Optical Sensor CHRocodile 2

Non-Contact Measurement for Distance and Layer Thickness

Protocol and Command Reference

Firmware Version
R1.5.2

© Precitec Optronik GmbH

Version 01/2025 • EN • Original Edition

This documentation is under the copyright of Precitec Optronik GmbH.

It may not be reproduced or used in a manner contrary to the company's legal interests without prior written approval of Precitec Optronik GmbH. It is strictly intended for use in the context of service operations. Any other use is impermissible. Any sharing of this documentation with third parties requires the prior, expressed written approval of Precitec Optronik GmbH.

EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.

Changes in the technical details from the descriptions, data and figures in this documentation are reserved.

Produced in the Federal Republic of Germany.

Original Edition

Responsible for contents:

Precitec Optronik GmbH
Schleussnerstraße 54
D – 63263 Neu-Isenburg / Deutschland

Telephone:	0049 (0)6102 / 36 76 - 100
Telefax:	0049 (0)6102 / 36 76 - 126
e-mail:	info@precitec-optronik.de
Website:	www.precitec.com

Contents

Chapter 1

CHRocodile Protocols

1.1 Introduction

Overview

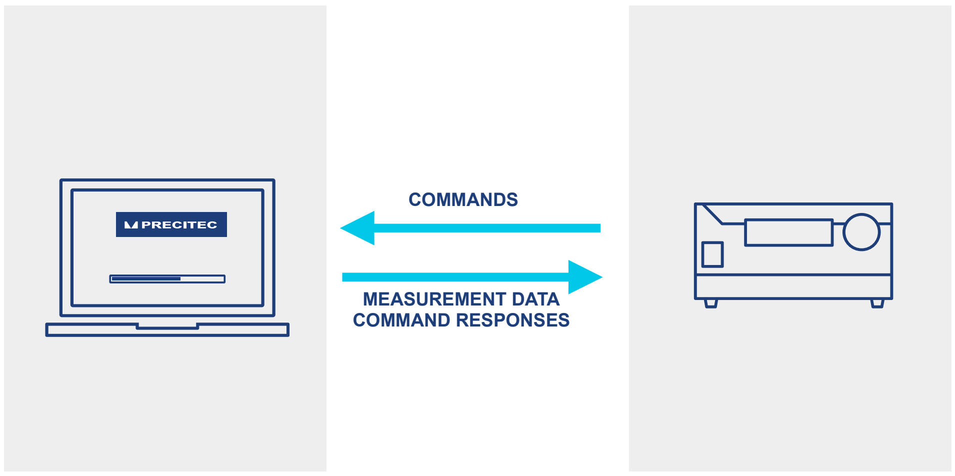
The following chapter delineates the transfer of measurement data via the device's interfaces and the configuration of the device with the corresponding commands. The two underlying protocols are described, the dollar protocol and the packet protocol.

Characteristics

During normal operation, the device continuously sends data packets. There are two unique protocols for communication between the sensor and external clients, typically being PCs. One is the dollar protocol, which is also used by CHRocodile sensors of the first generation, and the other is the binary packet protocol. The following table compares the main characteristics of the two types of data transmission:

Dollar Protocol	Packet Protocol
• Available on TCP port 7890 and serial port • Economic result data transmission (2 bytes per measured value plus 2 bytes telegram header) • Human readable command format, enabling setup and parameter adjustment with a simple terminal program. • ASCII measurement output option, making the data output human readable on a simple terminal program. • Real-time data output over serial port	• Available only on TCP port 7891 • Multi-client enabled • Multi-channel enabled (only on multi-channel devices) • Simple packet structure with reasonable packet overhead • Data packets, data format packets, command packets • Easily decodable • Easily extendable

Protocols similarities In both protocols, commands are streamed up to the device and measurement data and command responses are streamed down to clients. The information (e.g., distance, thickness, intensity, etc.) included in the data stream can be chosen by the user with a command (SODX). The command responses are sent interleaved with data samples. These relationships are outlined in the following figure:



Protocols differences Basically, the commands and the measured information available are identical on both protocols. However, there are some general differences and a few differences on individual commands. In the command reference, existing differences are mentioned for every affected command. For example, the command ULFW is supported by the packet protocol only.

Connection limitations The maximum number of supported simultaneous connections is device specific.

Multi-channel devices Devices such as the CHRocodile MPS, CLS or 2 DPS do not fully support the dollar protocol. Only the binary packet protocol is available for data transfer with those devices. However, the dollar protocol can be used for commands and parameter queries.

1.2 Dollar protocol

Overview

The dollar protocol got its name from the fact that every command begins with a dollar character. The $ character switches the CHRocodile into command receive mode.

In the dollar protocol, the format of the measurement data can be binary for optimal transmission speed ($BIN) or ASCII for easy readability ($ASC).

Notice: On multi-channel devices (such as the CHRocodile MPS, CLS or 2 DPS), the dollar protocol is not fully supported: Only global signals (start time, encoder values, sample counter, CALC 0 etc.) can be received, but not peak signals (distance, thickness, intensity etc.).

Interface

The dollar protocol is available on the Ethernet (TCP port 7890) and serial (RS422) interface.

Notice: If the dollar protocol is used with the serial interface, care has to be taken when choosing a combination of baud rate, sample rate, average, data format and selected output data that all data can be transmitted as the baudrate of the serial interface is far lower than that of the Ethernet interface.

ASCII mode

In ASCII mode, the selected signals are transmitted as decimal number strings, separated by space. Each line starts with a space. A data telegram is concluded with a CR/LF (#13#10) character combination.

Binary mode

In binary mode a data telegram begins with a 2-byte synchronization sequence. The default sequence is #255,#255 (0xFF, 0xFF) but it can be customized by the $SSQ command. After the synchronization sequence the selected signals are sent either as 2-byte or 4-byte values, according to the selected format.

The endianess of the values depends on the signal bit width according to the table below (MSB: most significant byte, LSB: least significant byte):

Bit width	Endianess	Byte order
16 bit	Big endian	MSB, LSB
32 bit	Little endian	LSB, MSB

Example:
The command SODX 16640 (16-bit integer) creates a data stream with the following structure (big endian):
`0xFF,0xFF,PeakPos1 MSB,PeakPos1 LSB`

The command SODX 65 (32-bit integer) creates a data stream with the following structure (little endian):
`0xFF,0xFF,EncoderX LSB ...EncoderX MSB`

The length of a telegram can be calculated from the size of the selected data signals plus 2 bytes (synchronization sequence). As the synchronization sequence is not unique in the data flow (there might be signal values that equal the synchronization characters), the data client must apply a special technique in order to achieve (and maintain) telegram synchronization:

1. Stop the data flow by sending a command. After completion of the command the sequence ready [CR/LF] will be sent and the CHRocodile starts with a new telegram.
2. When the client does not receive the synchronization sequence at the expected place, synchronization is lost. In this (and only in this) case, it should

wait for a new synchronization sequence. This will resynchronize the data flow in a few telegram cycles since the transmitted data words are usually changing and only the synchronization sequence is stable.

Command format `$COMMAND <arg0> <arg1> ... <argn>[CR]`

Rules Observe the following rules:

- Every command begins with “$” and is followed by at least three capital characters.
- The arguments have to be separated by a non-numeric character (preferably a whitespace, don’t use a comma as some parameters are in floating point format and the comma would be mistaken as decimal point).
- Most commands accept a question mark (“?”) as argument and then send back the current parameter setting as response.
- Commands which expect parameter values must be finished with a carriage return (#13) which is also echoed.
- Both in binary mode and in ASCII mode: All command argument and response values are in ASCII.

In the dollar protocol, every command must be preceded by a “$”-character and terminates with a carriage return (*CR*,\r, #13). At the reception of a “$”-character, the CHRocodile stops the sending of data telegrams at the next telegram boundary and the $ is echoed back. Between the “$”-character and the concluding *CR* all characters received by the device are immediately echoed back. Outside this command reception phase, no characters are echoed.
A command is composed of the leading 3 or 4 letter command name followed by a specific number of parameters. Commands may have optional parameters, which means that the last parameter(s) can be left away. Parameters must be separated by one (or more) space (#32) characters. Numerical parameters are given in human readable ASCII format, float values use a dot (.) as decimal separator.
Example: `$AAL 1 100.5`*[CR]*
When the device has completely received the command, the command is interpreted and a response is given. In the dollar protocol, the response always terminates with the string `ready\r\n`. After this termination, the sending of data telegrams is resumed.
Besides the concluding `ready\r\n`, most commands don’t have an additional dedicated response in the dollar protocol (there are exceptions, see the detailed command description). However, if the command is a query, there is a response that carries the queried information. Numerical values in responses are output in human readable ASCII format, float numbers use a dot (.) as decimal separator. If there is more than one response parameter, parameters are separated by a space (#32) character. The response always terminates with the string `ready\r\n`.

1.3 Binary packet protocol

Overview	This topic describes characteristics of the packet protocol and how communication between the CHRocodile device and the local network is carried out.
Interface	The packet protocol is available on the Ethernet interface. TCP port 7891 is used.
Byte order	The fields defined in the binary packet protocol are arranged in little endian order, i.e., the least significant byte is followed by more significant bytes.

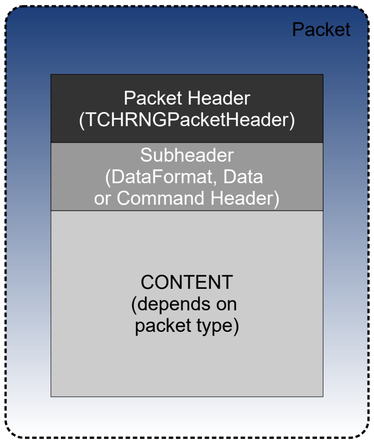
Packet types

The packet protocol currently has three different types of packets:

- Data packet
- Data format packet
- Command packet

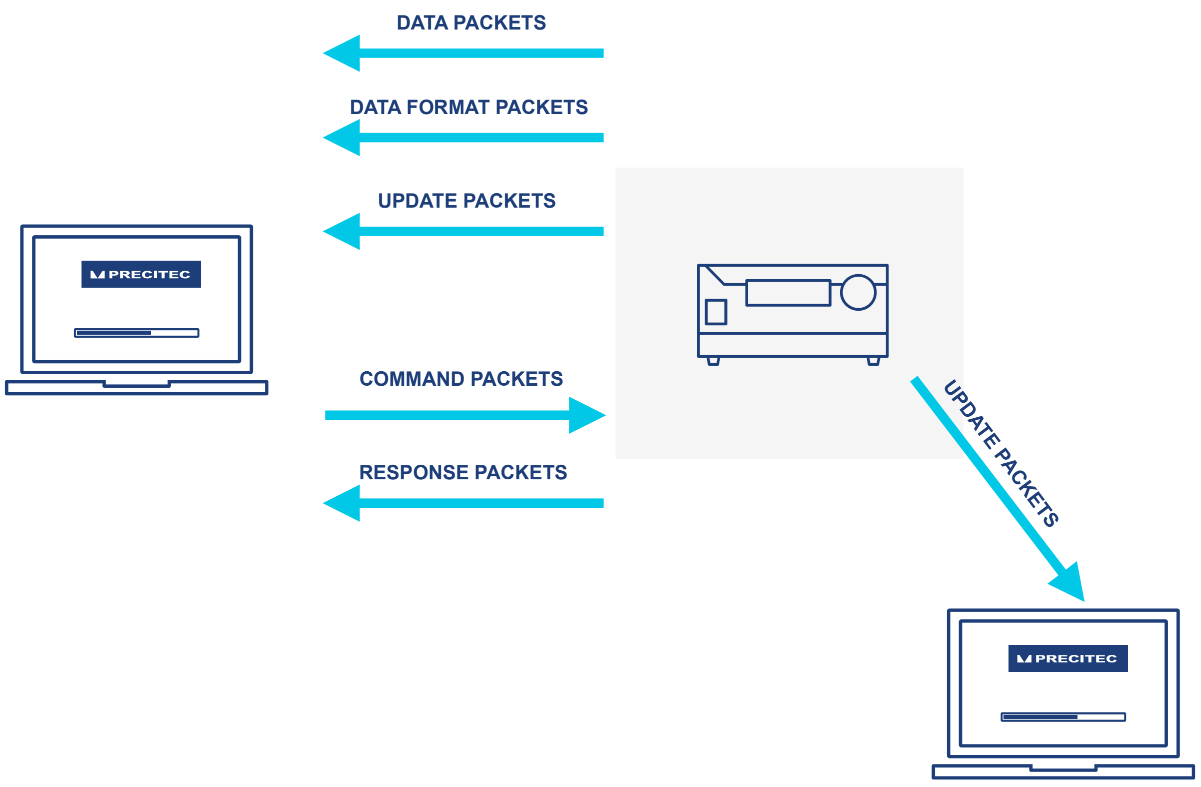
General structure

All data packets (command packets, data format packets and data packets) share the same general structure as shown in the following figure. C-like declarations of the related structures are given in the following sections.


Packet
Packet Header (TCHRNGPacketHeader)
Subheader (DataFormat, Data or Command Header)
CONTENT (depends on packet type)

Illustration

The following illustration describes how the different types of packets support communication between client and device:



Type	Function
Data packet	Contains signals (e.g., distance values, intensity, etc.)
Data format packet	Contains information about the structure of data packets, e.g., which data signals are included in which order, what type they have, how many bytes do they occupy, etc.
Command packet	Contains commands and associated arguments.
Response packet	Response of the device upon a command, can contain various flags (error, warning, etc.) indicating anomalies in the command processing. Structurally identical to command packets.
Update packet	Informs other clients about an internal state change of the device as result of a command. Structurally identical to command packets.

1.3.1 Packet header

Overview

All types of packets share a common main header at the beginning of the packet.

Packet header fields

The packet header contains:

- Magic Number (4 bytes): constant 0xAA55AA55, marks the beginning of the packet
- Packet Length (4 bytes, integer): length of a complete packet including its header, which is limited to 4096 bytes.
- Reserved (8 bytes): currently ignored, should be set to zero
- Packet Type (4 bytes): The type of the packet. There are currently 3 types:
  - 0x00444D43 (ASCII "CMD") for command packet, including response and update packets.
  - 0x00544644 (ASCII "DFT") for data format packet.
  - 0x00544144 (ASCII "DAT") for data packet.

After the packet header, the packet data section follows. It contains the packet type specific subheader and the content.

Source code

In the following there is a listing of the structure and constants used in the packet header in the C language.

```

const u32 cMagicNumber = 0xAA55AA55;

enum class TCHRNGPacketType : u32 {
    CommandTelegram      = 0x00444D43, //''CMD''
    DataTelegram         = 0x00544144, //''DAT''
    DataFormatTelegram   = 0x00544644, //''DFT''
};

struct TCHRNGPacketHeader {
    u32 MagicNumber;
    s32 PacketLength;
    s32 Reserved1;
    u32 Reserved2;
    TCHRNGPacketType TelegramType;
};
```

1.3.2 Command packet definition

Overview

Command packets contain commands and associated arguments. They consist of the packet header and a command subheader, possibly followed by arguments. Commands are defined in terms of a 32 bit ID while parameters carry their type and their value. Supported types are (32 bit) integer, (single precision) float, string, char and blob (binary large object). The number of parameters that can be added to the command is merely limited by the maximum total size of a packet (currently 4096 bytes). Sending a command to the device from the client application basically means to send a command packet and wait for the echo / response to that packet. The command response packet has the same structure as the original command packet and usually contains all incoming arguments (possibly clipped or otherwise corrected) plus any additional parameters that might have been queried for.

Packet subheader

The command packet subheader includes:

- **Command** (4 bytes): three to four ASCII letters such as SODX
- **Destination filter ID** (4 bytes): will be reflected to client in the response packet.
- **Source filter ID** (4 bytes): will be reflected to client in the response packet.
- **Flags** (2 bytes). This field may contain a bitwise OR combination of any of the flags defined below:
  - cCmdFlagQuery = 0x0001
    This flag declares a parameter query. The sensor will respond with a command packet containing the current parameter value.
  - cCmdFlagError = 0x8000
    This flag is returned by the sensor in the command response in case of an error, i.e., if for any reason the command has not been executed. Additional information may be added as arguments.
  - cCmdFlagWarning = 0x4000
    Returned in the response packet in case the command has been executed, however, with possibly modified parametrization or other implications that the user has not expected. Additional information may be added as arguments.
  - cCmdFlagUpdate = 0x2000
    The device may send *"updates"*, i.e., command responses to the connected client whenever the current configuration / parametrization changes, e.g., due to some user action at the front panel or through another connection. The client will receive such packets *without* prior command packets. In such cases, the update flag will be set.
- **Reserved** (2 bytes)
- **Ticket** (2 bytes): The client attributes an arbitrary ticket number to a command packet. The response packet for the contained command will have the same ticket number. Thereby a command response packet can be unambiguously related to its originating command packet.
- **Arguments count** (2 bytes, integer): number of command arguments added behind this subheader

Arguments section

The command packet subheader is followed by the arguments section. Every argument starts with a type specifier defined as:

Type specifier	Type of command argument
0	integer (32 bit)
1	float (single precision)
2	string
3	char
4	blob (chunk of untyped, binary data)

The type specifier is followed by data formatted in a type-specific way.

For int, float and char parameters, the data directly follows the type specifier:

```
<Type Specifier(4 bytes)><Value(4 bytes)>
```

String and blob parameters do not have fixed length. Therefore, the length of the data is given in the first four bytes after the type specifier:

```
<Type Specifier><Str/Blob Length(4 bytes)><String/Blob Value(ASCII)>
```

For alignment reasons, the string or blob parameters must contain multiples of 4 bytes. If a string has a length of, say, nine, then three zeros must be added to the end to fill twelve bytes which are divisible by four. The real string/blob length is, as said, stored in the length field.

Multiple arguments may be concatenated to form the argument part of the command packet.

Response packet

Once the command packet has been received and processed completely, the device will respond with a command response packet. This response packet has the same format as the command packet and acts as a receipt for the command. Please note that:

- The response packet may not only contain the arguments set by the client, but also other parameters that reflect the state of the device.
- The ticket of the response packet is the same as the ticket of the command packet.

Update packet

Beside responding to the client that sent a command, the resulting internal change(s) will be communicated to all other connected clients using update packets. The only differences between those and the response packet are their update flags and their ticket number. While the response packet contains the original ticket number assigned by the issuing client and an unset update flag, the update packets will have their ticket number set to zero and their update flags set.

Source code

The types and constants involved in command packets are listed below:

```

struct TCHRCmdHeader {
    TCHRCmd ID; // Three to four ASCII letters such as "SODX"
    u32 DestFilterID;
    u32 SourceFilterID;
    u32 Flags: 16;
    u32 Reserved: 16; //ignored, should be set to 0
    u32 Ticket: 16; // arbitrary number, repeated in the response
    u32 ArgsCount: 16; // The number of following arguments
};

struct TIntParam { // An integer argument
    TParamType ParamType;
    s32 Value;
};
```

```

struct TFloatParam { // A floating point argument
    TParamType ParamType;
    float Value;
};

struct TStringParam { // A string argument
    TParamType ParamType;
    u32 Length;
    char[n] string_chars; // length n as given in length field
    char[m] zeros; // additional zeros to fill multiple of four bytes
};

// A blob argument has the same structure as a string argument.
// A char argument is formatted like an integer argument.
```

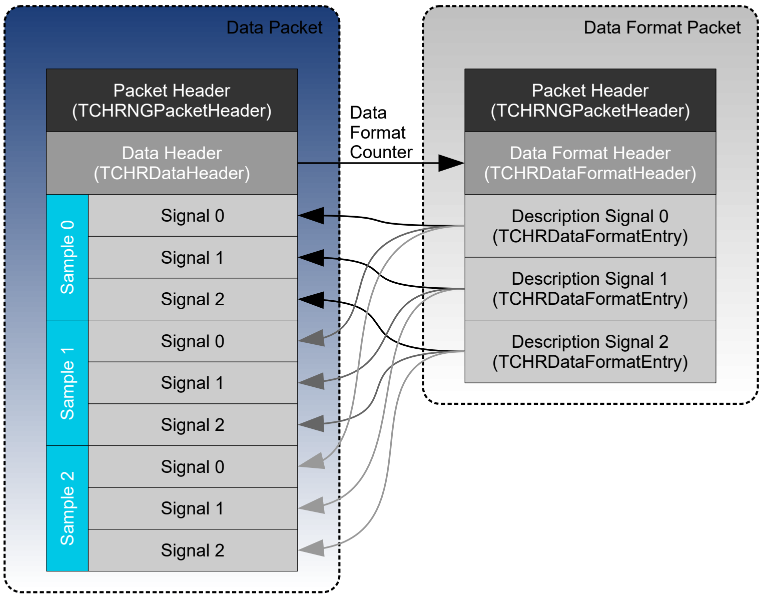
1.3.3 Data format packet definition

Overview

Data format packets contain information about the structure of data packets, e.g., which data signals are sent from the device to the client. For details about data packets please refer to next section. The signals definition contained in a data format packet is valid for all the following data packets until a new data format packet arrives.

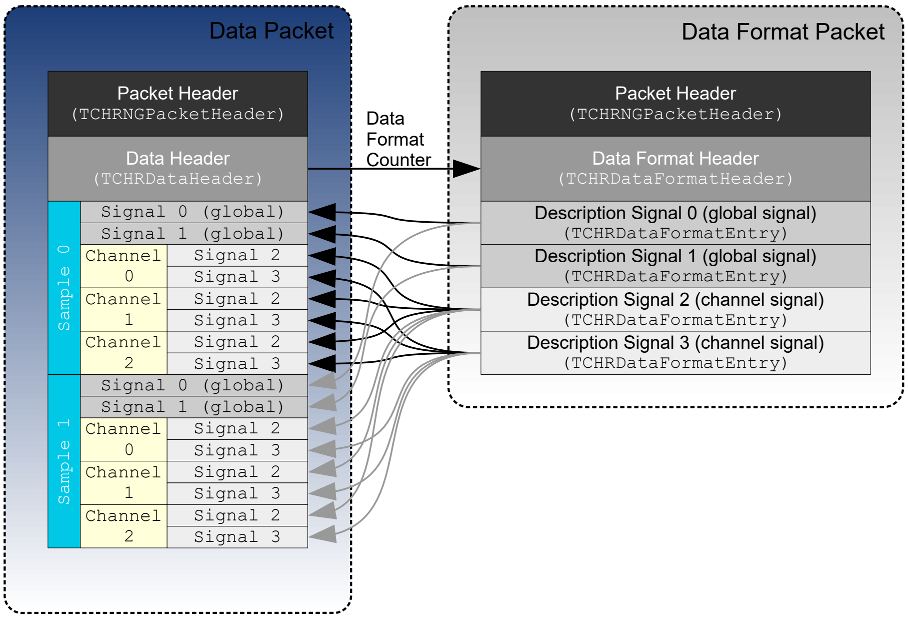
Single channel device

The relationship between a data format packet and a data packet is described in the following figure. As illustrated, a data packet contains one or more *samples*, each being a group of *signal values*:



Data Packet
Packet Header (TCHRNGPacketHeader)
Data Header (TCHRDataHeader)
Sample 0
Sample 1
Sample 2
Signal 0
Signal 1
Signal 2
Data Format Counter
Data Format Packet
Packet Header (TCHRNGPacketHeader)
Data Format Header (TCHRDataFormatHeader)
Description Signal 0 (TCHRDataFormatEntry)
Description Signal 1 (TCHRDataFormatEntry)
Description Signal 2 (TCHRDataFormatEntry)

Multi-channel device The relationship between a data format packet and a data packet in a multi-channel device. Note that global signals are placed at the beginning, channel signals are placed behind:


Data Packet
Packet Header (TCHRNGPacketHeader)
Data Header (TCHRDataHeader)
Sample 0
Signal 0 (global)
Signal 1 (global)
Channel 0
Signal 2
Signal 3
Channel 1
Channel 2
Sample 1
Data Format Counter
Data Format Packet
Packet Header (TCHRNGPacketHeader)
Data Format Header (TCHRDataFormatHeader)
Description Signal 0 (global signal) (TCHRDataFormatEntry)
Description Signal 1 (global signal) (TCHRDataFormatEntry)
Description Signal 2 (channel signal) (TCHRDataFormatEntry)
Description Signal 3 (channel signal) (TCHRDataFormatEntry)

Sample and signal A sample is a collection of signal values simultaneously measured during one detector exposure.

A signal can be, e.g., the distance to the surface measured, some encoder value as latched during exposure, a time stamp or a peak intensity – whatever has been requested by the user by means of an SODX command. A complete list of available signals is given in chapter 3 CHRocodile Signal IDs.

We distinguish between global, channel signals (or measuring point related signals) and peak related signals. Global signals, like encoder positions or exposure time stamps are common for the whole sample, whereas the channel signals are individual for each channel of a multi-channel device. However, the set of transmitted channel signals is the same for all channels. The peak related signals represent different aspects of a measured peak, e.g., its value, its position in the spectrum, its median, etc.

Data packets contain this data with only a minimum of declarative overhead. Instead, all information is given in the data format packets which have to be sent only once by the sensor whenever the signal arrangement has been changed via the SODX command.

Packet subheader The data format subheader consists of the following information:

- The data stream ID (4 bytes, integer), always set to one
- The data format counter (4 bytes, integer), which is incremented on any new data format packet. Each data packet contains the same data format counter value as the data format packet that describes it.
- The sample rate of the stream (samples per second, 4 bytes, float)

- The data signals count (4 bytes, integer), which declares how many *different* signals are included in a sample. One sample consists of one set of global signals (transmitted first) and then n (= number of channels) sets of channel signals. One data packet can contain multiple samples.

  In the example of the figure in Section 1.3.3 for multi-channel devices, the signals count is 4: 2 global signals and 2 channel signals. On the left of the data format packet we see a data packet that is built according to this data format packet. It contains 2 samples. Each sample has 3 channels with 2 signals each. The signal sets of all channels are the same (here signal 2 and signal 3).

Signal description

The section after the data format subheader contains descriptions of every signal present in the data packets:

- The data type (1 byte), e.g., "integer" or "single precision float"
- Reserved (1 byte) – ignore this field
- The number of channels (2 bytes, integer) ("points") in case of a multi-channel device, 1 otherwise
- The number of the first channel to be transferred (2 bytes, integer). Is zero unless otherwise configured.
- The signal ID (2 bytes, integer)

Requesting data

When the client is connected to the device, by default the device will not send any data to the client. The client has to first order data by sending an SODX command to the device. Such an SODX command contains a list of IDs of those signals requested by the client. Provided the requested signals are valid and available, the SODX command will be responded by an SODX response packet – followed by a data format packet based on the SODX command packet sent by the client. Note, however, that the binary packet protocol does not guarantee the order of signals to be the same as given in the SODX command. This guarantee is only given in the dollar protocol.

Signal sorting

In a data format packet and its corresponding data packet, all *g*lobal signals (i.e., signals which appear only once per exposure for both single channel and multi-channel devices, such as exposure start time, exposure period and encoder values) are always placed at the beginning. The so-called "peak signals" (signals that are related to specific measurement peaks, e.g., "Distance 1 / 2 / 3" or "Intensity 1 / 2 / 3") are placed behind the sample global signals. See the next section for the details of the arrangement of the data blocks.

Source code

```
struct TCHRDataFormatHeader {
    u32 DataStreamID;
    s32 DataFormatCounter;
    float SampleRate;
    s32 SignalsCount;
};
```

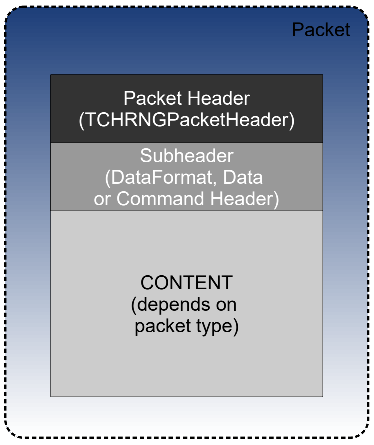
For each signal, one entry of type `TCHRDataFormatEntry` is added to the data format packet.

```
typedef enum {
    sitU8 = 0, //byte
    sitS8,     //signed char
    sitU16,    //unsigned 16bit
    sitS16,    //signed 16bit
    sitU32,    //unsigned 32bit
    sitS32,    //signed 32bit
    sitFloat   //single precision float
} TSignalType;

// Attention: the following definition uses bitfields
// in order to create a packed record
struct TCHRDataFormatEntry {
    TSignalType DataType: 8;
    u32 Reserve: 8;
    u32 PointCount: 16;
    u32 FirstPoint: 16;
    u32 SignalID: 16;
};
```

1.3.4 Data packet definition

Overview

As shown in the following figure, data packets consist of the general packet header, a data packet subheader, and the content (i.e., the data):


Packet
Packet Header (TCHRNGPacketHeader)
Subheader (DataFormat, Data or Command Header)
CONTENT (depends on packet type)

Packet subheader

The data packet subheader defines:

- Data stream ID (4 bytes, integer), always 0x1
- Data format counter (4 bytes, integer), so that the corresponding data format packet can be associated.
- Time stamp (8 bytes), the time stamp of the first sample inside the packet. It has the 32.32 bit fixed point format, where the most significant 32 bits represent the full seconds and the least significant 32 bits contain the fraction of one second. For example:

Time [s]	Representation in 32.32 fixed point format
1	0x0000 0001 0000 0000
1.5	0x0000 0001 8000 0000
2	0x0000 0002 0000 0000
2.00025	0x0000 0002 0010 624D

  Given a constant data rate, any other sample's time stamp can be calculated from this time stamp, the stream sample rate, and the index of the data sample inside the packet.
- Data row count (4 bytes, integer), the number of samples in this packet

Packet payload

The data packet subheader is followed by the actual data. In the data section, multiple samples may be placed, where each sample consists of (number of global signals + number of channels * number of channel signals) values. In the example of the figure in Section 1.3.3 for multi-channel devices, 8 values are transmitted per sample. Samples are transferred one after another. There is no padding (alignment filling with zeros) between the samples. However, in order to keep the total packet length a multiple of four, padding can occur at the end of a data packet.

Signal sorting

As mentioned earlier, data is ordered such that global signals (time stamps, encoder values etc.) are located at the beginning, followed by the channel signals.

For example, if signals 256 and 257 (which is Distance 1 and Intensity 1 in confocal mode) are requested for a *single channel device* (e.g., CHRocodile 2 S), then the data block will appear as:

```
<Signal Value 256><Signal Value 257>
```

If multiple samples are included in one single data packet, the data block may look like:

```
<Signal Value 256 - sample 1><Signal Value 257 - sample 1>
<Signal Value 256 - sample 2><Signal Value 257 - sample 2>
<Signal Value 256 - sample 3><Signal Value 257 - sample 3>
...
```

In case of a *multi-channel device* (e.g., CHRocodile CLS or MPS), if signals 256 and 257 are requested, the data block will appear as:

```
<Signal Value 256 of Channel 1><Signal Value 257 of Channel 1>
<Signal Value 256 of Channel 2><Signal Value 257 of Channel 2>
...
<Signal Value 256 of Channel 191><Signal Value 257 of Channel 191>
<Signal Value 256 of Channel 192><Signal Value 257 of Channel 192>
```

Note the pattern of the data values: values of signal 256 and 257 are given consecutively for each channel. In this example, only one sample is contained in the data packet.

Source code

Below is the data packet header declaration.

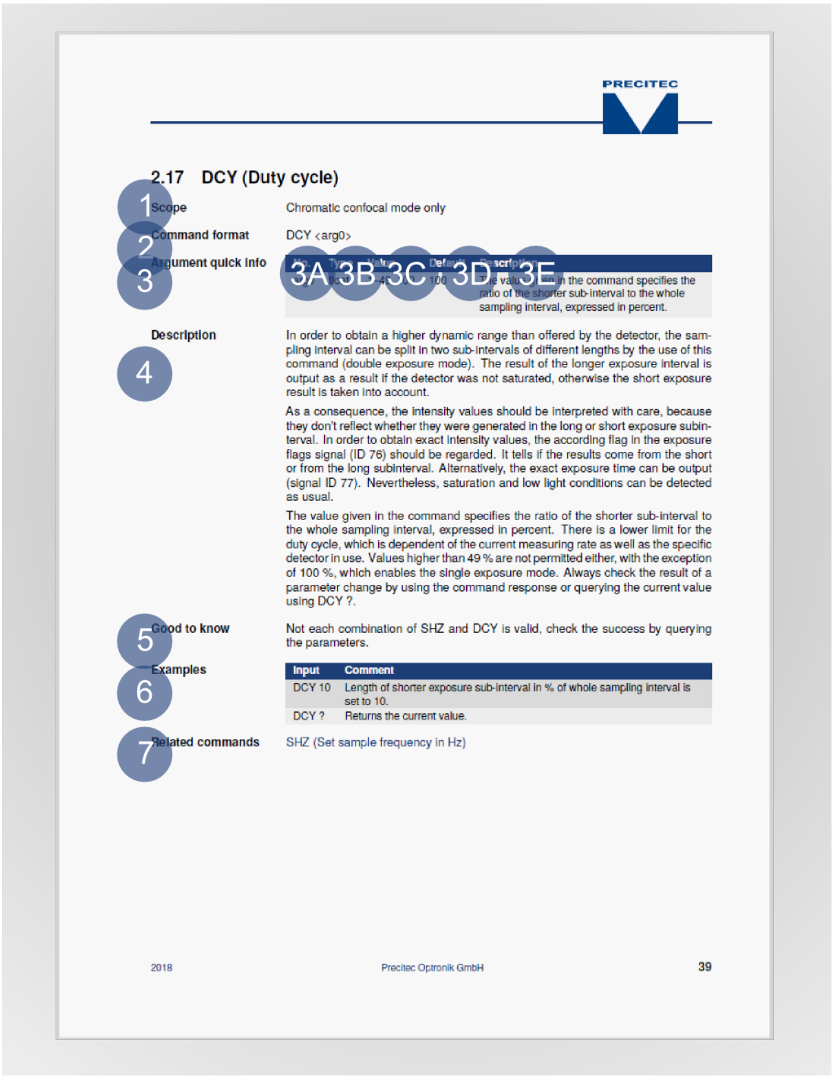
```

    struct TCHRDataHeader {
        u32 DataStreamID;       //always zero
        s32 DataFormatCounter;  //refers to respective data format packet
        u64 TimeStamp;          //Exposure start time of first exposure
        s32 DataRowCount;       //Number of samples contained in this packet
    };
```

Chapter 2

CHRocodile Commands

How to read the Command Reference

Illustration The following figure indicates the different blocks in the Command Reference:



PRECITEC

2.17 DCY (Duty cycle)

Scope Chromatic confocal mode only

Command format DCY <arg0>

Argument quick info

No.	Type	Value	Default	Description
arg0	float	1–49, 100	100	The value given in the command specifies the ratio of the shorter sub-interval to the whole sampling interval, expressed in percent.

Description

In order to obtain a higher dynamic range than offered by the detector, the sampling interval can be split in two sub-intervals of different lengths by the use of this command (double exposure mode). The result of the longer exposure interval is output as a result if the detector was not saturated, otherwise the short exposure result is taken into account.

As a consequence, the intensity values should be interpreted with care, because they don't reflect whether they were generated in the long or short exposure subinterval. In order to obtain exact intensity values, the according flag in the exposure flags signal (ID 76) should be regarded. It tells if the results come from the short or from the long subinterval. Alternatively, the exact exposure time can be output (signal ID 77). Nevertheless, saturation and low light conditions can be detected as usual.

The value given in the command specifies the ratio of the shorter sub-interval to the whole sampling interval, expressed in percent. There is a lower limit for the duty cycle, which is dependent of the current measuring rate as well as the specific detector in use. Values higher than 49 % are not permitted either, with the exception of 100 %, which enables the single exposure mode. Always check the result of a parameter change by using the command response or querying the current value using DCY ?.

Good to know Not each combination of SHZ and DCY is valid, check the success by querying the parameters.

Examples

Input	Comment
DCY 10	Length of shorter exposure sub-interval in % of whole sampling interval is set to 10.
DCY ?	Returns the current value.

Related commands SHZ (Set sample frequency in Hz)

2018 Precitec Optronik GmbH 39

Description

The following table describes each block in detail:

No.	Criterion	Explanation
1	Scope	Scope of command, e.g., limitation to certain modes or communication protocols
2	Command format	Short form of command with associated arguments. Arguments are listed in angle brackets. Optional arguments are additionally enclosed in square brackets.
3	Argument quick info	Info table on command with the following information (3A–3E):
3A	No.	Argument index
3B	Type	Specifies data type of the corresponding argument: int: abbreviation for integer float: abbreviation for floating point number str: abbreviation for string blob: binary large object
3C	Value	Accepted values of the corresponding argument. Two numerical values separated by "–" denote a range, values separated by commas define a fixed set of possibilities for a value. A type name followed by "[]" denotes an array of that type.
3D	Default	Default value. This value will be set when using the SFD command. Note: The default values do not necessarily correspond to the as-delivered settings.
3E	Description	Description of argument
4	Description	Detailed description of command
5	Good to know	Tips and hints when dealing with the command
6	Examples	Examples of good practice to explain the functionality of command
7	Related commands	Links to related commands

Value range and type

The following table describes notations of valid parameter values:

Notation	Explanation
a–b	Values between a and b are valid.
a, b, d	Only parameter values of a, b and d are valid.
full range	No limitation of parameter values
u8, u16, u32	Unsigned integers that are 8, 16 or 32 bits wide
s8, s16, s32	Signed integers that are 8, 16 or 32 bits wide
float	Floating point numbers according to IEEE 754 (single precision)
s16[]	Array of signed 16 bit integers

2.1 AAL (Auto adapt light source)

Command format AAL <arg0> [<arg1>]

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0, 1	0	Enables / disables auto adapt mode
arg1	float	Packet protocol: 0–100	30.5	Packet protocol: Only needed if <arg0> = 1: desired detector level in % saturation level (0–100), approximately 30 is recommended
arg1	int	Dollar protocol: 0–255	78	Dollar protocol: Only needed if <arg0> = 1: desired detector level in % of saturation level (0–255), approximately 78 is recommended

Description

An algorithm tries to keep the level of the detector signal at the given level by constantly adapting the exposure time. The setpoint value is entered as fraction of the saturation level of the detector by the second parameter. If the second parameter is not specified, the device keeps its current setting.
There is a historic difference in the range scaling of the second parameter between the dollar protocol and the packet protocol: On the packet protocol, the setpoint value is given as a percentage of the saturation level in float format (0–100 %). On the dollar protocol, the setpoint value is given as an integer in the range from 0–255.

Good to know

- The auto adapt mode is disabled by the LAI command.
- When the light source is switched off by a LAI 0 command, switching the device to Auto adapt mode using the AAL command mode (AAL 1 n) will turn on LED again.

Examples

Input	Comment
AAL 1 50	Activates auto adapt light source, set target saturation level to 50 %.
AAL 1	Activates auto adapt mode with the previous detector level settings.
AAL 0	Deactivates auto adapt light source.
AAL ?	Returns the current value(s).

Related commands LAI (Lamp intensity)

2.2 ABE (Abbe number)

Command format ABE <arg0> ...

Argument quick info

No.	Type	Value	Default	Description
arg0	float	0–500	0	Abbe number of layer 1
arg1	float	0–500	0	Abbe number of layer 2
...	...	...	0	As many as number of layers (NOP - 1)

Description

Sets Abbe number to achieve a correct thickness measure by modeling the dependency of the refractive index on the wavelength (dispersion). A low Abbe number means a strong dispersion, a high number means little dispersion. A value of 0 codes zero dispersion (constant refractive index for all wavelengths).

The Abbe number on a certain layer only takes effect, if SRT..0.. is selected (no preloaded index table) for the corresponding layer. You should give as many Abbe numbers as there are layers to be measured, which is (number of peaks - 1).

The reply of the query includes Abbe numbers of all layers even though not all are used according to the setting of NOP.

Good to know ABE 0 on a certain layer disables dispersion correction for that layer.

Examples

Input	Comment
ABE 0 155 32.5	Sets the Abbe numbers for three layers.
ABE ?	Returns the current value(s).

Related commands

NOP (Number of peaks)
SRI (Set refractive indices)
SRT (Set refractive index table)

2.3 ALI (Aiming laser intensity)

Command format ALI <arg0>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0–255	255	255: Maximum intensity of the aiming laser 0: Turns off the aiming laser completely.

Description Sets the intensity of the pilot/aiming laser.

Good to know This command is only relevant for devices with a pilot/aiming laser installed.

Examples

Input	Comment
ALI ?	Queries for current aiming laser intensity.
ALI 100	Sets the aiming laser to 100.

2.4 ANAM (Analog output mode)

Command format ANAM <arg0>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0, 1	0	0: Voltage output (-10–10 V) with internal reference 1: Voltage output with LVDT mode

Description The analog outputs support voltage output with an internal reference or LVDT mode. The same type will be used on both outputs. It is selected with the ANAM command.

Related commands ANAX (Analog output function, extended)

2.5 ANAX (Analog output function, extended)

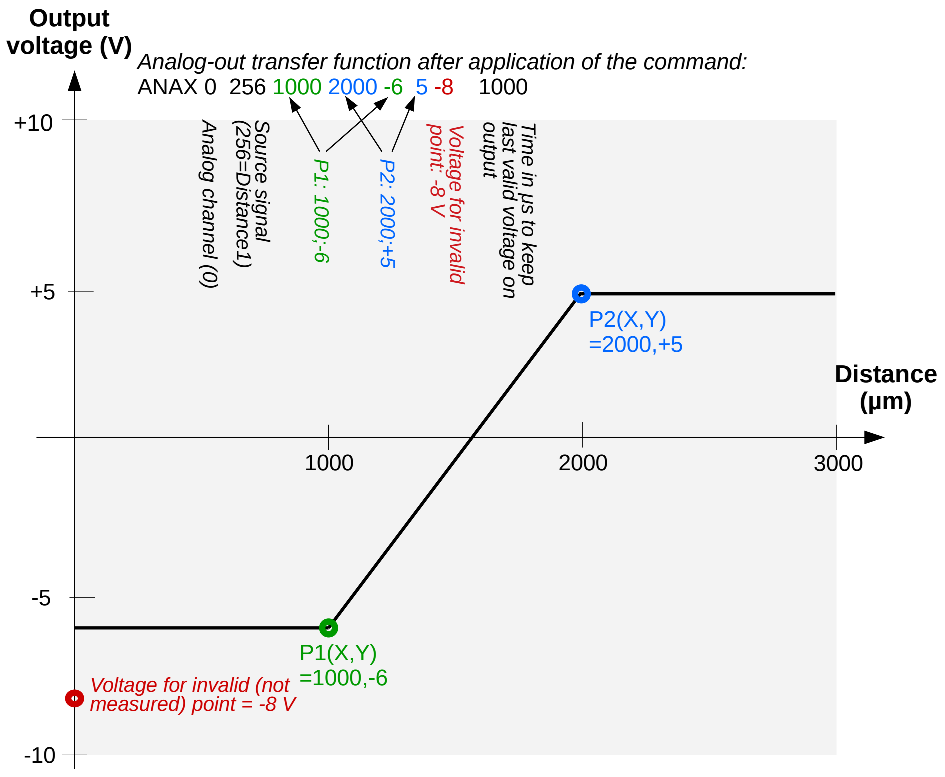
Command format ANAX <arg0> to <arg7>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0, 1	-	Analog output to parameterize
arg1	int	u16	256, 257	Index of result value to be put on Analog Out (default depends on <arg0>)
arg2	float	-	0	Value that should result in output voltage/current given by <arg4>
arg3	float	-	1000	Value that should result in output voltage/current given by <arg5>
arg4	float	-10–10	0	Voltage attributed to value given by <arg2>
arg5	float	-10–10	10	Voltage attributed to value given by <arg3>
arg6	float	-10–10	-1	Invalid output value. Put to output after the hold time (see <arg7>) when no new valid value arrives in the meantime.
arg7	int	u32	0	Hold time of last valid result in µs, 0 means holding for one measurement interval. When no new valid value arrives during the hold time, the invalid value (<arg6>) is output.

Description

The analog output translates an interval given by <arg2>, <arg3> of an output signal defined by <arg1> linearly into a voltage or current range defined by <arg4>, <arg5>. Outside the interval the output voltage/current stays constant at the nearer limit value (<arg4> or <arg5>). When no new valid result is measured, the output changes after a hold time (<arg7>) to the invalid output voltage/current given by <arg6> (see also the following illustration).



Examples

Input	Comment
ANAX 0 256 0 1000 0 10 -1 1000	Distances 0 to 1000 µm are transmitted as 0 to 10 V on output 0. If during 1000 µs no valid result is measured, the last valid voltage is replaced by -1 V.
ANAX 0 ?	Queries settings of first analog out channel.
ANAX 1 ?	Queries settings of second analog out channel.
ANAX ?	Yields a complete list of all settings for both analog out channels.

Related commands ANAM (Analog output mode)

2.6 ASC (ASCII mode)

Scope	Dollar protocol only
Command format	ASC
Argument quick info	No arguments supported
Description	Sets the dollar protocol output in ASCII format.
Related commands	BIN (Binary mode)

2.7 AVD (Data averaging)

Command format

AVD <arg0>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0–10000	1	Data average depth Note: AVD 0 starts window averaging mode.

Description

This command relates to averaging of peak signal data (e.g. distance, thickness, intensity). It averages the results of n samples before outputting. Averaging is not implemented as moving average, so it slows down the output rate by a factor of n. Invalid samples (due to low signal intensity, or low quality) are not taken into account for averaging and thus do not disturb the result. In the case of invalid samples, these are skipped, but the averaging interval is not extended! So, the output rate is not affected by invalid samples.

In Trigger each mode, one trigger event will start a sequence of AVD*AVS exposures. The AVD*AVS exposures are executed with the current sample rate (SHZ). One averaged result will be output after the exposure sequence. There will be only one trigger event on the Sync-out signal per n samples to be averaged. The pulse marks the beginning of the first exposure of the averaging interval. In the other trigger modes, there is one Sync-out pulse for every exposure, regardless of averaging.

Good to know

In double exposure mode (DCY other than 100 %), the intensity result value doesn't contain useful information when averaging is active, as intensities from short and from long exposures might be averaged.

Be careful when using the data averaging in the interferometric mode: The detected thicknesses in the interferometric mode may be ordered according to their signal quality (POD 0). These qualities tend to vary locally quite heavily. Hence, the thicknesses of different layers in a multilayer system could be erroneously mixed together by averaging!
AVD 0 activates the window averaging mode. In this mode, samples are averaged during the high period of the trigger signal, which means one high period of the trigger signal results in one output data sample. If encoder trigger is disabled (ETR 3 0), this signal is combined using the logical AND operator from the Sync-in hardware signal and the level of software trigger signal defined by command STR. Otherwise it reflects the encoder trigger signal.

Examples

Input	Comment
AVD 5	Average over 5 data samples
AVD 1	No data averaging is active.
AVD ?	Returns the current value(s).

Related commands

AVDS (Serial port averaging)
AVS (Spectra averaging)

2.8 AVDS (Serial port averaging)

Command format AVDS <arg0>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	u16	1	Indicates how many samples are averaged.

Description

The peak related values output on the serial port can be averaged in addition to the AVD averaging. If AVDS is set to 3, one sample is output on the serial port every 3 samples that are output on the TCP port.

Examples

Input	Comment
AVDS 5	Sets the serial port averaging factor to 5.

2.9 AVS (Spectra averaging)

Command format AVS <arg0>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	1–256	1	Spectra average depth

Description

During the signal processing, the spectra obtained from the spectrometer can be averaged in order to reduce the noise. This allows extending the dynamic range of the optical sensor as the noise is reduced by a factor of $\sqrt{\frac{1}{n}}$ while the saturation limit stays the same. To make use of the extended dynamic, the detection threshold (THR) should be lowered correspondingly.

Generally speaking, AVS only makes sense for special applications: when a stable measurement signal is available but is subject to high levels of noise, i.e. the signal-to-noise ratio is low.

In case of rapidly changing distances, the peak in the spectrum will be broadened by the spectra averaging and thus the calculation of the result might become less reliable.

In Trigger each mode, one trigger event will start a sequence of AVD*AVS exposures. The AVD*AVS exposures are executed with the current sample rate (SHZ). One averaged result will be output after the exposure sequence. There will be only one trigger event on the Sync-out signal per n samples to be averaged. The pulse marks the beginning of the first exposure of the averaging interval. In the other trigger modes, there is one Sync-out pulse for every exposure, regardless of averaging.

Good to know

- In double exposure mode (DCY other than 100 %), spectra averaging is not possible!
- In chromatic confocal mode, the raw spectra data will be averaged. In interferometric mode, the device will average the Fast Fourier-transformed spectra data.
- On CHRocodile 2 SX, AVS is not supported. If AVS is specified, it is always reset to AVS 1.

Examples

Input	Comment
AVS 5	Average over 5 spectra
AVS 1	No spectra averaging is active.
AVS ?	Command returns the current value.

Related commands AVD (Data averaging)

2.10 BCAF (Binary command argument format)

Command format BCAF <arg0>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0, 1	0	0: binary (default) 1: 2 HEX chars per byte

Description

Format of data contained in commands or command responses. Examples:

- Spectrum download (DNLD)
- Firmware upload (ULFW)

2.11 BDR (Baud rate and hardware handshaking)

Command format BDR <arg0> <arg1>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0–8, n	4	If <arg> $\leq$ 8: interpreted as a baud rate index, otherwise as a free custom baud rate.
arg1	int	0, 1	0	Hardware RTS/CTS handshake on/off

Response quick info

No.	Type	Value	Default	Description
arg0	int	-	-	Baud rate
arg1	int	-	-	Effective baud rate
arg2	int	-	-	Hardware handshake on/off

Description

As described in the section “Argument quick info” above, you can input a baud rate index or a free baud rate. Consider the following table when using index input:

Index	Baud rate [bit/s]
0	9600
1	19200
2	38400
3	57600
4	115200
5	230400
6	460800
7	921600
8	1843200

The baud rate of the serial port can be adjusted to values between 9600 and 1843200 (115200 is the default and recommended value). When entering a free custom baud rate, even higher baud rates are accepted.

The effective baud rate, which might be slightly different from the intended baud rate due to frequency dividing limitations, will be replied as second argument <arg1> for information.

The second argument <arg1> according to the command format definition enables or disables RTS/CTS hardware handshaking.

Good to know

The baud rate takes effect immediately after the command is sent and therefore a serial port sending the command cannot receive the response if the command changes the baud rate.

Examples

Input	Comment
BDR 4	Selects the baud rate by index.
BDR 115200 1	Enters baud rate directly and switches hardware handshaking on.
BDR ?	Returns the current value(s).

2.12 BIN (Binary mode)

Scope	Dollar protocol only
Command format	BIN
Argument quick info	No arguments supported
Description	Sets the dollar protocol data output to binary format.
Related commands	ASC (ASCII mode)

2.13 CONF (Send configuration)

Scope	The command is not supported in the dollar protocol as this protocol does not implement update packets.
Command format	CONF
Argument quick info	No arguments supported
Description	This command serves to publish all current parameter settings at once to a client. Though it has no direct reply arguments, it triggers the device to send out update packets of all parameter settings. As last update packet, a CONF update packet is sent in order to signal the termination of the parameter cycle to the client. The command affects only the client that has sent it, all other clients don't notice it.
Good to know	An equal update packet burst is sent unsolicited at the establishment of a new connection. It is recommended to wait until the CONF update is received before sending commands to the device.

2.14 CRA (Set active detector range)

Scope Interferometric mode only

Command format CRA <arg0> <arg1>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0–1000	0	Start pixel of the active detector range
arg1	int	0–1000	Detector specific	Stop pixel of the active detector range

Description Sets start and stop pixel of the active detector range.

Good to know

- The actual permitted range depends on the detector built into the respective device. Remember to confirm the result by monitoring the command response.
- On CHRocodile 2 SX: Only the detector range of the first light source LS1 can be adjusted. The active detector range for the second light source LS2 is fixed (start pixel 200, stop pixel 700) and cannot be queried.

Examples

Input	Comment
CRA 40 600	Sets the detector range to start at pixel 40 and stop at pixel 600.

2.15 CRDK (Continuous refresh dark factor)

Command format CRDK <arg0>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	u16	0	Refresh factor

Description

This command configures the continuous refresh dark reference mode. The refresh factor 0 turns continuous refresh off. A refresh factor of 65535 configures that the dark reference data is always replaced completely by the spectrum of the preceding exposure. Typical values are quite low (1–10), which means that the dark reference is adapted quite slowly.

The continuous refresh dark reference mode is useful if either the raw spectrum changes quickly and permanently, as in interferometric mode when measuring high thicknesses, or if objects cross the measurement range only rarely and quickly.

Taking a dark reference by an FDK or DRK command ends the CRDK mode (equals CRDK 0).

Good to know

When CRDK is turned off (refresh factor = 0), a new DRK command needs to be executed.

Examples

Input	Comment
CRDK 8192	Sets the refresh factor to 8192. 8192 divided by 65536 gives 0.125. This means that the dark reference for each pixel is calculated as: 0.125*(current spectrum) + 0.875*(previous dark reference)
CRDK ?	Returns the current value.

Related commands

DRK (Dark reference)
FDK (Fast dark reference)

2.16 CTN (Continue in free run mode)

Command format	CTN
Argument quick info	No argument supported
Description	Recover from TRE/TRG/TRW command to go into free run mode. In this mode, new measurements are started periodically based on the sampling rate.
Good to know	• The trigger modes TRE, TRW and CTN are stored in the non-volatile memory after issuing an SSU command (Save setup).
Related commands	TMOD (Trigger mode) TRE (Trigger each) TRG (Trigger once) TRW (Trigger window)

2.17 DCY (Duty cycle)

Scope Chromatic confocal mode only, on CHR 2 SX in interferometric mode

Command format DCY <arg0>

Argument quick info

No.	Type	Value	Default	Description
arg0	float	1–49, 100 (CHR 2 SX: 0–100)	100	The value given in the command specifies the ratio of the shorter sub-interval to the whole sampling interval, expressed in percent.

Description

In order to obtain a higher dynamic range than offered by the detector, the sampling interval can be split in two sub-intervals of different lengths by the use of this command (double exposure mode). The result of the longer exposure interval is output as a result if the detector was not saturated, otherwise the short exposure result is taken into account.

As a consequence, the intensity values should be interpreted with care, because they don't reflect whether they were generated in the long or short exposure sub-interval. In order to obtain exact intensity values, the according flag in the exposure flags signal (ID 76) should be regarded. It tells if the results come from the short or from the long sub-interval. Alternatively, the exact exposure time can be output (signal ID 77). Nevertheless, saturation and low light conditions can be detected as usual.

The value given in the command specifies the ratio of the shorter sub-interval to the whole sampling interval, expressed in percent. There is a lower limit for the duty cycle, which is dependent of the current measuring rate as well as the specific detector in use. Values higher than 49 % are not permitted either, with the exception of 100 %, which enables the single exposure mode. Always check the result of a parameter change by using the command response or querying the current value using DCY ?.

On CHRocodile 2 SX:
The CHRocodile 2 SX device uses two light sources (LS1 and LS2) for the thickness measurement. During one sample measurement, two exposures are taking place with individual exposure times for each light source. The DCY value sets the ratio between the exposure time of LS1 and the sum of the exposure times of light source LS1 and LS2. For more details see the CHRocodile 2 SX User Manual.

Good to know Not each combination of SHZ and DCY is valid, check the success by querying the parameters.

Examples

Input	Comment
DCY 10	Length of shorter exposure sub-interval in % of whole sampling interval is set to 10.
DCY ?	Returns the current value.

Related commands SHZ (Set sample frequency in Hz)

2.18 DNLD (Download spectrum)

Scope	Packet protocol only
Command format	DNLD <arg0>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0–2	-	Spectrum ID

Response quick info

No.	Type	Value	Default	Description
arg0	int	0–2	-	Spectrum ID
arg1	int	u16	-	Exposure number
arg2	float	full range	-	Micrometers per bin
arg3	int	s32	-	Block exponent of spectrum data
arg4	blob	s16[]	-	Spectrum data, one s16 per pixel

In chromatic confocal measurement mode, <arg2> and <arg3> can be ignored.

Description

Clients use this command to request a spectrum from the device. Using <arg0>, you can specify the spectrum type as follows:

Spectrum ID	Spectrum type
0	Raw spectrum (unprocessed signal)
1	Spectrum (chromatic confocal mode)
2	FFT spectrum (interferometric mode)

Since acquiring the requested spectrum can take up several sample periods, a DNLD request from the client causes the device to first respond with a DNLD response that does not contain any spectrum data. It just acknowledges the request. Response arguments are always identical to the command's ones.

Later, when the spectrum is available, the device sends one or more update packets containing the spectrum data. Their arguments are specified in the block Response quick info.

Data packets may arrive in the meantime, i.e., between the response and the updates.

2.19 DRK (Dark reference)

Command format	DRK
Argument quick info	No argument supported

Description

Takes a dark reference and stores the result in the non-volatile flash memory. The operation takes about 3 seconds.

The command response returns the result of the last DRK operation. This result indicates the amount of stray light that was registered. It represents in fact a virtual measuring rate in Hz at which the detector would just be saturated by the stray light.

Good to know

- The action takes place immediately after the command.
- While executing this command, the optical probe must not point to any object in the measuring range.
- If this value is very high (>100 Hz), try to get it lower by cleaning the fiber end faces (see the device´s User Manual for details).
- On CHRocodile 2 SX: The dark reference is taken for both light sources. The command´s response shows the higher value of both light sources (LS1 or LS2).

Examples

Input	Comment
DRK	Takes the dark reference and returns the (virtual) measuring rate in Hz.

Related commands

CRDK (Continuous refresh dark factor)
FDK (Fast dark reference)

2.20 DTF (Data format query)

Scope Dollar protocol only

Command format DTF

Response quick info

No.	Type	Value	Default	Description
arg0	int	-	-	Total length in bytes of a data telegram
arg1	int	-	-	1st signal ID
arg2	int	2–6	-	1st signal type
arg3	int	1–4	-	1st signal offset in bytes
...	...	...	...	Repeats the last 3 arguments for each of the signals.

Description

Query only. Mainly for dollar protocol connections, used to make binary telegram processing easier. In the dollar protocol there is no data format packet, so it can be explicitly queried by this command. Returns total length in bytes of a data telegram, ID, type and byte offset for every signal included in a telegram.

Type number	Type description
2	unsigned 16 bits integer
3	signed 16 bits integer
4	unsigned 32 bits integer
5	signed 32 bits integer
6	float

Examples

Input	Comment
DTF	Returns values in bytes for every signal included in the telegram.

2.21 DWD (Detection windows definition)

Command format DWD <arg0> <arg1> ...

Argument quick info

No.	Type	Value	Default	Description
arg0	float	0–range (current probe)	0	Left edge of window 1 in micrometers
arg1	float	0–range (current probe)	0	Right edge of window 1 in micrometers
arg2	float	0–range (current probe)	0	Left edge of window 2 in micrometers
arg3	float	0–range (current probe)	0	Right edge of window 2 in micrometers
. . .	. . .	. . .	-	Max. 14 additional windows (16 max. in total)

Description

Using this command, up to 16 detection windows can be defined. If LMA is active (1), only the peaks inside the windows will be taken into consideration when calculating peaks and thicknesses.

Good to know

- The left and right edges must be given as pairs.
- The arguments in the response might differ somewhat from the parameters given in the command. The cause of this deviation is that the edges are internally aligned to the detector pixels.
- The value of the right edge of each window must be larger than its left edge.
- DWD without parameters disables windowing so that the whole range is active.

Examples

Input	Comment
DWD 0 190.3 200.5 612.4 745 822	Defines 3 detection windows.
DWD ?	Returns the effective windows currently active.

2.22 ENC (Encoder functions)

Command format ENC <arg0> to <arg2>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0–4	-	Axis index
arg1	int	0–4	-	Encoder subfunction index
arg2	int	s32	-	Encoder subfunction argument

Response quick info

No.	Type	Value	Default	Description
arg0	int	-	-	Axis index
arg1	int	-	-	Encoder subfunction index
arg2	int	-	-	Encoder subfunction argument response value

Description This command defines all aspects of encoder counting. More information on the subfunctions will follow on the next pages.

Examples

Input	Comment
ENC n 0 . . .	See ENC 0 (Set encoder counter value) for axis n.
ENC n 1 . . .	See ENC 1 (Set encoder counter source) for axis n.
ENC n 2 . . .	See ENC 2 (Set encoder preload value) for axis n.
ENC n 3 . . .	See ENC 3 (Set encoder preload event) for axis n.
ENC n 4 . . .	See ENC 4 (Set cut-off frequency for low-pass filter) for axis n or Sync-in.

Related commands ETR (Encoder trigger control)

2.22.1 ENC 0 (Set encoder counter value)

Command format ENC <arg0> 0 <arg2>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0–4	-	Axis index
arg1	int	0	-	Encoder subfunction index
arg2	int	s32	-	Counter value to be set

Description Sets the encoder counter value immediately. The third argument <arg2> gives the value to be taken by the encoder counter.

Good to know It is possible to shorten this command by skipping the second argument <arg1>. However, the command response will always contain three arguments.

Examples

Input	Comment
ENC 1 0 123	Sets the current Y axis counter value to 123.
ENC 1 123	Sets the current Y axis counter value to 123, short-hand version.
ENC 2 0 ?	Queries the current Z axis counter value.
ENC 2 ?	Queries the current Z axis counter value, short-hand version.

Related commands ENC (Encoder functions)
ETR (Encoder trigger control)

2.22.2 ENC 1 (Set encoder counter source)

Command format ENC <arg0> 1 <arg2>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0–4	-	Axis index
arg1	int	1	-	Encoder subfunction index
arg2	int	15, 0–10	15	Source index

Description

This command sets the input source for the encoder counter given by the axis index <arg0>.

The meaning of the third argument <arg2> is as follows:

Index	Input pin(s)	Count mode
0	A0	Pulse
1	B0	Pulse
2	A1	Pulse
3	B1	Pulse
4	A2	Pulse
5	B2	Pulse
6	A3	Pulse
7	B3	Pulse
8	A4	Pulse
9	B4	Pulse
10	Sync-in	Pulse
11–14	open	No counting
15	A<arg0> and B<arg0>	Quadrature

Quadrature count mode (15):

Quadrature input of the axis defined by the axis index argument (A/B encoder count). This is the standard case of operation and permits forward and backward position counting.

Pulse count mode:

Alternatively, single inputs A0, B0, A1 . . . can be used for pulse counting (see pulse count mode description in the CHRocodile User Manual). In that case, the counter only counts up.

For furter discussion and examples, please see also chapter 5 Encoder Interface.

Examples

Input	Comment
ENC 0 1 10	Connects counter 0 to Sync-in for pulse counting.
ENC 2 1 15	Connects counter 2 to quadrature encoder input A2/B2.

Related commands

ENC (Encoder functions)
ETR (Encoder trigger control)

2.22.3 ENC 2 (Set encoder preload value)

Command format ENC <arg0> 2 <arg2>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0–4	-	Axis index
arg1	int	2	-	Encoder subfunction index
arg2	int	s32	0	Preload value

Description

Sets the value that will be loaded into the encoder counter <arg0> when a preload event occurs. The preload event is defined with ENC 3, see subchapter ENC 3 (Set encoder preload event). The third argument <arg2> gives the value that will be preloaded.
For further discussion and examples, please see also chapter 5 Encoder Interface.

Examples

Input	Comment
ENC 0 2 4321	Sets preload value of counter 0 to 4321.

Related commands

ENC (Encoder functions)
ETR (Encoder trigger control)

2.22.4 ENC 3 (Set encoder preload event)

Command format ENC <arg0> 3 <arg2>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0–4	-	Axis index
arg1	int	3	-	Encoder subfunction index
arg2	int	s32	0	Preload event specification

Description

Sets preload event. The preload functionality can be used to reference the incremental encoder counter. The preload event generation is defined by the third argument <arg2> where the bits have the following meaning:

Index	Preload condition
Bit 7	Turns preloading on/off / Inactive (0), Active (1)
Bit 6	Edge (0) / State (1)
Bit 5	Rising edge or high state (0) / Falling edge or low state (1)
Bit 4	Only once, first event (0) / Every event, always (1)
Bits [3..0]	Preload event source selector, see the following table.

Preload event source selector:

Index	Input pin
0	A0
1	B0
2	A1
3	B1
4	A2
5	B2
6	A3
7	B3
8	A4
9	B4
10	Sync-in
11–14	Reserved
15	Immediate preload

The <arg2>-value 178 from the example below is composed of the bit field as follows:

7	6	5	4	3	2	1	0	Decimal value
1	0	1	1	0	0	1	0	
128 +	0 +	32 +	16 +	0 +	0 +	2 +	0	= 178

For further discussion and examples, please see also chapter 5 Encoder Interface.

Examples

Input	Comment
ENC 0 2 1234	Sets the preload value to 1234.
ENC 0 3 178	Configures the encoder counter to load the preload value 1234 into the counter register whenever a falling edge occurs on A1, i.e., the A input of encoder channel 1.

Related commands	ENC (Encoder functions) ETR (Encoder trigger control)

2.22.5 ENC 4 (Set cut-off frequency for low-pass filter)

Command format ENC <arg0> 4 <arg2>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0–5	-	Axis index (0–4), Sync-in (5)
arg1	int	4	-	Encoder subfunction index
arg2	int	0–7	2	Index of cut-off frequency of low-pass filter

Description

This command sets the cut-off frequency for a low-pass filter which is applied to the encoder and Sync-in signals in order to suppress high-frequency interferences on poor signal quality. Keep in mind that with a stronger low-pass filter setting, the maximum usable signal frequency also decreases.

The first argument <arg0> denotes the input signal. The low-pass filter can individually be parametrized for each encoder input or Sync-in signal.

The cut-off frequencies for the low-pass filter <arg 2> settings for signals with 50 % duty cycle are as follows:

Index	Cut-off frequency
0	17 MHz
1	10 MHz
2	5.6 MHz
3	2.9 MHz
4	1.5 MHz
5	0.77 MHz
6	0.38 MHz
7	0.19 MHz

For the quadrature signals of the encoders counting frequencies up to four times the specified signal frequencies are possible. The default setting of the low-pass filter is 2, which means that the maximum encoder count frequency is 22 million increments/sec.

Good to now

If the low-pass filter is set to a cut-off frequency of 17 MHz, 68 million increments/sec are possible in quadrature mode. This maximum value can only be achieved under the following conditions: The high and low phases of a signal in quadrature mode must be longer than $\frac{1}{2*17\ MHz}$.

For further discussion and examples, please see also chapter 5 Encoder Interface.

Examples

Input	Comment
ENC 0 4 3	Sets the cut-off frequency of low-pass filter to 2.9 MHz to X axis.
ENC 5 4 5	Sets the cut-off frequency of low-pass filter to 0.77 MHz to Sync-in.

Related commands

ENC (Encoder functions)
ETR (Encoder trigger control)

2.23 EQN (Equalize Fourier noise)

Scope Interferometric mode only

Command format EQN <arg0> <arg1>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0, 1, 3141	-	Turn off, turn on or acquire new equalization function
arg1	float	-	-	Reserved

Description This command determines the amplification function on FFT pixels in the interferometric mode to achieve the same noise level over the measuring range. It will be stored permanently in non-volatile memory.

This command may only be necessary after significant changes of the detector range in interferometric mode.

Good to know No object shall be in the measuring range of the device. Required command sequence: DRK, FDK, EQN 3141, DRK

Examples

Input	Comment
EQN 0	Turns off equalization function.
EQN 1	Turns on equalization function.
EQN 3141	Takes a new equalization function and turns it on.

Related commands DRK (Dark reference)
FDK (Fast dark reference)

2.24 ETR (Encoder trigger control)

Command format ETR <arg0> <arg1>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0–5, 7	-	Encoder trigger subfunction index
arg1	...	...	-	See ETR subcommands

Description

The ETR command groups several functions related to encoder triggering.

The encoder trigger is implemented as a state machine. In the idle state, it waits for the encoder counter of the selected axis to pass the start position (in either direction) where it generates the first trigger event. Then the trigger interval value is added to the current position and when this position is reached, the next trigger event is generated. This step is repeated until the stop position is encountered. The generation of trigger events is now stopped.

If triggering during return movement is selected, the state machine waits for the stop position to be passed once again and generates trigger events similarly to the forward movement (the trigger interval is now subtracted instead of added) until the start position is reached. The state machine then goes back to the idle state. If no trigger during return movement is selected, the state machine waits for the start position to be passed over (during return movement) and then passes to the idle state.

Learn more about encoder triggering and triggered measurements in chapter 6 Triggered Measurements.

Good to know

The state machine is reset by the ETR 0 (Encoder trigger start position) subcommand.

Examples

Input	Comment
ETR 0	See ETR 0 (Encoder trigger start position).
ETR 1	See ETR 1 (Encoder trigger stop position).
ETR 2	See ETR 2 (Encoder trigger interval).
ETR 3	See ETR 3 (Encoder trigger state control).
ETR 4	See ETR 4 (Encoder trigger active during return).
ETR 5	See ETR 5 (Encoder trigger source).
ETR 7	See ETR 7 (Encoder trigger roundtrip / endless mode).

Related commands

CTN (Continue in free run mode)
ENC (Encoder functions)
STR (Set trigger)
TRG (Trigger once)
TRE (Trigger each)
TRW (Trigger window)

2.24.1 ETR 0 (Encoder trigger start position)

Scope Only relevant if roundtrip trigger is active (see ETR 7).

Command format ETR 0 <arg1>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0	-	Encoder trigger subfunction index
arg1	int	s32	0	Start position value

Description Sets start position and resets the encoder trigger state machine.

Good to know This subcommand must be the last one in a sequence of ETR commands so that all encoder trigger configurations can be applied.

Examples

Input	Comment
ETR 0 100	Sets trigger start position to 100.

Related commands
CTN (Continue in free run mode)
ENC (Encoder functions)
STR (Set trigger)
TRG (Trigger once)
TRE (Trigger each)
TRW (Trigger window)

2.24.2 ETR 1 (Encoder trigger stop position)

Scope	Only relevant if roundtrip trigger is active (see ETR 7).
Command format	ETR 1 <arg1>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	1	-	Encoder trigger subfunction index
arg1	int	s32	1000	Stop position value

Description Sets the stop position for encoder trigger.

Examples

Input	Comment
ETR 1 1000	Sets trigger stop position to 1000.

Related commands

CTN (Continue in free run mode)
ENC (Encoder functions)
STR (Set trigger)
TRG (Trigger once)
TRE (Trigger each)
TRW (Trigger window)

2.24.3 ETR 2 (Encoder trigger interval)

Command format ETR 2 <arg1>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	2	-	Encoder trigger subfunction index
arg1	float	full range	100	Trigger interval value

Description

Sets trigger interval value. The argument <arg1> is a float so that the trigger interval can be given in fractions of encoder counts (e.g., 100.5).

In detail: The encoder trigger interval can be specified as a floating point value. Internally, the device converts this value to a 16 bit integer using floor rounding (truncation). This means that any decimal part is removed, and the value is rounded down to the nearest whole number.
However, the device does not discard the decimal part completely. Instead, it keeps track of the accumulated value internally. This means that over time, the rounding error is corrected naturally.

Example behavior with 100.5 as input:
1. First trigger: 100.5 → rounded to 100
Internal counter now at 100.5
2. Second trigger: 100.5 + 100.5 = 201 → rounded to 201
3. Third trigger: 201 + 100.5 = 301.5 → rounded to 301
4. Fourth trigger: 301.5 + 100.5 = 402 → rounded to 402

This means that, on average, the trigger interval is 100.5, even though individual intervals may vary slightly. Over time, the device ensures the correct timing. If precise timing is critical, consider choosing values that are exact integers to avoid minor variations.

Examples

Input	Comment
ETR 2 10.5	Sets trigger interval to 10.5.

Related commands

CTN (Continue in free run mode)
ENC (Encoder functions)
STR (Set trigger)
TRG (Trigger once)
TRE (Trigger each)
TRW (Trigger window)

2.24.4 ETR 3 (Encoder trigger state control)

Command format ETR 3 <arg1>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	3	-	Encoder trigger subfunction index
arg1	int	0, 1	0	Inactive / active

Description

Controls the encoder trigger state:

- <arg1> = 0: (default) Encoder trigger is inactive.
- <arg1> = 1: Encoder trigger is active.

Examples

Input	Comment
ETR 3 1	Activates encoder trigger.

Related commands

CTN (Continue in free run mode)
ENC (Encoder functions)
STR (Set trigger)
TRG (Trigger once)
TRE (Trigger each)
TRW (Trigger window)

2.24.5 ETR 4 (Encoder trigger active during return)

Scope	Only applicable for roundtrip trigger mode.
Command format	ETR 4 <arg1>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	4	-	Encoder trigger subfunction index
arg1	int	0, 1	0	Inactive / active during return movement

Description

Enables trigger during return movement:

- <arg1> = 0: (default) Encoder trigger is only active during the movement from start position to stop position.
- <arg1> = 1: Encoder trigger is also active during the return movement from stop position to start position.

Examples

Input	Comment
ETR 4 0	Disables trigger during return movement.

Related commands

CTN (Continue in free run mode)
ENC (Encoder functions)
STR (Set trigger)
TRG (Trigger once)
TRE (Trigger each)
TRW (Trigger window)

2.24.6 ETR 5 (Encoder trigger source)

Command format ETR 5 <arg1>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	5	-	Encoder trigger subfunction index
arg1	int	0–4	0	Encoder counter axis index

Description Chooses an encoder counter as trigger source using its index.

Examples

Input	Comment
ETR 5 0	Chooses axis 0 as encoder trigger source.

Related commands

CTN (Continue in free run mode)
ENC (Encoder functions)
STR (Set trigger)
TRG (Trigger once)
TRE (Trigger each)
TRW (Trigger window)

2.24.7 ETR 7 (Encoder trigger roundtrip / endless mode)

Command format ETR 7 <arg1>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	7	-	Encoder trigger subfunction index
arg1	int	0, 1	0	Roundtrip / endless trigger mode

Description

Roundtrip / endless mode:

- <arg1> = 0: (default) Roundtrip trigger. Start and stop positions are used.
- <arg1> = 1: Endless trigger. Generates one trigger event on every interval regardless of any start / stop position.

Examples

Input	Comment
ETR 7 1	Activates endless trigger.
ETR 7 0	Activates roundtrip trigger.

Related commands

CTN (Continue in free run mode)
ENC (Encoder functions)
STR (Set trigger)
TRG (Trigger once)
TRE (Trigger each)
TRW (Trigger window)

2.25 EWD (Exclude windows)

Command format EWD <arg0> <arg1> ...

Argument quick info

No.	Type	Value	Default	Description
arg0	float	0–range (current probe)	0	Left edge of window 1 in micrometers
arg1	float	0–range (current probe)	0	Right edge of window 1 in micrometers
arg2	float	0–range (current probe)	0	Left edge of window 2 in micrometers
arg3	float	0–range (current probe)	0	Right edge of window 2 in micrometers
. . .	. . .	. . .	-	Max. 14 additional windows (16 max. in total)

Description

Using this command, measurement ranges can be ignored. Thus, up to 16 exclude windows can be defined.

Good to know

- The left edge and right edge must be given in pairs.
- The reply argument float of each window edge describes the physical distance (or thickness) corresponding to the detector pixel closest to the user input (affected by the calibration table). Therefore it may deviate slightly from the input value.
- The value of the right edge of each window must be larger than its left edge.
- EWD without parameters disables windowing so that the whole range is active.

Examples

Input	Comment
EWD 0 190.3 200.5 612.4 745 822	Excludes 3 windows with the mentioned edges.
EWD ?	Returns the effective positions of the currently active windows.

2.26 FDK (Fast dark reference)

Command format FDK [<arg0> <arg1>]

Argument quick info

No.	Type	Value	Default	Description
arg0	int	1–255	-	Number of spectra to average for the dark reference
arg1	int	u16	-	Refresh factor

Response quick info

No.	Type	Value	Default	Description
arg0	float	-	-	(Virtual) measuring rate in Hz at which the detector would be saturated by the stray light

Description

This command takes a dark reference at the current measuring rate. This dark reference result is not stored in the non-volatile memory. It permits to take very fast dark references very frequently, for example in an inline application. You can specify the number of spectra that are averaged to obtain the new dark reference. If not given, the value of 100 is assigned.

With a second optional parameter you can specify a refresh factor which takes effect as follows:

$$newRef = \frac{RefreshFactor}{65535} * AverageOfSpectra + (1 - \frac{RefreshFactor}{65535}) * OldRef \tag{2.1}$$

A big value (65535) for refresh factor replaces the old reference by the new one, a small value modifies the old reference only by a small portion. When FDK is used with one parameter, this parameter gives the average number and the refresh factor defaults to 65535 (replace old dark reference completely).

The command response indicates the amount of stray light that was registered. It represents in fact a virtual measuring rate in Hz at which the detector would just be saturated by the stray light.

Good to know

- The fast dark reference is only valid for the current measuring rate and LAI setting.
- The dark reference acquired by FDK won't be saved to non-volatile memory. It will be replaced by the previous reference acquired by the DRK command as soon as the device restarts.
- On CHRocodile 2 SX: The dark reference is taken for both light sources. The command´s response shows the higher value of both light sources (LS1 or LS2).

Examples

Input	Comment
FDK	Carries out a fast dark reference and returns (virtual) measuring rate in Hz.
FDK 50	Carries out a fast dark reference (average of 50 spectra) and returns (virtual) measuring rate in Hz.
FDK 50 10020	Carries out a fast dark reference (average of 50 spectra, refresh factor 10020) and returns (virtual) measuring rate in Hz.

Related commands	CRDK (Continuous refresh dark factor) DRK (Dark reference)

2.27 GAN (Detector gain setting)

Command format GAN <arg0>

Argument quick info

CHR 2 IT, CHR 2 IT RW, CHR 2 IT DW, CHR 2 HTW, CHR 2 HDW 250/500:

No.	Type	Value	Default	Description
arg0	int	Detector specific	0	Gain setting index

CHR 2 S (only 70 kHz version), CHR 2 SE (only 70 kHz version), CHR 2 S HP (only 70 kHz version), CHR 2 LR (only 70 kHz version):

No.	Type	Value	Default	Description
arg0	int, str	Detector specific	1	Gain setting index (0—2); “hdr”: activating HDR mode

Description

This command changes the detector gain setting. Gain refers to the ability of the electronic circuit to increase the signal amplitude. Increasing the gain will increase the signal amplitude, however noise will also increase. To increase the gain can be useful if the signal is at the limit of measurability with the currently set gain and the exposure time cannot be increased further (because a certain measuring rate is required).
Valid values of the command argument are specific to each detector type. Furthermore, for certain CHRocodile devices (see examples below) a High Dynamic Range (HDR) can be activated. The HDR mode extends the dynamic range of a detector exposure.

Good to know

- After changing the gain, a dark reference (DRK) must be performed.
- Gain settings will be saved in non-volatile memory when issuing the command SSU (Save setup).

Examples

Sensor CHR 2 IT, CHR 2 IT RW, CHR 2 IT DW:

Input	Comment
GAN 0	Highest gain
GAN 1	Medium gain = 0.12 x highest gain
GAN 2	Lower gain = 0.012 x highest gain
GAN 3	Lowest gain = 0.007 x highest gain

Sensor CHR 2 HTW, CHR 2 HDW 250/500:

Input	Comment
GAN 0	High sensitivity mode
GAN 1	High Dynamic Range (HDR) mode

Sensor CHR 2 S, CHR 2 SE, CHR 2 S HP, CHR 2 LR (only 70 kHz versions):

Input	Comment
GAN 0	Highest gain = 6 x medium gain
GAN 1	Medium gain (default setting)
GAN 2	Lowest gain = 0.23 x medium gain
GAN hdr	Activates High Dynamic Range (HDR) mode. Note: To deactivate, switch back to one of the other GAN modes.

2.28 GLE (Get last errors)

Command format GLE [<arg0>]

Argument quick info

No.	Type	Value	Default	Description
arg0	int	-	-	Number of errors to be queried

Response quick info

No.	Type	Value	Default	Description
arg0	int	-	-	Error code
arg1	int	-	-	Sub-error code
arg2	int	-	-	ID of client causing the error
arg3	int	-	-	Ticket number of command causing the error
arg4	str	-	-	String of the command causing the error
arg5	str	-	-	String of error description

Description

Queries latest errors. If no argument is given, only the last error will be returned.

The following table gives detailed information about error and sub-error codes:

Error code	Sub-error code	Explanation
	0	Unknown_Error
1	1	Device_Cannot_Connect
1	3	Device_Not_Connected
3	1	Device_Unknown_CMD
3	2	Device_No_Support_CMD
3	3	Device_Invalid_Operation_CMD
3	4	Device_Operation_Failed_CMD
4	1	Device_Param_OverRange
4	2	Device_Param_WrongType
4	3	Device_Param_Invalid_Value
4	4	Device_Param_WrongArgCount
4	5	Device_Param_Mismatch
4	6	Device_Param_Malform
8		Client_Connect_Error
9		Packet_Error
10		High_Level_Error

Examples

Input	Comment
GLE 1	Returns the last error.

2.29 IDE (Identification)

Command format IDE

Argument quick info No arguments supported

Description Queries device specific parameters. The response has a string argument that contains key-value pairs.

Examples

Input	Comment

IDE	Returns for example the following string:
	
	`personality.caps.light=led/sld`
	`personality.caps.max_frequency=...`
	`personality.caps.spectrometer=...`
	`personality.channelcount=...`
	`personality.oem.devicetype=...`
	`personality.oem.id=...`
	`personality.proddate=...`
	`personality.revision=...`
	`personality.serial=...`
	`personality.softwareconfiguration=0`

Related commands VER (Version information)

2.30 IPCN (IP configuration)

Command format IPCN <arg0> [<arg1> . . . <arg9>]

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0, 1	0	DHCP on (1)/off (0)
arg1	int	0–255	192	IP address byte 1
arg2	int	0–255	168	IP address byte 2
arg3	int	0–255	170	IP address byte 3
arg4	int	0–255	2	IP address byte 4
arg5	int	0–255	255	Subnet mask byte 1
arg6	int	0–255	255	Subnet mask byte 2
arg7	int	0–255	255	Subnet mask byte 3
arg8	int	0–255	0	Subnet mask byte 4
arg9	int	1500–9000	1500	Ethernet Maximum Transmission Unit (MTU)

Description

Command to define TCP/IP communication settings.

If DHCP is on (<arg0> = 1), you should neither specify an IP address nor a subnet mask. If DHCP is off (<arg1> = 0), IP address should be specified and subnet mask is optional (if not given, the default value is used). Common restrains on IP address and subnet mask also apply.

Good to know

- If the Ethernet Maximum Transmission Unit MTU (<arg9>) is set to values larger than 1530 bytes, the user must ensure that the network interface card supports jumbo frames. Jumbo frames might help to increase the data throughput when using a point-to-point connection, but must be individually tested.
- After the input of IPCN, the CHRocodile's Ethernet device will be reset and thus all TCP/IP connections will be closed, without receiving the response/update of IPCN.
- Power cycling the device after issuing this command is strongly recommended.

Examples

Input	Comment
IPCN 0 192 168 170 2 255 255 255 0	Turns DHCP client off, sets the IP address and subnet mask.
IPCN 1	Turns DHCP client on.
IPCN ?	Returns the current value(s).

2.31 LAI (Lamp intensity)

Command format LAI <arg0>

Argument quick info

No.	Type	Value	Default	Description
arg0	float	0–100	100	Lamp intensity in percentage

Description

This function sets the effective brightness of the light source. Depending on the device type, this takes place via changing either the intensity of the light source or the detector exposure time. In case of a LED/SLD/laser diode as light source, a LAI value of 0 switches the lightsource off.

Good to know

- This command also disables the auto adapt mode.
- Because of detector's frame overhead time, a target value of 100 % can't always be reached. The maximum reachable value depends on the detector type and the current sample rate (SHZ).

Examples

Input	Comment
LAI 80	Sets the effective brightness of the light source to 80 %.
LAI ?	Queries the current parameter value.

Related commands AAL (Auto adapt light source)

2.32 LAMP (External lampbox commands)

Command format	LAMP . . .
Description	Various commands to control the optional external Halogen Dual Lamp Module. The communication is implemented via RS422 serial interface. For details see the Halogen Dual Lamp Module User Manual.

2.33 LMA (Detection limits active)

Command format LMA <arg0>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0, 1	0	Status of detection limits (0 for inactive and 1 for active)

Description Activates or deactivates the detection windows. If inactive, the full thickness/distance detection range is active. If active, the preset windows configured by DWD are used.

Examples

Input	Comment
LMA 1	Activates the detection limits.
LMA ?	Returns the current value.

Related commands DWD (Detection windows definition)

2.34 LOC (Lock front panel keyboard)

Command format LOC [<arg0>]

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0, 1	0	Whether to lock the front panel inputs (jogwheel and keys)

Description This command locks and unlocks the front panel, namely the input keys and the jogwheel. The locking state can be given explicitly using the single integer argument. Without argument, the command toggles the locking state.

Good to know The locking state is not saved by the SSU command. After a power cycle, the keyboard is always accessible.

Examples

Input	Comment
LOC	Toggles the locking state.
LOC ?	Queries the locking state.
LOC 0	Unlocks inputs.
LOC 1	Locks inputs.

2.35 MAN (Manual)

Command format MAN <arg0>

Argument quick info

No.	Type	Value	Default	Description
arg0	str	-	-	Command string

Description Shows the description of the requested command as a response (e.g. on the display of the device, in CHRocodileExplorer etc.).

Good to know Not all commands are supported. Using MAN ?, a list of commands for which a manual entry exists can be queried.

Examples

Input	Comment
MAN ENC	Shows a description of the ENC command.
MAN ?	Lists supported commands.

2.36 MED (Median width)

Command format MED <arg0>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	3–31	7	Median filter width

Description

This command defines how many measurement values are to be used for the calculation of the median. Calculating the median makes sense if a considerable number of outliers are to be expected which would unnecessarily falsify the average value of the measurement. Up to 8 median-filtered signals can be included in the output telegram.
The result of the median calculation can be selected by requesting the corresponding signal IDs. The median is determined by setting bit No. 2 of the raw distance value´s signal ID. For example, if the raw signal ID of Distance 1 is 256, the result is output in signal ID 260. For more information on signal ID composition refer to chapter 3 CHRocodile Signal IDs.

Good to know

- Only thicknesses and distances can be considered for median calculation. Its output is a separate signal (PeakValue Median, which is PeakValue signal ID + 4). For more information, please refer to chapter 3 CHRocodile Signal IDs.
- The width setting is globally applied in all median calculations on all peaks being processed.
- If there are not enough data points (e.g. after booting the device) or the previous data points are based on different measurement settings, the PeakValue Median Signal might deliver inappropriate results.
- The filter window is reset each time a MED or MEDX command is encountered.

Examples

Input	Comment
MED 7	In this case 7 measured values are used for median calculation.

2.37 MEDX (Extended median filter)

Command format MEDX <arg0> [<arg1>] [<arg2>] [<arg3>]

Argument quick info

No.	Type	Value	Default	Description
arg0	int	3–31	7	Extended median filter width
arg1	float	0–100	50	The percentile threshold for the first result
arg2	float	0–100	-	The percentile threshold for the second result (only relevant for Precitec IDM)
arg3	char	‘v’	-	Whether only valid values are considered

Description This command is used to configure the extended median filter. The filter sorts a specified number of measured values given by <arg0> in ascending order, then outputs the result at the percentile threshold given by <arg1>. <arg2> is not considered and exists only for compatibility reasons. Its input is any measured distance/thickness. Its output is a separate signal (PeakValue Median). For more information, please refer to chapter 3 CHRocodile Signal IDs. Up to 8 median-filtered signals can be included in the output telegram.

Good to know

- Only thicknesses and distances can be considered for median calculation. Its output is a separate signal (PeakValue Median, which is PeakValue signal ID + 4). For more information, please refer to chapter 3 CHRocodile Signal IDs.
- The width setting is globally applied in all median calculations on all peaks being processed.
- If there are not enough data points (e.g. after booting the device) or the previous data points are based on different measurement settings, the PeakValue Median Signal might deliver inappropriate results.
- The filter window is reset each time a MED or MEDX command is encountered.

Examples

Input	Comment
MEDX 30 10 90 v	Sets the extended median filter to consider 30 valid values and outputs the first result at 10 %, second result at 90 % of those values (second result only relevant for Precitec IDM).
MEDX 30 90 10 v	Sets the extended median filter to consider 30 valid values and outputs the first result at 90 %, second result at 10 % of those values (second result only relevant for Precitec IDM).

Related commands MED (Median width)

2.38 MESG (Message from device to client)

Scope Packet protocol only

Command format No argument supported

Response quick info

No.	Type	Value	Default	Description
arg0	int	-	-	Message type: 0: error 1: warning 2: info
arg1	int	-	-	Message severity: Lower number means higher severity with 0 representing the most severe event.
arg2	int	-	-	Message ID: reserved
arg3	str	-	-	Message content: individual content

Description This command is used to let the device inform the client about events/issues.

2.39 MMD (Measurement mode)

Command format MMD <arg0>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0, 1	0	0: chromatic confocal mode 1: interferometric mode

Description Sets the measurement mode to chromatic confocal (0) or interferometric (1) independent of the number of detected peaks.

Examples

Input	Comment
MMD 0	Sets the device to chromatic confocal mode.
MMD ?	Returns the current value(s).

Related commands NOP (Number of peaks)

2.40 NOP (Number of peaks)

Command format NOP <arg0>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	1–max. NOP	1	Number of peaks

Description Selects maximum number of peaks (surfaces) to be detected.

Good to know

- When reducing the NOP, signals related to then invalid peaks will automatically have a value of 0. NOP also affects the number of valid layers or thicknesses (NOP - 1) and thus all signals referring to invalid layers will have a value of zero.
  Example: When changing from NOP 3 to NOP 2, eventually selected third distance and second thickness will have a value of zero.
- In confocal mode, if less than NOP peaks are detected, all thickness signals will be invalidated because peak identification is not possible.
- On CHRocodile 2 SX devices, all valid peaks from both light sources are being tracked internally in CHRocodile 2 first before applying the NOP-command-value. The peak order for the final peaks output from both light sources is defined by POD command as usual.

Examples

Input	Comment
NOP 3	Sets the number of peaks.
NOP ?	Returns the currently specified number of peaks.

2.41 OFN (Output function)

Command format OFN <arg0> [<arg1>]

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0–5	-	Output function index
arg1	int	0–15	0	Output function value

Description

The OFN command enables users to either activate certain predefined functions to control the state of the outputs or control the state of the outputs directly. Note that these outputs are located on the Interface connector (pin 9...12, AUX 3...0). In the following, the state of the four outputs AUX OUT 3 – AUX OUT 0 is represented as a binary pattern that can be interpreted as a number. The bit value of “0” represents a voltage of 0 V (with respect to GND) on the respective output, whereas the bit value “1” represents either 5 V (no external voltage on pin 13 of the Interface connector) or V_{ext} (up to 24 V) if V_{ext} is supplied and >5 V.

The decimal representation of the binary number is as follows:
nAUX OUT = Bitval(AUX OUT 3) * 8
+ Bitval(AUX OUT 2) * 4
+ Bitval(AUX OUT 1) * 2
+ Bitval(AUX OUT 0) * 1

Each binary digit in the provided value Bitval corresponds to one of the digital outputs, where “1” signifies the output is enabled and “0” signifies it is disabled. Note that the binary number nAUX OUT is used in its decimal representation for the OFN command.

OFN 0 x: Set the bit pattern encoded in nAUX OUT directly to the outputs. All outputs are affected.

Examples:

OFN 0 13 $\rightarrow$ nAUX OUT = 10. The binary representation of decimal 10 = b1010. Thus, OFN 0 10 sets the output pattern b1010, enabling outputs 1 and 3 while disabling outputs 0 and 2.

OFN 0 15 sets the output pattern to b1111, enabling all four outputs simultaneously.

OFN 0 7 sets the output pattern to b0111, enabling outputs 0, 1 and 2 while disabling output 3.

Examples:

AUX OUT	3 2 1 0	3 2 1 0	3 2 1 0
	🟨 ⬛ 🟨 ⬛	🟨 🟨 🟨 🟨	⬛ 🟨 🟨 🟨
Binary pattern	1 0 1 0	1 1 1 1	0 1 1 1
Decimal representation	10	15	7
Command	OFN 0 10	OFN 0 15	OFN 0 7

🟨 On, +5 V / V_{ext} ⬛ Off, 0 V

OFN 0: Turns off all digital outputs.

OFN 1: Digital outputs represent 4 LSBits (least significant bits) of active calibration table index.

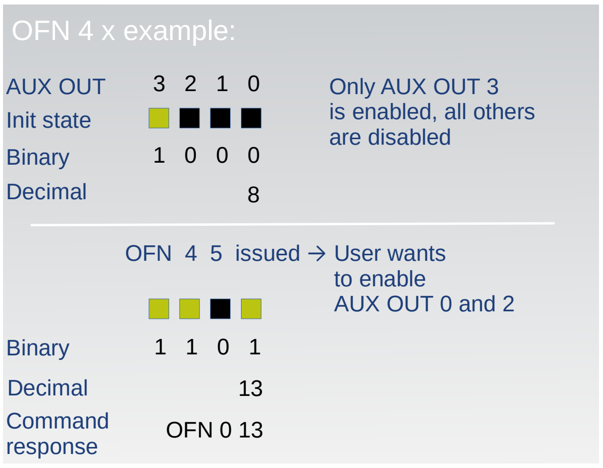
OFN 2: Output [0] represents LSBit of active calibration table index, output [1] represents measuring mode (0 for chromatic confocal, 1 for interferometric).

OFN 3: Outputs [2..0] represent 3 LSBits of active calibration table index, output [3] represents measuring mode (0 for chromatic confocal, 1 for interferometric).

OFN 4 x: Set the bits defined by the argument x in the output pattern nAUX OUT, leave the other outputs unchanged. The equivalent bit operation is:

```
nAUX OUT <= nAUX OUT | x
```

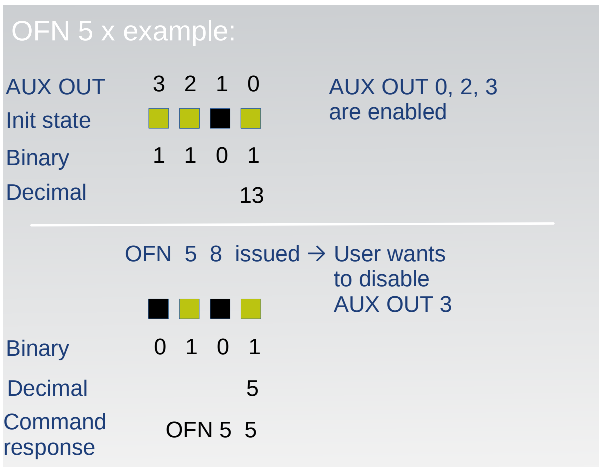
Example:
Assume that only AUX OUT 3 is enabled. nAUX OUT is currently: 8 (b1000). Issue command OFN 4 5 to enable AUX OUT 0 and 2: nAUX OUT = 13 (b1101) $\rightarrow$ device response is OFN 013 $\rightarrow$ AUX OUT 3 was enabled and, additionally, OFN 4 5 enabled 0 and 2.


OFN 4 x example:
AUX OUT 3 2 1 0
Init state 🟩 ■ ■ ■
Binary 1 0 0 0
Decimal 8
Only AUX OUT 3 is enabled, all others are disabled
OFN 4 5 issued → User wants to enable AUX OUT 0 and 2
🟩 🟩 ■ 🟩
Binary 1 1 0 1
Decimal 13
Command response OFN 0 13

OFN 5 x: Reset the bits defined by the argument x in the output pattern nAUX OUT, leave the other outputs unchanged. The equivalent bit operation is:

```
nAUX OUT <= nAUX OUT & (~x)
```

Example:
Assume that nAUX OUT is currently 13 (b1101); AUX OUT 0, 2, 3 are enabled. Issue command OFN 5 8 to disable AUX OUT 3: nAUX OUT = 5 (b0101)



Good to know

The first argument (arg0) is stored in the non-volatile memory after issuing an SSU command (Save setup). In contrast, the second argument encoding the binary representation of the digital outputs (output function value) will not be saved in non-volatile memory even after issuing an SSU command.

Examples

Input	Comment
OFN 0	Sets digital outputs: output[3..0] = b0000
OFN 0 7	Sets digital outputs: output[3..0] = b0111
OFN 0 11	Sets digital outputs: output[3..0] = b1011
OFN 0 ?	Returns the output signal state.

2.42 OPD (Operation data)

Command format OPD <arg0>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0, 1, 2, 3, 5, 6	-	Subcommand index

Description

This command queries operational data (e.g., total operation time in seconds) and controls the time stamp synchronization (OPD 6).

Examples

Input	Comment
OPD 0 ?	See OPD 0 (Lamp lifetime).
OPD 2 ?	See OPD 2 (Total operation time).
OPD 3 ?	See OPD 3 (Number of power ups).
OPD 5 ?	See OPD 5 (Uptime since last power cycle).
OPD 6 ?	See OPD 6 (Time stamp synchronization).

2.42.1 OPD 0 (Lamp lifetime)

Command format OPD <arg0> <arg1>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0	-	Subcommand index
arg1	int	314	-	Magic number

Response quick info

No.	Type	Value	Default	Description
arg0	int	0	-	Subcommand index
arg1	str	-	-	Operation time since the last lamp change. String is composed as follows: 000000h, 00min

Description

This command marks the point in time at which the last change of lamp was done. It always returns the device's operation time since the last lamp change. Resetting the lamp lifetime is carried out with a fixed magic number ("314").

Examples

Input	Comment
OPD 0 314	Marks that the lamp was changed.
OPD 0 ?	Returns operation time in human readable form [h, min] since the last lamp change.

2.42.2 OPD 2 (Total operation time)

Command format OPD <arg0>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	2	-	Subcommand index

Response quick info

No.	Type	Value	Default	Description
arg0	int	2	-	Subcommand index
arg1	int	s32	-	Total operation time [s]

Description This command queries the total operation time of the device.

Good to know Command must be a query.

Examples

Input	Comment
OPD 2 ?	Returns total operation time [s].

2.42.3 OPD 3 (Number of power ups)

Command format OPD <arg0>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	3	-	Subcommand index

Response quick info

No.	Type	Value	Default	Description
arg0	int	3	-	Subcommand index
arg1	int	s32	-	Total number of power ups

Description This command queries the total number of power ups.

Good to know Command must be a query.

Examples

Input	Comment
OPD 3 ?	Returns total number of power ups.

2.42.4 OPD 5 (Uptime since last power cycle)

Command format OPD <arg0>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	5	-	Subcommand index

Response quick info

No.	Type	Value	Default	Description
arg0	int	5	-	Subcommand index
arg1	int	s32	-	Uptime of unit since last power on [s]

Description This command queries the uptime of the device since the last power on.

Good to know Command must be a query.

Examples

Input	Comment
OPD 5 ?	Returns uptime [s] since the last power on.

2.42.5 OPD 6 (Time stamp synchronization)

Command format OPD <arg0> <arg1> <arg2>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	6	-	Time stamp synchronization control
arg1	int	0, 1	0	Master function status: off (0) / on (1)
arg2	int	0, 1	0	Slave function status: off (0) / on (1)

Response quick info

No.	Type	Value	Default	Description
arg0	int	6	-	Time stamp synchronization control
arg1	int	0, 1	-	Master function status: off (0) / on (1)
arg2	int	0, 1	-	Slave function status: off (0) / on (1)
arg3	int	0, 1	-	Synchronization status: failure (0) / synchronized (1)
arg4	int	u16	-	Number of correctly received synchronization packets
arg5	int	u16	-	Number of “hard” synchronizations
arg6	int	u16	-	Number of erroneously received synchronization packets

Description This command controls and queries the time stamp synchronization.

Good to know A “hard” synchronization is executed if the time stamp of the slave differs too much from the master’s. In that case, the time stamp of the master replaces the time stamp of the slave. Otherwise, only the increment value of the time counter is adjusted.

Examples

Input	Comment
OPD 6 1 0	Activates the synchronization master function.
OPD 6 0 1	Activates the synchronization slave function.
OPD 6 ?	Returns the synchronization status: if master is active, if slave is active, if device time stamp is synchronized, how many packets were OK, how many “hard” synchronizations happened and how many packets were erroneous.

2.43 ORDB (Overlap region dual band)

Scope CHRocodile 2 SX only

Command format ORDB <arg0>

Argument quick info

No.	Type	Value	Default	Description
arg0	float	0; 1–100	5	Sets a value in µm for the center of the overlap region.

Description

The CHRocodile 2 SX device uses two light sources (LS1 and LS2) for the thickness measurement, one for thinner (LS1) and one for thicker materials (LS2). There is a certain thickness range where both light sources can generate valid results, the so called “overlap region”. (Note: There is no overlap in the wavelength bands of both light sources.)
The ORDB command controls how the result is generated in this overlap region. Below the overlap region, the thickness is obtained from the LS1 measurement. Above the overlap region, the thickness is obtained from the LS2 measurement. Inside the overlap region, a quality weighted average is applied on the detected peaks from both light sources.
By the arg0 value (e.g.: 5), a center for the overlap region is defined. Internally in CHRocodile 2, the begin and the end for the overlap region is calculated by using the fixed value 0.17 µm for silicon:

- Overlap region begin (e.g.: for arg0 = 5 µm): arg0 - 0.17 µm = 5 µm - 0.17 µm = 4.83 µm
- Overlap region end (e.g.: for arg0 = 5 µm): arg0 + 0.17 µm = 5 µm + 0.17 µm = 5.17 µm
- Resulting in the following overlap region: 4.83 µm – 5.17 µm.

Examples

Input	Comment
ORDB 5	Sets a value for the center of the overlap region of 5 µm.
ORDB 0	The overlap region is deactivated, all detected peaks from both light sources are output. (Resulting in a double output for thicknesses in a certain thickness range.)
ORDB ?	Displays the currently valid value of the center for the overlap region.

Related commands TRW (Trigger window)

2.44 POD (Peak ordering)

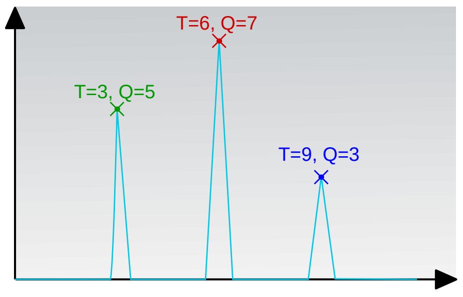
Scope	Interferometric mode only
Command format	POD <arg0>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0–2	0 (for CHR 2 SX: 1)	Peak order: 0: according to quality (peak quality descending) 1: according to position (ascending) 2: according to position (descending)

Description

This command sets the output peak order in interferometric mode (in confocal mode the output peak order is always according to increasing distance). If the value of <arg0> is 0, the peaks are sorted according to their peak quality, with the highest peak first. If <arg0> is 1, the peaks are sorted according to their position (i.e. layer thickness) in an ascending manner, with the thickest layer last. Conversely, if <arg0> is 2, peaks are sorted according to decreasing layer thickness.



POD setting	Peak 0	Peak 1	Peak 2

0	Thickness=6
	Quality=7
		Thickness=3
		Quality=5
			Thickness=9
			Quality=3

1	Thickness=3
	Quality=5
		Thickness=6
		Quality=7
			Thickness=9
			Quality=3

2	Thickness=9
	Quality=3
		Thickness=6
		Quality=7
			Thickness=3
			Quality=5

Good to know

For CHRocodile 2 SX: Peaks are detected in two different spectra from different light sources. The peak quality is calculated independently for each spectrum. Comparing peak quality between peaks from different spectra can lead to unexpected results. Therefore, POD 0 is not recommended; use POD 1 or POD 2 instead for this device.

Examples

Input	Comment
POD 0	Sets peak ordering according to the peak quality in a descending manner.
POD ?	Returns the current value.

Related commands SODX (Set output data extended)

2.45 PSM (Peak separation minimum)

Scope Chromatic confocal mode only

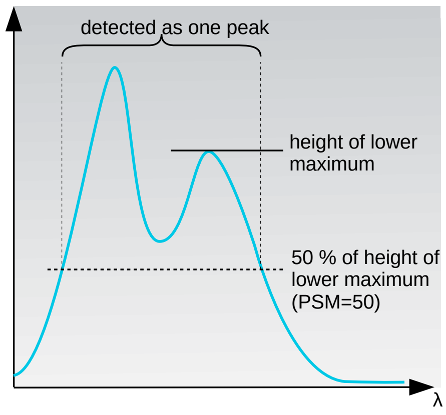
Command format PSM <arg0>

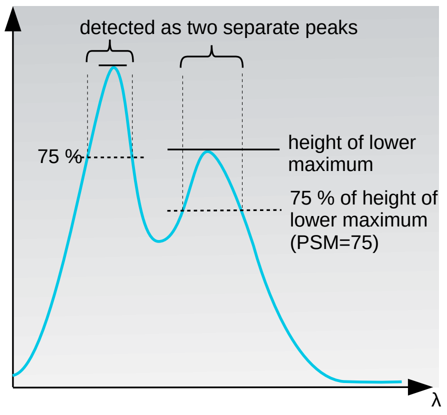
Argument quick info

No.	Type	Value	Default	Description
arg0	float	10–90	50	Peak separation minimum in percent of the height of the lower peak

Description

The peak detection algorithm detects 2 neighbouring peaks as separate peaks if the signal minimum between the two peaks is less than a certain fraction of the lower peak height. This fraction (PSM value) is given in percent of the height of the lower peak.




detected as two separate peaks
75 %
height of lower maximum
75 % of height of lower maximum (PSM=75)
λ

Examples

Input	Comment
PSM 60	Sets the threshold for signal minimum value between two peaks to 60 % of the lower peak.
PSM ?	Queries the current value of this parameter.

2.46 QTH (Quality threshold, interferometric mode)

Scope Interferometric mode only

Command format QTH <arg0> [<arg1>]

Argument quick info

No.	Type	Value	Default	Description
arg0	float	1–1000	15	Sets a quality threshold.
arg1	float	1–1000	15	Only on CHRocodile 2 SX: Sets a quality threshold for the second light source LS2. Note: If arg1 is not set, quality threshold from arg0 is set for both light sources.

Description

This command lets you specify a quality threshold for the peak detection. The threshold parameter is used to tune the peak detection algorithm to the desired sensitivity. It should be set as low as possible to be able to measure dark surfaces, but high enough to suppress the detection of noise spikes when there is no object in the detection range.

The threshold is in arbitrary units.

If the sensor doesn't detect a signal which passes the threshold, 0 is output for distance and intensity. However, this behavior doesn't disturb the averaging algorithm, as invalid results are excluded from averaging.

Good to know

For CHRocodile 2 SX, the command has been extended by a further argument for the second light source. If the second argument is not defined, quality threshold from the first argument is set for both light sources.

Examples

Input	Comment
QTH 30	Sets a quality threshold of 30.
QTH 30 50	Only on CHRocodile 2 SX: Sets a quality threshold of 30 for the lower band (first light source LS1) and 50 for the upper band (second light source LS2)..
QTH ?	Displays the currently valid quality threshold.

Related commands

THR (Threshold, confocal mode)
POD (Peak ordering)

2.47 RAF (Refractive adjustment factor)

Scope	Interferometric mode only
Command format	RAF <arg0>

Argument quick info

No.	Type	Value	Default	Description
arg0	float	0.001–50	1	Sets an adjustment factor to the thickness result.

Description

The RAF command applies a multiplicative adjustment factor to the thickness result, providing a possibility to the user to fine-tune the result.

Good to know

The refractive adjustment factor applies to all thickness values (Thickness 1, Thickness 2, Thickness 3 ...).

Examples

Input	Comment
RAF 1.002	Sets an adjustment factor to 1.002 to increase the result by 2‰.
RAF ?	Displays the currently valid refractive adjustment factor.

2.48 RST (Restart device)

Command format	RST
Argument quick info	No argument supported
Description	Reboots the CHRocodile device. This may be useful after a software update or a calibration in order to reinitialize the system completely without switching the power off and on again.

Examples

Input	Comment
RST	Reboots the CHRocodile.

2.49 SCA (Scale)

Command format SCA <arg0>

Response quick info

No.	Type	Value	Default	Description
arg0	int	u32	-	Full scale in micrometers

Description

For query only. Gives the full scale value for distances and thicknesses that are selected for transmission in 16 bit mode.

In 16 bit mode, the value is not transmitted in micrometer or nanometer but in normalized form. A distance value of 32768 would mean a distance of (Full Scale) micrometers in air. To convert the integer distance value (d) in air received from the serial interface to a value in micrometers (D), use Equation 2.2.

$$D[\mu m] = \frac{d[integer]}{32768} * FullScale \tag{2.2}$$

For thickness measurements the result has to be multiplied by the refractive index of the layer material.

Note for CHRocodile 2 SX: For the above formula (2.2) the full scale of LS2 is used.

Good to know

When using a refractive index table stored in the device (SRT different from 0), query the needed index from the device with the SRI ? command. When no table is used, the refractive index has to be provided by the user and must also be forwarded to the CHRocodile via the SRI command.

Examples

Input	Comment
SCA ?	Returns the scale in micrometers.

Related commands

ABE (Abbe number)
MMD (Measurement mode)
SRI (Set refractive indices)
SRT (Set refractive index table)

2.50 SCAN (Scanner control)

Scope Packet protocol only

Command format SCAN <arg0> ...

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0–11	-	Scanner subcommand index
...	...	...	...	Individual argument lists, see the following descriptions of subcommands for more information

Description

This command controls the scanner mode of CHRocodile 2 devices. It can be used to control the Flying Spot Scanner (FSS). In the scanner mode, a CHRocodile 2 device uses its 2 analog outputs (analog mode) or the Sync-in / Sync-out pins (digital mode) to control 2 galvanometer scan axes (output 1 to axis X, output 2 to axis Y). The control mode (analog/digital) can be switched using the SCAN 6 command. Scan paths are to be predefined (called scan program) and are either stored as file on the internal SD card of CHRocodile 2 devices or executed directly. A limited number of device parameters can also be embedded into the scan programs.

Good to know

In the scanner mode, the current scanner set position (the position sent to scanner) can be read back using the encoder channel 3 (signal ID 68). The 32-bit value has to be interpreted as an array of 2 signed 16-bit integers, where the least significant 16 bits represent the X value and the most significant 16 bits the Y value.

Depending on the scanner support, its current position can be read back using encoder channel 0 (X value = signal ID 65) and channel 1 (Y value = signal ID 66).

Encoder channel 4 (signal ID 69) outputs a marker value that is controlled by the scan program. It serves to easily attribute measured results to the different parts of the scan path.

Please ensure that no other signal is routed to those channels to avoid interference. Disabling the encoder counting for ENC 3 and ENC 4 is achieved by selecting an undefined count source index like 14 using ENC 3 1 14 and ENC 4 1 14.

Examples

Input	Comment
SCAN 0 ...	See SCAN 0 (Activate / deactivate scanner mode).
SCAN 1 ...	See SCAN 1 (Scanner gain value).
SCAN 2 ...	See SCAN 2 (Scanner offset value).
SCAN 3 ...	See SCAN 3 (Scanner direct positioning).
SCAN 4 ...	See SCAN 4 (Raster scan parameters).
SCAN 5 ...	See SCAN 5 (Scanner coordinate alignment).
SCAN 6 ...	See SCAN 6 (Scanner control mode).
SCAN 7 ...	See SCAN 7 (Scanner XY-correction).
SCAN 8 ...	See SCAN 8 (Parametrize external axis).
SCAN 9 ...	See SCAN 9 (FSS status query).
SCAN 10 ...	See SCAN 10 (Scanner Z-correction coefficients).
SCAN 11 ...	See SCAN 11 (Scanner positioning delay times).
SCAN 13 ...	See SCAN 13 (Polynomial representation).

Related commands TRW (Trigger window)

2.50.1 SCAN 0 (Activate / deactivate scanner mode)

Scope

Form 3 is only supported via binary packet protocol.

Command format

Form 1: SCAN 0
Form 2: SCAN 0 <arg1>
Form 3: SCAN 0 <arg1> <arg2> <arg3>

Argument quick info

Form 1 (disable scanner mode):

No.	Type	Value	Default	Description
arg0	int	0	-	Scanner control subcommand index

Form 2 (select recipe or direct control):

No.	Type	Value	Default	Description
arg0	int	0	-	Scanner control subcommand index
arg1	str	-	-	“direct” or “raster” or filename of scan program on SD card

Form 3 (live recipe):

No.	Type	Value	Default	Description
arg0	int	0	-	Scanner control subcommand index
arg1	int	-	-	Byte offset of current blob in complete scan program being sent
arg2	int	-	-	Complete size in bytes of scan program being sent
arg3	blob	-	-	Current blob of scan program being sent

Description

This subcommand allows the user to activate or deactivate the scanner mode of the device. There are 3 forms of this subcommand.

- Form 1: Without any additional argument, this command deactivates the scanner mode.
- Form 2: Given a single additional string argument, there are 3 possibilities:
  If the string is `direct`, the device enters the direct positioning mode, where the scanner position is controlled by the SCAN 3 (Scanner direct positioning) command.
  If the string is `raster`, the device prepares a raster scan that was previously parametrized by the SCAN 4 (Raster scan parameters) command. The raster scan can then be started with a trigger event or a STR (Set trigger) command. At the beginning of the raster scan, the value of encoder 3 (“Marker”) is set to 1, after the end of the scan it is set to -2 to indicate the end of the scan. In order to make use of these marker values, make sure that the encoder 3 does not count on an input signal by setting the encoder count source to “open” (ENC 3 1 14).
  If any other than the two above mentioned strings is passed, the device tries to load a scan program file with the name given by the string from the device’s SD card. After loading it successfully that scan program will be started.
- Form 3: External software may use the form 3 to send a scan program file to the device and activate it without saving it on the internal SD card. This mechanism is also known as live recipe. A blob (<arg 3>) must be less than 4 kbytes, which is a general limitation of the blob argument format. The scan program must thus be transmitted in several blobs if required by the total scan program size.

Examples

Form 1 and 2: Activate and deactivate scanner mode using an internal stored scan program.

Input	Comment
SCAN 0 scan.rec	Loads scan program file “scan.rec” from SD card and activates the scanner mode at the end.
SCAN 0 direct	Activates the direct positioning mode where the scanner position is controlled by the SCAN 3 command.
SCAN 0	Deactivates the scanner mode.

Form 3: Receive a 7000 bytes scan program from external software using 2 blobs with different lengths.

Input	Comment
SCAN 0 0 7000 blob	Loads the first blob of a scan program into device. In our example the blob length is 4096 bytes.
SCAN 0 4096 7000 blob	Loads the remaining blob (2904 bytes) of the scan program into device then activates the scanner mode.

Related commands SCAN (Scanner control)

2.50.2 SCAN 1 (Scanner gain value)

Scope Packet protocol only

Command format SCAN 1 <arg1> <arg2>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	1	-	Scanner control subcommand index
arg1	float	-1–1	1	Normalized gain value, X axis
arg2	float	-1–1	1	Normalized gain value, Y axis

Description

The normalized gain values specified using this subcommand are applied on the corresponding scanner coordinates that are stored in the scan program. Thus the defined scan path can be scaled up or down as needed.

This scaling operation is done after the internal coordination alignment defined by SCAN 5 but before the shifting operation defined by SCAN 2.

Good to know

Negative values of arg1 correspond to an inversion of the scanners's X axis. Negative values of arg2 invert the Y axis.

Rebooting the device resets the command to its default values.

Examples

Input	Comment
SCAN 1 1 1	Sets normalized gain value of both axes to 1.

Related commands

SCAN (Scanner control)
SCAN 2 (Scanner offset value)
SCAN 5 (Scanner coordinate alignment)

2.50.3 SCAN 2 (Scanner offset value)

Scope Packet protocol only

Command format SCAN 2 <arg1> <arg2>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	2	-	Scanner control subcommand index
arg1	float	-1–1	0	Normalized offset value, X axis
arg2	float	-1–1	0	Normalized offset value, Y axis

Description The normalized offset values specified using this subcommand are applied on the corresponding scanner coordinates that are stored in the scan program. Thus the defined scan path can be shifted as needed.

This shifting operation is done after the internal coordination alignment defined by SCAN 5 and the scaling operation defined by SCAN 1.

Good to know The position in mm corresponding to a normalized position x is calculated as:

$$x \cdot \frac{32768}{\mathrm{ScalingFactorRWto32k}} \tag{2.3}$$

The input range [-1,1] corresponds to an offset range that is larger than the field of the scanner. Thus the scan path may be clipped or shifted outside the field of the scanner by using SCAN 2.

Rebooting the device resets the command to its default values.

Examples

Input	Comment
SCAN 2 0.1 0.1	Sets normalized offset value of both axes to 0.1.

Related commands SCAN (Scanner control)
SCAN 1 (Scanner gain value)
SCAN 5 (Scanner coordinate alignment)

2.50.4 SCAN 3 (Scanner direct positioning)

Scope	Packet protocol only
Command format	SCAN 3 <arg1> <arg2>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	3	-	Scanner control subcommand index
arg1	float	-1–1	-	Normalized X position
arg2	float	-1–1	-	Normalized Y position

Description

Moves the measurement spot to the given coordination.

Before using SCAN 3, the scanner has to be set to direct positioning mode by sending SCAN 0 direct.

Good to know

The position in mm corresponding to a normalized position x is calculated as:

$$x \cdot \frac{32768}{\mathrm{ScalingFactorRWto32k}} \tag{2.4}$$

The input range [-1,1] corresponds to a coordinate range that is larger than the field of the scanner. The spot is not visible when it is moved to a position outside of the scanning field.

Examples

Input	Comment
SCAN 3 0 0	Moves the measurement spot to the center of the scan area.

Related commands

SCAN (Scanner control)

2.50.5 SCAN 4 (Raster scan parameters)

Scope Packet protocol only

Command format SCAN 4 <arg1> ...

Argument quick info

No.	Type	Value	Default	Description
arg0	int	4	-	Scanner control subcommand index
arg1	float	-1–1	-1	Start X coordinate
arg2	float	-1–1	-1	Start Y coordinate
arg3	float	-1–1	1	Stop X coordinate
arg4	float	-1–1	1	Stop Y coordinate
arg5	str	‘x’/‘y’	‘x’	Axis parallel to scan direction
arg6	int	u16	11	Number of scan lines
arg7	int	u32	1000	Number of (averaged) measurement results per line
arg8	int	1–100000	10000	Scanner frequency
arg9	int	0–2^{31}-1	1000	Wait time in µs at the beginning of each line
arg10	int	0–2^{31}-1	1000	Wait time in µs at the end of each line

Description

This function supports users to quickly scan a surface to gather information about distance or thickness. During a scan, the measurement spot moves along lines that are parallel either to the X axis or the Y axis from the start towards the stop corner.

SCAN 4 configures the scanning parameters. In order to perform a raster scan, the scanner has to be set to raster scan mode using SCAN 0 raster. The trigger command STR starts the scan.

The device shall only measure while the spot is on a line, i.e., it shall not measure while the spot is jumping from the end of the current line to the beginning of the next line. In order to do this, the command TRW must be set in advance.

Additionally, it is possible to configure a wait time in µs after the scanner reaches the start position and another wait time (also in µs) after the scanner reaches the end position.

Good to know

The position in mm corresponding to a normalized position x is calculated as:

$$x \cdot \frac{32768}{\mathrm{ScalingFactorRWto32k}} \tag{2.5}$$

The input range [-1,1] corresponds to a coordinate range that is larger than the field of the scanner. The scan area set by the command may be clipped by the available scanning field.

Measurement results can be mapped to lines using scanner markers.

Related commands

AVD (Data averaging)
AVS (Spectra averaging)
SCAN (Scanner control)
SHZ (Set sample frequency in Hz)
TRW (Trigger window)

2.50.6 SCAN 5 (Scanner coordinate alignment)

Scope

Packet protocol only

Command format

SCAN 5 <arg1> <arg2> <arg3>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	5	-	Scanner control subcommand index
arg1	float	full range	0	Rotation, given in *radian*
arg2	int	-32768–32767	0	Shifting value, X axis
arg3	int	-32768–32767	0	Shifting value, Y axis

Description

This command sets parameters of the scanner internal coordinate alignment, i.e. a rotation of the coordinate system around the origin and a shift of the coordinates. The shifting operation is performed after the rotation. The coordinate alignment is done before both the scaling and shifting operations defined by SCAN 1 and SCAN 2.

Good to know

The position in mm corresponding to a normalized position x is calculated as:

$$\frac{x}{ScalingFactorRWto32k} \tag{2.6}$$

The command is reset to its default values when rebooting the device.

Related commands

SCAN (Scanner control)
SCAN 1 (Scanner gain value)
SCAN 2 (Scanner offset value)

2.50.7 SCAN 6 (Scanner control mode)

Scope Packet protocol only

Command format SCAN 6 <arg1>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	6	-	Scanner control subcommand index
arg1	int	0, 1	0	Scanner type: 0: analog 1: digital

Description

This command switches between two scanner control modes: analog and digital. Value 0 means analog, value 1 means digital. In analog control mode, the scanner position is determined by the analog output voltage and the actual position is fed back by the X and Y encoder inputs. In digital scanner mode, both scanner set point position transmission and position feedback are accomplished via a highspeed serial link.

The scanner control mode must be set before entering scan mode. Changes of the mode during scan mode active will not be recognized!

Good to know

In the analog control mode, both of the normal CHRocodile analog outputs are used by the scanner.

In the digital control mode, the Sync-in and Sync-out pins are occupied and thus can not be used as a trigger source. Instead, the differential A0+/- pins of the encoder interface are used for triggering. The time stamp synchronization feature also needs these two pins for its communication, thus no time stamp synchronization is possible in the digital scanner control mode.

Related commands

ANAX (Analog output function, extended)
OPD (Operation data)
SCAN (Scanner control)

2.50.8 SCAN 7 (Scanner XY-correction)

Scope Packet protocol only

Command format Form 1: SCAN 7 <arg1> to <arg30>
Form 2: SCAN 7 str

Argument quick info Form 1 (classic form, providing up to 30 specific polynomial coefficients):

No.	Type	Value	Default	Description
arg0	int	7	-	Scanner control subcommand index
arg1	float	full range	0	XY-correction coefficient no. 1
...	...	...	...	...
arg30	float	full range	0	XY-correction coefficient no. 30

Form 2 (extended command):

No.	Type	Value	Default	Description
arg0	int	7	-	Scanner control subcommand index
arg1	str	-	-	"LUT", "POLY" or "NONE" as correction types for XY-correction, see description below for details.

Description This command sets XY-coefficients of the XY-correction. There are two forms of this subcommand:

- Form 1: Following the subcommand index, 30 float values are listed, which are separated by space.
- Form 2: Furthermore, there is an extended form of the command that allows to select the dedicated correction types as follows:

Command	Description
SCAN 7 LUT	Uses an look-up-table ("LUT") for correction of XY-positions. This look-up-table is more precise than using polynomial coefficients and therefore preferable especially for FSS310.
SCAN 7 POLY	Re-activates form 1 of the command using the 30 polynomial coefficients. Initially, on re-activation, the coefficients stored in the FSS memory are automatically loaded and applied.
SCAN 7 NONE	Resets the polynomial coefficients to the factory default settings ("001001"). This will disable the XY-correction.

Good to know A connected FSS is initialized when entering the scanning mode, loading stored data from the FSS memory and caching it on CHRocodile 2. This data is valid throughout the connection between the CHRocodile 2 and the FSS and is cleared from the cache if the scanning mode is quit or the FSS connection is lost.

Examples

Input	Comment
SCAN 7 0 0 1 0 0 1 ...	Sets XY-correction coefficients.
SCAN 7 ?	Queries active XY-correction coefficients. Note: Stored scanner coefficients will be loaded on detection of the connected FSS.

Related commands SCAN (Scanner control)

2.50.9 SCAN 8 (Parametrize external axis)

Command format SCAN 8 <arg1> <arg2> <arg3> <arg4>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	8	-	External axis subcommand index
arg1	float	full range	0.0	factorX = delta scanner X increments/delta external axis increments
arg2	float	full range	0.0	factorY = delta scanner Y increments/delta external axis increments
arg3	int	-32768–32767	32767	AutoScannerWaitPos (scanner wait position)
arg4	int	0, 1	0	ExtAxisMovesNegative. If the encoder counter decreases when the sample moves on the external axis under the scanner during the measurement, this parameter must be set to 1, otherwise 0.

Description

SCAN 8 serves to parametrize the scanner use together with an external axis that moves the sample. Because the set point values for the scanner are calculated in real time as the sum of the external axis position and the recipe positions, both positions must have the same dimension.
For this purpose, the two first parameters in the SCAN 8 command (<arg1> and <arg 2>) define two factors that convert external axis encoder counts into scanner encoder counts. The first factor is for the scanner X direction, the second factor is for the scanner Y direction. Definition of the factors:
factorX: (delta scanner X increments)/(delta external axis increments)
factorY: (delta scanner Y increments)/(delta external axis increments)

Experimentally, the two factors can be determined using the following procedure:

1. Place a sample with an easily visible mark (“feature”, like a small hole or a corner) on the external axis under the scanner.
2. Reset the position of the external axis counter to 0 by issuing the command:
   ENC 2 0
3. Switch off the axis position compensation by setting factorX and factorY to 0 with the command:
   SCAN 8 0 0 32767 0
4. Position the aiming laser on the feature (see step 1) by issuing the commands:
   ```
   SCAN 6 1          // (digital scanner)
   SCAN 0 direct     // (direct positioning using SCAN 3 command)
   repeat SCAN 3 X Y // (refine X,Y values until aiming laser hits feature exactly)
   ```
5. Note the X and Y values determined in the previous step as X1, Y1.
6. Move the external axis slightly (half the scan area). Note the external axis value.
7. Repeat step 4 (you can omit SCAN 6 and SCAN 0 direct commands as you are already in direct positioning mode).
8. Note the X and Y values as X2, Y2.

9. Calculate factorX as (X2-X1)/(Z2-Z1)*32768
   Calculate factorY as (Y2-Y1)/(Z2-Z1)*32768
   where Z is the encoder count and Z1 = 0.

   Example:
   X2=-0.1 , X1=0.3
   Y2=Y1=0
   Z2=20000 (20000x 2 µm increments, yields 40 mm), Z1=0

   Result:
   factorX is (-0.1-0.3)/(-2000-0)*32768=0.65536
   factorY is 0.
   Note: The factors might be negative (if positive direction of scanner coordinates and positive external axis direction are opposite).
10. Enter the command:
    SCAN 8 factorX factorY 32767 ExtAxisMovesNegative
    (in the example above: SCAN 8 0.65536 0 32767 1)
11. Observe the movement of the aiming laser spot when you move the external axis. It should follow the feature.

Good to know

- In order to function precisely, the scanner has to be X/Y calibrated. Because otherwise, the actual size of the scanner X and Y increments depends (slightly) on the current scanner position. In that case, axis and scanner positions cannot be added together correctly.
- The factors are stored in the non-volatile memory after issuing an SSU command (Save setup).

Related commands

SCAN (Scanner control)
SCAN 0 (Activate / deactivate scanner mode)
SCAN 2 (Scanner offset value)
SCAN 3 (Scanner direct positioning)
SCAN 6 (Scanner control mode)

2.50.10 SCAN 9 (FSS status query)

Scope	Packet protocol only
Command format	SCAN 9

Argument quick info

No.	Type	Value	Default	Description
arg0	int	9	-	Scanner control subcommand index

Response quick info

No.	Type	Value	Default	Description
arg0	strg	-	-	FSS status

Description

Queries the FSS status and FSS personality, which outputs to console as follows:

- Firmware version
- Down counter channel 0
- Down counter channel 1
- Shutter state (optional according to hardware)
- Configuration mode (optional according to hardware)
- Operation time in hours (optional according to hardware)
- Creation date for the stored scanner correction coefficients (optional according to hardware)
- FSS personality consisting of FSS type, positioning range, production date, serial number, item number and FSS revision

Must be a query, so use SCAN 9 ? to output the properties mentioned above.

Note: A transmission error is issued if no (stable) connection between CHRocodile and FSS is possible. The dedicated transmission error can be queried by GLE command.

Examples

Input	Comment
SCAN 9 ?	Queries FSS status and FSS personality. Up to three different hardware versions can be detected on connection. Unique IDs are attached to the response of a detected hardware, which are described as follows: 127[NewsonFirmwareVersion $\leq$ 1.2]: Cyclone10LP hardware detected, which supports the tuneset-feature, handling scanner correction coefficients and firmware remote update. 127[NewsonFirmwareVersion $\geq$ 1.3]: Additionally supports FSS parameter and FSS personality. 126: Max10 hardware detected, which supports the tuneset-feature. 125: Max10 hardware detected (no tuneset-feature). 124: A transmission error occurred: no (stable) connection between CHR2 and FSS. 123: The FSS hasn´t been initialized!

Related commands SCAN (Scanner control)

2.50.11 SCAN 10 (Scanner Z-correction coefficients)

Scope	Packet protocol only
Command format	SCAN 10 <arg1> to <arg45>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	10	-	Scanner control subcommand index
arg1	float	full range	0	Z-coefficient no. 1
...	...	...	...	...
arg45	float	full range	0	Z-coefficient no. 45

Description

This command serves to set Z-coefficients of the Z-correction. For distance measurements, the different paths of the light beam through the telecentric lens cause a U-shaped height distortion. The polynomial Z-correction eliminates this distortion.
Following the subcommand index, 45 float values are listed, which are separated by space.

Good to know

A connected FSS is initialized when entering the scanning mode, loading stored data from the FSS memory and caching it on CHRocodile 2. This data is valid throughout the connection between the CHRocodile 2 and the FSS and is cleared from the cache if the scanning mode is quit or the FSS connection is lost.

Examples

Input	Comment
SCAN 10 0.01 0.02 0.03 0.04 0.05 ...	Sets Z-correction coefficients.
SCAN 10 ?	Queries active Z-correction coefficients. Note: Stored scanner coefficients will be loaded on detection of the connected FSS.

Related commands

SCAN (Scanner control)

2.50.12 SCAN 11 (Scanner positioning delay times)

Scope Packet protocol only

Command format SCAN 11 <arg1> <arg2>

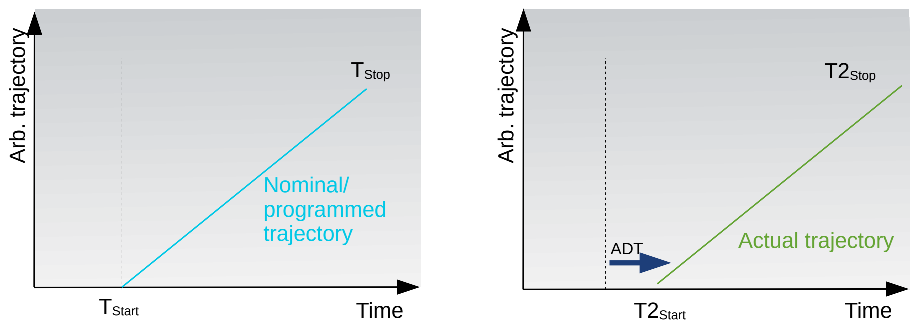
Argument quick info

No.	Type	Value	Default	Description
arg0	int	11	-	Scanner control subcommand index
arg1	int	0–1023	150	Active Delay Time [µs]
arg2	int	0–1023	25	Real Position Measurement Delay Time [µs]

Description

Active Delay Time
Active Delay Time is related to the filtering algorithm implemented in the Flying Spot Scanner. The filtering algorithm introduces a delay in the response to control positions, the so-called Active Delay Time. The filtering is necessary to achieve smoother and more stable movements of the galvo mirrors of the Flying Spot Scanner. Because we want to start the CHRocodile data acquisition exactly on the *actual* start of the scanner movement and stop it exactly at the *actual* end of the movement, the start and stop of the measurement have to be delayed with the same amount as the filtering delay with respect to the command.

For example, the user programs a trajectory that commences at T_{Start} and concludes at T_{Stop} (left graph). However, the actual trajectory experiences delay attributable to the filtering algorithm integrated into the Flying Spot Scanner. Consequently, the actual starting time becomes $T2_{Start} = T_{Start}$ + Active Delay Time (ADT). Similarly, the end of the trajectory is at $T2_{Stop} = T_{Stop}$ + ADT. These relationships are illustrated in the following figure:



Procedure to determine the Active Delay Time:

1. Create a scan program which contains a horizontal line from the left to the right and another one, vertically shifted, backwards.
2. Set the CHRocodile measuring rate and the scanner sample rate as high as possible, for example 70 kHz.
3. Perform the scan. Check the acquired actual positions compared to the programmed positions.
4. Change the Active Delay Time and repeat the scan. Set the parameter such that the acquired actual positions overlap with the programmed positions.

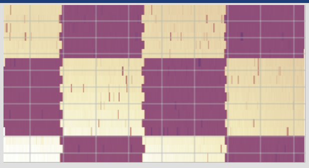
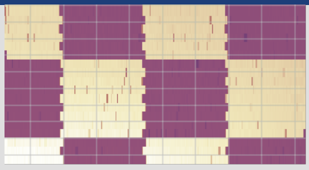
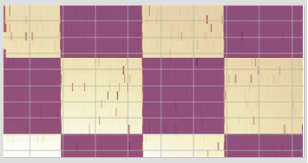
Note that the Real Position Measurement Delay Time should be determined first in case the user wants to perform bi-directional scans because the Real Position Measurement Delay Time might influence the Active Delay Time via the post processing as well.

Real Position Measurement Delay Time
The current actual scanner position is acquired by the scanner and transmitted to the CHRocodile sensor. At the beginning of each exposure, the CHRocodile latches the current scanner position and attributes it to the sample being exposured. Due to the acquisition and transmission delays of the actual position from scanner to CHRocodile, the latched position has in fact been acquired earlier. This effect is directly visible, for example when scanning a trajectory in both directions. The error occurs in direction of movement. The parameter Real Position Measurement Delay Time is used to compensate for this effect.

Procedure to determine the Real Position Measurement Delay Time:

1. Set the CHRocodile measuring rate and the scanner sample rate as high as possible, for example 70 kHz.
2. Perform a bi-directional raster scan on a target with vertical lines, e.g. a checkerboard target like in the illustration below. Set the number of lines to e.g. 20 lines and the number of points per line to e.g. 400. The selected area of the raster scan should be small (15 mm x 10 mm for FSS 310).
3. Execute the scan and check the intensity pattern for a zig-zag pattern. Adjust the Real Position Measurement Delay Time and repeat the scan.
4. Set the parameter such that the acquired actual positions align with the programmed positions and the zig-zag pattern vanishes.

Scan result with a Real Position Measurement Delay Time of . . .

0 µs	**50 µs**	**22 µs**
		
–	–	✓
Each odd line is shifted to the right and each even line is shifted to the left, respectively.	Each odd line is shifted to the left and each even line is shifted to the right, respectively.	Real positioning measurement delay is compensated. No zig-zag pattern observable anymore.

Note that determining the Real Position Measurement Delay Time should precede any attempts to perform bi-directional scans. This is important because the Real Position Measurement Delay Time might affect the Active Delay Time through post-processing procedures as well.

Note that this parameter is neither processed by the CHRocodile sensor nor by the Flying Spot Scanner. It is stored in the FSS memory and can only be read out for post-processing purposes only.

Good to know

A connected FSS is initialized when entering the scanning mode, loading stored data from the FSS memory and caching it on CHRocodile 2. This data is valid throughout the connection between the CHRocodile 2 and the FSS and is cleared from the cache if the scanning mode is quit or the FSS connection is lost.

Examples

Input	Comment
SCAN 11 122 24	Aligns the actual positions with the measurement data; Active Delay Time 122 µs, Real Position Measurement Delay Time = 24 µs.
SCAN 11 ?	Queries scanner positioning delay time settings.

Related commands SCAN (Scanner control)

2.50.13 SCAN 13 (Polynomial representation)

Scope	Packet protocol only
Command format	SCAN 13 <arg1>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	13	-	Scanner control subcommand index
arg1	str	-	-	ASCII-string, including the Flying Spot Scanner calibration data. See description below for details.

Description

SCAN 13 was implemented to have an ASCII-based, human readable form for polynomial calibration, including flexible meta information. The string consists of two parts, the header and the content with the XYZ-coefficients.

The header must contain the following key-value-pairs and separators:

```
[x type=poly; dir=RWTo32k; deg=4]
```

The provided XYZ-coefficients are given as special key value pairs of the form:

```
<i>,<j>:<coeff_ij>
```

Example:

```
[x type=poly; dir=RWTo32k; deg=4]
0,0:1.23
1,0:5.6
0,1:5.7
1,1:5.8
2,1:5.9
1,2:6.7
...
0,4:0
```

Rules:
The structure for the header of XYZ-coefficients and the coefficient-values must be entered exactly as shown above to be recognized by a CHRocodile 2. In the header, the first entry is “x” for X-coefficients, “y” for Y-coefficients and “z_offset” for Z-coefficients. For XY-coefficients, only the polynomial order 4 (deg = 4) is supported, for z-coefficients 4, 6 and 8. Note that every line of both header and content have to be finalized with carriage return / line feed (DOS) or line feed (Unix).

XYZ-coefficients can be uploaded individually for X, Y and Z or as a complete XYZ-package.

Good to know

- Make sure that Digital Scan Mode has been activated once using SCAN 6 1.
- Make sure that SCAN 7 POLY is active.

Examples

Input	Comment
SCAN 13 <ASCII-string>	Uploads a Flying Spot Scanner calibration.
SCAN 13 ?	Downloads a calibration of a connected Flying Spot Scanner.

Related commands

SCAN (Scanner control)
SCAN 6 (Scanner control mode)

SCAN 7 (Scanner XY-correction)
SCAN 10 (Scanner Z-correction coefficients)

2.51 SEN (Select chromatic calibration)

Command format SEN <arg0>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0–15	0	Index of currently used chromatic calibration table

Description This command selects the chromatic calibration table by its index on the device.

Good to know For exact measurements, assure that the calibration table selected by this command matches the probe serial number(s) as all probes are individually calibrated! If you are not sure about the serial number of the calibration table, use the SENX ? command which outputs more information.

Examples

Input	Comment
SEN 0	Selects the chromatic calibration table 0 with nominal measuring range.
SEN ?	Queries which calibration is active and if the extended measuring range is active.

Related commands SENX (Extended chromatic calibration table query)

2.52 SENI (Sensor interferometric)

Scope

Only for CHRocodile 2 LR, CHRocodile 2 LR Detector V2, CHRocodile 2 IT DW 250 70 kHz, CHRocodile 2 IT DW 500 70 kHz, CHRocodile 2 IT DW 250 V2, CHRocodile 2 IT DW 500 V2 and Firmware $\geq$ R1.4.1

Command format

SENI <arg0>

Argument quick info

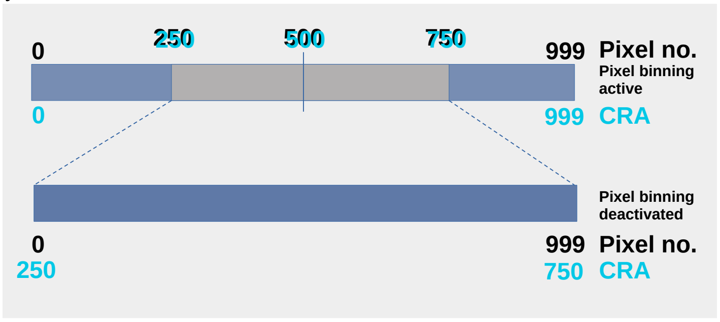
No.	Type	Value	Default	Description
arg0	int	0, 1	0	SENI 0: Switches standard interferometric measuring range on (pixel binning on) SENI 1: Switches extended interferometric measuring range on (pixel binning off).

Description

In the interferometric measuring mode the SENI command allows the user to extend the standard measuring mode. In detail:
SENI 0 sets the standard interferometric measuring mode (see topic Technical Data in User Manual)
SENI 1 switches to the extended interferometric measuring mode:

- Number of active pixels is halved.
- Measuring range is doubled.
- Minimum measurable thickness is doubled.
- Signal to noise ratio decreases.

The SENI command activates different areas of the detector pixel line and deactivates/activates pixel binning. Thus, the active detector range (CRA) changes accordingly:



SENI 0 (standard interferometric measuring range): CRA command: parameter limits are 0–999 (default) which corresponds to 1000 pixels with pixel binning mode active.

SENI 1 (extended interferometric measuring range): The active detector area is reduced to the pixels 250–750 of the standard measuring range. As a consequence, the parameters of the CRA command are restricted to CRA 250–750. In order to equalize the number of pixels to the standard interferometric measuring range, the pixel binning is deactivated. Thus, for both measuring ranges (SENI 0 and SENI 1) the same number of total pixels are available, namely 1000.

Good to know

- When changing from SENI 0 to SENI 1 (or vice versa), a dark reference (DRK) must be carried out anew.
- When changing from SENI 1 to SENI 0, the active detector range must be set anew (CRA).

- When using the extended interferometric measuring range (SENI 1), a higher signal noise must be considered.
- Note that binning in confocal mode is always activated. Thus, if you are in SENI 1 mode and then switch to confocal mode, a new dark reference has to be carried out (and also when switching back to the interferometric mode).
- When using CHRExplorer or myCHRocodile: Note that the representation of the active detector range is not correct in the spectrum view.

2.53 SENX (Extended chromatic calibration table query)

Command format SENX [<arg0>]

Argument quick info

No.	Type	Value	Default	Description
arg0	str	enum	-	Indicates that properties of all calibrated optical probes shall be queried.

Response quick info

Response on SENX ?

No.	Type	Value	Default	Description
arg0	str	-	-	Properties of the current optical probe

Response on SENX enum ?

No.	Type	Value	Default	Description
arg0	str	enum	-	Indicates that properties of all optical probes will follow.
arg1	str	-	-	Properties of the first calibrated optical probe
arg2	str	-	-	Properties of the second calibrated optical probe
...	...	...	...	Repeats until all calibrated optical probes are displayed.

Description

Must be a query. Use SENX ? for querying active optical probe properties.
If called with the string parameter “enum” (SENX enum ?), properties of all calibrated optical probes are returned in a list of strings.
In the dollar protocol, the strings are separated by “,”.

Each response string is composed as follows:

1. Tableindex, followed by a “,”
2. “`SNr: `”, followed by the probe serial number
3. “`Range: `”, followed by the measuring range in micrometers, followed by “`µm`”

Examples

Input	Comment
SENX ?	Gets properties of the current chromatic calibration table of a single channel device, for example: `0, SNr:  200001, Range:  2291 µm`
SENX enum ?	Gets properties of all chromatic calibration tables of a single channel device, for example: `0, SNr:  200001, Range:  2291 µm; 1, SNr: 200002, Range:  2320 µm`

Related commands SEN (Select chromatic calibration)

2.54 SFD (Set factory defaults)

Command format SFD [<arg0>]

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0, 1	0	Whether network settings shall be reset

Description This command resets all device parameters to their factory default values. The only exception are the network settings. They will not be reset, if no argument or "0" is given together with this command. Only SFD 1 will also reset those network settings.

Good to know This command triggers the device to send out update packets of all parameter settings (similar to the update packet burst at the establishment of a new connection or after sending a CONF command).

Examples

Input	Comment
SFD	Sets parameters to factory default values except the network settings.
SFD 0	Sets parameters to factory default values except the network settings.
SFD 1	Sets parameters to factory default values including the network settings.

Related commands SSU (Save setup)

2.55 SHZ (Set sample frequency in Hz)

Command format SHZ <arg0>

Argument quick info

No.	Type	Value	Default	Description
arg0	float	Detector specific	4000	Sample frequency in Hz

Description

Using this command, users can set the sample rate to an arbitrary value in Hz. Every value between a lower boundary given by the parasitic light during dark reference and the upper boundary given by the detector type may be specified.

The upper boundary can be determined by issuing IDE and looking for the value of the key `personality.caps.max_frequency`.

Due to the nature of the internal time base, not every sample rate can be realized exactly. The exact frequency to which the sample rate has been "rounded" can always be queried with SHZ ?. The SHZ command response as well as its command updates also contains this information.

Good to know

If you don't intend to use the double exposure mode, check that the duty cycle setting (DCY) is 100 % (or simply set it to 100 %).

Examples

Input	Comment
SHZ 400	Sets the sample frequency of the device in Hz.
SHZ ?	Returns the current value.

Related commands IDE (Identification)

2.56 SOD (Set output data)

Scope	Dollar protocol only, for backwards compatibility
Command format	SOD [<arg0> to <arg15>]

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0, 1	-	Enables (1) or disables (0) signal 0.
arg1	int	0, 1	-	Enables (1) or disables (0) signal 1.
arg2	int	0, 1	-	Enables (1) or disables (0) signal 2.
...	int	0, 1	-	...
arg15	int	0, 1	-	Enables (1) or disables (0) signal 15.

Description

This command is for dollar protocol backwards compatibility only. Only a subset of the result signals can be selected by this command. For the 16 possible 16 bit data words, 1 selects the word for transmission, 0 deselects the word. When less than 16 parameters are sent, the words with higher indices will not be included in the telegram.

A table describing the 16 data words can be found in Section 3.5.

Good to know

Do not use both SOD and SODX! Only use SOD for compatibility reasons. SODX should be used if possible.

Examples

Input	Comment
SOD 1 0 0 1	Includes the output words Distance 1 and Intensity 1 in the output telegram.
SOD ?	Returns the current values.

Related commands

SODX (Set output data extended)

2.57 SODX (Set output data extended)

Scope For packet protocol, the sequence of signals in the response may be reordered. For dollar protocol, the data will be sent in the same order as SODX input.

Command format SODX [<arg0> <arg1> <arg2> . . .]

Argument quick info

No.	Type	Value	Default	Description
arg0	int	u16	-	Signal ID to be output
. . .	. . .	. . .	. . .	Up to 32 signal IDs

Description This command selects the signal IDs that will be included in the output packet. This setting is specific to each individual client. Learn more about signal IDs in chapter 3 CHRocodile Signal IDs.

Good to know

- Do not use both SOD and SODX! Use only SODX and new signal ID definitions if possible (as outlined in chapter 3 CHRocodile Signal IDs).
- SODX without an argument lets the device send data packets with an empty payload. In order to stop the output data stream, use STO instead.
- Data transmission starts upon client connection. By factory default, the signal IDs 256 and 257 are transmitted (Distance 1 and Intensity 1).
- It is possible to choose a new signal set by means of SODX and store it persistently using SSU. Thus, the most recent signal set stored by a client will automatically be applied when a client connects.
- For CHRocodile 2 SX, the full scale of LS2 is used. Thus, the obtained thickness from integer values might have a lower resolution than the one obtained from float values. For this reason, the integer signals IDs (16***) should not be used with the CHRocodile 2 SX. Use the float signal IDs instead, for example 256 for Thickness 1.

Examples

Input	Comment
SODX 83 256 257	Selects sample counter, first detected distance and intensity for output packets.
SODX ?	Queries the currently active signal IDs.

Related commands STO (Stop data)

2.58 SRI (Set refractive indices)

Command format SRI <arg0> <arg1> ...

Argument quick info

No.	Type	Value	Default	Description
arg0	float	1.0–5.0	1	Refractive index of layer 1
arg1	float	1.0–5.0	1	Refractive index of layer 2
...	...	...	...	As many as number of layers (NOP - 1)

Description

Command to correct display of thickness and correct dispersion model. The parameter is given in floating point format.

In order to obtain correct thickness values, the thickness results have to be multiplied (or divided in the case of interferometric measurements) by the refractive index in the user application according to the formula specified in the description of the SCA command. The SRI setting on the CHRocodile is responsible for the dispersion correction and a correct absolute value on the display. The thickness and position output values on the analog outputs and the serial interface are still normalized to the respective full scale value and are only slightly affected by this parameter through the dispersion correction function (Abbe number).

Good to know

- Changing the refractive index may fail in interferometric mode (if the dispersion model calculation fails). Monitor the response in order to detect this situation and try other settings for CRA.
- In interferometric mode, only one refractive index value will be applied.
- In chromatic confocal mode, you should give as many refractive indices as there are layers to be measured, that is (number of peaks - 1).
- If the number of arguments sent is lower than the number of activated peaks, remaining SRI arguments are set to default value (1).

Examples

Input	Comment
SRI 1.2 1.3 2.1	Sets the refractive index for three layers.
SRI ?	Returns the current value(s).

Related commands

ABE (Abbe number)
CRA (Set active detector range)
NOP (Number of peaks)
SCA (Scale)
SRT (Set refractive index table)

2.59 SRT (Set refractive index table)

Command format SRT <arg0> <arg1> to <arg14>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0–16	0	Refractive index table of layer 1
arg1	int	0–16	0	Refractive index table of layer 2
...	...	...	...	As many as number of layers (NOP - 1)

Description

Instead of modeling the refractive index vs. wavelength function by a rather simple model based on n_d and the Abbe number ν_d, the CHRocodile offers up to 16 different user definable dispersion tables. These dispersion tables are stored in the non-volatile memory. With the SRT command, one of these tables can be activated (the table corresponding to the parameter is selected).
The parameter value 0 deselects any refractive index tables and instead enables the dispersion model based on n_d and the Abbe number ν_d.
If a selected table is not filled with valid data, the CHRocodile will default to the n_d/ν_d dispersion model. The user has to interpret the response of the CHRocodile to the SRT command in order to know if the selected setting was accepted and applied.
After the command the CHRocodile responds with the active table index and its name.
When the thickness is output in the normalized integer format (16 or 32 bit), then the thickness output values are normalized to a fixed reference refractive index value (as with the n_d/ν_d dispersion model, where n_d is the reference refractive index). This reference refractive index value is part of the table and has to be queried by the SRI ? command in order to scale the output values correctly.

Good to know

- In interferometric mode, only one refractive index table will be applied.
- In chromatic confocal mode, you should give as many refractive indices as there are layers to be measured, that is (number of peaks - 1).
- If the number of arguments sent is lower than the number of activated peaks, remaining SRT arguments are set to default value (0).

Examples

Input	Comment
SRT 3 1 2	Sets three refractive index tables by index.
SRT ?	Returns the current value(s).

Related commands

ABE (Abbe number)
NOP (Number of peaks)
SRI (Set refractive indices)

2.60 SSQ (Synchronization sequence)

Scope	Dollar protocol only
Command format	SSQ <arg0> <arg1>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	u8	255	Synchronization sequence byte 1
arg1	int	u8	255	Synchronization sequence byte 2

Description

Allows the user to set a customized telegram start sequence (instead of the default $FFFF) in dollar protocol and binary data mode. The 2 bytes immediately following the command will be used to indicate the beginning of every new data telegram (in binary mode). These bytes must follow the command directly with no separation character in between and must not be sent in hexadecimal notation.

There can be a permanent ambiguity about the start of the telegram when using this default sequence and the last telegram value is an intensity and the CHRocodile is in saturation. Under these circumstances, external data acquisition software will not be able to synchronize safely and it is good practice to change the synchronization sequence.

Good to know

A custom synchronization sequence will not be saved in non-volatile memory even after issuing an SSU command.

Examples

Input	Comment
SSQxy	Sets the new telegram synchronization sequence to xy.

2.61 SSU (Save setup)

Command format	SSU
Argument quick info	No argument supported
Description	Saves the current setting to non-volatile memory. The sensor will start on the next power up with the saved configuration.

2.62 STA (Start data)

Command format	STA
Argument quick info	No argument supported
Description	Starts output data stream. This command only applies to the connection through which this command was received. E.g., if the command was received from the serial interface, the serial data output is started.
Good to know	Use STO to stop output data stream.
Related commands	STO (Stop data)

2.63 STO (Stop data)

Command format	STO
Argument quick info	No argument supported
Description	Stops output data stream. This command only applies to the connection through which this command was received. E.g., if the command was received from the serial interface, the serial data output is stopped. Note that with this command, only the output of measurement data is stopped, the device still performs measurements.
Good to know	Use STA to start output data stream.
Related commands	STA (Start data)

2.64 STR (Set trigger)

Command format STR [<arg0>] [<arg1>]

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0, 1, 2	0	0 = Sets the software trigger state to low. 1 = Sets the software trigger state to high. 2 = Activates pull-up/pull-down parametrizing mode (see arg1). Without parameter = Generates trigger event
arg1	int	0, 1	0	Only if first argument is 2: Specifies the pull-up/pull-down behaviour to be applied to the Sync-in trigger signal. 0 = Enables pull-up. 1 = Enables pull-down.

Description

There are different forms of this command: without parameters or with parameter(s).

If used without parameter, a single "software" trigger event is generated.

If issued with one parameter, the command sets the software trigger state according to the parameter value. For use with TRG and TRE: If the software trigger state is set to 0, it triggers on the rising edge of the Sync-in signal. If the software trigger state is set to 1, it triggers on the falling edge of the Sync-in signal. For use with TRW: If the software trigger state is set to 0, the active level for TRW corresponds to a voltage level above the trigger threshold. If it is set to 1, the active level for TRW corresponds to a voltage level below the trigger threshold.

This parameter is not stored to non-volatile memory and always initializes with 0 after power up.

If issued with two parameters, the second parameter (<arg1>) specifies the pull-up/pull-down behaviour to be applied to the Sync-in trigger signal. Using the command STR 2 0 will enable pull-up (which is the default), STR 2 1 will enable pull-down. The pull-up/pull-down behaviour as well as the trigger behavior settings will be saved in non-volatile memory when issuing the command SSU (Save setup). Note: This feature is not available for older devices. Please contact our Support team for details.

Related commands

TRG (Trigger once)
TRE (Trigger each)
TRW (Trigger window)

2.65 TABL (Table handling)

Scope Packet protocol only

Command format TABL <arg0> <arg1> <arg2> <arg3> [<arg4>]

Argument quick info

No.	Type	Value	Default	Description
arg0	int	1–15	-	Table type or ID
arg1	int	0–16	-	Table index
arg2	int	u32	-	Offset in bytes of <arg4> within the complete table
arg3	int	u32	-	Table total size in bytes
arg4	blob	-	-	Fragment of table data in binary format

Description

This command is used to upload (or download in case of TABL <arg0> ?) various binary data blocks, e.g., calibration or spectral correction tables.

Tables that are larger than 4096 bytes have to be split in chunks of 4096 bytes or smaller. Each of the chunks shall be uploaded or downloaded by a separated TABL command. Note the correct ordering of those commands. No other command is allowed in between.

The blob argument <arg4> must be given if an upload is intended. The corresponding command response will not reflect that blob back to a client. In case of a download, a given <arg4> in the command will be simply ignored. The binary data is contained in the blob argument of the command response.

Available table types are specified in the following table:

ID	Indices	Bytes	Query	Name
1	0–15	4096	Yes	Confocal calibration
2	0	4096	Yes	Interferometric calibration
3	1–16	420	Yes	Refractive index table
4	0	2048	Yes	Constant dark correction
7	0	2048	Yes	Fourier amplifier table
9	0	2048	Yes	Exposure dark correction
10	0	2048	Yes	Exposure lighting dark correction
11	0	2048	Yes	White correction
13	0	2048	No	Simulated input spectrum

Further details on selected tables can be found in Appendix A.1 and A.2.

Good to know

Accidental erroneous application of this command may lead to incorrect adjustment of the device (e.g., by overwriting the calibration). To prevent property damage when uploading tables, please contact Precitec Optronik for further information.

2.66 THR (Threshold, confocal mode)

Scope	Chromatic confocal mode only
Command format	THR <arg0>

Argument quick info

No.	Type	Value	Default	Description
arg0	float	0–1000	40	Confocal peak detection threshold

Description

This command lets you specify an intensity threshold for the peak detection. The threshold parameter is used to tune the peak detection algorithm to the desired sensitivity. It should be set as low as possible to be able to measure dark surfaces, but high enough to suppress the detection of noise spikes when there is no object in the detection range.

The threshold is in arbitrary units.

If the sensor doesn't detect a signal which passes the threshold, 0 is output for distance and intensity. However, this behavior doesn't disturb the averaging algorithm, as invalid results are excluded from averaging.

Good to know

In most cases, high <arg0>-values are not very useful, since noise is no longer detected even at small values. Especially for weak signals or high measuring rates, you should choose a small threshold and increase the threshold empirically upwards.

Examples

Input	Comment
THR 30	Sets the threshold to 30.
THR ?	Returns the current values.

Related commands

QTH (Quality threshold, interferometric mode)

2.67 TMOD (Trigger mode)

Command format TMOD <arg0>

Argument quick info

No.	Type	Value	Default	Description
arg0	str	CTN, TRE, TRG, TRW	-	Specifies the trigger mode (CTN, TRE, TRG and TRW).

Description

This command queries the trigger mode. The query will return a string to indicate the current trigger mode. The current string values are CTN, TRE, TRG and TRW.

Furthermore, this command in combination with the strings CTN, TRE, TRG and TRW can also be used to set the trigger mode (see section Examples below for details).

Examples

Input	Comment
TMOD ?	Queries the trigger mode. For example: If the device operates in free run mode, this command replies “TMOD CTN”.
TMOD CTN	Recover from TRE/TRG/TRW command to go into free run mode (measurements are started periodically based on the measuring rate).

Related commands

CTN (Continue in free run mode)
TRE (Trigger each)
TRG (Trigger once)
TRW (Trigger window)

2.68 TRE (Trigger each)

Command format TRE

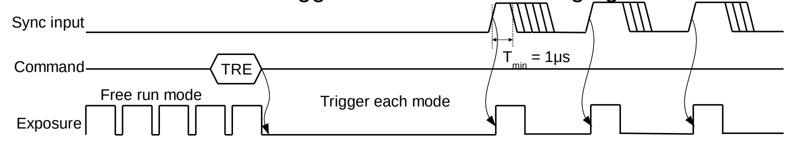
Argument quick info No argument supported

Description Switches to Trigger Each mode: In this mode, every trigger event triggers one exposure or a burst of AVD*AVS exposures, if averaging has been activated by setting AVD (Data averaging) and/or AVS (Spectra averaging) to values >1. Each exposure (or the first exposure of an averaging burst, respectively) will begin exactly at the trigger event.

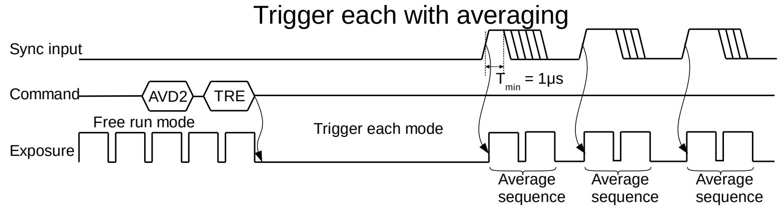
If the previous exposure is still ongoing, the trigger will be delayed until the detector is ready. If the device receives a trigger event while the preceding delayed trigger still waits for execution, the trigger event will be lost and the trigger lost counter will be incremented. Delayed and skipped trigger events are flagged in the "Exposure-flags" result signal (ID 76, see subchapter Global signals).

Data (AVD) and spectral (AVS) averaging is possible in Trigger Each mode. In Trigger each mode, one trigger event will start a sequence of AVD*AVS exposures. The AVD*AVS exposures are executed with the current sample rate (SHZ). One averaged result will be output after the exposure sequence. There will be only one trigger event on the Sync-out signal per n samples to be averaged. The pulse marks the beginning of the first exposure of the averaging interval. In the other trigger modes, there is one Sync-out pulse for every exposure, regardless of averaging.

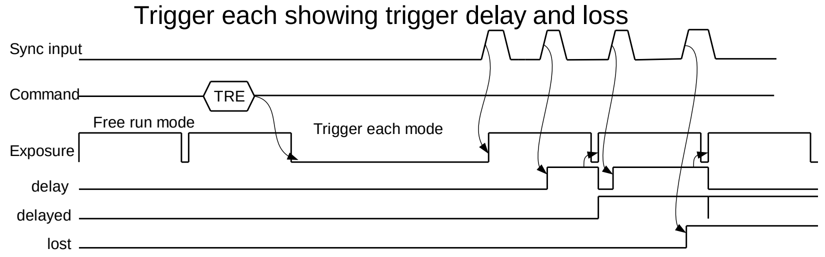
Trigger each without averaging



Trigger each with averaging



Trigger each showing trigger delay and loss



Good to know

- Note that when using Trigger Each mode, the free-run frequency SHZ should be set slightly higher than the maximum expected trigger frequency. This guarantees that the exposure cycle (whose duration is defined by the SHZ command) is completed before a new trigger event occurs.

- The command CTN resumes normal operation (free run mode).
- The trigger modes TRE, TRW and CTN are stored in the non-volatile memory after issuing an SSU command (Save setup).
- The command TRE resets the sample counter.

Related commands

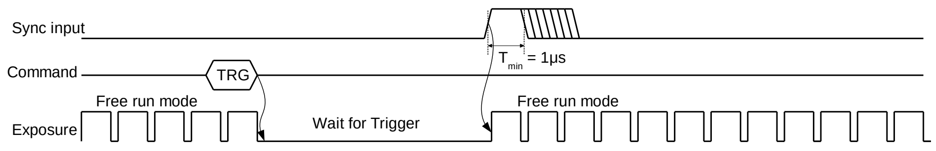
AVS (Spectra averaging)
AVD (Data averaging)
CTN (Continue in free run mode)
STR (Set trigger)
TMOD (Trigger mode)
TRG (Trigger once)
TRW (Trigger window)

2.69 TRG (Trigger once)

Command format TRG

Argument quick info No argument supported

Description Switches to Trigger Once mode: Stops the free run mode and waits for a trigger event. The trigger event restarts the free run mode. The first exposure will occur immediately at the trigger event.



Good to know

- In addition to a trigger event, the CTN command can be used to resume normal operation (free run mode).
- The command TRG resets the sample counter.

Related commands

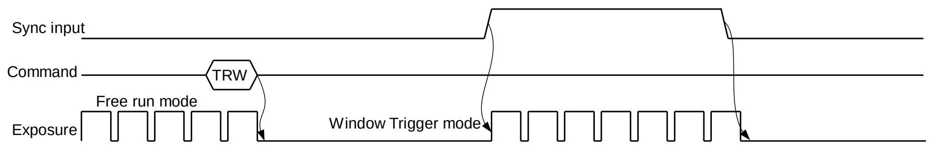
CTN (Continue in free run mode)
TRE (Trigger each)
TMOD (Trigger mode)
TRW (Trigger window)

2.70 TRW (Trigger window)

Command format TRW

Argument quick info No argument supported

Description Switches to Window Trigger mode: While the voltage difference between Sync-In and GND is at the active level as defined with STR, the device is in free-run mode. If the voltage level changes from the inactive level to the active level, the first exposure is synchronized with the edge of the voltage change. If the voltage is already at the active level before the TRW command is applied, a new exposure cycle will start as soon as the TRW command is processed.



Good to know

- The command CTN resumes normal operation (free run mode).
- The trigger modes TRE, TRW and CTN are stored in the non-volatile memory after issuing an SSU command (Save setup).
- The Window Trigger mode can be combined with the window averaging mode (AVD 0). In this case, one result sample is produced for each high period of the trigger signal. This sample averages all measurements started between a rising and a falling edge of the trigger signal.

Related commands

AVD (Data averaging)
CTN (Continue in free run mode)
STR (Set trigger)
TMOD (Trigger mode)
TRE (Trigger each)
TRG (Trigger once)

2.71 ULFW (Upload firmware)

Command format ULFW <arg0> <arg1> <arg2>

Argument quick info

No.	Type	Value	Default	Description
arg0	int	-	-	Offset of the current chunk within the firmware file
arg1	int	-	-	Total firmware file size in bytes (not size of the current chunk)
arg2	blob	-	-	Blob argument containing the current chunk

Description

A firmware can be uploaded by this command. The data packets must be sent in sequential order.

The maximum allowed blob size is 4096 bytes. All chunks must be sent in their correct order. The offset of the first chunk must be 0. For the next chunks:

start offset current chunk = start offset previous chunk + blob size previous chunk

2.72 VER (Version information)

Command format VER

Argument quick info

No.	Type	Value	Default	Description
arg0	str	-, list, json	-	Specifies command mode (see below for details).

Description

This command queries identification key-value pairs. The classical VER command without arguments returns value pairs of the form:
<key1>=<value1><crlf><key2>=<value2><crlf>...

The other two commands VER list and VER json provide extended information in alternative formats (see below for details).

Good to know

- For VER: The keys "firmware_version", "hardware_serial_number" and "device_serial_number" are always present. The other keys are not guaranteed in format and position, new ones can be added in the future.

- For VER list and VER json: The keys "platform", "product", "firmware_type", "major_version", "minor_version", "build_hash", "device_serial_no" and "hardware_serial_no" are always present. The other keys are not guaranteed in format and position, new ones can be added in the future.

Examples

Input	Comment

VER	In classic form (i.e. without argument), the command returns for example the following string:
	
	`firmware_version=1.4.1`
	`build=r # b4878b0be # 1867 #`
	`2023-01-25T12:19:26+0000 # mainline`
	`hardware_serial_number=3034`
	`device_serial_number=203034`

Input	Comment

VER list	Alternative command form VER list, returns more detailed information in the following form:
	
	`platform="CHRocodile2"`
	`product="SE"`
	`firmware_type="R"`
	`major_version="1"`
	`minor_version="4"`
	`running_no="1"`
	`build_no="1867"`
	`build_hash="25758b4a40"`
	`build_date="20221029"`
	`device_serial_no="1234"`
	`hardware_serial_no="0000243432"`
	`main_board_revision="2880_C.0210"`
	`firmware_variant="mainline"`
	`friendly_namen="CHRocodile ..."`
	`product_code="5106129"`
	`...`

Input	Comment

VER json	Alternative command form VER json, returns a standardized version information as a JSON-formatted string:
	
	`"platform": "CHRocodile2",`
	`"product": "SE",`
	`"firmware_type": "R",`
	`"major_version": "1",`
	`"minor_version": "4",`
	`"running_no": "1",`
	`"build_no": "1867",`
	`"build_hash": "25758b4a40",`
	`"build_date": "20221029",`
	`"device_serial_no": "1234",`
	`"hardware_serial_no": "0000243432",`
	`"main_board_revision": "2880_C.0210",`
	`"firmware_variant": "mainline"`
	`"friendly_namen": "CHRocodile ..."`
	`"product_code": "5106129"`
	`...`

Related commands IDE (Identification)

Chapter 3

CHRocodile Signal IDs

3.1 Introduction

Overview

The following chapter describes the use of signal IDs. These IDs are used together with the SODX (Set output data extended) command to control the composition of the output packet that is produced for every measured sample.

First, an overview diagram is provided that explains how the signal ID is composed. Then, examples for defining some signal IDs are given, followed by a list of global signals (with detailed description of the ExposureFlags signal). Finally, a list is appended with the signal ID range from 0 to 63, which serves for compatibility to CHRocodile devices of older generations.

Peak/global signals

Signals are either peak signals or global signals. Peak signals relate to measured surfaces. There can be several peak signals (belonging to different measurement channels and/or surfaces) in one sample, whereas global signals represent information that is common for all channels of a sample.

Examples global signals	Examples peak signals
Time stamp, encoder values, sample counter ...	Distance, Thickness, Intensity ...

Thus, while some values are defined only once per sample, others can be specified for each detected peak. In contrast to a global signal, a peak signal is always a combination of the measured value and the number of the corresponding peak.

Selecting signals

Signal IDs to be included in the output packet can be selected with the SODX command, see SODX (Set output data extended) command for details.

3.2 Signal ID definition

	Bit number of the signal ID															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Global exposure information	Value encoding		Statistics for averaging interval (not for CHR C/Mini, DPS, CLS, MPS)			0	0	0	Global signal ID (see separate table)							
Peak signal, single channel devices	Value encoding		Statistics for averaging interval (not for CHR C/Mini, DPS, CLS, MPS)			0	Mode 0: distance 1: thickness	1	Peak index					Peak signal type		
Peak signal, multi-channel devices	Value encoding		Statistics for averaging interval (not for CHR C/Mini, DPS, CLS, MPS)			0	Mode 0: distance 1: thickness	1	Reserved (all bits 0)			Peak index		Peak signal type		

Bit number			Statistics (in case of averaging, see AVD)
13	12	11	
0	0	0	Raw value / Average of raw values
0	0	1	Standard deviation (in average interval)
0	1	0	Number of valid values (in average interval)
0	1	1	Minimum (in average interval)
1	0	0	Maximum (in average interval)
1	0	1	Peak-to-Peak difference (max-min) (in average interval)

Note: Statistics are not available for CHR C/Mini, DPS, CLS, MPS!

Bit number					Peak number
7	6	5	4	3	
0	0	0	0	0	1st peak
0	0	0	0	1	2nd peak
0	0	0	1	0	3rd peak
⋮					

Bit number	Mode
9	
0	Distance (in confocal mode) Thickness (in interferometric mode)
1	Thickness in confocal mode (distance difference between selected and next peak)

Bit number			Signal Name	Description
2	1	0		
0	0	0	PeakValue	Distance / Thickness
0	0	1	PeakIntensity / PeakQuality	Confocal mode: Intensity of peak Interferometric mode: Quality of FFT peak
0	1	0	CCDSaturation	Confocal mode: Peak height on detector in percent of full well
0	1	1	PeakPos	Confocal: detector pixel position Interferometric mode: Bin position of FFT peak
1	0	0	PeakValue Median	Median/first percentile of PeakValue (median filter width defined with MEDX/MED command)
1	1	0	PeakValue Median 2	Second percentile (only for Precitec IDM devices, defined by <arg2> of MEDX command)
1	0	1	PeakWidth	Confocal mode only: peak width, given in pixels

Bit number		In case of geometrical quantities (e.g. distance or thickness)	In case of non-geometrical quantities	
15	14		Native number values (e.g. peak intensity)	Native integer values (e.g. encoder)
0	0	Float in µm, including refractive index (geometrical thickness)	Float (IEEE 754)	32-bit integer
0	1	Integer 16 bit, scaled as fraction of measurement range, without refractive index. Distance and thickness values are given as: d[µm] = value * SCA[µm] / 2^{15}. To get the geometrical thickness, the value has to be multiplied (chrom. confocal case) or divided (interferometric case) by the index of refraction of the material.	Truncated 16-bit integer	Least significant 16-bit
1	0	32-bit integer value in nanometers, including refractive index (geometrical thickness)	32-bit integer value	Most significant 16-bit
1	1	Integer 32 bit, scaled as fraction of measurement range, without refractive index. Distance and thickness values are given as: d[µm] = value * SCA[µm] / 2^{31}. To get the geometrical thickness, the value has to be multiplied (chrom. confocal case) or divided (interferometric case) by the index of refraction of the material.	undefined	undefined

3.3 Examples of some signal IDs

What is the signal ID of Distance 1 in 16-bit integer format?

Bit number	Binary value	Description
15 to 14	0 1	For 16-bit integer format
13 to 11	0 0 0	Average, in case averaging is activated by AVD
10 to 09	0 0	For Distance
8	1	For a peak signal
7 to 3	0 0 0 0 0	For the first peak = Distance 1 / Thickness 1
2 to 0	0 0 0	For the distance or thickness value
Total signal ID	0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0	= 16640 (decimal)

To request Distance 1 in 16-bit integer format, the command is SODX 16640.

What is the signal ID of Distance 2 in float format?

Bit number	Binary value	Description
15 to 14	0 0	For float format
13 to 11	0 0 0	Average, in case averaging is activated by AVD
10 to 09	0 0	For Distance
8	1	For a peak signal
7 to 3	0 0 0 0 1	For the second peak = Distance 2 / Thickness 2
2 to 0	0 0 0	For the distance or thickness value
Total signal ID	0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0	= 264 (decimal)

To request Distance 2 in float format, the command is SODX 264.

What is the signal ID of Thickness 1 in float format?

Bit number	Binary value	Description
15 to 14	0 0	For float format
13 to 11	0 0 0	Average, in case averaging is activated by AVD
10 to 09	0 1	For Thickness
8	1	For a peak signal
7 to 3	0 0 0 0 0	For the first peak = Distance 1 / Thickness 1
2 to 0	0 0 0	For the distance or thickness value
Total signal ID	0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0	= 768 (decimal)

To request Thickness 1 (in chromatic confocal mode) in float format, the command is SODX 768.

What is the signal ID for the encoder counter X in 32-bit integer format?

Bit number	Binary value	Description
15 to 14	0 0	For 32-bit integer format
13 to 11	0 0 0	Average, in case averaging is activated by AVD
10 to 09	0 0	In case a global signal is selected
8	0	For global signal
7 to 0	0 1 0 0 0 0 0 1	For the x-encoder, decimal value = 65
Total signal ID	0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1	= 65 (decimal)

To request the value of encoder counter X in 32-bit integer format, the command is SODX 65.

3.4 Global signals

Signal ID	Signal name	Native type	Remarks
64	StartTime	u32	In ns, free running time base
65	Start_PositionX	s32	X-encoder position at the beginning of the exposure
66	Start_PositionY	s32	Y-encoder position at the beginning of the exposure
67	Start_PositionZ	s32	Z-encoder position at the beginning of the exposure
68	Start_PositionU	s32	U-encoder position at the beginning of the exposure
69	Start_PositionV	s32	V-encoder position at the beginning of the exposure
70	Stop_PositionX	s32	X-encoder position at the end of the exposure
71	Stop_PositionY	s32	Y-encoder position at the end of the exposure
72	Stop_PositionZ	s32	Z-encoder position at the end of the exposure
73	Stop_PositionU	s32	U-encoder position at the end of the exposure
74	Stop_PositionV	s32	V-encoder position at the end of the exposure
75	ExposureCount	u16	Exposure number of the first exposure in the averaging cycle
76	ExposureFlags	u16	Bits containing information about the exposure (see details below)
77	RealExpTimeNs	u32	Exposure time of detector (ns)
78	RealLightingTimeNs	u32	Time when the light source is on during exposure (ns)
79	TriggerLostCounter	u16	Number of trigger events that have been ignored since the last sample because the detector was not ready to be triggered
80	NumberOfValidPeaks	u16	Number of peaks that have been found in the spectrum
81	TicketNumber	u16	Command ticket number, for diagnosis
82	InterferomIntensity	float	Highest intensity on detector in percent of full well
83	SampleCounter	u16	Counts samples that have been processed
84	Reserved	-	Reserved
85	interfEnergy	float	Accumulated energy of the completed spectrum (interferometric mode only)
86	Health_DSPLoad	u32	DSP load in 1/1000 of full load
87	Health_TicketWrongOrder	u32	Error counter
88	Health_UPPLostCount	u32	Error counter: spectrum lines lost during transmission
89	Health_ExposureLostCount	u32	Error counter
90	Health_UPPNotFinished	u32	Error counter
91	PacketTimestampOffset	s32	Only defined for packet protocol; never ordered by SODX. For detailed description see below.
92	Reserved	-	Reserved
93	Internal temperature	s16	Format: degree Celsius * 100 Note: Not available for all devices. If available, the measurement location is device-specific (for CHRocodile 2 S/S HS/S HP/SE/DPS: measurement at light source).
94	LostAnalogValues	s16	Counts result values written to analog output that have been lost because of buffer overflow.
95	PixelBlackValue	u16	Signal value on a black pixel of the detector
96	Counter80MHzMSB	u16	16 most significant bits of a counter running at 80 MHz (for backwards compatibility reason)
97	Counter80MHzLSB	u16	16 least significant bits of a counter running at 80 MHz (for backwards compatibility reason)

Signal ID	Signal name	Native type	Remarks
. . .	Reserved	-	Reserved
240	CALC0Result	float	Result of calculation defined by CALC 0 (applies only to double channel devices (CHRocodile 2 DPS))
241	CALC1Result	float	Result of calculation defined by CALC 1 (applies only to double channel devices (CHRocodile 2 DPS))
242	CALC2Result	float	Result of calculation defined by CALC 2 (applies only to double channel devices (CHRocodile 2 DPS))
243	CALC3Result	float	Result of calculation defined by CALC 3 (applies only to double channel devices (CHRocodile 2 DPS))

ExposureFlags (ID 76)

In case of averaging, the ExposureFlags (ID 76) value is the bitwise OR combination of the ExposureFlags of the averaged exposures.

Bit number	Description
0 (LSB)	**Saturation** occurred on at least one detector pixel in current sample.
1	At least one **trigger was skipped** since last sample (Trigger lost).
2	Present sample based on a **delayed Trigger** event (system was not ready when trigger occurred).
3	Reserved
4	Reserved
5	Reserved
6	Reserved
7	Reserved
8	Second exposure (in double exposure mode)
9	Double exposure active
10	State of the Sync-in input (if usable for trigger)
11	Short Sync Pulse (2.5 µs) sent on Sync-out. Occurs every 2 s if time stamp synchronization (see OPD 6) is not active.
12	Short Sync Pulse (<3.5 µs) received from Sync-in. Can occur only if time stamp synchronization (see OPD 6) is not active.

PacketTimestampOffset (ID 91)

In “Trigger Each” mode, sample time stamps are not equidistant as in free run mode and therefore cannot be defined in terms of a starting time stamp and a sample frequency inside a data packet. Instead, every sample starts with a time stamp offset (32bit) which has to be added to the data packet time stamp (64bit) to obtain the effective time stamp of the sample:

Effective time stamp in seconds: $t_{sample} = \frac{t_{packet}+t_{offset}}{2^{32}}$

3.5 Definition of signal ID in range 0 to 63

The IDs of the global signals do not start at 0 but at 64. The range from 0 to 63 is reserved for compatibility with older CHRocodile generations that supported only 16 bit integer signals. It aliases a subset of the normal signals.

Example:
ID 0 is an alias for ID 16640: Peak 0 distance value, 16 bit integer
Signals as defined for the CHRocodile 2 device.

For a complete list, see the following table:

Signal ID	Alias for signal in mode 0 (1 peak, distance)	Alias for signal in mode 1 (2 peaks, thickness)	Alias for signal in mode 2 (3 peaks, interferometric)
0	16640 (Distance 1 / 16 bit)	17152 (Thickness 1 / 16 bit)	16640 (Thickness 1 / 16 bit)
1	–	16640 (Distance 1 / 16 bit)	16648 (Thickness 2 / 16 bit)
2	–	16648 (Distance 2 / 16 bit)	16656 (Thickness 3 / 16 bit)
3	16641 (Intensity 1 / 16 bit)	–	16641 (Quality 1 / 16 bit)
4	–	16641 (Intensity 1 / 16 bit)	16649 (Quality 2 / 16 bit)
5	–	16649 (Intensity 2 / 16 bit)	16657 (Quality 3 / 16 bit)
6	16643 (peak 1 position, pixels)	16643 (peak 1 position, pixels)	16466 (Intensity / 16 bit)
7	–	–	–
8	32844 (ExposureFlags)		
9	32832 (Exposure time in 12.5 ns units)		
10	32833 (Encoder 0 position, 16 bit, most significant word)		
11	16449 (Encoder 0 position, 16 bit, least significant word)		
12	32834 (Encoder 1 position, 16 bit, most significant word)		
13	16450 (Encoder 1 position, 16 bit, least significant word)		
14	32835 (Encoder 2 position, 16 bit, most significant word)		
15	16451 (Encoder 2 position, 16 bit, least significant word)		
16	83 (Sample Counter)		
17	93 (internal temperature in °C * 100)		
32	75 (Exposure count)		
33	96 (Timecounter, 80 MHz most significant word)		
34	97 (Timecounter, 80 MHz least significant word)		

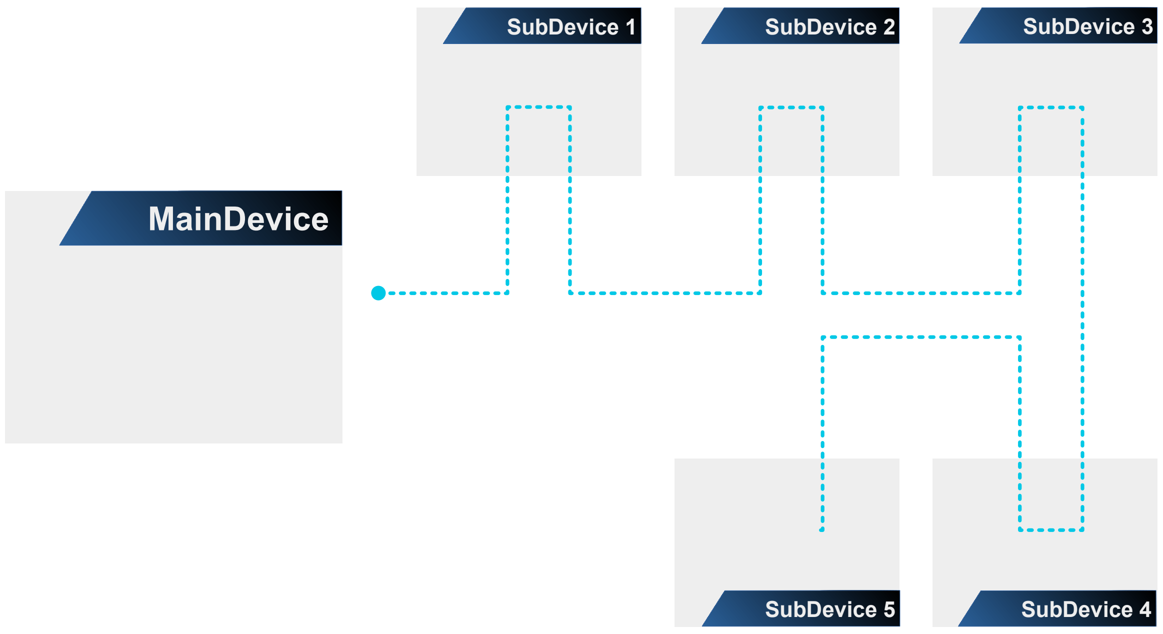
Chapter 4

EtherCAT Protocol

4.1 Introduction

Overview EtherCAT® (Ethernet for Control Automation Technology) is a modern, highly efficient protocol for industrial real-time automation and machine control. Introduced in 2003, it became an industrial fieldbus standard in the following years. EtherCAT is widely used in nearly any industrial branch and offers advantages in fast processes and in the networking of several devices and machines.

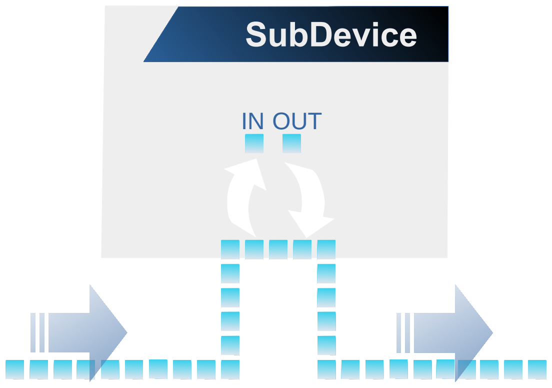
EtherCAT system A typical EtherCAT system consists of an EtherCAT main unit (a so-called MainDevice) and several EtherCAT sub units (referred to as SubDevices, e.g. CHRocodile sensor). The MainDevice, typically integrated in the machine controller (e.g., a PC-based controller, a PLC or numeric control) represents the communication controller device. Up to 65535 EtherCAT SubDevices may be supported, i.e. field devices like sensors, actuators, drives.
Communication is realized via standard Ethernet cables with 2x RJ-45 connectors. The EtherCAT MainDevice sends telegrams through the network. These run from SubDevice to SubDevice, whereby the SubDevices take out data that is intended for them (while the telegram is running) and put in data that they want to send to other units.



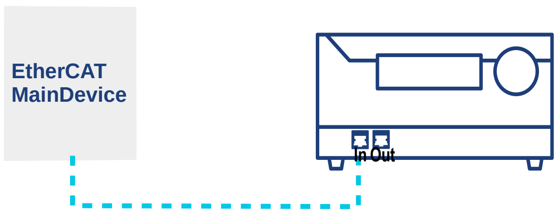
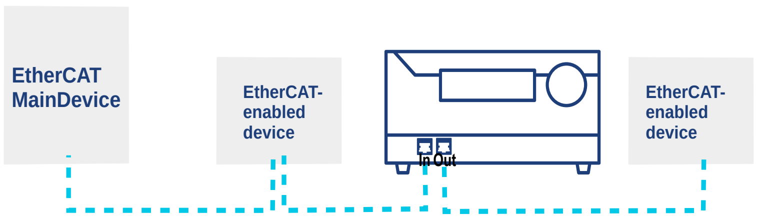
EtherCAT does not send data to individual SubDevice nodes, but passes Ethernet frames through all of the SubDevice nodes. From the last EtherCAT sub unit the completely processed frame is sent back to the MainDevice. The figure above illustrates these relationships.

Read and write

The MainDevice controls the EtherCAT communication and is the only device being able to send new EtherCAT frames. The MainDevice sends and receives the frames via a standard MAC (medium access control), while SubDevices process the frames. During circulation, data is read and written at runtime. At the SubDevices, individual bits or even larger data packets can be taken from or added to the EtherCAT telegram.



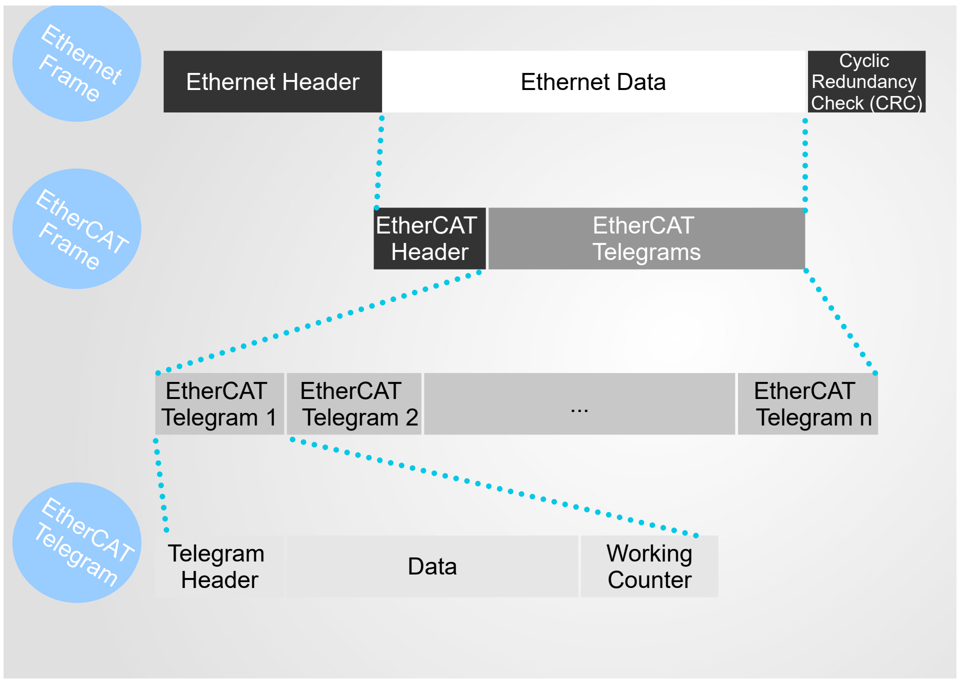
Bus topologies The following table illustrates the most common EtherCAT topologies:

MainDevice-SubDevice topology	
CHRocodile as only SubDevice connected to EtherCAT MainDevice (only the EtherCAT In port must be used to connect the device in this case).	
Chain or star topology	
CHRocodile as SubDevice in a chain or star topology (both EtherCAT In port and EtherCAT Out port must be used to connect the device).	 EtherCAT MainDevice EtherCAT-enabled device In Out EtherCAT-enabled device

A ring topology is not very common and is usually chosen only for ensuring cable redundancy (if one connection to the MainDevice is broken, the other may still be functional). Note that in an EtherCAT system, communication is always “circular”: That is, an EtherCAT packet travels from the MainDevice through all connected devices and then back to the MainDevice.

EtherCAT frame

EtherCAT data is packaged in Ethernet data packets. The following diagram illustrates the structure of Ethernet frame, EtherCAT frame and EtherCAT telegram:



The Ethernet frame, exchanging data between the EtherCAT main unit and the EtherCAT sub units, contains the EtherCAT frame. The EtherCAT frame itself contains up to 15 EtherCAT telegrams. An EtherCAT telegram starts with an telegram header (data length, including address of one or more SubDevices, etc.), followed by the EtherCAT data and the working counter (counts the number of interactions with SubDevices in a given telegram).

Basic information

More detailed information on the basics of the protocol can be found on the webpage of the EtherCAT® Technology Group (www.ethercat.org). This is a global organization in which manufacturers, end users and technology providers support and promote further EtherCAT technology development.

Mounting and wiring

Information on setting up your CHRocodile and wiring your hardware for EtherCAT purposes can be found in the corresponding CHRocodile User Manual.

4.2 EtherCAT and CHRocodile

Overview

The CHRocodile device supports the EtherCAT fieldbus according to the IEC 61158 standard with no specific device profile. EtherCAT comes with the following characteristics:

- CANopen-over-EtherCAT (CoE)
- EtherCAT Tx Process Data Object (PDO) mechanism for transferring measurement results, including an oversampling mechanism
- Device configuration via EtherCAT Service Data Objects (SDO)

A CHRocodile device will utilize the PDO mechanism only for sending data and not for receiving process data (that is, the RxPDO option is not used).
Note: The EtherCAT interface can be used in parallel with other interfaces in a multi-client configuration. Active switching between EtherCAT and other protocols (e.g. packet protocol over TCP/IP) is not necessary.

Data objects

SDO (Service data objects) and PDO (Process data objects) allow the communication in the EtherCAT system. Both data objects have distinct functions: SDOs are used for non-cyclic communication, which is typically configuration, whereas PDOs are used for cyclically transferred data through the EtherCAT system during each cycle.
The following table lists characteristics of both data object types:

Service data objects (SDO)	Process data objects (PDO)
Transmits service data	Transmits process data in real-time
Standard parameterization of the controller	Quick exchange of process data
Sent upon request	Sent every cycle

Each device has an object dictionary (OD), i.e. a data structure addressed by index and subindex that contains various data objects. The object dictionary bears entries that specify, e.g., the device configuration. This object dictionary can be accessed via SDOs (read and write).

All entries of the object dictionary and their addresses/subindices are listed in the ESI file of the device (see subchapter ESI Files below).

Availability

The EtherCAT interface supports a subset of the CHRocodile packet protocol commands. So far, the following commands are available:

AAL (Auto adapt light source)
ABE (Abbe number)
AVD (Data averaging)
AVS (Spectra averaging)
CALC (Calculation on signal values; only CHRocodile 2 DPS)
CTN (Continue in free run mode)
DNLD (Download spectrum)
DRK (Dark reference)
LAI (Lamp intensity)
LOC (Lock front panel keyboard)
MMD (Measurement mode)
NOP (Number of peaks)
SCA (Scale)

SEN (Select chromatic calibration)
SENX (Extended chromatic calibration table query)
SFD (Set factory defaults)
SHZ (Set sample frequency in Hz)
SRI (Set refractive indices)
SSU (Save setup)
STA (Start data)
STO (Stop data)
STR (Set trigger)
THR (Threshold, confocal mode)
TMOD (Trigger Mode)
TRE (Trigger each)
TRG (Trigger once)

Note: A more detailed description of the commands can be found in the chapter CHRocodile Commands of this Protocol and Command Reference.

4.3 Executing CHRocodile commands

Commands

The CHRocodile packet protocol commands can be executed by utilizing EtherCAT SDOs. Each command is represented by a specific SDO (unless explicitly noted, all object dictionary entries can also be read):

Command	Index:Subindex	Data type	Usage
AAL	0x2000:01	u8	0 - Disables auto adapt mode. 1 - Enables auto adapt mode; desired saturation level specified in 0x2000:02.
	0x2000:02	f32	Saturation level in % Note: For enabling the auto adapt mode, write the saturation level into SDO 0x2000:02, then set the value of subindex 01 to "1". Once enabled, writing to the SDO 0x2000:02 will apply the new saturation level immediately.
ABE	0x200F:01	u8	Number of layers
	0x200F:02 ... 0x200F:11	f32	Abbe number for up to 16 layers Note: Use the subindices 02...11 to set up the layer values. Then set the value of subindex 01 to the number of layers that have been configured. Setting the number of layers will execute the command.
AVD	0x200D	u16	
AVS	0x2010	u16	
CALC	0x2015:01	u8	CALC 0 Format ID: 0 - Standard format 1 - Reference format
	0x2015:02	f32	Standard format: CALC 0 Offset value
	0x2015:03	f32	Standard format: CALC 0 First factor
	0x2015:04	f32	Standard format: CALC 0 Second factor
	0x2015:05	u8	Standard format: CALC 0 Channel 1 peak no.
	0x2015:06	u8	Standard format: CALC 0 Channel 2 peak no.
	0x2015:07	f32	Reference format: CALC 0 Height/thickness of reference object
	0x2015:08	f32	Reference format: CALC 0 Averaging of offset values
	...	...	...
CALC	0x2018:01	u8	CALC 3 Format ID: 0 - Standard format 1 - Reference format
	0x2018:02	f32	Standard format: CALC 3 Offset value
	0x2018:03	f32	Standard format: CALC 3 First factor
	0x2018:04	f32	Standard format: CALC 3 Second factor
	0x2018:05	u8	Standard format: CALC 3 Channel 1 peak no.
	0x2018:06	u8	Standard format: CALC 3 Channel 2 peak no.
	0x2018:07	f32	Reference format: CALC 3 Height/thickness of reference object
	0x2018:08	f32	Reference format: CALC 3 Averaging of offset values

Command	Index:Subindex	Data type	Usage
CTN			See command TMOD
DNLD	0x2012:01	s32	Spectrum ID
	0x2012:02	u16	Exposure number
	0x2012:03	f32	Micrometers per bin
	0x2012:04	s32	Block exponent of spectrum data
	0x2012:05	octetstr	Spectrum data Note: For detailed instructions on downloading spectra, see subchapter Downloading a spectrum.
	0x2012:06	u32	Spectrum data length in bytes
DRK	0x2001:01	u8	Dark reference state, set to 1 for initiating a dark reference determination operation
	0x2001:02	f32	Virtual measuring rate in Hz, available after dark reference operation has completed. Note: To determine if operation is still progressing, the MainDevice can read the entry 0x2001:01. If the returned value is '1', the operation is still progressing.
LAI	0x2002	f32	Lamp intensity in %
LOC	0x2013	u8	Lock front panel inputs 0 - Keys and jogwheel unlocked 1 - Keys and jogwheel locked
MMD	0x200B	u8	0 - Chromatic confocal mode 1 - Interferometric mode
NOP	0x2003	u8	Number of peaks
SCA	0x200C	u32	Full scale in micrometers
SEN	0x200A	u8	Chromatic calibration table in use
SENX	0x2014:01 0x2014:02	u8 vstr	Query current/all probe calibrations To query probe calibration(s), set 0x2014:01 to either '0' (which is the equivalent of a 'SENX ?' query), or '1' (being the equivalent of a 'SENX enum ?' query). In either case, the response string can be obtained by reading from 0x2014:02.
SFD	0x200E	u8	Set to '0' to reset the device configuration to factory defaults; the TCP/IP network settings will be kept. Set to '1' to reset the device configuration to factory defaults, including the TCP/IP network settings. Note: This entry cannot be read.
SHZ	0x2004	f32	Sample frequency in Hz

Command	Index:Subindex	Data type	Usage
SRI	0x2011:01	u8	Number of layers
	0x2011:02 ... 0x2011:11	f32	Refractive index for up to 16 layers Note: Use the subindices 02...11 to set up the layer values. Then set the value of subindex 01 to the number of layers that have been configured. Setting the number of layers will execute the command.
SSU	0x2008	u8	Saves the current setting to non-volatile memory.
STA STO	0x2009	u8	Set to ’1’ (true) for starting sample output, set to ’0’ (false) for stopping sample output
STR	0x2007:01	u8	Generates a single trigger event when set to ’1’. The device will reset the value automatically. Note: This entry cannot be read.
	0x2007:02	u8	Trigger state: 0 – Low trigger state 1 – High trigger state Note: This entry cannot be read.
THR	0x2005	u16	Confocal peak detection threshold
TMOD	0x2006	vstr	The trigger mode to be applied: CTN – Continue in free run mode TRE – Trigger each TRG – Trigger once ECAT – Synchronize with EtherCAT signals (see Acquisition triggering and data transfer for details)
TRE			See command TMOD
TRG			See command TMOD

Data types mentioned in the table above:

Notation	Explanation
u8, u16, u32	Unsigned integers of 8, 16 or 32 bit length
s8, s16, s32	Signed integers of 8, 16 or 32 bit length
f32	32-bit floating point value (according to IEE 754 standard)
vstr	EtherCAT variable-length character string
octetstr	Array of unsigned integer values (8 bit)

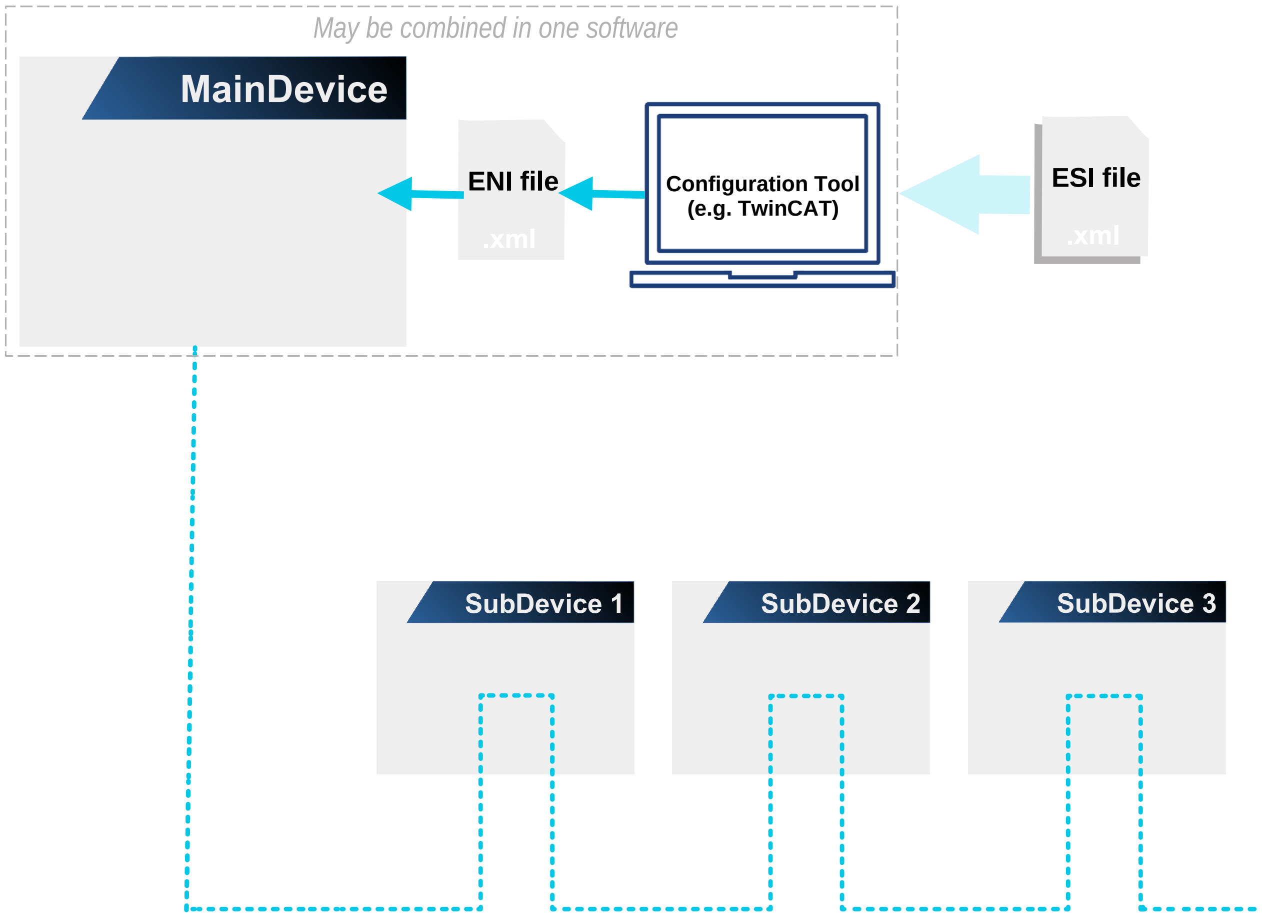
SODX not supported The SODX command for configuring the set of signals to be acquired and transmitted by the device is not supported. Instead, use the EtherCAT PDO mapping and assignment mechanism for configuring the sample contents as described in subchapter Sample data and signal IDs below.

SDO info The CHRocodile device supports the EtherCAT/CANopen “SDO info” service for providing details of the object dictionary configuration to an EtherCAT main component. However, “SDO complete access” read or write operations are currently not supported.

4.4 ESI Files

Overview

The EtherCAT SubDevice Information (ESI) is a standardized device description file. It is used to describe the communication characteristics and functionalities supported by an EtherCAT-enabled device. Specifically, the ESI file is utilized to define certain communication properties, the mapping of process data (PDO), the object dictionary contents (SDO) and other relevant information for EtherCAT Sub-Devices.
The ESI of an EtherCAT-enabled device is usually provided as a file in XML format. It will be processed by an EtherCAT MainDevice in order to communicate with a specific device correctly.



The EtherCAT Configuration Tool is the software application which allows to define the architecture of the EtherCAT network. The architecture will be adopted by the EtherCAT MainDevice to run the network itself during operation. This comprises a description of the network initialization commands and the cyclic commands. This description can be provided to the EtherCAT MainDevice as a file in a standardized XML format (EtherCAT Network Information, ENI).

Download ESI File

The ESI file of a CHRocodile device is stored on the device itself. To define the EtherCAT network layout, the ESI file from the CHRocodile sensor can be provided to the Configuration Tool for configuration purposes. The ESI file is downloaded using a standard web browser (such as Chrome, Firefox etc.) as described below:

1. Connect the CHRocodile device to a computer using the Ethernet socket (TCP/IP), see User Manual of the corresponding CHRocodile for details.
2. Open a web browser.
3. Type the following URL into the address field of the browser and press return: http://<device-IP-address>/esi (e.g. http://192.168.170.2/esi)

Result: The XML code is shown in the browser window.

4.5 Acquisition triggering and data transfer

Overview

This subchapter deals with synchronization modes, triggering and data acquisition of a CHRocodile sensor in an EtherCAT network.
Note: For the following information, basic knowledge of CoE (CAN over EtherCAT) and a deeper understanding of the EtherCAT protocol are necessary. More detailed information on the basics of the protocol can be found on the webpage of the EtherCAT® Technology Group (www.ethercat.org). There, also terms used in the following like “Outputs Valid” and “Input Latch” are defined.

Trigger modes

The triggering modes “trigger each” (TRE command), “trigger once” (TRG command), “trigger window” (TRW command) and “free run” (CTN command) will always function independently of EtherCAT bus events: Acquisition operations are either triggered by the device’s trigger input or by means of the encoder trigger functions. In “free run” mode, acquisition operations will be triggered automatically with the configured sample frequency, which is also generated independently of the EtherCAT bus timing. Note, however, that a CHRocodile device will express the start time of an exposure (represented by the global signal ID 64) as EtherCAT bus time if the device is connected to an EtherCAT bus.

ECAT mode

An additional mode is available for use if the CHRocodile device participates in an EtherCAT system. This acquisition triggering mode is referred to as “ECAT”. It is similar to the “free run” mode. However, it synchronizes a device with EtherCAT bus events and basically matches the EtherCAT SubDevice synchronization mode “DC” (distributed clocks), subordinate “µC cycle”, shifting of “Outputs Valid” and/or “Input Latch”. In this mode, a CHRocodile device will still utilize the configured sample frequency, but the start of each acquisition operation will be synchronized with the EtherCAT Sync1 signal (shifted by the offset time specified by means of the “Shift time” located at object dictionary index 0x1C32:03). This behavior results in some implications that must be considered when configuring acquisition operations, see section “Configuring Sync1” below.

To enable EtherCAT bus-based acquisition triggering, set the object entry 0x2006 (“TMOD”) to the string value “ECAT”.

Configuring Sync1

To ensure consistent, uniform sample intervals, the Sync1 signal frequency must be an integral multiple of the total sample frequency resulting from the configured sample frequency (SHZ), data averaging (AVD) and spectrum averaging (AVS) parameters. This can be expressed as AVD * AVS / SHZ. This means that the Sync1 period must always be configured to be longer than (or equal) to the total sample period. If the sample period is instead longer than the Sync1 signal period, the device might still acquire exposure data when a subsequent Sync1 signal is recognized. As a result, the active acquisition operation would be aborted and the sample data resulting from this operation gets discarded. In addition, a new acquisition operation would be triggered.

In practice, this means that the MainDevice must configure the Sync1 signal frequency to match the lowest sample frequency that may occur. For example, for a minimum sample frequency of 500 Hz, set the Sync1 cycle time to 1/500 Hz = 2000 microseconds. In addition, it is important that the sample frequency must always be chosen to be an integral multiple of the Sync1 signal frequency. For example, if the minimum sample frequency and the Sync1 signal frequency are at 500 Hz, proper sample frequencies would be 500 Hz, 1 kHz, 1.5 kHz etc.

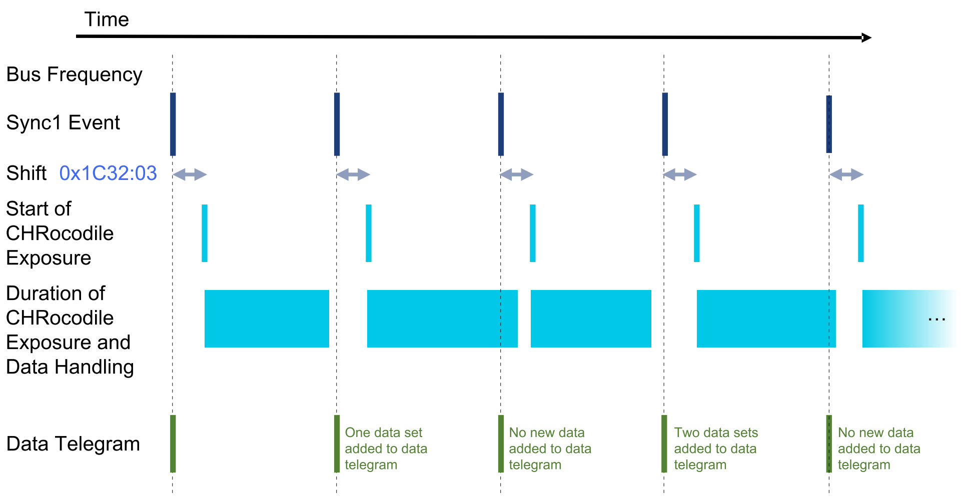
Scenarios

The data handling time (reading of the detector, data analysis, writing data into the buffer) is not necessarily constant. In addition, with the AAL setting, the exposure time of the CHRocodile sensor can vary.
Consequently, a data set might not be ready once an EtherCAT telegram reaches the device. In this case, no new data is added to the telegram. Once the data set is ready in the buffer, it is added to the subsequent telegram. That means that the number of data sets in a telegram can vary. The actual number is documented in PDO entry 0x1A00:03.
The CHRocodile clock is synchronized to the bus time and each data set has a time stamp. Thus, the time of recording for each data set is known, regardless of when it arrives at the MainDevice.

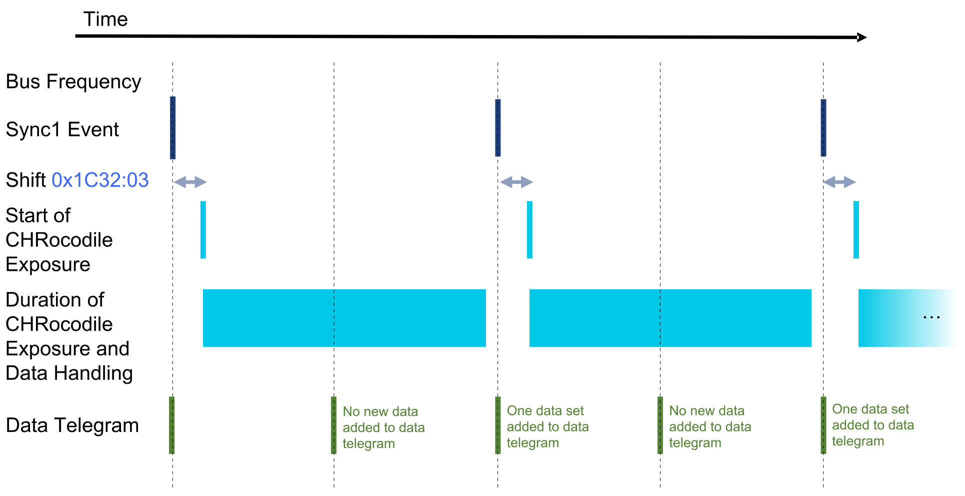
The figures below depict the timing in the scenarios:

1. CHRocodile sample frequency matches the bus frequency.
2. CHRocodile sample frequency is a fraction of the bus frequency.
3. CHRocodile sample frequency is a multiple of the bus frequency.

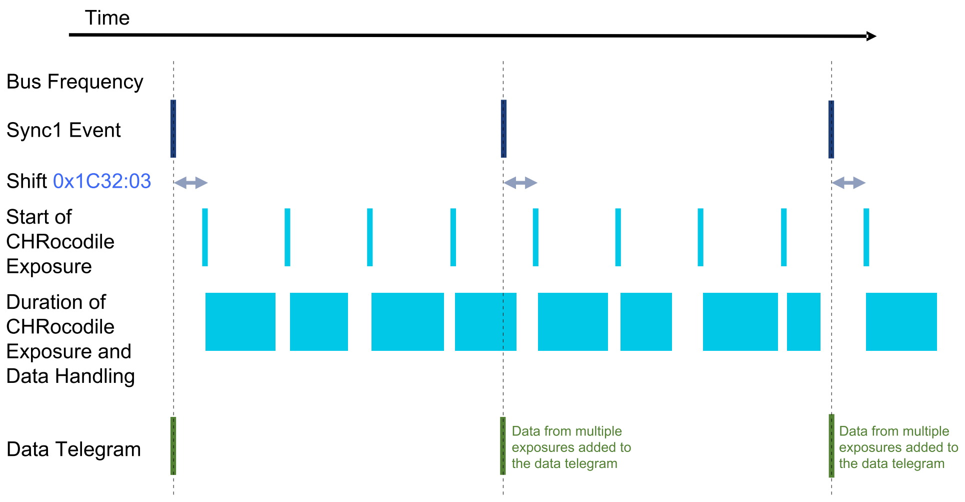
In detail:
1. CHRocodile sample frequency matches the bus frequency:


Time
Bus Frequency
Sync1 Event
Shift 0x1C32:03
Start of CHRocodile Exposure
Duration of CHRocodile Exposure and Data Handling
...
Data Telegram
One data set added to data telegram
No new data added to data telegram
Two data sets added to data telegram
No new data added to data telegram

2. CHRocodile sample frequency is a fraction of the bus frequency:


Time
Bus Frequency
Sync1 Event
Shift 0x1C32:03
Start of CHRocodile Exposure
Duration of CHRocodile Exposure and Data Handling
...
Data Telegram
No new data added to data telegram
One data set added to data telegram
No new data added to data telegram
One data set added to data telegram

3. CHRocodile sample frequency is a multiple of the bus frequency:


Time
Bus Frequency
Sync1 Event
Shift 0x1C32:03
Start of CHRocodile Exposure
Duration of CHRocodile Exposure and Data Handling
Data Telegram
Data from multiple exposures added to the data telegram
Data from multiple exposures added to the data telegram

Transfer timing

The data acquisition process of a CHRocodile device is fundamentally different compared to simple signal processing hardware (such as an Analogue-to-Digital conversion unit). While such devices can acquire and transfer data almost immediately when triggered, a CHRocodile sensor will always demonstrate a significant delay between the start of a sample acquisition operation and the transfer of the resulting sample data: First, sample data does not represent a "snapshot" of information acquired at a particular point in time. Instead, data is accumulated during a specific period (the exposure time). Second, after the exposure time has expired, the resulting information must be read from the detector and processed afterwards. The final sample data can only be transferred (using PDOs) when the entire process (exposure, detector read-out, post-processing) has completed. However, detector read-out and data processing can run in parallel to an exposure cycle. This means that after an exposure is complete, the subsequent exposure cycle can start immediately.

Due to the timing constraints outlined above and because the time required to complete the data processing operations can slightly vary, it is difficult to determine during which bus cycle a particular sample will actually be transferred. Thus, it is important to notice that an EtherCAT MainDevice receiving PDOs containing sample data must not assume that the respective data has been acquired during the most recent bus cycle. However, the MainDevice can configure the contents of a sample to include the start time of the associated exposure (represented by the global signal ID 64). Because the start time represents the EtherCAT bus time when that exposure started, this information can be used to determine during which bus cycle a specific sample has been acquired.

A CHRocodile device will transfer acquired samples in the following case: The Sync0 signal is encountered and the input latch time configured by means of the object dictionary entry 0x1C33:03 has expired and there are samples that have been acquired but not yet been sent. I.e., samples that have not yet been sent are sent in the next EtherCAT frame when Sync0 appears and the latch time has expired.

Available samples will be transferred with the next EtherCAT frame that passes through the device. The frequency of the Sync0 signal should hence usually match the bus frequency. The associated input latch time (i.e. the time during which the device will collect acquired samples for transfer within the next EtherCAT frame) must be chosen so that the device can still safely provide the data before the next EtherCAT frame. In practice, setting entry 0x1C33:03 to two-thirds of the bus cycle time (e.g. 666,000 nanoseconds, if the bus frequency is 1 kHz) has proven to be a suitable value.

First PDO

The CHRocodile device will provide additional information about the transferred samples in the first PDO (object dictionary index 0x1A00) transmitted in each EtherCAT frame:

PDO index:subindex	Signal index	Data type	Description
0x1A00:01	0x2406	u8	Frame counter
0x1A00:02	0x2400	u8	Number of configured samples
0x1A00:03	0x2401	u8	Number of recorded samples
0x1A00:04	0x2407	u8	Flags: Bit No. 0: 1 = sample(s) lost 0 = no losses

The EtherCAT MainDevice can use this information to determine how many samples have actually been transmitted and whether one or more samples were “lost” because more samples have been acquired than fit into the configured PDO set (see subchapter Oversampling below for more information on this topic).

Frame counter

The frame counter will be increased every time the CHRocodile device writes to the PDO data in an EtherCAT frame. If two or more consecutive frames contain PDO 0x1A00 with an identical frame counter, this indicates that the respective data is old. I.e., the device did not try to modify the PDO contents. Note that this scenario indicates a communication problem and should not occur.

4.6 Sample data and signal IDs

Overview

The following subchapter describes the use of signal IDs with EtherCAT on a CHRocodile device. The different types of signals are introduced. Furthermore, it is decribed how to define the signal set to be sampled and transferred by the device.
Note: For basic information on the composition of CHRocodile signal IDs, see chapter CHRocodile Signal IDs in this document.

Data transmission

Sample data will be transmitted using the EtherCAT PDO mechanism. An EtherCAT MainDevice can configure a set of PDOs describing the contents of a single sample or several samples (if oversampling is required). In the oversampling case, all configured PDOs must contain an identical signal set.
To configure the contents of a sample, the MainDevice must specify the signal set a sample shall comprise. A signal can either be a peak signal or a global signal. While peak signals represent acquired measurement results, global signals provide additional information such as time stamps, exposure information, encoder data and similar.
When using the EtherCAT interface, peak signals and global signals can be accessed as follows:

- All global signals are represented by dedicated object dictionary entries that can easily be mapped into a PDO.
- Likewise, common peak signals — peak signals that are frequently used by applications — are represented in the same manner.
- A dedicated mechanism allows a MainDevice to define arbitrary peak signals using the well-known signal encoding scheme supported by CHRocodile devices. These signals may then also be mapped into a PDO.

Signal ID mappings

The following table lists the object dictionary entries associated with the various device signals:

Object dictionary entries	Usage
0x2640 — 0x26FF	Global signals (starting from ID 64)
0x2700 — 0x3FFF	Common peak signals
0x2501 — 0x2560	Arbitrary signals

These different signal ID types are described in detail on the following pages.

Global signals

Global signals represent peak-independent signals like for example time stamp, encoder values, sample counter etc. The table below lists these global signals and the corresponding object dictionary entries (left column):

Index:Subindex	Signal name	Data type	Usage/Remarks
0x2640:00	64_StartTime	u32	In ns, free running time base
0x2641:00	65_Start_PositionX	s32	X-encoder position at the beginning of the exposure
0x2642:00	66_Start_PositionY	s32	Y-encoder position at the beginning of the exposure
0x2643:00	67_Start_PositionZ	s32	Z-encoder position at the beginning of the exposure
0x2644:00	68_Start_PositionU	s32	U-encoder position at the beginning of the exposure
0x2645:00	69_Start_PositionV	s32	V-encoder position at the beginning of the exposure
0x2646:00	70_Stop_PositionX	s32	X-encoder position at the end of the exposure
0x2647:00	71_Stop_PositionY	s32	Y-encoder position at the end of the exposure
0x2648:00	72_Stop_PositionZ	s32	Z-encoder position at the end of the exposure
0x264A:00	74_Stop_PositionV	s32	V-encoder position at the end of the exposure
0x264B:00	75_ExposureCount	u16	Exposure number of the first exposure in of the averaging cycle
0x264C:00	76_ExposureFlags	u16	Bits containing information about the exposure
0x264D:00	77_RealExpTimeNs	u32	Exposure time of detector (ns)
0x264E:00	78_RealLightingTimeNs	u32	Time when the light source is on during exposure (ns)
0x264F:00	79_TriggerLostCounter	u16	Number of trigger events that have been ignored since the last sample
0x2650:00	80_NumberOfValidPeaks	u16	Number of peaks that have been found in the spectrum
0x2651:00	81_TicketNumber	u16	Command ticket number, for diagnosis
0x2652:00	82_InterferomIntensity	float	Highest intensity on detector in percent of full well
0x2653:00	83_SampleCounter	u16	Counts samples that have been processed
0x2655:00	85_interfEnergy	float	Accumulated energy of the completed spectrum (interferometric mode only)
0x2656:00	86_Health_DSPLoad	u32	DSP load in 1/1000 of full load
0x2657:00	87_Health_TicketWrongOrder	u32	Error counter
0x2658:00	88_Health_UPPLostCount	u32	Error counter: spectrum lines lost during transmission
0x2659:00	89_Health_ExposureLostCount	u32	Error counter
0x265A:00	90_Health_UPPNotFinished	u32	Error counter
0x265B:00	91_PacketTimestampOffset	s32	Only defined for packet protocol; never ordered by SODX.
0x265D:00	93_Internal temperature	s16	Format: degree Celsius * 100 Note: Not available for all devices. If available, the measurement location is device-specific (for CHRocodile 2 S/S HS/S HP/SE/DPS: measurement at light source).
0x265E:00	94_LostAnalogValues	s16	Counts result values written to analog output that have been lost because of buffer overflow.
0x265F:00	95_PixelBlackValue	u16	Signal value on a black pixel of the detector
0x2660:00	96_Counter80MHzMSB	u16	16 most significant bits of a counter running at 80 MHz (for backwards compatibility reason)
0x2661:00	97_Counter80MHzLSB	u16	16 least significant bits of a counter running at 80 MHz (for backwards compatibility reason)

Index:Subindex	Signal name	Data type	Usage/Remarks
. . .	Reserved	. . .	Reserved
0x26F0	240_CALC0Result	f32	Applies only to double channel devices (CHRocodile 2 DPS)
0x26F1	241_CALC1Result	f32	Applies only to double channel devices (CHRocodile 2 DPS)
0x26F2	242_CALC2Result	f32	Applies only to double channel devices (CHRocodile 2 DPS)
0x26F3	243_CALC3Result	f32	Applies only to double channel devices (CHRocodile 2 DPS)

Note: For more information on the global signals supported by a CHRocodile device, see chapter CHRocodile Signal IDs in this document.

Peak signals

Peak signals relate to measured surfaces, for example Distance, Thickness, Intensity etc. The tables below list the pre-defined common peak signals supported by the CHRocodile software and the corresponding object dictionary entries (left column). The object dictionary entries 0x2700 through 0x2707 represent the Distance/Thickness signals for peaks No. 1 through No. 8. Subindices are used to represent the two channels of a double channel device (i.e. CHRocodile 2 DPS). Specifically, subindex “1” is dedicated to channel No. 0 while subindex “2” is dedicated to channel No. 1. For example:

Index:Subindex	Data type	Usage/Remarks
0x2700:01	f32	256_Distance/Thickness Peak 1 channel No. 0
0x2700:02	f32	256_Distance/Thickness Peak 1 channel No. 1
. . .	. . .	. . .
0x2707:01	f32	312_Distance/Thickness Peak 8 channel No. 0
0x2707:02	f32	312_Distance/Thickness Peak 8 channel No. 1

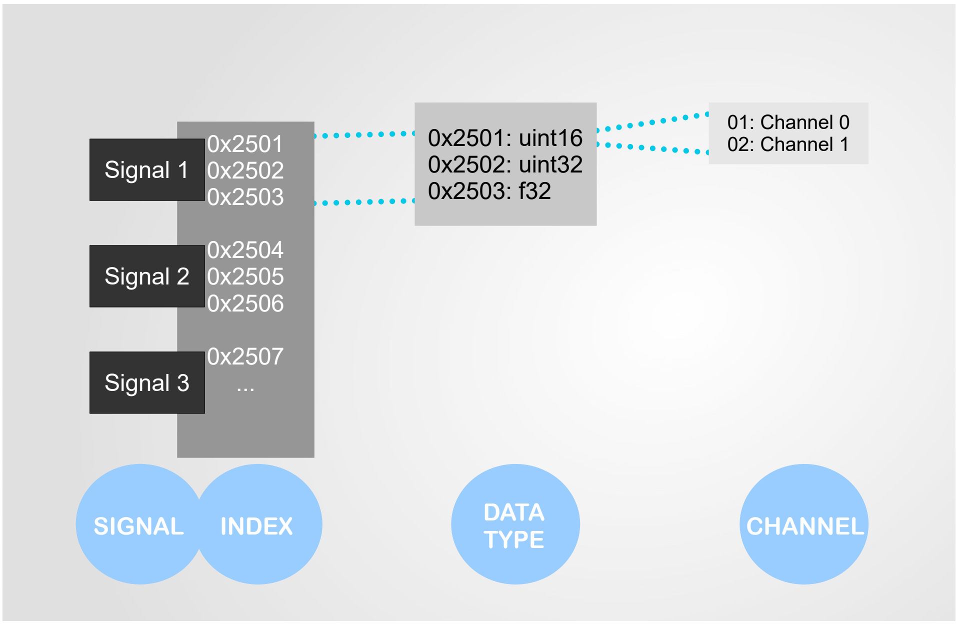
For single channel devices, always use subindex 1 of each entry. Note that because a double channel device (i.e. CHRocodile 2 DPS) will always measure on both supported channels, a PDO must also map both signals in the correct order, first channel 0, then channel 1, directly next to each other. For example, if a PDO includes 0x2700:01 “256_Distance/Thickness Peak 1 Channel 0”, it must also include 0x2700:02 “256_Distance/Thickness Peak 1 Channel 1”.

The object dictionary entries 0x2708 through 0x270F represent the Intensity/Quality signals for peaks No. 1 through No. 8. Again, subindices are used to represent the two channels of a CHRocodile 2 DPS device. Subindex “1” is dedicated to channel No. 0, while subindex “2” is dedicated to channel No. 1.
For example:

Index:Subindex	Data type	Usage/Remarks
0x2708:01	f32	257_Intensity/Quality Peak 1 channel No. 0
0x2708:02	f32	257_Intensity/Quality Peak 1 channel No. 1
. . .	. . .	. . .
0x270F:01	f32	313_Intensity/Quality Peak 8 channel No. 0
0x270F:02	f32	313_Intensity/Quality Peak 8 channel No. 1

Arbitrary signals

All peak signals that have not been pre-defined but are to be used must be created as arbitrary signals. The user has to define those signals manually. Example: Signal ID 16640 = Distance of peak No. 1 as 16 bit integer.
The mechanism for defining arbitrary signals as PDO contents requires an additional level of indirection: In a first step, the MainDevice must write the identification codes of the signals into a dedicated area of the object dictionary (starting at index 0x2501). In a second step, it must map the respective object dictionary entries to a PDO, selecting the appropriate signal data types and channel No. (which are represented by object dictionary entry subindices).
The object dictionary entries 0x2501 through 0x2560 are provided for signals in three possible data types: uint16, uint32, f32. Subindices 01 and 02 are used to select the device channel for double channel devices (for single channel devices, only subindex 01 is used). In total, the CHRocodile supports a maximum of 32 signals per sample. The following diagram illustrates the underlying concept:



The following table depicts some examples for defining an arbitrary signal:

Index:Subindex	Usage
0x2501:01	Unsigned 16-bit mapping type, channel No. 0
0x2501:02	Unsigned 16-bit mapping type, channel No. 1
0x2502:01	Unsigned 32-bit mapping type, channel No. 0
0x2502:02	Unsigned 32-bit mapping type, channel No. 1
0x2503:01	Floating point 32-bit mapping type, channel No. 0
0x2503:02	Floating point 32-bit mapping type, channel No. 1
. . .	. . .

The procedure for assigning an arbitrary signal is thus:

1. Choose the object dictionary entries representing the next free signal, starting from 0x2501 to 0x2560.
   Example: Signal 1 corresponds to 0x2501, 0x2502 or 0x2503.
2. Select an object dictionary entry representing the data type of the signal to be mapped.
   Example: For peak signal 16640 (distance/thickness peak No. 1 as 16 bit value), use 0x2501:01.

3. Select the subindex: For a single channel device, write the signal ID to subindex 01 of the respective entry (in this example: 0x2501:01). For a double channel device (CHRocodile 2 DPS), write the signal ID to both subindex 01 and 02.

Setting up PDOs

To define the PDOs and their contents a CHRocodile device shall send, a Main-Device may utilize the following Tx PDO entries in the object dictionary:

Index	Entry name	Contents
0x1A00	Control PDO 1	Sample information
0x1A01	Data PDO Ovs 1	Sample contents description
0x1A02	Data PDO Ovs 2	Oversampling PDO 2
...	...	...
0x1A49	Data PDO Ovs 73	Oversampling PDO 73

The PDO describing a single sample is defined by entry 0x1A01. The entries 0x1A02 through 0x1A49 are used for specifying data oversampling (see subchapter Oversampling below for details). The PDO at 0x1A00 contains four important special signals and must always be the first in the PDO assignment set. The signals comprised by this PDO are 0x2406 ("Frame counter"), 0x2400 ("Number of configured samples") and 0x2401 ("Number of recorded samples"). In addition, there is a special "Flags" signal. The EtherCAT-MainDevice can use the respective data to determine how many samples have actually been transmitted and whether the device has acquired more samples than could be fit into the configured PDOs (see subchapter Oversampling below for more information on these topics).

Instructions

To define the signal set to be sampled and transferred by the device, carry out the following steps:

1. Put the device into state "Pre-Operational".
   Note: This step is mandatory, because the PDO assignment can only be modified in this state and the CHRocodile approach to data transmission implements oversampling by means of assigning multiple PDOs with identical mapping.
2. Set the PDO entry 0x1A01, subindex 0, to "0".
3. Map the signals a sample shall consist of into the PDO 0x1A01, using predefined or MainDevice-defined arbitrary signals. To include arbitrary signals, the EtherCAT MainDevice must have defined them using the object dictionary starting at index 0x2501, as described above.
   Note: Take care to set the value of all sub-entries of the PDO that are not used to "0". For example, if you configure a sample consisting of three signals, make sure that you clear the sub-entries 0x1A00:04 through 0x1A00:20.
4. Set PDO entry 0x1A01, subindex 0 to the number of signals mapped.
5. Assign the PDOs to be transferred using the object dictionary entry "Tx PDO Assign" as follows:
   • Set the object dictionary entry 0x1C13, subindex 0, to "0".
   • Set object dictionary entry 0x1C13, subindex 1, to the value 0x1A00. This includes the "Sample information" data in the transmitted PDO set.
   • Set object dictionary entry 0x1C13, subindex 2, to the value 0x1A01. This includes the defined signal set in the transmitted PDO set.
   • Set object dictionary entry 0x1C13, subindex 0, to "2" (i.e., two Tx PDO objects assigned).
6. Set the object dictionary entry at index 0x23FF to "1" to apply the configured PDO set.

7. Put the device into state “Operational” (OP) or “Safe-Operational” (SAFEOP).

EtherCAT and SODX

Using the SODX command supported by other communication protocols of the CHRocodile firmware will not affect the sample data transmitted on the EtherCAT bus.

4.7 Downloading a spectrum

Overview

This topic explains how to request a spectrum from the device via EtherCAT. The spectrum data is provided as an array of 8-bit values actually representing 16 bit words. The values are stored as low byte/high byte pairs (“little endian”).
Note: When downloading large amounts of data as in the case of the spectrum download in Operational mode, process data (i.e. raw data collected by the sensor) may be lost.

Subindices usage

Subindices and their usage can be found in the following table:

Command	Index:Subindex	Data type	Usage
DNLD	0x2012:01	s32	Spectrum ID
	0x2012:02	u16	Exposure number
	0x2012:03	f32	Micrometers per bin (can be ignored in chromatic confocal measurement mode)
	0x2012:04	s32	Block exponent of spectrum data (can be ignored in chromatic confocal measurement mode)
	0x2012:05	octetstr	Spectrum data
	0x2012:06	u32	Spectrum data length in bytes

Using subindex 01, you can specify the spectrum type as follows:

Spectrum ID	Spectrum type
0	Raw spectrum (unprocessed signal)
1	Spectrum (chromatic confocal mode)
2	FFT spectrum (interferometric mode)

Instructions

To download a spectrum from the device, carry out the following steps:

1. Write the spectrum ID (see table above) to subindex 01. This will trigger the spectrum acquisition.
2. Attempt to read from the subindices 02 through 05. If the spectrum is not yet available, the device will respond with an “abort” message to the SDO read attempt.
3. Once the spectrum is available, an SDO read operation will complete successfully. The spectrum data can be read from subindex 05.

4.8 Oversampling

Overview

The CHRocodile device will acquire samples at the configured sample frequency. If this frequency is higher than the EtherCAT bus frequency, oversampling occurs. Sample data resulting from oversampling must be explicitly included in the PDO set. For this purpose, the CHRocodile device supports up to 73 PDOs reflecting the additional samples (for the exact derivation of this figure, see subchapter Useful hints).
Depending on the sample frequency, the EtherCAT MainDevice must configure a suitable set of PDOs representing the resulting oversampling. For example, for a bus frequency of 1 kHz and a sample frequency of 4 kHz, the resulting oversampling would be n = 4 kHz / 1 kHz = 4, hence requiring at least the sample PDOs 0x1A01 through 0x1A04 (actually, the MainDevice should configure up to three more “spare” PDOs in order to cope with jitter effects).
If too few PDOs have been configured, the CHRocodile will be unable to transfer all data acquired during a bus cycle, and sample data loss will result (the device will set bit No. 0 of the “Flags” signal in PDO 0x1A00 in this situation). In contrast, configuring too many PDOs does not impose problems. An application can hence configure a static number of PDOs suitable for the maximum sample frequency intended.

Sample information

The contents of PDO “Control PDO” (0x1A00) from the sensor ships four bytes with the following layout:

Index:Subindex	Signal Index	Data type	Usage
0x1A00:01	0x2406	u8	Frame counter
0x1A00:02	0x2400	u8	Number of configured samples
0x1A00:03	0x2401	u8	Number of recorded samples
0x1A00:04	0x2407	u8	Flags: Bit No. 0: 1 = Sample(s) lost 0 = All acquired samples transmitted Bit No. 1—7: Reserved

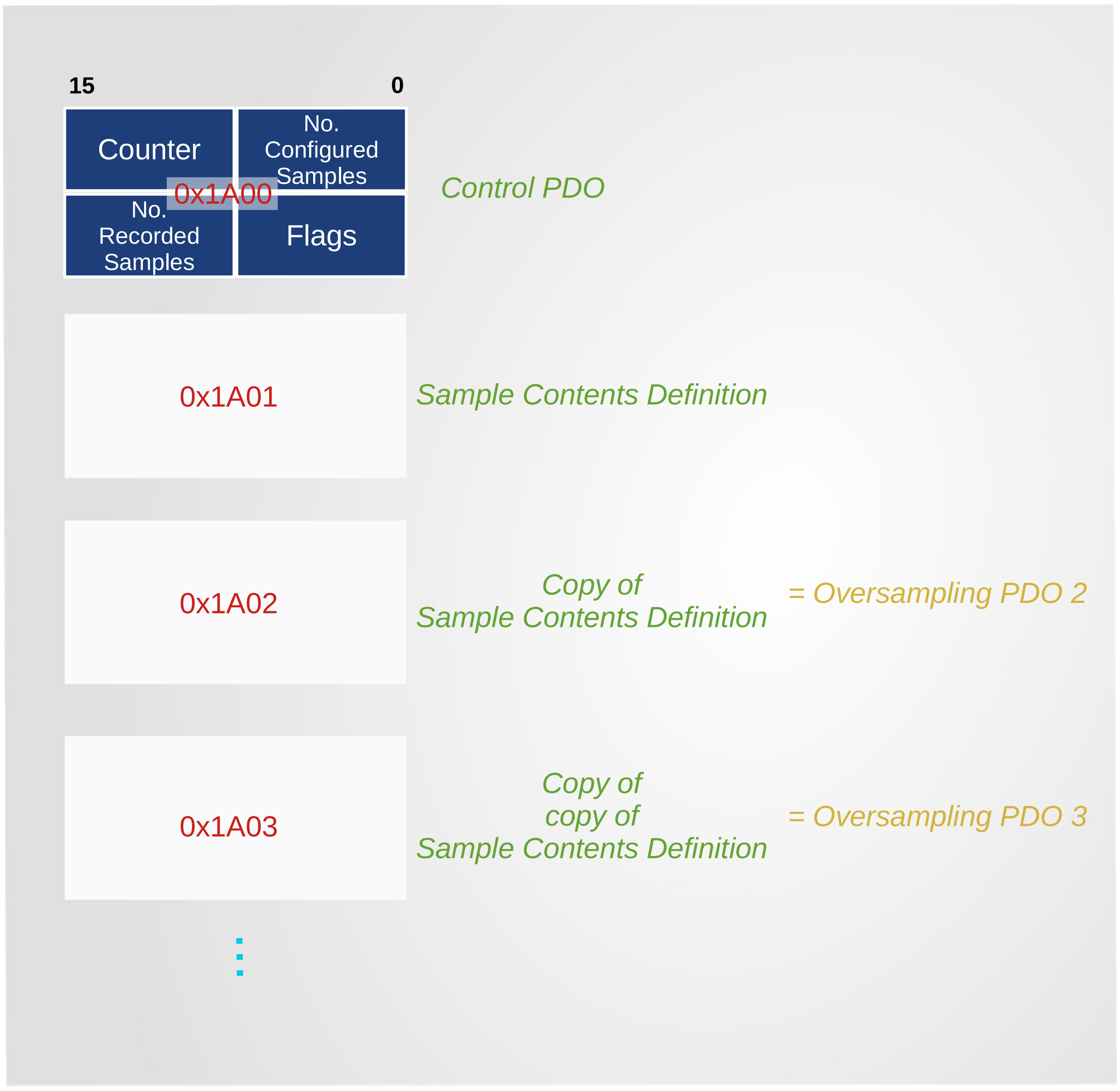
The EtherCAT MainDevice can use the respective data to determine how many samples have actually been transmitted and whether one or more samples were “lost” because more samples have been acquired than fit into the configured PDO set.

The number of samples actually transmitted can vary. To determine the actual number of samples received, inspect the contents of the first PDO 0x1A00: The second entry (“Number of configured samples”) contains the number of configured samples, i.e. the number of sample PDOs you have configured. The third entry (“Number of transmitted samples”) specifies the number of samples that have actually been recorded and transmitted.

The frame counter will be increased every time the device accesses the PDO data in an EtherCAT frame. If two or more consecutive frames contain PDO 0x1A00 with an identical frame counter, this indicates that the respective data is actually old, i.e., the device did not make an attempt to modify the PDO contents for some reason. Note that this scenario indicates a communication problem and should not occur.

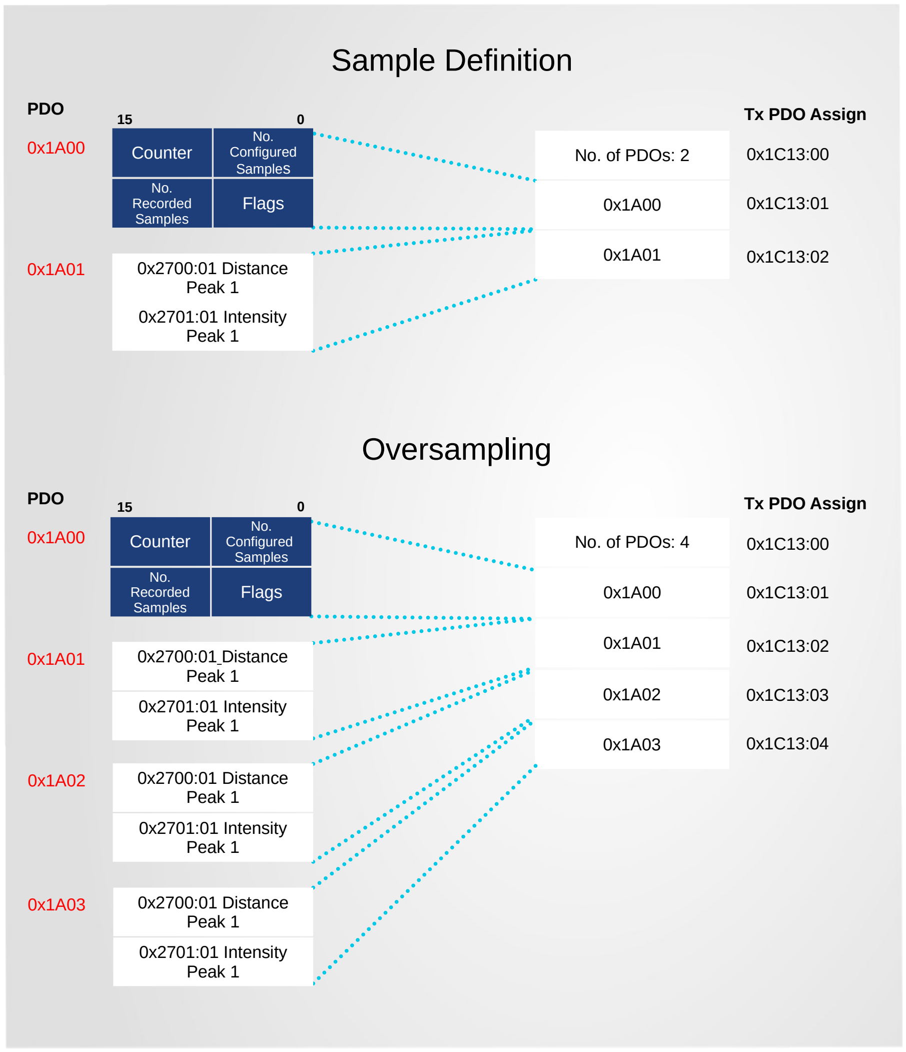
Illustration: Layout

The following illustration describes the layout of sample information and sample content. The sample content is defined in 0x1A01. In case of oversampling, copies of the sample contents definition of 0x1A01 are transferred:



Illustrated example

The following illustration describes the case that a simple sample with two signals (256 and 257) is ordered (upper part of illustration). The lower part illustrates the case that the same sample with an oversampling factor of “3” is required.



Instructions

To configure an oversampling PDO set, carry out the following steps:

1. Define the signal set of the first sample using the PDO at object dictionary entry 0x1A01 as described in subchapter Sample data and signal IDs.
2. For the first oversampling PDO, map an identical signal set using the PDO at object dictionary entry 0x1A02 (copy of sample contents definition of 0x1A01).
3. Repeat step 2 for each additional oversampling PDO, using the PDOs at index 01A03 and following. Notice: Make sure that unused sub-entries of any PDO are set to 0.
4. Specify the PDOs to be transmitted as described in subchapter Sample data and signal IDs.
5. Set the object dictionary entry at index 0x23FF to “1” to apply the configured PDO set.
6. Put the device into state “Operational” (or “Safe-Operational”).

4.9 States and capabilities

Overview

The EtherCAT state machine provides the following states:

- Initialization (INIT)
- Pre-Operational (PREOP)
- Safe-Operational (SAFEOP)
- Operational (OP)

This topic defines, which processes are carried out in which state.

States in details

The following table lists which tasks are possible in the above mentioned states:

State	SDO communications	PDO transmission	PDO reception	Capabilities
INIT	☐	☐	☐	Communications are being initialized. Communications are not possible.
PREOP	🟦	☐	☐	SDO (message) communications are possible in this state, as well as CHRocodile command execution and PDO configuration. This state is entered after initialization has been completed. It is used to configure the device.
SAFEOP	🟦	🟦	☐	In this state, PDO transmissions are possible in addition to SDO (message) communications.
OP	🟦	🟦	🟦	Normal communication state. PDO transmissions possible, SDO communications also supported.

4.10 Useful hints

Overview

This section collects frequently asked EtherCAT questions and provides an answer. For further questions, please contact the Precitec Support team. Our support will assist you personally with all problems related to EtherCAT.

FAQs

Frequently asked questions on EtherCAT when using a CHRocodile device:

? Can I change the sample frequency in any EtherCAT bus state and what happens then?

! Yes. The sample frequency (object dictionary entry 0x2004) can be changed in any EtherCAT bus state (except INIT). However, the sample frequency can only be changed within the limits set by the Sync1 signal and the configured oversampling PDOs:

- If the trigger mode ECAT is selected, the sample frequency of the sensor must not be set to a value lower than the Sync1 signal frequency. Otherwise, samples will not be acquired at the configured sample frequency, but at the frequency of the Sync1 signal.
- The sample frequency must not be set to a value that exceeds the maximum oversampling supported by the configured PDO set. Otherwise, acquired samples will get lost.

See subchapters Acquisition triggering and data transfer and Oversampling for details.

? How do I proceed if the oversampling and therefore the number of PDOs changes? Do I really have to go back to the PREOP state and change the PDO configuration?

! Yes, this is due to the fact that EtherCAT provides the Tx PDO mapping only in this state. In order not to constantly reconfigure the PDOs (for which you have to set the whole EtherCAT bus to the PREOP state) when the sample frequency changes, you should configure your PDO set for the maximum occurring sample frequency of the application.

? Is the maximum oversampling limited?

! Yes. For CHRocodile devices, the current maximum oversampling is limited to 73. This is due to the fact that oversampling is calculated as follows (the “+ 3” in the equation below accounts for a few samples for jitter):
oversampling = SHZ / EtherCAT_bus_freq + 3

? Downloading spectrum data: Is it possible to retrieve spectra in every EtherCAT state?

! No. It is not possible to retrieve spectra in the INIT state, but it is possible in the other EtherCAT states. However, retrieving spectra in OP state could lead to data loss in the process data.

? Can I use SODX to order CHRocodile signals like when using packet protocol or dollar protocol?

! No. In EtherCAT, the SODX command for configuring the set of signals to be transmitted by the device is not supported. Instead, use the EtherCAT PDO mapping and assignment mechanism for configuring the sample contents as described in subchapter Sample data and signal IDs.

Chapter 5

Encoder Interface

5.1 Encoder interface

Overview

The CHRocodiles support interfacing incremental encoders in order to relate real world positions to the measurements in real time. Depending on the CHRocodile type 5 or 3 (CHRocodile C/Mini+) encoder channels are supported. The encoder inputs are digital differential RS422-level A/B quadrature inputs. 120 Ohm termination resistors can be activated through control pins for subgroups of the encoder signals (see device manual).

The encoder interface records the exact position of the encoder-equipped axes at the acquisition moment of the measurement. Furthermore, it allows to trigger measurements at defined positions (see subchapter Encoder trigger control)

The encoder interface is controlled with the ENC command, which has several subfunctions.

The encoder functions let the user set and query the current encoder position, define the source signals for the encoder counters and provide means to automatically set (preset) the encoder counter to a programmable value based on external signals.

Format

The encoder command has the following format: ENC ⟨*axis*⟩ ⟨*function*⟩ ⟨*arg*⟩ where *axis* is the axis index 0..4, and *function* is:

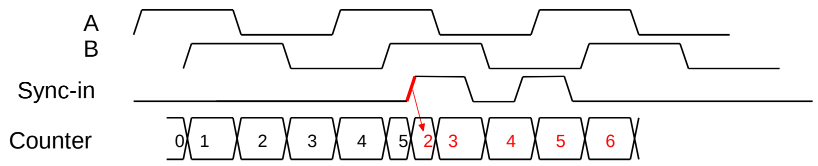
- **Function 0:** Set the encoder counter immediately to *arg* (example: `ENC 1 0 123` – sets the current axis 1 counter position to 123).
- **Function 1:** Set count source for counter:
  - *arg* = 15: Quadrature input of the axis defined by the axis argument (A/B encoder count). This is the standard case of operation and permits forward and backward position counting.
  - Alternatively, single inputs A0, B0, A1, . . . , B4 and Sync-in can be used for pulse counting (see pulse count mode below). In that case, the counter only counts up. The meaning of *arg* is as follows:

0:	A0
1:	B0
2:	A1
3:	B1
. . .	. . .
. . .	. . .
. . .	. . .
9:	B4
10:	Sync-in (not for CHRocodile C/Mini)

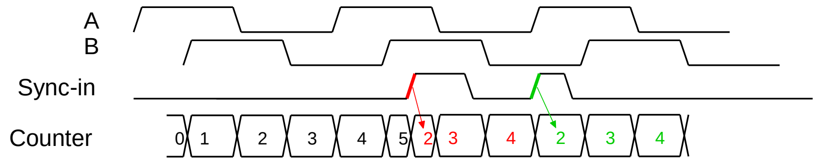
- **Function 2:** Set preload value. *arg* sets the value that will be loaded into the counter when a preload event occurs. The preload event is selected with function 3, see below.
- **Function 3:** Set preload event. The preload event generation is defined by *arg* where the bits have the following meaning:
  - Bit 7: active (1) / inactive (0) – turns preloading on / off
  - Bit 6: State (1) / Edge (0) – determines whether the preload event is triggered on state or edge of the selected input. Please note that the state mode only works in conjunction with Bit 4 (always) set to 1.
  - Bit 5: Falling edge (or low state) (1) / Rising edge (or high state) (0)
  - Bit 4: always (at every event) (1) / single (only once) (0)
  - Bits 3..0: Preload Event selector:

0:	A0
1:	B0
2:	A1
3:	B1
. . .	. . .
. . .	. . .
. . .	. . .
9:	B4
10:	Sync-in (not for CHRocodile C/Mini)
11–14:	Reserved
15:	Immediate preload

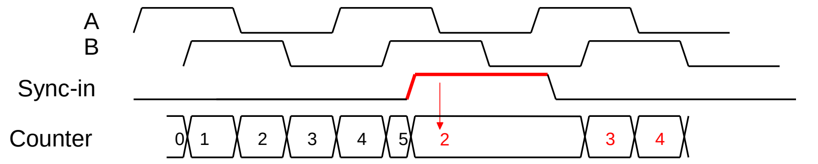
The following waveforms illustrate different preload scenarios. In all examples, the preload value register has been preloaded with the value of 2. The preload event is derived from the Sync-in input.



Preload event behaviour after Command ENC0 3 138 (active/edge/rising/single/Sync-in)



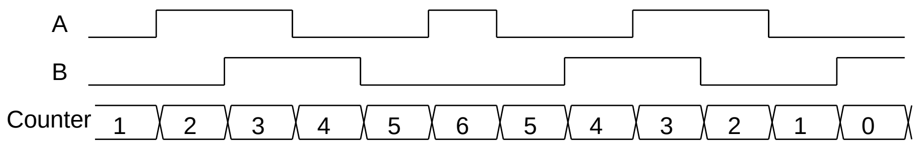
Preload event behaviour after Command ENC0 3 154 (active/edge/rising/multiple/Sync-in)



Preload event behaviour after Command ENC0 3 202 (active/state/high/single/Sync-in)

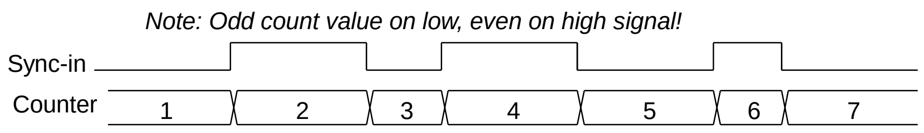
Quadrature count mode

In quadrature mode (default), the phase shift between the rectangle signals on A and B decides if the counter is incremented or decremented. As an example, we assume a quadrature signal on encoder channel 0 (`ENC 0 1 15`):



Pulse count mode

If a single encoder signal Ax or Bx or Sync-in is selected as count source, then the counter is in pulse count mode. It increments on every edge and never counts down. Another speciality of this mode is that the bit 0 of the count value always reflects the current state of the selected input: If the input is low, the count will always be odd and if the signal is high, the count will always be even. Thus, the state of an input can be monitored. For the example below we assume that Sync-in has been selected as count source for counter 0 (`ENC 0 1 10`):



Note: For CHRocodile C/Mini, Sync-in is not provided as count source.

Examples

Examples:

Input	Comment
`ENC 0 1 15`	Connects counter 0 to quadrature encoder input A0/B0.
`ENC 3 1 15`	Connects counter 3 to quadrature encoder input A3/B3.
`ENC 4 1 10`	Connects counter 4 as pulse counter to Sync-in (not for CHRocodile C/Mini).
`ENC 0 2 1234`	Sets the preload value of counter 0 to 1234.
`ENC 0 3 178`	(178 = 128 + 0 + 32 + 16 + 2) Configures the encoder counter to load the preload value 1234 into the counter 0 whenever a falling edge occurs on A1, i.e. the A input of encoder channel 1. Can be used e.g. for axis referencing.

Chapter 6

Triggered Measurements

6.1 Triggered measurements

Overview

The CHRocodile can perform measurements either in regular intervals (the so-called free running mode) or as a reaction on trigger events in one of the so-called trigger modes. There are 3 different trigger modes: *Trigger once* (TRG), *Trigger each* (TRE) and *Trigger window* (TRW).

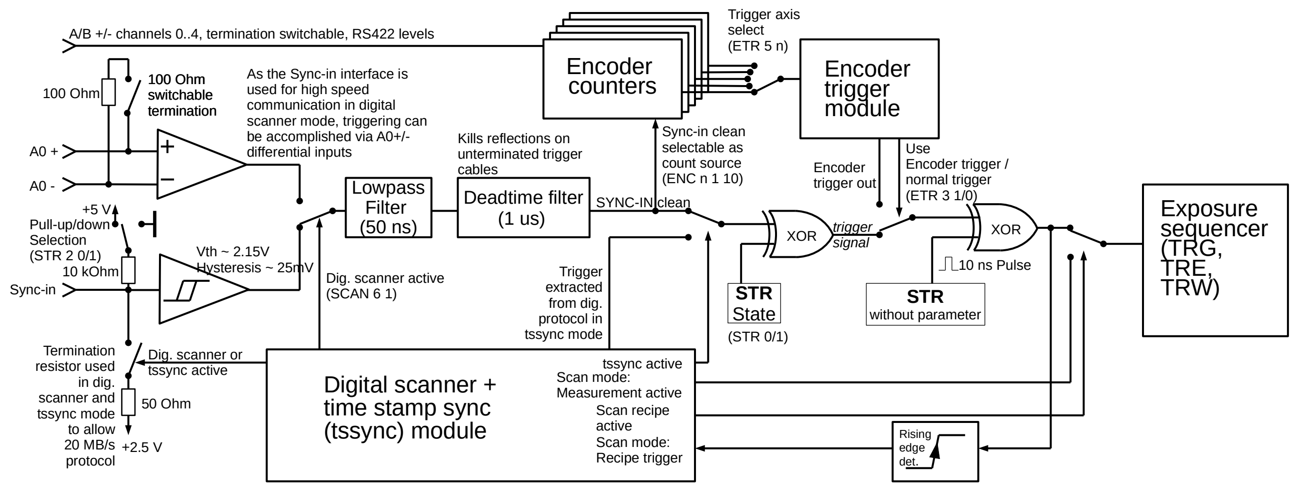
Trigger event

Trigger events can be generated by external signals, via software command or by the encoder trigger module. If encoder based trigger generation is disabled by ETR 3 0 (normal trigger), a trigger event is generated at each rising edge of the *trigger signal*. Otherwise, if encoder triggering is enabled using ETR 3 1, a trigger event occurs every time the specified encoder counter reaches the next trigger position. For details on encoder triggering, see subchapter Encoder trigger control.

Trigger signal

The trigger signal is a logical XOR combination of the digital input signal Sync-in and the software trigger signal set by command STR. Thus, the signal edge of the Sync-in signal that generates a trigger event can be selected (STR 0: rising edge, STR 1: falling edge). The STR state is initialized to 0 when the device boots up. For details, see STR (Set trigger). The level of Sync-in signal is pulled high +5 V by an internal pullup resistor (10 kOhm) if the connector is left open.

The following figure illustrates the trigger signal treatment:



A/B +/- channels 0..4, termination switchable, RS422 levels
Encoder counters
Trigger axis select (ETR 5 n)
Encoder trigger module
100 Ohm
100 Ohm switchable termination
As the Sync-in interface is used for high speed communication in digital scanner mode, triggering can be accomplished via A0+/- differential inputs
A0 +
A0 -
+5 V
Pull-up/down Selection (STR 2 0/1)
10 kOhm
Sync-in
Vth ~ 2.15V Hysteresis ~ 25mV
Lowpass Filter (50 ns)
Kills reflections on unterminated trigger cables
Deadtime filter (1 us)
SYNC-IN clean
Sync-in clean selectable as count source (ENC n 1 10)
Encoder trigger out
Use Encoder trigger / normal trigger (ETR 3 1/0)
XOR
trigger signal
XOR
10 ns Pulse
STR State
(STR 0/1)
STR without parameter
Exposure sequencer (TRG, TRE, TRW)
Dig. scanner active (SCAN 6 1)
Trigger extracted from dig. protocol in tssync mode
Termination resistor used in dig. scanner and tssync mode to allow 20 MB/s protocol
Dig. scanner or tssync active
50 Ohm
+2.5 V
Digital scanner + time stamp sync (tssync) module
tssync active
Scan mode: Measurement active
Scan recipe active
Scan mode: Recipe trigger
Rising edge det.

Setting trigger mode

The following table lists commands to set the trigger mode:

Command	Description
TRG (Trigger once)	This command stops data acquisition. The sensor waits for a trigger event. When the trigger event occurs, data acquisition will continue in free run mode with the currently selected sample rate.
TRE (Trigger each)	With this command the sensor enters the trigger each mode. Every trigger event will generate one single sample. This mode is particularly useful in conjunction with encoder based trigger generation.
TRW (Trigger window)	With this command, the sensor enters the window trigger mode. The signal acquisition is stopped as long as the trigger signal is low. During the high periods of the trigger signal, the device's behavior is identical to the free run mode. The first exposure of a high period is synchronized to the rising edge of the trigger signal. When the signal goes low again, the acquisition stops.
CTN (Continue in free run mode)	The command CTN resumes free running operation mode.

TRE and averaging

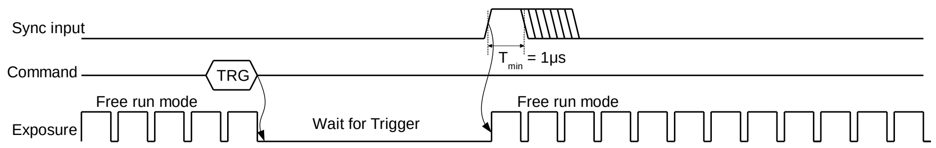
In Trigger each mode, one trigger event will start a sequence of AVD*AVS exposures. The AVD*AVS exposures are executed with the current sample rate (SHZ). One averaged result will be output after the exposure sequence. There will be only one trigger event on the Sync-out signal per n samples to be averaged. The pulse marks the beginning of the first exposure of the averaging interval. In the other trigger modes, there is one Sync-out pulse for every exposure, regardless of averaging.

Notice

The window trigger mode (TRW) can not be used meaningfully in combination with encoder triggering (ETR 3 1).

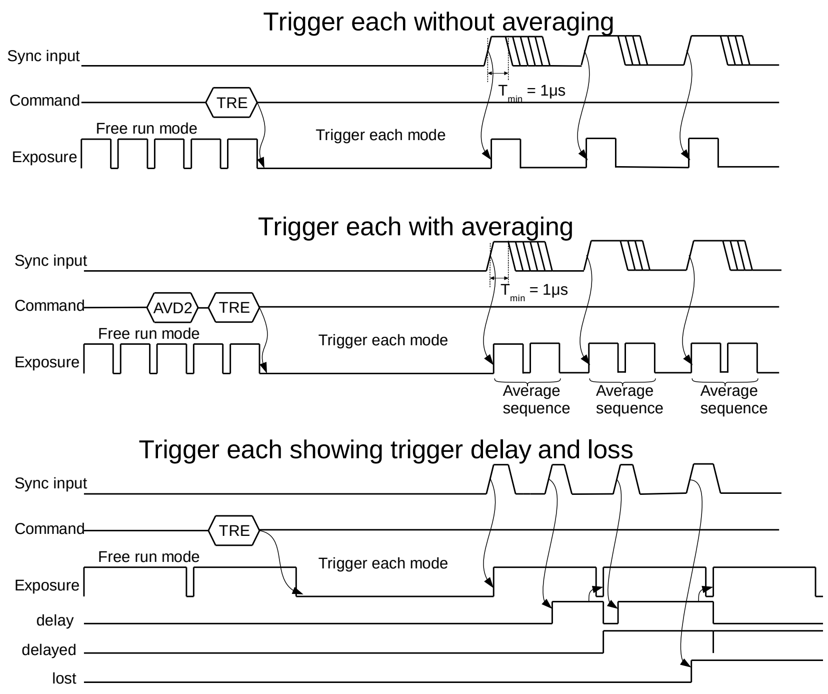
Trigger once

The figure below illustrates Trigger once mode:



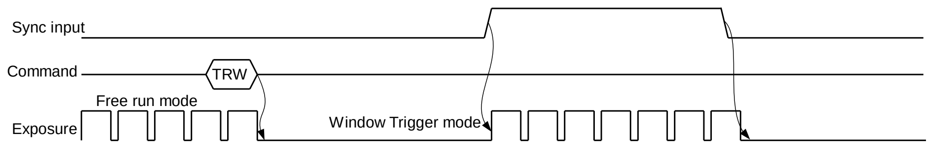
Trigger each

The figure below illustrates Trigger each mode:



Trigger window

The figure below illustrates Trigger window mode:



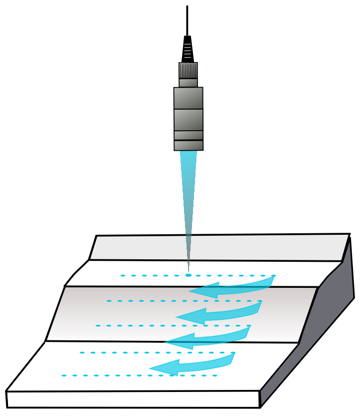
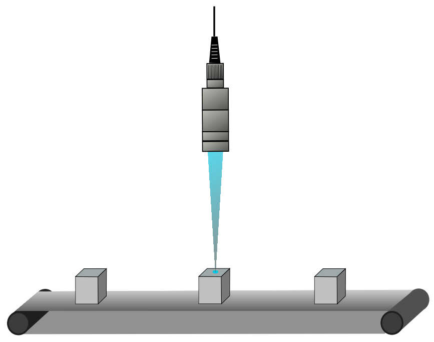
6.2 Encoder trigger control

Overview

The following sections deal with the use of the encoder trigger. Encoder positions are captured via the encoder input and allow precise allocation of the measuring points to the axis positions. In addition to just recording the axis position at the measurement moment, the encoder unit can be used to trigger measurements at exact positions.

Encoder trigger

The encoder trigger can operate in two ways (see the following pages for details):

Mode	Description	Illustration
Roundtrip trigger	For the use in raster scanning applications, where one scanning axis goes back and forth and the trigger is used to align the measurement to the axis positions	
Endless trigger	For applications where the encoder primarily moves in one direction as for example in production lines or on rotation tables	

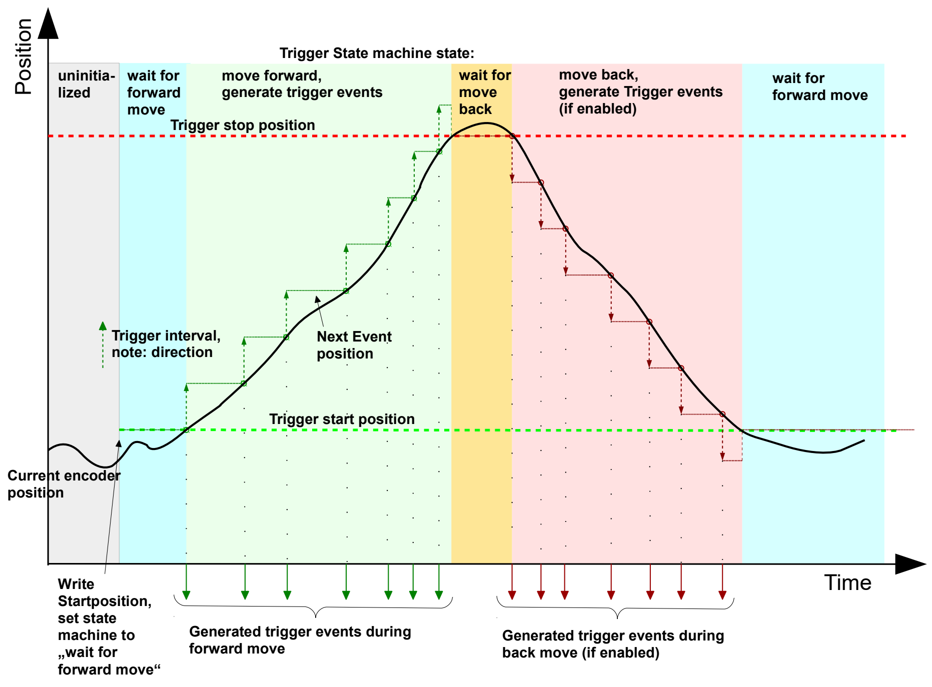
6.2.1 Encoder roundtrip trigger

Overview

The roundtrip trigger mode is provided for the use in raster scanning applications, where one scanning axis goes back and forth and the trigger is used to align the measurement of the CHRocodile to the scan axis positions. You can generate trigger events at programmable regular position intervals beginning with a starting position and ending with a stop position. You can select if trigger events are only generated during the move in one direction or also during the return movement.

Illustration

The following figure illustrates the operating principle of the roundtrip trigger. In this mode the encoder trigger is implemented as a state machine:


Position
Trigger State machine state:
uninitia-lized
wait for forward move
move forward, generate trigger events
wait for move back
move back, generate Trigger events (if enabled)
wait for forward move
Trigger stop position
Trigger interval, note: direction
Next Event position
Trigger start position
Current encoder position
Time
Write Startposition, set state machine to „wait for forward move“
Generated trigger events during forward move
Generated trigger events during back move (if enabled)

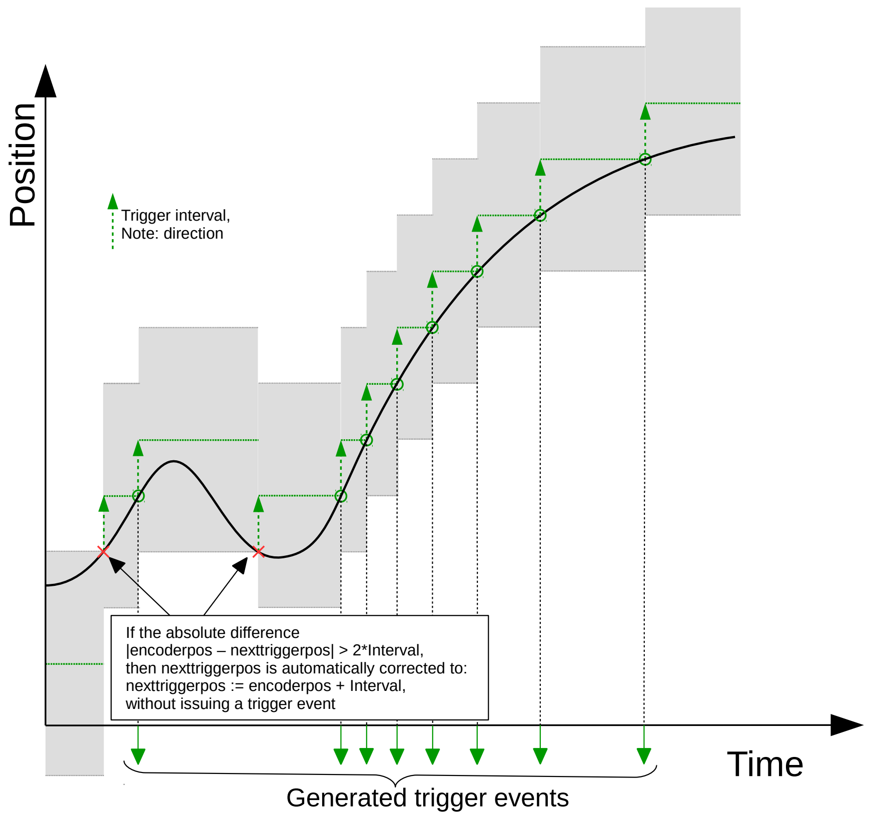
- In order to function correctly, the state machine has to be initialized to the “wait for forward move” state. This is accomplished automatically when setting the start position.
- In the “wait for forward move” state, the encoder trigger state machine waits for the encoder counter of the selected axis to pass the start position (in either direction) where it generates the first trigger event. The state machine changes to the “move forward” state. The trigger interval is added to the start position in order to calculate the next trigger position.
- If the trigger position is reached, a trigger event is generated. The trigger interval is added to the trigger position in order to generate the next trigger position. This step is repeated until the stop position is encountered. The generation of trigger events is then stopped and the state machine changes to the “wait for move back” state.
- If triggering during return movement is selected, the state machine waits for the stop position to be passed once again and then changes to “move back” state. It then generates trigger events similarly to the forward movement (the trigger interval is now subtracted instead of added) until the start position is reached. The state machine then goes back to the “wait for forward move” state.
  If no trigger during return movement is selected, the state machine waits for

the start position to be passed over (during return movement) and then passes to the “wait for forward move” state.

6.2.2 Encoder endless trigger

Overview The endless trigger mode is designed for applications where the encoder primarily moves in one direction as for example in production lines or on continuously rotating samples. In order to use this mode, only a trigger interval has to be parametrized.

Illustration The following figure illustrates the operating principle of the endless trigger:


Position
Trigger interval,
Note: direction
If the absolute difference
|encoderpos – nexttriggerpos| > 2*Interval,
then nexttriggerpos is automatically corrected to:
nexttriggerpos := encoderpos + Interval,
without issuing a trigger event
Time
Generated trigger events

6.2.3 Encoder trigger programming

ETR command family

The ETR (Encoder trigger control) command groups several functions related to encoder triggering. It controls how encoder counters can generate trigger events. For more information about trigger modes, see TRG (Trigger once), TRW (Trigger window) and TRE (Trigger each) description. The command format is as follows:

ETR <*function*> <*arg*>

where *function* is:

- 0: Set start position (*arg* = start position (int) to set, see figure)
- 1: Set stop position (*arg* = stop position (int) to set, see figure)
- 2: Set trigger interval (*arg* = trigger interval (float)). Note that the interval can be given in fractions of encoder counts! (e.g. float value 100.5). The next trigger position is calculated by adding the Interval value to the last trigger position. **If (Stop position) - (Start position) is negative, the trigger interval value must also be negative!**
- 3: Select encoder trigger source
  - *arg* = 0: (default) Deactivate encoder trigger, trigger by Sync-in or software (STR command).
  - *arg* = 1: Activate encoder trigger.
- 4: Enable trigger during return movement
  - *arg* = 0: (default) Encoder trigger is only active during the movement from start position to stop position.
  - *arg* = 1: Encoder trigger is also active during the return movement from stop position to start position.
- 5: Choose axis: *arg* = index of the encoder counter used as trigger source. Default source is encoder counter 0.
- 6: Reserved (0)
- 7: Endless/Roundtrip trigger
  - *arg* = 0: (default) Round trip trigger. Start/stop positions are used.
  - *arg* = 1: Endless trigger. Generate one trigger event on every interval regardless of any start/stop position.

Examples

Example command sequence 1:

No.	Command	Type Description
1	`ENC 0 0 0`	(Re-)Set current encoder position of axis 0 to 0.
2	`ETR 4 0`	Don't trigger during return movement.
3	`ETR 5 0`	Choose axis 0 as encoder trigger source.
4	`ETR 7 0`	Round trip trigger mode (for back and forth movements, as opposed to continuous movements)
5	`ETR 1 1000`	Set trigger stop position to 1000.
6	`ETR 2 10.5`	Set trigger interval to 10.5 counts.
7	`ETR 0 100`	Set trigger start position to 100 (must come after ETR 2 and ETR 1 in order to reset the trigger state machine).
8	`ETR 3 1`	Activate Encoder Trigger (deselect Sync-in / software as trigger source).
9	`TRE`	Set CHRocodile to trigger each mode.

This command sequence sets the current counter 0 to zero and configures the trigger logic to start triggering at position 100 of encoder counter 0, stop counting at position 1000, and fire one trigger every ten counts. The last command TRE (Trigger each) sets the device to trigger each mode. At positions 100, 110, 121, . . . , 971, 982 and 992 one sample will be acquired every 10 or 11 counts (10.5 as interval, 85 samples in total).

Example command sequence 2:

No.	Command	Type Description
1	`ENC 2 0 0`	(Re-)Set current encoder position of axis 2 to 0.
2	`ETR 5 2`	Choose axis 2 as encoder trigger source.
3	`ETR 1 2000`	Set trigger stop position to 2000.
4	`ETR 2 -100`	Set trigger interval to -100 (negative, as start position is greater than stop position.
5	`ETR 0 2950`	Set trigger start position to 2950 (must come after ETR 2 and ETR 1 in order to reset the trigger state machine).
6	`ETR 3 1`	Activate encoder trigger (deactivate Sync-in / software as trigger source).
7	`TRE`	Set CHRocodile to trigger each mode.

In example 2, the start position is greater than the stop position. Therefore, a negative value must be given to the interval value. In this setting, 10 samples will be generated.

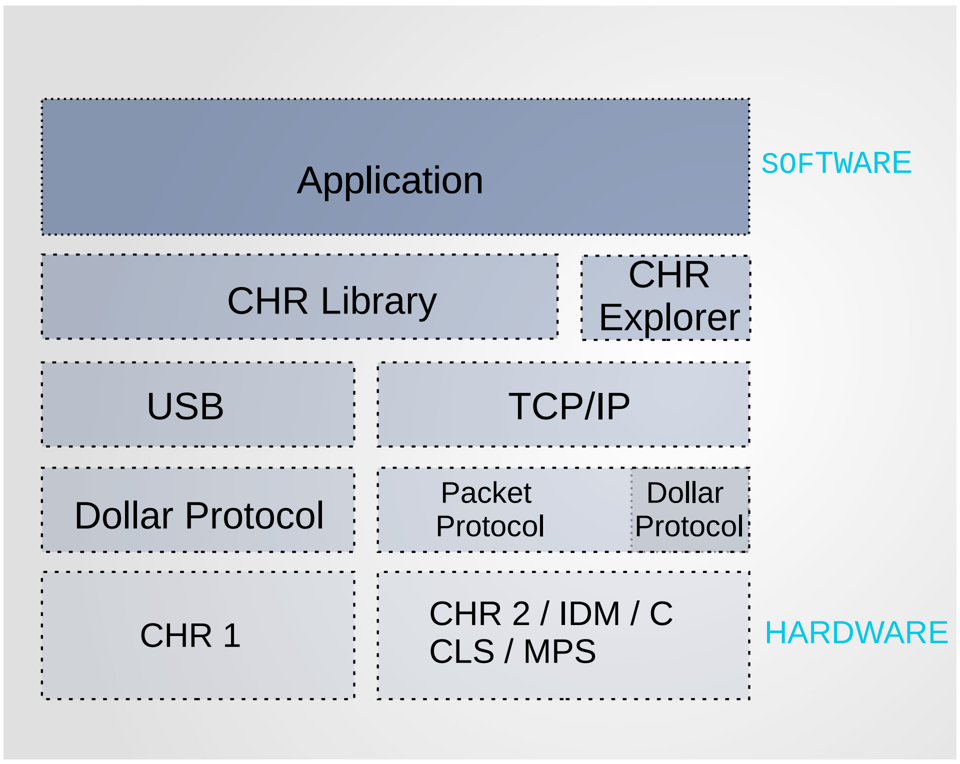
Chapter 7

CHRocodile Library

7.1 Using the CHRocodile Library (CHRocodileLib)

Overview

The following figure gives a general overview on integration of the CHRocodile device into the application running on measuring systems. As illustrated, communication with the CHRocodile device can be carried out on the hardware side (using packet protocol or dollar protocol) or on the software side (using CHRocodile Library or CHR Explorer):



Note: Multi-channel devices such as CHRocodile MPS and CLS as well as the double channel device CHRocodile 2 DPS do not fully support the dollar protocol. Only the binary packet protocol is available for data transfer with those devices.

CHRocodile Library

The library provides a universal interface for integrating CHRocodile devices. It encompasses basic functions for communicating with devices to acquire data and to set up the CHRocodile. This is an enormous advantage, as the different communication protocols mentioned in the previous chapters (dollar protocol, packet protocol) are device-specific and support a different set of commands.

Further features

Main features of the CHRocodile Library are:

- Multi-device support: Multiple devices can be operated in parallel and independently.
- Multi-connection support: One device can be operated by multiple library-hosted logical connections where each connection can send commands, receive replies, updates, and data independently.
- Synchronous / asynchronous operation
- Multi-threading support: Sending commands asynchronously and receiving data can be done in different threads.

Which devices?

The CHRocodile Library supports all kinds of CHRocodile devices:

- CHRocodile 1
- CHRocodile 2
- CHRocodile CLS /MPS / 2 DPS
- CHRocodile C / Mini
- Precitec IDM

The CHRocodile Library provides the same interface for all these devices. Therefore, to connect to a different CHRocodile device, the client software does not need to change anything.

Which platform?

The CHRocodile Library is available as:

- Dynamic Link Library (DLL) for Windows XP, 7, 8, 10, 11
- Shared object library for the Linux platform

Please contact our Support team for details.

Further details

For a more detailed description of the CHRocodile Library features and functions, refer to the document “CHRocodileLib XX API Reference” (see the USB stick included in delivery).

Chapter 8

Command Availability

8.1 Availability of commands

Purpose In some cases it may be useful to know if other CHRocodile devices also support a certain command. To facilitate the request, the table below provides an overview of command availability throughout all CHRocodile classes.
Differences in implementation are only very briefly indicated – so for details, the user should consult the description under chapter 2 “CHRocodile Commands” of the Command Reference of the corresponding device.

Comparison Availability of commands on different CHRocodile devices:

Command name	CHR 2	Precitec IDM	CHR C/Mini	CHR CLS/MP S/DPS	CHR CLS 2/2Pro	Comment
AAL	🟦	🟦	🟦	🟦	☐	*CHR 2 DPS:* The exposure of both channels is adapted separately, but the AAL setpoint value is common for both channels.
ABE	🟦	🟦	🟦	🟦	☐	*CHR C/Mini:* ⟨argn⟩ value range 0–999, other device types 0–500
ALI	🟦	☐	☐	☐	☐	Only relevant for devices with a pilot laser installed.
ANAM	🟦	🟦	■	■	☐	*CHR C/Mini:* Only CHR C with Extension Box or CHR Mini+ *CHR CLS/MPS/DPS:* Only double channel devices (DPS) Note the different output options for different CHRocodile devices. Refer to the command description for details.
ANAX	🟦	🟦	■	🟦	☐	*CHR C/Mini:* Only CHR C with Extension Box or CHR Mini+; new subcommand ANAX ref *CHR CLS/MPS/DPS:* ⟨arg0⟩ . . . ⟨arg9⟩, other device types ⟨arg0⟩ . . . ⟨arg7⟩
ASC	🟦	🟦	🟦	🟦	☐	
AVD	🟦	🟦	🟦	🟦	☐	*CHR C/Mini:* ⟨arg0⟩ value range 0–999, other device types 0–10000
AVDS	🟦	🟦	🟦	☐	☐	
AVM	☐	☐	🟦	☐	☐	Available for Firmware 1.4 or later
AVS	🟦	🟦	🟦	☐	☐	*CHR C/Mini:* ⟨arg0⟩ value range 1–999, other device types 1–256 *CHR 2:* In double exposure mode (DCY other than 100%), spectra averaging is not possible. *CHR 2 SX:* AVS is not supported.

Command name	CHR 2	Precitec IDM	CHR C/Mini	CHR CLS/MP S/DPS	CHR CLS 2/2Pro	Comment
BCAF	🟦	🟦	🟦	🟦	☐	
BDR	🟦	🟦	🟦	🟦	☐	*CHR C/Mini:* Note that hardware handshaking is only available when RS232 mode is selected by the SRPT command.
BIN	🟦	🟦	🟦	🟦	☐	
CALC	☐	☐	☐	■	☐	*CHR CLS/MPS/DPS:* Only double channel devices (DPS)
CONF	🟦	🟦	🟦	🟦	🟦	*CHR CLS 2/2Pro:* Command still exists, but different response strings.
CRA	🟦	🟦	☐	🟦	🟦	*CHR 2:* <arg0>: 0–2048; <arg1>: 0–2048 *CHR CLS:* <arg0>: 0–124; <arg1>: 50–174 *CHR MPS:* <arg0>: 0–22; <arg1>: 2–24 *CHR CLS 2/2Pro:* <arg0>: 0–136; <arg1>: 64–200 Note: Default values are specific for each device type.
CRDK	🟦	🟦	🟦	🟦	☐	
CTN	🟦	🟦	🟦	🟦	🟦	Common
DCY	🟦	🟦	☐	☐	🟦	Works slightly different for *CHR CLS 2/2Pro*, see DCY command description in chapter 2 for details.
DNLD	🟦	🟦	🟦	🟦	🟦	This command works very different for each CHRocodile class. Refer to the command description for details.
DRK	🟦	🟦	🟦	🟦	🟦	This command works very different for each CHRocodile class. Refer to the command description for details.
DTF	🟦	🟦	🟦	🟦	☐	*CHR C/Mini:* response different
DWD	🟦	🟦	🟦	■	☐	*CHR CLS/MPS/DPS:* Only double channel devices, works different than for other CHRocodiles.
ENC	🟦	🟦	🟦	🟦	🟦	*CHR C/Mini:* Only CHR C with Extension Box or CHR Mini+; less axes supported, furthermore no ENC 4 available
EQN	🟦	🟦	☐	☐	☐	
ETR	🟦	🟦	🟦	🟦	🟦	*CHR C/Mini:* Only CHR C with Extension Box or CHR Mini+
EWD	🟦	🟦	☐	☐	☐	

Command name	CHR 2	Precitec IDM	CHR C/Mini	CHR CLS/MP S/DPS	CHR CLS 2/2Pro	Comment
FDK	🟦	🟦	🟦	🟦	🟦	This command works very different for each CHRocodile class. Refer to the command description for details.
GAN	🟦	🟦	□	□	□	
GLE	🟦	🟦	🟦	🟦	🟦	*CHR CLS 2/2Pro:* Modifications of error codes and sub-error codes.
GLL	□	□	🟦	□	□	
IDE	🟦	🟦	🟦	🟦	🟦	This command provides different responses for each CHRocodile class. Refer to the command description for details.
IDMP	□	🟦	□	□	□	
IDMX	□	🟦	□	□	□	
IPCN	🟦	🟦	🟦	🟦	🟦	*CHR C/Mini:* <arg4> default is 4 instead of 2 for other CHRocodile class. *CHR CLS 2/2Pro:* <arg4> is 3.
LAI	🟦	🟦	🟦	🟦	🟦	*CHR CLS/MPS/DPS:* Only CHR 2 DPS: second argument valid for second channel
LMA	🟦	🟦	🟦	🟦	□	
LOC	🟦	□	□	□	□	Only relevant for CHR 2 devices with front panel keyboard
LTC	□	□	🟦	□	□	
MAN	🟦	🟦	□	🟦	□	
MED	🟦	🟦	□	■	🟦	*CHR CLS/MPS/DPS:* Only CHR 2 DPS *CHR CLS 2/2Pro:* Command format and functionality completely changed. This median command allows 1D filtering of distances or intensities using a configurable filter width (window size) or 2D filtering of distances or intensities using a configurable number of exposures.
MEDX	🟦	🟦	□	■	□	*CHR CLS/MPS/DPS:* Only CHR 2 DPS
MESG	🟦	🟦	🟦	🟦	🟦	Common
MMD	🟦	🟦	□	□	🟦	

Command name	CHR 2	Precitec IDM	CHR C/Mini	CHR CLS/MP S/DPS	CHR CLS 2/2Pro	Comment
NCH	□	□	□	□	🟦	
NOP	🟦	🟦	🟦	🟦	🟦	*CHR CLS 2/2Pro:* Different functionality available on CHRocodile CLS 2 series: NOP (number of peaks) has been extended. It is possible to specify the number of peaks to detect and specify the number of peaks to transmit. This is in addition to normal NOP usage.
OFN	🟦	□	🟦	■	□	*CHR C/Mini:* Different arguments compared to standard *CHR CLS/MPS/DPS:* Only CHR 2 DPS
OPD	🟦	🟦	🟦	🟦	🟦	*CHR C/Mini:* Not all operational data is available as query. *CHR CLS 2/2Pro:* Not all operational data is available as query.
ORDB	■	□	□	□	□	Only available on CHR 2 SX
POD	🟦	🟦	□	□	□	
PSM	🟦	🟦	🟦	🟦	🟦	Common
QTH	🟦	🟦	□	□	□	
RAF	🟦	🟦	□	□	□	
RST	🟦	🟦	🟦	🟦	🟦	Common
SCA	🟦	🟦	🟦	🟦	🟦	Common
SCAN	🟦	□	□	□	□	
SEN	🟦	🟦	🟦	🟦	🟦	*CHR C/Mini:* optional ⟨arg1⟩ (extended measuring range enabled/disabled) *CHR CLS/MPS/DPS:* On multi-channel devices, one calibration can contain multiple probes.
SENI	■	□	□	□	□	Only available for certain CHRocodile 2 representatives and a firmware version $\geq$ R1.4.1. Refer to the command description for details.
SENX	🟦	🟦	🟦	🟦	🟦	
SFD	🟦	🟦	🟦	🟦	🟦	*CHR CLS/MPS:* Sets parameters to factory default values except the network settings.
SHZ	🟦	🟦	🟦	🟦	🟦	
SODX	🟦	🟦	🟦	🟦	🟦	*CHR C/Mini:* Up to 16 signal ID´s to be ordered, other device types up to 32 signal ID´s.

Command name	CHR 2	Precitec IDM	CHR C/Mini	CHR CLS/MPS/DPS	CHR CLS 2/2Pro	Comment
SRI	🟦	🟦	🟦	🟦	☐	*CHR C/Mini:* The reply of the query includes values of 3 layers even though not all are used according to the setting of NOP.
SRPT	☐	☐	🟦	☐	☐	
SRT	🟦	🟦	🟦	🟦	☐	
SSQ	🟦	🟦	🟦	🟦	☐	
SSU	🟦	🟦	🟦	🟦	🟦	Common
STA	🟦	🟦	🟦	🟦	🟦	Common
STO	🟦	🟦	🟦	🟦	🟦	Common
STR	🟦	🟦	🟦	🟦	🟦	*CHR 2, CHR C/Mini, CLS 2/2Pro:* Optional additional argument that specifies the pull-up/pull-down behaviour to be applied to the Sync-in trigger signal.
TABL	🟦	🟦	🟦	🟦	🟦	This command works very different for each CHRocodile class. Refer to the command description for details.
THR	🟦	🟦	🟦	🟦	🟦	Note that there are different defaults and value ranges for each CHRocodile class.
TMOD	🟦	🟦	🟦	🟦	🟦	
TRE	🟦	🟦	🟦	🟦	🟦	
TRG	🟦	🟦	🟦	🟦	🟦	Common
TRW	🟦	🟦	🟦	🟦	🟦	
ULFW	🟦	🟦	🟦	🟦	☐	This command works very different for each CHRocodile class. Refer to the command description for details.
VER	🟦	🟦	🟦	🟦	🟦	*CHR C/Mini:* Provides different set of key-value pairs than the CHR 2 platform. *CHR CLS 2/2Pro:* Command still exists, but different response string.

🟦 *Command implemented;* ■ *Command available for certain representatives (see comment);* ☐ *Command not implemented*

Chapter A

Further Information

A.1 Specification of selected tables for single channel devices

Confocal calibration

A confocal calibration table yields a micrometer value for every pixel of the spectrum. The table descriptor contains some additional data such as the calibration date. A confocal calibration table for a single channel device is structured as follows:

```
typedef struct
{

u32 Calib_Date;          // simple date coding:
                         //      byte 0 / 5 LSBs: day;
                         //      byte 1 / 4 LSBs: month;
                         //      byte 2         : year - 2000
u32 TableMagicNumber;    // 0x6F4F960A
s32 MeasurementRange;    // in micrometers
u32 SerialNumber;        // of the probe
} ts_ConfocTableDescriptor

typedef struct
{
float LUT[n];                                        // in micrometers
ts_ConfocTableDescriptor ConfocTableDescriptor; //as declared above
} ts_ConfocTable
```

Note that n is not statically defined. Instead, the constant size of the table descriptor is subtracted from the total size of the binary data block. The remaining size divided by four is the number of spectral pixels of the calibration.

Interferometric calibration

This table provides a translation from CCD pixels to wavelengths and is structured as follows:

```
typedef struct
{
    float Lambda[1024 - 1];      //in nanometers
    s16 Reserve;
    s16 LastCCDPixel;            // number of detector pixels
} ts_Lambda_Pix;
```

Refractive indices

A user-defined table provides the wavelength-dependent index of refraction for a given material. 32 $n(\lambda)$ pairs can be passed to this list. For wavelengths between two such pairs, n will be interpolated. Each table can be named with a string of up to 32 characters. The checksum is the ones complement of the sum of the whole table (without the leading “checks” member), casted as 32 integers.

```
typedef struct
{
    float lambdanm; // lambda in nm
    float RI;       // refractive index, real part
    float RI_imag;  // refractive index, imag. part (reserved)
} ts_lambdaRI;

typedef struct
{
    u32 checks;                  //not used
    s8 name[32];                 //0-terminated ASCII string
    ts_lambdaRI lambdaRI[32];    //as declared above
}ts_RI_Sourcetable;
```

A.2 Specification of selected tables for multi-channel devices

Confocal calibration

There are several formats for multi-channel calibration tables. One can differentiate between calibrations with a single optical probe and thus a common measuring range for all channels (CLS) and a multi-channel device using different single and multi-channel probes at the same time (MPS). In the first case, there is one common header (MCLUTDescriptor) for all channels. In the second case, which is signaled by the measurement range entry in the main header being 0, every channel has its own subheader that carries channel specific data.

For the actual calibration data of the channels, there are 2 formats which are distinguished by the "degree" entry of the main header. If "degree" is smaller than or equal to 9, the calibration data is in polynomial format, otherwise it is in sampled format.

Polynomial format

This format is deprecated.

In order to save table space, multi-channel calibrations are defined by polynomials for every channel. Polynomials of 8th degree (max.) are used. The complete calibration structure for one probe consists of a descriptor containing some meta information and the polynomial coefficients in the order:

```
(a8, a7, a6,..., a0) channel0,
(a8, a7, a6,..., a0) channel1,
...
```

Sampled format

For higher precision, the sampled format is used which consists of sampled distance values as 16 bit integers, normalized to the measuring range (unsigned short d16=(u16)(dist*65536.0/measuringrange)). The degree entry of the main header is used to describe the number of samples.

Notice: All the following structures are "packed". Make sure that the compiler does not add any alignment padding!

Main header

```
typedef struct
{

    u32 Calib_Date;          // simple date coding:
                             //      byte 0 / 5 LSBs: day;
                             //      byte 1 / 4 LSBs: month;
                             //      byte 2          : year - 2000
    u32 MCLUTMagicNumber;    // = utCLSConfocalCalibration (5)
    s32 MeasurementRange;    // in micrometer
    u32 SerialNumber;        // serial number of the probe
    s16 degree;              // degree of polynomial or number of sampled points
    s16 NrOfChannels;        // number of channels (fibers) actually used

    u32 spectrumlength;      // only used with sampled format:
                             // length of the spectrum in pixels
                             // covered by the ``degree'' samples
    float channelpitch_um;   //
    u32 reserve[25];         // Reserved words for alignment reasons
} ts_MCLUTDescriptor
```

Subheader (MPS)

```
typedef struct
{
    u16 MeasurementRange;        // in micrometer
    s16 res16;                   // = 0, reserve for alignment
    u32 SerialNumber;            // serial number of the probe
    float DistToNextChannel_um;
} ts_MCLUTChannelDescriptor;
```

Channel specific (MPS)

Don't use any `sizeof()` construct of this type as the actual size is determined by the number of samples given by the "degree"-entry in the main header.

```


typedef struct
{
    ts_MCLUTChannelDescriptor MCLUTChannelDescriptor;
    u16 distances[1];            //array of "degree" distances in micrometers
}t_sMCLUT_indiv_Channel;
```

Chapter B

Topic-Related Command Lists

B.1 Timing related commands

B.2 Dark/white reference related commands

B.3 Peak detection related commands

B.4 Trigger related commands

B.5 Dispersion related commands

B.6 Communication settings related commands