TECHNICAL INFORMATION

Protocol and command reference

OD7000

SICK

Sensor Intelligence.

Described product

OD7000

Manufacturer

SICK AG
Erwin-Sick-Str. 1
79183 Waldkirch
Germany

Legal information

Original document

This document is an original document of SICK AG.

Contents

1 OD7000 Protocols

1.1 Introduction

Overview

The following section 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.

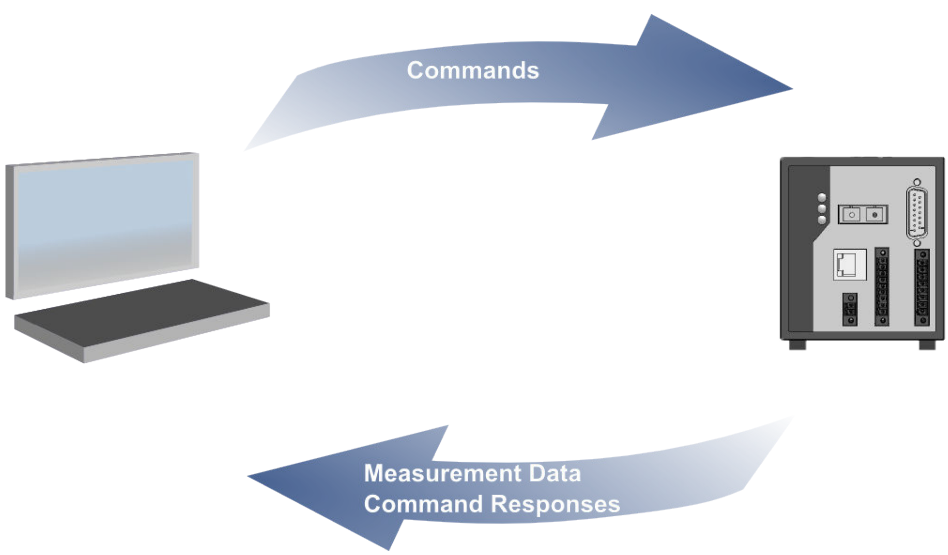
Characteristics

During normal operation, the device continuously sends data packets. There are two unique protocols for communication between the sensor and external clients, typically being computers. One is the dollar protocol, which is also used by OD7000 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 • 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.

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 OD7000 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).

Interface

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

NOTE
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 baud rate 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 comma characters (,). 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 endianness of the values depends on the signal bit width according to the table below (MSB: most significant byte, LSB: least significant byte):

Bit width	Endianness	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 OD7000 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 OD7000 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 OD7000 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

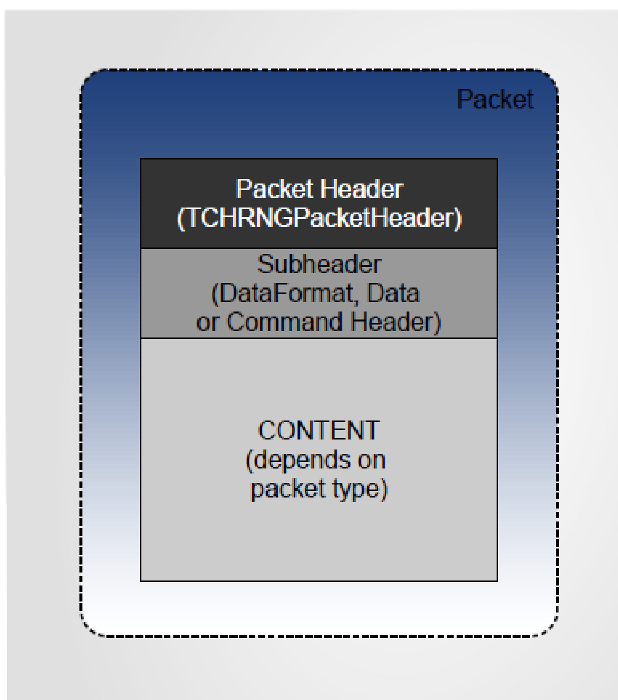
The fields defined in the binary packet protocol are arranged in little endian order, i.e., the least significant byte is followed by more significant bytes.

Packet types

The packet protocol currently has three different types of packets:
- Data packet
- Data format packet
- Command packet

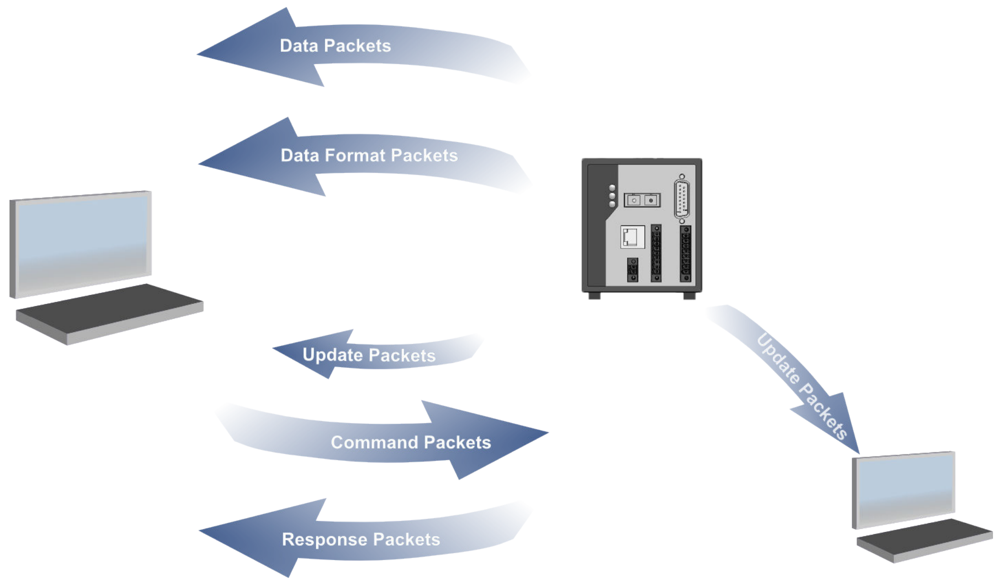
General structure

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.



Illustration

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



Functions

The following table denotes the functions of the different types of packets:

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 three 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:
  - `cCrdFlagQuery = 0x0001`. This flag declares a parameter query. The sensor will respond with a command packet containing the current parameter value.
  - `cCrdFlagError = 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.

  - `cCrdFlagWarning = 0x4000`. Returned in the response packet in case the command has been executed, however, with possibly modified parameterization or other implications that the user has not expected. Additional information may be added as arguments.
  - `cCrdFlagUpdate = 0x2000`. The device may send “updates”, i.e., command responses to the connected client whenever the current configuration / parameterization 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 4 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 12 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.
```

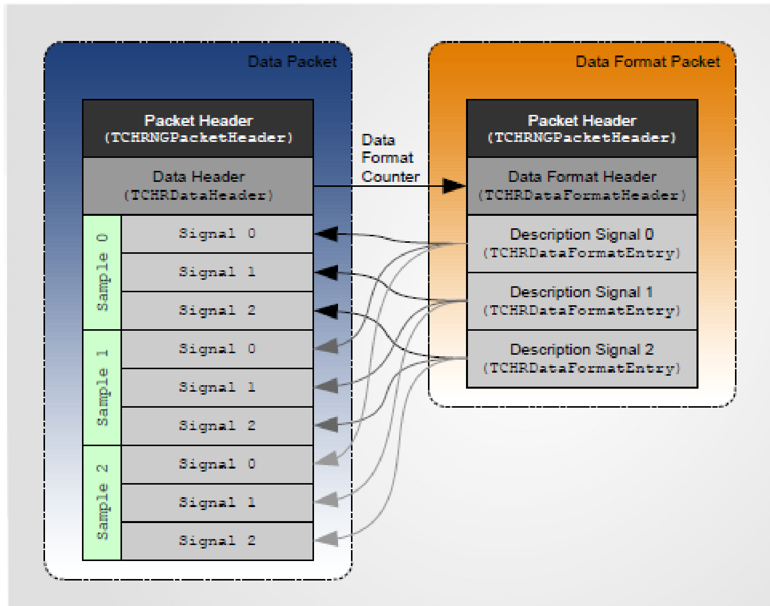
1.3.3 Data format packet definition

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:



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, see "OD7000 Signal IDs", page 66.

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. 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.

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”) is always 1.
- The number of the first channel to be transferred (2 bytes, integer) is always 0.
- 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 global signals (i.e., signals which appear only once per exposure, 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;
};
```

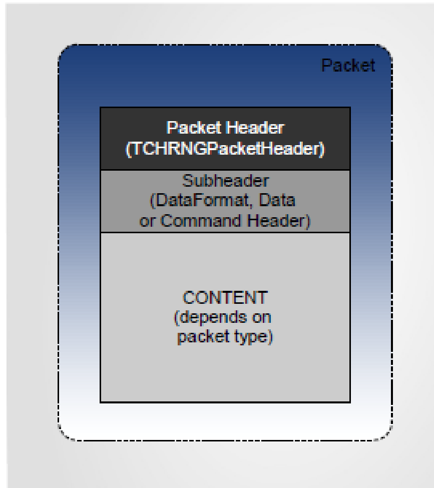
For each signal, one entry of type `TCHRDataFormatEntry` is added to the data format packet.

```
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 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 channel signals) values. 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 the OD7000, 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>

...

Source code

Below is the data packet header declaration.

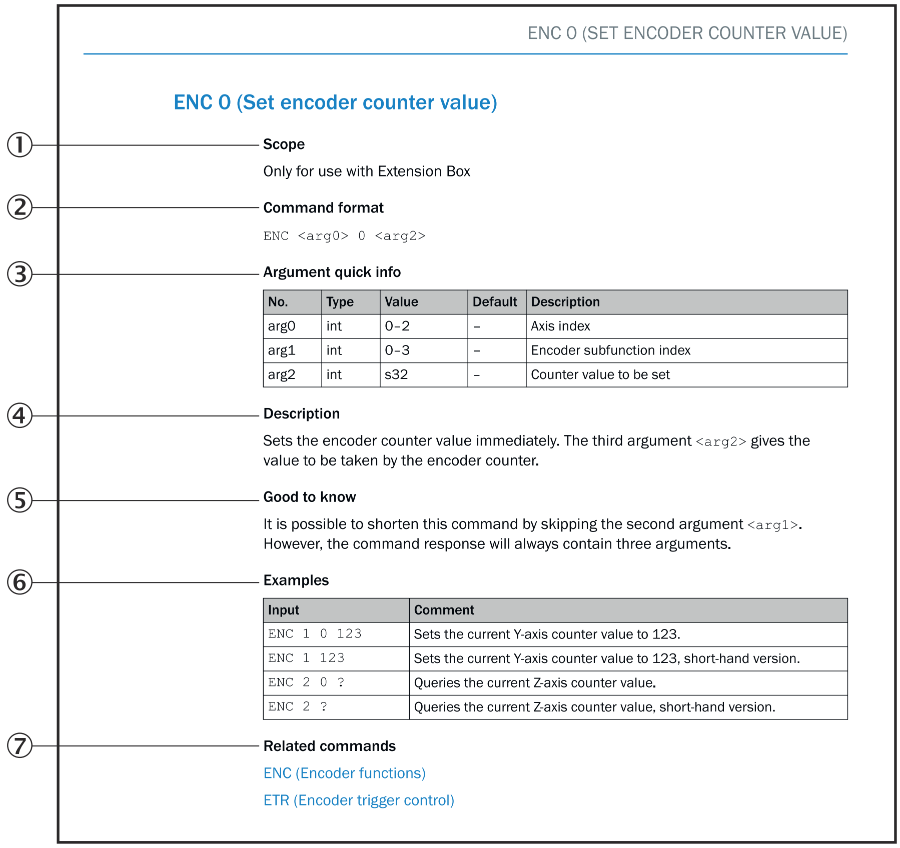
```

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
};
```

2 OD7000 Commands

2.1 How to read the Command Reference

Illustration

The following figure indicates the different blocks in the Command Reference.


ENC 0 (SET ENCODER COUNTER VALUE)

ENC 0 (Set encoder counter value)

① **Scope**

Only for use with Extension Box

② **Command format**

`ENC <arg0> 0 <arg2>`

③ **Argument quick info**

No.	Type	Value	Default	Description
arg0	int	0–2	–	Axis index
arg1	int	0–3	–	Encoder subfunction index
arg2	int	s32	–	Counter value to be set

④ **Description**

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

⑤ **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.

⑥ **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.

⑦ **Related commands**

ENC (Encoder functions)

ETR (Encoder trigger control)

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 argu-ments are additionally enclosed in square brackets.

No.	Criterion	Explanation
3	Argument quick info	Info table on command with the information (3A–3E).
	No.	Argument index
	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
	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.
	Default	Default value. This value will be set when using the SFD command.
	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 bits, 16 bits or 32 bits wide
s8, s16, s32	Signed integers that are 8 bits, 16 bits or 32 bits wide
float	Floating point numbers according to IEEE 754 (single precision)
s16[]	Array of signed 16-bit integers

2.2 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	33	Only needed if <arg0> = 1: desired detector level in % saturation level (0–100), 33 is recommended
arg1	int	Dollar protocol: 0–255	84	Dollar protocol: Only needed if <arg0> = 1: desired detector level in % of saturation level (0–255), 84 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.

Examples

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

Related commands

LAI (Lamp intensity)

2.3 ABE (Abbe number)

Command format

`ABE <arg0> ...`

Argument quick info

No.	Type	Value	Default	Description
arg0	float	0–999	0	Abbe number of layer 1
arg1	float	0–999	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)

2.4 ANAM (Analog output mode)

Scope

Only valid for OD7000-xxxxxxx1.

Command format

`ANAM <arg0>`

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0, 2	0	0: Voltage output (-10–10 V) 2: Current output (0–20 mA)

Description

The analog outputs support voltage or current output. 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)

Scope

Only valid for OD7000-xxxxxxx1.

Command format

`ANAX <arg0> to <arg7>`

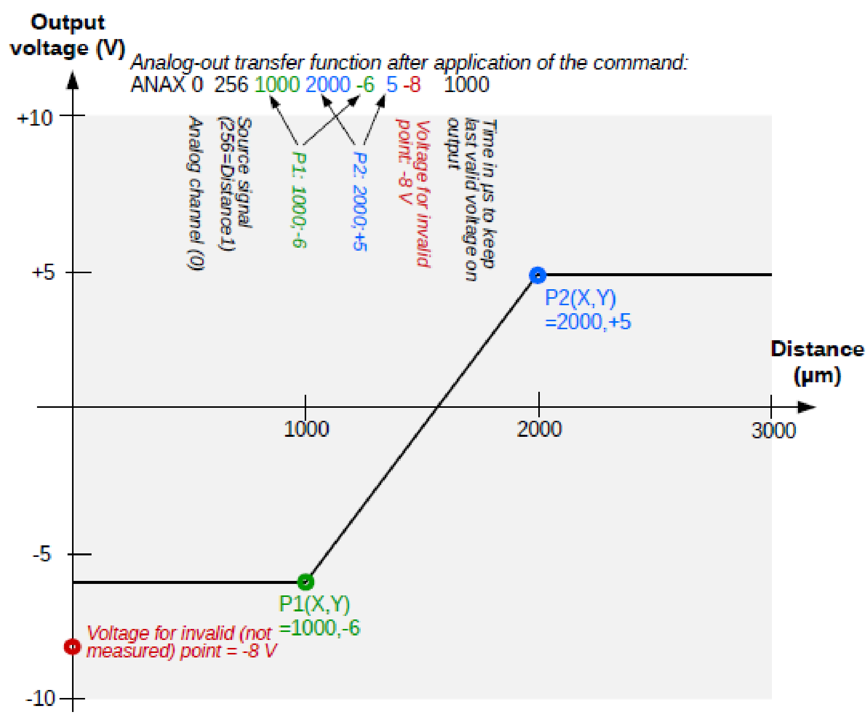
Argument quick info

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 / 0–20	0	Voltage/current attributed to value given by `<arg2>`

No.	Type	Value	Default	Description
arg5	float	-10–10 / 0–20	10	Voltage/current attributed to value given by `<arg3>`
arg6	float	-10–10 / 0–20	-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 µm ... 1000 µm are transmitted as 0 V 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 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	0–999	1	Data average depth

Description

This command relates to averaging of distance / intensity data. 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.

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

AVS (Spectra averaging)

AVM (Apply moving average)

2.8 AVM (Apply moving average)

Command format

`AVM <arg0>`

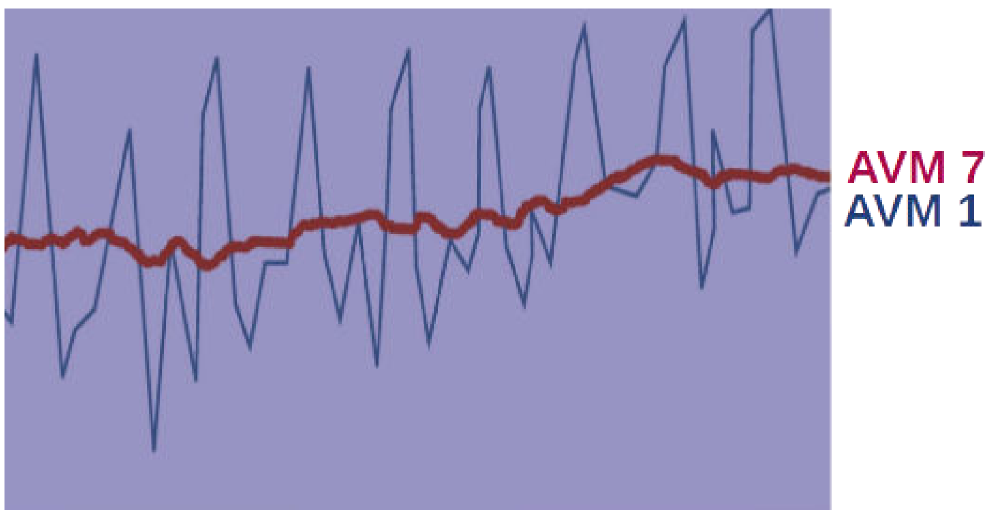
Argument quick info

No.	Type	Value	Default	Description
arg0	int	1–1000	1	Window size of the moving average filter

Description

This command provides a moving average of the distance and intensity data. It is used for smoothing data series. The degree of smoothing depends on the window size of the moving average filter.

The argument `<arg0>` specifies the window size. This number may range between 1 (no moving average applied) and 1000 (window size for the average operation is 1000 samples). If the window size is greater than 1, the `AVM` command will apply a simple moving average filter on the distance/intensity data sampled when all requested peaks have been detected.



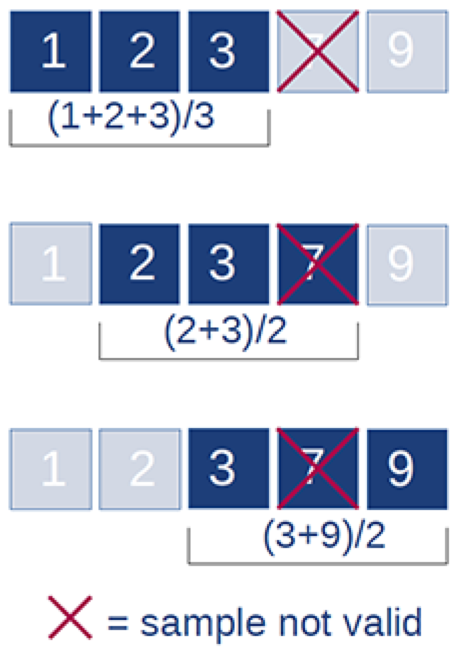
Invalid samples caused by low signal quality or intensity will be skipped and are not considered when computing the average. In this regard, the `AVM` command works similar to `AVD` (Data averaging) and `AVS` (Spectra averaging). In contrast to `AVD` and `AVS`, because of the moving average filter applied, the moving average operation does not reduce the sample rate.

In the illustration below, calculation of the moving average with a window size of three samples is shown:

(Top) The first three samples: Result is first moving average.

(Middle) Shift window one position to the right, one of the three samples is not valid: Next average has only two elements.

(Bottom) Shift window one position to the right, one of the three samples is not valid: Next average has only two elements.



Good to know

The command can be used in conjunction with `AVD` and `AVS`. The moving average option is applied after the average spectrum (`AVS`) and average data (`AVD`) operation.

Examples

Input	Comment
`AVM 50`	Window size for the average operation is 50 samples.
`AVM 1`	No moving average is active.
`AVM ?`	Returns the window size that is currently active.

Related commands

AVD (Data averaging)

AVS (Spectra averaging)

2.9 AVS (Spectra averaging)

Command format

```
AVS <arg0>
```

Argument quick info

No.	Type	Value	Default	Description
arg0	int	1–999	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.

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.

Examples

Input	Comment
AVS 5	Average over 5 spectra
AVS 1	No spectra averaging is active.
AVS ?	Returns the current value(s).

Related commands

AVD (Data averaging)

AVM (Apply moving average)

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.

Example: Spectrum download (DNLD)

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>≤ 8: interpreted as a baud rate index, otherwise as a free custom baud rate.

No.	Type	Value	Default	Description
arg1	int	0, 1	0	Hardware RTS/CTS handshake on/off (handshake is only available if RS-232 is selected, see `SRPT` command)

Response quick info

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

Description

The baud rate of the serial port can be adjusted to values between 9600 and 1843200 (115200 is the default and recommended value on the RS-232 port).

As described in the arguments, you can input a baud rate index or a free baud rate. 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.

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

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. Please note that hardware handshaking is only available when RS-232 mode is selected by the `SRPT` command.

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).

Related commands

SRPT (Configure serial port (RS422 RS-232))

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 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 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(s).

Related commands

DRK (Dark reference)

FDK (Fast dark reference)

2.15 CTN (Continue in free run mode)

Command format

`CTN`

Argument quick info

No arguments 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.

Related commands

TRE (Trigger each)

TRG (Trigger once)

TRW (Trigger window)

2.16 DNLD (Download spectrum)

Scope

Packet protocol only

Command format

```
DNLD <arg0>
```

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0, 1	-	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	-	Reserved (1)
arg3	int	s32	-	Reserved (1)
arg4	blob	s16[]	-	Spectrum data, one s16 per pixel

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)

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.

2.17 DRK (Dark reference)

Command format

```
DRK
```

Argument quick info

No arguments supported

Description

Takes a dark reference and stores the result in the non-volatile flash memory. The operation takes about 3 seconds. The sensor returns the (virtual) measuring rate in Hz at which the detector would 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).

Examples

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

Related commands

CRDK (Continuous refresh dark factor)

FDK (Fast dark reference)

2.18 DTF (Data format query)

Scope

Dollar protocol only

Command format

```
DTF
```

Response quick info

No.	Type	Value	Default	Description
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 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.

2.19 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.
- Don’t mix the use of `LLM/RLM` and `DWD`.

Examples

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

2.20 ENC (Encoder functions)

Scope

Only valid for OD7000-xxxxxxx1.

Command format

```
ENC <arg0> to <arg2>
```

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0–2	–	Axis index
arg1	int	0–3	–	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 sub-functions will follow on the next pages.

Examples

Input	Comment
`ENC n 0` ...	ENC 0 (Set encoder counter value)
`ENC n 1` ...	ENC 1 (Set encoder counter source)
`ENC n 2` ...	ENC 2 (Set encoder preload value)
`ENC n 3` ...	ENC 3 (Set encoder preload event)

Related commands

ETR (Encoder trigger control)

2.20.1 ENC 0 (Set encoder counter value)

Scope

Only valid for OD7000-xxxxxxx1.

Command format

`ENC <arg0> 0 <arg2>`

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0–2	–	Axis index
arg1	int	0–3	–	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)

2.20.2 ENC 1 (Set encoder counter source)

Scope

Only valid for OD7000-xxxxxxx1.

Command format

```
ENC <arg0> 1 <arg2>
```

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0–2	–	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
10–14	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 OD7000 user manual). In that case, the counter only counts up.

For further discussion and examples, see "Encoder Interface (OD7000-xxxxxxx1)", page 72.

Examples

Input	Comment
ENC 2 1 15	Connects counter 2 to quadrature encoder input A2/B2.

Related commands

ENC (Encoder functions)

ETR (Encoder trigger control)

2.20.3 ENC 2 (Set encoder preload value)

Scope

Only valid for OD7000-xxxxxxx1.

Command format

```
ENC <arg0> 2 <arg2>
```

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0–2	–	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 (Set encoder preload event). The third argument <arg2> gives the value that will be preloaded.

For further discussion and examples, see "Encoder Interface (OD7000-xxxxxxx1)", page 72.

Examples

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

Related commands

ENC (Encoder functions)

ETR (Encoder trigger control)

2.20.4 ENC 3 (Set encoder preload event)

Scope

Only valid for OD7000-xxxxxxx1.

Command format

```
ENC <arg0> 3 <arg2>
```

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0–2	–	Axis index
arg1	int	3	–	Encoder subfunction index

No.	Type	Value	Default	Description
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
10–14	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, see "Encoder Interface (OD7000-xxxxxxx1)", page 72.

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)

2.21 ETR (Encoder trigger control)

Scope

Only valid for OD7000-xxxxxxx1.

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, see "Triggered Measurements", page 75.

Good to know

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

Examples

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

Related commands

CTN (Continue in free run mode)

ENC (Encoder functions)

TRG (Trigger once)

TRE (Trigger each)

TRW (Trigger window)

2.21.1 ETR 0 (Encoder trigger start position)

Scope

Only relevant if roundtrip trigger is active (see ETR 7). Only valid for OD7000-xxxxxxx1.

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 reset 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)

TRG (Trigger once)

TRE (Trigger each)

TRW (Trigger window)

2.21.2 ETR 1 (Encoder trigger stop position)

Scope

Only relevant if roundtrip trigger is active (see `ETR 7`). Only valid for OD7000-xxxxxxx1.

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)

TRG (Trigger once)

TRE (Trigger each)

TRW (Trigger window)

2.21.3 ETR 2 (Encoder trigger interval)

Scope

Only valid for OD7000-xxxxxxx1.

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).

Examples

Input	Comment
`ETR 2 10.5`	Sets trigger interval to 10.5.

Related commands

CTN (Continue in free run mode)

ENC (Encoder functions)

TRG (Trigger once)

TRE (Trigger each)

TRW (Trigger window)

2.21.4 ETR 3 (Encoder trigger state control)

Scope

Only valid for OD7000-xxxxxxx1.

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)

TRG (Trigger once)

TRE (Trigger each)

TRW (Trigger window)

2.21.5 ETR 4 (Encoder trigger active during return)

Scope

Only relevant if roundtrip trigger is active (see `ETR 7`). Only valid for OD7000-xxxxxxx1.

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)

TRG (Trigger once)

TRE (Trigger each)

TRW (Trigger window)

2.21.6 ETR 5 (Encoder trigger source)

Scope

Only valid for OD7000-xxxxxxx1.

Command format

```
ETR 5 <arg1>
```

Argument quick info

No.	Type	Value	Default	Description
arg0	int	5	–	Encoder trigger subfunction index
arg1	int	0–2	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)

TRG (Trigger once)

TRE (Trigger each)

TRW (Trigger window)

2.21.7 ETR 7 (Encoder trigger roundtrip / endless mode)

Scope

Only valid for OD7000-xxxxxxx1.

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

Chooses an encoder counter as trigger source using its index.

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)

TRG (Trigger once)

TRE (Trigger each)

TRW (Trigger window)

2.22 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.

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 argument is the (virtual) measuring rate in Hz at which the detector would 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 or timing-related commands like `SHZ` or `LAI` are applied.
- It is possible to query the last `FDK` response using `FDK ?`. If a query is sent, only the virtual measuring rate will be replied and no dark reference is taken.

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)

2.23 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

Error code	Sub-error code	Explanation
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.

2.24 GLL (Get last logs)

Scope

Packet protocol only

Command format

```
GLL [<arg0>]
```

Argument quick info

No.	Type	Value	Default	Description
arg0	int	n	–	Boot count (0 means current boot, 1 means last boot, etc.)

Response quick info

No.	Type	Value	Default	Description
arg0	int	–	–	Boot count for current entry
arg1	int	–	–	Total operation time in seconds
arg2	int	–	–	Severity (higher number means higher severity)
arg3	int	–	–	Log type (0: info; 1: warning; 2: error)
arg4	str	–	–	Time of log after last start-up (milliseconds)
arg5	str	–	–	Log strin

Description

A series of `GLL` commands will be replied, each containing a log entry during the boot specified (counting back). When n=0, the current boot logs (lethal and normal) will be replied. When n>0, the non-volatile stored logs (only lethal) will be replied for the backward nth boot.

Examples

Input	Comment
`GLL 0`	Returns information on the current boot, see table above for details.

2.25 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	4	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 OD7000'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 4 255 255 255 0`	Turns DHCP client off, sets the IP address and subnet mask.
`IPCN 192 168 170 4`	Sets the IP address statically, implicitly turning off DHCP client.
`IPCN 1`	Turns DHCP client on.
`IPCN ?`	Returns the current value(s).

2.26 LAI (Lamp intensity)

Command format

`LAI <arg0>`

Argument quick info

No.	Type	Value	Default	Description
arg0	float	0–100	0	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 as light source, a `LAI` value of 0 switches off the LED/SLD.

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)

2.27 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(s).

Related commands

DWD (Detection windows definition)

2.28 LTC (Latency)

Scope

Packet protocol only

Command format

`LTC <arg0>`

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0–10000	1	Latency in milliseconds

Description

Experimental parameter. It is not saved to non-volatile memory. Determines the time after which a data packet is closed and sent.

Examples

Input	Comment
`LTC 10`	Sets the data packet latency to 10 milliseconds.
`LTC ?`	Returns the current value(s).

2.29 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.

Examples

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

2.30 OFN (Output function)

Command format

```
OFN <arg0> [<arg1>]
```

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0	-	Output function index
arg1	int	0-7	0	Output function value

Description

This command is used to control the two digital outputs (Digital Out 0/1) and the reserve signal of a OD7000-xxxxxxx1.

OFN 0: Turns off all digital outputs.

OFN 0 <0..7>: Sets the output signal states using the following bit assignment:

- Bit 0: Reserve signal out
- Bit 1: Digital Out 0
- Bit 2: Digital Out 1

By setting bit #16 of <arg 1>, one can switch the reserve signal into "input" mode.

Examples

Input	Comment
`OFN 0 0`	Sets digital outputs: output[2..0] = b000, i. e. Digital Out 1 = 0 Digital Out 0 = 0 Reserve signal out = 0
`OFN 0 2`	Sets digital outputs: output[2..0] = b010, i. e. Digital Out 1 = 0 Digital Out 0 = 1 Reserve signal out = 0
`OFN 0 6`	Sets digital outputs: output[2..0] = b110, i. e. Digital Out 1 = 1 Digital Out 0 = 1 Reserve signal out = 0
`OFN 0 ?`	Returns the output signal state.

2.31 OPD (Operation data)

Command format

```
OPD <arg0>
```

Argument quick info

No.	Type	Value	Default	Description
arg0	int	2, 3	–	Subcommand index

Description

This command queries operational data (e.g., total operation time in seconds).

Examples

Input	Comment
`OPD 2 ?`	OPD 2 (Total operation time)
`OPD 3 ?`	OPD 3 (Number of power ups)

2.31.1 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].

2.31.2 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.

2.32 PSM (Peak separation minimum)

Scope

Chromatic confocal mode only

Command format

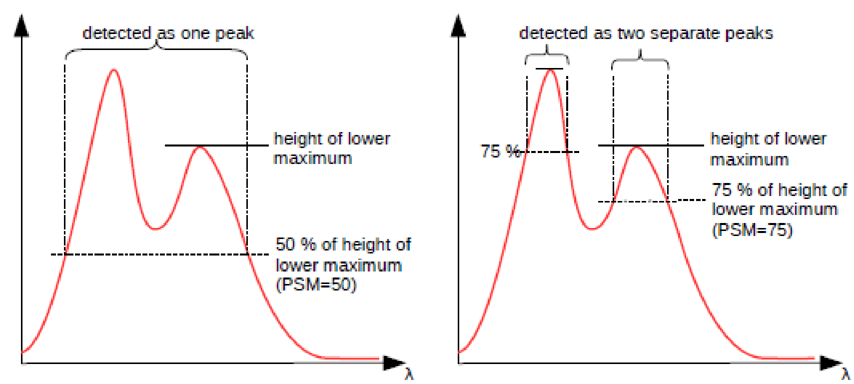
`PSM <arg0>`

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 neighboring 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.



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.

2.33 RST (Restart device)

Command format

`RST`

Argument quick info

No arguments supported

Description

Reboots the OD7000. 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 OD7000.

2.34 SCA (Scale)

Scope

The full scale is affected by the commands SRI, SRT, ABE. Thus, always query the full scale value after issuing these commands!

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.

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 OD7000 via the SRI command.

Examples

Input	Comment
SCA ?	Returns the scale in micrometers.

Related commands

ABE (Abbe number)

SRI (Set refractive indices)

SRT (Set refractive index table)

2.35 SEN (Select chromatic calibration)

Command format

```
SEN <arg0> [<arg1>]
```

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0–7	0	Index of currently used chromatic calibration table
arg1	int	0, 1	0	Extended measuring range enabled (1) or disabled (0)

Description

This command selects the chromatic calibration table by its index on the device. The second optional parameter permits to extend the nominal measuring range of each probe. However, the precision in the additional measuring range is reduced. When the second parameter is left out, the nominal range is selected.

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.
`SEN 0 0`	Selects calibration 0 with nominal measuring range.
`SEN 0 1`	Selects calibration 0 with extended measuring range.

Related commands

SENX (Extended chromatic calibration table query)

2.36 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

No.	Type	Value	Default	Description
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 Table index, followed by a ","
2 "`SNr: `", followed by the probe serial number
3 "`Range: `", followed by the measuring range in micrometers, followed by "`µm`"
4 Items 2 to 5 can repeat multiple times on multi-channel configurations that use multiple probes. In these cases, the probe descriptions are separated by a "`|`" character.

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)

2.37 SFD (Set factory defaults)

Command format

`SFD [<arg0>]`

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0, 1	–	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.

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)

2.38 SHZ (Set sample frequency in Hz)

Scope

Dollar protocol: if the value is not accepted, the device responds with the string “not valid”.

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.

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.

Examples

Input	Comment
`SHZ 40`	Sets the sample frequency of the device in Hz.
`SHZ ?`	Returns the current value.

2.39 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.

Indices and significations for the possible result words are:

Index	Mode 0 (distance)	Mode 1 (thickness)
0	Distance 1	Thickness
1	Not used	Distance 1 (bigger)
2	Not used	Distance 2 (bigger)
3	Intensity 1	Not used
4	Not used	Intensity 1
5	Not used	Intensity 2
6	Pixelpos. 1	
7	Extended precision bits Distance 1	Extended precision bits Distance 1 / 2
8	Flags: Bit2: IGNOREDTRIGGER In Trigger each mode: Trigger pulse was ignored because it arrived too soon after preceding trigger pulse and there was already a delayed trigger pending. Bit3: DELAYEDTRIGGER In Trigger each mode: Trigger of the exposure was delayed with respect to the trigger pulse because it arrived too soon after preceding trigger pulse. Bit4: CCD_SATURATED At least at one pixel the detector was saturated. The accuracy of the result may be impaired.	
9	Actual exposure time in units of 1/640000 sec	
10	Encoder 0 Position, most significant word	
11	Encoder 0 Position, least significant word	
12	Encoder 1 Position, most significant word	
13	Encoder 1 Position, least significant word	
14	Encoder 2 Position, most significant word	
15	Encoder 2 Position, least significant word	

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`	Includes the output word Distance 1 in the output telegram.
`SOD ?`	Returns the current values.

Related commands

SODX (Set output data extended)

2.40 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 16 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 section OD7000 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 section OD7000 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.

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)

2.41 SRI (Set refractive indices)

Command format

`SRI <arg0> <arg1>` ...

Argument quick info

No.	Type	Value	Default	Description
arg0	float	1–4	1	Refractive index of layer 1
arg1	float	1–4	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 OD7000 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

- 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).
- The reply of the query includes values of 3 layers even though not all are used according to the setting of `NOP`.

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)

NOP (Number of peaks)

SCA (Scale)

SRT (Set refractive index table)

2.42 SRPT (Configure serial port (RS422 RS-232))

Command format

```
SRPT <arg0>
```

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0, 1, ?	0 (RS-232)	Selects RS422 (1) or RS-232 (0) on serial port.

Description

The serial port can be configured as RS-232 or RS422 connection. The RS422 provides terminated differential signaling with a Rx and Tx signal. The receiving side is terminated inside the OD7000 by a 110-Ω resistor in order to avoid signal reflections. The transmitting signal must be terminated at the receive input of the host device. Due to the termination, the RS422 allows for higher baud rates (up to 10 MBaud) than the RS-232. There is no handshaking available for the RS422 link, so the host must be capable to receive the data produced by the OD7000 at any moment.

Good to know

The RS-232 is a nonterminated, single ended interface with Rx, Tx, RTS and CTS signals. The hardware handshake with the `RTS`, CTS signals is optional and can be activated with the `BDR` command. The RS-232 allows for baud rates up to 1843200 Baud.

Related commands

BDR (Baud rate and hardware handshaking)

2.43 SRT (Set refractive index table)

Command format

```
SRT <arg0> <arg1> to <arg14>
```

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0–14	0	Refractive index table of layer 1
arg1	int	0–14	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 v_d, the OD7000 offers up to 15 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 v_d.

If a selected table is not filled with valid data, the OD7000 will default to the n_d/ v_d dispersion model. The user has to interpret the response of the OD7000 to the `SRT` command in order to know if the selected setting was accepted and applied.

After the command the OD7000 responds with the active table index and its name.

When the thickness is output in the normalized integer format (16 bit or 32 bit), then the thickness output values are normalized to a fixed reference refractive index value (as with the n_d/v_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 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).
- The reply of the query includes values of three layers even though not all are used according to the setting of `NOP`.

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)

2.44 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 OD7000 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 a `SSU` command.

Examples

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

2.45 SSU (Save setup)

Command format

`SSU`

Argument quick info

No arguments supported

Description

Saves the current setting to non-volatile memory. The sensor will start on the next power-up with the saved configuration.

2.46 STA (Start data)

Command format

`STA`

Argument quick info

No arguments 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)

2.47 STO (Stop data)

Command format

```
STO
```

Argument quick info

No arguments 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)

2.48 STR (Software trigger)

Command format

```
STR [<arg0>]
```

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0, 1	0	0 = Sets the software trigger state to low. 1 = Sets the software trigger state to high. Without parameter = Generates trigger event.

Description

There are two forms of this command: without parameters or with one parameter.

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

If issued with a parameter, the command sets the software trigger state according to the parameter value. As the internal trigger state is an XOR combination of the software trigger state and the input state of the Sync-in input, the software trigger state can be used to choose if the rising or falling edge of the Sync-in signal generates a trigger event:

If the software trigger state is set to 1, the OD7000 will trigger on the falling edge of the Sync-in signal, whereas it triggers on the rising edge if `STR` is set to 0.

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

Related commands

TRG (Trigger once)

TRE (Trigger each)

TRW (Trigger window)

2.49 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

ID	Indices	Bytes	Query	Name
9	0	2048	Yes	Exposure dark correction
12	0	2064	Yes	White correction

Further details on selected tables can be found in Appendix, see "Specification of selected tables", page 83.

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 SICK for further information.

2.50 THR (Threshold, confocal mode)

Scope

Chromatic confocal mode only

Command format

`THR <arg0>`

Argument quick info

No.	Type	Value	Default	Description
arg0	int	0–1000	50	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.

Examples

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

2.51 TRE (Trigger each)

Command format

`TRE`

Argument quick info

No arguments 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 "Global signals", page 69).

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.

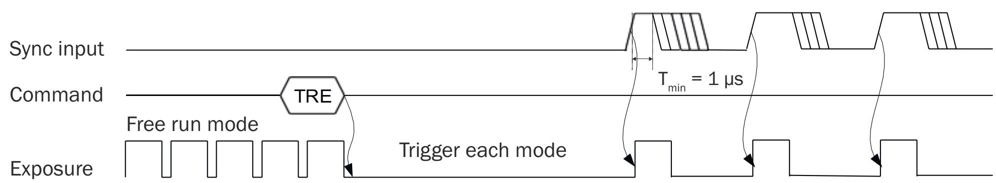


Figure 1: Trigger each without averaging

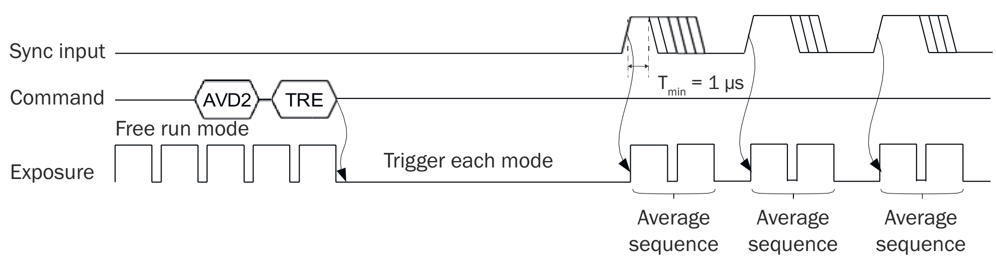


Figure 2: Trigger each with averaging

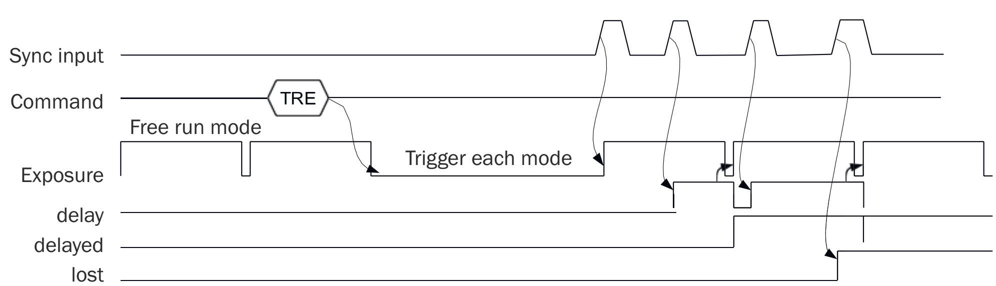


Figure 3: Trigger each showing trigger delay and loss

Good to know

The command `CTN` resumes normal operation (free run mode).

Related commands

AVS (Spectra averaging)

AVD (Data averaging)

CTN (Continue in free run mode)

TRG (Trigger once)

TRW (Trigger window)

2.52 TRG (Trigger once)

Command format

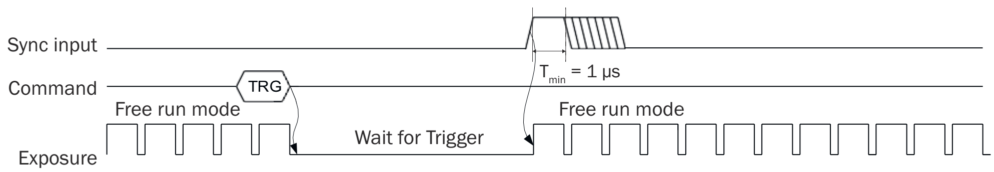
`TRG`

Argument quick info

No arguments 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).

Related commands

CTN (Continue in free run mode)

TRE (Trigger each)

TRW (Trigger window)

2.53 TRW (Trigger window)

Command format

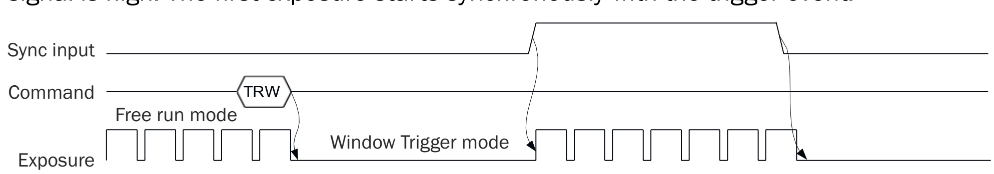
`TRW`

Argument quick info

No arguments supported

Description

Switches to Window Trigger mode: The device is in free run mode as long as the trigger signal is high. The first exposure starts synchronously with the trigger event.



Good to know

The command `CTN` resumes normal operation (free run mode).

The Window Trigger mode can be combined with the window average 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)

TRE (Trigger each)

TRG (Trigger once)

2.54 VER (Version Information)

Command format

`VER`

Argument quick info

No arguments supported

Description

This command queries identification key-value pairs. It returns value pairs of the form: <key1>=<value1><crlf><key2>=<value2><crlf>. . .

Examples

Input	Comment
VER	Returns for example the following string: OD7000 24174ae7042464d1 00f140803871fd1f Board Revision = 4 device_serial_number = 4000537 firmware_version = R1.6.1 build = 2023-12-15 9bf3a684a0 Extension Box FW = UD1.6.1 2023-08-29 25e992c819

3 OD7000 Signal IDs

3.1 Introduction

Overview

The following section 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.

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 surfaces) in one sample, whereas global signals represent information that is common for all signals.

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	0	0	0	0	Global signal ID (see separate table)	
Peak signal, single channel devices	Value encoding	0	0	Mode 0: distance 1: thickness	1	Peak index	Peak signal type

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

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

Bit number 2	1	0	Signal Name	Description
0	0	0	PeakValue	Distance / Thickness
0	0	1	PeakIntensity	Confocal mode: Intensity of peak
0	1	0	CCDSaturation	Confocal mode: Peak height on detector in percent of full well
0	1	1	PeakPos	Confocal: detector pixel position
1	0	0	PeakValue Median	Median of PeakValue (median depth defined with MEDX/MED command)
1	0	1	PeakWidth	Confocal mode only: peak width, given in pixels

Bit number 15	14	In case of geometrical quantities (e.g. distance or thickness)	In case of non-geometrical quantities: Native number values (e.g. peak intensity)	In case of non-geometrical quantities: 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) 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) 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

Bit number	Binary value						Description
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 = Dis-tance 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 of 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
75	ExposureCount	u16	Exposure number of the first exposure in of 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
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
88	Health_UPPLostCount	u32	Error counter: spectrum lines lost during transmission
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; measurement location is device-specific.

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).

Bit number	Description
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	Reserved
9	Reserved
10	State of the Sync-in input (if usable for trigger)
-	-

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 (32 bit) which has to be added to the data packet time stamp (64 bit) 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. 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 OD7000 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)		

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)
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 (Time counter, 80 MHz most significant word)		
34	97 (Time counter, 80 MHz least significant word)		

4 Encoder Interface (OD7000-xxxxxxx1)

4.1 Encoder interface

Overview

The OD7000-xxxxxxx1 supports interfacing incremental encoders in order to relate real world positions to the measurements in real time. 3 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 "Encoder trigger control", page 77.

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.

The encoder command has the following format: `ENC <axis> <function> <arg>` where axis is the `axis` index 0...2, 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:

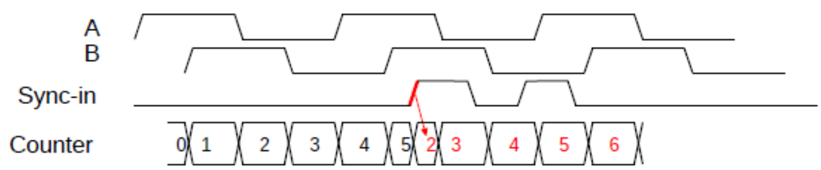
0:	A0
1:	B0
2:	A1
3:	B1
. . .	. . .
. . .	. . .
. . .	. . .
9:	B4
10:	Sync-in

- **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:

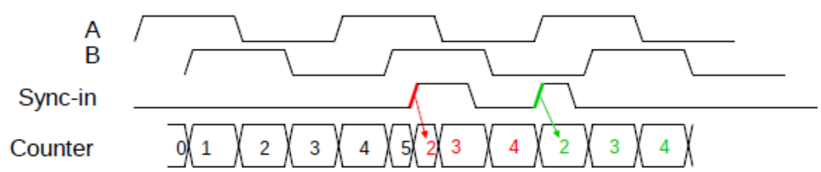
0:	A0

1:	B0
2:	A1
3:	B1
. . .	. . .
. . .	. . .
. . .	. . .
9:	B4
10:	Sync-in
11–14:	Reserved
15:	Immediate preload

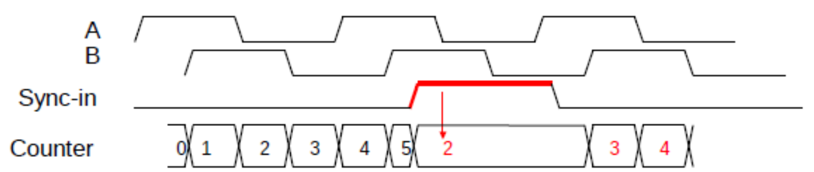
The following waveforms illustrate different preload scenarios. In all examples, the preload value register has been preloaded with the value of 2. The preload event is derived from the Sync-in input.



Preload event behaviour after Command ENC0 3 138 (active/edge/rising/single/Sync-in)



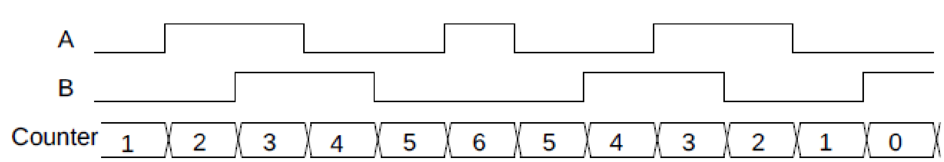
Preload event behaviour after Command ENC0 3 154 (active/edge/rising/multiple/Sync-in)



Preload event behaviour after Command ENC0 3 202 (active/state/high/single/Sync-in)

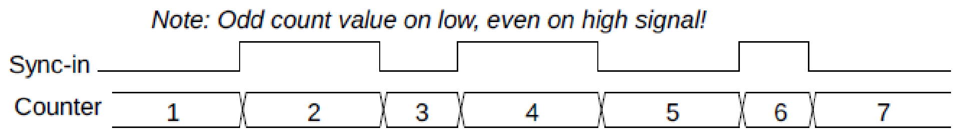
Quadrature count mode

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 specialty 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`):



Examples

`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.
`ENC 0 2 1234`	Sets the preload value of counter 0 to 1234.
`ENC 0 3 8`	(178 = 128 + 0 + 32 + 16 + 2) Configures the encoder counter to load the preload value 1234 into the counter 0 whenever a rising edge occurs on A1, i.e. the A input of encoder channel 1. Can be used e.g. for axis referencing.

5 Triggered Measurements

5.1 Triggered measurements

Overview

The OD7000 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 three different trigger modes: `Trigger once` (TRG), `Trigger each` (TRE) and `Trigger window` (TRW).

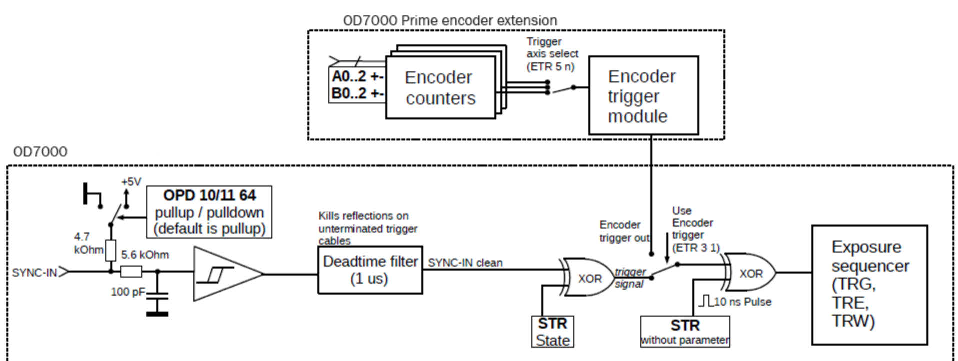
Trigger event

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 "Encoder trigger control", page 77.

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 (Software trigger)", page 60. 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 in an example:



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.

Command	Description
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

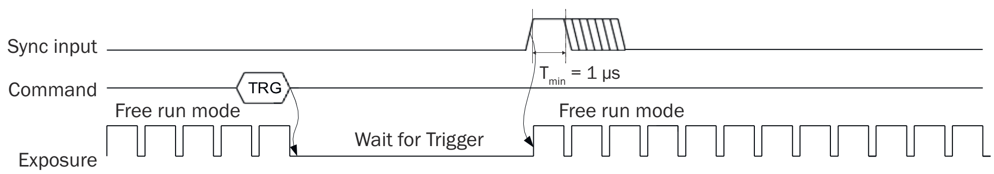
In Trigger each mode, one trigger event will start a sequence of AVD*AVS exposures. The AVD*AVS exposures are executed with the current sample rate (SHZ). One averaged result will be output after the exposure sequence. There will be only one trigger event on the Sync-out signal per n samples to be averaged. 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:

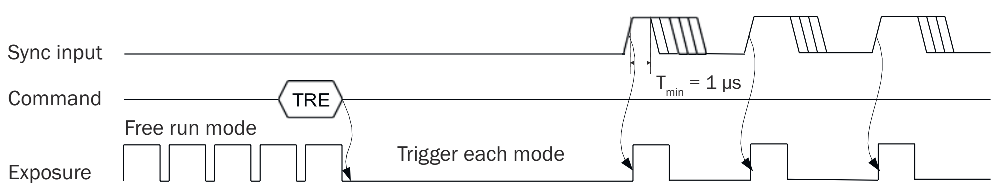


Figure 4: Trigger each without averaging

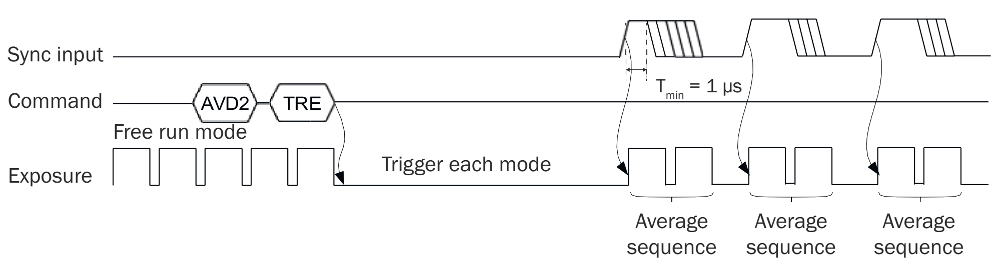


Figure 5: Trigger each with averaging

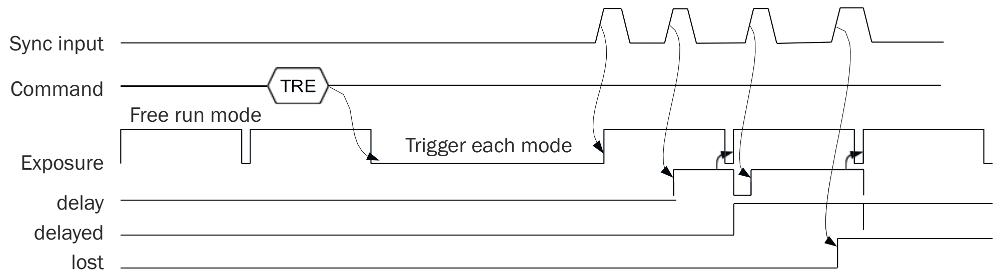
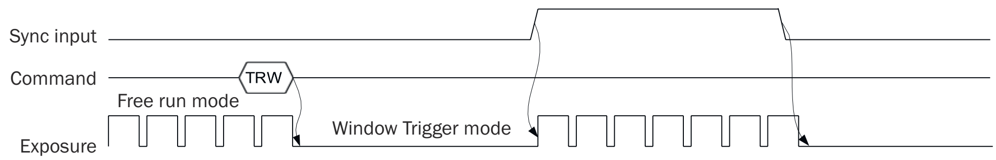


Figure 6: Trigger each showing trigger delay and loss

Trigger window

The figure below illustrates Trigger window mode:



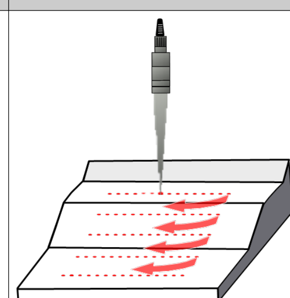
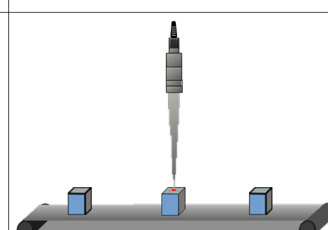
5.2 Encoder trigger control

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	

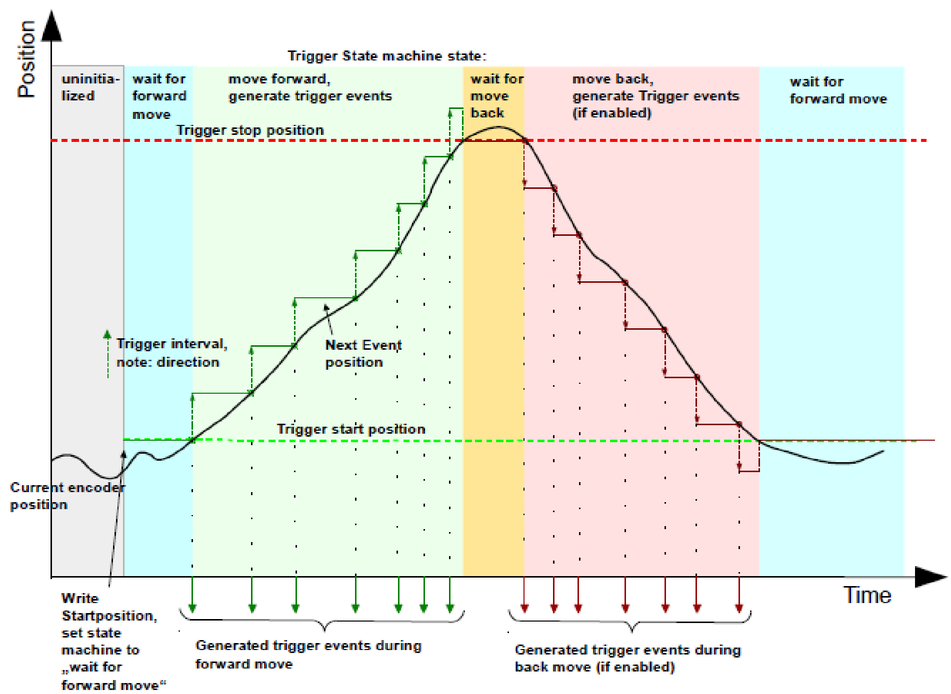
5.2.1 Encoder roundtrip trigger

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 OD7000 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:



- 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.

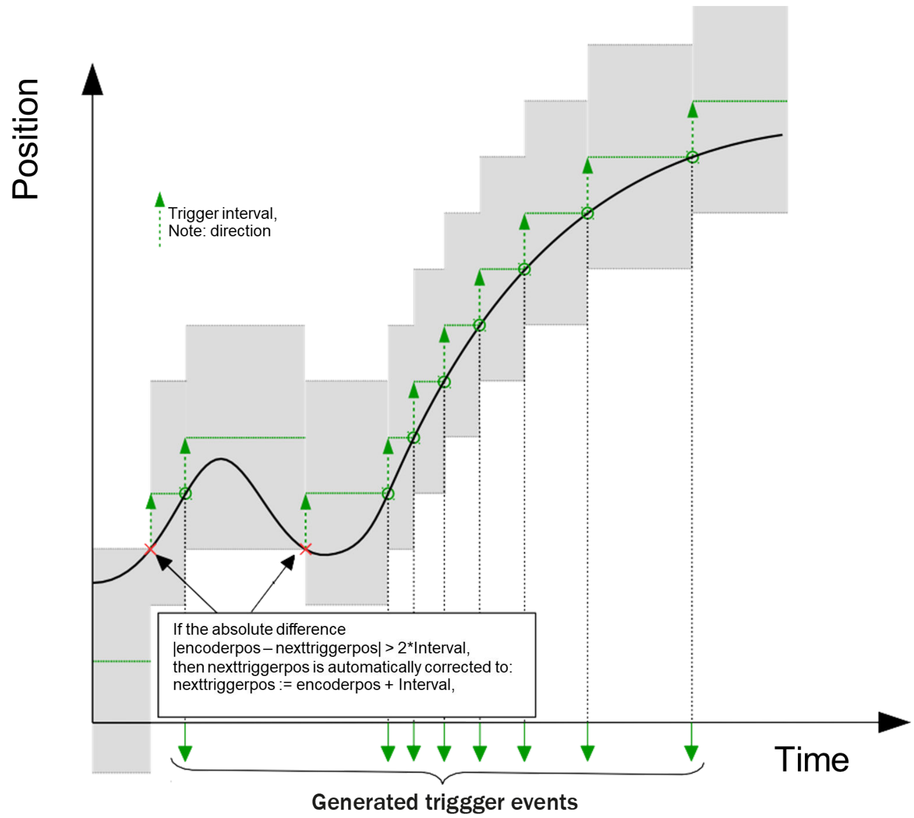
5.2.2 Encoder endless trigger

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:



5.2.3 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.

NOTE
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 OD7000 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 OD7000 to trigger each mode.

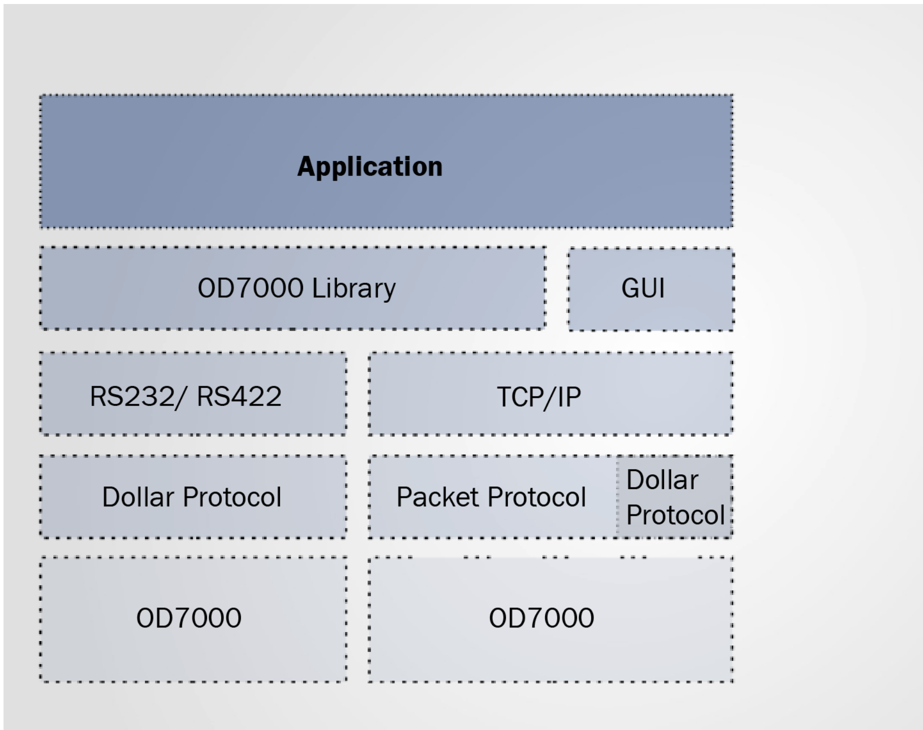
In example 2, the start position is greater than the stop position. Therefore, a negative value must be given to the interval value. In this setting, ten samples will be generated.

6 OD7000 Library

6.1 Using the OD7000 Library (OD7000Lib)

Overview

The following figure gives a general overview on integration of the OD7000 device into the application running on measuring systems. As illustrated, communication with the OD7000 device can be carried out on the hardware side (using packet protocol or dollar protocol) or on the software side (using OD7000 Library or OD7000):



OD7000 Library

The library provides a universal interface for integrating OD7000 devices. It encompasses basic functions for communicating with devices to acquire data and to set up the OD7000. This is an enormous advantage, as the different communication protocols mentioned in the previous sections (dollar protocol, packet protocol) are device-specific and support a different set of commands.

Further features

Main features of the OD7000 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 platform?

The OD7000 Library is available as:

- Dynamic Link Library (DLL) for Windows, 7, 8, 10
- Shared object library for the Linux platform Please contact our Support team for details.

Further details

For a more detailed description of the OD7000 Library features and functions, refer to the document “OD7000LibAPI.pdf”.

7 Further Information

7.1 Specification of selected tables

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.

Refractive indices

A user-defined table provides the wavelength-dependent index of refraction for a given material. 32 n(λ) 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 charac-ters. 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;
```

8 Topic-Related Command Lists

8.1 Timing related commands

"AAL (Auto adapt light source)", page 19
"LAI (Lamp intensity)", page 46
"SHZ (Set sample frequency in Hz)", page 54

8.2 Dark/white reference related commands

"CRDK (Continuous refresh dark factor)", page 28
"DRK (Dark reference)", page 30
"FDK (Fast dark reference)", page 42

8.3 Peak detection related commands

"DWD (Detection windows definition)", page 31
"LMA (Detection limits active)", page 46
"NOP (Number of peaks)", page 47
"PSM (Peak separation minimum)", page 50
"THR (Threshold, confocal mode)", page 62

8.4 Trigger related commands

"CTN (Continue in free run mode)", page 29
"ENC (Encoder functions)", page 32
"ETR (Encoder trigger control)", page 37
"STR (Software trigger)", page 60
"TRE (Trigger each)", page 62
"TRG (Trigger once)", page 64
"TRW (Trigger window)", page 64

8.5 Dispersion related commands

"ABE (Abbe number)", page 20
"SRI (Set refractive indices)", page 56
"SRT (Set refractive index table)", page 58

8.6 Communication settings related commands

"ASC (ASCII mode)", page 23
"BCAF (Binary command argument format)", page 26
"BDR (Baud rate and hardware handshaking)", page 26
"BIN (Binary mode)", page 28
"DTF (Data format query)", page 31
"IPCN (IP configuration)", page 45
"SRPT (Configure serial port (RS422 RS-232))", page 57

"SSQ (Synchronization sequence)", page 59

"STA (Start data)", page 59

"STO (Stop data)", page 60

8027510/ 1MIY/2024-02-14/en

Australia
Phone +61 (3) 9457 0600
1800 33 48 02 – tollfree
E-Mail sales@sick.com.au

Austria
Phone +43 (0) 2236 62288-0
E-Mail office@sick.at

Belgium/Luxembourg
Phone +32 (0) 2 466 55 66
E-Mail info@sick.be

Brazil
Phone +55 11 3215-4900
E-Mail comercial@sick.com.br

Canada
Phone +1 905.771.1444
E-Mail cs.canada@sick.com

Czech Republic
Phone +420 234 719 500
E-Mail sick@sick.cz

Chile
Phone +56 (2) 2274 7430
E-Mail chile@sick.com

China
Phone +86 20 2882 3600
E-Mail info.china@sick.net.cn

Denmark
Phone +45 45 82 64 00
E-Mail sick@sick.dk

Finland
Phone +358-9-25 15 800
E-Mail sick@sick.fi

France
Phone +33 1 64 62 35 00
E-Mail info@sick.fr

Germany
Phone +49 (0) 2 11 53 010
E-Mail info@sick.de

Greece
Phone +30 210 6825100
E-Mail office@sick.com.gr

Hong Kong
Phone +852 2153 6300
E-Mail ghk@sick.com.hk

Hungary
Phone +36 1 371 2680
E-Mail ertekesites@sick.hu

India
Phone +91-22-6119 8900
E-Mail info@sick-india.com

Israel
Phone +972 97110 11
E-Mail info@sick-sensors.com

Italy
Phone +39 02 27 43 41
E-Mail info@sick.it

Japan
Phone +81 3 5309 2112
E-Mail support@sick.jp

Malaysia
Phone +603-8080 7425
E-Mail enquiry.my@sick.com

Mexico
Phone +52 (472) 748 9451
E-Mail mexico@sick.com

Netherlands
Phone +31 (0) 30 229 25 44
E-Mail info@sick.nl

New Zealand
Phone +64 9 415 0459
0800 222 278 – tollfree
E-Mail sales@sick.co.nz

Norway
Phone +47 67 81 50 00
E-Mail sick@sick.no

Poland
Phone +48 22 539 41 00
E-Mail info@sick.pl

Romania
Phone +40 356-17 11 20
E-Mail office@sick.ro

Russia
Phone +7 495 283 09 90
E-Mail info@sick.ru

Singapore
Phone +65 6744 3732
E-Mail sales.gsg@sick.com

Slovakia
Phone +421 482 901 201
E-Mail mail@sick-sk.sk

Slovenia
Phone +386 591 78849
E-Mail office@sick.si

South Africa
Phone +27 10 060 0550
E-Mail info@sickautomation.co.za

South Korea
Phone +82 2 786 6321/4
E-Mail infokorea@sick.com

Spain
Phone +34 93 480 31 00
E-Mail info@sick.es

Sweden
Phone +46 10 110 10 00
E-Mail info@sick.se

Switzerland
Phone +41 41 619 29 39
E-Mail contact@sick.ch

Taiwan
Phone +886-2-2375-6288
E-Mail sales@sick.com.tw

Thailand
Phone +66 2 645 0009
E-Mail marcom.th@sick.com

Turkey
Phone +90 (216) 528 50 00
E-Mail info@sick.com.tr

United Arab Emirates
Phone +971 (0) 4 88 65 878
E-Mail contact@sick.ae

United Kingdom
Phone +44 (0)17278 31121
E-Mail info@sick.co.uk

USA
Phone +1 800.325.7425
E-Mail info@sick.com

Vietnam
Phone +65 6744 3732
E-Mail sales.gsg@sick.com

Detailed addresses and further locations at **www.sick.com**

SICK AG | Waldkirch | Germany | www.sick.com

SICK
Sensor Intelligence.