BiSS Interface

BiSS LINE PROTOCOL DESCRIPTION

preliminary

FEATURES

- One-Cable Technology
- Combined transmission of power and data signal
- 2-wire and 4-wire Cable Support
- Bidirectional Asynchronous Serial Communication
- Forward Error Correction
- Point-to-Point and Multi Slave Support
- BiSS C Protocol Compatibility
- Compact and Cost Effective
- Open Standard

APPLICATIONS

- Single Cable Systems
- Motor Feedback Systems
- Sensor/Actuator Interface
- Smart Sensors
- Rotary Positioning
- Linear Positioning
- Multi-Axis Systems
- Condition Monitoring with Multi-Sensor Systems
- High Disturbance Environments
- BiSS C Extension

OVERVIEW

The open source *BiSS Interface* protocol implements a real time interface for a digital, serial and secure communication between drive, sensor and actuator. It is used on the lower sensor/actuator communication level in industrial applications which require short cycle times, safety, flexibility and minimum implementation effort.
In addition to its technical advantages, two characteristics have ensured the success of the global standard: the free license of *BiSS* for applications and the stability and continuity of the protocol itself since its introduction.

BiSS Line follows the industrial trend of fully digital communication, functional safety capabilities and motor power supply over a single cable. However, *BiSS Line* provides all specific features of *BiSS C*.
The data transmission is based on the industrial standard RS485 and is protected by Forward Error Correction (FEC) against disturbances coupled from the motor line to the data line.
For example, the cycle time for FEC protected sensor data of 128 bit sent over a cable with a length of up to 100 m is about 31.25 µs.

For *BiSS Line* communications, there is one BiSS Line Master (BLM) controlling the data transfer with the BiSS Line Slaves (BLS) in both directions. The BLM is connected to the Host and the BLS are connected to the actuators and sensors of the system. In a *BiSS Line* point-to-point topology one BLM is connected to one BLS. In a *BiSS Line* bus topology multiple BLS are connected to one BLM as shown in Figure 1.

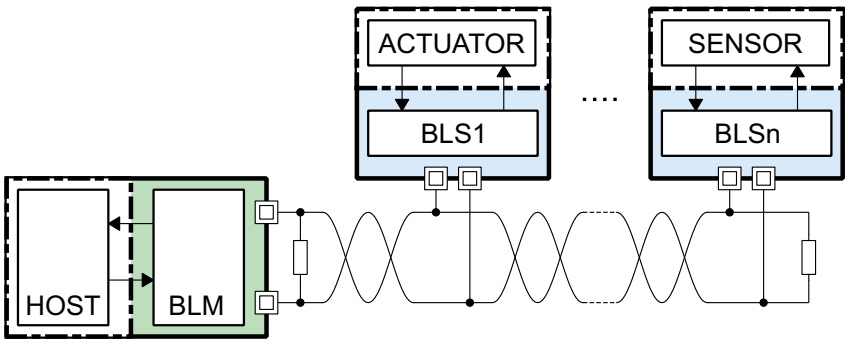

Figure 1: General *BiSS Line* Topology

The communication between the BLM and BLS is organized in frames. Two types of frames are available that can be used for data transmissions:

- Process Data Frame (PDF)
- Auxiliary Data Frame (ADF)

The PDF is primarily used for cyclic transmission of sensor and actuator data. Once configured, the PDF is automatically triggered by the BLM. The actuator and sensor data are referred to as Process Data. Similar to the *BiSS C* frame, the PDF includes Single-Cycle Data (SCD) that is completely transmitted within one frame cycle and Control Data that need several frame cycles to be transmitted completely. The Control Data are merged into the Control Data Frame that can be used for register communication without interrupting the transmission of the Process Data.

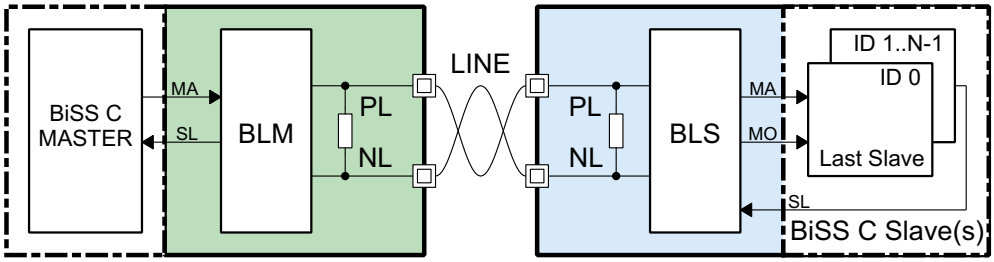

Figure 2: Example for *BiSS Line* Extension of a *BiSS C* System (Point-to-Point Topology)

Due to its similarity to the structure of *BiSS C*, the PDF can easily be used to transform *BiSS C* into *BiSS Line* communications. Figure 2 shows the extension of a *BiSS C* system into a *BiSS Line* system by connecting

the BLM and BLS in between the *BiSS C* devices. However, *BiSS Line* provides an independent data transmission between BLM and BLS that can be controlled by a Host through an arbitrary interface. The Serial Peripheral Interface (SPI) is typically used for on-board Host interfaces while parallel interfaces are usually implemented on-chip (within ASICs and FPGAs).

The ADF explicitly addresses a BLS to exchange common data, that does not need to be transmitted periodically. It can be used for fast configuration of one BLS before starting the cyclic transmission of the PDF, for example. Additionally, the ADF provides several opcodes for controlling data transmission and enables bus topologies with several BLS.

BiSS LINE SYSTEM

The *BiSS Line* system contains a BLM connected to one or several BLS. In this document, the elements related to the BLM are colored in green while BLS related elements are colored in blue.

In Figure 3, the *BiSS Line* components for a motor feedback system are shown. The Drive on the left hand side includes the BLM, which is controlled by the Host, either through a parallel or a serial data interface. The Host and the BLM are connected to the Motor on the right hand side via one common power and feedback cable only. The Encoder is used to transmit the Motor's rotary position information to the Drive. Therefore, the Sensor of the Encoder is connected to the BLS. The communication between the BLS and the Sensor follows the standard *BiSS C* communication protocol.

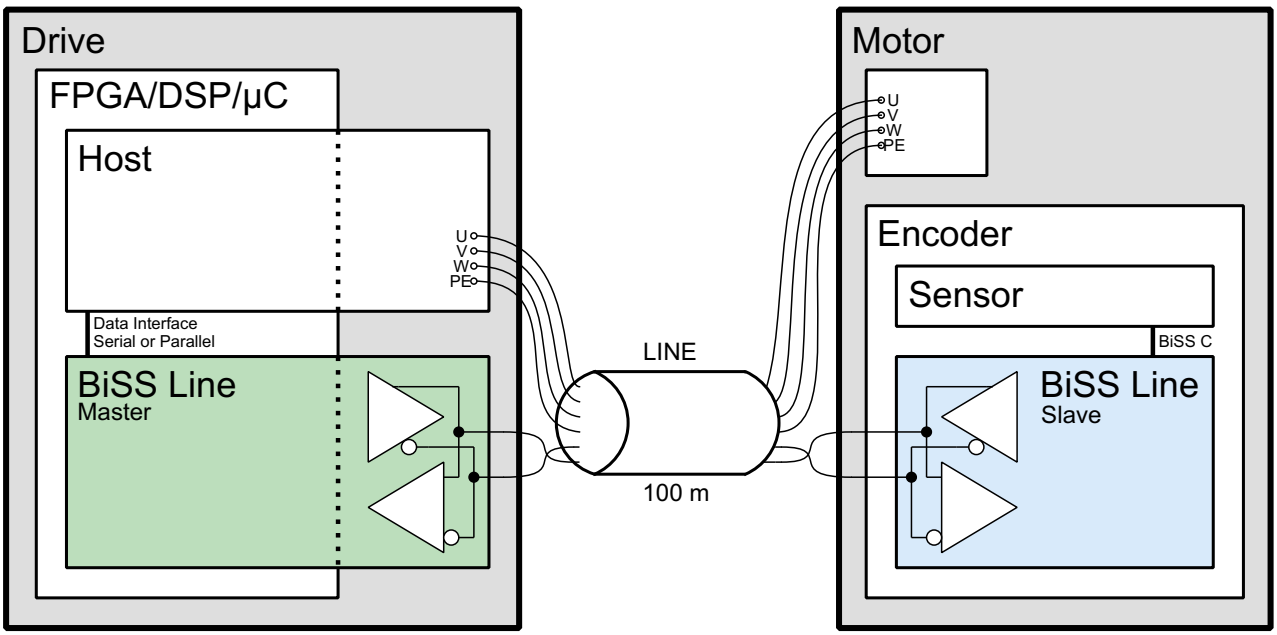

Figure 3: *BiSS Line* Motor Feedback System

Digital data and the power supply are transmitted over the same cable simultaneously. The connection between the BLM and the BLS can be realized using a 4-wire or a 2-wire configuration. The 4-wire configuration uses four dedicated lines for power supply and communication. The internal structure of the BLM and the BLS in 4-wire configuration is illustrated in Figure 4.

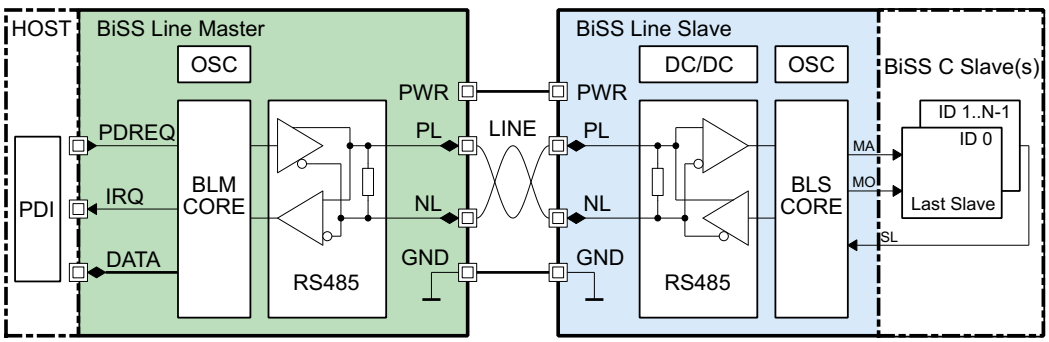

Figure 4: *BiSS Line* 4-wire Configuration

Typically the standard BiSS Line Master consists of the BLM core, an accurate oscillator (OSC) for system timing and a RS485 bus transceiver.The Host is connected to the BLM core by the Process Data Interface (PDI). A signal on Process Data Request (PDREQ) triggers the BLM to request new data from the BLS. At the end of

the *BiSS Line* communication the Host is notified about new data by an Interrupt Request (IRQ).
On the opposite side, the standard BiSS Line Slave usually consists of a RS485 bus transceiver, a DC/DC converter, an OSC for system timing and the BLS core.

Due to the protocol similarity, a BLS core typically provides a standard *BiSS C Master* interface to connect any *BiSS C* sensor that is already available on the market.

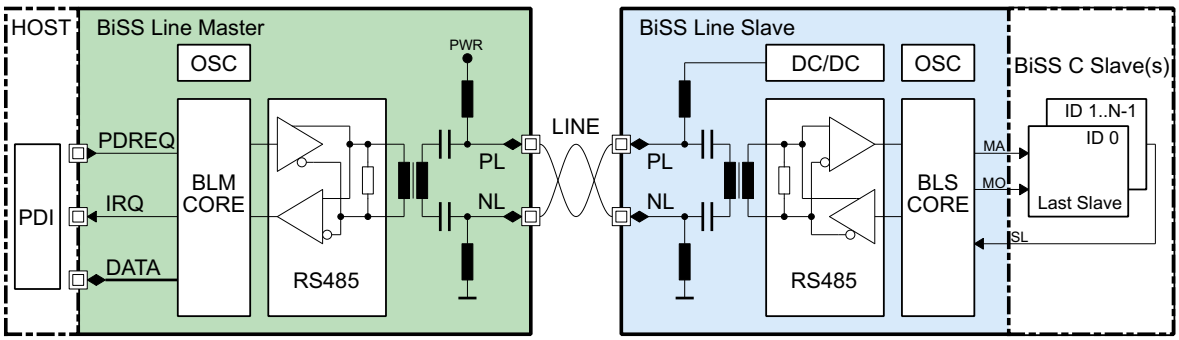

Figure 5: *BiSS Line* 2-wire configuration

In the 2-wire configuration, the power supply and the data signals are transmitted over the same twisted pair cable as shown in Figure 5. Compared to the 4-wire configuration, it requires additional passive elements for power supply (de)coupling but reduces the number of required wires.

BiSS LINE FRAME

The *BiSS Line* communication is realized with frames using a RS485 half-duplex baseband serial transmission. Figure 6 shows the structure of a general BiSS Line Frame. As defined before, the sections of the BiSS Line Frame that are driven by the BLM are colored in green. The blue sections indicate that the BLS is driving the line.

A BiSS Line Frame is initiated by the BLM and usually finished by a BLS. In between frames, the BLM sends IDLE symbols to prevent a DC charge of the line.

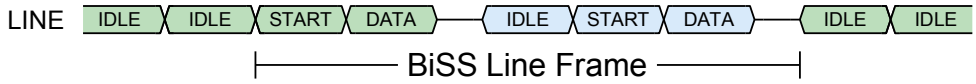

Figure 6: Structure of the BiSS Line Frame

At the beginning of a BiSS Line Frame, the BLM pauses the transmission of the IDLE symbol and sends one of several START symbols followed by DATA. When the BLM finishes the transmission, the BLS takes over the line. The response of the BLS starting with the IDLE symbol needs to be sent immediately to match the response timeout of the BLM. The BLS transmits the START symbol and DATA as soon as the requested data is available. At the end of the communication, the BLS releases the line and the BLM continues to send the IDLE symbol until the next BiSS Line Frame begins. Technical details about the BiSS Line Frame can be found in chapter CHARACTERISTICS on page 36.

IDLE Symbol

The IDLE symbol is an alternating code producing high frequency digital switching that is suitable for clock recovery. The maximum length of the idle time is not limited and the active transmitter can stop any IDLE symbol at a digital low to start the transmission. However, a long idle time helps the receiver to adapt to the clock phase of the current symbol on the line in disturbed environments. Table 1 shows the 8b10b coding of the IDLE symbol.

Symbol	Description	8b10b	Binary
IDLE	BLM and BLS Idle time for clock recovery	21.5	1010101010

Table 1: IDLE Symbol

START Symbols

The codes of the START symbol are intently chosen to provide a strong distinction between each other and the IDLE symbol. *BiSS Line* defines several START symbols for different requests of the BLM and responses of the BLS. The available START symbols are listed in Table 2.

START Symbol	Description	8b10b	BINARY
REQ	Sent only by BLM Request of Process Data Initialization of a PDF	26.7	0101101110
AUX	Sent only by BLM Request of Auxiliary Data Initialization of a ADF	no 8b10b	1101110000
RSP0	Sent only by BLS Response to a REQ from BLM, if CDS = 0 Response to an ADF	29.4	0100011101
RSP1	Sent only by BLS Response to a REQ from BLM, if CDS = 1	26.7	0101101110

Table 2: Listing of START Symbols

Each START symbol is used to determine the beginning of a new BiSS Line Frame transmission at the respective receiver. Since the beginning of the transmissions is a delicate phase of the communication, the detection of the START symbol must be robust and definite. Therefore, the codes of all valid symbols are

chosen to always produce a Hamming Distance (HD) of five and above while clocked into the receivers shift register bit by bit. Figure 7 shows the HD of the current content clocked into the shift register to the START symbol transmitted over the line. Because of the HD being a minimum of five constantly, the receiver is able to reconstruct the end of the START symbol exactly even if one or two bits are flipped during transmission. A HD below three always indicates a received START symbol explicitly.

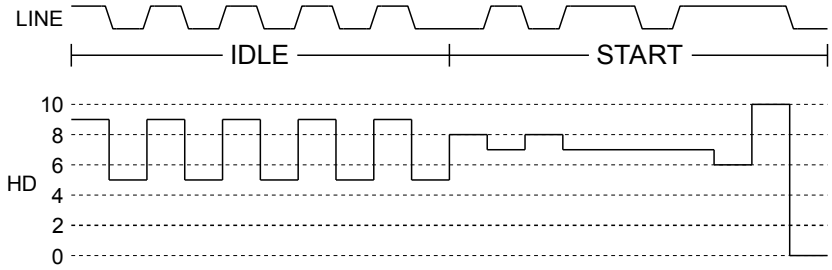

Figure 7: *BiSS Line* START Symbol Detection

Process Data Frame (PDF)
The START symbol REQ is used to initialize a PDF. The PDF is intended to exchange sensor and actuator data between the BLM and BLS on a regular basis. The PDF represents the main functionality of the *BiSS Line* protocol.

The Process Data is transmitted within the segment DATA of the PDF sent by the BLS and BLM respectively. *BiSS Line* matches the data structure of *BiSS C* exactly. As described in the *BiSS C* protocol, the Process Data is divided into multiple data channels optionally protected by Cyclic Redundancy Check (CRC). Usually, each data channel is transmitted completely in the segment DATA of every PDF.

The segment DATA of the BiSS Line Frame is 8b10b coded with the benefit of DC balance and sufficient transitions for clock recovery. The standard *BiSS Line* device supports FEC to further protect the Process Data against transmission errors. Since the applied FEC method does not only detect transmission errors, but is also able to correct corrupted data, the availability of the communication improves. To reduce the error rate in very disturbed environments, the Process Data can be transmitted repeatedly by request or pre-configuration. Detailed information about the PDF can be found in chapter PROCESS DATA FRAME (PDF) on page 8.

Control Data Frame
Similar to *BiSS C*, it is possible to access register data of the connected devices without stopping the Process Data transmission. Therefore, the PDF includes one data bit for register communication in each directions. The CDM bit is sent by the BLM and the CDS bit is sent by the BLS. Since the Control Data Frame is composed of the CDM and CDS bits over several PDF, the Control Data communication is intended for data that does not need to be refreshed on a high frequency basis. For more information about the Control Data Frame see chapter CONTROL DATA FRAME on page 14.

Auxiliary Data Frame (ADF)
The START symbol AUX initializes an ADF that can be used for general data transmissions without capturing new sensor data. There are several opcodes for predefined operations such as read and write of up to 8 data bytes within the address space of the BLS for example. As for the PDF, the opcode, the address and the data segments are 8b10b coded and protected by FEC.

At start-up, the ADF is intended to configure the *BiSS Line* communication parameters such as the number of data channels and the data length for example. Further opcodes may be implemented in the future. Further details about the ADF are described in Chapter AUXILIARY DATA FRAME (ADF) on page 18.

PROCESS DATA FRAME (PDF)

FRAME AND SYMBOLS

The capture of Process Data is usually triggered by the drive. Typically, it is not synchronized to the signals on the line. Figure 8 shows the sequence of a PDF requesting sensor data of a connected *BiSS C* slave.

The drive starts the sequence by asserting the process data request signal PDREQ at the PDI. Thereupon, the BLM initiates a new PDF by transmitting the START symbol REQ that induces the BLS to capture new sensor data by starting a *BiSS C* frame for communication with the connected sensor. As defined in the BiSS C Protocol Description, the sensor data is captured at the first rising edge on the *BiSS C* MA line.

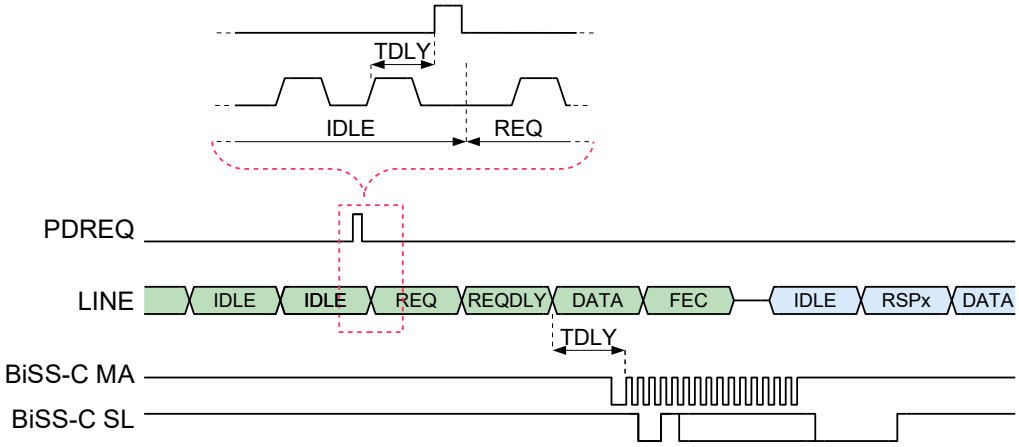

Figure 8: Process Data Frame

REQDLY
BiSS Line minimizes the jitter between the position request from the drive and the latch event at the sensor. Therefore, the delay time TDLY between the trigger event at PDREQ and the last rising edge of the IDLE symbol on LINE is measured. TDLY is coded into the symbol REQDLY which is transmitted to the BLS directly after the START symbol REQ. The BLS delays the start of the *BiSS C* frame by the received TDLY to ensure equidistant data acquisition. Technical information about TDLY can be found in Table 39 on page 36.

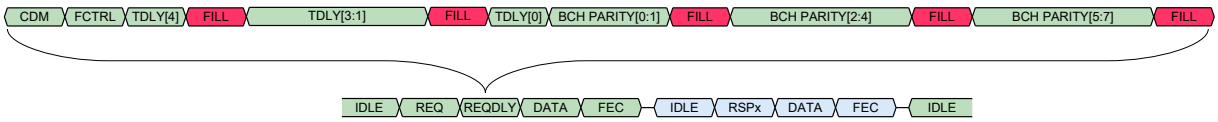

Figure 9: Detailed View of the *BiSS Line* Symbol REQDLY

Similar to the START symbols, REQDLY is coded in consideration of an adequate HD to provide appropriate error corrections. A Bose-Chaudhuri-Hocquenghem (BCH) code is used to transmit the CDM (1 bit), FCTRL (1 bit) and TDLY (5 bits). CDM is used for the Control Data communication and is described in chapter CONTROL DATA FRAME. The frame control bit FCTRL is reserved for future purposes regarding the PDF processing at the BLS.

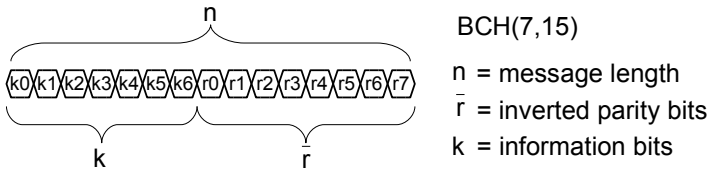

Figure 10: Illustration of a BCH(7,15) Protected Message

The BCH code is an FEC method that adds eight parity bits to REQDLY for redundancy. The polynomial used to generate the parity bits is $g(x) = x^8 + x^7 + x^6 + x^4 + x^0$. In total, the bit length of REQDLY is 15 bit and its FEC is able to correct any transmission errors of up to two bits. Figure 10 shows the structure of the resulting BCH (7,15) code.

Since the BCH (7,15) code is not DC balanced, it is necessary to insert fill bits (FILL) that prevent continuous series of zeros or ones in the symbol REQDLY. The BLM adds one FILL bit after every third BCH bit. Furthermore, the BCH PARITY bits are inverted before transmission to prevent information bits of exclusive zeros.

DATA
The PDF segment DATA holds the Process Data of *BiSS Line*. The Process Data is identically formatted as described in the BiSS C Protocol Description. 8 logical Process Data channels can be activated for transmission of actuator and sensor data to and from one BLS. Each Process Data channel can be configured for up to 64 data bits protected by a CRC of up to 16 bit. However, in systems with multiple slaves the maximum payload for *BiSS Line* of 64 byte cannot be exceeded.

Figure 11 shows a *BiSS Line* system connected to multiple *BiSS C* slaves of which one device is an actuator. Since *BiSS C* transfers sensor and actuator data by means of a large shift register, the unused actuator data for SENSORA and SENSORB must be clocked through the system resulting in an unnecessarily reduced frame rate. This is not the case for *BiSS Line*. The PDF only transmits the actuator data needed for the corresponding BLS.

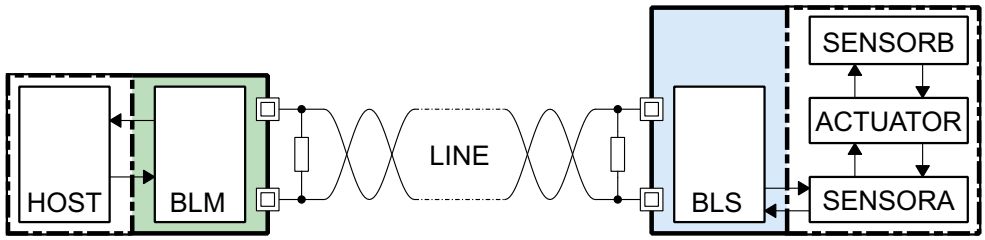

Figure 11: Multiple *BiSS C* Devices

The segment DATA is encoded using 8b10b*.
Figure 12 shows an example of 14 bit Process Data that are encoded into two 10-bit symbols sent by the BLM in the DATA section of the BiSS Line Frame.

First, the data to be transmitted is divided into portions of 8 bits. Incomplete bytes are left aligned and padded with zeros. Second, the 8b10b encoding adds two bits for each byte to complete the Line Code.
The segment DATA sent by the BLM transmits the actuator data and the segment DATA sent by the BLS transmits the sensor data using the same 8b10b coding scheme.

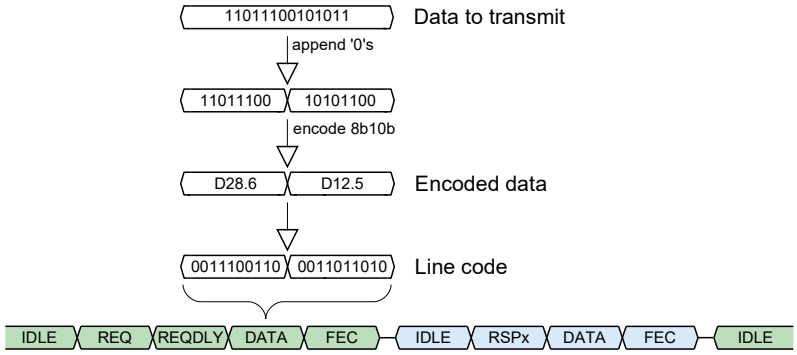

Figure 12: Example of 8b10b Encoding for DATA

FEC
To additionally protect the control, actuator and sensor data against burst errors, *BiSS Line* uses a Forward

* A detailed description of the 8b10b coding can be found in patent US 4486739 A "Byte oriented DC balanced (0,4) 8B/10B partitioned block transmission code".

Error Correction (FEC) based on a Reed Solomon (RS) code which is a special subclass of BCH codes. Since the FEC is 8b10b coded as well, the RS code is calculated for DATA before the 8b10b coding. In comparison to the BCH code, the RS code applies to bytes instead of single bits. Hence, it is possible to correct complete *BiSS Line* symbols only. *BiSS Line* implements the RS(247, 255) code, which uses the generator polynomial $g(x) = x^8 + x^4 + x^3 + x^2 + x^0$. For calculations of the 8 parity bytes, DATA is extended to a fixed length of 247 bytes as shown in Figure 13.

Unused data bytes are filled with zeros, which are not transmitted over the line. The data bytes must be considered right aligned to execute the parity calculation correctly. In total, the RS(247, 255) code is able to correct four symbols distorted during transmission. More information about the RS(247, 255) can be found in "Error Control Coding: Fundamentals and Applications, SHU LIN/Danie J. Costello,Jr.".

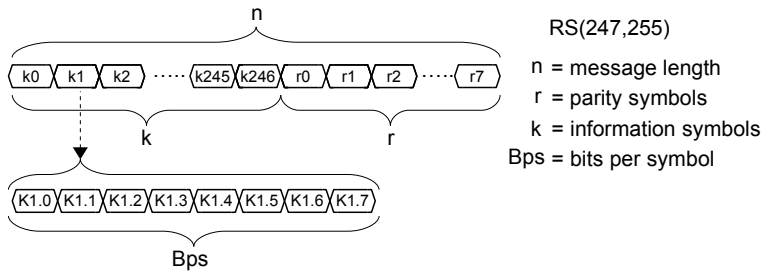

Figure 13: Illustration of a RS(247,255) Protected Message

RSPx

In contrast to the BLM, the BLS sends the START symbol RSPx which includes the CDS bit. In case of a connected *BiSS C* sensor the recently received CDS bit is passed on by the Internal BiSS C Master. However, the BLS is able to independently generate CDS bits for Control Data communications.
The Internal BiSS C Master controls the communication to the connected *BiSS C* device at the BLS. As described in chapter CONTROL DATA FRAME, the CDS bit is used for Control Data communications.

In case of a PDF, the CDS bit information is coded into the START symbol of the BLS response. There are two possible START symbols RSP0 and RSP1 representing CDS = 0 and CDS = 1 respectively. More information about the START symbol coding can be found in Table 2 on page 6.

STATE DIAGRAM

Figure 14 shows the sequence of the internal states of a standard BLM during Process Data communication as illustrated in Figure 8 on page 8. As before, the bubbles colored in green represent the states in which the BLM is sending data and the blue bubbles represent the states in which the BLS is responding accordingly. Figure 15 on the other hand shows the sequence of the internal states of a standard BLS during the same communication procedure. Thus, the BLM and BLS cycle the same states when executing a PDF.
The following section describes the different states of the BLM and BLS as presented in Figures 14 and 15 in detail. The description of the states refer to the BLM and BLS as labeled.

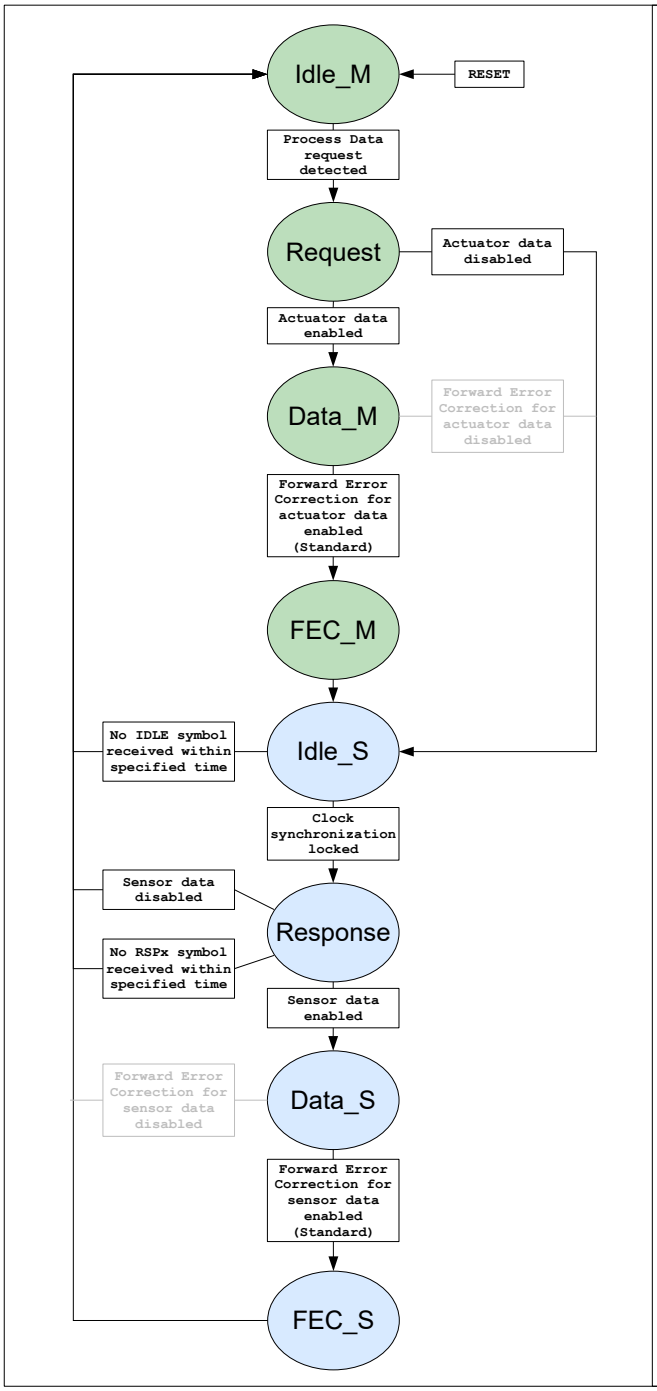

Figure 14: State Diagram of the BLM During PDF Execution

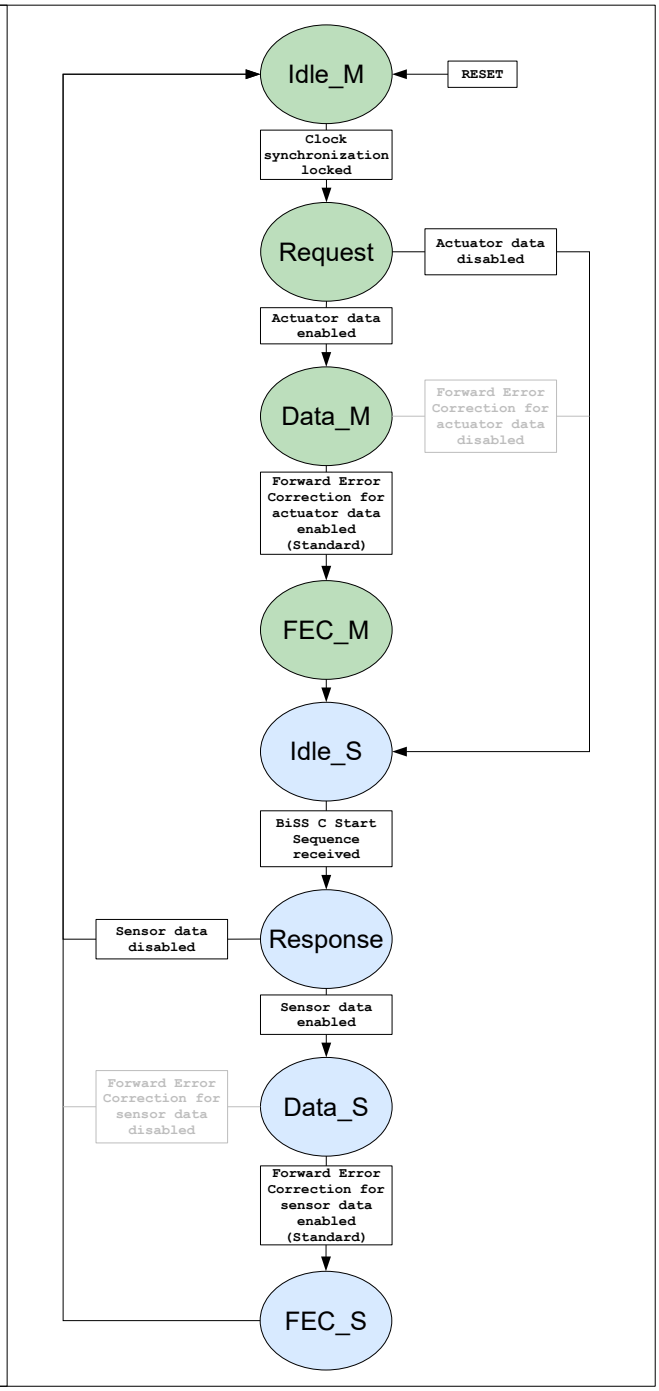

Figure 15: State Diagram of the BLS During PDF Execution

State: Idle_M
After system reset, the BLM and BLS start in the state Idle_M.

Master:
At the beginning, the BLM sends the symbol IDLE to allow the BLS to synchronize to the *BiSS Line* clock signal. The IDLE symbol is chosen following the 8b10b coding D21.5. Thus, the transmission line continuously switches from digital high to digital low. That helps to keep the transmission line DC balanced and produces sufficient level changes for high precision clock recovery.
When receiving a process data request signal (PDREQ) signal from the drive, the BLM stores the TDLY and enters the state Request.

Slave:
The BLS enters the state Request, if the clock synchronization is locked to the incoming transmission clock of the BLM.

State: Request
After receiving the trigger, the synchronized data transmission can be started by the BLM.

Master:
First, the BLM transmits the start symbol REQ to the BLS. REQ starts the BiSS Line Frame and prepares the BLS to receive data. The symbol REQ is chosen following 8b10b coding D26.7.
Second, the BLM transmits the symbol REQDLY to the BLS which contains the TDLY of the current data request signal. The CDM bit and the FCTRL bit coming from the drive are coded within the symbol REQDLY as well. To generate the symbol REQDLY the BLM uses BCH(7,15) for error correction. If the BLM is enabled to transmit actuator data to the BLS, the BLM continues to control the communication and enters the state Data_M. Otherwise, the BLM stops the transmission, enters the state Idle_S and waits for the BLS to drive the line.

Slave:
After receiving the start symbol REQ completely, the synchronized BLS is able to correctly sample the information from the BiSS Line Frame. The REQDLY data is extracted, analyzed and corrected, if the parity bits indicate any errors during transmission. The reconstructed FCTRL is evaluated and CDM and TDLY are passed on to the Internal BiSS C Master. The Internal BiSS C Master controls the communication to the connected *BiSS C* slaves and is fully compatible to the standard BiSS C Protocol Description. Therefore, *BiSS Line* can easily be used to extend a standard *BiSS C* communication system.
The Internal BiSS C Master uses TDLY to delay the start of the *BiSS C* frame. Thus, the jitter of the sensor data acquisition timing can be compensated, which is an important characteristic for proper control design at the connected drive.
Similar to the BLM, the BLS enters the state Data_M only, if it expects actuator data coming over the line. If not, the BLS takes over the communication control and enters the state Idle_S.

State: Data_M
The state Data_M is used to transfer actuator data from the BLM to the BLS.

Master:
The BLM divides the applied data and generates the corresponding 8b10b codes to be sent over the line. The BLM continues to state FEC_M after all data has been transmitted.

Slave:
The BLS receives the 8b10b coded data packets and decodes the original actuator information to be passed on to the *BiSS C* Slave. The BLS enters state FEC_M before starting transmission of actuator data at the *BiSS C* interface.

State: FEC_M
With FEC erroneous transmissions in heavily disturbed environments can be corrected and data availability is increased.

Master:
First, the BLM divides the raw actuator data into portions of eight bits. Incomplete bytes are filled with zeros as described in the state Data_M. REQDLY and the data byte symbols are extended to 247 bytes by adding zero bytes in front of them. Finally, the RS parity symbols are calculated. The zero bytes are for calculation purposes only.
Second, the 8 parity symbols are encoded using 8b10b before sent over the line.
After sending the FEC information, the BLM finishes data transmission and is ready to receive the response of the BLS. Therefore, the BLM releases the transmission line, continues to the state Idle_S and hands the communication control over to the BLS.

Slave:
The BLS receives the FEC symbols and decodes the 8b10b code to recover the original parity bytes. Afterwards, the data bytes and parity bytes are used to correct possible transmission errors and the actuator data is forwarded to the Internal BiSS C Master. The Internal BiSS C Master generates the start bit and continues

the *BiSS C* frame as shown in the *BiSS C* protocol description.
For FEC calculations, the received DATA has to be extended by zero bytes as described for the BLM. The BLS continues to the state Idle_S and prepares to transmit its data to the BLM.

State: Idle_S
The state Idle_S is equal to the state Idle_M except of the fact that the control of the communication is handled by the BLS. In this phase of the communication procedure, the BLS sends its response back to the BLM.

Slave:
Since the direction of data transmission changes, the BLM needs to be synchronized to the clock of the BLS. Therefore, the BLS also sends the symbol IDLE as described in the state Idle_M for the BLM.
If the Internal BiSS C Master receives the complete start sequence including the CDS bit from the *BiSS C* slave, the BLS is triggered to start the *BiSS Line* response. The BLS stops sending the IDLE symbol and continues to the state Response.
Since the start sequence of the *BiSS C* protocol is initiated considering any line delay and processing times, *BiSS Line* automatically delays the response depending on the connected sensor.

Master:
If the clock of the BLM for receiving locks to the clock of the BLS for transmitting, the BLM continues to the state Response. However, if the BLS does not respond in time, the BLM aborts the PDF, returns to the state Idle_M and notifies the Host about the erroneous communication.

State: Response
The state Response is similar to the state Request. The START symbol synchronizes the data transmission between BLS and BLM.

Slave:
In contrast to the BLM, the BLS sends the START symbol RSPx which includes the recently received CDS bit from the Internal BiSS C Master. There are two possible START symbols RSP0 and RSP1 representing CDS = 0 and CDS = 1 respectively. More information about the START symbol coding can be found in Table 2 on page 6.
If the BLS is enabled to transmit sensor data over the line, the next state of the communication procedure is Data_S. Otherwise, the BLS returns to the state Idle_M and the BiSS Line Frame is completed.

Master:
The BLM waits for the START symbol RSPx to be received completely.
If the BLM is configured to receive sensor data from the line the BLM continues to the state Data_S. If there is no sensor data transmission, the BLM returns to Idle_M and waits for the next PDREQ to start another BiSS Line Frame.

State: Data_S
In the state Data_S, the BLS sends the sensor data of the Internal BiSS C Master to the BLM. The process is similar to the transmission of the actuator data as described for the state Data_M.

State: FEC_S
In the state FEC_S, the BLS calculates and sends the FEC parity bytes of the sensor data to the BLM including the bit CDS.
The process is similar to the transmission of the FEC parity bytes as described in state FEC_M.

CONTROL DATA FRAME

As described in chapter PROCESS DATA FRAME (PDF) on page 8, *BiSS C* uses the Control Data Frame for register access. Therefore, each *BiSS C* Frame includes the CDM bit, to send information from the master to the slaves, and the CDS bit for the opposite direction. The Control Data Frame is composed of the single bit transmissions included in multiple *BiSS C* Frames. More information about the Control Data Frame of the *BiSS C* protocol can be found in BiSS C Protocol Description.

Since *BiSS Line* provides the functionality of *BiSS C*, the BiSS Line Frame can also be used to transmit Control Data. Similar to the *BiSS C* protocol, the Control Data communication of *BiSS Line* is handled during execution of the PDF. The CDM bit is included in the REQDLY symbol send by the BLM and the CDS bit is coded into the RSPx symbol from the BLS. For more information about the PDF see chapter PROCESS DATA FRAME (PDF) on page 8.

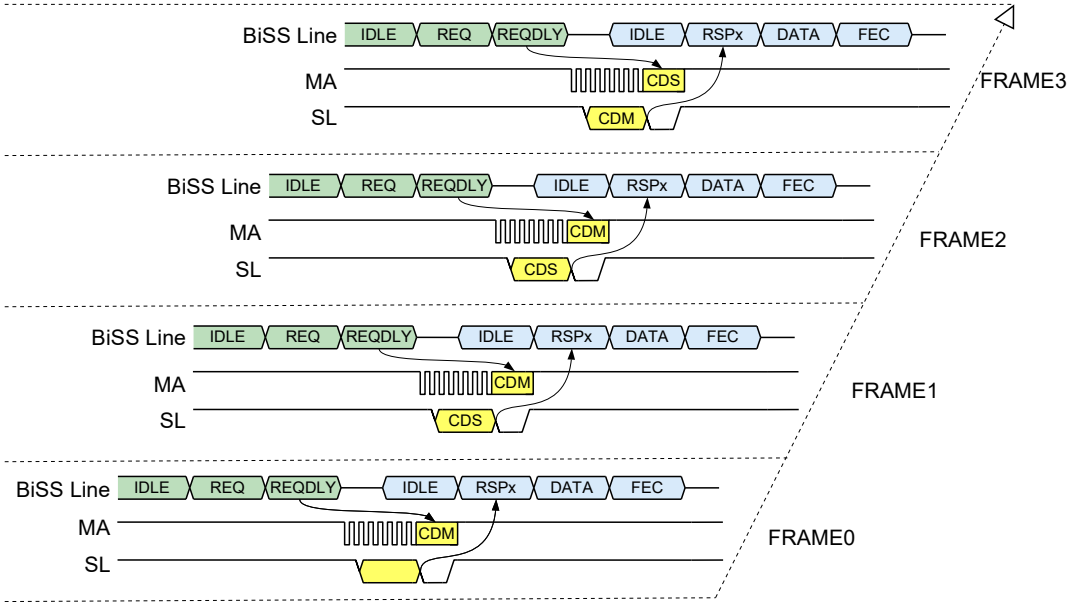

Figure 16: *BiSS Line* Control Data Communication for *BiSS C* Devices

Figure 16 illustrates the Control Data Frame of *BiSS Line*. Each of the four PDF that are sent in sequence, transmit one bit of the Control Data Frame in both directions. The *BiSS Line* symbol REQDLY includes the CDM bit of the corresponding *BiSS* C Frame FRAME0 ... FRAME3 to be sent to the connected BLS. The BLS passes the received CDM bit on to the Internal BiSS C Master which transmits it during the *BiSS C* timeout of the *BiSS C* Frame as explained in the *BiSS C* protocol description. The *BiSS C* Slave merges the received CDM bits to build the Control Data Frame.

The data response of the *BiSS C* Slave is coded into the symbol RSPx as soon as the start sequence including the CDS bit is received by the Internal BiSS C Master. CDS = 0 produces a *BiSS Line* response of RSP0 and CDS = 1 produces a *BiSS Line* response of RSP1. As for the CDM bit, the BLM needs to merge the CDS bits to form the Control Data Frame. The Control Data Frame is started by the Host.

The *BiSS C* Master sends its CDM bit at the end of the *BiSS C* Frame after receiving the CDS bit. Hence, the CDM bit can be altered in reaction to the CDS bit received within one *BiSS C* frame. The only case an immediate reaction to the CDS bit is necessary is, if the start bit of the Control Frame is delayed by the *BiSS C* slave during register access.
In the *BiSS Line* protocol on the other hand, the symbol REQDLY is sent before the symbol RSPx arrives at the BLM as shown in Figure 16. That means, it is not possible to alter the CDM bit in reaction to the CDS bit within the same frame cycle.
In case of a read register access, the BLS extends the start bit to address the incompatibility without significant impact on the communications.
For writing, a delayed access is not supported. However, since the write access for the first byte is usually not delayed, the limitation only affects sequential write accesses of multiple bytes.

As explained in the *BiSS C* protocol description, *BiSS C* automatically assigns a unique slave ID to each of the connected *BiSS C* slaves at the beginning of every Control Frame. Since the *BiSS C* Frame is passed from one *BiSS C* slave to another, the *BiSS C* slave picks its slave ID locally by occupying the CDS bit of the current *BiSS C* Frame. See the *BiSS C* protocol description for further details.

For configuration of the *BiSS Line* components, the Control Data Frame can be used to address the BLS as well. Since each occupied CDS bit from the *BiSS* chain is processed for generating the appropriate *BiSS Line* response, the BLS is capable of detecting the end of the *BiSS C* slave ID assignment. The BLS occupies the first unused *BiSS C* slave ID by setting the CDS bit of the particular ID slot (IDL). Hence, the BLS will always be assigned to the highest slave ID.

After the assignment, the BLS responds to any Control Data Frame addressed to that specific slave ID. However, the configuration of the BLS can also be done using the ADF.

BiSS LINE BUS

Additionally to the point-to-point configuration as discussed in chapter BiSS LINE SYSTEM on page 4, it is possible to connect multiple *BiSS Line* devices in a bus topology. The *BiSS Line* bus can be used to control the system of an industrial robot arm, for example. Only one cable is necessary to power and communicate to several position sensors built into the joints of the robot. However, *BiSS Line* enables connections to multiple BLS that are equipped with multiple sensors and actuators providing every feature as discussed for the point-to-point configuration.

Figure 17 shows an example of a *BiSS Line* bus system that contains three BLS connected to multiple sensors and actuators.

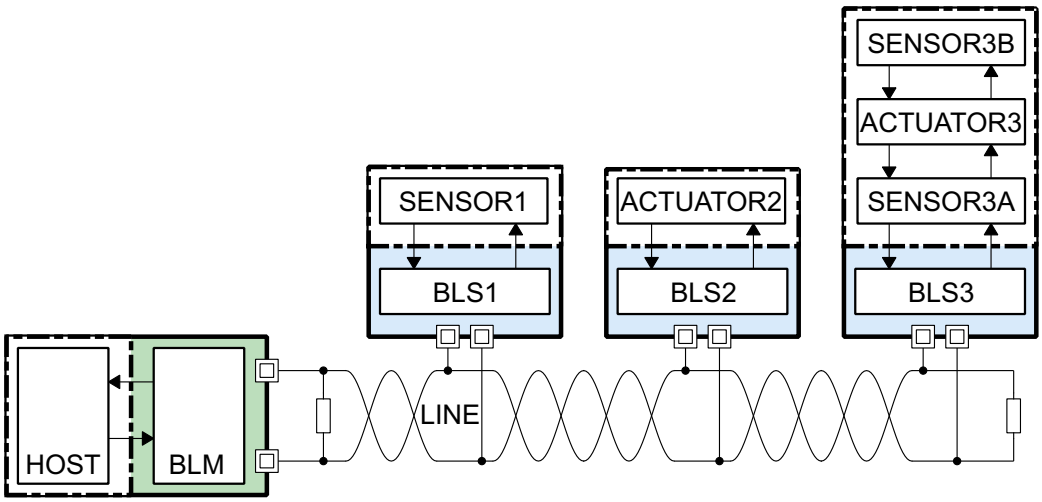

Figure 17: *BiSS Line* Bus with Multiple *BiSS C* Slaves

As described in chapter PROCESS DATA FRAME (PDF) the PDF consists of a request sent by the BLM and a response sent by the BLS. If more than one BLS is connected to the BLM, there must be some sort of organization to handle the response of each device. Since the drive in the robot arm for example processes the position of every joint in every control cycle, *BiSS Line* needs to transmit the sensor data of every BLS within one PDF.

The data transmission for PDF is organized using specific time slots that are assigned to every BLS connected to the system. Hence, the BLS respond to a request by the BLM in succession. Figure 18 shows the PDF for the *BiSS Line* bus topology from Figure 17.

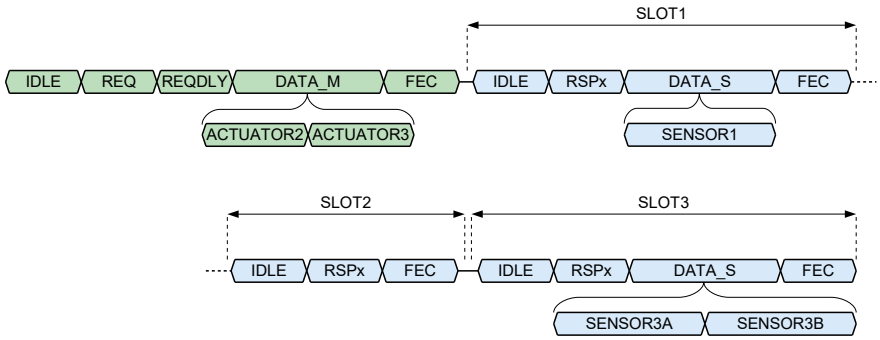

Figure 18: Process Data Frame for *BiSS Line* Bus Topology

Similar to the PDF of the point-to-point configuration, the request of the BLM initiates the communication. Since all BLS are connected to the BLM in parallel, each device processes the request as described before.
The actuator data for ACTUATOR2 and ACTUATOR3 are transmitted in the section of the frame that is driven by the BLM. Every BLS that receives actuator data

needs to be configured to the position of the corresponding data within DATA_M.

Before starting any PDF, each BLS that is transmitting sensor data needs to be configured to a specific time slot. If a BLS processes actuator data only, a time slot does not necessarily need to be assigned to it. However, for communications via Control Data Frame, the BLS must be assigned to a time slot in order to respond accordingly.
As it can be seen in Figure 18, BLS1, BLS2 and BLS3 from the example network above are assigned to SLOT1, SLOT2 and SLOT3 respectively. For the calculation of the point in time when to start the response, each BLS needs to be configured to the data lengths of the other time slots as well. Hence, the timing of the responses is individually determined by every BLS.

If time slots overlap due to configuration errors, the corruption of the transmission is detected by the data protection mechanism as described before.

Naturally, the maximum *BiSS Line* frame rate is affected by the number of slots that are activated in the system. If the BLM is supposed to start another PDF before every time slot is processed, the Host receives an error signal.

For configuration of a *BiSS Line* bus topology, each BLS connected to the BLM needs to be identified and explicitly configured. The ADF provides special operations for configuration of the BLS. Detailed information about the ADF can be found in chapter AUXILIARY DATA FRAME (ADF).

AUXILIARY DATA FRAME (ADF)

The Auxiliary Data Frame (ADF) is used to communicate with a specific BLS and execute predefined operations. The ADF is required to establish a *BiSS Line* bus topology by configuring each BLS according to the system setup. Furthermore, it is suitable for transmissions of a larger amount of data from a selected BLS to BLM and vice versa. Since the data rate is much higher than for a Control Data Frame, the ADF is intended for addressing additional data storages such as external EEPROMs and communication to a microcontroller. codes the ADF enables functions for data processing far beyond simple read and write.

An ADF can individually be started by the BLM in the configuration phase of the system's operation but also while already transmitting PDF regularly. In the latter case, one PDF is replaced by the ADF, if activated. Hence, the functionality of the ADF is available without stopping the overall *BiSS Line* cycle. However, it is important to understand, that an ADF does not trigger the BLS to capture new sensor data.

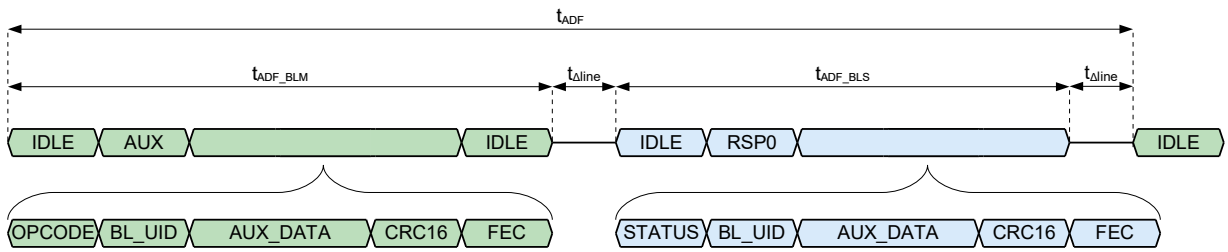

Figure 19: Auxiliary Data Frame

In comparison to the PDF, the ADF includes a unique identification number that addresses a specific BLS. Consequentially, in general only one *BiSS Line* device of the system responses to the start symbol AUX. Therefore, the *BiSS Line* Unique Identifier (BL_UID), a 48-bit unique number, is assigned to every BLS during production. Especially for bus systems with several *BiSS Line* devices, this enables the BLM to communicate to a particular BLS exclusively since the time slot mechanism is temporarily deactivated. However, it is also possible to address every BLS in the system by sending broadcast operations via ADF.

The function of an ADF is defined by the operation code transmitted by the BLM. Since the structure of the ADF stays the same for each opcode, the frame length is fixed. In Figure 19 the individual segments of an ADF are shown.

Segment	Length	Data	Description
IDLE	40 bits	-	Synchronization and Frame Processing at BLS
AUX	10 bits	-	START symbol "1101110000" for the Request of the BLM
OPCODE	20 bits	2 bytes	The Code Selects the Operation to be Executed
BL_UID	60 bits	6 bytes	Unique Identifier of a *BiSS Line* Slave BL_UID(47:32) = Manufacturer ID BL_UID(31:0) = Serial number
AUX_DATA	80 bits	8 bytes	Data for the Selected Operation
CRC16	20 bits	2 bytes	16-bit CRC Checksum for OPCODE/STATUS, BL_UID and AUX_DATA.
FEC	80 bits	8 bytes	FEC Protection for the Entire ADF including CRC16
RSP0	10 bits	-	START symbol "0100011101" for the Response of a BLS
STATUS	20 bits	2 bytes	Status of the BLS sent During Response

Table 3: Segments of the Auxiliary Data Frame

As explained previously, the BLS needs to be synchronized to the data clock of the BLM before starting any *BiSS Line* frame. For the ADF at least four IDLE symbols are necessary to ensure an accurate data transmission on the line.
The start symbol to initialize a new ADF is AUX. OP-

CODE, BL_UID and AUX_DATA are used to control the frame operation and for transmission of necessary data. Finally, the frame is protected by CRC16 and FEC against disturbances.
For ADF it is necessary to add another four IDLE symbols at the end of the BiSS Line Frame segments driven by the BLM. The additional time is needed by the BLS to decode the received data considering any FEC corrections and to evaluate if the ADF has to be responded to.
After synchronization of the BLM, the response of the BLS is initiated by the start symbol RSP0. The rest of the response is identical to the BLM section of the BiSS Line Frame except for STATUS that replaces the OPCODE. Table 3 overviews the different segments of the ADF.

IDLE, AUX and RSP0
After the four mandatory IDLE symbols are sent, the synchronization phase can be interrupted anytime as described for the PDF. The start symbol AUX initiates a new ADF which always consists of the same frame segments listed in Table 3. The invariable time duration of the frame section that is driven by BLM is $t_{REQUEST\_AUX}$ as shown in Figure 19.

For the response of the BLS the IDLE time is the same as for the BLM except that there is no further IDLE segment at the end of the transmission. Apart from the second IDLE, the frame section that is driven by the BLS is structured similarly to the request resulting in the overall time duration for one complete ADF of t_{FRAME_AUX}. The actual physical characteristics of *BiSS Line* are listed in Table 39 on page 36.

The start symbol of the response is RSP0 and matches the start symbol of the PDF when returning a CDS bit of 0. Since the Control Data communication is deactivated, the BLM does not need to distinguish between multiple start symbols. However, a Control Data communication in progress is only paused for processing the ADF. The next PDF continues to execute the current Control Data Frame without any effect to the data transfer.

OPCODE and STATUS
The data length of OPCODE before 8b10b coding is two byte. The first byte holds the OPCODE_HEAD, which determines the operation, the ADF is supposed to trigger. There are several operations available as listed in Table 7 on page 22. Further operations for data processing may be added in the future.

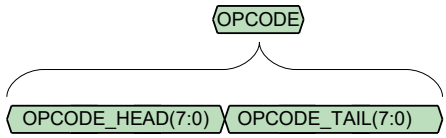

Figure 20: OPCODE of Auxiliary Data Frame

The second byte of OPCODE is the OPCODE_TAIL. It is used for additional information to control the execution of the operation. The operations AUX_READx and AUX_WRITEx for instance use the OPCODE_TAIL to determine the data address within the recipient to read from or write to. However, OPCODES such as AUX_NOP do not need any further configuration and transmit a constant OPCODE_TAIL of zero. Figure 20 shows the composition of OPCODE as described above.

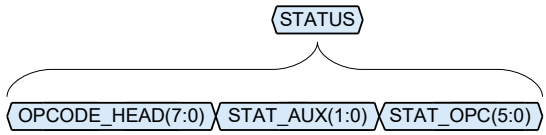

Figure 21: STATUS of Auxiliary Data Frame

In response to an OPCODE sent by the BLM, BLS returns a two byte STATUS back to the BLM as it can be seen in Figure 21. The first byte of STATUS repeats the OPCODE_HEAD that was previously sent by BLM. The second byte reports the status of the BLS processing the opcode.

It consists of a general 2 bit status STAT_AUX for the current operation and a specific 6 bit status STAT_OPC for the selected OPCODE. STAT_AUX is always the same for all operations providing general information about the operation in progress. Table 4 shows the coding of STAT_AUX in detail.
In contrast to STAT_AUX, STAT_OPC transmits specific information about the status of the current OPCODE. Hence, the structure of STAT_OPC depends on the OPCODE that is currently in progress.

STAT_AUX(1:0)	Status	Description
00	Idle	No Operation in Progress
01	Busy	Operation in Progress
10	Acknowledge	Operation Successfully Completed
11	Error	Operation Failed

Table 4: General Operation Status

Single frame operations can be executed within one BiSS Line Frame and are immediately acknowledged by the BLS. Multi frame operations need several BiSS Line Frames and are indicated by STAT_AUX = Busy and OPCODE_HEAD = STAT_OPC = 0. If the BLS is busy executing a multi frame operation, the OPCODE AUX_NOP must be sent by the BLM until STAT_AUX changes to 'Acknowledge' or 'Error'. All multi frame operations use OPCODES $\geq$ 0x10.

Table 5 summarizes the content of OPCODE and STATUS as described above. Detailed information about the different operations and their configuration and status can be found in chapter AUXILIARY DATA FRAME OPERATIONS.

Segment	Content	Length	Description
OPCODE(15:8)	OPCODE_HEAD	8 bit	Determines the Operation for the BLS to be executed
OPCODE(7:0)	OPCODE_TAIL	8 bit	Configuration of the Operation Different for every OPCODE_HEAD
STATUS(15:8)	OPCODE_HEAD	8 bit	Repetition of last OPCODE_HEAD
STATUS(7:6)	STAT_AUX	2 bit	General Status of the Operation
STATUS(5:0)	STAT_OPC	6 bit	Specific Status of the Operation Different for every OPCODE_HEAD

Table 5: OPCODE and STATUS

BL_UID
In general, the communication of an ADF is point to point. The BLM addresses one BLS that exclusively responses accordingly. As shown in Table 3 on page 18 the BL_UID, which is used for addressing a specific *BiSS Line* device, consists of the Manufacturer Identification (MFR_ID) and the Serial Number (DEV_SN). The MFR_ID, a 16-bit number which must be requested from the *BiSS* Association e.V.*, identifies the manufacturer of the *BiSS Line* device. The DEV_SN is a 32-bit value exclusively assigned to each BLS. Both identification parameters have already been introduced for *BiSS C* as can be read in the BiSS C Protocol Description.

For verification, the BLS sends its own BL_UID back to the BLM. However, since the BLM does not automatically check the BL_UID, the Host needs to verify that the response on the line is sent by the requested device.

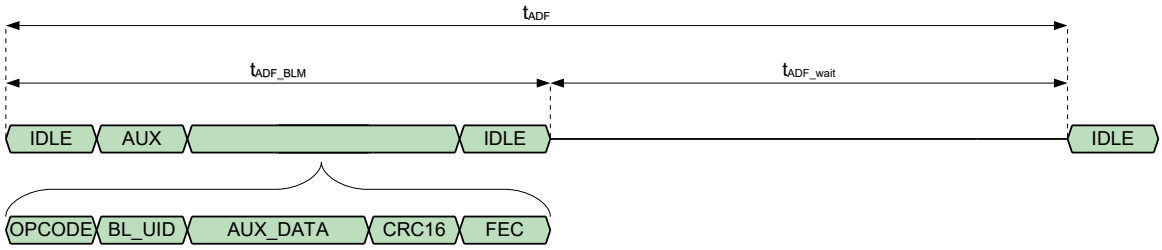

Figure 22: Auxiliary Data Frame for Broadcast Operations

In contrast to its main purpose to address a specific *BiSS Line* device, the ADF provides the functionality to send broadcast messages to every BLS connected to the system. Therefore, *BiSS Line* defines a broadcast address for one way operations. If BL_UID equals zero, the ADF is processed by every BLS but no response is returned to the BLM. Hence, a broadcast operation is not acknowledged by STATUS as described for point-to-point communications. As shown in Figure 22 the time duration of an ADF for broadcast operations stays the same.

AUX_DATA
The Host configures the parameters within the BLM section and triggers a new ADF. After completion of the communication with the addressed slave, the Host checks the BL_UID_S and collects the desired data within the BLS section.

To transmit data to and from the BLS the data section AUX_DATA is used. An ADF provides up to 8 byte for data transmissions in both directions. Although not every operation occupies all data bytes, each ADF transmits the entire AUX_DATA. Unused bytes in AUX_DATA_M are ignored and unused bytes in AUX_DATA_S are set to zero by the BLS. The AUX_DATA bytes used for a particular operation are specified in chapter AUXILIARY DATA FRAME OPERATIONS on page 22.

General Data Field								
Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BiSS Line Master								
OPCODE								
00	OPCODE_HEAD(7:0)							
01	OPCODE_TAIL(7:0)							
BL_UID_M								
02	BL_UID(47:40) = MFR_ID (15:8)							
03	BL_UID(39:32) = MFR_ID (7:0)							
04	BL_UID(31:24) = DEV_SN (31:24)							
05	BL_UID(23:16) = DEV_SN (23:16)							
06	BL_UID(15:8) = DEV_SN (15:8)							
07	BL_UID(7:0) = DEV_SN (7:0)							
AUX_DATA_M								
08	AUX_DATA(63:56)							
09	AUX_DATA(55:48)							
10	AUX_DATA(47:40)							
11	AUX_DATA(39:32)							
12	AUX_DATA(31:24)							
13	AUX_DATA(23:16)							
14	AUX_DATA(15:8)							
15	AUX_DATA(7:0)							
BiSS Line Slave								
STATUS								
16	OPCODE_HEAD(7:0)							
17	STAT_AUX(1:0)		STAT_OPC(5:0)					
BL_UID_S								
18-23	BL_UID(47:0)							
AUX_DATA_S								
24-31	AUX_DATA(63:0)							

Table 6: General Data Field & Transmission Sequence of an Auxiliary Data Frame

CRC16 and FEC
There are two mechanisms to protect the data transmission of an ADF. First, a 16-bit CRC is calculated for OPCODE, BL_UID and AUX_DATA with the polynomial 0x190d9 and a start value of zero. The checksum is inverted and added to the frame as CRC16.

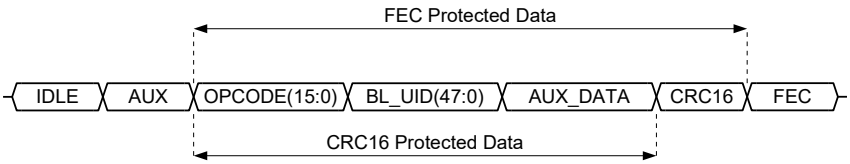

Figure 23: CRC and FEC Protection of an ADF

Afterwards, the *BiSS Line* device calculates the FEC as already described for the PDF. Figure 23 shows the data ranges the two protection checksums are calculated for.

Before transmission, all bytes of the ADF are 8b10b coded as described in chapter PROCESS DATA FRAME (PDF) on page 8.

AUXILIARY DATA FRAME OPERATIONS

As described in chapter AUXILIARY DATA FRAME (ADF) on page 18, the ADF provides several operations for data transmission. According to the specific function, the OPCODE, the AUX_DATA and the STATUS of each operation is formatted differently. Based on the definition of the general data field in Table 5, this chapter describes the available operations, their function and directions for usage. Table 7 overviews the operations available and defines their OPCODE_HEAD.

OVERVIEW

Operation	OPCODE_HEAD	Description	Reference
Single Frame Operations			
-	0x00	Reserved	-
AUX_PING	0x01	Induce BLS to Respond to Bitmasked BL_UID	Page 24
AUX_CFGPDF	0x02	Configure the Process Data Frame for Multiple Slaves	Page 26
AUX_GETCFGPDF	0x03	Gets the Process Data Frame Configuration stored in the BLS	Page 29
-	0x04 ... 0x0F	Reserved	-
Multi Frame Operations			
AUX_NOP	0x10	No Operation	Page 31
-	0x11 ... 0x16	Reserved (Unused)	-
AUX_SETAOFFS	0x17	Extend Slave Address for R/W Access	Page 35
AUX_READ1	0x18	One Byte Read Operation	Page 32
AUX_READ2	0x19	Two Byte Read Operation	Page 32
AUX_READ4	0x1A	Four Byte Read Operation	Page 32
AUX_READ8	0x1B	Eight Byte Read Operation	Page 32
AUX_WRITE1	0x1C	One Byte Write Operation	Page 34
AUX_WRITE2	0x1D	Two Byte Write Operation	Page 34
AUX_WRITE4	0x1E	Four Byte Write Operation	Page 34
AUX_WRITE8	0x1F	Eight Byte Write Operation	Page 34
-	0x20 ... 0xFF	Reserved	-

Table 7: OPCODE_HEAD

There are two groups of operations. The operations of the group Single Frame Operations are processed within one BiSS Line Frame. Hence, the immediate response of the BLS corresponds to the request of the BLM and can be evaluated in the same frame cycle.

On the other hand, operations which cannot be processed within one BiSS Line Frame belong to the group Multi Frame Operations. Because the operation is not yet completely processed, the immediate response of the BLS does not show STAT_OPC of the operation sent by the BLM, and STAT_AUX equals 'Idle' or 'Busy'. If STAT_AUX equals 'Idle', the BLS is occupied with another task and has not yet started to process the new operation. In contrast, 'Busy' indicates that the BLS is processing the new operation which is not yet finished.

However, for Multi Frame Operations it is necessary to send another ADF to trigger a response of the BLS without starting a new operation. This is what the operation AUX_NOP is intended for. AUX_NOP can be used to check the status of the latest operation. If the BLS receives AUX_NOP, it returns the section of the general data field associated with the slave without starting any other process. Hence, AUX_NOP transmits the current STATUS from the BLS to the BLM. Detailed information about AUX_NOP can be found on page 31.

Figure 24 shows an example of an operation from the Single and Multi Frame Operations group. In the latter case, AUX_NOP is used to transmit STATUS back to the BLM.

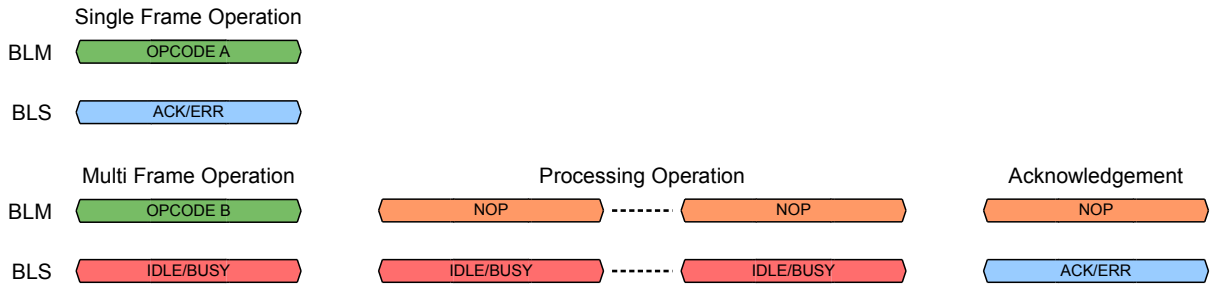

Figure 24: Procedure of Single and Multiple Frame Operations

Each rectangle represents a segment of the general data field (transmission sequence) of one *BiSS Line* frame cycle as described in chapter AUXILIARY DATA FRAME (ADF) on page 18. The upper half shows the BLM section and the lower half shows the BLS section of the general data field.

At the top, for the Single Frame Operation, the green region is the response to the OPCODE A in the blue region of the same *BiSS Line* frame cycle. Hence, after the transmission of one BiSS Line Frame, the operation is finished completely.

At the bottom, the red region indicates that the operation from the blue region above is not completed. Hence, OPCODE B is an operation from the group Multi Frame Operations. For the next ADF, AUX_NOP in the yellow region is used to request the status of the operation triggered by OPCODE B. AUX_NOP is repeated as long as the BLS section of the general data frame acknowledges the operation as shown within the green region of the last *BiSS Line* frame cycle in Figure 24.

If another OCPODE_HEAD is sent while the status still reports 'Busy', STAT_AUX changes back to 'Idle' and waits for the BLS to start processing the new operation. However, an operation of the group Single Frame Operations always responses immediately, no matter what STAT_AUX was before.

OPERATION AUX_PING

For setting a new *BiSS Line* system up, the AUX_PING operation can be used to automatically identify any BLS connected to the drive. Instead of using the individual addresses of each BLS in the system explicitly, AUX_PING enables the BLM to request multiple devices at once. Therefore, a bitmask is used to group the BL_UID of the *BiSS Line* devices AUX_PING is intended for. Every BLS of the system processes AUX_PING and analyzes the bitmask and its own BL_UID to decide, if a response is demanded. Hence, it is possible that multiple devices drive the line generating an illegal response. The BLM detects the unusable frame by data protection mechanisms as described before and starts to refine the bitmask to identify individual BL_UIDs. If every BL_UID is identified, other OPCODES can be used to catch information about the individual devices.
However, AUX_PING can also be used to test, if individual *BiSS Line* devices still respond to the BLM. Table 7 shows the general data field and transmission sequence of operation AUX_PING.

AUX_PING								
Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BiSS Line Master								
OPCODE								
00	0	0	0	0	0	0	0	1
01	0	IDENT	BITMASK(5:0)					
BL_UID_M								
02 ... 07	BL_UID(47:0)							
AUX_DATA_M								
08 ... 15	unused							
BiSS Line Slave								
STATUS								
16	0	0	0	0	0	0	0	1
17	1	0	0	0	0	0	0	0
BL_UID_S								
18 ... 23	BL_UID(47:0)							
AUX_DATA_S								
24 ... 31	unused							

Table 8: General Data Field & Transmission Sequence of Operation AUX_PING

OPCODE consists of OPCODE_HEAD = 0x01 to select AUX_PING, and IDENT as well as BITMASK to configure the operation. BITMASK is a 6-bit value used to select a subset of the BL_UID for addressing multiple *BiSS Line* devices. Since the bit width of BL_UID is 48, BITMASK can be set in the range from 0 to 48.

BITMASK	Byte 01; Bit 5:0	BLM
0	All BL_UIDs	
0 ... 47	Group of BL_UIDs	
48	Specific BL_UID	
>48	undefined	

Table 9: Bitmask for AUX_PING

BITMASK = 48 means that every bit of the BL_UID at the slave must match the BL_UID_M sent within the AUX_PING frame in order to induce the BLS to respond. Hence, this is the configuration to test an individual BLS explicitly. If BITMASK = 0 then every *BiSS Line* slave of the system responds to the operation. It is the job of the drive to delimit the bitmask accordingly.
Figure 25 shows an example of BL_UID matching with BITMASK(5:0) = 17. The blue box exactly enframes 17 bits that are considered for the selection of the BLS to respond. All bits outside of the box are ignored. Hence, any BLS with the BL_UID = xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxx0 10100100 01110010 responds to the AUX_PING operation of the example.

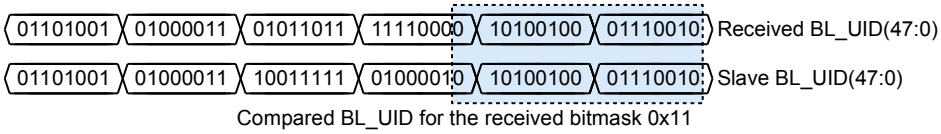

Figure 25: Example with BITMASK(5:0) = 17

If a single BLS and its BL_UID is identified, it can be disabled by IDENT to prevent it from responding during further searches. If IDENT is set, the BLS addressed with BL_UID_M and BITMASK = 48 is internally flagged as identified. For further AUX_PING operations with IDENT = 1 and BITMASK < 48, the flagged BLS are not responding. However, a search with IDENT = 0 suppresses this mechanism.
To clear an identification flag, AUX_PING with IDENT = 0 must be addressed to a specific (BITMASK = 48) BLS. A Broadcast AUX_PING operation with BITMASK = 0 and IDENT = 0 can be used to reset the internal identification flags of all connected BLS at once.

IDENT	Byte 01;	Bit 6 BLM
Code	Condition	Description
0	BITMASK = 0, BL_UID = 0	Reset Identification Flag of all connected BLS
0	BITMASK =48	Reset Identification Flag
0	BITMASK≤48	Search Without Considering Identification Flags
1	BITMASK =48	Set Identification Flag
1	BITMASK≤48	Search With Considering Identification Flags

Table 10: Identification flag for AUX_PING

AUX_PING belongs to the group of Single Frame Operations and responds with the STATUS as shown in Table 7 within the same ADF.

OPERATION AUX_CFGPDF

AUX_CFGPDF can be used to configure the PDF for a single BLS in a bus system.
As described in chapter BiSS LINE BUS on page 16, to correctly respond within the assigned time slot scheme of a PDF, each BLS need to be configured for the data length of every *BiSS Line* device. Since the configuration exceeds 8 bytes, the operation is divided into two pages. Although AUX_CFGPDF belongs to the group of Single Frame Operations, two ADF are necessary to completely configure one BLS. Table 10 shows the general data field and transmission sequence of AUX_CFGPDF.

AUX_CFGPDF								
Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BiSS Line Master								
OPCODE								
00	0	0	0	0	0	0	1	0
01	0	0	0	0	0	0	PAGE(1:0)	
BL_UID_M								
02 ... 07	BL_UID(47:0)							
AUX_DATA_M (PAGE = 0)								
08	FEC_AD	DLEN_AD(6:0)						
09	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0
11	Reserved for future use (Must be 0x00)							
12	EN_CFGP	0	IDLE_REST(5:0)					
13	0	0	IDLE_FIRST(5:0)					
14	DIS_SLID	AD_OFFS						
15	IDL_OFFS(3:0)				SLOT_NO(3:0)			
AUX_DATA_M (PAGE = 1)								
08	FEC_SD8	DLEN_SD8(6:0)						
09	FEC_SD7	DLEN_SD7(6:0)						
10	FEC_SD6	DLEN_SD6(6:0)						
11	FEC_SD5	DLEN_SD5(6:0)						
12	FEC_SD4	DLEN_SD4(6:0)						
13	FEC_SD3	DLEN_SD3(6:0)						
14	FEC_SD2	DLEN_SD2(6:0)						
15	FEC_SD1	DLEN_SD1(6:0)						
BiSS Line Slave								
STATUS								
16	0	0	0	0	0	0	1	0
17	1	0	0	0	0	0	0	0
BL_UID_S								
18 ... 23	BL_UID(47:0)							
AUX_DATA_S								
24 ... 31	unused							

Table 11: General Data Field & Transmission Sequence of Operation AUX_CFGPDF

OPCODE_HEAD is 0x02 and OPCODE_TAIL includes PAGE(1:0) only. In 10, the field AUX_DATA_M is presented twice, once for every page. The parameter PAGE(1:0) is used to select which page to use.

PAGE	Byte 01; Bit 1:0	BLM
0	Use AUX_DATA_M (PAGE = 0)	
1	Use AUX_DATA_M (PAGE = 1)	
2, 3	Reserved	

Table 12: Page Selection for AUX_CFGPDF

For configuration of the actuator data, DLEN_AD, FEC_AD and AD_OFFS are available.

DLEN_AD	Byte 08 (PAGE = 0); Bit 6:0	BLM
0	No Actuator Data Used	
1 ... 64	Length of Actuator Data in Byte	
>64	not defined	

Table 13: Data Length of Actuator Data for AUX_CFGPDF

DLEN_AD is used to determine the total number of bytes all BLS expect for actuator processing. FEC_AD activates the forward error correction for the Actuator Data as described in chapter PROCESS DATA FRAME (PDF) on page 8.

FEC_AD	Byte 08 (PAGE = 0); Bit 7	BLM
0	Deactivate FEC for Actuator Data	
1	Activate FEC for Actuator Data	
Notes	In standard *BiSS* Line systems, FEC is enabled for all BLS.	

Table 14: Forward Error Correction of Actuator Data of AUX_CFGPDF

The segment DATA of a PDF includes the actuator data of every slave and each BLS needs to extract its bytes correctly. Therefore, the position of the bytes for any BLS needs to be configured accordingly. AD_OFFS is used to program that offset for the actuator data of each slave. For example, if the third byte of DATA is the first actuator data byte of a BLS, then AD_OFFS needs to be set to 2.

AD_OFFS	Byte 14 (PAGE = 0); Bit 6:0	BLM
0	First Bytes Hold Actuator Data for Slave	
1 ... 63	Byte Offset of Actuator Data	
>64	not defined	

Table 15: Actuator Data Position Offset for AUX_CFGPDF

Since there is one BLM only, the configuration of the actuator data needs to be done just for the current BLS. For the sensor data on the other hand, each *BiSS Line* device needs to consider the data length of the other sensors for calculation of the time slot as well. Hence, the data length for senor data, DLEN_SDx must be configured for each of the 8 possible devices.

DIS_SLID	Byte 14 (PAGE = 0); Bit 7	BLM
0	Activate IDL assignment for BLS	
1	Deactivate IDL assignment for BLS	

Table 16: Disable Control Communication IDL assignment for *BLS*

With DIS_SLID = 1 BLS occupies no IDL address in the Control Communication. Usually BLS takes the last free IDL in a *BiSS C* bus. This also must be considered for IDL_OFFS programming.

DLEN_SD8	Byte 08 (PAGE = 1); Bit 6:0	BLM
DLEN_SD7	Byte 09 (PAGE = 1); Bit 6:0	BLM
DLEN_SD6	Byte 10 (PAGE = 1); Bit 6:0	BLM
DLEN_SD5	Byte 11 (PAGE = 1); Bit 6:0	BLM
DLEN_SD4	Byte 12 (PAGE = 1); Bit 6:0	BLM
DLEN_SD3	Byte 13 (PAGE = 1); Bit 6:0	BLM
DLEN_SD2	Byte 14 (PAGE = 1); Bit 6:0	BLM
DLEN_SD1	Byte 15 (PAGE = 1); Bit 6:0	BLM
0	Time Slot not Used	
1 ... 64	Length of Sensor Data in Byte by Time Slot	
65 ... 126	Not Defined	
127	No Sensor Data, Control Data Communication Activated	

Table 17: Data Length of Sensor Data for AUX_CFGPDF

DLEN_SDx = 0 deactivates the time slot. The time slots used must be configured in direct sequence, it is not possible to skip any time slot. The length of the sensor data for each BLS can be configured in the range of 1 to 64. However, as described in chapter PROCESS DATA FRAME (PDF) *BiSS Line* only supports DATA of up to 64 byte in total. If no sensor data transmission is used for a BLS but the drive must communicate to it via Control Data Frame, DLEN_SDx = 127 occupies a time slot with an empty segment DATA.

FEC_SD8	Byte 08 (PAGE = 1);	Bit 7	BLM
FEC_SD7	Byte 09 (PAGE = 1);	Bit 7	BLM
FEC_SD6	Byte 10 (PAGE = 1);	Bit 7	BLM
FEC_SD5	Byte 11 (PAGE = 1);	Bit 7	BLM
FEC_SD4	Byte 12 (PAGE = 1);	Bit 7	BLM
FEC_SD3	Byte 13 (PAGE = 1);	Bit 7	BLM
FEC_SD2	Byte 14 (PAGE = 1);	Bit 7	BLM
FEC_SD1	Byte 15 (PAGE = 1);	Bit 7	BLM
0	Deactivate FEC for Sensor Data by Time Slot		
1	Activate FEC for Sensor Data by Time Slot		
Notes	In standard *BiSS* Line systems, FEC is enabled for all BLS.		

Table 18: Forward Error Correction for Sensor Data of AUX_CFGPDF

Similar to the actuator data, FEC_SDx enables and disables the forward error correction for each BLS. Since each device holds the configuration of every slave, SLOT_NO is used to determine which configuration applies to the particular BLS. SLOT_NO = 8 entirely deactivates the device for the PDF.

SLOT_NO	Byte 15 (PAGE = 0);	Bit 3:0	BLM
0 ... 7	Time slot Assignment for BLS		
8	Process Data Frame Deactivated for BLS		
>8	not defined		

Table 19: Time Slot Number for AUX_CFGPDF

More configurations for the calculation of the appropriate BLS response are IDLE_FIRST and IDLE_REST.

IDLE_FIRST	Byte 13 (PAGE = 0);	Bit 5:0	BLM
0 ... 63	Duration of Idle Time in Symbols		

Table 20: Idle Time for BLS of First Time Slot of AUX_CFGPDF

IDLE_FIRST is used to define the idle time of the BLS within the first time slot. It is necessary to consider the maximum processing time of the system to ensure proper data generation.

IDLE_REST	Byte 12 (PAGE = 0);	Bit 5:0	BLM
0 ... 63	Duration of Idle Times in Symbols		

Table 21: Idle Time for BLS of Other Time Slots of AUX_CFGPDF

IDLE_REST is used for the other *BiSS Line* slaves in the system. If IDLE_FIRST is sufficiently chosen, IDLE_REST must only assure adequate synchronization for flawless *BiSS Line* data transmissions.

EN_CFGP	Byte 12 (PAGE = 0);	Bit 7	BLM
0	BLS will not use the AUX_CFGPDF configuration.		
1	BLS will use the AUX_CFGPDF configuration.		

Table 22: Enables AUX_CFGPDF parameters

With EN_CFGP = 0 the bls will not use the provided parameters by AUX_CFGPDF for the time slot assignment and the IDLE durations.

IDL_OFFS	Byte 15 (PAGE = 0);	Bit 7:4	BLM
0 ... 15	Offset of *BiSS C* ID for *BiSS Line* System		

Table 23: Offset of Automatic Identification Assignment for *BiSS C* of AUX_CFGPDF

The BLS is effected by the automatic assignment of *BiSS C* slave identification for Control Data Frame as described in chapter PROCESS DATA FRAME (PDF). In a system with multiple slaves, the numeration of *BiSS C* devices for one BLS must consider the connected devices of the other BLS. Therefore, IDL_OFFS can be used.
For example, three *BiSS C* slaves connected to BLS1 are assigned to the IDL = 0, 1, and 2. Another three *BiSS C* slaves connected to BLS2 are assigned to the same identifications as for the *BiSS C* slaves connected to BLS1. Using IDL_OFFS = 3 for BLS2 causes the Internal BiSS C Master to occupy the first three IDL resulting in IDL = 3, 4, 5 for its *BiSS C* sensors. Hence, every *BiSS C* device at every BLS can be explicitly addressed by the BLM without adaption of the identification scheme of *BiSS C*.
However, the connected *BiSS C* sensors must support the MO line of *BiSS C* in order to provide automatic device identification.

OPERATION AUX_GETCFGPDF

AUX_GETCFGPDF can be used to read the PDF configuration stored in the BLS. Since the PDF configuration data has a size of 32 bytes, the operation is divided into four pages. Although AUX_GETCFGPDF belongs to the group of Single Frame Operations, four ADF are necessary to completely get the configuration of one BLS.

AUX_GETCFGPDF								
Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BiSS Line Master								
OPCODE								
00	0	0	0	0	0	0	1	1
01	0	0	0	0	0	0	PAGE(1:0)	
BL_UID_M								
02 ... 07	BL_UID(47:0)							
08 ... 15	unused							
BiSS Line Slave								
STATUS								
16	0	0	0	0	0	0	1	1
17	1	0	0	0	0	0	0	0
BL_UID_S								
18 ... 23	BL_UID(47:0)							
AUX_DATA_S (PAGE = 0)								
24	ENACT1	ENSENS1	PDLEN1(5:0)					
25	0	CRCLEN1(6:0)						
26	CRCSTART1(7:0)							
27	CRCSTART1(15:8)							
28 ... 0x31	PDF Configuration for Data Channel 2							
AUX_DATA_S (PAGE = 1)								
24 ... 31	PDF Configuration for Data Channel 3 ... 4							
AUX_DATA_S (PAGE = 2)								
24 ... 31	PDF Configuration for Data Channel 5 ... 6							
AUX_DATA_S (PAGE = 3)								
24 ... 31	PDF Configuration for Data Channel 7 ... 8							

Table 24: General Data Field & Transmission Sequence of Operation AUX_GETCFGPDF

OPCODE_HEAD is 0x03 and OPCODE_TAIL includes PAGE(1:0) only. In Table 23, the field AUX_DATA_S is presented four times, once for every page. The parameter PAGE(1:0) is used to select which page to use.

PAGE	Byte 01; Bit 1:0	BLM
0	Use AUX_DATA_S (PAGE = 0)	
1	Use AUX_DATA_S (PAGE = 1)	
2	Use AUX_DATA_S (PAGE = 2)	
3	Use AUX_DATA_S (PAGE = 3)	

Table 25: Page Selection for AUX_GETCFGPDF

ENACT1	Byte 24 (PAGE=0);	Bit 7	BLS
ENACT2	Byte 28 (PAGE=0);	Bit 7	BLS
ENACT3	Byte 24 (PAGE=1);	Bit 7	BLS
ENACT4	Byte 28 (PAGE=1);	Bit 7	BLS
ENACT5	Byte 24 (PAGE=2);	Bit 7	BLS
ENACT6	Byte 28 (PAGE=2);	Bit 7	BLS
ENACT7	Byte 24 (PAGE=3);	Bit 7	BLS
ENACT8	Byte 28 (PAGE=3);	Bit 7	BLS

0	Actuator Data not sent
1	Actuator Data sent

Table 26: Enable Actuator Data for Channel x

ENSENS1	Byte 24 (PAGE=0);	Bit 6	BLS
ENSENS2	Byte 28 (PAGE=0);	Bit 6	BLS
ENSENS3	Byte 24 (PAGE=1);	Bit 6	BLS
ENSENS4	Byte 28 (PAGE=1);	Bit 6	BLS
ENSENS5	Byte 24 (PAGE=2);	Bit 6	BLS
ENSENS6	Byte 28 (PAGE=2);	Bit 6	BLS
ENSENS7	Byte 24 (PAGE=3);	Bit 6	BLS
ENSENS8	Byte 28 (PAGE=3);	Bit 6	BLS

0	Sensor Data not sent
1	Sensor Data sent

Table 27: Enable Sensor Data for Channel x

PDLEN1	Byte 24 (PAGE=0);	Bit 4:0	BLS
PDLEN2	Byte 28 (PAGE=0);	Bit 4:0	BLS
PDLEN3	Byte 24 (PAGE=1);	Bit 4:0	BLS
PDLEN4	Byte 28 (PAGE=1);	Bit 4:0	BLS
PDLEN5	Byte 24 (PAGE=2);	Bit 4:0	BLS
PDLEN6	Byte 28 (PAGE=2);	Bit 4:0	BLS
PDLEN7	Byte 24 (PAGE=3);	Bit 4:0	BLS
PDLEN8	Byte 28 (PAGE=3);	Bit 4:0	BLS

0	1
1 ... 62	Code + 1
63	64

Table 28: Process Data Length of Channel x

CRCLEN1	Byte 25 (PAGE=0);	Bit 6:0	BLS
CRCLEN2	Byte 29 (PAGE=0);	Bit 6:0	BLS
CRCLEN3	Byte 25 (PAGE=1);	Bit 6:0	BLS
CRCLEN4	Byte 29 (PAGE=1);	Bit 6:0	BLS
CRCLEN5	Byte 25 (PAGE=2);	Bit 6:0	BLS
CRCLEN6	Byte 29 (PAGE=2);	Bit 6:0	BLS
CRCLEN7	Byte 25 (PAGE=3);	Bit 6:0	BLS
CRCLEN8	Byte 29 (PAGE=3);	Bit 6:0	BLS

0x00	Single-Cycle Data not sent, CRC not used
0x03	0b1011 = 0xB
0x04	0b1.0011 = 0x13
0x05	0b10.0101 = 0x25
0x06	0b100.0011 = 0x43
0x07	0b1000.1001 = 0x89
0x08	0b1.0010.1111 = 0x12F
0x10	0b1.1001.0000.1101.1001 = 0x190D9

Table 29: CRC Length with Predefined Polynomials

CRCSTART1(7:0)	Byte 26 (PAGE=0);	Bit 7:0	BLS
CRCSTART1(15:8)	Byte 26 (PAGE=0);	Bit 7:0	BLS
CRCSTART2(7:0)	Byte 30 (PAGE=0);	Bit 7:0	BLS
CRCSTART2(15:8)	Byte 30 (PAGE=0);	Bit 7:0	BLS
CRCSTART3(7:0)	Byte 26 (PAGE=1);	Bit 7:0	BLS
CRCSTART3(15:8)	Byte 26 (PAGE=1);	Bit 7:0	BLS
CRCSTART4(7:0)	Byte 30 (PAGE=1);	Bit 7:0	BLS
CRCSTART4(15:8)	Byte 30 (PAGE=1);	Bit 7:0	BLS
CRCSTART5(7:0)	Byte 26 (PAGE=2);	Bit 7:0	BLS
CRCSTART5(15:8)	Byte 26 (PAGE=2);	Bit 7:0	BLS
CRCSTART6(7:0)	Byte 30 (PAGE=2);	Bit 7:0	BLS
CRCSTART6(15:8)	Byte 30 (PAGE=2);	Bit 7:0	BLS
CRCSTART7(7:0)	Byte 26 (PAGE=3);	Bit 7:0	BLS
CRCSTART7(15:8)	Byte 26 (PAGE=3);	Bit 7:0	BLS
CRCSTART8(7:0)	Byte 30 (PAGE=3);	Bit 7:0	BLS
CRCSTART8(15:8)	Byte 30 (PAGE=3);	Bit 7:0	BLS

0 ... 65535	Code

Table 30: CRC Start Value

OPERATION AUX_NOP

For all multiple frame operations, AUX_NOP can be used to check their status as described before. Table 30 shows the general data field and transmission sequence of AUX_NOP. The section of the BLM is static. OPCODE_HEAD is 0x10 without any further configurations.

The section of the BLS depends on the status of the operation in progress. If STAT_AUX equals 'Idle' or 'Busy', the section of the BLS is static too as shown in Table 30. But if STAT_AUX equals 'Acknowledge' or 'Error', the section of the BLS changes to the response of the operation that has just finished. The section of the BLS then corresponds to the OPCODE as described in this chapter.

AUX_NOP								
Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BiSS Line Master								
OPCODE								
00	0	0	0	1	0	0	0	0
01	0	0	0	0	0	0	0	0
BL_UID_M								
02 ... 07	BL_UID(47:0)							
AUX_DATA_M								
08 ... 15	unused							
BiSS Line Slave (STAT_AUX = 'Idle' or 'Busy')								
STATUS								
16	0	0	0	0	0	0	0	0
17	STAT_AUX		0	0	0	0	0	0
BL_UID_S								
18 ... 23	BL_UID(47:0)							
AUX_DATA_S								
24 ... 31	unused							
BiSS Line Slave (STAT_AUX = 'Acknowledge' or 'Error')								
STATUS								
16	OPCODE_HEAD(7:0) of Last Operation							
17	STAT_AUX		STAT_OPC(5:0) of Last Operation					
BL_UID_S								
18 ... 23	BL_UID(47:0)							
AUX_DATA_S								
24 ... 31	Depends on Last Operation							

Table 31: General Data Field & Transmission Sequence of Operation AUX_NOP

AUX_NOP does not execute any actual operation or produce its own status. OPCODE_HEAD and STAT_OPC of STATUS as well as AUX_DATA_S are invalid before STAT_AUX acknowledges the operation in progress.

OPERATION AUX_READx

For fast transmissions of up to 8 data bytes from the BLS to the BLM, the operations AUX_READx is intended. It can be used to transfer larger amounts of data from sources such as external EEPROMs and for communications to a connected microcontroller. AUX_READx is a Multi Frame Operation.

The configuration of the ADF to be conducted is the number of bytes and the address of the first byte to read. In the case of multiple byte accesses, the address is automatically increased by the BLS. Table 31 shows the general data field of the AUX_READx operations.

AUX_READx								
Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BiSS Line Master								
OPCODE								
00	0	0	0	1	1	0	NBR(1:0)	
01	0	ADDRESS(6:0)						
BL_UID_M								
02 ... 07	BL_UID(47:0)							
AUX_DATA_M								
08 ... 15	unused							
BiSS Line Slave (STAT_AUX = 'Acknowledge' or 'Error')								
STATUS								
16	0	0	0	1	1	0	NBR(1:0)	
17	STAT_AUX		0	0	NBP(3:0)			
BL_UID_S								
18 ... 23	BL_UID(47:0)							
AUX_DATA_S								
24	RDATA_ADDR7(7:0)							
25	RDATA_ADDR6(7:0)							
26	RDATA_ADDR5(7:0)							
27	RDATA_ADDR4(7:0)							
28	RDATA_ADDR3(7:0)							
29	RDATA_ADDR2(7:0)							
30	RDATA_ADDR1(7:0)							
31	RDATA_ADDR0(7:0)							

Table 32: General Data Field & Transmission Sequence of Operation AUX_READx

As a part of the OPCODE_HEAD, NBR(1:0) determines the number of requested bytes to be read from the BLS.

NBR	Byte 00;	Bit 1:0	BLM
NBR	Byte 16;	Bit 1:0	BLS
00	Read 1 Byte from BLS		
01	Read 2 Bytes from BLS		
10	Read 4 Bytes from BLS		
11	Read 8 Bytes from BLS		

Table 33: Number of Requested Bytes for AUX_READx

The start address of the BLS address space is defined by ADDRESS(6:0). If multiple bytes are selected by NBR(1:0), the bytes are processed in the BLS by incrementing the address accordingly. As explained on page 35, the limited address space can be extended by the operation AUX_SETAOFFS.

ADDRESS	Byte 01;	Bit 6:0	BLM
0 ... 127	BLS Start Address for Read Operation		

Table 34: Address of the First Byte for AUX_READx

Since AUX_READx is a Multi Frame Operation, the Host must wait until STAT_AUX changes from 'Busy' to 'Acknowledge' before collecting the requested read data bytes. Therefore, the OPCODE AUX_NOP is used as explained on page 31. However, if STAT_AUX = 'Error' for read operations of multiple bytes, NBP(3:0) indicates how many of the requested data bytes could successfully be processed.

NBP	Byte 17;	Bit 3:0	BLS
0 ... 8	n Bytes Read from the BLS		
>8	not possible		

Table 35: Number of Processed Bytes for AUX_READx

If the operation is successfully completed, the requested data bytes can be collected from the segment AUX_DATA_S. Unused data bytes are set to zero. For example, if NBR(1:0) = 01 and NBP(3:0) = 2, the requested data bytes are written to RDATA_ADDR0 and RDATA_ADDR1.

OPERATION AUX_WRITEx

The operations AUX_WRITEx work similar as described for AUX_READx on page 32. In contrast to the latter, AUX_WRITEx transmits data from the BLM to the BLS. Hence, the only difference is that the Host writes the data bytes to be transferred into the segment AUX_DATA_M.
For example, to write 1 byte at address 0x20 of the BLS address space, the Host must set ADDRESS(6:0) = 0x20 and the data must be copied to WDATA_ADDR0 for transmission. WDATA_ADDR1 ... WDATA_ADDR7 are not used and are ignored by the BLS. Table 35 shows the general data field for AUX_WRITEx.

AUX_WRITEx								
Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BiSS Line Master								
OPCODE								
00	0	0	0	1	1	1	NBR(1:0)	
01	0	ADDRESS(6:0)						
BL_UID_M								
02 ... 07	BL_UID(47:0)							
AUX_DATA_M								
08	WDATA_ADDR7(7:0)							
09	WDATA_ADDR6(7:0)							
10	WDATA_ADDR5(7:0)							
11	WDATA_ADDR4(7:0)							
12	WDATA_ADDR3(7:0)							
13	WDATA_ADDR2(7:0)							
14	WDATA_ADDR1(7:0)							
15	WDATA_ADDR0(7:0)							
BiSS Line Slave (STAT_AUX = 'Acknowledge' or 'Error')								
STATUS								
16	0	0	0	1	1	1	NBR(1:0)	
17	STAT_AUX		0	0	NBP(3:0)			
BL_UID_S								
18 ... 23	BL_UID(47:0)							
AUX_DATA_S								
24 ... 31	unused							

Table 36: General Data Field & Transmission Sequence of Operation AUX_WRITEx

As described for AUX_READx, AUX_NOP must be used to check the status of the Multi Frame Operation AUX_WRITEx as well. Although no actual data is transferred back to the BLM, the Host should check, if the write operation has been processed completely before continuing to execute any other tasks.

OPERATION AUX_SETAOFFS

As described above, the address space for AUX_READx and AUX_WRITEx is limited to 128 byte. For BLS with larger address spaces, the operation AUX_SETAOFFS can be used to extend the available address space. Therefore, an address offset can be set within the BLS that is automatically added to the address used for AUX_READx and AUX_WRITEx operations. Table 36 shows the parameters necessary to configure AUX_SETAOFFS.

AUX_SETAOFFS								
Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BiSS Line Master								
00	0	0	0	1	0	1	1	1
01	0	0	0	0	0	0	0	0
BL_UID_M								
02 ... 07	BL_UID(47:0)							
AUX_DATA_M								
08 ... 13	unused							
14	ADDROFFS(15:8)							
15	ADDROFFS(7:0)							
BiSS Line Slave (STAT_AUX = 'Acknowledge' or 'Error')								
STATUS								
16	0	0	0	1	0	1	1	1
17	STAT_AUX		0	0	0	0	0	0
BL_UID_S								
18 ... 23	BL_UID(47:0)							
AUX_DATA_S								
24 ... 31	unused							

Table 37: General Data Field & Transmission Sequence of Operation AUX_SETAOFFS

The OPCODE_HEAD is 0x17. OPCODE_TAIL does not include any configuration parameters. Once set, ADDROFFS(15:0) is a 16-bit number that is used for all subsequent address operation as described above. It is not only considered for the next data access. To deactivate the address offset, AUX_SETADDROFFS needs to be executed with ADDROFFS = 0.
Since AUX_SETAOFFS belongs to the group of Multi Frame Operations, AUX_NOP is needed to complete the operation.

ADDROFFS(15:8)	Byte 14;	Bit 7:0	BLM
ADDROFFS(7:0)	Byte 15;	Bit 7:0	BLM
Code	Description		
0	No address offset (address offset disabled)		
1 ... 2^{16}-1	Address offset as defined by Code		

Table 38: Address Offset for AUX_READx and AUX_WRITEx

CHARACTERISTICS

In this chapter, general physical conditions about the *BiSS Line* protocol are listed.

Table 39 defines the physical characteristics for *BiSS Line*. The timing constants are referred to in the chapters accordingly.

No.	Symbol	Parameter	Min.	Max.	Unit
01	$1/T_{BL}$	*BiSS Line* Bit Rate	12.49875	12.50125	MHz
02	$l_{BiSSLine}$	Length of the Transmission Line	0	100	m
03	$t_{response}$	Maximum Response Time Measured at BLM	42.8		µs
04	$t_{release}$	Release Time Between BLM Sending and BLS Sending	0	240 [1]	ns
05	t_{IDLE_min}	Minimum Idle Time [2]	0.8		µs
06	RES_{TDLY}	Resolution of TDLY	5		bit
07	t_{LSB_TDLY}	Value of the Least Significant Bit of TDLY	5		ns
08	t_{TDLY}	Value of TDLY	0	155	ns
09	$1/T_{MA}$	BiSS C Bit Rate	$0.8 * 1/T_{BL}$		MHz
10	t_{ADF}	Duration of a Auxiliary Data Frame	54.4		µs
11	t_{ADF_BLM}	Duration of the Auxiliary Data Frame Request (BLM sending)	28		µs
12	t_{ADF_BLS}	Duration of the Auxiliary Data Frame Response (BLS sending)	24.8		µs
13	$t_{\Delta line}$	Delay Caused by the Transmission Line		0.8	µs
14	t_{ADF_wait}	Wait Time during broadcast ADF		26.4	µs

[1] It is recommended to keep $t_{release}$ as short as possible to keep the LINE in a defined state.
[2] The minimum idle time is one complete IDLE symbol. The required idle time of a system depends on the cable and the environment and can be longer than the minimum idle time.

Table 39: Table of characteristics

Timing Diagram of a general PDF

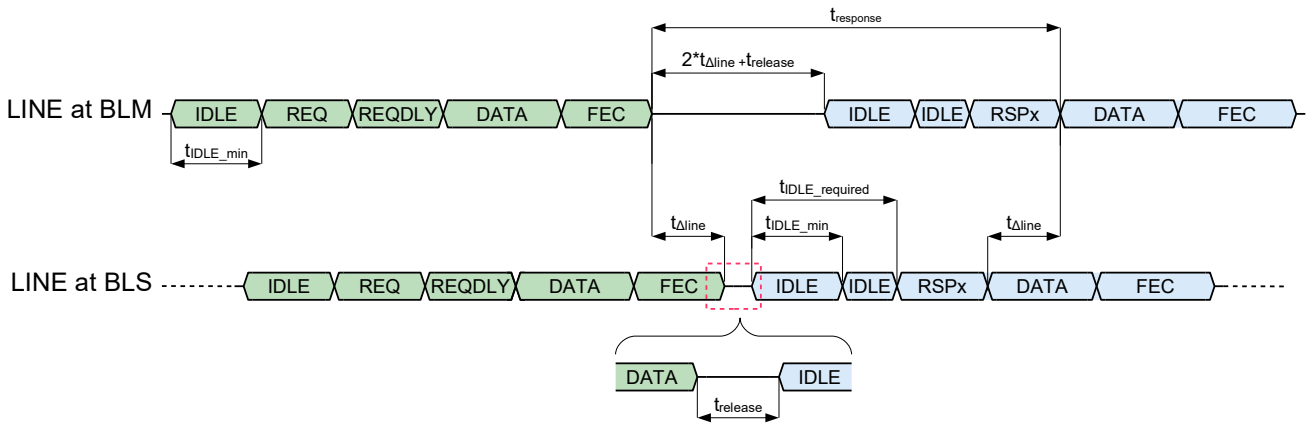

Figure 26: Timing diagram of a Process Data Frame

Timing Diagram of a general ADF

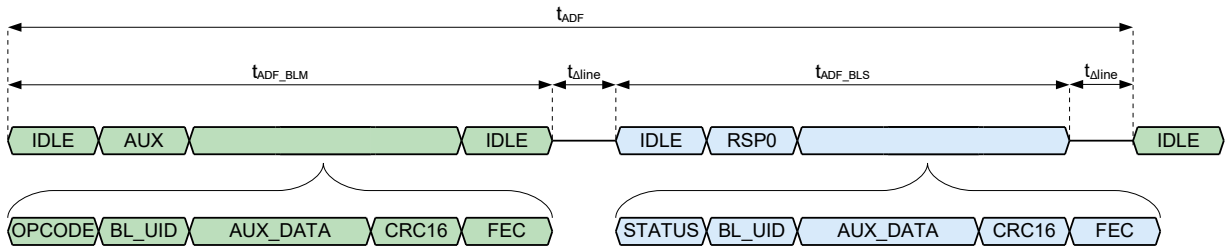

Figure 27: Timing diagram of an Auxiliary Data Frame

REVISION HISTORY

Rel.	Rel. Date*	Chapter	Modification	Page
A1	2022-04-04	All	Initial Preliminary Release	All

* Release Date format: *YYYY-MM-DD*