iLinkRT™

Communication Protocol Definition

Protocol Revision: 1.3
Document Revision: 1.4
Status: Released

Contents

1 General

This protocol definition document describes a multi-client / multi-server architecture for the purpose of fast and channel based communication between servers and clients.

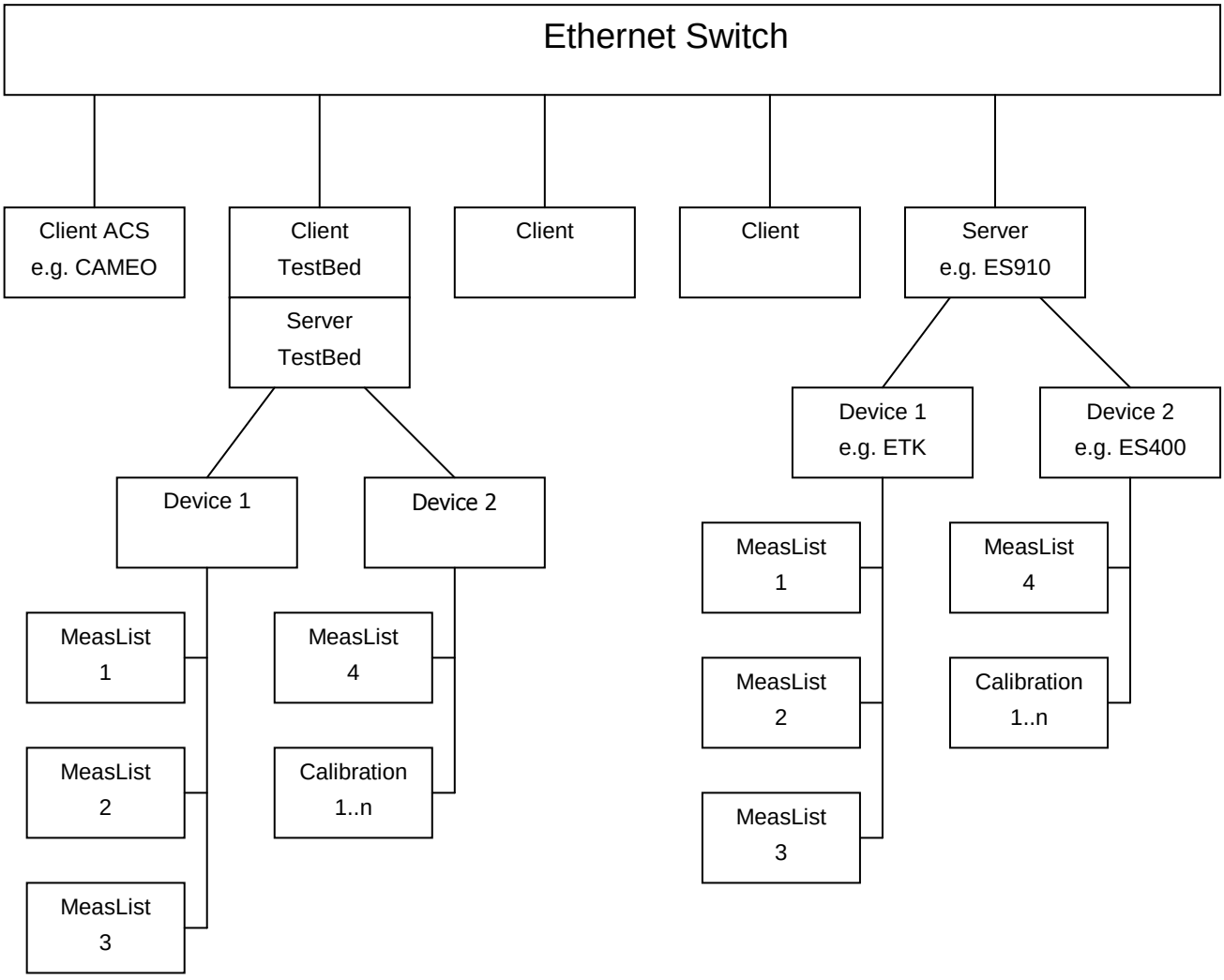
The implementation is based on XCP on Ethernet, UDP/IP.

The following description will focus on the "High Performance ECU Calibration" to explain the server and client implementations.

This document describes which features are implemented in the XCP protocol, which user defined commands are available and the present deviations to the XCP standard.

The supported XCP protocol version number is V1.0.

1.1 Architectural overview



1.2 High Performance ECU Calibration (HPEC)

For several automated ECU calibration tasks it is necessary to have a fast interface between the participating systems (ACS, AuSy, ECU, etc.).

For fast data exchange between an ECU and client applications the ES910 is utilized. To avoid a communication bottle neck between the ECU and the ES910 an ETK access is used.

Since client applications require physical ECU values, the ES910 simulates a physical ECU that already contains the hex-to-phys-converted data. The client application can set up an XCP based communication with this "Virtual Physical ECU". It can measure variables and exchange calibration values that were specified in INCA in advance.

1.3 XCP Parameters

1.3.1 Time-out handling

The time-out value t1 is defined to be 1s.

1.3.2 MAX_DTO

MAX_DTO is 1468 Bytes (maximum length of DTO packets in Bytes).

1.3.3 Endianess

The byte order mode is BYTE_ORDER = 1 (Motorola).

1.3.4 Address Granularity

The address granularity is AG = 1

1.3.5 Communication modes

The ES910 supports standard communication mode and interleaved mode. Block mode is not supported.
The interleaved mode runs with a fixed QUEUE_SIZE of 4

1.3.6 Data Acquisition Configuration Type

Only static DAQ lists are supported. Dynamic DAQ configuration is not supported.

1.3.7 Number of DAQ lists and events

There is a one to one relation between the DAQ lists and events. This means every event channel contains exactly one DAQ list.

1.3.8 Predefined DAQ Lists

All DAQ lists are predefined and cannot be modified by the master.

1.3.9 Identification Field Type

The ES910 uses the „CTO Packet Code and absolute ODT number" identification field type.

Every DAQ list contains only one ODT.

1.3.10 Object Descriptor Table

The ES910 uses the following properties for DAQ. STIM is not supported.

- GRANULARITY_ODT_ENTRY_SIZE = 1
- MAX_ODT_ENTRY_SIZE = 8
- ADDRESS_GRANULARITY = 1

1.3.11 Overload Indication

Overload indication is not supported.

1.3.12 Prescaler

Prescaler is not supported.

1.3.13 Bit Offset

Bit offset is not supported.

1.3.14 Timestamp

The ES910 supports 32 bit timestamp. It is not possible to switch the timestamp off (TIMESTAMP_FIXED).

1.3.15 DAQ List Types

The ES910 supports only DAQs, no STIMs are supported.

1.3.16 Event Channels

Event channels for DAQs are supported.

1.3.17 Freeze

FREEZE is not supported.

1.4 XCP on UDP Restrictions

IP fragmentation is not supported.

2 Definitions and Abbreviations

Abbreviation	Description
UDP/IP	Unified Data Protocol / Internet Protocol
TS	Time Stamp
PTV	Protocol version
STA	Status of device
PID	Packet identification
DVID	Device ID
DTY	Data type
CTY	Parameter dimension, calibration label type
CMT	Bitmask, possible calibration methods
CID	Compatibility ID

2.1 Protocol Versions

Protocol Version	Response GET_ALL_SERVER	Compatibility ID
V1.1	0x07	Not applicable
V1.2	0x08	0x01
V1.3	0x09	0x01

2.2 PID, Packet Identification Codes

PID	Type	Description
0xF1 01	BYTE	GET_ALL_SERVER
0xFF	BYTE	GET_ALL_SERVER, response
0xF1 02	BYTE	READ_DAQ_EXTENDED
0xFF	BYTE	READ_DAQ_EXTENDED, response
0xF1 03	BYTE	READ_CAL_EXTENDED(deprecated)
0xFF	BYTE	READ_CAL_EXTENDED, response (deprecated)
0xF1 04	BYTE	CAL_DOWNLOAD_ADVANCED
0xFF	BYTE	CAL_DOWNLOAD_ADVANCED, response
0xF1 05	BYTE	CAL_UPLOAD_ADVANCED (deprecated)
0xFF	BYTE	CAL_UPLOAD_ADVANCED, response (deprecated)
0xF1 06	BYTE	READ_CAL_EXTENDED_V2
0xFF	BYTE	READ_CAL_EXTENDED_V2, response
0xF1 07	BYTE	CAL_UPLOAD_ADVANCED_V2
0xFF	BYTE	CAL_UPLOAD_ADVANCED_V2, response

2.3 CMT, calibration methods

CMT	Type	Description
0x00	BYTE	Flat [1]
0x01	BYTE	Increment
0x02	BYTE	Offset
0x03	BYTE	Area [1]

2.4 Data type definitions

2.4.1 Data type codes (DTY)

DTY	Type	Description
0x00	Float32_IEEE	4-byte floating point number (IEEE format)
0x01	Float64_IEEE	8-byte floating point number (IEEE format)
0x02	UBYTE, BYTE	1-byte unsigned integer
0x03	SBYTE	1-byte signed integer
0x04	UWORD, WORD	2-byte unsigned integer
0x05	SWORD	2-byte signed integer
0x06	ULONG	4-byte unsigned long
0x07	SLONG	4-byte signed integer

2.4.2 Parameter dimension, calibration label type (CTY) – see standard ASAM-MCD-2 MC

CTY	Type	Description
0x00	BYTE	Scalar (“VALUE”)
0x01	BYTE	Curve - 1-dimensional array with physical axes
0x02	BYTE	Map - 2-dimensional array with physical axes
0x03	BYTE	Array (“VAL_BLK”) – 1-dimensional or 2-dimensional array without physical axes

2.4.3 String

Position	Type	Description
0,1	WORD	Number of characters (including termination)
2,3..n	BYTE	UTF-8 characters (terminated with \0)

[1] In “Version 1.2” the only supported calibration method is “flat” and “area”

3 3 Deviations to the XCP Standard

The XCP standard demands that all communication is done via the same UDP port and that there is one counter for all packages sent from the slave to the master.

In the standard XCP protocol all packages from the master and the slave are sent on one and the same UDP port. In this implementation two separate UDP ports are used for XCP commands and Data Acquisition Packages. Furthermore the CMDs and DAQs increment their counters independently.

The UDP port for the CMD communication will be the port the master used to send the CONNECT command. The port for DAQ list will be transmitted in the user defined command GET_ALL_SERVER.

The DAQ lists are sent out via multicast or unicast messages. The master is informed of the method used in the response to GET_ALL_SERVER.

XCP allows a 1 byte definition for MAX_CTO, which leads to a maximum command size of 256 Bytes. In this implementation, the MAX_CTO size is defined to be 1468 Bytes (maximum length of CTO packets in Bytes). However, the slave returns 0xFF for MAX_CTO in the response to CONNECT.

4 XCP commands

4.1 Standard commands

Mandatory commands:

- CONNECT
- DISCONNECT
- GET_STATUS
- SYNCH

Available optional commands:

- SHORT_UPLOAD

4.2 Calibration commands

Mandatory commands:

- DOWNLOAD – not implemented

Available optional commands:

- SHORT_DOWNLOAD

4.3 Page Switching commands

No page switching available.

4.4 Data Acquisition commands

Mandatory commands:

- CLEAR_DAQ_LIST – not implemented (static DAQs!)
- WRITE_DAQ – not implemented (static DAQs!)
- SET_DAQ_LIST_MODE – not implemented (static DAQs!)
- SET_DAQ_PTR
- GET_DAQ_LIST_MODE
- START_STOP_DAQ_LIST
- START_STOP_SYNCH

Available optional commands:

- GET_DAQ_EVENT_INFO
- GET_DAQ_RESOLUTION_INFO

4.5 Event Packets

Available optional events:

- EV_CMD_PENDING
- EV_SESSION_TERMINATED

5 User Defined Commands

In addition to the above listed standard XCP command some user defined commands are used. The purpose of the user defined commands is to avoid the necessity for the master to parse the a2l file and to increase the performance of downloads of entire maps or curves.

5.1 GET_ALL_SERVER

The GET_ALL_SERVER command is sent from the client to the slave before the communication is started with a CONNECT. The slave responds to the command with different information the master needs to know to set up a communication. This command is sent via a broadcast message. The master receives the IP address of the slave with the response package.

The UDP port and multicast address for DAQ lists will be communicated from the slave to the master in the GET_ALL_SERVER response. If the slave is configured to send DAQs via unicast 0 is returned for the multicast address.

Master -> Slave

Position	Type	Description
0	BYTE	PID = 0xF1
1	BYTE	Sub command 0x01
n…m	STRING	Senders name e.g. "CAMEO"

5.1.1 Response, GET_ALL_SERVER

Slave -> Master

Position	Type	Description
0	BYTE	PID = 0xFF
1	BYTE	Reserved
2	BYTE	PTV, Protocol version
3	BYTE	CID, Compatibility ID
3..n	STRING	Server name e. g. "ES910_1"
n+1, n+2	WORD	UDP port of DAQ lists
n+3..n+6	4 BYTE	Multicast address of DAQ lists (0 in case of unicast)
n+7	BYTE	Number of devices
n+8...m	-	Device information 1
...	...	...
m+1	…	Device information x

Device information

For each device

Position	Type	Description
n+8	BYTE	Device ID (unique)
n+9	BYTE	Number of DAQ lists
n+10..m	STRING	Name of device (e. g. "ETK1")

5.2 READ_DAQ_EXTENDED

With the READ_DAQ_EXTENDED command the master can request some additional information about the variables transmitted in the predefined DAQ lists. The command is used in the same way as the standard XCP command READ_DAQ. Also the SET_DAQ_PTR needs to be set before DAQ entries can be read.

Master -> Slave

Position	Type	Description
0	BYTE	PID = 0xF1
1	BYTE	Sub command 0x02

5.2.1 Response, READ_DAQ_EXTENDED

Slave -> Master

Position	Type	Description
0	BYTE	PID = 0xFF
1	BYTE	Address extension
2-5	4 BYTE	Address of DAQ element
6..n	STRING	Name of DAQ label
n+1..m	STRING	Unit of DAQ label
m+1..k	STRING	Description of DAQ label
k+1	BYTE	Data type body (DTY)

5.3 READ_CAL_EXTENDED (deprecated)

This command is deprecated (see Sub command 0x06 for latest version).

READ_CAL_EXTENDED is used by the master to get information about all labels that can be calibrated. The SHORT_DOWNLOAD or the CAL_DOWNLOAD_ADVANCED commands can only download to addresses that are available in the response to READ_CAL_EXTENDED.

If the calibration element number exceeds the number of calibration elements on the target an error frame will be sent as response.

Master -> Slave

Position	Type	Description
0	BYTE	PID = 0xF1
1	BYTE	Sub command 0x03
2	BYTE	DVID, Device ID
3,4	WORD	Calibration element number

Note

The counting of the "calibration element number" restarts for each device at "0"

5.3.1 Response, READ_CAL_EXTENDED (deprecated)

Slave -> Master

Position	Type	Description
0	BYTE	PID = 0xFF
1	BYTE	Address extension
2-5	4 BYTE	Address of CAL element
6..n	STRING	Name of calibration label
n+1..m	STRING	Unit of calibration label
m+1..k	STRING	Description of calibration label
k+1	BYTE	Parameter dimension, calibration label type (CTY)
k+2	BYTE	Data type body (DTY)
k+3	WORD	Bitmask, possible calibration methods (CMT)
k+5	WORD	Size of x-Axis for maps, curves and arrays (1 for others)
k+7	WORD	Size of y-Axis for maps and 2D arrays (1 for others)

5.4 CAL_DOWNLOAD_ADVANCED

To enable the download of entire maps, curves and arrays, the CAL_DOWNLOAD_ADVANCED command is introduced. Two different modes are supported.

5.4.1 Flat mode

In the flat mode the master sends one value to the client. This value will then be written to the whole area of the map, curve or array. This reduces the communication traffic. If a value is transmitted with this command the record layout is calculated by the slave and the desired values are downloaded to the corresponding addresses.

As an alternative to SHORT_DOWNLOAD, Scalars are also supported in this mode.

Master -> Slave

Position	Type	Description
0	BYTE	PID = 0xF1
1	BYTE	Sub command 0x04
2	BYTE	Address extension
3-6	4 BYTE	Address
7	BYTE	0 (flat mode)
8…	ELEMENT	Data element

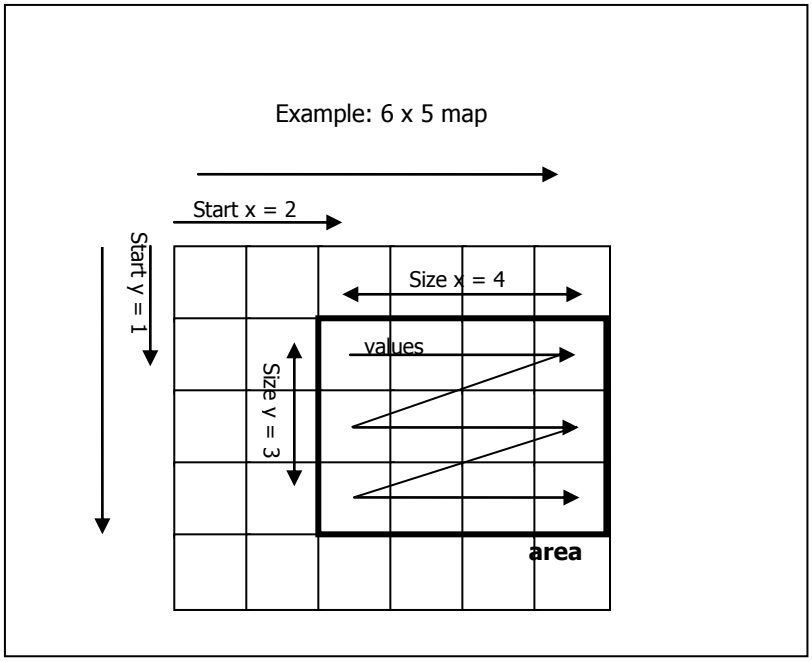
5.4.2 Area mode

In the area mode only the desired area of a map, curve or array is filled with values.

As an alternative to SHORT_DOWNLOAD, Scalars are also supported in this mode. (Start x & Start y must be 0; Size x & Size y must be 1).

Master -> Slave

Position	Type	Description
0	BYTE	PID = 0xF1
1	BYTE	Sub command 0x04
2	BYTE	Address extension
3-6	4 BYTE	Address
7	BYTE	3 (area mode)
8,9	WORD	Start x (starts with 0)
10,11	WORD	Start y (starts with 0)
12,13	WORD	Size x
14,15	WORD	Size y (1 for curves and 1D arrays)
16…	ELEMENTS	Z data elements (size x * size y values, row oriented)



The picture shows which values are filled with an example command for a 6 by 5 map.

5.4.3 Response, CAL_DOWNLOAD_ADVANCED

Slave -> Master

Position	Type	Description
0	BYTE	PID = 0xFF

5.5 CAL_UPLOAD_ADVANCED (deprecated)

This command is deprecated (see Sub command 0x07 for new version).

This command is to upload a sub area of a map, curve or array. It has the same parameters like the CAL_DOWNLOAD_ADVANCED command in area mode (except for the mode and data elements). The same meaning applies to the parameters Start x, Start y, Size x, Size y.

As an alternative to SHORT_UPLOAD, Scalars are also supported by this command. (Start x & Start y must be 0; Size x & Size y must be 1).

Master -> Slave

Position	Type	Description
0	BYTE	PID = 0xF1
1	BYTE	Sub command 0x05
2	BYTE	Address extension
3-6	4 BYTE	Address
7,8	WORD	Start x (starts with 0)
9,10	WORD	Start y (starts with 0)
11,12	WORD	Size x
13,14	WORD	Size y (1 for curves and 1D arrays)

5.5.1 Response, CAL_UPLOAD_ADVANCED (deprecated)

Slave -> Master

Position	Type	Description
0	BYTE	PID = 0xFF
1…	ELEMENTS	Z data elements (size x * size y values, row oriented)

5.6 READ_CAL_EXTENDED_V2

READ_CAL_EXTENDED_V2 is used by the master to get information about all labels that can be calibrated. The SHORT_DOWNLOAD or the CAL_DOWNLOAD_ADVANCED commands can only download to addresses that are available in the response to READ_CAL_EXTENDED_V2.

If the calibration element number exceeds the number of calibration elements on the target an error frame will be sent as response.

Master -> Slave

Position	Type	Description
0	BYTE	PID = 0xF1
1	BYTE	Sub command 0x06
2	BYTE	DVID, Device ID
3,4	WORD	Calibration element number

Note

The counting of the "calibration element number" restarts for each device at "0"

5.6.1 Response, READ_CAL_EXTENDED_V2

Slave -> Master

Position	Type	Description
0	BYTE	PID = 0xFF
1	BYTE	Address extension
2-5	4 BYTE	Address of CAL element
6..n	STRING	Name of calibration label
n+1..m	STRING	Unit of calibration label
m+1..k	STRING	Description of calibration label
k+1	BYTE	Parameter dimension, calibration label type (CTY)
k+2	BYTE	Data type body (DTY)
k+3	WORD	Bitmask, possible calibration methods (CMT)
k+5	WORD	Size of x-Axis for maps, curves and arrays (1 for others)
k+7	WORD	Size of y-Axis for maps and 2D arrays(1 for others)
k+8	BYTE	DTY of x-Axis for maps and curves (0xFF for others)
k+9	BYTE	DTY of y-Axis for maps (0xFF for others)

5.7 CAL_UPLOAD_ADVANCED_V2

This command is to upload a sub area of a map, curve or array. It has the same parameters like the CAL_DOWNLOAD_ADVANCED command in area mode (except for the data elements). The same meaning applies to the parameters Start x, Start y, Size x, Size y.

As an alternative to SHORT_UPLOAD, Scalars are also supported by this command. (Start x & Start y must be 0; Size x & Size y must be 1).

Master -> Slave

Position	Type	Description
0	BYTE	PID = 0xF1
1	BYTE	Sub command 0x07
2	BYTE	Address extension
3-6	4 BYTE	Address
7	BYTE	Mode (3: z-values only, 4: z-values and axis values)
8,9	WORD	Start x (starts with 0)
10,11	WORD	Start y (starts with 0)
12,13	WORD	Size x
14,15	WORD	Size y (1 for curves and 1D arrays)

5.7.1 Response, CAL_UPLOAD_ADVANCED_V2 with z-values only (Mode 3)

Slave -> Master

Position	Type	Description
0	BYTE	PID = 0xFF
1..m	ELEMENTS	Z data elements (size x * size y values, row oriented)

5.7.2 Response, CAL_UPLOAD_ADVANCED_V2 with z-values and axis values (Mode 4)

Scalars and arrays do not have physical axes. Mode 4 is only possible for maps and curves and is not allowed for arrays or scalars.

Slave -> Master

Position	Type	Description
0	BYTE	PID = 0xFF
1..m	ELEMENTS	Z data elements (size x * size y values, row oriented)
m+1..n	ELEMENTS	X data elements (size x)
n+1..o	ELEMENTS	Y data elements (size y), not applicable for curves

6 Change History

6.1 Document Revision 1.4

Protocol Revision 1.2:

1. Added definition for XCP parameter timeout t1
2. Added definition of MAX_DTO
3. Erased Address Extension from XCP Parameters, including comment that it is not used
4. Changed QUEUE_SIZE in interleaved mode from 3 to 4
5. Erased Busy Error response if QUEUE_SIZE is violated
6. Deletion of CID from abbreviations (Calibration ID) -> Not used in spec
7. Introduction of CID (Compatibility ID)
8. Updated footnote for calibration methods Increment & Offset
9. Definition table added
10. Support XCP command GET_DAQ_RESOLUTION_INFO
11. Support XCP events EV_CMD_PENDING, EV_SESSION_TERMINATED
12. UTF-8 for Strings
13. More precise definition of Calibration Types
14. DAQ-Lists to be sent via multicast or unicast
15. Added definition of MAX_CTO to deviations from XCP
16. Added CAL_UPLOAD_ADVANCED (Sub command 0x05) to PID overview
17. Changed Status of Device in GET_ALL_SERVER to "reserved"
18. Minor modification of description of READ_CAL_EXTENDED, no effect on command itself
19. Defined "size y" for READ_CAL_EXTENDED and CAL_UPLOAD_ADVANCED to properly include 2D arrays
20. Defined "size x" and "size y" for READ_CAL_EXTENDED, CAL_DOWNLOAD_ADVANCED & CAL_UPLOAD_ADVANCED to start with the value "1"
21. CAL_DOWNLOAD_ADVANCED & CAL_UPLOAD_ADVANCED descriptions changed to include arrays & scalars

Protocol Revision 1.3:

1. Declared Sub command 0x03 deprecated
2. Declared Sub command 0x05 deprecated
3. Introduced Sub Command 0x06 (READ_CAL_EXTENDED_V2 to return data type (DTY) of axes)
4. Introduced Sub Command 0x07 (CAL_UPLOAD_ADVANCED_V2 with option to upload axis points)