
# VC nano 3D Z Laser Scanner

## Manual Version 1.8.8

2. April 2019

Document name: VC 3D Z Laser Scanner Manual.pdf



*Vision Components GmbH*
*Ottostr. 2*
*76275 Ettlingen*
*Germany*
*Phone: +49 (0) 7243 2167-0*

WWW.VC-LINUX.COM
WWW.VISION-COMPONENTS.COM

**Foreword and Disclaimer**

This documentation has been prepared with most possible care. However Vision Components GmbH does not take any liability for possible errors. In the interest of progress, Vision Components GmbH reserves the right to perform technical changes without further notice.
Please notify **support@vision-components.com** if you become aware of any errors in this manual or if a certain topic requires more detailed documentation.
This manual is intended for information of Vision Component's customers only. Any publication of this document or parts thereof requires written permission by Vision Components GmbH.

**Trademarks**

Linux, the Tux logo, Vivado, Xilinx and Zynq, Windows XP, Total Commander, Tera Term, Motorola are registered Trademarks. All trademarks are the property of their respective owners.

# Table of Contents

# 1 Introduction

Vision Components introduces the VC nano 3D Z, an ultra-compact machine vision system that can be freely configured for 3D and 2D inspection tasks, providing a cost-efficient solution for a wide range of applications. Measuring merely 140 x 80 x 40 mm, its housing includes an intelligent camera and a line laser with up to 500 mW performance which enables the real-time recording of images at a scan rate of up to 2000 Hz according to the triangulation method. The images can be analyzed by the Dual-Core ARM processor and the integrated FPGA of the Smart Camera which has a computing power of with 2 x 866 MHz. The scan results can be transferred as attitude profile in mm or pixel as well as the grey image itself. Additional further calculations like line detection, gap detection etc. could be done inside the sensor.

## 1.1 General features

| **Technical Data** | **Specification** |
|---|---|
| Laser | Class 2, <130mW average, wave lenght 450nm |
| CMOS Sensor | 1/2.9" sony IMX273LLR, monochrome version |
| Active pixels | 1408(H) x 1080(V) |
| Pixel size | 3.45(H) x 3.45(V) µm |
| Active sensor size | 4.8(H) x 3.7(V) mm |
| High-speed shutter | 1 µsec + steps of 1 µsec |
| Integration | Global shutter |
| Data acquisition | program-controlled or external high speed trigger, jitterfree acquisition full-frame up to 174 frames per second |
| A/D conversion | 10 bit, 8 bits used after LUT application |
| Input LUT | 1024 x 8 bits |
| Processor | Dual-Core ARM® Cortex®-A9 with 866MHz and integrated FPGA |
| FPGA | Laser line processing during image acquisition in FPGA |
| RAM | 512 MB DDR-SDRAM |
| Flash EPROM | 16 GB flash memory (nonvolatile) industrial eMMC |
| Encoder Inputs | Yes, and encoder inputs can be used as additional 5-24V inputs. |
| Trigger Input | Encoder Signal A can be used as trigger input. |
| Ethernet interface | 1 Gbit / 100 Mbit / 10 Mbit |
| CE certification | CE Certification from Vision Components |
| Power Supply | 24V DC, 300mA without I/O usage. |
| Power Consumption | 7.2 W typical without I/O usage. |

# 2 Documentation

Please have a look at the following documentations:

General information about the sensor and the different model types:
**https://www.vision-components.com/de/produkte/oem/3d-systeme/arm-laser-triangulation/**

Hardware documentation of the sensor:
**https://www.vision-components.com/fileadmin/external/documentation/hardware/VC_nano_3D_Z/index.html**

Quick start documentation about how to work with VC Linux Cameras:
**https://www.vision-components.com/en/service-support/download-center/documentation-arm-linux/**
Please have a look at: **"Manual for VC Z Smart Camera set up introduction"**
- Kapitel 1.2. Video Tutorial
- Kapitel 2. Camera Access
- Appendix E: Recovering a camera from bad network
- Appendix C: Changing the IP address and DHCP

For special features of the sensor (at camera folder):
**"/usr/share/doc/libvclinux-doc/html/index.html"**
Please have a look at: **Modules → Platform Customisations → VC nano3D-Z**

For new version changes (at camera folder):
**"/usr/share/doc/libvclinux-doc/chlog"**

**Latest version:**

Latest version for camera server (Linux Z or TI) and Windows PC client (VC Smart Shape GUI):
**http://files.vision-components.com/nano3D/NewVersion/VCnano3D_SmartShape.zip**

**Support:**

If you need any help, please use our contact form:
**https://www.vision-components.com/en/contact/contact-form/**

# 3 Working with the VC 3D Z Laser Scanner Software

Vision Components offers two software packages (server, client) for an easy setup and hardware check:

## 3.1 SERVER - Laser Scanner Camera Program (vcnano3d_main_x.x.x)

The program includes the main function calls of the Laser Scanner and an easy data protocol for TCP/IP connection to the outer world. It shows, how to use the parallel image acquisition and processing and provides a menu to change all sensor and debug parameters.

## 3.2 CLIENT - Laser Scanner MS Windows Program (VC Smart Shape Vx.x)

The MS Windows PC program displays the results of the Laser Scanner camera. It allows an easy setting of the main scanner parameters. You can use the program to show the real 2D image from the sensor as well as the measured 3D profil. It works together with the standard TCP/IP protocol from the Laser Scanner demo program.

VC Smart Shape is just a simple demo program, which gives you a quick overview about the main functions of the sensor.

There is also a DLL / source code demo for the communication protocol available, if you like to build your own client. Please contact our support form for more information.

# 4 VC 3D Z Laser Scanner Camera Program

Typically VC delivers the VC 3D Laser Scanner with the latest software version of vcnano3d_main_x.x.x and starts the program via autoexec. This allows you to connect the camera directly to the client VC Smart Shape.

As the program starts directly via autoexec, there is no easy access to the program debug output or program menu.

If you like to have access to the program menu, please kill the running program and start it manually by following SHELL (SSH) commands:

| | |
|---|---|
| ps -e | (gives you the task number (PID) of vcnano3d_main_x.x.x) |
| kill [PID_NR] | (kill the process) |
| ./vcnano3d_main_x.x.x | (starts the program) |

Then you have full access to sensor menu and debug outputs.

If the VC 3D Laser Scanner doesn't have the latest version ready, you will find the software on our website. In the software package you will get the vcnano3d_main_x.x.x program as an executable file for the camera. Please upload the file (via WinSCP) to the camera and start it.

An Ethernet TCP/IP (10 / 100 / 1000 MBit/s) connection is used to communicate between the camera and the PC.

Files will be transferred over SCP or SFTP protocol via standard port 22.
Console access will be done via the SSH protocol also via standard port 22.

The initial and fallback network settings are as follows:

IP Address 192.168.003.015
Subnet mask 255.255.255.000

Due to these subnet mask settings, your PC must have at least one different IP address in the range of 192.168.3.[0–254]. Contact your system administrator for assistance.

The initial login settings are as follows:

User Name: root
Password: root

In order to start the program make sure, the file has exec rights:

```
chmod +x PrgName
```

The laser is more times hardware protected in order to fulfill the laser class 2 conditions. Before you can use the laser light, activate it with the command:

```
vcio -r 1
```

These steps can be done automatically during the boot process. Therefore please change the file

```
/etc/vcinit/user_init.sh
```

Example file user_init.sh:

```
#!/bin/bash
/usr/bin/vcio -r 360
cd /root && ./vcnano3d_main_x_x_x
```

Especially in multitask mode, it is sometimes necessary to use a higher priority. In this case start the program like:

```
chrt -f 98 ./PrgName -b 100
```

After starting the camera program file (`./PrgName`) the following messages appears:

```
Reading sensor parameter file VC3DPar.txt successful.
Product parameter file ProdPar.txt doesn't exist. Use default values.

New stream socket created at port 1096
New stream socket created at port 1097

VCImgNetSrv not running - no debug images available.

VCLINUX : 3.7.1
VCLIB   : 6.7.0
FASTLIB : 5.1.0
Cam SNR : 4090059
Created : Apr  2 2019 14:20:22

--- VC NANO 3D Z Version: Main=1.8.8 / Lib=0 / Product=57 ---

Press  'm'  to change sensor parameter
Press  'p'  to change product parameter
Press  'd'  to change debug parameter

RUN Mode. Ready for commands.
```

Customer settings can be stored on the flash memory as well. The software will start with the new parameter values. If the file "`VC3DPar.txt`" is not available, the demo file starts with the standard parameter settings. You can store your settings inside the menu 'm'. It is also possible to change the "VC3DPar.txt" file with a standard editor.

Additional for special product tasks like "angle detection", "gap detection" etc., a special setting file "ProdPar.txt" is maybe available.

The standard TCP/IP connection will be at port 1096. You can change the port number inside the source code. An additional port 1097 is available for debug purpose.

```
SERVER_PORT[0] =  1096;    // scanner communication port 0 (master)
SERVER_PORT[1] =  1097;    // scanner communication port 1 (slave)
```

The next lines reports the version number of all used libraries and the serial number (SNR) of the sensor. It also shows the compilation date and time of the demo file.

Next line shows the program version and a short menu.

## 4.1 Menu “m” (main menu for sensor parameters)

The menu gives you the possibility to change all sensor parameters. An other way to change the parameter is with the PC Windows Client. The parameters are described in the following chapters.

```
+-------------------------------------------+
|  nano3D_Z Sensor parameter                |
+-------------------------------------------+
>            MainVersion:             1.8.7 |
|              CameraSNR:           4090054 |
|               DataMode:     [            0]|
|      TransferAllPos_MM:     [            0]|
|                NbrLines:     [            0]|
|            ShutterTime:     [           10]|
|                GainVal:     [          100]|
|              LaserMode:     [            1]|
|              RlcThresh:     [           35]|
|            LaserSelect:     [            1]|
|               OptWidth:     [            8]|
|               MinWidth:     [            1]|
|               MaxWidth:     [           40]|
|        SpeckleFilterDx:     [            5]|
|        SpeckleFilterDy:     [            3]|
|        LaserMaskFilter:     [            0]|
|        ReflexionFilter:     [            0]|
|              ROI_X_PIX:     [            0]|
|              ROI_Y_PIX:     [            0]|
|             ROI_DX_PIX:     [         1408]|
|             ROI_DY_PIX:     [         1080]|
|            TriggerMode:     [            0]|
|         TriggerDelayNs:     [            0]|
|           ExposureMode:     [            0]|
|         AutoShutterVal:     [          100]|
|      AutoShutterFilter:     [            0]|
|         AutoShutterMin:     [            5]|
|         AutoShutterMax:     [          100]|
|         DoubleShutter1:     [            5]|
|         DoubleShutter2:     [          100]|
|         EthernetPackNr:     [            1]|
|         AutoTriggerFPS:     [    50.000000]|
|       AutoTriggerError:     [            0]|
|     EthernetSendNoWait:     [            1]|
|           MedianFilter:     [            3]|
|              SensorLUT:     [            0]|
|  Save sensor parameter:     [          GO!]|
|  Load sensor parameter:     [          GO!]|
```

```
|              Quit menu:     [          GO!]|
+-----------------------------------------+
```

Menu “p” (main menu for product parameter)
The menu ‘p’ gives you the possibility to change all product parameters.

```
+-----------------------------------------+
|  nano3D_Z Product parameter              |
+-----------------------------------------+
>         AdjustSensorRoi:          [      2]|
|         J00_ProductType:          [      1]|
|           J00_PolygonNr:          [      2]|
|      Show parameter J00:          [    GO!]|
|         J01_ProductType:          [      0]|
|           J01_PolygonNr:          [      0]|
|      Show parameter J01:          [    GO!]|
|         J02_ProductType:          [      0]|
|           J02_PolygonNr:          [      0]|
|      Show parameter J02:          [    GO!]|
|         J03_ProductType:          [      0]|
|           J03_PolygonNr:          [      0]|
|      Show parameter J03:          [    GO!]|
|  Save product parameter:          [    GO!]|
|  Load product parameter:          [    GO!]|
|               Quit menu:          [    GO!]|
+-----------------------------------------+
```

```
+-----------------------------------------------+
|  nano3D_Z Product parameter                    |
+-----------------------------------------------+
>          J00_TargetAngle:    [      0.000000]|
|          J00_AngleTolPos:    [      5.000000]|
|          J00_AngleTolNeg:    [      5.000000]|
|         J00_LineAlgoMode:    [             2]|
|       J00_BestLineFilter:    [            80]|
|      J00_HistoLineFilter:    [             1]|
|               J00_PhiMin:    [      0.000000]|
|               J00_PhiMax:    [    179.900009]|
|     J00_MinNrLaserPoints:    [             8]|
|  J00_PolygonPoint_X00_MM:    [     -2.998510]|
|  J00_PolygonPoint_Z00_MM:    [    143.000000]|
|  J00_PolygonPoint_X01_MM:    [    -20.998510]|
|  J00_PolygonPoint_Z01_MM:    [    185.000000]|
|   Save product parameter:    [           GO!]|
|   Load product parameter:    [           GO!]|
|               Quit menu:     [           GO!]|
+-----------------------------------------------+
```

```
|  Press '<-'/'->'/'Enter' to change Value.    |
|  Press 'd' to set Default Value.             |
+----------------------------------------------+
```

### 4.2 Menu “d” (debug parameter menu)

The menu “d” gives you additional information about the scanner and offers a fast way for debugging the software. You have the opportunity to find memory leaks, display the measure time or shutter time. You also can display the position of the laser line detection. In order to debug the TCP/IP host communication please select the necessary values.

```
+-------------------------------------------------------------------------------+
|  nano3D_Z Debug parameter                                                     |
+-------------------------------------------------------------------------------+
>          DebugPrint: 1=Mem 2=fps 3=Sh 4=ProdRes 5=ParSet 6=ROI 7=ATrig:    [  0]|
|                           DebugDraw: Bit0=LaserCenter Bit1=LaserBorder:    [  0]|
|  TCP/IP: Bit0=Host->Cam Bit1=Cam->Host Bit2=SendData Bit3=GlobalHeader:    [  0]|
|                                                 Save scan at scan.txt:    [GO!]|
|                                                          Quit program:    [GO!]|
|                                                             Quit menu:    [GO!]|
+-------------------------------------------------------------------------------+
|  Press '<-'/'->'/'Enter' to change Value.                                     |
|  Press 'd' to set Default Value.                                              |
+-------------------------------------------------------------------------------+
```

# 5 VC 3D Z Laser Scanner connection to Windows

We recommend to use our new “VC3D Smart Shape” PC program as a Host client.

3DSmartShape5.x works with the nano 3d Z laser scanner from Vision Components and the server program vcnano3d_main starting at version V1.7.0.

### 5.1 Camera Hardware Connection

A 100 MBit or 1 Gbit TCP/IP connection is necessary in order to communicate with the camera. You can directly connect the scanner camera to a TCP/IP Hub or just use a TCP/IP cross cable for a direct camera connection to your PC. A cross cable is available from Vision Components.

The scanner camera listen to the TCP/IP Port 1096. This port number can be changed in the camera source code. The standard IP address of the laser camera is 192.168.3.15. The IP address could be

different, if there is a different setting in the file “vcsetip.scr”. Additional settings for gateway or MASK are possible.

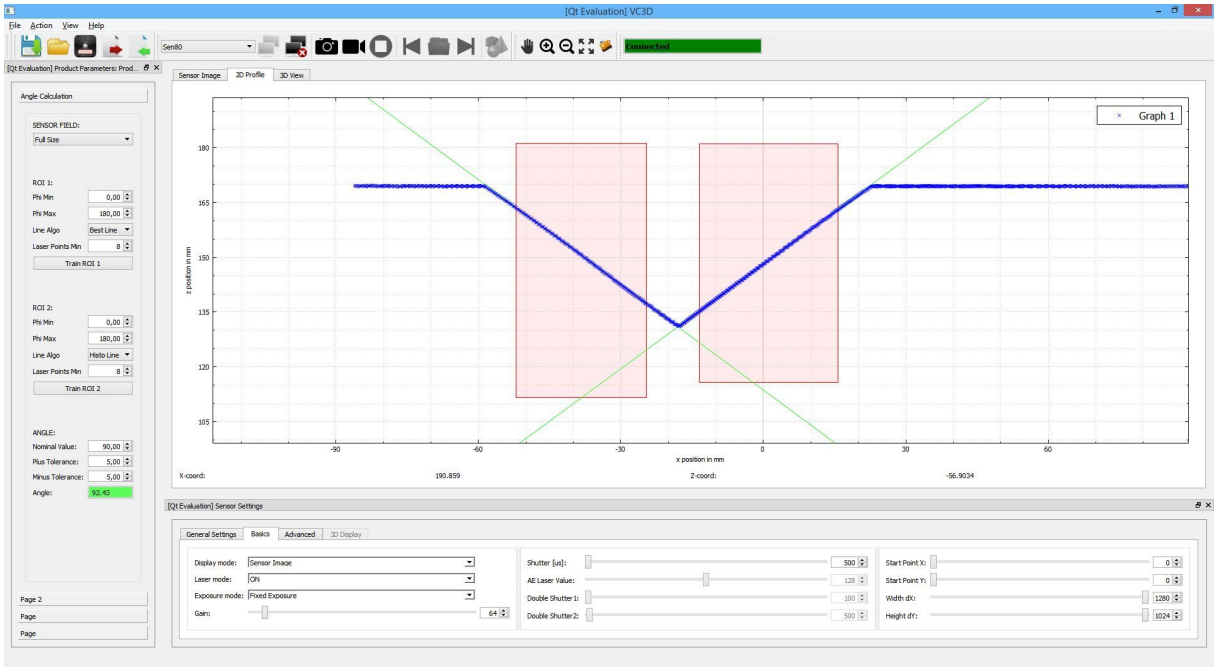
### 5.2 PC Connection

First you have to start the Laser Scanner camera software (vcnano3d_main) if the shell autoexec file is not available. Then start the VC 3D Z PC program and try to establish a TCP/IP connection to the camera. Choose the right IP and port address and press the Connect Button of the VC 3D Z PC application.

# 6 Windows program “VC 3D Z SmartShape”

The VC 3D Z PC SmartShape is a MS Windows PC program for displaying the results from the Laser Scanner camera. It allows an easy setting of the scanner parameters.

We recommend to use this new VC3D Smart Shape PC program for all VC 3D Z Laser Scanners. Please follow the program description in the documentation “VC 3D Z SmartShape Manual.pdf”.



# 7 Sensor Parameter

| Parameter | Effect |
|---|---|
| LaserMode | 1: set laser on<br>0: set laser off |
| ROI_X_PIX | Set the ROI x starting point |
| ROI_Y_PIX | Set the ROI y starting point |
| ROI_DX_PIX | Set the horizontal ROI width |
| ROI_DY_PIX | Set the vertical ROI width |
| GainVal | Set the gain |
| ShutterTime | Set the shutter [µs] |
| TriggerMode | 0: set the line trigger off (free running image acquisition)<br>+1: external hardware trigger or encoder<br>-1: Auto trigger mode (camera trigger, see AUTO_TRIG_PERIODE)<br>-2: Auto trigger mode (like -1, starting at external hardware trigger) |
| TriggerDelayNs | Delay time after hardware trigger (ns) |
| DataMode | -1: sensor active, no result transfer via TCP/IP<br>0: MODE_ STANDBY, no sensor activity<br>1: MODE_ TAKE_PIC<br>2: MODE_2D _POINTLIST<br>3: MODE_2D _POINTLIST_AND_LASER_IMG<br>4: MODE_2D _POINTLIST_MM (order x0 z0 / x1 z1 / x2 z2 / ...)<br>5: MODE_2D _POINTLIST_MM (order x0 x1 x2 ... / z0 z1 z2 ...)<br>6: MODE_BINARY_PIC<br>7: Not used any more<br>8: MODE_PRODUCT_RESULTS if sensor works in product mode<br>9: combination of Mode 4 and 8 (sends first Mode 4 than Mode 8) |
| TransferAllPos_MM | 0: transfer only valid laser detection points to the Client<br>1: transfer all positions to the Client (invalid positions will be NAN) |
| NbrLines | Set the number of scans that will be taken<br>If X=-1 there will be endless capturing until RESET has been sent<br>If X=0 there will be no scans and X=-2 will take the value from the other port |
| LaserSelect | Select the proper laser line method:<br>0 takes the laser line where the width is closest to OptWidth<br>1 takes the laser line which is defined by ReflexionFilter<br>2 takes the brightest laser line |
| OptWidth | Defines the optimal laser line width in pixel |
| ReflexionFilter | Laser Line Detection Filter Mode |

| | |
|---|---|
| | >0 : takes the n-th laser line from top (if more lines are in the image)<br><0 : takes the n-th laser line from bottom (if more lines are in the image) |
| MinWidth | Set min. Laser Line Width |
| MaxWidth | Set max. Laser Line Width |
| LaserMaskFilter | 0: no Filter<br>>0: mask filter to delete background |
| RlcThresh | Set threshold for laser line detection |
| ExposureMode | Set exposure mode:<br>0: FIX_EXPOSURE (use ShutterTime)<br>1: AUTO_EXPOSURE (overwrites ShutterTime, range [AutoShutterMin...AutoShutterMax])<br>2: DOUBLE_EXPOSURE (use DoubleShutter1 and 2) |
| AutoShutterVal | AUTO_EXPOSURE mode:<br>Select Shutter time for an average laser brightness of laser value |
| AutoShutterMin | AUTO_EXPOSURE mode:<br>Min. allowed Shutter time [µs] |
| AutoShutterMax | AUTO_EXPOSURE mode:<br>Max. allowed Shutter time [µs] |
| AutoShutterFilter | Auto shutter mode to calculate laser brightness based on:<br>0: all valid laser points<br>1: 50% of median valid laser points<br>2: 50% of brightest valid laser points<br>3: 50% of darkest valid laser points<br>4: 50% of centre valid laser points<br>5: 50% of left valid laser points<br>6: 50% of right valid laser points |
| DoubleShutter1 | DOUBLE_EXPOSURE mode:<br>Shutter time 1 [µs] – main image shutter |
| DoubleShutter2 | DOUBLE_EXPOSURE mode:<br>Shutter time 2 [µs] – second image, if no laser detection in main image |
| ROI_X_MM | Read the ROI x starting point [mm] |
| ROI_Y_MM | Read the ROI y starting point at y |
| ROI_DX_MM | Set the horizontal ROI width received at X |
| ROI_DY_MM | Set the vertical ROI width received at X |
| EthernetPackNr | Add n scan lines together in one Ethernet package.<br><br>0: write as much scan lines into camera memory and send the results at the end or if memory is full (about 11000 scans). This method is the fastest scan mode, as full processor power is available for scanning. But it will take more time at the end to transfer all |

| | |
|---|---|
| | data at once. Don't use this mode, if you scan permanently.<br>1: send scan line results directly after every scan (recommended)<br>n: put n scan lines together in one Ethernet package.<br>Recommended is max. of n=4 scan lines!<br><br>Please have a look at SEND_NOWAIT ! |
| AutoTriggerFPS | Sensor auto trigger mode in frames per second, 50.0 means 50 fps. See also AUTO_TRIG_ERR |
| AUTO_TRIG_ERR | nr scans which are not in time during Autotrigger (allowed jitter < MAX_JITTER_US, typ. 50µs)<br>>0 means, camera is not fast enough for selected scan speed, n lines are out of tolerance. Customer have to reset the value. |
| EthernetSendNoWait | 0: send will wait until all data are transferred<br>If host is not ready, it will lead to a delay of the next scan line<br>1: send as much data as possible (recommended)<br>If host is not ready, it will temporally fill camera memory. If memory is full, it stops scanning, until all data are transferred |
| SpeckleFilterDx<br>SpeckleFilterDy | Low pass filter to improve laser speckles noise (dx x dy) |
| MedianFilter | Median filter to improve laser detection errors (3x1, 5x1, 7x1 or 9x1) |
| SensorLUT | 10 to 8 Bit sensor LUT: 0: linear 1:sqrt 2:log |

## 7.1 Double Exposure

In double exposure mode the scanner takes two images. The first image is the main image and will be taken at the shutter time DoubleShutter[0]. The second image, with the shutter time DoubleShutter[1], will be taken directly after the first image. If a laser detection at any x position is not possible in the first image, it will try to find the laser in the second image.

Double Exposure mode reduce the scan rate!

The first image can be triggered by hardware or auto trigger as well as in free running mode. The second image will always be triggered directly after the first one. It doesn't wait for hardware or auto trigger.

# 8 Communication Protocol between Host and Sensor

Directly after establishing the TCP/IP connection the following communication is possible:

1. String Commands from Host (Client) to Sensor (Server)

2. Responses from Sensor (Server) to Host (Client)
   - Command Response after STRING Command
   - Result Response during RUN mode

## 8.1 Global Header

For an easy communication way, there is a fixed **GLOBAL HEADER** of 32 Bytes for every Client or Server communication. The size and structure is always the same.

| Size | Type | Description of Global Header | Value |
|---|---|---|---|
| 4 | U32 | Synchronisation Bytes | 0x10101010, 0x00001111, 0x01010101, 0x1111000<br><br>or<br><br>0xAA0F55F0 |
| 4 | U32 | Checksum (Global Header) | [0 .. 0x11111111] |
| 4 | U32 | Checksum (Data) | [0 .. 0x11111111] |
| 4 | U32 | Counter (0 = Reset Counter) | [0 .. 2^32-1] |
| 4 | I32 | Data waiting time in ms (0 = Infinite) | [0 .. 10000] |
| 4 | I32 | Reserved | |
| 4 | I32 | Error | [-2^32 .. 2^32-1] |
| 4 | U32 | Data Length N in Bytes (Without Global Header) | [0 .. 2^32-1] |
| | | ---------- END OF GLOBAL HEADER ---------- | |
| N | U8 | Data Bytes | [0 .. 2^8-1] |

The Global Header is the leading part of every command. The following byte stream with the size of Data Length includes the main information.

The synchronisation Bytes are a fixed value for an easy 'start of header' recognition.
The Global Header Checksum starts from Counter until the end of DataLength. The Data checksum revers only to the Data bytes. Checksum is an easy XOR combination of all bytes.

Counter is a global value, which allows a simply check for missing commands. Every part (Client or Server) increases their individual counter. The opposite part can verify this value and recognize any lack. A counter value of 0 resets the counter of the opposite part. It is recommended to reset the value after each connection. After reaching the max. value, the counter restarts at 0.

The data waiting time works as a time out and can limit the waiting time for receiving the data bytes.
Error can be used for fast error messages like TimeOut, Checksum Error etc.
Data Length defines the following data size in bytes.

## 8.2 Host (Client) Data Bytes

The “Command Type” defines the action mode:

0: STRING (ASCII) Command (allows one or more commands at once)

The “Parameter ID” is the value corresponding to the parameter list (see separate chapter).

### 8.2.1 Host “STRING” - Command

| Size | Type | Description of Meta Header | Value |
|---|---|---|---|
| 4 | U32 | Command Type | [0] |
| 4 | U32 | Parameter ID | [0 .. 2^32-1] |
| 4 | U32 | Host Command ID | [0 .. 2^32-1] |
| 4 | U32 | STRING Length N in Bytes (N=0, if no STRING Bytes) | [0 .. 2^32-1] |
| N | U8 | STRING Bytes | [0 .. 2^8-1] |

For valid “STRING” commands, please refer to the separate chapter. It is possible, to send one, more or all “STRING” parameters in one command. Every parameter setting must be terminated by Line Feed (0x0A) or NULL (0x00).

## 8.3 Sensor (Server) Data Bytes

The “Command Type” defines the action mode:

0: STRING (ASCII) command response

The “Parameter ID” is the value corresponding to the parameter list (see separate chapter). For every “Parameter ID” you will find the possible “Command Types” in the table.

The “Error” value provides the result of the parameter handling (see separate chapter).

### 8.3.1 Sensor “STRING” - Response

| Size | Type | Description of Meta Header | Value |
|---|---|---|---|
| 4 | U32 | Command Type | [0] |
| 4 | U32 | Parameter ID | [0 .. 2^32-1] |
| 4 | U32 | Host Command ID Reply | [0 .. 2^32-1] |
| 4 | I32 | Error | [-2^32 .. 2^32-1] |
| 4 | U32 | STRING Length N in Bytes (N=0, if no STRING Bytes) | [0 .. 2^32-1] |
| N | U8 | STRING Bytes | [0 .. 2^8-1] |

## 8.4 Sensor (Server) Result Response Data Bytes

### 8.4.1 Server “DATA” - Result Response during RUN mode

| Size | Type | Description of Meta Header | Value |
|---|---|---|---|
| 4 | U32 | Command Type | [100] |
| 4 | I32 | Data Mode | [1 .. 9] |
| 4 | U32 | Sensor Result Counter | [0 .. 2^32-1] |
| 8 | U64 | Image Time Stamp after Camera Boot [ms] | [0 .. 2^64-1] |
| N | U8 | Data | [-2^8 .. 2^8-1] |

The “Data” structure and size depends on the selected “Data Mode” (see separate chapter).

# 9 Parameter ID

Nearby all commands are usable as Read-Commands and Set-Commands. Some are only Read- or Set-Commands. Please refer to the list below. You will find Valid “command types” for every parameter in the table “CT” (0=STRING, 1=GET and 2=SET).

| **Parameter ID** | **ID** | **Effect** |
|---|---|---|
| RESET | 0 | Stop image acquisition and set NBR_LINES to 0 |
| LOAD_SENSOR_FILE | 42 | reads the product parameter from the camera file "fd:/VC3DPar.txt" |
| SAVE_SENSOR_FILE | 33 | Saves latest sensor settings in camera flash. Camera will start with this parameters |
| RECEIVE_SENSOR_DATA | 43 | Set sensor parameter Host ->  Cam from STRING. |
| SEND_SENSOR_DATA | 44 | Send sensor parameter Cam  -> Host as STRING. |
| LOAD_PRODUCT_FILE | 38 | reads the product parameters from the camera file "fd:/ProdPar.txt" |
| SAVE_PRODUCT_FILE | 45 | Saves latest product settings in camera flash. Camera will start with this parameters |
| RECEIVE_PRODUCT_DATA | 40 | Set product parameter Host ->  Cam from STRING. |
| SEND_PRODUCT_DATA | 41 | Send product parameter Cam  -> Host as STRING. |

# 10 STRING parameters

All valid PARAMETER STRINGs and sensor depending parameter ranges will be transferred with the commands SEND_SENSOR_DATA or SEND_PRODUCT_DATA.

File or Data example:

```
#
#  nano3D_Z Sensor parameter
#
#   Configuration file generated automatically on 2018-02-22 11:06:37
#

          DataMode = 0;
          NbrLines = 0;
       ShutterTime = 100;
           GainVal = 100;
         LaserMode = 0;
         RlcThresh = 24;
       LaserSelect = 0;
          OptWidth = 8;
          MinWidth = 1;
          MaxWidth = 40;
   SpeckleFilterDx = 5;
   SpeckleFilterDy = 3;
   LaserMaskFilter = 0;
   ReflexionFilter = 0;
         ROI_X_PIX = 0;
         ROI_Y_PIX = 0;
        ROI_DX_PIX = 1384;
        ROI_DY_PIX = 1080;
       TriggerMode = 0;
      ExposureMode = 0;
    AutoShutterVal = 100;
 AutoShutterFilter = 0;
    AutoShutterMin = 1;
    AutoShutterMax = 100;
    DoubleShutter1 = 5;
    DoubleShutter2 = 100;
    EthernetPackNr = 1;
    AutoTriggerFPS = 50.000000;
  AutoTriggerError = 0;
EthernetSendNoWait = 1;


# End of File
```

```
#  nano3D_Z Product parameter
#
#   Configuration file generated automatically on 2019-01-30 17:31:50
#

         ProductVersion = 57;
        AdjustSensorRoi = 2;
        J00_ProductType = 1;
          J00_PolygonNr = 2;
        J01_ProductType = 1;
          J01_PolygonNr = 2;
        J02_ProductType = 0;
          J02_PolygonNr = 0;
        J03_ProductType = 0;
          J03_PolygonNr = 0;
        J00_TargetAngle = 0.000000;
        J00_AngleTolPos = 5.000000;
        J00_AngleTolNeg = 5.000000;
       J00_LineAlgoMode = 2;
     J00_BestLineFilter = 80;
    J00_HistoLineFilter = 1;
              J00_PhiMin = 0.000000;
              J00_PhiMax = 180.000000;
   J00_MinNrLaserPoints = 8;
J00_PolygonPoint_X00_MM = -2.998510;
J00_PolygonPoint_Z00_MM = 143.000000;
J00_PolygonPoint_X01_MM = -20.998510;
J00_PolygonPoint_Z01_MM = 185.000000;
        J01_TargetAngle = 0.000000;
        J01_AngleTolPos = 5.000000;
        J01_AngleTolNeg = 5.000000;
       J01_LineAlgoMode = 2;
     J01_BestLineFilter = 80;
    J01_HistoLineFilter = 1;
              J01_PhiMin = 0.000000;
              J01_PhiMax = 180.000000;
   J01_MinNrLaserPoints = 8;
J01_PolygonPoint_X00_MM = 2.001490;
J01_PolygonPoint_Z00_MM = 143.000000;
J01_PolygonPoint_X01_MM = 22.001490;
J01_PolygonPoint_Z01_MM = 185.000000;


# End of File
```

# 11 Parameter Error List

| Error Name | Error Number | Effect |
|---|---|---|
| ERR_NONE | 0 | success |
| ERR_TCP_IP | -3001 | Wrong TCP/IP ParameterID |
| ERR_PARM | -3003 | Illegal parameter value or type<br>(out of range) |
| ERR_COMMAND | -3004 | Not a valid Parameter ID |
| ERR_CMD_STRG_LEN | -3010 | Bad string length at STRING command |
| ERR_CMD_TYPE | -3011 | Not a valid Command Type |
| ERR_CMD_DATA_LEN | -3012 | Command length does not fit to received bytes |
| ERR_MEM_PROD_ANGLE | -3018 | Could not allocate memory for angle calculation |
| ERR_PROD_INACTIV | -4000 | job was not executed |
| ERR_PROD_ROI_NR | -4001 | less than expected ROI points |
| ERR_PROD_ANGLE | -4002 | no line in selected angle range |
| ERR_PROD_LINE_NR | -4003 | less than expected points for line |
| ERR_PROD_GENERAL | -4004 | general product calculation error |
| ERR_PROD_RESULT | -4005 | couldn't calculate product result |

# 12 Measure Response (Data Mode)

Once the server has been initialized and in measure mode, it will send back to the client a result response that will contain the data depending on the selected MODE.

The Mode_ID is an enum value corresponding to the commands asked by the client, the counter has the same role as in the Command Header.

| Data Mode | Effect | Sub Header (Bytes) | Sub Header | Data | Data size |
|---|---|---|---|---|---|
| 1 | grey image | 8 | <width(4)><br><height(4)> | copy of the image memory in U8 | Width * height * sizeof (U8) |
| 2 | laser line profile (Sensor values) | 16 | <Data_Type(4)><br><Number_of_Point(4)><br><StartX(4)><br><dX(4)> | If Data_Type=0, data array coded in I16.<br>If Data_type=1, data array in floating point | <Number_of_Point> * sizeof ( I16 or float ) |
| 3 | laser line profile and laser brightness | 16 | <Data_Type(4)><br><Number_of_Point(4)><br><StartX(4)><br><dX(4)> | If Data_Type=0, data array coded in I16.<br>If Data_type=1, data array in floating point | <Number_of_Point> * sizeof ( I16 or float ) + <Number_of_Point> * sizeof(U8) |
| 4<br>5 | laser line profile (world coordinates in mm) | 4 | <Number_of_Point(4)> | 4: x0z0x1z1...<br>5: x0x1...z0z1... | <Number_of_Point> * 2 (x, y) * sizeof (float) |
| 6 | binary image | 8 | <width(4)><br><height(4)> | copy of the image memory in U8 | Width * height * sizeof (U8) |
| 7 | laser line profile, laser brightness and product results | 16 | <Data_Type(4)><br><Number_of_Point(4)><br><StartX(4)><br><dX(4)> | If Data_Type=0, data array coded in I16.<br>If Data_type=1, data array in floating point | <Number_of_Point> * sizeof ( I16 or float ) + <Number_of_Point> * sizeof(U8) |
| 8 | product results | 4 | <Data String Size(4)> | <1> AngleMeasure | Depending on product type different ASCII data sets will be send. |
| 9 | laser line profile in mm and product results | 8 | <Number_of_Point(4)<br><Data String Size(4)> | <1> AngleMeasure | <Number_of_Point> * 2 (x, y) * sizeof (float) + Depending on product type different ASCII data sets will be send. |

**Description of the main Modes:**

This chapter describes the states and the data that will be returned from the camera in the corresponding state.

After the camera has been set to a special state with the command CMD_MODE it may return data.

**Mode 1 ( MODE_TAKE_PIC )**:

Capture an image then send the array containing the header and the image.

.Size of the Sub Header: 8 Bytes
.Description of the Sub Header: <width (4 Bytes)> <height (4 Bytes) >
.Data: The data will be an exact copy of the image memory with size <width> *<height>, coded in U8.
.The number of images that will be returned depends on the previously sent command CMD_NBR_LINES. If this value was -1 the sending will be endless.
Image transfer always can be stopped if there is a command „CMD_RESET“ .

**Mode 2 (MODE_2D_POINTLIST)**:

Capture a frame and extract the laser line points in pixel coordinates, then send the array containing the header and the laser line points.

.Size of the Sub Header: 16 Bytes
.
.Description of the Sub Header:

.the data sent are I16 points (integer coded in 2 bytes):
.<Data_Type = 0 (4 Bytes)> <Number_of_Point(4 Bytes)> <StartX(4 Bytes)> <dX (4 Bytes)>

.the data are floating points (4 Byte float values) :
<Data_Type = 1 (4 Bytes)> <Number_of_Point(4 Bytes)> <StartX(4 Bytes)> <dX (4 Bytes)>

.Data: For each profile the Z-Position in pixels will be returned in either integer points or floating points. The floating values are always in little endian format.

.The number of frames that will be returned depends on the previously sent command CMD_NBR_LINES. If this value was -1 the sending will be endless.
Frame transfer always can be stopped if there is a command „CMD_RESET“ .

**Mode 3 (MODE_2D_POINTLIST_AND_LASER_IMG)**:

Capture a frame, extract the laser line points in pixel coordinates, then send the array containing the header and the laser line points. Additionally, send an array representing the laser line information : each value of the array will be the sum of each column of the laser line image.

.Size of the Sub Header: 16 Bytes

.Description of the Sub Header:

.For I16 data:
<Data_Type = 0 (4Bytes)> <Number_of_Point (4 Bytes)> <StartX (4 Bytes)> <dX (4Bytes)>

.For floating point data:
<Data_Type = 1 (4Bytes)> <Number_of_Point (4 Bytes)> <StartX (4 Bytes)> <dX (4 Bytes)>

.Data:
.For each profile the Y-Position in pixels will be returned in either short points or floating points followed by the laser line image information. The starting point and the shift in X can be used to go directly to some point in the data array.
.As the laser line image and the 2D PointList have the same number of points, the data size is : Number_of_Point *sizeof(type) + Number_of_Point * sizeof(U8))
.The array representing it will be sent in U8, the floating values are always in little endian format.

.The number of frames that will be returned depends on the previously sent command CMD_NBR_LINES. If this value was -1 the sending will be endless.
Frame transfer always can be stopped if there is a command „CMD_RESET“

**Mode 4 (MODE_2D_POINTLIST_MM)**: capture a frame and extract the laser line points in mm coordinates, then send the array containing the header and the laser line points.

.Size of the Sub Header: 8 Bytes
.
.Description of the Sub Header:

. <Number_of_Point(4 Bytes)> <LineCounter(4 Bytes)>

.the data are floating points (4 Byte float values) :
<x> <y>

.The number of frames that will be returned depends on the previously sent command CMD_NBR_LINES. If this value was -1 the sending will be endless.
Frame transfer always can be stopped if there is a command „CMD_RESET“ .

# 13 New Version Changes

For new version changes, please have a look at the camera folder:
**“/usr/share/doc/libvclinux-doc/chlog”**

A short history of the changes is documented in the head of the file vcnano3d_z_X.X.X.c.
Please have a look for more information.

Visit the Vision Components site **www.vision-components.com** for further information and documentation and software downloads:

| Web Site Menu Links | Content |
|---|---|
| Home | Latest News from VC |
| Competences | VC Company Information<br>VC Network |
| Solutions | Applications<br>Industry |
| **Products**<br>VC Smart Cameras | Product Overview: |
| | VC Base<br>VC Professional<br>VC Optimum<br>VC Line<br>Visicube<br>VC Board Cameras<br>VC Customized<br>Accessories |
| VC Smart Camera Software | |
| VC Software Development Kit Ti: | VCRT Operating System<br>VCLIB Image Processing Library |
| VC Special Libraries: | Color Lib<br>Extension Lib<br>VCOCR Text Recognition Library<br>VC Smart Reader<br>VC Smart Finder<br>VC Solar Solution |
| News and Events | VC News<br>Trade Show dates<br>VC Seminars & Workshops |
| Service & Support: | |
| Contact | Contact Vision Components |
| Download Center | Download of: |
| Documentation<br>(User Registration required) | - Product Brochures<br>- Camera Manuals<br>- Getting Started<br>- Programming Manuals<br>- Training Manuals and Demo Code |
| Software<br>(User- and SW License Registration required) | - Software Updates (VCRT & Libs)<br>- Demo Code<br>- Software utilities |
| Tech News | Tech News – new SW and Documentation |
| Knowledge Base / FAQ | FAQ Database with programming Examples and Demo Code |
| Return / Repair Service | Form for Allocation of Repair Numbers. |
| Loan systems | Info about VC loan cameras |