XY3-100 Laser Scanner
Protocol Format Specification

Version 1.0 LIA202002

Table of Contents

1 Copyright

2 Document history

Date	Changes in document
09/2020	Version 1.0 released
08/2020	Copyright information updated
07/2020	Missing index values at backchannel data added
07/2020	Description of Frame Error Count Packet corrected
07/2020	Description of 32 bit frame clarified
07/2020	Frame length information corrected to 10 usec
07/2020	Initial version

3 Overview

This document describes the complete protocol of the digital XY3-100 laser scanner control interface. It includes all, the hardware layer, signal description as well as the transmitted data format.

The XY3-100 protocol is intended to be used as successor of the XY2-100 standard. Comparing to it it offers the following features:

	XY2-100	**XY3-100**
Resolution (bitrate)	16 bit (18 bit via XY2-100E)	Variable from 16 to 26 bit
Resolution (frames)	100 kHz	Variable, 100 kHz typically
Transmission rate	100 ks/sec	Variable, 100ks/sec typically
Backchannel	20 data bits synchronous to XY2-100 clock	Flexible, asynchronous RS485 serial communication protocol
Hardware	DB25 connector	DB25 connector
Hardware Compatibility		Same pinout as XY2-100(E), no hardware changes needed
Error correction	Parity bit	Parity counter on position/command data, binary protocol on backchannel

3.1 Features

The XY3-100 standard provides the following features:

- pin-compatible to XY2-100, so upgrade is possible via firmware modification
- SPI-like, so no longer requires FPGA custom designs but can be implemented with standard hardware (like MCUs) too
- copyrighted by LasIA but open standard, can be used for free in both scanheads and scanner controllers (**please note the copyright information and licensing conditions** mentioned in "1 Copyright") with minor restrictions, can be certified for full XY3-100 compliance
- variable resolution in range 16..26 bit possible
- support for 2D and 3D position data as well as two additional axes for different purposes
- enhanced asynchronous backchannel with expandable, up- and down-compatible data format
- expanded error detection for more secure position data transmission
- collision avoidance: when a XY2-100 and a XY3-100 device are connected to each other, there is no undefined behaviour or random data reception

3.1.1 XY3-100 feature subsets

The core-function of the XY3-100 protocol is the **position data transmission via SYNC, CLK, X and Y lines only** (for a detailed description please refer below). All other functions beyond that (including channels Z, U, W, the backchannel, the data submitted via the backchannel, command transmission to the scanhead and the command data submitted to the scanhead) **are optional**. When the backchannel is used, only the synchronisation packet is mandatory, all other possible data structures are optional
So neither a data source (which is typically a scanner controller card) needs to support all of these features nor a data sink (typically a scanhead) can expect to receive all of these data.

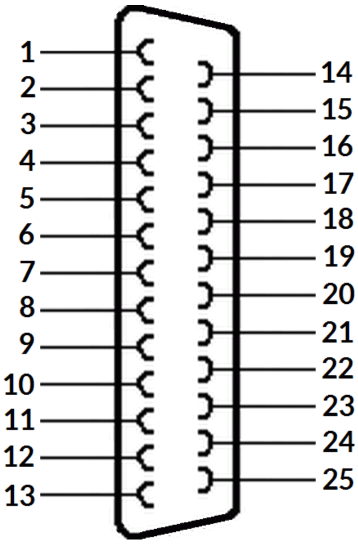
Following table gives a quick overview about the possible subsets. The subsets to be supported grow from left to right and from top to bottom, mandatory elements are marked in **bold red**:

<table>
<tr><td>2D position data
forward transmission
SYNC, CLK, X, Y</td><td>3D position data
forward transmission
Z</td><td>4-channel position data
forward transmission
U</td><td>5-channel position data
forward transmission
W</td></tr>
<tr><td colspan="4">Backchannel
BACK
Synchronisation packet (mandatory only when backchannel is used)</td></tr>
<tr><td colspan="4">Backchannel
BACK
all other data packets</td></tr>
</table>

4 Hardware interface

The XY3-100 interface makes use of DB25 connector with following pinout:

1 – SYNC- (A-)
2 – CLK- (B-)
3 – X- (C-)
4 – Y- (D-)
5 – Z- (E-)
6 – BACK- (F-)
7 – U- (G-)
8 – W- (H-)
9 – V+
10 – V+
11 – GND
12 – V-
13 – V-



14 – SYNC+ (A+)
15 – CLK+ (B+)
16 – X+ (C+)
17 – Y+ (D+)
18 – Z+ (E+)
19 – BACK+ (F+)
20 – U+ (G+)
21 – W+ (H+)
22 – V+
23 – GND
24 – GND
25 – V-

Alternatively a (preferentially white) 26-pin IDC-connector can be used

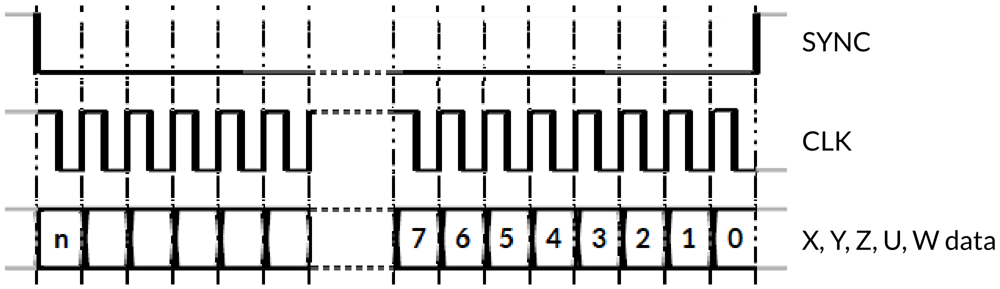
Upper Row Of Pins	Signal	Voltage	Remarks	Lower Row Of Pins	Signal	Voltage	Remarks
1	SYNC- (A-)	RS485		**2**	SYNC+ (A+)	RS485	
3	CLK- (B-)	RS485		**4**	CLK- (B+)	RS485	
5	X- (C-)	RS485		**6**	X+ (C+)	RS485	
7	Y- (D-)	RS485		**8**	Y+ (D+)	RS485	
9	Z- (E-)	RS485	optional	**10**	Z+ (E+)	RS485	optional
11	BACK- (F-)	RS485	optional back-channel	**12**	BACK+ (F+)	RS485	optional back-channel
13	U- (G-)	RS485	optional	14	U+ (G+)	RS485	optional
15	W- (H-)	RS485	optional	16	W+ (H+)	RS485	optional
17	V+			**18**	V+		
19	V+			**20**	GND	GND	
21	GND	GND		**22**	GND	GND	
23	V-			**24**	V-		
25	V-			26			available only on IDC-connector, do not connect

At least it has to support the two position axes X and Y (named as C and D in end user documents) for transmission of 2D position data and the control lines SYNC and CLK (named as A and B in end user documents).
The Z-channel (named as E in end user documents) is optional and can be used for 3D-capable hardware.
The BACK-channel (named as F in end user documents) is optional and can be used when backchannel information are supported and provided by a scanhead.
The U- and W-channels (named as G and H in end user documents) are optional and can be used for hardware which supports additional axes (e.g. for keeping a camera focus).

5 Data protocol

5.1 Forward transmission data

16..26 bit position data are transferred from scanner controller data to scanner card using lines SYNC+-, CLK+-, X+-, Y+- and optional Z+-, U+- and W+- using a fixed frame-length (10 usec typically for 100 ks/sec) but a variable frame-size regarding the amount of contained data bits:



The protocol works similar to a standard SPI interface with $\overline{CS}$ (also named $\overline{SS}$), SCK and SDI (also named MOSI) lines. Therefore reception does not necessarily require a custom FPGA but can be done in hardware of an appropriate MCU too. The beginning of a frame is marked by the falling edge at the SYNC-channel ($\overline{CS}$). Whenever data at X, Y, Z, U and W channel (SDI) are valid, this is signalled by a falling edge on the CLK-channel (SCK).

The first bit (n, 31 or 23) signalises the length of the whole frame. When it is 0, the frame has a total length of 24 bits with a payload of 22 bits. When it is 1, it has a length of 32 bits with a payload of 30 bits. The related data structures are described below.

The second bit (n-1) specifies the mode of operation. When it is set to 1, position data will follow. A value of 0 specifies a frame that contains commands that can be read and executed by the data sink (optionally) but not to be converted to position data. For a description please refer below.

Position data always end with parity bits which are the lower bits of a counter that identifies the number of position-bits set to 1 (see description below).

Structure of a 24 bit position frame:

Bit	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SYNC	LOW																							
CLK	falling edge when X/Y/Z are valid																							
Data	0	1	D19..D0 position data																				P_1	P_0

A receiver (scanhead) is free to choose which accuracy to accept depending on its own capabilities, it can use the full amount of 20 received data bits or less. In case of less, the lower, minor bits have to be ignored by the scanhead.

A sender (controller card) is free to choose which resolution to send depending on its own capabilities, it can use the full available amount of 20 bits or less. In case of less, the lower, minor bits have to be set to 0.

The P_1/P_0 – parity bits have to be calculated as follows:

- count the number of bits set to 1 in D19..D0 (bits 21..2)
- mask the counting result with 0x03 to get only the lower two bits
- write the masked result to P_1 (0x02 mask) and P_0 (0x01 mask)

In case of 100 ks/sec transmission rate the whole frame has a length of 10 usec which is equal to a clock-frequency of 2.4 MHz. Depending on the hardware capabilities on both ends, higher transmission rates are allowed too (e.g. 200 ks/sec: 5 usec frame length and clock frequency of 4,8 MHz).

Structure of a 32 bit position frame:

Bit	31	30	29	28	27	26	..	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SYNC	LOW																							
CLK	falling edge when X/Y/Z are valid																							
Data	1	1	D25..D0 position data																		P_3	P_2	P_1	P_0

A receiver (scanhead) is free to choose which accuracy to accept depending on its own capabilities, it can use the full amount of 26 received data bits or less. In case of less, the lower, minor bits have to be ignored by the scanhead.

The 26 bit accuracy not necessarily needs to be used for outputting them at the scanhead. Assumed the output device has to perform some own calculations internally and assumed a scanner card already provides data in a higher resolution, this position frame can be used to transmit the full resolution intermediate data to the scanner to avoid a loss of data because of the transport. Due to less rounding errors the scanhead then can provide better accuracy results also in case the real output resolution is smaller than 26 bits.

A data source (controller card) is free to choose which resolution to send depending on its own capabilities, it can use the full available amount of 26 bits or less. In case of less, the lower, minor bits have to be set to 0.

The P_3/P_2/P_1/P_0 – parity bits have to be calculated as follows:

- count the number of bits set to 1 in D25..D0 (bits 29..4)
- mask the counting result with 0x0F to get only the lower four bits
- write the masked result to P_3 (0x08 mask), P_2 (0x04 mask), P_1 (0x02 mask) and P_0 (0x01 mask)

In case of 100 ks/sec the whole frame has a length of 10 usec which is equal to a clock-frequency of 3.2 MHz. Depending on the hardware capabilities on both ends, higher transmission rates are allowed too (e.g. 200 ks/sec: 5 usec frame length and clock frequency of 6,4 MHz).

Structure of a 24 bit command frame:

Bit	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SYNC	LOW																							
CLK	falling edge when X/Y/Z are valid																							
Data	0	0	D19..D0 command data																				P_1	P_0

Command frames make use of the same parity feature like described for position frames above. The commands itself, which are encoded in bits D19..D0, are described below.

Structure of a 32 bit command frame:

Bit	31	30	29	28	27	26	25	24	23	..	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SYNC	LOW																							
CLK	falling edge when X/Y/Z are valid																							
Data	1	0	X	X	X	X	X	X	D19..D0 command data												P_3	P_2	P_1	P_0

To make use of exactly the same commands like for a 24 bit frame, the bits 24..29 are unused and have to be ignored. Nevertheless they are part of the parity check. It is recommended to set them to 0.
Command frames make use of the same parity feature like described for position frames above. The control commands itself, which can be encoded in bits D19..D0, are described below.

5.1.1 Control commands

Control commands have to be transmitted via the X channel. Optionally they also can be submitted via the Y, Z, U, W channels in parallel too, but a data sink is free to ignore control commands on all other channels except X.

The control commands itself all are optional, a data sink has to react on them only when it supports the related function. All other or unknown control commands can be ignored.

Following 20 bit wide control commands are defined:

`XY3_CMD_AUTOCALIB_ON` – `0x800XX` – when a data sink supports some kind of automatic adjustment or calibration function, this command can be used to turn it on. It is up to the data sink to persist the current state set with this function or fall back to a default value on restart.
Here the lower 5 bits specify for which axes this command is valid. They either can be 0 when the command has to be executed for all available axes, or they can be a combination of `XY3_CMD_AXIS_FLAG_xxx`-flags specifying the exact axes where the operation has to be performed at. For a list of `XY3_CMD_AXIS_FLAG_xxx`-flags see below.

`XY3_CMD_AUTOCALIB_OFF` – `0x40000` – when a data sink supports some kind of automatic adjustment or calibration function, this command can be used to turn it off. It is up to the data sink to persist the current state set with this function or fall back to a default value on restart. When turned off, the data sink may operate in some raw mode which no longer guarantees the desired accuracy or functionality.
Here the lower 5 bits specify for which axes this command is valid. They either can be 0 when the command has to be executed for all available axes, or they can be a combination of `XY3_CMD_AXIS_FLAG_xxx`-flags specifying the exact axes where the operation has to be performed at. For a list of `XY3_CMD_AXIS_FLAG_xxx`-flags see below.

`XY3_CMD_CALIB_START` – `0xC0000` – when a data sink supports some kind of calibration or adjustment function which is <u>not</u> done automatically but needs to be triggered actively, this command invokes such a calibration operation.
Here the lower 5 bits specify for which axes this command is valid. They either can be 0 when the command has to be executed for all available axes, or they can be a combination of `XY3_CMD_AXIS_FLAG_xxx`-flags specifying the exact axes where the operation has to be performed at. For a list of `XY3_CMD_AXIS_FLAG_xxx`-flags see below.

`XY3_CMD_REF_START` – `0x20000` – when a data sink supports or requires some kind of referencing which is <u>not</u> done automatically but needs to be triggered actively, this command invokes such a reference run.
Here the lower 5 bits specify for which axes this command is valid. They either can be 0 when the command has to be executed for all available axes, or they can be a combination of `XY3_CMD_AXIS_FLAG_xxx`-flags specifying the exact axes where the operation has to be performed at. For a list of `XY3_CMD_AXIS_FLAG_xxx`-flags see below.

`XY3_CMD_BACK_RATE57` – `0xA0000` – when the backchannel is used, this command changes its data transmission rate to 57600 bit/s

`XY3_CMD_BACK_RATE115` – `0xE0000` – when the backchannel is used, this command changes its data transmission rate back to the default value of 115200 bit/s

`XY3_CMD_BACK_RATE230` – `0x10000` – when the backchannel is used, this command changes its data transmission rate to 230400 bit/s

`XY3_CMD_BACK_RATE460` – `0x90000` – when the backchannel is used, this command changes its data transmission rate to 460800 bit/s

`XY3_CMD_BACK_RATE912` – `0xD0000` – when the backchannel is used, this command changes its data transmission rate to 921600 bit/s

`XY3_CMD_TEMPCOMP_ON` – `0x880XX` – when a data sink supports some kind of automatic temperature compensation function, this command can be used to turn it on. It is up to the data sink to persist the current state set with this function or fall back to a default value on restart.
Here the lower 5 bits specify for which axes this command is valid. They either can be 0 when the command has

to be executed for all available axes, or they can be a combination of `XY3_CMD_AXIS_FLAG_xxx`-flags specifying the exact axes where the operation has to be performed at. For a list of `XY3_CMD_AXIS_FLAG_xxx`-flags see below.

`XY3_CMD_AUTOCALIB_OFF` – `0x48000` – when a data sink supports some kind of automatic temperature compensation function, this command can be used to turn it off. It is up to the data sink to persist the current state set with this function or fall back to a default value on restart. When turned off, the data sink may operate in some raw mode which no longer guarantees the desired accuracy or functionality.
Here the lower 5 bits specify for which axes this command is valid. They either can be 0 when the command has to be executed for all available axes, or they can be a combination of `XY3_CMD_AXIS_FLAG_xxx`-flags specifying the exact axes where the operation has to be performed at. For a list of `XY3_CMD_AXIS_FLAG_xxx`-flags see below.

Following axis flags exist which can be OR-concatenated with commands that support selection of a specific axis:
`XY3_CMD_AXIS_FLAG_X` – `0x00001` – flag for axis X
`XY3_CMD_AXIS_FLAG_Y` – `0x00002` – flag for axis Y
`XY3_CMD_AXIS_FLAG_Z` – `0x00004` – flag for axis Z
`XY3_CMD_AXIS_FLAG_U` – `0x00008` – flag for axis U
`XY3_CMD_AXIS_FLAG_W` – `0x00010` – flag for axis W

5.2 Backchannel data

The backchannel via the BACK–lines (F+/F+) is completely independent from the CLK line of the position data channels. It is an RS485 serial interface with a default transmission rate of 115200 bps, 8 data bits, 1 stop bit and no parity (can be changed via control commands optionally). Data packets transmitted via this interface always have a fixed structure:

Head	Type	Length	Payload
8 bit value, always set to 0x48	8 bit value, specifies type of data packet	8 bit value, specifies length of following data	Payload with the length specified in previous byte

To synchronise with an interrupted or already running transmission, a receiver has to wait for a sync-packet:

Head	Type	Length	Payload
0x48	0x41	0	none

So when the receiver finds a sequence 0x48 0x41 0x00 0x48 within a data stream, it can assume it is back in sync with the data (this sequence is a SYNC-packet followed by the head of the next packet). When a receiver does not find the 0x48 packet at the end of a previous one, or when a known packet with an invalid size arrives, it has to assume it is out of sync. In this case all received data have to be dropped until it is back in sync via the above procedure.
On the other hand, a sender from time to time should send such a sync-packet in order to give receivers the chance to resynchronise properly.

All data types, structures and constants are defined within the header file xy3_100.h which is provided together with this specification. The base packet structure is:

```
struct xy3_backframe
{
   unsigned char sync;  // has always to be set to XY3_SYNC_IDENTIFIER
   unsigned char mtype; // packet type XY3_TYPE_xxx
   union
   {
      …; // packet-specific sub-structures, for a description refer below
   } d;
};
```

Available packet types are:

Synchronisation Packet:

Head	Type	Length	Payload
0x48	0x41	0	none

The type identifier is `XY3_TYPE_SYNC`, there is no separate structure for this packet type.

Vendor Identifier Packet:

Head	Type	Length	Payload
0x48	0x01	3..200 bytes	Name of the manufacturer of the scanhead in 7 bit ASCII

The type identifier is `XY3_TYPE_VENDOR`, the related structure is `struct xy3_generic_text`.

Model Identifier Packet:

Head	Type	Length	Payload
0x48	0x02	3..200 bytes	Type/model of the scanhead in 7 bit ASCII

The type identifier is `XY3_TYPE_MODEL`, the related structure is `struct xy3_generic_text`.

Firmware Version String Packet:

Head	Type	Length	Payload
0x48	0x03	3..200 bytes	Firmware version of the scanhead in 7 bit ASCII

The type identifier is `XY3_TYPE_MODEL`, the related structure is `struct xy3_generic_text`.

Serial Number String Packet:

Head	Type	Length	Payload
0x48	0x04	3..200 bytes	Serial number string of the scanhead in 7 bit ASCII

The type identifier is `XY3_TYPE_SN`, the related structure is `struct xy3_generic_text`.

Temperature Packet:

Head	Type	Length	Payload
0x48	0x05	2..44 bytes	Array of 16 bit signed integers specifying the temperature at a specific point in scanhead

Here the 16 bit values specify a temperature in unit 1/100 degrees Celsius each. When one of the values is not supported, the sender has set it to -32767 (equal to a temperature of -327,67 °C). When some of the last

temperatures in the index list below are not supported, they can be shortened by dropping these values and submitting shorter packets.
Following temperature values are supported:

Index	Name	Description
0	Head	General head temperature inside the housing
1	DSP	Temperature at the central, general DSP controlling the scanhead
2	DAC X	Temperature at the DAC of the X-channel
3	DAC Y	Temperature at the DAC of the Y-channel
4	DAC Z	Temperature at the DAC of the Z-channel
5	DAC U	Temperature at the DAC of the U-channel
6	DAC W	Temperature at the DAC of the W-channel
7	Driver X	Temperature at the driver of the X-galvo
8	Driver Y	Temperature at the driver of the Y-galvo
9	Driver Z	Temperature at the driver of the Z-galvo/actuator
10	Driver U	Temperature at the driver of the U-galvo/actuator
11	Driver W	Temperature at the driver of the W-galvo/actuator
12	Galvo X	Temperature of the X-galvo
13	Galvo Y	Temperature of the Y-galvo
14	Galvo Z	Temperature of the Z-galvo/actuator
15	Galvo U	Temperature of the U-galvo/actuator
16	Galvo W	Temperature of the W-galvo/actuator
17	Mirror X	Temperature at the optics for the X-channel
18	Mirror Y	Temperature at the optics for the Y-channel
19	Mirror Z	Temperature at the focus optics
20	Mirror U	Temperature at the U optics
21	Mirrror W	Temperature at the W optics

The type identifier is `XY3_TYPE_TEMP`, the related structure is `struct xy3_temperatures` and the array indices are defined in `enum components`.

Frame Error Count Packet:

Head	Type	Length	Payload
0x48	0x06	20 bytes	5x 32 bit unsigned integers giving the absolute number of erroneous frames (parity error) received at X, Y, Z, U and W since last power-up

The type identifier is `XY3_TYPE_FRAMEERRCNT`, the related structure is `struct xy3_error_count`.

Error State Packet:

Head	Type	Length	Payload
0x48	0x07	1..22 bytes	Array of 8 bit unsigned integers returning an error code for a specific part of the scanhead

Here the 8 bit values specify if the related component is somehow in error state or not. A 0 means „no error“ while a value >0 signals an error. The error codes itself can be one of:

- 0 – no error, related component works as expected
- 1 – temperature error
- 2 – data error
- 3 – out of range error
- 4 – power supply error
- 5 – other electrical error
- 6 – adjustment error
- 7 – other mechanical error
- 100..255 – vendor specific error codes

Following error array indices are known:

Index	Name	Description
0	Head	General head error code
1	DSP	Central, general DSP error code
2	DAC X	Error code for the DAC of the X-channel
3	DAC Y	Error code for the DAC of the Y-channel
4	DAC Z	Error code for the DAC of the Z-channel
5	DAC U	Error code for the DAC of the U-channel
6	DAC W	Error code for the DAC of the W-channel
7	Driver X	Error code for the driver of the X-galvo
8	Driver Y	Error code for the driver of the Y-galvo
9	Driver Z	Error code for the driver of the Z-galvo
10	Driver U	Error code for the driver of the U-galvo/actuator
11	Driver W	Error code for the driver of the W-galvo/actuator
12	Galvo X	Error code for the X-galvo
13	Galvo Y	Error code for the Y-galvo
14	Galvo Z	Error code for the Z-galvo
15	Galvo U	Error code for the U-galvo/actuator
16	Galvo W	Error code for the W-galvo/actuator
17	Mirror X	Error code for the optics of the X-channel
18	Mirror Y	Error code for the optics of the Y-channel
19	Mirror Z	Error code for the focus optics
20	Mirror U	Error code for the U optics
21	Mirror W	Error code for the W optics

The type identifier is `XY3_TYPE_ERRORSTATE`, the related structure is `struct xy3_error_codes` and the array indices are defined in `enum components`.

Debug Packet:

Head	Type	Length	Payload
0x48	0x08	3..200 bytes	Free to use field for debugging data, should not be used by any connected scanner card for normal operation purposes but during development only.

The type identifier is `XY3_TYPE_DEBUG`, the related structure is `struct xy3_generic_text`.

Working Hours Packet:

Head	Type	Length	Payload
0x48	0x09	4..88 bytes	Array of 32 bit unsigned integers specifying the total amount of working hours for the related component of the scanhead

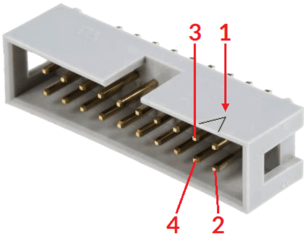
Here the 32 bit values specify a temperature in unit hours. When one of the values is not supported, the sender has set it to 0xFFFFFFFF. When some of the last fields in the index list below are not supported, they can be shortened by dropping these values and submitting shorter packets.
As some of the components neither can't collect working hours for their own nor can be watched by the logic of the scanhead, they either have to be ignored or they have to be counted "blindly". In second case a procedure has to be established during maintenance in order to reset these implicitly counted working hours on replacement.
For following components working hour information values are supported:

Index	Name	Description
0	Head	General head working hours (to be used when no more detailed information are available)
1	DSP	Working hours of the central, general DSP controlling the scanhead
2	DAC X	Working hours of the DAC of the X-channel
3	DAC Y	Working hours of the DAC of the Y-channel
4	DAC Z	Working hours of the DAC of the Z-channel
5	DAC U	Working hours of the DAC of the U-channel
6	DAC W	Working hours of the DAC of the W-channel
7	Driver X	Working hours of the driver of the X-galvo
8	Driver Y	Working hours of the driver of the Y-galvo
9	Driver Z	Working hours of the driver of the Z-galvo
10	Driver U	Working hours of the driver of the U-galvo/actuator
11	Driver W	Working hours of the driver of the W-galvo/actuator
12	Galvo X	Working hours of of the X-galvo
13	Galvo Y	Working hours of of the Y-galvo
14	Galvo Z	Working hours of of the Z-galvo
15	Galvo U	Working hours of of the U-galvo/actuator
16	Galvo W	Working hours of of the W-galvo/actuator
17	Mirror X	Working hours of the optics for the X-channel
18	Mirror Y	Working hours of the optics for the Y-channel
19	Mirror Z	Working hours of the focus optics
20	Mirror U	Working hours of the U optics
21	Mirror W	Working hours of the W optics

The type identifier is `XY3_TYPE_WORKHOUR`, the related structure is `struct xy3_workinghours` and the array indices are defined in `enum components`.

APPENDIX A – IDC-connector pin numbering

Pin numbering of the IDC-connectors (according to pinout-tables shown in hardware description section above) can be seen in below image:



The first pin is marked by a small arrow in connector. Second pin is below of it, counting continues column-wise.

Index

W

X