MOST

Media **O**riented **S**ystems **T**ransport

Multimedia and Control Networking Technology

MOST High Protocol Specification
Rev 2.3.1
05/2011

MOSTCO CONFIDENTIAL

See page 3 for the terms of disclosure

Legal Notice

COPYRIGHT

LICENSE DISCLAIMER

Nothing on any MOST Cooperation Web Site, or in any MOST Cooperation document, shall be construed as conferring any license under any of the MOST Cooperation or its members or any third party's intellectual property rights, whether by estoppel, implication, or otherwise.

CONTENT AND LIABILITY DISCLAIMER

MOST Cooperation or its members shall not be responsible for any errors or omissions contained at any MOST Cooperation Web Site, or in any MOST Cooperation document, and reserves the right to make changes without notice. Accordingly, all MOST Cooperation and third party information is provided "AS IS". In addition, MOST Cooperation or its members are not responsible for the content of any other Web Site linked to any MOST Cooperation Web Site. Links are provided as Internet navigation tools only.

MOST COOPERATION AND ITS MEMBERS DISCLAIM ALL WARRANTIES WITH REGARD TO THE INFORMATION (INCLUDING ANY SOFTWARE) PROVIDED, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. Some jurisdictions do not allow the exclusion of implied warranties, so the above exclusion may not apply to you.

In no event shall MOST Cooperation or its members be liable for any damages whatsoever, and in particular MOST Cooperation or its members shall not be liable for special, indirect, consequential, or incidental damages, or damages for lost profits, loss of revenue, or loss of use, arising out of or related to any MOST Cooperation Web Site, any MOST Cooperation document, or the information contained in it, whether such damages arise in contract, negligence, tort, under statute, in equity, at law or otherwise.

FEEDBACK INFORMATION

Any information provided to MOST Cooperation in connection with any MOST Cooperation Web Site, or any MOST Cooperation document, shall be provided by the submitter and received by MOST Cooperation on a non-confidential basis. MOST Cooperation shall be free to use such information on an unrestricted basis.

TRADEMARKS

MOST Cooperation and its members prohibit the unauthorized use of any of their trademarks. MOST Cooperation specifically prohibits the use of the MOST Cooperation LOGO unless the use is approved by the Steering Committee of MOST Cooperation.

SUPPORT AND FURTHER INFORMATION

For more information on the MOST technology, please contact:

MOST Cooperation
Administration
Bannwaldallee 48
D-76185 Karlsruhe
Germany

Tel: (+49) (0) 721 966 50 00
Fax: (+49) (0) 721 966 50 01
E-mail: contact@mostcooperation.com
Web: www.mostcooperation.com

This Specification is Confidential Information of the MOST Cooperation. It may only be disclosed to member companies. Member companies wishing to discuss these Specifications with suppliers or other third parties must ensure that a commercially standard form of non-disclosure agreement has been previously executed by the party receiving such Specifications. Use of these Specifications may only be for purposes for which they are intended by the MOST Cooperation. Unauthorized use or disclosure is a violation of law.

Contents

Document References

All documents which this MOST document have references to are listed here with the actual revision this document is referring to.

Number	Document	Revision
[1]	MOST Specification	2.5
[2]	MOST Specification	3.0

Document History

Changes MOST High Protocol Specification Rev. 2.3 to Rev. 2.3.1

Change Ref.	Section	Changes
2V31_001	General	– Clear distinction between “MHP Commands” and “MHP Data” and consistent use of these terms. – Unified spelling of “0-FRAME”. – Consistently use “frame” instead of “telegram” or other variants. – Consistently use “DSO” and “DSI” instead of “sender”, “transmitter”, “receiver”, or similar other terms. – Unified spelling of timer names and retry value names in MSCs and textual description. – Removed RxTxLog column from all MHP commands because addressing between devices is not relevant in that context. – Split FrAck field into high (FrAckH) and low (FrAckL) byte in frame depictions. – Added 0x00 as value for all “Reserved” MHP command fields. – Added missing captions for figures and tables.
2V31_002	Document References	– Added MOST Specification Rev. 3.0.
2V31_003	1	– Removed “Bps” and “Stream” from the Glossary because these are not sufficiently relevant. – Modified and moved glossary entry for “0-FRAME” to the introduction. – Modified description of MHP frame.
2V31_004	2	– New figures for MHP frame and MHP Data frame, block, and packet. – More generic description of the maximum number of usable bytes per MHP Data frame. – Revised description for improved clarity.
2V31_005	3.1	– Modified Figure 3-1: An MHP connection also requires the OPType, marked MHP modules with DSO and DSI; marked figure as example because in a common MHP connection scenario the DSO could represent the FBlock. – Revised basic flow description to improve clarity.
2V31_006	3.1.1	– Removed values for timers in the description because tolerances have been introduced for timers.
2V31_007	4.1	– Revised to remove redundant information. – MHP commands and MHP data separated in two distinct sections. – The FRAME ACK command was renamed to ACKNOWLEDGE, which serves for FRAME ACKNOWLEDGE, BLOCK ACKNOWLEDGE, and NEGATIVE ACKNOWLEDGE. – Modified description of HOLD CONNECTION TX and HOLD CONNECTION RX: continuation is not indicated with this command. – Added optional MaxBlkSize and HoldFlag fields.
2V31_008	4.1.1	– Increased maximum value for NDF to 0x05EC. – Different protocol revisions in the DSO and DSI must not lead to the rejection of a new connection.
2V31_009	4.1.1.2	– Completely revised.
2V31_010	4.1.1.3	– Different protocol revisions in the DSO and DSI must not lead to the rejection of a new connection.
2V31_011	4.1.2	– Increased maximum value for NDFAck to 0x05EC. – Changed RevID value used in the frame to 0x02. – Different protocol revisions in the DSO and DSI must not lead to the rejection of a new connection.
2V31_012	4.1.2.1	– Removed reference to NDFAck limit because it is not relevant here.

Change Ref.	Section	Changes
2V31_013	4.1.2.2	– Different protocol revisions in the DSO and DSI must not lead to the rejection of a new connection.
2V31_014	4.1.2.4	– Completely revised.
2V31_015	4.1.2.5	– Revised description of the performance category. – Modified example so that 500 interrupts per second correspond to an AIR value of 2000.
2V31_016	4.1.2.6	– Added note that points to the appendix for legacy implementations.
2V31_017	4.1.4	– Combined the completely revised descriptions of FRAME ACKNOWLEDGE, BLOCK ACKNOWLEDGE, and NEGATIVE ACKNOWLEDGE under ACKNOWLEDGE.
2V31_018	4.1.5	– Added description of MULTIPLE FRAMES REQUEST command use.
2V31_019	4.1.8	– Added 0x01…0xFE range as “not used”.
2V31_020	4.1.9	– Added 0x00…0xFE range as “not used”.
2V31_021	4.2.1	– FrAck low byte (total number of MHP Data frames in current block) does not include 0-FRAME.
2V31_022	4.2.1.3.1	– Added note that SFA mode is deprecated.
2V31_023	4.2.1.3.2	– Stated that MHP Data frames are transmitted in the order of their frame number.
2V31_024	4.2.2	– For clarity, modified data byte index numbers to start with “1” instead of “0”.
2V31_025	6.1	– Completely revised timeout table, added min./typ./max. values.
2V31_026	7.1.1	– Added reference to appendix for legacy implementations. – Added alternative where the connection is held. – Removed SFA mode.
2V31_027	7.1.2	– Moved description of END CONNECTION TX behavior to section 7.1.5. – Added description of behavior related to START CONNECTION retries. – A new REQUEST CONNECTION TX is an implicit END CONNECTION TX for the previous connection.
2V31_028	7.1.3	– Changed behavior: If the DSO attempts to send 0-length packets and a connection is already open, the attempt is ignored. – Added remark that SFA mode is deprecated. – Current frame counting starts with “1” instead of “0”.
2V31_029	7.1.4	– Changed behavior: If the DSO attempts to send 0-length packets and a connection is already open, the attempt is ignored. – Block count becomes a parameter to the MSC. – Current frame counting starts with “1” instead of “0”. – Modified t_{AIR_Delay} behavior. – Added alternative where $t_{retrans}$ expires.
2V31_030	7.1.5	– Added description of END CONNECTION TX retries and reaction on REQUEST CONNECTION. – MSC revised—considers application requests to end the connection and does not contain a hold phase anymore. – MSC revised—considers regular termination of connection and kill connection.
2V31_031	7.1.6	– Removed reference to “malfunctions” because the scenario regularly occurs as part of a regular transmission.
2V31_032	7.1.7	– New section “HOLD CONNECTION by DSO—using t_{Delay_End}”
2V31_033	7.1.8	– Using NEGATIVE ACKNOWLEDGE instead of FrameAck
2V31_034	7.1.9	– MSC revised—considers $t_{retrans}$ and BLOCK ACKNOWLEDGE; removed use of $t_{receive}$ after a MULTIPLE FRAMES REQUEST.
2V31_035	7.2.3	– The number of NEGATIVE ACKNOWLEDGE retries is limited by r_{negack}.

Change Ref.	Section	Changes
2V31_036	7.2.4	– Renamed MSC from MH_Sc_NoFrameAckSFA to MH_Sc_NoFrameAcknowledge.
2V31_037	7.2.5-7.2.7	– New sections.
2V31_038	7.2.8	– Renamed MSC from MH_Sc_NoFrameAckBA to MH_Sc_BlockAcknowledge. – Modified to fit into MH_Gen_BasicFlow. – Renamed NoTrans to NumTrans.
2V31_039	7.2.9	– Removed statement indicating that $t_{receive}$ is started when an MULTIPLE FRAMES REQUEST is sent. – END_CONNECTION_RX no longer sent after NEGATIVE ACKNOWLEDGE retries.
2V31_040	7.2.10	– Replaced MH_Sc_FrameNAK with more comprehensive MH_Sc_NegativeAcknowledge.
2V31_041		– Removed section “DSO terminates connection”, which was merged into 7.1.5.
2V31_042	8.5	– Added reference to appendix for legacy implementations.
2V31_043	9.1.1	– Added reference to appendix for legacy implementations. – Flow chart can be entered from “Idle” or “Removing Connection”. – The hold phase can be interrupted if data is queued. – Considers priority handling.
2V31_044	9.1.2	– Exit to “Normal Operation” renamed to “Idle”.
2V31_045	9.1.3	– The repeated sending of END CONNECTION TX is interrupted if data is queued. – Exit to “Normal Operation” renamed to “Idle”.
2V31_046	9.2.1	– Rejection of connection and priority check refined.
2V31_047	10	– New appendix.

Changes MOST High Protocol Specification 2V2 to MOST High Protocol Specification 2V3

Change Ref.	Section	Changes
2V3_001	General	– Updated use of MOST terms to match MOST Specification Rev. 2.5 (e.g., Asynchronous Channel becomes Packet Data Channel, Loss of Lock becomes Unlock). – Fixed clerical errors.
2V3_002	1	– Renamed chapter to “Glossary” – Removed use of MOST High over Control Channel, which is not supported any longer. – Removed entries that are already covered by the MOST Specification. – Removed definition of Baud, which is not used in this specification. – Removed redundant and misleading reference to MHP user data max. value.
2V3_003	2	– Added statement that MHP is based on the Packet Data Channel. – Removed reference to Control Channel. – Changed max. number of frames in Block to 255. – Updated Control Data and User Data diagrams to match MOST Specification. – Referenced MOST Specification for details on structure and capacity of the Packet Data Channel. – Clarified that MHP user data always contains the FrAck field, which reduces the usable area to 1006 bytes. – Packet size is not limited by the protocol but by implementation.
2V3_004	-	– Removed redundant chapter “Structure of Control and Asynchronous

Change Ref.	Section	Changes
		Channel".
2V3_005	-	– Removed redundant chapter "Capacity of the Channels".
2V3_006	3.1	– Removed Length field from application message. – Modified diagrams so that controller and FBlock are distinguishable. – Improved description of basic flow. – Removed redundant parts – Removed statement on limitation of packet size.
2V3_007	3.1.1	– New section "Termination of Connections"
2V3_008	4.1	– Removed ADJUST RATE command. – Removed REQUEST command.
2V3_009	4.2.1 – 5.4	– Removed reference to Control Channel. – Removed content from fields whenever it was merely an example. – Changed "High Cmd" to "MHP Cmd" to avoid confusion with High byte. – Added RevID 0x02 for MHP revision 2.3.
2V3_010	4.2.1	– Merged NDFH and NDFL into description of NDF.
2V3_011	4.2.2	– Merged NDFAckH and NDFAckL into description of NDFAck. – Merged AIRH and AIRL into description of AIR. – Added MaxBlkSize field that contains the actual buffer size. – Added RevID 0x02 for MHP revision 2.3.
2V3_012	4.2.2.1	– Number of bytes per block is restricted to 65535, not 65536. – Removed redundant description and 48 byte limitation
2V3_013	4.2.2.5	– Replaced "must" with "should" in "The transmission rate should not exceed the rate that corresponds to AIR."
2V3_014	4.2.2.6	– New section "MaxBlkSize"
2V3_015	4.2.3	– Added description.
2V3_016	4.2.4.3.1	– Described termination of connection in case of exhausted retries more clearly.
2V3_017	4.2.6	– Added optional HoldFlag field to support an implicit HOLD_CONNECTION.
2V3_018	4.2.6.1	– New section "HoldFlag"
2V3_019	4.2.9 4.2.10	– Renamed RSVD to Reserved. – Changed "Priority info" range (0x02...0x7F) to "not used" – Added value 0x80 for hold by application.
2V3_020	4.2.11	– Replaced "connection is terminated since the data is no longer valid" with "kill connection".
2V3_021	5.1	– Changed heading to Reaction on Unlock.
2V3_022	5.2	– New section NetInterface PowerOff
2V3_023	-	– Removed section "Loss of Signal".
2V3_024	5.4	– Added t_{Hold} timeout as additional trigger to continue transmission.
2V3_025	5.5	– Added statement about sender of an open connection using END CONNECTION TX when terminating in System State NotOK. – When a connection is needed after reaching OK again, a new connection has to be established.
2V3_026	6.1	– Explicitly stated that t_{Hold_Resend} refers to the max. value. – HOLD CONNECTION can be sent before t_{Hold_Resend} expires.
2V3_027	7.1.2	– END CONNECTION TX retries can be interrupted by REQUEST CONNECTION TX. – Broadcast addresses are ignored.
2V3_028	7.1.3 7.1.4	– Described behavior when DSO attempts to send packets of length 0.
2V3_029	7.1.5	– If the connection is re-opened, END CONNECTION TX commands are interrupted immediately. – Removed reference to Control Channel.

Change Ref.	Section	Changes
2V3_030	-	– Removed section on Single Frame Request
2V3_031	-	– Removed section on Block Request
2V3_032	7.1.8	– Removed reference to Control Channel.
2V3_033	7.1.9	– Corrected transmission rate adaptation to reflect the MULTIPLE FRAME REQUEST scenario instead of the misleading HOLD CONNECTION RX scenario.
2V3_034	-	– Removed section Adjust Transmission Rate.
2V3_035	8	– Added chapter on details of concurrent connections.
2V3_036	9.1.1 – 9.2.2	– Replaced numeric retry values in diagrams with corresponding names.

Changes MOST High Protocol Specification 2V1 to MOST High Protocol Specification 2V2

Change Ref.	Section	Changes
2V2_001	General	Replaced Bibliography chapter with “Document References” Removed “Chapter 1 Introduction”.
2V2_002	General	Chapter 6 reworked, contains information from old chapter 6 and 9.
2V2_003	General	Old chapter 10 and 12 removed.
2V2_004	General	Old chapter 11 moved to Appendix A.
2V2_005	General	BlckCnt changed to BlockCnt for consistency.
2V2_006	2, 3.1, 6.1	NetServices changed to Network Services.
2V2_007	3.1	Clarified last point in Basic Flow.
2V2_008	3.1	Replaced timer t_{Hold_Max} with $t_{Hold\_Max\_Buf}$
2V2_009	5.1	12 removed from list as it was an error case and this is a successful scenario.
2V2_010	4.2 5.4 5.4	FktID moved to the area of extension functions.
2V2_011	4.2.1	Changed TelLen from 4 to 5 in the “REQUEST CONNECTION” command.
2V2_012	4.2.1.2	Parallel combined mode changed to alternative packet service. “must not support” changed to “may not support”.
2V2_013	4.2.1.3 4.2.2.2	RevID is a new field in “Request Connection” and “Start Connection” command. The RevID field defines which version of MHP that is implemented.
2V2_014	4.2.2.3	Clarification of how a connection is rejected.
2V2_015	4.2.2.5	Added information about transmission rate.
2V2_016	4.2.4.2	SegID is a new field in the “0-Frame”. The purpose of the segment ID is for the receiver to detect the beginning and the ending of a segmented packet.
2V2_017	4.2.4.3	Reworked text that describes acknowledgment modes.
2V2_018	4.2.4.3	Reworked text that describes Block and Single-Frame acknowledgment mode. Updated Figure 6-2.
2V2_019	4.2.4.3	Reworked text that describes Block and Single-Frame acknowledgment mode. Added description of timers and timeouts (t_{trans} and $t_{retrans}$).
2V2_020	5.5	New chapter.
2V2_021	6.1	Removed timers $t_{Hold\_Max\_Appl}$ and $t_{Hold\_Max\_Busy}$ Changed description for $t_{retrans}$.
2V2_022	6.1	Changed description of timers t_{trans} and $t_{retrans}$
2V2_023	6.1 6.2	Updated values for timeouts and retransmissions.
2V2_024	6.1	New timers t_{dwn_NegAck}, $t_{TxSpeedRecovery}$ t_{AIR_Delay} added.
2V2_025	7	New chapter.

Changes MOST High Protocol Specification 2V0 to MOST High Protocol Specification 2V1

Change Ref.	Section	Changes
2V1_001	2.1	“Please note: ..... “ is modified.
2V1_002	general	All MOST High Protocol frames are now illustrated for the control and the asynchronous channel to clarify the difference. Sign “C:” is used to indicate the structure of a frame on the control channel. Sign “A:” is used to indicate the structure on the asynchronous channel.
2V1_003	6.1, 9.1, 9.2.2, 10.1.1.2, 10.2.1.2, 10.2.1.3	Command “Start Connection” contains an additional operand “AIRH/AIRL”.
2V1_004	6.2, 10.1.2.3, 12.2	In case of the Single Frame Acknowledge mode the next frame is not sent by the DSO until the previous frame has been acknowledged by the DSI.
2V1_005	6.2, 9.1, 9.2.10, 10.2.2.5	Additional kind of request: Multiple Frames Request Command: <FF> <ListOfFrameIDs>
2V1_006	6.2, 9.1, 9.2.11	New command to increase (<F0><01>) or decrease (<F0><02>) the transmission rate.
2V1_007	7.1, 10.2.2	Behavior on Loss Of Lock events is modified.

Changes MOST High Protocol Specification 1V0 to MOST High Protocol Specification 2V0

Change Ref.	Section	Changes
1V1_001	6.1.1	New section.
1V1_002	6.2, 9.2.4, 10.1.2.1.1	“0-Frame”, new field: “BlckCnt”
1V1_003	6.2, 9.1, 9.2.6	Command “Frame Acknowledge”, new field: “BlckCnt”
1V1_004	10.2.1.4, 10.2.2.2	Command “Negative Acknowledge”, new field: “BlckCnt”
1V1_005	6.1, 9.1, 9.2.1, 10.1.1.1, 10.1.1.2, 10.2.1.1	Command “Request Connection”, new fields: “NDFH” and “NDFL”
1V1_006	6.1, 9.1, 9.2.2, 10.2.1.2	Command “Start Connection”, new fields: “NDFAckH” and “NDFAckL”

Change Ref.	Section	Changes
1V1_007	1.1, 2.1	The block size is limited to 64kByte. The frame size is not longer limited to 48 byte.
1V1_008	1.2	Figure modified.
1V1_009		“Please note: ...” is added.

1 Glossary

Term	Definition
MOST	Media Oriented Systems Transport.
TCP	Transmission Control Protocol.
MHP Commands	MOST High Protocol (MHP) Commands, e.g., for establishing or terminating connections.
MHP Data	MOST High Protocol (MHP) Data represents the actual data contents of a message.
Packet	Message consisting of m blocks (that are related to each other with respect to their contents), which will be processed by a higher layer. The packet is marked by a defined end. The packet size is not limited by the protocol but by implementation only.
Block	A packet is composed of several blocks. A block consists of a “0-FRAME” and a defined number of “n” MHP Data frames (n= 1 up to 255) and is protected by a securing mechanism. The Block size is limited to 64kByte.
MHP frame	The MHP frame is the smallest unit of transport, which is secured by the Data Link Layer (DLL), which is given by MOST.
DLL	The Data Link Layer secures data transfer on the lowest level by a CRC check on the Packet Data Channel.
DSO	Data source
DSI	Data sink
MSC	Message Sequence Chart

2 Basics

The MOST High Protocol (MHP) is based on the MOST Packet Data Channel. MHP uses some of the established mechanisms of TCP (Transmission Control Protocol). For MHP, the header is reduced to the essential extent, and adapted to the MOST environment. MHP can be used for providing secure data transmission within MOST Networks. It is not suitable for communication with the external world. Nevertheless it could transport TCP secured data within the MOST Network, for example, from and to a telephone.

MHP fits harmoniously into the definition of MOST. This applies to the structure of the software, as well as to the functional way of looking at the application level.

In MHP, two basic types of frames are transported: MHP Commands and MHP Data. They are differentiated by the TelID.

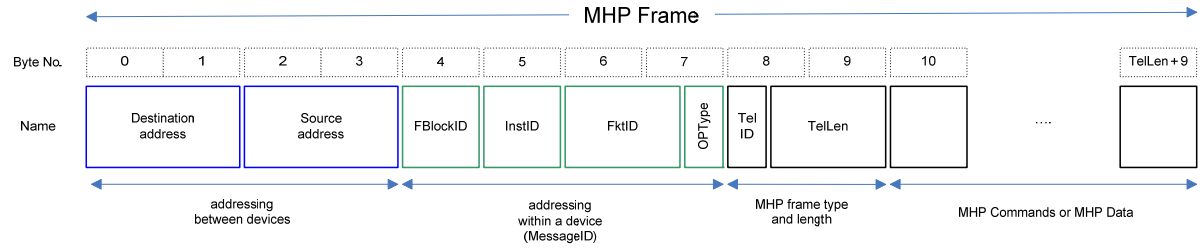

Figure 2-1: Structure of an MHP Frame

A complete MHP frame consists of the destination and source addresses and usable data bytes.

The data area of the MHP frame is reduced by 6 bytes, which are reserved for MessageID, TelID, and TelLen. (refer to [1] and [2]). The usable payload is further reduced for MHP Data frames, which contain the 2-byte FrAck (frame acknowledge) field.

Note: *For details on typical values for the maximum number of usable bytes per MHP Data frame see 10.1 MHP Frame Size in the appendix.*

The 0-FRAME is the MHP Data frame that indicates the start of a new block.

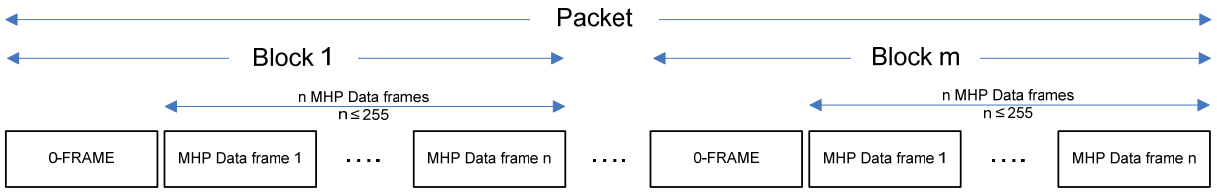

Figure 2-2: MHP Data Frame, Block, and Packet

The packet size is not limited by the protocol but by implementation only.

With the use of the FrAck field, the reliability of transmission and data flow control is increased. The first byte of the FrAck field serves the consecutive numbering of MHP Data frames in a block while the second byte provides the block size (number of MHP Data frames per block). It is possible to send single MHP Data frames or entire blocks with up to 255 MHP Data frames in total.

The maximum number of usable data bytes per MHP Data frame is determined when the connection is established. It depends on the hardware restrictions on Data source (DSO) and on Data sink (DSI) side. In addition, it is possible to tell the DSO, with the help of scaling, how many MHP Data frames per block the DSI can accept.

Since continuation MHP Data frames are sent only if the preceding block was confirmed completely, flooding the DSI with data can be avoided. Single MHP Data frames or all MHP Data frames within a block are confirmed by a separate message, which is transported in an MHP Command frame.

With the help of the "***Options***" field, it is possible to determine whether the DSI shall send the confirmation after each MHP Data frame or after all MHP Data frames within a block.

The number of valid bytes in an MHP frame (specified length) is determined by "***TelLen***".

MHP Data Frame

Data Area MOST Network Interface Controller (16 bit addressing)

16 bits	16 bits	8 bits	8 bits	12 bits	4 bits	4 bits	12 bits	16 bits	8 bits		8 bits
Destination Address	Source Address	FBlock ID	Inst. ID	Fkt ID	OP Type	Tel ID	Tel Len	FrAck	Data	...	Data

Figure 2-3: MHP Data Frame

MHP Data frames have TelID 0x8.

MHP Command Frame

Data Area MOST Network Interface Controller (16 bit addressing)

16 bits	16 bits	8 bits	8 bits	12 bits	4 bits	4 bits	12 bits	8 bits	8 bits		8 bits
Destination Address	Source Address	FBlock ID	Inst. ID	Fkt ID	OP Type	Tel ID	Tel Len	MHP Cmd	Oper-and	...	Oper-and

Figure 2-4: MHP Command Frame

MHP Command frames have TelID 0x9.

For more detailed information about the structure and the capacity of the Packet Data Channel, please refer to [1] and [2].

3 Integration in Software

3.1 Application View

In the communication model (control model) of MOST, a Controller sends an application message, addressing a function in an FBlock. The function replies to its own image located in the Controller. The same model applies to MHP. MHP is only used as alternative transport mechanism for the message in addition to single transfer and segmented transfer on the Control Channel. The Controller delivers the application message in the following form to the transport layer of MHP:

```
DeviceID.FBlockID.InstID.FktID.OPType (Data)
```

MHP will then use MHP Command frames to establish a connection to the device containing the addressed function. Then the data is transmitted by the help of MHP Data frames. During transmission, data flow is controlled by MHP Command frames. If no further messages are to be sent to the same DSI, the connection is closed.

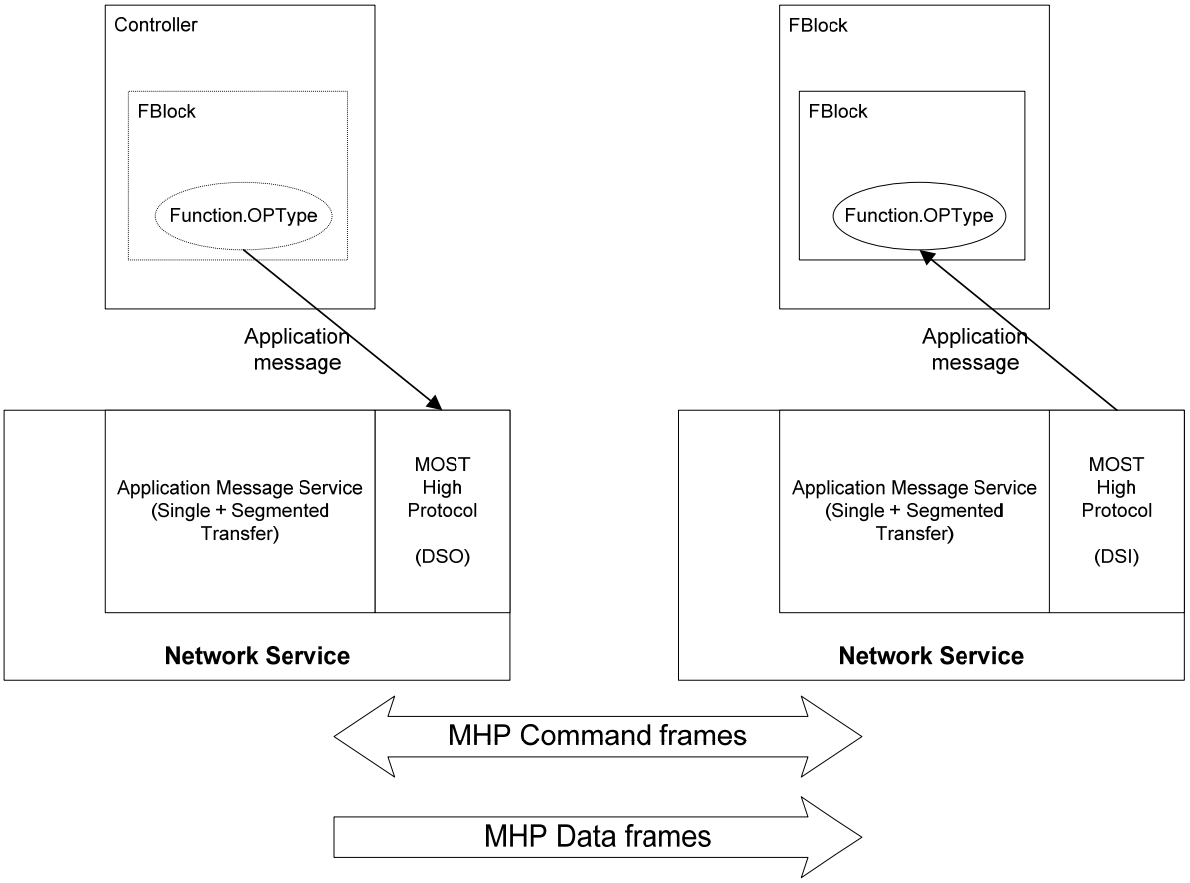

Figure 3-1: Example for the Transmission of an Application Messages with the Help of MHP

Note: *In case of a duplex data transmission, both partners implement such a DSI function and both DSOs establish a connection independent from each other.*

The administration of connections is not transparent to the receiving and the sending application. If an application message shall be sent, the transport layer of the DSO establishes a connection to the transport layer of the DSI. Then it transmits the message (data packet) divided up into blocks. If no new application message is delivered within timeout t_{Delay_End} (from the DSO to the same function in the DSI), the transport layer of the DSO removes the connection.

Basic flow:

1. The sending application passes an application message to MHP.
2. Using MHP, the DSO asks the DSI for a connection.
3. On the DSI side, the MHP module requests a receive buffer from the DSI function.
4. If the DSI function provides a buffer, the DSI—through MHP—confirms the connection. Otherwise, a timeout occurs at the DSO's MHP module, which will then notify the application.
5. If the connection was confirmed, the DSO's MHP module starts the data transmission. Data transmission is organized in blocks, which consist of single MHP Data frames. The DSI's MHP module confirms the correct reception of the data or initiates the repetition of transmissions.
6. If the receive buffer is full although the application message was not transmitted completely, the DSI's MHP module notifies the function of the application and stops transmission by sending HOLD CONNECTION RX. The connection will be kept active until $t_{Hold\_Max\_Buf}$ expires.
   a) After that, the DSI sends END CONNECTION RX and removes the connection.
   b) If the buffer is released before the timeout occurred, the transmission can be continued by sending NEGATIVE ACKNOWLEDGE.
7. Even after transmission of the application message is complete, the connection stays open until a t_{Delay_End} timeout occurs or it is explicitly closed by the application. The DSO may keep the connection open so that it can handle a potential successive transmission.
8. If the connection is still open after t_{Delay_End}, the connection is removed by END CONNECTION TX.

3.1.1 Termination of Connections

MHP connections are terminated under the following conditions:

1. A timeout occurs on the DSO side. No END CONNECTION TX command is sent.

   For example, this scenario applies when a timeout occurs after $(r_{request} + 1) * t_{send}$ while trying to build a connection (see *6.1 Timeouts* , *6.2 Retries*).

   In other cases, a timeout may occur when trying to send a data block, which is not acknowledged by the DSI, after $(r_{trans} + 1) * t_{trans}$.

   Other reasons for this kind of timeout may be implementation-specific. For example, a buffer may be full or an I/O interface might operate outside the nominal range.

2. A timeout occurs on the DSI side. No END CONNECTION RX command is sent.

   For example, this scenario applies when a timeout occurs after $(r_{start} + 1) * t_{ready}$ while trying to confirm a new connection.

   Other timeouts may occur when waiting for the 0-FRAME after $(r_{negack} + 1) * t_{frame}$ or when receiving MHP Data frames after $(r_{negack} + 1) * t_{receive}$.

3. If the DSO closes a connection, it has to perform retries of the END CONNECTION TX command. Retrying END CONNECTION TX commands is interrupted immediately whenever the connection needs to be re-opened (by REQUEST CONNECTION command). The DSI shall interpret a new REQUEST CONNECTION as implicit END CONNECTION TX command if it failed to receive the previous END CONNECTION TX.

4. If the DSI kills a connection, it sends only a single END CONNECTION RX command (without retries).

4 MHP Commands and MHP Data

4.1 MHP Commands

MHP Command	Name	Description
0xCA <Prio><NDFH> <NDFL><RevID>	REQUEST CONNECTION	Indicates the existence of data, the priority of connection, the maximum number of usable data bytes per MHP Data frame, and the revision number of MHP, which is provided by the DSO.
0xF2 <Scale><RevID> <PrioAck><NDFAckH> <NDFAckL><AIRH> <AIRL> (<MaxBlkSize >)	START CONNECTION	Command for establishing communication. Contains statements about the capacity of the DSI, MHP revision number of the DSI, the priority of connection, the resulting number of usable data bytes per MHP Data frame and the performance category of the receiving device.
0xF3 <Event> **0xFC** <Event>	END CONNECTION TX (from the DSO) END CONNECTION RX (from the DSI)	Terminates a connection between two devices.
0xFA <FrAck> <BlockCnt> (<HoldFlag>)	ACKNOWLEDGE	Acknowledge or negative acknowledge of the DSI about expected MHP Data frame or block.
0xFF <ListOfFrameIDs>	MULTIPLE FRAMES REQUEST	Request for several MHP Data frames of the current block.
0xFD	READY FOR DATA	Answer on the command for establishing connection and acknowledge of the connection to be established.
0xF1 <Event> **0xFE** <Event>	HOLD CONNECTION TX (from the DSO) HOLD CONNECTION RX (from the DSI)	Indicates the interruption of the communication. The Event field transports the kind of interruption.

Table 4-1: Overview of MHP Commands

4.1.1 REQUEST CONNECTION

FBlock ID	Inst ID	Fkt ID	OP Type	Tel ID	Tel Len	MHP Cmd	Prio	NDFH	NDFL	RevID
				0x9	**0x005**	**0xCA**	**aa**	**bb**	**cc**	**0x02**

Operand	Description	Range
Prio	Priority of connection	0x01...0x7F
NDF	Maximum number of usable data bytes per MHP Data frame that can be sent by the DSO. NDFH - High byte of NDF NDFL - Low byte of NDF	0x0028...0x05EC
RevID	Protocol revision: Different protocol revisions in the DSO and DSI must not lead to the rejection of a new connection.	0x00 - MHP revision 2.1 or older 0x01 - MHP revision 2.2 0x02 - MHP revision 2.3/2.3.1 [0x03...0xFF] - reserved

4.1.1.1 Prio

If two devices are trying to send data at the same time, the priority specifies which of the two is more important. Definition and administration of priorities is the task of the application. For avoiding the lock during the establishing of a connection, the default value for the priority must be set to 1, which is the lowest priority.

4.1.1.2 NDF

The NDF field contains the maximum MHP Data frame size that can be sent by the DSO. NDF is set depending on the restriction by hardware or the driver layer, which may not support the possible MHP Data frame size.

Note: *For details on typical NDF values (maximum number of usable bytes per MHP Data frame) see 10.1 MHP Frame Size in the appendix.*

4.1.1.3 RevID

The RevID field defines which version of MHP is implemented in the DSO. Different protocol revisions in the DSO and DSI must not lead to the rejection of a new connection.

4.1.2 START CONNECTION

FBlock ID	Inst ID	Fkt ID	OP Type	Tel ID	Tel Len[1]	MHP Cmd	Scale	RevID	Prio Ack	NDF AckH	NDF AckL	AIRH	AIRL	MaxBlk Size
				0x9	**0x00A**	**0xF2**	**xx**	**0x02**	**aa**	**bb**	**cc**	**dd**	**ee**	**ssss**

Operand	Description	Range
Scale	Maximum number of MHP Data frames per block	0x01..0xFF (Scale * NDFAck ≤ 65535)
RevID	Protocol revision: Different protocol revisions in the DSO and DSI must not lead to the rejection of a new connection.	0x00 - MHP revision 2.1 or older 0x01 - MHP revision 2.2 0x02 - MHP revision 2.3 [0x03...0xFF] - reserved
PrioAck	Priority of connection (acknowledge by the DSI)	0x01…0x80 (PrioAck ≥ Prio)
NDFAck	Maximum number of usable data bytes per MHP Data frame. NDFAckH - High byte of NDFAck NDFAckL - Low byte of NDFAck	0x0028...0x05EC (NDFAck ≤ NDF) (Scale * NDFAck ≤ 65535)
AIR	Performance category of the DSI (average interrupt rate) AIRH - High byte of AIR AIRL - Low byte of AIR	full range
MaxBlkSize	Optional 16 bit value, containing the actual size of the receive buffer on block level.	MaxBlkSize ≥ Scale * NDFAck MaxBlkSize < (Scale + 1) * NDFAck

4.1.2.1 Scale

By Scale, the DSI indicates to the DSO how many MHP Data frames per block it can receive.
In case the optional parameter MaxBlkSize is not transmitted, the Scale parameter together with NDFAck indicates the maximum block size that can be received.
In case the optional parameter MaxBlkSize is transmitted but not supported by the DSO, the Scale parameter together with NDFAck is used to determine the maximum block size.
Because the number of bytes per block is restricted to 65535, the product of Scale and NDFAck must not exceed that limit.

The number of MHP Data frames per block (Scale) is limited to 255.
The number of bytes per block is restricted to 65535 (64kByte).
That means: **Scale * NDFAck ≤ 65535**

Note: *If the optional parameter MaxBlkSize is transmitted, the size of the DSI's buffer is contained in MaxBlkSize.*

[1] If the optional MaxBlkSize field is not transmitted, TelLen is 0x008.

4.1.2.2 RevID

The RevID field defines which version of MHP is implemented in the DSI. Different protocol revisions in the DSO and DSI must not lead to the rejection of a new connection.

4.1.2.3 PrioAck

In the PrioAck field, the received priority number is returned by the DSI, if no message with higher priority is pending. The priority check considers pending connections to the same object (referring to FBlockID.InstID.FktID.OPType) in the DSI.

If a message with higher priority is pending, the DSI rejects the connection. This is done by inserting the higher priority value into the PrioAck field.

4.1.2.4 NDFAck

The NDFAck field contains the maximum MHP Data frame size of the connection. It is restricted by the maximum supported MHP Data frame size of the DSO (NDF in REQUEST CONNECTION) and the DSI.
That means: **NDFAck ≤ NDF**

All MHP Data frames (except the last MHP Data frame per block) must use the maximum number of Data Bytes indicated by the NDFAck value.
MHP Data frames using a length that is not equal to the NDFAck value shall be discarded. Exceptions are allowed only for the last MHP Data frame.

Note: *For details on typical NDFAck values (maximum number of usable bytes per MHP Data frame) see 10.1 MHP Frame Size in the appendix.*

4.1.2.5 AIR

By AIR (Average Interrupt Rate) the DSI reports its performance category. The performance category determines how many MHP Data frames the DSI can receive per second.

If this value is set to 0, the transmission rate depends just on the capability of the DSO. If this value is set greater than 0, the DSO is able to reduce the transmission rate to avoid unacknowledged MHP Data frames. The transmission rate should not exceed the rate that corresponds to AIR.

Example: The interrupt handler in the DSI is able to service about 500 RX interrupts per second. In that case the AIR is set to 2000.

4.1.2.6 MaxBlkSize

This optional field contains the actual size of the receive buffer.

Note: *For legacy implementations (which refers to MOST25 and MOST50 devices based on MOST Specifications with Major Revision 2) please see 10.2 Miscellaneous Requirements in the appendix.*

4.1.3 READY FOR DATA

Answer on the command for establishing connection and acknowledge of the connection to be established.

FBlock ID	Inst ID	Fkt ID	OP Type	Tel ID	Tel Len	MHP Cmd
				0x9	**0x001**	**0xFD**

Operand	Description	Range
(none)	--	--

4.1.4 ACKNOWLEDGE

The ACKNOWLEDGE command is sent by the DSI. It is used in the form of BLOCK ACKNOWLEDGE in BA (block acknowledge) mode or FRAME ACKNOWLEDGE in SFA (single frame acknowledge) mode to confirm the successful transmission of a block or frame, respectively. The structure of BLOCK ACKNOWLEDGE is identical to FRAME ACKNOWLEDGE.

ACKNOWLEDGE is used as NEGATIVE ACKNOWLEDGE by the DSI in BA and SFA mode to request the transmission of the indicated block.

4.1.4.1 BLOCK ACKNOWLEDGE

BLOCK ACKNOWLEDGE confirms the reception of an MHP block and is only sent after all MHP Data frames have been received.

Apart from a regular BLOCK ACKNOWLEDGE, an MHP block can be acknowledged implicitly by a NEGATIVE ACKNOWLEDGE. This is the case when the DSO, after re-sending the 0-FRAME of a block that was not acknowledged, receives a NEGATIVE ACKNOWLEDGE with a BlockCnt that is one greater than the current BlockCnt of the DSO (see Figure 7-20: MH_Sc_NegativeAcknowledge).

FBlock ID	Inst ID	Fkt ID	OP Type	Tel ID	Tel Len[1]	MHP Cmd	FrAckH	FrAckL	Block Cnt	Hold Flag
				0x9	**0x005**	**0xFA**	**nn**	**nn**	**yy**	**hh**

Operand	Description	Range
FrAck	In BLOCK ACKNOWLEDGE, the High Byte (FrAckH) and the Low Byte (FrAckL) of the FrAck parameter have the same value. This value corresponds to the "total number of MHP Data frames per block".	High byte: 0x01...0xFF Low byte: 0x01...0xFF
BlockCnt	ID of current block.	full range
HoldFlag	Bit 0 = True: implicit HOLD CONNECTION RX. Bit 0 = False: The DSO is allowed to transmit the next block. Bit 1-7: reserved	0x00…0x01

[1] If the optional HoldFlag parameter is not transmitted, TelLen contains 0x004.

4.1.4.1.1 HoldFlag

The HoldFlag is an optional field. It provides the possibility to issue an implicit HOLD CONNECTION RX command when a block is acknowledged. This reduces protocol overhead and bus load because the DSI does not have to send a HOLD CONNECTION RX command on reception of the next 0-FRAME.

If either the HoldFlag is not present, or the HoldFlag Bit 0 = False, or the DSO does not support the HoldFlag, the DSO is allowed to transmit the next block.

Note: *For legacy implementations (which refers to MOST25 and MOST50 devices based on MOST Specifications with Major Revision 2) please see 10.2 Miscellaneous Requirements in the appendix.*

4.1.4.2 FRAME ACKNOWLEDGE

FRAME ACKNOWLEDGE is sent after every received MHP Data frame, including the 0-FRAME.

FBlock ID	Inst ID	Fkt ID	OP Type	Tel ID	Tel Len[1]	MHP Cmd	FrAckH	FrAckL	Block Cnt	Hold Flag
				0x9	**0x005**	**0xFA**	**nn**	**xx**	**yy**	**hh**

Operand	Description	Range
FrAck	High byte (FrAckH): Total number of MHP Data frames in the current block Low byte (FrAckL): ID of received MHP Data frame	High byte: 0x01...0xFF Low byte: 0x00...0xFF
BlockCnt	ID of current block.	full range
HoldFlag	Bit 0 = True: implicit HOLD CONNECTION RX. Bit 0 = False: The DSO is allowed to transmit the next block. Bit 1-7: reserved	0x00…0x01

4.1.4.2.1 HoldFlag

The HoldFlag is an optional field. It provides the possibility to issue an implicit HOLD CONNECTION RX command when a block is acknowledged (last FRAME ACKNOWLEDGE). This reduces protocol overhead and bus load because the DSI does not have to send a HOLD CONNECTION RX command on reception of the next 0-FRAME.

If either the HoldFlag is not present, or the HoldFlag Bit 0 = False, or the DSO does not support the HoldFlag, the DSO is allowed to transmit the next block.

Note: *For legacy implementations (which refers to MOST25 and MOST50 devices based on MOST Specifications with Major Revision 2) please see 10.2 Miscellaneous Requirements in the appendix.*

4.1.4.3 NEGATIVE ACKNOWLEDGE

NEGATIVE ACKNOWLEDGE by the DSI requests the transmission of a block and is used to

- indicate the failed transmission of a block in BA and SFA mode
- indicate an implicit BLOCK ACKNOWLEDGE in BA mode
- continue transmission after HOLD CONNECTION RX in BA and SFA mode

FBlock ID	Inst ID	Fkt ID	OP Type	Tel ID	Tel Len	MHP Cmd	FrAckH	FrAckL	Block Cnt
				0x9	**0x004**	**0xFA**	**0x00**	**0x00**	**yy**

Operand	Description	Range
FrAck	High byte (FrAckH): Zero on Negative Acknowledge. Low byte (FrAckL): Zero on Negative Acknowledge.	High byte: 0x00 Low byte: 0x00
BlockCnt	ID of current block.	full range

4.1.5 MULTIPLE FRAMES REQUEST

In general, a MULTIPLE FRAMES REQUEST contains all FrameIDs that were not received yet. Optionally, the DSI may limit the first sent MULTIPLE FRAMES REQUEST to those FrameIDs that are lower than the highest received FrameID. As a consequence, if no DATA FRAME was received yet, no MULTIPLE FRAMES REQUEST is sent on the first timeout of t_{MFR}.

The triggers for a MULTIPLE FRAMES REQUEST are detailed in sections 7.2.7 and 7.2.8. Apart from sending a MULTIPLE FRAMES REQUEST after timeout or after the last frame, the DSI may optionally send a MULTIPLE FRAMES REQUEST earlier, as soon as loss of data frames is detected.

FBlock ID	Inst ID	Fkt ID	OP Type	Tel ID	Tel Len	MHP Cmd	ListOfFrameIDs
				0x9	**n+1**	**0xFF**	**f1 f2 f3 ... fn**

Operand	Description	Range
ListOfFrameIDs	List of requested MHP Data frames. Each requested MHP Data frame reserves one byte. The number of requested MHP Data frames is limited to 41.	
FrameID	Requested MHP Data frame	0x01…0xFF

4.1.6 HOLD CONNECTION TX

FBlock ID	Inst ID	Fkt ID	OP Type	Tel ID	Tel Len	MHP Cmd	Reserved	Event
				0x9	**0x003**	**0xF1**	**0x00**	**xx**

Operand	Description	Range	
Event	Indicates the reason for hold state.	0x00...0x7F	Not used
		0x80	The connection is forced to be in hold state by application (DSO).
		0x81...0x82	Not used
		0x83	Data has been transmitted successfully. Connection is kept open, but no new data is available at the moment.
		0x84	The TX section is servicing another connection.
		0x85...0xFF	Not used

4.1.7 HOLD CONNECTION RX

FBlock ID	Inst ID	Fkt ID	OP Type	Tel ID	Tel Len	MHP Cmd	Reserved	Event
				0x9	**0x003**	**0xFE**	**0x00**	**xx**

Operand	Description	Range	
Event	Indicates the reason for hold state.	0x00…0x7F	Not used
		0x80	The connection is forced to be in hold state by application (DSI).
		0x81	0-FRAME has been received, but the RX section is servicing another connection at the moment.
		0x82	0-FRAME has been received, no other connection is serviced, but the receive buffer has not yet been established.
		0x83...0xFF	Not used

4.1.8 END CONNECTION TX

FBlock ID	Inst ID	Fkt ID	OP Type	Tel ID	Tel Len	MHP Cmd	Reserved	Event
				0x9	**0x003**	**0xF3**	**0x00**	**xx**

Operand	Description	Range	
Event	Indicates the reason for termination.	0x00	Regular termination of connection
		0x01…0xFE	Not used
		0xFF	Kill connection

4.1.9 END CONNECTION RX

FBlock ID	Inst ID	Fkt ID	OP Type	Tel ID	Tel Len	MHP Cmd	Reserved	Event
				0x9	**0x003**	**0xFC**	**0x00**	**FF**

Operand	Description	Range	
Event	Indicates the reason for termination.	0x00…0xFE 0xFF	Not used Connection is killed by the DSI.

4.2 MHP Data

4.2.1 0-FRAME

FBlock ID	Inst ID	Fkt ID	OP Type	Tel ID	Tel Len	FrAckH	FrAckL	SegID	Options	Block Cnt
				0x8	**0x005**	**0x00**	**nn**	**0x00**	**yy**	**zz**

Operand	Description	Range
FrAck	High byte (FrAckH): ID of current MHP Data frame (FrameID) (0-FRAME) Low byte (FrAckL): Total number of MHP Data frames (excluding the 0-FRAME) in the current block	High byte 0x00 Low byte 0x01...0xFF
SegID	Segment ID: The purpose of the Segment ID is for the DSI to detect the beginning and the ending of a possibly segmented packet.	0x00 The packet contains only 1 block, or the DSO does not support segment IDs (downward compatibility). 0x01 First block of a segmented transmission. 0x02 Middle block of a segmented transmission. 0x03 Last block of a segmented transmission.
Options	Options of transmission	Bit 0-1: 0x01 Acknowledge after transmission of the entire block (BlockAck mode). 0x02 Acknowledge after transmission of each single MHP Data frame (single frame acknowledge mode). Bit 2-7: Reserved.
BlockCnt	ID of current block. 0 on first block. It is incremented on each new block.	full range

4.2.1.1 FrAck

The 0-FRAME (MHP Data frame that indicates the start of a block) is indicated by the value 0x00nn in the FrAck field where nn is the total number of MHP Data frames in this block (0x00nn = MHP Data frame 0 of nn).

4.2.1.2 SegID

The SegID parameter was introduced in MHP Rev. 2.2. With respect to downward compatibility, SegID was added to a field that was reserved in previous versions of MHP. When a new DSI (MHP Rev. 2.2 or later) will assemble blocks from an old DSO (MHP Rev. 2.1), that does not support this additional segmentation information, the behavior of the DSI will be the same as in MHP Rev. 2.1. It could be treated like flat segmentation (no end of packet information within an open connection cycle).

↓Start of connection End ↓ ↓ Start

Transmitted Info ?									
No	Packet #	0				1			0
No	Block #	0	1	2	3	0	1		0
Yes	MHP Data Frame #	0 1 2 3	0 1 2 3	0 1 2 3	0 1 2	0 1 2 3	0 1		0 1 2
Yes	Total (Data-) Frames/Block	3	3	3	2	3	1		2
Yes	BlockCnt	0	1	2	3	4	5		0
Yes	SegID	1	2	2	3	1	3		0
No	Block # Calculated on Rx side (MHP layer)	0	1	2	3	0	1		0
No	Packet # Calculated on Rx side (MHP layer)	0				1			0

Table 4-2: Segmentation on MHP

4.2.1.3 Options

4.2.1.3.1 Single Frame Acknowledge (SFA) Mode

Note: *single frame acknowledge (SFA) mode is deprecated!*

In case of SFA mode, each MHP Data frame is acknowledged individually by the DSI. The next MHP Data frame is not sent by the DSO until the previous frame has been acknowledged by the DSI. If the DSO has not received the acknowledgment of a transmitted MHP Data frame within $t_{retrans}$, the MHP Data frame is resent to the DSI. An acknowledged MHP Data frame is never resent by the DSO. The DSI acknowledges correctly received MHP Data frames by including the sequence number of the received MHP Data frame in the low byte of the FrAck field. The high byte transports the total number of MHP Data frames expected in the block. The acknowledgment is then sent to the DSO. By this a reliable transmission is guaranteed.

Whenever the transmission of a block is initiated (first or proximate block within a packet), the timer t_{trans} is started. If t_{trans} timeouts before all MHP Data frames in a block have been acknowledged, the transmission will start over with the unacknowledged MHP Data frame and t_{trans} is restarted. Already acknowledged MHP Data frames are never resent. There will be at most r_{trans} retries to send the remaining unacknowledged MHP Data frames in a block. The connection will be terminated without sending END CONNECTION TX if one MHP Data frame in a block has not been acknowledged after r_{trans} retries.

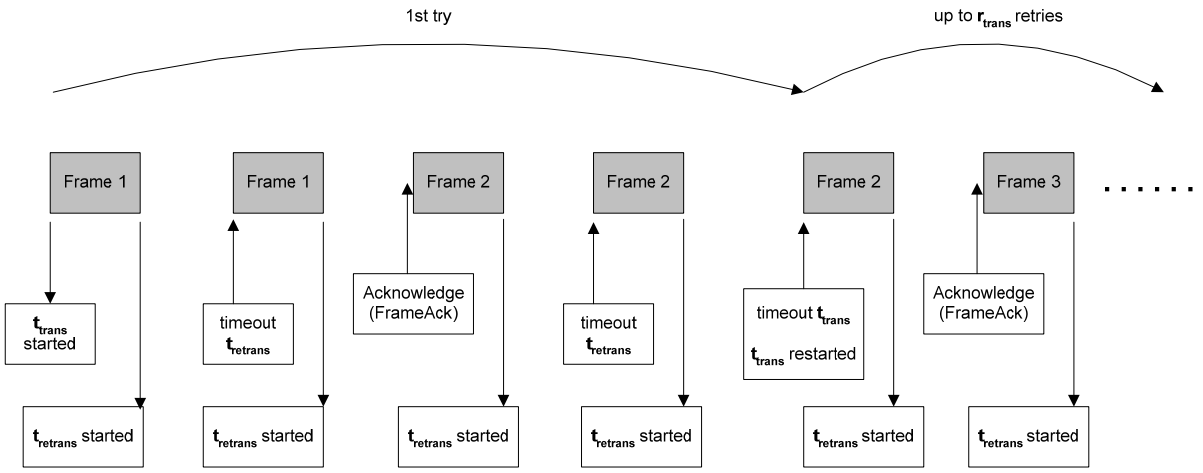

Figure 4-1: Timers, Timeouts and Retransmissions in SFA-Mode

4.2.1.3.2 Block Acknowledge (BA) Mode

In case of block acknowledge mode, only the last MHP Data frame in each block is acknowledged by the DSI. The DSO sends all MHP Data frames in a sequence and waits for the DSI to acknowledge the last MHP Data frame in the block. When the last MHP Data frame is acknowledged all MHP Data frames in the block are considered acknowledged. The next block is not sent by the DSO until the previous block has been acknowledged by the DSI. If the DSO has not received the acknowledgment of a sent block within $t_{retrans}$, all MHP Data frames in the block are resent to the DSI. When the DSI receives the last MHP Data frame in the block, the entire block is acknowledged by including the sequence number of the received (last) MHP Data frame in the low byte of the FrAck field. The high byte transports the total number of MHP Data frames expected in the block. The acknowledgment is then sent to the DSO.

Whenever the transmission of a block is initiated (first or proximate block within a packet), the timer t_{trans} is started. If t_{trans} timeouts before the block has been acknowledged, the complete block will be resent and t_{trans} restarted. There will be at most r_{trans} retries to resend a block. The connection will be terminated (without transmitting END CONNECTION TX) if a block has not been acknowledged after r_{trans} retries.

MHP Data frames are transmitted in the order of their frame number, that is, the MHP Data frame with the lowest number in the remaining set is transmitted first.

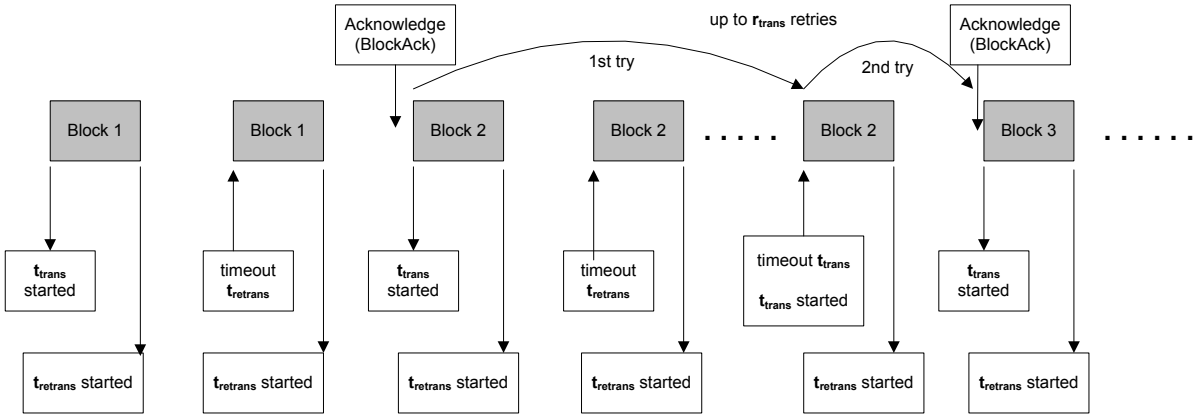

Figure 4-2: Timers, Timeouts and Retransmissions in BA-Mode

4.2.1.4 Block Counter (BlockCnt)

Whenever a connection is established, the DSO and the DSI set the block counter to zero. This block counter is incremented on each new block. It is transmitted in the first MHP Data frame of each block (0-FRAME). It is also used, whenever the DSI acknowledges single MHP Data frames or whole blocks. When the block counter reaches 0xFF, it starts again at 0x00.

4.2.2 DATA FRAME

FBlock ID	Inst ID	Fkt ID	OP Type	Tel ID	Tel Len	FrAckH	FrAckL	Data	Data	Data	...	Data
				0x8	**m+2**	**xx**	**nn**	**D1**	**D2**	**D3**		**Dm**

Operand	Description	Range
FrAck	High byte (FrAckH): ID of current MHP Data frame (FrameID) Low byte (FrAckL): Total number of MHP Data frames in the current block	High byte: 0x01...0xFF Low byte: 0x01...0xFF

5 System Dependent Behavior on Error

5.1 Reaction on Unlock

In consideration of data transmission via MHP it is not necessary to react on an Unlock event, because the single MHP frames are CRC protected by the Data Link Layer on the Packet Data Channel. The defined retry mechanisms and the possibility to request the missed MHP Data frames provide a reliable transmission of data even in case of “short” Unlock events.

In case of a Critical Unlock (please refer to [1] and [2]), the network is shut down. On the system’s re-start, MHP is re-initialized and the transmission must be restarted.

5.2 NetInterface Off

If the state of the NetInterface changes from Normal Operation to Off (the modulated signal is switched off), all MHP connections have to be reset. No END CONNECTION command can be exchanged since no MOST communication is possible in NetInterface Off.

5.3 Low Voltage

Caused by different low voltage thresholds, it is not clear at which time the single devices stop communicating. After the low voltage event, or the system's restart, MHP must be initialized, and the transmission must be restarted.

5.4 Transmission during HOLD CONNECTION

If the receiving node sends a HOLD CONNECTION RX caused by a malfunction during transmission, the rest of the MHP Data frames are not sent; this keeps bus load low. After the first missing of HOLD CONNECTION RX, the transmission is continued after a timeout of t_{Hold} or after a Negative Acknowledge is received for this connection.

FBlock ID	Inst ID	Fkt ID	OP Type	Tel ID	Tel Len	MHP Cmd	Reserved	Event
				0x9	**0x003**	**0xFE**	**0x00**	**yy**

5.5 System State NotOK

Application communication is forbidden in System State NotOK. In case of Configuration.Status(NotOK), all connections are considered terminated. However, the DSO may transmit END CONNECTION TX.

If the application requires the use of the MHP connection after the system has reached System State OK again, a new connection has to be established.

6 Timeouts and Retries

6.1 Timeouts

Timeout	Value [ms]			Description
	min.	typ.	max.	
t_{send}	50	100	150	Timeout between each attempt when the DSO is building the connection with the REQUEST CONNECTION command.
t_{trans}	2500	3000	3500	Maximum time of one attempt, when the DSO tries to transmit one block.
t_{end}	50	100	150	Timeout between each attempt when closing a connection with the END CONNECTION command.
t_{ready}	50	100	150	Timeout between each attempt when the DSI is confirming the connection with the START CONNECTION command.
t_{frame}	150	200	250	Timeout between each attempt when the DSI is waiting for a 0-FRAME.
$t_{receive}$	150	200	250	Timeout for reception of MHP Data frames.
t_{Hold}	650	700	800	Timeout after receiving a HOLD CONNECTION command.
t_{Hold_Resend}	0	500	550	Maximum interval of sending a HOLD CONNECTION command. The DSO or the DSI may transmit a HOLD CONNECTION command before t_{Hold_Resend} expires.
t_{Delay_End}	5500	6000	∞	Delay of the DSO before closing the connection, if no application request to close the connection exists. During this time, the DSO is waiting for further packets from the application.
$t_{Hold\_Max\_Buf}$	11000	12000	13000	Maximum time of each hold cycle whenever data buffer is locked by application.
$t_{retrans}$	0	50	100	**SFA Mode:** Timeout between each attempt to retransmit an unacknowledged MHP Data frame. If a sent MHP Data frame not acknowledged within $t_{retrans}$, the frame is resent. This value is valid only for single frame acknowledge mode.
	50	200	250	**BA Mode:** Timeout for reception of BLOCK ACKNOWLEDGE or “implicit block acknowledge” (see 4.1.4.1). If sent block not acknowledged within $t_{retrans}$ the block is resent. This value is valid only for block acknowledge mode.
t_{dwn_NegAck}	10	200	250	Delay between retries of NEGATIVE ACKNOWLEDGE after received MHP Data frames without 0-FRAME.
$t_{TxSpeedRecovery}$	50	100	150	Timeout between each transmission rate increase in block acknowledge mode.

Timeout	Value [ms]			Description
	min.	typ.	max.	
t_{AIR_Delay}	0		25	Average Interrupt Delay This delay is set to AIR received from the DSI. If AIR = 0 or smaller than AIR in the DSO, the delay is set by the DSO. Section 7.1.10 describes how the delay can be adapted during transmission.
t_{mfr}	0	50	100	Timeout in the DSI while receiving MHP Data frames. On timeout, a MULTIPLE FRAMES REQUEST is triggered.
t_{mfr_retry}	0	50	100	Timer in the DSI between transmission of consecutive MULTIPLE FRAMES REQUEST.

Table 6-1: Timeouts

6.2 Retries

Retry	Value	Description
$r_{request}$	4	Retries of the DSO when establishing a connection (REQUEST CONNECTION command).
r_{end}	4	Retries of the DSO closing the connection.
r_{trans}	2	Retries of the DSO during data transmission.
r_{start}	4	Retries of the DSI confirming the connection.
r_{negack}	8	Retries of the DSI sending a NEGATIVE ACKNOWLEDGE.

Table 6-2: Retries

7 Dynamic Requirements

This section describes the requirements for the dynamic behavior of MHP.

7.1 General MSCs

This section describes the general flow of MHP when used successfully. Error handling and special cases are described in 7.1.10.

7.1.1 Basic Flow

This MSC describes the basic flow when an application transmits data from DSO to DSI through MHP.

Note: *For legacy implementations (which refers to MOST25 and MOST50 devices based on MOST Specifications with Major Revision 2) please see 10.2 Miscellaneous Requirements in the appendix.*

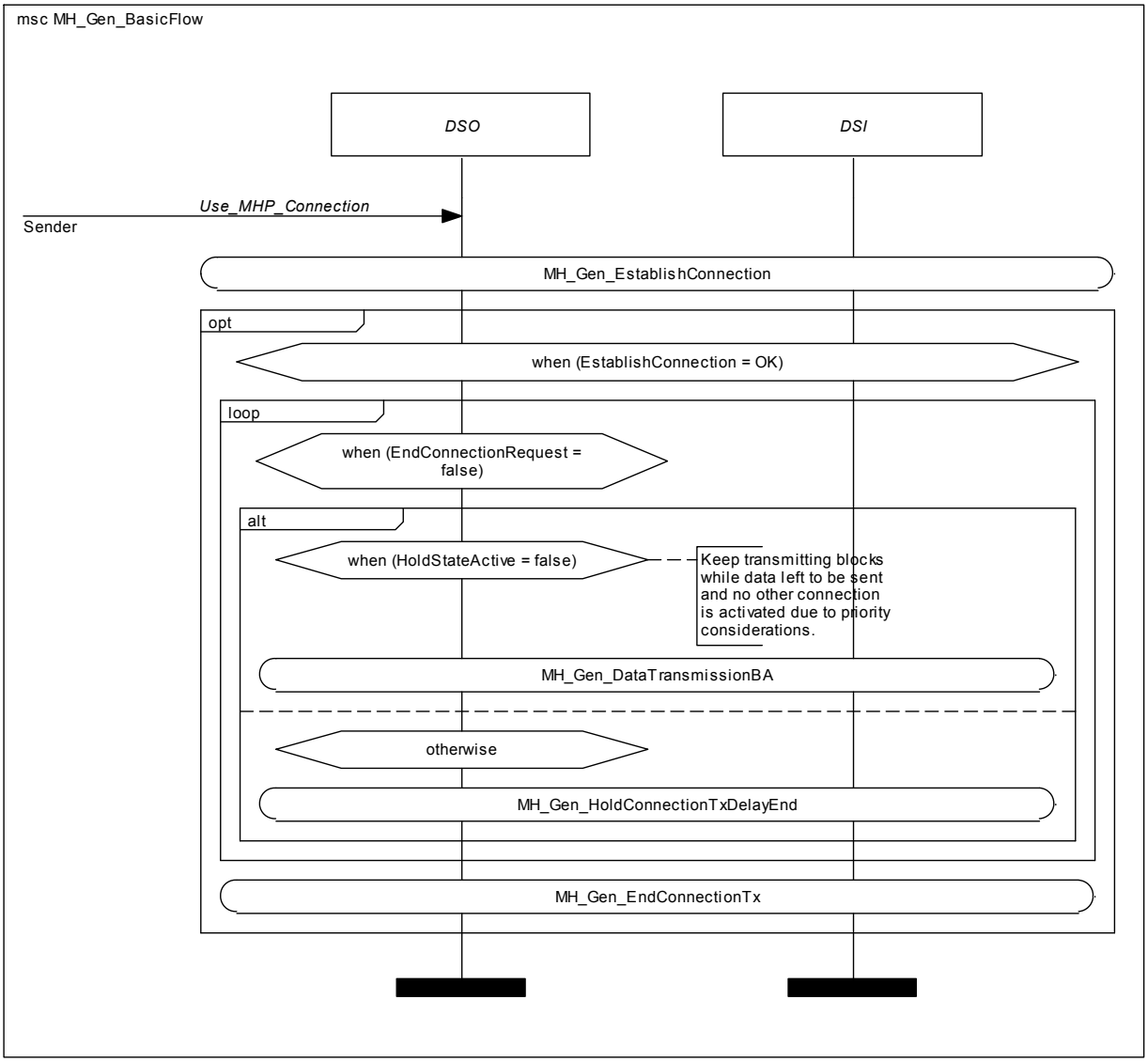

Figure 7-1: MH_Gen_BasicFlow

7.1.2 Establish Connection

This MSC describes how a connection is established between DSO and DSI. The priority check considers pending connections only to the same object (referring to FBlockID.InstID.FktID.OPType) in the DSI. If the requested connection refers to another object, the connection is not rejected. But it might be set into a hold state, if the priority is too low.

If the DSO receives a retry of a START CONNECTION command, the DSO behaves as follows:
- During transmission of first block (between READY FOR DATA and reception of the first acknowledge), the DSO must be resynchronized. It retransmits READY FOR DATA and restarts the transmission of the first block.
- After reception of first acknowledge from the DSI, the START CONNECTION command must be ignored.

Note: *Groupcasts or broadcasts are not supported by MHP. Thus, all requests with FBlockID 0xFF or InstID 0xFF are ignored without error response.*

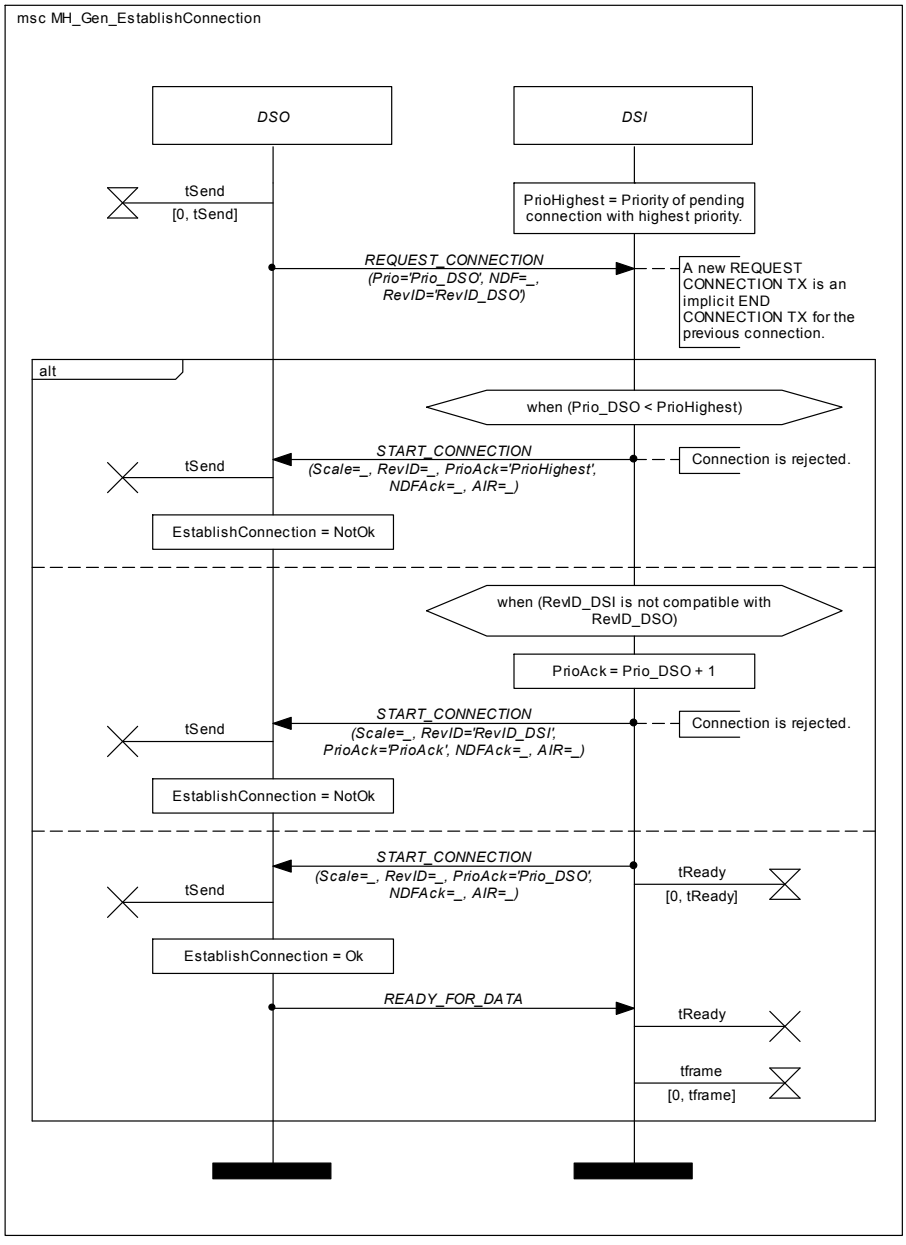

Figure 7-2: MH_Gen_EstablishConnection

7.1.3 Data Transmission with Single Frame Acknowledge

This MSC describes how a block of data shall be transmitted from DSO to DSI if SFA mode is used.

Note: *If the DSO attempts to send packets with length 0, the following behavior applies.*
a) If a connection is not opened yet, a connection is not established.
b) If a connection is already open, the attempt is ignored.

Single frame acknowledge (SFA) mode is deprecated!

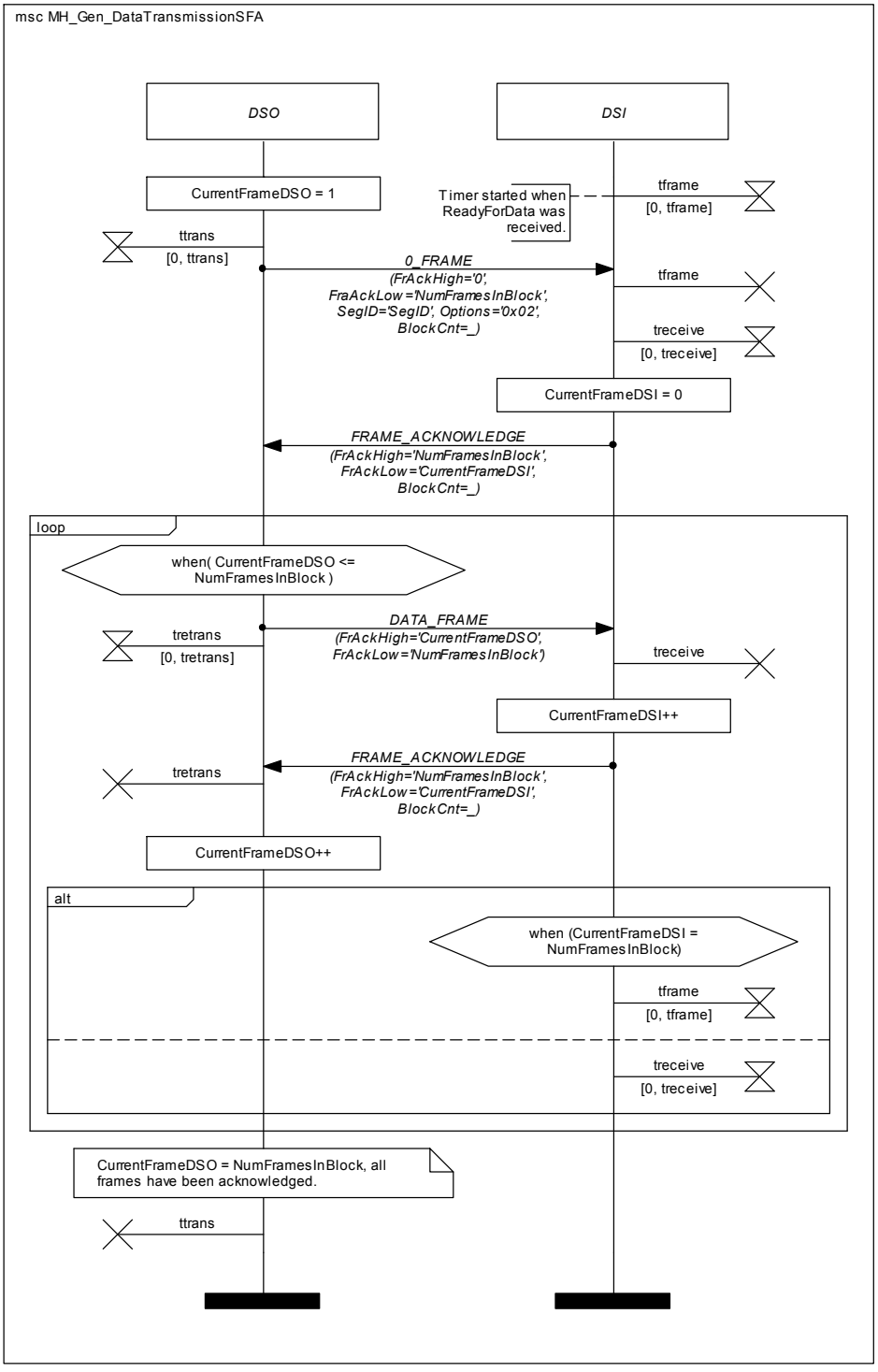

Figure 7-3: MH_Gen_DataTransmissionSFA

7.1.4 Data Transmission with Block Acknowledge

This MSC describes how a block of data shall be transmitted from DSO to DSI if block acknowledge mode is used.

Note: *If the DSO attempts to send packets with length 0, the following behavior applies.*
a) If a connection is not opened yet, a connection is not established.
b) If a connection is already open, the attempt is ignored.

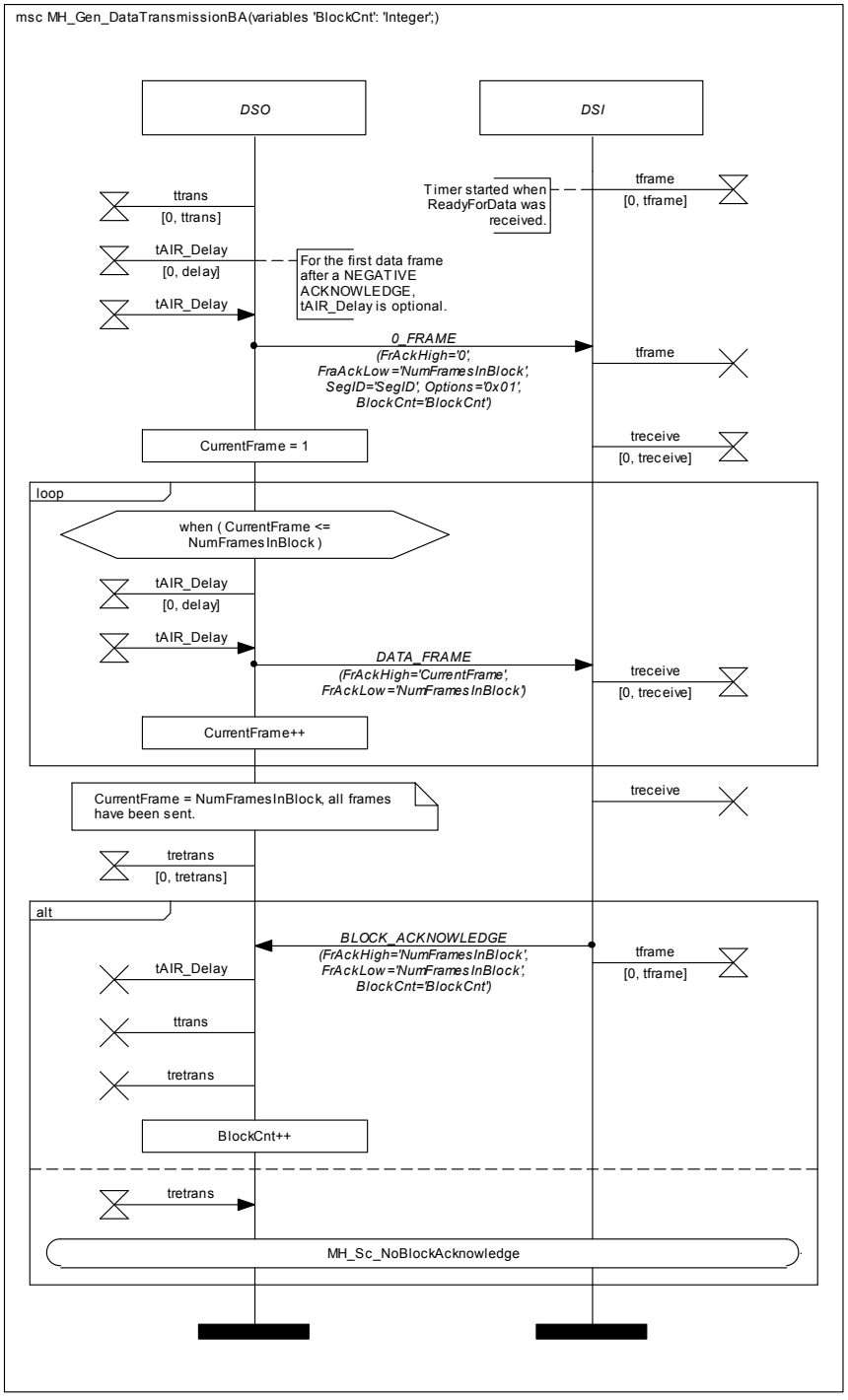

Figure 7-4: MH_Gen_DataTransmission_BA

7.1.5 END CONNECTION by DSO

A connection is closed with an END CONNECTION TX command.

The DSO sends the END CONNECTION TX command using Event 0x00 (regular end) or 0xFF (kill connection). The DSO sends the command r_{end} times, but the connection must be regarded as closed after the first attempt. The DSI regards the connection as closed, as soon as an END CONNECTION TX was received, or after a timeout.

The repeated sending of END CONNECTION TX is interrupted when the connection needs to be re-established.
The DSI has to accept a new REQUEST CONNECTION, even if the previous connection was not terminated. A new REQUEST CONNECTION is an implicit END CONNECTION TX for the previous connection.

This MSC specifies how the DSO shall close the connection:

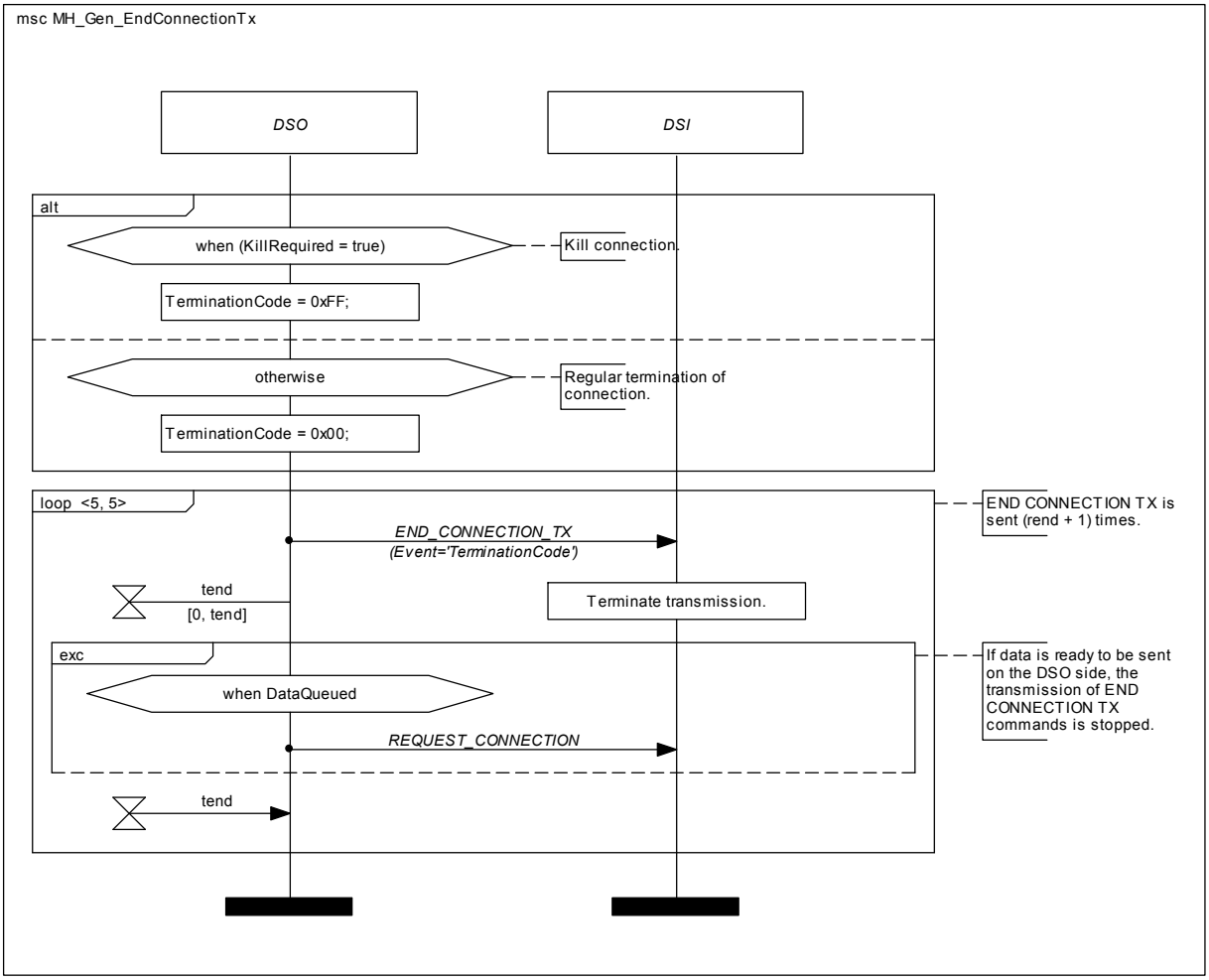

Figure 7-5: MH_Gen_EndConnectionTx

7.1.6 HOLD CONNECTION by DSO

The DSO has the possibility to pause or interrupt a connection if requested by the application.. Pending timers shall be stopped before HOLD CONNECTION TX is transmitted.

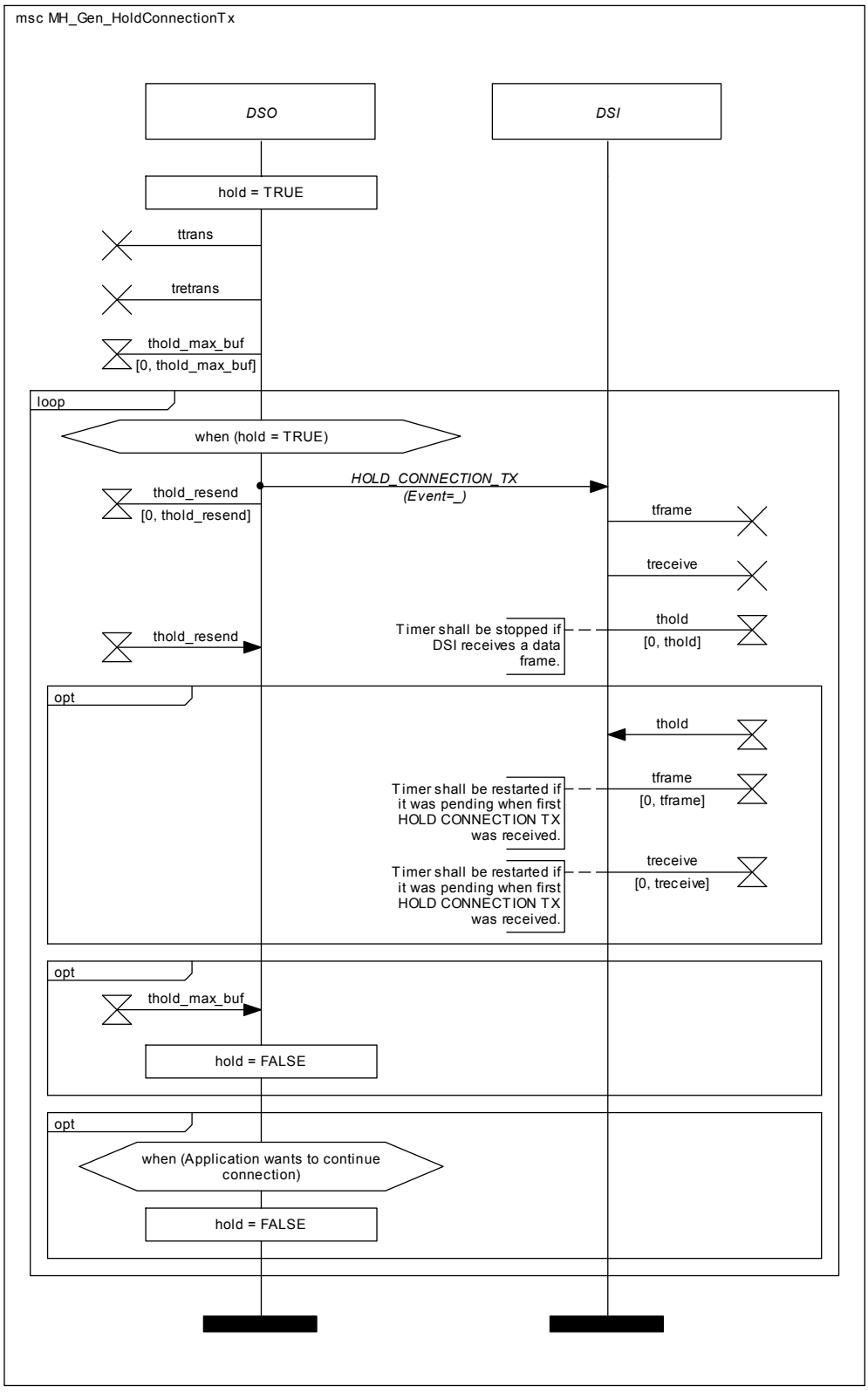

Figure 7-6: MH_Gen_HoldConnectionTx

7.1.7 HOLD CONNECTION by DSO—using t_{Delay_End}

This MSC describes the sending of HOLD CONNECTION TX commands in the case that no data is available to send or in the case that data is available to send, but another connection is active at this time. If interleaving on frame level (optional behavior) is not supported, only one connection is active at one time (for more information see section *8 Concurrent Connections*).
This scenario is started after the connection was opened or after data was sent. If the timer t_{Delay_End} expires, the connection is closed as described in *7.1.5 END CONNECTION by DSO*.

Note: *HOLD CONNECTION TX is sent after the timer t_{hold_resend} expires or a NEGATIVE ACKNOWLEDGE is received.*
The events DataQueued and ActivateConnection will not trigger the immediate sending of a HOLD CONNECTION TX command.

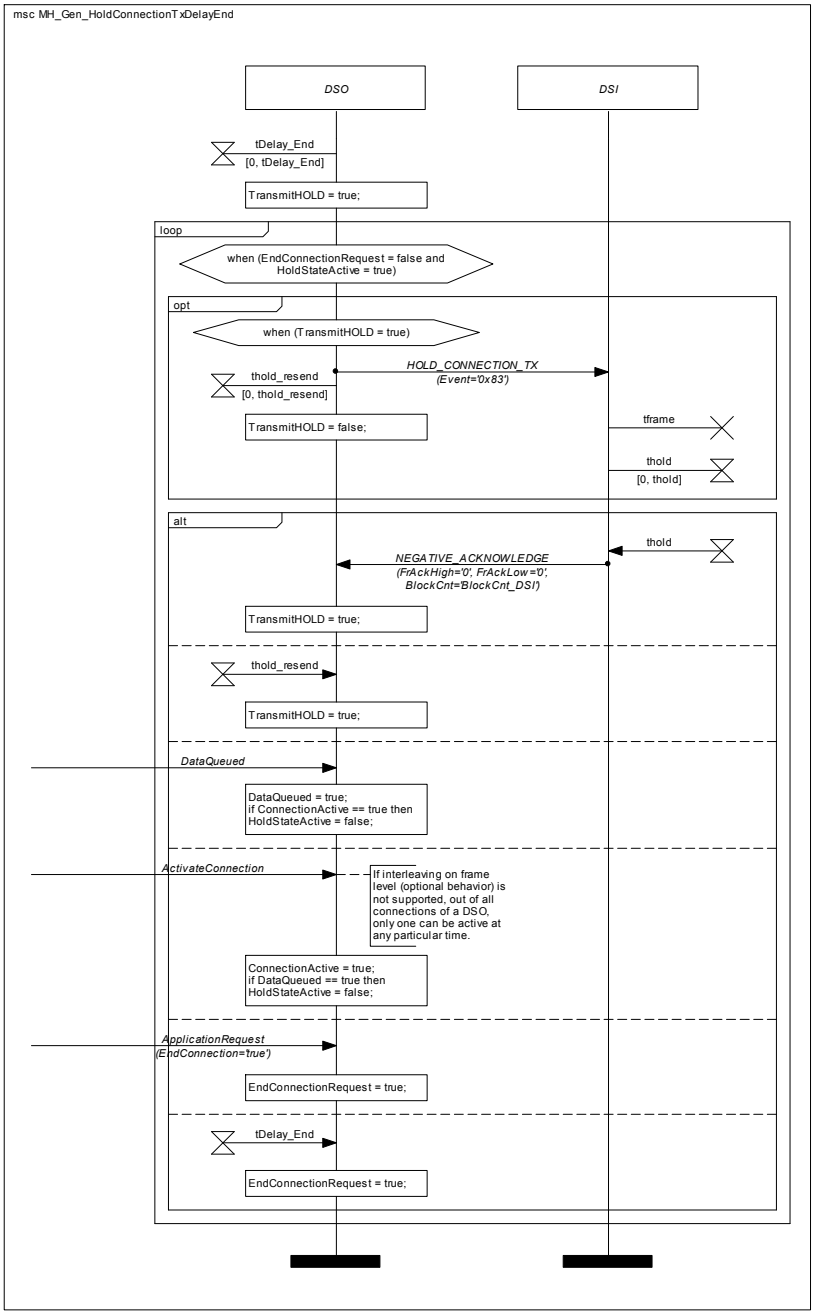

Figure 7-7: MH_Gen_EndConnectionTxDelayEnd

7.1.8 Hold Connection by DSI

The DSI has the possibility to pause or interrupt a connection if it recognizes malfunctions during transmission or a connection at higher priority is received during transmission. Pending timers shall be stopped before HOLD CONNECTION RX is transmitted and restarted when connection is not hold any longer.

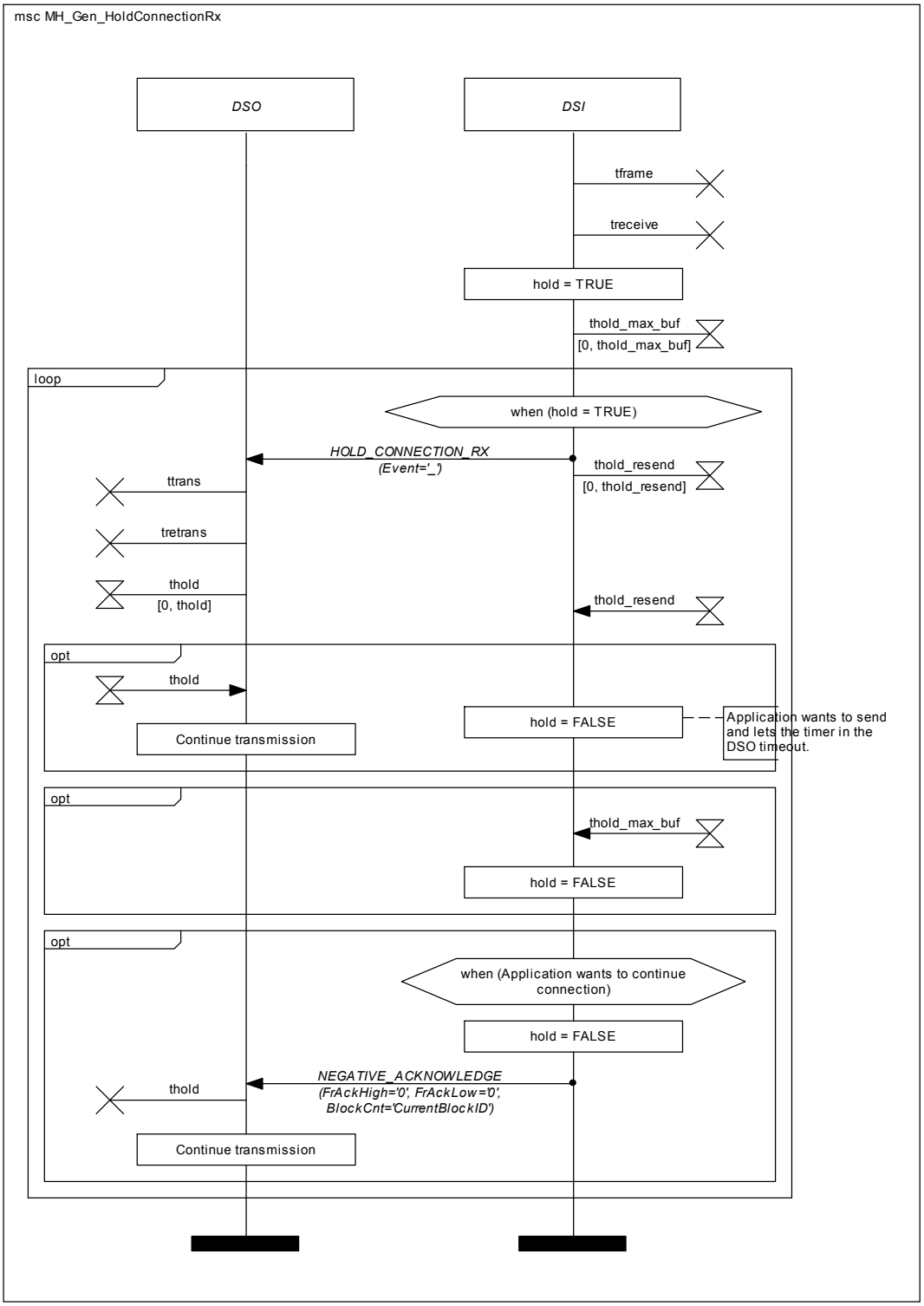

Figure 7-8: MH_Gen_Hold ConnectionRx

7.1.9 MULTIPLE FRAMES REQUEST

In the MULTIPLE FRAMES REQUEST command, each FrameID reserves one byte. Up to 41 MHP Data frames can be requested within one MULTIPLE FRAMES REQUEST command.

This MSC describes how the DSI shall request missing MHP Data frames in block acknowledge mode. For MULTIPLE FRAMES REQUEST with transmission rate adaptation please see 7.1.10.

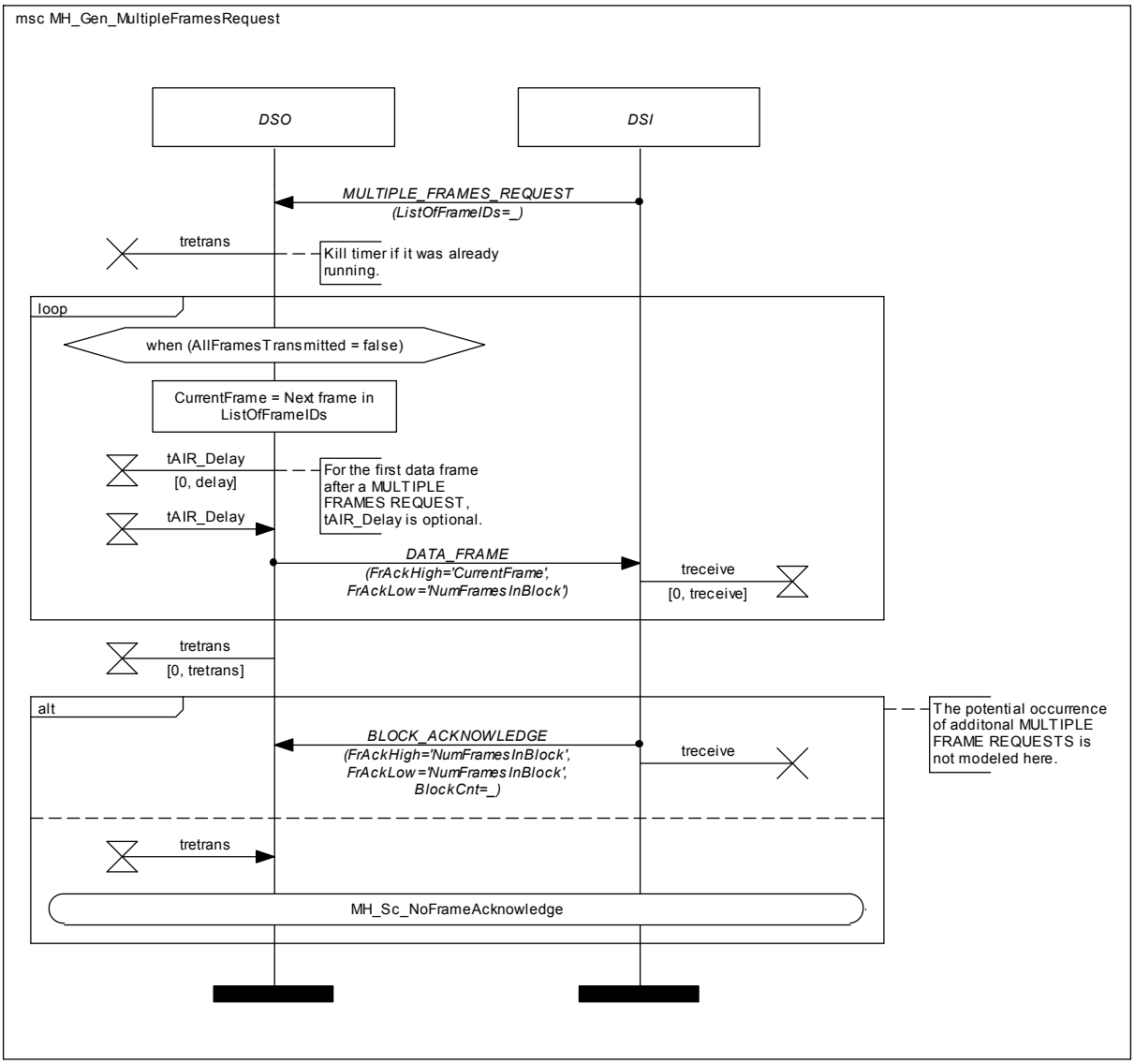

Figure 7-9: MH_Gen_MultipleFramesRequest

7.1.10 Transmission Rate Adaptation

The following MSC describes how the transmission rate can be adapted during transmission. This is applicable in block acknowledge mode. The initial transmission rate is set by AIR in the DSI and the DSO.
The DSO has the possibility to increase the transmission rate during transmission. This can be done until the DSI sends a MULTIPLE FRAMES REQUEST. After that, the DSO shall decrease the transmission rate.

Note: *The transmission rate can also be adapted in case of a NEGATIVE ACKNOWLEDGE of the same block.*

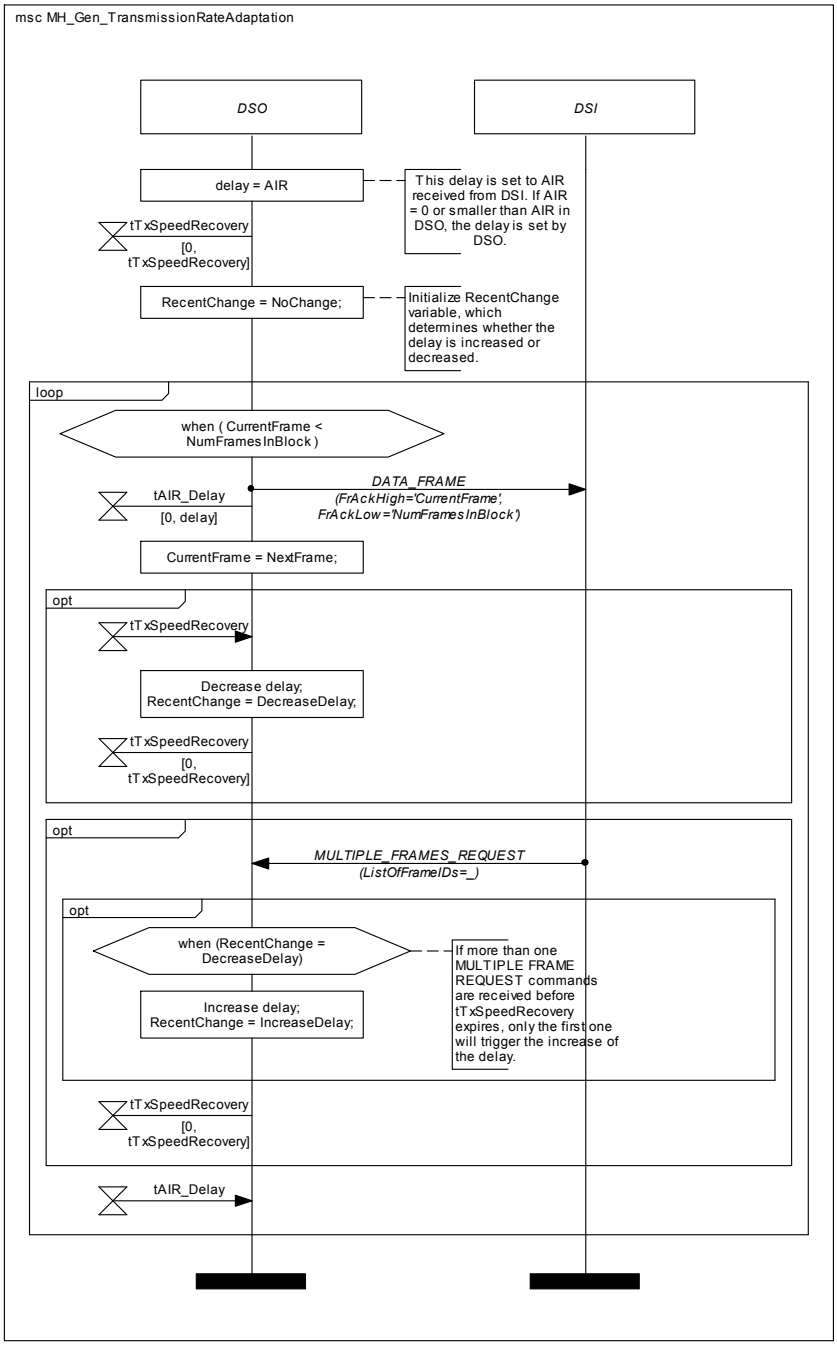

Figure 7-10: MH_Gen_TransmissionRateAdaptation

7.2 Scenario MSCs

7.2.1 REQUEST CONNECTION Failure

This MSC specifies the behavior when the DSO tries to establish a connection by sending REQUEST CONNECTION but does not receive START CONNECTION within t_{Send}. Timer started when REQUEST CONNECTION was transmitted.

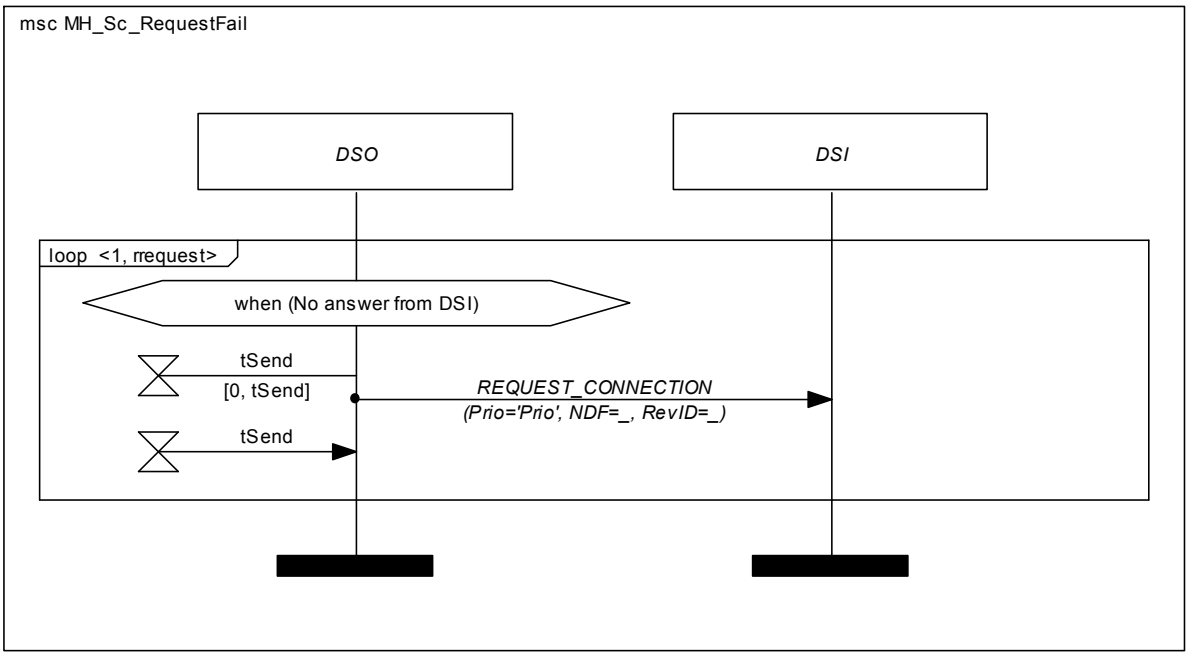

Figure 7-11: MH_Sc_RequestFail

7.2.2 READY FOR DATA not Received

This scenario is started when the DSI has not received READY FOR DATA within t_{Ready}. Timer started when START CONNECTION was transmitted.

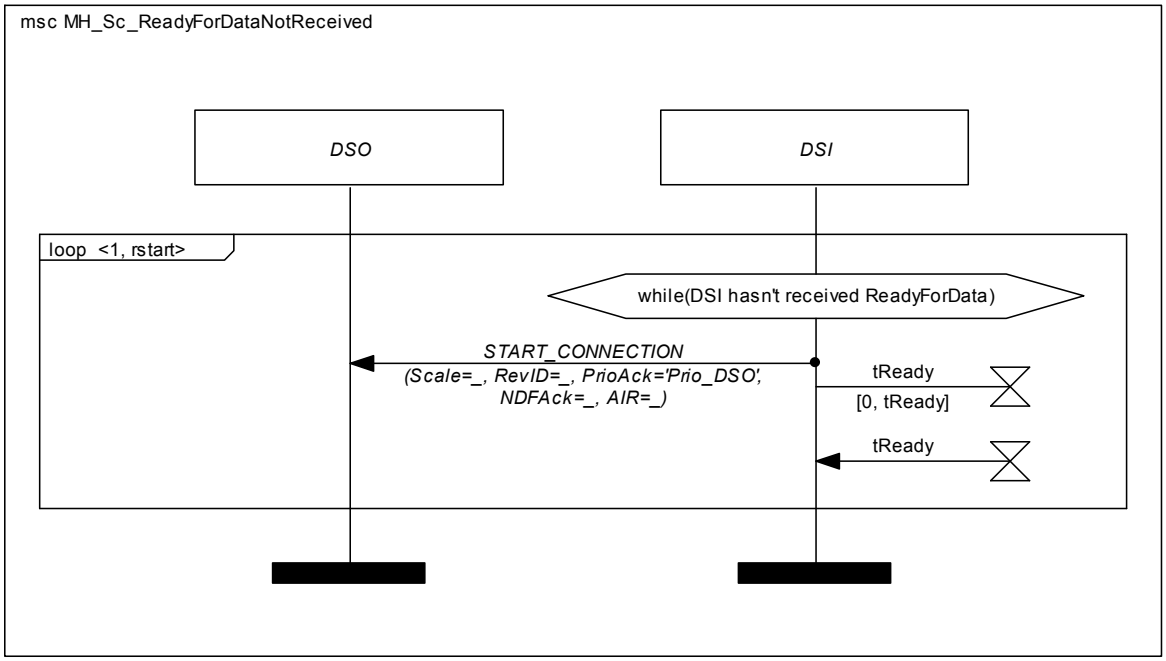

Figure 7-12: MH_Sc_ReadyForDataNotReceived

7.2.3 DATA FRAME Received Without 0-FRAME

This scenario is started when the DSI receives an MHP Data frame without a proceeding 0-FRAME. The number of NEGATIVE ACKNOWLEDGE retries is limited by r_{negack}.

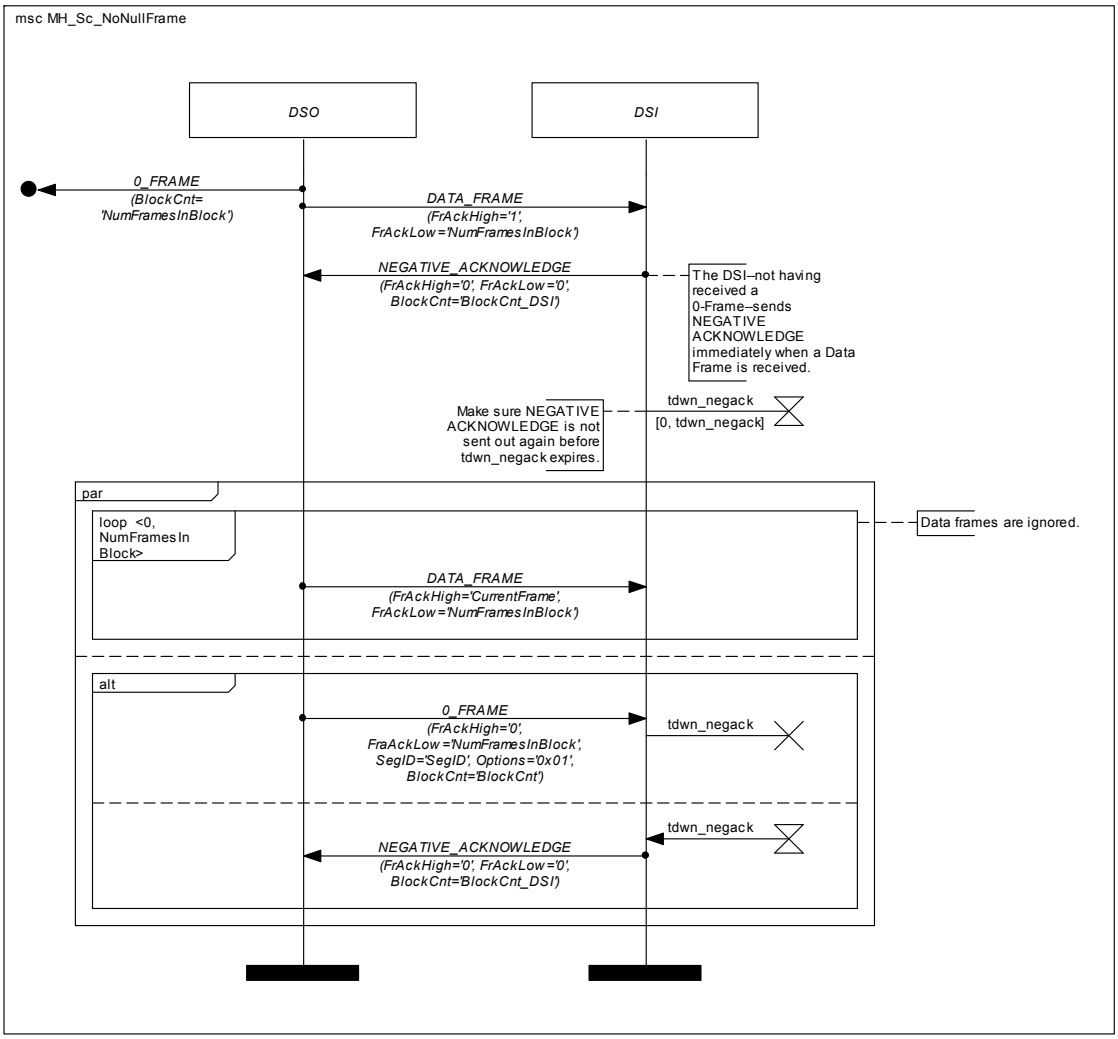

Figure 7-13: MH_Sc_NoNullFrame

7.2.4 DATA FRAME not Acknowledged

This scenario is started when a transmitted MHP Data frame has not been acknowledged by the DSI and timer $t_{retrans}$ or timer t_{trans} has expired. Timer t_{trans} started when 0-FRAME was transmitted and timer $t_{retrans}$ was started when last data MHP Data frame was transmitted. This scenario is only applicable in SFA mode.

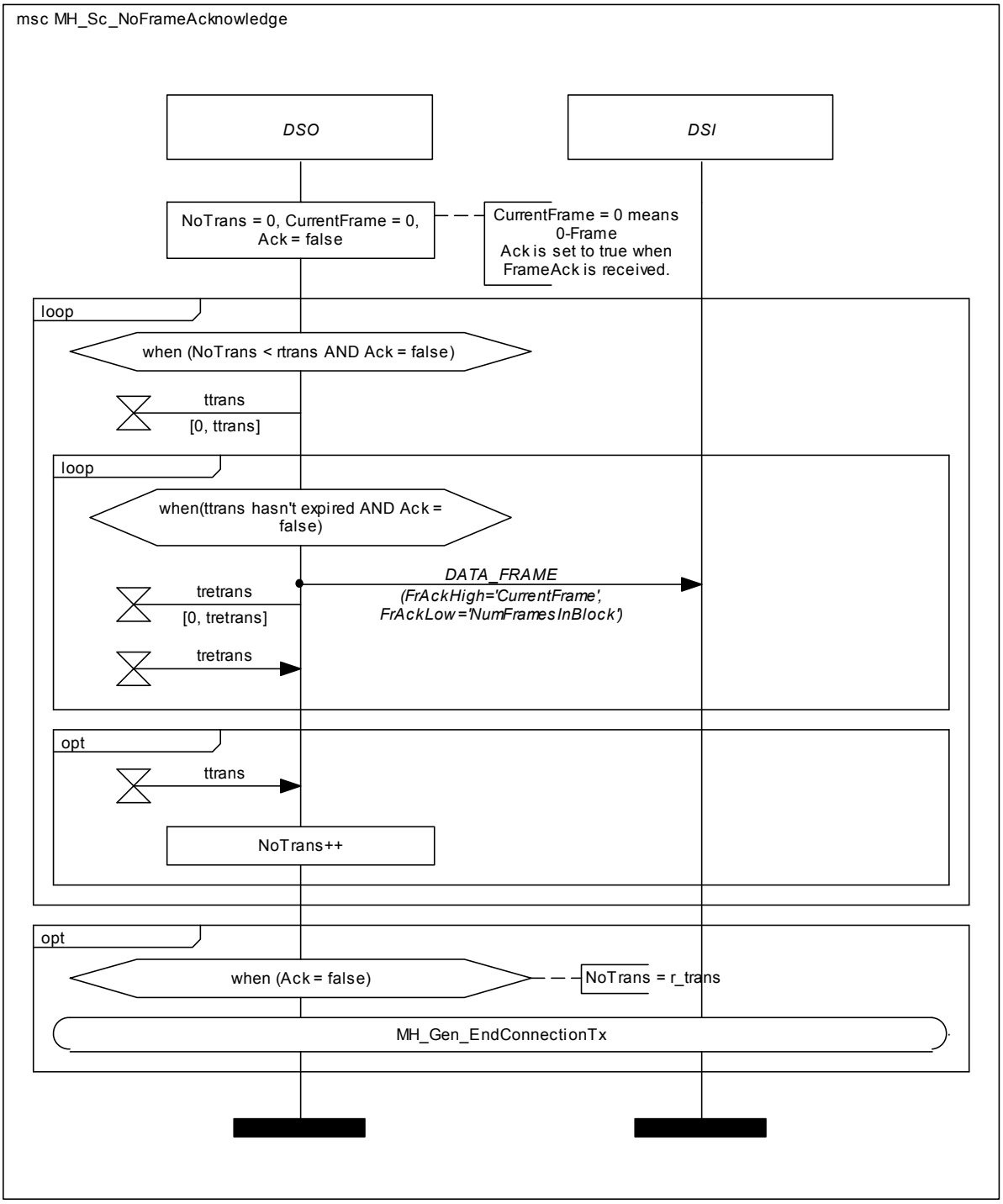

Figure 7-14: MH_Sc_NoFrameAcknowledge

7.2.5 MULTIPLE FRAMES REQUEST After Timeout

This scenario describes a use case for the occurrence of a MULTIPLE FRAMES REQUEST. Here, the DSO fails to send data frames within the specified time limits and triggers a MULTIPLE FRAMES REQUEST in the DSI.

Note: *The focus of this scenario is on the behavior of the DSI.*

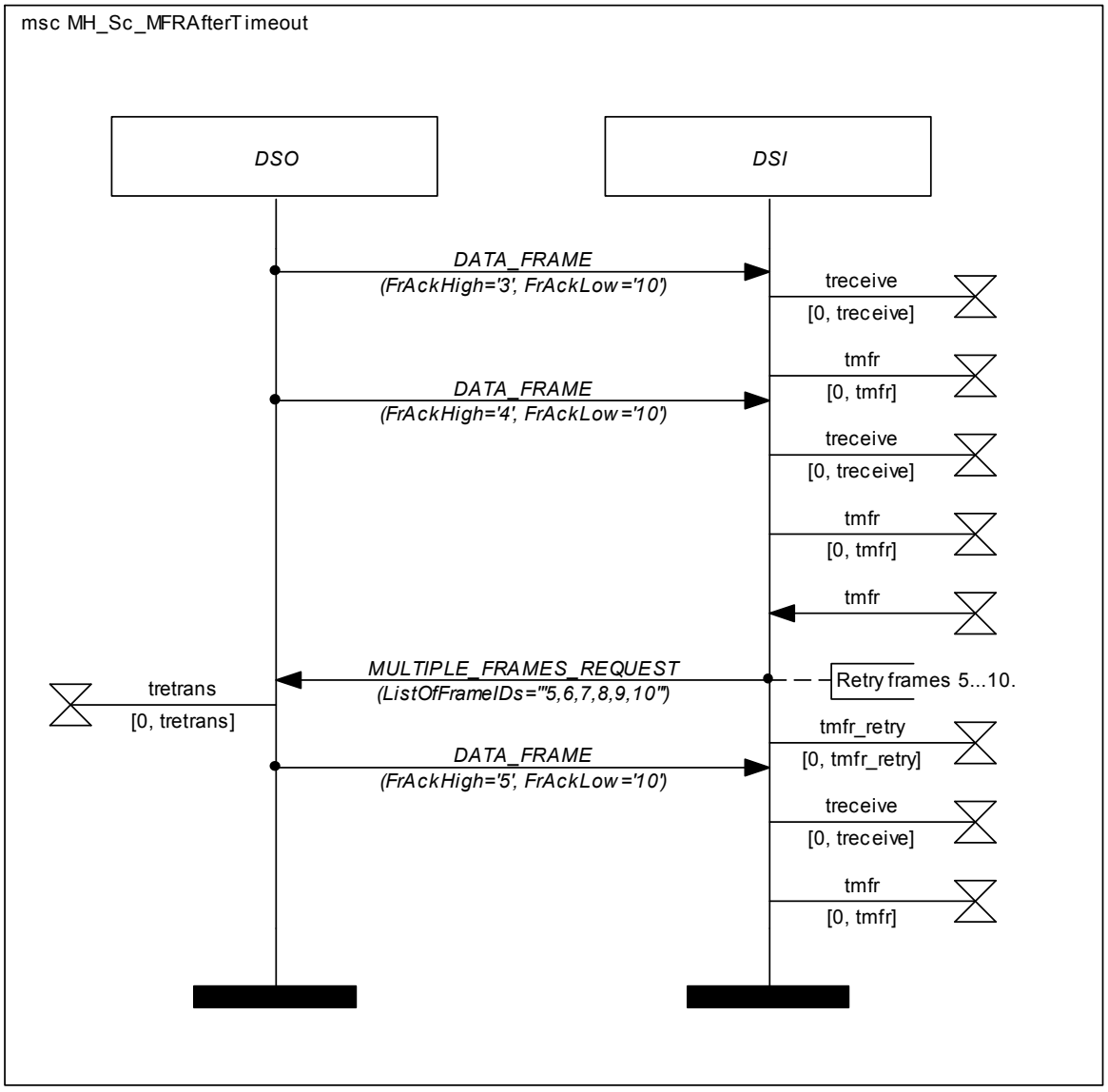

Figure 7-15: MH_Sc_MFRAfterTimeout

7.2.6 MULTIPLE FRAMES REQUEST After Last Frame

This scenario describes a use case for the occurrence of a MULTIPLE FRAMES REQUEST. Here, an expected data frame is not contained in the set of frames received by the DSI; that causes the DSI to send a MULTIPLE FRAMES REQUEST.

Note: *The focus of this scenario is on the behavior of the DSI.*

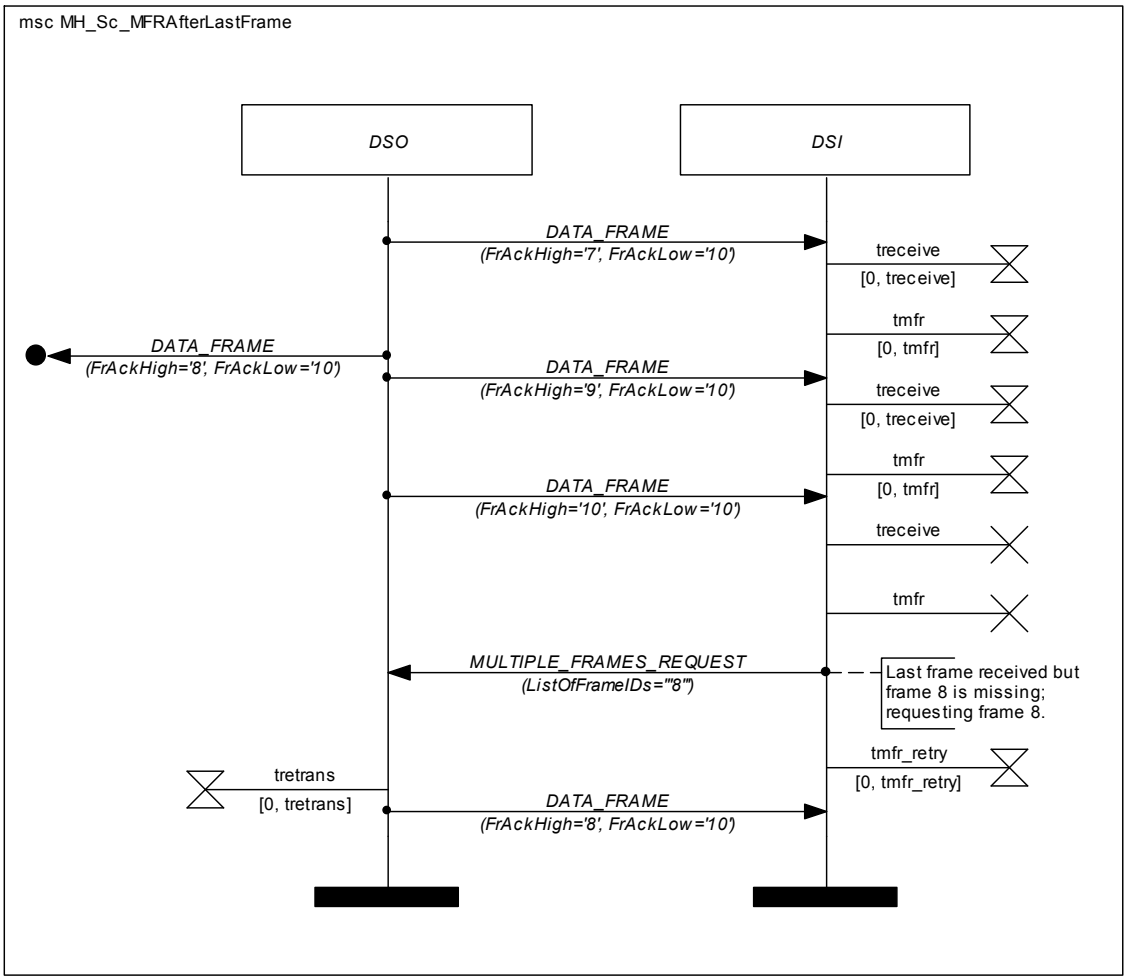

Figure 7-16: MH_Sc_MFRAfterLastFrame

7.2.7 Merging MULTIPLE FRAMES REQUESTs

In the case that the DSO is sending MHP Data frames for a previous MULTIPLE FRAMES REQUEST and a new MULTIPLE FRAMES REQUEST is received, the requests are merged by the DSO.
In this MSC, the timer $t_{Retrans}$, which exists on the DSO side, is not modeled because it is not relevant for the scenario.

Note: *For legacy implementations (which refers to MOST25 and MOST50 devices based on MOST Specifications with Major Revision 2) please see 10.2 Miscellaneous Requirements in the appendix.*

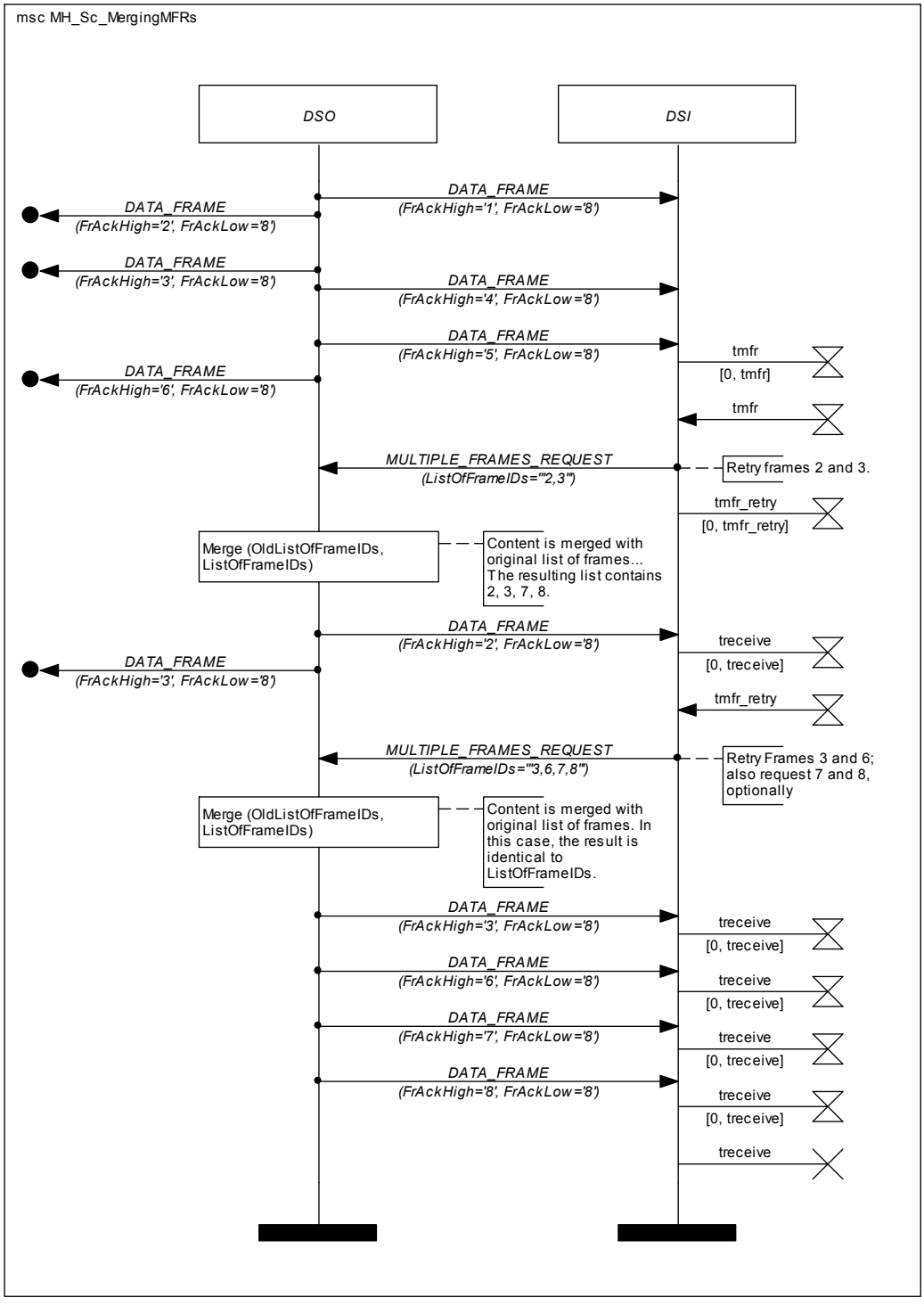

Figure 7-17: MH_Sc_MergingMFRs

7.2.8 Data Block not Acknowledged

This scenario is started when a transmitted data block has not been acknowledged by the DSI and timer $t_{retrans}$ or timer t_{trans} has expired. Timer t_{trans} started when 0-FRAME was transmitted and timer $t_{retrans}$ was started when last MHP Data frame was transmitted. This scenario is only applicable in block acknowledge mode.

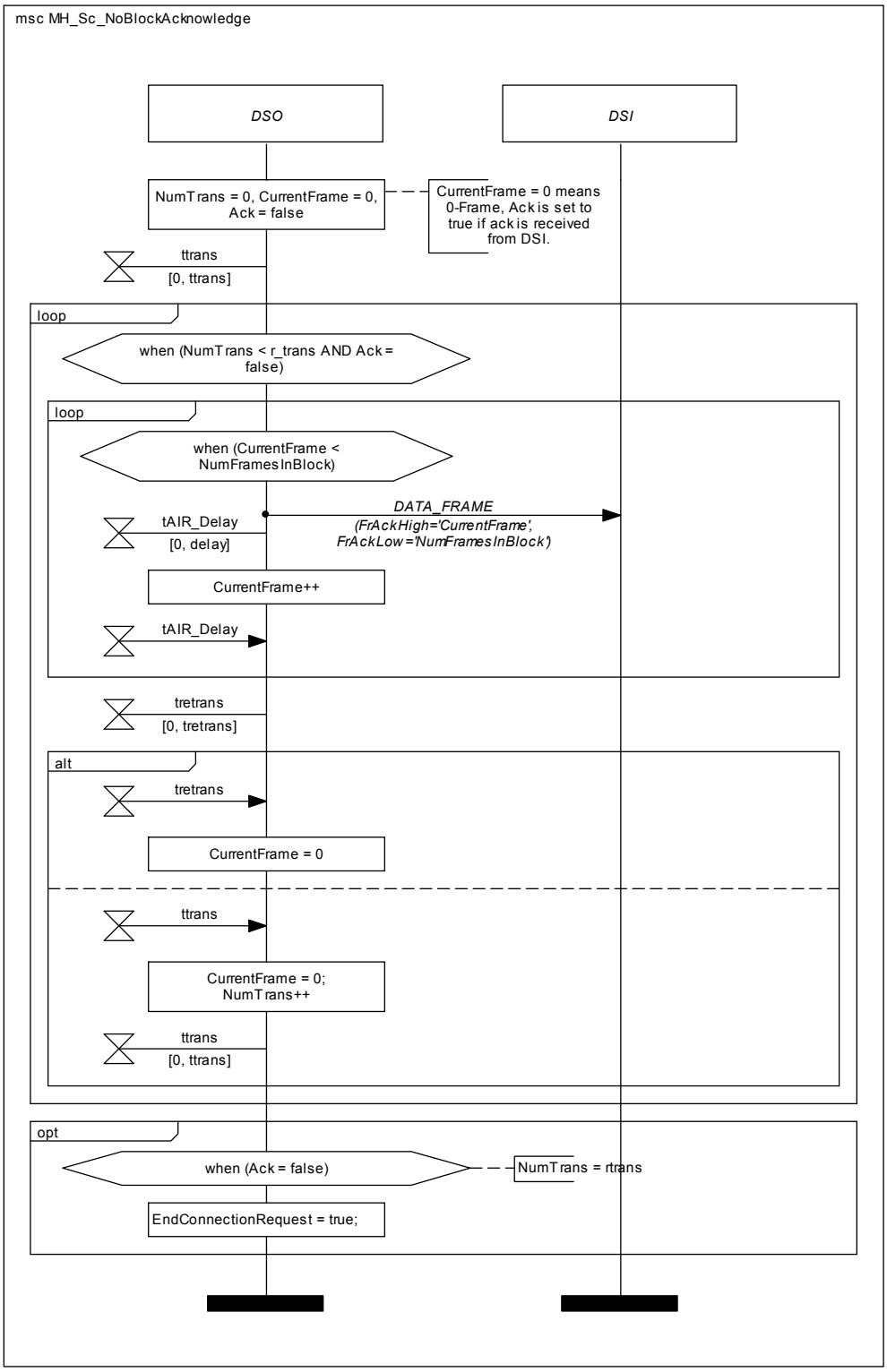

Figure 7-18: MH_Sc_NoBlockAcknowledge

7.2.9 Connection not Terminated by DSO

This MSC specifies how the DSI shall behave if data no longer is received from the DSO and t_{frame} or $t_{receive}$ has expired. Timer t_{frame} is started when READY FOR DATA is received or when acknowledge after last received MHP Data frame is sent to the DSO. Timer $t_{receive}$ is started every time the DSI receives an MHP Data frame from the DSO (except for last MHP Data frame).

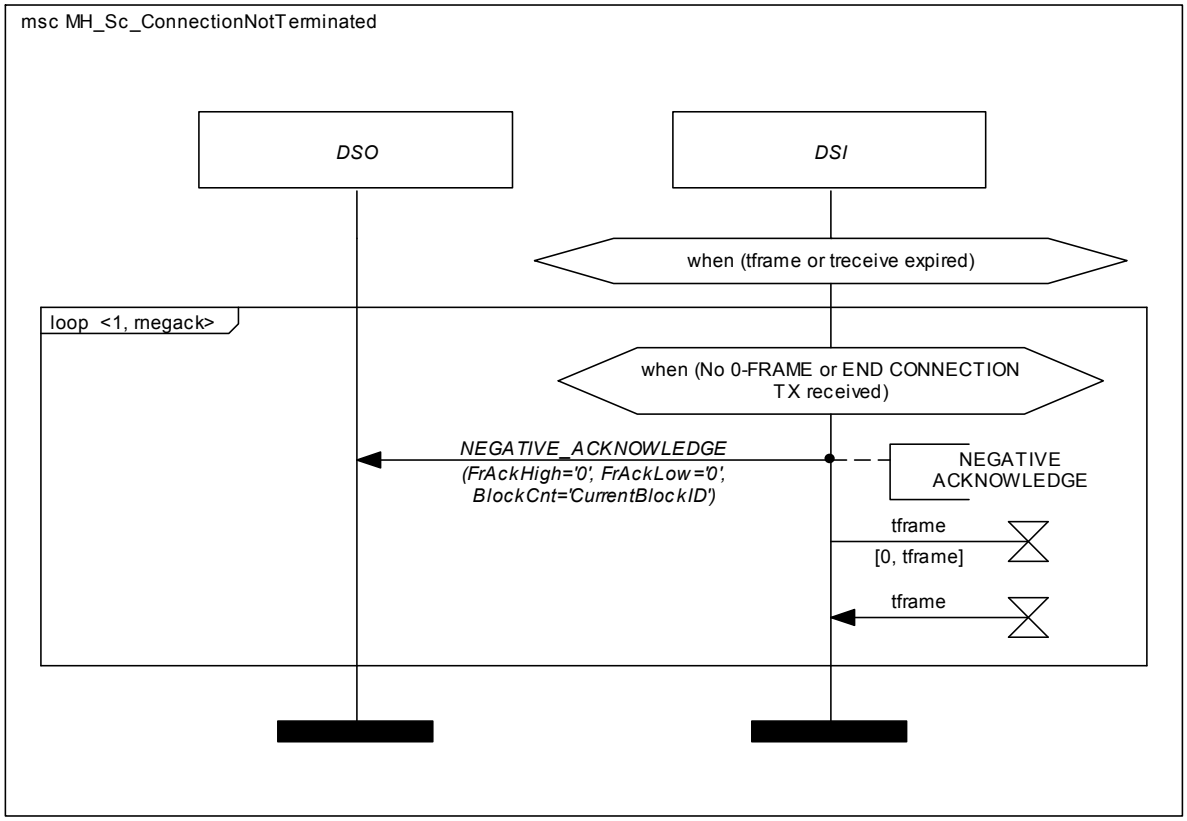

Figure 7-19: MH_Sc_ConnectionNotTerminated

7.2.10 NEGATIVE ACKNOWLEDGE

This MSC specifies how the DSO shall behave if a transmitted MHP Data frame is answered by a NEGATIVE ACKNOWLEDGE.

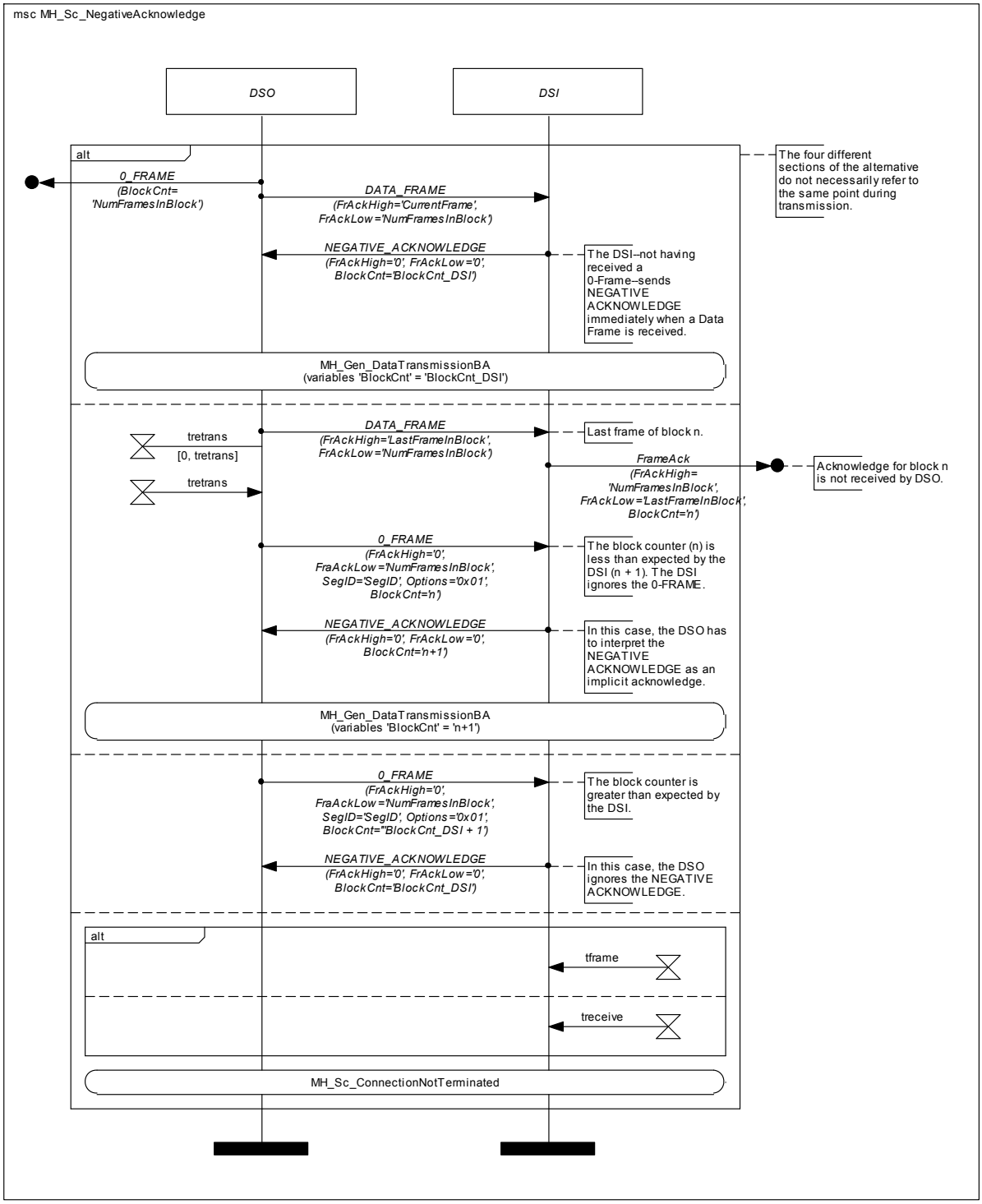

Figure 7-20: MH_Sc_NegativeAcknowledge

7.2.11 DSI Terminates Connection

The DSI can always terminate an established connection. END CONNECTION RX is used for this purpose and it is illustrated in the following MSC.

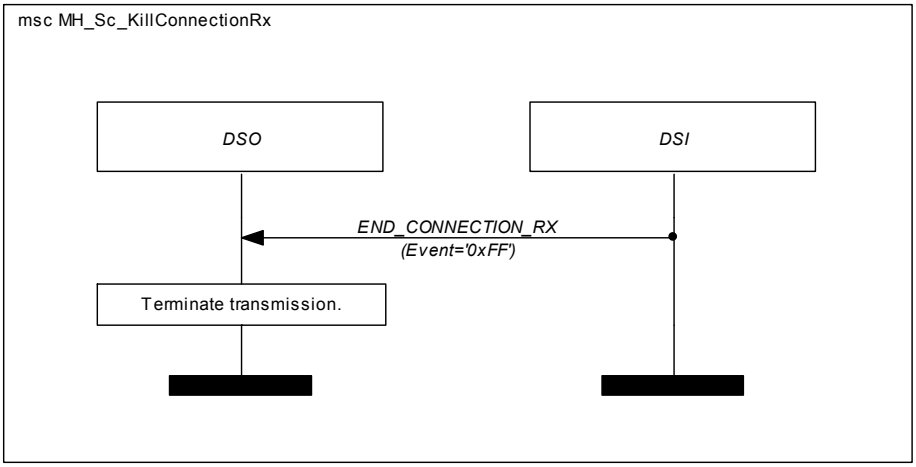

Figure 7-21: MH_Sc_KillConnectionRx

8 Concurrent Connections

8.1 Single Connection

Scenario	Single Connection
Description	One connection between one DSO and one DSI. Con1: SrcAddr.TgtAddr.FBlockID.InstID.FktID.OPType
Behavior	The connection is established. Data is transmitted. Connection is held by the DSO. Connection is closed by the DSO.

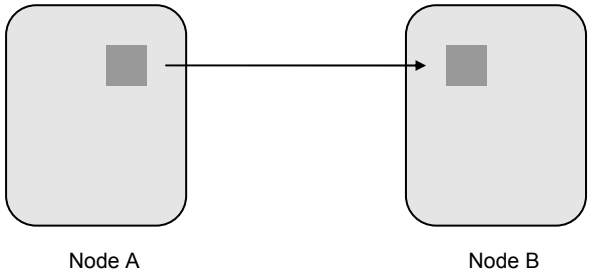

Figure 8-1: Concurrent Connections - Single Connection

8.2 Twin Connection

Scenario	Twin Connection
Description	One DSO tries to establish two identical connections to the same DSI. Both connection IDs are identical This case is NOT supported by MHP → The connection IDs must be different (refer to use case “Parallel Connection”). Con1: SrcAddr.TgtAddr.FBlockID.InstID.FktID.OPType Con2: SrcAddr.TgtAddr.FBlockID.InstID.FktID.OPType Con1 = Con2
Behavior	Connection #1 is established. Connection #2 cannot be established. Data of connection #1 is transmitted. Connection #1 is held by the DSO. Connection #1 is closed by the DSO.

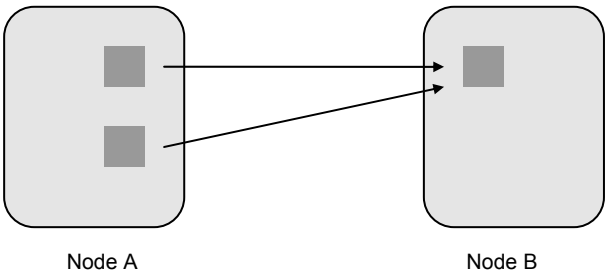

Figure 8-2: Concurrent Connections - Twin Connection

8.3 Parallel Connection

Scenario	Parallel Connection
Description	One DSO tries to establish two different connections to the same DSI. Con1: SrcAddr1.TgtAddr1.FBlockID1.InstID1.FktID1.OPType1 Con2: SrcAddr1.TgtAddr1.FBlockID2.InstID2.FktID2.OPType2 Con1 ≠ Con2 Con1 is established before Con 2 is established.
Behavior	The behavior depends on the priority of both connections. Case A) Prio2 = Prio1 Case B) Prio2 < Prio1 Case C) Prio2 > Prio1 See subchapters below.

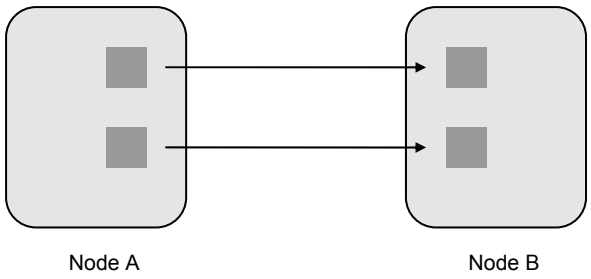

Figure 8-3: Concurrent Connections - Parallel Connection

8.3.1 Parallel Connections Using Same Priority

Scenario	Parallel Connections using the same priority
Description	One DSO tries to establish two different connections to the same DSI. Con1: SrcAddr1.TgtAddr1.FBlockID1.InstID1.FktID1.OPType1 Con2: SrcAddr1.TgtAddr1.FBlockID2.InstID2.FktID2.OPType2 Con1 ≠ Con2 Con1 is established before Con2 is established. The second connection (Con2) has the same priority as the first connection (Con1): Prio2 = Prio1
Behavior	Connection #1 is established. Connection #2 is established. Connection #2 is held (paused) by the DSO. Block 0 of Connection #1 is transmitted Connection #1 is held (paused) by the DSO. Block 0 of Connection #2 is transmitted Connection #2 is held (paused) by the DSO. Block 1 of Connection #1 is transmitted Connection #1 is held (paused) by the DSO. Block 1 of Connection #2 is transmitted Connection #2 is held (paused) by the DSO. ... and so on ....

8.3.2 Parallel Connections Using Lower Priority (Prio2 < Prio1)

Scenario	Parallel Connections using a lower priority
Description	One DSO tries to establish two different connections to the same DSI. Con1: SrcAddr1.TgtAddr1.FBlockID1.InstID1.FktID1.OPType1 Con2: SrcAddr1.TgtAddr1.FBlockID2.InstID2.FktID2.OPType2 Con1 ≠ Con2 Con1 is established before Con2 is established. The second connection (Con2) has a lower priority as the first connection (Con1): Prio2 < Prio1
Behavior	Connection #1 is established. Connection #2 is established. Connection #2 is held (paused) by the DSO. Block 0 of Connection #1 is transmitted Block 1 of Connection #1 is transmitted Block 2 of Connection #1 is transmitted ….. Block N of Connection #1 is transmitted (till all available blocks are transmitted) Connection #1 is held (paused) by the DSO. Block 0 of Connection #2 is transmitted Block 1 of Connection #2 is transmitted Block 2 of Connection #2 is transmitted ….. When Connection #1 has the need to transmit further data, the Connection #2 is held (paused) by the DSO, after the current block (of Con2) is completed. Data on lower priority is transmitted only, when no other connection using a higher priority is waiting for transmission.

8.3.3 Parallel Connections Using Higher Priority (Prio2 > Prio1)

Scenario	Parallel Connections using a higher priority
Description	One DSO tries to establish two different connections to the same DSI. Con1: SrcAddr1.TgtAddr1.FBlockID1.InstID1.FktID1.OPType1 Con2: SrcAddr1.TgtAddr1.FBlockID2.InstID2.FktID2.OPType2 Con1 ≠ Con2 Con1 is established before Con2 is established. The second connection (Con2) has a higher priority as the first connection (Con1): Prio2 > Prio1
Behavior	Connection #1 is established. Block 0 of Connection #1 is transmitted Block 1 of Connection #1 is transmitted Block 2 of Connection #1 is transmitted Meanwhile Connection #2 is established. Connection #2 is held (paused) by the DSO, until the current block of the lower connection is completed. Connection #1 is held (paused) by the DSO. Block 0 of Connection #2 is transmitted Block 1 of Connection #2 is transmitted Block 2 of Connection #2 is transmitted ... Block N of Connection #2 is transmitted (till all available blocks are transmitted) Connection #2 is held (paused) by the DSO. Block 3 of Connection #1 is transmitted Block 4 of Connection #1 is transmitted Block 5 of Connection #1 is transmitted ….. When Connection #2 has the need to transmit further data, the Connection #1 is held (paused) by the DSO, after the current block (of Con1) is completed. Data on lower priority is transmitted only, when no other connection using a higher priority is waiting for transmission.

8.4 Multiple Connections to Multiple DSIs

Scenario	Multiple connections to different DSIs
Description	One DSO tries to establish two different connections to different target devices. Con1: SrcAddr.TgtAddr1.FBlockID1.InstID1.FktID1.OPType1 Con2: SrcAddr.TgtAddr2.FBlockID2.InstID2.FktID2.OPType2 Con1 ≠ Con2 Con1 is established before Con 2 is established.
Behavior	The behavior depends on the priority of both connections. Case A) Prio2 = Prio1 Case B) Prio2 < Prio1 Case C) Prio2 > Prio1 See subchapters below.

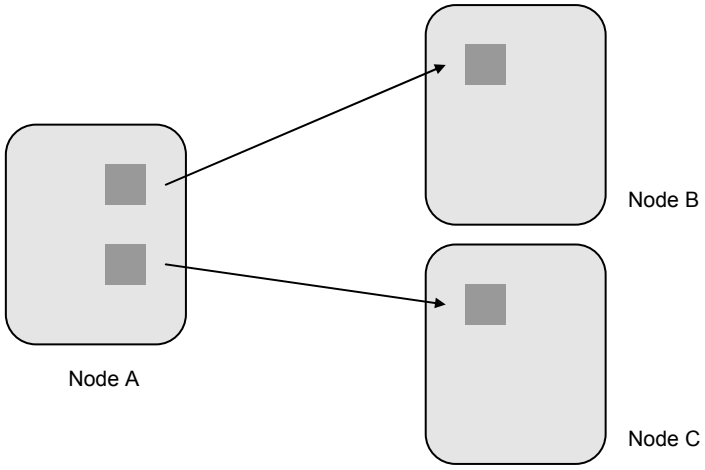

Figure 8-4: Concurrent Connections - Multiple Connections to Different DSIs

8.4.1 Connections to Multiple DSIs Using Same Priority

8.4.1.1 Interleaving on Block Level (Standard Behavior)

Scenario	Multiple Connections to Multiple DSIs using the same priority
Description	One DSO tries to establish two different connections to different target devices. Con1: SrcAddr.TgtAddr1.FBlockID1.InstID1.FktID1.OPType1 Con2: SrcAddr.TgtAddr2.FBlockID2.InstID2.FktID2.OPType2 Con1 ≠ Con2 Con1 is established before Con 2 is established. The second connection (Con2) has the same priority as the first connection (Con1): Prio2 = Prio1
Behavior	Connection #1 is established. Connection #2 is established. Connection #2 is held (paused) by the DSO. Block 0 of Connection #1 is transmitted Connection #1 is held (paused) by the DSO. Block 0 of Connection #2 is transmitted Connection #2 is held (paused) by the DSO. Block 1 of Connection #1 is transmitted Connection #1 is held (paused) by the DSO. Block 1 of Connection #2 is transmitted Connection #2 is held (paused) by the DSO. ... and so on ....

8.4.1.2 Interleaving on Frame Level (Optional Behavior)

Optionally, the DSO may service and transmit MHP Data frames on multiple connections in an interleaved manner. This is allowed only for connections that are aimed at different DeviceIDs.

Note: *The* DSI *must not interleave on MHP Data frame level but only on block level. Otherwise, the AIR (Average Interrupt Rate, see 4.1.2.5) regulation would not be operational.*

8.4.2 Connections to Multiple DSIs Using Lower Priority

Scenario	Multiple Connections to Multiple DSIs using different priorities (Prio2<Prio1)
Description	One DSO tries to establish two different connections to different target devices. Con1: SrcAddr.TgtAddr1.FBlockID1.InstID1.FktID1.OPType1 Con2: SrcAddr.TgtAddr2.FBlockID2.InstID2.FktID2.OPType2 Con1 ≠ Con2 Con1 is established before Con 2 is established. The second connection (Con2) has a lower priority than the first connection (Con1): Prio2 < Prio1
Behavior	Connection #1 is established. Connection #2 is established. Connection #2 is held (paused) by the DSO. Block 0 of Connection #1 is transmitted Block 1 of Connection #1 is transmitted Block 2 of Connection #1 is transmitted ….. Block N of Connection #1 is transmitted (till all available blocks are transmitted) Connection #1 is held (paused) by the DSO. Block 0 of Connection #2 is transmitted Block 1 of Connection #2 is transmitted Block 2 of Connection #2 is transmitted ….. When Connection #1 has the need to transmit further data, the Connection #2 is held (paused) by the DSO, after the current block (of Con2) is completed. Data on lower priority is transmitted only, when no other connection using a higher priority is waiting for transmission.

8.4.3 Connections to Multiple DSIs Using Higher Priority

Scenario	Multiple Connections to Multiple DSIs using different priorities (Prio2>Prio1)
Description	One DSO tries to establish two different connections to different target devices. Con1: SrcAddr.TgtAddr1.FBlockID1.InstID1.FktID1.OPType1 Con2: SrcAddr.TgtAddr2.FBlockID2.InstID2.FktID2.OPType2 Con1 ≠ Con2 Con1 is established before Con 2 is established. The second connection (Con2) has a higher priority than the first connection (Con1): Prio2 > Prio1
Behavior	Connection #1 is established. Block 0 of Connection #1 is transmitted Block 1 of Connection #1 is transmitted Block 2 of Connection #1 is transmitted Meanwhile Connection #2 is established. Connection #2 is held (paused) by the DSO, until the current block of the lower connection is completed. Connection #1 is held (paused) by the DSO. Block 0 of Connection #2 is transmitted Block 1 of Connection #2 is transmitted Block 2 of Connection #2 is transmitted ... Block N of Connection #2 is transmitted (till all available blocks are transmitted) Connection #2 is held (paused) by the DSO. Block 3 of Connection #1 is transmitted Block 4 of Connection #1 is transmitted Block 5 of Connection #1 is transmitted ….. When Connection #2 has the need to transmit further data, the Connection #1 is held (paused) by the DSO, after the current block (of Con1) is completed. Data on lower priority is transmitted only, when no other connection using a higher priority is waiting for transmission.

8.5 Coexistent Connections from Multiple DSOs

Scenario	Coexistent connections from multiple DSOs
Description	Different DSO devices try to establish multiple connections to one and the same target device. Each DSO targets to another object in the same DSI. Con1: SrcAddr1.TgtAddr.FBlockID1.InstID1.FktID1.OPType1 Con2: SrcAddr2.TgtAddr.FBlockID2.InstID2.FktID2.OPType2 Con1 ≠ Con2 Con1 is established before Con 2 is established.
Behavior	The behavior depends on the priority of both connections. Case A) Prio2 < Prio1 Case B) Prio2 = Prio1 Case C) Prio2 > Prio1 See subchapters below. ***Note:*** *For legacy implementations (which refers to MOST25 and MOST50 devices based on MOST Specifications with Major Revision 2) please see 10.2 Miscellaneous Requirements in the appendix.*

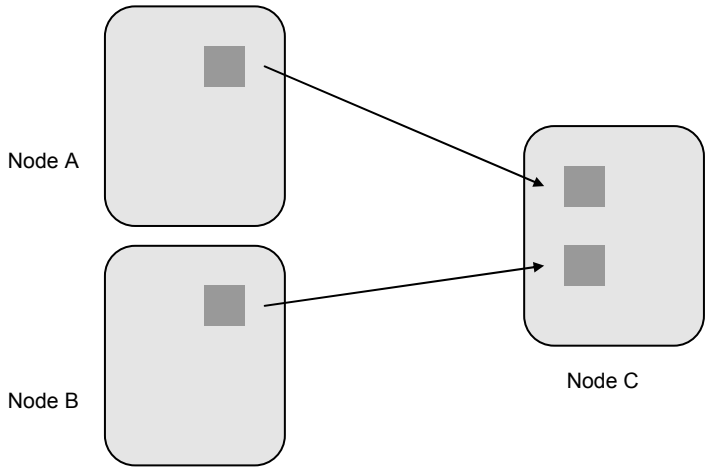

Figure 8-5: Concurrent Connections - Coexistent Connections from Multiple DSOs

8.5.1 Coexistent Connections Using a Lower Priority

Scenario	Coexistent connections from multiple DSOs using a lower priority
Description	Different DSO devices try to establish multiple connections to one and the same target device. Each DSO targets to another object in the same DSI. Con1: SrcAddr1.TgtAddr.FBlockID1.InstID1.FktID1.OPType1 Con2: SrcAddr2.TgtAddr.FBlockID2.InstID2.FktID2.OPType2 Con1 ≠ Con2 Con1 is established before Con 2 is established. The second connection (Con2) has a lower priority as the first connection (Con1): Prio2 < Prio1
Behavior	Connection #1 is established. Block 0 of Connection #1 is transmitted Block 1 of Connection #1 is transmitted … Meanwhile Connection #2 is established Connection #2 is held (paused) by the DSI (on reception of 0-FRAME) Block 2 of Connection #1 is transmitted Block 3 of Connection #1 is transmitted … Block N of Connection #1 is transmitted (till all available blocks are transmitted) Connection #1 is held (paused) by the DSO. Connection #2 is continued (triggered by a Negative Acknowledge by the DSI) Block 0 of Connection #2 is transmitted Block 1 of Connection #2 is transmitted ... and so on .... Data from a further connection (with lower priority) is only accepted, when no other connection using a higher priority is serviced. During this time, the DSI pauses the low priority connection by sending hold commands.

8.5.2 Coexistent Connections Using the Same Priority

Scenario	Coexistent connections from multiple DSOs using the **same** priority
Description	Different DSO devices try to establish multiple connections to one and the same target device. Each DSO targets to another object in the same DSI. Con1: SrcAddr1.TgtAddr.FBlockID1.InstID1.FktID1.OPType1 Con2: SrcAddr2.TgtAddr.FBlockID2.InstID2.FktID2.OPType2 Con1 ≠ Con2 Con1 is established before Con 2 is established. The second connection (Con2) has the same priority as the first connection (Con1): **Prio2 = Prio1**
Behavior	Connection #1 is established. Block 0 of Connection #1 is transmitted Block 1 of Connection #1 is transmitted … Meanwhile Connection #2 is established Connection #2 is held (paused) by the DSI (on reception of 0-FRAME), until the current block of Connection #1 is completed. Connection #1 is held (paused) by the DSI, as soon as the block was completed. Connection #2 is continued (triggered by a Negative Acknowledge by the the DSI) Block 0 of Connection #2 is transmitted Connection #1 is kept in hold state by the DSI. Connection #2 is held (paused) by the DSI, as soon as the block was completed. Connection #1 is continued (triggered by a Negative Acknowledge by the the DSI). ... and so on .... Data from a further connection using a lower priority is only accepted, when no other connection using a higher (or same) priority is serviced. During this time, the DSI pauses the low priority connection by sending hold commands.

Summary:
- All connections using the same priority are handled in an interleaved way (round robin)
- Interleaving on block level, not on MHP Data frame level
- Connections using a lower priority are NOT serviced, as long as higher priority connections are active (i.e., existent and not held by the DSO or the DSI application)
- As soon as all higher priority connections are forced to hold state (by the application of the DSO or the DSI) the next lower priority is serviced. If multiple lower priority connections are waiting the same interleaving approach applies.

8.5.3 Coexistent Connections Using Higher Priority

Scenario	Coexistent connections from multiple DSOs using a higher priority (Prio2 > Prio1)
Description	Different DSO devices try to establish multiple connections to one and the same target device. Each DSO targets to another object in the same DSI. Con1: SrcAddr1.TgtAddr.FBlockID1.InstID1.FktID1.OPType1 Con2: SrcAddr2.TgtAddr.FBlockID2.InstID2.FktID2.OPType2 Con1 ≠ Con2 Con1 is established before Con 2 is established. The second connection (Con2) has a higher priority than the first connection (Con1): Prio2 > Prio1
Behavior	Connection #1 is established. Block 0 of Connection #1 is transmitted Block 1 of Connection #1 is transmitted Block 2 of Connection #1 is transmitted Meanwhile Connection #2 is established. Connection #2 is held by the DSI (till current block of connection #1 is completed) Connection #1 is held by the DSI (on reception of next 0-FRAME) Block 0 of Connection #2 is transmitted Block 1 of Connection #2 is transmitted Block 2 of Connection #2 is transmitted Block N of Connection #2 is transmitted (till all available blocks are transmitted) Connection #2 is held (paused) by the DSO. Connection #1 is continued by the DSI (by sending a Negative Acknowledge) Block 3 of Connection #1 is transmitted Block 4 of Connection #1 is transmitted Block 5 of Connection #1 is transmitted … and so on…. Data on a low priority connection is going to be received only, if no other connection with a higher priority is waiting to be serviced. The connection is not switched by the DSI, before the previous block (of a lower priority) has been completed.

8.6 Competing Connections from Multiple DSOs

Scenario	Competing connections from multiple DSOs
Description	Different DSO devices try to establish multiple connections to one and the same target device. All DSO devices aim to the same object in the same DSI. Con1: SrcAddr1.TgtAddr.FBlockID.InstID.FktID.OPType Con2: SrcAddr2.TgtAddr.FBlockID.InstID.FktID.OPType Con1 = Con2 Con1 is established before Con 2 is established.
Behavior	The behavior depends on the priority of both connections. Case A) Prio2 ≤ Prio1 Case B) Prio2 > Prio1 See subchapters below.

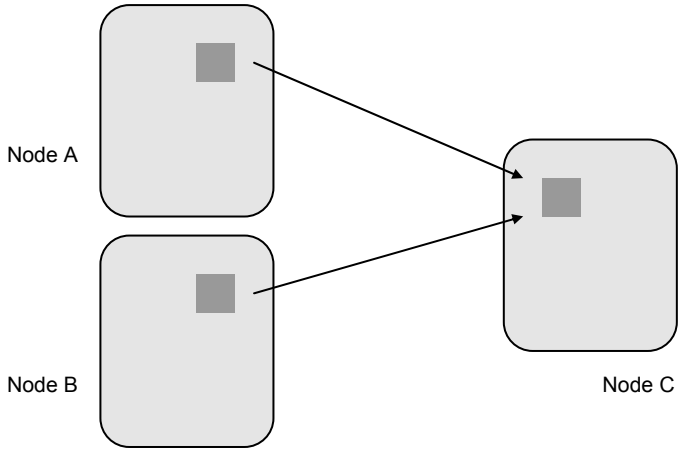

Figure 8-6: Concurrent Connections - Competing Connections from Multiple DSOs

8.6.1 Competing Connections Using Same or Lower Priority

Scenario	Competing connections from multiple DSO using the same or lower priority
Description	Different DSO devices try to establish multiple connections to one and the same target device. All DSO devices aim to the same object in the same DSI. Con1: SrcAddr1.TgtAddr.FBlockID.InstID.FktID.OPType Con2: SrcAddr2.TgtAddr.FBlockID.InstID.FktID.OPType Con1 = Con2 Con1 is established before Con 2 is established. The second connection (Con2) has the same or a lower priority as the first connection (Con1): Prio2 ≤ Prio1
Behavior	Connection #1 is established. Block 0 of Connection #1 is transmitted Block 1 of Connection #1 is transmitted Block 2 of Connection #1 is transmitted … Meanwhile Connection #2 tries to get established Connection #2 is rejected (cannot be established) Block 3 of Connection #1 is transmitted Block 4 of Connection #1 is transmitted … and so on… A new connection to the **same object** (same FBlockID.InstID.FktID.OPType in the same device) can only be established, if the **priority is higher** than the priority of an already existing connection.

8.6.2 Competing Connections Using Higher Priority

Scenario	Competing connections from multiple DSOs using a higher priority (Prio2 > Prio1)
Description	Different DSO devices try to establish multiple connections to one and the same target device. All DSO devices aim to the same object in the same DSI. Con1: SrcAddr1.TgtAddr.FBlockID.InstID.FktID.OPType Con2: SrcAddr2.TgtAddr.FBlockID.InstID.FktID.OPType Con1 = Con2 Con1 is established before Con 2 is established. The second connection (Con2) has a higher priority as the first connection (Con1): Prio2 > Prio1
Behavior	Connection #1 is established. Block 0 of Connection #1 is transmitted Block 1 of Connection #1 is transmitted Block 2 of Connection #1 is transmitted … Meanwhile Connection #2 is established. Connection #1 is terminated by MHP, due to a higher priority onto the same object. Block 0 of Connection #2 is transmitted Block 1 of Connection #2 is transmitted Block 2 of Connection #2 is transmitted .... and so on .... An already existing connection using a lower priority is killed, as soon as a new connection is established to the **same object** (same FBlockID.InstID.FktID.OPType in the same device).

9 Appendix A: Data Flows

This Appendix contains flow charts that describe MHP. The flow charts shall be read as an overview, the requirements for the protocol are not specified here.

9.1 DSO

9.1.1 Establishing Connection

Note: *For legacy implementations (which refers to MOST25 and MOST50 devices based on MOST Specifications with Major Revision 2) please see 10.2 Miscellaneous Requirements.*

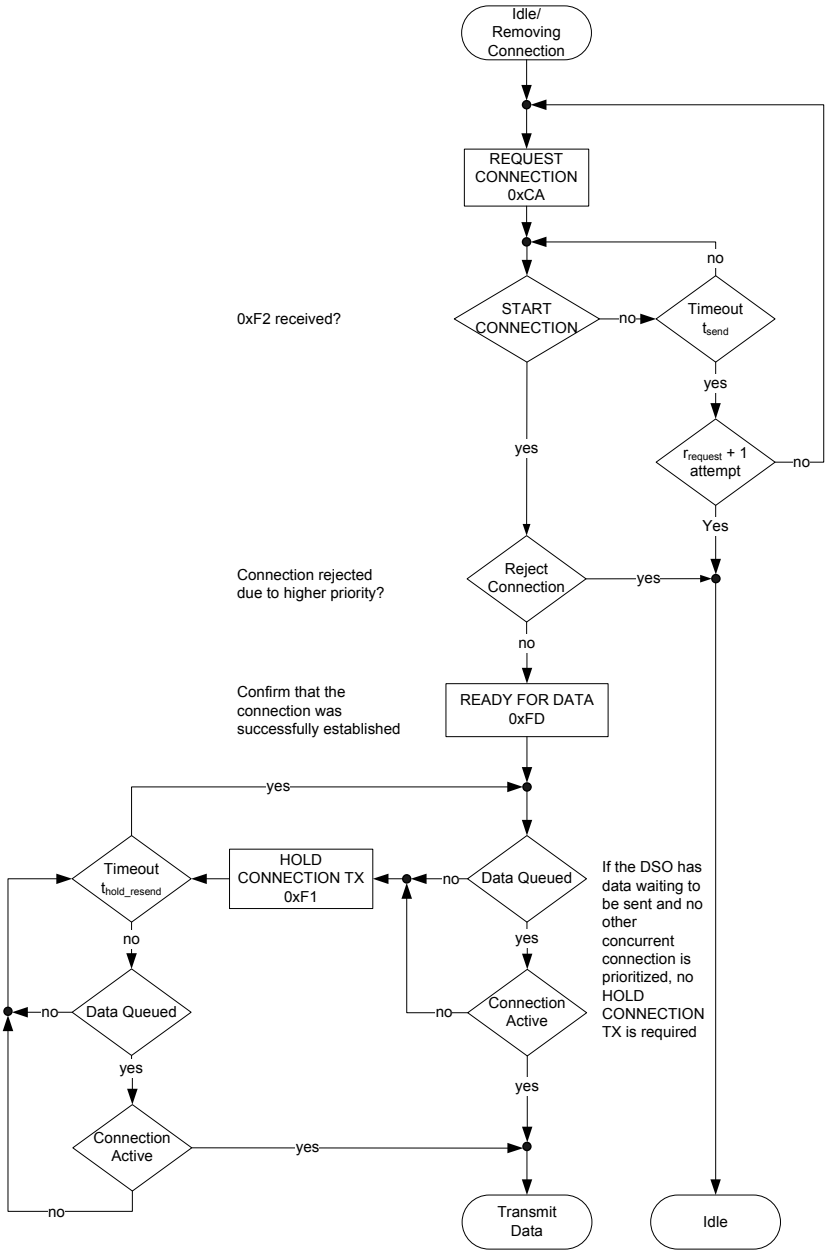

Figure 9-1: DSO: Establishing Connection

Note: *If a connection cannot be established after $r_{request}$ + 1 attempts, the stability of the system should be checked by the application. This helps in avoiding endless loops.*

9.1.2 Transmit Data

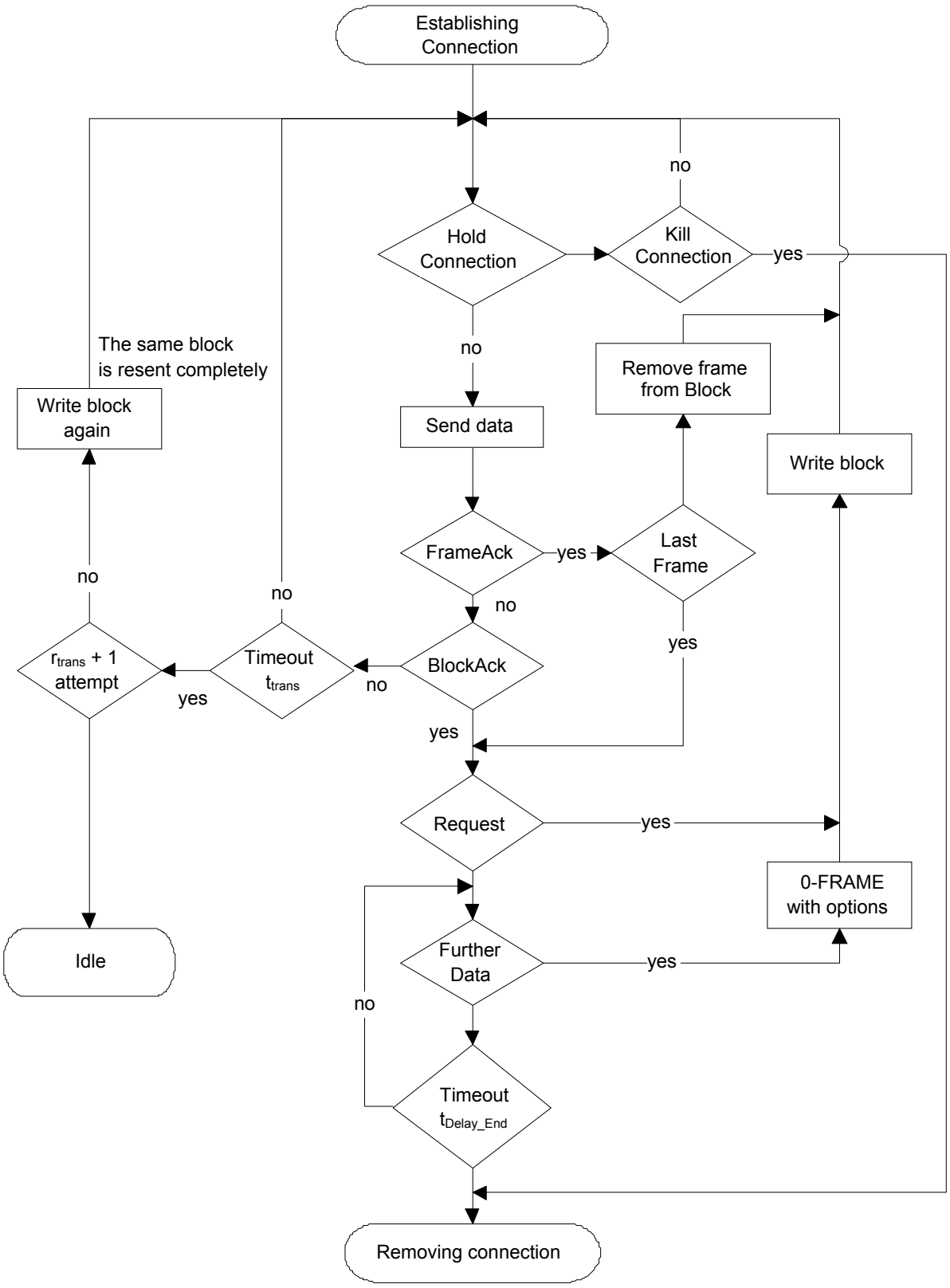

Figure 9-2: DSO: Transmit Data

9.1.3 Removing Connection

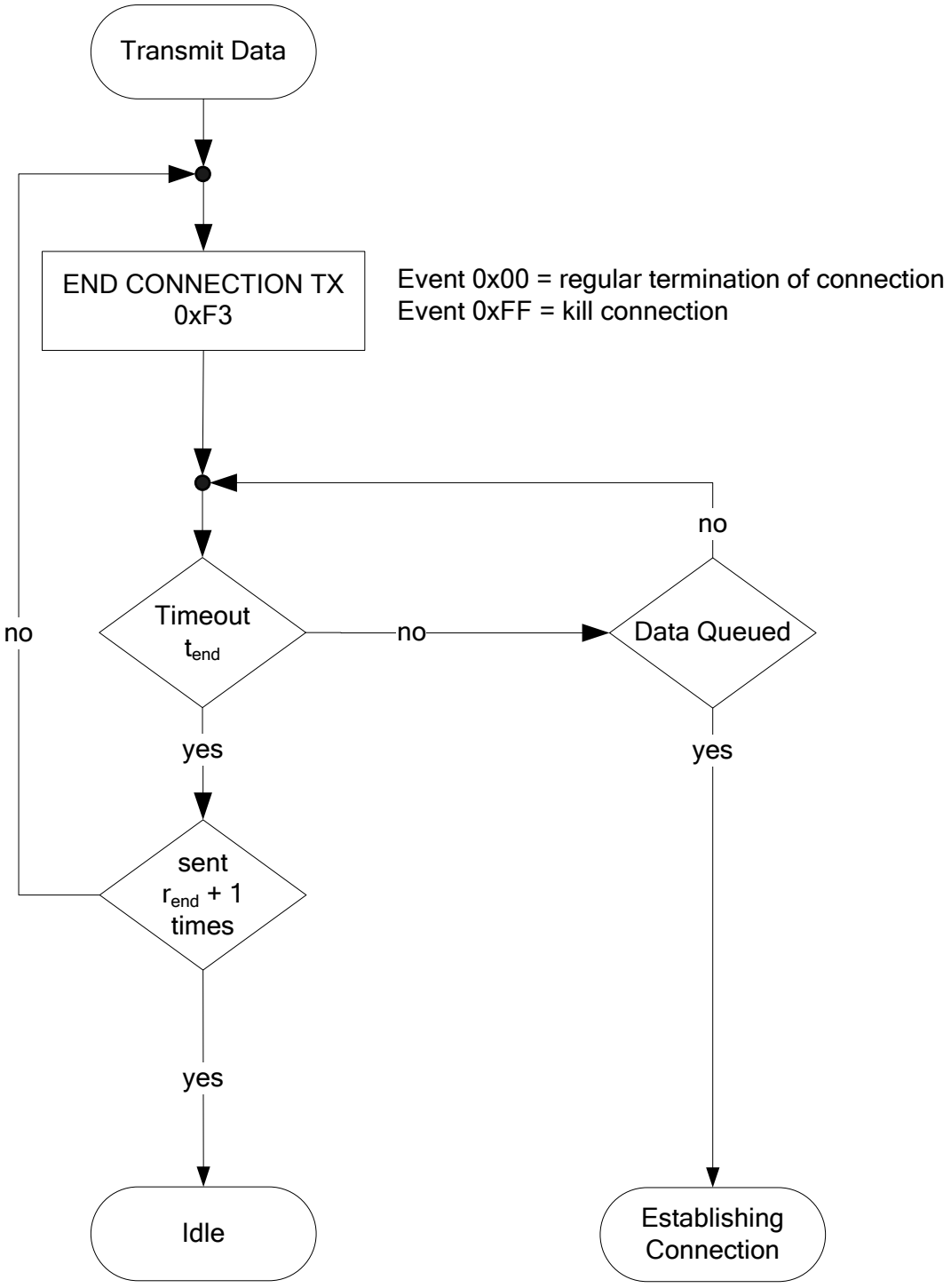

Figure 9-3: DSO: Removing Connection

9.2 DSI

9.2.1 Establishing Connection

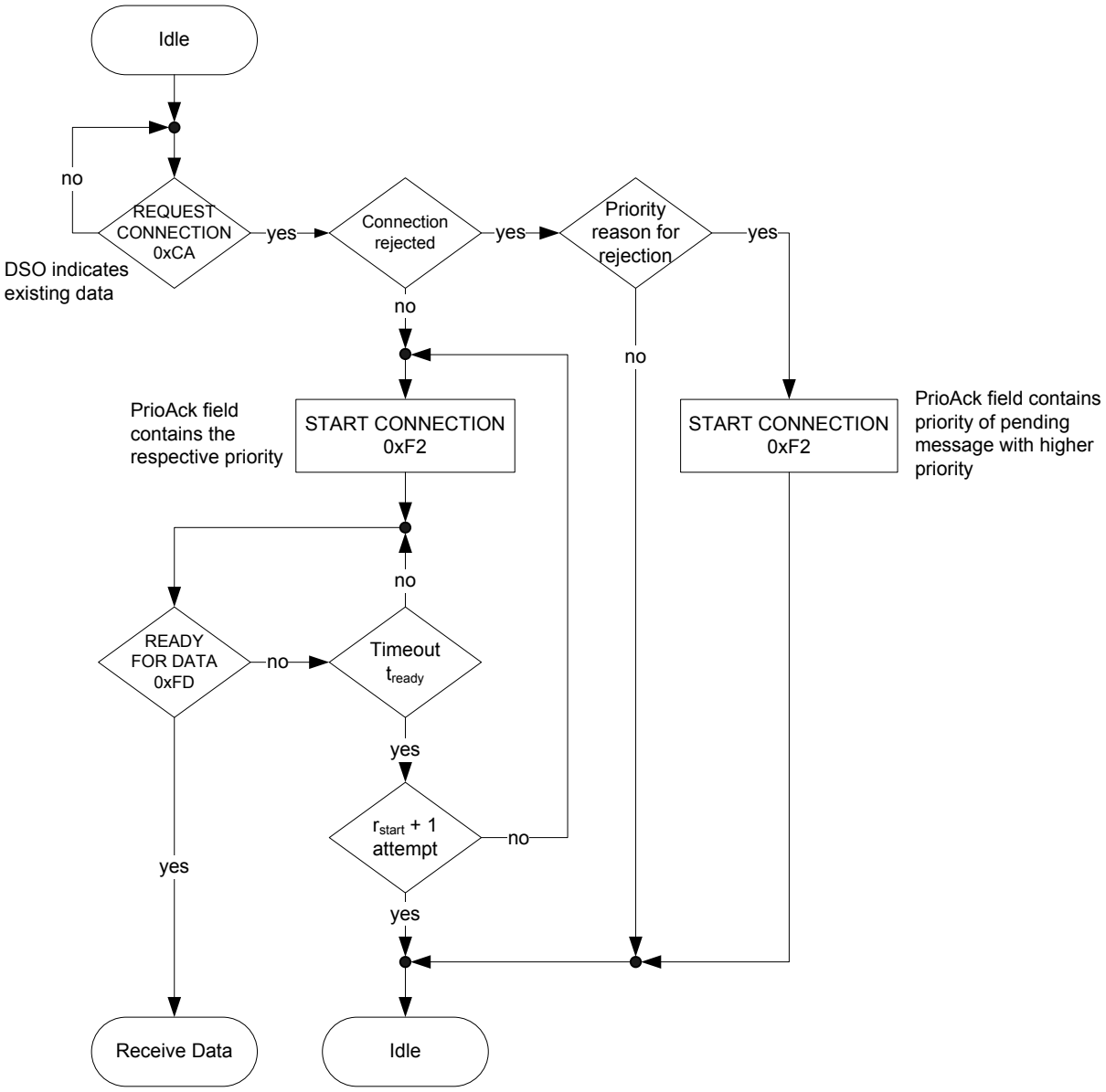

Figure 9-4: DSI: Establishing Connection

9.2.2 Receive Data

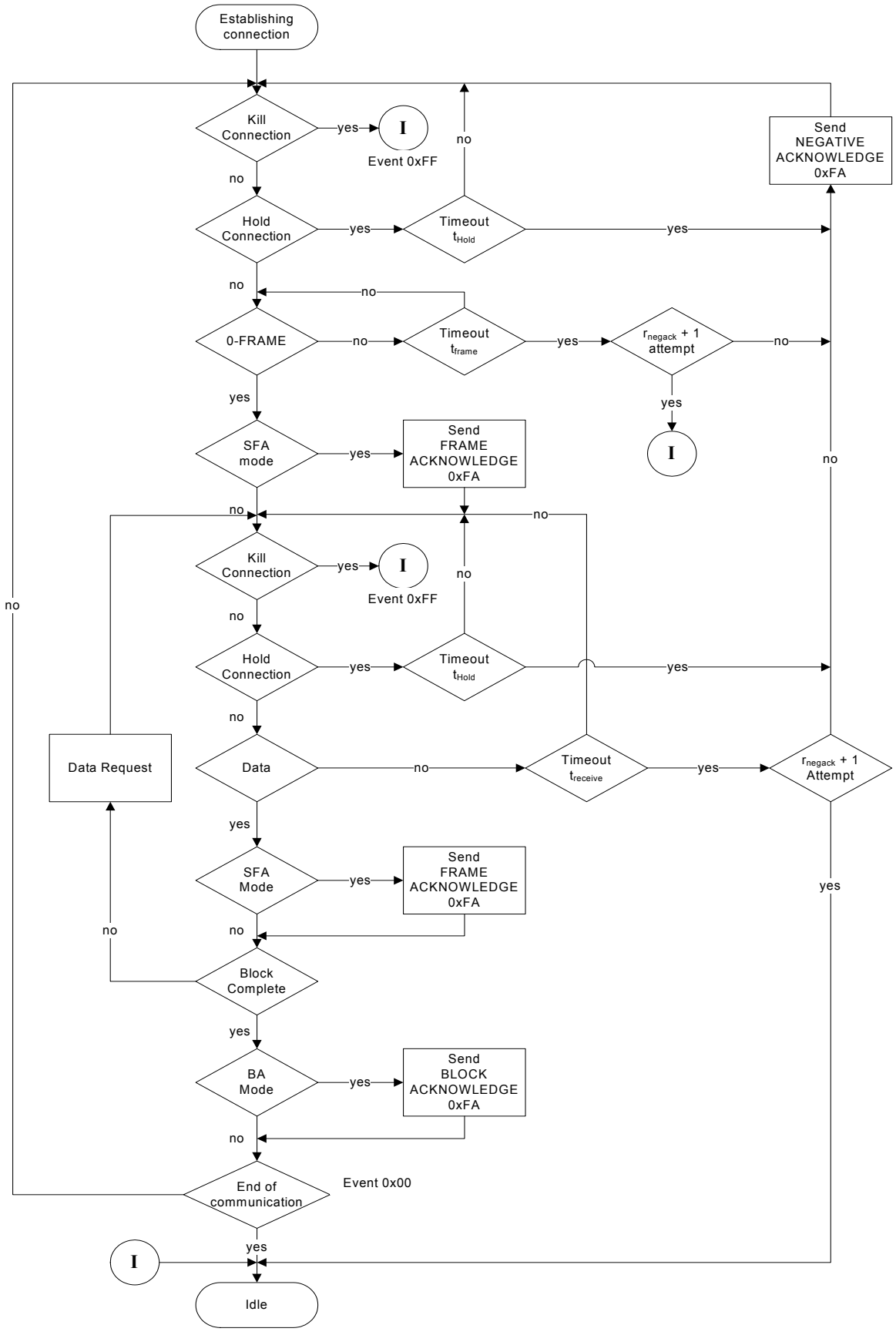

Figure 9-5: DSI: Receive Data

10 Appendix B: Conditional Features (informative)

This appendix lists features that are not necessarily available for all speed grades or implementations. The term “legacy implementations” refers to MOST25 and MOST50 devices that were built based on MOST Specification Revisions with Major Revision 2.

10.1 MHP Frame Size

The MHP frame size that a device supports is determined by the revision of the MOST Specification the device complies with but also by device-specific parameters, for example, the I/O interface. The following table lists the most typical MHP frame size limits.

Variant	Max. MHP frame size (data area, excluding destination and source addresses)	Available for MHP Commands	Maximum number of usable MHP Data bytes (typical NDF/NDFAck values)
MOST Spec. Rev. 2.5	1014	42	1006
MOST Spec. Rev. 3.0	1524	42	1516
Exemplary I/O limitation (I^2C)	48	42	40

Table 10-1: MHP Frame Size Limits

10.2 Miscellaneous Requirements

Section and item	Legacy Implementations (MOST25 / MOST50)
Section 4.1.4.2.1: FRAME ACKNOWLEDGE, HoldFlag	The optional field HoldFlag is not supported: – It is never set by the DSI. – It is ignored by the DSO.
Section 4.1.2.6: START CONNECTION, MaxBlkSize	The optional field MaxBlkSize is not supported. – It is never set by the DSI. – It is ignored by the DSO.
Sections 7.1.1 and 9.1.1 contain an alternative use case within the basic flow: “Open a connection without sending data”	The use case is not supported. The connection is opened only on demand, that is, if data is available for transport.
Section 7.2.5: In the case that the DSO is sending MHP Data frames for a previous MULTIPLE FRAMES REQUEST and a new MULTIPLE FRAMES REQUEST is received, the requests are merged by the DSO.	MULTIPLE FRAMES REQUESTs are queued in the DSO. The requests are not merged.
Section 8.5: Coexistent Connections from Multiple DSOs Case A) Prio2 < Prio1 Case B) Prio2 = Prio1	Case A and Case B are not distinguished. In both cases (Prio2 ≤ Prio1), the behavior that is described in section 8.5.1 applies.

Table 10-2: Restrictions for Legacy Implementations

11 Appendix C: Index of Figures

12 Appendix D: Index of Tables

Notes: