BiSS Interface

BiSS C Protocol Description

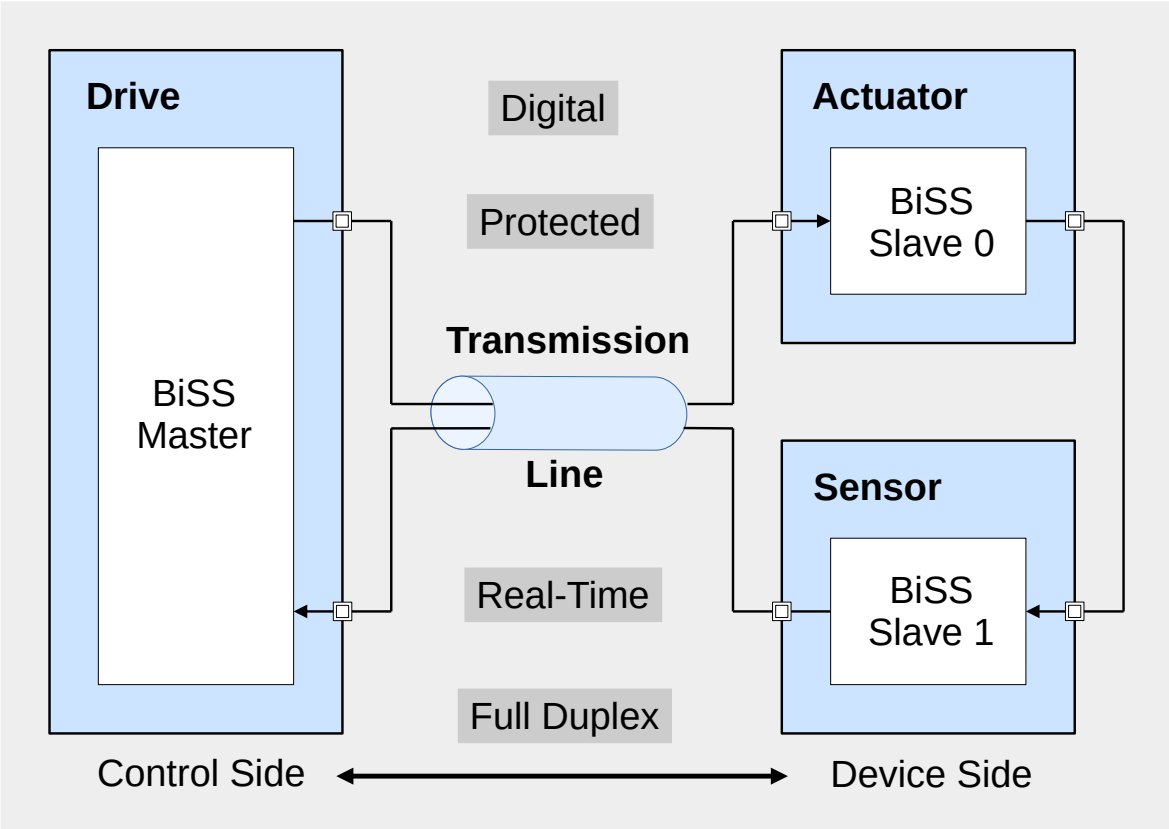
FEATURES

- Sensor/actuator interface
- Isochronous, real-time-capable data transmission
- Fast, serial
- Permanently bidirectional
- Point-to-point or multi slave networks
- Compact and cost-effective
- Open standard

APPLICATIONS

- Drives
- Rotary/Linear Encoders
- Robotics
- Smart sensors
- Safe actuators

BLOCK DIAGRAM



CONTENTS

OVERVIEW 3

POINT-TO-POINT CONNECTION 4

BiSS FRAME 6
- Idle . . . 6
- Header . . . 6
- Data Channels . . . 7
- Timeout . . . 7
  - Static Timeout . . . 7
  - Adaptive Timeout . . . 7
- Line Delay . . . 8
- BiSS Cycle . . . 9

PROCESS DATA COMMUNICATION 9
- Latch Point . . . 9
- Processing Time . . . 10
- Daisy Chain . . . 10
- Bus Reset/ Initialization . . . 12
- Propagation Delays . . . 12
- Null Value . . . 13

CONTROL COMMUNICATION 13
- Register Communication . . . 15
  - Register Read Access . . . 15
  - Register Write Access . . . 17
- *BiSS* Commands . . . 17
  - CMD = "00" (Process Data Channel) . . . 19
  - CMD = "01" (Control Communication) . . . 19
  - CMD = "10" (Bus Coupler/User-Defined) . . . 19
  - CMD = "11" (User-Defined/Reserved) . . . 19
- Short *BiSS* frame . . . 19
- Reduced *BiSS* Frame . . . 19

BiSS MEMORY MAP 20
- Register Protection Level . . . 20
- Bank Selection . . . 20
- EDS Bank . . . 21
  - Electronic Data Sheet . . . 21
  - User Banks (Optional) . . . 21
- BiSS Profile ID . . . 21
- Device Serial Number . . . 21
- Free Registers . . . 21
- Manufacturer ID and Device ID . . . 22

BUS CONNECTION 23

APPLICATION HINTS 26
- Initialization Example . . . 26
- Calculation of BiSS Cycle Time . . . 27
  - General Calculation for n Slaves: . . . 27
  - Simplified Calculation for one Slave: . . . 27
- Bus Coupler . . . 28

CHARACTERISTICS 29

LIST OF ACRONYMS 31

REVISION HISTORY 32

OVERVIEW

BiSS (**Bi**directional/**S**erial/**S**ynchronous) is a digital, serial interface protocol for fast and safe isochronous process data transmission, particularly used in motor feedback systems. Simultaneous to the reception of sensor process data and the transmission of actuator process data in real-time, the *BiSS* protocol is capable of register data transfers without interruption of the process data stream.

Today, BiSS Interface (abbreviated "BiSS") refers to the established BiSS C protocol that was introduced in 2007. The previous protocol definitions (e.g. BiSS B) are not recommended for new designs.

One *BiSS* Master module implemented in the drive is connected to one or more *BiSS* Slave modules that are included in the sensor and actuator devices of the system. In general, the *BiSS* Slave modules are connected to the *BiSS* Master module by three signals: a clock input signal (Clock) for synchronization purposes, a data input signal (Data In) transmitted to the *BiSS* Slave modules and a data output signal (Data Out) received from the *BiSS* Slave modules. In the following text, the signal directions for inputs and outputs are always referenced to the *BiSS* Slave modules as described previously.

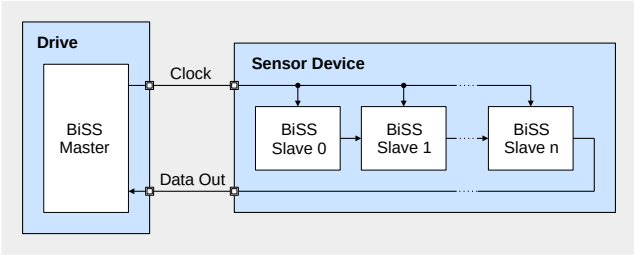


Figure 1: Point-to-point connection: master device connected to single sensor device

In the point-to-point connection, which is suitable for systems that consist of one sensor device only, the drive is connected to the sensor device solely with the clock line and the data output line. The data input line is omitted. Hence, *BiSS* devices in point-to-point configuration are hardware compatible with drives that only support the SSI protocol.

As shown in Figure 1, it is possible to daisy-chain multiple *BiSS* Slave modules within one sensor device by forwarding the data output line to each succeeding *BiSS* Slave module and then finally back to the *BiSS* Master module. Subsequently, the short terms *BiSS* Master and *BiSS* Slave are used to indicate the *BiSS* Master module and the *BiSS* Slave module respectively in contrast to the *BiSS* devices.

The data exchange between the *BiSS* Master and the *BiSS* Slave is controlled by consecutive *BiSS* frames that are isochronously triggered in configurable *BiSS* cycles. A *BiSS* frame is divided into several logical process data channels that are individually protected by a cyclic redundancy check (CRC). Each *BiSS* Slave occupies one logical process data channel that is transmitted in serial through the entire daisy-chain for every *BiSS* frame completely. Hence, the minimum *BiSS* cycle time depends on the current process data channel configuration.

For register data transfer, every *BiSS* frame includes one control data bit in each direction to and from the *BiSS* Slave. These bits are collected from successive *BiSS* frames to form a subordinate transmission frame with slower throughput. The control data frame is used for secondary data transmission to and from a specific *BiSS* Slave. Each *BiSS* Slave is automatically assigned a unique identification number determined by its position in the daisy-chain.

The register data transfer is typically used for configuration/calibration purposes and device information in the form of an Electronic Data Sheet. However, the control data frame can also be used to transmit additional sensor data that is more static in nature, such as temperature.

In addition to the register data transfer, the control data frame offers predefined and custom commands that can be addressed to specific *BiSS* Slaves or broadcast to every *BiSS* Slave connected to the system. The *BiSS* commands can be used to facilitate the startup procedure at the *BiSS* Master or to synchronize certain events regarding specific application for the entire system, position data presets for example. To ensure proper data transmission even in high-interference environments, each control data frame is also protected by its own Cyclic Redundancy Check in addition to the data protection of the *BiSS* frame.

For systems with more than one sensor device, the bus connection is used. In contrast to the point-to-point connection, the bus connection also enables data transmission to actuator devices. Therefore, the third line for data input is connected to the *BiSS* Slaves. Figure 2 shows several *BiSS* Slaves that are connected to the *BiSS* Master using the bus configuration.

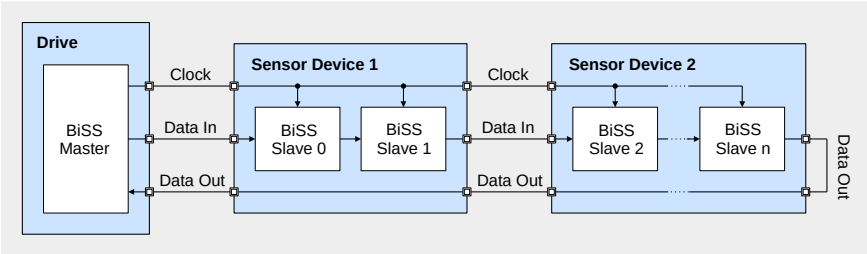


Figure 2: Bus connection: master device connected to several daisy-chained sensor devices

Since the daisy-chain is effectively a long shift register, the actuator data is shifted into the chain and the sensor data is shifted out of it. Therefore, the *BiSS* Slaves that represent actuators use the same process data channels with the same configuration properties as described for the point-to-point connection. The bus connection can even be used to daisy-chain entire *BiSS* devices that include multiple *BiSS* Slave modules themselves to create an even more complex *BiSS* system.

In order to synchronously capture and apply process data for the entire system, the *BiSS* protocol offers a mechanism that can be used to simultaneously trigger data processing within every *BiSS* Slave connected to the *BiSS* Master. The mechanism is designed to reduce the timing jitter that is critical for high precision control loop applications. Furthermore, it is possible to delay the process data transmission at the beginning of the *BiSS* frame in order to support devices that require additional internal processing time.

The *BiSS* cycle is synchronized by a predefined timeout that each *BiSS* Slave recognizes at the end of the *BiSS* frame separately. For ease of implementation, any line delay due to long transmission lines and slow line drivers are automatically compensated by the *BiSS* Master. Due to the extensive data protection, the *BiSS* protocol is also suitable for safety-critical applications up to SIL 3 considering the BiSS Safety Profile.

POINT-TO-POINT CONNECTION

As described in chapter OVERVIEW, *BiSS* is primarily used for multi-slave isochronous process data transmission in real-time control loop applications. For this purpose, one *BiSS* Master device is connected to one or more *BiSS* Slave devices. The single *BiSS* Master device only includes one *BiSS* Master, the *BiSS* Slave devices on the other hand may include multiple *BiSS* Slaves.

However, in the point-to-point configuration, the *BiSS* Master device is only connected to a single *BiSS* Slave device. The most common *BiSS* Slave device for point-to-point connections includes only one *BiSS* Slave as shown in Figure 3.

In point-to-point connection only one *BiSS* slave device is connected to the master device. The *BiSS* slave device may only contain sensors.

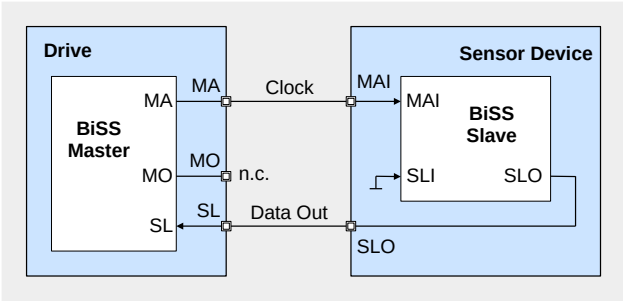


Figure 3: Point-to-point connection with a single *BiSS* Slave

The *BiSS* Master is connected to the clock line by its MA pin and to the data output line by its SL pin. The MO pin for the data input line is not used, therefore it is not possible to use an actuator *BiSS* Slave in point-to-point configuration.

The *BiSS* Slave within the sensor device is connected to the clock line by its MAI pin and to the data output line by its SLO pin. Since the data input signal is not used, the SLI pin of the *BiSS* Slave is connected to ground.

Note that in devices intended solely for point-to-point connections, the data input pin SLI is not accessible outside the device. It is internally connected to ground.

The clock signal generated at the MA pin controls the communication frame. It delivers the clock source for the *BiSS* Slave and defines the start and end of the transmission cycle. Furthermore, the clock signal is used for transmitting the single control data bit from the *BiSS* Master to the *BiSS* Slave for control communication. More information about the clock signal and control communication can be found in chapters *BiSS* FRAME and CONTROL COMMUNICATION respectively.

The data output signal generated at pin SLO is used to transmit the sensor data from the *BiSS* Slave back to the *BiSS* Master. Additionally, the data output signal is used to determine the appropriate digital sample point for the *BiSS* Master and to transmit the response of the *BiSS* Slave for control communication. More information about the flexibility of the digital sample point in consideration of signal delay due to long transmission lines can be found in chapter Line Delay.

In general, *BiSS* is a digital protocol and the *BiSS* Master and the *BiSS* Slave generate their data signals on the rising edge on the clock signal MA. The sample point of the *BiSS* Slave is at the falling edge of MA. As mentioned previously, the sample time of the *BiSS* Master referred to the clock signal MA is not fixed and can be used to compensate for line delays. However, as a starting point, it is convenient to assume that the system does not introduce any line delay and SL is sampled at an appropriate point in time during the clock period on MA as shown in Figure 4.

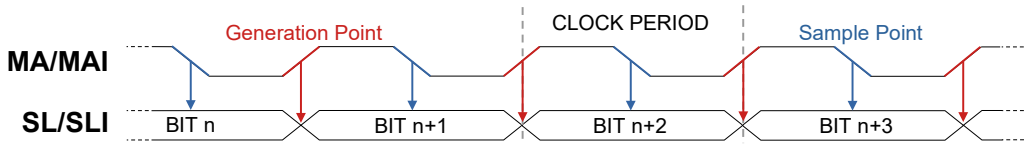


Figure 4: *BiSS* data generation and sampling

Amongst other factors, the clock period in Figure 4 affects the throughput of the *BiSS* data transmission. The frequency range and duty cycle of the clock is specified in chapter CHARACTERISTICS. In addition to the restrictions of the *BiSS* protocol itself, the clock signal is subject to the transmission line as well. For proper data transmissions it is recommended to follow the standard RS422 protocol as shown in Figure 5. More information about RS422 can be found in standard TIA-422 of the Telecommunications Industry Association.

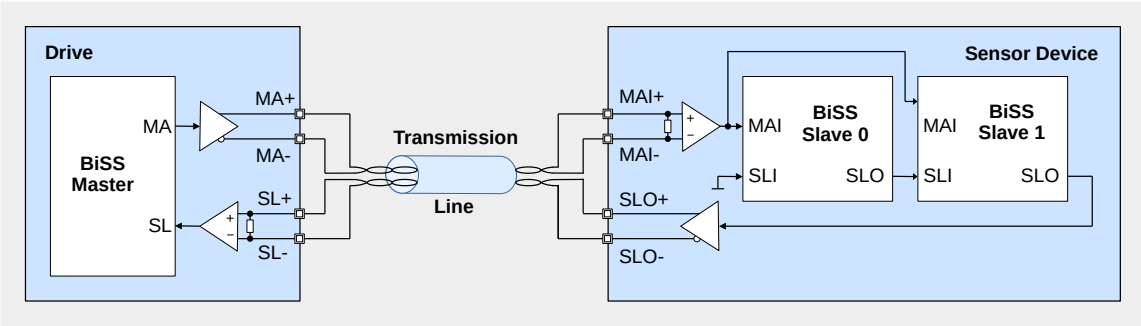


Figure 5: RS422 in point-to-point connection

If necessary, it is possible to daisy-chain multiple *BiSS* Slaves even in point-to-point connections. Figure 5 shows an example of a sensor device with two integrated *BiSS* Slaves. However, for point-to-point configurations it is not possible to connect multiple sensor devices to the system. More information about daisy-chaining of *BiSS* Slaves can be found in chapter PROCESS DATA COMMUNICATION in section Daisy Chain.

BiSS FRAME

Since the *BiSS* frame does not change considerably for different physical connections, it is reasonable to introduce its structure referring to the simple system as shown in Figure 3 on page 4. An example *BiSS* frame for this system is shown in Figure 6. The following descriptions correspond to the signals MAI and SLO at the *BiSS* Slave.

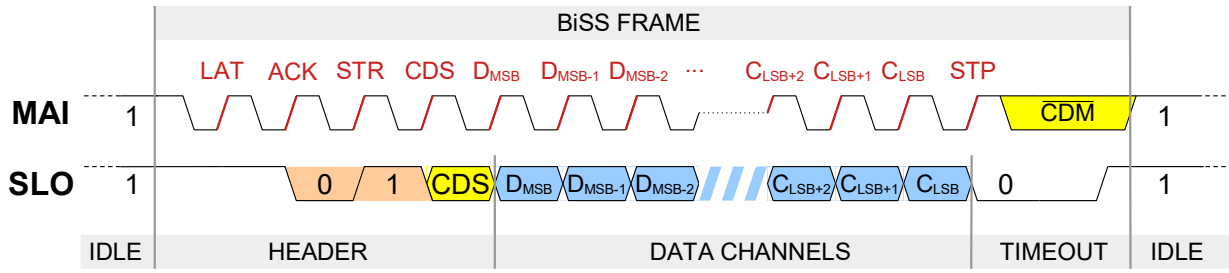


Figure 6: A simple *BiSS* frame

As described in chapter OVERVIEW, the *BiSS* frame is controlled by the *BiSS* Master via the clock signal at pin MAI. The sequence of MAI can be divided into four main phases:

- Idle
- Header
- Data Channels
- Timeout

During Idle, there is no data transmission. The Header initializes a new *BiSS* frame and the Data Channels are used for transmission of the process data. The Timeout synchronizes the *BiSS* Slave to be ready for the next *BiSS* cycle.

The length of each phase actually depends on the *BiSS* Slave characteristics and the current *BiSS* configurations. Each phase (Header, Data Channels and Timeout) determines the minimum *BiSS* cycle time supported by the *BiSS* system. The *BiSS* Master needs to be configured accordingly. More information about the *BiSS* cycle can be found in section BiSS Cycle. However, for ease of understanding it is beneficial to cover the simplest case first. The different phases from Figure 6 are discussed in the following sections.

Idle

Since the *BiSS* frame is typically transmitted within regular cycles, there is a phase after the current cycle ends and before the next frame starts. This phase is called Idle. Typically, there is no data transmission during Idle and both pins MAI and SLO are constantly high (1). The only exception is the feature Hold-CDM, which is described in chapter CONTROL COMMUNICATION.

Header

Figure 7 shows the Header of a typical *BiSS* frame that starts a new *BiSS* cycle and prepares the data of the *BiSS* Slave to be transmitted via SLO. The start of the *BiSS* Frame is initiated by a falling edge followed by a rising edge on MAI. The first rising edge LAT (latch) is used as a trigger to latch or generate the process data within the *BiSS* Slave. More information about data synchronization can be found in section Latch Point.

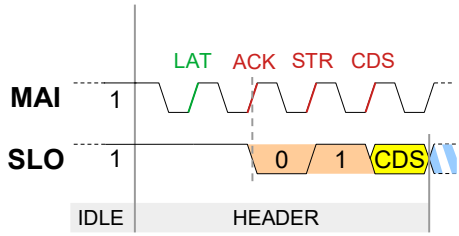


Figure 7: The Header of a simple *BiSS* frame

As described in chapter POINT-TO-POINT CONNECTION, the *BiSS* Slave generates its response on SLO at the rising edge of the clock signal on MAI. Hence, except for LAT, each rising edge represents one bit of the *BiSS* frame. The first bit (ACK) is used by the *BiSS* Slave to acknowledge the beginning of a new *BiSS* cycle to the *BiSS* Master. Since the Idle phase of SLO is "1", the acknowledge bit is always "0". Detailed information about how the *BiSS* Master processes the acknowledge bit can be found in section Line Delay.

The start bit (STR) indicates that the *BiSS* Slave is ready to transmit its process data to the *BiSS* Master. As described in chapter PROCESS DATA COMMUNICATION in section Processing Time, the start bit can be delayed for devices that introduce processing time to the system. After the start bit, the actual data transmission starts with the Control Data Slave bit (CDS). In contrast to the process data, the CDS bit is part of the control communication and therefore dedicated to the

Header of the *BiSS* frame. The control communication is described in chapter CONTROL COMMUNICATION.

Data Channels
The Header is followed by the Data Channels as shown in Figure 8. Following the CDS bit, the Data Channels transmit process data of up to 64 bits for each connected *BiSS* Slave. Since the example above includes one single *BiSS* Slave, this *BiSS* frame transmits one data channel (D_{MSB}...D_{LSB}) only. Each data channel is typically protected by a cyclic redundancy check (CRC) of up to 16 bits (C_{MSB}...C_{LSB}). The CRC bits are inverted before transmission. The Data Channels are completely transmitted in each *BiSS* cycle and therefore offer the highest throughput. More information about the Data Channels can be found in chapter PROCESS DATA COMMUNICATION.

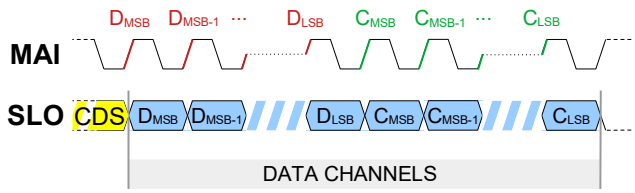
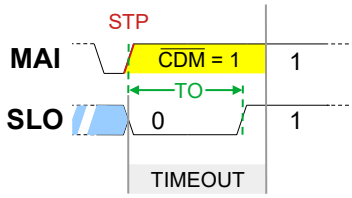


Figure 8: The Data Channels of a simple *BiSS* frame

Timeout
The last phase of a *BiSS* frame is the Timeout. The Timeout is used to synchronize the end of a *BiSS* cycle at the *BiSS* Slaves and the *BiSS* Master. For that reason, a handshake mechanism is provided. The *BiSS* Master stops the clock signal on MAI and the Timeout (TO) is individually measured by each *BiSS* Slave. Once expired, the Timeout is indicated to the *BiSS* Master by a rising edge at SLO as shown in Figure 9. Therefore, the stop bit is generated by the *BiSS* Slave at the last rising edge on MAI (STP). In order to transmit the Control Data Master bit (CDM) for control communication, the *BiSS* Slave samples the clock line MAI at the end of the Timeout.

The handshake mechanism ensures, that the *BiSS* Master does not start another *BiSS* cycle before all connected *BiSS* Slaves are ready to process another *BiSS* frame. As explained before, each *BiSS* Slave determines its own Timeout interval independently. Therefore, at the point in time when the clock line MAI stops toggling, the *BiSS* Slave starts an internal timer. When the timer expires, the data output SLO is set to "1", the CDM bit is evaluated and the *BiSS* Slave goes back to Idle. If the stop bit "0" was properly generated, the *BiSS* Master is notified about the expired Timeout by a rising edge on SLO and enters its Idle state as well. The whole system is now ready to start a new *BiSS* cycle.

CDM = 0:



CDM = 1:

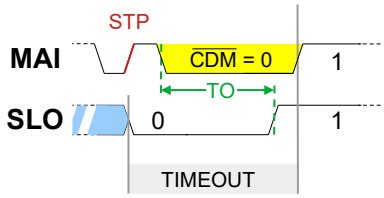


Figure 9: The Timeout of a simple *BiSS* frame

There are two types of Timeouts that are supported in standard *BiSS* systems:

- Static Timeout
- Adaptive Timeout

The static Timeout is a constant time interval (often about 20 µs) predefined in the *BiSS* Slave. The adaptive Timeout on the other hand is adjusted at the beginning of every *BiSS* frame considering the current *BiSS* clock frequency in order to minimize the cycle time for high performance applications.

Static Timeout
Figure 10 shows a *BiSS* frame with static Timeout. For high MAI frequencies, the Timeout interval is much longer than the clock period (T_{MA}).

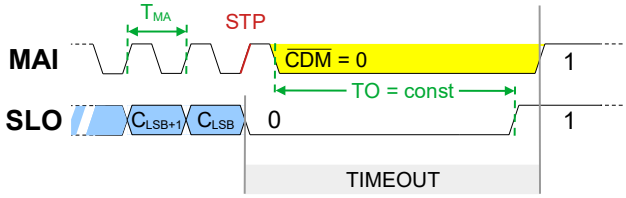


Figure 10: Static Timeout

The limits of the static Timeout are specified in chapter CHARACTERISTICS. Since every *BiSS* Slave at least needs to include the static Timeout, the *BiSS* Master is able to cancel a *BiSS* frame at any time. After waiting for the predefined maximum static Timeout interval, each *BiSS* Slave must be in the state Idle and ready to start a new *BiSS* cycle.

Adaptive Timeout
The adaptive Timeout as shown in Figure 11 depends on the currently configured clock frequency on MAI. The

adaptive Timeout interval is 1.5 times the MAI clock period (T_{MA}) and therefore compatible with the entire *BiSS* frequency range as well.

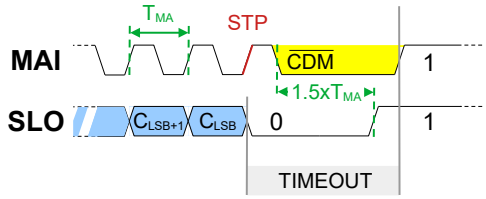


Figure 11: Adaptive Timeout

In comparison to Figure 10, the Timeout phase is much shorter which provides shorter *BiSS* cycle times and higher throughput. However, the adaptive Timeout is only as accurate as the sample clock of the *BiSS* Slave.

Figure 12 shows the measurement of the adaptive Timeout during the Header phase of the *BiSS* frame. Slow sample clocks may produce longer adaptive Timeouts. *BiSS* devices that support the adaptive Timeout still use the static Timeout as maximum during measurement of the first 1.5 clock periods.

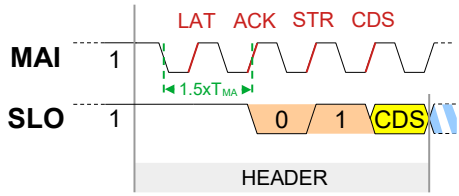


Figure 12: Measurement of the adaptive Timeout

Line Delay

As described before, the *BiSS* frame as shown in Figure 6 represents the signals MAI and SLO at the *BiSS* Slave. In real world applications, the clock signal MA generated by the *BiSS* Master and the data output SLO generated by the *BiSS* Slave are transferred via transmission lines that might add signal delay to the *BiSS* communication system. Therefore the *BiSS* frame from the example above at the *BiSS* Master could look as shown in Figure 13.

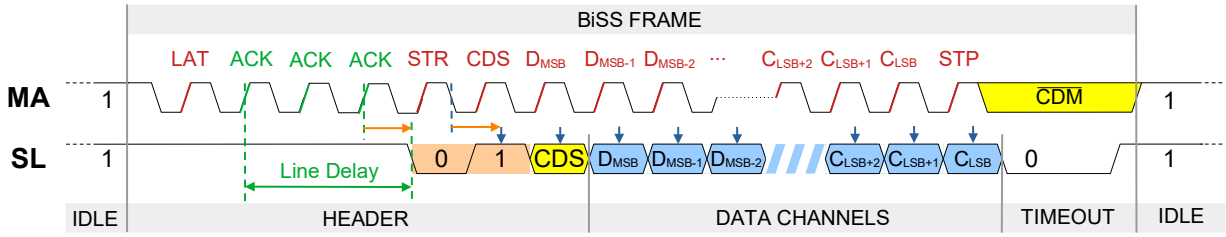


Figure 13: A *BiSS* frame with line delay at the *BiSS* Master

The length of the Header is extended by two clocks due to the delayed response of the *BiSS* Slave to the acknowledge bit (ACK). The *BiSS* Master inserts additional MA clocks into the Header until the acknowledge bit "0" is received on SL. Furthermore, the *BiSS* Master analyzes the delay within the last acknowledge clock period to adapt its sample points for the residual data bits. Since the additional MA clocks only shift the entire frame, the following sequence of the *BiSS* communication stays the same as described before.

The line delay compensation is suitable for systems with long transmission lines. Additionally, even protocol translators can be included into the transmission line without interfering with the *BiSS* communication. However, the response of the *BiSS* Slave by means of the acknowledge bit has to occur within a certain time interval. Exceeding this time interval must be indicated by the drive as an error. The maximum line delay that is allowed for standard *BiSS* systems is specified in chapter CHARACTERISTICS.

BiSS Cycle

Since *BiSS* can be used for isochronous process data transmission in high precision control applications, it is reasonable to repeatedly start a *BiSS* frame with a pre-configured period. Typically, the *BiSS* Master is set up to implement a fixed *BiSS* cycle. The *BiSS* cycle time is defined by the application. However, it is required to consider the length of the maximum *BiSS* frame in order to determine the minimum cycle time possible. The minimum cycle time can be calculated as described in chapter APPLICATION HINTS section Calculation of BiSS Cycle Time.

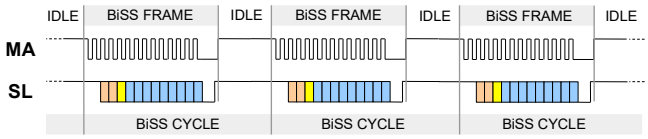


Figure 14: Isochronous *BiSS* cycles

Figure 14 shows several *BiSS* cycles that are isochronously triggered by the *BiSS* Master.

The *BiSS* frame must be finished by the Time-out indicated with the rising edge on SLO before the next *BiSS* cycle is started.

PROCESS DATA COMMUNICATION

The Data Channels as introduced in chapter *BiSS* FRAME are used for a protected, isochronous process data transmission with high throughput. Each *BiSS* Slave connected to the *BiSS* Master can be configured to occupy one data channel of up to 64 bits protected by a Cyclic Redundancy Check (CRC) of up to 16 bits. Typically, the data channels are used for sensor and actuator data transfers that need to be transmitted completely in every *BiSS* cycle. For ease of understanding, this chapter examines sensor data only. Actuator data is explained in chapter BUS CONNECTION.

Figure 15 shows a typical use case for a *BiSS* Slave transmitting multi-turn and single-turn position data of a rotary encoder. However, the data channels can be used for arbitrary data. The *BiSS* protocol does not prescribe the type and format of data that is transferred via process data communications. For standardization, predefined *BiSS* Profiles and Electronic Datasheets (EDS) are used.

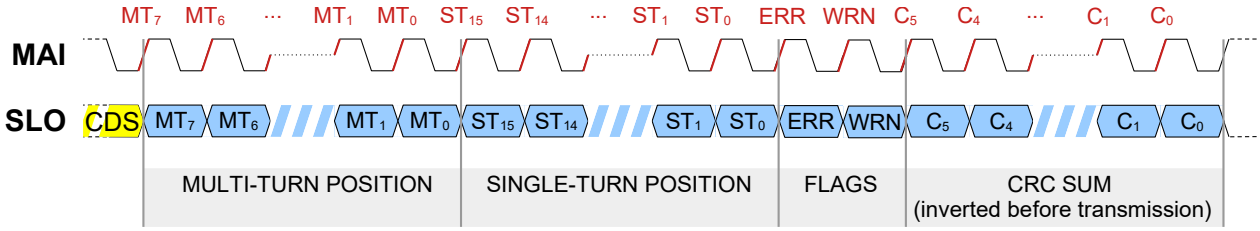


Figure 15: Typical Data Channel content

Referring to the simple example from Figure 3 on page 4 only one data channel is occupied. The data channel is configured for position data of 8-bit multi-turn and 16-bit single-turn. Furthermore, the data channel transmits status information in the form of error and warning flags. The entire data channel is protected by a 6-bit Cyclic Redundancy Check. If another data channel is activated, the additional data bits directly follow the first data channel. Information about multiple data channels can be found in section Daisy Chain.

For high precision control applications, it is important to generate new sensor data on a regular basis. Therefore, the *BiSS* frame offers a mechanism to synchronously trigger the generation of new sensor data at the beginning of each *BiSS* cycle as described in section Latch Point.

If the *BiSS* device needs significant time to generate or prepare its process data, the transmission of the data channels can be delayed. The start bit within the Header notifies the *BiSS* Master about the required processing time. More information about the delay of the start bit can be found in section Processing Time.

Latch Point

For high precision control applications it is important to generate sensor data at a specific point in time. For that reason, the generation of process data is triggered at the beginning of each *BiSS* cycle at the first rising edge on MAI (see LAT in Figure 7 on page 6).

If the *BiSS* cycle is isochronously triggered by the *BiSS* Master, the sample edge of the *BiSS* frame LAT and the latch point for process data occurs isochronously as well. If the sample edge of the *BiSS* frame is processed such that the generation of the process data has a constant delay only, the control always receives sensor data from a fixed point in time referenced to the start of the *BiSS* cycle. Consequently, the time jitter of the data generation is reduced to the processing performance of the *BiSS* slave device. Figure 16 shows the constant delay of the data generation in reference to the sample edge of the current *BiSS* frame.

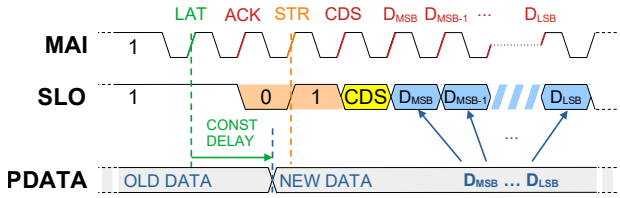


Figure 16: Constant delay of data generation

The process of data generation is triggered by the rising edge LAT within the Header. The *BiSS* slave device requires time to process the data acquisition and latch the new process data to the *BiSS* Slave (PDATA). The new process data is transmitted back to the *BiSS* Master in Data Channels following the CDS bit.

In a point-to-point connection, the clock signal MA is applied in parallel to any *BiSS* Slave connected to the system. The latch point can be used to synchronize the process data acquisition of the entire daisy chain. More information about multiple *BiSS* Slaves connected in point-to-point connection can be found in section Daisy Chain.

Processing Time

The time between the latch point (LAT) and the start bit is defined as the processing time (t_{busy}). Due to the frame's characteristics, the processing time of a *BiSS* slave device is at least two clock periods long. For devices with significant processing time (greater than two clock periods), the generation of the start bit within the Header is delayed. This feature enables the *BiSS* system to latch sensor data at the beginning of a *BiSS* cycle and process it before transmission. Figure 17 shows an example of a *BiSS* Slave that needs more than two clock periods for generating the process data.

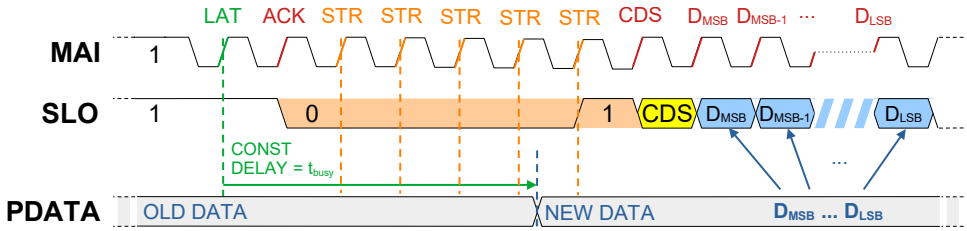


Figure 17: Delayed *BiSS* frame with significant processing time

In a point-to-point connection, the transmission of the start bit is solely controlled by the *BiSS* Slave. If the device needs more time to generate its process data, the *BiSS* Slave sends "0"s following the acknowledge bit (ACK) instead of the start bit (STR). The start bit is suppressed until the process data is ready for transmission, then the sequence of the *BiSS* frame continues as usual. The maximum value for the processing time (t_{busy}) that has to be supported by the *BiSS* Master is defined in chapter CHARACTERISTICS.

For systems in a bus connection, the start bit delay is controlled by the *BiSS* Master as described in chapter BUS CONNECTION.

Daisy Chain

Even in point-to-point connections, it is possible to include multiple *BiSS* Slaves into the *BiSS* slave device. The clock signal MA is connected to each of the *BiSS* Slaves in parallel. The data output line SLO is connected to the data input line SLI of the subsequent *BiSS* Slave, so that the frame data of the left-most *BiSS* Slave is shifted through the entire daisy chain. Figure 18 shows three *BiSS* Slaves daisy-chained within one *BiSS* slave device which is connected to the *BiSS* Master in a point-to-point connection.

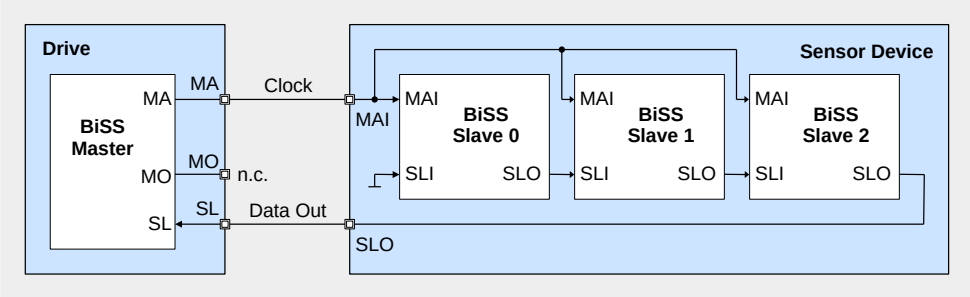


Figure 18: Daisy chain within a *BiSS* Slave device in point-to-point connection

The daisy-chain works like a long shift register that moves the process data of each *BiSS* Slave back to the *BiSS* Master. Since the data output signal SLO of the right-most *BiSS* Slave 2 is directly connected to SL, the *BiSS* Master receives its process data first. Due to an identification scheme, that is described in chapter CONTROL COMMUNICATION, the *BiSS* Slave that connects the daisy chain with the *BiSS* Master gets the highest slave number (here Slave 2). The *BiSS* Slave on the other end of the daisy chain, whose data input signal SLI is connected to ground, is always referred to as Slave 0 of the *BiSS* device.

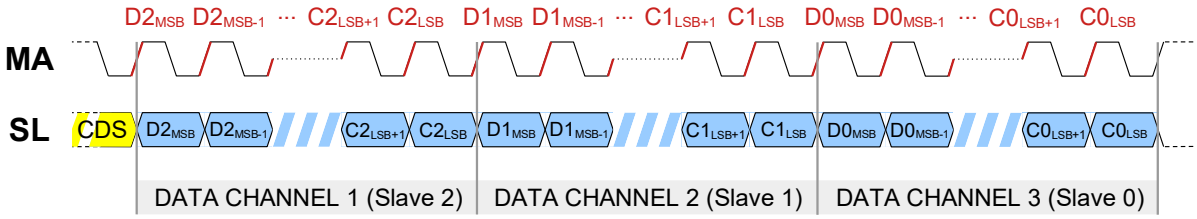


Figure 19: Data Channels of daisy-chained *BiSS* Slaves

Figure 19 shows the Data Channels of the three daisy-chained *BiSS* Slaves. As described above, the three data channels are transmitted one after another, each one independently protected by its own CRC. The first data received by the *BiSS* Master (Data Channel 1) belongs to Slave 2 as labeled in Figure 18. The second and third data channel (Data Channel 2 and 3) belong to Slave 1 and Slave 0 respectively.

The order of the data channels for process data communication is inverted to the order of the slave IDs for control communication.

The reason for this inverse order is that the mechanism for the identification scheme always starts at Slave 0, but the data channel received first by the *BiSS* Master corresponds to the *BiSS* Slave that is directly connected to SL.

Since the clock signal MA is connected in parallel, each *BiSS* Slave is triggered for process data generation simultaneously at LAT. If the *BiSS* Slaves need different processing times for process data generation, the *BiSS* Slave with the longest processing time determines the start bit delay as explained in section Processing Time. Therefore, after finishing its own data generation, each *BiSS* Slave hands over the data output signal SLO of the predecessor which is connected to the data input signal SLI. Only Slave 0 actually generates the start bit which is shifted through the entire daisy chain back to the *BiSS* Master.

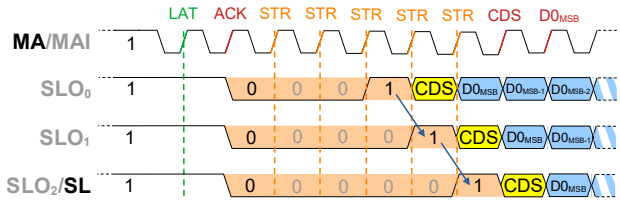


Figure 20: Start bit delay of daisy-chained *BiSS* Slaves

Figure 20 shows the start bit delay for the three daisy-chained *BiSS* Slaves as described previously. The acknowledge bit (ACK) forces each data output signal SLO low (0). Only Slave 0 (SLO_0) actively generates the start bit at SLO when ready. The other *BiSS* Slaves (SLO_1 and SLO_2) capture the data input signal SLI that is handed over to the corresponding SLO after their process data generation is done. The data output signal of Slave 2 is connected to the *BiSS* Master via SL. Since the start bit is delayed by one clock per slave due to this capture mechanism, the *BiSS* Master always detects processing time for any daisy chain, even if the *BiSS* Slaves themselves do not need additional time to generate their process data.

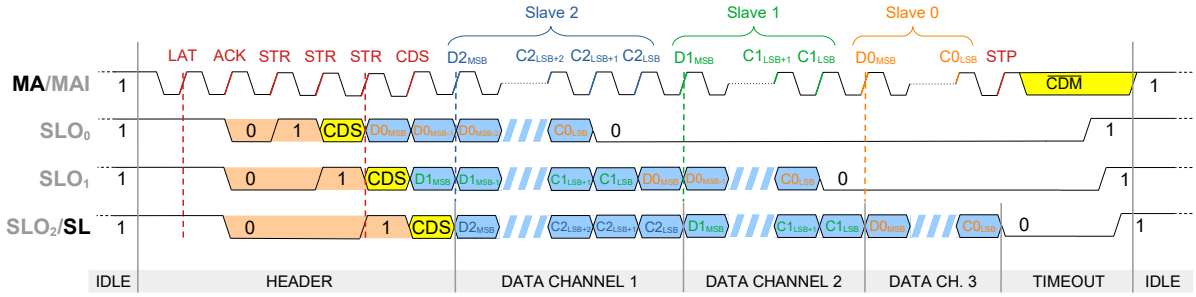


Figure 21: An entire *BiSS* frame passing through the daisy chain

As the following *BiSS* Slaves wait for the start bit before clocking out their process data, the *BiSS* Slave with the largest processing time is placed as Slave 0.

Figure 21 shows an entire *BiSS* frame passing through the daisy chain. As described above, the *BiSS* Master detects processing time at Slave 2 although Slave 0 (SLO_0) does not produce any actual start bit delay for process data generation. After capturing the CDS bit from its predecessor, each *BiSS* Slave in the daisy chain processes it for control communication before generating its own CDS bit. More information about the control communication can be found in chapter CONTROL COMMUNICATION.

During output of its own process data (Data Channel 1 at SLO_2 in blue) after the start bit, Slave 2 needs to buffer the process data of its predecessors (Data Channel 2 in green and Data Channel 3 in orange at SLO_1) as connected to the data input line SLI (SLI_2). Slave 1 in the middle on the other hand needs to buffer the frame data of Slave 0 accordingly. Hence, each *BiSS* Slave of a daisy chain must be able to buffer frame data from its predecessor up to the size of its maximum process data configuration plus CDS. This is why the *BiSS* Slave with the largest processing time is placed as Slave 0.

When finished outputting its own process data, each *BiSS* Slave again hands over the frame data received at the input data signal SLI as described for the start bit mechanism at the beginning of the frame. Thus, it is not necessary to configure a single *BiSS* Slave for the frame length of the entire daisy chain. Instead, only Slave 0 actively generates the stop bit for timeout indication.

When the *BiSS* Master stops toggling the clock signal MA, the data output signal SLO of any *BiSS* Slave is automatically set low (0). The timeout is independently processed by each *BiSS* Slave as described in chapter Timeout. However, to synchronize the timeout mechanism, each *BiSS* Slave only enters Idle when its predecessor indicates the detection of the timeout at SLI. Otherwise, the data output signal SLO is held low for correctly notifying the *BiSS* Master about the timeout detection of the entire daisy chain.

Bus Reset/ Initialization

After power-on and after an error, the *BiSS* Master must break *BiSS* communication for 40 µs before starting a new *BiSS* frame to ensure that the *BiSS* timeout has expired and all slaves are ready for the data transmission.

In point-to-point connection start and stop bit of the *BiSS* frame have to be generated by the Slave 0. In order to inform Slave 0 about its active role, its data input line is connected to ground and the system is initialized after power-on. To this end the *BiSS* Master sends at least two low pulses at MA as shown in Figure 22. After timeout expiration, the system is ready for *BiSS* communication and the idle level of the data signal SL should be "1".

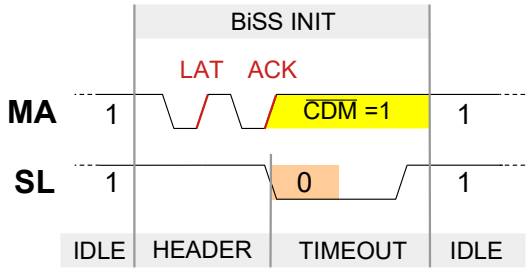


Figure 22: Example for a typical initialization sequence

Propagation Delays

In order to ensure proper data communications, it is important to pay attention to the propagation delays of the clock input signal MAI and the data input signal SLI. For large lags between MAO and SLO, there might be an issue with synchronous data transmission between two *BiSS* Slaves of a daisy chain. Since there is no line delay compensation as described for the *BiSS* Master, the data from the predecessor is sampled at the next falling edge considering the parallel connection to the

clock line MAI. If the data output signal SLO is not stable before the falling edge, the data shifting fails, which leads to errors during *BiSS* frame processing.

In order to prevent lags between MAO and SLO, MAO can be delayed to be synchronous to SLO as described in chapter BUS CONNECTION.

Null Value
In *BiSS*, the Null Value (all data bits D_{MSB} ... D_{LSB} equal "0") is used to indicate invalid sensor or actuator data. It is typically transmitted by a *BiSS* Slave to indicate that its sensor data has not been updated since the previous *BiSS* frame or by the *BiSS* Master to indicate that the actuator data has not been updated. The Null Value is also transmitted in a daisy-chain or bus topology, if data received by a *BiSS* Slave at SLI cannot be (completely) buffered. In that case, the start bit might be delayed before transmitting the Null Value.

Typically, a data channel contains status bits that are active-low (e.g. error and warning bit as defined in BiSS Profile BP3). This enables output of valid sensor data equal to zero (error bit is "1"). In case of invalid sensor data, all position and status bits will be "0" indicating an error.

CONTROL COMMUNICATION

In contrast to the *SSI* protocol, *BiSS* is capable of data transmissions in both directions, from the *BiSS* Master to the *BiSS* Slave and the other way round. The data transfer from the *BiSS* Master to the *BiSS* Slave can be used to access the *BiSS* Slave's memory resources, e.g. for sensor configuration and calibration. Since the control communication is embedded into the *BiSS* frame, the process data transmission is not interrupted for these data transfers.

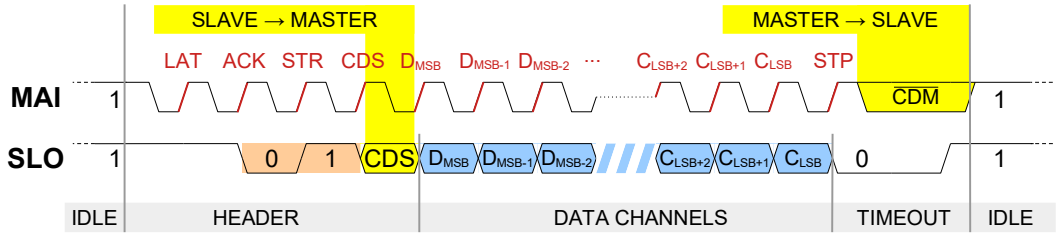


Figure 23: CDM and CDS bits

As specified in chapter *BiSS* FRAME, each *BiSS* frame includes two bits for control communication. Figure 23 shows the CDM bit and the CDS bit that are used for control data transmission between the *BiSS* Master and *BiSS* Slave in addition to the process data transmission. The CDM as well as the CDS bit are transmitted within every *BiSS* frame and composed into a subordinate data frame with lower throughput compared to the process data transmission. Figure 24 shows several frames that transmit multiple CDM and CDS bits for control communication.

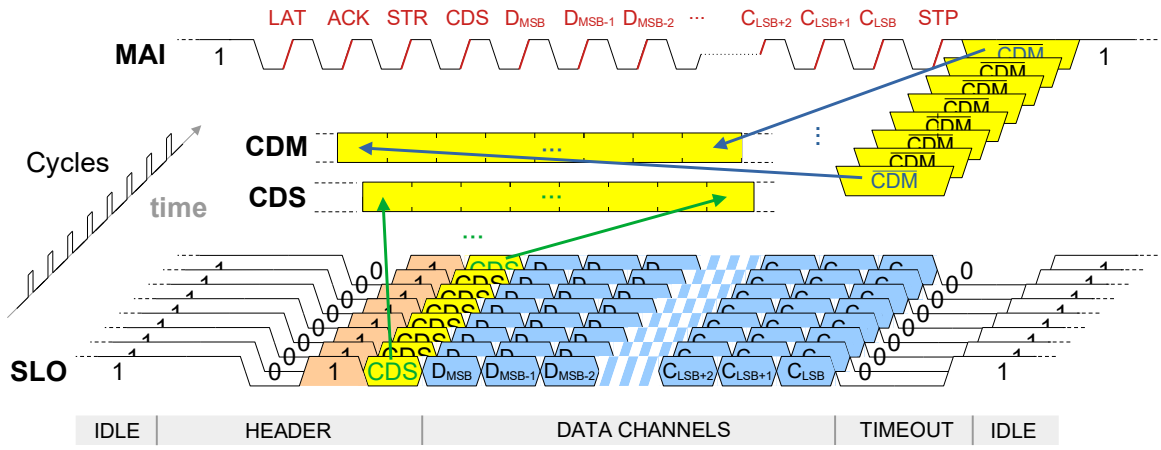


Figure 24: Multiple CDM and CDS bits compose the control data frame

The single CDM bits (inverted before transmission) are generated by the *BiSS* Master at the end of the *BiSS* cycle and captured by the *BiSS* Slave during Timeout as described in chapter *BiSS* FRAME. The *BiSS* Slave processes the transmitted information and responds to the *BiSS* Master by generating the CDS bit at the beginning of the next *BiSS* cycle. Thus, the *BiSS* Master is responsible for starting, defining the type and terminating the control communication.

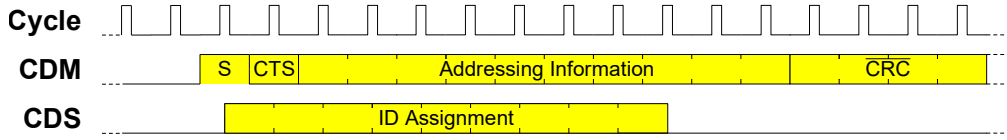


Figure 25: Header of the control data frame

Figure 25 shows the header of each control data frame. As long as no control data frame is being processed, the *BiSS* Master sends CDM = "0" and the *BiSS* Slave responds with CDS = "0". The *BiSS* Master triggers a new control communication by sending a control data start bit (S) on CDM, which starts the processing of a new control data frame in the *BiSS* Slave. The *Control Select* bit (CTS) defines what type of control communication is being selected by the *BiSS* Master.

- CTS=0: *BiSS* Commands (page 17)
- CTS=1: Register Communication (page 15)

After the *Control Select* bit, the *BiSS* Master transmits the 10-bit addressing information, which is processed by every *BiSS* Slave connected to the daisy chain. The header of each control data frame is protected by a 4-bit Cyclic Redundancy Check (polynomial 0x13). The CRC is inverted before transmission and considers CTS and the addressing bits. The start bit is excluded from CRC calculation.

For flexibility of the *BiSS* system, the addressing scheme of the control communication is dynamically executed at the beginning of each new control data frame on CDS. Therefore, after receiving the start bit, each *BiSS* Slave tries to fetch the next free *BiSS Slave ID* according to its position in the daisy chain.

The control communication supports up to eight different *BiSS Slave ID* at the same time. The *BiSS Slave ID* assignment works similarly to the mechanism of the start bit delay and timeout detection for each *BiSS* cycle: Since there is only one CDS bit per *BiSS* frame, each *BiSS* Slave receives the CDS bit generated by its predecessor. During ID assignment, each bit on CDS represents a different *BiSS Slave ID*. If the CDS bit is set to one, the current *BiSS Slave ID* is already taken and the *BiSS* Slave waits for the next cycle to occupy the next *BiSS Slave ID* if possible.

Each *BiSS* Slave that has already taken a *BiSS Slave ID* shifts the unset CDS bit to its successor. After nine cycles, the ID assignment is complete and the first eight *BiSS* Slaves of the daisy chain can be addressed by their corresponding *BiSS Slave ID*.

The ninth bit indicates to the *BiSS* Master that there are more *BiSS* Slaves connected to the system than can be addressed without further measures. More information about *BiSS Slave ID* assignment for systems with more than eight *BiSS* Slaves can be found in section *BiSS* Commands. Figure 26 shows the *BiSS Slave ID* assignment for the example system from Figure 18 on page 11.

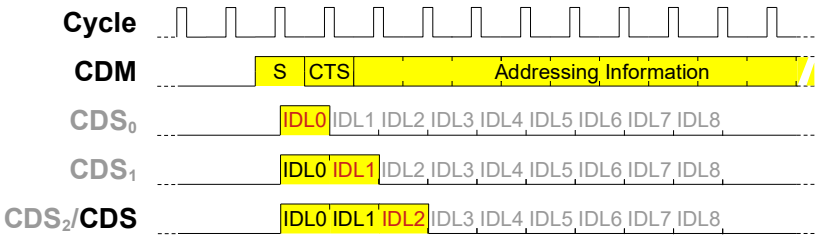


Figure 26: *BiSS Slave ID* assignment for 3 *BiSS* Slaves

After receiving the start bit, Slave 0 sets the lock bit IDL0 to "1" and stores the *BiSS Slave ID* = 0. From that moment on, Slave 0 will answer to any communication to *BiSS Slave ID* = 0 until the next ID assignment. Within the same *BiSS* cycle, Slave 1 and Slave 2 receive IDL0=1 and wait for the next *BiSS* cycle to obtain their *BiSS Slave ID*. In the next *BiSS* cycle, Slave 0 does not occupy CDS so that Slave 1 sets IDL1 to "1" signaling to Slave 2 that *BiSS Slave ID* = 1 is already taken. In the third *BiSS* cycle, IDL2 is ignored by Slave

0 and Slave 1 and *BiSS Slave ID* = 2 gets assigned to Slave 2.

After ID assignment is done, all three *BiSS* Slaves can exclusively be addressed by their corresponding *BiSS Slave ID*. Further addressing mechanisms for the different types of control data frame are explained in the following chapters.

The control data frame can be aborted any time by sending CDM="0" for 14 *BiSS* cycles. Each *BiSS* Slave receiving CDM="0" 14 times in a row must reset to idle and be ready to process a new control data frame.

After reboot, if the *BiSS* system is not configured properly, it is possible to read the necessary data channel configuration from the connected *BiSS* Slaves for devices with an electronic datasheet.

In order to ensure proper sampling of CDM for the control communication the *BiSS* Master should be able to hold the level of the inverted CDM bit until the next *BiSS* cycle (the slave's timeout needs to expire) instead of releasing it to idle. This 'holding' of the $\overline{CDM}$ level at MA is shown in Figure 27. If holding the $\overline{CDM}$ level is activated and CDM = "1", the rising edge on MA is delayed until the beginning of the next *BiSS* frame.

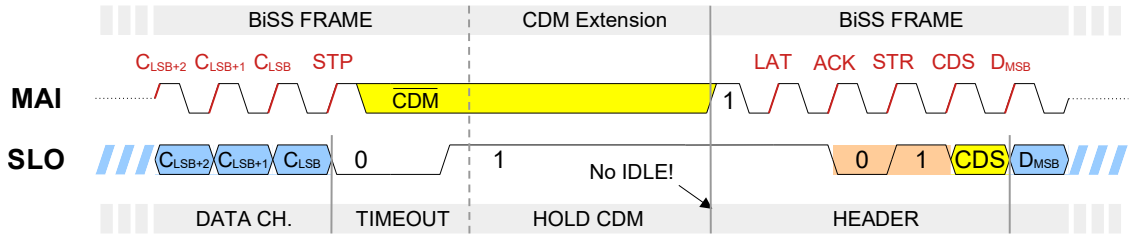


Figure 27: Two consecutive *BiSS* frames with CDM level held constant

Register Communication

As described previously, there are two types of control data frame supported by the *BiSS* protocol. Register communication is used for byte-addressed data access to the memory map of one specific *BiSS* Slave connected to the *BiSS* system. This data access is used for reading and writing memory-mapped resources of the *BiSS* Slave, such as device configuration, calibration data and EDS parameters.

Register communication can be used for single-byte data accesses as well as automatically incremented data access for entire address ranges, if supported by the device. The data accesses are acknowledged by the *BiSS* protocol and can be protected by several register protection levels.

The *BiSS* device can be accessed for additional addressed data exchange without interrupting the process data transmission.

Register Read Access

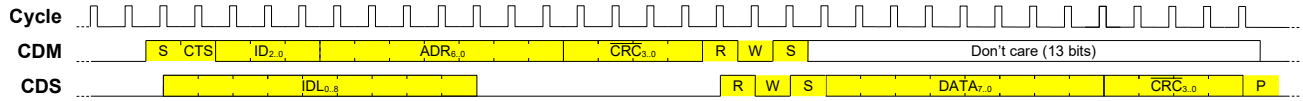


Figure 28: Register communication for single-byte read access

Figure 28 shows a control data frame for reading a single byte from the slave with the corresponding *BiSS Slave ID* at the specified address. The bit CTS=1 following the start bit (S) informs all *BiSS* Slaves about the beginning of a control data frame for register communication. The binary-coded 3-bit address ID specifies the recipient of the frame. Only the *BiSS* Slave that is assigned to the *BiSS Slave ID* identified by the ID completely processes the control data frame. The other *BiSS* Slaves ignore the rest of the control data frame after processing the frame header.

The 7-bit address (ADR) specifies the resource within the selected *BiSS* Slave for the data request. Thus, the *BiSS* protocol supports up to 2^7 = 128 bytes of directly addressable access with a single control data frame. However, it is possible to extend the usable address space by introducing memory banks that can be switched within the *BiSS* Slave. More information about memory banks can be found in chapter *BiSS* MEMORY MAP.

After verification of the 4-bit Cyclic Redundancy Check within the frame header, the control data frame for register communication continues with the read (R) and write (W) bits. These bits are generated by the *BiSS* Master to inform the addressed *BiSS* Slave about the purpose of the request. The read access is initiated by R=1 and W=0.

If a CRC error is detected, the *BiSS* Slave behaves as if it was not addressed and ignores the residual control data frame.

As shown in Figure 28, the bits R and W are directly repeated by the addressed *BiSS* Slave on CDS. This mechanism is used for acknowledging the current access to the *BiSS* Master. Only if the R and W bits are repeated identically on CDS is the data access accepted by the addressed *BiSS* Slave.

If the selected address is not available or the operation is not permitted, the *BiSS* Slave inverts the W bit (1) and aborts further processing of the current control data frame. The *BiSS* Master, when detecting an inverted W bit, also aborts the current control data frame. If the data access is accepted by the *BiSS* Slave, the control data frame continues with the actual data transmission.

The data transmission starts with another start bit (S) generated by the *BiSS* Master that is again repeated on CDS. If the *BiSS* Slave is not ready to send the requested data due to internal processing, the repetition of the start bit (S) can be delayed to inform the *BiSS* Master about the required processing time. An example of the start bit delay is shown in Figure 29. The maximum processing time for register communication is specified in chapter CHARACTERISTICS. If the *BiSS* Slave exceeds the maximum processing time, the *BiSS* Master aborts the control data frame.

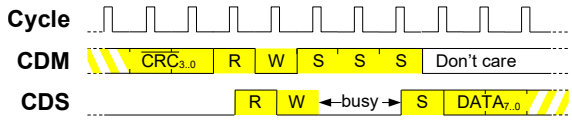


Figure 29: Start bit delay for single-byte read access

Once the *BiSS* Slave repeats the start bit (S), the following 13 CDM bits are not relevant for the current register access. However, it is recommended to send CDM="0" to reset control communication with the 14th CDM="0" at the end of the read access.

For devices that do not support the write bit inversion for invalid read access as described above, the start bit delay can also be used to inform the *BiSS* Master about the failed register communication. However, in order to properly detect the invalid access, the *BiSS* Master must wait the maximum processing time as defined for register communication.

After receiving the repeated start bit (S) from the *BiSS* Slave, the *BiSS* Master expects the requested data byte on CDS followed by another 4-bit Cyclic Redundancy Check (polynomial 0x13) for protection against interference on the transmission line. The CRC is inverted before transmission and considers solely the data byte. The start bit is excluded from CRC calculation. The control data frame for a single byte read access is concluded with the stop bit P=0 that is generated by the *BiSS* Slave.

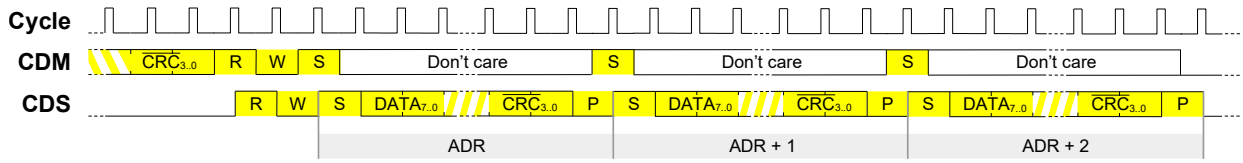


Figure 30: Register communication for sequential read access

After receiving the stop bit (P) the *BiSS* Master can directly start another data transfer by sending another start bit (S) on CDM, if supported by the connected device. Thus, it is possible to transmit multiple data bytes without restarting the entire frame for higher throughput. In this case, the addressed *BiSS* Slave automatically increments the received address and responds with the next data byte. Figure 30 shows an example of a control data frame for reading of 3 consecutive data bytes.

The stop bits (P) at the end of each data transmission use the same acknowledge mechanism as described for the write bit (W). If the following address is accepted for the selected operation the stop bit is P=0, otherwise, the *BiSS* Slave sends P=1 on CDS to inform the *BiSS* Master to terminate the frame. The fourth byte in Figure 30 for example is not available for read access and therefore acknowledged with P=1.

Since for sequential register accesses the R and W bits are not repeated, it is not possible to change the access mode for the current control communication without starting a new control data frame. The start bit delay as explained for single-byte accesses is supported for each byte of a sequential register access as well. The maximum number of bytes for one sequential register access is 64.

Register Write Access

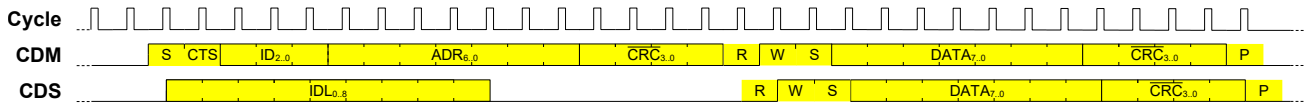


Figure 31: Register communication for single-byte write access

In contrast to the control data frame for register read access, the read and write bits for initiating a register write access are R=0 and W=1 as shown in Figure 31. The frame header is exactly the same.

As before, the *BiSS* Slave acknowledges the availability of the requested resource by repetition of the R and W bits, inverting the W bit (0) if the transmitted address is not accepted. The permission for read and write access to the same address can be different. Detailed information about the register protection levels supported by *BiSS* can be found in chapter *BiSS* MEMORY MAP.

After receiving the start bit (S), the *BiSS* Master transmits the data byte to be written to the selected address. The start bit ahead of the first data byte must not be delayed. The following start bits within a sequential write access may be delayed, if processing time is required. The *BiSS* Master repeats the start bit until it is confirmed by the *BiSS* Slave.

For verification purposes, the *BiSS* Slave repeats the received data information and the 4-bit CRC, that must be verified at the recipient.

To ensure success of the write operation, it is recommended to verify a write access by reading back the data from the corresponding address.

As with register read access, the transmission is concluded with the stop bit P, which is used for access acknowledgement of sequential register write accesses.

Figure 32 shows an example of three consecutive data bytes written to the addressed *BiSS* Slave. The start bit (S) for the second and third data byte is delayed due to the *BiSS* Slave's processing time. As before, the 4th address counting from the transmitted address information is not available for write access (P=1).

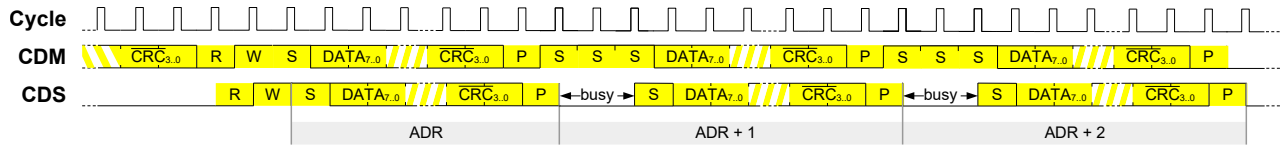


Figure 32: Register communication for sequential write access with start bit delay

BiSS Commands

In order to trigger a specific function for multiple *BiSS* Slaves at the same time, *BiSS* offers a command mechanism built into the protocol itself. In addition to protocol specific commands (explained later on in this section), it is even possible to execute slave-specific commands that are individually implemented in the *BiSS* devices. Following the specification of the control data frame header, a *BiSS Command* frame is defined as shown in Figure 33

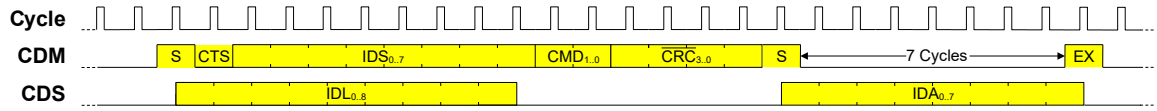


Figure 33: *BiSS Command* frame

The *BiSS Command* frame is initiated by CTS=0 following the start bit (S). In contrast to the *Register Communication Frame*, the *BiSS Command* frame does not need an address to be send to the recipient. Instead, the addressing information of the frame header is used for the linear-select 8-bit ID (IDS) and the 2-bit command (CMD) itself.

Each IDS bit represents one of the automatically assigned *BiSS Slave ID* starting with *BiSS Slave ID* = 0. The linear-select code enables the *BiSS* Master to address multiple *BiSS* Slaves within one control data frame and send one *BiSS* command to several *BiSS* Slaves, if necessary. If no IDS bit is set, the transmitted command is supposed to be executed by all *BiSS* Slaves connected to the system.

The broadcast of the *BiSS Command* frame can be used even for systems with more than eight *BiSS* Slaves. Thus, the *BiSS* Slaves that are not currently assigned to a *BiSS Slave ID* need to process the upcoming *BiSS Command* frame as well. More information about the broadcast of a *BiSS Command* frame can be found below.

The 2-bit CMD defines the command that is supposed to be executed at the selected *BiSS* Slaves. Hence, it is possible to trigger 4 different commands to one or more *BiSS* Slaves connected to the system. Details about the different *BiSS* commands are described below. As before, the header of the *BiSS Command* frame concludes with the 4-bit Cyclic Redundancy Check transmitted on CDM.

After transmitting the control data frame header, the *BiSS* Master continues with another start bit (S) on CDM in order to verify the reception of the command information. Therefore, all *BiSS* Slaves addressed by IDS acknowledge the reception and acceptance of the transmitted command to the *BiSS* Master with an ID acknowledge bit (IDA).

As with the IDL and IDS bits, the IDA bits represent the *BiSS* Slaves in ascending order starting with *BiSS Slave ID* = 0. The *BiSS* Slaves that are processing the *BiSS* command according to the IDS bits of the *BiSS Command* frame header and that are ready for execution, set the corresponding IDA bit to "1". The other *BiSS* Slaves that are assigned to a valid *BiSS Slave ID* send IDA = "0". Thus, the *BiSS* Master is able to check if the requested *BiSS* Slaves accept the transmitted *BiSS* command before triggering the actual execution with the execute bit (EX) on CDM.

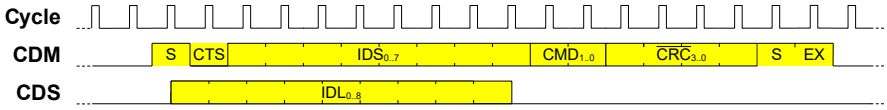


Figure 34: Broadcast *BiSS Command* frame

In contrast to the *BiSS Command* frame addressed to specific *BiSS* Slaves, a broadcast transmission (Figure 34) is initiated with IDS = "0000 0000". The transmitted *BiSS* command is executed by all *BiSS* Slaves connected to the system, even those without a valid *BiSS Slave ID* assigned to them. Therefore, the IDA bits are omitted and the *BiSS* Master directly sends the EX bit following the start bit (S) on CDM.

Since the operation of the predefined *BiSS* commands changes for a broadcast *BiSS Command* frame, in total there are eight possible functions to be triggered by one *BiSS Command* frame. Table 1 shows all *BiSS* commands supported by the control communication.

BiSS Commands are optional. It is recommended to implement *BiSS* Commands "00" and "01" particularly in devices with more than one slave and bus capable devices.

BiSS Protocol Commands	
CMD	Function
Addressed	
00	Activate process data channels of addressed slaves
01	Deactivate control communication of addressed slaves
10	Activate "Standard Operation" for bus coupler (or user defined)
11	User defined
Broadcast (to all slaves) with IDS="0000 0000"	
00	Deactivate process data channels of all slaves
01	Activate control communication of all slaves
10	Activate "Feedback Operation" of bus coupler (or reserved)
11	Reserved

Table 1: *BiSS* Commands

CMD = "00" (Process Data Channel)
With the general start-up procedure in mind, the broadcast command CMD = "00" can be used to deactivate the process data of all *BiSS* Slaves connected to the system. If implemented, the command enables the shortest *BiSS* frame possible that can be used for bidirectional control communication. The *Short BiSS Frame* as described in section Short *BiSS* frame is suitable for fast transfer of register data, for reading electronic datasheet data during initialization or sensor calibration for example.

CMD = "00" addressed to specific *BiSS* Slaves reactivates the process data solely for the *BiSS* Slaves assigned to the corresponding *BiSS Slave ID*. The command can be used to initially set up the *BiSS* frame for the desired application.

CMD = "01" (Control Communication)
CMD = "01" addressed to specific *BiSS* Slaves deactivates the ID assignment for the selected *BiSS* Slaves of the system. This function is essential for systems with more than eight *BiSS* Slaves. For example, the only way to address the ninth *BiSS* Slave of the daisy chain for register communication is to deactivate one of the first eight devices occupying a valid *BiSS Slave ID*.

The broadcast command CMD = "01" reactivates the register communication for all connected *BiSS* Slaves. Thus, if nine *BiSS* Slaves are connected, only the first eight *BiSS* Slaves can be addressed. After start-up every *BiSS* Slave tries to occupy a valid *BiSS Slave ID* during ID assignment of the control data frame (without activation through broadcast command CMD = "01").

CMD = "10" (Bus Coupler/User-Defined)
The broadcast command CMD = "10" is used to activate the "Feedback Operation" mode for all *BiSS* Slaves with implemented bus coupler. For *BiSS* Slaves without bus coupler, broadcast CMD = "10" is deactivated.

The addressed CMD = "10" activates the "Standard Operation" mode for the selected *BiSS* Slaves with implemented bus coupler. If no bus coupler is is implemented, the command can be used for device-specific functions, e.g. sensor presets.

Details about bus couplers are explained in chapter APPLICATION HINTS.

CMD = "11" (User-Defined/Reserved)
The command CMD = "11" addressed to specific *BiSS* Slaves is intended for user-defined functions as well. Multiple user defined commands can be implemented by changing the active command via register communication. The broadcast command CMD = "11" is not used and is reserved for future use.

Short BiSS frame
The Short *BiSS* Frame is a complete frame as described in chapter *BiSS* FRAME with deactivated data channels. Therefore, the *BiSS* frame ends after transmission of the stop bit. Figure 35 shows a Short *BiSS* Frame that is activated with a broadcast command CMD = "00".

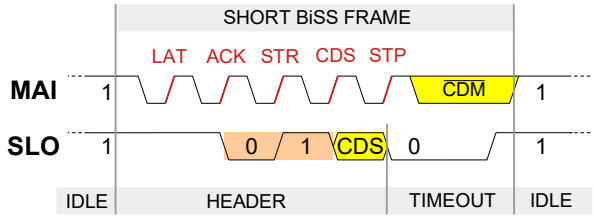


Figure 35: Short *BiSS* Frame

Since the Short *BiSS* Frame transmits the bits CDS and CDM it can be used for bidirectional control communication. The Short *BiSS* Frame supports the Adaptive *BiSS* Timeout as described in chapter *BiSS* FRAME and enables proper automatic timeout detection at the *BiSS* Master. The Short *BiSS* Frame enables the shortest *BiSS* cycle time for the fastest control communication possible.

Reduced BiSS Frame
In contrast to the Short *BiSS* Frame, the Reduced *BiSS* Frame does not transmit the CDS bit for bidirectional register communication. Instead, the frame is concluded after the acknowledge edge on MA even without any response from the *BiSS* Slave. Figure 36 shows a Reduced *BiSS* Frame as described above.

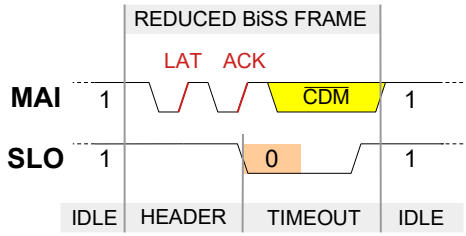


Figure 36: Reduced *BiSS* frame

The Reduced *BiSS* Frame is initiated solely by the *BiSS* Master when it stops toggling MA after the acknowledge bit. Since the CDS bit is not transmitted, the Reduced *BiSS* Frame can only be used for broadcast *BiSS* commands, which do not need any response from the *BiSS* Slaves. Hence, the Reduced *BiSS* Frame is supposed to be used for system initialization when the *BiSS* Master is not properly configured according to the correct parameters of the connected *BiSS* Slaves. In order to enable fast register communication, the Reduced *BiSS* Frame can be used to activate the Short *BiSS* Frame with broadcast command CMD = "00", for example.

BiSS MEMORY MAP

As described in the chapter CONTROL COMMUNICATION register communication can be used to address 128 bytes of each *BiSS* Slave' internal data resources. The upper 64 bytes (addresses 0x40 to 0x7F) are referred to as *Fixed Addresses* and are directly mapped to the most important data resources of the *BiSS* Slave. Some of the *Fixed Addresses* are predefined for necessary information regarding the *BiSS* protocol itself, others are free to be used for device-specific information.
The remainder of the address range (addresses 0x00 to 0x3F) can be used for additional data resources that are organized as memory banks. Therefore, the Fixed Address space includes the bank select byte (BSEL). The bank select byte can be used to switch the data represented at addresses 0x00 to 0x3F for extending the addressable data resources if necessary. Hence, in total it is possible to address 256x64 = 16384 bytes in the Register Bank space. A register bank can contain configuration parameters, an Electronic Data Sheet (EDS) or User Data (e.g. a motor info sheet) for instance. Thus, the access level for each bank is defined by the device manufacturer according to its content. Table 2 provides an overview of the *BiSS* memory map.

Address	Name	RPL	Content
Register Bank			
0x00 ...0x3F	Register 1 ... Register 64	R/W, R, N/A	Register bank with 64 registers (e.g. configuration data, EDS, User Data). The register bank is selected with BSEL. Access level of each register is defined by device manufacturer.
Fixed Addresses			
0x40	BSEL	R/W, N/A	Bank selection, if more than one bank is implemented.
0x41	EDS_BANK	R, N/A	Pointer to EDS bank
0x42 0x43	BP_ID	R	BiSS Profile ID
0x44 ... 0x47	DEV_SN	R, N/A	Device serial number
0x48 ...0x77	Free Registers	R/W, R, N/A	Free registers for project-related use (e.g. status information, a device specific command register, etc.). Access level of each register is defined by device manufacturer.
0x78 ... 0x7D	DEV_ID	R	Device ID
0x7E ... 0x7F	MFR_ID	R	Manufacturer ID
Notes: [1] Parameters with more than one byte are saved as big-endian, i.e. with the highest-value byte at the lowest-value address.			

Table 2: BiSS Memory Map

Register Protection Level

In order to protect the data and configuration parameters, the *BiSS* protocol provides a three-stage Register Protection Level (RPL). The stages are available as shown in Table 3 and can be assigned to any address in the *BiSS* address space as defined above.

Register Protection Levels	
RPL	**Description**
R/W	The corresponding register address can be read from and written to.
R	The corresponding register address can only be read from. Write accesses are rejected.
N/A	The corresponding register address rejects read and write accesses.

Table 3: Register protection levels

Each address from the *BiSS* address range can be fully accessible, read-only or fully protected. It is not possible to differentiate between fully protected and not implemented since the mechanism of indication is the same. The indication of protected register accesses is described in chapter CONTROL COMMUNICATION.

Bank Selection

The bank selection (BSEL) byte is mapped to slave address 0x40. BSEL enables the implementation of up to 256 banks with 64 data bytes each that are addressed at 0x00 to 0x3F. In order to switch the current bank mapped to the accessible address space, a *Register Communication Frame* is used for write access. BSEL can be read to check the currently active bank. If only one bank is implemented, read and write access to BSEL are refused.

BSEL(7:0)	Addr. 0x40; bit 7:0	R/W
Code	Description	
0x00 ...0xFF	Registers of selected bank can be accessed at addresses 0x00...0x3F.	
Notes	If only one bank is implemented, read and write access to BSEL are refused (RPL=N/A).	

Table 4: Bank Selection

EDS Bank

Address 0x41 holds the EDS_BANK pointer. This byte is read-only and can be accessed by the *BiSS* Master to get the bank number of the Electronic Data Sheet (EDS) of the *BiSS* device. The EDS memory space is described in section Electronic Data Sheet. The EDS should be implemented in every *BiSS* device. If it is not possible to implement an EDS (e.g. due to missing memory space), at least a Profile ID must be implemented.

EDS_BANK(7:0) Addr. 0x41; bit 7:0	R
Code	Description
0x00	No EDS implemented.
0x01 ...0xFF	Number of bank which contains the electronic datasheet (EDS).
Notes	In legacy products access to EDS_BANK can also be refused (RPL=N/A) if no EDS is implemented.

Table 5: EDS Bank

Electronic Data Sheet

The Electronic Data Sheet (EDS) can be used to automatically configure the *BiSS* Master according to the connected *BiSS* Slaves.The EDS contains all the characteristics of the slave necessary to establish *BiSS* communication, e.g. data channel configurations, the maximum clock frequency and the minimum *BiSS* cycle time. The EDS definition can be found at www.biss-interface.com.

User Banks (Optional)

The User Banks hold specific information about the system the *BiSS* device is implemented in. These optional banks are typically defined by the user who implements the *BiSS* device into the final system. For example, in the case of a motor feedback application, the User Banks can be used to store a motor info sheet with details about the motor and its specified limits. Therefore, the manufacturer of the final product need to be able to write User Banks and prevent changes to it by increasing the register protection level to read-only later on.

It is highly recommended to implement an EDS **and** a Profile ID. Only if sufficient non-volatile memory is not available to store an EDS, it may be neglected. In that case, at least a valid Profile ID (≠0x0000) must be implemented.

BiSS Profile ID

In order to simplify the compatibility of *BiSS* C devices and to support interoperability of *BiSS* products, there are profile definitions for common applications and frequently required device types. The *BiSS* Profiles define data channel configurations and protection mechanism to be supported by the compatible systems.

The *BiSS* Profile ID (BP_ID) is a 16-bit big endian number that can be read from addresses 0x42 and 0x43. If no *BiSS* Profile is supported, BP_ID equals 0x0000. Since the profile describes a single data channel, a profile ID needs to be implemented in each *BiSS* slave. *BiSS* Profiles are only defined by iC-Haus GmbH or BiSS Association e.V. Details about the BP_ID are available at www.biss-interface.com.

BP_ID(15:8) Addr. 0x42; bit 7:0	R
BP_ID(7:0) Addr. 0x43; bit 7:0	R
Code	Description
0x00 ...0xFF	BiSS Profile ID as defined by iC-Haus GmbH or BiSS Association e.V.
Notes	If no BiSS profile ID is supported, BP_ID=0x0000.

Table 6: BiSS Profile ID

Device Serial Number

To identify each *BiSS* device in the field, it is possible to assign a 32-bit serial number (DEV_SN) to addresses 0x44 to 0x47. It is stored as big-endian. Together with MFR_ID and DEV_ID, the serial number provides a globally unique tag for error tracking and charge identification for example. 0x00000000 and 0xFFFFFFFF are reserved and cannot be assigned as a valid serial number. If a *BiSS* device does not provide a serial number, the value must be set to 0x00000000 or the addresses 0x44 to 0x47 must be indicated as not accessible.

DEV_SN(31:24) Addr. 0x44; bit 7:0	R
DEV_SN(23:16) Addr. 0x45; bit 7:0	R
DEV_SN(15:8) Addr. 0x46; bit 7:0	R
DEV_SN(7:0) Addr. 0x47; bit 7:0	R
Code	Description
0x00 ...0xFF	Device Serial Number as defined by device manufacturer
Notes	If no Device Serial Number is supported, DEV_SN=0x00000000. In legacy products access to DEV_SN might be refused (RPL=N/A), if no serial number is implemented. DEV_SN=0xFFFFFFFF is reserved.

Table 7: Device Serial Number

Free Registers

The address space from address 0x48 to 0x77 contains unused registers. Since they are not predefined for any *BiSS* parameter, they can be used for device-specific information, e.g. status information or device commands. The access level of the registers is defined by the device manufacturer. If implemented, a register must be available independent of BSEL.

Parameters with more than one byte are saved as big-endian, i.e. with the highest-value byte at the lowest-value address.

Manufacturer ID and Device ID

The 16-bit Manufacturer ID at addresses 0x7E and 0x7F identifies the manufacturer of the *BiSS* device. It is unique and assigned by BiSS Association e.V. The Manufacturer ID can be requested at www.biss-interface.com.

MFR_ID(15:8)	Addr. 0x7E; bit 7:0	R
MFR_ID(7:0)	Addr. 0x7F; bit 7:0	R
Code	Description	
0x00 ...0xFF	Unique manufacturer ID as defined by iC-Haus GmbH or BiSS Association e.V.	
Notes	The Manufacturer ID must be implemented. Only the Manufacturer ID assigned by iC-Haus or BiSS Association may be used. Contact BiSS Association for further details.	

Table 8: Manufacturer ID

Together with the Manufacturer ID, the 48-bit Device ID uniquely identifies the product type of a *BiSS* device. The addresses 0x78 to 0x7D can be read by the *BiSS* Master in order to check if a device is responding as expected. It is recommended to assign different Device IDs to devices with different configurations regarding the *BiSS* interface, e.g. different data lengths or timing parameters. Only for uniquely identifiable *BiSS* devices is it possible to implement any automatic system initialization processes.

While EDS and *BiSS* Profile ID are standardized by BiSS Association e.V., the identification procedure using Manufacturer and Device ID requires an XML file for decoding that is provided by the manufacturer.

DEV_ID(47:40)	Addr. 0x78; bit 7:0	R
DEV_ID(39:32)	Addr. 0x79; bit 7:0	R
DEV_ID(31:24)	Addr. 0x7A; bit 7:0	R
DEV_ID(23:16)	Addr. 0x7B; bit 7:0	R
DEV_ID(15:8)	Addr. 0x7C; bit 7:0	R
DEV_ID(7:0)	Addr. 0x7D; bit 7:0	R
Code	Description	
0x00 ...0xFF	Device ID as defined by the device manufacturer.	
Notes	If no Device ID is supported, DEV_ID=0x000000000000. In legacy products access to DEV_ID might be refused (RPL=N/A) if no Device ID is implemented.	

Table 9: Device ID

BUS CONNECTION

In contrast to the point-to-point connection described in chapter POINT-TO-POINT CONNECTION on page 4, a system using bus connection is capable of daisy chaining multiple *BiSS* devices with sensor and actuator slaves included. The third *BiSS* signal line for data input MO is used to transmit actuator data from the *BiSS* Master to the *BiSS* Slaves and/or to delay the start bit according to the maximum processing time of the connected *BiSS* Slaves. Figure 37 shows an example of a simple *BiSS* system using bus connection for an actuator device with a single *BiSS* Slave.

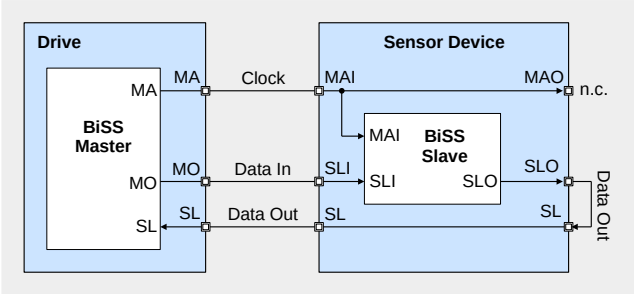


Figure 37: Bus connection with a single actuator slave

In general, the *BiSS* slave operation is the same for both point-to-point and bus connections. Only the signal SLI of Slave 0 is not connected to ground. This delegates the control mechanism as described for the point-to-point connection to the *BiSS* Master itself. Thus, the start bit generation and timeout detection is initiated on MO.

The data to be processed by the actuator device is transmitted to SLI. It is then clocked through and is accessible by the *BiSS* Slave. The data output SLO is externally fed back to the *BiSS* Master by SL. For actuator devices, the *BiSS* Slave returns the process data of the last *BiSS* cycle on SLO.

The pin MAO outputs the *BiSS* clock, which is necessary for connecting multiple devices as described below. The clock forwarding is the reason for the different signal naming as discussed in chapter POINT-TO-POINT CONNECTION.

Figure 38 shows the corresponding *BiSS* frame of the system as discussed above. Since SLI is not connected to ground at the beginning of the *BiSS* frame, the *BiSS* Slave does not generate the start bit. Instead, the *BiSS* Slave waits for the start bit to be transmitted via MO following the acknowledge bit, before outputting new process data.

Since the *BiSS* Master behaves like the Slave 0, it is necessary to transmit a valid CDS bit for further processing by the *BiSS* Slave. Therefore, the next bit is always CDS = "0" as shown in Figure 38. CDS = "0" has no effect on the control communication or automatic ID assignment as described in chapter CONTROL COMMUNICATION.

The last bit of the Header is the stop bit "0" on MO. The stop bit is necessary for timeout detection at the end of the *BiSS* frame. Since each *BiSS* Slave outputs the data sampled on SLI after transmitting its own process data, the stop bit ensures a rising edge on SLO at timeout to be detected by the *BiSS* Master. For point-to-point connections, the stop bit is naturally generated by connecting SLI of Slave 0 to ground.

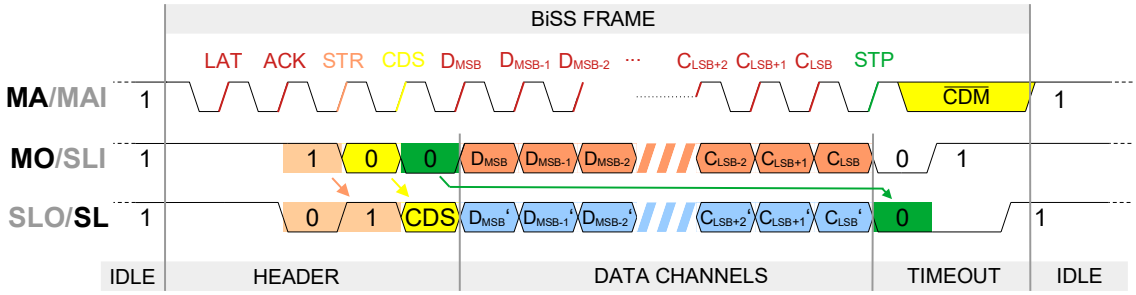


Figure 38: *BiSS* frame of single actuator bus system

The Data Channels in bus connection perfectly illustrate the shift register mechanism for any *BiSS* device. The actuator data on MO (D0-C15) are shifted into the register, while the process data as received during the last *BiSS* cycle (D0'-C15') are shifted out of it. This is, for actuator slaves, the process data are exchanged during *BiSS* clocks on MAI. The configuration of the actuator data channels equals the configuration

of the sensor data channels in point-to-point connections with optional Cyclic Redundancy Check protection as described in chapter PROCESS DATA COMMUNICATION.

The *BiSS* frame concludes with the stop bit on MO followed by a rising edge during transmission of the CDM bit as described before. Overall, the *BiSS* frame on SLO does not differ for bus connection compared to the point-to-point connection, neither does the behavior of the connected *BiSS* Slaves.

In contrast to the point-to-point connection, the bus connection can be used for a daisy chain of multiple *BiSS* Slaves even distributed within multiple devices. Each *BiSS* Slave can either be a sensor or actuator slave, which enables complete control signal processing in motor feedback applications using the same protocol and transmission line.

Figure 39 shows an example bus with two *BiSS* devices. *BiSS* device 1 contains two *BiSS* Slaves and *BiSS* device 2 contains one *BiSS* Slave. Although, the single *BiSS* Slaves behave absolutely the same, there are some effects that need to be considered when implementing systems with multiple devices in bus connections.

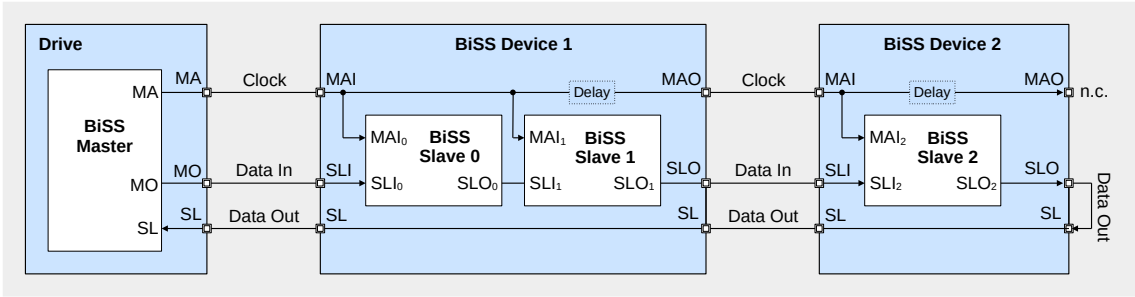


Figure 39: Daisy chain in bus connection

First of all it is important to consider the line delay between daisy-chained *BiSS* Slaves. The line delay compensation as described in section Line Delay on page 8 is a feature that is implemented for signal transmission back to the *BiSS* Master only. Signals from *BiSS* device 1 to 2 for example are not compensated for any delay. Since MA is transmitted through *BiSS* device 1, it is important to match the delay of the *BiSS* clock to the delay of the data signal on SLO. If clock and data signal at *BiSS* device 2 are not synchronous, the daisy chain might not work properly. The maximum amount of delay between MAO and SLO is defined in chapter CHARACTERISTICS.

The ID assignment for the *BiSS* Slaves within the different devices works as described before. The counting of the *BiSS Slave ID* does not consider the total number of devices but only the *BiSS* Slave modules inside the devices. Overall, control communication also works for bus connection as described in chapter CONTROL COMMUNICATION. Figure 40 shows the corresponding *BiSS* frame for the example system as introduced in Figure 39.

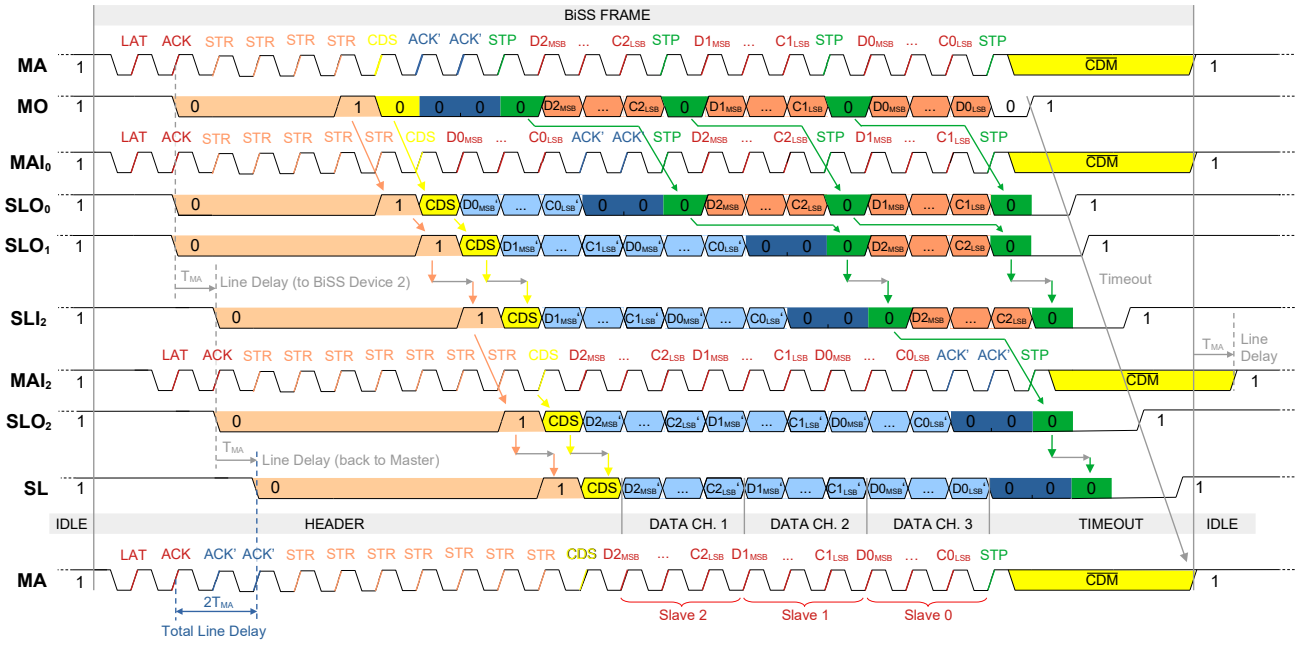


Figure 40: *BiSS* frame for daisy chain in bus connection with processing time

As usual, the *BiSS* frame starts with MA. All sensor process data are sampled at LAT as described before. It is best practice to enable actuator data at LAT as well to achieve deterministic timings for jitter-free control loop applications.

After the acknowledge bit, the start bit is generated on MO as described above. In contrast to the point-to-point connection, the start bit is not generated by Slave 0 as described in section Processing Time on page 10. Instead, the *BiSS* Master is configured to delay the start bit on MO in accordance to the maximum processing time of the slowest *BiSS* Slave in the system. This way, each *BiSS* Slave shifts the start bit through the daisy chain without any further delay as shown in Figure 40.

After processing the CDS bit, the process data is exchanged the same way as described for the single data channel in Figure 38. Each actuator data channel on MO is concluded by a stop bit that needs to be clocked into the shift registers for proper timeout detection by every *BiSS* Slave. The stop bits are output on SLO_x by every *BiSS* Slave at the end of the frame. Due to the shift register mechanism, no stop bits are output in between the actuator data channels transmitted at SLO_2 to the *BiSS* Master.

Since there is line delay between the *BiSS* devices, the signals input at *BiSS* device 2 are delayed in comparison to the output signals at *BiSS* device 1. As mentioned before, it is important to match the timing of the signal MAI_2 and SLI_2 in order to correctly process the received frame data. Therefore, it might be necessary to implement additional delay to the clock between MAI and MAO within *BiSS* device 1. However, it is best practice to use the same bus drivers for the clock and data signal for each *BiSS* device in any case. The total line delay at SLO_2 is automatically compensated by the *BiSS* Master as described in chapter *BiSS* FRAME in section Line Delay.

APPLICATION HINTS

Initialization Example

The following sequence provides an initialization example implemented in a *BiSS* Master for point-to-point connection with a single *BiSS* Slave. Note that the BiSS device's individual start-up time is not considered.

1. **Set clock frequency:**
   250 kHz $\leq$ f_{MA} $\leq$ 1,000 kHz
   (according to CHARACTERISTICS No.2).
2. **Set cycle time: t_{Cycle} = 250 µs**
   (according to CHARACTERISTICS No.12).
3. **Disable all data channels** in the *BiSS* Master configuration.
4. **Enable Hold-CDM** (as shown in Figure 27). This ensures proper control communication until the data channel configuration is known.
5. **Wait 40 µs** before initiating the first *BiSS* frame. This ensures that the *BiSS* timeout has expired and all *BiSS* Slaves are ready for communication.
6. **Send INIT sequence** (as shown in Figure 22). To ensure that the data output signals SLO of all *BiSS* Slaves are "high" level, an initialization sequence (a frame with at least two low pulses at MA) is transmitted.
7. **Wait 40 µs** to ensure the *BiSS* Slave's timeout has expired.
8. **Read the BiSS Slave's (slave ID = 0) Electronic Datasheet (EDS).** If an EDS is not available, continue with step 9.
   - **Read BiSS Slave's register address 0x41.** This register points to the register bank which contains the Electronic Datasheet (EDS).
   - **Write content of address 0x41 into 0x40** to select the EDS register bank.
   - **Read the EDS at addresses 0x00 ... 0x3F.** Note, the Content depends on the EDS version.
   - **Configure the BiSS Master according to the EDS** (e.g. data length, CRC, clock frequency, cycle time).
9. **Read the BiSS Slave's (slave ID = 0) BiSS Profile ID,** if an EDS is not available.
   - **Read BiSS register addresses 0x42 and 0x43.** These registers contain the *BiSS* Profile ID.
   - **Configure the BiSS Master according to the BiSS Profile** (e.g. data length, CRC). Note, timing parameters (e.g. clock frequency, cycle time) are not defined by the *BiSS* Profile and need to be configured manually.

Calculation of BiSS Cycle Time

The minimum possible cycle time T_{Cycle_min} can be calculated by the *BiSS* master considering

- The slave's characteristics
  - Processing time (t_{busy}, rounded up)
  - Additional start bit delay (busy_s, in clocks)
  - Data length of each slave (DLENx)
  - CRC length of each slave (CRCLENx)
  - Timeout (t_{TO}, rounded up)
  - Minimum possible clock period at MAI (T_{MAI_min})
- The transmission line's characteristic ($t_{LineDelay}$)
- The protocol's characteristic (Additional header bits)
  - The first clock period ($t_{FirstClock}$)
  - CDS bit (t_{CDS})
  - One start bit shift for each slave ($t_{StartBitShift}$)
- The currently applied clock period at MA (T_{MA}).

The slave's characteristic can be read from its electronic datasheet. The clock period applied by the master must not underrun the minimum possible clock period of the slave ($T_{MA} \geq T_{MA\_min}$). The line delay can be typically measured by the *BiSS* master.

> The calculated minimum cycle time must not be underrun. Furthermore, the slave may define an absolute minimum cycle time due to its implementation. Neither the calculated nor the absolute minimum cycle time may be underrun.

General Calculation for n Slaves:

$$T_{Cycle\_min} = \underbrace{T_{MA}}_{\text{First Clock}} + \underbrace{t_{LineDelay}}_{\text{Line Delay}} + \underbrace{t_{busy\_max} + T_{MA}}_{\text{Processing Time}} + \underbrace{busy\_s\_max}_{\text{Additional Start Bit Delay}} + \underbrace{T_{MA}}_{\text{CDS}} + \underbrace{T_{MA} * \sum_{x=1}^{n}[1 + DLENx + CRCLENx]}_{\text{Data Channels (incl.Start Bit Shift)}} + \underbrace{t_{TO}}_{\text{Timeout}} + T_{MA}$$

$$= 4T_{MA} + t_{LineDelay} + t_{busy\_max} + busy\_s\_max + T_{MA} * \sum_{x=1}^{n}[1 + DLENx + CRCLENx] + t_{TO}$$

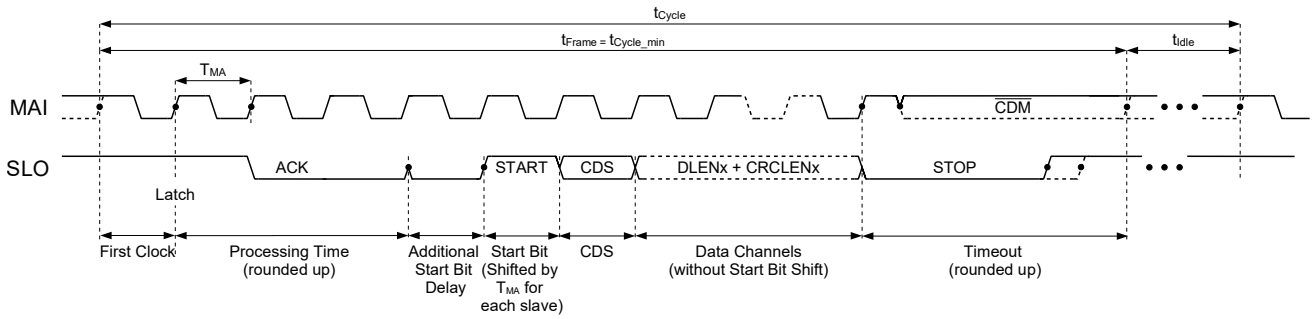


Figure 41: *BiSS* frame composition for calculation of minimum cycle time (line delay excluded)

Simplified Calculation for one Slave:

$$T_{Cycle\_min} = T_{MA} * (5 + DLEN + CRCLEN) + t_{LineDelay} + t_{busy\_max} + busy\_s\_max + t_{TO}$$

Bus Coupler

A bus coupler is a feature of a *BiSS* Slave that enables direct feedback of the device's SLO line to the SL line. At the same time the data output of the following *BiSS* devices is disconnected from the *BiSS* Master. Therefore, the *BiSS* Slave implements a switch to operate in two modes as shown in Figure 42.

- **Standard Operation:** SLO_x is connected to SLI_{x+1} and the SL input is connected to the SL output.
- **Feedback Operation:** SLO_x is connected to the SL output to bypass the following devices.

A bus coupler is suitable for hardware diagnosis, e.g. to detect short circuits or open circuits in the bus connection. Using broadcast command CMD = "10", all bus couplers are switched to "Feedback Operation" (*BiSS* Slaves without bus coupler ignore this command). Then, one bus coupler after the other is switched to "Standard Operation" beginning with the first slave (SLO_0) using the addressed command CMD = "10". By checking the response of each instance a faulty device can be detected and a service induced.

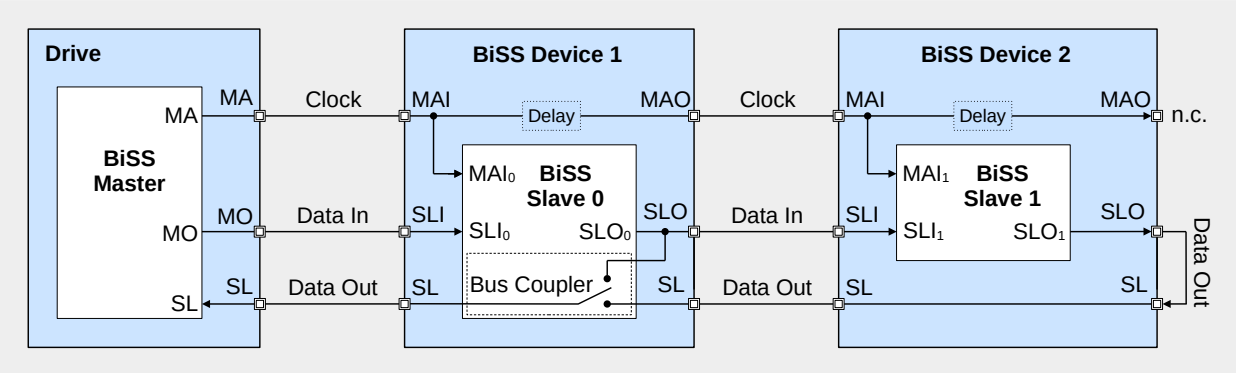


Figure 42: Bus Coupler in Device 1

CHARACTERISTICS

No.	Symbol	Parameter	Condition	Min.	Max.	Unit
01	$1/T_{MA}$	Permissible clock frequency		80	10000 [1]	kHz
02	$1/T_{MA}$ [2]	Required clock frequency	Max. cable length of 100m	250	1000	kHz
03	t_{MAI_lo}	Clock signal low level duration	MAI = "0" at receiver	40		$\%T_{MA\_min}$ [1]
04	t_{MAI_hi}	Clock signal high level duration	MAI = "1" at receiver	40		$\%T_{MA\_min}$ [1]
05	t_{TO}	Static *BiSS* timeout		12.5	40	µs
06	t_{TOA}	Adaptive *BiSS* timeout	For slaves with automatic *BiSS* timeout adaption on T_{MA}	t_{TOA_ref}	$t_{TOA\_ref} + 3 \cdot T_{CLK}$ [3]	
07	$t_{LineDelay}$	Line delay MA → SL	Measured from second rising edge at MA to first falling edge at SL at receiver	0	40	µs
08	$t_{LineJitter}$	Line delay jitter MA → SL	Permissible deviation from $t_{LineDelay}$ at receiver	-25	25	$\%T_{MA}$
09	t_{Lag_SLO}	Lag SLO	For bus configurations, referenced to rising edge of MAO	-50	50	$\%t_{MAlo}$ $\%t_{MAhi}$
10a	t_{busy}	Processing time		$2T_{MA}$	40µs	
10b	busy_s	Additional start bit delay (in clocks)	t_{busy}+busy_s ≤ 40µs	0	8	T_{MA}
11	t_{busy_r}	Processing time for register access		0	20	ms
12	t_{Cycle_min}	Minimum cycle time	Supported by all slaves		250	µs

Notes:
[1] The maximum clock frequency depends on the transmission medium and on the individual devices. The corresponding minimum clock period T_{MAI_min} is usually stored in the EDS.
[2] The required clock frequency has to be supported by any BiSS slave. A BiSS master can use it for initial operation to read the slave's EDS.
[3] $1/T_{CLK}$ is the *BiSS* slave's minimum sampling frequency. t_{TOA_ref} is measured from first falling to second rising edge at MAI.

Table 10: Table of characteristic

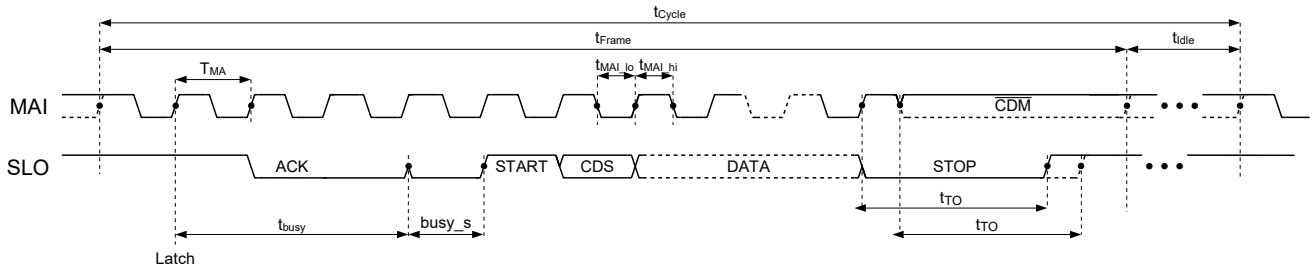


Figure 43: *BiSS* slave timings for point-to-point configuration

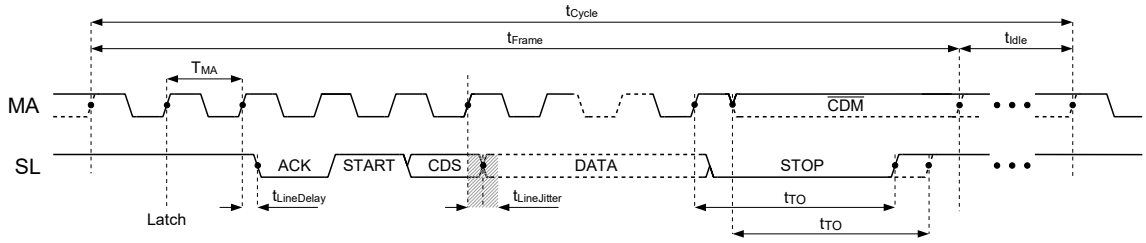


Figure 44: *BiSS* master timings for point-to-point configuration

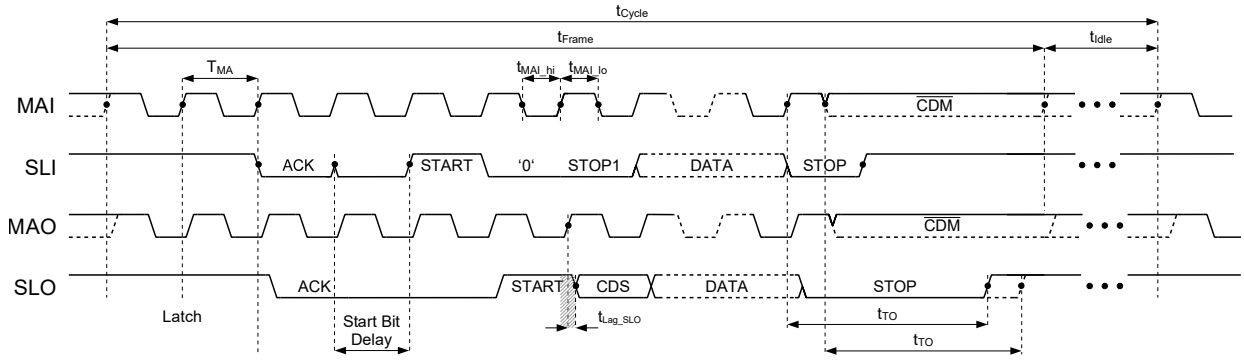


Figure 45: *BiSS* slave timings for bus configuration

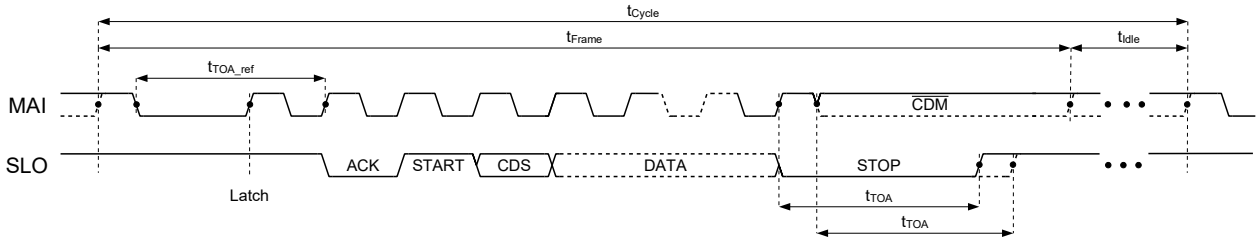


Figure 46: Adaptive *BiSS* timeout

LIST OF ACRONYMS

ACK	Acknowledge bit sent by BiSS slave to confirm the beginning of a new BiSS cycle
ADR	Register address for register access sent by BiSS master
BiSS	Bidirectional/Serial/Synchronous
CDM	Control data bit sent by BiSS master ($\overline{CDM}$ is the inverted control data bit)
CDS	Control data bit sent by BiSS slave
CMD	BiSS command sent by BiSS master
CRC	Cyclic redundancy check ($\overline{CRC}$ is the inverted CRC)
CTS	Control select bit for control communication sent by BiSS master
EX	Execute bit for BiSS commands sent by BiSS master
ID	Slave ID for register access sent by BiSS master
IDA	ID acknowledge bit for BiSS commands sent by BiSS slave
IDL	ID lock bit for control communication sent by BiSS slave
IDS	Slave ID for BiSS commands sent by BiSS master
LAT	"Latch point" that indicates BiSS sensor to capture its current data
MA	Clock output pin at master (connected to MAI of slave)
MAI	Clock input pin at slave (connected to MA of master)
MO	Data output pin at master (connected to SLI of slave)
P	Stop bit for control communication sent by BiSS slave
R	Read access bit for register access sent by BiSS master/slave
S	Start bit for control communication sent by BiSS master
SL	Data input pin at master (connected to SLO of slave)
SLI	Data input pin at slave (connected to MO of master)
SLO	Data output pin at slave (connected to SL of master)
SSI	Synchronous Serial Interface
STP	Stop bit sent by BiSS slave to indicate end of BiSS frame
STR	Start bit sent by BiSS slave to indicate that sensor data is ready for transmission
TO	Timeout of BiSS slave
W	Write access bit for register access sent by BiSS master/slave

REVISION HISTORY

Rel.	Rel. Date*	Chapter	Modification	Page
A1	2007-07-09		Initial release	all

Rel.	Rel. Date*	Chapter	Modification	Page
C1 ...C5	2007 ...2016		See corresponding data sheet for modifications	

Rel.	Rel. Date*	Chapter	Modification	Page
C6	2016-02-26		Drive controllers, Rotary encoder, Linear encoder and Communication watchdogs added to applications	1
		OPERATING DESCRIPTION	Figure 3 updated: T_{busy} starts at first rising edge of MA(latch), not first falling edge of SLO	4
		CONTROL COMMUNICATION	Figure 10 reference in text updated	9
		CONTROL COMMUNICATION	Details on the reduced BiSS frame added	10
		CONTROL COMMUNICATION	BiSS Commands 0b00 and 0b01 added	11
		CONTROL COMMUNICATION	Details on applications with more than 8 IDs added	11
		CONTROL COMMUNICATION	Details on Register Protection added	13
		TERMS AND ABBREVIATIONS	Abbreviation table updated	20
		REVISION HISTORY	Chapter added	21
			Minor text corrections	5, 8, 11, 13, 14, 15, 17

Rel.	Rel. Date*	Chapter	Modification	Page
D1	2023-08-08	All	Overall update, improved descriptions and figures in all chapters.	All
		BiSS MEMORY MAP	Improved descriptions to clarify mandatory BiSS registers and contents.	20
		APPLICATION HINTS	Added application hints	26
		CHARACTERISTICS	Improved parameter description and condition. No.01 Updated to "permissible clock frequency", adjusted footnote No.02 Updated to "required clock frequency", adjusted minimum value (80 kHz → 250 kHz) and footnote No.03 Updated to minimum "clock signal low level duration" of 40 ns No.04 Updated to minimum "clock signal high level duration" of 40 ns No.05 Updated symbol and parameter No.06 Removed characteristic for reduced timeout No.07 Updated symbol and simplified formula and added footnote No.08 Improved parameter description and condition No.09 Added characteristics for lag between SLO and MAO signals No.10 Updated to No.10a/10b characteristics No.12 Added "minimum cycle time" Added timing diagrams	29
		LIST OF ACRONYMS	Added list of acronyms	31

Rel.	Rel. Date*	Chapter	Modification	Page
D2	2023-11-08	PROCESS DATA COMMUNICATION	Minor clarifications and updated Figure 21 (it showed Figure 19 a second time).	11
		CONTROL COMMUNICATION	Register Read Access: added info for slave behavior in case of CRC error in header. Updated Figures 28, 29 and 30: CDM after "S" is "Don't Care". Added info below Figure 29. Renamed "Single-Cycle Data" → "Process Data". Replaced Figure 32 by picture with processing time and updated description.	13ff
		CHARACTERISTICS	Line delay jitter (No.8) updated to 25% (before 20%).	29
		LIST OF ACRONYMS	Typo correction in CRC acronym.	31

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