

XCP

Version 1.1

Part 3 - Transport Layer Specification

XCP on FlexRay

Part 3 – XCP on FlexRay – Transport Layer Specification



**Association for Standardisation of
Automation and Measuring Systems**

**Dated: 31-03-2008
© ASAM e. V.**

Status of Document

Date:	31-03-2008
Authors:	Roel Schuermans, Vector Informatik GmbH Andreas Zeiser, Vector Informatik GmbH Oliver Kitt, Vector Informatik GmbH Hans-Georg Kunz, VDO Automotive AG Hendirk Amsbeck, dSPACE GmbH Bastian Kellers, dSPACE GmbH Boris Ruoff, ETAS GmbH Reiner Motz, Robert Bosch GmbH Dirk Forwick, Robert Bosch GmbH
Version:	Version 1.1
Doc-ID:	
Status:	Release
Type	

Disclaimer of Warranty

Although this document was created with the utmost care it cannot be guaranteed that it is completely free of errors or inconsistencies.

ASAM e.V. makes no representations or warranties with respect to the contents or use of this documentation, and specifically disclaims any expressed or implied warranties of merchantability or fitness for any particular purpose. Neither ASAM nor the author(s) therefore accept any liability for damages or other consequences that arise from the use of this document.

ASAM e.V. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Revision History

This revision history shows only major modifications between release versions.

Date	Author	Filename	Comments
2008-03-31	R.Schuermans		Released document

Table of contents

<u>0</u>	<u>Introduction</u>	<u>9</u>
0.1	The XCP Protocol Family	9
0.2	Documentation Overview	10
0.3	Definitions and Abbreviations	11
0.4	Mapping between XCP Data Types and ASAM Data Types	12
0.5	Normative Reference	13
<u>1</u>	<u>The XCP Transport Layer for FlexRay</u>	<u>14</u>
1.1	FlexRay frames transporting XCP	14
1.1.1	LPDU_ID	14
1.1.2	FlexRay frame type	15
1.1.3	Maximum FlexRay length	15
1.2	Addressing	16
1.2.1	XCP-dedicated LPDU_IDs on the network	16
1.2.2	XCP-dedicated buffers in the slave	16
1.2.2.1	Assignment of LPDU_IDs to buffers	17
1.2.2.2	Assignment of XCP-packets to buffers	18
1.2.3	Multiple XCP-slaves using exclusive LPDU_IDs	19
1.2.4	Multiple XCP-slaves competing for LPDU_IDs	20
1.2.4.1	Node address for XCP	20
1.2.4.2	Mutually exclusive activation of XCP-dedicated buffers	20
1.2.4.3	Reconfiguration of XCP-dedicated buffers	20
1.2.5	Use of XCP-dedicated buffers within the slave	21
1.3	Communication Model	22
1.4	Header and Tail	23
1.4.1	Header	23
1.4.1.1	Node Address for XCP (NAX)	24
1.4.1.2	Counter (CTR)	24
1.4.1.3	Fill	24
1.4.1.4	Length (LEN)	24
1.4.2	Tail	25
1.4.2.1	Single XCP message in one FlexRay frame	25
1.4.2.2	Multiple XCP messages in one FlexRay frame	26
1.5	The Limits of performance	27
<u>2</u>	<u>Specific commands for XCP on FlexRay</u>	<u>28</u>
2.1	Assign/Deassign FlexRay LPDU-Identifiers to buffers	29
2.2	Activate Communication of a FlexRay buffer	31

2.3	Deactivate Communication of a FlexRay buffer	32
2.4	Get DAQ List FlexRay Buffer(s)	33
2.5	Set DAQ List FlexRay Buffer(s)	34
<u>3</u>	<u>Specific events for XCP on FlexRay</u>	<u>35</u>
<u>4</u>	<u>Interface to ASAM MCD 2MC description file</u>	<u>36</u>
4.1	ASAM MCD 2MC AML for XCP on FlexRay	36
4.2	IF_DATA example for XCP on FlexRay	39
<u>5</u>	<u>Interface to ASAM MCD-2 [FBX] description file</u>	<u>42</u>

Table of diagrams:

Diagram 1 : Data Link Layer Protocol Data Unit Identifier	14
Diagram 2 : Theoretical (top) vs. practical (bottom) max. length of payload segment	15
Diagram 3 : XCP dedicated LPDU_IDs on the network	16
Diagram 4 : multiple XCP slaves using exclusive LPDU_IDs	19
Diagram 5 : Header and Tail for XCP on FlexRay	23
Diagram 6 : Header types for first XCP message of a FlexRay frame	23
Diagram 7 : Header for subsequent concatenated messages	23
Diagram 8 : Tail for rounding up to even FlexRay length	25
Diagram 9 : Tail for filling up to MAX_FLX_LEN_BUF_x	25
Diagram 10 : Tail for aligning concatenated XCP messages and rounding up to even FlexRay length	26
Diagram 11 : LEN = 0 for detecting the end of the used payload segment	26
Diagram 12 : Relation between MAX_CTO, MAX_DTO and MAX_FLX_LEN_BUF_x	27

0 INTRODUCTION

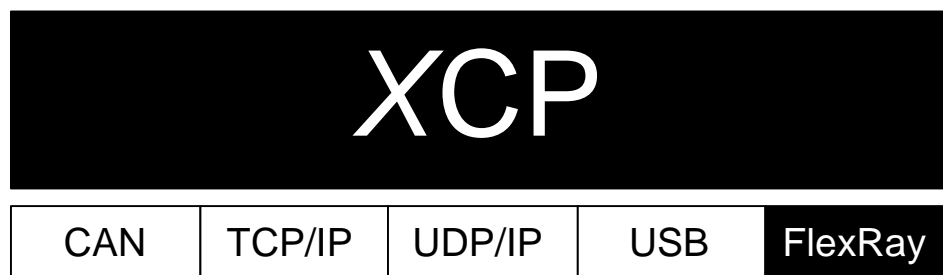
0.1 THE XCP PROTOCOL FAMILY

This document is based on experiences with the **CAN Calibration Protocol (CCP)** version 2.1 as described in feedback from the companies Accurate Technologies Inc., Compact Dynamics GmbH, DaimlerChrysler AG, dSPACE GmbH, ETAS GmbH, Kleinknecht Automotive GmbH, Robert Bosch GmbH, Siemens VDO Automotive AG and Vector Informatik GmbH.

The XCP Specification documents describe an improved and generalized version of CCP.

The generalized protocol definition serves as standard for a protocol family and is called “XCP” (Universal Measurement and **C**alibration **P**rotocol).

The “**X**” generalizes the “various” transportation layers that are used by the members of the protocol family e.g “XCP on CAN”, “XCP on TCP/IP”, “XCP on UDP/IP”, “XCP on USB” , “XCP on FlexRay” and so on.



XCP is not backwards compatible to an existing CCP implementation.

0.2 DOCUMENTATION OVERVIEW

The XCP specification consists of 5 parts. Each part is a separate document and has the following contents:

Part 1 “Overview” gives an overview over the XCP protocol family, the XCP features and the fundamental protocol definitions.

Part 2 “Protocol Layer Specification” defines the generic protocol, which is independent from the transportation layer used.

Part 3 “Transport Layer Specification” defines the way how the XCP protocol is transported by a particular transportation layer like CAN, TCP/IP, UDP/IP and FlexRay.

This document describes the way how the XCP protocol is transported on FlexRay.

Part 4 “Interface Specification” defines the interfaces from an XCP master to an ASAM MCD 2MC description file and for calculating Seed & Key algorithms and checksums.

Part 5 “Example Communication Sequences” gives example sequences for typical actions performed with XCP.

Everything not explicitly mentioned in this document, should be considered as implementation specific.

0.3 DEFINITIONS AND ABBREVIATIONS

The following table gives an overview about the most commonly used definitions and abbreviations throughout this document.

Abbreviation	Description
A2L	File Extension for an ASAM 2MC Language File
AML	ASAM 2 Meta Language
ASAM	A ssociation for S tandardization of A utomation and M easuring Systems
BYP	BYP assing
CAL	CAL ibration
CAN	C ontroller A rea N etwork
CCP	C an C alibration P rotocol
CHI	C ontroller H ost I nterface
CMD	CoMmanD
CS	C heck S um
CTO	C ommand T ransfer O bject
CTR	CounTeR
DAQ	D ata AcQ uisition, D ata AcQ uisition Packet
DTO	D ata T ransfer O bject
ECU	E lectronic C ontrol U nit
ERR	ERR or Packet
EV	E vent Packet
FIBEX	F ield B us EX change Format
FLX	FLeX Ray
LEN	LEN gth
LPDU_ID	Data L ink Layer P rotocol D ata U nit I Dentifier
MCD	M easurement C alibration and D iagnostics
MTA	M emory T ransfer A ddress
NAX	N ode A ddress for XCP
ODT	O bject D escriptor T able
PAG	PAG ing
PGM	ProGraM ming
PID	P acket I Dentifier
RES	command RES ponse packet
SERV	SERV ice request packet
STD	STanD ard
STIM	Data STIM ulation packet
TCP/IP	T ransfer C ontrol P rotocol / I nternet P rotocol
TS	T ime S tamp
UDP/IP	U nified D ata P rotocol / I nternet P rotocol
USB	U niversal S erial B us
XCP	Universal C alibration P rotocol

Table 1: Definitions and Abbreviations

0.4 MAPPING BETWEEN XCP DATA TYPES AND ASAM DATA TYPES

The following table defines the mapping between data types used in this specification and ASAM data types defined by the Project Data Harmonization Version 2.0 (ref. www.asam.net).

XCP Data Type	ASAM Data Type
BYTE	A_UINT8
WORD	A_UINT16
DWORD	A_UINT32
DLONG	A_UINT64

0.5 NORMATIVE REFERENCE

The following standards contain provisions which, through reference in this text, constitute provisions of this document.

FIBEX	FIBEX Field Bus Exchange Format	Version 1.1.5
FlexRay	FlexRay Communications System Specification	Version 2.1

1 THE XCP TRANSPORT LAYER FOR FLEXRAY

1.1 FLEXRAY FRAMES TRANSPORTING XCP

1.1.1 LPDU_ID

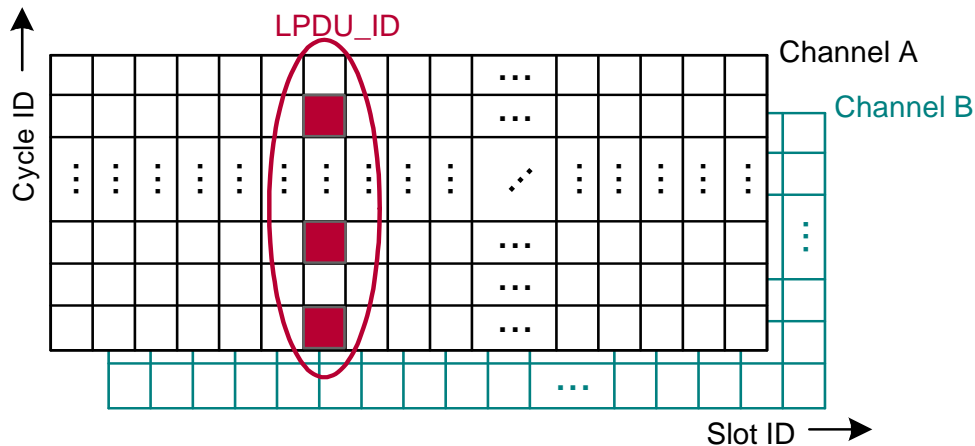


Diagram 1 : Data Link Layer Protocol Data Unit Identifier

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: $b < 2^r$

\swarrow cycle repetition

\swarrow generator

\swarrow 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 (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 (FLX_SLOT_ID)
- cycle counter offset (OFFSET)
- cycle counter repetition (CYCLE_REPETITION)
- FlexRay channel (FLX_CHANNEL).

XCP on FlexRay messages are transported inside FlexRay LPDUs.

1.1.2 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.

1.1.3 MAXIMUM FLEXRAY LENGTH

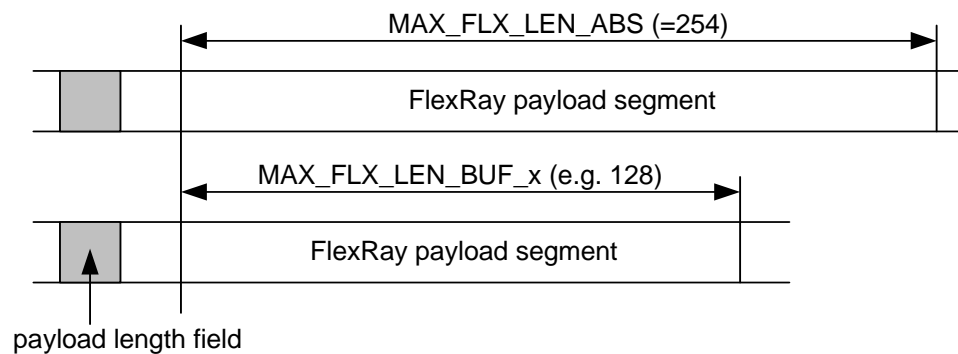


Diagram 2 : Theoretical (top) vs. practical (bottom) max. length of payload segment

The maximum theoretical data length of a FlexRay frame and therefore the maximum length of an XCP on FlexRay message is 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 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 MAX_FLX_LEN_BUF_x_init.

The actual length of the FlexRay payload segment of a specific FlexRay frame, expressed in bytes, is called 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:

$$\text{payload length field} = \text{FLX_LEN_BUF_x} / 2$$

1.2 ADDRESSING

1.2.1 XCP-DEDICATED LPDU_IDS ON THE NETWORK

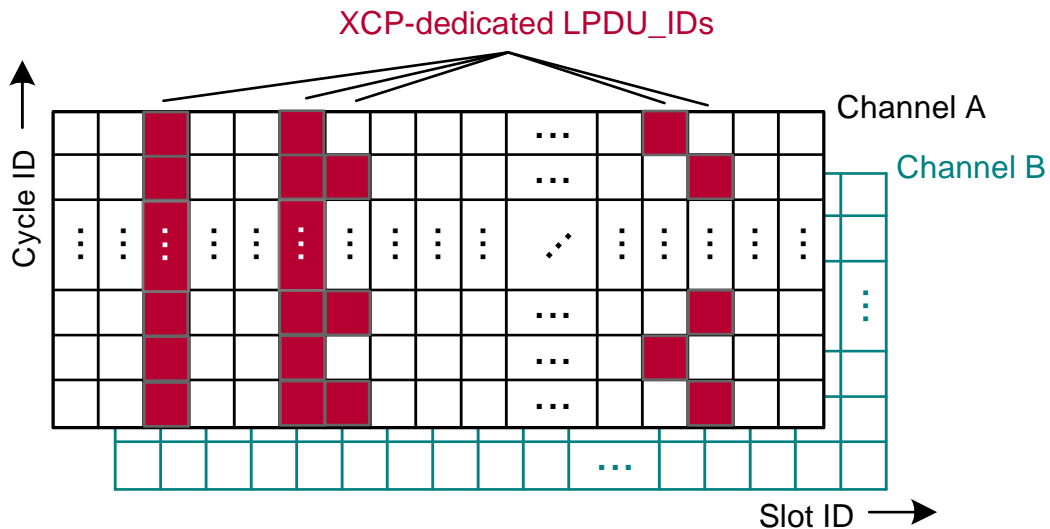


Diagram 3 : XCP dedicated LPDU_IDS on the network

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.

1.2.2 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 2MC description file describes to which extent the XCP-dedicated buffers of a specific slave can be configured for XCP communication.

1.2.2.1 ASSIGNMENT OF LPDU_IDS TO 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)

123, A initial value () configurable
 * uninitialised parameter

Table 2: Example of the assignment of LPDU_IDS to XCP-dedicated buffers

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 uninitialised 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 initialised but it could be changed by the XCP on FlexRay master.

For Buffer_4 the parameters for the set of cycle counters are uninitialised and have to be configured by the XCP on FlexRay master.

Buffer_5 is completely uninitialised 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 initialised but it could be changed by the XCP on FlexRay master.

1.2.2.2 ASSIGNMENT OF XCP-PACKETS TO BUFFERS

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

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

Table 3: Example of the assignment of XCP packets to buffers

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 can not 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.

1.2.3 MULTIPLE XCP-SLAVES USING EXCLUSIVE LPDU_IDS

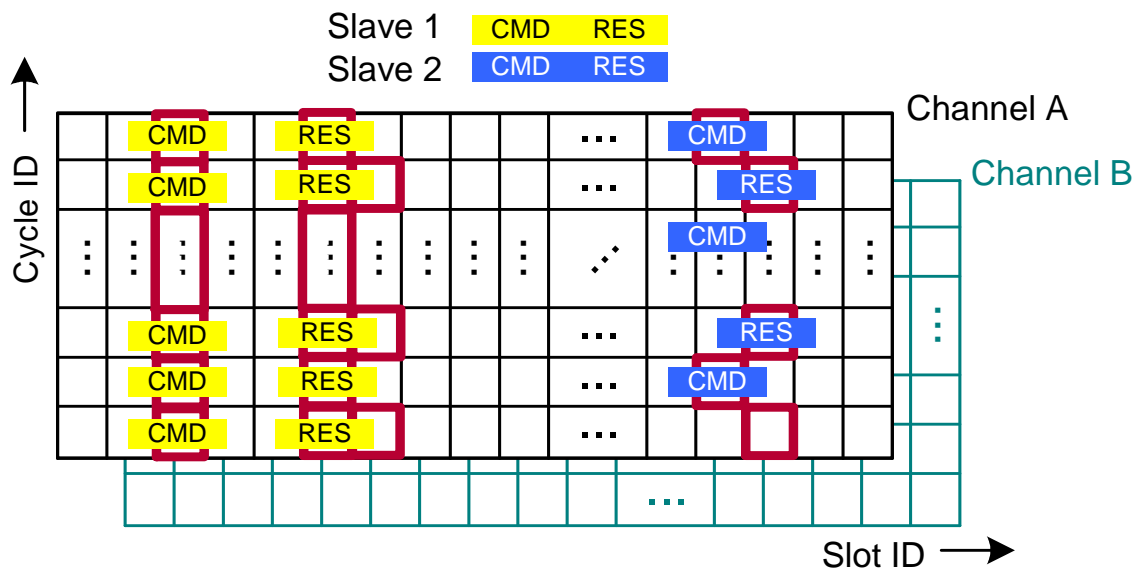


Diagram 4 : multiple XCP slaves using exclusive LPDU_IDS

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. The XCP master gets the information about the slave's XCP-dedicated buffers from the ASAM MCD 2MC description file 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.

1.2.4 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.

1.2.4.1 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. 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.

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.

1.2.4.2 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.

1.2.4.3 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.

1.2.5 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.

1.3 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.

1.4 HEADER AND TAIL

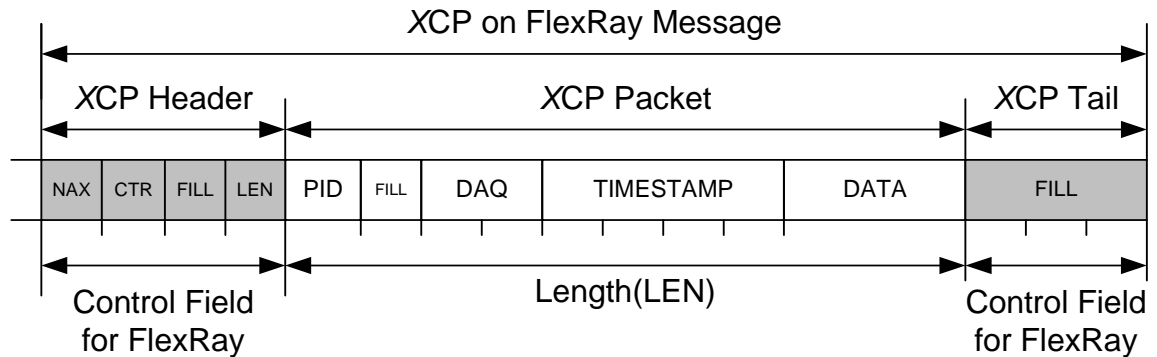


Diagram 5 : Header and Tail for XCP on FlexRay

1.4.1 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.

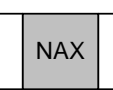
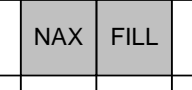
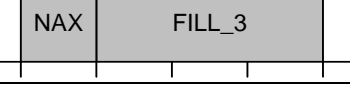
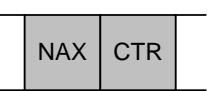
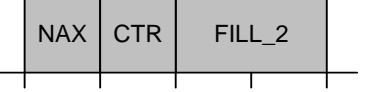
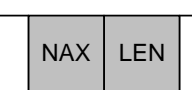
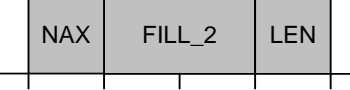
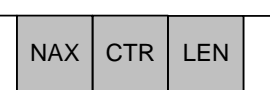
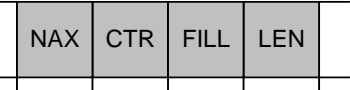
	sequence correctness not guaranteed	sequence correcting possible
no concatenation allowed	<p>8 bit alignment</p>  <p>16 bit alignment</p>  <p>32 bit alignment</p> 	<p>8 / 16 bit alignment</p>  <p>32 bit alignment</p> 
concatenation allowed	<p>8 / 16 bit alignment</p>  <p>32 bit alignment</p> 	<p>8 bit alignment</p>  <p>16 / 32 bit alignment</p> 

Diagram 6 : Header types for first XCP message of a FlexRay frame

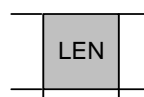


Diagram 7 : Header for subsequent concatenated messages

1.4.1.1 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.

1.4.1.2 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.

1.4.1.3 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 "don't 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.

1.4.1.4 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.

1.4.2 TAIL

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

1.4.2.1 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 “don’t 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.

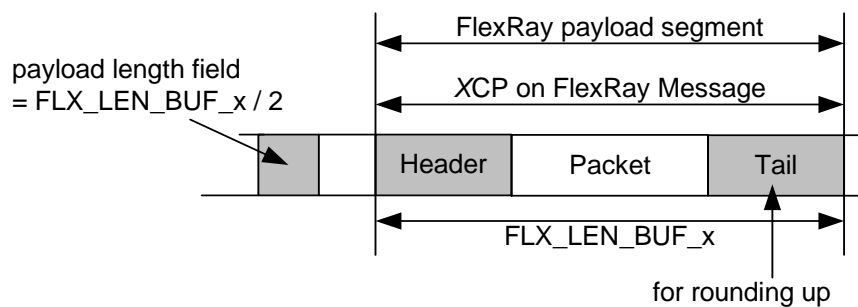


Diagram 8 : Tail for rounding up to even FlexRay length

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 $MAX_FLX_LEN_BUF_x$.

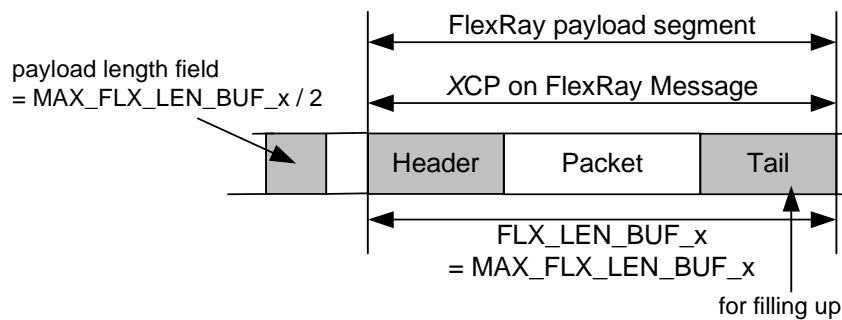


Diagram 9 : Tail for filling up to $MAX_FLX_LEN_BUF_x$

1.4.2.2 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 build 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 build in such a way that the overall length `FLX_LEN_BUF_x` becomes an even number.

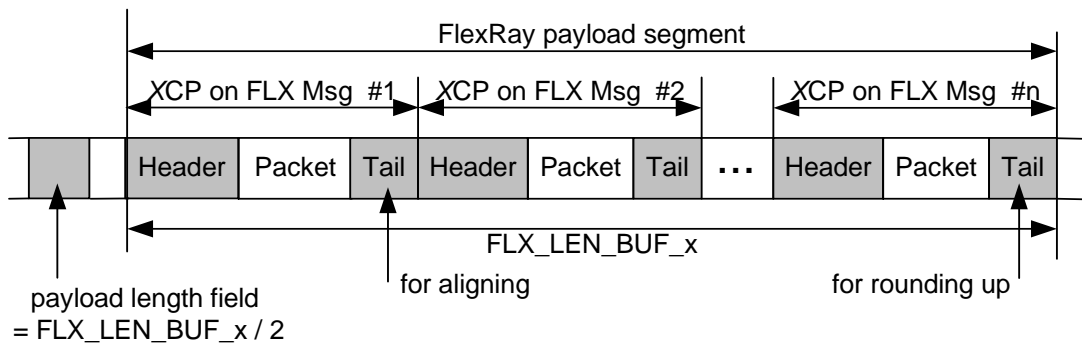


Diagram 10 : Tail for aligning concatenated XCP messages and rounding up to even FlexRay length

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.

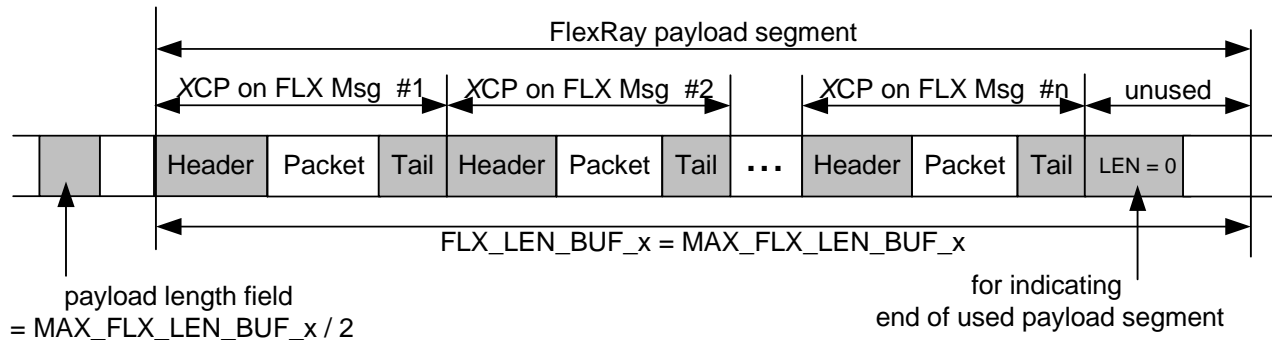


Diagram 11 : LEN = 0 for detecting the end of the used payload segment

1.5 THE LIMITS OF PERFORMANCE

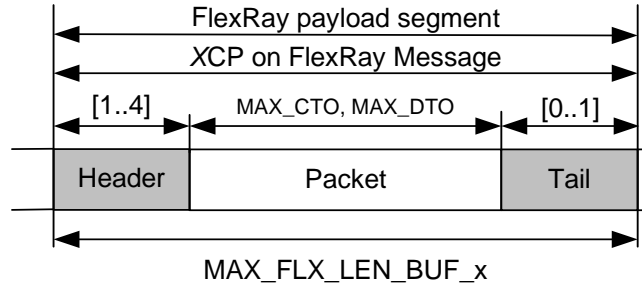


Diagram 12 : Relation between MAX_CTO, MAX_DTO and MAX_FLX_LEN_BUF_x

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.

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.

2 SPECIFIC COMMANDS FOR XCP ON FLEXRAY

Table of Command Codes

Command	Code	Timeout	Remark
FLX_ASSIGN	0xFF	t1_FLX	optional
FLX_ACTIVATE	0xFE	t1	optional
FLX_DEACTIVATE	0xFD	t1	optional
GET_DAQ_FLX_BUF	0xFC	t1	optional
SET_DAQ_FLX_BUF	0xFB	t1	optional

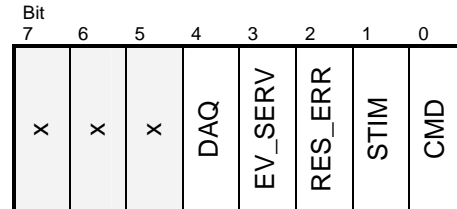
2.1 ASSIGN/DEASSIGN FLEXRAY LPDU-IDENTIFIERS TO BUFFERS

Category FlexRay only, optional
Mnemonic FLX_ASSIGN

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.



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.

buffer type	allowed values for XCP_PACKET_TYPE
receive	0x01, 0x02, 0x03
transmit	0x04, 0x08, 0x0C, 0x10 0x14, 0x18, 0x1C

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 “uninitialised” 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 \leq 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.

2.2 ACTIVATE COMMUNICATION OF A FLEXRAY BUFFER

Category FlexRay only, optional
Mnemonic FLX_ACTIVATE

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.

2.3 DEACTIVATE COMMUNICATION OF A FLEXRAY BUFFER

Category FlexRay only, optional
Mnemonic FLX_DEACTIVATE

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.

2.4 GET DAQ LIST FLEXRAY BUFFER(S)

Category FlexRay only, optional
Mnemonic GET_DAQ_FLX_BUF

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:

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.

2.5 SET DAQ LIST FLEXRAY BUFFER(S)

Category FlexRay only, optional
Mnemonic SET_DAQ_FLX_BUF

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.

3 SPECIFIC EVENTS FOR *XCP* ON FLEXRAY

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

4 INTERFACE TO ASAM MCD 2MC DESCRIPTION FILE

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

4.1 ASAM MCD 2MC AML FOR XCP ON FLEXRAY

```

/*****
/*
/*  ASAP2 meta language for XCP on FlexRay V1.0
/*
/*  2005-10-13
/*
/*
/*  Datatypes:
/*
/*  A2ML      ASAP2      Windows    description
/*  -----
/*  uchar     UBYTE      BYTE      unsigned 8 Bit
/*  char      SBYTE      char      signed 8 Bit
/*  uint      UWORD      WORD      unsigned integer 16 Bit
/*  int       SWORD      int       signed integer 16 Bit
/*  ulong     ULONG      DWORD     unsigned integer 32 Bit
/*  long      SLONG      LONG      signed integer 32 Bit
/*  float     FLOAT32_IEEE      float 32 Bit
/*
*****/
/***** start of FLX *****/
enum packet_assignment_type {
    "NOT_ALLOWED",
    "FIXED",
    "VARIABLE_INITIALISED",
    "VARIABLE"
}; /* end of packet_assignment_type */

struct buffer {

    uchar; /* FLX_BUF */

    taggedstruct {

        "MAX_FLX_LEN_BUF" taggedunion {
            "FIXED" uchar; /* constant value */
            "VARIABLE" uchar; /* initial value */
        }; /* end of MAX_FLX_LEN_BUF */

        block "LPDU_ID" taggedstruct {

            "FLX_SLOT_ID" taggedunion {
                "FIXED" uint;
                "VARIABLE" taggedstruct{
                    "INITIAL_VALUE" uint;
                };
            }; /* end of FLX_SLOT_ID */

            "OFFSET" taggedunion {
                "FIXED" uchar;
                "VARIABLE" taggedstruct{
                    "INITIAL_VALUE" uchar;
                };
            }; /* end of OFFSET */
        };
    };
};

```

```

"Cycle_Repetition" taggedunion {
    "FIXED" uchar;
    "VARIABLE" taggedstruct{
        "INITIAL_VALUE" uchar;
    };
}; /* end of CYCLE_REPETITION */

"Channel" taggedunion {
    "FIXED" enum {
        "A" = 0,
        "B" = 1
    };
    "VARIABLE" taggedstruct{
        "INITIAL_VALUE" enum {
            "A" = 0,
            "B" = 1
        };
    };
}; /* end of CHANNEL */

}; /* end of LPDU_ID */

block "XCP_PACKET" taggedstruct {

    "CMD" enum packet_assignment_type; /* end of CMD */
    "RES_ERR" enum packet_assignment_type; /* end of RES_ERR */
    "EV_SERV" enum packet_assignment_type; /* end of EV_SERV */
    "DAQ" enum packet_assignment_type; /* end of DAQ */
    "STIM" enum packet_assignment_type; /* end of STIM */

}; /* end of XCP_PACKET */

};

}; /* end of buffer */

struct FLX_Parameters {

    uint; /* XCP on FlexRay version */
           /* e.g. "1.0" = 0x0100 */

    uint; /* T1_FLX [ms] */

    char[256]; /* FIBEX-file including CHI information */
              /* including extension */
              /* without path */

    char[256]; /* Cluster-ID */

    uchar; /* NAX */

    enum {
        "HEADER_NAX" = 0,
        "HEADER_NAX_FILL" = 1,
        "HEADER_NAX_CTR" = 2,
        "HEADER_NAX_FILL3" = 3,
        "HEADER_NAX_CTR_FILL2" = 4,
        "HEADER_NAX_LEN" = 5,
        "HEADER_NAX_CTR_LEN" = 6,
        "HEADER_NAX_FILL2_LEN" = 7,
        "HEADER_NAX_CTR_FILL_LEN" = 8
    };
};

```

```
enum {
    "PACKET_ALIGNMENT_8" = 0,
    "PACKET_ALIGNMENT_16" = 1,
    "PACKET_ALIGNMENT_32" = 2
};

taggedunion {
    block "INITIAL_CMD_BUFFER" struct buffer;
};

taggedunion {
    block "INITIAL_RES_ERR_BUFFER" struct buffer;
};

taggedstruct {

    (block "POOL_BUFFER" struct buffer)*;

};

};/***** end of FLX *****/
```

4.2 IF_DATA EXAMPLE FOR XCP ON FLEXRAY

```
/begin XCP_ON_FLX

0x0100          /* XCP on FlexRay 1.0 */

0x0019          /* T1_FLX [ms] */
"MyFlexRayNetwork.xml" /* FIBEX-file including CHI information */
"MyCluster"      /* Cluster-ID */
0x02            /* NAX */
HEADER_NAX_CTR_LEN
PACKET_ALIGNMENT_8

/begin INITIAL_CMD_BUFFER

0x01 /* FLX_BUF */
MAX_FLX_LEN_BUF FIXED 32

/begin LPDU_ID

FLX_SLOT_ID FIXED 0x07B
OFFSET FIXED 0
CYCLE_REPETITION FIXED 1
CHANNEL FIXED A

/end LPDU_ID

/begin XCP_PACKET

CMD    FIXED
RES_ERR NOT_ALLOWED
EV_SERV NOT_ALLOWED
DAQ    NOT_ALLOWED
STIM   FIXED

/end XCP_PACKET

/end INITIAL_CMD_BUFFER

/begin INITIAL_RES_ERR_BUFFER

0x02 /* FLX_BUF */
MAX_FLX_LEN_BUF FIXED 32

/begin LPDU_ID

FLX_SLOT_ID FIXED 0x07C
OFFSET FIXED 1
CYCLE_REPETITION FIXED 1
CHANNEL FIXED A

/end LPDU_ID

/begin XCP_PACKET

CMD    NOT_ALLOWED
RES_ERR FIXED
EV_SERV FIXED
DAQ    FIXED
STIM   NOT_ALLOWED

/end XCP_PACKET

/end INITIAL_RES_ERR_BUFFER
```

```
/begin POOL_BUFFER

0x03 /* FLX_BUF */
MAX_FLX_LEN_BUF FIXED 32

/begin LPDU_ID

FLX_SLOT_ID VARIABLE INITIAL_VALUE 0x07D
OFFSET FIXED 0
CYCLE_REPETITION FIXED 1
CHANNEL FIXED A

/end LPDU_ID

/begin XCP_PACKET

CMD NOT_ALLOWED
RES_ERR VARIABLE
EV_SERV VARIABLE
DAQ VARIABLE_INITIALISED
STIM NOT_ALLOWED

/end XCP_PACKET

/end POOL_BUFFER

/begin POOL_BUFFER

0x04 /* FLX_BUF */
MAX_FLX_LEN_BUF VARIABLE 64

/begin LPDU_ID

FLX_SLOT_ID FIXED 0x07E
OFFSET VARIABLE
CYCLE_REPETITION VARIABLE
CHANNEL FIXED A

/end LPDU_ID

/begin XCP_PACKET

CMD VARIABLE
RES_ERR VARIABLE
EV_SERV VARIABLE
DAQ VARIABLE
STIM VARIABLE

/end XCP_PACKET

/end POOL_BUFFER

/begin POOL_BUFFER

0x05 /* FLX_BUF */
MAX_FLX_LEN_BUF VARIABLE 64

/begin LPDU_ID

FLX_SLOT_ID VARIABLE
OFFSET VARIABLE
CYCLE_REPETITION VARIABLE
CHANNEL VARIABLE

/end LPDU_ID
```



```
/begin XCP_PACKET  
    CMD    VARIABLE  
    RES_ERR VARIABLE  
    EV_SERV VARIABLE  
    DAQ    VARIABLE  
    STIM    VARIABLE  
  
/end XCP_PACKET  
  
/end POOL_BUFFER  
  
/end XCP_ON_FLX
```

5 INTERFACE TO ASAM MCD-2 [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 FRAMETYPE "OTHER" and MANUFACTURER-EXTENSION "FRAME-TYPE-EXTENSION" = "XCP" stays valid.

ASAM e.V.
Arnikastraße 2
D-85635 Höhenkirchen
Germany

Tel.: (+49) 08102 / 8953 17
Fax.: (+49) 08102 / 8953 10
E-mail: info@asam.net
Internet: www.asam.net