XCP
Version 1.1

Part 3- Transport Layer Specification
XCP on SxI

Part 3 – XCP on SxI – Transport Layer Specification

Association for **S**tandardisation of
Automation and **M**easuring Systems

Dated:31-03-2008

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	

Revision History

This revision history shows only major modifications between release versions.

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

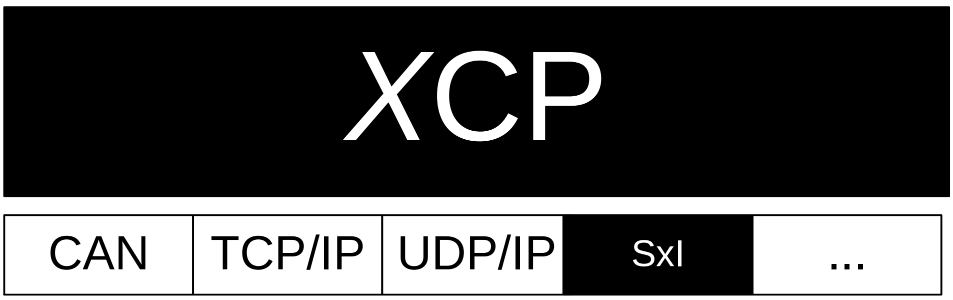
Table of contents

Table of diagrams:

0 INTRODUCTION

0.1 THE *X*CP PROTOCOL FAMILY

This document is based on experiences with the **C**AN **C**alibration **P**rotocol (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 *X*CP Specification documents describe an improved and generalized version of CCP.
The generalized protocol definition serves as standard for a protocol family and is called “*X*CP” (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 “*X*CP on CAN”, “*X*CP on TCP/IP”, “*X*CP on UDP/IP”, “*X*CP on USB” and so on.



0.2 DOCUMENTATION OVERVIEW

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

Part 1 “Overview” gives an overview over the *X*CP protocol family, the *X*CP 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 *X*CP protocol is transported by a particular transportation layer like CAN, TCP/IP and UDP/IP.

This document describes the way how the *X*CP protocol is transported on SxI interfaces.

Part 4 “Interface Specification” defines the interfaces from an *X*CP 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 *X*CP.

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 **A**SAM **2**MC **L**anguage File
AML	**A**SAM 2 **M**eta **L**anguage
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
CMD	**C**o**M**man**D**
CS	**C**heck**S**um
CTO	**C**ommand **T**ransfer **O**bject
CTR	**C**oun**T**e**R**
DAQ	**D**ata **A**c**Q**uisition, **D**ata **A**c**Q**uisition Packet
DTO	**D**ata **T**ransfer **O**bject
ECU	**E**lectronic **C**ontrol **U**nit
ERR	**ERR**or Packet
EV	**EV**ent Packet
LEN	**LEN**gth
MCD	**M**easurement **C**alibration and **D**iagnostics
MTA	**M**emory **T**ransfer **A**ddress
ODT	**O**bject **D**escriptor **T**able
PAG	**PAG**ing
PGM	**P**ro**G**ra**M**ming
PID	**P**acket **ID**entifier
RES	command **RES**ponse packet
SERV	**SERV**ice request packet
SPI	**S**erial **P**eripheral **I**nterface
STD	**ST**an**D**ard
STIM	Data **STIM**ulation packet
TCP/IP	**T**ransfer **C**ontrol **P**rotocol **/** **I**nternet **P**rotocol
TS	**T**ime Stamp
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

1 THE XCP TRANSPORT LAYER FOR SXI (SPI AND SCI)

1.1 ADDRESSING

In general SPI and SCI (SxI) are no bus interfaces, they are used as a point to point connection. Therefore an addressing feature is not part of the transport layer.

1.2 COMMUNICATION MODEL

*X*CP on SxI makes use of the standard communication model.

The block transfer communication is optional.

The interleaved communication model is optional.

1.3 HEADER AND TAIL

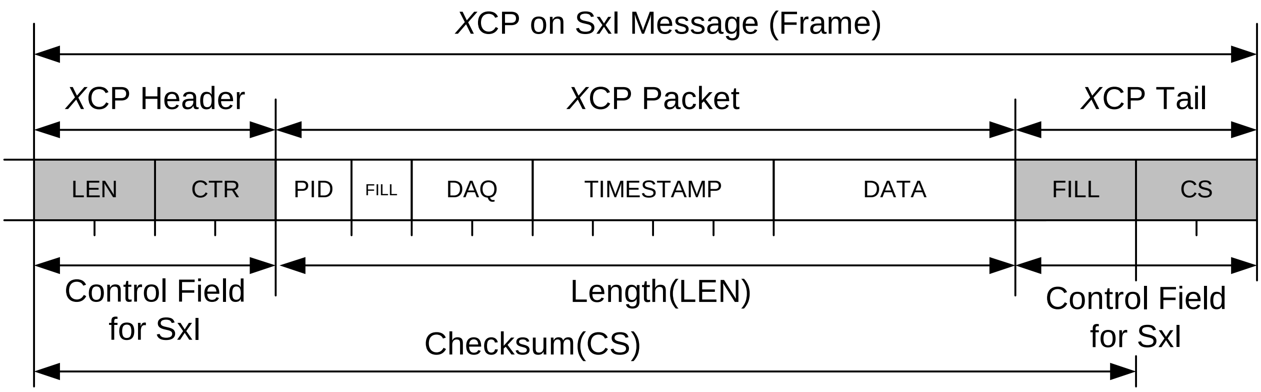


Diagram 1 : Header and Tail for *X*CP on SxI

1.3.1 HEADER

The *X*CP packet header for SxI consists of a Control Field containing a **LEN**gth (LEN) and an optional **C**oun**T**e**R** (CTR).

1.3.1.1 LENGTH

LEN is the number of bytes in the original *X*CP Packet. LEN can be BYTE or WORD (Intel format).

1.3.1.2 COUNTER

The CTR value in the *X*CP Header allows to detect missing Packets.

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, SRM or DAQ messages. The CTR value must be incremented for each new packet sent from slave to the master.

If available, CTR always has the same size as LEN.

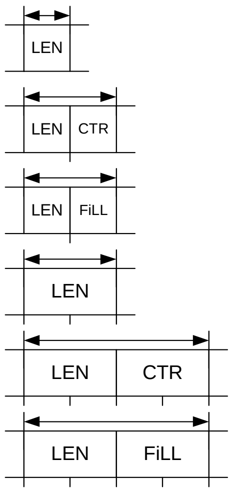


Diagram 2 : Header Types for *X*CP on SxI

1.3.2 TAIL

1.3.2.1 FILL BYTES

Depending on the alignment (when using the SPI in WORD or DWORD mode) and the minimum packet size (when Master/Slave SPI mode is used), LEN_FILL (= MAX_CTO(DTO)–LEN) optional fill bytes can be added at the end of the *X*CP Message.

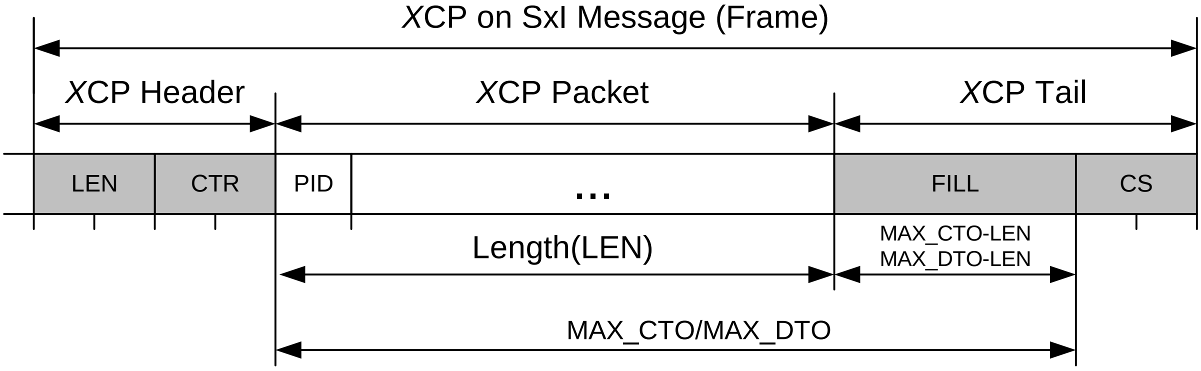


Diagram 3 : Fill bytes in Tail for *X*CP on SxI

1.3.2.2 CHECKSUM

The *X*CP Tail may contain an optional BYTE or WORD size checksum.

For a BYTE checksum the calculation must be done byte-wise, for a WORD checksum the calculation must be done word-wise. The checksum is calculated by adding the bytes of the *X*CP Header, the bytes of the *X*CP Packet and the Fill bytes of the *X*CP Tail into a BYTE or WORD checksum, ignoring overflows.

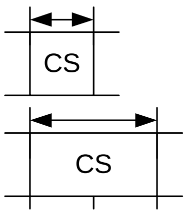


Diagram 4 : Checksum in Tail for *X*CP on SxI

1.4 THE LIMITS OF PERFORMANCE

There are no additional restrictions of MAX_CTO and MAX_DTO for *X*CP on SxI.

Name	Type	Representation	Range of value
MAX_CTO	Parameter	BYTE	0x08 – 0xFF
MAX_DTO	Parameter	WORD	0x0008 – 0xFFFF

1.5 COMMUNICATION MODES

1.5.1 ASYNCHRONOUS COMMUNICATION MODE (SCI)

In asynchronous (SCI) full duplex mode each direction is fully independent of the other and there are no restrictions regarding the protocol.

1.5.1.1 ASYNCHRONOUS COMMUNICATION MODE WITH FRAMING

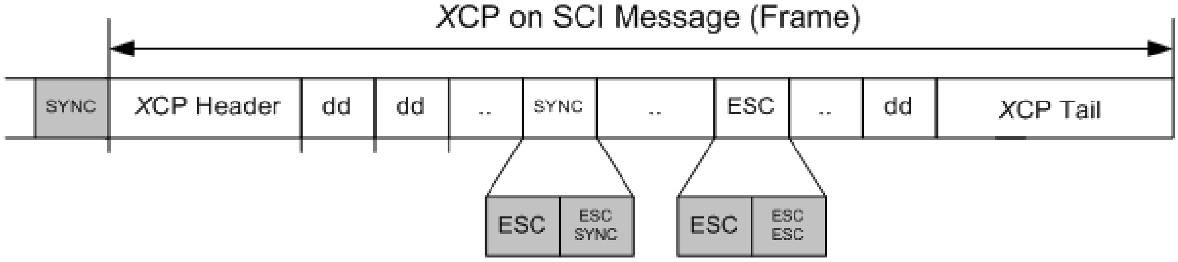


Diagram 5 : framing for XCP on SCI

For improving frame detection capabilities, a framing mechanism can be used.

The framing protocol defines two special characters: SYNC and ESC. If framing is used, every XCP on SCI Message is preceded by a SYNC character. If Inside the XCP on SCI Message a data byte occurs that is the same as SYNC, it is replaced by the two-byte sequence ESC+ESC_SYNC. If inside the XCP on SCI Message a data byte occurs that is the same as ESC, it is replaced by the two-byte sequence ESC+ESC_ESC.

With the FRAMING block in the ASAM MCD 2MC description file the slave can inform the master that it has to use the framing mechanism.

SYNC and ESC are configurable, ESC_SYNC = 0x01, ESC_ESC = 0x00.

1.5.2 Synchronous communication mode (SPI)

1.5.2.1 Full duplex mode

In synchronous (SPI) full duplex mode each direction has its own clock line.

Both directions are fully independent of each other and there are no restrictions regarding the protocol.

This mode is available for BYTE, WORD and DWORD SPI interfaces.

When using a WORD or DWORD SPI interface, alignment requirements must be met.

In this case the Identification_Field_Type for DAQ packets must be 0x01 or 0x03. Also the timestamp size must be 2 or 4 byte.

For *X*CP messages with odd length, a fill byte must be added in the *X*CP Tail.

Example:

LEN		CTR		ODT	DAQ	Timestamp		Data					
9	0	X	X	3	1	T_l	T_h	D[0]	D[1]	D[2]	D[3]	D[4]	0

DAQ message : WORD LEN, WORD CTR, WORD SPI, no CS, 1 fill byte

1.5.2.2 MASTER/SLAVE MODE

In synchronous (SPI) master/slave mode, one clock line is used for both directions.

The device which supplies the clock is called the SPI master.

In this case the SPI slave can only send a message, if the SPI master sends a message in parallel, because the clock is required from SPI master. The SPI slave must ensure, that the message to be transmitted starts synchronously to the message to be received.

For DAQ purposes the *X*CP slave should be the SPI master to ensure that it is able to transmit a DAQ packets with low latency.

During configuration time, when no DAQ is running, the *X*CP slave must transmit dummy packets in order to enable the *X*CP master to send command packets for configuration. This needs to be done frequently.

The dummy packet is defined as an event packet with the event code EV_TRANSPORT. All other bytes of this event packet must be zero to be compatible with future extensions.

Example:

LEN		CTR		PID	EV	Fill bytes					
2	0	X	X	0xFD	0xFF	0	0	0	0	0	0

Dummy message with WORD LEN and CTR:

The minimum length for all packets sent by the *X*CP slave must be at least MAX_CTO.

This is to ensure that all kind of command packets could be sent by the *X*CP master.

This mode is available for BYTE, WORD and DWORD SPI interfaces.

When using a WORD or DWORD SPI interface, the same alignment requirements as for Full Duplex Mode must be met

2 SPECIFIC COMMANDS FOR *X*CP ON SXI

There are no specific commands for *X*CP on SxI at the moment.

3 SPECIFIC EVENTS FOR *X*CP ON SXI

Table of Event Codes:

Event	Code	Remark
EV_DUMMY	0xFF	Optional

3.1 DUMMY PACKET

Category SPI Master/Slave mode only, optional
Mnemonic EV_DUMMY

Position	Type	Description
0	BYTE	Event Packet = 0xFD
1	BYTE	EV_TRANSPORT = 0xFF

The DUMMY packet is used for SPI applications when the SPI is used in Master/Slave mode.

In this case an event packet must be sent by the *X*CP slave (which is the SPI master) frequently to keep the communication alive.

If DAQ is running, no DUMMY packets are required.

Note:

The minimum message size must be at least MAX_CTO bytes, plus the size of the *X*CP Header, plus the size of the Checksum in the *X*CP Tail.

Therefore additional fill bytes must be added in the Tail of the event message.

4 INTERFACE TO ASAM MCD 2MC DESCRIPTION FILE

The following chapter describes the parameters that are specific for *X*CP on SxI.

4.1 ASAM MCD 2MC AML FOR *X*CP ON SXI

```
/*************************************************************************************/
/*                                                                                   */
/*   ASAP2 meta language for XCP on SxI V1.0                                         */
/*                                                                                   */
/*   2007-08-07                                                                      */
/*                                                                                   */
/*   Vector Informatik, Schuermans                                                   */
/*                                                                                   */
/*   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 SxI *************************************/

 struct SxI_Parameters { /* At MODULE */

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

  ulong;                      /* BAUDRATE [Hz] */

  taggedstruct { /* exclusive tags */
    "ASYNCH_FULL_DUPLEX_MODE" struct {
                                      enum {
                                         "PARITY_NONE" = 0,
                                         "PARITY_ODD"   = 1,
                                         "PARITY_EVEN" = 2
                                      };

                                      enum {
                                         "ONE_STOP_BIT"    = 1,
                                         "TWO_STOP_BITS" = 2
                                      };

                                       taggedstruct {
                                           block "FRAMING" struct {
                                                      uchar;   /* SYNC */
                                                      uchar;   /* ESC    */
                                                     };
                                       };
```

```
    "SYNCH_FULL_DUPLEX_MODE_BYTE”;
    "SYNCH_FULL_DUPLEX_MODE_WORD”;
    "SYNCH_FULL_DUPLEX_MODE_DWORD”;
    "SYNCH_MASTER_SLAVE_MODE_BYTE”;
    "SYNCH_MASTER_SLAVE_MODE_WORD”;
    "SYNCH_MASTER_SLAVE_MODE_DWORD”;
  };

  enum {
    "HEADER_LEN_BYTE"          = 0,
    "HEADER_LEN_CTR_BYTE"   = 1,
    "HEADER_LEN_FILL_BYTE”   = 2,
    "HEADER_LEN_WORD"         = 3,
    "HEADER_LEN_CTR_WORD" = 4,
    "HEADER_LEN_FILL_WORD" = 5
  };

  enum {
    "NO_CHECKSUM"      = 0,
    "CHECKSUM_BYTE"   = 1,
    "CHECKSUM_WORD" = 2
  };

};/*************************** end of SxI ***********************************/
```

4.2 IF_DATA EXAMPLE FOR XCP ON SXI

```
/begin XCP_ON_SxI

   0x0100           /* XCP on SxI version */

    25000           /* BAUDRATE */

    ASYNCH_FULL_DUPLEX_MODE
        PARITY_ODD
        TWO_STOP_BITS
        /begin FRAMING
           0x9A   /* SYNC */
           0x9B   /* ESC   */
        /end FRAMING

    HEADER_LEN_CTR_WORD
    NO_CHECKSUM

/end XCP_ON_SxI
```

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