ConoProbe MKIII
OEM Manual

Version 1.00

OPTIMET MANUAL P/N 3J06007

Notice to Users:

Disclosure Restrictions:

Disclaimer:

Contact Information:

Europe and Israel:

Optimet, Optical Metrology Ltd.
Tel: +972-2-548-2900
+972-2-548-2916
+972-2-548-2903
Fax: +972-2-586-5387
P.O.B. 45021, Jerusalem 91450, Israel
support@optimet.com
http://www.optimet.com

USA and Canada:

Optimet, Optical Metrology Inc.
Tel: +1-978-657-6303
Fax: +1-978-657-6054
sales@optimet.com
http://www.optimet.com

Japan:

Optimet Division
Ophir Japan Ltd.
TEL: +81-48-646-4150
FAX: +81-48-646-4155
info@ophirjapan.co.jp
http://ophirjapan.co. jp

Table of Contents

1 Introduction

Thank you for purchasing the ConoProbe Mark III

This manual provides information for integrating the Hardware and Software of the ConoProbe Mark III into your system.

This chapter includes the following sections:

Product Overview – An overview of the ConoProbe Mark III

Contents - Descriptions of the various product components.

Safety Warnings – Safety warnings one should follow when handling and using the ConoProbe Mark III.

Compatibility – A list of the supported operating systems and development environments in which the ConoProbe Mark III OEM has been successfully integrated.

Physical Dimensions – Physical dimensions for the ConoProbe MKIII with various lenses.

Physical Dimensions **Box Description** – Description of the ConoProbe communication box.

1.1 Product Overview

The **ConoProbe Mark III** (**MKIII**) is the newest generation of single point non-contact optical sensors developed and manufactured by **Optimet**. The MKIII is a state of the art sensor, which is the result of over seven years of field experience and over 100 different OEM applications around the world.

In accordance with Optimet's line of non-contact sensors, the MKIII is based on our unique and patented Conoscopic Holography technology. The MKIII is designed for integration in a large variety of industrial applications such as: Quality Control, In-process inspection, and Reverse Engineering

The MKIII embedded is a "SOC" (system on a chip) platform and all data processing, including pre-programmed functions, is in the sensor head. The exceptional features of the MKIII include:

- Optimum measurements up to 3000 Hz with the MKIII extended frequency version.
- Low weight of 720 grams.
- Compact.
- Modular setup with interchangeable objective lenses enabling various standoffs and working ranges in the same sensor.
- Sub-micron precision with short focal length objectives.
- Simultaneous measurement on highly reflective and diffusive surfaces.
- Extensive angular coverage of over 170° width.
- Measurement of hard to measure geometries, steep grooves and angles.
- Integration capability with relay optics.
- The ConoProbe Mark III can act in a system as a Master synchronizer or as a Slave synchronized by the system

The ConoProbe Mark III integration software is designed to be backwards compatible with the ConoProbe Mark II software using DLL files. It is possible to have multiple sensors integrated and work in parallel, using standard Ethernet LAN communication. The MKIII also offers an optional analog output.

1.2 Contents

The ConoProbe Mark III package consists of the Mark III ConoProbe, the communication box, a crossed (twisted-pair) Ethernet cable, a D-15 to ODU cable connecting the MKIII to the communication box, and a 12VDC power supply powering the system.

The MKIII software package contains the interface files: Fifo.DLL, Fifo_PS.dll, and a set of supporting files. It also provides the following software utilities: a set of sample programs, the *SMART Scanner* for scanning objects, and the *Viewer* for viewing profiles of scan results.

1.2.1 ConoProbe Mark III

The ConoProbe MKIII is an optical non-contact "point" measurement sensor. Using Optimet's patented conoscopic holography technology the laser beam returned from the object surface is translated into a measured distance. This is done using an opto/mechanical base and implementation of mathematical algorithms in the electronics.

1.2.2 Communication Box

The communication box is the signal distribution hub for the MKIII and provides a connection between the MKIII and the host PC via an Ethernet 10/100 LAN link. It also has an external trigger input enabling measurements to be triggered externally, and a ROG (Read Out Gate) signal output for synchronizing systems where the MKIII is the master device.

1.2.3 Lenses

A variety of lenses is available to the customer according to his needs and requirements. Each lens is calibrated to a specific MKIII sensor in order to achieve optimal performance. The Serial Number on the MKIII should match the Serial Number on the Lenses.

1.2.4 Cables

The ConoProbe Mark III package contains the following cables:

- A 15-pin D-Type to ODU connector cable (for connecting the MKIII to the Communication box). Length: 2 meters.
- One Ethernet crossover cable. Length: 2 meters.

Power Supply

- An AC to DC power supply supplying 12VDC @ 0.5Amp.

1.2.5 OPTIMET OEM CD

The Optimet OEM CD contains important information about Optimet software package.

The CD Contains:

1. Help menu.
2. System requirements.
3. Installation.
4. List of CD content (README file)
5. How to contact Optimet.
6. SDK which contains sample code in C++, C# and Visual Basic

1.2.6 The FIFO DLL

The Fifo DLL is a 32-bit DLL. To allow your application to communicate with the MKIII, integrate this DLL into your application. For more information about integrating this DLL into your application, refer to the Optimet OEM CD and section 4.2 of this manual.

Note: The FIFO.dll enables backwards compatibility with applications written for the Mark 2.0 ConoProbe. (For a compatibility table see section 4.10). Note that is some situations, such as scanning with pulses, slight modifications must be made in software originally written for the Mark 2.0 to allow it to work correctly with the Mark 3.0. Please see section 4 and the sample code for more information.

Fifo_PS DLL is a 32-bit DLL, which extends the functionality of the Fifo DLL. Add the Fifo_PS DLL to the same program folder in which that you have installed the Fifo.DLL.

Please NOTE: The FIFO.DLL is not thread safe. One must therefore make all calls to this DLL from the same thread.

.

1.2.7 The SMART Probe Tester Program

The SMART Probe Tester Program allows you to experiment with the ConoProbe Mark III by changing its settings and by performing measurements without having to write your own program.

Using SMART Probe Tester, you can:

- Gain experience using the MKIII's controls through the SMART Probe Tester's GUI.
- Obtain the best operating parameters for your MKIII's measurements.
- Configure the measurement working frequency and laser power.
- Select a lens to use.
- Perform infinite and finite stream measurements as well as measurement acquisition to a buffer in a different thread.

The full source code for the Smart Probe Tester Program is in the SDK folder in the Smart Probe Tester sub folder.

1.2.8 The Viewer Program

The *Viewer* program allows you to view specimen profiles created with SMART Scanner. Using the *Viewer*, you can filter scan results to obtain optimal accuracy and export scan information to other formats or to other programs. In addition, the Viewer can perform an advanced analysis of the scan information. For more information about using the *Viewer*, refer to the **Viewer OEM manual**.

1.3 Safety Warnings

Exercise caution when using and servicing both the ConoProbe Mark III and the Communication-Box.

The MKIII emits laser radiation of less than 1mW per Class II specifications. Never stare directly into the beam.

The MKIII has no user serviceable components, and the laser radiation inside the sensor can reach 3mW. The MKIII is only to be opened by qualified Optimet service personnel.

" CAUTION- USE OF CONTROLS, ADJUSTMENTS OR PERFORMING PROCEDURES OTHER THAN THOSE SPECIFIED HEREIN MAY RESULT IN HAZARDOUS RADIATION EXPOSURE"

FDA Class II **IEC Class 2**

Figure 1-1 Safety Warnings

1.4 Compatibility

The ConoProbe Mark III OEM software is compatible with Windows™ 2000, Windows™ XP and Windows™ Vista. The software API can be used with any programming language that works with a 32-bit DLL (unmanaged). The Fifo DLL has been successfully integrated into systems written in the following development environments (among others): Microsoft .NET C#, VB.NET, Visual C++ 6.0, Visual Basic 6, Borland C++ Builder 5.0, and Delphi 5.0.

1.5 Physical Dimensions

NOTE:

Optimet reserves the right to make changes to its products without notice, and advises all customers to contact Optimet in order to obtain the latest information.

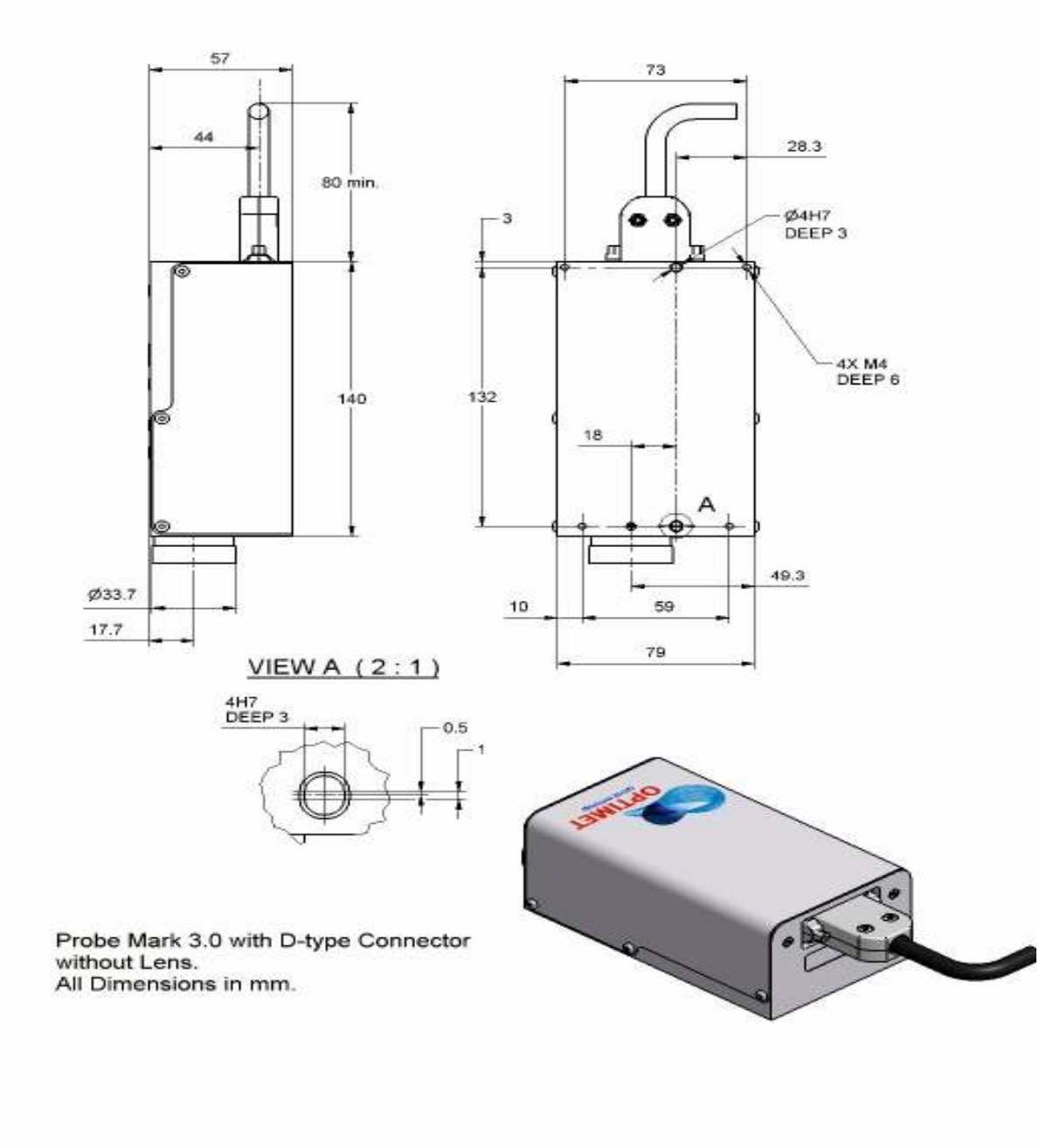
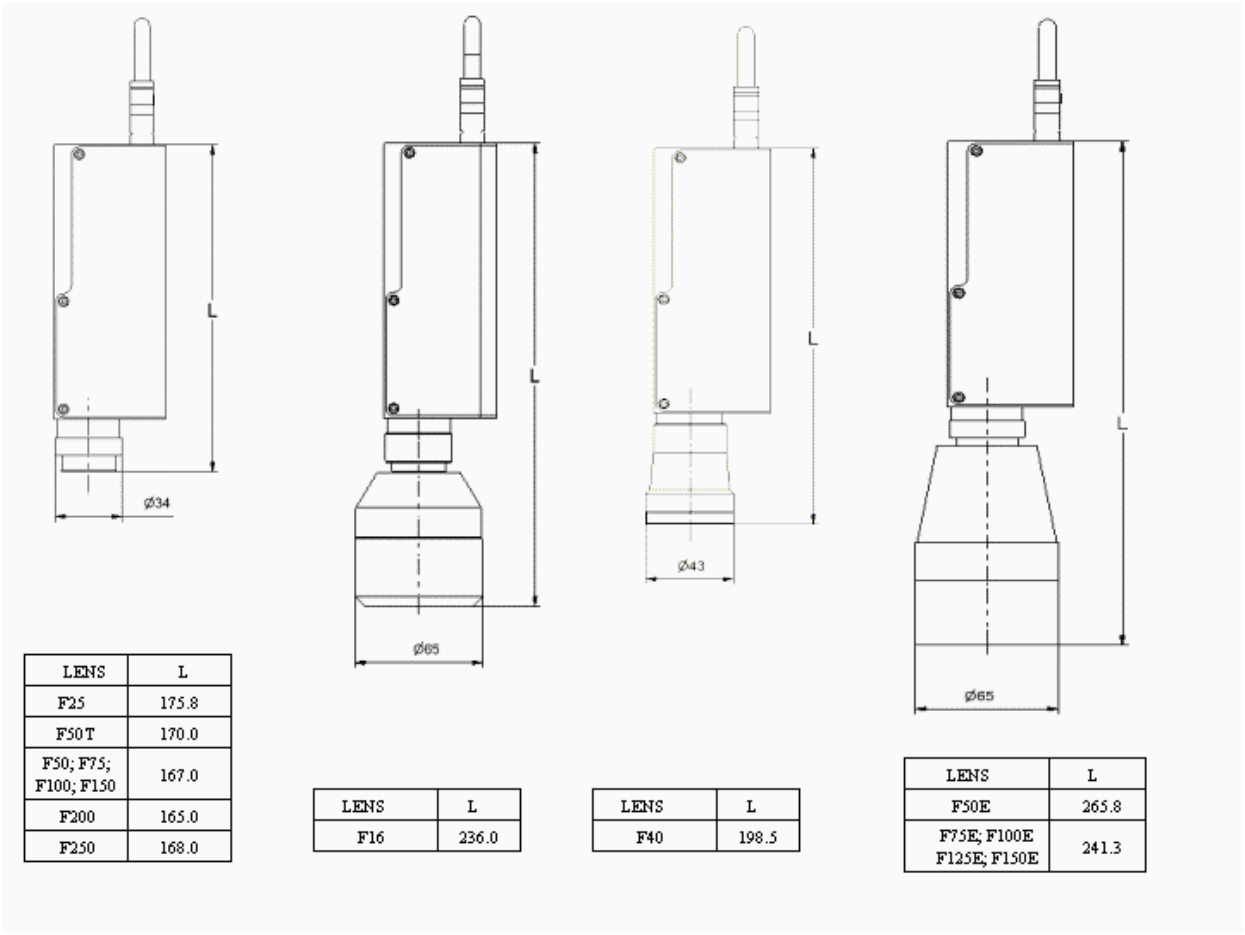


Figure 1–2 ConoProbe Mark III Dimensions no Lens



LENS	L
F25	175.8
F50T	170.0
F50; F75; F100; F150	167.0
F200	165.0
F250	168.0

LENS	L
F16	236.0

LENS	L
F40	198.5

LENS	L
F50E	265.8
F75E; F100E F125E; F150E	241.3

Figure 1–3 ConoProbe Mark III and Lens Dimensions

1.6 Communication Box Description

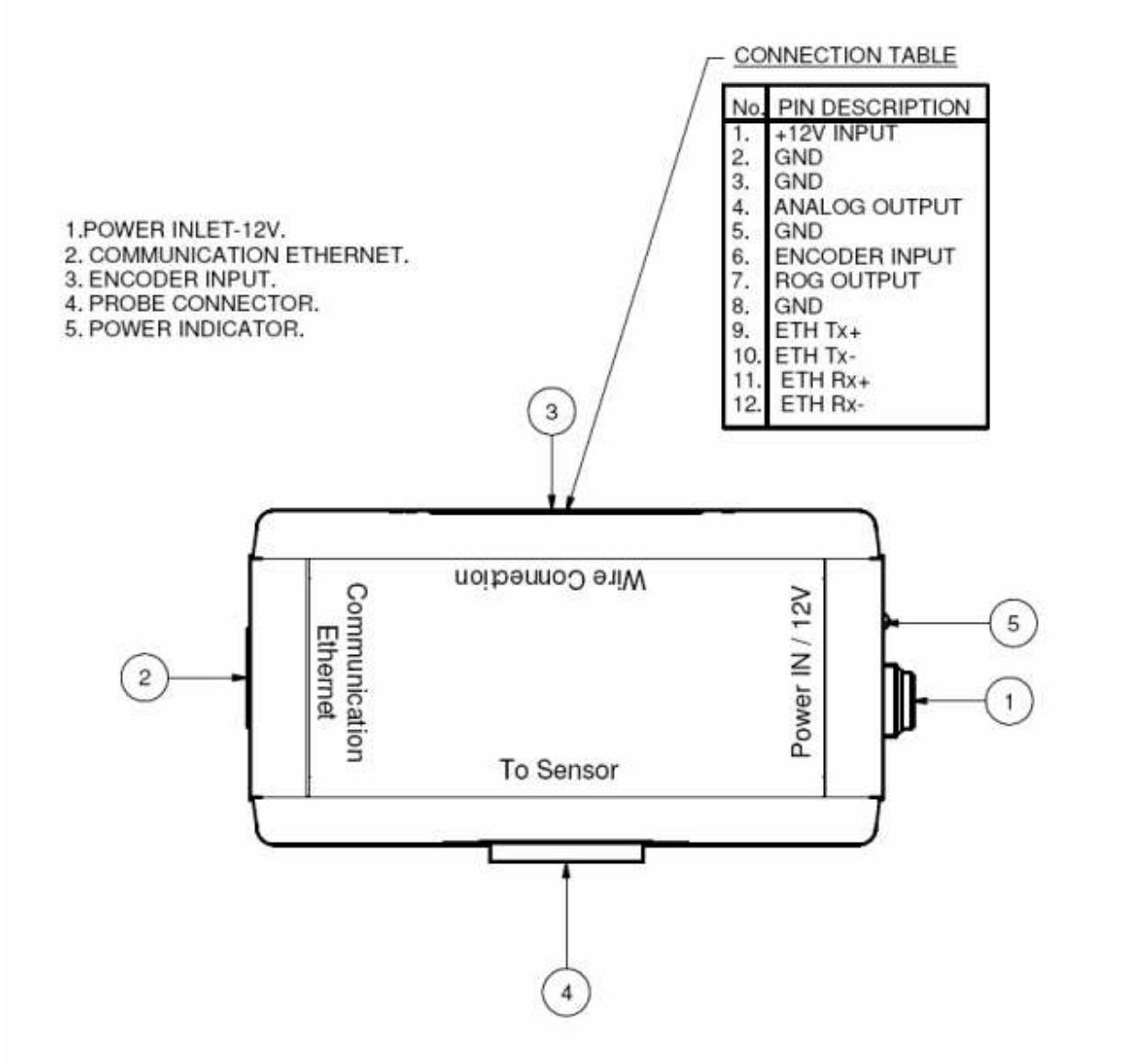


Figure 1–4 ConoProbe Mark III communication box

2 Connecting the ConoProbe Mark III

This chapter explains how to connect the ConoPobe Mark III and install its support software. It includes the following sections:

System Requirements – PC requirements for the MKIII.

Connecting the MKIII – A description of how the MKIII, the communication box unit, and the host PC are connected.

Connecting the Installing the ConoProbe Mark III OEM Software - Installing Instructions on how to install the OEM Software.

2.1 System Requirements

The ConoProbe Mark III uses a PC for operation. The PC must meet the following requirements:

A Pentium III processor (Pentium IV recommended).

At least 256MB physical RAM (512MB recommended).

SVGA display (800x600 resolution, 16 bit color palette).

A 10/100 Ethernet LAN network connection.

Windows™ 2000, Windows™ XP, or Windows™ Vista operating systems.

2.2 Connecting the ConoProbe Mark III

The communication box unit must be connected to both the ConoProbe Mark III and the host PC. To connect the communication box to the probe and the PC (refer to *Figure 2-1*):

2. Connect the Communication Box to the probe using the 14-pin D-Type connector cable.
3. Connect the Communication Box to the PC with an Ethernet crossover cable.
4. Connect the Communication Box to the power supply (12v @ 0.5 Amp).

To learn how to use the external trigger input, or for more information about the analog output, encoder inputs, and start of measurement output (ROG), see Chapter 5: The Input/Output Description.

.

Figures 2–1 and 2–2 display how the probe, the Connection Box, and the host PC are connected.

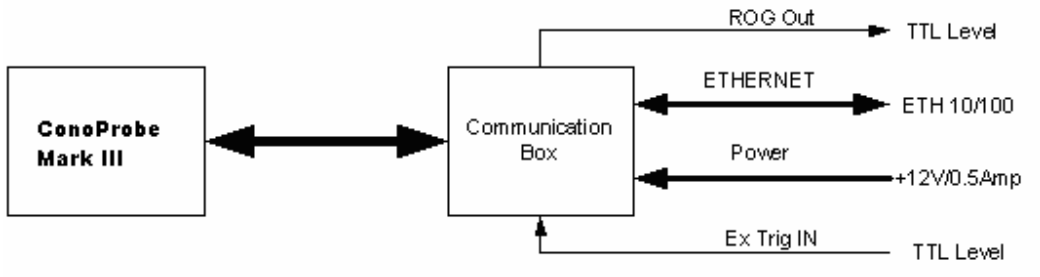


Figure 2-1: Communication Box Cables and Connections

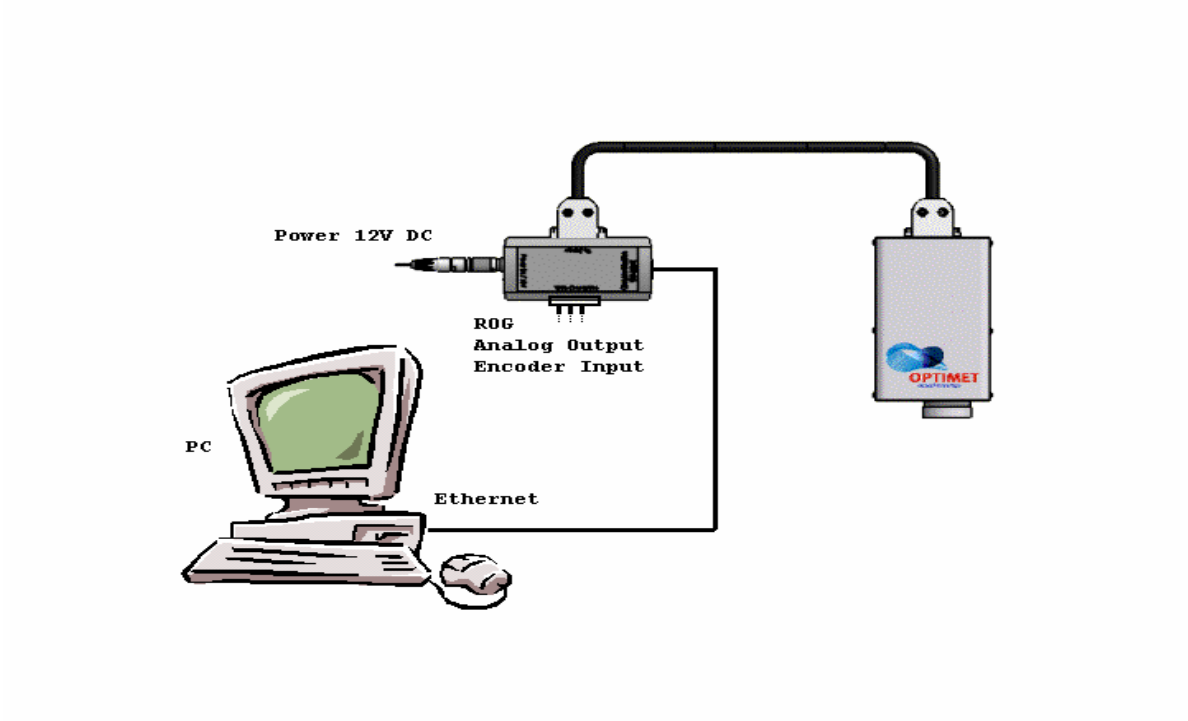


Figure 2–2 ConoProbe Mark III System Layout

WARNING:

When connecting or disconnecting the probe to the Communication Box, make sure that the Communication Box power is unplugged. Connecting or disconnecting the probe to the Communication Box while its power is on could damage the probe.

2.3 Installing the ConoProbe Mark III OEM Software

NOTE:

For installation in Windows™ 2000, Windows™ XP or Windows™ Vista, you must have administrator rights.

To install the software:

1. Insert the installation CD-ROM in your CD-ROM drive.
2. From the `Setup` folder, run `Setup.exe`. The **Welcome** screen appears.
3. Click **Next**. The **Software License Agreement** screen appears.
4. After you have read the agreement and accept its terms, click **Yes**. The **Choose Destination Location** screen appears.
5. Browse to the directory in which you want to install the OEM software, or use the default directory. Click **Next**. The **Select Program Folder** appears.
6. Enter a name for the OEM program folder, or leave the default name, and click **Next**. The **Start Copying Files** screen appears.
7. The setup program installs the ConoProbe Mark III OEM software in the selected directory. After it has finished, the **Setup Complete** screen appears.
8. If required to restart your computer, click 'Yes, I want to restart my computer now.' or 'No, I will restart my computer later.' Click **Finish**.

2.3.1 Communication Set-up

The MKIII is connected to the host PC via an Ethernet port on the host PC. This port can be configured by the connection wizard of either Windows™ 2000, Windows™ XP or Windows™ Vista. This wizard can be found by pressing the Network Connections icon from the control panel of the host PC. This will start the connection wizard. Right click the properties menu item on the network connection you will use for the MKIII. The network properties dialog will then be displayed as shown below in figure 2-3. Remove the checks from the first three items, Client for Microsoft Networks, File and Printer Sharing for Microsoft Networks and QoS Packet Scheduler

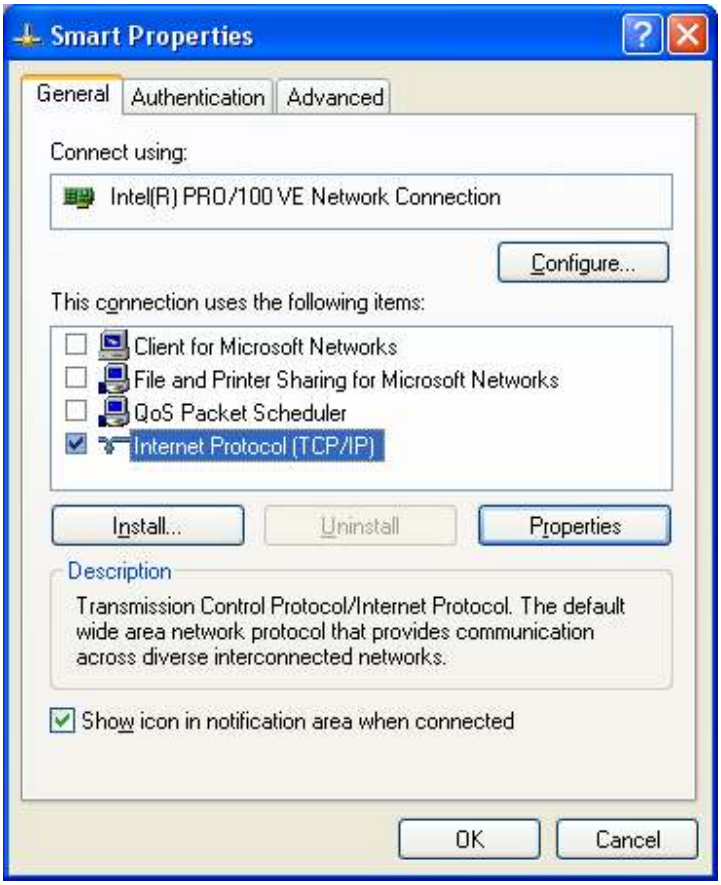


Figure 2–3 The Network Properties dialog

Next, select the item **Internet Protocol (TCP/IP)** and click the **Properties** button. The General page will appear. In this dialog do the following:

1. Click on the button labeled **Use the following IP address**:
2. Change the IP address field to 1.2.3.9
3. Change Subnet mask to 255.255.255.0
4. Make sure the Default Gateway field is empty.

After the changes have been made verify that the settings are identical to those in figure 2–4, then press OK.

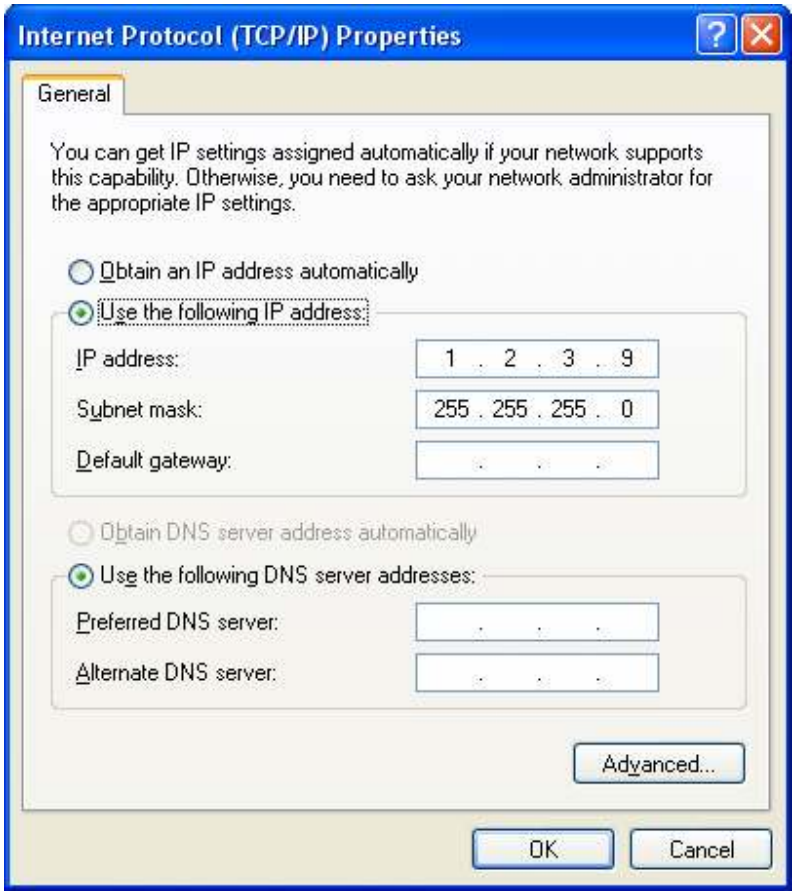


Figure 2-4 The Network Properties dialog displaying TCP/IP properties.

After configuring the settings, rename the network connection to Smart Probe in the Network Connections Window. See Figure 2-5.

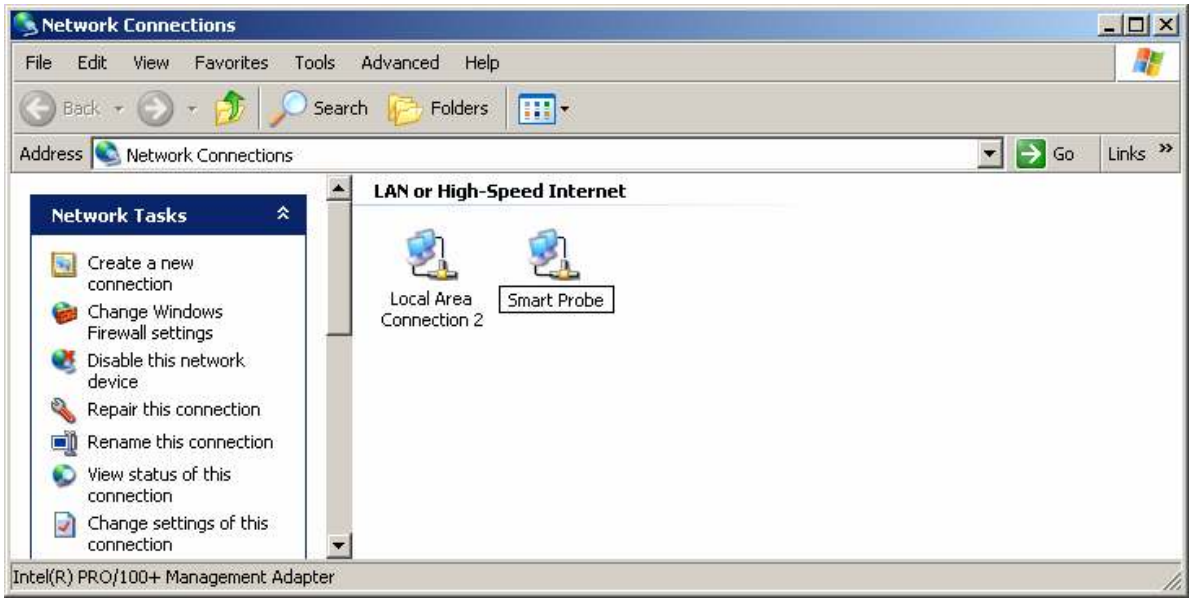


Figure 2-5 Renaming the Network Connection.

2.3.2 Verifying the Network Connection with the Probe

The network communication with the probe can be verified by running the communication test utility "SmartCommTest.exe" which is located in the Utilities folder. Run the utility (see Figure 2-6) and hit the Test 1 button. A dialog indicating that the successful communication with the probe will be displayed if the connection was setup properly.

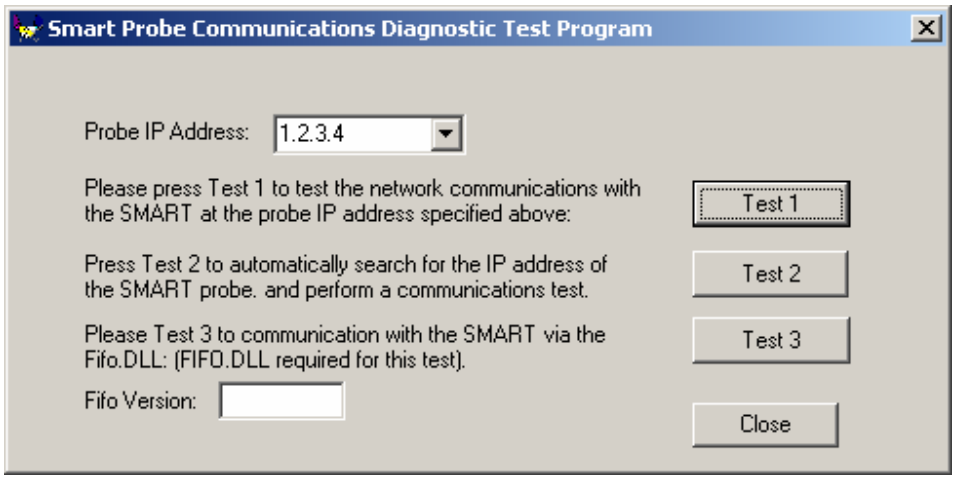


Figure 2-6: The Smart Probe Communication Utility

3 Mark III Specifications and Lens Type

Below is a table of specifications that are arranged by type and focal length. All lenses that come with the specific Mark III are custom calibrated to that particular sensor.

	Lens Assembly Type (By Focal Length in mm)*													
	Standard										High Definition			
ConoProbe Technical Specifications	16	25	40	50	50 ext.	75	100	150	200	250	16	25	40	50
Z (Vertical) axis														
Static Resolution [10] (µm)	<0.1						0.1	0.35	0.75	1.5	<0.1			
Precision [1,4] (µm)	<2**	<3**	<4	<6	<6	<10	<15	<35	<70	<100	<0.5	<1	<2.0	<2.5
Precision [1,4] (µm) at 80%	**													
Reproducibility 1σ [2] (µm)	<0.15	<0.4	<0.7	<1	<1	<2	<4	<15	<15	<15	<0.1	<0.2	<0.4	<0.5
Working Range (mm)	0.6	1.8	4	8	8	18	35	70	125	180	0.2	0.6	1.4	2
Standoff [3] (mm)	12	15	40	42	85	65	90	140	185	240	11	14	37	40
Measurement Aspect Ratio	**													
Lateral Axes (X and Y)														
Laser Spot Size (X) (appr.) [5] (µm)	11	22	30	45	60	65	75	120	170	220	3.5	6	TBD	15
Lateral Resolution (X) [6] (µm)	5	12	14	15	20	25	35	50			2	4	TBD	10
Laser Spot Size (Y) (appr.) [5] (µm)	**													
Weight														
Lens (g)	460	40	122	25	400	25	25	25	25	25	460	40	122	25
Probe (g)	700													
Communication Box (g)	120													
Data Handling														
Max Measurement Rate (Hz)	3000													
Export Data to:	Excel, ASCII text file, BMP, JPEG, UBM, VRML**													
Interface														
Communication	Ethernet 10/100 UDP													
Discrete Signal Control	ROG – Output, External Trigger – Input, Analog Signal Range – +/- 5V													
Applications														
Precision for radius measurements [7]	Relative to lens accuracy													
Angle measurement [8]	170°													
Working temperature [7]	18 to 35°C													
Continuous shock resistance 245 m/s2-25g-6ms	>6000 shocks 6 directions													
Supply Voltage	12 VDC													
Light Source	Visible red laser diode – Wavelength 655nm													
Laser Class	FDA Class II – IEC Class 2													

NOTES:

- Extended lenses for focal lengths from 50 to 200 and other objectives for different focal lengths are available upon request.
- **Glossary**
  a) **Measurements Range** - Range over which distance measurements meets measurements specifications.

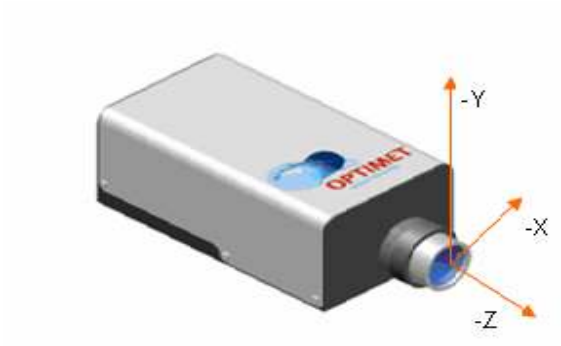
b) **Relative Distance Measurement** – Distance measurement between two points in the Measurement Range (in contrast to Absolute Distance Measurement).

c) **ROG** Read-Out Gate – Internal measurements synchronization pulse, used when **Mark III** is the **master**.

d) **External Trigger** – External start of measurements, used when **Mark III** is the **slave**.

e) **Analog Signal** – Conversion of Z measurement to an analog representation (Optional)

f) **ConoProbe Mark III** – Axis definitions see below:



(1) Specifications are defined under the following parameters and conditions:

a. Measured object - Diffusive surface of certified Johansson blocks.

b. Relative distance measurements covering 50% of the measurement range.

c. Averaging over 200 measured points.

d. Sampling step of ½ of the spot size.

e. Measurements done along Y axis.

f. Environmental temperature and Laser Power same as in the calibration.

Different measurement parameters such as reflective surface (fine-machined-N6 surfaces) or smaller sampling steps may affect precision.

(2) **Reproducibility:** The Standard Deviation of 5 relative measurements of parallel profiles scans with offset -normal to scan direction (parallel to X axis) Measurement parameters same as (1).

(3) **Standard deviation** - Standard deviation of the random error (Measurement Uncertainty) of absolute distance measurements at the middle of the measurement range. Measurement parameters same as (1).

(4) **Standoff** - The distance from the tip of the objective lens housing to the center of the measurement range. The tolerance of the standoff distance may deviate up to 10% of the stated value.

(5) **Laser beam spot size** - the effective spot width in the Y-Axis measured at the center of the measurement range at FWHM points. The Lateral resolution can be estimated as half of the spot size.

(6) **Angle coverage** - Measurable points up to ± 85° from normal incidence angle. For angles up to 60°, the angle estimation (by indirect measurement) is within a tolerance of 0.05°. For angles exceeding 60° the angle estimation (by indirect measurement) is within a tolerance of 0.25° (assuming minimum of 50 measurable points). Measured in Y direction.

(7) Calibration is done at 22-24°C. Temperature dependence of relative distance measurement is less than 0.025%/°C.

4 ConoProbe Mark III OEM API Reference

This chapter describes the ConoProbe MKIII API library. It describes the various data types and functions available in the MKIII and the measurement modes used to drive it.

This chapter includes the following sections:

Installing the FIFO DLL – Describes how to install the DLL for OEM programming in Windows™ 2000, and Windows™ XP.

Type Definitions – Describes the types available to the OEM programmer.

Setup Functions – Describes the functions that set up communication with the probe and configure measurement parameters.

Information Functions – Describes the functions that provide information about the system configuration.

Measurement Functions – Describes the functions that drive the probe measurement process.

Error Handling Functions – Describes the error handling functions.

Timeout Handling Functions – Describes the functions that set and provide information about the timeout parameters.

4.1 Software Differences between the Mark III and the Mark II ConoProbes

The ConoProbe Mark III has different electronics and timings than the Mark II ConoProbe. The communication between the PC and the ConoProbe Mark III is accomplished via an Ethernet cable using the UDP Protocol and the FIFO.DLL as the driver. (The latest released version is 6.0.0.0)

The following API logic has been changed in the ConoProbe Mark III. If you are a user of Mark II ConoProbe, please note the following important changes:

1. **Lens logic**
   a. Every Mark III has a "0 mm" lens as a first lens which should NOT be used for any scanning purposes. This lens is only for internal use by OPTIMET.
   b. FifoGetLensCount() returns the total number of all calibrated lenses + 1 for the "0 mm" lens.
   c. When using FifoGetActiveLensIndex(), FifoSetActiveLensIndex() and FifoGetLens() be aware of the "0 mm" lens.

2. **Setting the measurement parameters before making any measurements**
   a. After powering ON the Mark III, the user should ensure the following parameters are set before making any measurements. These parameters are persistant until the probe is powered off.
   b. Power by calling SetPower()
   c. Frequency by calling SetFrequency()

3. **The role of the DILUTION factor in measurement acquisition**

a. The measurement acquisition process is started by calling either FifoBeginReadMeasurementStream(), FifoBeginReadMeasurements(), or FifoReadMeasurementStream. In the **ConoProbe Mark II**, the parameter **DelayBetweenMeasurements** used in the above functions is defined as the time between successive measurements in micro seconds in **TIME** mode (1/Scan Frequency) and the time between successive pulses in **PULSE** mode. For the **ConoProbe Mark III**, the **TIME** mode definition of this parameter has not been changed. In **PULSE** mode, however, **DelayBetweenMeasurements** is the factor which one wishes to dilute pulses and is independent of the CCD frequency*. For example: One wishes to perform a 20 mm scan at probe frequency = 1000 (one measurement every 1ms) using a motor stage operating at 2000 pulses per mm at a work velocity of 10 mm/sec. The motor stage will produce 20000 trigger pulses per second. Since the maximum measurement rate is 1000 measurements per second, we require a dilution factor of 20 (20000/1000), to measure at every 20^{th} pulse to perform an accurate scan. Note that before making a pulse mode measurement; make sure that the CCD frequency is set to the proper value by calling FifoSetFrequency.
b. **Jitter and Noise Considerations:** Due to jitter and imperfections of the signal coming from the motor stage encoder, this dilution factor should be increased by approximately 10%
c. For more information on how measurements are generated and the delay time between measurements please refer Chapter 5, **The Input/Out Description**.
d. If FifoBeginReadMesurementStream, FifoBeginReadMeasurements, or FifoReadMeasurementStream is used in time mode, the probe frequency will be set according to the resulting quotient of the integer division of 100000/DelayBetweenMeasurements rounded up to the nearest integer. This could result in a slightly different value for the desired frequency. If one wishes to use an exact value of frequency, FifoSetFrequency should be called prior to calling the above mentioned functions and DelayBetweenMeasurements should be set to 0.

*Frequency defines the integration time of the CCD and provides lower bounds to the dilution factor.

4. **Tag field behavior in TMeasurement structure using PULSE mode**
   a. When an acquisition process is started (for example, calling FifoBeginReadMeasurementStream()), the Tag field value for the first received measurement will NOT necessary be 0.
   b. The Tag value counts cyclically (0...255), the number of pulses after dilution. If the ConoProbe Mark III receives all the pulses during the scan the Tag values will be successive. If missing any pulses occurred during the scan, the difference between two successive Tag values minus one will represent the number of missing pulses between those two measurements. For example: CCD Frequency = 3 kHz, Pulse frequency = 12 kHz, DelayBetweenMeasurements / Dilution= 2. We will miss each second pulse (12 / 2 / 3 = 2), in this case the Tag values will be: …, 7, 9, 11, 13 …
5. **Acquisition modes:**
   Currently, only Time and Pulse Modes are supported in the ConoProbe Mark III. Time-trigger is under development and will be available in the next software release which will be later in 2007.

6. **Buffer Acquisition:**
   The Mark III uses the physical memory of the host computer to receive buffer mode measurements and can make up to 65535 acquisitions in one scan, whereas the Mark II utilizes an internal buffer which is limited to 7000 measurements. In the Mark III, the buffer acquisition process utilizes a separate program thread in the FIFO.DLL whereas the Mark II this processing is performed as a separate process in the hardware.

4.2 Installing and Using the FIFO DLL

The dynamic link library, "FIFO.DLL" is automatically installed during installation of the customer probe to …\SDK\Library folder. You must copy FIFO.DLL file before you begin OEM programming to your project development folder. The DLL may be linked to your C, C++, VB, C# or Delphi application in two ways: Statically or Dynamically.

Statically:
With static linking, the executable using the FIFO.DLL links to an import library FIFO.LIB file (located in SDK\Lib\Microsoft or SDK\Lib\Borland folders). The operating system loads the FIFO.DLL when the executable using it is loaded. The client executable calls the DLL's exported functions just as if the functions were contained within the executable.
You must copy FIFO.LIB file from the …\SDK\Library folder to your project development folder.
In order to support Mark III ConoProbe in already existing applications which work with the Mark II ConoProbe you must just overwrite the Mark II FIFO.DLL with Mark III FIFO.DLL (please refer to Chapters 4.1 and 4.10).

Dynamically:
With dynamic linking, the executable using the FIFO.DLL must make function calls to explicitly load and unload the DLL and to access the DLL's exported functions. The client executable must call the exported functions through a function pointer. Please read in the MSDN about the following Win32 functions: LoadLibrary(), GetProcAddress(), FreeLibrary().
If you are intending to operate multiple ConoProbe devices in a single computer or if you want to work with Mark II (Parallel communication) and Mark III (Ethernet communication) in the same application you must use dynamic DLL loading:

Example:

```
if (ConfigurationFile.Mode == Parallel)
{
            hModule = LoadLibrary(".\MarkII\Fifo.dll");
}
else // Ethernet
{
            hModule = LoadLibrary(".\MarkIII\Fifo.dll");
}
if (hModule != NULL)
{
            FifoInit = (FifoInit)GetProcAddress(hModule, "FifoInit");
            …
}
```

Note: If you are using ShowProbeDialog() (see Chapters 4.104.11.1) function you must copy and deploy with your application FIFO_PS.DLL file which is located in …\Developer\Library folder. This

DLL is loaded dynamically by FIFO.DLL once you call ShowProbeDialog() function and must be located in the same folder as FIFO.DLL.

4.3 Type Definitions

This section describes the various data types available to the OEM programmer in the ConoProbe OEM. These types serve as both input and output parameters for various OEM functions.

4.3.1 BYTE

Definition

```
typedef unsigned char BYTE;
```

Explanation

Standard nomenclature to distinguish between signed and unsigned char. This is a standard Windows™ API type.

Used By

Types: TLensIndex, TPower and TLens.

4.3.2 WORD

Definition

```
typedef unsigned short WORD;
```

Explanation

Standard nomenclature to distinguish between signed and unsigned short. This is a standard Windows™ API type.

Used By

Types: TLens and TMeasurement.

Functions: `FifoBeginReadMeasurements`, `FifoGetMeasurements`, `FifoGetPower` and `FifoSetPower`.

4.3.3 TProbeVersionInfo

Definition

```
typedef struct
{
  ULONG  SN;
  ULONG  Date;
  ULONG  Time;
  WORD    Version_LOW_SW;
  WORD    Version_HIGH_SW;
  WORD    Version_LOW_HW;
  WORD    Version_HIGH_HW;
} TProbeVersionInfo;
```

Explanation

20 byte probe version information structure returned by the porbe to the PC.

Following are explanations of the function's parameters:

`SN` – The serial number of the probe.

`Date` – Used to distinguish between various calibrations.

`Time` – The number of times the probe has been calibrated with this lens.

`Version_LOW_SW` – probe internal software version, low word.

`Version_HIGH_SW` – probe internal software version, high word.

`Version_LOW_HW` – probe internal hardware version, low word.

`Version_HIGH_HW` – probe internal hardware version, high word.

Example

```
TProbeVersionInfo versionInfo;
versionInfo = FifoGetProbeInfo();
      …
```

Used By

Functions: FifoGetProbeInfo.

4.3.4 TPower

Definition

```
typedef BYTE TPower;
typedef enum {pFine = 1, pCoarse = 2, pComposite = 3} Power;
```

Explanation

The coarse and fine tuning power level of the probe's laser. Using the parameter pComposite, one can set or get the Fine power in one unsigned short variable where bits 0-5 represent the fine power and bits 6-11 represent the coarse power.

Used By

Functions: `FifoGetPower` and `FifoSetPower`.

4.3.5 TMode

Definition

```
typedef BYTE TMode;
typedef enum {mTime, mPulse} Mode;
```

Explanation

The mode in which every measurement is made, either by number of milliseconds, number of pulses, or number of milliseconds but returned at every pulse.

Used By

Functions: `FifoBeginReadMeasurements`, `FifoBeginReadMeasurementStream` and `FifoReadMeasurementStream`.

4.3.6 TLensIndex

Definition

```
typedef BYTE TLensIndex;
```

Explanation

Index of lens installed on the probe. Each probe has a set of lenses installed, and each lens has an index through which it can be accessed. Legal values are 1 to 31.

Used By

Functions: `FifoGetActiveLensIndex`, `FifoGetLens`, `FifoGetLensCount`, `FifoGetLensZTweakUser`, `FifoSetActiveLensIndex` and `FifoSetLensZTweakUser`.

4.3.7 TEdges

Definition

```
typedef WORD TEdges;
typedef enum {eBoth = 0, eFalling = 0x4000, eRising = 0x8000} Edges;
```

Explanation

For use in external trigger mode (pulse mode).

The MKIII only identifies the Rising Edge of the External Trigger Pulse. (For ConoProbe MKII users, please note, the ConoProbe MKIII only supports Rising Edge. It does not support the parameters Falling/Both Edges.)

Used By

Functions: `FifoBeginReadMeasurements`, `FifoReadMeasurementStream` and `FifoBeginReadMeasurementStream`.

4.3.8 TLoadError

Definition

```
typedef BYTE TLoadError;
typedef enum {loeNone,
      loeCommunication,
      loeParam,
      loeTimeOut,
} LoadError;
```

Explanation

System error code as follows:

`loeNone` – No error.

`loeCommunication` – Communication error occurred.

`loeParam` – Parameter error.

`loeTimeOut` – Probe failed to respond within timeout period.

Used By

Function: `FifoGetError`.

4.3.9 TLens

Definition

```
typedef struct {
      short FocalDistance;
      char Type;
      char Iteration;
      struct {
            BYTE Day;
            BYTE Month;
            WORD Year;
      } DateOfCalibration;
      float DistanceMin;
      float DistanceMax;
      short Power;
      short FinePower;
      short MCUVersion;
      short TemperatureTenthsDegree;
      BYTE Reserved[232];
} TLens;
```

Explanation

Lens data is stored in the probe's *Flash* memory for each lens. This data is derived during calibration of the lens. The following are explanations of the lens' parameters:

`FocalDistance` – The distance from the lens to the focal point of the laser beam emitted by the probe.

`Type` – Used to distinguish between Standard, Telescopic, and High Precision lenses. Value

range: (a–z/A–Z).

`Iteration` – The number of times the probe has been calibrated with this lens.

`DateOfCalibration` – The date of calibration of the probe with this lens.

`DistanceMin` – The shortest distance from the probe that can be measured with this lens.

`DistanceMax` – The longest distance from the probe that can be measured with this lens.

`Power` – This value should be set to zero and is for backward compatibility only.

`FinePower` – This value should be set to zero and is for backward compatibility only.

`MCU Version` – This value should be set to zero and is for backward compatibility only.

`TemperatureTenthsDegree` – This value should be set to zero and is for backward compatibility only.

`Reserved` – Reserved information.

The Lens ID is composed of `FocalDistance`, `Type` and `Iteration`.

Used By

Function: `FifoGetLens`.

4.3.10 TMeasurement

Definition

```
typedef struct {
            float Frequency;
            float Distance;
            WORD Snr;
            short Sat;
            WORD Total;
            BYTE BitSet;
            BYTE Tag;
} TMeasurement;
```

Explanation

16 byte measurement structure returned by the probe to the PC, where:

`Frequency` – The optical signal described in terms of Power Spectrum Distribution (*this is not the scanning frequency*).

`Distance` – The distance measured.

`SNR` – The signal quality, given as a value between 0 and 1023. To convert this value to a percentage, multiply it by 100 and divide by 1024 (refer to *Example* below).

`Sat` – For in-house test purposes only.

`Total` – The energy detected by the CCD sensors. This can be used to optimize the measurement and to detect surface changes such as edges or color changes (refer to **Section 5.4.6 Optimizing Signal Quality***)*.

`BitSet` – The measurement status flag (0 = object in range; 2 = object is too close to the probe or there is a bad signal; 4 = object is too far from the probe or the signal is too noisy; 255 = unknown).

`Tag` – The tag of the measurement that can be used to identify missed measurements. Each

`TMeasurement` has a sequential tag number.

Example

This example illustrates use of the SNR parameter.

```
TMeasurement m;
FifoReadMeasurement(&m);
if (m.Snr x 100/1024 < 80)
       printf ("Signal quality is below 80%");
```

Used By

Functions: `FifoGetMeasurements`, `FifoReadMeasurement` and `FifoGetMeasurementStream`.

Event Function: `TMeasurementStreamEvent`.

4.3.11 TBeginMeasurementStreamEvent

Definition

```
typedef void (_stdcall *TBeginMeasurementStreamEvent)(void);
```

Parameters

None

Returns

None

Explanation

Pointer to function to be called when beginning to drive the probe to report measurements in stream (real–time). For example, use `TBeginMeasurementStreamEvent` to show the dialog box representing the scan results.

Used By

Function: `FifoReadMeasurementStream`.

4.3.12 TMeasurementStreamEvent

Definition

```
typedef BYTE (_stdcall *TMeasurementStreamEvent)(int nMeasurement, TMeasurement*
Measurement);
```

Parameters

- `nMeasurement` - number of measurements the probe has made up to now.
- `TMeasurement*` - Pointer to TMeasurement

Returns

Function returns 0 to stop measuring in stream, otherwise measuring continues

Explanation

Pointer to function to be called when driving the probe to report measurements in stream (real–time).

For example, use `TMeasurementStreamEvent` to add a new point to the graph representing the scan results.

Used By

Function: `FifoReadMeasurementStream`.

4.3.13 TFinishMeasurementStreamEvent

Definition

```
typedef void (_stdcall *TFinishMeasurementStreamEvent)(int MeasurementsCount,
int MeasurementsDeliveredCount);
```

Parameters

- `MeasurementsCount` - number of measurements the probe has made up to now.
- `MeasurementsDeliveredCount` - number of measurements sent to `TMeasurementStreamEvent`

Returns

None

Explanation

Pointer to function to be called when the PROBE has finished acquiring measurements in stream (real–time) mode. For example, use `TFinishMeasurementStreamEvent` to launch a filtering or shading algorithm on the scan results or to prompt the user to save a file.

Used By

Function: `FifoReadMeasurementStream`.

4.3.14 TMeasurementFilter

Definition

```
typedef BYTE TMeasurementFilter;
typedef enum {mfNone, mfSnr, mfTotal} MeasurementFilter;
```

Explanation

Available filters to preprocess measurements with errors. (refer to 4.4.7 FifoSetMeasurementFilter).

Used By

Function: `FifoSetMeasurementFilter`.

4.3.15 TLanguage

Definition

```
typedef BYTE TLanguage;
typedef enum {lEnglish, lFrench, lGerman} Language;
```

Used By

Type: `TParameters`.

Function: `ShowProbeDialog`.

4.3.16 TUnits

Definition

```
typedef BYTE TUnits;
typedef enum {uMM = 0, uMils = 1, uMicroM = 2} Units;
```

Explanation

Measurement units used in the Probe dialog.

Used By

Type: TParameters.

Function: `ShowProbeDialog`.

4.3.17 TGraphType

Definition

```
typedef BYTE TGraphType;
typedef enum {gtNone, gtRawA, gtRawB, gtRawAB, gtRawAMinusB, gtRawAPlusB,
gtContrast, gtFrequency} GraphType;
```

Explanation

Graph type selected in the Probe dialog drop-down list.

Used By

Type: TParameters.

Function: `ShowProbeDialog`.

4.3.18 TDlgProc

Definition

```
typedef int (_stdcall *TDlgProc)(void *hWnd, int Msg, int wParam, int
lParam);
```

Explanation

Callback function of the Probe dialog

Used By

Type: TParameters.

Function: `ShowProbeDialog`.

4.3.19 TParameters

Definition

```
typedef struct {
```

```
  TLanguage Language;
  TUnits Units;
  TGraphType GraphType;
  BYTE UseExStatisticsView;
  BYTE UseExGraphView;
  BYTE UseAO;
  short Power;
  short FinePower;
  short Frequency;
  TDlgProc DlgProc;
} TParameters;
```

Explanation

TParameters structure used by the ShowProbeDialog function. The parameters are as follows:

`Language` – The language used in the Probe dialog.

`Units` – The measurement units used in the Probe dialog (millimeters, mils, microns).

`GraphType` – The default graph type selected in the Probe dialog.

`UseExStatisticsView` – If True, displays Extended Statistics view. Otherwise, displays Standard Statistics view.

`UseExGraphView` – If True, displays axis values on graph. Otherwise, displays graph without axis values.

`UseAO` – If True, uses Analog Output.

`Power` – The default coarse power.

`FinePower` – The default fine power.

`Frequency` – The default frequency.

`DlgProc` – The callback function, which can be used to customize the probe screen (can be NULL).

Used By

Function: `ShowProbeDialog`.

4.3.20 TResults

Definition

```
typedef struct {
  short Power;
  short FinePower;
  int Frequency;
} TResults;
```

Explanation

Structure of the results, representing the parameter adjusted by the user, returned by the `ShowProbeDialog` function

1. `Power` – The adjusted coarse power.
2. `FinePower` – The adjusted fine power.
3. `Frequency` – The adjusted frequency.

Used By

Function: `ShowProbeDialog`.

4.4 Setup Functions

Setup functions are used to set up communication with the PROBE, as well as to configure measurement parameters.

4.4.1 FifoInit

Initiates communication with the ConoProbe Mark III with the default Probe IP Address "1.2.3.4". The default host adapter IP address is "0.0.0.0" which represents any IP address, and the default socket reception buffer is sufficient for 10000 measurements. This function should be used if you have only one probe connected to the host computer and the IP of the probe is "1.2.3.4". If you are working with more than one probe, use `FifoInitEx or FifoIpInit` to initialize communication and pass the IP address for each device. Note that the parameter Port is only used for backwards compatibility and should be set to 1. After successful initialization the frequency of the probe is set to 500Hz and the composite power value 2000.

Prototype

```
BOOL _stdcall FifoInit(WORD Port);
```

Parameters

```
WORD Port; // Ignored
```

Returns

`True` if successful in communicating with the probe.

Example

```
WORD Port = 1;
if (!FifoInit(Port))
      printf ("Cannot communicated with probe.  Please check your
      connections")
 else
      /* begin working with probe */
```

…

See Also

`FifoInitEx` and `FifoTerminate`.

4.4.2 FifoInitEx

Initializes communication with the ConoProbe Mark III with a specified Probe IP Address. The default host adapter IP address is "0.0.0.0" which represents any IP address, and the default socket reception buffer is sufficient for 10000 measurements. The probe IP address parameter should be a 32-bit integer representation of the real address. For example if the IP address is 1.2.3.6, the value IPAdress = 0x01020306. This function should be used when working with more than one probe on the same host computer. After successful initialization the frequency of the probe is set to 500Hz and the composite power value 2000.

Prototype

```
BOOL _stdcall FifoInitEx(int IPAddress);
```

Parameters

`INT IPAdress`

Returns

`True` if successful in communicating with the probe.

Example

```
…
      INT IPAddress = 0x01020304; // ProbeIPAddress 1.2.3.4
      if (!FifoInitEx(IPAddress))
            printf ("Probe not found.  Please check your connections and IP
            address")
      else
            /* begin working with probe */
…
```

See Also

`FifoInit`, FifoIpInit

4.4.3 FifoIpInit

Initiates communication with the ConoProbe Mark III using specified IP addresses for the Probe and Host computer and the size of the sockets receive buffer in number of measurements. The actual size of socket buffer will be sizeof (TMeasurement) * number of measurements. The IP address parameter should be a 32-bit integer representation of the real address. For example if the IP address is 1.2.3.6, the value IP address = 0x01020306. This function should be used when working with more than one probe and when the user wishes to modify the size of the socket reception buffer, or use different host adapters. After successful initialization the frequency of the probe is set to 500Hz and the composite power value 2000.

Prototype

```
BOOL _stdcall FifoIpInit(int dwProbeIPAddress, int dwHostIpAddress, int
socketRcvMeasurementsBufferSize)
```

Parameters

`INT` dwProbeIPAddress, // IP address of probe

`INT` dwHostIpAddress, // IP address of host computer adapter

`INT` socketRcvMeasurementsBufferSize, // number of measurments

Returns

`True` if successful in communicating with the probe., false if function fails.

Example

```
…
       INT IPAddress = 0x01020304; // Probe IPAddress 1.2.3.4
       INT HostIPAddress = 0x01020309; // IPAddress of HOST Computer 1.2.3.9
       INT RecvBufferSize = 10000; // size of network buffer in bytes
       if (!FifoIpInit(IPAddress, HostIPAddress, RecvBufferSize))
            printf ("Probe not found.  Please check your connections and IP
            address")
       else
            /* begin working with probe */
…
```

See Also

`FifoInit`., FifoInitEx

4.4.4 FifoSetActiveLensIndex

Configures the probe to calculate distance based on specified lens calibration data. The lenses are numbered from 1 to `FifoGetLensCount()`.

Prototype

```
BOOL _stdcall FifoSetActiveLensIndex(TLensIndex Index);
```

Parameters

`TLensIndex`.

Returns

`True` if successful. Otherwise, returns `False`.

Example

```
…
       if (!FifoSetActiveLensIndex(Index))
            printf ("Lens index out of range");
       else
            /* set up for measurement process */
…
```

See Also

`FifoGetActiveLensIndex`, `FifoGetLens` and `FifoGetLensCount`.

4.4.5 FifoSetFrequency

Sets the probe's CCD frequency. This may vary between 50 Hz and the allowed maximum frequency (approximately 875Hz for the Mark III Standard and 3000Hz for the Mark III Extended version).

Prototype

```
BOOL _stdcall FifoSetFrequency(int Frequency);
```

Parameters

`Frequency`.

Returns

`True` if successful. Otherwise, returns `False`.

Example

```
…
  if (!FifoSetFrequency (frequency))
       printf ("Frequency too high");
  else
       /* Drive Probe in Single Capture mode */
```

See Also

`FifoGetMaxFrequency`, `FifoGetMCUVersion` and `FifoReadMeasurement`.

4.4.6 FifoSetMeasurementParams

Sets all of the measurement parameters with one function call. This function is convenient to use when more than one measurement parameter needs to be set at one time. This parameter is only relevant for Pulse mode measurement functions.

Prototype

```
BOOL _stdcall FifoSetMeasurementParams(int Frequency, WORD CompositePower,
TMode Mode, int Dilution);
```

Parameters

Frequency, CompositePower, Mode, Dilution

Returns

`True` if successful. Otherwise, returns `False`.

See Also

`FifoSetPower`, `FifoSetFrequency`, `FifoSetDilution`

4.4.7 FifoSetMeasurementFilter

Activates a filter to preprocess all measurements. Two filters are currently defined: SNR and Total. If a

measurement has a value outside of the range set by the user, the `function GetError() returns a parameter error, loeParam.`

Prototype

```
BOOL _stdcall FifoSetMeasurementFilter(TMeasurementFilter MeasurementFilter,
WORD wMinValue, WORD wMaxValue);
```

Parameters

`MeasurementFilter -` the filter type to activate: `mfSnr` – SNR filter; `mfTotal` – Total Filter; `mfNone` – disable all filters. To activate both filters, call `FifoSetMeasurementFilter` twice, once with `MeasurementFilter` set to `mfSnr` and once with `MeasurementFilter` set to `mfTotal`.

`wMinValue` – the minimum value for the range (0 for SNR filter, 0 for Total filter).

`wMaxValue` – the maximum value for the range (1023 for SNR filter, 32767 for Total filter).

Returns

`True` if successful. Otherwise, returns `False`.

Example

```
…
if (!FifoSetMeasurementFilter (mfSnr, 0, 1023))
       printf ("Error" );
  else
       /* Set up rest of measurement parameters */
…
```

See Also

None.

4.4.8 FifoSetPower

Sets the probe's laser power level. Use this function after you determine the appropriate values in the Probe Dialog .

Prototype

```
BOOl _stdcall FifoSetPower(TPower Power, WORD Value);
```

Parameters

`Power - pFine` for fine tuning, `pCoarse` for coarse tuning.

`Value` – power level (between 0–63).

Returns

`True` if successful. Otherwise, returns `False`.

Example

```
…
    if (!FifoSetPower (pCoarse, powerLevel))
         printf ("Requested Power Level too high" );
```

```
       else
            /* Set up rest of measurement parameters */
…
```

See Also

`FifoGetPower`.

4.4.9 FifoTerminate

Closes down communication between the PC and the probe.

Prototype

```
void _stdcall FifoTerminate();
```

Parameters

None.

Returns

None.

Example

```
…
/* Perform measurements until you decide to finish */
FifoTerminate();
...
```

See Also

`FifoInit`.

4.5 Information Functions

Information functions provide the OEM programmer with system configuration information. The most frequently used information function is `FifoDetect`, which is called before and assists the `FifoInit` function (refer to 4.4.1 FifoInit). Other information functions provide important information, such as information that is displayed when your program starts.

4.5.1 FifoDetect

This function is obsolete and should no longer be used. It is retained for backwards compatibility only.

Prototype

```
Word _stdcall FifoDetect();
```

Parameters

None.

Returns

The value 1.

See Also

`FifoInit, FifoInitEx, FifoIpInit.`

4.5.2 FifoGetActiveLensIndex

Queries the probe about the lens data currently in use for distance calculation.

Prototype

```
TLensIndex _stdcall FifoGetActiveLensIndex();
```

Parameters

None.

Returns

Index of lens calibration data currently in use.

Example

```
…
printf ("Presently active lens index is %d", FifoGetActiveLensIndex());
…
```

See Also

`FifoGetLens`, `FifoGetLensCount` and `FifoSetActiveLensIndex`.

4.5.3 FifoGetHeadSN

Queries the probe's serial number.

Prototype

```
LONG _stdcall FifoGetHeadSN(void);
```

Parameters

None.

Returns

The serial number of the probe unit. If no probe is connected or found, this function will return 0.

Example

```
…
printf ("Serial Number of Probe unit is %ld", FifoGetHeadSN ());
…
```

See Also

None.

4.5.4 FifoGetLens

Uploads data about a specific lens from the probe. The specified lens must be one of the set of lenses

supported by the user's probe. The lenses are numbered from 1 to `FifoGetLensCount()`.

Prototype

```
BOOL _stdcall FifoGetLens(TLensIndex Index, TLens *Lens);
```

Parameters

`TLensIndex`.

Pointer to `TLens` structure.

Returns

`True` if successful. Otherwise, returns `False`.

Example

```
TLens lens;
…
FifoGetLens(FifoGetActiveLensIndex(), &lens);
printf ("Focal length of present lens is %d millimeters", lens.FocalDistance);
…
```

See Also

`FifoGetActiveLensIndex`, `FifoGetLensCount` and `FifoSetActiveLensIndex`.

4.5.5 FifoGetLensCount

Queries the probe for the total number of lenses with which it has been calibrated. Note that this number may include some additional virtual lenses. You can check whether a lens is real or virtual by checking the Focal Distance field in the TLens structure, using the function `FifoGetLens`.

Prototype

```
TLensIndex _stdcall FifoGetLensCount();
```

Parameters

None.

Returns

`TLensIndex`: Count of lens stored in the probe.

Example

```
…
printf ("Total number of lenses calibrated with this Conoprobe is %d",
FifoGetLensCount());
…
```

See Also

`FifoGetActiveLensIndex`, `FifoGetLens` and `FifoSetActiveLensIndex`.

4.5.6 FifoGetProbeInfo

Queries the probe for its version information and serial number.

Prototype

```
TProbeVersionInfo _stdcall FifoGetProbeInfo ();
```

Parameters

None.

Returns

TProbeVersionInfo structure.

Example

```
WORD Port = 0;
TProbeVersionInfo probeVesionInfo;
if (!FifoInit(Port))
{
        /* obtain probe information */
        probeVesionInfo =  FifoGetProbeInfo ();
        printf("Probe SN = %d", probeVesionInfo.SN);
        printf("Probe Software Version = %d.%d",
        probeVesionInfo.Version_HIGH_SW, probeVesionInfo.Version_HIGH_SW );
        printf("Probe Hardware Version = %d.%d",
        probeVesionInfo.Version_HIGH_HW, probeVesionInfo.Version_HIGH_HW );
 }
```

See Also

TProbeVersionInfo.

4.5.7 FifoGetMaxFrequency

Queries the probe for its maximum measurement frequency.

Prototype

```
int _stdcall FifoGetMaxFrequency();
```

Parameters

None.

Returns

Maximum working frequency possible for the probe.

Example

```
…
printf ("Maximum measurement frequency is %d", FifoGetMaxFrequency());
…
```

See Also

`FifoGetFrequency`, `FifoSetFrequency` and `FifoBeginReadMeasurements`.

4.5.8 FifoGetMaxFrequencyEx

Queries the PROBE for its maximum measurement frequency. The parameter InBitSet should be set for zero. This function works the same as FifoGetMaxFrequency and is only provided for backwards compatibility. This function is obsolete.

Prototype

```
int _stdcall FifoGetMaxFrequencyEx(int InBitSet);
```

Parameters

```
int InBitSet =0;
```

Returns

Maximum working frequency for the PROBE.

Example

```
…
  int Freq = FifoGetMaxFrequencyEx(0);
  printf ("Max frequency is %d", Freq);
…
```

See Also

`FifoGetFrequency and FifoGetMaxFrequency.`

4.5.9 FifoGetMCUVersion

Queries the PROBE for the version of its micro–controller software.

Prototype

```
int _stdcall FifoGetMCUVersion();
```

Parameters

None.

Returns

MCU version.

Example

```
…
printf ("MCU Version of PROBE is %d", FifoGetMCUVersion());
…
```

See Also

None.

4.5.10 FifoGetMCUVersionRequired

Queries `Fifo` library for its software version. This function is obsolete.

Prototype

```
int _stdcall FifoGetMCUVersionRequired();
```

Parameters.

None.

Returns

version number.

Example

```
…
printf ("MCU Version required for tue Fifo library is %d",
FifoGetMCUVersionRequired());
…
```

4.5.11 FifoGetPower

Queries the PROBE for the current setting of its laser power.

Prototype

```
WORD _stdcall FifoGetPower(TPower Power);
```

Parameters

`Power – pFine` for fine tuning, `pCoarse` for coarse power level, pComposite for returns the fine power in bits 0-5 and the coarse power in bits 6-11.

Returns

Power level (between 0–63).

Example

```
…
printf ("Present setting of Conoprobe laser is Coarse :%d\tFine :%d\n",
       FifoGetPower (pCoarse), FifoGetPower (pFine));
…
```

See Also

`FifoSetPower`.

4.6 Measurement Functions

Measurement functions drive the PROBE measurement process. They form the nucleus of the OEM software.

In single measurement mode, use function `FifoReadMeasurement` (for example, to set up the probe in an optimal position).

In stream measurement mode, use function `FifoBeginReadMeasurementStream` to initiate the measurements. Use `FifoGetMeasurementStream` to read the measurements in a loop, and use the function `FifoReadMeasurementStream` to read the measurements via callback functions. Use `FifoAbortMeasurements` to stop measuring.

In buffered measurement mode, use functions `FifoBeginReadMeasurements` and

`FifoGetMeasurements`. The probe will automatically stop measuring when the number of requested measurements has been acquired. Use `FifoAbortMeasurements` to reset measurement clock.

Note: the meaning of the parameter DelayBetweenMeasurements that is used in the measurement initiation functions has a different meaning for pulse mode in the Mark III ConoProbe, please refer to section 4.1 for an in depth explanation.

4.6.1 FifoAbortMeasurements

Forces the probe to stop measuring. This function is useful when driving the probe's measurement cycle via external trigger from an encoder that is attached to a moving stage. If the stage stops moving due to unknown circumstances, use this command to force the probe out of its measurement cycle. Also use this function when you are finished acquiring a stream of measurements.

This function is relevant for all measurement modes.

Prototype

```
int _stdcall FifoAbortMeasurements();
```

Parameters

None.

Returns

Number of measurements performed before aborting.

Example

```
…
if (MotorStopped()) {
      FifoAbortMeasurements();
      /* upload partially filled buffer */
  }
…
```

See Also

`FifoBeginReadMeasurements`, `FifoGetMeasurements`, `FifoGetMeasurementsCount` and `FifoBeginReadMeasurementStream`.

4.6.2 FifoBeginReadMeasurementStream

Puts the probe in stream measurement mode.

Prototype

```
BYTE _stdcall FifoBeginReadMeasurementStream(int DelayBetweenMeasurements,
TMode DelayBetweenMeasurementsMode, TEdges Edges);
```

Parameters

`DelayBetweenMeasurements:`

TimeMode – The number of microseconds between measurements. The smallest value is 10^6/`FifoGetMaxFrequency`. The value is calculated by dividing $10^6$by the scan frequency and rounding up to the nearest integer. By

setting DelayBetweenMeasurements =0, the currently set value of probe frequency will be used.

PulseMode – The factor for diluting pulses. See section 4.1 for an example of deriving this factor.

`DelayBetweenMeasurementsMode` – The delay mode between measurements `(mPulse or mTime)`. This parameter refers to the `DelayBetweenMeasurements` parameter preceding it. If the mode is `mPulse`, the `DelayBetweenMeasurements` is the dilution factor. If the mode is `mTime`, the `DelayBetweenMeasurements` is the number of microseconds between successive measurements.

`Edges` –Trigger on the rising edge.

Returns

`True` if successful. Otherwise, returns `False`.

Example

```
TMeasurement M;
FifoBeginReadMeasurementStream(2000, mTime, eRising);
/* Read 200 points */
while (i <= 200) {
       if (FifoGetMeasurementStream(&M)) {
              printf ("Distance measured = %f", M.Distance);
              i++;
       }
  }
FifoAbortMeasurements();
```

See Also

`FifoAbortMeasurements` and `FifoGetMeasurementStream`.

4.6.3 FifoBeginReadMeasurements

Puts the probe in buffered measurement mode, creates a measurement thread. Measurements start being acquired after a delay that can be set by FifoSetMotorDelay. Calling GetMeasurements() will return the measurement buffer to the caller after the measurement thread has finished.

Prototype

```
BYTE _stdcall FifoBeginReadMeasurements(WORD nMeasurements, WORD Wait, TMode
WaitMode, int DelayBetweenMeasurements, TMode DelayBetweenMeasurementsMode,
TEdges Edges);
```

Parameters

`nMeasurements` – The number of measurement requests.

`Wait` – This parameter is no longer used, and is provided for backwards compatibility only. Set to 0.

`WaitMode` – This parameter is no longer used, and is provided for backwards compatibility only. Set to `mTime`.

`DelayBetweenMeasurements:`

TimeMode – The number of microseconds between measurements. The smallest value is 10^6/`FifoGetMaxFrequency`. The value in microseconds is calculated by

dividing 10^6 by the scan frequency and rounding to the nearest integer. By setting DelayBetweenMeasurements=0, the currently set value of probe frequency will be used.

PulseMode – Same as TimeMode but scaled by 1/Dilution factor to compensate for the pulserate generated by a motor stage controller. See section 4.1 for an example of deriving this factor.

`DelayBetweenMeasurementsMode` – `(mPulse or mTime)`. This parameter refers to the `DelayBetweenMeasurements` parameter preceding it. If the mode is `mPulse`, the `DelayBetweenMeasurements` is the dilution factor. If the mode is `mTime`, the `DelayBetweenMeasurements` is the number of microseconds between successive measurements.

`Edges` – Trigger on the rising edge.

Returns

`True` if successful. Otherwise, returns `False`.

Example

```
…
/* Set up for measurements */
…
/* Take 100 readings with no delay before starting. Measuring at every 3
milliseconds */
if (FifoBeginReadMeasurements (100, 0, mTime, 3, mTime, eRising))
       /* Get results when ready */
  else
       printf ("Failed to start measurement process");
…
```

See Also

`FifoAbortMeasurements`, `FifoGetMeasurements`, `FifoGetMeasurementsCount` and `FifoSetTimeOut`, `FifoSetMotorDelay`

4.6.4 FifoGetMeasurementStream

Uploads one measurement from the probe (FIFO queue) when the probe is in stream measurement mode.

Prototype

```
BYTE _stdcall FifoGetMeasurementStream(TMeasurement *Measurement);
```

Parameters

`Pointer to TMeasurement.`

Returns

`True` if successful. Otherwise, returns `False`.

Example

```
TMeasurement M;
FifoBeginReadMeasurementStream(2000, mTime, eRising);
```

```
/* Read 200 points */
while (i <= 200) {
      if (FifoGetMeasurementStream(&M)) {
            printf ("Distance measured = %f", M.Distance);
            i++;
      }
 }
FifoAbortMeasurements();
```

See Also

`FifoAbortMeasurements` and `FifoBeginReadMeasurementStream`.

4.6.5 FifoGetMeasurements

Uploads measurement buffer from the probe. Note that this function is called after calling FifoBeginReadMeasurements. FfoGetMeasurements does not return until the probe's measurement thread has finished.

Prototype

```
BYTE _stdcall FifoGetMeasurements(TMeasurement *Measurements, WORD nMeasurements);
```

Parameters

`Pointer to TMeasurement, number of measurements to upload.`

Returns

`True` if successful. Otherwise, returns `False`.

Example

```
…
if (FifoGetMeasurements(Measurements, 100)) {
      /* Display all distance measurements */
 else
      printf ("Failed to upload data");
…
```

See Also

`FifoAbortMeasurements`, `FifoBeginReadMeasurements`, `FifoGetMeasurementsCount` and FifoSetTimeOut.

4.6.6 FifoGetMeasurementsCount

Queries the probe for the total number of measurements processed and waiting to be read. `FifoGetMeasurementsCount` is important when the measurement process is aborted and you must know the exact number of measurements that were completed.

Prototype

```
int _stdcall FifoGetMeasurementsCount();
```

Parameters

None.

Returns

Number of measurements.

Example

```
…
if (MotorStopped()) {
       /* Force MKIII to finish measuring */
       FifoAbortMeasurements();
       /* Upload partially filled buffer */
       if (FifoGetMeasurements(Measurements, FifoGetMeasurementsCount())) {
              /* Display all distance measurements */
       else
              printf ("Failed to upload data");
  }
…
```

See Also

`FifoAbortMeasurements`, `FifoBeginReadMeasurements` and `FifoGetMeasurements`.

4.6.7 FifoReadMeasurement

Instructs the ConoProbe Mark III to perform a single capture and to upload the result. Use this function when you initially set up your object for scanning. For example, you can use this function to check the SNR at periodic intervals, for example, 100 ms. At the same time, you can change the power level or the distance from the probe to the object and thereby improve the scanning quality. You can also move the motors along the object in order to check the possible differences in quality of scanning. Calling this function every 100 ms allows you to see the resulting changes in the SNR.

Prototype

```
BOOL _stdcall FifoReadMeasurement(TMeasurement *Measurement);
```

Parameters

`Pointer to TMeasurement.`

Returns

`True` if successful. Otherwise, returns `False`.

Example

```
…
TMeasurement Measurement;
while (FifoReadMeasurement(&Measurement))
  printf ("Distance measured = %f", Measurmement.Distance);
…
```

See Also

`FifoSetFrequency` and `FifoSetPower`.

4.6.8 FifoReadMeasurementStream

Puts the ConoProbe Mark III in stream measurement mode, reporting measurements via callback

functions.

Prototype

```
BOOL _stdcall FifoReadMeasurementStream(int DelayBetweenMeasurements, TMode
DelayBetweenMeasurementsMode, TBeginMeasurementStreamEvent
BeginMeasurementStreamEvent, TMeasurementStreamEvent MeasurementStreamEvent,
TFinishMeasurementStreamEvent FinishMeasurementStreamEvent, TEdges Edges);
```

Parameters

- `DelayBetweenMeasurements`:

TimeMode – The number of microseconds between measurements. The smallest value is 10^6/`FifoGetMaxFrequency`. The value is calculated by dividing is calculated by dividing 10^6 by the scan frequency and rounding up to the nearest integer. By setting DelayBetweenMeasurements=0, the currently set value of probe frequency will be used.

PulseMode – The dilution factor. See section 4.1 for an example of deriving this factor.

- `DelayBetweenMeasurementsMode` – The delay mode between measurements `(mPulse or mTime)`. This parameter refers to the `DelayBetweenMeasurements` parameter preceding it. If the mode is `mPulse`, the `DelayBetweenMeasurements` is the dilution factor. If the mode is `mTime`, the `DelayBetweenMeasurements` is the number of microseconds.
- `BeginMeasurementStreamEvent` – The function to call when beginning measurements in stream.
- `MeasurementStreamEvent` – The function to call when a new measurement is received.
- `FinishMeasurementStreamEvent` – The function to call when finishing measurements in stream.
- `Edges` – Trigger on rising edge.

Returns

`True` if successful. Otherwise, returns `False`.

Example

```
BYTE MeasurementStreamEvent(int nMeasurement, TMeasurement* Measurement)
{
 printf("Distance: %f Snr: %hd BitSet: %hd n: %i\n",
        Measurement->Distance, Measurement->Snr,
        Measurement->BitSet, nMeasurement);
 return !kbhit(); // wait until a key is hit
 }
 ...
 SetUpProbeParameters();
 /* Measure after every 10th rising edge (Dilution factor - 1), reporting the
 reading immediately. */
FifoReadMeasurementStream (10, mPulse, NULL, MeasurementStreamEvent, NULL,
eRising);
```

```
...
```

Note

`BeginMeasurementStreamEvent` and `FinishMeasurementStreamEvent` parameters may be NULL. `MeasurementStreamEvent` function returns `0` to stop measuring in stream; otherwise, measuring in stream continues.

See Also

`FifoSetTimeOut` and `FifoSetTimeOutEx`.

4.7 Error Handling Functions

Error handling functions return information about an error that has occurred. These functions may be called after performing each OEM operation, in order to understand the error that occurred. Since the ConoProbe OEM can be used with any programming language, including Visual Basic, the exception mechanism is not used.

4.7.1 FifoGetError

Retrieves the status of the last error.

Prototype

```
TLoadError _stdcall FifoGetError();
```

Parameters

None.

Returns

Last error.

See Also

`FifoGetErrorString`.

4.7.2 FifoGetErrorString

Retrieves the last error information string.

Prototype

```
char *_stdcall FifoGetErrorString();
```

Parameters

None.

Returns

Error string.

Example

```
…
if (FifoGetError() != loeNone)
            printf ("Error reported : %s", FifoGetErrorString());
```

See Also

`FifoGetError`.

4.8 Timeout Handling Functions

Timeout handling functions set and return information about the timeout parameter. The timeout parameter determines how long the program waits for a response from the probe before displaying an error message. By default, this parameter is set for 3 seconds and is not persistent. FifoGetTimeOut and FifoSetTimeOut operate with a granularity of 1 millisecond and provide a time range from 1 millisecond to about 595 hours. FifoGetTimeOutEx and FifoSetTimeOutEx operate with a granularity of 1 microsecond and provide a time range from 1 microsecond to about 35 minutes.

One must exercise caution when using the timeout handling functions interchangeably. For example, if one sets a timeout value of less than 1000 microseconds with FifoSetTimeOutEx and uses FifoGetTimeOut to read the value, a value of zero will be returned since the smallest value returned by FifoGetTimeOut is 1 millisecond.

4.8.1 FifoGetTimeOut

Retrieves the communication timeout parameter (in ms).

Prototype

```
int _stdcall FifoGetTimeOut();
```

Parameters

None.

Returns

Timeout value.

Example

```
…
printf ("Communication timeout value is %d msec", FifoGetTimeout());
…
```

See Also

`FifoSetTimeOutEx, FifoSetTimeOut, FifoGetTimeOutEx`.

4.8.2 FifoSetTimeOut

Sets the communication timeout parameter (in ms).

Prototype

```
void _stdcall FifoSetTimeOut(int TimeOut);
```

Parameters

Timeout value.

Returns

None.

Example

```
…
/* Set communication timeout value to 500 ms */
FifoSetTimeOut (500);
…
```

See Also

FifoSetTimeOutEx, FifoGetTimeOut, FifoGetTimeOutEx.

4.8.3 FifoGetTimeOutEx

Retrieves the communication timeout parameter in microseconds (µs).

Prototype

```
long _stdcall FifoGetTimeOut();
```

Parameters

None.

Returns

Timeout value.

Example

```
…
printf ("Communication timeout value is %d µsec", FifoGetTimeOutEx());
…
```

See Also

FifoSetTimeOutEx, FifoSetTimeOut, FifoGetTimeOut.

4.8.4 FifoSetTimeOutEx

Sets the communication timeout parameter in microseconds (µs).

Prototype

```
void _stdcall FifoSetTimeOutEx(long TimeOut);
```

Parameters

Timeout value.

Returns

None.

Example

```
…
/* Set communication timeout value to 500 µs */
FifoSetTimeOutEx (500);
…
```

See Also

`FifoGetTimeOutEx, FifoSetTimeOut, FifoGetTimeOut.`

4.9 New FIFO Functions

The following functions are only available in the ConoProbe Mark III and other SMART family probes.

4.9.1 FifoGetFrequency

Return the probe's CCD frequency. The frequency can range from 50 Hz to 875 Hz in the Mark III Standard version and 3000 Hz in the Mark III Extended version.

Prototype

```
int _stdcall FifoGetFrequency ();
```

Parameters

None.

Returns

Current frequency of probe in Hz.

Example

```
…
printf ("The current probe frequency is %d Hz", FifoGetFrequency());
```

See Also

`FifoGetMaxFrequency, FifoSetFrequency` and `FifoBeginReadMeasurements.`

4.9.2 FifoGetProbeInfo

Queries the probe for its version information and serial number.

Prototype

```
TProbeVersionInfo _stdcall FifoGetProbeInfo ();
```

Parameters

None.

Returns

TProbeVersionInfo structure.

Example

```
WORD Port = 0;
TProbeVersionInfo probeVesionInfo;
if (!FifoInit(Port))
{
        /* obtain probe information */
        probeVesionInfo =  FifoGetProbeInfo ();
        printf("Probe SN = %d", probeVesionInfo.SN);
        printf("Probe Software Version = %d.%d",
        probeVesionInfo.Version_HIGH_SW, probeVesionInfo.Version_HIGH_SW );
        printf("Probe Hardware Version = %d.%d",
        probeVesionInfo.Version_HIGH_HW, probeVesionInfo.Version_HIGH_HW );
 }
```

See Also

TProbeVersionInfo.

4.10 ConoProbe API Functions Supported by the Mark III

Function	Supported in Mark III	Notes
FifoDetect	✓	Obsolete, returns 1
FifoInit	✓	
FifoInitEx	✓	Accepts custom Probe IP Addresses Mark III ONLY
FifoIpInit	✓	Accepts custom IP Addresses for the Probe and Host Computer Mark III ONLY
FifoGetMcuVersion	✓	Returns internal software version in Mark III
FifoSetFrequency	✓	
FifoSetPower	✓	
FifoSetDilution	✓	Sets pulse dilution factor Mark III ONLY
FifoSetInBitSet		Not relevant in Mark III
FifoSetLensZTweakUser		Not relevant in Mark III
FifoSetMeasurementFilter	✓	
FifoTerminate	✓	
FifoGetHeadSN	✓	
FifoGetInBitSetAvailability		Not relevant in Mark III
FifoGetLensZTweakUser		Not relevant in Mark III
FifoGetMaxFrequency	✓	
FifoGetMaxFrequencyEx	✓	Same as FifoGetMaxFrequency
FifoGetMCUVersion	✓	
FifoGetMCUVersionRequired	✓	Same as FifoGetMCUVersion
FifoGetPower	✓	
FifoGetFrequency	✓	New FIFO Function returns current frequency of probe
FifoGetActiveLensIndex	✓	
FifoSetActiveLensIndex	✓	
FifoGetLensCount	✓	
FifoGetLens	✓	
FifoSetLens	✓	
FifoReadMeasurementStream	✓	Rising edges only Supported

FifoAbortMeasurements	✓	
FifoReadMeasurement	✓	
FifoBeginReadMeasurements	✓	Rising edges only Supported
FifoGetMeasurements	✓	
FifoBeginReadMeasurementStream	✓	Rising edges only Supported
FifoGetMeasurementStream	✓	Same as FifoReadMeasurement
FifoGetMeasurementsCount	✓	
FifoGetError	✓	
FifoGetErrorString	✓	
FifoSetTimeOut	✓	Represents ETHERNET communication timeout in ms
FifoGetTimeOut	✓	Represents ETHERNET communication timeout in ms
FifoSetTimeOutEx	✓	Represents ETHERNET communication timeout in µs – Mark III ONLY
FifoGetTimeOutEx	✓	Represents ETHERNET communication timeout in µs – Mark III ONLY
FifoGetProbeInfo	✓	Returns information on the Probe – Mark III ONLY
ShowProbeDialog	✓	Must have Fifo_PS.DLL.

4.11 Other Functions

4.11.1 ShowProbeDialog

Displays a Probe dialog. You can customize the Probe dialog by adding a Windows message processing callback function and setting the parameter `Parameters->PlgProc` to point to your function.

The Probe dialog helps the user set up the object for optimal scanning. The Probe dialog displays the parameters of the current lens and single point scan results with SNR, and the user can try to improve the scan quality by changing the power level and/or the distance between the probe and the object.

Prototype

```
BOOL _stdcall ShowProbeDialog(BYTE Modal, TParameters *Parameters, TResults
*Results);
```

Parameters

`Modal` –This parameter is no longer used, and is provided for backwards compatibility only. Set to `True`.

`Parameters` – pointer to `TParameters` structure.

`Results` – pointer to `TResults` structure.

Returns

`True` if user clicks OK. Otherwise, returns `False`.

Example

```
int _stdcall MyDlgProc(void *hWnd, int Msg, int wParam, int lParam) /* Callback
function */
{
 switch (Msg) {
 case WM_CREATE:
        SetWindowText(hWnd, "Customized Probe dialog");
        break;
        }
        return 1;
 }
 TParameters Parameters = {lEnglish, uMM, gtRawAB, 1, 1, 1, 15, 0, 400,
 MyDlgProc};
 TResults Results;

 if (FifoInit(FifoDetect()))
 if (ShowProbeDialog(1, &Parameters, &Results)) /* user clicks OK */
        printf("Adjusted power: %d\n", Results.Power);
```

See Also

`FifoInit` and `FifoSetActiveLensIndex`.

5 The Smart Probe Tester Program

5.1 Introduction

This chapter explains how to use the Probe Tester program to experiment with the probe and to acquire measurements. The entire source code for the application is included in the SDK folder of the OEM CD so that one can gain a better understanding of how to use the API. The user can also use the source code as basis for a custom probe application project. In order to successfully build the project, one needs to use Visual Studio 2005 C++. One can obtain the express edition of Visual Studio 2005 free from the Microsoft website.

The Main Screen – Describes the functions associated with the main screen of the application.

Using the Probe Screen Dialog – Describes how to use the probe screen dialog to obtain optimal measurements.

Start of Measurement Output – (also called ROG) describes how to use the Start of Measurement output signal to synchronize the current stage location with the measurement

5.2 The Main Screen

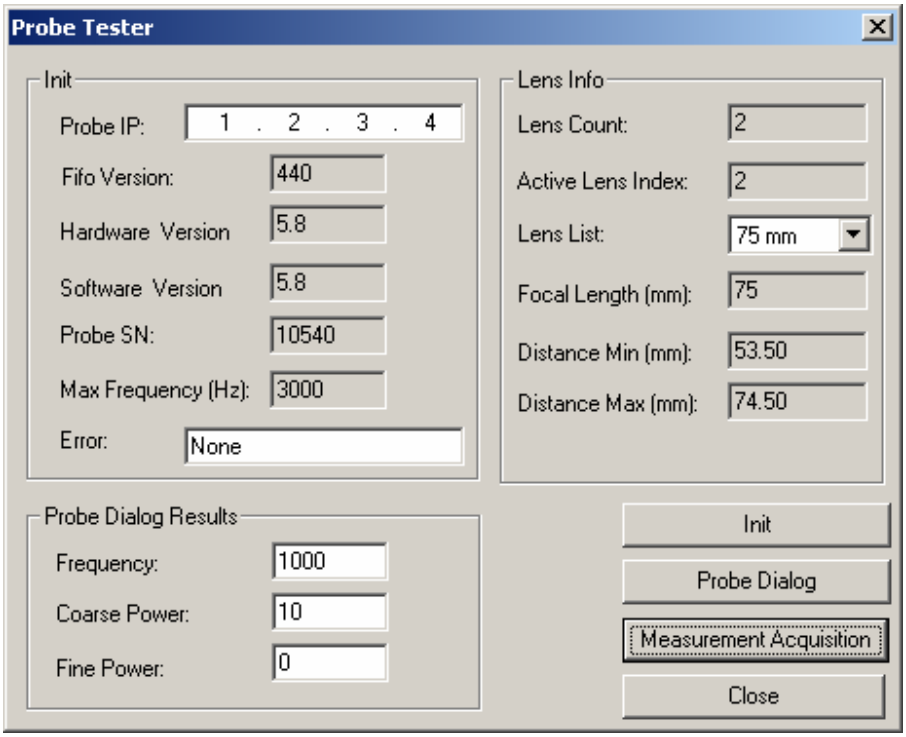


Figure 5-1: The Probe Tester Main Screen

The **Figure 5-1: The Probe Tester Main Screen** shows the main screen of the Probe Tester. To start the application and to initialize the probe, press the <Init> button. If there is a problem in the

initialization, (for example, the probe is not connected to the communication box), an error message will appear in the Error box. Otherwise, the various probe parameters will be displayed in their respective fields. A summary of the fields is shown below:

In the Init window:

Fifo Version – Version of Fifo.DLL being used.

Probe IP – IP Address being used to communicate with the probe.

Max Freq – The maximum frequency supported by the probe.

Probe SN – The serial number of the probe.

In the Lens window:

Lens Count – The number of lenses in the probe (including the "0" lens).

Active Lens Index – The index of the active lens

Lens List – List of all the lenses in the probe by focal length.

Focal Length – The focal length of the active lens.

Distance Min – The minimum distance limit of the focal range of the active lens.

Distance Max – The maximum distance limit of the focal range of the active lens.

5.3 Changing the Active Lens

The active lens of the probe can be clicking the desired lens index in the lens list box. If the operation is successful, the new active lens will be highlighted in the list box and the lens parameters will be updated to those of the active lens.

5.4 Using the Probe Screen Dialog

Before one acquires measurements using the Probe Tester program, the signal coming from the probe should first be analyzed with the probe dialog. Based upon the quality of the signal, the power, frequency, and probe distance should be changed to obtain the best measurements. Below is a discussion of how to understand the graphical output of the probe dialog in order to make these corrections.

When scanning an object, you must make sure that the signal quality is good and that you are receiving optimal scan results. The Probe screen may be used to adjust the probe settings in order to control and analyze the signal quality. This section describes the Probe screen and its components.

Figure 5-2 displays the probe screen showing a probe output signal with high SNR from an object whose distance is 63.808 mm from the probe.

In the signal display area, the X-axis of the graph represents the pixel number of the CCD Detector, and the Y-axis represents the units of intensity. The signal's frequency and wavelength depend on the distance between the probe lens and the object.

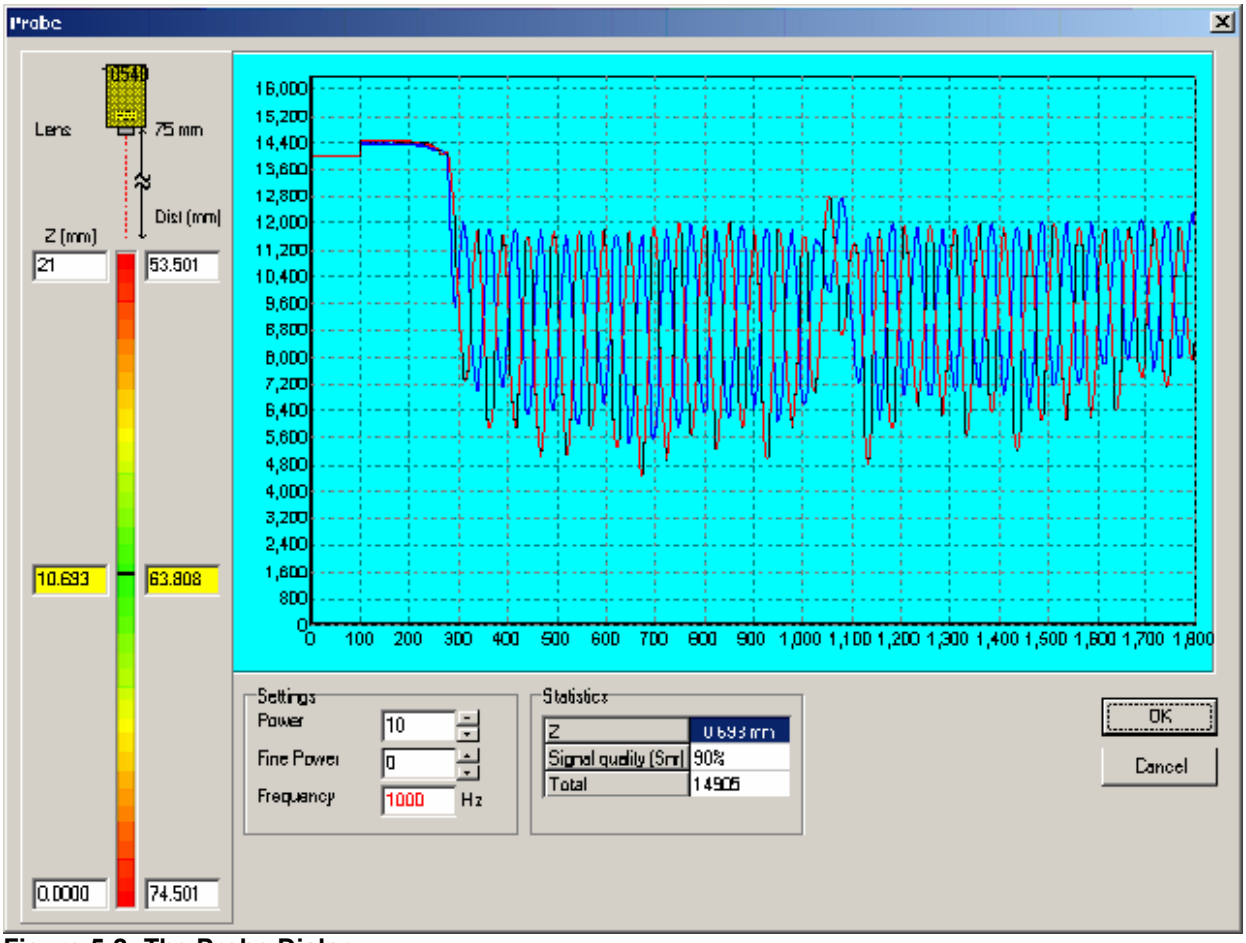


Figure 5-2: The Probe Dialog

The Probe screen is divided into several sections:

5. **Vertical Distance Scale** – Located on the left side of the Probe screen. For more information about the vertical distance scale, refer to **Section 5.4.5**
6. **Signal Display** – Located on the upper right-hand side of the Probe screen. For more information about the signal display, refer to **Section 5.4.1**
7. **Statistics Section** – Located under the signal display to the left. For more information about the *Statistics section*, refer to **Section 5.4.2.**
8. **Settings Field** – Located to the right of the Statistics section. For more information about the Settings section, refer to **Section 5.4.3.**

To return to the main window, click **OK** or **Cancel** in the Probe screen.

5.4.1 The Signal Display

Scattered light reflected off the object enters the probe's optical system and is detected by two CCD detectors inside the probe.

The signal display allows you to analyze and optimize the signal quality. This section explains how to read and understand the signal display in order to analyze the signal quality.

You can enlarge a section of the signal display graph for easier viewing. To enlarge a section of the graph,

hold down the left mouse button and drag the cursor from the top-left to the bottom-right of the section you want to enlarge.

To zoom out, hold down the left mouse button and drag the cursor over the graph from bottom-right to top-left.

5.4.2 Statistics Section

Table 5-1: Statistics Section Fields describes the fields in the Statistics section of the Probe screen.

Table 5-1: Statistics Section Fields

Field	Description
Z	The height of the object.
Signal quality (Snr)	Signal to noise ratio. The percentage of the signal's power out of the total reading (Signal/(signal+noise)).
Total	The estimated total amount of light in the picture.

5.4.3 Settings Field

Table 5-2 Settings Section Fields describes the fields in the Settings section of the Smart Probe screen.

Table 5-2 Settings Section Fields

Field	Description
Power	Valid power settings range from 0 to 63. One power unit is equivalent to approximately 40-50 fine power units. For most applications, set the power level above 10 for more accurate measurements (refer to **Section 3.5.5.3** Setting laser power level)
Fine power	Fine-tune the power level to achieve a better Snr. Effective at low Power levels 6-11.
Frequency	The Working Frequency sets the CCD detectors' exposure time for every measurement. The exposure time equals 1/frequency.

5.4.4 Adjusting the Smart Probe Settings

You can adjust various probe settings to ensure that the object is within the working range, and the power level and Working Frequency are set correctly.

5.4.4.1 Adjusting the Smart Probe Height

You may have to adjust the height of the probe so that the object is within the working range of the lens installed. To position the probe at the correct distance from the object:

- Increase the power level until the laser spot becomes visible. This can be done by adjusting the power in the Settings section of the Probe Screen.
- Move the object so that the probe is over a relatively flat portion of the object between the highest and lowest points you want to measure. When the probe is over the object, a red dot appears on the object. The red dot marks the exact point the probe is measuring.

- Move the probe up or down until the distance from the lens to the point measured on the object is approximately the standoff of the lens. As you move the probe up and down, watch the size of the red dot on the object. If the red dot gets smaller and more focused, the lens is moving towards its optimum distance. If you do not see the laser spot, increase the laser power.
- If the Probe Screen is activated, you can fine-tune the probe height after the laser spot is focused, using the vertical distance scale on the Probe Screen.
  If a valid reading is indicated, but the measured distance moves in the opposite direction than expected (i.e., as you move the object closer to the probe, the cursor moves lower on the scale in the Probe Screen), you are in the region of the reversed fringes. Increase the probe height until you are within the normal range. This may occur with 25 and 50mm lenses
- Place the object as close as possible to the center of the working range. If the cursor is not approximately at the center of the range, adjust the height of the probe. If you change the height of the probe, readjust the power level.
- Move the object so that the laser beam is on the highest point that you want to measure on the object and check the distance by using the vertical distance scale in the probe screen to make sure the object is still within range. Then move the object so the laser beam falls on the lowest point that you want to measure on the object and check that the object is still within range. If necessary, adjust the probe's height so that all areas of the object that you want to measure are within range.

5.4.4.2 Setting the Working Frequency (Data Acquisition Rate)

Set the Working Frequency according to the following guidelines:

- In general, it is preferable to work with the highest Working Frequency possible, since measurement error can be minimized by better use of averaging filters.
- As a general rule, for shorter focal lengths - use higher working frequencies, for darker (light absorbing) surfaces and higher angles-use lower frequencies. Start with 1000 – 1500 for lenses up to 75 and 500-700 for lenses from 100mm up.
- If you are acquiring measurements with a moving table, then calculate the correlation working frequency – table velocity in such a way to allow sufficient measurement points on the desired feature.

5.4.4.3 Setting the Laser's Power Level

In general, there is a direct relationship between the Working Frequency and the corresponding required laser power. This means that if you are working at high CCD frequencies you will require a higher laser power level, and if you are working at low CCD frequencies you will require a lower laser power level.

The laser power level should be set at 11 or higher. When the power level is below 9, the wavelength is unstable. This may cause an unacceptably high uncertainty of measurement.

To configure for optimal signal quality, you should set the initial laser power to 20 when working with lenses greater than 25 mm. When working with 25 mm lenses, set the initial laser power to 11. Gradually change the power while monitoring the following parameters in the **Statistics Section.**

.

- **Total** is within the acceptable range between 1200 and 16,000; in extreme cases (objects with different surface finishes, steep angles, etc.) values from 900 to 18000 will allow a fair measurement. If the Total is too low, then increase the power level. If the Total is too high,

then decrease the power level.

- **SNR** is at its' maximal value over the range of the power.

NOTE:

When scanning with or 25mm lens, on diffusive white materials, it may be necessary to scan using a power level lower than 11.

To set the laser's power level:

- In the Settings section of the Probe screen (see **Figure 5-3**), set the Power field to 0.
- Increase the power while monitoring the SNR. Continue to increase the power level until the SNR decreases, and then decrease the power by one level. Repeat the process to set the Fine power field.

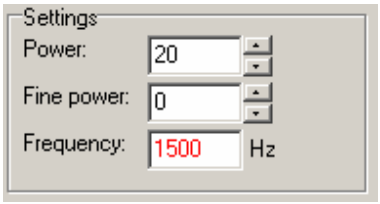


Figure 5-3 Settings Section of the Probe Screen

NOTES:

- If the SNR begins to decrease only when the power level is less than 11, increase the Working Frequency and repeat the process.
- This procedure is sufficient in most cases. However, for optimal tuning of the system, results obtained with the SNR reading are not sufficient and you must also use the signal display. For more information about using the signal display, refer to **Section 3.4** *Optimizing Signal Quality*

5.4.5 Using the Vertical Distance Scale

Use the vertical distance scale to ensure that the object is within the working range of the lens. The vertical distance scale provides a graphical and digital display of the lens' working range and the object's position within the working range. **Figure 5-4** displays the vertical distance scale.

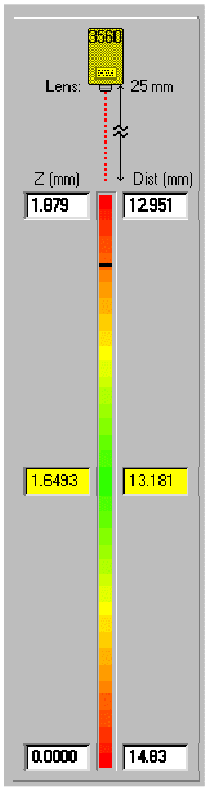


Figure 5-4: Probe Screen - The Vertical Distance Scale

The numbers on the left side of the vertical distance scale represent the working range of the lens. The bottom number represents the furthest point (from the lens) in the working range, and is always 0. The top number represents the length of the working range. The middle number represents the actual distance from the object's surface at the point where the laser beam is directed and the bottom of the working range. If this point is not within the working range, the message Out of Range appears across the screen.

The numbers on the right side of the vertical distance scale represent the probe location relative to the working range. The bottom number represents the farthest distance that the lens can measure. The top number represents the nearest distance that the lens can measure. The middle number represents the absolute distance from the lens to the object's surface at the point where the laser beam is directed.

The location of the object's surface is indicated graphically by a horizontal cursor in the middle column.

For example, in **Figure 5-4:** Probe Screen - The Vertical Distance Scale, the working range is 1.879 mm and is located between 14.83 mm and 12.951 mm from the probe. The probe is currently directed at a point in the object surface that is near the top of the working range, 13.181 mm from the probe.

Move the object so that the laser beam is on the highest point that you want to measure on the object and check the vertical distance scale to make sure the object is still within range. Then move the object so the laser beam falls on the lowest point that you want to measure on the object and check that the object is still within range. If necessary, adjust the probe's height so that all areas of the object that you want to measure are within range.

Place the object as close as possible to the center of the working range. When the laser beam falls on a point halfway between the highest and lowest points you want to measure, the cursor in the vertical distance scale should be approximately at the center of the range. If the cursor is not approximately at the center of the range, adjust the height of the probe. If you change the height of the probe, readjust the power level.

5.4.6 Optimizing Signal Quality

The ideal signal would be sinusoidal except in its center area. The Z values are calculated from the dominant frequency of the signal's FFT. As the signal's resemblance to a sine increases, the measurement precision increases.

To optimize the signal quality, follow these guidelines:

- Strengthen the signal. Always try to have the strongest signal without saturation. You can control the signal strength through the power and Working Frequency settings. First adjust the laser power. If this is inadequate, adjust the Working Frequency. For more information about adjusting the power and Working Frequency settings, refer to **Section 5.4.4 Adjusting the Smart Probe settings**

- Consider the object's surface and set the probe settings accordingly. For more information on various surface types, refer to **Section 5.4.7.11: Considering the Object's Surface.**

5.4.7 Analyzing the Signal

Analyze the signal quality to ensure that the power level, Working Frequency, and other parameters are set correctly, as well as to ensure that the probe is positioned correctly in relation to the object. You can analyze the signal quality:

- Using the Signal to Noise Ratio (SNR).
- Using the Total values

5.4.7.1 Using the Probe Screen Graph

The probe screen graph displays the raw signals detected by the CCDs. This graph is very useful in analyzing the signal and improving signal quality.

5.4.7.2 Ideal Probe Signal Graph

In an ideal probe signal graph, the channels appear as two perfect sine waves that are out of phase by 180°. The **Figure 5-5** shows an example of a good signal.

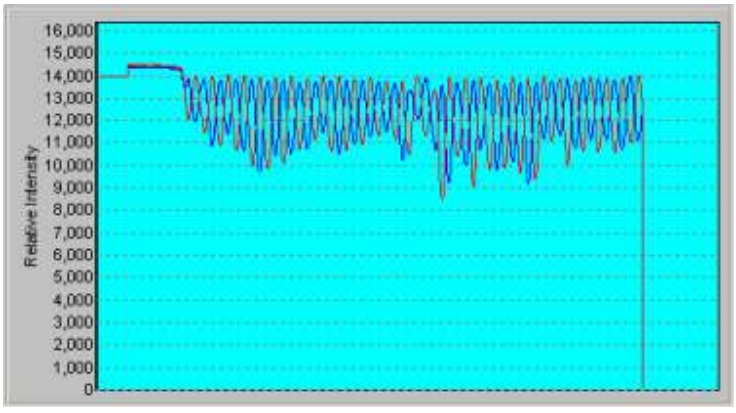


Figure 5-5: An Example of a Good Probe Signal

5.4.7.3 Saturated Signal

A saturated signal occurs when the laser power level is set too high. In a saturated signal, the bottom of the wave is clipped, as shown below in F**igure 5-6**. To improve a saturated signal, reduce the power level. However, if you are measuring a reflective surface, a semi-saturated signal results in the best SNR ratio.

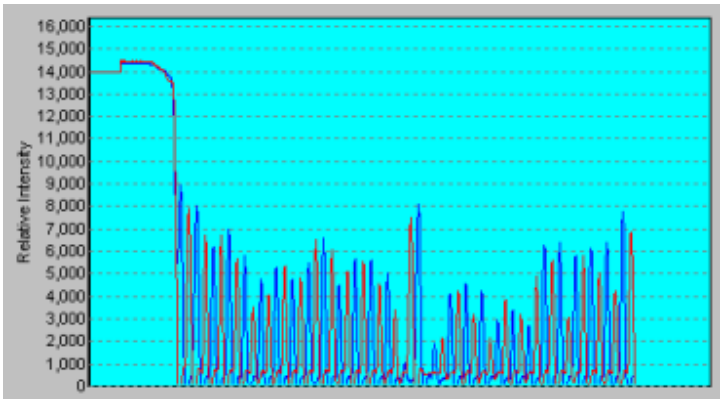


Figure 5-6: A Saturated Signal

5.4.7.4 Fully Saturated Signal

A fully saturated signal occurs when the laser power is too high and/or the Working Frequency is too low. When the signal is fully saturated, the SNR ratio is very low (close to 0) and an out-of-range message appears on the vertical distance scale. To improve the signal, lower the power level and/or raise the Working Frequency. The **Figure 5-7** illustrates a fully saturated signal.

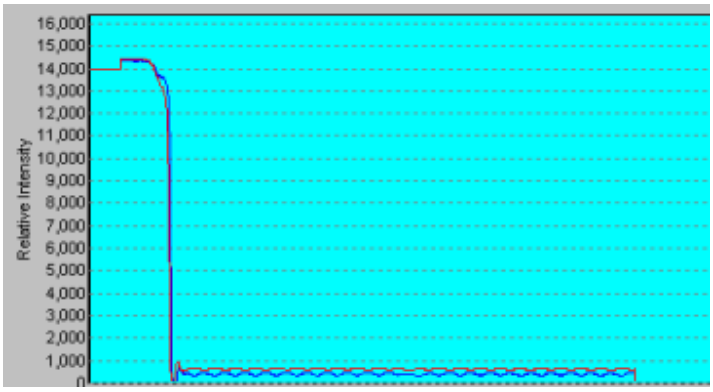


Figure 5-7: A Fully Saturated Signal

5.4.7.5 Low-Level Signal

A low-level signal occurs when the amount of light reflected off the object is not enough for the CCD detectors. This may occur if the object's surface has low reflectivity or if the object's surface scatters the laser light (like the reflection from a mirror-like ball). To improve the signal, raise the power level and/or lower the Working Frequency. The F**igure 5-8** illustrates a 'flat', low-level signal.

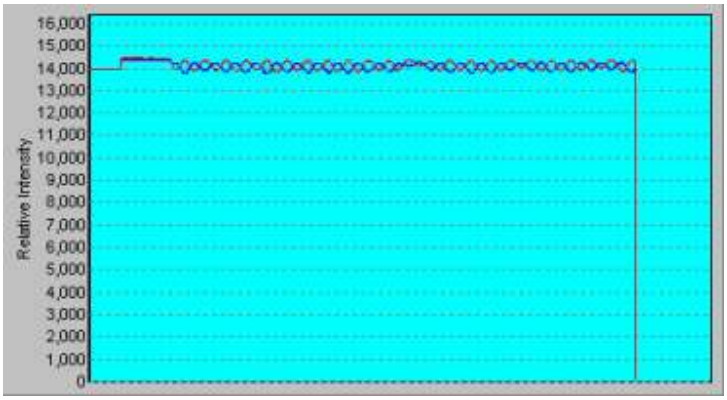


Figure 5-8: A Low-Level Signal

5.4.7.6 Signal due to multiple reflection or split last spot

A signal due to multiple reflection or split spot (different distances measured simultaneously) is shown in F**igure 5-9.**

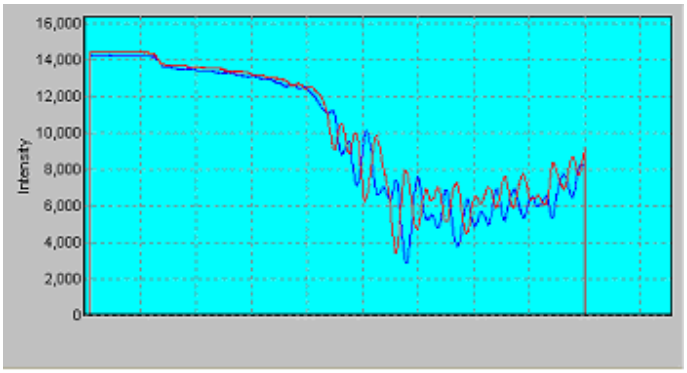


Figure 5-9: Probe signals from multiple reflection or split laser spot

5.4.7.7 Smart Probe too close to object

The Frequency graph, as well as other indicators, may indicate that the probe is too close to the object you are measuring. If this is the case, increase the distance between the probe and the object until the object is within working range. The following circumstances indicate that the probe is too close to the object:

- The signal graph looks similar to the graph displayed in F**igure 5-10**. The signal is wider and starts before pixel 300.
-
- Using the Vertical Distance Scale
- .There is a low SNR value
- Numbers appear on the vertical distance scale, but the cursor in the middle of the scale moves in the opposite direction than expected (i.e., as you move the object closer, the cursor moves lower on the scale). These numbers are not correct. As you move the probe further, an out-of-range message appears and then the correct numbers begin to appear on the scale. For more information about the vertical distance scale, refer to **Section 5.4.5.**

.

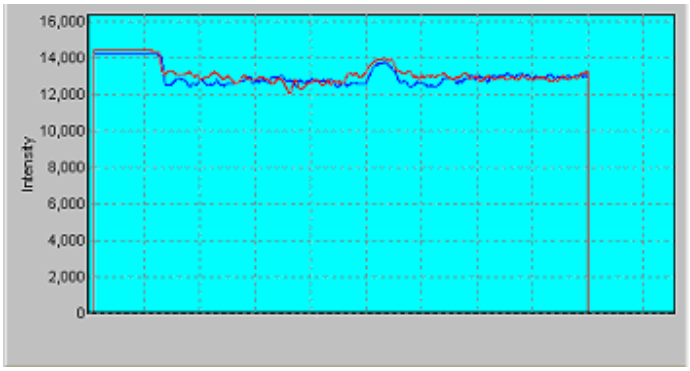


Figure 5-10: Probe too close to object

5.4.7.8 Probe too far from object

The Frequency graph, as well as other indicators, may indicate that the probe is too far from the object you are measuring. If the probe is too distant from its target, decrease the distance between the probe and the object until the object falls within the working range. The following circumstances indicate that the probe is too far from the object:

- The signal graph looks similar to the graph displayed in **Figure 5-11**. The signal looks 'narrow' and starts beyond pixel 500.
- An out-of-range message appears on the vertical distance scale. As you move the probe further and reenter the working range of the lens, the applicable numbers reappear on the vertical distance scale. For more information about the vertical distance scale, refer to *Using* the Vertical Distance Scale
- There is a low SNR value.

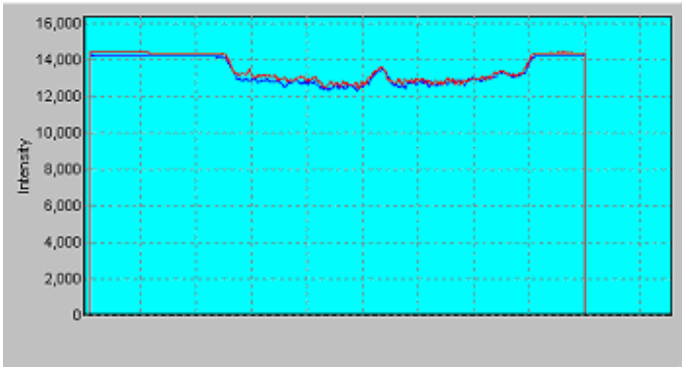


Figure 5-11: Probe too far from object

5.4.7.9 Using the Signal to Noise (SNR) Ratio

In the Probe screen, the Signal quality (SNR) field represents the signal to noise ratio. SNR characterizes the measurement quality and may range from 0% to 100%. A high SNR value indicates high signal quality, which results in high measurement quality. Generally, a SNR value below 30% indicates a not reliable measurement, and a SNR value above 50% indicates accurate measurement results.

NOTE

Values between 30% and 50% are acceptable only if they are consistent over the entire object to be measured. This may occur in the case of semitransparent or absorbing materials, usually plastics or ceramics; use of objectives with large focal length (75mm and up) is recommended.

5.4.7.10 Using Total

The Total represents a value proportional to the area limited by the signal envelope and increases with signal intensity. Signals with small total values may not be as accurate as expected, even though the SNR is high. Recommended values: 2000 to 16000, though values of 1200 to 18000 may still yield accurate results in most cases.

5.4.7.11 Considering the Object's Surface

The optical properties of different surface types can affect the measurement results. Following are some guidelines on the optical properties of various surface types.

- Materials that scatter light tend to provide good, even signals. Some examples are paint, molded plastics, and matte surface metals.
- Light-colored materials reflect more light than dark-colored materials and therefore need a lower power setting. Adjust the laser power accordingly.
- Reflective surfaces with simple contours (flat surfaces, simple angles, etc) may be difficult to measure. For best results, try tilting the object so that it forms an angle of about 0.5-2.5 degrees around an axis parallel to the probe's bottom plane. This deflects the reflected beam away from the instrument's optical axis. With partially reflective surfaces, it is also recommended to eliminate bad points in the scan.
- Reflective surfaces with complex contours (waves, holes, etc.), and translucent surfaces, cannot be measured unless coated with paint or talcum powder. To coat a purely reflective surface, do one of the following:
- Apply a thin, flat layer of white spray paint to the surface. Apply the layer in several very thin coats until the surface is about half coated. It is not necessary to cover the surface completely. If the spray paint is carefully applied, the added thickness should be less than 1.5 µm and the measurement results are only affected to the extent of 0.5 - 1 µm.
- Apply a non-permanent coating by dusting the object very lightly with talcum powder (baby powder) or spraying it with a spray spot remover, which also applies a thin coat of talc to the surface. If done carefully, this method applies a coat with a thickness of about 1 µm.
- The angle formed by the object's surface and the laser beam has a strong effect on the quality of measurement. When the object's surface is non-reflective, the probe can measure the object at an angle of up to 85 degrees. If the object's surface is reflective, the maximum angle is smaller. Try to keep from exceeding these measurement angles wherever possible.

  If you are measuring an angle, rotate the object to find an angle that results in a good signal throughout the scanning area
- When scanning with a 25mm lens, on diffusive white materials, it may be necessary to scan using a power level lower than 11. Note that the resultant measurements may not be

stable over time.

- When scanning non-homogeneous surfaces with both diffusive and reflective components (for example, an object that is black and white, or an object that is partly reflective and partly non-reflective), you should perform multiple scans. Change the laser power level between scans so that every portion of the object is scanned at least once with the appropriate settings. Then combine the results into one full scan of the object using the highest SNR
- When using a lens with a working range that is smaller than the required height range, you cannot measure the whole object in a single scan. In this case, you can perform several scans at different heights in order to cover the entire required height range. You can set the reference heights for each scan as follows:
    - Set a reference height using an external encoder.
    - Measure a reference height on the object surface using the Smart.
    - Measure the reference height on an external reference surface (e.g. gage blocks) using the Smart.
    - After performing the multiple scans, the scans can be combined into a single image as follows:
    - Correct the Z values by adding the various reference heights.
    - For each point scanned, choose the optimal Z value from the different measurements, based on maximal SNR
- When measuring objects which return a very low signal intensity (for example: objects made of graphite or black rubber), and using a large lens (75mm and up), the sampling rate can be set as low as 20Hz, in order to increase the exposure time and the **Total** value
- When scanning features that generate multiple reflections (grooves, blind holes, etc.) it is recommended to scan in the direction of the probe base plane. This is due to the fact that the probe's optical aperture is smaller in this direction than in the perpendicular direction. In order to get better results or to eliminate bad measurements it is recommended to scan in the direction of the smallest aperture value, so that the parasite signals (from multiple reflections) are minimized.
  Note that the width of the spot size increases faster in this direction (parallel to base) than in the other direction when moving close to the working range limits and generally yields a higher measurement 'noise'.

5.5 Acquiring Measurements

After the parameters for a scan are set, one can perform a variety of measurement acquisition scans by pressing the Measurement Acquisition button. After this button is pressed, the Measurement Acquisition screen appears:

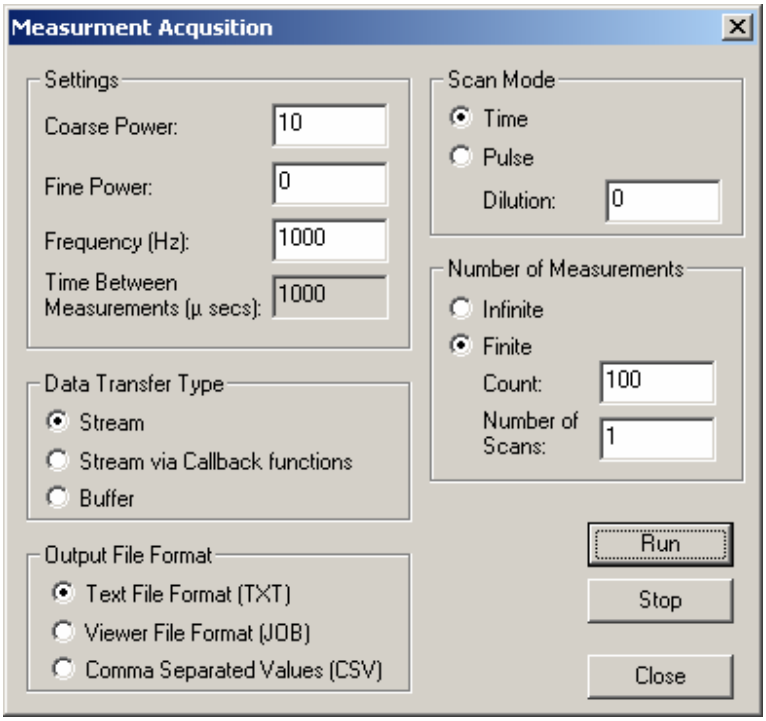


Figure 5-11: The Measurement Acquisition Dialog

Pressing the Measurement Acquisition button. After this button is pressed, the Measurement Acquisition dialog appears. See **figure 5-11**.

5.5.1 Time mode

In Time mode measurements are taken at preset time intervals and are controlled by an internal clock. The period between measurements is equal to 1/scan frequency.

5.5.2 Pulse mode

In Pulse mode measurements are acquired when triggered by external pulses generated by a motor encoder, wave generator or other source. A measurement is acquired at an interval of pulses determined by the dilution factor. Therefore, a dilution of 2, a measurement will be acquired at every other pulse. For more information about operating the probe in pulse mode, please refer to the OEM manual.

5.5.3 Number of Measurements

There are two possible ways of determining the quantity of measurements during a scan session.

- Infinite – Measurements will be continuously acquired in a single scan until the user presses the <Stop> button. This option only works for Stream or Stream via Callback functions mode and for only a single scan.
- Finite – A fixed number of measurements are acquired for each scan. The number of scans is entered in the *Number of Scans box*.

5.5.4 Data Transfer Type

There are three possible data delivery types that can be utilized to acquire measurements:

- Stream Mode – Measurements are acquired from a stream. After the desired number of measurements is acquired, FifoAbortMeasurements is called to stop the probe from measuring.
- Stream Mode via Callback Functions – Same as Stream mode, however the acquisition processed can be controlled by three events, *Begin* which is called at the start of measurement acquisition, *Measurement*, which is called every time a measurement is received, and *Finish* which occurs at the completion of the measurement acquisition.
- Buffer – Measurements are acquired in a separate thread in a buffer passed by the caller program.

5.5.5 Save File Format

After the scanning parameters have been set, the desired data transfer type, and mode, press the <Run>

Measurements can be saved in one of three formats

- Text – Saves measurements in standard text format.
- Job – Saves measurements in a format which one can use the viewer tool.
- CSV – Saves measurement in a format suitable for a viewing by in spreadsheet.

5.5.6 Performing an Acquisition of Measurements

After the scanning parameters have been set, the desired data transfer type, and mode, press the <Run> button to start the measurement acquisition process. After a successful run, the measurements will appear in the data format selected in the Save File Format box. When job file is selected for the saved file format, the viewer will be invoked and one can visually see and analyze the data. See **Figure 5-12** below:

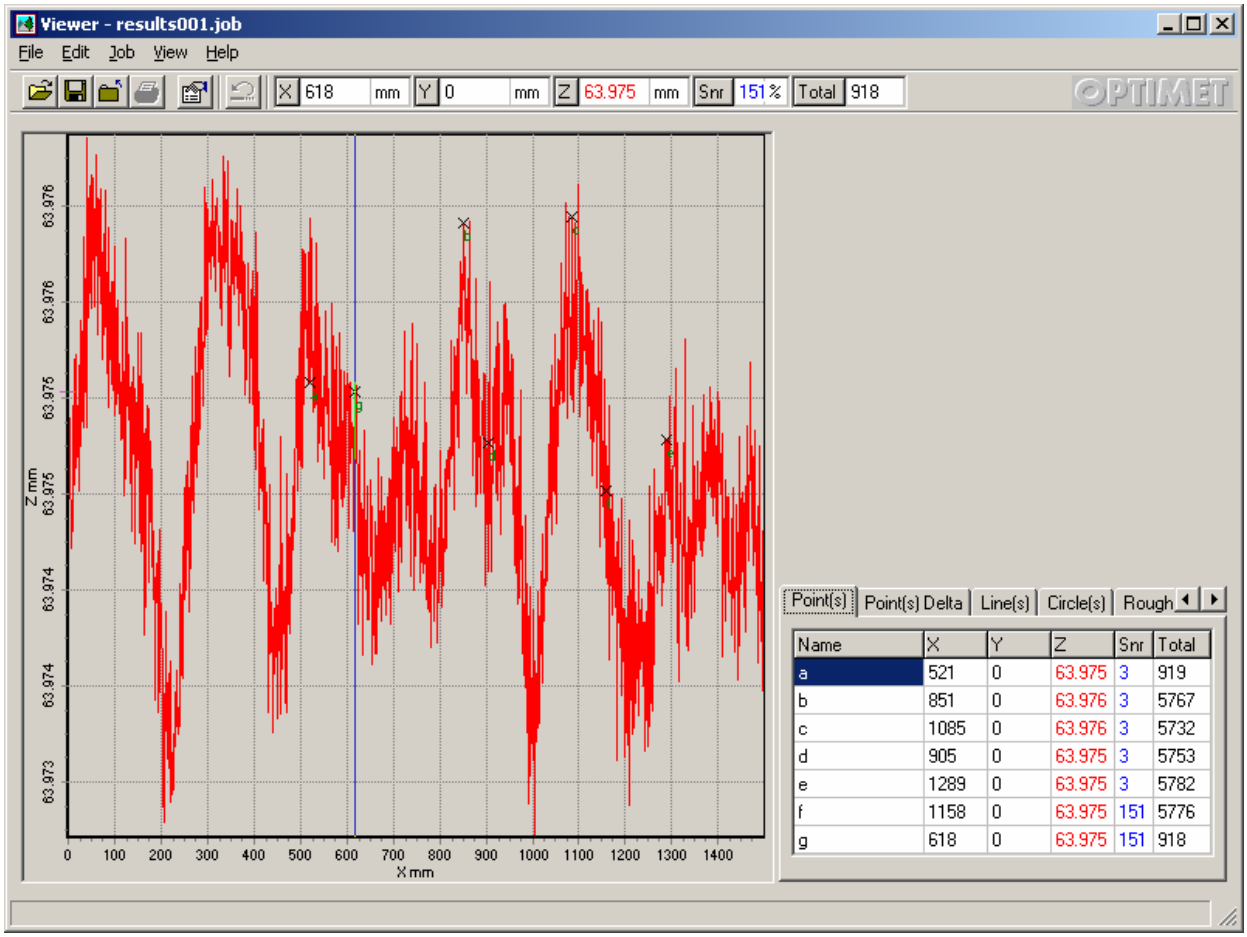


Figure 5-12: A job file being displayed by the Viewer application

For an-depth treatment on how to use the Viewer to analyze measurement data, please look at the Viewer help file.

6 The System Input/Output Description

6.1 Introduction

This chapter reviews the input/output discrete signals as follows:

External Trigger Input – Describes how to use external triggers to drive the probe.

Analog Output – Describes how to use analog output to obtain additional measurement information.

Start of Measurement Output – (also called ROG) describes how to use the Start of Measurement output signal to synchronize the current stage location with the measurement

6.2 External Trigger Inputs

The probe can be externally triggered to perform a measurement at a pre-set rate. This feature is used to synchronize the probe to the host system.

The probe can measure up to the maximum rate in any of the measurement modes. This section describes the hardware and software interface of the external trigger inputs.

6.2.1 Hardware Description

On the panel of the Communication Box, there is a connection used for an external trigger input. The input is a standard TTL level signal (common ground with the system).

Pin Number	Description
x	External trigger signal.
y	GND.

Table 6-1 Trigger pin assignments

6.2.2 External Trigger Signal

The external trigger is a TTL hardware level signal.

Triggering is established on the RISING-EDGE of the signal with a minimum pulse width of 10 μsec.

6.2.3 External Trigger Dilution of Signals

In addition the External trigger mode facilitates an option for diluting the incoming triggers.

.

Example 1: Using trigger input rising edges, with 200 Hz frequency (short positive pulses), and N=1. In this case, there are 200x1 active edges. Measurements are at 200 Hz.

Example 2: Using trigger input rising edges, with 10000 Hz frequency (short positive pulses), and N=20. In this case, there are 10000 active edges, and measurements are at 10000/20 =500 Hz.

6.3 Timing diagrams for the External Trigger

6.3.1 General sequence

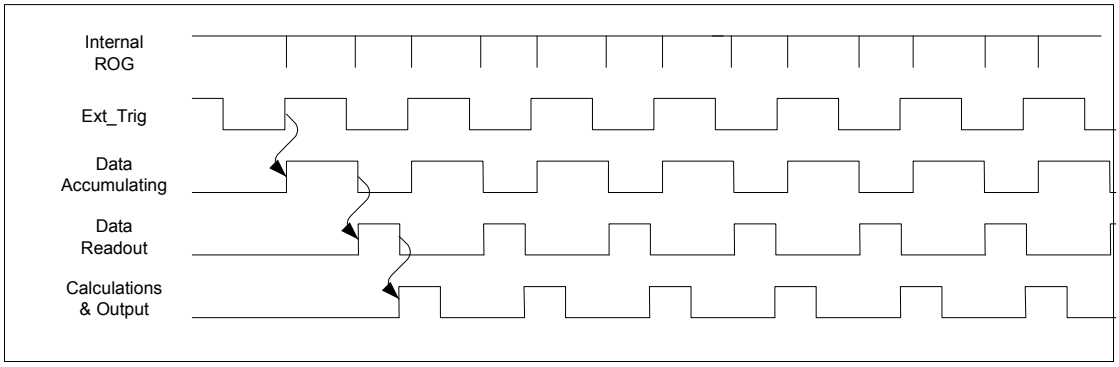


Figure 6-1 General sequence

6.3.2 Timing blow-up

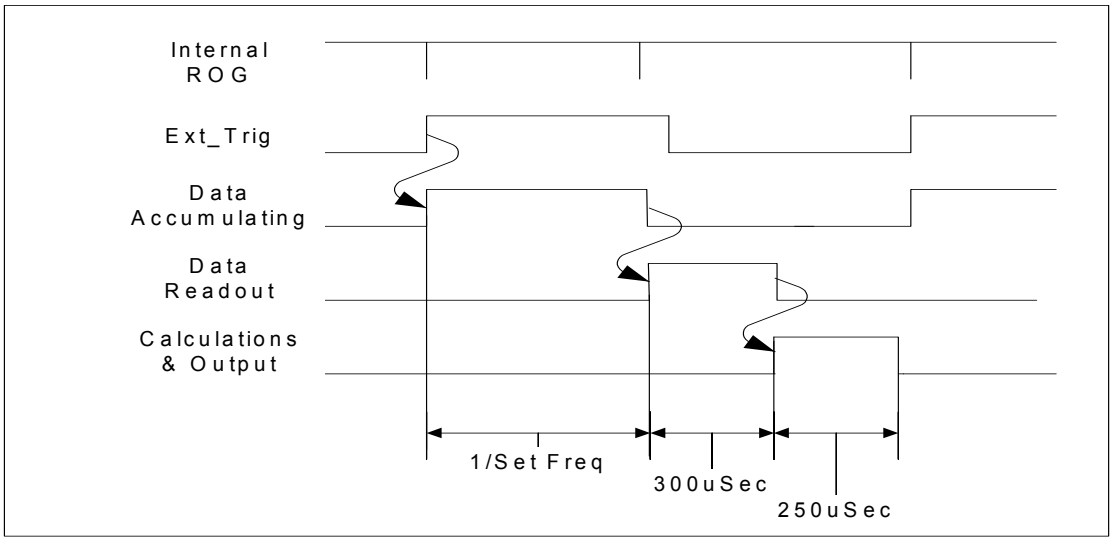


Figure 6-2 Timing blow-up

6.4 Analog Output

The MKIII features an analog output voltage that indicates the measured Z value. It does not provide other information about the measurement, such as the SNR Etc.

6.4.1 Analog Output Specifications

Table 6-1 below defines the analog output specifications.

This analog output is used for display only and should not be used to drive mechanisms.

Voltage Output, Z = Minimum	--5V +/- 5 mV (1).
Voltage Output, Z = Maximum	+5V +/- 5 mV (1).
Resolution (Bits)	11 (1).
Output uncertainty of measurement (relative to Z values)	W.R./1000 +/- 0.1% of differential height or actual used range (1) (2).
Output Noise Level	10 mV peak-to-peak (max) (1) (3).
Output Impedance	100 ohm.
Maximum Current Drive (normal operation)	1 mA (4).
Absolute Maximum Current Drive	20 mA (5).
Output Rise/Fall Time	200 us (0 to 99%, step change in voltage).
Initial Output Value (after power up)	0.0V +/- 5mV.

Table 6-2 Analog Output Specifications

NOTES:

(1) Analog output is generated using a 12 bit digital-to-analog converter, which yields an lsb of +/-2.4mV. For practical reasons 10mV should be considered the uncertainty of measurement.

(2) Accuracy refers to analog output voltage relative to the Z value measured by the system. The measurement accuracy of Z is not included in this value.

(3) Value depends on the measurement method and detection system employed and the environment in which the measurement is made (lens, material, speed).

(4) Maximum Current Drive is based on a 1% error deviation from calibrated voltage output.

(5) This refers to maximum limit to avoid damage to the output buffer.

6.5 Start of Measurement Output (ROG – Read Out Gate)

A pin located on the Communication Box labeled “ROG” facilitates the “start of measurement “signal.

In this section the term ‘measurement’ is defined as the integration of a light pattern on a CCD, which is used to calculate the distance from the probe to the object being measured. The CCD chip contains photodiode sensors, which accumulate an electrical charge proportional to the amount of light integrated. This electrical charge is then converted to an analog signal. The analog signal is outputted from the CCD chip and analyzed in the probe.

The measurement time is controlled by the signal ROG. When the ROG is high, the analog signal reflecting the previous measurement is outputted, and the CCD integrates the light entering the photodiode sensors for the next measurement. When the ROG is low, the photodiode sensors are discharged. (Note that the photodiode sensors continue to integrate light even when the ROG is low). Therefore, the new measurement begins and the previous measurement ends when the ROG goes high.

The ROG is high for approximately 1/Sampling Frequency. The Sampling frequency is in the range of 50 – 3000 Hz; therefore, ROG is high for between approximately 1/freq ms. ROG is low for 5.0 µs.

Figure 6-3 contains a timing diagram of the ROG signal:

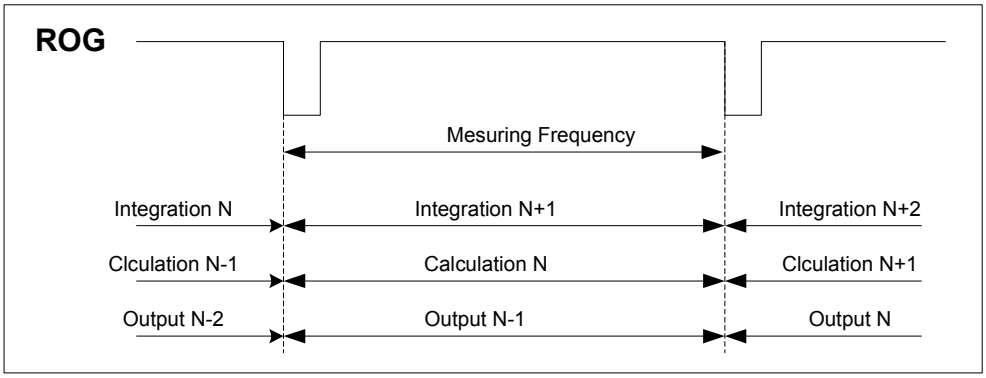


Figure 6-3 Timing Diagram of the ROG signal

In order to prevent saturation of the photodiode sensors, they are constantly being emptied. In other words, the ROG signal is active all the time, even when no measurements are requested via the software. A modified ROG signal (called START_OF_MEASUREMENT) is generated from the ROG signal. START_OF_MEASUREMENT is active only for measurements requested by the user.

Figure 6-4 contains a timing diagram of the START_OF_MEASUREMENT signal:

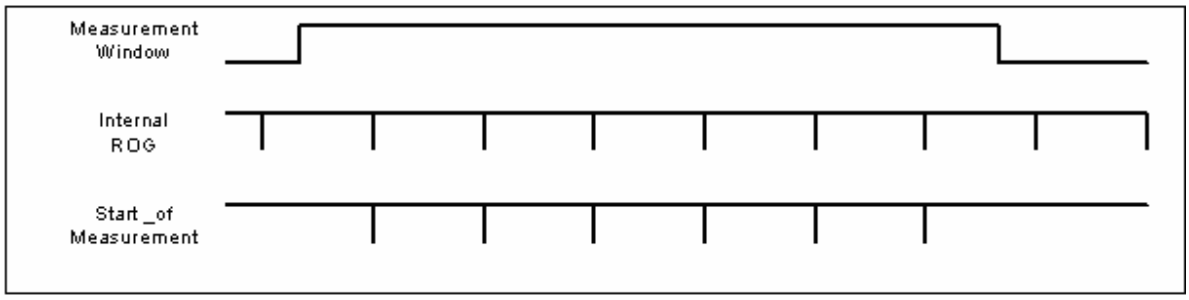


Figure 6-4 Timing Diagram of the START_OF_MEASUREMENT signal

It is anticipated that the customer hardware will latch the current location of the stages when it detects a rising edge on the START_OF_MEASUREMENT signal. This information will be used to provide the precise location of the measurements.

NOTES:

- START_OF_MEASUREMENT is a standard 5V TTL output. The reference GND voltage to this signal may be taken from the communication box connection.
- The first START_OF_MEASUREMENT pulse is extraneous and should be ignored.
- The measurement window is active until after the end of the final measurement. Therefore there are likely to be additional START_OF_MEASUREMENT pulses at the end of the scan. These additional START_OF_MEASUREMENT pulses should be ignored.

6.6 External Trigger Mode Operation

When driving the ConoProbe Mark III in external trigger mode, missing or incorrect measurements may occur. Readings of zero distance or measurements that were expected but never registered indicate missing or incorrect measurements. These errors may be caused by problems with any of the following:

6.6.1 Time between Pulses

The MKIII can measure at up to a rate of 3000Hz (875Hz in the Standard version). The MKIII can perform each measurement as long as the pulse timing of the external trigger greater than or equal to 1/3000 sec (1/875 sec in the Standard version).

Remark: The above is valid for ideal external pulses without Jitter. Jitter can cause the pulse timing to be lower than 1/3000 sec resulting in missed measurements.

6.6.2 Hardware Configuration

If the external trigger inputs are not properly connected to the COMMUNICATION BOX, the pulses may be badly timed and which can cause measurements to be missed. Check the hardware configuration to ensure that:

a. Standard TTL or CMOS type drivers are used at the input.

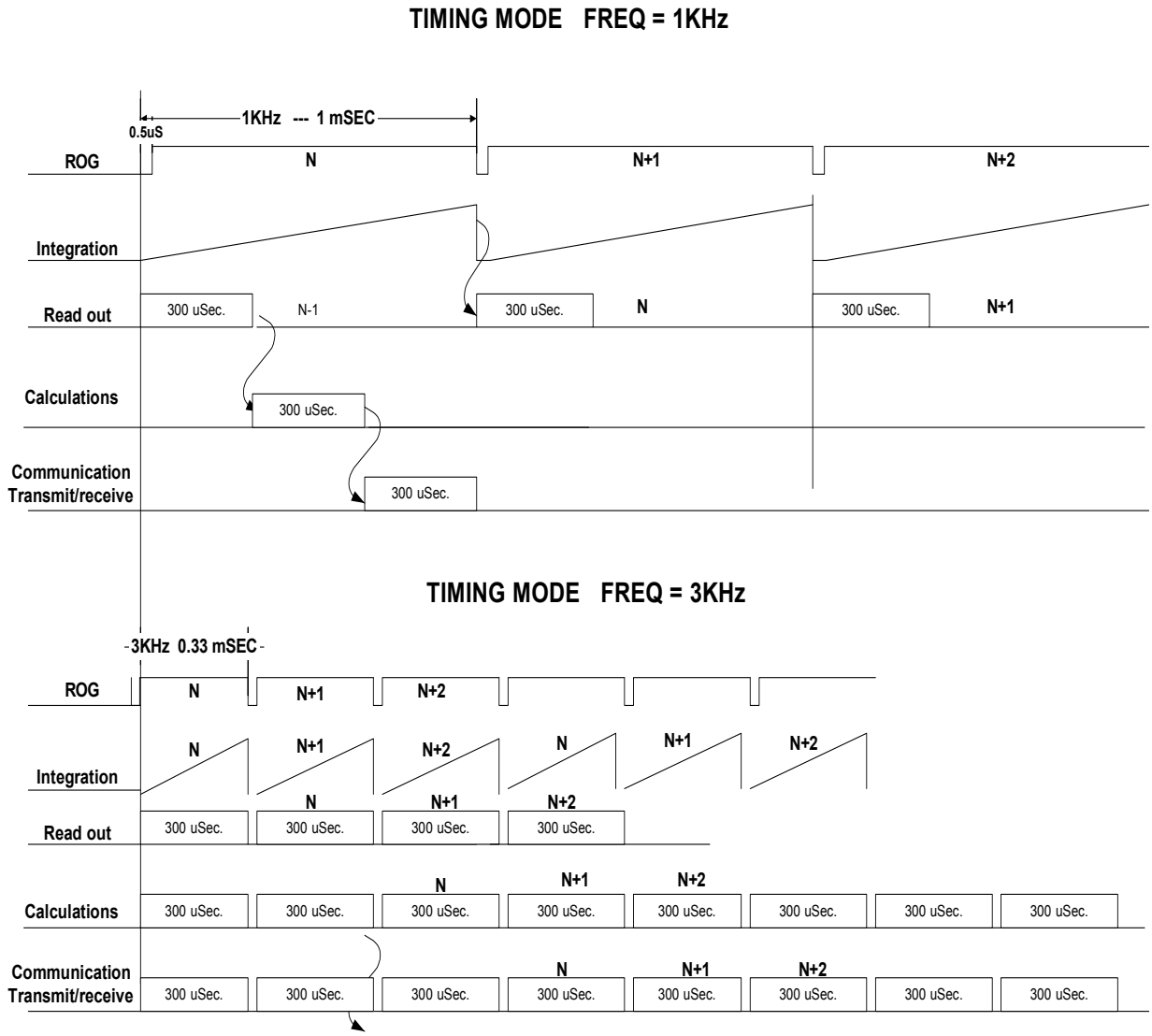
b. Input edges are short enough (<1 μs rise and fall times).

c. The correct current drive is being used (2.2 mA minimum to 5 mA maximum).

d. A good common ground exists between communication box and system

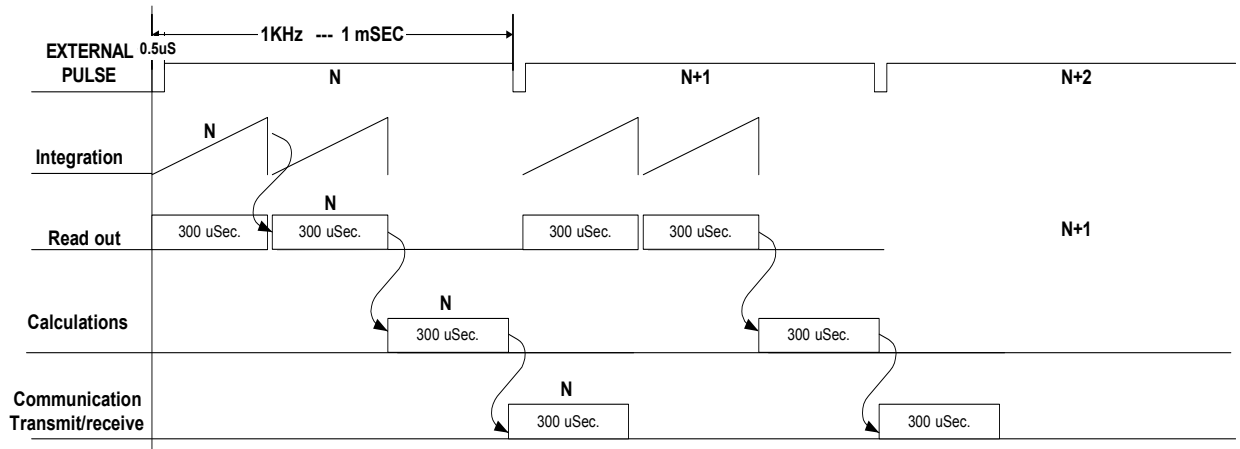
6.7 Timing Diagram elaborations

6.7.1 Time Mode

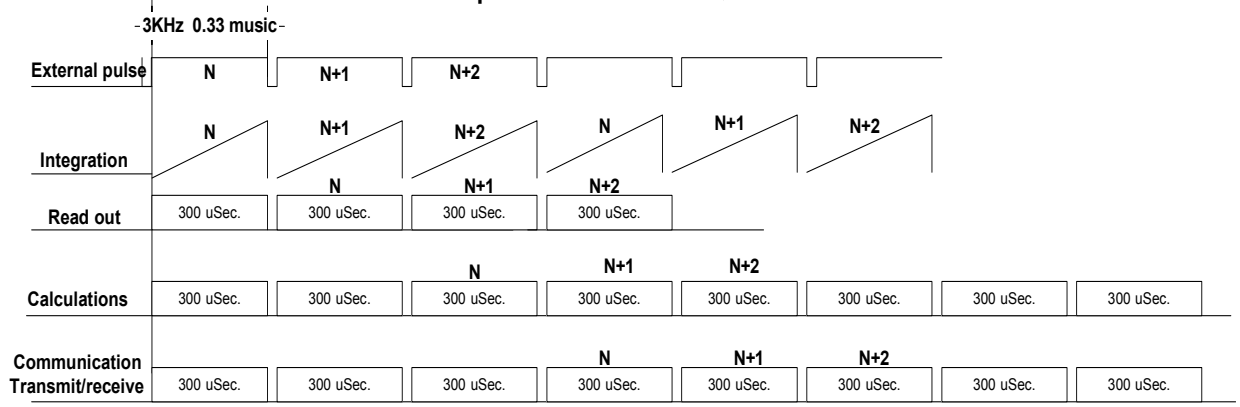


6.7.2 External Triggering

EXT-PULSE MODE PULE
Repetition = 1KHz
Freq = 3Khz



EXTERNAL PULSE MODE
Repetition = 3KHz FREQ = 3KHz



7 Appendix A: Comparisons between Probes

Sensor	Smart ConoProbe	ConoProbe Mark II	ConoProbe Mark III
Size	94X85X62 mm	80X180X60 mm	80X180X60 mm
Weight	380 gr.	750 gr.	750gr
Communication	Ethernet	Parallel	Ethernet
Software	FIFO.dll, Active X, .NET Assembly	FIFO.dll	FIFO.dll, ActiveX, .NET Assembly
Working range to precision ratio	2500	1300	1300
Standard deviation	X2 better than ConoProbe		Like ConoProbe II
Linearity	0.05%	0.1%	0.1%
Lens availability	Limited	Extended lenses & periscopes	Extended lenses & periscopes
TV camera option	No	Yes	Yes
Multiple sensor interface to one PC	Yes	No (only 2)	Yes
Wireless communication	Yes, possible	No	Yes, possible
Angle coverage	85 degrees on one axis 82 in the other	85 degrees on both axes	85 degrees on both axes
Measurement on shinny surfaces	Good, limited by multiple reflections from the side in Y axis	Very good in HRS version only	Very good in HRS version only
Intensity dynamic range	Very large	Acceptable	Large
CMM interface capability	Yes, possible	No	No
Robustness	Robust	Very robust	Very robust
Measuring speed	3000 Hz	750 Hz	875/3000 Hz
Temperature dependence	High	Medium	Medium
Temperature compensation	Available as option	No	Future option
Auto laser power / Exposure control	Not available	Not available	Not available
External electronics	None-small interface box	EC1000	None-small interface box
ROHS electronics	On demand	No	Yes
Input power	12V.DC	100-240V AC	12V DC

8 Appendix B: Sample OEM Programs

The ConoProbe Mark III Software Development Kit (SDK) provides many code samples that illustrate measurement acquisition techniques using the ConoProbe Mark III API. The samples provided illustrate routines for obtaining lens information, setting lens, using the probe dialog, setting measurement parameters, stream and buffer measurement acquisitions, and measurement acquisition using callback functions. The SDK is divided into the following subfolders:

Borland C++ (VCL) – Borland C++ Builder sample project with GUI
C – C Console code examples
Delphi – Delphi sample project with GUI
Dot Net (C#) – C# sample project with GUI
Dot Net (VB) – Visual Basic.Net sample project with GUI
Visual Basic 6 – Visual Basic 6 sample project with GUI
Visual C++ (MFC) – Visual C++ sample project with GUI

Library – Folder containing the FIFO.DLL and support files needed to run the projects.
Lib – Folder containing export libraries for the Microsoft and Borland development systems.

The following development tools are recommended to compile and run the samples with GUI:
Delphi 5
Borland C++ 5
Microsoft Dot Net 2003
Visual Basic 6
Visual C++ 6

The samples written in C require a 32-bit compiler that supports the WIN32 API.

In order to run the samples, the files FIFO.DLL and FIFO_PS.DLL must be copied from the Library folder to either your system32 folder or to the folder where the program EXE file resides.

9 Appendix C: Conoscopic Holography

The ConoProbe Mark III uses an innovative technique known as conoscopic holography to perform non-contact profile measurements. Conoscopic holography has many advantages over classical holographic and other techniques. This appendix illustrates how the probe uses conoscopic holography to provide superior two-dimensional profile measuring results.

Classical vs. Conoscopic Holography

In classical holography, a hologram is created by detecting and recording the interference pattern formed between an object beam and reference beam using a coherent light source. The object beam and reference beam propagate with the same velocity, but follow different geometrical paths, creating a Gabor Zone Lens (GZL).

In conoscopic holography, however, the separate coherent beams are replaced by the ordinary and extraordinary components of a single beam traversing a uni-axial crystal. With this method, it is possible to produce holograms using incoherent light, with a fringe period compatible with the resolution of conventional electronic imaging devices. Since both beams propagate through the same path, conoscopic holography is highly stable.

How the ConoProbe Mark III Works

The ConoProbe Mark III emits a laser beam. This beam is reflected by a beam splitter, and hits the specimen being measured. Scattered light travels from the specimen through the beam splitter and birefringent crystal, and is detected by the probe's CCD camera (refer to *Figure A-9-1*).

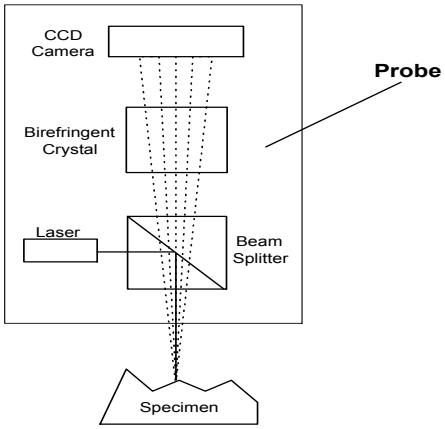


Figure B-9-2: Paths of Light Inside the ConoProbe Mark III

The birefringent crystal modifies light rays in a way that is dependent on the angle of each ray of light. This creates a high contrast pattern on the CCD camera..

Benefits of Conoscopic Technology

In the ConoProbe Mark III, each camera pixel contributes to the signal. If some of the light is blocked or is of poor quality, other areas can provide enough of a signal for measurement. Comparison to Other Methods:

Triangulation is a method of non-contact distance measurement. It is not co-linear, and requires a rigid, expensive, temperature sensitive optical system. In contrast to triangulation, with the probe, the outgoing laser beam is in the same optical axis as the returned signal. This co-linearity allows the probe to measure inside holes, through folding optics (mirrors), and measure steeply inclined surfaces (even at angles greater than 85 degrees).

Dynamic focus systems have moving parts and are less reliable and accurate. In addition, dynamic focus does not work well measuring surfaces that have abrupt changes in distance. Unlike dynamic focus systems, the ConoProbe Mark III has no moving parts, and can perform point-by-point mapping of abrupt changes in distance.