Open System Protocol

Application Note

Published by ams-OSRAM AG
Tobelbader Strasse 30,
8141 Premstaetten Austria
Phone +43 3136 500-0
ams-osram.com

Open System Protocol

Application Note No. AN162

Valid for:
OSIRE® E3731i

Abstract

This document defines the basic communication protocol for single primary ("controller") - multiple secondary ("responder") node topologies with focus on, but not limited to, automotive interior ambient lighting applications. The protocol description includes all basic characteristics of an electrical interface including the physical layer, the data link layer, and the network layer.

Note: Any application and/or device specific commands and registers are not part of this document but are provided elsewhere. For example, the application of the OSP for an LED requires LED-relevant commands and registers, such as PWM and current settings. These are part of a device specific document, e.g., datasheet or application note.

Table of contents

1 Basic information

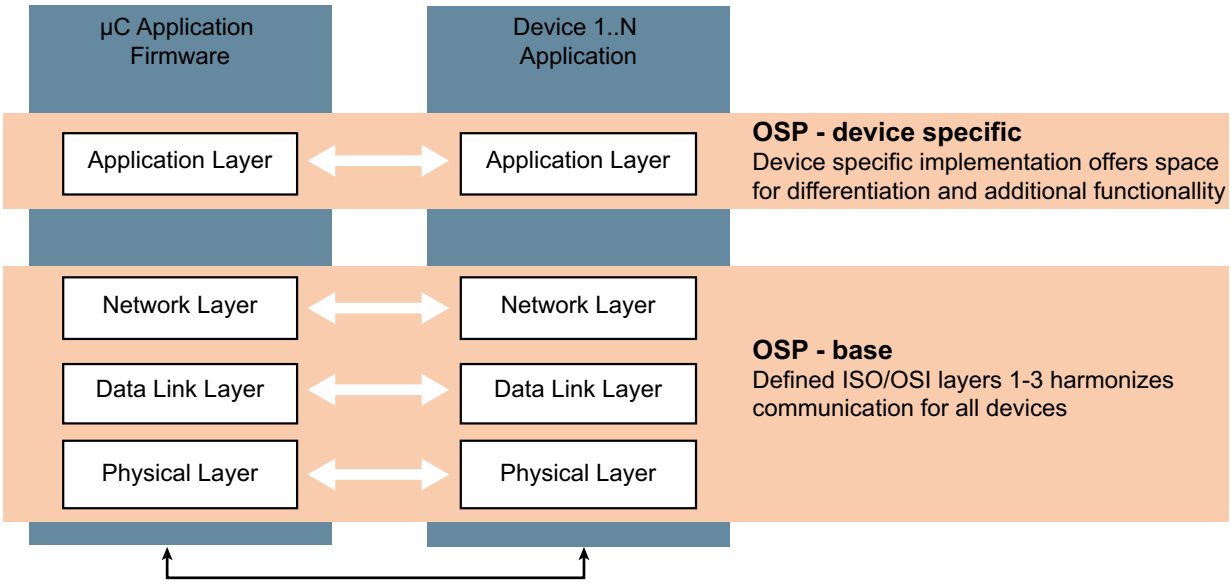
ams OSRAM created an open bus protocol, called "Open System Protocol" (OSP), to support the integration of its intelligent light-sources on customer side based on the following key arguments:

1. An open protocol is essential for a software-defined lighting application
2. The application software (in the µC) includes the knowledge about the application and the µC must be programmed specifically for the application.
3. Full transparency on each of the protocol layers (see Figure 1) is necessary to:
   - allow free choice of the primary node (MUC, left side Figure 1) with performance adjusted to the individual needs of the application
   - allow anybody to build a compatible secondary device (device 1.. N, right side Figure 1)
   - to improve the application using own solutions

The OSP is covering those protocol layers in two steps:

- A base protocol, which needs to be the same for all devices to ensure that communication along the chain is working
- A device-specific application layer, which includes the specific device features

Figure 1: Layer architecture



The first layer, the physical layer, is responsible for the actual transmission over a physical medium.

The second layer, the data link layer, describes the message frame format and how messages are encoded.

The third layer, the network layer, provides commands and conventions for interpreting the data.

And the fourth layer, the application layer, covers all device-specific commands tailored at the specific features of individual nodes.

1.1 Glossary and abbreviations

Table 1 shows the abbreviations for this documentation.

Table 1: Abbreviations

Term	Description
ACK / NACK	Acknowledged / not acknowledged
CAN (FD)	Controller area network (flexible data-rate)
CRC	Cyclic redundancy check
ECU	Electronic control unit, e.g., microcontroller
EMC	Electromagnetic compatibility
EMI	Electromagnetic interference
EOL	End of line
LED	Light emitting diode

Table 1: Abbreviations

Term	Description
LSB	Least significant bit; also used as unit for the quantization step size
LVDS	Low-voltage differential signaling
ME	Manchester encoded data
MCU	Microcontroller (unit)
MSB	Most significant bit
NRZ	Non return to zero
OSP	Open system protocol
PCB	Printed circuit board
POR	Power-on reset
PSI	Payload size indicator
PWM	Pulse-width modulation
RGB	Red, green, blue (also used to refer to a module with three colored LEDs)
RX	Receiving
SE	Single-ended (signaling, one wire carries the signal with GND as reference)
SPI	Serial peripheral interface
TX	Transmitting
USE	Unidirectional single-ended
V_{pp}	Volt peak-to-peak, used for differential voltages

Table 2 shows the glossary for this documentation and explains the terms with a description.

Table 2: Glossary

Term	Description
Node / Device	A member of the OSP network
Primary Node	A unique node in the network (usually an MCU) that initiates communication and may be connected to a backbone network (e.g., LIN, CAN or Ethernet)
Secondary Node	A node in the network that serves the communication requests of the primary node, e.g., an RGB LED with a driver or a sensor
Device specific	Unique protocol implementation in hardware
Message / Telegram	A packet of information exchanged between nodes as defined in chapter "Message frame format"
Downstream	Direction of propagation from primary to secondary node
Upstream	Direction of propagation from secondary to primary node
0x”value”	Number in hexadecimal format, e.g., 0xff = 255
0b”value”	Number in binary format, e.g., 0b101 = 5

1.2 Conformance levels

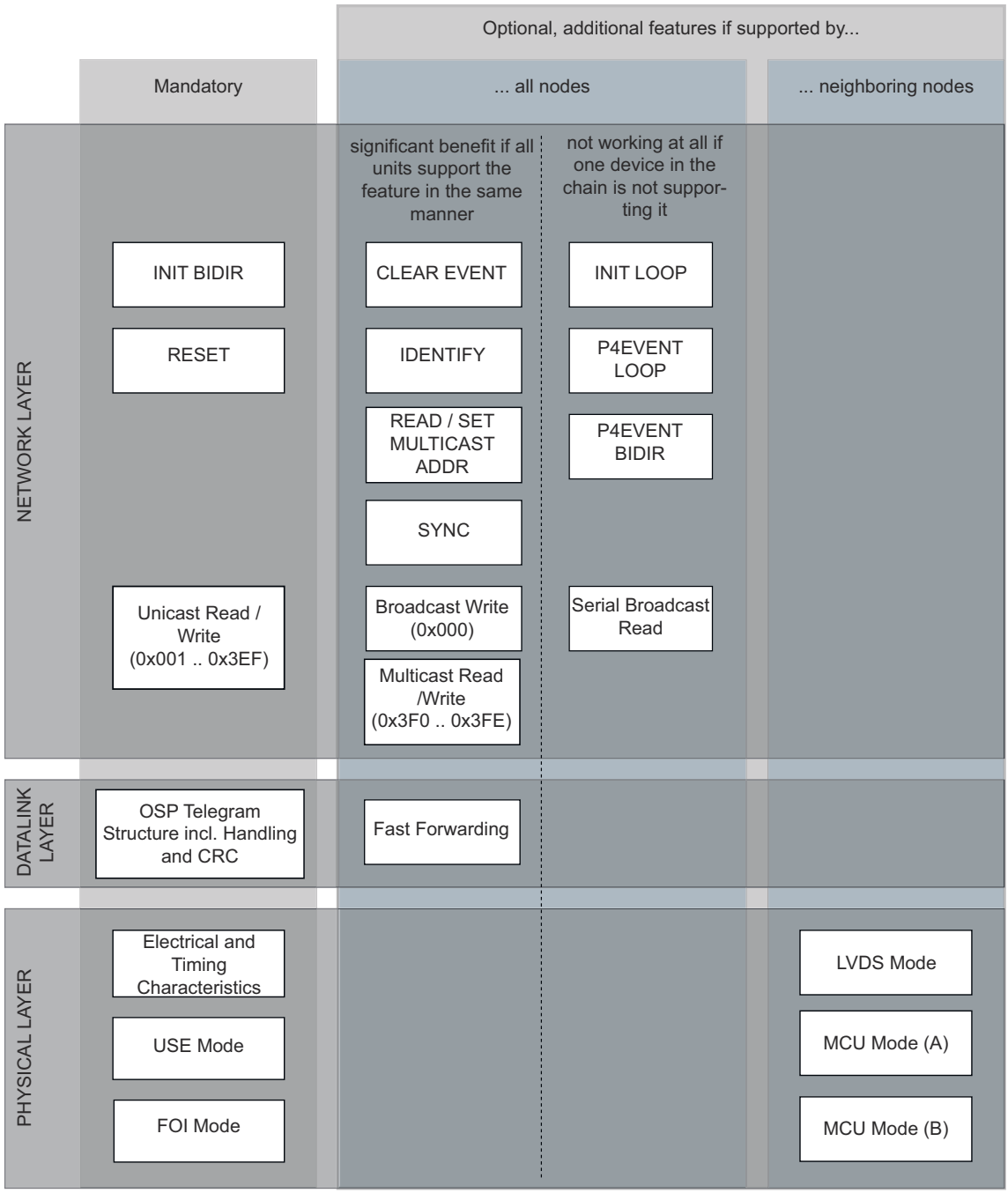
The nomenclature shown in Table 3 will be used throughout this document to illustrate the different levels of conformance with the OSP specification.

Table 3: Conformance level nomenclature

Term	Description
Expected / Assumed	A key word used to describe the behavior of the hardware or software in the design models assumed by this specification. Other hardware and software design models may also be implemented. Expected behavior is the consequence of implementing requirements, it is NOT a requirement in itself.
May / Optional / Device Specific	Key words that indicate flexibility of choice with no implied preference.
Shall	A key word that indicates a mandatory requirement. Designers are required to implement all such mandatory requirements.
Should / Recommended	A key word that indicates flexibility of choice with a strongly preferred alternative

Some of the features described in this specification are mandatory, some are recommended but optional, and some are free to be implemented on a device-specific level. Figure 2 gives an overview of the different conformance levels of all features and indicates what to expect in a mixed chain situation.

Figure 2: Overview of mandatory and optional features



Mandatory

The listed features and characteristics are mandatory for every node in the chain, i.e., the primary and all secondary nodes.

If a device does not support these, the chain is broken and cannot be operated, i.e., no communication over the chain is possible.

Additional features if supported by all nodes

The listed features provide a significant benefit if all nodes in the chain support it, i.e., the primary and all secondary nodes.

They can be assigned to two different categories:

1. The function works even if not all units support it but the benefit for the customer is strongly reduced. In this case, only nodes supporting the feature will react. Examples include the SYNC and IDENTIFY features and the multicast read / write access.

2. The function does not work at all if not all units support it. Examples include the serial broadcast readout or the loopback communication. If one of these functions is to be realized in the chain, the system integrator / customer needs to ensure that all selected devices support this function in exactly the same manner.

Additional features if supported by neighboring nodes

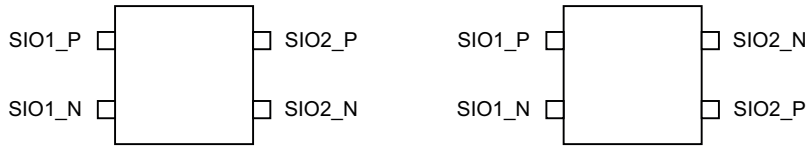
The listed features provide additional functionality and benefits if they are supported by at least two neighboring nodes in the chain, i.e., the primary and the first secondary node or two neighboring secondary nodes.

Examples include the different MCU modes and the LVDS mode. The other nodes in the chain are not affected if they do not support this feature.

2 Physical layer

Each node provides two identical I/O ports with two pins each. In the following, these will be called SIOx_P and SIOx_N where x = 1, 2. Figure 3 shows two possible implementations with symmetric alignment (left, default) and crossed alignment (right). The crossed configuration is beneficial when the inter-node connectivity is realized via the USE-mode. The symmetric configuration is recommended when the inter-node connectivity is realized via LVDS. Refer to chapter "2.2.1 Mode selection" for details on these modes.

Figure 3: Symmetric and crossed alignment



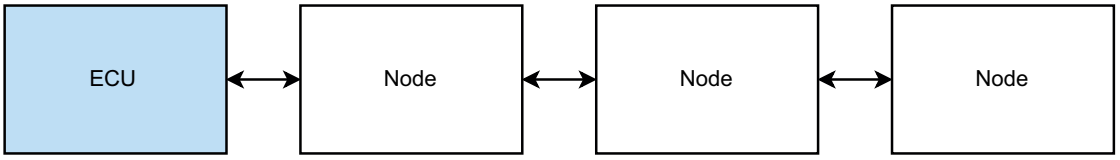
2.1 Supported network topologies

There are two supported network topologies:

- The daisy chain bidirectional
- The daisy chain loop-back network

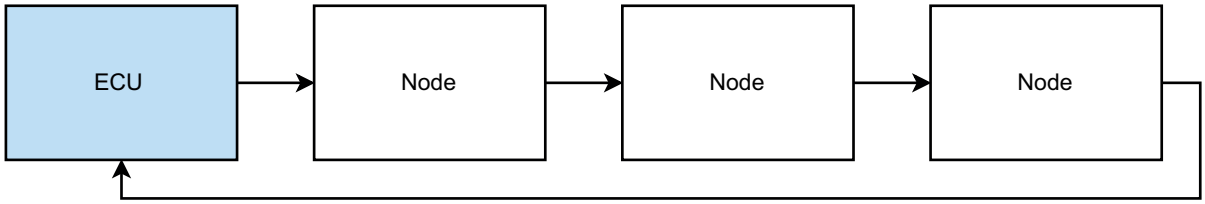
In the daisy chain bidirectional network, one network node is connected to the next in a chain. The communication runs through the chain once and then returns the answer along the same path. The first and last node are not directly connected (see Figure 4).

Figure 4: Daisy chain bidirectional network



In the daisy chain loop back network, both, the first and the last secondary nodes are connected to the primary node. The answers are forwarded and will return with this loop-back (see Figure 5).

Figure 5: Daisy chain loop-back network



Each I/O interface only can operate in semi-duplex mode, i.e., it can either receive or transmit a telegram at the same time. It is possible that a device does receive a telegram at one I/O interface and start transmitting at the other I/O interface (fast-forwarding).

Refer to chapter "6 Examples" in the appendix for more examples on supported chain configurations.

2.2 Physical layer types and communication modes

Each of the two I/O interfaces independently supports different communication modes, that are selected through pull-up and pull-down resistors to GND and VDD, respectively.

2.2.1 Mode selection

Figure 6 and Table 4 show the possible resistor arrangements and resulting communication modes. Communication modes shall be selected directly after POR or reset either through a pin voltage measurement or via an impedance measurement on the pins. The mode selection process shall be done within the mode selection delay time (see chapter "2.8 Parameters"). This voltage is set through pull-up and pull-down resistor combinations as indicated in the table below A typical resistance value of 10 kΩ is recommended. In the following the three different modes will be explained.

Figure 6: Possible resistor arrangements

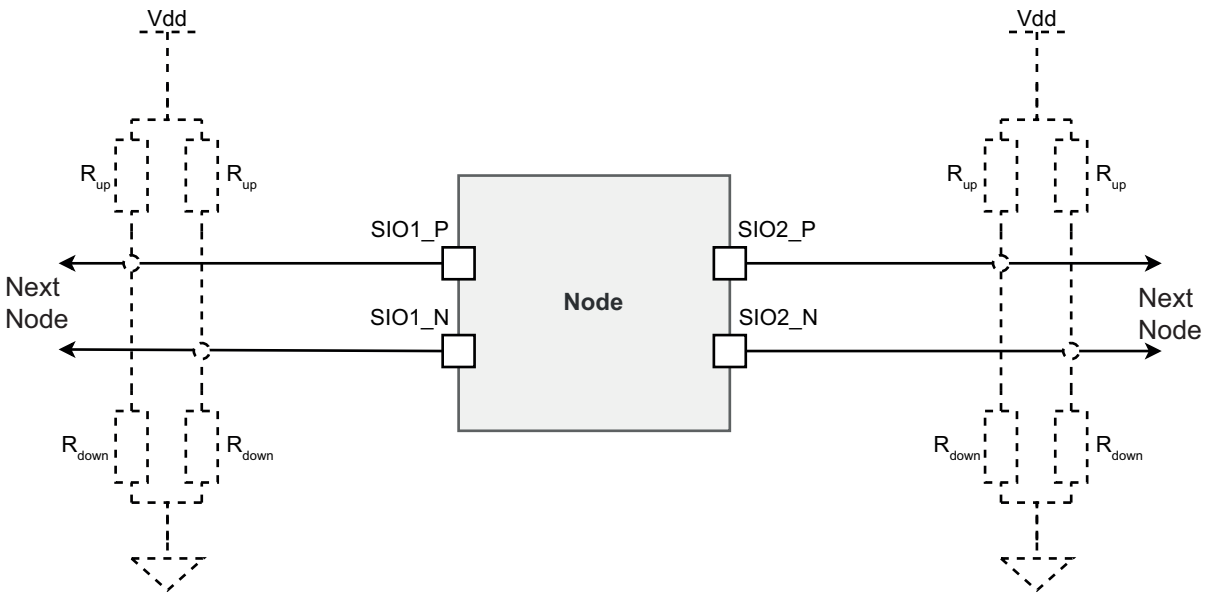


Table 4: Possible resistor arrangements

Mode	R @ SIOx_P	R @ SIOx_N	SIOx_P Input	SIOx_P Output	SIOx_N Input	SIOx_N Output
USE	up	up	-	ME	ME	-
LVDS	(down)	(down)	ME	ME	ME	ME
EOL	up	down	device specific			
	down	up	device specific			

Values in parentheses are not necessary but recommended. ME = Manchester encoded.

The recommended typical value for the internal pull down resistors for the LVDS mode is 100 kΩ. These may be integrated into each secondary node to simplify the board layout and reduce the number of external components required.

Note: It shall not be possible to change the communication mode during operation. However, mode detection may be restarted also by the RESET command.

Note: In EOL / MCU mode, the lines may be pulled to a value different from the idle value suggested by the resistor configuration after the mode detection has been successfully concluded. This may be used to simplify the MCU interaction on a device specific basis.

USE mode

This mode supports communication with MCUs, CAN-FD transceivers and node-to-node communication via two separate wires for transmission and reception (Tx and Rx). The communication over each wire is unidirectional.

Data is received via SIO_N only and is sent via SIO_P only. During receiving, SIO_P is switched to a high-Z state and during sending, data on SIO_N is ignored.

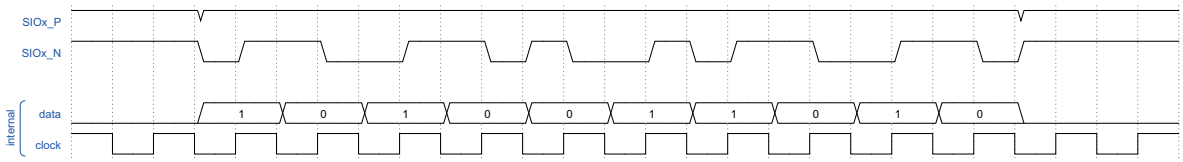
The bit stream is Manchester encoded. This mode is selected through two pull-up resistors and the idle state in this mode is high on SIO_P and SIO_N.

Implementation of this mode is mandatory.

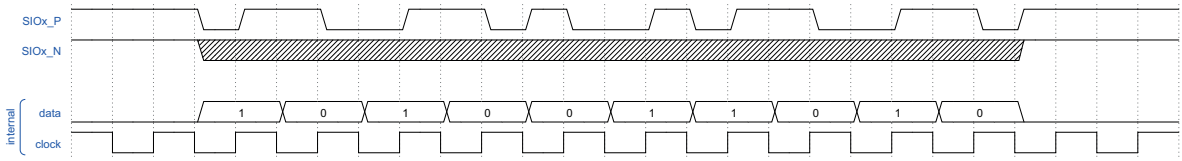
Note: This mode is also referred to as "CAN" mode, as it enables the use of CAN-FD transceivers. Due to the Manchester coding, the CAN-FD transceiver needs to support at least twice the data rate of the OSP

Figure 7: Communication in USE mode

Example waveform showing receiving communication in USE mode:



Example waveform showing transmitting communication in USE mode:



LVDS mode (optional)

This mode supports node-to-node communication and reduces EMI via differential signaling.

The bit stream is Manchester encoded. This mode is selected through two pull-down resistors (may be included in the node) and the idle state in this mode is $\Delta V = 0$ (recessive).

During idle state, the common mode voltage may be pulled to GND.

Implementation of this mode is recommended.

MCU mode / EOL mode (optional, device specific)

This mode is used to indicate that a unit is at the endpoints of the chain, i.e., either the first in line or the last in line and is necessary for proper initialization of the chain. The exact implementation of the end-of-line detection itself is device specific.

This mode may be selected through two different pull-up resistor configurations: either a pull-up resistor on SIO_P and a pull-down resistor on SIO_N, or vice versa. The idle state in this mode is consequently either high on SIO_P and low on SIO_N, or vice versa.

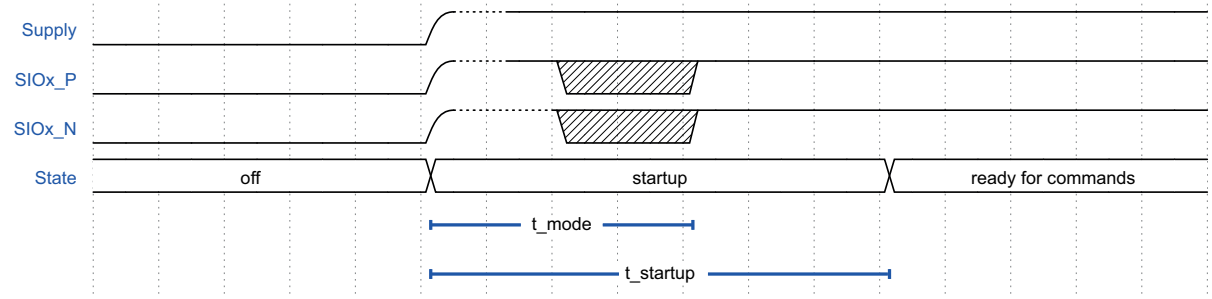
In addition to the end-of-line detection, this mode may be used to simplify communication with an MCU in a device specific fashion. Some examples for different implementations of this MCU mode are provided in the chapter "5.3 MCU mode implementation".

Startup Timing

When the supply voltage is connected, or the RESET command is received, the OSP node shall execute as part of its (device specific) startup sequence also the mode detection.

The Figure 8 illustrates the typical case of the USE mode.

Figure 8: Illustration of the startup behavior for the USE mode



When the supply voltage is applied, the two I/O pins assume their respective default voltage level as determined by the resistor configuration. The mode detection shall take place within the mode selection delay time t_{mode} (see chapter "2.8 Parameters"). After the startup time $t_{startup}$, the node is expected to be ready for communication.

During the mode detection, the device may apply a load to the I/O pins to improve the reliability of detecting the respective pin voltage and resistor configuration. As a change of voltage level might be interpreted by the MCU as a valid data transition, it is recommended to only activate the I/O interface of the MCU after the startup time (see chapter "2.8 Parameters").

2.3 Timeout and idle condition

A static line, i.e., no signal change is detected on any wire (SIOx_P and SIOx_N) for a duration of more than $t_{timeout}$ is interpreted as timeout and communication is canceled. The I/O port of this node will enter an idling state and be ready for a new transmission (both incoming and outgoing).

In general, the protocol does not expect interruptions during the telegram which are above the specified duty cycle distortion limits.

Note: The primary node is responsible for handling of response timeouts, i.e., when no response has been received for a time longer than the maximum expected wait time. There shall not be any built-in timeout mechanisms within a secondary node to ensure interoperability of different devices in a chain.

2.4 Start and stop conditions

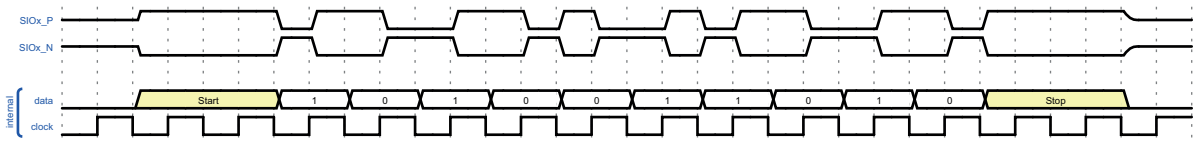
When a node is in idle state (no signal change for at least the timeout period) and any of the two wires changes its logic state, this is considered as the START condition. When a transmission has been finished and the line has been held static for at least the timeout period, this is considered as the STOP condition.

Note: In addition to the physical STOP condition, there is also a STOP condition defined on the data link layer (see chapter "End of message").

2.5 Start and stop sequence for LVDS mode

As the default condition for the LVDS mode, $\Delta V = 0\ V_{pp}$, VCM ~ 0 V, does not correspond to a logic high or low state, a special start and stop sequence is required. These start and stop sequences consist of a logic-high held for a duration of 2 bits as shown in Figure 9.

Figure 9: Start and stop sequence



Note: There are no such start / stop sequences defined or necessary for the USE or EOL / MCU modes.

2.6 Collisions

The OSP does not define any arbitration (collision avoidance) and does not enforce support for simultaneous receiving of upstream and downstream messages on the same node. With the exception of fast-forwarding of the same message, receiving a message on one port while receiving or transmitting a different message on the other port is not enforced and shall be treated as not supported.

The recommended behavior is the following:

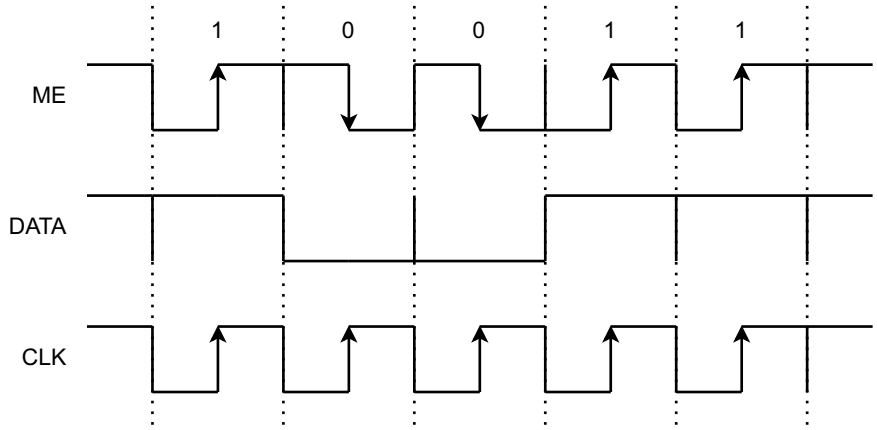
- Give priority to the I/O interface which first receives or starts sending a telegram
- Ignore all data received at the other I/O interface while the prioritized interface is still receiving or sending

Note: If the prioritized interface has finished receiving or sending data and the other port is still receiving, priority will then be given to the other port. This may cause receiving a truncated message, which will trigger one of the communication failures listed in chapter "3.3 Communication errors on data link level" or chapter "4.3 Communication errors on network layer level".

2.7 Manchester coding

The USE and LVDS modes use Manchester encoding based on the IEEE 802.3 standard, where a logical "1" is encoded by a rising edge and a logical "0" by a falling edge. Figure 10 shows DATA and CLK for illustration only.

Figure 10: Illustration DATA and CLK of Manchester encoding



Note: Manchester encoding can also be used in the EOL / MCU mode depending on the device specific implementation.

2.8 Parameters

Mode selection

The parameters for the mode selection are shown in Table 5.

Table 5: Mode selection

Parameter	Value	Unit	Conditions
Resistor value	min. 2.2. typ. 10 max. 25*	kΩ	
Mode sel. high	min. 2.85	V	
Mode sel. low	max. 0.7	V	
Mode sel. delay	min. 5 max. 500	µs	t_{mode}, after POR and reset
Startup Time**	typ. 1	ms	$t_{startup}$, time after POR and reset, device ready for communication

* Higher values up to 60 kΩ are possible when 3.3 V logic components are used
** The startup time is device specific. If different devices are operated in one chain, the recommended startup time for the MCU communication is the maximum of the individual startup times of the nodes in the chain. It is expected that devices are designed to minimize the startup time.

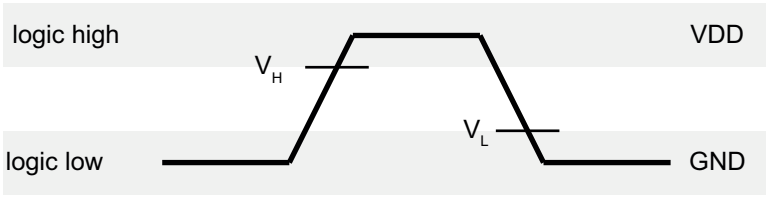
Single-ended signaling

The single-ended (SE) signaling (Table 6 and Figure 11) applies to USE, MCU and EOL modes from chapter "2.2.1 Mode selection".

Table 6: Single-ended signaling parameters

Parameter	Value	Unit	Comment
Input low level	max. 0.8	V	V_{IL}
Input high level	min. 2	V	V_{IH}
Output low level	typ. 0 max. 0.7	V	V_{OL}
Output high level	min. VDD - 0.7 typ. VDD	V	V_{OH}
Load capacitance	max. 35	pF	

Figure 11: Single-ended signaling



Differential signaling

The differential signaling is shown in Table 7 and Figure 12. Logic high is encoded by a positive differential voltage, V(SIO_P) - V(SIO_N) > 0, while logic low is encoded by a negative differential voltage, V(SIO_P) - V(SIO_N) < 0.

Table 7: Differential signaling parameters

Parameter	Value	Unit	Comment
Line termination	typ. 200	Ω	included in each node
Tx current	typ. ± 1.5	mA	
Common mode	min. 0.3 typ. 1.2 max. VDD + 0.3	V	active state
Common mode	typ. 0	V	idle state
Input low level	max. -0.15	V_{pp}	dV_{IL}=V(SIO_P) - V(SIO_N)
Input high level	max. 0.15	V_{pp}	dV_{IH}
Output low level	typ. -0.3 max. -0.2	V_{pp}	dV_{OL}
Output high level	min. 0.2 typ. 0.3	V_{pp}	dV_{OH}

Table 7: Differential signaling parameters

Parameter	Value	Unit	Comment
Load capacitance	max. 35	pF	

Note: The line termination resistor shall be integrated in the OSP node with each I/O interface to avoid additional external components.

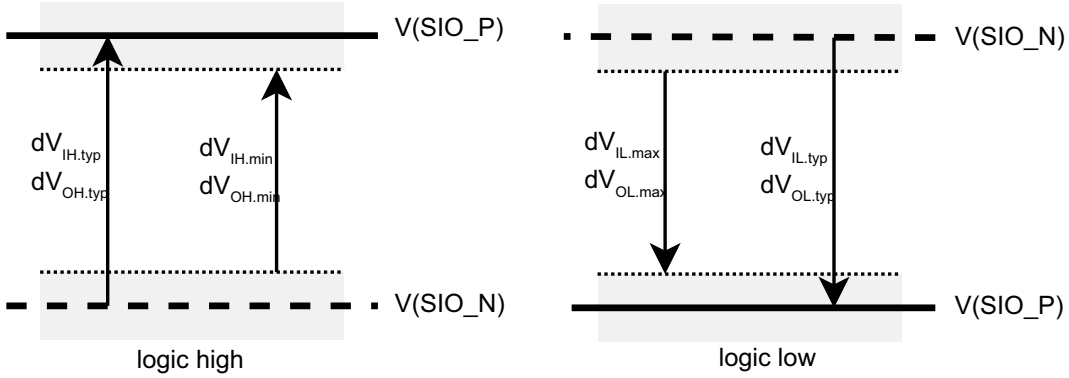
Depending on the PCB layout / technology used, it may not be possible to obtain impedance matching on the board. In this case, the OSP nodes should not be placed further apart than the critical distance L_{crit}:

$$L_{crit} \approx \frac{7,8}{\sqrt{\varepsilon_{eff}}}[m]$$

where ε_{eff} is the effective dielectric constant of the PCB board.

Differential lines shall be balanced and be routed parallel to minimize interference with radiated electromagnetic fields and noise injection

Figure 12: Schematic differential signaling



Communication Timing

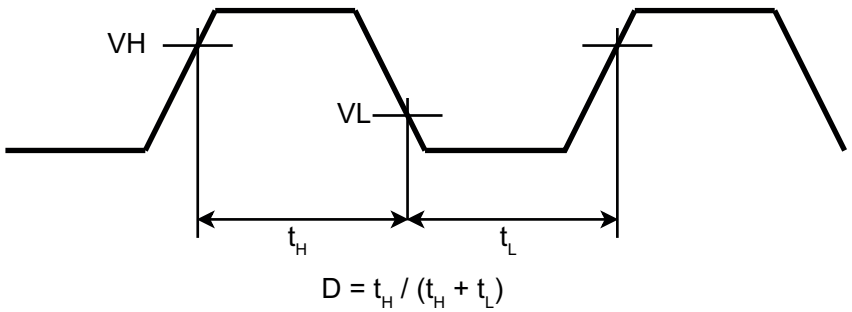
The communication timing is shown in Table 8 and the duty cycle for Manchester encoded signals is defined for a sequence of equal bits (0000…, 1111…) and for a sequence of alternating bits (010101…) as indicated in Figure 13.

Table 8: Communication timing

Parameter	Value	Unit	Conditions
Data rate	typ. 2.4	Mbit/s	
Rx rate variance	min. 8 max. 8	%	
Tx rate variance	min. 5 max. 5	%	
ME input duty cycle	min. 40 max. 55	%	H:L, see below
ME output duty cycle	min. 45 typ. 50 max. 55	%	H:L, see below
Timeout	min. 0.58	µs	$t_{timeout}$ ~ 1.5 bits
LVDS start / stop seq.	min. 0.77	µs	2 bits, can be longer
Fast-forward delay	min. 5.6	µs	t_{FF}, 14 bits, start in-to-start out
Response delay* (generic)	typ. 1.7	µs	3-4 bits, end-to-start
Response delay** (MCU mode, multicast)	min. 5.0	µs	12 bits, end-to-start

* The response delay, in general, is device and command specific and therefore not regulated in this specification; the given value reflects the expected typical behavior.
** In MCU/EOL bidirectional mode and for responses to multicast read commands, there shall be a minimum response delay introduced to enable the MCU to reconfigure the interface from sending to receiving between sending a request and receiving the response.

Figure 13: Duty cycle for Manchester encoded signals



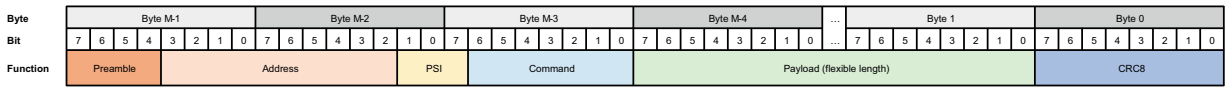
Note: The duty cycle, data rate, and clock variance requirements together with the logic levels define the slew rate and I/O timing requirements. These are not listed separately.

3 Data link layer

3.1 Message frame format

The communication is message based and its frame format is shown in Figure 14.

Figure 14: Message frame format



The byte and bit numbering refers to the order for transmission. Byte and bit order is most significant first. The message length is variable but limited to 12 bytes. The format is the same for upstream and downstream communication.

Preamble (4 bit)

Fixed to 0b1010.

Address (10 bit)

Downstream messages contain the address of the destination and upstream messages always contain the address of the sender in the address field. Allowed values for downstream messages are given in Table 9.

Table 9: Allowed values for downstream messages with write and read commands

Address value	Name	Write	Read	Description
0x000	Broadcast	yes	no	Use this value to address all nodes.
0x001 ... 0x3EF	Unicast	yes	yes	Individual address to select a specific device 1-1007
0x3F0 ... 0x3FR	Multicast /Groupcast	yes	yes	Multicast groups 0 - 14
0x3FF	INIT VALUE	no	no	Expected default value of an uninitialized node after POR or reset.

Note: The init value is listed for completeness only. It is not allowed as target address in a message. Messages with this address shall be silently discarded.

PSI (3 bit)

The PSI field indicates the length of the payload in bytes. The PSI value (Table 10) must match the specified value for each command (may be different for up- and downstream directions).

Table 10: Allowed values for PSI field

PSI	0b000	0b001	0b010	0b011	0b100	0b101	0b110	0b111
Payload	0	1	2	3	4	5	6	8

The total message length is PSI + 4 bytes.

Note: A payload of 7 bytes is not supported.

Command (7 bit)

Commands are mostly device specific. See network layer description.

Payload (0 to 64 bit)

Variable payload field. The length must match the value given in the PSI field.

CRC (8 bit)

The CRC field contains the CRC8 check word calculated over the full message excluding the CRC field.

CRC Implementation: $x^8 + x^5 + x^3 + x^2 + x + 1$.

Polynomial: 0x2F (normal notation)

Initial value: 0x00

XOR value: 0x00

Direction: MSB first

Note: A device can select not to implement the CRC. In this case, the CRC field is still mandatory.

Note: This implementation uses the same polynomial but is not compatible with the AUTOSAR implementation because of the different initial and XOR values of 0xFF.

Example:

Telegram = 0xA0 04 02

CRC check word = 0xA9

→ Sent Message = 0xA0 04 02 A9

Example:

Received Message = 0xA0 04 02 A9

CRC check word = 0x00

3.2 Message handling

Preamble

Messages without valid preamble are ignored, i.e., they are neither interpreted nor forwarded.

Note: A node shall not raise a communication error if a wrong preamble is received (see chapter "3.3 Communication errors on data link level").

Forwarding and fast forwarding

Message forwarding refers to the direct forwarding of a message without modification. A device shall not extended, shorten or modify a forwarded telegram.

Messages are forwarded to the next device if the address in the message frame is different from the node's address (this includes broadcast and multicast and applies to both downstream and upstream directions).

Multicast messages, for which the node is registered, should be forwarded only after the message has been fully processed (refer also to chapter "4.1.6 Multicasting").

All other messages should be forwarded immediately after the preamble and address field have been received, i.e., after 14 bits. This is referred to as fast-forwarding.

If a node is uninitialized, all messages should be fast-forwarded as the telegram address is always different from the node's address.

Note: The fast-forward latency or delay (time between start of telegram and start of forwarding) directly affects the minimum wait time between two consecutive messages and therefore it is recommended to start the forwarding as close to 14 bits as possible.

End of message

The PSI field is used to determine the STOP condition on the data link layer. Additional data received after the expected number of bits shall be silently discarded.

Note: When a transmission ends, the sending node should be able to directly release the lines. These are subsequently pulled to their respective idle state levels by the pull-up and pull-down resistors. Depending on their state at the end of the message, this will lead to an additional transition, which could be falsely interpreted as additional data.

Note: This behavior shall also apply for fast-forwarded messages.

Responses

Some commands trigger a response from the addressed node. The address field of the response message is set to the node's address to indicate the origin of the message.

Note: Response latency is device and command specific and therefore, with some exceptions, not regulated in this specification. Expected behavior is to send a response directly after the request has been received.

A minimum response delay is required at least in bidirectional mode when in MCU mode (to allow the MCU to switch between sending and receiving mode) or in a multicast read event (to avoid collisions at the first unit in the chain due to the response delay in MCU mode). Devices may implement this response delay with every response.

Note: It is recommended to use a timeout in the MCU to prevent the system from being stuck when the command or the response has been lost due to a collision or one of the nodes has stopped working properly. In this case, the P4EVENT command can be used to gather more information.

3.3 Communication errors on data link level

The OSP requires certain communication errors to be detected by each unit in a daisy chain independent of whether the unit is addressed (directly or via broadcast / multicast).

Incorrect preamble

Messages with incorrect preamble shall silently discarded without raising an error flag. They shall not be forwarded. The message shall not be interpreted by the node.

Invalid address

Messages with an address 0x3FF shall be silently discarded without raising an error flag. They shall not be forwarded. The message shall not be interpreted by the node.

Premature end of message

If the transmission ends before the expected number of bits (given by the PSI value) has been received, an error flag shall be raised irrespective of whether the message is fast-forwarded or targeted at the current node. The message shall not be interpreted by the node.

The message should not be forwarded. However, fast forwarding would start before this error can be detected and shall not be interrupted.

Note: The device shall not append additional data to match the message length to the PSI field, e.g., by using data from a previous message still in the message buffer.

Note: The CLEAREVENT command can be used to clear the error flags in such a case.

Incorrect CRC

If the CRC check byte is incorrect, an error flag shall be raised. The message shall not be interpreted by the node.

The message should not be forwarded. However, fast forwarding would start before this error can be detected and shall not be interrupted.

Note: the CLEAREVENT command can be used to clear the error flags in such a case.

Additional data after end of message

If the transmission continues after the expected number of bits (given by the PSI value) have been received, these additional bits shall be silently discarded. No error flag shall be raised.

Note: If the value in the PSI field is incorrect, the forwarded message length will match the PSI value but bits will be truncated from the original telegram. In this case, the CRC information will likely be incorrect and the addressed node will discard the telegram and raise an error flag. If the CRC check is disabled, however, the telegram may still be categorized as incorrect due to a mismatch on network layer level, e.g., PSI to command mismatch.

4 Network layer

The OSP defines a minimum set of commands (see chapter "Commands") on the network layer required to ensure interoperability of nodes of various types (mixed nodes) in a daisy chain. These are summarized in the following chapters. Refer to chapter "Conformance levels" for a classification of mandatory, recommended, and optional commands.

Note: Application and device specific commands are provided in specific device datasheets.

4.1 High level protocol features

4.1.1 Transaction acknowledgments

The purpose of an ACK mechanism is to signal the primary node that a transaction has been correctly received and executed by a secondary node.

The OSP does not directly define an ACK/NACK mechanism on physical or data link layer, but a similar mechanism can be implemented on the network layer on a device specific level using the non-OSP specific commands (value > 0x10).

When a command triggers a response, the response message itself serves as ACK.

When a command does not trigger a response and was not sent via broadcast, a device is free to send a message as response, e.g., its status register, which again serves as ACK.

When a command is sent via broadcast, the last device in the chain (indicated by the EOL condition) could again send a message as response, which then serves as ACK from this specific device and indicates that the chain is intact.

The implementation should be done in a way that allows to enable and disable the ACK mechanism, either via a register setting (preferred) or a free bit in the command structure (for example bit 5, but not recommended as this reduces the number of possible commands).

4.1.2 Loop-back communication

By default, communication shall be bidirectional, half-duplex, i.e., responses to the primary node are be returned over the same port that was used to receive the requests.

Additionally, devices may implement a loop-back mode, where the first and last units of the chain need to be connected with the primary node and responses are sent in the same direction along the chain as the request, i.e., responses are sent using the port opposite to the one used to receive the requests.

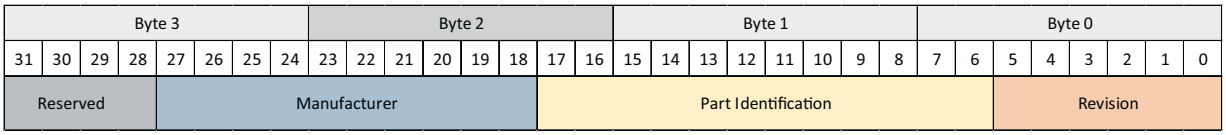
The loop-back mode is activated during initialization using the INITLOOP command.

4.1.3 Node identification

In order to handle mixed chains, the primary node needs to be able to identify the device type of each node in the chain. This is covered by a 32-bit, read-only device identification code (Figure 15), giving the following information:

- manufacturer (10 bits)
- part identification (12 bits)
- part revision (6 bits)

Figure 15: Definition of device identification code



Refer to chapter "5 Appendix" for a list of manufacturer codes.

The part identification and part revision codes are manufacturer-specific and shall be stated in the device datasheet.

The four bits 31 to 28 are reserved for later use and shall be set to zero.

4.1.4 Synchronized updates

When more nodes are used in a system, the network latency may lead to visible differences between the update moment of the first and last node. Especially, PWM settings for LEDs need to be updated on a single-node-level, leading to a "tearing" effect over the chain.

In order to solve this, the protocol foresees a SYNC command, that will enable each node to apply settings, that have been set beforehand and latched, with a minimal delay of broadcasting the command over the chain.

The exact implementation of both latch-able commands and the synchronize command is thereby device specific.

4.1.5 Broadcasting

An address of 0x000 initiates a broadcast, i.e., the same message is sent and received by all nodes in the chain with the lowest possible latency.

Messages with address 0x000 should always be fast forwarded.

Messages that cause a response cannot be sent with the broadcast address, i.e., broadcast readouts are not allowed.

4.1.6 Multicasting

In addition to the general broadcast, multicast groups can be defined. Devices can be assigned to one or more multicast groups via SETMULT. Once assigned, these devices respond to both the general broadcast address 0x000 and the multicast group address as well as to their respective unicast address.

Multicast group (Table 11) addresses are within the range from 0x3F0 to 0x3FE.

There are 15 different addresses for multicast groups available. A node can be registered to any of these by setting the respective bit in the mask register:

Table 11: Multicast groups

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Group	-	3FE	3FD	3FC	3FB	3FA	3F9	3F8	3F7	3F6	3F5	3F4	3F3	3F2	3F1	3F0

The set of commands compatible with a multicast call is device specific. In particular, devices can use this to enable multicast readouts.

If a device is registered to a multicast group and receives a multicast telegram with a command it does not support, the device shall raise a communication error but continue forwarding the telegram.

Messages addressed to a multicast group, for which the node is not registered, should be fast-forwarded.

Messages addressed to a multicast group, for which the node is registered, should be forwarded only after the message has been fully processed, i.e., the message has been fully received and, if applicable, the response telegram has been completely transmitted. This is necessary to avoid collisions between responses.

Note: If a node does not support multicast readout, all multicast messages can be fast-forwarded irrespective of whether the node is registered or not.

4.1.7 Serial broadcast read

Serial broadcasting allows to interact with and request information from more than one secondary node with only a single command sent by the primary node.

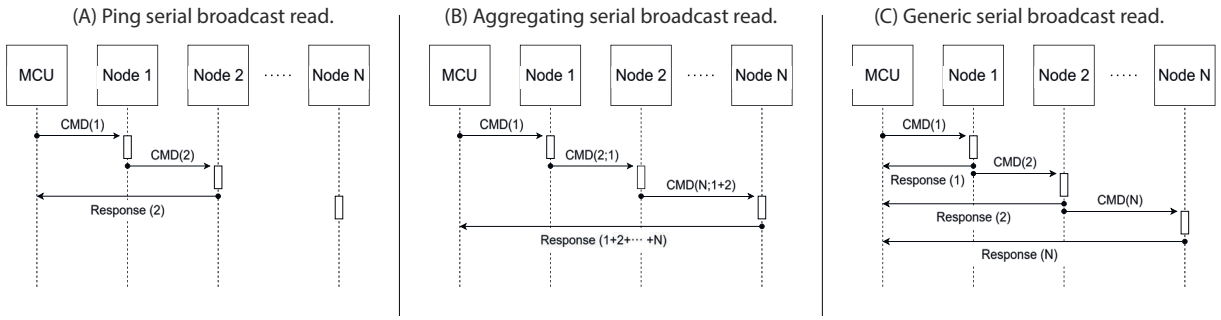
In contrast to the general broadcast via address 0x000, a serial broadcast read is always addressed to a specific node in the chain with a *specific* command and propagates through the chain by serially incrementing the address field (the INIT command is an example of a serial broadcast command).

Broadcast and multicast addresses are not allowed for serial broadcast commands.

Figure 16 illustrates the different types of serial broadcast readouts. The readout can be a readout of a single node in the chain based on a condition (ping), a consolidated or aggregated readout over the entire chain (aggregating), or an individual readout for every node (generic).

Note: Apart from the commands listed in chapter "4.4 Commands", each device may choose to implement serial broadcast readouts as device specific commands. This feature does then generally not work in mixed chains as every node would require an identical implementation.

Figure 16: Different serial broadcast readout flavors. (A) ping readout for certain condition, (B) aggregating readout over the chain, (C) individual readout over the chain.



Ping serial broadcast readout

The command propagates along the chain until a certain condition is fulfilled and the currently active node stops the forwarding, executes the command, and returns the response to the primary node.

A stop condition could be, for example, EOL (for the INIT command) or a pending event or failure flag (P4EVENT command).

Aggregating serial broadcast readout

The same mechanism can be used to retrieve a consolidated response over the chain, e.g., request the maximum / minimum value of some parameter.

In this case, a node receives the command with some parameter, aggregates its own value, then sends the messages to the next node with the parameter replaced with the aggregated value. Examples: determine max/min temperature, max/min forward voltage, count of nodes with an error, etc.

The serial readout stops at the EOL node, which returns the response to the primary node.

Generic serial broadcast readout

Each unit sends its response to the primary node in either bidirectional or loop-back mode, before sending the request with an incremented address field to the next unit.

Note: In loop-back mode, the sending node needs to maintain the minimum required wait time between sending the response and forwarding the command (see chapter "5.4 Waiting time between two messages").

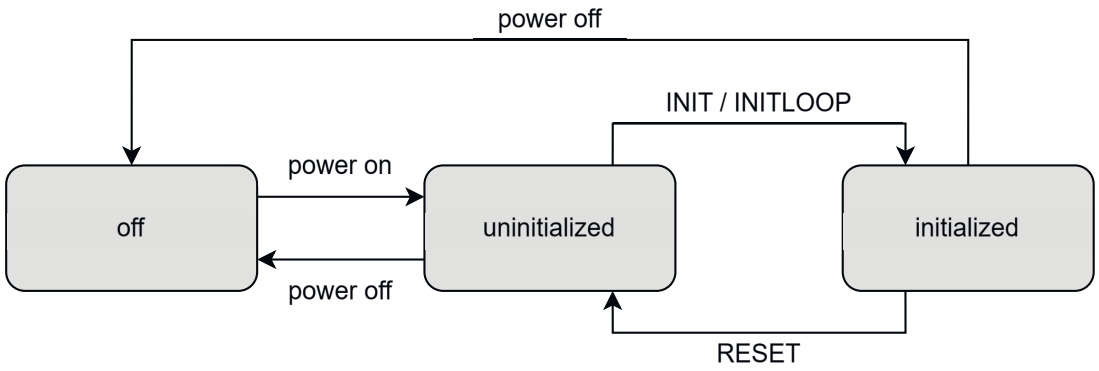
4.2 Node addressing

Node addressing is completely covered on protocol level through a dynamic initialization and automatic addressing procedure.

4.2.1 State diagram

A simplified state diagram for node addressing is shown in Figure 17.

Figure 17: State diagram for node addressing



The device is initially in the off state and automatically transitions to the uninitialized state after power has been applied. Nodes are responsive to (limited) communication (see next section).

Once the node has received and executed the INIT command and thus has received an address, its state changes to the initialized state, where it supports the full OSP command set including device specific commands.

A RESET command shall reset all registers to default values including the address state machine.

4.2.2 Uninitialized nodes

A node will reach the uninitialized state automatically after power-up or reset. In this state, it should only respond to a limited set of commands:

- INIT (bi-directional or loop back)
- RESET (via broadcast)
- CLEAREVENT (via broadcast)
- P4EVENT (bi-directional or loop back)
- All write commands (via broadcast)

Note: The possibility to send configuration settings via broadcast even to uninitialized nodes is useful, for example, to set the correct signal polarity for MCU mode or enable / disable the CRC check before initialization.

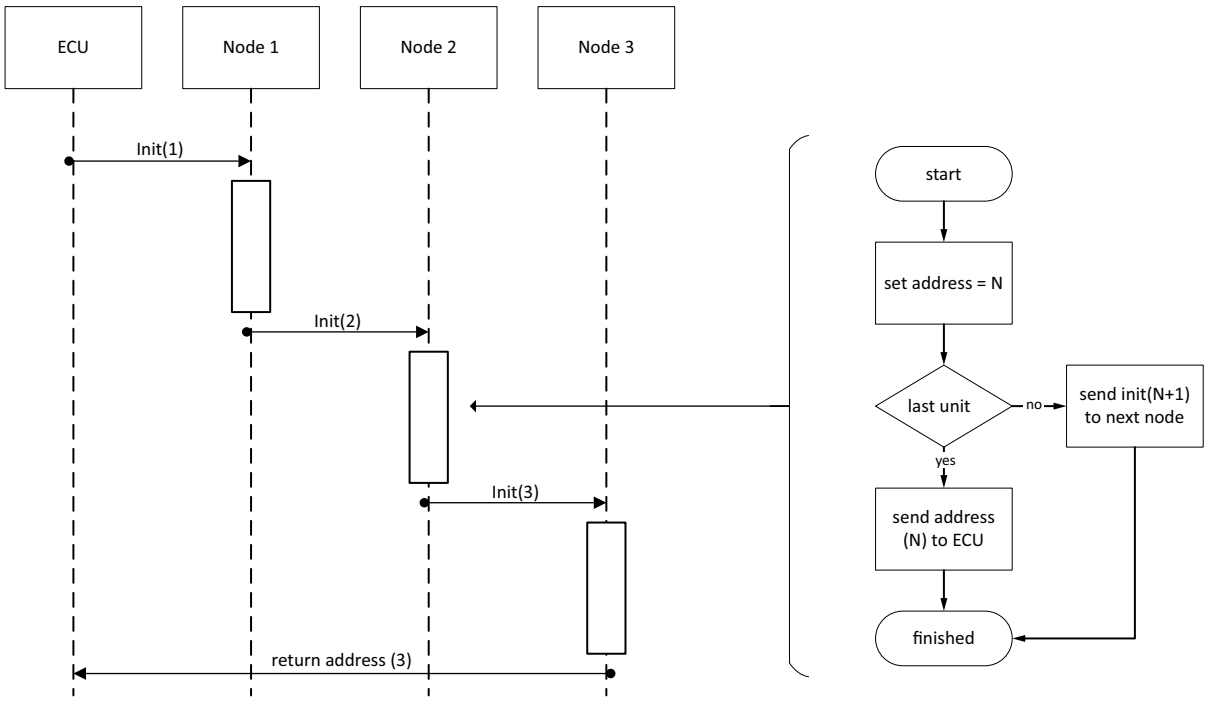
4.2.3 Chain initialization

The OSP requires dynamic addressing of units in a chain to support flexible arrangements. The addresses are assigned by an initialization command sent by the primary node (see Figure 18).

There are two flavors of INIT-commands, one for bidirectional communication and one for loopback communication mode (see chapter "INIT" and chapter "INITLOOP").

A device receives the INIT-command with some address set in the address field. It then sets its own address to the telegram's address, increments the address by one and sends the updated INIT-command to the next unit in the chain. The last unit in the chain returns its own address to the primary node and stops the initialization procedure. The response is either sent in backward or forward direction depending on the command flavor used.

Figure 18: Illustration of automatic addressing



Subsequent commands should only be sent after the response has been received by the MCU.

Note: It is possible to start the chain initialization with any address in the range 0x001 to 0x3EF. For example, initialization with address 0x100 will set the first node to 0x100, the second to 0x101, and so on.

If the maximum number of node addresses has been reached, the device is expected to stop the initialization procedure and to return its address to the primary node. Subsequent nodes should not be initialized.

Note: When the last unit in the chain is not in EOL mode, it will not recognize itself as last unit and will not send a response telegram.

If a device has already been initialized, it shall nevertheless execute the initialization procedure as outlined if it is directly addressed, i.e., if the INIT telegram has an address that matches the address of the (already initialized) node. This ensures proper incrementation of the address field

along the chain. Typical use cases include re-initialization of the chain if one or more nodes in the chain encountered a power cycle, for instance.

Note: An initialized node receiving an INIT-command with a target address different from its own will fast-forward the INIT-command without incrementing the address and without interpreting the command. Thus, it is possible to initiate an uninitiated node somewhere downstream the chain with a specific address that could be different from the address it would receive by the automatic addressing. This procedure is not recommended due to the risk of creating duplicate addresses. While not disturbing the chain, the second unit with the duplicate address will not be accessible anymore directly through unicast commands.

4.3 Communication errors on network layer level

The following errors will only be detected by nodes when they are addressed by the message (directly or via broadcast) after the full message has been received and the basic checks have been passed (see chapter "3.3 Communication errors on data link level").

Note: Unicast messages not addressed to a node, broadcast and multicast messages shall always be forwarded, independent of any potential protocol errors. Serial broadcast read messages shall not be forwarded in case of a communication error.

Unknown command

If a node receives an unknown command with code > 0x0F, an error flag shall be raised.

If a node receives an unknown command with code ≤ 0x0F (OSP specific command), no error shall be raised to enable compatibility with future protocol extensions.

In both cases, the command shall not be executed.

Incorrect length of argument

Some commands expect an argument delivered in the payload. If the length of the argument does not match the expectation, an error flag shall be raised. The command shall not be executed.

Incorrect argument

If an argument is expected and the data in the argument does not meet the requirements, an error flag is raised. The command shall not be executed.

Note: The decision whether an argument meets the expectations is device specific.

Broadcast readout attempt

If a readout command is received via the broadcast address 0x000, an error flag shall be raised. The command shall not be executed.

Note: A broadcast write command is allowed to have the transaction acknowledgement feature implemented (see chapter "4.1.1 Transaction acknowledgments").

Inappropriate broadcast attempt

If a serial broadcast read command (e.g., INIT, INITLOOP, P4EVENT, and P4EVENTLOOP) is received via the broadcast address 0x000 or via a mutlicast address, an error flag shall be raised. The command shall not be executed.

4.4 Commands

Table 12 shows a list of recommended and required commands to ensure the interoperability of mixed nodes in a daisy chain.

Refer also to chapter "1.2 Conformance levels" for more details..

Table 12: List of recommended and required commands

Command	Code	Description	Mandatory / Optional
	0x00 … 0x0F	OSP Specific commands	
RESET	0x00	Node reset	Mandatory
CLEAREVENT	0x01	Clear pending event / error flags	Optional
INIT	0x02	Initialize chain in bidirectional mode	Mandatory
INITLOOP	0x03	Initialize chain in loop-back mode	Optional
IDENTIFY	0x07	Returns node identification code	Optional
P4EVENT	0x08	Poll unread notifications from all nodes in the chain, bidirectional mode	Optional
P4EVENTLOOP	0x09	Poll unread notifications from all nodes in the chain, loop-back mode	Optional
READMULT	0x0C	Returns the multicast mask register	Optional
SETMULT	0x0F	Writes the multicast mask register	Optional
SYNC	0x0F	Synchronize the chain	Optional
	0x10 … 0x7F	Device specific commands	

The command range 0x00 to 0x0F is reserved for OSP specific commands. Nodes will not raise an error if they receive an OSP protocol command which is not implemented in this particular node.

All commands shall support unicast, i.e., direct addressing of individual nodes. Whether a command shall additionally support multicast and / or broadcast is indicated with every command.

Device specific commands and register access commands (0x10 to 0x7F) are free to be implemented for any functionality, to be decided by the node (type) manufacturer.

RESET

Performs a complete reset of one (unicast) or more (multicast / broadcast) devices and brings the node into the uninitialized state with all values set to the device specific default values.

The properties of the RESET command:

- Takes no arguments
- No response
- Broadcast is possible
- Multicast is possible

CLEAREVENT

Clears all pending events, e.g., failure flags, of a device. If the condition that led to the event still persists, the event will be raised again.

The properties of the CLEAREVENT command:

- Takes no arguments
- No response
- Broadcast is possible
- Multicast is possible

INIT

Initiates the automatic addressing of the chain (see chapter "4.2.3 Chain initialization") and sets the communication direction to bidirectional. The command shall be addressed to the first secondary node after the primary node.

The last unit in the chain shall send a message with its address back to the primary node. The exact implementation of the return message is device specific (e.g., a node could return a status register).

The properties of the INIT command:

- Takes no arguments
- Broadcast is NOT possible
- Multicast is NOT possible

Note: As the address is set through the address field of the telegram, the command does not have a payload.

Note: This is a serial broadcast ping command with EOL as stop condition.

INITLOOP

Initiates the automatic addressing of the chain (see chapter "4.2.3 Chain initialization") and sets the communication direction to loop-back. The command shall be addressed to the first secondary node after the primary node.

The last unit in the chain shall send a message with its address via loop-back, i.e., in downstream direction, to the primary node. The exact implementation of the return message is device specific (e.g., a node could return a status register).

The properties of the INITLOOP command:

- Takes no arguments
- Broadcast is NOT possible
- Multicast is NOT possible

Note: As the address is set through the address field of the telegram, the command does not have a payload.

Note: This is a serial broadcast ping command with EOL as stop condition.

IDENTIFY

Returns the device type identifier (see chapter "4.1.3 Node identification") with a length of 32 bit.

The properties of the IDENTIFY command:

- Takes no arguments
- Broadcast is NOT possible
- Multicast is possible but multicast support is device specific

P4EVENT

Poll the event, error or notification status of all nodes following the addressed node in bidirectional mode. The command may be addressed to any secondary node in the chain.

For initialized devices:

If new events are pending, the respective device shall send a message with information on the events to the primary node and stop the further processing of the command. The type of information and length of payload is device specific. For example, it could include failure flags or interrupt flags.

If no new events are pending, the address field shall be incremented by one and the message shall be forwarded to the next node in the chain.

The last unit in the chain shall return its event status in any case to signal the completion of the command.

For uninitialized devices:

Uninitialized nodes shall send a response to the primary node indicating that they are not yet initialized and stop the further processing of the command. The exact response is device specific.

The properties of the P4EVENT command:

- Takes no arguments
- Broadcast is NOT possible
- Multicast is NOT possible

Note: This is a serial broadcast ping command with pending events and EOL as stop conditions.

P4EVENTLOOP

Poll the event, error or notification status of all nodes following the addressed node in loop-back mode. The command may be addressed to any secondary node in the chain.

For initialized devices:

If new events are pending, the respective device shall send a message with information on the events to the primary node in downstream direction and stop the further processing of the command. The type of information and length of payload is device specific. For example, it could include failure flags or interrupt flags.

If no new events are pending, the address field shall be incremented by one and the message shall be forwarded to the next node in the chain.

The last unit in the chain shall send its event status to the primary node in any case to signal the completion of the command.

For uninitialized devices:

Uninitialized nodes shall send a response to the primary node in downstream direction indicating that they are not yet initialized and stop the further processing of the command. The exact response is device specific.

The properties of the P4EVENTLOOP command:

- Takes no arguments
- Broadcast is NOT possible
- Multicast is NOT possible

Note: As this command is also used to poll the initialization status of the chain and uninitialized nodes do not have a way to detect the expected response direction, the P4EVENTLOOP command is needed to explicitly denote the correct response direction.

Note: This is a serial broadcast ping command with pending events and EOL as stop conditions.

READMULT

Return the 16-bit multicast register (see chapter "4.1.6 Multicasting").

The properties of the READMULT command:

- Takes no arguments
- Broadcast is NOT possible
- Multicast is possible but support for multicast is device specific

SETMULT

Register a node to any of the 15 multicast groups 0x3F0 to 0x3FE by setting the 16-bit multicast register (see chapter "4.1.6 Multicasting").

A node can be assigned to multiple groups.

The properties of the SETMULT command:

- Expects 16-bit multicast register as argument
- No response
- Broadcast is possible
- Multicast is possible but support for multicast is device specific

SYNC

Synchronization command that can be used to apply settings that have been latched before, for example, PWM values. The exact implementation of the command is device specific.

Properties of the SYNC command:

- Takes no arguments.
- No response.
- Broadcast is possible.
- Multicast is possible.

5 Appendix

5.1 Identification codes

Manufacturer ID

Table 13 lists assigned 10-bit manufacturer IDs.

Table 13: 10-bit manufacturer IDs

Code	Description	Comment
0x000	ams OSRAM	
0x002	DOMINANT OPTO	

The manufacturer IDs are assigned by ams-OSRAM. Interested parties should contact ams-OSRAM (osp.info@ams-osram.com)

Part Identification

Part identification codes are manufacturer specific.

It is recommended to assign new part identification codes to every device having a distinct functionality or every device undergoing a major change compared to the predecessor.

Revision

Revision codes are manufacturer specific.

It is recommended to assign a new revision code to every new generation of a device if there is only a minor change in functionality.

5.2 EOL detection

Each node needs to be able to detect whether it is the last node in the chain as required for proper initialization. The exact mechanism is device specific.

Some examples for possible implementations are given below.

Separate MCU / EOL mode with physical discrimination

Use the resistor configuration to create a unique discrimination between the first-in-line (MCU) and the last-in-line (EOL) secondary node

Table 14: Resistor configuration for unique discrimination

Mode	R @ SIOx_Po	R @ SIOx_N	SIOx_P Inout	SIOx_P Output	SIOx_N Input	SIOx_N Output
MCU	up	down	device specific			
EOL	down	up	-	device specific	-	device specific

Only the EOL device will terminate the enumeration procedure during the initialization phase and return its address to the primary node.

Note: The EOL device should support sending messages in downstream direction, i.e., towards the port with the EOL resistor configuration, to enable the loop-back mode.

Combined MCU / EOL mode with link layer discrimination

In this case, the discrimination between MCU and EOL node is not done via resistor configuration but via the propagation direction of the INIT command.

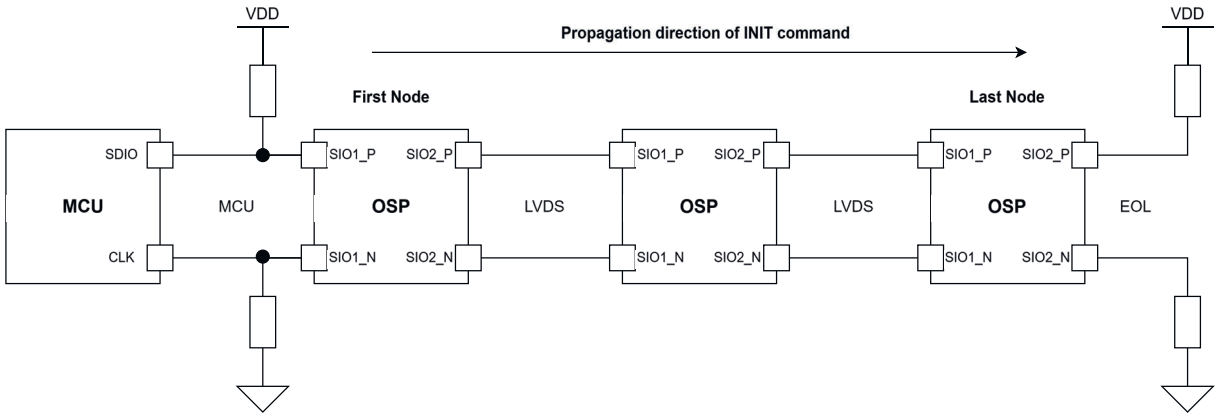
The combined MCU/EOL mode can be selected using any of the two possible resistor configurations shown in Table 15.

Table 15: Resistor configuration for combined MCU/EOL

Mode	R @ SIOx_Po	R @ SIOx_N	SIOx_P Inout	SIOx_P Output	SIOx_N Input	SIOx_N Output
MCU/ EOL	up	down		device specific		
	down	up		device specific		

The EOL detection is based on the direction of propagation of the initialization message along the chain as illustrated in Figure 19. A node is considered as last in chain if it receives an INIT telegram on a port opposite of a port configured as MCU/EOL.

Figure 19: Working principle



Possible configurations are shown in Table 16.

Table 16: MCU / EOL configuration. x - other mode (USE, LVDS)

Resistor configuration at port...		Node classification, when INIT received at port...	
SIO1	SIO2	SIO1	SIO2
MCU / EOL	x	first node	last node
x	MCU / EOL	last node	first node
MCU	MCU / EOL	last node	last node
x	x	x	x

Note: The exact implementation is device specific. For example, a device might choose to support both options or only a specific option.

5.3 MCU mode implementation

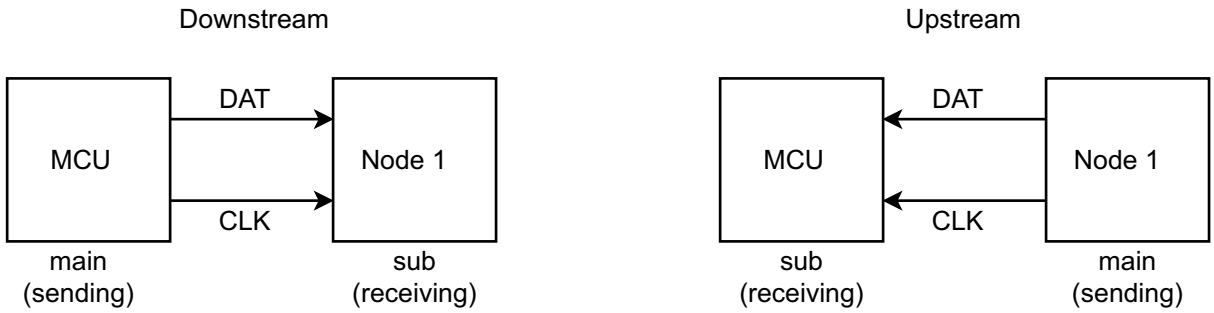
The following sections illustrate some possible implementations of the MCU mode.

The exact implementation is device specific.

MCU mode (A) - 2-wire SPI

Both downstream and upstream communication (Figure 20) use a two-wire SPI interface with clock and data signals (no chip select) with NRZ coding (no Manchester encoding).

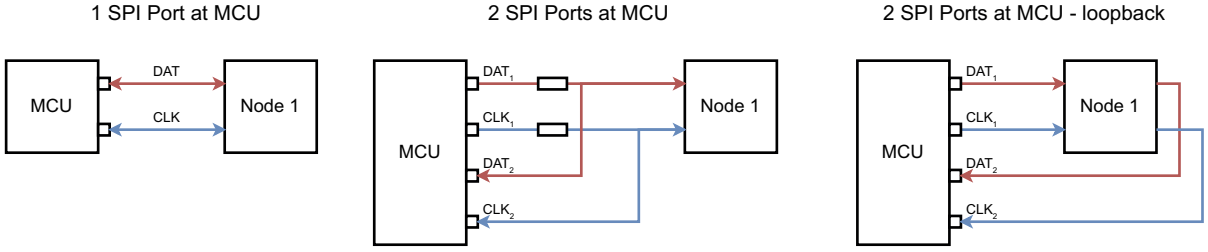
Figure 20: SPI interface for downstream and upstream



The SPI main role is always taken by the transmitting device, i.e., for downstream communication by the MCU and for upstream communication by the first (bidir) or last (loop) node.

There are three possible options to realize this on MCU side like shown in Figure 21.

Figure 21: MCU options



The first two options work in bidirectional mode, the third option only for the loop-back mode.

When 2 SPI ports are used on the MCU, the first is always main and the second is always sub. The two resistors in the second case are needed to decouple the main and sub pins in case both the MCU and node 1 become SPI main.

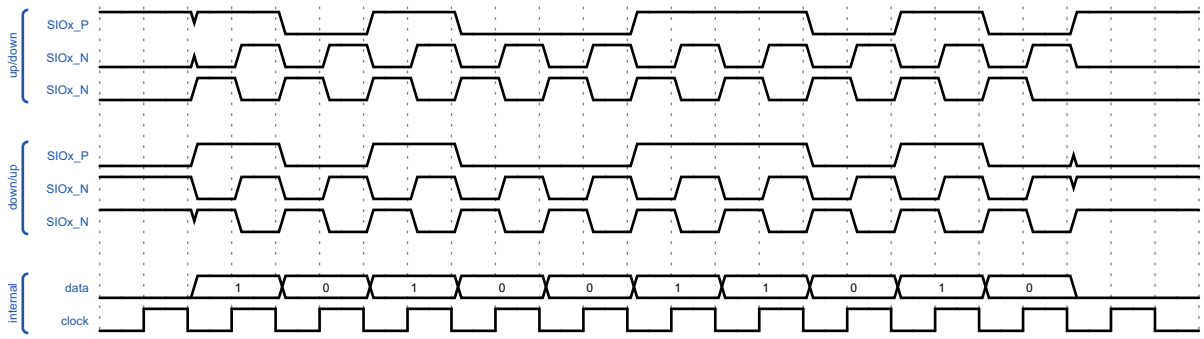
When only 1 SPI port is used, the MCU needs to toggle the SPI role between main and sub after a message has been sent and before the response is received.

There is a wait time of min. 5 µs between the end of an incoming message and the start of a response that is sent back to the primary node.

Ideally, the clock polarity on SIOx_N is selectable through register settings.

Figure 22 illustrates the possible signals for the two EOL / MCU modes (pull-up on SIO_P and pull-down on SIO_N and vice versa) and the two possible clock polarities, respectively.

Figure 22: Possible signal for EOL / MCU modes



These curves apply both to upstream and downstream communication. There is NO dedicated start / stop sequence.

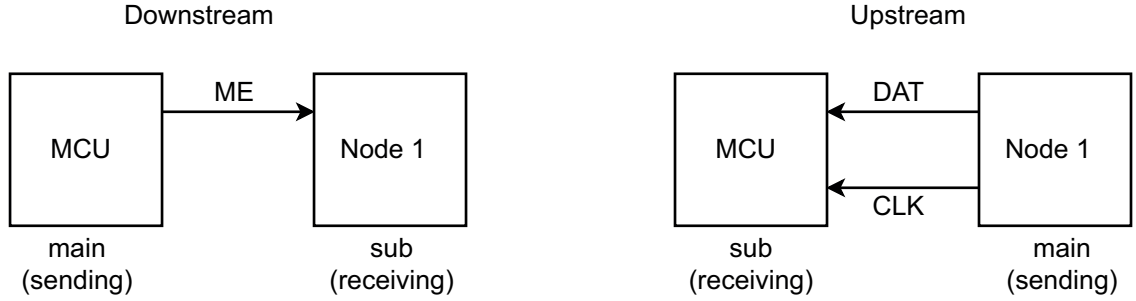
The same timing requirements apply as given in chapter "2.8 Parameters" for communication timing.

Note: The OSP node should support SCK interruptions from the MCU of up to 1 bit (limited by $t_{timeout}$, see chapter "2.8 Parameters").

MCU Mode (B) - Mixed SPI

This option is identical to the pure SPI given in the previous section with the modification that the downstream communication (Figure 23) uses a single-wire Manchester encoded signal.

Figure 23: Downstream communication



The advantage lies in a simpler design for the nodes as no additional blocks for receiving are needed (the receiving part is equal to the USE mode). The upstream direction still offers the advantage of easy decoding by the MCU (no Manchester decoder needed).

Refer to the previous section for possible connection options.

There is a wait time of min. 5 µs between the end of an incoming message and the start of a response that is sent back to the master.

The downstream signal follows the timing diagram below for the two possible EOL / MCU modes (pull-up on SIO_P and pull-down on SIO_N and vice versa). See chapter "2.7 Manchester coding" for the definition of the Manchester encoding.

It is possible to use SPI with twice the nominal data rate to construct the ME signal in the MCU as shown in the Figure 24 and Figure 25. The clock signal created by the SPI is ignored by the node (if connected to SIOx_N).

Figure 24: ME signal in the MCU

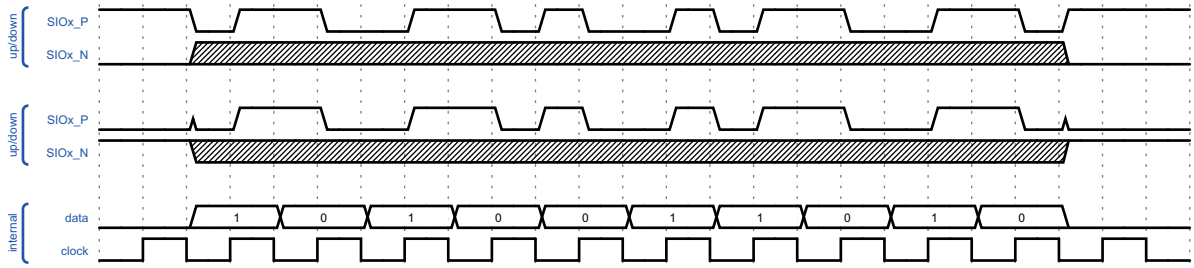
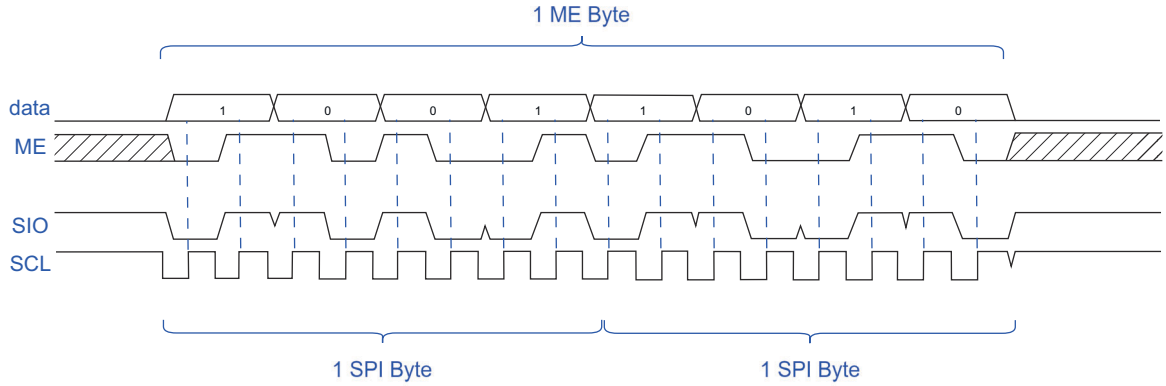


Figure 25: Manchester encoding via SPI



The upstream signal is identical to the one given in the previous section, i.e., consisting of clock and data.

There is NO dedicated start / stop sequence.

The same timing requirements apply as given in chapter "2.8 Parameters" for communication timing. The downstream bit rate needs to be twice the nominal rate, i.e., 4.8 Mbit/s.

Note: When the line is released by the MCU after the transmission, it will approach its idle state as defined by the resistor configuration. Depending on the RC constant of the interface, this transition might be interpreted as an additional data bit by the node. Therefore, it is recommended to either drive and hold the line in a defined state for a period of 4 bits (which is the minimum possible when using SPI with twice the data rate, or 2 bits when using GPIO), or ensure that the node is using the PSI value to determine the message length and ignores any additional bits (see chapter "Additional data after end of message").

5.4 Waiting time between two messages

To avoid collisions between two messages sent by the MCU, a waiting time t_{wait} has to be added by the MCU after one message has been sent and before the next message can be sent.

This wait time is dependent on the length of the telegram (m), the number of nodes in the chain to pass (N), the internal and external oscillator frequencies, the synchronization jitter, and on systematic asymmetries in the receiver of transmitter parts that could lead to a systematic difference in the fast-forward latency between telegrams addressed to even and odd nodes (e.g., differences in rise and fall times of the receiver circuitry).

Therefore, the recommendation is to evaluate and set the wait time for specific scenarios based on the recommendations given in the manufacturers' datasheets.

In general, the wait time t_{wait} as function of number of nodes (N) to pass and number of bits in a telegram (m) can be estimated as

$t_{wait} = max_{ij}(t_{wait,ij})$

where

$$t_{wait,\,ij} \approx t_{FF} + \frac{|f_j - f_i|}{f_j f_i} m + \frac{3}{a f_i} \sqrt{\frac{N}{6}} + \Delta_{even-odd}\, N$$

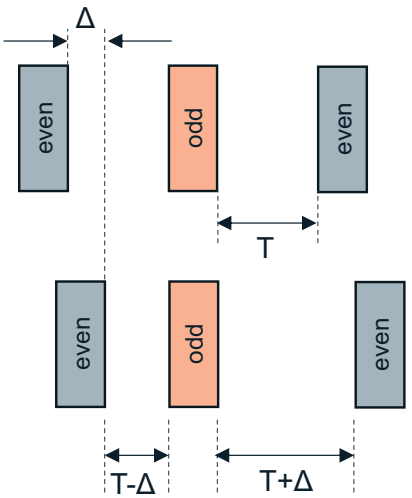
and

- t_{wait} is the minimum end-to-start wait time between two consecutive telegrams,
- $t_{wait,ij}$ is the end-to-start wait time between two consecutive telegrams required by nodes *i* and *j,*
- t_{FF} is the fast-forward delay,
- f_i is the data rate of the sending node i,
- f_j is the data rate of the receiving node j,
- *a* is the oversampling ratio between the internal oscillator and the OSP data rate (e.g., 8),
- $\Delta_{even-odd}$ is the timing asymmetry between even and odd telegrams per node

and the 3-sigma value of the synchronization jitter has been used.

Note: the wait time depends on the number of nodes N to pass, i.e., the target address - however, it is recommended to set N to the total number of nodes in the chain.

Figure 26: Illustration of the definition of the timing asymmetry between even and odd telegrams.



Note: if consecutive telegrams have the same address parity, e.g., 0x001 - 0x003 - ... or 0x002 - 0x004 - ..., the timing asymmetry between even and odd telegrams does not affect the wait time and drops out of the equation. The same applies to multiple broadcast telegrams sent consecutively, as all have the same address 0x000. Sending all telegrams with the same parity first before switching to the other parity, therefore, provides the fastest way to individually address every node in the chain.

6 Examples

6.1 Command range partitioning

Device specific commands and register access commands (0x10 to 0x7F) are free to be implemented by each node individually.

The table below shows a possible partitioning having a device specific command range and a dedicated register range supporting 5-bit register addressing and a read/write bit.

Table 17: Example command field partitioning

Meaning	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OSP Command	0	0	0	OCMD[3:0]			
Device Specific Command (DCMD > 0x0F)	0	DCMD[5:0]					
Register Read	1	REG[4:0]					0
Register Write	1	REG[4:0]					1

6.2 Code examples

CRC calculation

A possible C++ implementation of the CRC algorithm is the following:

```
uint8_t crc(uint8_t *InBytes, uint8_t InSize)
{
    uint8_t crc = 0x00;
    uint8_t extract;
    for(uint8_t i = 0;i < InSize; i++) {
        extract = *InBytes;

        for (uint8_t mask = 8; mask > 0; mask--) {
            if (((extract >> (mask - 1)) & 0x01) == ((crc >> 7) & 0x01))
                crc = crc << 1;
            else
                crc = (crc << 1) ^ 0x2F;
            crc &= 0xFF;
        }
```

```
        InBytes++;
    }
    return crc;
}
```

Manchester encoding

A possible C++ implementation of the Manchester encoding is the following:

```
void manchester_encoding(uint8_t *p_buffer, uint8_t *p_bufferNew, uint8_t byteCount)
{
    uint16_t *p_buffer16Bit = (uint16_t*) p_bufferNew;
    uint8_t in;
    uint16_t out;
    for (uint8_t i = 0; i < byteCount; i++) {
        in = *(p_buffer + i);
        out = (uint16_t) 0;
        for (uint8_t j = 0; j < 8; j++) {
            out <<= 2;
            if (in & 0x80)  out |= 0x1;
            else            out |= 0x2;
            in <<= 1;
        }
        *p_buffer16Bit = (out >> 8) | ((out & 0xFF) << 8);
        p_buffer16Bit++;
    }
}
```

The 8-bit input buffer is converted to an 8-bit output buffer with twice the number of bytes.

Example:

0x00 --> ME 0xaaaa

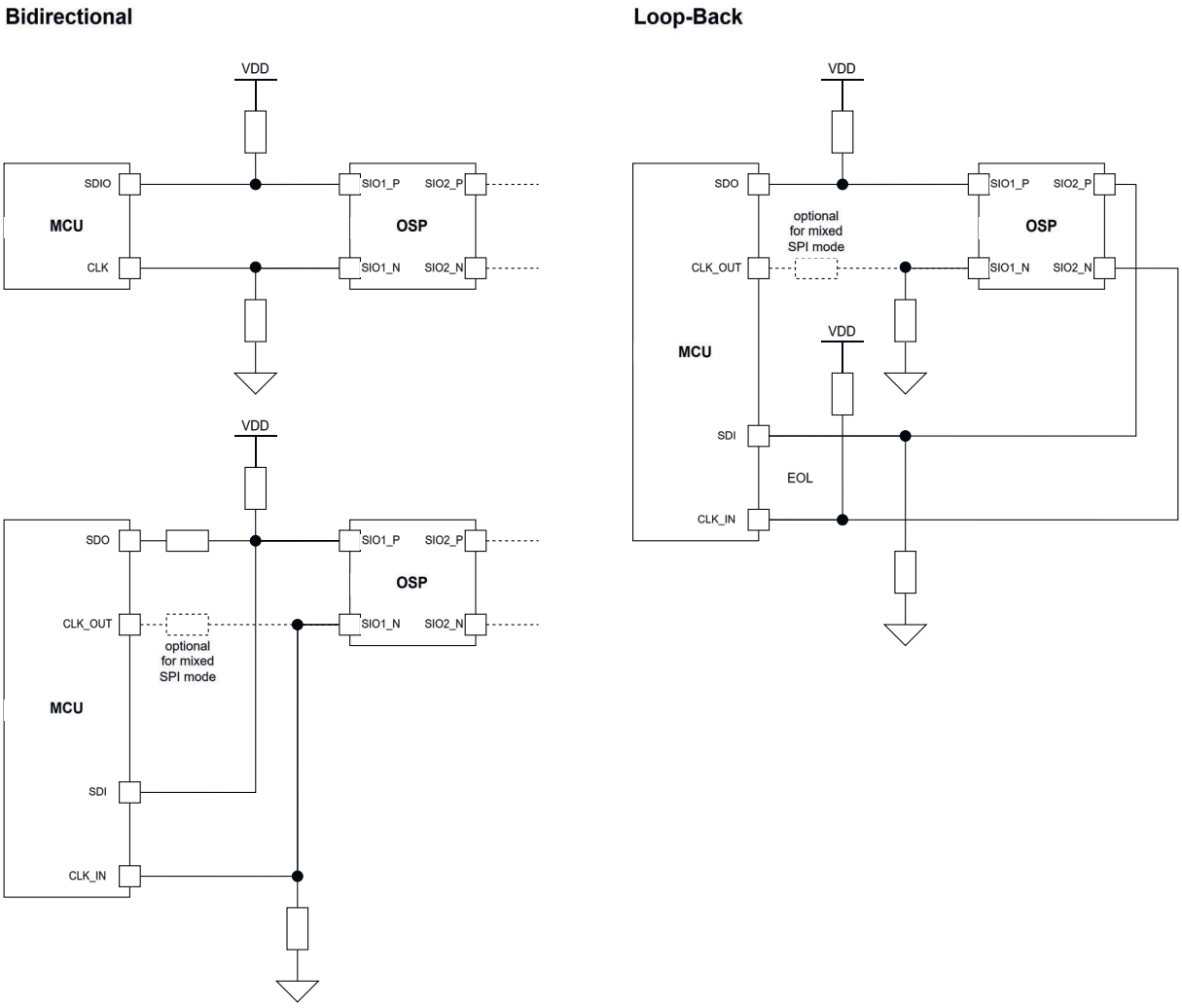
0x01 --> ME 0xaaa9

6.3 Example chain configurations

MCU connection schemes

Figure 27 illustrates basic connection options between the OSP nodes and an MCU.

Figure 27: MCU connection schemes



The two figures on the left show connection options suitable for bidirectional communication using a single SPI port (top) and using two SPI ports (bottom). If the single port option is used (top figure), the MCU should be sufficiently fast to switch between sending and receiving mode in order to be able to receive responses from the first unit in the chain (see chapter "Communication Timing" for timing requirements).

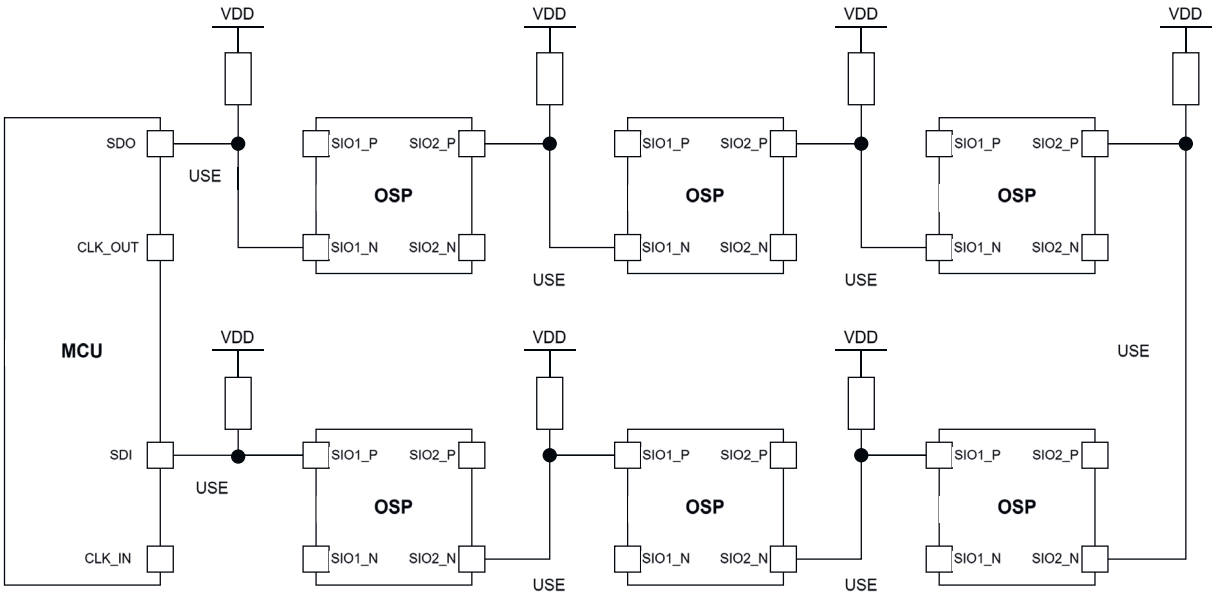
The resistors between the main SPI port of the MCU and the first node in the bottom figure have to be chosen together with the total capacitance and the pull-up and -down resistors to ensure voltage levels and signal rise- and fall times are within specification (see chapter "Single-ended signaling"). Ideally, the MCU pins should be set to a high impedance state while receiving data from the OSP node.

Note: Some MCUs require a chip select (CS) signal for receiving messages in sub mode. A possible solution is to drive the CS pin of the SPI port with another GPIO pin of the same MCU.

Note: All single-ended connections should be kept as short as possible to ensure signal integrity and to lower EMC concerns.

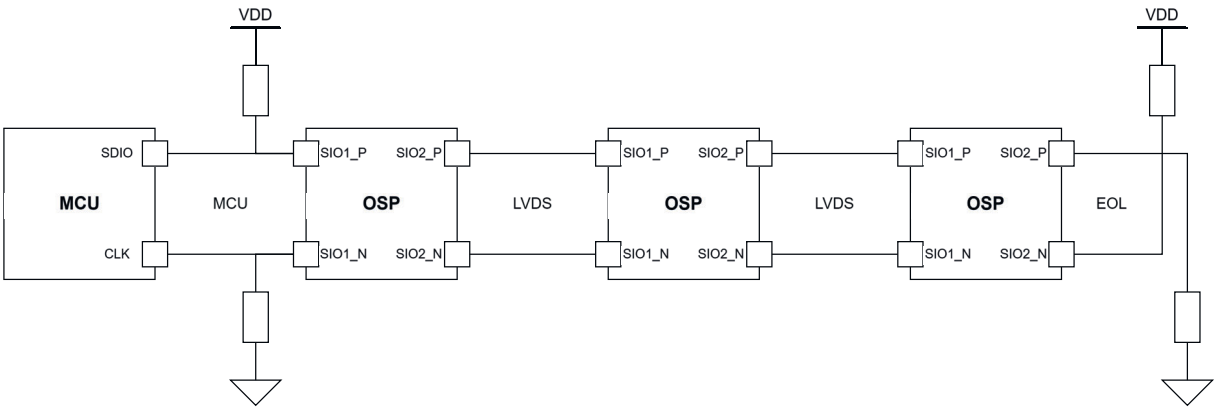
Loop-Back chain with USE mode only

Figure 28: Loop-Back chain with USE mode only



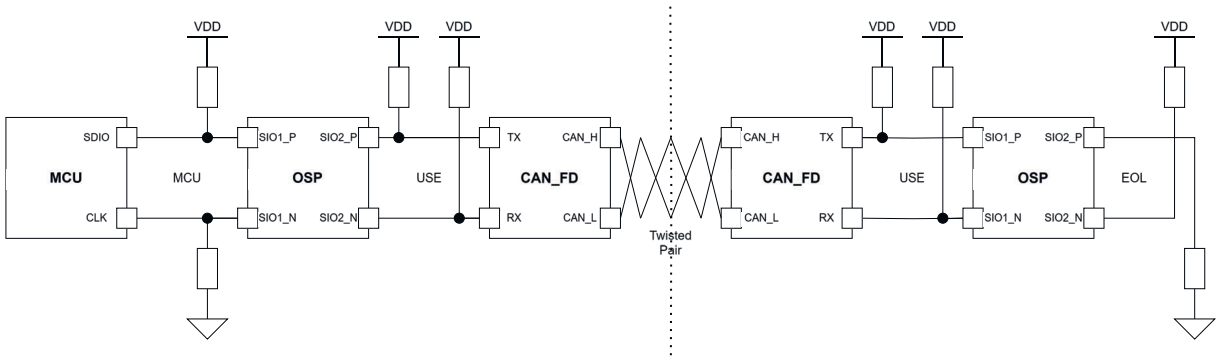
Bidirectional chain with MCU and LVDS modes

Figure 29: Bidirectional chain with MCU and LVDS modes



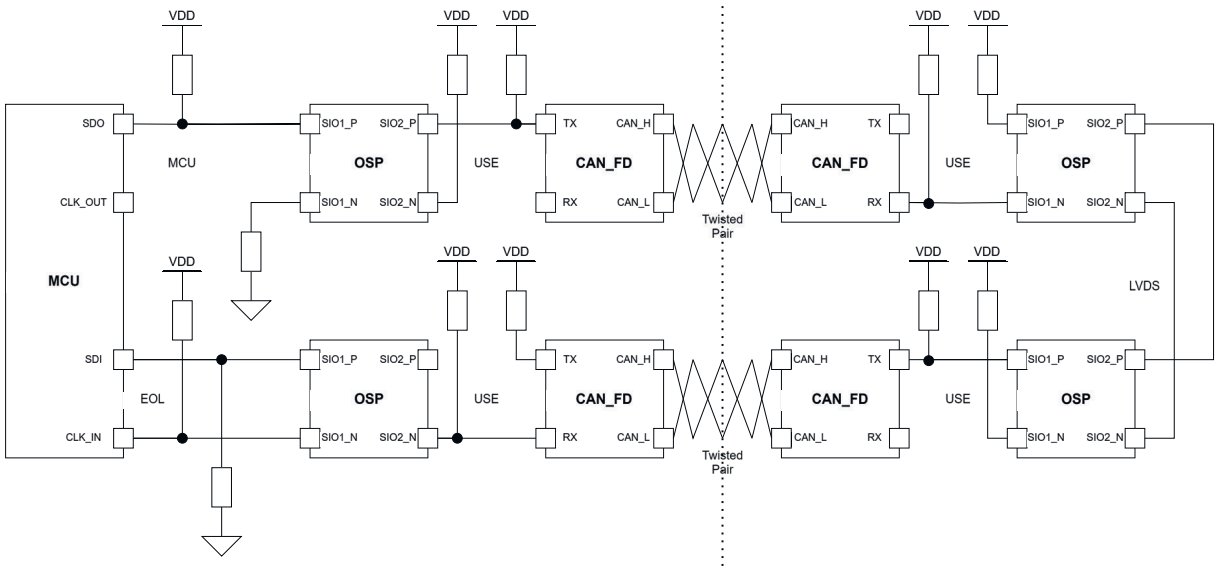
Bidirectional chain with MCU, LVDS and USE modes

Figure 30: Bidirectional chain with MCU, LVDS and USE modes



Loop-Back chain with MCU, LVDS and USE modes

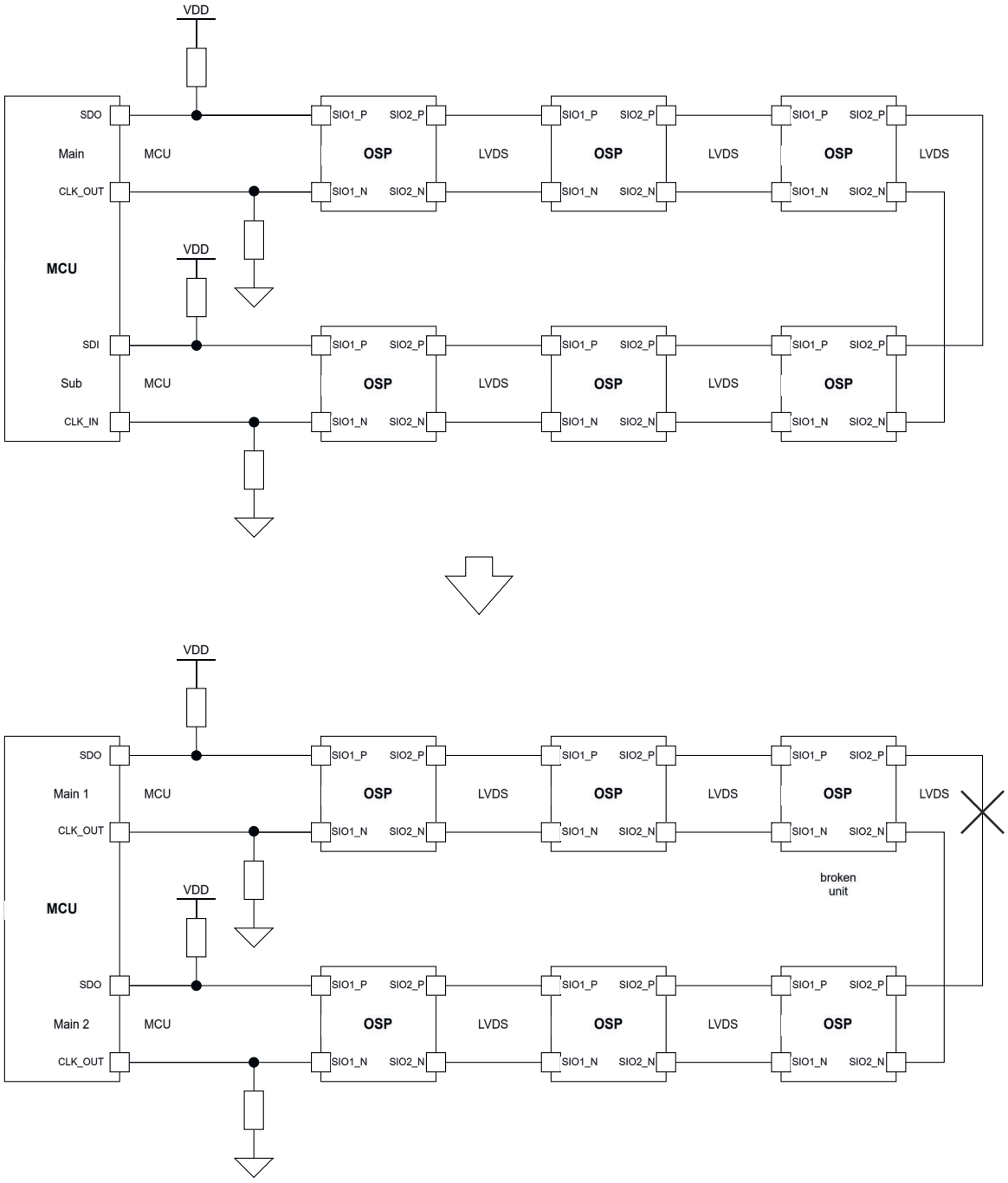
Figure 31: Loop-Back chain with MCU, LVDS and USE modes



Recovery of chain after node defect

The following proposal shown in Figure 32 illustrates how to recover part of the chain after a node or wire connection has failed.

Figure 32: Proposal for a chain configuration which is more robust against node defects.



Starting point is a loop-back chain configuration but using the MCU mode for both ends of the chain (left).

For initialization, the first I/O interface of the MCU is configured as main and sends the INITLOOP command into the chain. The second I/O interface of the MCU is configured as sub and will receive the INITLOOP command with address N+1 where N is the number of nodes in the chain. The last node is not configured as EOL and therefore the MCU sub I/O needs to handle this, respectively.

Assuming a connection somewhere within the chain breaks, communication would no longer be possible along the whole chain and, in the example, the lower part of the chain would no longer be accessible. The MCU will realize this eventually by the lack of response telegrams reaching the sub I/O, for example, from reading the status or temperature information.

To recover this part of the chain, the MCU reconfigures both I/O interfaces as main and uses the INIT command to initialize each section of the chain in bidirectional mode.

As the nodes adjacent to the broken connection are not configured as EOL devices, there will be no response sent back to the MCU. The MCU needs to use a timeout mechanism to wait for the initialization procedure to be completed.

The remaining length of each chain segment (or the location of the broken connection) may be found by iteratively polling some information from consecutive nodes starting from the first one in each segment. The last node answering then gives the remaining chain length.

7 Revision information

Revisions

Version	Release Date	Changes
1.0	2023-07-06	initial revision
1.1	2025-01-29	updated layer model in basic information (sec. 1) updated glossary (sec. 1.1) added conformance levels (sec. 1.2) removed point-to-point topology from sec. 2.1 added startup timing (sec. 2.2.1) added information on collision handling (sec. 2.6) updated electrical and timing parameters (sec. 2.8) updated node identification table (sec. 4.1.3 and sec. 5.1) added multicast feature (sec. 4.1.6 and sec. 4.4) added serial broadcast readout feature (sec. 4.1.7) added information on waiting time between commands (sec. 5.4) added example section (sec. 6)

Revision Control

Version	Approval Date	Company
1.0	2023-07-06	ams OSRAM AG
1.1	2025-01-29	ams OSRAM AG
	2025-01-29	DOMINANT Opto Technologies Sdn. Bhd.

ABOUT ams OSRAM Group (SIX: AMS)

The ams OSRAM Group (SIX: AMS) is a global leader in optical solutions. By adding intelligence to light and passion to innovation, we enrich people's lives. This is what we mean by Sensing is Life. With over 110 years of combined history, our core is defined by imagination, deep engineering expertise and the ability to provide global industrial capacity in sensor and light technologies. Our around 24,000 employees worldwide focus on innovation across sensing, illumination and visualization to make journeys safer, medical diagnosis more accurate and daily moments in communication a richer experience. Headquartered in Premstaetten/Graz (Austria) with a co-headquarters in Munich (Germany), the group achieved over EUR 5 billion revenues in 2021. Find out more about us on https://ams-osram.com

DISCLAIMER

PLEASE CAREFULLY READ THE BELOW TERMS AND CONDITIONS BEFORE USING THE INFORMATION SHOWN HEREIN. IF YOU DO NOT AGREE WITH ANY OF THESE TERMS AND CONDITIONS, DO NOT USE THE INFORMATION.

The information provided in this general information document was formulated using the utmost care; however, it is provided by ams-OSRAM AG or its Affiliates* on an "as is" basis. Thus, ams-OSRAM AG or its Affiliates* does not expressly or implicitly assume any warranty or liability whatsoever in relation to this information, including – but not limited to – warranties for correctness, completeness, marketability, fitness for any specific purpose, title, or non-infringement of rights. In no event shall ams-OSRAM AG or its Affiliates* be liable – regardless of the legal theory – for any direct, indirect, special, incidental, exemplary, consequential, or punitive damages arising from the use of this information. This limitation shall apply even if ams-OSRAM AG or its Affiliates* has been advised of possible damages. As some jurisdictions do not allow the exclusion of certain warranties or limitations of liabilities, the above limitations and exclusions might not apply. In such cases, the liability of ams-OSRAM AG or its Affiliates* is limited to the greatest extent permitted in law.

ams-OSRAM AG or its Affiliates* may change the provided information at any time without giving notice to users and is not obliged to provide any maintenance or support related to the provided information. The provided information is based on special conditions, which means that the possibility of changes cannot be precluded.

Any rights not expressly granted herein are reserved. Other than the right to use the information provided in this document, no other rights are granted nor shall any obligations requiring the granting of further rights be inferred. Any and all rights and licenses regarding patents and patent applications are expressly excluded.

It is prohibited to reproduce, transfer, distribute, or store all or part of the content of this document in any form without the prior written permission of ams-OSRAM AG or its Affiliates* unless required to do so in accordance with applicable law..

* ("Affiliate" means any existing or future entity: (i) directly or indirectly controlling a Party; (ii) under the same direct, indirect or joint ownership or control as a Party; or (iii) directly, indirectly or jointly owned or controlled by a Party. As used herein, the term "control" (including any variations thereof) means the power or authority, directly or indirectly, to direct or cause the direction of the management and policies of such Party or entity, whether through ownership of voting securities or other interests, by contract or otherwise.)

For further information on our products please see the Product Selector and scan this QR Code.

Published by ams-OSRAM AG
Tobelbader Strasse 30, 8141 Premstaetten, Austria

Published by ams-OSRAM AG
Tobelbader Strasse 30,
8141 Premstaetten Austria
Phone +43 3136 500-0
ams-osram.com