of these blocks will be discussed.

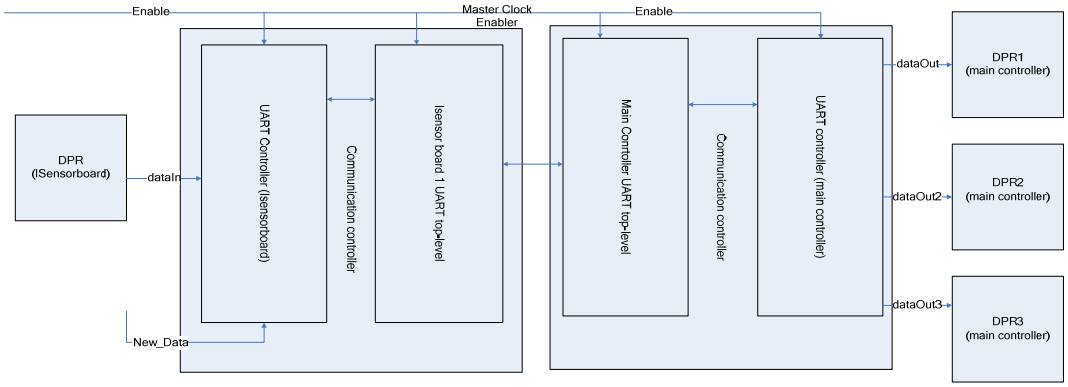


Figure 4-25: ISensorboard and Main controller communication functional architecture

4.12.1 DPR

DPR units are instantiated on both the ISensorboard and the Main controller. The DPR on the ISensorboard is used as the memory space where the X, Y and Z position values are written into. These values can then be accessed by the communication controller. Three different dual port rams are also instantiated on the main controller. These DPRs are used to store the X, Y and Z position values in separate DPRs. Different error codes are also stored in the DPR to be analysed by the main controller.

4.12.2 UART– top level modules

The UART – top level modules is the next higher assembly unit which consists of four lower assembly units, a RX - and TX FIFO unit and a RX- and TX UART unit. The UART units are responsible for data framing, implementing the first error detection and transmitting or receiving data over the specified link. The FIFO units are used as elastic storage units between two subsystems.

4.12.3 UART controller module (ISensorboard)

A UART controller is instantiated on the FPGA of the ISensorboard. The main function of the UART controller is to obtain the data from the ISensorboard memory space, calculate the CRC and transmit the data to the main controller. By default ***all*** the UART controller is triggered (enabled) by the ***same*** master clock every 20 kHz, however when the master clock is lost due to noise the UART controller will receive prompt from the main controller which will trigger the transmission of the X, Y and Z position values.

The UART controller must flag the error conditions listed in Table 4-4.

Table 4-4: Error conditions

Error number	Condition
Error 1	CRC error – previous data received was corrupted

4.12.4 UART controller module (Main controller)

A UART controller is instantiated on the Virtex 5 FPGA located on the main controller. This UART controller receives the position values from the ISensorboard and checks for possible data corruption. If the ISensorboard stops transmitting position values the UART controller situated on the main controller prompts the UART controller on the ISensorboard to start transmitting. This will ensure that when the synch signal is lost, that the ISensorboard still functions.

The UART controller must flag the error conditions listed in Table 4-5.

Table 4-5: Error conditions

Error number	Condition
Error 1	Receiver timeout due to loss of sync signal
Error 2	CRC error – previous data received was corrupted

4.12.5 Communication controller modules

The communication controller modules are the next higher assembly units that consist of two lower assembly units; the UART top-level and the UART controller. The designed communication controllers will be instantiated twice, keeping in mind that there are two communication channels between the ISensorboard and Main controller. In the next section the communication controllers will be described in detail using state machines.

4.13 ISensorboard communication controller

4.13.1 TX state machine

As previously mentioned, the communication controllers will be designed by using state machines. Only the ISensorboard communication controller and the interfacing communication controller on the main controller will be discussed, for the rest of the state machines refer to

Appendix C.3 and Appendix C.4. The first state machine that will be discussed, is the TX state machine implemented in the communication controller situated on the ISensorboard. Figure 4-26 shows the state machine and Table 4-6 describes the states. The state machine should be interpreted according to the following convention:

1. The name of each state is written in the circle.
2. The double circle indicates the start of the state machine
3. The important processes executed in each state are written next to the circle.
4. Events that triggered state transitions are written near the arrows.

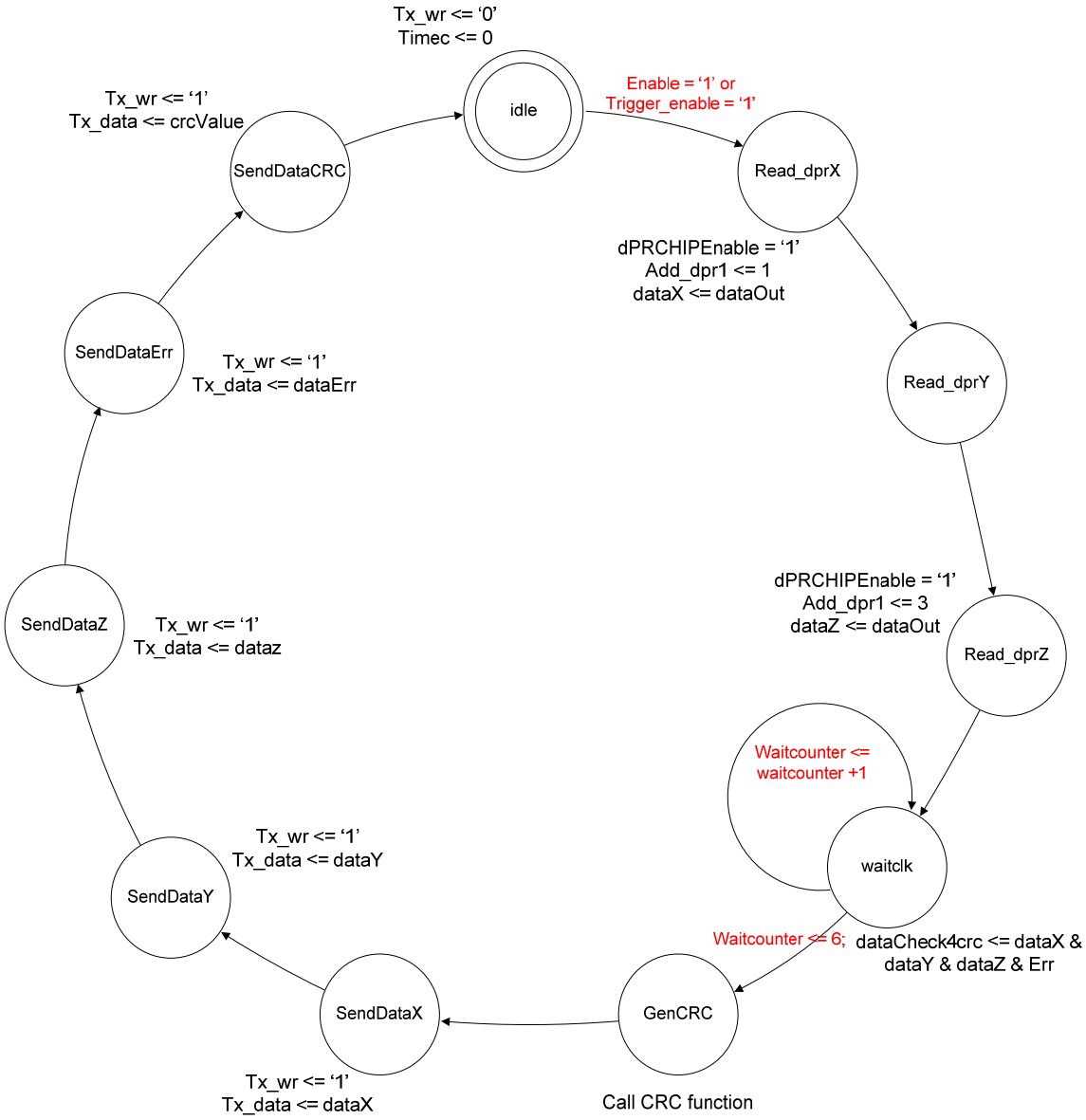
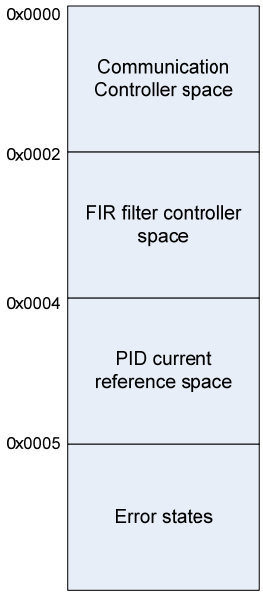


Figure 4-26: Transmitter state machine on the ISensorboard

Table 4-6: State description of TX process

State	State description
Idle	During the idle state, no data will be transmitted. Data transfer will only start when the communication controller is enabled by the master clock or prompted by the main controller. After the communication controller has been enabled the new data input will be checked. The new data input checks whether new data is available in the DPR. If no new data is available an error will be flagged and state transition occurs. If new data is available an error will not be flagged.
Read_dprX, Read_dprY, Read_dprZ	During the Read_dprX, Read_dprY and Read_dprZ states data will be read from the DPR. During each of the respective states the data will be read from a different specified address. Once the position value is obtained the Read_dprX state changes to the Read_dprY state. Once the last value was read out of the DPR the Read_dprZ state transitions to the waitclk state. Memory spaces are allocated to all the controllers which forms a part of the ADES control system. All the communication controller data will be written in between address space 0x0000 to address space 0x0002. 
Waitclk	During the waitclk state the X, Y, Z and Err values are concatenated. This is done to obtain the data necessary for the CRC function. Concatenating the values takes more than one clock cycle; therefore a counter is used to avoid timing errors. After the counter has timeout out state transition occurs.

GenCRC	During the GenCRC state the CRC value is calculated by calling a function. The functions name is next_crc and the value returned is the crcValue. The CRC function will be discussed later on. After the crcValue is returned, state transition occurs.
SendDataX, SendDataY, SendDataZ, SendDataErr, SendDataCRC,	During these states the data is written into the TX FIFO by strobing the write control signal high for the duration of all the states. The UART transmitter is informed that new data is available in the FIFO to be transmitted. Once all these states have executed the SendDataCRC state transitions back to the default state.

4.13.2 RX state machine

Two different state machines execute continuously on the ISensorboard communication controllers. In the previous section the TX state machine has been discussed. In this section the focus will shift toward the RX state machine as illustrated in Figure 4-27, where after the states will be explained in Table 4-7.

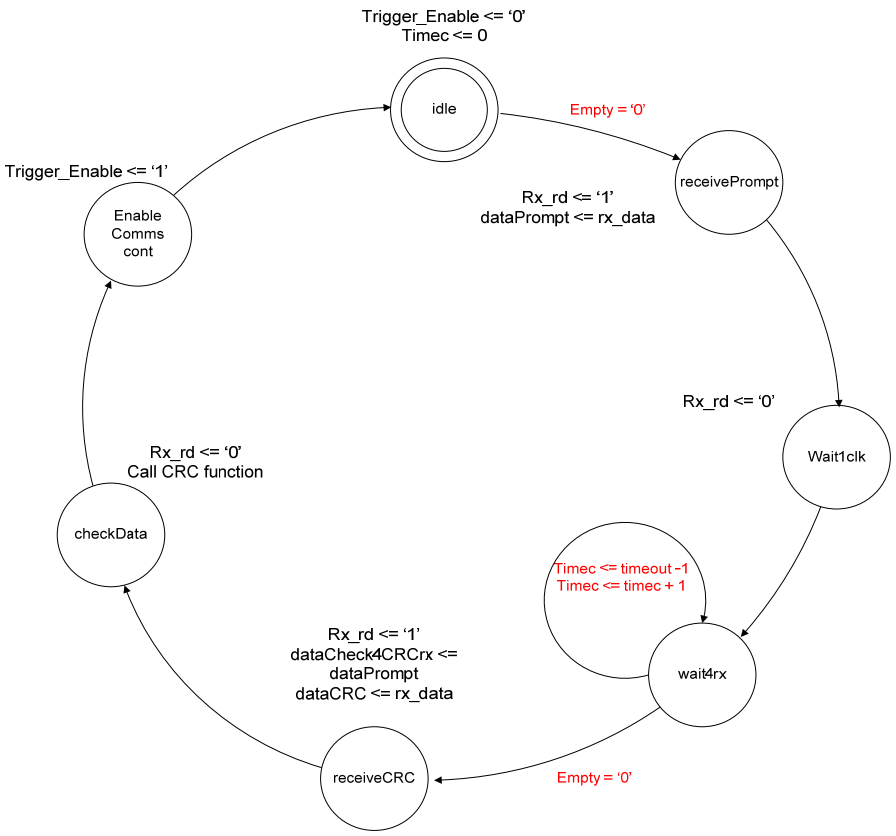


Figure 4-27: Receiver state machine on ISensorboard.

Table 4-7: State description of RX process

State	State description
Idle	During the idle state, the system is waiting for a prompt signal from the main controller indicating that the sync signal or master clock has been lost. Once the empty signal goes low, a prompt has been received and the ReceivePrompt state is entered. If the empty signal stays high no prompt has been received and the RX state machine will remain in the idle state.
ReceivePrompt	In this state data available in the FIFO buffer is read into the communication controller. After the data has been read a state transition occurs.
Wait1clk	The wait1clk state is invoked to prevent an off by one timing error.
Wait4rx	During the Wait4rx state, the system waits for another value to become available in the FIFO. If a value becomes available the FIFO empty signal will become '0'. If the empty signal becomes '0' a state transition occurs.
ReceiveCRC	In this state the CRC value is read out of the FIFO and a state transition occurs once again.
checkData	In this state the received CRC is checked by calling the CRC function and calculating the CRC with the data received in the ReceivePrompt state. If the CRC value calculated and the CRC value received does not correspond an error will be flagged. After the value has been calculated and compared state transition occurs.
enableCommsCont	During the enableCommsCont state the ISensorboard sets the Trigger_enable signal to '1'. This will trigger the TX procedure to start transmitting the X, Y and Z values, ensuring that the ISensorboard will still communicate with the main controller although the sync signal is lost. State transition occurs to idle.

4.14 Main controller communication controller

4.14.1 Main controller state machine

The communication controller situated on the main controller that interfaces with the ISensorboard consists of only one state machine that transmits and receives sequentially. The state machine is illustrated in Figure 4-28 and the state descriptions follow in Table 4-8.

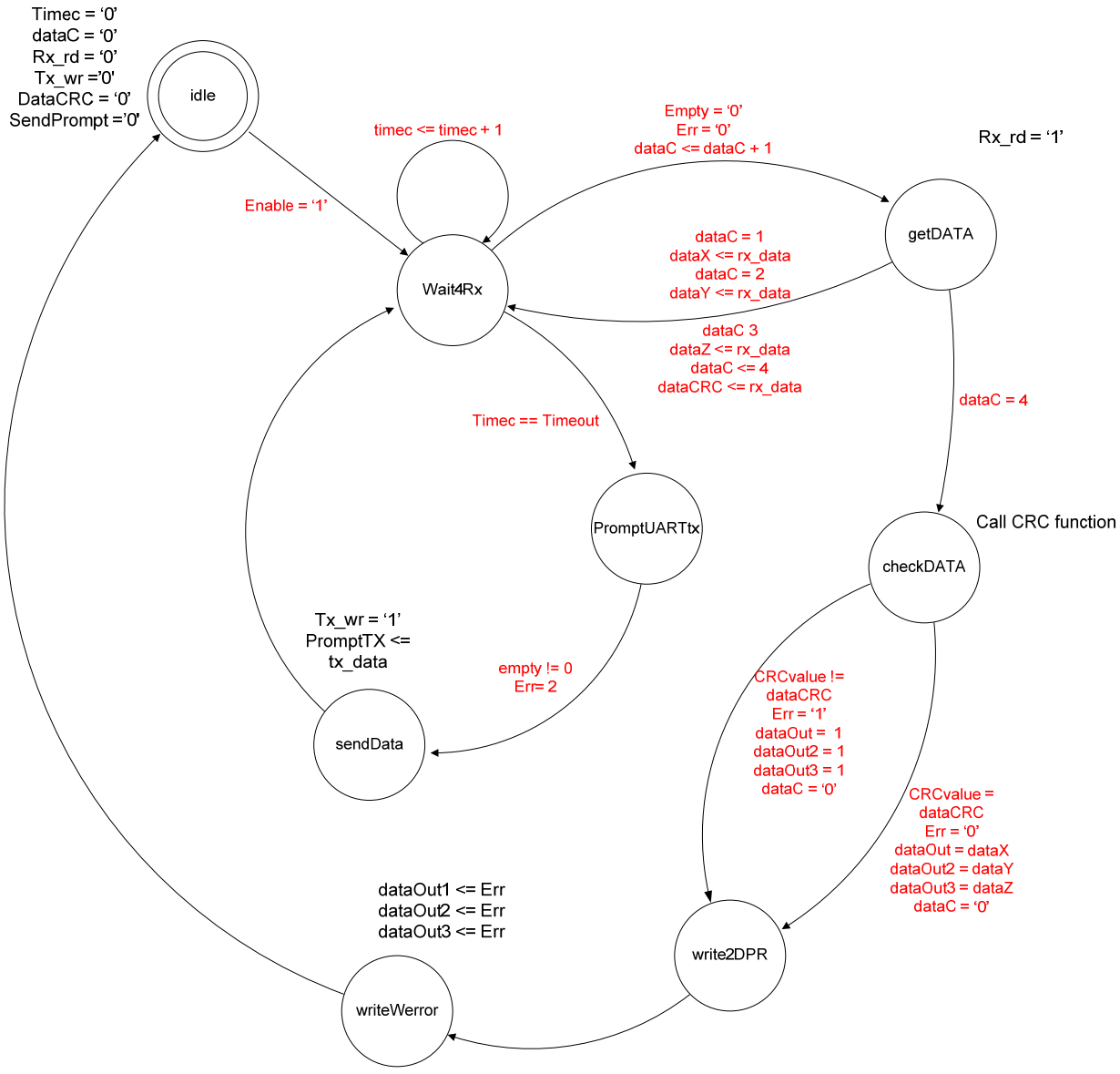


Figure 4-28: Communication controller state machine on the Main controller

Table 4-8: State description of the communication controller situated on the main controller

State	State description
Idle	During the **idle** state the communication controller waits to be enabled by die sync signal. Once the communication controller has been enabled, state transition occurs to **wait4rx**.
Wait4rx	During the **wait4rx** state, a timeout is incremented. Once data becomes available in the FIFO by setting the empty signal to '0' state transition occurs to the **GetData** state. In the case where no data becomes available in the FIFO, the timeout will be reached and state transition will occur to the **PromptUARTtx** state.
GetData	In the **getdata** state, the data available in the FIFO is read out. Once all the data is read out state transition occurs to **checkData**.
PromptUARTtx	This state will prompt the ISensorboard to transmit data back to the main controller, in the event that the sync signal malfunctioned. Once the error has been flagged state transition occurs to **sendData**.
SendData	The prompt command is now send to the ISensorboard. State transition occurs to the wait4rx state.
CheckData	In this state the received CRC value from the ISensorboard is verified. State transition occurs to **write2dpr**.
Write2DPR	If the CRC values corresponded, the new X, Y and Z positions are written into the respective DPRs. If the CRC values did not match, 1 is written in the DPRs indicating that the previous sensor values must be used. State transition occurs to **WriteError**
WriteError	In the **WriteError** state, all the errors flagged are written into the DPRs specified. After these commands have executed state transition occurs to the default state **idle**.

4.15 Power amplifier and Main controller interconnection

Figure 4-29 shows the various functional blocks and how they are connected. In this section the functionality of each of these blocks will be discussed briefly. It is important to note that each of these components will be instantiated five times, because there are five power amplifier boards.

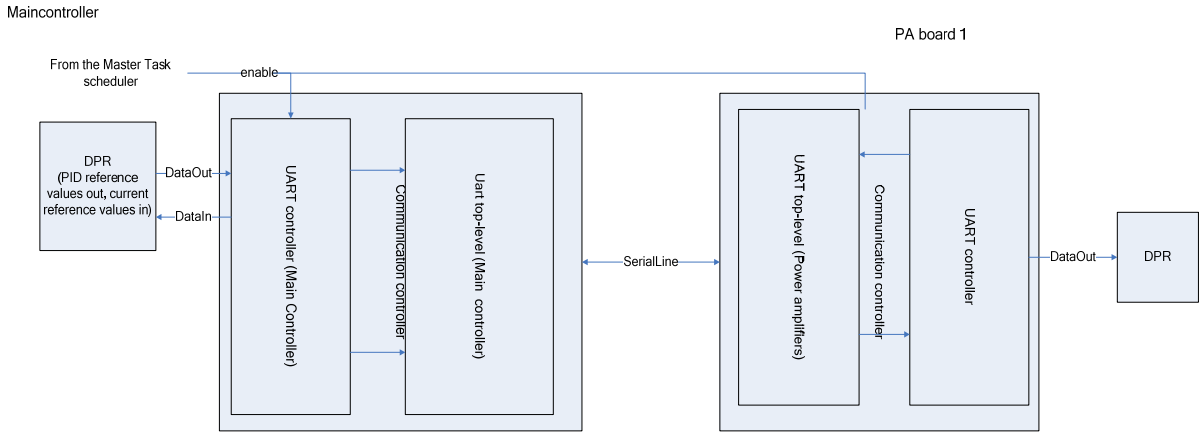


Figure 4-29: Connection between the power amplifiers and the main controller

4.15.1 DPRs

The DPR situated on the main controller will be used as the memory space between the PowerPC and the communication controller. The current reference values are read out of the DPR and the true current values are written into the DPR. The DPR situated on the power amplifier will be used as the memory space between the power amplifier controller and the communication controller.

4.15.2 UART top-levels

Refer to Section 4.12.2.

4.15.3 UART controller (main controller)

A UART controller is instantiated on the FPGA of the main controller. The main function of the UART controller is to retrieve data (reference current values) from the allocated memory space when new data becomes available. When new data becomes available the CRC is calculated. The reference value, power amplifier status (on or off) and the CRC value are transmitted to the power amplifiers, where after the UART controller waits to receive two true current values and error condition and a CRC. After the data is received, the UART controller tests for possible data corruptions by calling the CRC function. The calculated CRC value and the received CRC value are compared; in the event of a CRC mismatch an error is flagged.

The UART controller must flag the following errors conditions as listed in Table 4-9.

Table 4-9: Error conditions

Error number	Condition
Error 1	No new data available
Error 2	Receiver timeout occurred
Error 3	CRC error
Error 4	Incomplete data received

4.15.4 UART controller (power amplifiers)

The UART controller implemented on the power amplifier will receive data from the main controller and check for possible data corruption, by calling the CRC function. In the event of a CRC mismatch a '1' will be written into the DPR, indicating that the previous current reference value must be used. If no data corruption had occurred the received current reference value will be written into the DPR to be accessed by the power amplifier controller.

After the values have been received the true current values will be read out of the DPR and the true current values together with a CRC will be transmitted back to the main controller.

The UART controller must flag the following error conditions as listed in Table 4-10.

Table 4-10: Error conditions

Error number	Condition
Error 1	CRC error
Error 2	Receiver timeout occurred

4.15.5 Communication controller modules

Refer to Section 4.12.5. For more information about communication controller design refer to for this interface refer to Appendix C.3 and Appendix C.4

4.16 Conclusion

This chapter focussed on two important components of the communication sub-system for the ADES. Initially it focussed on the hardware selected according to the specified architecture and finally it focussed on the detailed design of an internal protocol for AMB systems. In the next chapter the focus will shift toward verifying and validating this protocol.