



Association for Standardisation of  
Automation and Measuring Systems

---

## **ASAM MCD-1 (XCP on FlexRay)**

Universal Measurement and Calibration  
Protocol

### **FlexRay Transport Layer**

Version 1.3.0

Date: 2015-05-01

### **Associated Standard**

## **Disclaimer**

This document is the copyrighted property of ASAM e.V.  
Any use is limited to the scope described in the license terms. The license  
terms can be viewed at [www.asam.net/license](http://www.asam.net/license)

---

**Table of Contents**

<b><u>1</u></b>	<b><u>Foreword</u></b>	<b><u>5</u></b>
<b><u>2</u></b>	<b><u>Introduction</u></b>	<b><u>6</u></b>
<b><u>3</u></b>	<b><u>Relations to Other Standards</u></b>	<b><u>7</u></b>
3.1	Backward Compatibility to Earlier Releases	7
3.1.1	Flexray Transport Layer	7
3.1.2	The Compatibility Matrix	7
3.2	References to other Standards	7
<b><u>4</u></b>	<b><u>The XCP Transport Layer for FlexRay</u></b>	<b><u>8</u></b>
4.1	FlexRay Frames Transporting XCP	8
4.1.1	LPDU_ID	8
4.1.2	FlexRay Frame Type	9
4.1.3	Maximum FlexRay Length	9
4.2	Addressing	10
4.2.1	XCP-Dedicated LPDU_IDS on the Network	10
4.2.2	XCP-Dedicated Buffers in the Slave	10
4.2.2.1	Assignment of LPDU_IDS to Buffers	11
4.2.2.2	Assignment of XCP Packets to Buffers	12
4.2.3	Multiple XCP Slaves Using Exclusive LPDU_IDS	13
4.2.4	Multiple XCP Slaves Competing for LPDU_IDS	13
4.2.4.1	Node Address for XCP	13
4.2.4.2	Mutually Exclusive Activation of XCP-Dedicated Buffers	14
4.2.4.3	Reconfiguration of XCP-Dedicated Buffers	14
4.2.5	Use of XCP-Dedicated Buffers Within the Slave	14
4.3	Communication Model	15
4.4	Header and Tail	15
4.4.1	Header	16
4.4.1.1	Node Address for XCP (NAX)	16
4.4.1.2	Counter (CTR)	16
4.4.1.3	Fill	17
4.4.1.4	Length (LEN)	17
4.4.2	Tail	17
4.4.2.1	Single XCP Message in One FlexRay Frame	17
4.4.2.2	Multiple XCP Messages in One FlexRay Frame	18
4.5	The Limits of Performance	19
<b><u>5</u></b>	<b><u>Specific Commands for XCP on FlexRay</u></b>	<b><u>21</u></b>
5.1	Assign/Deassign FlexRay LPDU Identifiers to Buffers	21
5.2	Activate Communication of a FlexRay Buffer	23
5.3	Deactivate Communication of a FlexRay Buffer	24
5.4	Get DAQ List FlexRay Buffer(s)	25
5.5	Set DAQ List FlexRay Buffer(s)	26

---

5.6	DAQ Clock Multicast	26
5.7	Communication Error Handling	28
5.7.1	Error code handling	28
<b>6</b>	<b><u>Specific Events for XCP on FlexRay</u></b>	<b>29</b>
<b>7</b>	<b><u>Interface to ASAM MCD-2 MC Description File</u></b>	<b>30</b>
7.1	ASAM MCD-2 MC aml for XCP on FlexRay	30
7.2	IF_DATA Example for XCP on FlexRay	30
<b>8</b>	<b><u>Interface to ASAM MCD-2 NET [FBX] Description File</u></b>	<b>31</b>
<b>9</b>	<b><u>Symbol and Abbreviated Terms</u></b>	<b>32</b>
<b>10</b>	<b><u>Bibliography</u></b>	<b>33</b>
	Figure Directory	34
	Table Directory	35

---

# 1 FOREWORD

XCP is short for Universal Measurement and Calibration Protocol. The main purpose is the data acquisition and calibration access from electronic control units. Therefore a generic protocol layer is defined. As transport medium different physical busses and networks can be used. For each authorized transport medium a separated transport layer is defined. This separation is reflected in standard document structure, which looks like follows:

- One Base Standard
- Associated Standards for each physical bus or network type

The Base Standard describes the following content:

- Protocol Layer
- Interface to ASAM MCD-2 MC
- Interface to an external SEED&KEY function
- Interface to an external Checksum function
- Interface to an external A2L Decompression/Decrypting function
- Example Communication sequences

This associated standard describes the XCP on FlexRay transport layer.

The "X" inside the term XCP generalizes the "various" transportation layers that are used by the members of the protocol family. Because XCP is based on CCP the "X" shall also show that the XCP protocol functionality is extended compared to CCP.

---

## 2 INTRODUCTION

This standard describes how XCP is transported on FlexRay [3] as transport layer. It is shown how addressing shall be realized and the usage of the different communication models (see Chapter 4.3). Also the content of the control field of the XCP message frame format is described. For details about the frame format structure please refer the base standard [1]. Chapter 5 shows the specific commands which are defined for the transport on FlexRay. The interface to the ASAM MCD-2 MC description file is described in chapter 7. The network information can be found in the FIBEX description file. For further information please see chapter 8.

---

## 3 RELATIONS TO OTHER STANDARDS

### 3.1 BACKWARD COMPATIBILITY TO EARLIER RELEASES

#### 3.1.1 FLEXRAY TRANSPORT LAYER

This Transport layer uses the version number 1.3. This version number is represented as 16 bit value, where the high byte contains the major version (U) and low byte contains the minor version (V) number.

If this associated standard is modified in such a way that a functional modification in the slave's driver software is needed, the higher byte of its XCP Transport Layer Version Number will be incremented. This could be the case e.g. when modifying the parameters of an existing command or adding a new command to the specification.

If this associated standard is modified in such a way that it has no direct influence on the slave's driver software, the lower byte of its XCP Transport Layer Version Number will be incremented. This could be the case e.g. when rephrasing the explaining text or modifying the AML description.

The slave only returns the most significant byte of the XCP Transport Layer Version Number for the current Transport Layer in the response upon CONNECT.

#### 3.1.2 THE COMPATIBILITY MATRIX

The Compatibility Matrix gives an overview of the allowed combinations of Protocol Layer and Transport Layer parts. For details about the Compatibility Matrix please refer the base standard [\[1\]](#).

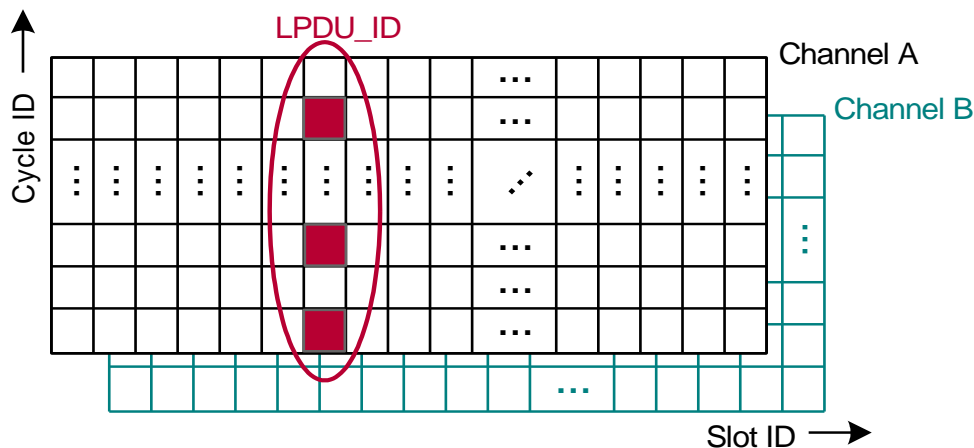
### 3.2 REFERENCES TO OTHER STANDARDS

For details about the References to other standards please refer the base standard [\[1\]](#).

## 4 THE XCP TRANSPORT LAYER FOR FLEXRAY

### 4.1 FLEXRAY FRAMES TRANSPORTING XCP

#### 4.1.1 LPDU\_ID



**Figure 1** Data link layer protocol data unit identifier

Basically a FlexRay slot-ID identifies a FlexRay frame.

However, in the dynamic part of a cycle multiple FlexRay nodes can share a slot for transmitting by using the cycle counter as a demultiplexer. A node can transmit in the slot if the current cycle counter matches an element of its set of valid cycle counter values. For a specific slot-ID the set of valid cycle counter values is generated by using the following formula:

$$\text{valid cycle counter} = (b + n \cdot 2^r) \bmod 64$$

constraint:  $b < 2^r$

└─ cycle repetition

└─ generator

└─ offset

$r = \{0, 1, \dots, 6\}$

$n = \{0, 1, \dots, 64\}$

$b = \{0, 1, \dots, 63\}$

Additionally, up to two FlexRay nodes can share a slot that is not used redundantly, by using the channel as a demultiplexer.

A FlexRay Data Link Layer Protocol Data Unit Identifier ( $\text{FLX\_LPDU\_ID}$ ) generalizes the notion of a slot-ID by taking into account these multiplexing possibilities. The following 4 parameters describe a FlexRay LPDU Identifier:

- FlexRay slot identifier ( $\text{FLX\_SLOT\_ID}$ )
- cycle counter offset ( $\text{OFFSET}$ )
- cycle counter repetition ( $\text{CYCLE\_REPETITION}$ )
- FlexRay channel ( $\text{FLX\_CHANNEL}$ ).

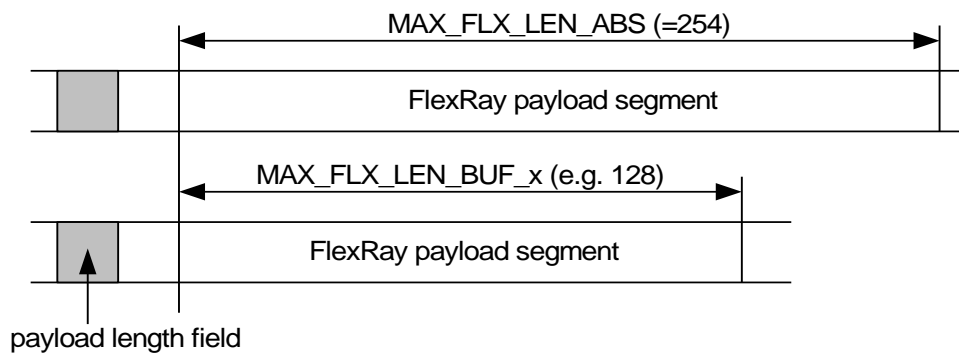


XCP on FlexRay messages are transported inside FlexRay LPDUs.

#### 4.1.2 FLEXRAY FRAME TYPE

An XCP on FlexRay message can be located in the static, guaranteed dynamic or non-guaranteed dynamic part of any cycle on any channel of the network. FlexRay distinguishes between LPDU\_IDs with a transmission mode of type “state driven” and “event driven”. If there are no updated data to be transmitted, for an event driven LPDU\_ID the corresponding currently used FlexRay cell in the static part of a cycle carries a Null Frame and in the dynamic part of a cycle stays empty. All XCP on FlexRay LPDU\_IDs always are event driven.

#### 4.1.3 MAXIMUM FLEXRAY LENGTH



**Figure 2 Theoretical (top) vs. practical (bottom) max. length of payload segment**

The maximum theoretical data length of a FlexRay frame and therefore the maximum length of an XCP on FlexRay message is  $\text{MAX\_FLX\_LEN\_ABS} = 254$  (bytes).

In actual practice the maximum data length of a FlexRay frame can be considerably shorter due to FlexRay controller limitations. The actual maximum data length of a FlexRay frame, which a specific slave is able to receive or transmit in a specific XCP-dedicated buffer, is called  $\text{MAX\_FLX\_LEN\_BUF}_x$ .

The initial maximum data length of a FlexRay frame, which a specific slave is able to receive or transmit in a specific XCP-dedicated buffer, is called  $\text{MAX\_FLX\_LEN\_BUF}_x\text{init}$ .

The actual length of the FlexRay payload segment of a specific FlexRay frame, expressed in bytes, is called  $\text{FLX\_LEN\_BUF}_x$ .

The payload length field in the FlexRay frame header always is set to the number of payload data bytes divided by two. Therefore the following relation always applies:

$$\text{payload length field} = \text{FLX\_LEN\_BUF}_x / 2$$

## 4.2 ADDRESSING

### 4.2.1 XCP-DEDICATED LPDU\_IDS ON THE NETWORK

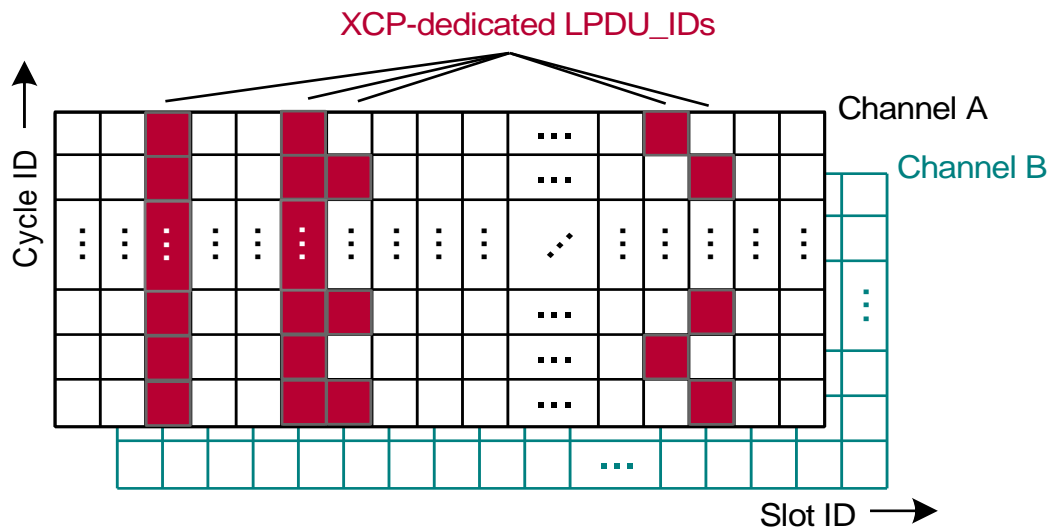


Figure 3 XCP dedicated LPDU\_IDS on the network

The FlexRay system designer for a specific network specifies where the LPDU\_IDS that can be used for XCP communication are located within the schedule.

These XCP-dedicated LPDU\_IDS define the maximum bandwidth available for XCP from the point of view of the network.

The XCP-dedicated LPDU\_IDS may be non-concatenated.

### 4.2.2 XCP-DEDICATED BUFFERS IN THE SLAVE

An XCP-slave connected to a FlexRay network has a limited number of transmit / receive buffers that are exclusively dedicated to XCP communication.

Before a slave can use one of those XCP-dedicated buffers for XCP communication, the buffer has to be linked to one of the XCP-dedicated LPDU\_IDS from the network.

Also master and slave for every buffer have to know what type of XCP packet is transported in the corresponding LPDU\_ID.

In actual practice not every slave offers the flexibility to let all parameters for its XCP-dedicated buffers be freely configured.

The ASAM MCD-2 MC description file [4] describes to which extent the XCP-dedicated buffers of a specific slave can be configured for XCP communication.

#### 4.2.2.1 ASSIGNMENT OF LPDU\_IDS TO BUFFERS

**Table 1 Example of the assignment of LPDU\_IDs to XCP-dedicated buffers**

Buffer	FLX_LPDU_ID				MAX_FLX_LEN_BUF_x
	FLX_SLOT_ID	OFFSET	CYCLE_REPETITION	CHANNEL	
Buffer_1	123	0	1	A	32
Buffer_2	124	1	2	A	32
Buffer_3	125	0	2	A	32
Buffer_4	126	(*)	(*)	A	64
Buffer_5	(*)	(*)	(*)	(*)	64

Legend:

123, A,... Example for initial values  
( ) configurable  
\* uninitialized parameter

An LPDU\_ID parameter of a specific buffer can be non-configurable. The XCP on FlexRay master cannot change the value of this parameter. The buffer always has the same value which also is the initial value after reset for this parameter.

An LPDU\_ID parameter of a specific buffer also can be configurable. The XCP on FlexRay master with FLX\_ASSIGN can change the value of this parameter. If the parameter is uninitialized after reset, the XCP on FlexRay master has to assign a value to this parameter.

The values of non-configurable LPDU\_ID parameters and the values assigned by the XCP on FlexRay master, always have to result in an LPDU\_ID that is in the valid set of XCP-dedicated LPDU\_IDs of the network.

In the example Buffer\_1 and Buffer\_2 have completely unconfigurable LPDU\_ID parameters.

For Buffer\_3 the Slot-ID is initialized but it could be changed by the XCP on FlexRay master.

For Buffer\_4 the parameters for the set of cycle counters are uninitialized and have to be configured by the XCP on FlexRay master.

Buffer\_5 is completely uninitialized and has to be completely configured by the XCP on FlexRay master.

MAX\_FLX\_LEN\_BUF\_x of a specific buffer can be non-configurable. The XCP on FlexRay master then cannot change the value of this parameter. MAX\_FLX\_LEN\_BUF\_x always equals MAX\_FLX\_LEN\_BUF\_x\_init.

MAX\_FLX\_LEN\_BUF\_x of a specific buffer also can be configurable. The XCP on FlexRay master with FLX\_ASSIGN then can change the value of this parameter.

In the example Buffer\_1 till Buffer\_3 have non-configurable values for MAX\_FLX\_LEN\_BUF\_x.

For Buffer\_4 and Buffer\_5 MAX\_FLX\_LEN\_BUF\_x is initialized but it could be changed by the XCP on FlexRay master.

#### 4.2.2.2 ASSIGNMENT OF XCP PACKETS TO BUFFERS

**Table 2 Example of the assignment of XCP packets to buffers**

Buffer	CMD	RES, ERR	EV, SERV	DAQ	STIM
Buffer_1	✓	✗	✗	✗	✓
Buffer_2	✗	✓	✓	✓	✗
Buffer_3	✗	(*)	(*)	(ü)	✗
Buffer_4	(*)	(*)	(*)	(*)	(*)
Buffer_5	(*)	(*)	(*)	(*)	(*)

Legend:

- ✓ initial assignment
- ( ) configurable
- \* allowed assignment
- ✗ not allowed

Master and slave for every buffer have to know what type of XCP packet is transported in the corresponding `LPDU_ID`.

A buffer can have a fixed assignment to always transport a specific type of XCP packet. The XCP on FlexRay master cannot change this assignment.

A buffer also can have an initial assignment to transport a specific type of XCP packet. The XCP on FlexRay master then with `FLX_ASSIGN` can change this assignment.

A buffer for a specific XCP packet type initially can be unassigned. The XCP on FlexRay master however with `FLX_ASSIGN` could request this buffer to transport this specific XCP packet type.

Finally a buffer cannot allow to have an assignment for transporting a specific XCP packet type.

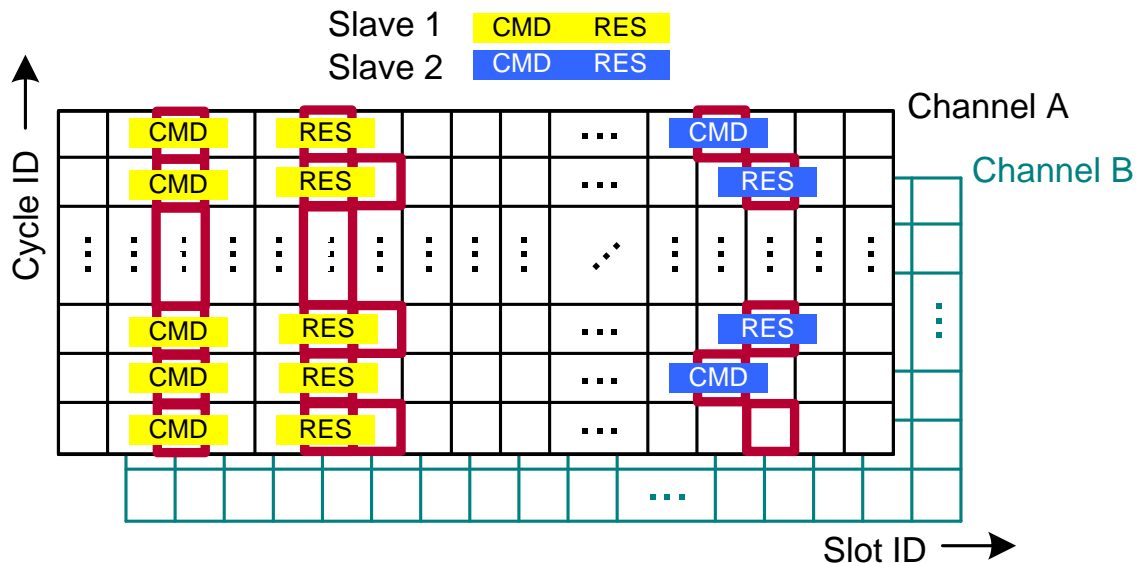
In the example Buffer\_1 has a fixed assignment to transport CMD and STIM packets. Buffer\_1 can neither transport RES, ERR, EV, SERV nor DAQ. Buffer\_1 for the slave looks like a “receive only buffer”.

Buffer\_2 has a fixed assignment to transport RES, ER, EV, SERV and DAQ. Buffer\_2 can neither transport CMD nor STIM. Buffer\_2 for the slave looks like a “transmit only buffer”.

Buffer\_3 has an initial assignment to transport DAQ. However, the XCP on FlexRay master could request this buffer to transport RES, ERR, EV or SERV. The XCP on FlexRay master cannot request this buffer to transport CMD neither STIM.

All other buffers are completely freely configurable. The XCP on FlexRay master can request these buffers to transport any type of XCP packet

### 4.2.3 MULTIPLE XCP SLAVES USING EXCLUSIVE LPDU\_IDS



**Figure 4 Multiple XCP slaves using exclusive LPDU\_IDS**

If an XCP master sets up a connection to an XCP on FlexRay slave, it gets the information about the XCP-dedicated LPDU\_IDS of the network from the FIBEX description file [2]. The XCP master gets the information about the slave's XCP-dedicated buffers from the ASAM MCD-2 MC description file ([4]) for this specific slave.

If the XCP master sets up a connection to a single slave, it has to check whether the values of non-configurable LPDU\_ID parameters of the slave result in an LPDU\_ID that is in the valid set of XCP-dedicated LPDU\_IDS of the network.

When assigning the configurable parameters, the XCP master itself has to make sure that these assignments result in an LPDU\_ID that is in the valid set of XCP-dedicated LPDU\_IDS of the network.

If the connected slaves all use their own exclusive LPDU\_IDS that are not shared with the other slaves, the XCP master has not to do any further management of the configuration.

### 4.2.4 MULTIPLE XCP SLAVES COMPETING FOR LPDU\_IDS

Multiple XCP slaves get in competition for XCP-dedicated LPDU\_IDS of the network if their fixed assigned buffers want to use the same LPDU\_ID. Another reason for slaves getting in competition is the limited bandwidth provided by the XCP-dedicated LPDU\_IDS of the network.

Finally the slaves themselves have a limited number of XCP-dedicated buffers to be shared by all XCP packet types.

Multiple XCP slaves sharing XCP-dedicated LPDU\_IDS, is only allowed if those LPDU\_IDS are located in the dynamic part of the FlexRay cycle.

#### 4.2.4.1 NODE ADDRESS FOR XCP

If multiple XCP slaves want to use the same LPDU\_ID in the direction from master to slave, the slaves are distinguished by their Node Address for XCP (NAX). In the direction from slave to master, the NAX is used for distinguishing responses.

Every node in a cluster that is an XCP slave has to have a NAX. Except for NAX = 255, the NAX has to be unique within the FlexRay cluster. In most cases it is practicable to take the diagnostic address of the ECU as the NAX. As an alternative, a cluster-wide unique node address can be taken for NAX.

NAX = 255 is reserved to realize XCP broadcast mechanisms on FlexRay. This special NAX identifier may only be used by the XCP master to send a broadcast command, e.g. the `GET_DAQ_CLOCK_MULTICAST` command. This NAX can thus only exist for messages with direction master to slave.

The first byte within the payload segment of a FlexRay frame that contains XCP on FlexRay contains the NAX. In the direction from master to slave the NAX is a target address and contains the address of the slave. In the direction from slave to master the NAX is a source address and thus also contains the address of the slave. The XCP master itself does not have a NAX.

#### 4.2.4.2 MUTUALLY EXCLUSIVE ACTIVATION OF XCP-DEDICATED BUFFERS

If multiple XCP slaves want to use the same LPDU\_ID for packets of type EV or SERV, a mechanism is needed for indicating to a slave whether it can use the LPDU\_ID for transmission.

With `FLX_ACTIVATE` / `FLX_DEACTIVATE` the XCP on FlexRay master can mutually activate the use of an LPDU\_ID that is shared among multiple slaves.

The XCP master is responsible to activate the communication on LPDU\_IDs in such a way that communication conflicts are avoided in every case.

#### 4.2.4.3 RECONFIGURATION OF XCP-DEDICATED BUFFERS

The bandwidth provided by the XCP-dedicated LPDU\_IDs of the network on FlexRay in typical automotive systems in practice will be limited. The “normal” functional frame transmission already claims most of the busses’ bandwidth.

On the other hand XCP needs a high bandwidth to perform sufficiently. A solution can be to have the XCP on FlexRay master perform a bandwidth distribution. Bandwidth distribution means sharing the limited resources of a FlexRay Bus by assigning the LPDU\_IDs dynamically to dedicated communication channels between master and slave.

With `FLX_ASSIGN` the XCP on FlexRay master can re-assign the configurable buffers of the slaves to the XCP-dedicated LPDU\_IDs provided by the network.

The XCP master is responsible to re-assign the communication on LPDU\_IDs in such a way that communication conflicts are avoided in every case.

The same re-assigning also can be used if the different XCP packet types of a slave internally are competing for the limited number of XCP-dedicated buffers.

### 4.2.5 USE OF XCP-DEDICATED BUFFERS WITHIN THE SLAVE

Not all XCP-dedicated buffers that are available also must be used. Buffers may be left unassigned.

A slave can only use an XCP-dedicated buffer if it has all LPDU\_ID parameters configured.

Furthermore the assignment of XCP packet type(s) has to be configured.

For the master being able to contact the slave, a slave always has to have an initial assignment for CMD. A slave processes a CMD on the specific LPDU\_ID if the message contains its NAX.

For the master being able to contact the slave, a slave always has to have an initial assignment for RES, ERR. A slave transmits RES, ERR on the specific `LPDU_ID` if the previous CMD message contains its NAX.

A slave transmits DAQ and receives STIM on the specific `LPDU_ID` if the XCP on FlexRay master previously started the DAQ lists.

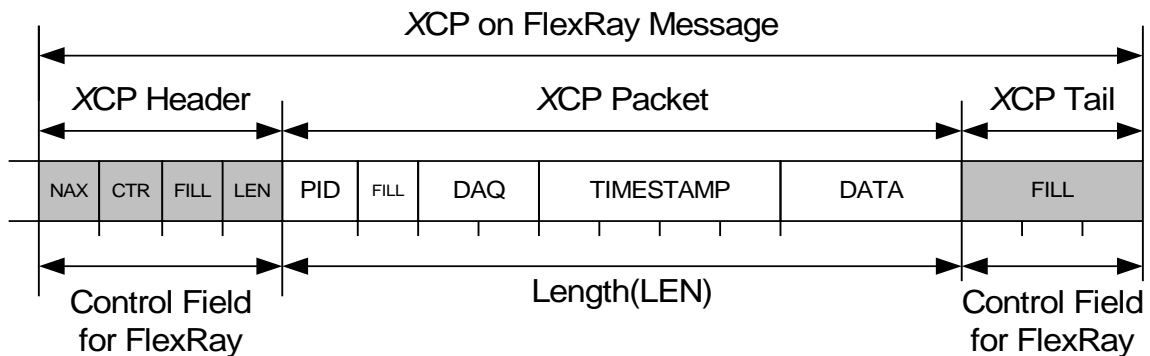
For EV, SERV a slave only transmits on the specific `LPDU_ID` if the XCP on FlexRay master previously with `FLX_ACTIVATE` explicitly has activated the `LPDU_ID`.

For CMD, RES, ERR, DAQ and STIM no explicit activation by the XCP on FlexRay master is necessary.

### 4.3 COMMUNICATION MODEL

XCP on FlexRay makes use of the standard communication model.  
The block transfer communication model is optional.  
The interleaved communication model is not allowed.

### 4.4 HEADER AND TAIL



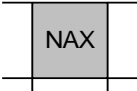
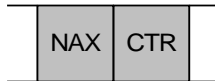
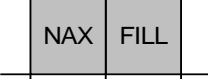
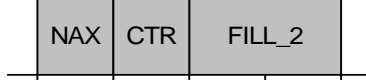

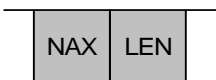

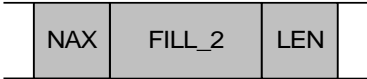

**Figure 5** Header and tail for XCP on FlexRay

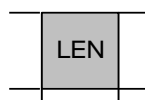
#### 4.4.1 HEADER

The XCP on FlexRay header consists of a control field containing a **N**ode **A**ddress for **X**CP (NAX), an optional **C**oun**T**er (CTR), optional **F**ILL bytes and an optional **L**ENgth (LEN).

Depending upon the requirements on sequencing correctness, alignment and net data throughput, different header types are possible. A slave for transmitting and receiving always uses one and the same header type.

**Table 3 Header types for first XCP message of a FlexRay frame**

	sequence correctness not guaranteed	sequence correcting possible
no concatenation allowed	8 bit alignment: 	8 / 16 bit alignment: 
	16 bit alignment: 	32 bit alignment: 
	32 bit alignment: 	
concatenation allowed	8 / 16 bit alignment: 	8 bit alignment: 
	32 bit alignment: 	16 / 32 bit alignment: 



**Figure 6 Header for subsequent concatenated messages**

##### 4.4.1.1 NODE ADDRESS FOR XCP (NAX)

The first byte within the payload segment of a FlexRay frame that contains XCP on FlexRay, always contains the Node Address for XCP.

If there is a concatenation of multiple XCP messages in one FlexRay frame, then only the header of the first XCP message contains the NAX field. The headers of the subsequent XCP messages do not contain the NAX field.

##### 4.4.1.2 COUNTER (CTR)

If sequence correcting should be possible, the slave has to use a header type that contains the CTR field. The 1-byte field CTR allows to detect missing packets and to correct sequencing errors.



The master has to generate a CTR value when sending a CMD or STIM message. The CTR value must be incremented for each new packet sent from master to the slave.

The slave has to generate a (second, independent) CTR value when sending RES, ERR, EV, SERV or DAQ messages. The CTR value must be incremented for each new packet sent from slave to the master.

If there is a concatenation of multiple XCP messages in one FlexRay frame, then only the header of the first XCP message contains the CTR field. The headers of the subsequent XCP messages do not contain the CTR field.

#### 4.4.1.3 FILL

The performance for accessing the contents of the XCP messages can be increased when an XCP packet starts on an aligned address.

`PACKET_ALIGNMENT_x` ( $x = 8, 16, 32$ ) indicates the alignment a slave uses for its XCP packets. The aligning is performed by optional FILL bytes within the header. The contents of the FILL bytes is "do not care".

If the slave uses a header type without LEN and without CTR, with `HEADER_NAX` the requirement for 8-bit alignment can be fulfilled. With `HEADER_NAX_FILL` the requirement for 16-bit alignment can be fulfilled. With `HEADER_NAX_FILL_3` the requirement for 32-bit alignment can be fulfilled.

If the slave uses a header type without LEN and with CTR, with `HEADER_NAX_CTR` the requirement for 8-bit and 16-bit alignment can be fulfilled. With `HEADER_NAX_CTR_FILL_2` the requirement for 32-bit alignment can be fulfilled.

If the slave uses a header type with LEN and without CTR, with `HEADER_NAX_LEN` the requirement for 8-bit and 16-bit alignment can be fulfilled. With `HEADER_NAX_FILL_2_LEN` the requirement for 32-bit alignment can be fulfilled.

If the slave uses a header type with LEN and with CTR, with `HEADER_NAX_CTR_LEN` the requirement for 8-bit alignment can be fulfilled. With `HEADER_NAX_CTR_FILL_LEN` the requirement for 16-bit and 32-bit alignment can be fulfilled.

If there's a concatenation of multiple XCP messages in one FlexRay frame, then only the header of the first XCP message contains the FILL field. The headers of the subsequent XCP messages do not contain the FILL field.

#### 4.4.1.4 LENGTH (LEN)

To improve the net data throughput, multiple XCP messages may be concatenated to build a single FlexRay frame.

If concatenation of multiple XCP messages in a single FlexRay frame is required, the slave has to use a header type that contains the LEN field. The 1-byte field LEN for each XCP on FlexRay message indicates the number of bytes of the original XCP packet.

If the slave uses a header type that contains the LEN field, a single FlexRay frame containing a single XCP message still is allowed.

### 4.4.2 TAIL

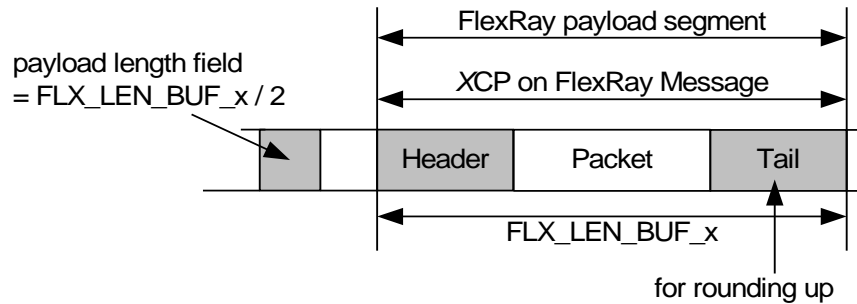
For XCP on FlexRay, the Tail consists of a Control Field containing optional FILL bytes.

#### 4.4.2.1 SINGLE XCP MESSAGE IN ONE FLEXRAY FRAME

The tail serves for rounding up `FLX_LEN_BUF_x` to an even number, which is required by the FlexRay protocol.

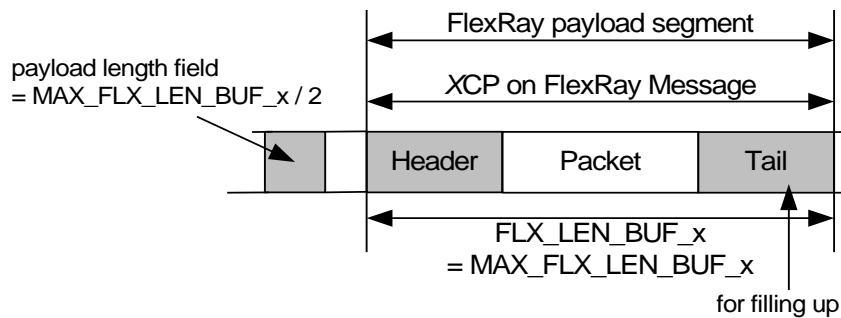
If the length of the XCP header added to the length of the XCP packet results in an odd number, then one FILL byte with a value of "do not care" is appended within the tail to make the XCP on FlexRay message have an even length.

If the length of the XCP header added to the length of the XCP packet gives an even number, then there is no tail.



**Figure 7 Tail for rounding up to even FlexRay length**

For avoiding the recalculation of `HEADER_CRC`, the slave can transmit its FlexRay frames with `MAX_FLX_LEN_BUF_x`. The tail serves for filling up the FlexRay payload segment to `MAX_FLX_LEN_BUF_x`.



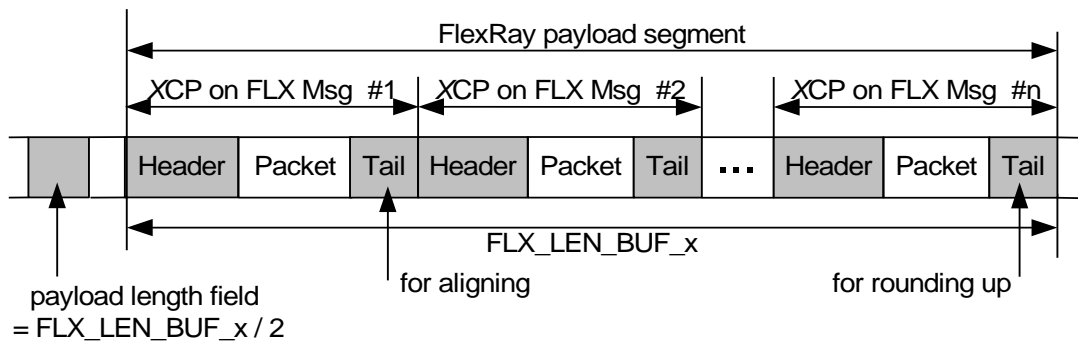
**Figure 8 Tail for filling up to `MAX_FLX_LEN_BUF_x`**

#### 4.4.2.2 MULTIPLE XCP MESSAGES IN ONE FLEXRAY FRAME

If there is a concatenation of multiple XCP messages in one FlexRay frame, the alignment indicated by `PACKET_ALIGNMENT_x` must be fulfilled for each XCP packet.

The aligning is performed by FILL bytes within the tail of each XCP message. The tail of a preceding XCP message is built in such a way that the required alignment of the XCP packet of the subsequent message is fulfilled. Depending on the length of each XCP header added to the length of its XCP packet and depending on the required alignment, the tail can consist of 0, 1, 2 or 3 FILL bytes.

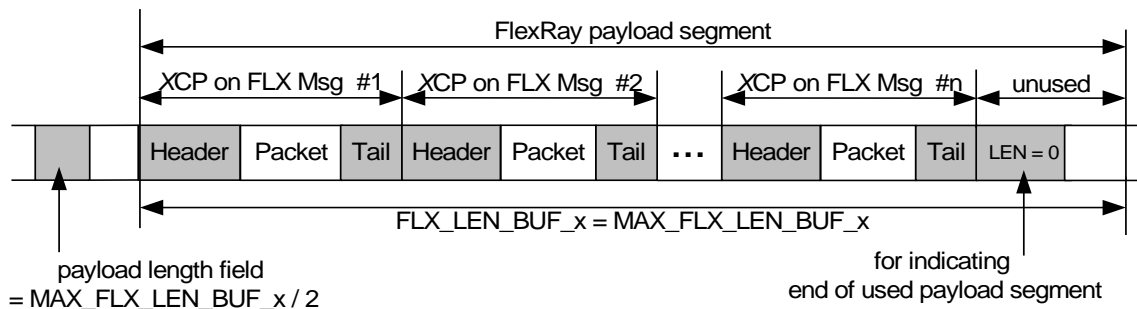
If there is a concatenation of multiple XCP messages in one FlexRay frame, the tail of the last XCP on FlexRay message is built in such a way that the overall length `FLX_LEN_BUF_x` becomes an even number.



**Figure 9** Tail for aligning concatenated XCP messages and rounding up to even FlexRay length

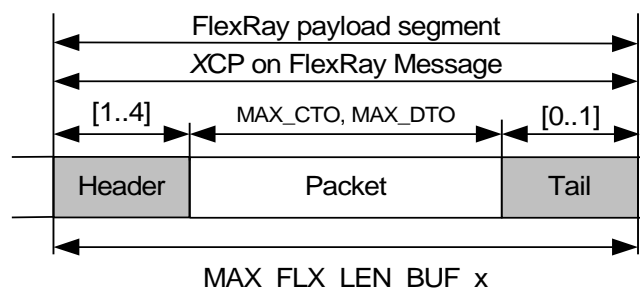
If there is a concatenation of multiple XCP messages in one FlexRay frame and the slave transmits its FlexRay frames with  $MAX\_FLX\_LEN\_BUF\_x$ , an empty XCP on FlexRay message is needed if the unused space after the last XCP message is longer than or equal to the length of a header.

The empty XCP on FlexRay message only consists of a Header with  $LEN = 0$ . The  $LEN = 0$  Header indicates the end of the actually used payload segment.



**Figure 10**  $LEN = 0$  for detecting the end of the used payload segment

## 4.5 THE LIMITS OF PERFORMANCE



**Figure 11** Relation between  $MAX\_CTO$ ,  $MAX\_DTO$  and  $MAX\_FLX\_LEN\_BUF\_x$

Since an XCP on FlexRay message always at least contains a NAX, the practical maximum length of a CTO or a DTO packet of a specific slave is  $MAX\_FLX\_LEN\_BUF\_x - 1$ .

**Table 4** CTO and DTO range

Name	Type	Representation	Range of value
MAX_CTO	Parameter	BYTE	0x08 .. (MAX_FLX_LEN_BUF_x-1)
MAX.DTO	Parameter	WORD	0x0008 .. (MAX_FLX_LEN_BUF_x-1)

If dynamic reconfiguration of parameters is supported, due to FLX\_ASSIGN a slave at least has to support CTO packets having a 12 bytes length.

## 5 SPECIFIC COMMANDS FOR XCP ON FLEXRAY

**Table 5 Command codes overview**

Command	Code	Timeout	Remark
FLX_ASSIGN	0xFF	t <sub>1</sub> _FLX	optional
FLX_ACTIVATE	0xFE	t <sub>1</sub>	optional
FLX_DEACTIVATE	0xFD	t <sub>1</sub>	optional
GET_DAQ_FLX_BUF	0xFC	t <sub>1</sub>	optional
SET_DAQ_FLX_BUF	0xFB	t <sub>1</sub>	optional
GET_DAQ_CLOCK_MULTICAST	0xFA	irrelevant	optional

### 5.1 ASSIGN/DEASSIGN FLEXRAY LPDU IDENTIFIERS TO BUFFERS

Category FlexRay only, optional

Mnemonic FLX\_ASSIGN

**Table 6 FLX\_ASSIGN command structure**

Position	Type	Description
0	BYTE	Command Code = TRANSPORT_LAYER_CMD = 0xF2
1	BYTE	Sub Command Code = FLX_ASSIGN = 0xFF
2	BYTE	FLX_BUF
3	BYTE	XCP_PACKET_TYPE
4	WORD	FLX_SLOT_ID
6	BYTE	OFFSET
7	BYTE	CYCLE_REPETITION
8	BYTE	FLX_CHANNEL (0 = Channel A, 1 = Channel B)
9	BYTE	MAX_FLX_LEN_BUF_x
10	WORD	HEADER_CRC

The “Assign FlexRay L-PDU-Identifiers to buffers” (FLX\_ASSIGN) command assigns an LPDU\_ID to an XCP-dedicated buffer of a slave (FLX\_BUF).

This command also assigns an XCP\_PACKET\_TYPE to the buffer.

**Table 7 XCP\_PACKET\_TYPE parameter bit mask structure**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
x	x	MULTICAST	DAQ	EV_SERV	RES_ERR	STIM	CMD

An XCP-dedicated buffer can either be a receive buffer or a transmit buffer. Depending on the type of buffer, only certain values of `XCP_PACKET_TYPE` are allowed.

Using this command with `XCP_PACKET_TYPE = 0x00` for a specific buffer `FLX_BUF` resets its assignment for both `LPDU_ID` used and type of XCP packet(s) transported. By resetting, the state of the configurable parameters becomes “uninitialized” or “unassigned” as shown in the slave assignment tables with “(\*)”.

Using this command with `XCP_PACKET_TYPE = 0x00` and `FLX_BUF = 0xFF` resets the assignment of all XCP-dedicated buffers of a slave to their initial assignments.

Any other values of `XCP_PACKET_TYPE` as explicitly mentioned here, are not allowed.

If the parameter is configurable, with `FLX_ASSIGN` the XCP on FlexRay master can change the initial value of `MAX_FLX_LEN_BUF_x` of a specific buffer.

The following relation has to be fulfilled:

$$\text{MAX\_FLX\_LEN\_BUF\_x} \leq \text{MAX\_FLX\_LEN\_BUF\_x\_init}$$

For easy configuration of the slave, the master with `FLX_ASSIGN` transfers the FlexRay `HEADER_CRC` which results from using `FLX_SLOT_ID` and `MAX_FLX_LEN_BUF_x`.

If the slave uses frame lengths `FLX_LEN_BUF_x ≤ MAX_FLX_LEN_BUF_x`, it has to calculate `HEADER_CRC` by itself.

By configuring `MAX_FLX_LEN_BUF_x` the XCP on FlexRay master can organize the XCP-dedicated buffers of the slave in such a way that the master’s communication requirements are optimally fulfilled.

## 5.2 ACTIVATE COMMUNICATION OF A FLEXRAY BUFFER

Category FlexRay only, optional

Mnemonic FLX\_ACTIVATE

Table 8 FLX\_ACTIVATE command structure

Position	Type	Description
0	BYTE	Command Code = TRANSPORT_LAYER_CMD = 0xF2
1	BYTE	Sub Command Code = FLX_ACTIVATE = 0xFE
2	BYTE	FLX_BUF

For EV and SERV this command activates the communication of a specific FLX\_BUF in a specific slave.

Before using this command, the XCP master shall have successfully assigned the FLX\_BUF to a FLX\_LPDU\_ID by the command FLX\_ASSIGN.

The XCP master is responsible to avoid multiple activated communication of different slaves using the same FlexRay cells.

### 5.3 DEACTIVATE COMMUNICATION OF A FLEXRAY BUFFER

Category FlexRay only, optional  
Mnemonic FLX\_DEACTIVATE

Table 9 FLX\_DEACTIVATE command structure

Position	Type	Description
0	BYTE	Command Code = TRANSPORT_LAYER_CMD = 0xF2
1	BYTE	Sub Command Code = FLX_DEACTIVATE = 0xFD
2	BYTE	FLX_BUF

For EV and SERV this command deactivates the communication of a specific FLX\_BUF in a specific slave.

Before using this command, the XCP master shall have properly assigned the FLX\_BUF to a FLX\_LPDU\_ID by the command FLX\_ASSIGN.



## 5.4 GET DAQ LIST FLEXRAY BUFFER(S)

Category FlexRay only, optional

Mnemonic GET\_DAQ\_FLX\_BUF

**Table 10 GET\_DAQ\_FLX\_BUF command structure**

Position	Type	Description
0	BYTE	Command Code = TRANSPORT_LAYER_CMD = 0xF2
1	BYTE	Sub Command Code = GET_DAQ_FLX_BUF = 0xFC
2	WORD	DAQ_LIST_NUMBER [0,1,...MAX_DAQ-1]

Positive Response:

**Table 11 GET\_SLAVE\_ID positive response structure**

Position	Type	Description
0	BYTE	Packet ID: 0xFF
1	BYTE	FLX_BUF_FIXED
		0 = FlexRay buffer can be configured
		1 = FlexRay buffer is fixed
2	BYTE	MAX_BUF
		maximum number of buffers = n
3	BYTE	FLX_BUF_1
		(First FlexRay buffer which is assigned to this DAQ List)
...		
n+2	BYTE	FLX_BUF_n
		(nth FlexRay buffer which is assigned to this DAQ List)

With GET\_DAQ\_FLX\_BUF, the master can detect whether a DAQ list uses one or more individual FlexRay buffer(s) and whether this FlexRay buffer(s) is (are) fixed or configurable.

If the FlexRay buffer is configurable, the master can configure the individual FlexRay buffer for this DAQ list with SET\_DAQ\_FLX\_BUF.

## 5.5 SET DAQ LIST FLEXRAY BUFFER(S)

Category FlexRay only, optional  
Mnemonic SET\_DAQ\_FLX\_BUF

**Table 12 SET\_DAQ\_FLX\_BUF command structure**

Position	Type	Description
0	BYTE	Command Code = TRANSPORT_LAYER_CMD = 0xF2
1	BYTE	Sub Command Code = SET_DAQ_FLX_BUF = 0xFB
2	WORD	DAQ_LIST_NUMBER [0,1,...MAX_DAQ-1]
4	BYTE	n = number of buffers to be assigned
5	BYTE	FLX_BUF_1
		(First FlexRay buffer which shall be assigned to this DAQ List)
...		
n+4	BYTE	FLX_BUF_n (nth FlexRay buffer which shall be assigned to this DAQ List)

The master can assign one or more individual FlexRay buffers to a DAQ list.  
If the given FlexRay buffer(s) is not (are not) successfully assigned, the slave returns an ERR\_OUT\_OF\_RANGE.

## 5.6 DAQ CLOCK MULTICAST

Category FlexRay only, optional  
Mnemonic GET\_DAQ\_CLOCK\_MULTICAST

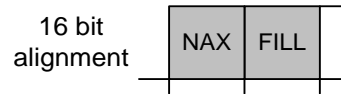
**Table 13 GET\_DAQ\_CLOCK\_MULTICAST command structure**

Position	Type	Description
0	BYTE	Command Code = TRANSPORT_LAYER_CMD = 0xF2
1	BYTE	Sub Command Code = 0xFA
2	WORD	Cluster Identifier (Intel byte order = little endian)
4	BYTE	Counter (allows for consistency checks at XCP master)

It is within the responsibility of the network designer to foresee LPDU\_IDs that are assigned to transport GET\_DAQ\_CLOCK\_MULTICAST commands. Thereby, each master might only use one LPDU\_ID out of the pool of LPDU\_IDs to establish a time synchronization domain. The availability of such a LPDU\_ID will be given in the A2L file, see the transport layer IF\_DATA for FlexRay, section XCP\_PACKET by using the keyword MULTICAST.

Different XCP header types are defined for transport layer FlexRay. In general – i.e. for point to point communication - there is the freedom that each slave uses its own header type out of the set of header types. However, to ensure that GET\_DAQ\_CLOCK\_MULTICAST messages are received and processed by all slaves, the

header type depicted in Figure 12 has to be used for sending `GET_DAQ_CLOCK_MULTICAST` commands.



**Figure 12 Header for `GET_DAQ_CLOCK_MULTICAST` command**

If an XCP slave shall support improved time synchronization capabilities it has to be ensured that `LPDU_ID` predefined by the network designer is processed by the XCP slave. This is indicated by the keyword `MULTICAST {FIXED|VARIABLE|VARIABLE_INITIALIZED}` as part of `XCP_PACKET` section of the `IF_DATA` description. In addition, the slave also must support the header type depicted in Figure 12 in order to receive `GET_DAQ_CLOCK_MULTICAST` commands.

When an XCP master makes use of `GET_DAQ_CLOCK_MULTICAST` command on transport layer FlexRay, a `GET_DAQ_CLOCK_MULTICAST` command must be sent every two seconds at the latest.

Upon reception of `GET_DAQ_CLOCK_MULTICAST` command, the XCP slave will respond an `EV_TIME_SYNC` event packet as defined in XCP Protocol Layer.

## 5.7 COMMUNICATION ERROR HANDLING

This chapter describes transport layer specific error code handling. It extends the error handling concepts specified in the chapter “Communication Error Handling” of the base standard [1]. Please refer to the base standard for obtaining fundamentals on XCP error handling.

### 5.7.1 ERROR CODE HANDLING

**Table 14 Transport Layer FlexRay sub commands error handling**

Command	Error	Pre-Action	Action
FLX_ASSIGN	ERR_CMD_SYNTAX	-	retry other syntax
	ERR_OUT_OF_RANGE	-	retry other parameter
	ERR_SUBCMD_UNKNOWN	-	display error or silently ignore error
FLX_ACTIVATE	ERR_OUT_OF_RANGE	-	retry other parameter
	ERR_SUBCMD_UNKNOWN	-	display error or silently ignore error
FLX_DEACTIVATE	ERR_OUT_OF_RANGE	-	retry other parameter
	ERR_SUBCMD_UNKNOWN	-	display error or silently ignore error
GET_DAQ_FLX_BUF	ERR_OUT_OF_RANGE	-	retry other parameter
	ERR_SUBCMD_UNKNOWN	-	display error or silently ignore error
SET_DAQ_FLX_BUF	ERR_CMD_SYNTAX	-	retry other syntax
	ERR_OUT_OF_RANGE	-	retry other parameter
	ERR_SUBCMD_UNKNOWN	-	display error or silently ignore error
GET_DAQ_CLOCK_MULTICAST	-	-	-

## **6 SPECIFIC EVENTS FOR XCP ON FLEXRAY**

There are no specific events for XCP on FlexRay at the moment.

## **7 INTERFACE TO ASAM MCD-2 MC DESCRIPTION FILE**

The following chapter describes the parameters that are specific for XCP on FlexRay.

### **7.1 ASAM MCD-2 MC AML FOR XCP ON FLEXRAY**

The AML for the XCP on FlexRay transport layer specific properties is defined in the file named XCP\_vX\_Y\_on\_FLX.aml where vX\_Y is the current transport layer version.

### **7.2 IF\_DATA EXAMPLE FOR XCP ON FLEXRAY**

The file XCP\_vX\_Y\_IF\_DATA\_example.a2l where vX\_Y is the current protocol layer version gives an IF\_DATA example for a XCP on FlexRay transport layer (see section beginning with `"/begin XCP_ON_FLX"`).

## 8 INTERFACE TO ASAM MCD-2 NET [FBX] DESCRIPTION FILE

If an XCP master sets up a connection to an XCP on FlexRay slave, it gets the information about the XCP-dedicated LPDU\_IDs of the network from the FIBEX description file.

The XCP master builds the set of valid XCP-dedicated LPDU\_IDs of the network by checking the frames for having a FRAME-TYPE "OTHER" and having a MANUFACTURER-EXTENSION called "FRAME-TYPE-EXTENSION" with the value "XCP".

```
<fx: FRAME ID="XCP_Frame">
  <ho:SHORT-NAME>XCP_Frame</ho:SHORT-NAME>
  ...
  <fx:FRAME-TYPE>OTHER</fx:FRAME-TYPE>
  ...
  <fx:MANUFACTURER-EXTENSION>
    <fx:FRAME-TYPE-EXTENSION>XCP</fx:FRAME-TYPE-EXTENSION>
  </fx:MANUFACTURER-EXTENSION>
  ...
</fx:FRAME>
```

As soon as the FIBEX description file format inherently supports the definition of XCP-dedicated LPDU\_IDs of the network, that definition will become recommended practice. However, the alternative with FRAME-TYPE "OTHER" and MANUFACTURER-EXTENSION "FRAME-TYPE-EXTENSION" = "XCP" stays valid.

## 9 SYMBOL AND ABBREVIATED TERMS

<i>A2L</i>	ASAM MCD-2 MC Language File
<i>CAN</i>	Controller Area Network
<i>CCP</i>	CAN Calibration Protocol
<i>CMD</i>	Command
<i>CTO</i>	Command Transfer Object
<i>CTR</i>	Counter
<i>DAQ</i>	Data Acquisition
<i>DTO</i>	Data Transfer Objects
<i>ECU</i>	Electronic Control Unit
<i>ERR</i>	Error
<i>EV</i>	Event
<i>FBX</i>	Field Bus Exchange Format
<i>FLX</i>	FlexRay
<i>IP</i>	Internet Protocol
<i>LEN</i>	Length
<i>LPDU</i>	Link Layer Protocol Data Unit
<i>NAX</i>	Node Address for XCP
<i>PID</i>	Packet Identifier
<i>RES</i>	Responses
<i>SCI</i>	Serial Communication Interface
<i>SERV</i>	Service
<i>SPI</i>	Serial Peripheral Interface
<i>STIM</i>	Synchronous Data Stimulation
<i>SxI</i>	Serial X Interface
<i>TCP</i>	Transfer Control Protocol
<i>UDP</i>	User Datagram Protocol
<i>USB</i>	Universal Serial Bus
<i>XCP</i>	Universal Measurement and Calibration Protocol



## 10 BIBLIOGRAPHY

- [1] ASAM AE MCD-1 XCP BS Protocol-Layer BS Version V1.3.0
- [2] ASAM AE MCD-2 NET Data Model for ECU Network Systems (FIBEX Field Bus Data Exchange Format) Version 3.1.1
- [3] FlexRay Communications System Specification Version 3.0.1
- [4] **ASAM MCD-2 MC** "Measurement and Calibration Data Specification", Version 1.7.x

---

**Figure Directory**

Figure 1	Data link layer protocol data unit identifier	8
Figure 2	Theoretical (top) vs. practical (bottom) max. length of payload segment	9
Figure 3	XCP dedicated LPDU_IDs on the network	10
Figure 4	Multiple XCP slaves using exclusive LPDU_IDs	13
Figure 5	Header and tail for XCP on FlexRay	15
Figure 6	Header for subsequent concatenated messages	16
Figure 7	Tail for rounding up to even FlexRay length	18
Figure 8	Tail for filling up to MAX_FLX_LEN_BUF_x	18
Figure 9	Tail for aligning concatenated XCP messages and rounding up to even FlexRay length	19
Figure 10	LEN = 0 for detecting the end of the used payload segment	19
Figure 11	Relation between MAX_CTO, MAX.DTO and MAX_FLX_LEN_BUF_x	19
Figure 12	Header for GET_DAQ_CLOCK_MULTICAST command	27

---

## **Table Directory**

Table 1	Example of the assignment of LPDU_IDs to XCP-dedicated buffers	11
Table 2	Example of the assignment of XCP packets to buffers	12
Table 3	Header types for first XCP message of a FlexRay frame	16
Table 4	CTO and DTO range	20
Table 5	Command codes overview	21
Table 6	FLX_ASSIGN command structure	21
Table 7	XCP_PACKET_TYPE parameter bit mask structure	22
Table 8	FLX_ACTIVATE command structure	23
Table 9	FLX_DEACTIVATE command structure	24
Table 10	GET_DAQ_FLX_BUF command structure	25
Table 11	GET_SLAVE_ID positive response structure	25
Table 12	SET_DAQ_FLX_BUF command structure	26
Table 13	GET_DAQ_CLOCK_MULTICAST command structure	26
Table 14	Transport Layer FlexRay sub commands error handling	28



Association for Standardisation of  
Automation and Measuring Systems

E-mail: [support@asam.net](mailto:support@asam.net)

Web: [www.asam.net](http://www.asam.net)

© by ASAM e.V., 2015