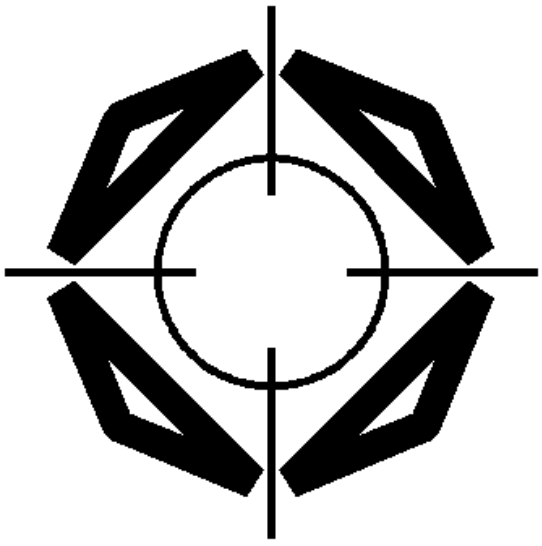
# XCP

**Version 1.0**

## “The Universal Measurement and Calibration Protocol Family”

**Part 4**

**Interface Specification**



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

**Dated:2003-04-08**

# Status of Document

| | |
|---|---|
| Date: | 2003-04-08 |
| Authors: | Roel Schuermans, Vector Informatik GmbH<br>Rainer Zaiser, Vector Informatik GmbH<br>Frank Hepperle, DaimlerChrysler AG<br>Hans Schröter, DaimlerChrysler AG<br>Reiner Motz, Robert Bosch GmbH<br>Andreas Aberfeld, Robert Bosch GmbH<br>Hans-Georg Kunz, Siemens VDO Automotive AG<br>Thomas Tyl, Siemens VDO Automotive AG<br>Robert Leinfellner, dSPACE GmbH<br>Hendirk Amsbeck, dSPACE GmbH<br>Harald Styrsky, Compact Dynamics GmbH<br>Boris Ruoff, ETAS GmbH<br>Lars Wahlmann, Accurate Technologies Inc. |
| Version: | 1.0 |
| Doc-ID: | XCP-Part 4- Interface Specification -1.0 |
| Status: | Released |
| Type | Final |

# Revision History

This revision history shows only major modifications between release versions.

| Date | Author | Filename | Comments |
|---|---|---|---|
| 2003-04-08 | R.Schuermans | | Released document |

## Table of contents

**0 Introduction ..... 6**

0.1 The *X*CP Protocol Family ..... 6

0.2 Documentation Overview ..... 7

0.3 Definitions and Abbreviations ..... 8

**1 Interface to ASAM MCD 2MC description file ..... 9**

1.1 ASAM MCD 2MC AML for *X*CP ..... 10

1.1.1 Protocol Layer and Transport Layer parts (XCP_definitions.aml) ..... 10

1.1.2 Overruling of default values (XCP_vX_Y_.aml) ..... 11

1.2 Example ASAM MCD 2MC ..... 12

1.2.1 Example of IF_DATA XCP (XCP_vX_Y_IF_DATA.a2l) ..... 12

1.2.2 Example of main *.a2l file (XCP_vX_Y_main.a2l) ..... 19

1.3 Consistency between ASAM MCD 2MC and slave ..... 23

**2 Interface to an external Seed&Key function ..... 24**

**3 Interface to an external Checksum function ..... 26**

## Table of diagrams:

Diagram 1 : structure of AML..... 9

# 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.

**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 (this document).

**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**

# 1 Interface to ASAM MCD 2MC description file

*X*CP consists of a generic Protocol Layer that can be transported on different Transport Layers.

**XCP_common_vX_Y.aml** in Part 2 of this specification specifies the AML description of the Common_Parameters of the Protocol Layer.

**XCP_on_##_vU_V.aml** in the respective Part 3 of this specification specifies the AML description of the specific parameters for each Transport Layer.

The main.a2l that describes a slave that supports *X*CP on different Transport Layers, includes an **XCP_definitions.aml** that contains a reference to the Common_Parameters and a reference to the parameters that are specific for the different Transport Layers the slave supports.

The main.a2l that describes a slave that supports *X*CP on different Transport Layers, also includes an **XCP_vX_Y.aml** that describes the structure of an "IF_DATA XCP ..". An "IF_DATA XCP .." has the possibility to describe default Transport Layer independent parameters, Transport Layer specific parameters and the overruling of the default parameters depending on the Transport Layer used.

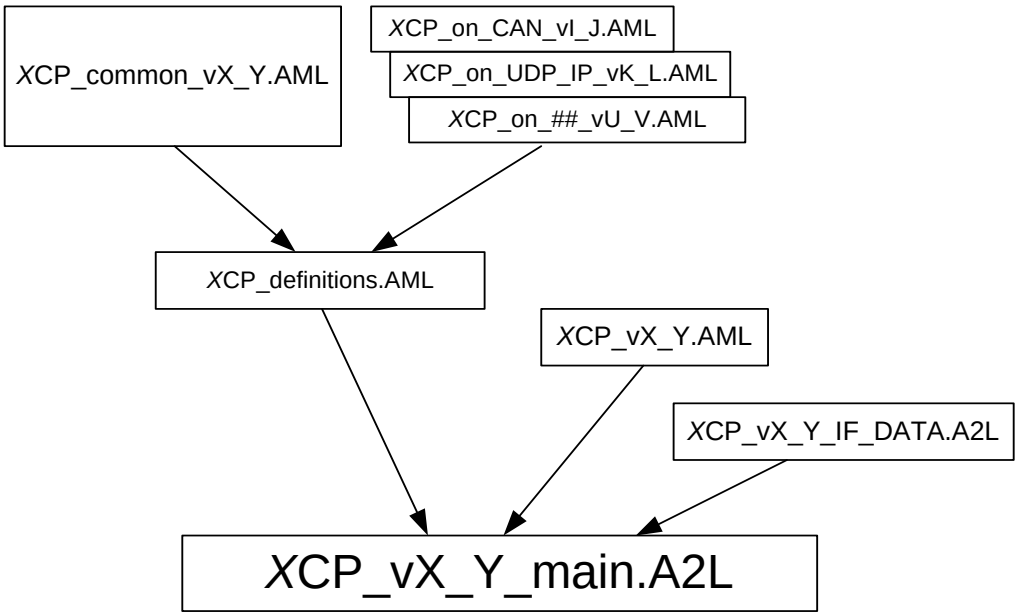



**Diagram 1 : structure of AML**

## 1.1 ASAM MCD 2MC AML for *X*CP

### 1.1.1 Protocol Layer and Transport Layer parts (XCP_definitions.aml)

The main.a2l that describes a slave that supports *X*CP on different Transport Layers, includes an **XCP_definitions.aml** that contains a reference to the Common_Parameters and a reference to the parameters that are specific for the different Transport Layers the slave supports.

Part 2 of the *X*CP specification "Protocol Layer Specification", defines the generic protocol, which is independent from the Transport Layer used.

**XCP_common_vX_Y.aml** in Part 2 specifies the AML description of the Common_Parameters of the Protocol Layer.

Part 3 of the *X*CP specification "Transport Layer Specification" defines the way how the *X*CP protocol is transported by a particular Transport Layer like CAN, TCP/IP and UDP/IP.

**XCP_on_##_vU_V.aml** in the respective Part 3 specifies the AML description of the specific parameters for each Transport Layer.

The Compatibility Matrix gives an overview of the allowed combinations of Protocol Layer and Transport Layer parts.

```
/*************************************************************************************/
/*    XCP_definitions.aml has to include                                             */
/*    a reference to a Protocol Layer part                                           */
/*    and                                                                            */
/*    (a) reference(s) to that(those) Transport Layer(s) your slave supports         */
/*                                                                                   */
/*    the Compatibility Matrix gives an overview of the allowed                      */
/*    combinations of Protocol Layer and Transport Layer parts                       */
/*                                                                                   */
/*************************************************************************************/
/************************ start of XCP definitions *********************************/

/include XCP_common_vX_Y.aml     /* Part 2 protocol layer part                    */

/include XCP_on_##_vU_V.aml      /* Part 3 transport layer part(s)                */

/************************ end of XCP definitions ***********************************/
```

**Example :**

This slave supports *X*CP protocol version 1.0, when transported on UDP/IP in version 1.0 and when transported on CAN in version 1.1

```
/************************ start of XCP definitions *******************************/

/include XCP_common_v1_0.aml     /* Part 2 protocol layer part                   */

/include XCP_on_UDP_IP_v1_0.aml  /* Part 3 transport layer UDP_IP                */
/include XCP_on_CAN_v1_1.aml     /* Part 3 transport layer CAN                   */

/************************ end of XCP definitions *********************************/
```

### 1.1.2 Overruling of default values (XCP_vX_Y_.aml)

The main.a2l that describes a slave that supports *X*CP on different Transport Layers, includes an **XCP_vX_Y.aml** that describes the structure of an "IF_DATA XCP ..".

An "IF_DATA XCP .." basically contains the Common_Parameters that are used as default values for communicating through XCP.

Inside a "/begin XCP_on_## .." there're the parameters that are specific for this particular Transport Layer. Also there exists the possibility to define Transport Layer specific values for the Common_Parameters that overrule the default Common_Parameters.

If looking for Common_Parameters for *X*CP on a specific Transport Layer, the master first has to check the availability of a Common_Parameters part inside the "/begin XCP_on_##" and use them if available. If this part is not available, the master has to use the default values for the Common_Parameters as defined in the "IF_DATA XCP ..".

```
/*************************************************************************************************/
/*  XCP_vX_Y.aml always has to have the same structure                                            */
/*  first there's a reference to the default parameters                                           */
/*  then there's (a) reference(s) to that(those) Transport Layer(s) your slave supports          */
/*                                                                                               */
/*************************************************************************************************/
/************************** start of XCP on different Transport Layers ***********************/
"XCP" struct {
  taggedstruct Common_Parameters ;          /* default parameters                                */
  taggedstruct {                            /* transport layer specific parameters               */
                                            /* overruling of the default parameters              */
    block "XCP_ON_##" struct {
      struct ##_Parameters ;                /* specific for ##                                   */
      taggedstruct Common_Parameters;       /* overruling of default                             */
    };
  };
};/***************************** end of XCP on different Transport Layers ***********************/
```

**Example :**

```
/************************** start of XCP on different Transport Layers ***********************/
"XCP" struct {
  taggedstruct Common_Parameters ;          /* default parameters                                */
  taggedstruct {                            /* transport layer specific parameters               */
                                            /* overruling of the default parameters              */
    block "XCP_ON_UDP_IP" struct {
      struct UDP_IP_Parameters ;            /* specific for UDP_IP                               */
      taggedstruct Common_Parameters;       /* overruling of default                             */
    };
    block "XCP_ON_CAN" struct {
      struct CAN_Parameters ;               /* specific for CAN                                  */
      taggedstruct Common_Parameters;       /* overruling of default                             */
    };
  };
};/***************************** end of XCP on different Transport Layers ***********************/
```

## 1.2 Example ASAM MCD 2MC

### 1.2.1 Example of IF_DATA XCP (XCP_vX_Y_IF_DATA.a2l)

This chapter gives an example of an IF_DATA XCP at MODULE for a slave that supports *X*CP on UDP/IP and *X*CP on CAN.

For *X*CP on UDP/IP the default values for the Common_Parameters are used.
For *X*CP on CAN the DAQ part of the Common_Parameters is overruled.

**Example:**

```
/begin IF_DATA XCP

  /begin PROTOCOL_LAYER

    0x0100                  /* XCP protocol layer 1.0 */

    0x0019                  /* T1 [ms] */
    0x0019                  /* T2 [ms] */
    0x0019                  /* T3 [ms] */
    0x0019                  /* T4 [ms] */
    0x0019                  /* T5 [ms] */
    0x0005                  /* T6 [ms] */
    0x00C8                   /* T7 [ms] */

    0x20                   /* MAX_CTO */
    0x00FF                  /* MAX_DTO */

    BYTE_ORDER_MSB_FIRST
    ADDRESS_GRANULARITY_WORD

    SEED_AND_KEY_EXTERNAL_FUNCTION     "MyS&K.DLL"

    OPTIONAL_CMD  GET_ID
    OPTIONAL_CMD  SET_REQUEST
    OPTIONAL_CMD  GET_SEED
    OPTIONAL_CMD  UNLOCK
    OPTIONAL_CMD  SET_MTA
    OPTIONAL_CMD  UPLOAD
    OPTIONAL_CMD  BUILD_CHECKSUM
    OPTIONAL_CMD  DOWNLOAD
    OPTIONAL_CMD  SET_CAL_PAGE
    OPTIONAL_CMD  GET_CAL_PAGE
    OPTIONAL_CMD  COPY_CAL_PAGE
    OPTIONAL_CMD  CLEAR_DAQ_LIST
    OPTIONAL_CMD  SET_DAQ_PTR
    OPTIONAL_CMD  WRITE_DAQ
    OPTIONAL_CMD  SET_DAQ_LIST_MODE
    OPTIONAL_CMD  START_STOP_DAQ_LIST
    OPTIONAL_CMD  START_STOP_SYNCH
    OPTIONAL_CMD  GET_DAQ_CLOCK

  /end PROTOCOL_LAYER
```

```
/begin DAQ

  DYNAMIC                    /* DAQ_CONFIG_TYPE */

  0x0100                    /* MAX_DAQ */
  0x0100                    /* MAX_EVENT_CHANNEL */
  0x05                      /* MIN_DAQ */

  OPTIMISATION_TYPE_ODT_TYPE_32
  ADDRESS_EXTENSION_FREE
  IDENTIFICATION_FIELD_TYPE_RELATIVE_WORD_ALIGNED

  GRANULARITY_ODT_ENTRY_SIZE_DAQ_WORD
  0x04                /* MAX_ODT_ENTRY_SIZE_DAQ */

  NO_OVERLOAD_INDICATION

  PRESCALER_SUPPORTED

  RESUME_SUPPORTED

  /begin STIM

    GRANULARITY_ODT_ENTRY_SIZE_STIM_WORD
    0x04              /* MAX_ODT_ENTRY_SIZE_STIM */
    BIT_STIM_SUPPORTED

  /end STIM

  /begin TIMESTAMP_SUPPORTED

    0x0100            /* TIMESTAMP_TICKS */

    SIZE_WORD
    UNIT_1MS

    TIMESTAMP_FIXED

  /end TIMESTAMP_SUPPORTED

  /begin EVENT

    "10_ms_task"            /* name */
    "10 ms"                /* short name */

    0x0000                  /* EVENT_CHANNEL_NUMBER */
    DAQ_STIM

    0x02                   /* MAX_DAQ_LIST */

    0x0A                    /* TIME_CYCLE */
    0x00                    /* TIME_UNIT */
    0x00                    /* PRIORITY */

  /end EVENT
```

```
  /begin EVENT

    "100_ms_task"             /* name */
    "100 ms"                /* short name */

    0x0001                  /* EVENT_CHANNEL_NUMBER */
    DAQ_STIM

    0x02                    /* MAX_DAQ_LIST */

    0x64                    /* TIME_CYCLE */
    0x00                    /* TIME_UNIT */
    0x10                    /* PRIORITY */

  /end EVENT

/end DAQ


/begin PAG

  0x01                    /* MAX_SEGMENTS */

  FREEZE_SUPPORTED

/end PAG


/begin PGM

  PGM_MODE_ABSOLUTE_AND_FUNCTIONAL

  0x02                    /* MAX_SECTORS */

  0x08                    /* MAX_CTO_PGM */

  /begin SECTOR

    "Lower sector"          /* name */
    0x00                  /* SECTOR_NUMBER */

    0x000000                /* address */
    0x20000               /* length */

    0x01                  /* Erase number */
    0x02                  /* Program number */

    0x00                  /* Programming method   */

  /end SECTOR
```

```
  /begin SECTOR

    "Upper sector"          /* name */
    0x01                  /* SECTOR_NUMBER */

    0x020000               /* address */
    0x20000               /* length */

    0x03                  /* Erase number */
    0x04                  /* Program number */

    0x00                  /* Programming method   */

  /end SECTOR

/end PGM

/begin XCP_ON_UDP_IP

    0x0100                    /* XCP on UDP_IP 1.0 */

    0x5555                    /* PORT          */

   ADDRESS "127.0.0.1"      /* ADDRESS   */

   /begin PROTOCOL_LAYER

      0x0100                  /* XCP protocol layer 1.0 */

      0x0019                  /* T1 [ms] */
      0x0019                  /* T2 [ms] */
      0x0019                  /* T3 [ms] */
      0x0019                  /* T4 [ms] */
      0x0019                  /* T5 [ms] */
      0x0005                  /* T6 [ms] */
      0x00C8                   /* T7 [ms] */

      0x20                   /* MAX_CTO */
      0x00FF                  /* MAX_DTO */

      BYTE_ORDER_MSB_FIRST
      ADDRESS_GRANULARITY_WORD

      OPTIONAL_CMD  FREE_DAQ
      OPTIONAL_CMD  ALLOC_DAQ
      OPTIONAL_CMD  ALLOC_ODT
      OPTIONAL_CMD  ALLOC_ODT_ENTRY

      COMMUNICATION_MODE_SUPPORTED  INTERLEAVED  0x0A

   /end PROTOCOL_LAYER

/end XCP_ON_UDP_IP
```

```
/begin XCP_ON_CAN

    0x0100                          /* XCP on CAN 1.0 */

    CAN_ID_BROADCAST  0x0100   /* auto-detection */

    CAN_ID_MASTER       0x0200   /* CMD/STIM */
    CAN_ID_SLAVE        0x0300   /* RES/ERR/EV/SERV/DAQ */

    BAUDRATE            500000   /* BAUDRATE */

    /begin DAQ_LIST_CAN_ID

      0x0000            /* for DAQ_LIST 0 */
      FIXED 0x310

    /end DAQ_LIST_CAN_ID

    /begin DAQ_LIST_CAN_ID

      0x0001            /* for DAQ_LIST 1 */
      FIXED 0x320

    /end DAQ_LIST_CAN_ID

    /begin DAQ_LIST_CAN_ID

      0x0002            /* for DAQ_LIST 2 */
      FIXED 0x330

    /end DAQ_LIST_CAN_ID

    /begin PROTOCOL_LAYER

      0x0100              /* XCP protocol layer 1.0 */

      0x000A              /* T1 [ms] */
      0x000A              /* T2 [ms] */
      0x000A              /* T3 [ms] */
      0x000A              /* T4 [ms] */
      0x000A              /* T5 [ms] */
      0x0000              /* T6 [ms] */
      0x0020              /* T7 [ms] */

      0x08                /* MAX_CTO */
      0x0008              /* MAX_DTO */

      BYTE_ORDER_MSB_FIRST
      ADDRESS_GRANULARITY_BYTE

      OPTIONAL_CMD  SHORT_UPLOAD
      OPTIONAL_CMD  SHORT_DOWNLOAD
      OPTIONAL_CMD  DOWNLOAD_NEXT

      COMMUNICATION_MODE_SUPPORTED  BLOCK  SLAVE  MASTER  0x0A 0x02

    /end PROTOCOL_LAYER
```

```
/begin DAQ

  STATIC                  /* DAQ_CONFIG_TYPE */

  0x0003                   /* MAX_DAQ */
  0x0002                   /* MAX_EVENT_CHANNEL */
  0x01                     /* MIN_DAQ */


  OPTIMISATION_TYPE_DEFAULT
  ADDRESS_EXTENSION_DAQ
  IDENTIFICATION_FIELD_TYPE_ABSOLUTE

  GRANULARITY_ODT_ENTRY_SIZE_DAQ_BYTE
  0x02               /* MAX_ODT_ENTRY_SIZE_DAQ */

  OVERLOAD_INDICATION_EVENT

  PRESCALER_SUPPORTED

  RESUME_SUPPORTED

  /begin DAQ_LIST

    0x0000                 /* DAQ_LIST_NUMBER */

    DAQ_LIST_TYPE DAQ

    MAX_ODT              0x01
    MAX_ODT_ENTRIES 0x02

    /begin PREDEFINED

       /begin ODT  0

          ODT_ENTRY  0  0x4000  0x00  0x01  0xFF
          ODT_ENTRY  1  0x4001  0x00  0x01  0xFF

       /end ODT

     /end PREDEFINED

  /end DAQ_LIST

  /begin DAQ_LIST

    0x0001                 /* DAQ_LIST_NUMBER */

    DAQ_LIST_TYPE DAQ_STIM

    MAX_ODT              0x03
    MAX_ODT_ENTRIES 0x10

  /end DAQ_LIST
```

```
      /begin DAQ_LIST

        0x0002                  /* DAQ_LIST_NUMBER */

        DAQ_LIST_TYPE DAQ_STIM

        MAX_ODT                 0x10
        MAX_ODT_ENTRIES 0x20

      /end DAQ_LIST

      /begin EVENT

        "10_ms_task"              /* name */
        "10 ms"                 /* short name */

        0x0000                  /* EVENT_CHANNEL_NUMBER */
        DAQ_STIM

        0x02                    /* MAX_DAQ_LIST */

        0x0A                    /* TIME_CYCLE */
        0x00                    /* TIME_UNIT */
        0x00                    /* PRIORITY */

      /end EVENT


      /begin EVENT

        "100_ms_task"             /* name */
        "100 ms"                /* short name */

        0x0001                  /* EVENT_CHANNEL_NUMBER */
        DAQ_STIM

        0x02                    /* MAX_DAQ_LIST */

        0x64                    /* TIME_CYCLE */
        0x00                    /* TIME_UNIT */
        0x10                    /* PRIORITY */

      /end EVENT

    /end DAQ

  /end XCP_ON_CAN

/end IF_DATA
```

### 1.2.2 Example of main *.a2l file (XCP_vX_Y_main.a2l)

This chapter gives an example of an ASAM MCD 2MC description file for a slave that supports *X*CP on UDP/IP and *X*CP on CAN.

**Example:**

```
/begin PROJECT XCP
          "XCP on different Transport Layers"

 /begin HEADER
     "Example of Default_Overruling principle"

  VERSION    "Sue01"
  PROJECT_NO XCPv01

 /end HEADER

 /begin MODULE XCP_Sim
           "Simulator by Vector Informatik GmbH"

  /begin A2ML

    /include XCP_definitions.aml

     block "IF_DATA" taggedunion if_data {

      /include XCP_v1_0.aml

    };

  /end A2ML

  /begin MOD_COMMON ""

   BYTE_ORDER MSB_LAST

  /end MOD_COMMON

  /include XCP_v1_0_IF_DATA.a2l
```

```
/begin MOD_PAR “”

 /begin MEMORY_SEGMENT

  Calib                  /* name */
  "Calibration data"        /* long identifier */
  DATA                    /* PrgType */
  FLASH                    /* Memory Type */
  INTERN                   /* Attribute */
  0x4000                  /* Address */
  0x200                   /* Size */
  -1 -1 -1 -1 -1           /* no mirrored segments */

  /begin IF_DATA XCP

   /begin SEGMENT

    0x00                  /* segment logical number */
    0x02                  /* number of pages */
    0x00                  /* address extension */

    0x00                  /* Compression method   */
    0x00                  /* Encryption method      */

    /begin CHECKSUM

     XCP_USER_DEFINED         /* checksum through external function */

     MAX_BLOCK_SIZE         0x100              /*  maximum block size */
     EXTERNAL_FUNCTION  "MyChecksum.DLL"    /*  name of function */

    /end CHECKSUM

    /begin PAGE

     0x00                  /* page number */

     ECU_ACCESS_ALLOWED_DONT_CARE
     XCP_READ_ACCESS_ALLOWED_DONT_CARE
     XCP_WRITE_ACCESS_NOT_ALLOWED

     INIT_SEGMENT 0x00       /* init segment */

    /end PAGE
```

```
      /begin PAGE

        0x01              /* page number */

        ECU_ACCESS_ALLOWED_DONT_CARE
        XCP_READ_ACCESS_ALLOWED_DONT_CARE
        XCP_WRITE_ACCESS_ALLOWED_WITH_ECU_ONLY

        INIT_SEGMENT 0x00     /* init segment */

      /end PAGE

      /begin ADDRESS_MAPPING

        0x04000           /* from */
        0x14000           /* to */
        0x100             /* length */

      /end ADDRESS_MAPPING

      /begin ADDRESS_MAPPING

        0x04100           /* from */
        0x24100           /* to */
        0x100             /* length */

      /end ADDRESS_MAPPING

    /end SEGMENT

  /end IF_DATA

 /end MEMORY_SEGMENT

/end MOD_PAR
```

```
  /begin MEASUREMENT

    Triangle                    /* name               */
    "Triangle test signal"      /* long identifier    */

    SBYTE                       /* DataType           */
    BitSlice.CONVERSION         /* conversion         */
    0                           /* resolution         */
    0                           /* accuracy           */
    -50  50                     /* lower, upper limit */

    BIT_MASK 0xFF

    ECU_ADDRESS 0x44A16

    FORMAT "%7.3"

    /begin IF_DATA XCP

      /begin DAQ_EVENT VARIABLE

        /begin AVAILABLE_EVENT_LIST
          EVENT 0001  EVENT 0002
        /end AVAILABLE_EVENT_LIST

        /begin DEFAULT_EVENT_LIST
          EVENT 0001
        /end DEFAULT_EVENT_LIST

      /end DAQ_EVENT

    /end IF_DATA

  /end MEASUREMENT

  /begin COMPU_METHOD

    BitSlice.CONVERSION
    ""
    RAT_FUNC
    "%2.0"
    "-"
    COEFFS 0 1 0 0 0 1

  /end COMPU_METHOD

 /end MODULE

/end PROJECT
```

## 1.3 Consistency between ASAM MCD 2MC and slave

The parameterization of the *X*CP protocol can be described in IF_DATA sections of an ASAM MCD 2MC description file.

If supported, the master also can read out almost all of these parameters directly from the slave.

If for a parameter there's both information in the ASAM MCD 2MC file and by reading it out from the slave, the master has to check the consistency of both values.

If the master detects an inconsistency, he has to inform the user about the detected inconsistency. The master has to give the user the possibility to decide whether the master for this parameter has to use the value from the ASAM MCD 2MC description file or the value read from the slave.

# 2 Interface to an external Seed&Key function

When calculating a Key from a Seed, the Master always has to use a user-defined algorithm. This algorithm is provided by the slave vendor. It contains functions to read out the provided privileges and to calculate a Key from a Seed.

The “SEED_AND_KEY_EXTERNAL_FUNCTION” parameter at the “PROTOCOL_LAYER” section in the ASAM MCD 2MC Description File, indicates the Name of the external function file the Master has to use. The parameter is an ASCII string that contains the name and the extension but does not contain the path to the file.

The integration of this function file is programming language and platform dependent. E.g. when using a Windows ® operating system, these “external functions” could be located in a MySeedNKey.DLL (Dynamically Linked Library). When using a UNIX ® operating system, these “external functions” could be located in a MySeedNKey.SO (Shared Object).

The mechanism required to include external functions files is tool specific.
However, the included functions and calling parameters themselves are specified in this chapter.

To have an easy handling for *X*CP there is only one external function file which may contain all algorithms to unlock all privileges or only a subset. That means the supplier can generate different external function files with different privilege level combinations.

The privilege levels are described based on the “Resource Mask” of *X*CP and coded as defined there. The ECU needs one algorithm for each privilege (if protected).

The external function file contains 2 functions: one to get information about the available privileges of this function file and one to calculate a key from a seed for the requested privilege.

**Function “XCP_GetAvailablePrivileges”:**

| Parameter name: | Data Typ | XCP_ComputeKeyFromSeed | Remarks |
|---|---|---|---|
| Return Value: | DWORD | Error Code | |
| Parameter 1: | BYTE * | Available Privilege | returns the privileges with available unlock algorithms in this external function file |

Function returns available privileges as XCP Resource Availability Mask.

The following error codes can be returned: XcpSkExtFncAck: o.k.

**Function: XCP_ComputeKeyFromSeed:**

| Parameter name: | Data Typ | XCP_ComputeKeyFromSeed | Remarks |
|---|---|---|---|
| Return Value: | DWORD | Error Code | |
| Parameter 1: | BYTE | Requested Privilege | => from Tool,<br>- input for external function<br>– input for GetSeed command |
| Parameter 2: | BYTE | Byte Length Seed | from answer of GetSeed |
| Parameter 3: | BYTE * | Pointer to Seed | |
| Parameter 4: | BYTE * | Byte Length Key | input: max bytes memory for key<br>output: byte length of key |
| Parameter 5: | BYTE * | Pointer to Key | |

The external function “XCP_ComputeKeyFromSeed “ should calculate Key from Seed for the requested privilege

Key = f(Seed, RequestedPrivilege) (only one privilege can be unlocked at once)

**Remark:**

Parameter 4 “Byte Length Key” must be initialised with the maximum Length of Key reserved by the Master when calling the external Seed&Key function. This makes sure that the Seed&Key function will not write into other memory than reserved. It is recommended to reserve 255 bytes since this is the maximum length that is possible.

The following error codes can be returned:

- XcpSkExtFncAck: o.k.
- XcpSkExtFncErrPrivilegeNotAvailable: the requested privilege can not be unlocked with this function
- XcpSkExtFncErrInvalidSeedLength: the seed length is wrong, key could not be computed
- XcpSkExtFncErrUnsufficientKeyLength: the space for the key is too small

**Example:**

Example source code for a Windows ® -DLL can be downloaded from .

www.asam.net under “Standards/ ASAM MCD/ I. Current specifications”

# 3 Interface to an external Checksum function

With the Checksum Type “XCP_USER_DEFINED”, the Slave can indicate that the Master for calculating the checksum has to use a user-defined algorithm implemented in an external function.

The integration of this function file is programming language and platform dependent. E.g. when using a Windows ® operating system, this “external function” could be located in a MyChecksum.DLL (Dynamically Linked Library). When using a UNIX ® operating system, this “external function” could be located in a MyChecksum.SO (Shared Object).

The mechanism required to include external functions files is tool specific.
However, the included function and calling parameters themselves are specified in this chapter.

| Type | Name | Description |
|---|---|---|
| 0xFF | XCP_USER_DEFINED | User defined algorithm, in externally calculated function |

The “EXTERNAL_FUNCTION” parameter at the “CHECKSUM” block at an *X*CP SEGMENT in the ASAM MCD 2MC Description File, indicates the Name of the external function file the Master has to use. The parameter is an ASCII string that contains the name and the extension but does not contain the path to the file.

Chapter “Win32 API for the ASAP1a Checksum Algorithm DLL” in the specification of the ASAM MCD 2MC Description File Format, describes the API for calling a Win32 Checksum.DLL.

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

Tel.: (+49) 8102 / 895317
Fax.: (+49) 8102 / 895310
E-mail: info@asam.net
Internet: www.asam.net