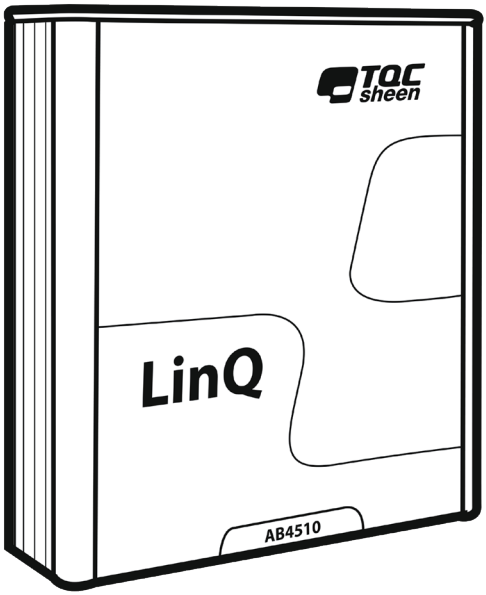


LinQ
AB4510

Operating Instructions (V1.0 0319)

IMPORTANT! Before taking this instrument in use we strongly advise you to read this manual carefully.

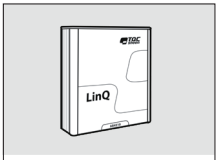
Index

1 Instrument Data

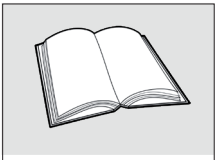
1.1 Name / Article Number

AB4510	LinQ

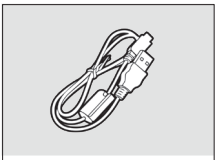
1.2 Scope of Supply



LinQ

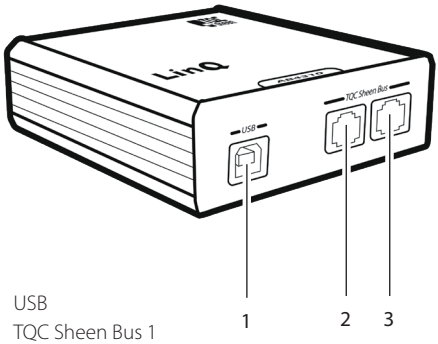


Manual



USB cable

1.3 Instrument Controls and Functions



1 USB
2 TQC Sheen Bus 1
3 TQC Sheen Bus 2

2 Installation

Connect the USB-cable from the LinQ box to the PC. If needed a driver can be found at: https://www.ftdichip.com/Drivers/VCP.htm.
A comport will be shown in the Device Manager on a Window PC. Connect the UTP-cable to the TQC-Bus port on the AFA.

Turn on the AFA. On the initialize page Press and hold the OK button until it shows 'This device is external controlled. The instrument is now ready for external control. First the instrument should be initialized by sending the Initialize command. After it is initialized the Instrument accept other commands such as Movement etc.

A Demo application with source code is available at TQC Sheen.

3 Device Protocol

This document covers the communication protocol with the Universal Lab Controller (ULC) equipped Lab Instruments. The protocol includes a description of a data transport and otherwise a declaration of the communication protocol. The main principle of the communication is master/slave. The data logger is the slave and will only respond to given requests. It will not independently give answers. The request and response buffers are limited to 64-byte.

4 Timeout

For ULC like devices the slave is expected to respond in 100 milliseconds on a command, if the slave has not responded the master will send the command again with a maximum of 3 retries.

5 Communication Protocol

Every command or response packet starts with two synch bytes filled with 0xCDCD.
Then a conversation byte a "serial number" to support the tracking of the commands. Next a byte as command ID which represent the specified command.
If the most significant bit in the first command ID byte is set then next byte is an optional command Id byte. Then follows an enumeration command or the data payload. At the end of every data packet there is a binary based checksum to verify the correctness.

CRC 32
The used check sum is the standard 32 and will be calculated of the whole packet, excluding the current CRC) (e.g. http://www.faqs.org/rfcs/rfc3309.html).

Code snipit:

```
private static uint CalcCrc(byte[] buffer, int length)
{
      uint crc = 0;
      const uint one = 0x00000001;

      for (uint i = 0; i < length; i++)
      {
            crc = crc ^ buffer[i];
            uint j;
            for (j = 0; j < 8; j++)
            {
                  if ((crc & one) == one)
                        crc = (crc >> 1) ^ 0xEDB88320;
                  else
                        crc = crc >> 1;
            }
      }

      return crc;
}
```

6 Base Packets

Below a schematic overview of the composition of the packets:

Orange	UINT16 0xCDCD	Error Data
Light Grey	UINT8	Conversation Id
Blue	UINT8	Command Id 1
Green	UINT8	Payload Length
White	n bytes *1	Payload
Dark Grey	UINT32 *2	32 Bit CRC checksum

*1 When the payload is defined as variable the first byte always present the number of bytes that follows.

*2 The checksum will include the synch bytes, conversation ID bytes, command bytes and the payload..

7 Request Packet Structure

[Synch16 bits] [Conversation Byte]
[CmdByte1][Payload length][Payload……][Checksum-32bit]

Request

CD	CD	CId	FE	Payload length	Payload	UINT32 checksum
1 byte	1 byte	1 byte	1 byte	1/2 byte	n bytes	4 bytes

If the most significant bit in the first command ID byte is set then next byte is an optional command Id byte.

If more than one commands is send in one data transaction, the device accepts only complete commands. If it doesn't fit start a new block, data transaction.

8 Response Packet Structure

[Synch16 bits] [Conversation Byte] [CmdByte1][Payload length]
[Payload……][Checksum-32bit]

Response

CD	CD	Cld	FE	Payload length	Payload	UINT32 checksum
1 byte	1 byte	1 byte	1 byte	1/2 byte	n bytes	4 bytes

The conversation id in the response will be the same as the one from the request.

Where CmdByte1 and or 2 are the Bit inversion of the request command byte(s).

So for example
Request 0x10 could look like the following…
CD CD 01 **10** {some 32 bit checksum}

Response
CD CD 01 **EF** {payload} {some other 32-bit checksum}

9 Special Response Packets

The device has some special responses to indicate something either so normal.
Payload will contain {Error Value}{Error Data}, 2x UINT16 = 4 bytes.

Response

CD	CD	Cld	FE	Payload length	2x UINT16 Error code	UINT32 checksum
1 byte	1 byte	1 byte	1 byte	1/2 byte	4 bytes	4 bytes

Error Value (UINT16)	Meaning	Error Data (UINT16)
0	OK	-
1	Unknown Error	-
2	Command Corrupt (CRC error)	-
3	Command Out of Sequence	-
4	Unexpected Command	-
5	Device Busy	-
6	Command NOT Supported	Cmd code
7	Cmd Enumeration NOT Supported	Cmd Enumeration
8	Batch NOT available	Batch Id
9	Data out of range	
10	Cmd Mode not Supported	Mode

The device will always respond to all commands even if it's **error 6**, "command not supported" (see above). This will allow the commands to be extended and enhanced in the future.

10 Data Types

To prevent obscurity about how the data is present follows here a definition of the used data types below.

CHAR	8-bit value to save characters
INT8	Signed 8-bit
UINT8	Unsigned 8-bit
INT16	Signed 16-bit
UINT16	Unsigned 16-bit
INT32	Signed 32-bit
UINT32	Unsigned 32-bit
FLOAT	32-bit IEEE-754 floating-point format
DOUBLE	64-bit IEEE-754 floating-point format

Compound data types derived from the core types…

TIMESTAMP	32-bit, (time_t) seconds since 12:00:00 AM 1970-01-01
SAMPLERATE	Signed 32 bit, tenths of seconds
TEMPERATUREINT	16, tenths of degrees C
GLOSS	INT16, tenths of points of gloss
PERCENT	INT16, tenths of percentage points
TIMESTAMP2	Signed 16 bit tenths of seconds from start of run
STRING[X]	X Chars long excluding a NULL
VARSTRING[X]	Byte 0 is the length, followed by a variable length non null terminated string.

All data will go across as Little-Endian.
Signed values are saved in 2-complement representation.

11 AFA Specific Commands

11.1 Setting of AFA values Command (0x64)

Request

CD	CD	Cld	64	Payload length	UINT16 enumeration	Payload	UINT32 checksum
1 byte	1 byte	1 byte	1 byte	1 byte		2 bytes	4 bytes

Response

CD	CD	Cld	9B	Payload length	Payload	UINT32 checksum
1 byte	1 byte	1 byte	1 byte	1 byte	n bytes	4 bytes

11.2 Reading of AFA values Command (0x65)

Request

CD	CD	Cld	65	Payload length	UINT16 enumeration	Payload	UINT32 checksum
1 byte	1 byte	1 byte	1 byte	1 byte	2 bytes		4 bytes

Response

CD	CD	Cld	9A	Payload length	Payload	UINT32 checksum
1 byte	1 byte	1 byte	1 byte	1 byte	2 bytes	

11.3 Conversation ID's

Conversation ID	
0x00	Mainboard AFA, including Drying Time Recorder option
0x01	Heated bed option

Commands **Conversation ID=0x00, Mainboard AFA**

Enumeration (HEX value)	Description	Payload Length (bytes)	Payload Definition
0x00	Hardwarestatus (ReadOnly)	U32	U32 Status (See HardwareStatus table)
0x02	Initialisation	U8	Write : Dummy byte starts initialisation Read: 0-100% Completion, 0xFF Failure
0x04	Generate audio	U8 + U16 + U16	U8 0-100 % Volume, U16 Fequency U16 Duration in ms
0x100	Move (programmed speed)	U32+U32	U32 speed in um/s U32 new position in um
0x101	Move (programmed time)	U32+U32	U32 time in ms U32 new position in um
0x103	Move manual	U32 + U8	U32 speed in um/s U8 direction
0x108	Stop movement (WriteOnly)	U8	Dummy byte,
0x10A	Movement (ReadOnly)	U8	Read: 0 = Not moving 1 = Moving
0x10B	Position (ReadOnly)	U32	Position in um
0x300	Clamp	U8	0 = Clamp closed, 1 = Clamp open
0x301	Vacuumpump	U8	0 = Vacuum off 1 = Vacuum on
0x302	Vacuumvalve	U8	0 = Valve off 1 = Valve on
0x303	Lifter position (ReadOnly)	U8	0 = Undefined 1 = A4 2= A3 4= Custom

HardwareStatus **Conversation ID=0x00, Mainboard AFA**

Hardwarestatus	Description
0x00 00 00 00	OK
0x00 00 00 01	Emergency button is activated
0x00 00 00 02	Device Not initialised
0x00 00 00 04	New ADD-ON detected
0x00 00 01 00	Motor Carrier is blocked
0x00 00 02 00	Motor Clamp is blocked
0x00 00 04 00	Paperlength sensor has changed position

12 AFA Heated Bed Specific Commands

Commands **Conversation ID=0x01, Heated bed option**

Enumeration (HEX value)	Description	Payload Length (bytes)	Payload Definition
0x00	Hardwarestatus (ReadOnly)	U32	U32 (See Hardware Status table)
0x300	SET Bed temperature	U16	0-250 deg C
0x301	Current probe temperature	U16	0-250 deg C

HardwareStatus **Conversation ID=0x01, Heated bed option**

Hardwarestatus	Description
0x00 00 00 00	OK
0x00 00 00 01	Sensor disconnected
0x00 00 00 02	Not Initialised
0x00 00 00 04	Overheated

13 Disclaimer

The right of technical modifications is reserved.

The information given in this manual is not intended to be exhaustive and any person using the product for any purpose other than that specifically recommended in this manual without first obtaining written confirmation from us as to the suitability of the product for the intended purpose does so at his own risk. Whilst we endeavor to ensure that all advice we give about the product (whether in this manual or otherwise) is correct we have no control over either the quality or condition of the product or the many factors affecting the use and application of the product. Therefore, unless we specifically agree in writing to do so, we do not accept any liability whatsoever or howsoever arising for the performance of the product or for any loss or damage (other than death or personal injury resulting from our negligence) arising out of the use of the product. The information contained in this manual is liable to modification from time to time in the light of experience and our policy of continuous product development.