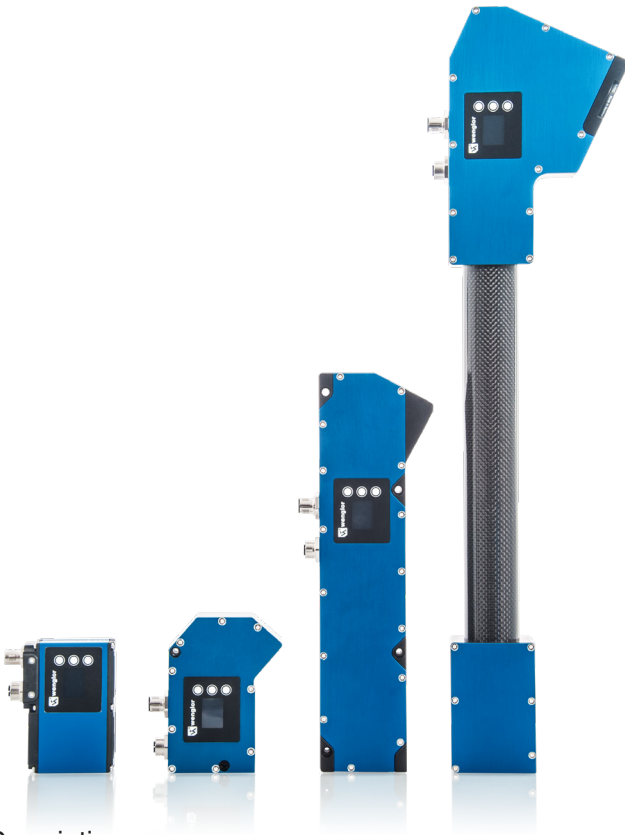


weCat3D MLSL/MLWL

2D/3D Profile Sensors



DLL Interface Description

Table of Contents

- 1. **Change Index** 6
- 2. **Document Information** 7
 - 2.1 References 7
- 3. **Introduction** 7
 - 3.1 System Requirements 7
- 4. **Application Example** 7
- 5. **SDK Functions** 10
 - 5.1 Connecting weCat3D Profile Sensor 11
 - 5.2 Closing Connection 11
 - 5.3 Check Connection 11
 - 5.4 Get General Sensor Information (deprecated) 12
 - 5.5 Get Scanned Profile 12
 - 5.6 Check DLL FiFo State 14
 - 5.7 Reset DLL FiFo 14
 - 5.8 Setup Profile Sensor 14
 - 5.9 Read DLL Version 15
 - 5.10 Read Property Value 15
- 6. **Setup Profile Sensor** 17
 - 6.1 Initiate Reboot 17
 - 6.2 Exposure Time 17
 - 6.2.1 Fixed Exposure Time 17
 - 6.2.2 Auto Exposure Time 17
 - 6.2.2.1 Set the Minimum of Auto Exposure Time 17
 - 6.2.2.2 Set the Maximum of Auto Exposure Time 17
 - 6.2.2.3 Set the Minimum of Intensity Range 17
 - 6.2.2.4 Set the Maximum of Intensity Range 18
 - 6.2.2.5 Set Minimum of Range X 18
 - 6.2.2.6 Set Maximum of Range X 18
 - 6.3 Setup Acquisition Line Time 18
 - 6.4 HDR Mode 19
 - 6.4.1 Set HDR 19
 - 6.4.2 Setup Exposure Time 2 19

6.5	Deactivate Laser	19
6.6	Set User LED	20
6.7	Enable Signal (Z)	20
6.8	Enable Signal (Strength)	20
6.9	Enable Signal (Width)	20
6.10	Enable Signal End Position Command	21
6.11	Setup Socket Connection Timeout	21
6.12	Setup The Heartbeat Signal	21
6.13	Start Acquisition	21
6.14	Stop Acquisition	21
6.15	Reset Settings	22
6.16	Reset Encoder	22
6.17	Reset Picture Counter	22
6.18	Save Settings	22
6.19	Load Settings	23
6.20	Setup Trigger Source	23
6.21	Setup Trigger Divider	24
6.22	Setup Trigger Delay	24
6.23	Software Trigger Command	24
6.24	Setup Encoder Trigger Function	24
6.25	Enable Fixed Frame mode	25
6.26	Setup Number of Profiles in Fixed Frame Mode	25
6.27	Setup Sync Out	25
6.28	Setup Delay of Sync Out	26
6.29	Enable Signal	26
6.30	Setup Signal Minimum Width	26
6.31	Setup Signal Maximum Width	26
6.32	Setup Signal Selection	26
6.33	Internal Profile Calculation	27
6.34	Setup Encoder Count Direction	27
6.35	Region of interest (ROI)	27
6.35.1	Setup ROI Width in X	27
6.35.2	Setup ROI Offset in X	28
6.35.3	Setup ROI Step X	28
6.35.4	Setup ROI Height in Z	28
6.35.5	Setup ROI Offset in Z	28
6.35.6	Setup ROI Step Z	28

6.36	E/A Functions	29
6.36.1	Setup E/A Functions	29
6.36.2	Setup E/A Function Laser Off	29
6.36.3	Setup E/A Function Profile Enable	29
6.36.4	Setup E/A Function Reset Enable	30
6.36.5	Setup E/A Repeat Reset Behaviour	30
6.36.6	Setup E/A Reset Signal Edge	30
6.36.7	Setup E/A Reset Base Time Counter	30
6.36.8	Setup E/A Reset Picture Counter	31
6.36.9	Setup E/A Reset Encoder HTL	31
6.36.10	Setup E/A Reset Encoder TTL	31
6.36.11	Setup E/A 1 Input Function	31
6.36.12	Setup E/A 1 Input Load	32
6.36.13	Setup E/A 1 Output	32
6.36.14	Setup E/A 1 Output Function	32
6.36.15	Setup E/A1 Input Counter	32
6.37	Setup User Data	33
6.38	Setup the Shared Library Internal FiFo Size	33
6.39	Setup the Shared Library Internal FiFo Mode	33
7.	Read Properties of weCat3D Profile Sensor	34
8.	Data Structure	37
8.1	General	37
8.1.1	Buffer Structure (one selected signal)	37
8.1.2	Buffer Structure (two selected signals)	38
9.	TCP/IP Socket Interface	38
9.1	Introduction	38
9.2	Setup the TCP/IP Socket Communication	38
9.3	Data Format Definition	39
9.3.1	Basic Data Formats	39
9.3.2	Complex Data Formats	39
9.4	General Structure	40
9.5	Structure of a Tag	40
9.6	Description of Tag	41
9.6.1	Container Tag	41
9.6.2	General Tag	41
9.6.3	Statistic Tag	42
9.6.4	Description Tag	43
9.6.5	ROI-X Tag	43
9.6.6	ROI-Z Tag	43

9.6.7	RegisterCameraMLSL	44
9.6.8	RegisterCameraMLWL	44
9.6.9	Register FPGAMLSL	44
9.6.10	Register FPGAMLWL	44
9.6.11	Linearization Table	45
9.6.12	ScanNonLinear	45
9.6.13	ScanLinear	45
9.6.14	SubID-ScanLinearHeader	46
9.6.15	SubID-ScanLinearData	48
9.6.16	ScaleParam	48
9.6.17	CRC	48
9.7	Typical Data Sets	49
9.7.1	Overview Typical Data Stream MSL	49
9.7.2	Overview Typical Data Stream MLWL	49
9.7.3	Example First Data After Connection	50
9.7.4	Example MSL Container	50
9.7.5	Example MLWL Container	53
9.8	Implementation Recommendation	56
9.9	CRC Calculation	56
10.	Appendix	58
10.1	GetInfo (XML mode)	58
10.2	GetInfo (Text mode)	58

1. Change Index

Version	Release Date	Description	DLL version
1.0.0	23.08.2016	Initial document	1.6.x
1.1.0	17.05.2017	<ul style="list-style-type: none"> • New document structure • Additional ASCII commands • New experimental functions • Changes for ASCII commands for DLL version 1.7.0 or later 	1.7.0
1.2.0	10.12.2018	<ul style="list-style-type: none"> • Fixing some typos and errors in the documentation • Updating the SDK Graphical User Interface • Extending the ASCII commands for the function EthernetScanner_WriteData • Extending the ASCII commands for the function EthernetScanner_ReadData • All EthernetScanner functions are now thread safe • Bug fixes in Linux version • DLL automatically initialized • ASCII commands with carriage return 	1.9.0
1.3.0	01.02.2019	<ul style="list-style-type: none"> • Fixing some typos • New commands: SetHDR and SetExposureTime2 (available from FW version 1.1.3) • New command SetTriggerDelay 	1.9.1
1.4.0	21.01.2020	<ul style="list-style-type: none"> • New commands for Auto Exposure Time, compatible with FW 1.2.0 or higher • New command SetLinearizationMode, compatible with FW 1.2.0 or higher • New command SetStatisticDataUserData available in FW 1.2.0 or higher • New command for activating the E/A input counter in FW 1.2.0 or higher • New command for reading the E/A counter value • New command for reading the user settings • Adaption of some values and command descriptions • New TCP/IP socket interface section available in FW 1.2.0 or higher 	1.10.0
1.5.0	27.07.2021	<ul style="list-style-type: none"> • Added ASCII command "GetCheckLinearization-Mode" for EthernetScannerReadData function (see section 7) • Fix bug related to hardware linearization mode • Correct typos in the document • Fix bug EthernetScanner_GetXZIEExtended returns always -1 after calling EthernetScanner_ResetDlIFiFo • Fix bug EthernetScanner_ReadData in xml mode • Fix bug memory leak in creating a new connection to the sensor • Reset FiFo buffer after stop --> start operation • Correct Trigger Divider • Added ASCII command "GetEncoder" (see section 7) • Internal code optimisation 	1.12.0

2. Document Information

2.1 References

Document	Version
Operating_Instructions_MLSL-MLWL.pdf	1.3.1

3. Introduction

This document describes the functions and the commands for using the DLL to realize custom application development for the weCat3D product series. The DLL is for users who want to create their own 2D/3D applications using the weCat3D Sensor series.

3.1 System Requirements

Applications development with the DLL/shared library requires a Microsoft operating system (WIN7 x64, WIN10 x64)/ Linux (Ubuntu x64, 14.04 or higher).

The weCat3D product series requires a 1 Gigabit network interface card.

The SDKs are available for download at www.wenglor.com in the product's separate download area.

The SDK is distributed through different packages, each package provides an example project on how to use the SDK functions. Each project is written under different IDEs using different programming languages.

4. Application Example

Each SDK has a demo project with the source code. The demo application is given as a mean to demonstrate the data transmission from the profile sensor to the application using the SDK functions.

Below is a screenshot from the SDK_Windows_QT_C++.



NOTE!

Please check functionality with the demo client before start programming.

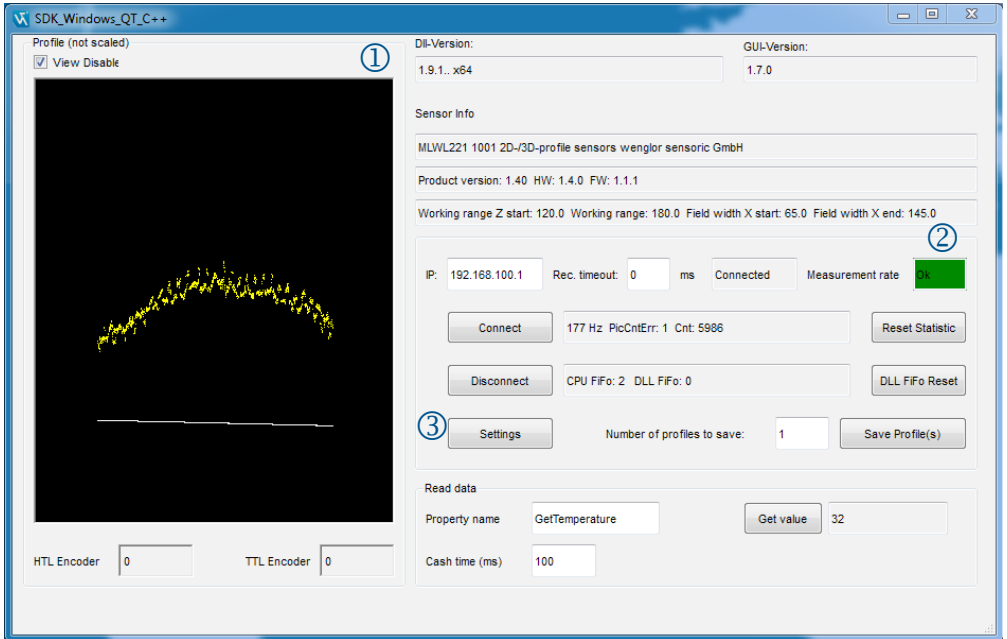


Fig. 1: Main window of the demo application delivered with the QT SDK for Windows (may differ in other demo projects)

- ① The demo project builds a connection to the profile sensor and shows a 2D representation of the scanned profile. The white points in the representation show the scanned profile, while the yellow dots display the intensity (signal strength) of each point.
- ② The main window in the demo project shows also the state of the measurement rate. If the measurement rate is within the allowed limits, the display field will show "Ok" (green background). If the measurement rate is too fast, the display field will show "too fast" (red background).
- ③ Click on the button "Settings" in order to check the ROI settings and the corresponding max scan request value. If the demo project fails to build a connection to the profile sensor, it will display the error message "Ethernet-Scanner_Connect: Error in connection":



NOTE!

Please check the IP address of your profile sensor and your network settings (check the reachability of the profile sensor in the network by pinging the profile sensor's IP address using the ping command in the operating system console, e.g. "ping 192.168.100.1").



NOTE!

You can check the connection state of the profile sensor through the web interface. Just type the IP address of the profile sensor in a web browser and look at the "connected to" field on the right side of the web interface.

The advance settings window (opens only if the connection to the profile sensor is established) allows to set-up the profile sensor and read the values of basic properties. It allows sending raw ASCII commands as well.

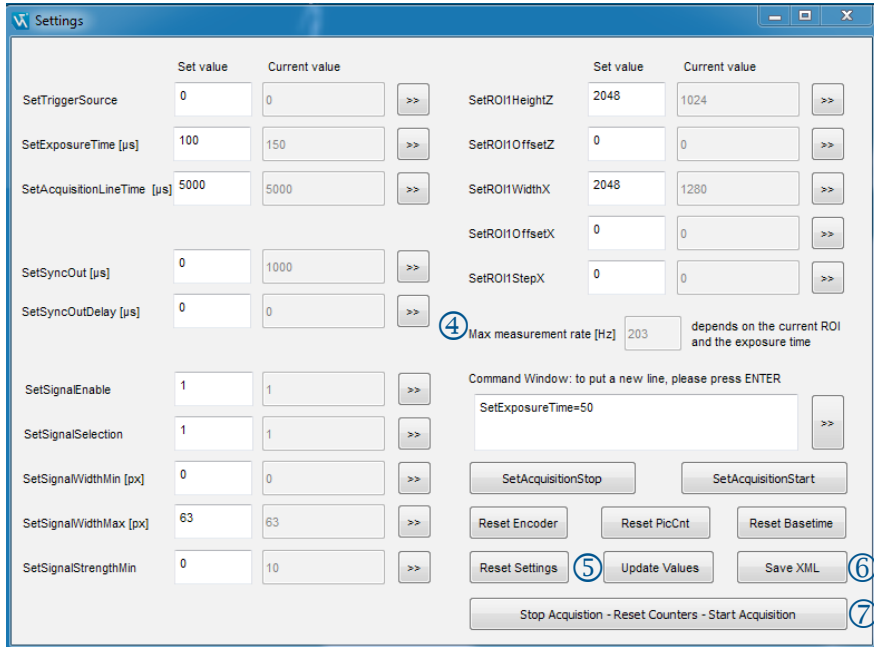


Fig. 2: The settings window of the demo project delivered with the QT SDK for windows (may differ in other demo projects)

④ The max measurement rate field computes the maximum measurement rate for triggering the profile sensor from the current ROI settings. The equation for computing the maximum value is available in the source code.



NOTE!

The computed max. measurement rate value is only an approximate value.

⑤ “Update values” button updates the values of some basic properties by calling and parsing the XML data description from the profile sensor.

⑥ “Save XML” saves the XML descriptor as XML file.

⑦ “Stop Acquisition - Reset Counters - Start Acquisition” button shows an example of the best behaviour to reset the profile sensor counters (like picture counter and system time counter) after stopping the acquisition.

In order to get scanned profiles from the profile sensor in a reliable way, the host application should send the following commands in the given sequence to the profile sensor to build a connection:

1. Build a connection to the profile sensor (EthernetScanner_Connect).
2. Check the connection status (EthernetScanner_GetConnectStatus).
3. (Optional) Set up the profile sensor according to application needs through ASCII commands (EthernetScanner_WriteData).
4. (Optional) Read the property values from the profile sensor (EthernetScanner_ReadData).
5. Read the scanned profiles from the profile sensor (EthernetScanner_GetXZI) and process the data accordingly.
6. Disconnect from sensor before ending the application (EthernetScanner_Disconnect).

NOTE!



In DLL version 1.7.0 or higher, there is no need to send the ASCII command “SetInitializeAcquisition” to the profile sensor after each connection. The DLL sends this command internally. If your program sends this command, the DLL (1.7.0 or higher) will ignore it. Sending the command “SetInitializeAcquisition” from the DLL has brought a lot of performance improvements to the DLL.

NOTE!



In DLL version 1.9.0 or higher there is no need to make sure that the DLL is initialized through calling the function EthernetScanner_GetInfo (see section 5.3 and 5.4 as well as the example code in the SDKs). The function EthernetScanner_GetConnectStatus (step 2) will return ETHERNETSCANNER_TCPSCANNERCONNECTED after building a valid connection to the profile sensor AND initializing the DLL.

5. SDK Functions

All the SDK functions are based on C function standard calls (_stdcall) and are compatible with all compilers that support C programming language. In fact, since the functions are based on C standard call, they can be deployed in a wide range of IDEs (QT, Visual studio C++, Visual Basic, C#, Delphi, Matlab, Labview, Embarcadero, etc.)



NOTE!

In DLL version 1.9.0 or higher all the SDK functions are thread safe.



NOTE!

All header definitions mentioned below are available in the header file "EthernetScannerSDKDefine.h" provided with the SDK.

5.1 Connecting weCat3D Profile Sensor

Command	<code>void* EthernetScanner_Connect(char *chIP, char *chPort, int iTimeOut)</code>
Parameter 1	char *chIP: the IP address of the profile sensor: „192.168.100.1“ with \0 at the end
Parameter 2	char *chPort: the port number of the profile sensor: „32001“ with \0 at the end
Parameter 3	int iTimeOut: Timeout in [ms] for the receive-function to close the connection, if no data is received. It is recommended to keep the timeout 0.
Response	void* a handle to the profile sensor. A NULL pointer is returned in case of failure
Description	This function will create a connection to the weCat3D sensor. The function will return a handle to the profile sensor, which will be used by other functions.



NOTE!

For checking the connection status with the profile sensor enter “EthernetScanner_GetConnectStatus”.

5.2 Closing Connection

Command	<code>void* _stdcall EthernetScanner_Disconnect(void *pEthernetScanner)</code>
Parameter 1	void*: the handle of the profile sensor (returned by the function EthernetScanner_Connect) to be disconnected
Response	void*: a handle to the profile sensor. In case of a successful disconnect operation, the function will return a null pointer, else it will return the profile sensor handle itself.
Description	Close the connection between the DLL and the weCat3D sensor.

5.3 Check Connection

Command	<code>void EthernetScanner_GetConnectStatus(void *pEthernetScanner, int *uiConnectStatus)</code>
Parameter 1	void* : a handle to the profile sensor returned by the function EthernetScanner_Connect
Parameter 2	int * : a pointer to an integer variable, through which the connection status is returned.
Response	---
Description	<p>This function checks the connection status to the profile sensor. The function is a non-blocking function.</p> <p>There are two states for the connection:</p> <ul style="list-style-type: none"> ETHERNETSCANNER_TCPSCANNERCONNECTED (3) meaning that the profile sensor is successfully connected to the DLL and the given IP and PORT in the function EthernetScanner_Connect are valid. In DLL version 1.9.0 or higher this state also means that the DLL is initialized (the DLL has received the linearization table from the profile sensor and all the internal parameters in the DLL have been set accordingly). From now on a valid profile can be polled using the function EthernetScanner_GetX-ZiExtended. ETHERNETSCANNER_TCPSCANNERDISCONNECTED (0) meaning that the profile sensor is disconnected or the given IP and PORT in the function EthernetScanner_Connect are not valid.

5.4 Get General Sensor Information (deprecated)

Command	<code>int EthernetScanner_GetInfo(void *pEthernetScanner, char *chInfo, int iBuffer, char *chMode)</code>
Parameter 1	<code>void*</code> : a handle to the profile sensor returned by the function <code>EthernetScanner_Connect</code>
Parameter 2	<code>char*</code> : a pointer to a raw buffer (of type <code>char</code>), where the profile sensor information will be written.
Parameter 3	<code>int</code> : the length of the raw buffer. The programmer should make sure that the length of the raw buffer is larger than the length of the returned sensor information. You can use the header definition <code>ETHERNETSCANNER_GETINFOSIZEMAX</code> provided in "EthernetScannerSDKDefine.h" to define the length of the raw buffer in parameter 2.
Parameter 4	<code>char*</code> : Defines the mode of the function. There are two different modes supported by the function: "text" and "XML" (see description below).
Response	<p><code>ETHERNETSCANNER_INVALIDHANDLE</code> (-1000) if the sensor handle (parameter 1) is NULL or invalid.</p> <p><u>In text mode:</u> If the size of the raw buffer (parameter 2) is smaller than the size of the data to be written, the function returns <code>ETHERNETSCANNER_GETINFOSMALLERBUFFER</code> (-2). In a successful operation the function returns the length of the data written into the raw buffer.</p> <p><u>In XML mode:</u> If the size of the raw buffer (parameter 2) is smaller than the size of the data to be written, the function returns <code>ETHERNET_GETINFOSMALLBUFFER</code> (-2). In a successful operation the function returns the length of the data written into the raw buffer. If the function fails to call the XML data from the profile sensor, it returns <code>ETHERNETSCANNER_GETINFOINVALIDXML</code> (-4).</p>
Description	<p><u>In text mode:</u> Returns basic information about the profile sensor as a text such as sensor name, working ranges, MAG, etc (see appendix 1 for an example)</p> <p><u>In XML mode:</u> Returns a full description of the profile sensor in XML format. The XML contains general information about the profile sensor, the current values of all features as well as all ASCII commands supported by the profile sensor in the firmware (see appendix 2 for an example).</p>

5.5 Get Scanned Profile

Command	<code>int EthernetScanner_GetXZExtended(void *pEthernetScanner, double *pdoX, double *pdoZ, int *piIntensity, int *piSignalWidth, int iBuffer, unsigned int *puiEncoder, unsigned int *pucUSRIO, int dwTimeOut, unsigned char *ucBufferRaw, int iBufferRaw, int *iPicCnt)</code>
Parameter 1	<code>void *</code> : a handle to the profile sensor returned by the function <code>EthernetScanner_Connect</code>
Parameter 2	<code>double*</code> : pointer to a raw buffer (of type <code>double</code>) used by the function to write in the X coordinates [in mm] of the measured profile. Pass NULL, if the buffer is not used
Parameter 3	<code>double*</code> : pointer to a raw buffer (of type <code>double</code>) used by the function to write in the Z coordinates [in mm] of the measured profile. Pass NULL, if the buffer is not used
Parameter 4	<code>int*</code> a pointer to a raw buffer (of type <code>int</code>) used by the function to write the intensity [10 bit] of the measured profile. Pass NULL, if this buffer is not used
Parameter 5	<code>int*</code> a pointer to a raw buffer (of type <code>int</code>) used by the function to write the peak width [in pixel < 32 pxl] . Pass NULL, if this buffer is not used

Parameter 6	int: the length of the raw buffers passed in parameter 2 to 5. The length of the raw buffers should be larger than the number of measured points returned by the profile sensor. You can use the header definition ETHERNETSCANNER_BUFFERSIZEMAX provided in "EthernetScannerSDKDefine.h" to define the length of the raw buffers in the parameters 2 to 5.
Parameter 7	int*: a pointer to a variable (of type int) which returns the encoder value of the current measured profile
Parameter 8	int*: a pointer to a variable (of type int) which returns the IO status of the current measured profile. The IO status is decoded as follows: bit0: EA1 bit1: EA2 bit2: EA3 bit3: EA4 bit4: TTL Encoder A bit5: TTL Encoder B bit6: TTL Encoder C Pass NULL, if this value is not used
Parameter 9	The value of the blocking time to wait for a new measured profile, until the function times out. The value 0 makes the function non-blocking (timeout in ms).
Parameter 10	Deprecated. Pass NULL
Parameter 11	Deprecated. Pass NULL
Parameter 12	int* : a pointer to variable (of type int) which returns the picture counter of current measured profile. This value is used to control the sequence of the received profiles.
Response	ETHERNETSCANNER_INVALIDHANDLE (-1000) if the sensor handle (parameter 1) is NULL or invalid. In the case of a success call, the function will return the total number of points of the measured profile written to the raw buffer (in parameter 2 to 5). The function will return ETHERNETSCANNER_GETXZINONEWSCAN (-1) if no new profile is available, ETHERNETSCANNER_GETXZIINVALIDBUFFER (-3) if the length of the buffer given in parameter 1 to 5 is shorter than the data to be written, ETHERNETSCANNER_GETXZIINVALIDLINDATA (-2) if the DLL is not initialized.
Description	The function pulls one scan from the internal FiFo in the DLL, if a new scan is available. The DLL saves all the scanned profiles received from the profile sensor in an internal FiFo buffer. The programer is responsible to pull the scanned profiles using this function as fast as possible to prevent overflow of the FiFo. If the program can not pull the scanned profiles fast enough, then it is recommended to decrease the output rate of the profile sensor. The function could be set to be blocking or non-blocking depending on the value of parameter 9. Set the function to be blocking (parameter 9 > 0) if you call the function from a secondary thread in your application. Set the function to be non blocking (parameter 9 = 0) if you call the function from the main thread in your application. To check the status of the FiFo see section 5.6. To know how to set up the output rate of the profile sensor see the ASCII commands SetAcquisitionLineTime in section 6.3.

5.6 Check DLL FiFo State

Command	int EthernetScanner_GetDllFiFoState(void *pEthernetScanner)
Parameter 1	void* : the handle to the profile sensor returned by the function EthernetScanner_Connect
Response	int: the status of the FiFo in the DLL in % (0 – 100) ETHERNETSCANNER_INVALIDHANDLE (-1000) if the sensor handle (parameter 1) is NULL or invalid.
Description	The function is used to check the status of the internal FiFo in the DLL to prevent the over-flow and hence, to prevent the loss of unpolled scanned profiles.

5.7 Reset DLL FiFo

Command	int EthernetScanner_ResetDllFiFo(void *pEthernetScanner)
Parameter 1	void * the handle to the profile sensor returned by the function EthernetScanner_Connect
Response	The function returns ETHERNETSCANNER_OK (0) if the calling was successful. ETHERNETSCANNER_INVALIDHANDLE (-1000) if the sensor handle (parameter 1) is NULL or invalid.
Description	The function is used to reset the internal FiFo in the DLL. However, that could lead to the loss of unpolled scanned profiles. This function is useful, if the application can not poll the scanned profiles fast enough and the programmer wants to process the latest scanned profile. In that case, it is recommended to call this function just before calling the function EthernetScanner_GetXZIEExtended.

5.8 Setup Profile Sensor



Command	int EthernetScanner_WriteData(void *pEthernetScanner, char *ucBuffer, int uiBuffer)
Parameter 1	void * the handle to the profile sensor returned by the function EthernetScanner_Connect
Parameter 2	char*: a pointer to a raw buffer (of type char) which contains the ASCII command to be sent to the profile sensor
Parameter 3	int: the length of the raw buffer passed in parameter 2
Response	The function returns the number of bytes sent to the profile sensor. Normally, it should be the same length as the ASCII comand. ETHERNETSCANNER_INVALIDHANDLE (-1000) if the scanner handle (parameter 1) is NULL or invalid.
Description	The function is used to send ASCII commands to setup the profile sensor. The supported ASCII commands can be found in section 6.

5.9 Read DLL Version

Command	<code>int EthernetScanner_GetVersion(unsigned char *ucBuffer, int uiBuffer)</code>
Parameter 1	char*: a pointer to a raw buffer (of type char) used by the function to write in the DLL version
Parameter 2	int: the length of the raw buffer used in parameter 1. You can use a length of 1024 to create the buffer passed in parameter1.
Response	The function returns the total length (in bytes) of the written data in the raw buffer. If the length of DLL version to be written in the raw buffer is larger than the length of the raw buffer given in parameter 2, the function returns ETHERNETSCANNER_ERROR (-1).
Description	The function is used to check the current version of the DLL.

5.10 Read Property Value

Command	<code>int EthernetScanner_ReadData(void *pEthernetScanner, char *chPropertyName, char *chRetBuf, int iRetBuf, int iCacheTime)</code>
Parameter 1	void * the handle to the profile sensor returned by the function EthernetScanner_Connect
Parameter 2	char * buffer with the ASCII command (ending with char \0)
Parameter 3	char * return buffer for the result of the ASCII command
Parameter 4	int the length of the return buffer. You can use the header definition ETHERNETSCANNER_BUFFERSIZEMAX provided in "EthernetScannerSDKDefine.h" to define the length of the raw buffer in parameter 3.
Parameter 5	int the cache time in ms; the value in this parameter defines the function mode (XML mode or scan mode). See the description below for details.
Response	<p>The function returns ETHERNETSCANNER_READDATAOK (0) in case of success operation, ETHERNETSCANNER_READDATASMALLBUFFER (-1) if the return buffer passed in parameter 3 is shorter than the length of the data available to be written in the buffer, ETHERNETSCANNER_READDATANOTSUPPORTEDMODE (-2) in the case where the given property is not supported in the current read mode (like PictureCounter in XML mode), ETHERNETSCANNER_READDATAFEATURENOTDEFINED (-3) if the property name is not supported,</p> <p>ETHERNETSCANNER_READDATANOSCAN (-4) if the function is called in scan mode and no scan is yet polled using the function EthernetScanner_GetXZiExtended,</p> <p>ETHERNETSCANNER_READDATAFAILED (-5) if the function failed to read data from XML data or from scan data.</p> <p>ETHERNETSCANNER_INVALIDHANDLE (-1000) if the sensor handle (parameter 1) is NULL or invalid.</p>

Description	<p>Starting from DLL version 1.9.0 or higher, the function <code>EthernetScanner_ReadData</code> is being introduced as a standard function in the SDK. The function reads the property values from the profile sensor. These values are cached in the DLL and the <code>iCacheTime</code> (parameter 5) defines how old the property value should be before writing it in the return buffer (parameter 3). The function and the supported property name does not depend on specific firmware of the profile sensor. The function is implemented in the DLL as a comfort function to make it easy for the programmer to read property values from the profile sensor.</p> <p>There are two operating function modes: XML mode and scan mode:</p> <ul style="list-style-type: none"> • XML mode is defined when the <code>iCacheTime</code> ≥ 0. In this mode, the data are fetched from the XML descriptor received from the profile sensor and cached in internal structure in the DLL. If the data cache is older than the given <code>iCacheTime</code> value, the DLL will call a new XML file from the profile sensor, parse it and cache the data in the internal structure and then write the property value in the return buffer. <p> NOTE! Setting a low value for <code>iCacheTime</code> in XML mode (i.e. <code>iCacheTime</code> = 0) will decrease the performance of the DLL since the DLL is then forced to read the full properties from the profile sensor and parse it each time the <code>EthernetScanner_ReadData</code> function is called. This would be evident if the DLL is working on low resource system or if the profile sensor works in range of kHz.</p> <ul style="list-style-type: none"> • Scan mode is defined when the <code>iCacheTime</code> = -1. The DLL in this mode reads the property value from the data delivered with the current scan (pulled using the function <code>EthernetScanner_GetXZIEExtended</code>). The property value in this mode will hold until the next successful call of the function <code>EthernetScanner_GetXZIEExtended</code>. <p>An example on how to use the new function can be found in the example code in the SDK.</p> <p> NOTE! Supported ASCII commands can be found in section 7. Not all properties are supported on both reading modes, see section 7 for more details.</p>
--------------------	--

6. Setup Profile Sensor

Below are the ASCII commands that are used to set up the profile sensor using the function EthernetScanner_WriteData.

6.1 Initiate Reboot

Command	SetReboot\r		
Description	Reboot the system		

6.2 Exposure Time

6.2.1 Fixed Exposure Time

Command	SetExposureTime=x\r		
Parameter	Range of value x: 0 ... 1000000	Default:	150
Description	Exposure time is set in μ s. If HDR mode is set (see section 6.4.1), SetExposureTime is the exposure time of the first profile. SetExposureTime2 is the exposure time of the second profile (see section 6.4.2).		

6.2.2 Auto Exposure Time

Command	SetAutoExposureMode=x\r		
Parameter	Values of x: 0: disabled 1: enabled	Default:	0
Description	Enables the automatic control of the exposure time		



NOTE!

Auto exposure time is available in firmware version 1.2.0 or higher.

6.2.2.1 Set the Minimum of Auto Exposure Time

Command	SetAutoExposureTimeMin=x\r		
Parameter	Range of value x: 10...100000	Default:	10
Description	Adjustment of the minimum exposure time in AutoExposureMode. The value is set in μ s.		

6.2.2.2 Set the Maximum of Auto Exposure Time

Command	SetAutoExposureTimeMax=x\r		
Parameter	Range of value x: 10...100000	Default:	1000
Description	Adjustment of the maximum exposure time in AutoExposureMode. The value is set in μ s.		

6.2.2.3 Set the Minimum of Intensity Range

Command	SetAutoExposureIntensityRangeMin=x\r		
Parameter	Range of value x: 0...1024	Default:	450
Description	Sets the lower limit of the intensity range.		

6.2.2.4 Set the Maximum of Intensity Range

Command	SetAutoExposureIntensityRangeMax=x\r		
Parameter	Range of value x: 0...1024	Default:	500
Description	Sets the upper limit of the intensity range.		



NOTE!

The intensity range should contain the area of the highest intensity. The exposure time is adjusted according to the average intensity of the selected range.

6.2.2.5 Set Minimum of Range X

Command	SetAutoExposureRangeXMin=x\r		
Parameter	Range of value x: MLSL: 0...1279 MLWL: 0...2047	Default:	MLSL: 64 MLWL: 64
Description	Sets the starting point of range X.		

6.2.2.6 Set Maximum of Range X

Command	SetAutoExposureRangeXMax=x\r		
Parameter	Range of value x: MLSL: 0...1279 MLWL: 0...2047	Default:	MLSL: 1215 MLWL: 1983
Description	Sets the ending point of range X.		



NOTE!

Range X defines the area where the control of the exposure time is applied.

6.3 Setup Acquisition Line Time

Command	SetAcquisitionLineTime=x\r		
Parameter	Range of value x: 166 ... 100000	Default:	MLWL: 5714 MLSL: 5000
Description	Time between two consecutive profiles in μ s. This function is only effective in internal trigger mode. 166 μ s = 6000 Hz Explanation: MLWL: 5714 μ s = 175 Hz MLSL: 5000 μ s = 200 Hz		

NOTE!

It is necessary to reduce the ROI settings and the scan contents in the profile sensor to get a higher LineTimeRate (please see [SetROI1HeightZ](#), section 6.35.4, [SetROI1WidthX](#), section 6.35.1, [SetSignalContentWidth](#), section 6.9 and [SetSignalContentReserved](#), section 6.10).





NOTE!

The profile sensor can transmit data through the network up to 30 MByte/s. Thus it is necessary to disable some signal contents to get a higher LineTimeRate (up to 6 kHz (166 μ s) in MLWL and 4 kHz (250 μ s) in MLSL)..

6.4 HDR Mode

High Dynamic Range Imaging (HDR) is used to record objects with a very high intensity contrast. With firmware version 1.1.3 and higher HDR is implemented in the weCat3D sensors using the method of recording two profiles with different exposure times. The generation of the HDR profile based on the two profiles must be done by the user.

6.4.1 Set HDR

(available from FW version 1.1.3)

Command	SetHDR=x\r		
Parameter	Values of x: 0: HDR disabled 1: HDR enabled	Default:	0
Description	Enables/disables HDR mode.		

6.4.2 Setup Exposure Time 2

(available from FW version 1.1.3)

Command	SetExposureTime2=x\r		
Parameter	Range of value x: 0 ... 1000000	Default:	150
Description	Exposure time 2 is set in μ s. If HDR mode is set (see section 6.4.1), SetExposureTime2 is the exposure time of the second profile. SetExposureTime is the exposure time of the first profile (see section 6.2.1).		

6.5 Deactivate Laser

Command	SetLaserDeactivated=x\r		
Parameter	Values of x: 0: Laser on 1: Laser off	Default:	0
Description	Default Laser on Software command to control the laser as a global function. If this function is set to 1(enabled), then all other enabled signals on the E/A do not have any effects.		

6.6 Set User LED

Command	SetUserLED=x\r		
Parameter	Values of x: 0: off 1: red 2: green 3: orange	Default:	0
Description	The command controls the user LED for optical display of the application status directly at the weCat3D sensor.		

6.7 Enable Signal (Z)

Command	SetSignalContentZ=x\r		
Parameter	Values of x: 0: disabled 1: enabled	Default:	1
Description	By default, the data sent from the profile sensor contains Z (the depth), X (the width), I (the intensity / signal strength) and the peak width. This command will disable sending the Z signal value to save the bandwidth of the network.		

6.8 Enable Signal (Strength)

Command	SetSignalContentStrength=x\r		
Parameter	Values of x: 0: disabled 1: enabled	Default:	1
Description	By default, the data sent from the profile sensor contains Z (the depth), X (the width), I (the intensity / signal strength) and the peak width. This command will disable sending the I signal value to save the bandwidth of the network.		

6.9 Enable Signal (Width)

Command	SetSignalContentWidth=x\r		
Parameter	Values of x: 0: disabled 1: enabled	Default:	1
Description	By default, the data sent from the profile sensor contains Z (the depth), X (the width), I (the intensity / signal strength) and the peak width. This command will disable sending the width signal value to save the bandwidth of the network.		

6.10 Enable Signal End Position Command

Command	SetSignalContentReserved=x\r		
Parameter	Values of x: 0: disabled 1: enabled	Default:	1
Description	By default, the data sent from the profile sensor contains Z (the depth), x (the width), I (the intensity / signal strength), the peak width and reserved. This command will disable sending the reserved signal value to save the bandwidth of the network.		

6.11 Setup Socket Connection Timeout

Command	SetSocketConnectionTimeout=x\r		
Parameter	Range of value x: 0 ... 60000ms	Default:	0
Description	Profile sensor Ethernet connection: rx-tx-timeout in ms. Default: 0 no connection will be closed, if no ethernet data has been transferred (rx/tx).		

6.12 Setup The Heartbeat Signal

Command	SetHeartBeat=x\r		
Parameter	Range of value x: 0 ... 10000 ms	Default:	0
Description	The command will activate the heartbeat signal in the profile sensor. If the profile sensor does not send/receive any data, it sends every x ms a heartbeat signal (XML file). x = 0 deactivates the heartbeat signal.		

NOTE!



It is recommended to activate the heartbeat signal in the profile sensor, the heartbeat signal will enable the profile sensor to detect a physical (electrical) connection drop (like in the case where the network cable is unplugged). Thus, the profile sensor closes the connection to the host and allowing the host to build a new connection to the profile sensor. The recommended value is 1000 ms.

6.13 Start Acquisition

Command	SetAcquisitionStart\r		
Description	After opening the socket connection this command is active and the profile data will be sent to the host (default).		

6.14 Stop Acquisition

Command	SetAcquisitionStop\r		
Description	The profile data will stop sending to the host.		

**NOTE!**

Continue reading data from the sensor until no data arrives to be sure that no data remains in the FiFo's.

6.15 Reset Settings

Command	SetResetSettings\r
Description	Loads the profile sensor default settings. The IP address of the profile sensor is retained.

**NOTE!**

A sleep time (1000 ms) should be added after executing the command „SetResetSettings“.

**NOTE!**

SetResetSettings command does not load the profile sensor factory settings. Only "Reset Sensor Settings" button in the profile sensor web interface resets the sensor settings to factory settings.

6.16 Reset Encoder

Command	SetResetEncoder\r
Description	Set the encoder counter of both encoders (HTL and TTL) to 0.

6.17 Reset Picture Counter

Command	SetResetPictureCounter\r
Description	Set the value of the picture counter to 0.

**NOTE!**

One of the usual cases is to set the picture counter to 0 during SetAcquisitionStop. After SetAcquisitionStop do all settings or reset counters then SetAcquisitionStart. Please look to the commands SetAcquisitionStop and SetAcquisitionStart.

6.18 Save Settings

Command	SetSettingsSave=x\r		
Parameter	Values of x: 0, 1, 2	Default:	0
Description	0: default (this setting will be saved automatically for the load after the power supply of the profile sensor is switched on) 1: Set1 2: Set2		

Command	SetResetBaseTimeCounter\r
Description	Set the basetime counter of the sensor to 0.

6.19 Load Settings

Command	SetSettingsLoad=x\r		
Parameter	Values of x: 0, 1, 2	Default:	0
Description	0: default (this setting will be loaded after the power supply of the profile sensor is switched on) 1: Set1 2: Set2		

6.20 Setup Trigger Source

Command	SetTriggerSource=x\r		
Parameter	Values of x: -1: Fixed trigger mode (read only) 0: Internal trigger mode 1: Hardware trigger mode over SyncIn function on E/A1...E/A4 2: Encoder trigger mode over HTL/TTL encoder 3: Software trigger mode	Default:	0
Description	The function is used to activate the trigger source of the profile sensor in dynamic trigger mode (see "Trigger Settings" in Operating Instructions of MLWL / MLSL sensors). Set 0 for internal triggering. Set 1 to trigger the profile sensor through hardware signal (useful for synchronizing multiple profile sensors in an application). Set 2 to trigger the profile sensor through the encoder signal (if E/A 1 and E/A 2 are defined as encoder function, the E/A encoder will be used as the trigger source, otherwise the TTL-RS422 is used). Set 3 to trigger the signal through the software command "SetTriggerSoftware". The function returns "-1" if the dynamic trigger mode in the profile sensor is switched off (the profile sensor is working in fixed trigger mode). The value "-1" can not be used in this function. If you want to switch off the dynamic trigger mode, please use the command SetTriggerAmountProfilesY .		

NOTE!



If the trigger source in the profile sensor is setup to encoder, hardware or software, and the profile sensor did not receive a trigger signal within the time defined in the iTimeOut input parameter (parameter 3) in "EthernetScanner_Connect"; the DLL will close the connection to the profile sensor and will build a new connection to it. To avoid this behaviour, you have either to set the iTimeOut value in "EthernetScanner_Connect" to 0 (see section 5.1) or setup the heartbeat signal to, for example, 1000 (see section 6.12).

6.21 Setup Trigger Divider

Command	SetTriggerEncoderStep=x\r		
Parameter	Range of value x: 0 ... 65535	Default:	0
Description	Set a trigger divider for both hardware trigger source (Syncln input) and encoder trigger source (Encoder HTL or TTL). The profile sensor will be triggered at the x+1 signal. This property is useful, if we have a high frequency external trigger source (either Encoder or Syncln signal).		

6.22 Setup Trigger Delay

Command	SetTriggerDelay=x\r		
Parameter	Range of value x: 0 ... 100 000	Default:	0
Description	Trigger delay is usually used in the slave sensor in multi-sensor setup. Trigger delay is set in μ s.		



NOTE!
Trigger delay + exposure time in the slave sensor should be smaller than the Acquisition-LineTime in the master sensor.

6.23 Software Trigger Command

Command	SetTriggerSoftware\r		
Parameter	---		
Description	Trigger the profile sensor to scan a profile over a software command. The profile sensor should be in software trigger mode.		

6.24 Setup Encoder Trigger Function

Command	SetEncoderTriggerFunction=x\r		
Parameter	Values of x: 0: DirectionUp 1: DirectionDown 2: Motion 3: PositionUp 4: PositionDown	Default:	2
Description	DirectionUp: The encoder will trigger the profile sensor only in one direction (counting up) DirectionDown: The encoder will trigger the profile sensor only in one direction (counting down). Motion: The encoder will trigger the profile sensor in both directions (counting up and down) PositionUp: The encoder will trigger the profile sensor in one direction (counting up) and only if the encoder position (counter value) is larger than the latest position. PositionDown: The encoder will trigger the profile sensor in one direction (counting down) and only if the encoder position (counter value) is smaller than the latest position.		

6.25 Enable Fixed Frame mode

Command	<code>SetTriggerAmountProfilesY=x\r</code>		
Parameter	Values of x: -1: Fixed trigger mode (read only) 0: Internal trigger mode 1: Hardware (SyncIn) trigger mode 2: Encoder trigger mode 3: Software trigger mode	Default:	-1
Description	This command is used to activate the fixed trigger mode in the profile sensor. In fixed trigger mode the profile sensor sends x number of profiles ($x = 0, 1, 2$ or 3)(see <code>SetAmountOfProfilesY</code>) to the host and then stops until the profile sensor receives a new <code>SetAcquisitionStart</code> command or hardware signal on ProfileEnabel pin (if defined). -1 means that the fixed frame mode in the profile sensor is switched off (the profile sensor is working in dynamic trigger mode). The value -1 can not be used in this function. If you want to switch off the fixed frame mode, please use the command <code>SetTriggerSource=x</code> .		

NOTE!



If the trigger source in the profile sensor is setup to encoder, hardware or software, and the profile sensor did not receive a trigger signal within the time defined in the `iTimeOut` input parameter (parameter 3) in "EthernetScanner_Connect"; the DLL will close the connection to the profile sensor and will build a new connection to it. To avoid this behaviour, you have either to set the `iTimeOut` value in "EthernetScanner_Connect" to 0 (see section 5.1) or setup the heartbeat signal to, for example, 1000 (see section 6.12).

6.26 Setup Number of Profiles in Fixed Frame Mode

Command	<code>SetAmountProfilesY=x\r</code>		
Parameter	Range of value x: 0...10000	Default:	0
Description	The command sets up the number of profiles to be sent to the host in the fixed frame mode (see 6.25).		

6.27 Setup Sync Out

Command	<code>SetSyncOut=x\r</code>		
Parameter	Range of value x: 10...100000	Default:	1000
Description	Defines the peak width (duration in μs) of the SyncOut signal (high) for the E/A SyncOut. The value of SyncOut peak width and the SyncOutDelay time (see 6.28) combined should be less than the AcquisitionLineTime value (see 6.3). This is important in order to prevent having one long SyncOut signal during the acquisition mode.		



NOTE!

The minimum peak width should be $10 \mu s$.

6.28 Setup Delay of Sync Out

Command	SetSyncOutDelay=x\r		
Parameter	Range of value x: 0...100000	Default:	0
Description	Defines the value of (switching) delay (in μ s) of the SyncOut trigger signal (high) for the E/A SyncOut.		

6.29 Enable Signal

Command	SetSignalEnable=x\r		
Parameter	Values of x: 1: Signal 1 2: Signal 2 3: Signal 1 + Signal 2	Default:	1
Description	The command sets the number of signals which are send with each profile. See the description of signal selection.		

6.30 Setup Signal Minimum Width

Command	SetSignalWidthMin=x\r		
Parameter	Range of value x: 0...63	Default:	0
Description	Peak width filter: This function is a filter to define the minimum peak width in pixels. Usual values: 2 or 3		

6.31 Setup Signal Maximum Width

Command	SetSignalWidthMax=x\r		
Parameter	Range of value x: 0...63	Default:	63
Description	Peak width filter: This function is a filter to define the maximum peak width in pixels. Usual values: 12		

6.32 Setup Signal Selection

Command	SetSignalSelection=x\r		
Parameter	Values of x: 0: top 1: strong 2: width 3: bottom	Default:	1
Description	Defines the signal which is to be used for the profile output. The sensor acquires internally two signals. Based on this selection the sensor provides the corrensponding signal.		

6.33 Internal Profile Calculation

Command	<code>SetLinearizationMode=x\r</code>		
Parameter	x = 0 (disabled) x = 1 (enabled)	Default:	0
Description	<p>The weCat3D profile sensors have the possibility to calculate the profile internally or externally using the SDK of the weCat3D sensor. If the profile is calculated internally, the calculated profiles are submitted via a TCP/IP protocol (please refer to section 9). If set to 1 the internal calculation is enabled.</p> <p>Before switching between internal or external calculation it must be ensured that no data are still transmitted. The program flow is:</p> <pre>SetAcquisitionStop\r //wait until no data are received by host SetLinearizationMode=1\r SetAcquisitionStart\r</pre>		



NOTE!

This command decreases the CPU load on the host.



NOTE!

SetLinearizationMode command is available in firmware version 1.2.0 or higher.

6.34 Setup Encoder Count Direction

Command	<code>SetEncoderCountDirection=x\r</code>		
Parameter	Values of x: 0: normal 1: invert	Default:	0
Description	The count direction of the encoder values can be inverted.		

6.35 Region of interest (ROI)

6.35.1 Setup ROI Width in X

Command	<code>SetROI1WidthX=x\r</code>		
Parameter	Range of value x: MLWL: 32...1280 MLWL: 32...2048	Default:	MLWL: 1280 MLWL: 2048
Description	<p>Amount of camera rows to readout:</p> <p>MLWL: no effect on the measurement rate, effect on the ethernet bandwidth</p> <p>MLSL: in steps of 16, effect on the measurement rate, effect on the ethernet bandwidth</p>		

6.35.2 Setup ROI Offset in X

Command	SetROI1OffsetX=x\r		
Parameter	Range of value x: MLWL: 0...1247 MLWL: 0...2047	Default:	0
Description	MLWL: in steps of 1 MLSL: in steps of 32 Defines the offset of the ROI in X-direction in relation to the first line.		

6.35.3 Setup ROI Step X

Command	SetROI1StepX=x\r		
Parameter	Values of x: 0: disabled 1: MLWL subsampling enabled, MLWL only step 1 2 ... x: only steps	Default:	0
Description	MLSL: If amount of pixel in the CMOS line (width X) set to half then the range of X looks like full. Speed can be increased by double. MLWL: Decreases only the amount of data, has no effect to speed.		

6.35.4 Setup ROI Height in Z

Command	SetROI1HeightZ=x\r		
Parameter	Range of value x: MLWL: 35...1024 MLWL: 35...2048	Default:	MLWL: 1280 MLWL: 2048
Description	Amount of camera lines to readout has an effect on the Ethernet bandwidth and the measurement rate.		

6.35.5 Setup ROI Offset in Z

Command	SetROI1OffsetZ=x\r		
Parameter	Range of value x: MLWL: 0...1023 MLWL: 0...2047	Default:	0
Description	Defines the offset of the ROI in Z-direction in relation to the first line.		

6.35.6 Setup ROI Step Z

Command	SetROI1StepZ=x\r		
Parameter	Values of x: 0: disabled 1: enabled	Default:	0
Description	If amount of CMOS lines (height Z) set to half then the range of Z looks like full. Speed can be increased by double.		

6.36 E/A Functions

6.36.1 Setup E/A Functions

The profile sensor offers 4 separate E/A functions. The following commands relate to these E/A functions and can be used for all E/As. The encoder HTL functions are only available for E/A 1 and E/A 2. The following explanation uses the syntax to set up the E/A 1. For addressing E/A 2 to E/A 4 use the same syntax:

SetEA1Function=1

SetEA2Function=2

SetEA3FunctionLaserOff=0

Command	SetEA1Function=x\r		
Parameter	Values of x: 1: sync_in 2: sync_out 3: input 4: output 5: encoder_ab	Default:	5
Description	Encoder_A/B (E/A 1 + E/A 2): Input function for connecting an HTL (5 to 24 V, A/B channel) rotary encoder. This function must be set for E/A 1 and E/A 2 at the same time. This function is only available for E/A 1 and E/A 2. If the encoder function is enabled on E/A 1/2, then the encoder value in the GetXZI function will be provided from this encoder! If no E/A 1/2 encoder function is selected, then the encoder value in the GetXZI function will be provided from TTL-RS422.		

6.36.2 Setup E/A Function Laser Off

Command	SetEA1FunctionLaserOff=x\r		
Parameter	Values of x: 0: disabled 1: enabled	Default:	0
Description	E/A high state: laser is off E/A low state: laser is on The E/A should be set to input (see section 6.36.1) for this function to work.		

6.36.3 Setup E/A Function Profile Enable

Command	SetEA1FunctionProfileEnable=x\r		
Parameter	Values of x: 0: disabled 1: enabled	Default:	0
Description	E/A high state: profiles will be send to the host The E/A should be set to input (see section 6.36.1) for this function to work.		

6.36.4 Setup E/A Function Reset Enable

Command	SetEA1FunctionResetCounter=x\r
Parameter	Values of x: 0: disabled 1: enabled
Description	Enables the E/A pin to reset one or more counters in the profile sensor (see example SetEA1ResetCounterEncoderHTL or SetEA1ResetCounterBaseTime). The E/A should be set to input (see section 6.36.1) for a working function.

6.36.5 Setup E/A Repeat Reset Behaviour

Command	SetEA1ResetCounterRepeat=x\r
Parameter	Values of x: 0: disabled 1: once 2: always
Description	If the function is disabled, the E/A will not reset any counter. If it is "1" the E/A will reset the counter only once when the E/A is active. If you need to reset the counter again, the command should be sent again to the profile sensor. "2" means that the reset counter E/A will always reset the counter each time the E/A is active.

6.36.6 Setup E/A Reset Signal Edge

Command	SetEA1ResetCounterSignaledge=x\r
Parameter	Values of x: 0: rising and falling edge 1: rising edge 2: falling edge
Description	Defines the edge of the signal to reset the counter. The E/A should be defined as an input, reset counter and reset counter repeat should be active (see section 6.36.1 , 6.36.4 and 6.36.5).

6.36.7 Setup E/A Reset Base Time Counter

Command	SetEA1ResetCounterBaseTimeCounter=x\r
Parameter	Values of x: 0: disabled 1: enabled
Description	Enables the E/A to reset the basetime counter in the scanner. The E/A should be defined as an input, reset counter and reset counter repeat should be active (see section 6.36.1 , 6.36.4 and 6.36.5).

6.36.8 Setup E/A Reset Picture Counter

Command	<code>SetEA1ResetCounterPictureCounter=x\r</code>		
Parameter	Values of x: 0: disabled 1: enabled		
Description	Enables the E/A to reset the picture counter in the scanner. The E/A should be defined as an input, reset counter and reset counter repeat should be active (see section 6.36.1, 6.36.4 and 6.36.5).		

6.36.9 Setup E/A Reset Encoder HTL

Command	<code>SetEA1ResetCounterEncoderHTL=x\r</code>		
Parameter	Values of x: 0: disabled 1: enabled		
Description	Enables the E/A to reset the HTL encoder counter in the profile sensor. The E/A should be defined as an input, reset counter and reset counter repeat should be active (see section 6.36.1, 6.36.4 and 6.36.5).		

6.36.10 Setup E/A Reset Encoder TTL

Command	<code>SetEA1ResetCounterEncoderTTLRS422=x\r</code>		
Parameter	Values of x: 0: disabled 1: enabled		
Description	Enables the E/A to reset the TTL encoder counter in the profile sensor. The E/A should be defined as an input, reset counter and reset counter repeat should be active (see section 6.36.1, 6.36.4 and 6.36.5).		

Example 1:

Setting E/A 3 to reset HTL encoder and TTL encoder each time it receives a high signal:

```
SetEA3Function=3\rSetEA3FunctionResetCounter=1\rSetEA3ResetCounterRepeat=2\rSetEA3ResetCounter-
Signaleedge=2\rSetEA3ResetCounterEncoderHTL=1\rSetEA3ResetCounterEncoderTTLRS422=1\r
```

6.36.11 Setup E/A 1 Input Function

Command	<code>SetEA1InputFunction=x\r</code>		
Parameter	Values of x: 0: Ub inactive 1: Ub active	Default:	1
Description	The input signal can be inverted as a function.		

6.36.12 Setup E/A 1 Input Load

Command	SetEA1InputLoad=x\r		
Parameter	Values of x: 0: input load disabled 0 mA 1: input load enabled 2 mA	Default:	0
Description	Enable/disable the extra load on the E/A input to get 0 level defined (Helpful for some PLC hardware).		

6.36.13 Setup E/A 1 Output

Command	SetEA1Output=x\r		
Parameter	Values of x: 1: Push-Pull 2: PNP 3: NPN	Default:	1
Description	Determines the output mode for the E/A (Push-Pull, PNP or NPN).		

6.36.14 Setup E/A 1 Output Function

Command	SetEA1OutputFunction=x\r		
Parameter	Values of x: 0: NO 1: NC	Default:	0
Description	0: NO (normally open) 1: NC (normally close)		

6.36.15 Setup E/A1 Input Counter

Command	SetEA1FunctionInputCounter=x\r		
Parameter	Values of x: 0: disable 2: enable	Default:	0
Description	Enables/disables the user counter function at the E/A. Use the ASCII command GetEAFunctionInputCounter to read the counter value, see section 7.		



NOTE!
The E/A should be set to input (Syncln or UserInput) for the counter to work.

6.37 Setup User Data

Command	SetStatisticDataUserData=x\r
Parameter	Range of value x: 0...65535
Description	This command helps the user to synchronize the communication between the host and the profile sensor. The command writes a user defined data into internal register (2 bytes) in the profile sensor. The user can read back the value using the command GetStatisticDataUserData . in function EthernetScanner_ReadData in scan mode, see section 5.10 and section 7.



NOTE!

Available in firmware version 1.2.0 or higher and DLL version 1.10.0 or higher.

6.38 Setup the Shared Library Internal FiFo Size

Command	SetLibraryScannerFiFoSize=x\r		
Parameter	Range of value x: 4198400 ... 4294967295 (in bytes)	Default:	41984000
Description	This command is used to setup the shared library internal FiFo size in bytes. Call this command before calling the function EthernetScanner_Connect. <u>Example:</u> EthernetScanner_WriteData(0,"SetLibraryScannerFiFoSize=4198400", strlen("SetLibraryScannerFiFoSize=4198400"));		



NOTE!

This command is implemented in the DLL internally and not supported by the FW in the profile sensor.

6.39 Setup the Shared Library Internal FiFo Mode

Command	SetLibraryScannerFiFoMode=x\r		
Parameter	Values of x: 0, 1	Default:	1
Description	x=0 deactivates the internal DLL FiFo buffer and the function EthernetScanner_GetXZExtended delivers in this mode the latest available received profile (ignoring all other older profiles). x=1 activates the internal DLL FiFo buffer functionality.		



NOTE!

This command is implemented in the DLL internally and not supported by the FW in the profile sensor.

7. Read Properties of weCat3D Profile Sensor

The following table shows the current ASCII commands that can be used to read the properties of the profile sensor using the function `EthernetScanner_ReadData`.

The table shows also the availability of each command for each read mode.

See the demo project in the SDK for a code example.

ASCII command	XML mode	Scan mode	Remarks
GetPictureCounter	o	x	
GetTimestamp (was GetSystemTime)	o	x	in μ s (GetSystemTime is deprecated. It is recommended to use from now on the new command GetTime-stamp).
GetStatisticDataUserData	o	x	
GetOrderNumber	x	o	
GetProductVersion	x	o	
GetProducer	x	o	
GetFirmwareVersion	x	o	
GetSerialNumber	x	o	
GetMAC	x	o	
GetWorkingRangeZStart	x	o	
GetWorkingRangeZEnd	x	o	
GetFieldWidthXStart	x	o	
GetFieldWidthXEnd	x	o	
GetPixelXMax	x	o	
GetPixelZMax	x	o	
GetOnOffCounter	x	o	
GetOnTimeCounter	x	o	
GetLinInfo	x	o	if the sensor is calibrated
GetUserString	x	o	
GetHeartBeat	x	o	
GetSocketConnectionTimeout	x	o	
GetIOState	x	x	bit0: E/A 1 bit1: E/A 2 bit2: E/A 3 bit3: E/A 4
GetEncoderHTL	x	x	
GetEncoderTTL	x	x	
GetEncoder	o	x	Returns the value of the active encoder. The active encoder could be the HTL or the TTL encoder depending on the E/A setup of the profile sensor.
GetTemperature	x	x	

ASCII command	XML mode	Scan mode	Remarks
GetScannerState	x	x	bit0: Profile scanner OK bit1: ExposureTime OK bit2: LaserONTime OK bit3: Not in use bit4: Not in use bit5: Measurement rate too fast bit6: Not in use bit7: Not in use
GetSignalEnable	x	x	The number of signals in each scan, see function SetSignalEnable
GetSignalContentZ	x	x	
GetSignalContentStrength	x	x	
GetSignalContentWidth	x	x	
GetSignalContentReserved	x	x	
GetSignalWidthMin	x	x	
GetSignalWidthMax	x	x	
GetSignalStrengthMin	x	x	
GetSignalSelection	x	x	
GetAcquisitionLineTime	x	x	
GetCameraRunning	x	x	
GetTriggerSource	x	x	
GetTriggerAmountProfilesY	x	x	
GetAmountProfilesY	x	x	
GetTriggerEncoderStep	x	x	
GetTriggerDelay	x	x	
GetExposureTime	x	x	
GetLaserActive	x	x	
GetROI1WidthX	x	x	
GetROI1OffsetX	x	x	
GetROI1StepX	x	x	
GetROI1HeightZ	x	x	
GetROI1OffsetZ	x	x	
GetSyncOut	x	x	
GetSyncOutDelay	x	x	
GetEncoderTriggerFunction	x	x	
GetEncoderCountDirection	x	x	
GetEA1Function	x	x	
GetEA1FunctionLaserOff	x	x	
GetEA1FunctionProfileEnable	x	x	
GetEA1FunctionResetCounter	x	x	
GetEA1InputFunction	x	x	
GetEA1InputLoad	x	x	
GetEA1Output	x	x	

ASCII command	XML mode	Scan mode	Remarks
GetEA1OutputFunction	x	x	
GetEA1ResetCounterRepeat	x	x	
GetEA1ResetCounterSignaledge	x	x	
GetEA1ResetCounterBaseTimeCounter	x	x	
GetEA1ResetCounterPictureCounter	x	x	
GetEA1ResetCounterEncoderHTL	x	x	
GetEA1ResetCounterEncoderTTLRS422	x	x	
GetEA2Function	x	x	
GetEA2FunctionLaserOff	x	x	
GetEA2FunctionProfileEnable	x	x	
GetEA2FunctionResetCounter	x	x	
GetEA2InputFunction	x	x	
GetEA2InputLoad	x	x	
GetEA2Output	x	x	
GetEA2OutputFunction	x	x	
GetEA2ResetCounterRepeat	x	x	
GetEA2ResetCounterSignaledge	x	x	
GetEA2ResetCounterBaseTimeCounter	x	x	
GetEA2ResetCounterPictureCounter	x	x	
GetEA2ResetCounterEncoderHTL	x	x	
GetEA2ResetCounterEncoderTTLRS422	x	x	
GetEA3Function	x	x	
GetEA3FunctionLaserOff	x	x	
GetEA3FunctionProfileEnable	x	x	
GetEA3FunctionResetCounter	x	x	
GetEA3InputFunction	x	x	
GetEA3InputLoad	x	x	
GetEA3Output	x	x	
GetEA3OutputFunction	x	x	
GetEA3ResetCounterRepeat	x	x	
GetEA3ResetCounterSignaledge	x	x	
GetEA3ResetCounterBaseTimeCounter	x	x	
GetEA3ResetCounterPictureCounter	x	x	
GetEA3ResetCounterEncoderHTL	x	x	
GetEA3ResetCounterEncoderTTLRS422	x	x	
GetEA4Function	x	x	
GetEA4FunctionLaserOff	x	x	
GetEA4FunctionProfileEnable	x	x	
GetEA4FunctionResetCounter	x	x	
GetEA4InputFunction	x	x	
GetEA4InputLoad	x	x	
GetEA4Output	x	x	

ASCII command	XML mode	Scan mode	Remarks
GetEA4OutputFunction	x	x	
GetEA4ResetCounterRepeat	x	x	
GetEA4ResetCounterSignaledge	x	x	
GetEA4ResetCounterBaseTimeCounter	x	x	
GetEA4ResetCounterPictureCounter	x	x	
GetEA4ResetCounterEncoderHTL	x	x	
GetEA4ResetCounterEncoderTTLRS422	x	x	
GetEAFunctionInputCounter	?	x	
GetSettings=0	x	x	Returns the saved settings of the profile sensor in default as xml structure
GetSettings=1	x	x	Returns the saved settings of the profile sensor in set1 as xml structure
GetSettings=2	x	x	Returns the saved settings of the profile sensor in set2 as xml structure
GetSettings=3	x	x	Returns the current settings of the profile sensor as xml structure
GetCheckLinearizationMode	x	x	Returns "1" if the sensor supports profile linearization internally, "0" if the sensor does not. See SetLinearizationMode command for more details.

(x) is available; (o) is not available

8. Data Structure

8.1 General

The profile information queried by the GetXZExtended function are displayed separately as buffer for each value (X,Z,I). If the measured object is located outside the measuring range, the measured value is set to 0.

8.1.1 Buffer Structure (one selected signal)

In case of just one selected signal (signal selection) the buffer structure appears in this order:

Buffer	X	Buffer	Z	Buffer	I	Buffer	Peakwidth	
0	double	0	double	0	int	0	int	1 st point
1	double	1	double	1	int	1	int	2 nd point
2	double	2	double	2	int	2	int	3 rd point
... *			

* to ...1280 MLSL / ...2048 MLWL

8.1.2 Buffer Structure (two selected signals)

If the signal selection is set up to get signal 1 and signal 2, the buffer contains the data in the following, different order:

Buffer	X	Buffer	Z	Buffer	I	Buffer	Peakwidth	
0	double	0	double	0	int	0	int	1 st point 1 st signal
1	double	1	double	1	int	1	int	1 st point 2 nd signal
2	double	2	double	2	int	2	int	2 nd point 1 st signal
3	double	3	double	3	int	3	int	2 nd point 2 nd signal
... *			

* to ...2560 MSLSL / ...4096 MLWL

9. TCP/IP Socket Interface

9.1 Introduction

The weCat3D sensor has a TCP/IP socket interface which needs only a working TCP/IP socket communication. Over the TCP/IP socket interface the commands can be transmitted in ASCII format. The data packet is in a binary format. The TCP/IP socket interface is available in FW 1.2.0 or higher.

9.2 Setup the TCP/IP Socket Communication

To establish a TCP/IP socket communication please follow the steps below:

- 1. Open a client TCP/IP socket communication to the sensor via port 32001
- 2. Initialize the TCP/IP socket interface of the sensor by sending following commands (\r = carriage return)
 - a. [SetAcquisitionStop\r](#)
 - b. Wait until all data is read out
 - c. [SetInitializeAcquisition\r](#)
 - d. [SetLinearizationMode=1\r](#)
 - e. [SetAcquisitionStart\r](#)

Now the sensor transmits sensor information and profile data via the TCP/IP socket.

To stop the transmission use

- a. [SetAcquisitionStop\r](#)
- b. Wait until all data is read out

9.3 Data Format Definition

9.3.1 Basic Data Formats

Type	Name	Size in bytes
Unsigned int	Unsigned integer	4
Unsigned short	Unsigned integer	2
Unsigned char	Unsigned integer	1
Signed char	Signed integer	1
Float	Floating point number	4
Void	Void data type	not defined
Unsigned int[n]	Array unsigned integer of length n	4*n
Unsigned short[n]	Array unsigned integer of length n	2*n
Unsigned char[n]	Array unsigned integer of length n	1*n
Float[n]	Array floating point number of length n	4*n

9.3.2 Complex Data Formats

Type	Name	Content	Description	Type	Size in bytes
ROIxDetail	Complex data type ROIx definition	Start	Start of ROI in X in pixel	unsigned short	6
		Length	Length of ROI in X in pixel	unsigned short	
		Step	Step size ROI in X MLWL: any, without speed increasing MLSL: 1 = subsampling (every second column is read out)	unsigned short	
ROIxDetail[n]	Array ROIxDetail of length n				6*n
ROIzDetail	Complex data type ROIz definition	Start	Start of ROI in Z in pixel	unsigned short	6
		Length	Length of ROI in Z in pixel	unsigned short	
		Step	Step size ROI in Z MLWL/MLSL: 0 = no subsampling MLWL/MLSL: 1 = subsampling (every second line is read out)	unsigned short	
ROIzDetail[n]	Array ROIzDetail of length n				6*n

9.4 General Structure

Each data packet (Container) starts with the Container-Tag and ends with a CRC-Tag (checksum). In the container other tags containing sensor information and measurement data.

Byte Offset	Tag	Name	Size in bytes	Type	Min. occurrence	Max. occurrence
0	0x021A01FF	Container ID	4	unsigned int	1	1
Variable	0x021A0101	General ID	4	unsigned int	0	1
Variable	0x021A0102	Statistic ID	4	unsigned int	0	1
Variable	0x021A0103	Description ID (xml)	4	unsigned int	0	1
Variable	0x021A0201	ID-ROI-X	4	unsigned int	0	1
Variable	0x021A0202	ID-ROI-Z	4	unsigned int	0	1
Variable	0x021A0301	ID-RegisterCameraMLSL	4	unsigned int	0	1
Variable	0x021A0302	ID-RegisterCameraMLWL	4	unsigned int	0	1
Variable	0x021A0401	ID-RegisterFPGAMLSL	4	unsigned int	0	1
Variable	0x021A0402	ID-RegisterFPGAMLWL	4	unsigned int	0	1
Variable	0x021A0601	ID-Scan	4	unsigned int	0	1
Variable	0x021A0602	ID-ScanLinear	4	unsigned int	0	1
Variable	0x021A0701	ID-FPGACoeffitionBlock	4	unsigned int	0	1
Variable	0x00000001	SubID-ScanLinearHeader	4	unsigned int	0	1
Variable	0x00000002	SubID-ScanLinearData	4	unsigned int	0	1
Variable	0x021A0801	ID-ScaleParam	4	unsigned int	0	1
Container-ID-Size-12	0x021AFFFF	CRC-ID	4	unsigned int	1	1

9.5 Structure of a Tag

Every tag starts with the tag ID and the total size of the tag in bytes.

Element	Description	Size in bytes	Type
Tag ID	Unique ID of the tag	4	unsigned int
Tag Size	Size of the tag in bytes	4	unsigned int
Tag Content	Content of the tag, depending on type	Tag size - 8 bytes	depending on tag

9.6 Description of Tag

The byte offset is always related to the beginning of the tag. All examples are in little endian formatted.

9.6.1 Container Tag

The container tag contains the root of the data structure.

Byte Offset	Tag Data	Description	Size in bytes	Type
0	Container-ID	0x021A01FF A complete data package is included in the container	4	unsigned int
4	Container-ID-Size	Total size of the tag in bytes	4	unsigned int

9.6.2 General Tag

The general tag contains informaton like encoder values.

Byte Offset	Tag Data	Description	Size in bytes	Type
0	General-ID	Content: 0x021A0101	4	unsigned int
4	Size	Total size of the tag in bytes	4	unsigned int
8	PicCnt	Picture counter (always +1)	2	unsigned short
10	BaseTimeCnt	Internal FPGA counter in μ s	4	unsigned int
14	Encoder HTL	Current HTL encoder value	4	unsigned int
18	SavedEncoderHTL	Stored HTL encoder value using reset encoder command	4	unsigned int
22	Encoder RS422	Current RS422 encoder value	4	unsigned int
26	SavedEncoderRS422	Stored RS422 encoder value using reset encoder command	4	unsigned int
30	USRIO1+USRIO2	Current state of digital I/O 1 and 2 Bit0: Load Bit1: Mirrored value Bit2: Reserved Bit3: Reserved	1	unsigned char
31	USRIO3+USRIO4	Current state of digital I/O 3 and 4 Bit0: Load Bit1: Mirrored value Bit2: Reserved Bit3: Reserved	1	unsigned char
32	Status Register	Bit0: Ready OK Bit1: Reserved Bit2: Reserved Bit3: Line numbers OK Bit4: Reserved Bit5: Overtrigger bit, triggering too fast Bit6: Reserved	2	unsigned short

34	Differential Inputs (Encoder422)	Signal TTL encoder inputs Bit0: ChA, Bit1: ChB, Bit2: ChC	1	unsigned char
35	Intensity-Peak1	Mean intensity of current profile, first peak	2	unsigned short
37	Intensity-Peak2	Mean intensity of current profile, second peak	2	unsigned short
39	ValidPoints-Peak1	Number of valid points in current profile, first peak	2	unsigned short
41	ValidPoints-Peak2	Number of valid points in current profile, second peak	2	unsigned short
43	Counter from input signal	Current counter of a user defined I/O (must be activated). Use SetEA1InputFunctionCounter... SetEA4InputFunctionCounter	4	unsigned int
47	CurrentExpTime	Current exposure time in μ s	3	unsigned char[3]
50	OPT30013	Bit0: Reserved Bit1: Blinking mode Bit2: Measurement mode Bit3: Profile enable status Bit4: Dynamic trigger status Bit5: Profile points detection status Bit6: Red laser status Bit7: Blinking mode profiles sending status	1	unsigned char
51	Reserved		1	unsigned char

9.6.3 Statistic Tag

The statistic tag contains sensor information like temperature.

Byte Offset	Tag Data	Description	Size in bytes	Type
0	Statistic-ID	0x021A0102	4	unsigned int
4	Statistic-Data-Size	Total size of the tag in bytes	4	unsigned int
8	Voltage1	Input voltage in Volt	2	unsigned short
10	Reserved	Reserved	2	unsigned short
12	CPU-FiFo	FiFo status CPU in bytes	4	unsigned int
16	FPGA-FiFo	FiFo status FPGA in bytes	4	unsigned int
20	Reserved	Reserved	6	void
26	OnOffCounter-CPU	Counter switching on sensor	2	unsigned short
28	OnTimeCounter-CPU	Operation timer in 1/4 seconds	4	unsigned int
32	Temperature-CPU	Temperature in grad Celsius of CPU	1	signed char
33	Reserved	Reserved	2	void
35	Temperature-Laser	Temperature in grad Celsius of laser	1	signed char

36	LaserPower	PWM-Signal, only for MSL2x7x and MLWLx7x	2	unsigned short
38	Mac address	Mac address	6	unsigned char[6]
44	Frequency: camera	In Hz	2	unsigned short
46	Bandwith: Ethernet	In *10 kBytes	2	unsigned short
48	Reserved	Reserved	5	void
53	User-Data	Can be set by command SetStatisticDataUserData=xxx	2	unsigned char[2]
55	Reserved	Reserved	1	void
56	Reserved	Reserved	4	unsigned char[4]

9.6.4 Description Tag

The description tag contains the XML description of the sensor settings.

Byte Offset	Tag Data	Description	Size in bytes	Type
0	Description-ID	0x021A0103	4	unsigned int
4	Description-Size	Total size of the tag in bytes	4	unsigned int
8	Description Data (xml)	Sensor data in XML format	Variable	unsigned char [size-8]

9.6.5 ROI-X Tag

The ROI-X Tag contains information about the ROI settings in X.

Byte Offset	Tag Data	Description	Size in bytes	Type
0	ROI-X ID	0x021A0201	4	unsigned int
4	ROI-X Size	Total size of the tag in bytes	4	unsigned int
8	X-Number	Number n of ROI in X	2	unsigned short
10	ROI-X Details	Definition of n ROI in X	6*n	ROI_XDetail[n]
10+6*n	Reserved	Reserved	2	unsigned short

9.6.6 ROI-Z Tag

The ROI-Z Tag contains information about the ROI settings in Z.

Byte Offset	Tag Data	Description	Size in bytes	Type
0	ROI-Z ID	0x021A0202	4	unsigned int
4	ROI-Z Size	Total size of the tag in bytes	4	unsigned int
8	Z-Number	Number n of ROI in Z	2	unsigned short
10	ROI-Z Details	Definition of n ROI in Z	6*n	ROI_ZDetail[n]
10+6*n	Reserved	Reserved	2	unsigned short

9.6.7 RegisterCameraMLSL

(only for MLSL sensor)

Byte Offset	Tag Data	Description	Size in bytes	Type
0	RegisterCameraMLSL	0x021A0301	4	unsigned int
4	Size	Total size of the tag in bytes	4	unsigned int
8	Reserved	Reserved	1024	unsigned char[1024]

9.6.8 RegisterCameraMLWL

(only for MLWL sensor)

Byte Offset	Tag Data	Description	Size in bytes	Type
0	RegisterCameraMLWL	0x021A0302	4	unsigned int
4	Size	Total size of the tag in bytes	4	unsigned int
8	Reserved	Reserved	128	unsigned char[128]

9.6.9 Register FPGAMLSL

(only for MLSL sensors)

Byte Offset	Tag Data	Description	Size in bytes	Type
0	RegisterFPGAMLSL	0x021A0401	4	unsigned int
4	Size	Total size of the tag in bytes	4	unsigned int
8	Reserved	Reserved	292	unsigned char[292]



NOTE!
Size may be changed in the case of firmware updates.

9.6.10 Register FPGAMLWL

(only for MLWL sensors)

Byte Offset	Tag Data	Description	Size in bytes	Type
0	RegisterFPGAMLWL	0x021A0402	4	unsigned int
4	Size	Total size of the tag in bytes	4	unsigned int
8	Reserved	Reserved	Size-8	unsigned char[Size-8]



NOTE!
Size may be changed in the case of firmware updates.

9.6.11 Linearization Table

The linearization table contains information used by the DLL which is provided by the SDK. The content is not documented.

Byte Offset	Tag Data	Description	Size in bytes	Type
0	Linearization tag	0x1907	2	unsigned short
4	Size	Total size of the tag in bytes	4	unsigned int
8	Data	Not documented	Size-10	void
Size-4	CRC	Checksum over all data without last 4 bytes	4	unsigned int

9.6.12 ScanNonLinear

Reserved in case that the data are not processed inside of the sensor.

Byte Offset	Tag Data	Description	Size in bytes	Type
0	Scan	0x021A0601	4	unsigned int
4	Size	Total size of the tag in bytes	4	unsigned int
8	Reserved	Reserved	Size-8	void

9.6.13 ScanLinear

Contains the data and information of the measured profile.

Byte Offset	Tag	Description	Size in bytes	Type	Min. occurrence	Max. occurrence
0	0x021A0602	ScanDataLinear	4	unsigned int	1	1
4	0x00000001	SubID-ScanDataLinearHeader	4	unsigned int	1	1
8	0x00000002	SubID-ScanDataLinearData	4	unsigned int	1	1

Byte Offset	Tag Data	Description	Size in bytes	Type
0	ScanLinear	0x021A0602	4	unsigned int
4	Size	Total size of the tag in bytes	4	unsigned int

9.6.14 SubID-ScanLinearHeader

Contains information how the data are formatted

Byte Offset	Tag Data	Description	Size in bytes	Type
0	SubID-ScanDataLinearHeader	0x00000001	4	unsigned int
4	ScanDataLinearHeader-Size	Total size of the tag in bytes	4	unsigned int
ScanDataLinearHeaderData				
8	NumberOfPoints	MLSL:1280 MLWL: 2048	4	unsigned int
12	NumberOfPeaks	1 or 2	1	unsigned char
13	NumberOfElementsPerPoint	Max 4	1	unsigned char
14	<u>HDR:</u> 0 = ExpTime1 1 = ExpTime2		1	unsigned char
15	Reserved	Reserved	5	unsigned char[5]
Element 1 of 4				
20	<u>ID-Name[0]:</u> 0 = Dummy 1 = X 2 = Z 3 = Y 4 = I 5 = Peak width (PW)	2 = Z	1	unsigned char
21	<u>Type:</u> 0 = unsigned int 1 = float	0	1	unsigned char
22	Size in bits	16	1	unsigned char
23	Reserved	Reserved	1	unsigned char
Element 2 of 4				
24	<u>ID-Name[0]:</u> 0 = Dummy 1 = X 2 = Z 3 = Y 4 = I (Bit7-0: Int-Bit 10-2) 5 = Peak width (PW)	4 = I	1	unsigned char
25	<u>Type:</u> 0 = unsigned int		1	unsigned char
26	Size in bits	10	1	unsigned char
27	Reserved	Reserved	1	unsigned char

Byte Offset	Tag Data	Description	Size in bytes	Type
Element 3 of 4				
28	<u>ID-Name[0]:</u> 0 = Dummy 1 = X 2 = Z 3 = Y 4 = I 5 = Peak width (PW) 5 = I-Low + PW(Bit7-6: Int-Low-Bit1-0, Bit5-0: PW-Bit5..0)	5 = PW	1	unsigned char
29	<u>Type:</u> 0 = insigned int		1	unsigned char
30	Size in bits	6	1	unsigned char
31	Reserved	Reserved	1	unsigned char

Element 4 of 4

32	<u>ID-Name[0]:</u> 0 = Dummy 1 = X 2 = Z 3 = Y 4 = I 5 = Peak width (PW) 5 = I-height (Int-height-Bit7-2)	1 = X	1	unsigned char
33	<u>Type:</u> 0 = unsigned int		1	unsigned char
34	Size in bits	16	1	unsigned char
35	Reserved	Reserved	1	unsigned char
36	Reserved	Reserved	4	unsigned char[4]

9.6.15 SubID-ScanLinearData

Contains the data.

Byte Offset	Tag Data	Description	Size in bytes	Type
0	SubID-ScanDataLinearData	0x00000002	4	unsigned int
4	ScanDataLinearData-Size	Total size of the tag in bytes	4	unsigned int
8	Z, I, PW, X	Structure as defined in SubID-ScanLinearHeader section 9.6.14	6*1280 for MLSL 6*2048 for MLWL	unsigned short[1280][3] for MLSL unsigned short[2048][3] for MLWL

9.6.16 ScaleParam

Contains the information how the dat must be scaled to convert it into mm dimension

Byte Offset	Tag Data	Description	Size in bytes	Type
0	ScaleParam	0x021A0801	4	unsigned int
4	Size	Total size of the tag in bytes	4	unsigned int
8	X-Scale	Scaling factor X in mm	4	float
12	X-Offset	Offset X in mm	4	float
16	Z-Scale	Scaling factor Z in mm	4	float
20	Z-Offset	Offset Z in mm	4	float

X value [mm] = X-Scale*integer value x + X-Offset

Z value [mm] = Z-Scale*integer value z + Z-Offset

9.6.17 CRC

Tag for checksum

Byte Offset	Tag Data	Description	Size in bytes	Type
0	CRC	0x021AFFFF	4	unsigned int
4	Size	Total size of the tag in bytes	4	unsigned int
8	Dummy data	Total size container must be modulo 64 bytes		unsigned int[Description-ID-Size-12]
Size-4	CRC-Sum	Check sum container without 4 last bytes (32 bit CRC Polynomial 0x04C11DB7)	4	unsigned int

9.7 Typical Data Sets

After connection to the sensor following data are typically transmitted by default.

1. The so called linearization table which is not used by the user and can be ignored.
2. The XML description of the sensor settings. The description is in plain terms formatted as XML.
3. The measurement data after the setup is done, s. section 2.

9.7.1 Overview Typical Data Stream MSL

Tag	Tag ID
After open socket communication	
Linearization table	
Linearization table	0x0719
Description sensor settings in XML	
Container	0x021a01ff
Description	0x021a0103
CRC	0x021AFFFF
After command SetAcquisitionStart	
Measurement data	
Container	0x021A01FF
ROI-X	0x021A0201
ROI-Z	0x021A0202
General	0x021A0101
Statistic	0x021A0102
ScaleParam	0x021A0801
ScanLinear	0x021A0602
RegisterFPGAMSL	0x021A0401
RegisterCameraMSL	0x021A0301
CRC	0x021AFFFF
Each new measurements generate a new container	

9.7.2 Overview Typical Data Stream MLWL

Tag	Tag ID
After open socket communication	
Linearization table	
Linearization table	0x0719
Description sensor settings in XML	
Container	0x021a01ff
Description	0x021a0103
CRC	0x021AFFFF
After command SetAcquisitionStart	
Measurement data	
Container-ID	0x021A01FF
General-ID	0x021A0101
Statistic-ID	0x021A0102
ID-RegisterCameraMLWL	0x021A0302
ID-RegisterFPGAMLWL	0x021A0402
ID-ROI-X	0x021A0201
ID-ROI-Z	0x021A0202
ScaleParam	0x021A0801
ScanLinear	0x021A0602
CRC	0x021AFFFF
Each new measurements generate a new container	

9.7.3 Example First Data After Connection

Linearization table:

Tag	Tag size in bytes	Offset in bytes	0	1	2	3	4	5	6	7	Tag element	Content
0x1907	182880	0	07	19	60	ca	02	00	4c	49	Linearization table	
		snipped data	22	7d	7d	00	91	8f	38	57	Size	182880
											Data	not documented
											CRC	1463324561

XML description:

Tag	Tag size in bytes	Offset in bytes	0	1	2	3	4	5	6	7	Tag element	Content
0x021a01ff	41388	0	ff	01	1a	02	ac	a1	00	00	Container	
											Size	41388
0x021a0103	41368	8	03	01	1a	02	98	a1	00	00	XML description	
			3c	3f	78	6d	6c	20	76	65	Size	41368
			72	73	69	6f	6e	3d	22	31	Content	<?xml version="1.0" encoding="UTF-8" ?><device><g...ice>
		snipped XML data	2e	30	22	20	65	6e	63	6f		
			69	63	65	3e	0d	0a	00	00		
0x021AFFFF	12	41376	ff	ff	1a	02	0c	00	00	00	CRC	
			06	3f	d6	ff					Size	12
											Dummy data	Dummy data to increase total container byte size to a value which is modulo 64 bytes (9280 bytes modulo 64 bytes =0).
											CRC-Sum (32 bit CRC Polynom 0x04C11DB7)	4292230918

9.7.4 Example MLSL Container

Tag	Tag size in bytes	Offset in bytes	0	1	2	3	4	5	6	7	Tag element	Content
0x021a01ff	9280		ff	01	1a	02	40	24	00	00	Container	
											Size	9280
0x021a0201	16	8	01	02	1a	02	10	00	00	00	ROI-X ID	
			01	00	00	00	00	05	00	00	Size	16
											X-Number	1
											ROI-X Details	0;1024;0
0x021a0202	16	24	02	02	1a	02	10	00	00	00	ROI-Z ID	
			01	00	00	00	00	04	00	00	Size	16
											X-Number	1
											ROI-X Details	0;1024;0

Tag	Tag size in bytes	Offset in bytes	0	1	2	3	4	5	6	7	Tag element	Content
0x021a0101	52	40	01 01 1a 02 34 00 00 00	06 38 6b 4d 22 e0 01 00	General							
			00 00 00 00 00 00 01 00		Size		52					
			00 00 00 00 00 00 88 84		PicCnt		14342					
			5f 00 07 c4 03 00 00 00		BaseTimeCnt		3760344427					
			05 00 00 00 00 00 00 96		EncoderHTL		1					
			00 00 00 00		SavedEncoderHTL							
					EncoderRS422		1					
					SavedEncoderRS422							
					USRIO1+USRIO- O2(Bit3:in,Bit2:oe,Bit1:in- n,Bit0:sk)							
					USRIO3+USRIO- O4(Bit3:in,Bit2:oe,Bit1:in- n,Bit0:sk)							
					M2GL-Status: Register 128		Bit 7 =1					
					Differential Inputs(Encoder422)		7					
					Bit0: ChA, Bit1: ChB, Bit2: ChC							
					Intensity-Peak1		964					
					Intensity-Peak2							
					ValidPoints-Peak1		1280					
					ValidPoints-Peak2							
					Counter from Input Signal							
					CurrentExpTime		150					
					Reserved							
0x021a0102	60	92	02 01 1a 02 3c 00 00 00	47 05 0e 00 02 00 00 00	Statistic							
			18 21 00 00 5f 00 84 ad		Statistic-Data-Size		60					
			49 00 5f 00 84 ad 49 00		Voltage1		1351					
			38 38 38 38 ff 00 54 4a		Reserved							
			05 0a 08 04 c7 00 01 07		CPU-FiFo		2					
			01 07 00 00 00 00 00 00		FPGA-FiFo		8472					
			00 00 00 00		Reserved							
					OnOffCounter-CPU		95					
					OnTimeCounter-CPU		4828548*1/4 [s]=1207137 s					
					Temperatur-CPU		56					
					Reserved							
					Temperatur-Laser		56					
					LaserPower		255					
					mac address		04:08:0a:05:4a:54					
					Frequency: Camera		199					
					Bandwidth: Eth		1793*10 kB=17930 kB					
					Reserved							
					User-Data		0x0000					
					Reserved							
					Reserved							
0x021a0801	24	152	01 08 1a 02 18 00 00 00	7c 85 79 3a 6e 56 ef c1	ScaleParam							
			2b ed 85 3a b5 ff 79 42		Description-ID-Size		24					
					X-Scale		0,00095185					
					X-Offset							
					Z-Scale		0,00102178					
					Z-Offset		62,4997139					
0x021a0602	7736	176	02 06 1a 02 38 1e 00 00	01 00 00 00 20 00 00 00	ScanLinear							
			00 05 00 00 01 04 00 00		ScanLinear-ID -Size		7736					
			00 00 00 00 02 00 10 00		SubID-ScanDataLinearHeader		0x00000001					
					ScanDataLinearHeader-Size		32					

Tag	Tag size in bytes	Offset in bytes	0	1	2	3	4	5	6	7	Tag element	Content
			04	00	0a	00	05	00	6	00	ScanDataLinearHeaderData:	
			01	00	10	00	00	00	00	00	NumberOfPoints	1280
			02	00	00	00	08	1e	00	00	NumberOfPeaks	1
			cc	59	08	ce	87	19	d0	59	NumberOfElementsPerPoint	4
			c8	d2	ae	19	fa	59	c8	d5	HDR: 0=ExpTime1, 1=ExpTime2	
			d0	19	fc	59	c8	d6	f8	19	Reserved	5 bytes
			fb	59	c8	d9	21	1a	e7	59	Element: 1 from 4	
			c9	d1	4c	1a	d2	59	c8	da	ID-Name[0]: 0=Dummy, 1=X, 2=Z, 3=Y, 4=I, 5=Peak width (PW)	2=Z
			78	1a	d0	59	08	d7	a1	1a	Type: 0=unsigned int, 1=float	0=unsigned int
			d1	59	88	d5	c9	1a	d1	59	Size in bits	16
			08	d8	f1	1a	d3	59	c8	db	Reserved	
			19	1b	d6	59	c8	dc	41	1b	Element: 2 from 4	
			ec	59	c9	d0	66	1b	ee	59	ID-Name[0]: 0=Dummy, 1=X, 2=Z, 3=Y, 4=I, 5=Peak width (PW)	4=I
			c9	cf	8e	1b	ee	59	89	d3	4=I (Bit7-0: Int-Bit10-2)	
		Snipped data	b6	1b	ed	59	c9	d6	df	1b	Type: 0=unsigned int	0=unsigned int
			eb	59	49	d7	08	1c	e9	59	Size in bits	10
			09	d5	30	1c	da	59	c8	da	Reserved	
			5b	1c	ee	59	89	cf	80	1c	Element: 3 from 4	
			f2	59	49	d3	a8	1c	f1	59	ID-Name[0]: 0=Dummy, 1=X, 2=Z, 3=Y, 4=I, 5=Peak width (PW)	5
			49	d0	d1	1c	db	59	48	d2	5=I-Low + PW(Bit7-6: Int-Low-Bit1-0, Bit5-0: PW-Bit5-0.0)	
			fc	1c	dc	59	08	d0	24	1d	Type: 0=unsigned int	0=unsigned int
			05	5a	48	d5	47	1d	0a	5a	Size in bits	6
			48	d8	6e	1d	0e	5a	c8	d9	Reserved	
			96	1d	14	5a	08	de	bd	1d	Element: 4 from 4	
			17	5a	88	e0	e5	1d	2b	5a	ID-Name[0]: 0=Dummy, 1=X, 2=Z, 3=Y, 4=I, 5=Peak width (PW)	1=X
			09	d5	0b	1e	2d	5a	c9	d2	Type: 0=unsigned int	0=unsigned int
			33	1e	2b	5a	89	d0	5b	1e	Size in bits	6
			10	5a	08	d6	88	1e	05	5a	Reserved	
			c8	ce	b2	1e	e1	59	08	cd	Reserved	
		snipped data	e0	1e	e0	59	48	ce	08	1f	SubID-ScanDataLinearData	0x0000002
			e0	59	88	ce	31	1f	f2	59	ScanDataLinearData-Size	7688
			47	dd	56	1f	f6	59	87	dd		
			7e	1f	09	5a	08	d1	a4	1f	Z,I,PW,X	1280 data values (size 7680 bytes)
			0b	5a	08	d5	cc	1f	0b	5a		
			8a	e5	7b	61	09	dd	b4	e5		
0x021a0401	300	7912	01	04	1a	02	2c	01	00	00	RegisterFPGAMLSL	
			5f	80	88	13	00	00	00	00	Description-ID-Size	300
			00	00	00	96	00	00	00	00	Reserved	292 bytes reserved
			00	00	00	00						
0x021a0301	1032	8212	01	03	1a	02	08	04	00	00	RegisterCameraMLSL	
			d0	50	01	00	00	00	00	00	Description-ID-Size	1032
			00	00	00	00	00	00	00	00	Reserved	1024 bytes reserved
			01	cd	01	c9	00	c1	20	00		
0x021affff	36	9244	ff	ff	1a	02	24	00	00	00	Description-ID(xml)	
			00	00	00	00	00	00	00	00	Description-ID-Size	36

Tag	Tag size in bytes	Offset in bytes	0	1	2	3	4	5	6	7	Tag element	Content
			00	00	00	00	00	00	00	00	Dummy data	Dummy data to increase total container byte size to a value which is modulo 64 bytes (9280 bytes modulo 64 bytes =0).
			00	00	00	00	00	00	00	00	CRC-Sum (32 bit CRC Polynom 0x04C11DB7)	3655526239
			5f	e7	e2	d9						
Total size		9280										

9.7.5 Example MLWL Container

Tag	Tag size in bytes	Offset in bytes	0	1	2	3	4	5	6	7	Tag element	Content
0x021a01ff	12992		ff	01	1a	02	c0	32	00	00	Container Size	12992
0x021a0101	52	8	01	01	1a	02	34	00	00	00	General	
			b7	b8	21	7e	ca	3b	00	01	Size	52
			00	00	00	00	00	00	01	00	PicCnt	8632
			00	00	00	00	00	00	00	20	BaseTimeCnt	1271561761
			5f	00	02	36	02	00	00	00	EncoderHTL	256
			08	00	00	00	00	00	00	95	SavedEncoderHTL	0
			00	00	7c	00					EncoderRS422	1
											SavedEncoderRS422	0
											USRIO1+USRIO2 (Bit3:in, Bit2:oe, Bit1:inn, Bit0:sk)	
											USRIO3+USRIO4 (Bit3:in, Bit2:oe, Bit1:inn, Bit0:sk)	
											M2GL-Status: Register_128	
											Differential Inputs(Encoder422)	2
											Bit0: ChA, Bit1: ChB, Bit2: ChC	
											Intensity-Peak1	0
											Intensity-Peak2	8
											ValidPoints-Peak1	0
											ValidPoints-Peak2	0
											Counter from Input Signal	0
											CurrentExpTime	149
											OPT3013	
											Reserved	
0x021a0102	60	60	02	01	1a	02	3c	00	00	00	Statistic	
			d6	08	0c	00	3c	04	00	00	Statistic-Data-Size	60
			00	00	00	00	10	00	a7	49	Voltage1	2262
			25	01	10	00	d7	09	49	00	Reserved	
			2b	2e	00	2b	ff	00	54	4a	CPU-FiFo	1084
			05	0a	06	8c	64	00	00	00	FPGA-FiFo	
			01	02	00	00	00	00	00	01	Reserved	
			00	00	00	00					OnOffCounter-CPU	16
											OnTimeCounter-CPU	4786647*1/4 [s]=1207137 s
											Temperatur-CPU	43
											Reserved	

Tag	Tag size in bytes	Offset in bytes	0	1	2	3	4	5	6	7	Tag element	Content
											Temperatur-Laser	43
											LaserPower	255
											mac address	84:74:5:10:6:140
											Frequency: Camera	100
											Bandwidth: Eth	0*10 kB=0 kB
											Reserved	
											User-Data	0x0000
											Reserved	
											Reserved	
0x021a0302	136	120	02	03	1a	02	88	00	00	00	RegisterCameraMLWL	
		Snipped data	00	00	08	00	00	00	00	00	Size	136
			00	00	00	00	00	00	00	00	Reserved	
			00	00	00	62	00	43	53	05		
0x021a0402	320	256	02	04	1a	02	40	01	00	00	RegisterFPGAMLWL	
		Snipped data	5f	80	10	27	00	00	00	00	Size	320
			00	00	00	96	00	00	00	00	Reserved	
			00	00	00	00	00	00	00	00		
0x021a0201	16	576	01	02	1a	02	10	00	00	00	ROI-X ID	
			01	00	00	00	00	08	00	00	Size	16
											X-Number	1
											ROI-X Details	0;8;0
											Reserved	
0x021a0202	16	592	02	02	1a	02	10	00	00	00	ROI-Z-ID	
			01	00	00	00	00	08	00	00	Size	16
											Z-Number	1
											ROI-Z Details	0;8;0
											Reserved	
0x021a0801	24	608	01	08	1a	02	18	00	00	00	ScaleParam	
			c8	cc	e6	3a	c6	16	66	c2	Size	24
			c1	cd	07	3b	6d	c2	a3	42	X-Scale	0.00176086
											X-Offset	-57.5222
											Z-Scale	0.0020722
											Z-Offset	81.8797
0x021a0602	12344	632	02	06	1a	02	38	30	00	00	ScanLinear	
			01	00	00	00	28	00	00	00	ScanLinear-ID -Size	12344
			00	08	00	00	01	04	00	00	SubID-ScanDataLinearHeader	0x00000001
			00	00	00	00	02	00	10	00	ScanDataLinearHeader-Size	40
			04	00	0a	00	05	00	06	00	ScanDataLinearHeaderData:	
			01	00	10	00	00	00	00	00	NumberOfPoints	2048
			02	00	00	00	08	30	00	00	NumberOfPeaks	1

Tag	Tag size in bytes	Offset in bytes	0	1	2	3	4	5	6	7	Tag element	Content
		Snipped data	4e	77	c8	ae	84	db	5b	77	NumberOfElementsPerPoint	4
			c8	ae	72	db	62	77	c8	b0	HDR: 0=ExpTime1, 1=ExpTime2	
			5c	db	6c	77	c8	b2	48	db	Reserved	5 bytes
			77	77	c8	b1	34	db	7e	77	Element: 1 from 4	
			89	a7	1f	db	84	77	08	af	ID-Name[0]: 0=Dummy, 1=X, 2=Z, 3=Y, 4=I, 5=Peak width (PW)	2=Z
			0a	db	84	77	88	af	f3	da	Type: 0=unsigned int, 1=float	0=unsigned int
			8c	77	c8	b0	de	da	8f	77	Size in bits	16
			88	b2	c8	da	95	77	08	b3	Reserved	
			b2	da	9d	77	88	af	9e	da	Element: 2 from 4	
			aa	77	88	ae	8a	da	ac	77	ID-Name[0]: 0=Dummy, 1=X, 2=Z, 3=Y, 4=I, 5=Peak width (PW)	4=I
			c8	b1	74	da	b6	77	48	b5	4=I (Bit7-0: Int-Bit10-2)	
			60	da	b6	77	c8	b3	49	da	Type: 0=unsigned int	0=unsigned int
			bc	77	c8	af	34	da	ca	77	Size in bits	10
			07	b7	20	da	ca	77	47	b6	Reserved	
			0a	da	cd	77	07	b6	f4	d9	Element: 3 from 4	
			ce	77	c7	b6	dd	d9	d3	77	ID-Name[0]: 0=Dummy, 1=X, 2=Z, 3=Y, 4=I, 5=Peak width (PW)	5=PW
			88	ae	c8	d9	da	77	48	ad	5=I-Low + PW(Bit7-6: Int-Low-Bit1-0, Bit5-0: PW-Bit-5..0)	
			b2	d9	e1	77	08	ac	9e	d9	Type: 0=unsigned int	0=unsigned int
			ef	77	47	b5	8a	d9	f0	77	Size in bits	6
			07	b7	74	d9	fd	77	88	b0	Reserved	
			60	d9	fd	77	88	af	4a	d9	Element: 4 from 4	
			02	78	48	ad	34	d9	0e	78	ID-Name[0]: 0=Dummy, 1=X, 2=Z, 3=Y, 4=I, 5=Peak width (PW)	1=X
			c8	ad	20	d9	18	78	87	b8	Type: 0=unsigned int	0=unsigned int
			0c	d9	20	78	c8	ae	f6	d8	Size in bits	16
			26	78	08	ae	e2	d8	29	78	Reserved	
			c8	af	cc	d8	2c	78	88	ae	Reserved	
			b6	d8	30	78	c8	ac	9f	d8	SubID-ScanDataLinearData	0x0000002
			3c	78	88	ad	8c	d8	45	78	ScanDataLinearData-Size	12296
			88	ae	77	d8	45	78	c8	ae	Z,I,PW,X	2048 data values (size 12288 bytes)
			60	d8	4e	78	48	ad	4c	d8		
			4d	78	08	ad	34	d8	4f	78		
			b7	14	1f	b2	c8	89	97	14		
0x021affff	16	12976	ff	ff	1a	02	10	00	00	00	Description-ID(xml)	
			00	00	00	00	c5	d0	65	9b	Description-ID-Size	16
											Dummy data	Dummy data to increase total container byte size to a value which is modulo 64 bytes (9280 bytes modulo 64 bytes =0).
											CRC-Sum (32 bit CRC Polynom 0x04C11DB7)	2607141061
Total size		12992										

9.8 Implementation Recommendation

For easy implementation it is recommended to define a complex data type in the structure of the container. The bits of the container are copied to the complex data type. See the provided SDK example for details.

9.9 CRC Calculation

The CRC checksum can be calculated using following algorithm provided in code snippets.

Definitions in header file:

```
#define CRCPOLYNOMIAL 0x04C11DB7L

/*!
 * Function to calculate the CRC checksum of the container tag.
 * \param[in] crc_accum start value of CRC calculation
 * \param[in] *data_blk_ptr pointer to the data in the container tag
 * \param[in] data_blk_size size of the data set equals to container size - 4
 * \return value of the calculated checksum
 */
unsigned int CalculateCRC(unsigned int crc_accum, unsigned char *data_blk_ptr,
unsigned int data_blk_size);
```


Implementation of function:

```
unsigned int CalculateCRC(unsigned int crc_accum, unsigned char *data_blk_ptr,
unsigned int data_blk_size)
{
    register unsigned int i, j;
    unsigned int uiCRCTable[256];
    boolean bCRCTableInitialize = false;;

    if (data_blk_size > 10000000)
    {
        return 0;
    }

    if (bCRCTableInitialize == false)
    {
        bCRCTableInitialize = true;
        register unsigned short int i, j;
        register unsigned int crc_accum;

        for (i = 0; i<256; i++)
        {
            crc_accum = ((unsigned int)i << 24);
            for (j = 0; j < 8; j++)
            {
                if (crc_accum & 0x80000000L)
                    crc_accum = (crc_accum << 1) ^ CRCPOLYNOMIAL;
                else
                    crc_accum = (crc_accum << 1);
            }
            uiCRCTable[i] = crc_accum;
        }
    }

    for (j = 0; j<data_blk_size; j++)
    {
        i = ((int)(crc_accum >> 24) ^ *data_blk_ptr++) & 0xFF;
        crc_accum = (crc_accum << 8) ^ uiCRCTable[i];
    }
    return crc_accum;
}
```

Example usage:

```
/*!
 * ucBuffer is a pointer to the data of the container tag
 * uiBuffer is the size of the container tag
 */
unsigned int uiCalculatedCRC = CalculateCRC(-1, ucBuffer, uiBuffer - 4);
```

10. Appendix

10.1 GetInfo (XML mode)

The following XML data description shows a part of the data returned by the function EthernetScanner_GetInfo (through parameter 2) in the XML mode:

```
<?xml version="1.0" encoding="UTF-8"?>
<device>
  <general>
    <ordernumber>MLWL221</ordernumber>
    <productversion>1.40</productversion>
    <producer>wenglor sensoric GmbH</producer>
    <description>2D-/3D-profile sensors</description>
    <hardwareversion>
      <general>1.4.0</general>
    </hardwareversion>
    .
    .
    .
    <encoder_ttl_rs422>
      <current>0</current>
      <default>0</default>
      <command>SetEA4ResetCounterEncoderTTLRS422</command>
      <parameter>0</parameter>
      <parameter>1</parameter>
      <help>"0: disabled 1: enabled"</help>
    </encoder_ttl_rs422>
    <help>"dependency ea functionresetcounter XML-section"</help>
  </resetcounter>
</ea4>
</usrio>
</settings>
</device>
```

10.2 GetInfo (Text mode)

The following data description shows an example of the data returned by the function EthernetScanner_GetInfo (through parameter 2) in the text mode:

```
[general]
sn=6
z_start=65.000
z_range=60.000
x_range_at_start=40.000
x_range_at_end=58.000
widthX=1280
widthZ=1024
```