

**N2100B  
PXIT Digital  
Communications  
Analyzer Module**

**User's Guide**



**Agilent Technologies**

## Notices

© Agilent Technologies, Inc. 2007

No part of this manual may be reproduced in any form or by any means (including electronic storage and retrieval or translation into a foreign language) without prior agreement and written consent from Agilent Technologies, Inc. as governed by United States and international copyright laws.

## Manual Part Number

N2100-90002

## Edition

August 2007  
Printed in USA

Agilent Technologies, Inc.  
Digital Signal Analysis Division  
1400 Fountaingrove Parkway  
Santa Rosa, CA 95403, USA

## Warranty

The material contained in this document is provided "as is," and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Agilent disclaims all warranties, either express or implied, with regard to this manual and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Agilent and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

## Technology Licenses

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

LZW compression/decompression: Licensed under U.S. Patent No. 4,558,302 and foreign counterparts. The purchase or use of LZW graphics capability in a licensed product does not authorize or permit an end user to use any other product or perform any other method or activity involving use of LZW unless the end user is separately licensed in writing by Unisys.

Portions of this software are licensed under one or more Open Source licenses, including version 2 of the General Public License (GPL) and version 2 of the Library or Lesser General Public License (LGPL). The source code of the GPL and LGPL licensed content is governed under the terms of the respective licenses and may be found on the CD shipped with the instrument.

## Restricted Rights Legend

If software is for use in the performance of a U.S. Government prime contract or subcontract, Software is delivered and licensed as "Commercial computer software" as defined in DFAR 252.227-7014 (June 1995), or as a "commercial item" as defined in FAR 2.101(a) or as "Restricted computer software" as defined in FAR 52.227-19 (June 1987) or any equivalent agency regulation or contract clause. Use, duplication or disclosure of Software is subject to Agilent Technologies' standard commercial license terms, and non-DOD Departments and Agencies of the U.S. Government will receive no greater than Restricted Rights as defined in FAR 52.227-19(c)(1-2) (June 1987). U.S. Government users will receive no greater than Limited Rights as defined in FAR 52.227-14 (June 1987) or DFAR 252.227-7015 (b)(2) (November 1995), as applicable in any technical data.

## Safety Notices

### **CAUTION**

Caution denotes a hazard. It calls attention to a procedure which, if not correctly performed or adhered to, could result in damage to or destruction of the product. Do not proceed beyond a caution sign until the indicated conditions are fully understood and met.

## **WARNING**

Warning denotes a hazard. It calls attention to a procedure which, if not correctly performed or adhered to, could result in injury or loss of life. Do not proceed beyond a warning sign until the indicated conditions are fully understood and met.

## Trademark Acknowledgements

Microsoft is a U.S. registered trademark of Microsoft Corporation.

Windows and MS Windows are U.S. registered trademarks of Microsoft Corporation.

## Instrument Markings



The CE mark is a registered trademark of the European Community.

ISM1-A This text denotes the instrument is an Industrial Scientific and Medical Group 1 Class A product.

ICES/NMB-001 This is a marking to indicate product compliance with the Canadian Interference-Causing Equipment Standard.



This product complies with the WEEE Directive (2002/96/EC) marking requirements. The affixed label indicates that you must not discard this electrical/electronic product in domestic household waste. Product

Category: With reference to the equipment types in the WEEE Directive Annex I, this product is classed as a "Monitoring and Control instrumentation" product. Do not dispose in domestic household. To return unwanted products, contact your local Agilent office, or refer to [www.agilent.com](http://www.agilent.com) for more information.



**N10149**

The C-Tick mark is a registered trademark of the Australian spectrum Management Agency.



This symbol indicates the Environmental Protection Use Period (EPUP) for the product's toxic substances for the China RoHS requirements.

---

# Contents

## **1 General Information**

- Introduction 1–2
- Electrostatic Discharge Information 1–5
- Connector Care 1–7
- Returning the N2100B to Agilent 1–14

## **2 Installation**

- Introduction 2–2
- Upgrading the Instrument Firmware 2–9

## **3 Configuring the DCA and Acquiring Data**

- Introduction 3–2
- Module Configuration Settings 3–4
- Acquiring Data 3–10
- Calibrating the N2100B 3–12

## **4 Measurements**

- Introduction 4–2
- Oscilloscope Measurements Mode 4–5
- Non Return to Zero Measurements Mode 4–16
- Eye Mask Test Measurements Mode 4–37
- Multiple Measurements Mode 4–44
- Pattern Sequence D. J. Measurements Mode 4–46

## **5 Displaying and Saving Results**

- Introduction 5–2
- Program Settings 5–3
- Copying and Saving Displayed Test Results 5–11

## **6 Pattern Acquisition**

- Pattern Acquisition 6–2

## **7 Mask & Mask Margin Definitions**

- Introduction 7–2
- How the Mask and Mask Margin are defined 7–3
- Agilent Mask Margin Rules 7–6
- Tektronix Mask Margin Rules 7–8

## Contents

### **8 Using the Simulator**

Introduction 8-2

Configuring the Simulator 8-5

Controlling with User-written Applications 8-7

### **9 Programming**

Introduction 9-2

DLL API Reference 9-6

ActiveX API Methods 9-42

ActiveX API Properties 9-75

### **10 Specifications**

Specifications 10-2

Introduction	1-2
Electrostatic Discharge Information	1-5
Connector Care	1-7
Returning the N2100B to Agilent	1-14

---

## General Information

---

## Introduction

The N2100B Digital Communications Analyzer (DCA) is a PXI based instrument that automatically performs accurate eye-diagram analysis to characterize the quality of sources (transmitters) from 1.063 Gb/s to 8.5 Gb/s for production ATE applications. The DCA has five classes of operation:

- Oscilloscope
- NRZ
- Eye mask
- Multiple measurements
- Pattern acquisition and digital filters

Refer to Chapter 4, “Measurements” for more information.

The N2100B implements a coherent vector under-sampling technique which combines the benefits and measurement capabilities of a real time scope with the bandwidth of a sampling scope.

### Main Features

- Optical input with a 750 to 1650 nm optical wavelength range.
- Includes up to four selectable filters in the optical input path.
- Electrical input with 1Vpp maximum level.
- Input bit rate from 155.52 Mb/s to 11.318 Gb/s.
- Internal clock recovery for signals from 155 Mb/s to 2.7 Gb/s; external clock operation with clock range of 10 MHz to 11.318 GHz.
- Pattern acquisition with software digital 4th-order Bessel-Thompson filters and no pattern trigger required.
- Optical performance and accuracy guaranteed at the calibrated wavelength of 850 nm (with offsets applied for 1310 and 1550 nm).

The DCA is a PXI based instrument. Either an embedded PC or a stand alone PC connected via a remote bridge (such as the NI MXI-4 product) can be used to control it via the PXI bus. It is also possible to connect to the instrument via a TCP/IP connection. The module is supplied with a Windows Control Panel application, an ActiveX interface usable from environments such as Visual

C++, Visual Basic, and LabWindows, a programmer's API and libraries callable from user-written applications. Software examples illustrating the use of the module interfaces are included in the supplied software.

---

## **General Safety Considerations**

This product has been designed and tested in accordance with IEC Publication 1010, Safety Requirements for Electronic Measuring Apparatus, and has been supplied in a safe condition. The instruction documentation contains information and warnings which must be followed by the user to ensure safe operation and to maintain the product in a safe condition.

<b>WARNING</b>	<b>Install the plug-in module according to the enclosure protection provided and placing filler panels in empty slots. This instrument does not protect against the ingress of water. This instrument protects against finger access to hazardous parts within the enclosure.</b>
<b>WARNING</b>	<b>Laser Safety Notice. The N2100B is used to measure optical signals. When connecting and disconnecting optical cables or equipment, all optical sources MUST be disabled. Failure to take proper safety precautions may result in eye damage. All unused optical ports MUST be covered when not in use to prevent light leakage or contamination.</b>
<b>WARNING</b>	<b>If this product is not used as specified, the protection provided by the equipment could be impaired. This product must be used in a normal condition (in which all means for protection are intact) only.</b>
<b>WARNING</b>	<b>No operator serviceable parts inside. Refer servicing to qualified service personnel. To prevent electrical shock do not remove covers.</b>
<b>WARNING</b>	<b>Use a dry cloth or one slightly dampened with water to clean the external case parts. Do not attempt to clean internally.</b>
<b>CAUTION</b>	<b>This product is designed for use in Installation Category II and Pollution Degree 2 per IEC 1010 and 664 respectively.</b>
<b>CAUTION</b>	<b>The N2100B is shipped in materials which prevent damage from static. The module should only be removed from the packaging in an anti-static area ensuring that correct anti-static precautions are taken.</b>
<b>WARNING</b>	<b>This product is NOT tested for use in medical or clinical applications.</b>
<b>WARNING</b>	<b>Use appropriate caution when using Agilent products for testing lasers.</b>



## Electrostatic Discharge Information

### CAUTION

Electrical channel input circuits and the trigger input circuit can be damaged by electrostatic discharge (ESD). Therefore, avoid applying static discharges to the front-panel input connectors. Prior to connecting any coaxial cable to the connectors, momentarily short the center and outer conductors of the cable together. Avoid touching the front-panel input connectors without first touching the frame of the instrument. Be sure that the instrument is properly earth-grounded to prevent buildup of static charge. Wear a wrist-strap or heel-strap.

Electrostatic discharge (ESD) can damage or destroy electronic components. All work on electronic assemblies should be performed at a static-safe work station. The following figure shows an example of a static-safe work station using two types of ESD protection:

- Conductive table-mat and wrist-strap combination.
- Conductive floor-mat and heel-strap combination.

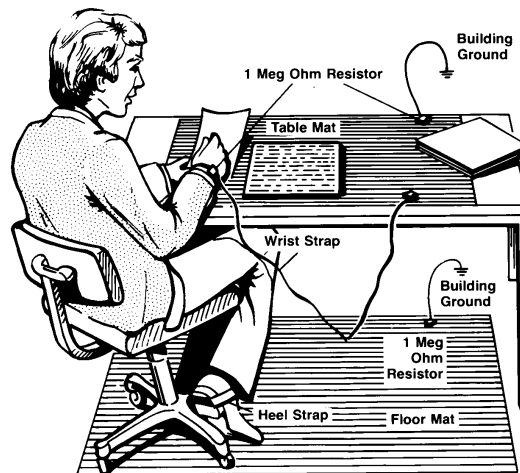


Figure 1-1. Static-safe Work Station

**Electrostatic Discharge Information**

Both types of techniques, when used together, provide a significant level of ESD protection. Of the two, only the table-mat and wrist-strap combination provides adequate ESD protection when used alone. To ensure user safety, the static-safe accessories must provide at least 1 M $\Omega$  of isolation from ground.

---

**WARNING**

---

**These techniques for a static-safe work station should not be used when working on circuitry with a voltage potential greater than 500 volts.**

---

## Connector Care

Today, advances in measurement capabilities make connectors and connection techniques more important than ever. Damage to the connectors on calibration and verification devices, test ports, cables, and other devices can degrade measurement accuracy and damage instruments. Replacing a damaged connector can cost thousands of dollars, not to mention lost time! This expense can be avoided by observing the simple precautions presented in this section.

---

### Electrical Connectors

Advances in measurement capabilities make connectors and connection techniques more important than ever. Observing simple precautions can ensure accurate and reliable measurements.

#### ***Handling and storage***

- Keep connectors clean.
- Extend sleeve or connector nut.
- Use plastic endcaps during storage.
- Do not touch mating plane surfaces.
- Do not set connectors contact-end down.

#### ***Visual inspection***

- Inspect all connectors carefully before every connection.
- Look for metal particles, scratches, and dents.
- Do not use damaged connectors.

#### ***Cleaning***

- Clean with compressed air first.
- Clean the connector threads.
- Do not use abrasives.
- Do not get liquid onto the plastic support beads.

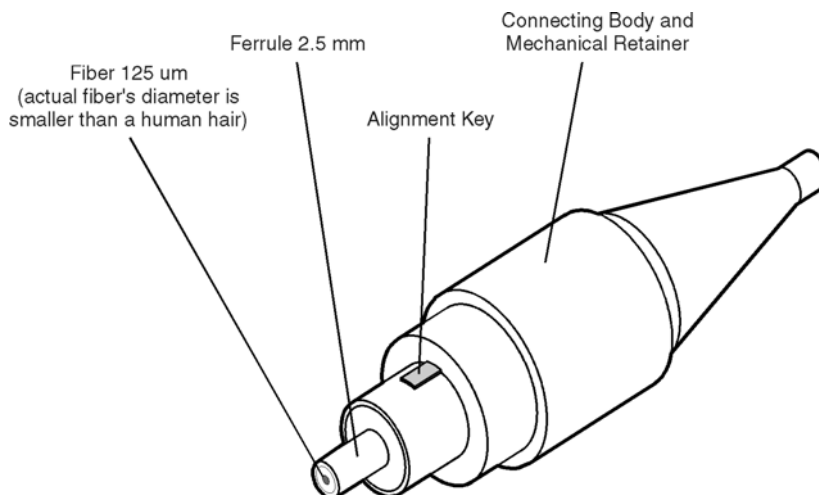
***Making connections***

- Use connector savers.
- Align connectors carefully.
- Make preliminary connection lightly.
- To tighten, turn connector nut only.
- Do not apply bending force to connection.
- Do not over tighten preliminary connection.
- Do not twist or screw in connectors.
- Use a torque wrench, and do not tighten past the "break" point of the torque wrench.

---

## **Optical Connectors**

Taking care of fiber-optic connectors is critical to making quality measurements. Because fiber-optic connectors are susceptible to damage that is not immediately obvious to the naked eye, poor measurements result without the user being aware. Microscopic examination and return loss measurements are the best way to ensure good measurements. Good cleaning practices can help ensure that optimum connector performance is maintained. With glass-to-glass interfaces, any degradation of a ferrule or the end of the fiber, any stray particles, or finger oil can have a significant effect on connector performance. Where many repeat connections are required, use of a connector saver or patch cable is recommended. [Figure 1-2 on page 1-9](#) shows the basic components of a typical connectors.



**Figure 1-2. Basic components of a connector.**

Figure 1-3 shows the end of a clean fiber-optic cable. The dark circle in the center of the micrograph is the fiber's 125  $\mu\text{m}$  core and cladding which carries the light. The surrounding area is either composed of a ceramic material or soft nickel-silver ferrule. Figure 1-4 shows a dirty fiber end from neglect or perhaps improper cleaning. Material is smeared and ground into the end of the fiber causing light scattering and poor reflection. Not only is the precision polish lost, but this action can grind off the glass face and destroy the connector.

Figure 1-5 shows physical damage to the glass fiber end caused by either repeated connections made without removing loose particles or using improper cleaning tools. When severe, the damage of one connector end can be transferred to another good connector endface that comes in contact with the damaged one. Periodic checks of fiber ends, and replacing connecting cables after many connections is a wise practice.

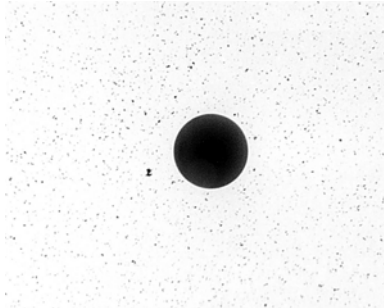
The cure for these problems is disciplined connector care as described in the following list and in [“Cleaning Non-lensed Connectors”](#) on page 1-12.

Use the following guidelines to achieve the best possible performance when making measurements on a fiber-optic system:

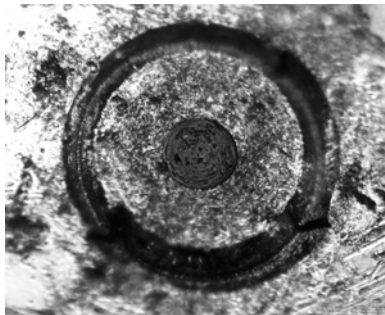
- Never use metal or sharp objects to clean a connector and never scrape the connector.

**Optical Connectors**

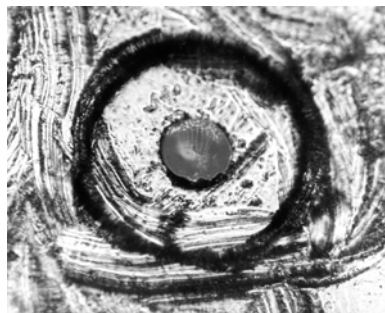
- Avoid matching gel and oils. The use of matching gel or oil will invalidate the N2100B system calibration. Contact Agilent Customer assistance if you have questions on this issue.



**Figure 1-3. Clean, problem-free fiber end and ferrule.**



**Figure 1-4. Dirty fiber end and ferrule from poor cleaning.**



**Figure 1-5. Damage from improper cleaning.**

While these often work well on first insertion, they are great dirt magnets. The oil or gel grabs and holds grit that is then ground into the end of the fiber. Also, some early gels were designed for use with the FC, non-contacting connectors, using small glass spheres. When used with contacting connectors, these glass balls can scratch and pit the fiber. If an index matching gel or oil must be used, apply it to a freshly cleaned connector, make the measurement, and then immediately clean it off. Never use a gel for longer-term connections and never use it to improve a damaged connector. The gel can mask the extent of damage and continued use of a damaged fiber can transfer damage to the instrument.

- When inserting a fiber-optic cable into a connector, gently insert it in as straight a line as possible. Tipping and inserting at an angle can scrape material off the inside of the connector or even break the inside sleeve of connectors made with ceramic material.
- When inserting a fiber-optic connector into a connector, make sure that the fiber end does not touch the outside of the mating connector or adapter.
- Avoid over-tightening connections.

Unlike common electrical connections, tighter is *not* better. The purpose of the connector is to bring two fiber ends together. Once they touch, tightening only causes a greater force to be applied to the delicate fibers. With connectors that have a convex fiber end, the end can be pushed off-axis resulting in misalignment and excessive return loss. Many measurements are actually improved by backing off the connector pressure. Also, if a piece of grit does happen to get by the cleaning procedure, the tighter connection is more likely to damage the glass. Tighten the connectors just until the two fibers touch.

- Keep connectors covered when not in use.
- Use fusion splices on the more permanent critical nodes. Choose the best connector possible. Replace connecting cables regularly. Frequently measure the return loss of the connector to check for degradation, and clean every connector, every time.

All connectors should be treated like the high-quality lens of a good camera. The weak link in instrument and system reliability is often the inappropriate use and care of the connector. Because current connectors are so easy to use, there tends to be reduced vigilance in connector care and cleaning. It takes only one missed cleaning for a piece of grit to permanently damage the glass and ruin the connector.

### ***Measuring insertion loss and return loss***

Consistent measurements with your lightwave equipment are a good indication that you have good connections. Since return loss and insertion loss are key factors in determining optical connector performance they can be used to determine connector degradation. A smooth, polished fiber end should produce a good return-loss measurement. The quality of the polish establishes the difference between the “PC” (physical contact) and the “Super PC” connectors. Most connectors today are physical contact which make glass-to-glass connections, therefore it is critical that the area around the glass core be clean and free of scratches. Although the major area of a connector, excluding the glass, may show scratches and wear, if the glass has maintained its polished smoothness, the connector can still provide a good low level return loss connection.

If you test your cables and accessories for insertion loss and return loss upon receipt, and retain the measured data for comparison, you will be able to tell in the future if any degradation has occurred. Typical values are less than 0.5 dB of loss, and sometimes as little as 0.1 dB of loss with high performance connectors. Return loss is a measure of reflection: the less reflection the better (the larger the return loss, the smaller the reflection). The best physically contacting connectors have return losses better than 50 dB, although 30 to 40 dB is more common.

### ***Visually inspecting fiber ends***

Visual inspection of fiber ends can be helpful. Contamination or imperfections on the cable end face can be detected as well as cracks or chips in the fiber itself.

---

## **WARNING**

---

**Always remove both ends of fiber-optic cables from any instrument, system, or device before visually inspecting the fiber ends. Disable all optical sources before disconnecting fiber-optic cables. Failure to do so may result in permanent injury to your eyes.**

Use a microscope (100X to 200X magnification) to inspect the entire end face for contamination, raised metal, or dents in the metal as well as any other imperfections. Inspect the fiber for cracks and chips. Visible imperfections not touching the fiber core may not affect performance (unless the imperfections keep the fibers from contacting).

### ***Cleaning Non-lensed Connectors***

The procedures in this section provide the proper steps for cleaning fiber-optic cables and Agilent Technologies universal adapters. The initial cleaning, using the alcohol as a solvent, gently removes any grit and oil. If a caked-on



layer of material is still present, (this can happen if the beryllium-copper sides of the ferrule retainer get scraped and deposited on the end of the fiber during insertion of the cable), a second cleaning should be performed. It is not uncommon for a cable or connector to require more than one cleaning.

---

**CAUTION**

---

Agilent Technologies strongly recommends that index matching compounds *not* be applied to their instruments and accessories. Some compounds, such as gels, may be difficult to remove and can contain damaging particulates. If you think the use of such compounds is necessary, refer to the compound manufacturer for information on application and cleaning procedures.

---

**CAUTION**

---

Do not use any type of foam swab to clean optical fiber ends. Foam swabs can leave filmy deposits on fiber ends that can degrade performance.

- 1 Apply pure isopropyl alcohol to a clean lint-free cotton swab or lens paper.  
Cotton swabs can be used as long as no cotton fibers remain on the fiber end after cleaning.
- 2 Clean the ferrules and other parts of the connector while avoiding the end of the fiber.
- 3 Apply isopropyl alcohol to a new clean lint-free cotton swab or lens paper.
- 4 Clean the fiber end with the swab or lens paper.

Do *not* scrub during this initial cleaning because grit can be caught in the swab and become a gouging element.

- 5 Immediately dry the fiber end with a clean, dry, lint-free cotton swab or lens paper.
- 6 Blow across the connector end face from a distance of 6 to 8 inches using filtered, dry, compressed air. Aim the compressed air at a shallow angle to the fiber end face.

Nitrogen gas or compressed air dust remover can also be used.

---

**CAUTION**

---

Do not shake, tip, or invert compressed air canisters, because this releases particles in the can into the air. Refer to instructions provided on the compressed air canister.

- 7 As soon as the connector is dry, connect it or cover it for later use.

If the performance, after the initial cleaning, seems poor try cleaning the connector again. Often a second cleaning will restore proper performance. The second cleaning should be more arduous with a scrubbing action.

---

## Returning the N2100B to Agilent

The instructions in this section show you how to properly package the instrument for return to an Agilent Technologies service office. If the instrument is still under warranty or is covered by an Agilent maintenance contract, it will be repaired under the terms of the warranty or contract. If the instrument is no longer under warranty or is not covered by an Agilent maintenance plan, Agilent will notify you of the cost of the repair after examining the unit.

When an instrument is returned to an Agilent service office for servicing, it must be adequately packaged and have a complete description of the failure symptoms attached.

When describing the failure, please be as specific as possible about the nature of the problem. Include copies of any instrument failure settings, data related to instrument failure, and error messages along with the instrument being returned.

Please notify the service office before returning your instrument for service. Any special arrangements for the instrument can be discussed at this time. This will help the Agilent service office repair and return your instrument as quickly as possible.

### ***Call Center***

For technical assistance, contact your local Agilent Call Center. In the Americas, call 1 (800) 829-4444. In other regions, visit <http://www.agilent.com/find/assist>. Before returning an instrument for service, you must first contact your local Agilent Call Center.

### ***Preparing the product for shipping***

- 1** Write a complete reason for returning the product and attach it to the instrument. Include any specific performance details related to the problem.
- 2** Pack the product. Use original packaging or comparable. Original materials are available through any Agilent office. Or, follow these recommendations:
  - Insert the product in an anti-static bag.
  - Use a double-walled, corrugated cardboard carton of 159 kg (350 lb) test

strength. The carton must allow approximately 7 cm (3 inches) on all sides of the kit for packing material and be strong enough to accommodate the weight of the kit.

- Surround the kit with approximately 7 cm (3 inches) of packing material, to protect the kit and prevent it from moving in the carton. If packing foam is not available, the best alternative is S.D-240 Air Cap™ from Sealed Air Corporation (Commerce, California 90001). Air Cap looks like a plastic sheet filled with air bubbles. Use the pink (antistatic) Air Cap™ to reduce static electricity. Wrapping the kit several times in this material will protect the kit and prevent it from moving in the carton.
- 3** Seal the carton with strong nylon adhesive tape.
  - 4** Mark the carton “FRAGILE, HANDLE WITH CARE”.
  - 5** Retain copies of all shipping papers.

General Information

**Returning the N2100B to Agilent**

Introduction	2-2
Step 1. Inspect the Shipment	2-2
Step 2. Install the Instrument Driver Software	2-3
Step 3. Install the N2100B	2-4
Quick Confidence Check	2-5
Powering Off the Instrument	2-8
Upgrading the Instrument Firmware	2-9

---

## Installation

---

## Introduction

The PXI chassis can be controlled using either an embedded PXI controller or an external PC using a PCI - cPCI/PXI remote bridge (such as the NI MXI-4 product).

---

### NOTE

You can control the DCA with Agilent Vee only through ActiveX.

If an external PC is used, the PC must meet the following specifications:

- Windows 2000 or XP operating system
- 1 GB RAM
- Pentium processor, 133 MHz or greater
- NI-VISA

---

### NOTE

When using an external PC as the controller, you must follow the installation steps in the sequence described for the PC BIOS to locate the instruments in the PXI chassis.

---

---

## Step 1. Inspect the Shipment

- 1 Inspect the shipping container and kit for damage. Keep the shipping container and cushioning material until you have inspected the contents of the shipment for completeness and have checked the kit mechanically and electrically.
- 2 Locate the shipping list. Verify that you have received all of the items listed.

To contact Agilent Technologies for technical assistance, contact your local Agilent Call Center. In the Americas, call 1 (800) 829-4444. In other regions, visit <http://www.agilent.com/find/assist>. Before returning an instrument for service, you must first contact your local Agilent Call Center.

- 3 Verify that the following environmental conditions exist before proceeding with the installation procedure.

**Step 2. Install the Instrument Driver Software**

---

**WARNING**

---

Ensure that the PXI chassis is connected to the specified power source using the correct power cord (noting country of use).

---

**WARNING**

---

Ensure that the PXI chassis provides adequate earth grounding.

---

**WARNING**

---

Ensure that the air supply to the chassis is working correctly. The N2100B requires an optimal air flow within the chassis. It is recommended to regularly change filters on PXI Chassis.

---

---

## Step 2. Install the Instrument Driver Software

---

**NOTE**

---

This procedure assumes that NI-VISA has already been installed.

- 1 Log onto the PC with administrator privileges, so that you can install the software.
- 2 Go to the Agilent website: [www.agilent.com/find/pxit](http://www.agilent.com/find/pxit)
- 3 Click on the Technical Support link and then the Drivers link.
- 4 Download the latest version of the following driver:  
Agilent N2100B DCA Driver
- 5 Once the download has completed, run the N2100B installation file.
  - a During the installation, you will enter the user name and organization.
  - b Select the all users option to ensure the software is available to all users of the PC. Click **Next**.
- 6 When installation process is finished, click **Finish**. The N2100B control software is now installed.
- 7 To ensure that the PC BIOS will be able to locate the instruments in the PXI chassis, follow this step according to the controller that you are using:
  - If you are using an external PC and remote bridge, turn the PXI chassis and PC power off.
  - If you are using an embedded controller, turn the PXI chassis and PC power off and remove all N2100B modules from the chassis.

### Step 3. Install the N2100B

---

**NOTE**

---

You can control the DCA with Agilent Vee only through ActiveX.

---

**NOTE**

---

As part of the installation, the N2100B User's Guide is made available on the Windows Start menu.

---

---

## Step 3. Install the N2100B

- 1 With the PC and chassis powered off, install the N2100B module in an available slot in a PXI chassis.
- 2 Power on the PXI chassis and wait for the power up sequence to complete.
- 3 Turn on the PC.

If the software is correctly installed, you will see an indication that new hardware is detected and that the system is attempting to locate the associated software driver. When this process is complete, a notification appears indicating that the hardware is ready for use.

If needed, you can use Windows Device Manager to determine if the instruments have been correctly identified by the BIOS. There should be an NI-VISA PXI Devices entry with your N2100B instrument.

---

**NOTE**

---

The N2100B User's Guide is provided as a PDF file. To locate and view the user's guide, go to the Windows **Start** menu and select **All Programs > Agilent > N2100B > Documents > User's Guide**.

---



---

## Quick Confidence Check

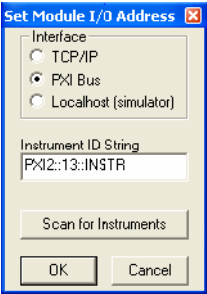
- 1 Apply a PRBS signal to the optical or electrical input connector and then configure the instrument accordingly.

---

**NOTE**

For the N2100B, both optical and electrical signals can be applied at the same time. For the N2100A, a signal can be applied to only one input.

- 2 Allow the DCA to warm up for at least 15 minutes.
- 3 Click the Windows **Start** menu.
- 4 Click **All Programs > Agilent > N2100B**.
- 5 Click **N2100B Control Panel**.
- 6 Click **I/O Config** to open the Set Module I/O Address dialog box, to then configure the communications between the host controller and the instrument.

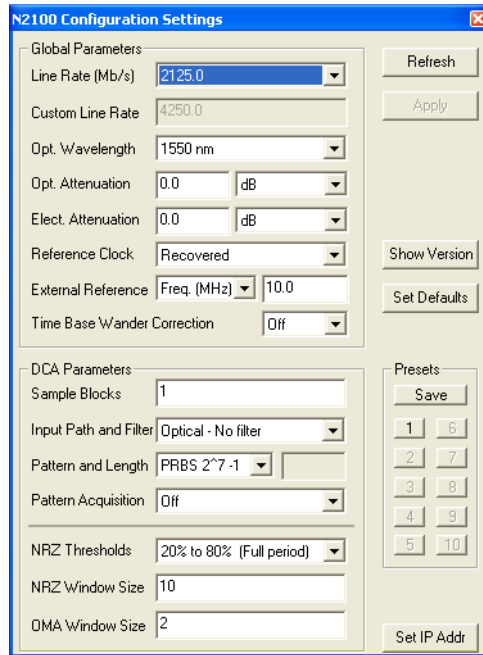


- 7 Select **PXI bus**.
- 8 Click **Scan for Instruments** and select the appropriate DCA instrument ID String.
- 9 Click **OK**. (When multiple DCAs are present in the chassis, use NI Measurement and Automation Explorer (NI MAX) to determine which DCA is in which chassis slot before selecting one.)
- 10 On the Control Panel, click **Connect**. A message panel appears briefly as the connection is established and then disappears.
- 11 Select the **Module Config** button.



**Quick Confidence Check**

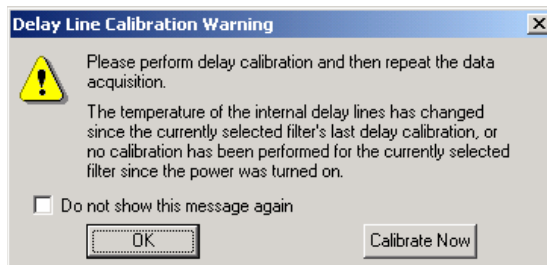
- 12** Select the Line Rate, the reference clock source, and the relevant Input Path and Filter option.



- 13** Click **Start Acquisition**.

Start Acquisition

A pop-up window appears when data acquisition is performed, notifying you that recalibration is advisable. Click **Calibrate Now**.



**Figure 2-1. Calibration Warning Dialog Box**

The Calibration dialog box appears and the calibration starts automatically. When the calibration is complete, the Calibration dialog box disappears.

---

**NOTE**

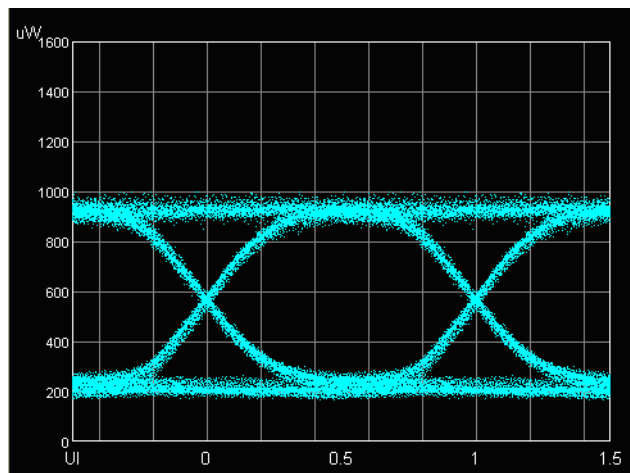
---

There must be a PRBS input signal to run the calibration. If the input signal is not PRBS, or if there is no input signal, click **OK** and ignore the warning that appears.

You should run the Delay Calibration when the following situations occur:

- The first time an input path/filter is selected.
- When the internal module temperature has changed.
- When the line rate changes. (Recommended only.)

- 14** The N2100B is functioning correctly when you can see an eye diagram as shown in the following figure.



If the input signal timing cannot be locked, the following message is displayed.



---

## **Powering Off the Instrument**

- 1** Click **Disconnect**.
- 2** Close the Control Panel.
- 3** Power down the PC.
- 4** Switch off the power to the PXI chassis.

## Upgrading the Instrument Firmware

To update the firmware and execute the FPGA Loader program, perform the following steps:

- 1** Close any open Control Panels.
- 2** On the Windows **Start** menu, click **All Programs > Agilent > N2100B > N2100B FW-FPGA Loader**.
- 3** Press the appropriate buttons to connect to a given module, and note the current module Serial Number, API, and Firmware Versions.
- 4** Select the new firmware and FPGA files located in the installation directory (typically C:\Program Files\Agilent\N2100B\fw), and press start.  
A progress bar appears as the module is updated.
- 5** Once complete, shut down the host PC and cycle the power on the PXI chassis.
- 6** After rebooting the PC, connect to the module and obtain Module Configuration, to confirm proper software upload.



Introduction	3-2
Module Configuration Settings	3-4
Acquiring Data	3-10
Calibrating the N2100B	3-12

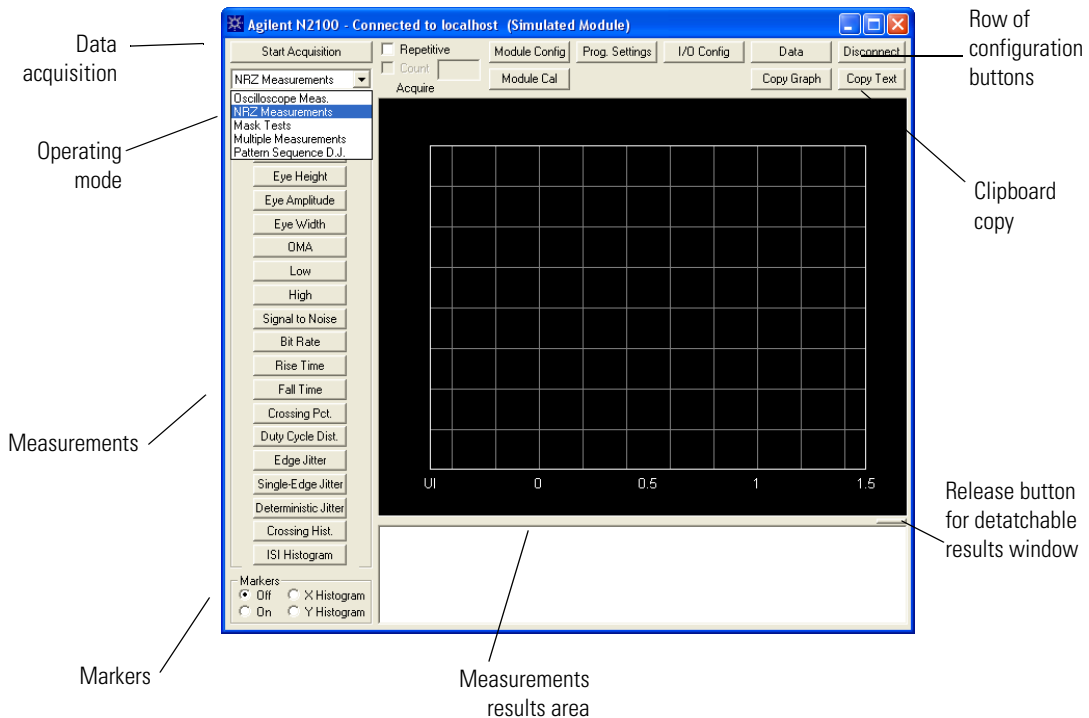
---

## Configuring the DCA and Acquiring Data

## Introduction

This chapter describes how to start and configure the DCA using the Control Panel. For information on controlling the five measurement modes, refer to Chapter 4, “Measurements”.

All aspects of the N2100B can be managed through the Control Panel. Use the top row of buttons to configure the module, I/O, and connect to the module. Use the list box in the top left corner of the main form to select the mode of operation for the N2100B.



**Figure 3-1. Control Panel**



---

## To start the Control Panel

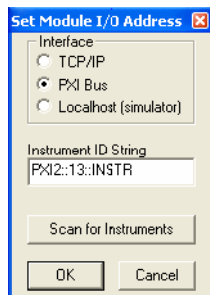
---

### NOTE

A single host computer or controller can control multiple DCAs up to the capability of the PXI backplane. An individual DCA cannot be controlled by more than one application at a time.

---

- 1 Click the Windows **Start** menu.
- 2 Click **All Programs > Agilent > N2100B**.
- 3 Click **N2100B Control Panel**.
- 4 Click **I/O Config** to open the Set Module I/O Address dialog box, to then configure the communications between the host controller and the instrument.

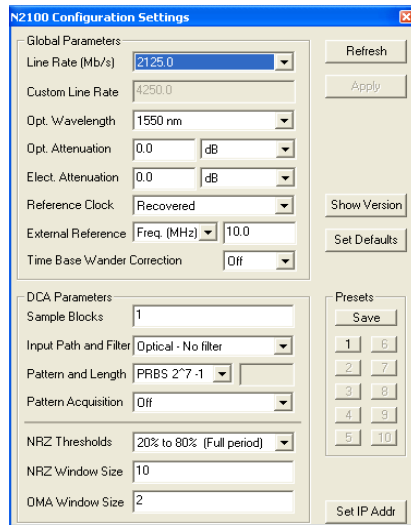


- 5 Select **PXI bus**.
  - If you want to set up a TCP/IP interface, [refer to “Setting the IP address for connecting with TCP/IP” on page 3-4](#).
- 6 Click **Scan for Instruments** and select the appropriate DCA instrument ID String.
- 7 Click **OK**. (When multiple DCAs are present in the chassis, use NI Measurement and Automation Explorer (NI MAX) to determine which DCA is in which chassis slot before selecting one.)
- 8 On the Control Panel, click **Connect**. A message panel appears briefly as the connection is established and then disappears.



## Module Configuration Settings

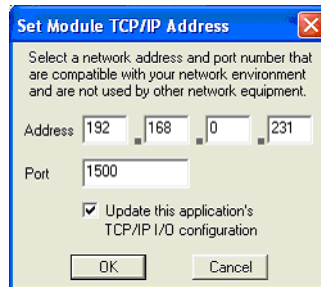
The Configuration Settings dialog box allows you to set up the operating parameters. The current DCA configuration appears in this panel when it first opens and any time the Refresh Button is pressed. Changes made on this screen can be applied to the DCA by pressing the Apply button (when highlighted), or automatically by pressing the Start Acquisition button. Factory defaults and ten (10) user selectable preset configurations are also available. Configuration and presets are persistent and are specific to a particular DCA module.



### *Setting the IP address for connecting with TCP/IP*

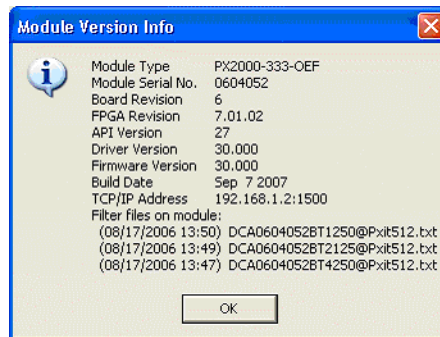
- 1** In the N2100B Control Panel, click **Module Config** to open the Configuration Settings dialog box.
- 2** Click **Set IP Addr**, located in the bottom-right corner of the dialog box.

The Set Module TCP/IP Address dialog box appears.



On a local network, the IP address must be within the range of active addresses for the network. (The DCA does not support DHCP automatic address assignment.) If you are connecting the DCA directly to a computer, a crossover network cable is required and the computer must have a static IP address. The first three fields of both the computer address and the DCA address must be the same.

- 3 Click **OK** and reboot the PXI system to activate the new IP address setting.
- 4 When the system has restarted, open the Control Panel and click **Show Version** to check the module version information and the current IP address setting.




---

## Global Parameters

### Line Rate (Mb/s)

Selects the desired Line Rate from the drop down list or 'Custom.'

### Custom Line Rate

Specifies the desired line rate if 'Line Rate' is set to 'Custom.'

**DCA Parameters**

<b>Optical Wavelength</b>	Specifies the wavelength of the signal under test. This affects the calibration coefficient used to calculate optical power.
<b>Optical Attenuation</b>	This field allows you to enter the value for an external optical attenuator (if any) in the input test signal path. You can enter it as a ratio or a value in dB, with no attenuation as the default. All DCA measurements will compensate for this attenuator and report effective values at its input.
<b>Electrical Attenuation</b>	This is the same function as the Optical Attenuation field described above, but for the electrical input path. You can set these two values independently.
<b>Reference Clock</b>	Sets the Reference Clock to one of the following modes: <ul style="list-style-type: none"> <li>• Internal. The internal reference clock. The recommended modes of operation are either the clock recovery or external clock option. The instrument specifications are valid only with those two modes.</li> <li>• External. A user-provided clock signal connected to the External Clock input. The frequency of this clock must be between 10 MHz and 11.318 GHz and must match the value of the External Reference Clock Frequency parameter.</li> <li>• Recovered. The clock derived from the received optical or electrical data. The recovered clock with a sub-rate pattern of K28.7 is usable with the API, but not the Control Panel interface. See FG_REFCLK_RECOVERED_RATIO in <a href="#">Table 9-4 on page 9-39</a>.</li> </ul>
<b>External Reference</b>	The N2100B can operate with a non-divided down clock signal or an “at rate” clock signal. It can also operate with a clock divided down signal. This parameter allows you to enter the reference as a particular frequency (freq). For example, freq 10 (MHz). You can also enter the value as a division of the line rate (rate/). For example, rate / 128.

---

**DCA Parameters**

<b>Sample Blocks</b>	Specifies the number of sample blocks that will be used to create the 'eye.' Each sample block consists of 1,024 sample points.
<b>Time Base Wander Correction</b>	Enables and disables the Time Base Wander Correction for a measurement. Small variations in the DCA's internal reference time base appear as low-frequency jitter (wander) in the acquired signal. For mask tests and edge jitter measurements, the DCA analyzes the acquired samples and removes the

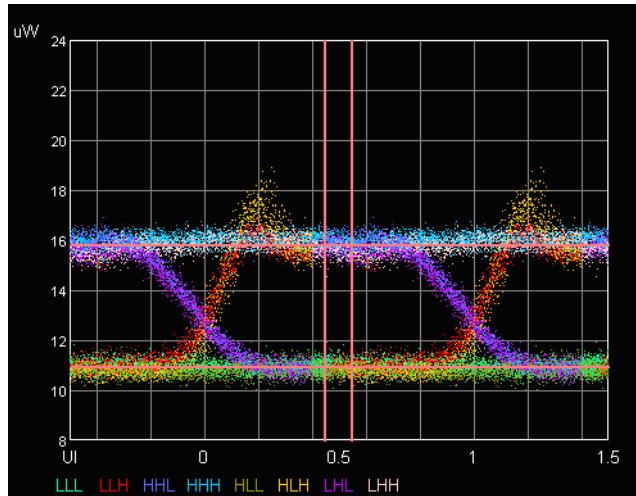
effect of this wander. Wander correction improves the accuracy of DCA measurements but may increase the time to acquire data. For applications where measurement speed is important, you may want to disable wander correction. Reducing the number of blocks to less than 20 significantly reduces the effect of wander.

<b>Input Path and Filter</b>	Sets either the electrical input path or the optical input path and the filter.
<b>Pattern Acquisition</b>	<p>Controls the Pattern Acquisition with the following mode choices:</p> <ul style="list-style-type: none"><li>• Off. The standard DCA eye capture is applied.</li><li>• Enabled (No filter). The DCA performs a pattern capture but does not apply any filtering.</li><li>• Any specific filter. The DCA performs a pattern capture and then applies the selected digital filter to this pattern.</li></ul>
<b>Pattern</b>	<p>Specifies the length of the repeating pattern sequence (PRBS 2<sup>7</sup>-1, PRBS 2<sup>9</sup>-1, PRBS 2<sup>11</sup>-1, K28.5, K28.7), and Set Length for patterns of arbitrary length. When Set Length is selected, the field to the right is enabled so you can enter the pattern length. The length must be between 4 and 2047.</p>
<b>NRZ Thresholds</b>	<p>Selects threshold levels used for the Rise &amp; Fall times calculation. The choices are 10% to 90%, 20% to 80%, and 30% to 70% of either the total Waveform Amplitude (Full Period), or the NRZ-Windowed high and low values.</p>

## DCA Parameters

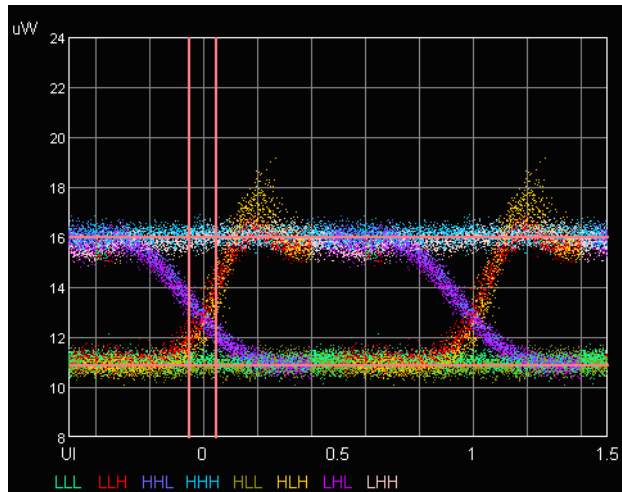
### NRZ Window Size

Sets the percentage of the bit period (unit interval) used for NRZ measurements. The NRZ window is centered between the eye crossings.



### OMA Window Size

Sets the percentage of the bit period (unit interval) used for OMA measurements. The OMA window is centered on the eye crossing.



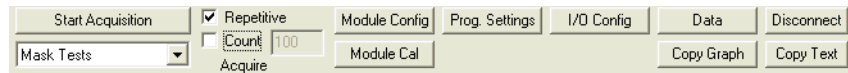
---

## Additional Controls

<b>Refresh</b>	Cancels unapplied changes on the screen and restores the current DCA configuration.
<b>Apply</b>	Sends the new configuration settings to the DCA, when you have made changes to parameters. Pressing Start Acquisition also applies any changes to parameters.
<b>Show Version</b>	Displays information about the Module Hardware and Software.
<b>Set Defaults</b>	Loads DCA Factory Defaults. These are applied with Apply or Start Acquisition buttons.
<b>Set IP Addr</b>	Sets the Module IP address. The first three fields for this address must be the same as the address for the computer that is trying to connect to the instrument.
<b>Presets</b>	Clicking <b>Save</b> , followed by one of the buttons [1 - 10] stores the current configuration parameters into one of 10 presets. Preset configurations are DCA specific, and persist until deleted. Clicking buttons [1 - 10] restores the previously saved configuration associated with that button. To delete a preset configuration, click <b>Save</b> (while holding the Ctrl key) followed by the preset button [1-10].

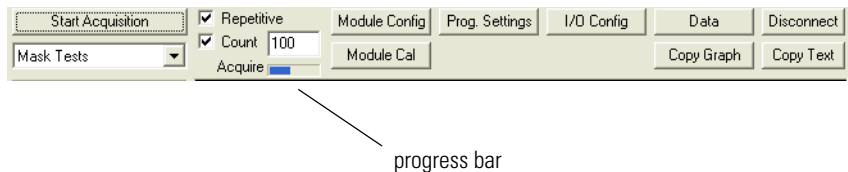
## Acquiring Data

After configuring the module and the Control Panel, click **Start Acquisition** to begin acquiring data. In the Control Panel, you can select **Repetitive** to have continuous acquisitions, as shown in the following figure.



**Figure 3-2. Repetitive Selected**

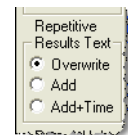
Select **Repetitive** and **Count** to specify a number of acquisitions to acquire. In the following figure, the count is set to 100. A progress bar indicates how many acquisitions have happened and how many are still to be taken.



**Figure 3-3. Count Selected with Progress Bar**

Use the General tab of the Program Settings dialog box to configure the repetitive results text.

- Select **Overwrite** during a repetitive acquisition sequence so each subsequent measurement overwrites the measurement on the display.
- Select **Add** to create a list of measurement results on the display where each new result is added to the list.
- Select **Add + Time** to create a time stamped list of measurement results. As

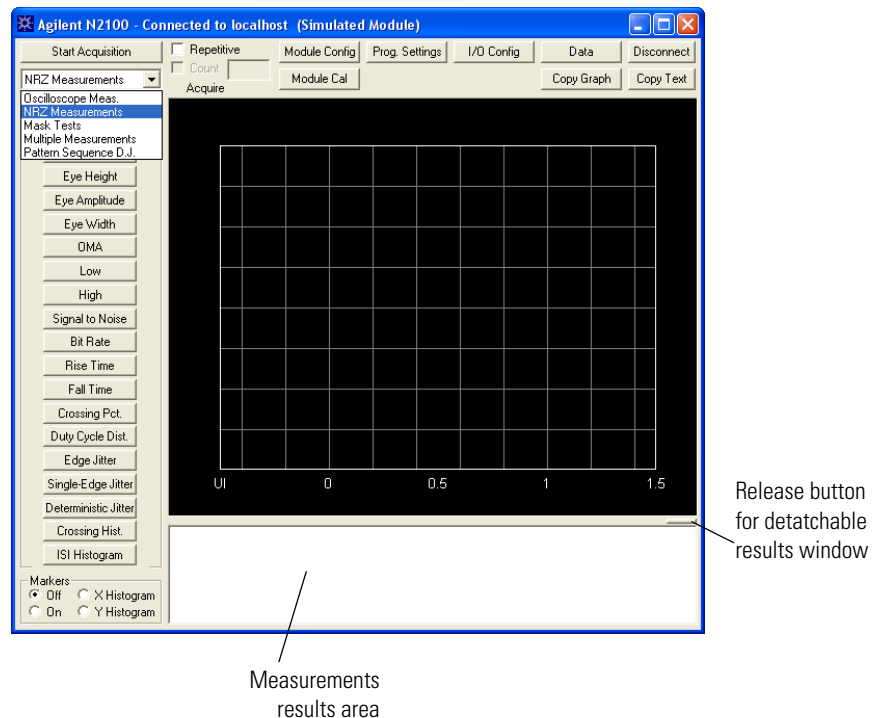




each measurement result is added to form a list on the screen, a start time, an end time, and an elapsed time of the most recent data collection is included. A time stamp is also included in the results list every 30 seconds.

### ***Viewing and moving the measurement results window***

The measurement results area is located at the bottom of the Control Panel. However, you can click the release button to detach the results window from the Control Panel for more convenient viewing.



- 1 Click the release button to detach the measurement results window.
- 2 Drag the measurement results window to a more convenient viewing position.
- 3 Drag the bottom border of the window to enlarge the area for viewing multiple measurement results without scrolling through the data.

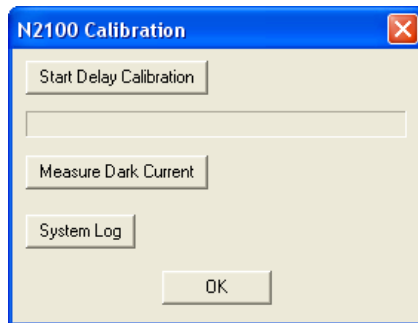
---

## Calibrating the N2100B

Clicking **Module Cal** displays the Calibration dialog box that provides access to the following features:



- Calibration of the delay lines used in vector sampling
- Dark current calibration
- System log



**Figure 3-4. Calibration Dialog Box**

### Delay Lines Calibration

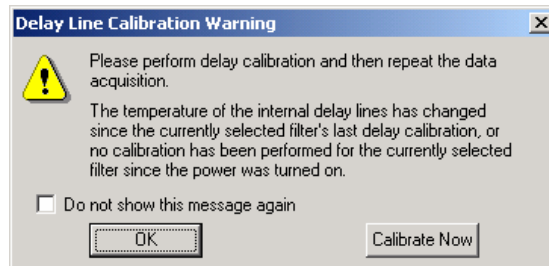
The Delay Calibration calibrates the delay timers that position the additional samples associated with the vector sampling technique. A PRBS signal must be applied to the N2100B for the calibration to work correctly. During the calibration, wait until the message “Calibration Complete” appears before continuing. You should run the Delay Calibration when the following situations occur:

- The first time an input path/filter is selected.
- When the internal module temperature has changed.
- When the Line rate changes. (Recommended only.)

### Temperature Monitor and Delay Calibration Notification

The DCA firmware monitors the internal module temperature and notifies the user or the client application when the temperature has changed enough that a delay recalibration is required.

In the Control Panel application, a pop-up window appears when data acquisition is performed, notifying you that recalibration is advisable. Calibration may be performed by clicking **Calibrate Now**, or at a later time using the Calibration dialog box described above.



**Figure 3-5. Calibration Warning Dialog Box**

Data acquisition functions that are in the ActiveX control and the module API return a status value, FG\_DELAYCAL\_REQD, indicating that recalibration is advisable.

### **Dark Current Calibration**

Calibration for optical signals depends on the optical detector's dark current. That is, the value measured with no light applied. The DCA can compensate for temperature- or time-related dark current drift. Use the following procedure to perform a dark current calibration:

- 1** Remove any optical signal from the optical input and cover the connector.
- 2** Click **Module Cal** on the Control Panel.
- 3** Click **Measure Dark Current**.

The DCA measures and stores the dark current measurement. This is relevant to extinction ratio and optical power measurements. To increase the accuracy of these values, a dark current calibration should be performed regularly.

### **System Log**

The **System Log** button allows you to view the log of system errors and events. The information collected depends on the Logging Level selection:

- Logging Level None. Logs nothing.
- Logging Level Fatal. Logs fatal errors only.
- Logging Level Error. Logs all errors.
- Warning. Logs all errors and warnings.
- Logging Level Info. Logs errors, warnings, and other events.

### **Calibrating the N2100B**

- Logging Level Debug. Logs errors, warnings, and internal information used for firmware debugging.
- Application Log. Associated with application-level firmware functions.
- System Log. Associated with system-level driver functions.
- Read Log. Displays the appropriate log file using Notepad. The file is deleted unless the 'Keep File' checkbox is selected.
- Clear Log. Clears the appropriate log. The 'Allow Clear' checkbox must be selected for this to be available.

---

**NOTE**

---

The logging setting should be set to Error, Fatal or None in normal use. Otherwise, the performance speed of the DCA will be affected.

Introduction	4-2
Oscilloscope Measurements Mode	4-5
Non Return to Zero Measurements Mode	4-16
Eye Mask Test Measurements Mode	4-37
Multiple Measurements Mode	4-44
Pattern Sequence D. J. Measurements Mode	4-46

---

## Measurements

---

## Introduction

This section describes how to control the various modes of operation for performing measurements:

- [“Oscilloscope Measurements Mode” on page 4-5.](#)
- [“Non Return to Zero Measurements Mode” on page 4-16.](#)
- [“Eye Mask Test Measurements Mode” on page 4-37.](#)
- [“Multiple Measurements Mode” on page 4-44.](#)
- [“Pattern Sequence D. J. Measurements Mode” on page 4-46.](#)

For information on all other aspects of using the Control Panel, [refer to “Configuring the DCA and Acquiring Data” on page 3-1.](#)

**Eye Diagram Display** The N2100B automatically performs accurate eye-diagram analysis to characterize the quality of transmitters from 1.063 Gb/s to 8.5 Gb/s. This chapter lists all possible measurements for each mode including:

- Definition (optical and electrical), including formula for parameter calculation
- Examples with screen shots (OC48 (STM4) signals were used)
- Display configuration
- Procedures for measurements using the API

Eye diagrams are a key figure of merit for most computer and communications system standards, including Gigabit Ethernet, Sonet, Infiniband, Rapid IO, PCI Xpress and others. The eye diagram provides a clear visual representation whether the interconnection, by itself, would meet the test specification for a given standard.

The Eye Diagram provides a longer term view of the signal, taking into account the relative time position of successive pulses. It provides a more thorough analysis of the cumulative effects of wander and jitter.

Eye diagrams are able to collect and display multiple waveforms. For a large number of waveforms (or blocks), the timing and pulse width variations are displayed as a widening of the eye-diagram traces, which must remain within the template to meet specification.

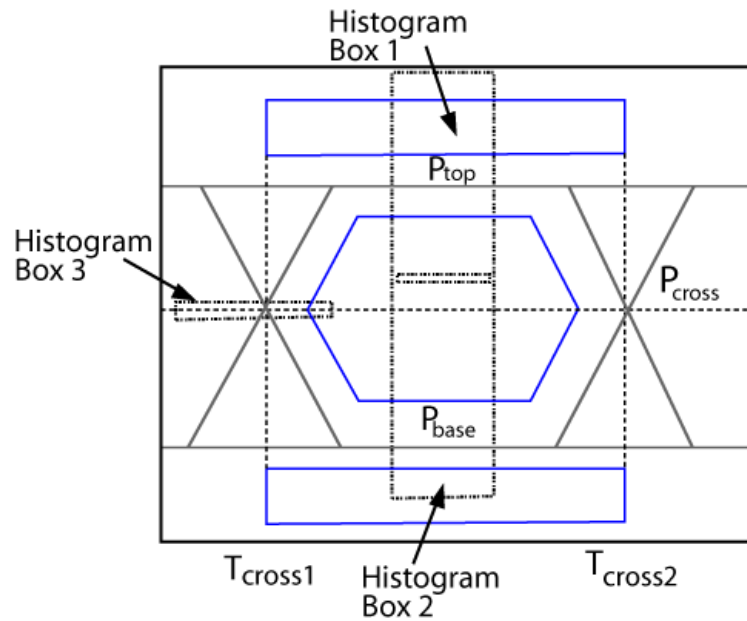


Figure 4-1. Eye Diagram

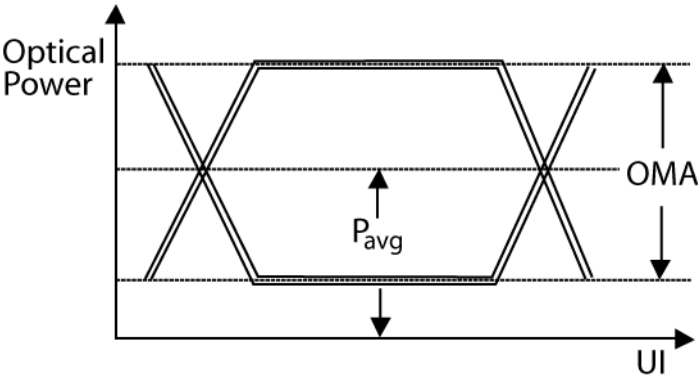


Figure 4-2. Optical Power



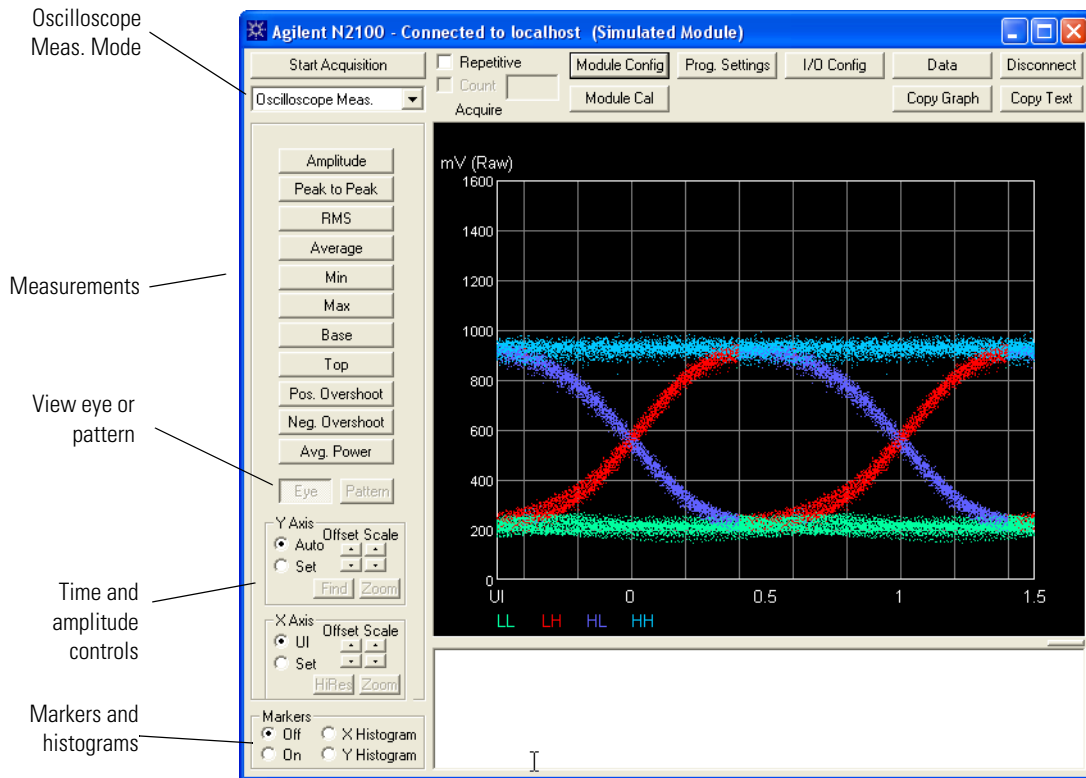
# Oscilloscope Measurements Mode

Oscilloscope Measurement mode offers the ability to select the display time scale and a selection of measurements related to the analog characteristics of the entire signal.

The following table summarizes the measurements the DCA can perform in Oscilloscope Mode. The DCA software installation includes an HTML document with full details of the ActiveX and API functions listed in the examples that follow in this section.

Table 4-1. Oscilloscope Mode Measurements

Measurement	Description
<a href="#">"Top" on page 4-8.</a>	The average of all sample values identified as high-to-high transitions.
<a href="#">"Base" on page 4-9.</a>	The average of all sample values identified as low-to-low transitions.
<a href="#">"Amplitude" on page 4-9.</a>	The difference between top level and base level of a displayed pulse waveform.
<a href="#">"Peak to Peak" on page 4-10.</a>	The difference between max and min of a displayed pulse waveform.
<a href="#">"RMS" on page 4-11.</a>	The RMS value of the waveform.
<a href="#">"Min" on page 4-12.</a>	The value of the sample point with the lowest amplitude.
<a href="#">"Max" on page 4-13.</a>	The value of the sample point with the highest amplitude.
<a href="#">"Positive Overshoot" on page 4-13.</a>	The amount by which the Max value exceeds the Top value, expressed as a percentage of the Amplitude.
<a href="#">"Negative Overshoot" on page 4-14.</a>	The amount by which the Min value falls below the Base value, expressed as a percentage of the Amplitude.
<a href="#">"Average Optical Power" on page 4-15.</a>	The average power read by the PIN diode (optical signals only).

**Oscilloscope Measurements Mode****Figure 4-3. Oscilloscope Measurement Mode Control Panel****X-Axis Control**

- **UI** always displays a single UI.
- **Set** displays the X axis with user-selectable time/division and offset.
- **Scale** zooms in and out.
- **Offset (UI)** adjusts the starting X value of the display.
- **HiRes** changes the offset by .05 UI or 0.005 UI.
- **Zoom** sets the closest pair of offset and scale settings to zoom within the area selected by the markers.

The offset and scale values are displayed on the bottom left side of the Eye-Diagram Display .

## **Y-Axis Control**

- **Auto** selects best fit scaling of eye diagram on screen.
- **Scale** zooms in and out.
- **Offset** creates an offset the eye within the display.
- **Find** auto-finds the trace on the screen. The Y-Axis scale and offset are adjusted accordingly.
- **Zoom** sets the closet pair of offset and scale settings to zoom within the area selected by the markers.

The amplitude values are displayed along the Y axis, with the units on top.

## **Markers**

When you switch on the Markers control, a pair of movable markers in both X-axis and Y-axis are displayed. The data at the position of the markers are displayed. The histogram function displays a histogram of the samples within the region bounded by the markers, selectable over X or Y values.

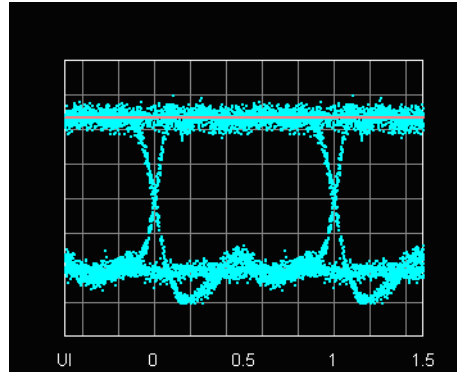
You can reposition the markers using the mouse. Select the marker with the left mouse button and move it by dragging the mouse. If you hold down the shift key while moving the mouse, the two markers will lock so that the distance between them is preserved.

The markers can also be used to control the scale of the display. The normal view (**UI**) presents the view of two UI. Switching to the **Set** option in the X Axis control allows you to zoom in and out on the eye diagram. In Y Axis (Auto) mode, the signal amplitude is scaled to best fit within the display screen.

---

## Top

Top is the average of all sample values identified as high-to-high transitions.



Top 660.1 mV

**Control Panel**

Click **Top** on the left side of the display.

**API Call**

FGDCAOscilloscopeAPI

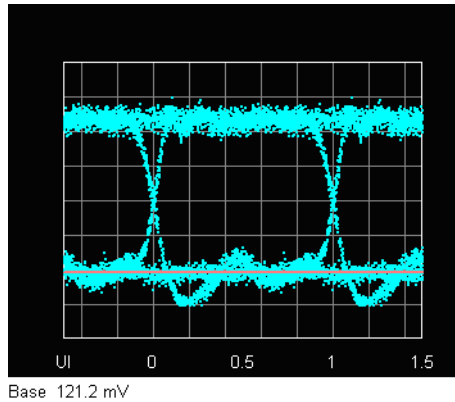
**ActiveX**

MeasOscTopLevel

---

## Base

Base is the average of all sample values identified as low-to-low transitions.



### Control Panel

Click **Base** on the left side of the display.

### API Call

FGDCAOscilloscopeAPI

### ActiveX

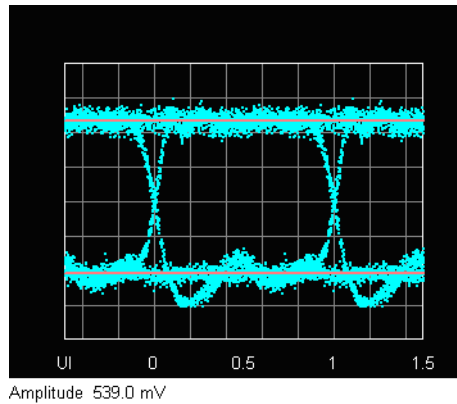
MeasOscBaseLevel

---

## Amplitude

Amplitude is the vertical difference between the Top and Base of the signal.

Top - Base = Amplitude

**Control Panel**

Click **Amplitude** on the left side of the display.

**API Call**

FGDCAOscilloscopeAPI

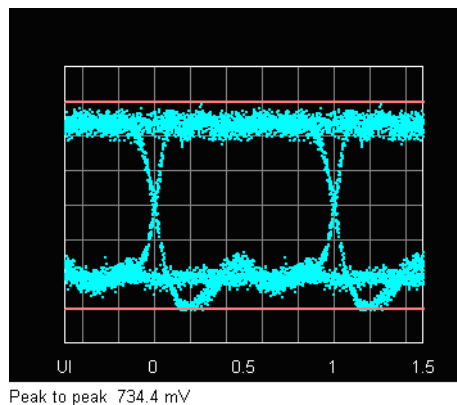
**ActiveX**

MeasOscAmplitude

---

## Peak to Peak

Peak-to-peak (pk-pk) is the difference between the maximum positive and the maximum negative amplitudes of a waveform.



**Control Panel**

Click **Peak to Peak** on the left side of the display.

**API Call**

FGDCAOscilloscopeAPI

**ActiveX**

MeasOscPeakToPeak

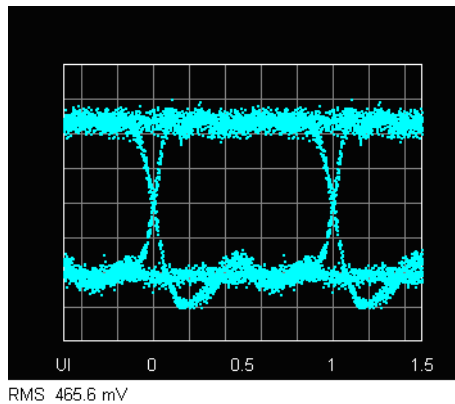
## RMS

Root-mean-square (rms) refers to the most common mathematical method of defining the effective voltage or current of an AC wave.

To determine rms value, four mathematical operations are carried out on the function representing the AC waveform:

- 1 Take the average of all samples.
- 2 Take the sum of (sample[i] - average)<sup>2</sup>.
- 3 Divide the sum by (# of samples).
- 4 Take the square root of the result.

$$value_{rms} = \sqrt{\left(\frac{1}{N}\right) \sum_{i=1}^N (sample[i] - average)^2}$$



**Oscilloscope Measurements Mode**

**Control Panel** Click **RMS** on the left side of the display.

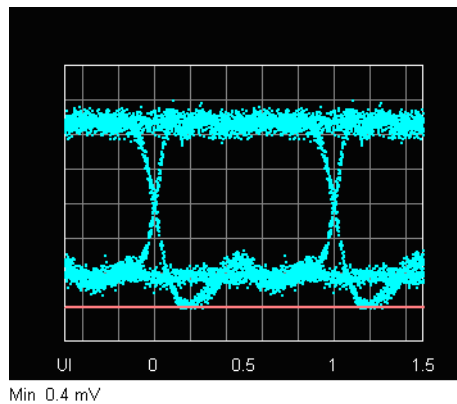
**API Call** FGDCAOscilloscopeAPI

**ActiveX** MeasOscRMS

---

**Min**

Min is the value of the sample point with the lowest amplitude.



**Control Panel** Click **Min** on the left side of the display.

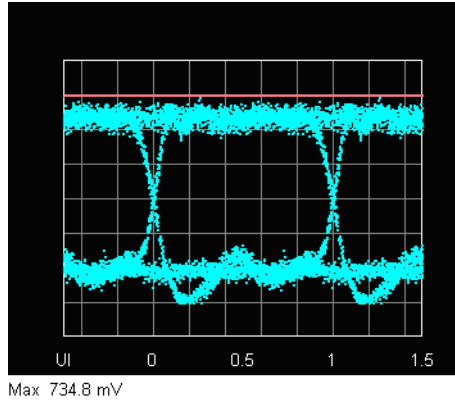
**API Call** FGDCAOscilloscopeAPI

**ActiveX** MeasOscMinLevel



## Max

Max is the value of the sample point with the highest amplitude.



### Control Panel

Click **Max** on the left side of the display.

### API Call

FGDCAOscilloscopeAPI

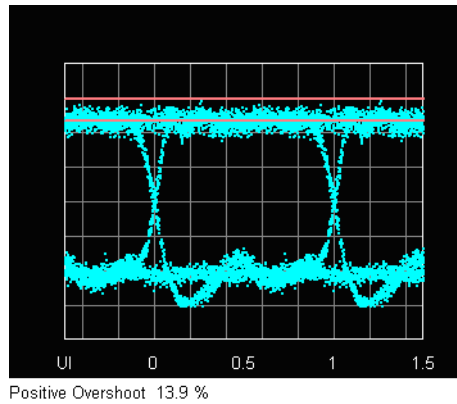
### ActiveX

MeasOscMaxLevel

## Positive Overshoot

Positive Overshoot is the difference between the maximum sample value and the signal's top level, expressed as a fraction of the signal amplitude.

$$Positive\ Overshoot = \frac{(Max - Top)}{(Top - Base)} \times 100$$

**Control Panel**

Click **Pos. Overshoot** on the left side of the display.

**API Call**

FGDCAOscilloscopeAPI

**ActiveX**

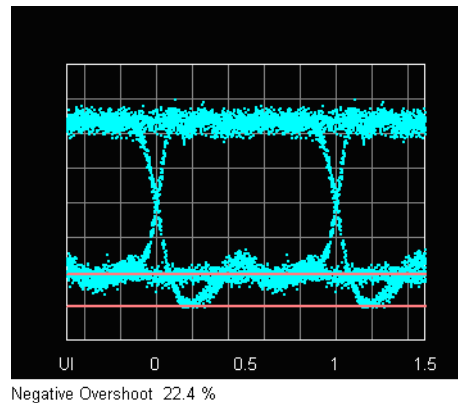
MeasOscPosOvershoot

---

## Negative Overshoot

Negative Overshoot is the difference between the minimum sample value and the signal's base level, expressed as a fraction of the signal amplitude.

$$\text{Negative Overshoot} = \frac{(\text{Base} - \text{Min})}{(\text{Top} - \text{Base})} \times 100$$



<b>Control Panel</b>	Click <b>Neg. Overshoot</b> on the left side of the display.
<b>API Call</b>	FGDCAOscilloscopeAPI
<b>ActiveX</b>	MeasOscNegOvershoot

---

## Average Optical Power

Applicable to optical signals only. The Average Optical Power is the true average component of an optical signal, expressed in microwatts or dBm (decibels relative to a power level of one milliwatt). This measurement results from the use of a hardware average-power monitor circuit rather than from the calculation of digitized waveform data.

<b>Control Panel</b>	Click <b>Avg. Power</b> on the left side of the display.
<b>API Call</b>	FGDCAAvgPowerAPI
<b>ActiveX</b>	MeasAvgPowerUW MeasAvgPowerDBM

---

## Non Return to Zero Measurements Mode

NRZ (Non-return-to-zero) Measurement Mode offers a selection of signal and timing measurements typically associated with NRZ signals. Re-acquisition of data is not required.

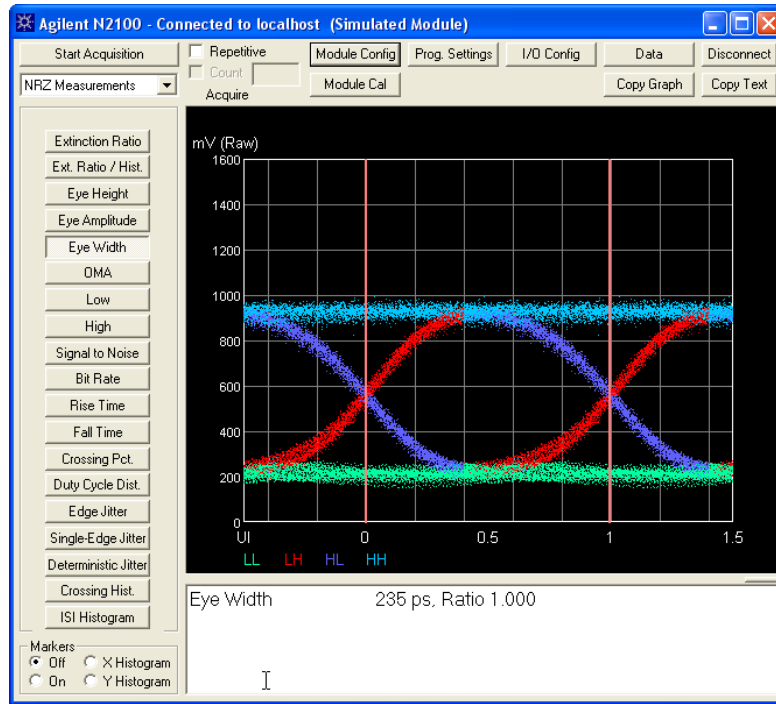
The following table summarizes the measurements the instrument is capable of performing in NRZ:

**Table 4-2. DCA NRZ Measurements (1 of 2)**

Measurement	Description
<a href="#">"Extinction Ratio" on page 4-18.</a>	For optical signals only. A measure of the ratio of optical power at the signal's '1' level to the power at the signal's '0' level.
<a href="#">"Eye Height" on page 4-20.</a>	Eye Height is the vertical opening of the eye measured as the difference between the -3 sigma value of the high samples within the NRZ window and the +3 sigma value of the low samples within the NRZ window. The measured value can be expressed as a ratio, percentage, or in dB.
<a href="#">"Eye Amplitude" on page 4-21.</a>	Difference between the average of the high samples within the NRZ window and the average of the low samples within the NRZ window.
<a href="#">"Eye Width" on page 4-22.</a>	Time between the latest possible occurrences of one eye crossing to the earliest possible occurrence of the next crossing.
<a href="#">"OMA (Optical Modulation Amplitude)" on page 4-23.</a>	Difference between the high and low levels measured in a window centered on the eye crossing.
<a href="#">"Low" on page 4-24.</a>	The average of all sample values within the NRZ window identified as low-to-low transitions.
<a href="#">"High" on page 4-25.</a>	The average of all sample values within the NRZ window identified as high-to-high transitions.
<a href="#">"Signal-to-Noise Ratio" on page 4-26.</a>	Ratio of the signal difference between high level and low level relative to the noise present at both levels.
<a href="#">"Bit Rate" on page 4-27.</a>	The rate of successive data transitions or potential data transitions of the input signal. Returns a value, depending on the reference clock selected:  Internal clock returns the line rate set by the user. External clock returns the frequency at the reference clock input. Recovered clock returns the measured line rate.

**Table 4-2. DCA NRZ Measurements (2 of 2)**

Measurement	Description
<a href="#">"Rise Time" on page 4-27.</a>	The mean transition time of the data on the upward slope of an eye diagram between two defined thresholds (for example, 20% to 80%) relative to signal amplitude (the full-UI Top and Base levels), or relative to the NRZ-window High and Low levels.
<a href="#">"Fall Time" on page 4-29.</a>	The mean transition time of the data on the downward slope of an eye diagram between two defined thresholds (for example, 90% and 10%) relative to the full-UI Top and Base levels, or the NRZ-window High and Low levels.
<a href="#">"Crossing Percentage" on page 4-30.</a>	The amplitude of the crossing level relative to the low and high levels.
<a href="#">"Duty Cycle Distortion" on page 4-30.</a>	The time separation between the rising edge and falling edge at the 50% level of the eye diagram.
<a href="#">"Edge Jitter" on page 4-32.</a>	A measure of the variation in time position of signal transitions, measured at the crossing.
<a href="#">"Single-Edge Jitter" on page 4-33.</a>	A measure of the variation in rising or falling transitions only in time position of signal transitions. A peak to peak value as well as an rms value is returned.
<a href="#">"Deterministic Jitter" on page 4-34.</a>	The time difference between transitions that immediately follow another transition (LHL and HLH) and transitions that immediately follow an interval with no transition (LLH and HHL).
<a href="#">"Crossing Hist." on page 4-35.</a>	Histogram showing the distribution of samples at the crossing point.
<a href="#">"ISI Histogram" on page 4-35.</a>	Histogram showing the distribution of the four different transition types separately.

**Non Return to Zero Measurements Mode****Figure 4-4. NRZ Measurements Mode Control Panel**

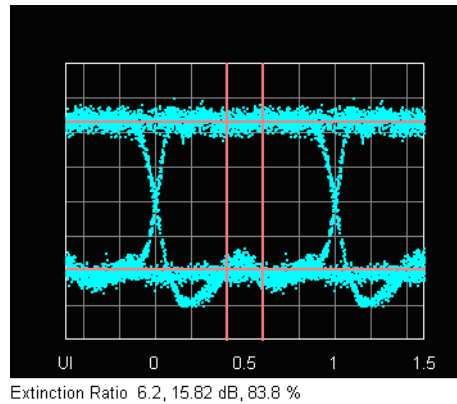
## Extinction Ratio

Extinction ratio is the ratio of the 'On' optical level (High) to the 'Off' optical level (Low) measured within the NRZ window. The result is expressed as Ratio, dB, and Percentage. These are calculated using the following formulas:

$$ER_{ratio} = \frac{High}{Low}$$

$$ER_{dB} = 10\log\left(\frac{High}{Low}\right)$$

$$ER_{\%} = \frac{(High - Low)}{High} \times 100$$

**Control Panel**

Click **Extinction Ratio** on the left side of the display.

**API Call**

FGNRZExtinctionRatioAPI

**ActiveX**

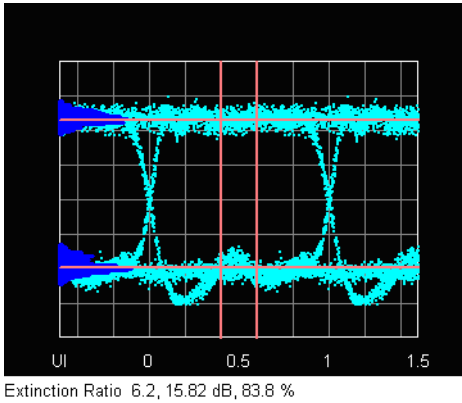
MeasNRZExtinctionRatio

MeasNRZExtinctionRatioDB

---

## Extinction Ratio with histogram

This measurement displays the same results as the Extinction Ratio measurement; additionally it displays the distribution of sample points within the NRZ window using a histogram.



**Control Panel** Click **Ext. Ratio/Hist.** on the left side of the display.

**API Call** N/A

**ActiveX** MeasNRZExtinctionRatioHist  
MeasNRZExtinctionRatioDBHist

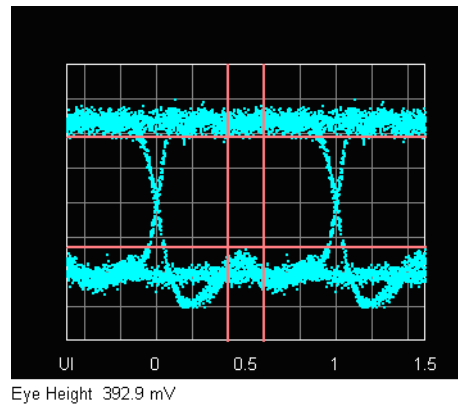
---

## Eye Height

Eye Height is the vertical opening of the eye and measured as the difference between the -3 sigma value of the high samples within the NRZ window and the +3 sigma value of the low samples within the NRZ window.

$$Eye\ Height = (P_{top} - 3\sigma_{top}) - (P_{base} + 3\sigma_{base})$$



**Control Panel**

Click **Eye Height** on the left side of the display.

**API Call**

FGNRZEyeHeightAPI

**ActiveX**

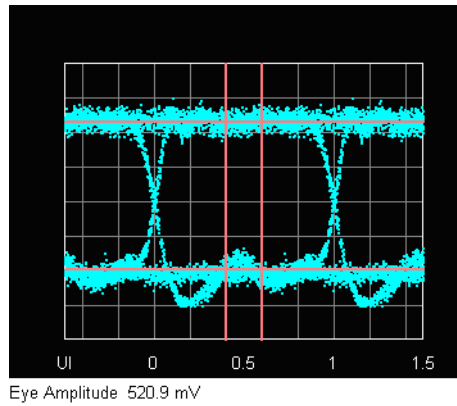
MeasNRZEyeHeight

---

## Eye Amplitude

Eye amplitude is the difference between the logic 1 level and the logic 0 level histogram mean values of an eye diagram measured within the NRZ window.

$$amplitude = high - low$$

**Non Return to Zero Measurements Mode****Control Panel**

Click **Eye Amplitude** on the left side of the display.

**API Call**

FGNRZEyeAmplitudeAPI

**ActiveX**

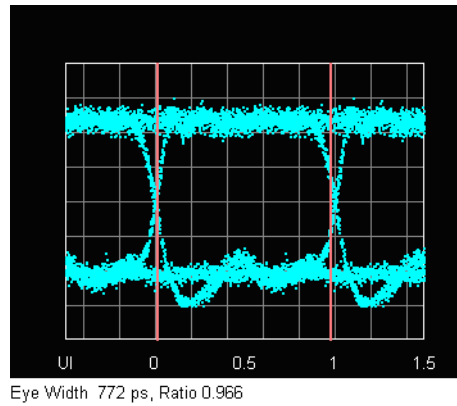
MeasNRZEyeAmplitude

---

## Eye Width

Eye Width is the horizontal opening of the eye.

$$Eye\ Width = (T_{cross2} - 2\sigma_{cross2}) - (T_{cross1} + 2\sigma_{cross1})$$

**Control Panel**

Click **Eye Width** on the left side of the display.

**API Call**

FGNRZEyeWidthAPI

**ActiveX**

MeasNRZEyeWidth

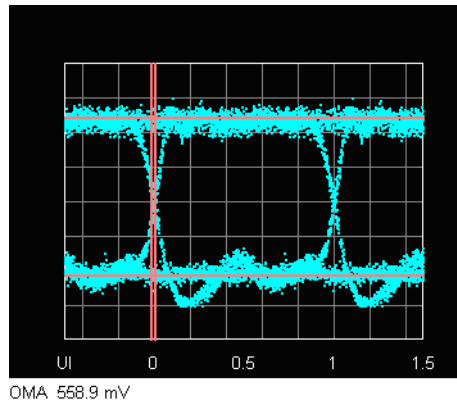
---

## OMA (Optical Modulation Amplitude)

Optical modulation amplitude is defined as the difference in power between the high and low levels as measured within the OMA window, centered on the crossing.

$$OMA = P_1 - P_0$$

## Non Return to Zero Measurements Mode



### Control Panel

Click **OMA** on the left side of the display.

### API Call

FGNRZOMAAPI

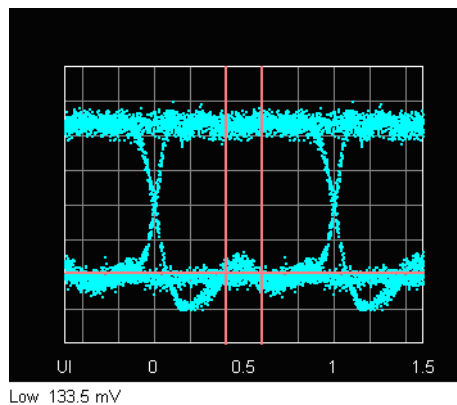
### ActiveX

MeasNRZOMA

---

## Low

Low measures within the NRZ window the average of LL samples during the interval defined by the window.



**Control Panel** Click **Low** on the left side of the display.

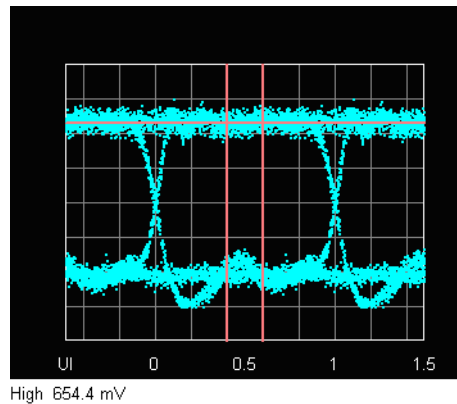
**API Call** FGNRZLowAPI

**ActiveX** MeasNRZLowLevel

---

## High

High measures within the NRZ window the average of HH samples during the interval defined by the window.



**Control Panel** Click **High** on the left side of the display.

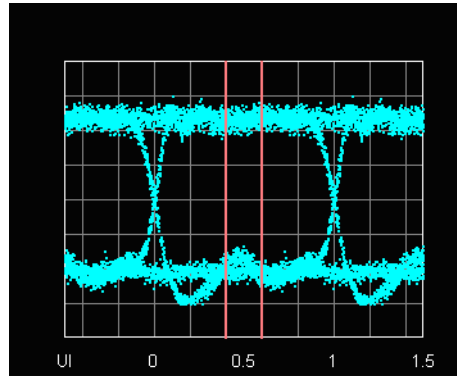
**API Call** FGNRZHighAPI

**ActiveX** MeasNRZHighLevel

---

## Signal-to-Noise Ratio

Signal-to-Noise is the ratio of the signal difference between high level and low level relative to the noise present at both levels.



Signal to Noise Ratio 10.5

**Control Panel**

Click **Signal to Noise** on the left side of the display.

**API Call**

FGNRZSNRAPI

**ActiveX**

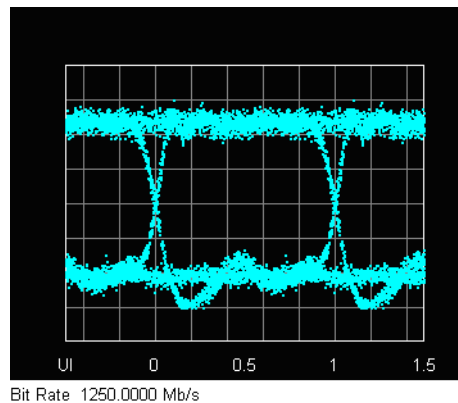
MeasNRZSNR

---

## Bit Rate

The value returned depends on the reference clock selected.

- Recovered: the measured frequency of the recovered clock.
- External: the measured frequency of the applied reference clock. The Control Panel also displays the implied bit rate.
- Internal: the bit rate as set..



### Control Panel

Click **Bit Rate** on the left side of the display.

### API Call

FGNRZBitRateAPI

### ActiveX

GetBitRate

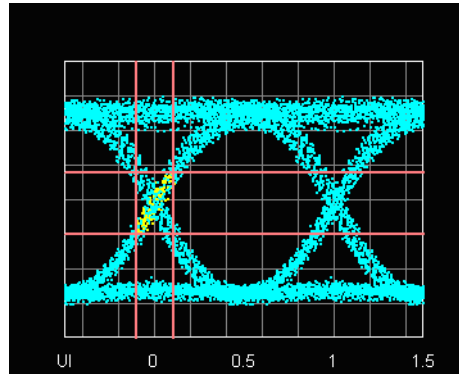
---

## Rise Time

The Rise Time measurement is made on the rising edge between the thresholds selected by the NRZ Threshold configuration setting. Choices are 10% to 90%, 20% to 80%, and 30% to 70% of either the total Waveform Amplitude (Full Period Top and Base), or the NRZ-Windowed High and Low Levels.

**Non Return to Zero Measurements Mode**

$$Rise\ Time = \bar{X}_{high\ threshold} - \bar{X}_{low\ threshold}$$



Rise Time 84.9 ps

**Control Panel**

Click **Rise Time** on the left side of the display.

**API Call**

FGNRZRiseTimeAPI

**ActiveX**

MeasNRZRiseTime

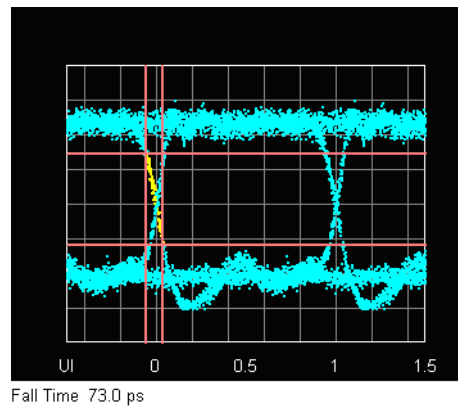


---

## Fall Time

The Fall Time measurement is made on the falling edge between the thresholds selected by the NRZ Threshold configuration setting. Choices are 10% to 90%, 20% to 80%, and 30% to 70% of either the total Waveform Amplitude (Full Period), or the NRZ-Windowed High and Low Levels.

$$Fall\ Time = \bar{X}_{low\ threshold} - \bar{X}_{high\ threshold}$$

**Control Panel**

Click **Fall Time** on the left side of the display.

**API Call**

FGNRZFallTimeAPI

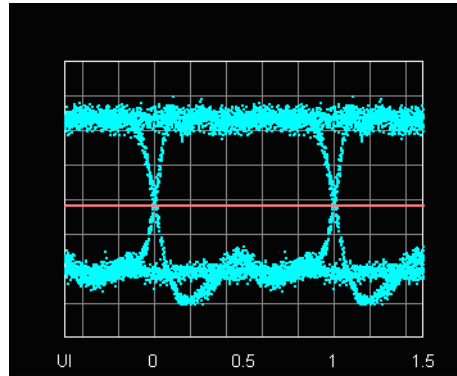
**ActiveX**

MeasNRZFallTime

---

## Crossing Percentage

Crossing Percentage is the signal level at the point where the rising and falling edges cross expressed as a percentage of the signal amplitude.



### Control Panel

Click **Crossing Pct.** on the left side of the display.

### API Call

FGNRZCrossingPercentAPI

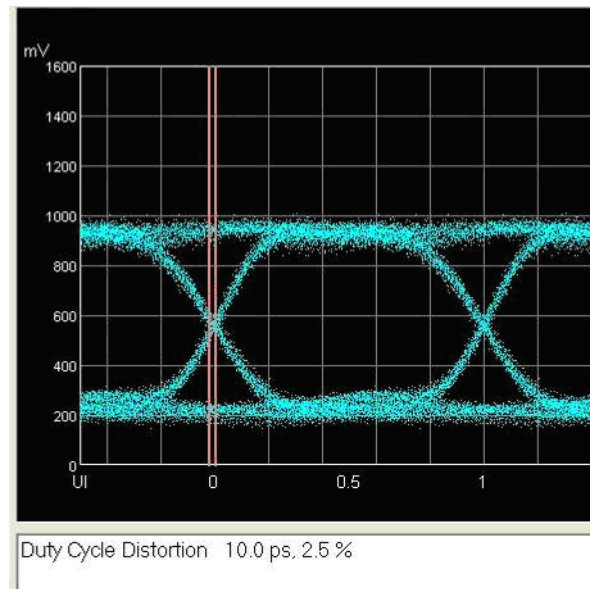
### ActiveX

MeasNRZCrossingPercent

---

## Duty Cycle Distortion

Duty Cycle Distortion is the time separation between the rising edge and falling edge at the mid-level of the eye diagram. The 50% threshold is calculated either on the total waveform amplitude (full period), or the NRZ-windowed high-low levels.

**Control Panel**

Click **Duty Cycle Distortion** on the left side of the display.

**API Call**

FGNRZDutyCyDistortionAPI

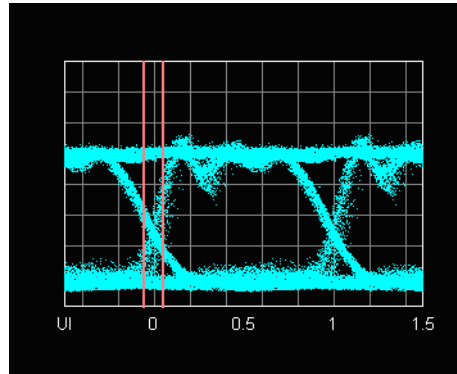
**ActiveX**

MeasNRZDutyCycleDistortion

---

## Edge Jitter

Edge Jitter is a measure of the difference in the time domain of the earliest transitioning sample and the latest. This is measured at the crossing point.



Jitter 41.5 ps Peak to Peak, 7.8 ps RMS

### Control Panel

Click **Edge Jitter** on the left side of the display.

### API Call

FGNRZJitterAPI

### ActiveX

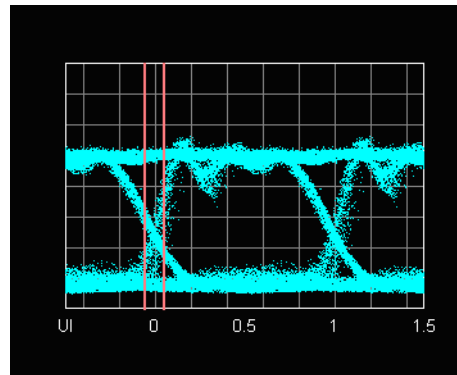
MeasNRZJitterPeakToPeak

MeasNRZJitterRMS

---

## Single-Edge Jitter

Single-Edge rising Jitter is a measure of the difference in the time domain of the earliest transitioning rising point and the latest. This is measured at the crossing point. The single edge falling jitter is also available.



Jitter 41.5 ps Peak to Peak, 7.8 ps RMS

### Control Panel

Click **Single Edge Jitter** on the left side of the display.

### API Call

FGNRZFallJitterAPI, FGNRZRiseJitterAPI

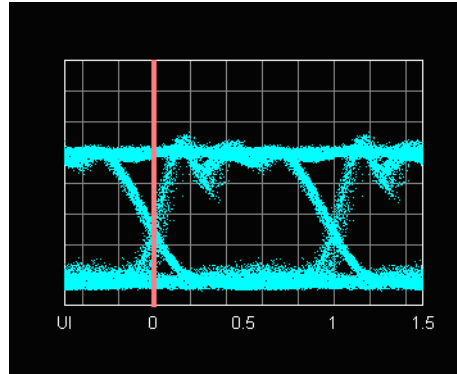
### ActiveX

MeasNRZSingleEdgeJitterPP  
MeasNRZSingleEdgeJitterRMS

---

## Deterministic Jitter

Deterministic Jitter is a measure of data dependent jitter.



Deterministic Jitter Rise 14.7 ps, Fall -1.8 ps

### Control Panel

Click **Deterministic Jitter** on the left side of the display.

### API Call

FGNRZDeterministicJitterAPI

### ActiveX

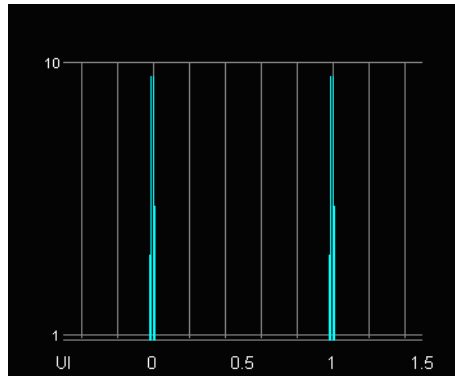
MeasNRZDeterministicJitterRise

MeasNRZDeterministicJitterFall

---

## Crossing Hist.

Crossing Histogram shows the distribution of samples at the crossing point.



### Control Panel

Click **Crossing Hist.** on the left side of the display.

### API Call

N/A

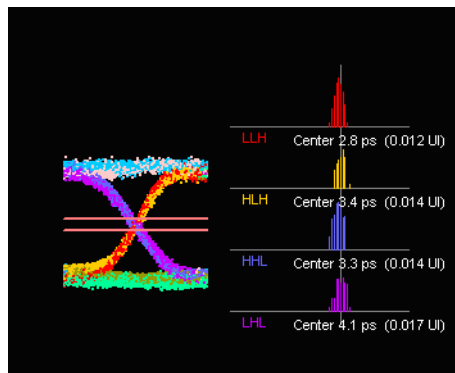
### ActiveX

N/A

---

## ISI Histogram

The ISI Histogram shows the distribution at the crossing points for LLH, HLH, HHL and LHL patterns separately.



**Non Return to Zero Measurements Mode**

**Control Panel**

Click **ISI Histogram** on the left side of the display.

**API Call**

N/A

**ActiveX**

N/A



---

## Eye Mask Test Measurements Mode

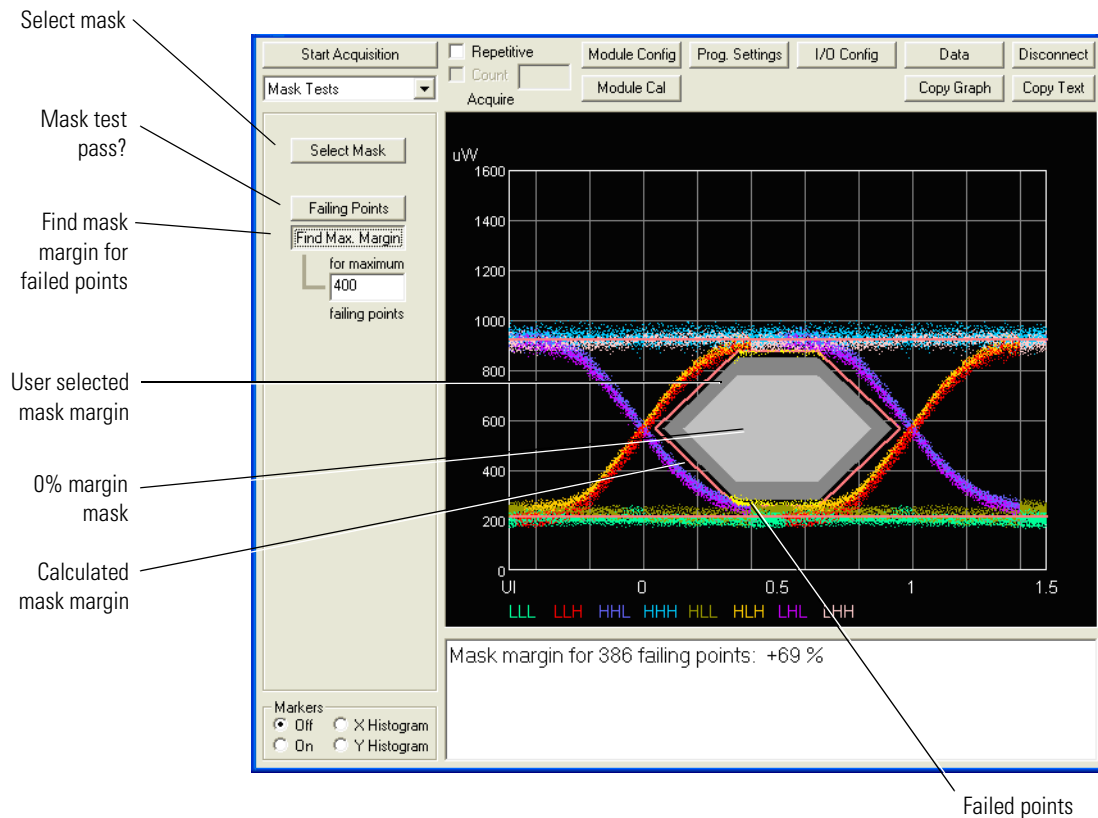
The N2100B allows the Host Application to select an eye mask and also to perform a comparison of the acquired NRZ Eye against the selected eye mask.

Use Mask Test mode to determine whether the signal meets defined mask requirements and calculate how much margin there is before failure.

- [“Mask Selection - Industry Standard and User Defined” on page 4-39](#)
- [“Pass/Fail” on page 4-42](#)
- [“Find Max Margin” on page 4-43](#)

**Eye Mask Test Measurements Mode**

Click **Select Mask** to select a Mask.



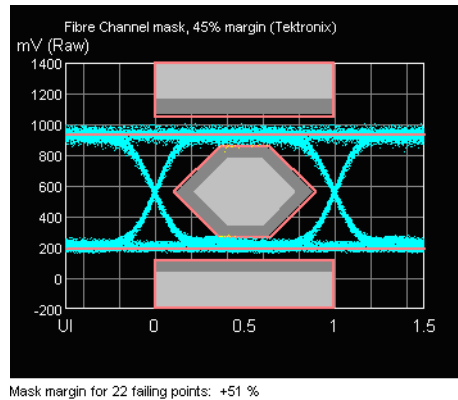
**Figure 4-5. Mask Tests Mode Control Panel**

**Failing Points Button** If you click **Failing Points**, the software reports any points that fall in the mask, or in the mask with margin applied, if a margin is selected. This reporting will be recalculated each time the Start Acquisition button is clicked or continuously in Repetitive Acquisition Mode. In the Trace Display, these reported points are highlighted in yellow. In the Results Display Area, the appropriate text is displayed: Test PASSED, or Test FAILED with the number of points that failed subtotaled by individual mask region. For example:

Test FAILED: 7 points of 16384 [45% margin]  
Center 7, Upper 0, Lower 0

### Find Max Margin Button

The **Find Max Margin** button allows you to find the maximum mask margin that results in no more than a specified number of failing points. The figure below shows an example where the user wants to determine the maximum mask margin available for the highest number of failing points, but no more than 25. The answer in this example is +51% for 22 points. Setting the margin below this level would result in less failing points total. Setting the margin above this level would result in fewer failing points.



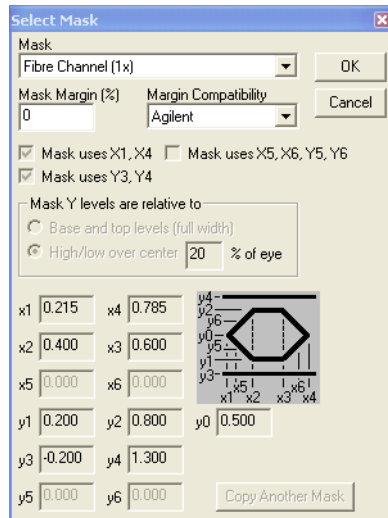
**Figure 4-6. Find Max Mask Margin**

The Find Max Margin calculates results without the need to reacquire data. This allows you to converge on a desired answer by trying different threshold values and the clicking the Find Max Margin for each one. When no data are present the error message NO DATA is displayed.

---

## Mask Selection - Industry Standard and User Defined

Clicking **Select Mask** invokes the Select Mask dialog box shown below. This allows you to select from a set of industry standard masks, or create a custom mask for your specific requirements. The predefined masks are listed in [Table 4-3 on page 4-41](#).

**Eye Mask Test Measurements Mode****Figure 4-7. Selecting a Mask - Industry Standard or User Defined**

The **Copy Another Mask** button allows you to easily create a custom mask by using an existing mask as a starting template. The existing mask can be either a standard mask or one that you previously defined. The new Mask is saved under a different mask number (#) after clicking **OK**.

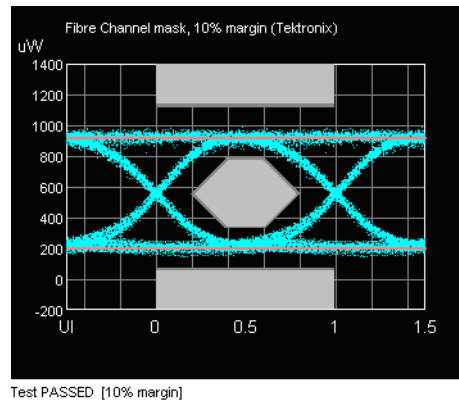
**Table 4-3. DCA Predefined Mask Definitions**

MASK	x1	x2	x3	x4	x5	x6	y0	y1	y2	y3	y4	y5	y6	Window
stm-1	0.15	0.35	0.65	0.85			0.5	0.2	0.8					Full UI
stm-4	0.25	0.4	0.6	0.75			0.5	0.2	0.8					Full UI
stm-16		0.4	0.6				0.5	0.25	0.75					Full UI
Gb Ether- net	0.22	0.375	0.625	0.758			0.5	0.2	0.8	-0.2	1.3			20%
Fibre chan- nel (1x)	0.215	0.4	0.6	0.785			0.5	0.2	0.8	-0.2	1.3			20%
Fibre chan- nel (2x,4x)	0.22	0.4	0.6	0.78			0.5	0.2	0.8	-0.2	1.3			20%
Fibre chan- nel (8x)	0.250	0.450	0.550	0.750	0.400	0.600	0.500	0.320	0.680	-0.400	1.400	0.350	0.650	20%
Fibre chan- nel (10x)	0.300	0.400	0.600	0.700			0.500	0.250	0.750	-0.400	1.400			20%

---

## Pass/Fail

A mask test fails if any points fall inside the mask regions. When a positive mask margin is applied, the mask regions are enlarged so that points falling within the original mask or the margin fail. When a negative margin is applied, the mask regions are reduced so that only points that fall within the reduced mask (original mask minus margin) fail.



### Control Panel

The Eye Mask Test is done automatically after acquisition in this mode. Click on the **Failing Points** button for this display.

### API Call

FGDCAMaskTestAPI

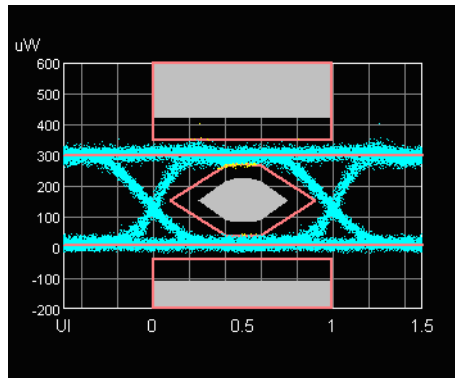
### ActiveX

GetMaskTestFailPoints

---

## Find Max Margin

This measurement finds the maximum mask margin value that results in no more than the specified number of sample points that fail the mask test.



Mask margin for 91 failing points: +61 %

### Control Panel

After an Acquisition, enter the desired number of failing points, and click on **Find Max Margin** button to perform the test. This sequence can be repeated on the same set of test data to determine the margin for several failing point thresholds.

### API Call

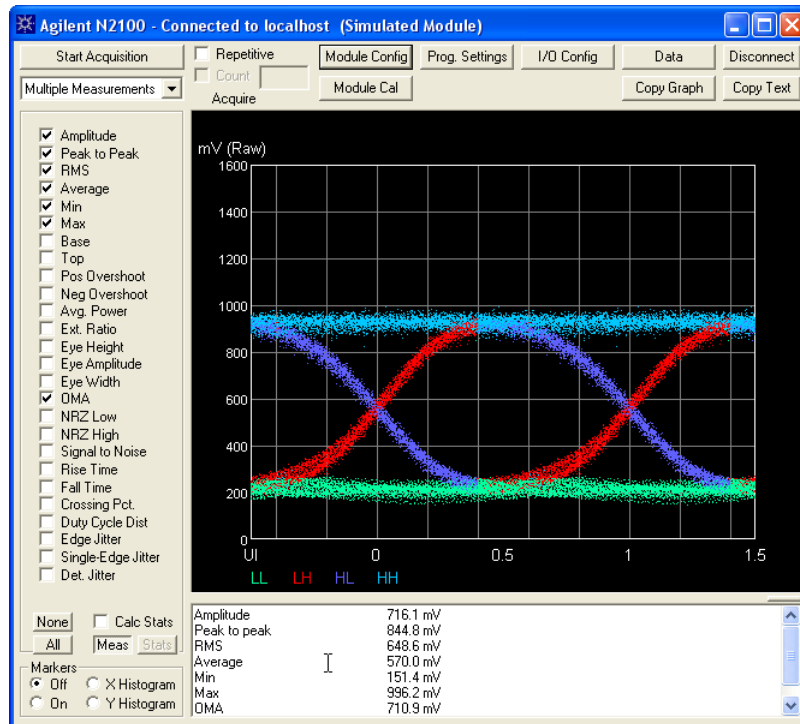
N/A

### ActiveX

FindThresholdMaskMargin

## Multiple Measurements Mode

The Multiple Measurements Mode allows you to gather and view all possible measurements on the eye diagram from one capture. After you select the measurements that you want to view, the Control Panel displays the results concurrently.



**Figure 4-8. Multiple Measurements Mode Control Panel**



### Enabling Statistics on the Multiple Measurements Tab

The Multiple Measurements tab includes access to the statistics function. When **Calc Stats** is selected the following values are collected over the measurements made during the current repetitive acquisition operation, for each of the parameters selected.

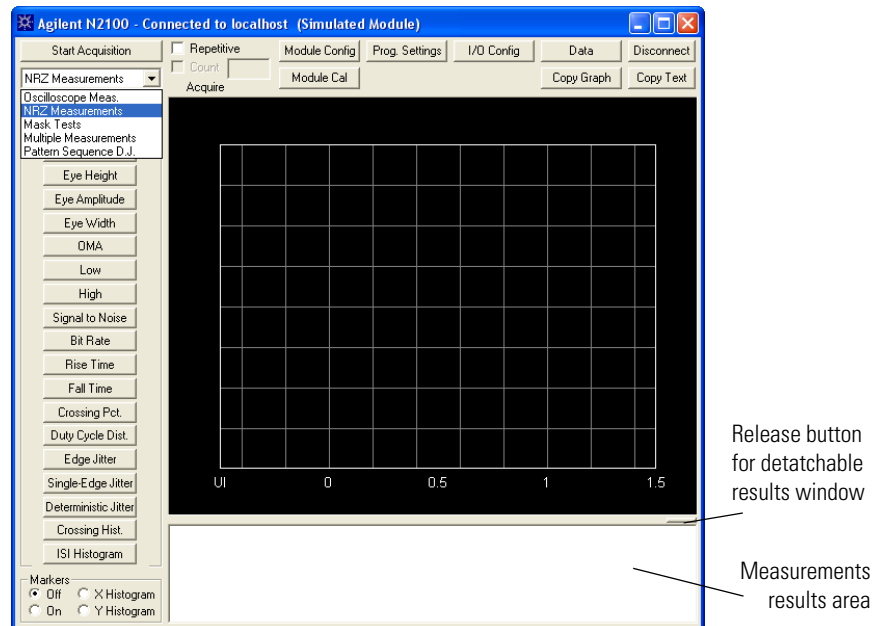
- Minimum
- Maximum
- Average
- Standard deviation

Clicking **Meas / Stats** switches the text window between the measured values and the statistics.

When the statistics results are displayed, clicking the **Copy Text** button copies the data displayed. Fields are separated by a single tab character.

### Viewing and moving the measurement results window

- 1 Click the release button to detach the measurement results window.



- 2 Drag the measurement results window to a more convenient viewing position.
- 3 Drag the bottom border of the window to enlarge the area for viewing multiple measurement results without scrolling through the data.

## Pattern Sequence D. J. Measurements Mode

The N2100B is capable of performing a pattern capture without the need for a pattern trigger.

A Pattern Sequence D.J. mode offers deterministic (pattern-dependent) jitter measurements on selected multibit sequences in the acquired data pattern, between specified bit sequences. This mode only applies to data acquired with pattern acquisition. For information on pattern acquisition, [refer to “Pattern Acquisition” on page 6-1](#). This section includes procedures for the following measurements:

- “To acquire a pattern” on page 4-48.
- “To apply software filters with pattern acquisition” on page 4-49.
- “D.J. based on Pattern Acquisition” on page 4-49.

Basic deterministic jitter measurement measures the average jitter between rising transitions 001 (transition following a steady low), and 101 (transition following a transition), and between falling transitions 110 and 010.

Pattern sequence deterministic jitter measurement measures the average jitter between two specified transition sequences up to 16 bits in length.

The DCA Pattern Sequence D.J. mode includes the following measurement capabilities:

- Standard deterministic measures the time between the earliest-occurring and latest-occurring rising transition, and the time between the earliest-occurring and latest-occurring falling transition.
- Custom deterministic jitter that is the deterministic jitter between two specified bit pattern sequences, or between one specified bit pattern sequence and all other bit pattern sequences.

The transition labeled, "Fast" refers to the transitions normally called HLH and LHL. The transition labeled, "Slow" refers to the transitions normally called LLH and HHL. These controls allow you to redefine these

The screenshot shows the 'Start Acquisition' dialog box. The 'Pattern Sequence D.J.' dropdown is selected. Below it are 'Standard' and 'Custom' buttons. Under 'Transition 1 ("Fast")', the '101 (Length 3)' radio button is selected, with a text field showing '101'. Under 'Transition 2 ("Slow")', the 'Length 2' dropdown is selected, with a text field showing '1001'. There is also an 'All' radio button. An 'Update' button is at the bottom.

transitions with arbitrary bit sequences for the purposes of this measurement. For example, a sequence consisting of a low-to-high transition preceded by exactly four low states would be expressed as 100001.

For the "Slow" transition, the Length selector refers to the number of low states preceding a rising transition, or the number of high states preceding a falling transition. The sequence 1001 corresponds to length 2, and the sequence 100001 corresponds to length 4.

Results of these measurements are reported as follows:

D.J., 1001 to 101 Rise 7.3 ps, Fall 4.3 ps

The eye display changes so that the only transition samples shown are those that match the specified patterns. If display colors are enabled, these are shown in the corresponding transition colors.

If you select the **All** choice, the measurement is made between the specified "Fast" pattern and all patterns of the form 10...01 with lengths between 2 and 11. When no transitions are found for a specified length, no result is reported for that length. The following example is a typical output from this measurement:

Reference pattern 101

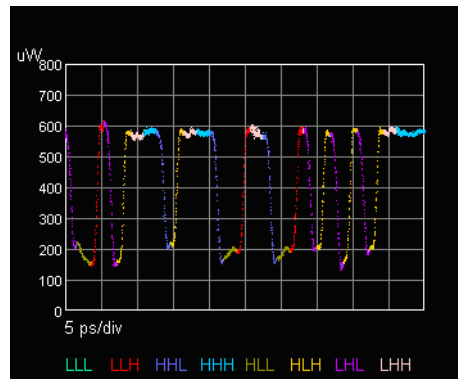
Length 2	Rise 7.3 ps, Fall 4.3 ps
Length 3	Rise 4.5 ps, Fall 4.7 ps
Length 4	Rise 1.3 ps, Fall 4.2 ps
Length 5	Rise -0.7 ps, Fall -0.2 ps
Length 6	Rise -1.9 ps, Fall 5.0 ps
Length 7	Rise -5.4 ps, Fall 9.5 ps
Length 8	Rise -7.0 ps, Fall 5.6 ps

## To acquire a pattern

### Control Panel

To display the following pattern sequence, perform the following steps:

- 1 In the **Module Config** subpanel, set the **Pattern Acquisition** option to **Enabled (No filter)** or the the desired filter selection.
- 2 Set either the pattern length or the pattern type.
- 3 To view the acquired pattern, set the instrument to the **Oscilloscope Meas.** mode and ensure that the **Pattern** option is selected.



### API Call

N/A

### ActiveX

PatternAcqFilterSel  
 PatternAcqLength  
 LoadPatternAcqFilterFile  
 LoadPatternAcqFilterFileByValue  
 MeasPatternAcqDetJitterRise  
 MeasPatternAcqDetJitterFall  
 MeasPatternAcqStdDetJitterRise  
 MeasPatternAcqStdDetJitterFall

---

## To apply software filters with pattern acquisition

<b>Control Panel</b>	To apply the software filters, perform the following steps:  <ol style="list-style-type: none"><li>1 In the Module Config sub panel, set the Pattern Acquisition Option to the desired filter.</li><li>2 Set either the pattern length or the pattern type.</li></ol>
<b>API Call</b>	N/A
<b>ActiveX</b>	PatternAcqFilter Sel PatternAcqLength LoadPatternAcqFilterFile LoadPatternAcqFilterFileByValue

---

## D.J. based on Pattern Acquisition

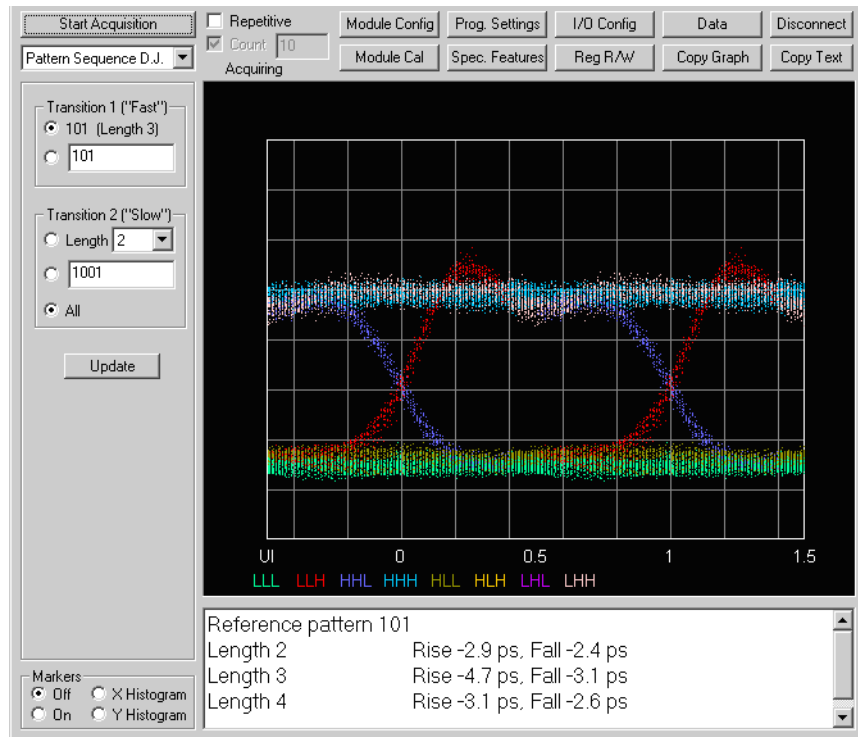
The Pattern Acquisition capability provides a Standard and a Custom Deterministic Jitter measurement.

The Standard D.J. measurement displays the maximum time deviations found among all rising transitions, and among all falling transitions, without the requirement of selecting a sequence. The result is one number each for rising and falling transitions.

The Custom deterministic jitter, as derived by two particular bit sequences, can be extracted using the pattern sequence capture option. You are allowed to define two sequences to compare. You are also allowed to define one base

**Pattern Sequence D. J. Measurements Mode**

sequence which can then be compared against all other sequences. The difference in ps of the crossing point between the two sequences under test is returned.

**Control Panel**

Select the **Pattern Sequence DJ** option in the main panel.

**API Call**

N/A

**ActiveX**

MeasPatternAcqStdDetJitterRise  
 MeasPatternAcqStdDetJitterFall  
 MeasPatternAcqDetJitterRise  
 MeasPatternAcqDetJitterFall

Introduction 5-2

Program Settings 5-3

Copying and Saving Displayed Test Results 5-11

---

## Displaying and Saving Results

---

## Introduction

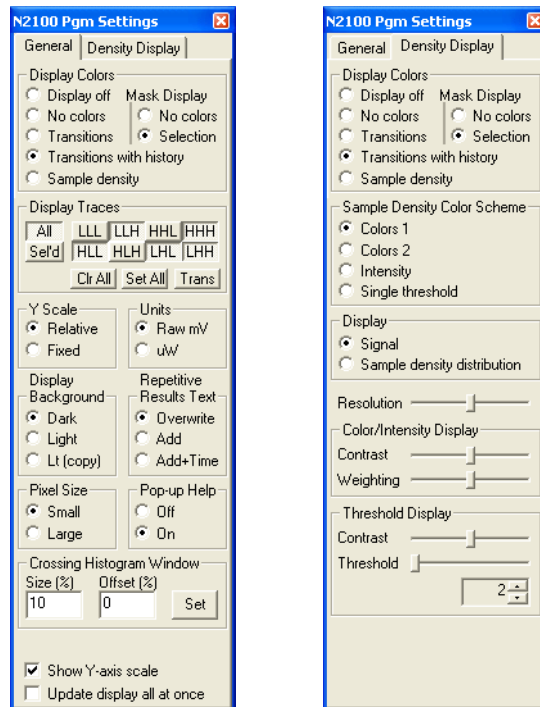
This chapter describes how to display the measurement results on the Control Panel and how to copy or save those displayed results.

- [“Program Settings” on page 5-3.](#)
- [“Copying and Saving Displayed Test Results” on page 5-11.](#)



## Program Settings

Use the Program Settings button on the Control Panel to change how data are displayed in the Control Panel. The dialog box has two tabs (General and Density Display), with Display Colors and Mask Display settings common to both.



**Figure 5-1. Program Settings dialog box**

## Settings on the General tab

### Display Colors

Controls the presentation of the sampled data through the following choices:

- Display Off. Prevents the window from being displayed, saving the time required to format and present the measured data.
- No Colors. All samples are displayed using the same color.
- Transitions. Samples are displayed using four colors to represent the different transition types: LL, LH, HL, and HH.
- Transitions with history. Samples are displayed using eight colors to represent transition types: LLL, LLH, LHH, LHL, HLL, HHL, HLH, and HHH. For this selection, the actual sample is taken on the transition between the latter two bits.
- Sample Density. The sample is displayed based on whichever option has been selected within the Sample Density tab.

### Mask Display

Controls the mask presentation through the following choices:

- No Colors. Sets the display for the mask test to the standard blue pixel display. In this mode, sample points failing the mask test are highlighted.
- Selection. Sets the Mask display to the selection in the display colors option.

---

## Settings on the General tab

### Display Traces

Controls the trace presentation through the following choices:

- All. All transitions are displayed.
- Sel'd. Only transitions selected using the LLL-HHH buttons are displayed.
- LLL-HHH. Allows you to switch individual traces On or OFF. The warning, 'Some traces not displayed' appears above the Trace Display Area when not all of the transitions are selected.
- Clr All. Sets all the LLL-HHH buttons to the OFF state.
- Set All. Sets all the LLL-HHH buttons to the ON state.
- Trans. Sets all the LLL-HHH buttons that represent a LH or HL transition to ON.

<b>Y-Scale</b>	<p>Controls the presentation of the Y-Axis through the following choices:</p> <ul style="list-style-type: none"><li>• Relative. Only displays the used part of the Y-axis.</li><li>• Fixed. Always includes the zero value on the Y-axis.</li></ul>
<b>Units</b>	<p>Controls the unit of measurement through the following choices:</p> <ul style="list-style-type: none"><li>• Signal: mV (electrical) or <math>\mu</math>W (optical). For a DCA with the optical input selected, the option is <math>\mu</math>W. For a DCA with an electrical input selected, the option is mV. Both of these are calibrated options.</li><li>• Raw: mV. In this mode, the internal (non-calibrated) mV value of the instrument is returned.</li></ul>
<b>Display Background</b>	<p>Controls the eye diagram background lighting through the following choices:</p> <ul style="list-style-type: none"><li>• Dark. The eye diagram has a dark background.</li><li>• Light. The eye diagram has a light background.</li><li>• Lt (Copy). The eye diagram is displayed with a dark background, but it is copied to the Windows clipboard with a light background for pasting into a document.</li></ul>
<b>Pixel Size</b>	<p>Small/Large. Selects the pixel size of the sample points that are drawn: "small" gives a more precise display; "large" gives a brighter display.</p>
<b>Repetitive Results Text</b>	<p>Controls the presentation of repeated measurement results through the following choices:</p> <ul style="list-style-type: none"><li>• Overwrite. During a repetitive acquisition sequence, each measurement is overwritten by the subsequent measurement in the display window.</li><li>• Add. Each measurement overwrites the previous measurement in the display window.</li><li>• Add + Time. The list of measurements is time stamped. As each measurement result is added to form a list in the display window along with a start time, an end time, and an elapsed time of the most recent data collection. A time stamp is also included in the results every 30 seconds.</li></ul>
<b>Crossing Histogram Window</b>	<p>Size (%). Defines the size of the sample window used to display the crossing histogram. Defined as a percentage of amplitude.</p> <p>Offset (%). Allows the window to be offset from the center by a percentage of the amplitude (+ or -).</p>

## Settings on the Density Display Tab

<b>Show Y-Axis Scale</b>	Sets whether the Y-axis scale is displayed.
<b>Update display all at once</b>	<p>Controls the elimination of possible display flicker through the following choices:</p> <ul style="list-style-type: none"><li>• Points are drawn individually as each point position is calculated.</li><li>• Points are stored in an internal buffer and displayed all at once at the end of each acquisition/calculation cycle.</li></ul>

---

## Settings on the Density Display Tab

The Density Display tab allows you to adjust the sample density display options.

<b>Sample Density Color Scheme</b>	<p>Controls the color of the sample density through the following choices:</p> <ul style="list-style-type: none"><li>• Colors 1. Blue and violet represent the highest sample densities.</li><li>• Colors 2. Orange and yellow represent the highest sample densities.</li><li>• Intensity. Different intensities of blue and green represent different densities.</li><li>• Single Threshold. A two-color display with adjustable density threshold.</li></ul>
<b>Display</b>	<p>Sets how you will view the acquired data through the following choices:</p> <ul style="list-style-type: none"><li>• Signal. View the Eye Diagram Display.</li><li>• Density Distribution. View a Histogram of the samples.</li></ul>
<b>Resolution</b>	<p>Selects the quantization grid that assigns sample densities.</p> <ul style="list-style-type: none"><li>• Move the slider to the left to select a coarser grid.</li><li>• Move the slider to the right to select a finer grid.</li></ul>
<b>Color Intensity Display</b>	<p>Sets the intensity of displayed pixels through the following controls:</p> <ul style="list-style-type: none"><li>• Contrast adjusts the intensity of the low-density pixels.</li><li>• Weighting controls the assignment of display colors to bins. Moving the slider to the right makes high-density samples relatively more prominent.</li></ul>

### Threshold Display

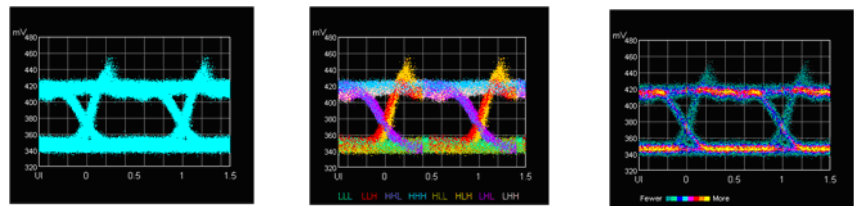
Sets the display of pixels associated with the threshold through the following controls:

- Contrast adjusts the intensity of pixels that indicate densities below the threshold.
- Threshold sets the display threshold value. An up/down button control allows a fine adjustment of this value.

### Sample Density Display Information

The data sample density display is a software tool that enables the eye diagram to display colors based upon the population of sample values. The following set of diagrams shows the variety of displays:

- The left diagram shows the typical single-color eye diagram of samples acquired by the DCA.
- The middle diagram shows multi-color pixels based on the types of transition to which the samples belong.
- The right diagram shows colored pixels based on the number of samples that occur at the location of that pixel.



**Figure 5-2. Variety of Displays for Sample Density Information**

To easily identify a data point that occurs many times in the sample data, you can set the pixel color to be different from a data point that occurs just once. With multiple pixel colors available in the DCA, you can control the display of sample density through the following sequence:

- 1 Define multiple thresholds for the number of times a given data point occurs.
- 2 Indicate each frequency-of-occurrence group in its own color.

The DCA restricts the data samples to a fixed quantization grid with X- and Y-axes, where each sample's "density," and resulting color is based on the number of samples that fall into the same grid square. The grid remains the same even when the display window size changes, retaining the same appearance.

**NOTE**

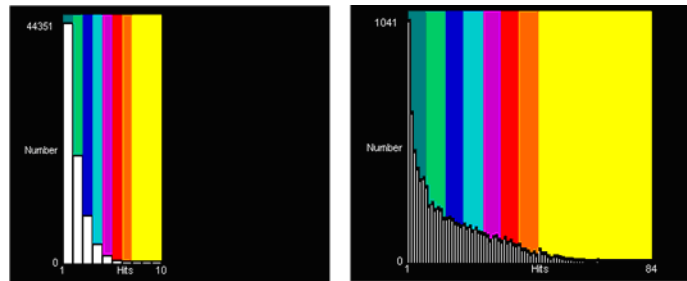
Sample densitydisplay affects only how the data samples are displayed. This display does not affect measurements made on the acquired data or the export of acquired data.

***Choosing the quantization grid***

The quantization grid size is selectable, giving the you some control over the way samples are grouped:

- Choosing a coarse grid results in the highest frequency-of-occurrence values but minimal resolution in the X- and Y-axes.
- Choosing a fine grid gives the best resolution in the X- and Y-axes but reduces the likelihood that samples will fall into the same grid square.

This concept is best shown with the following histograms of sample density.



**Figure 5-3. Histograms of Quantization Grid Choices**

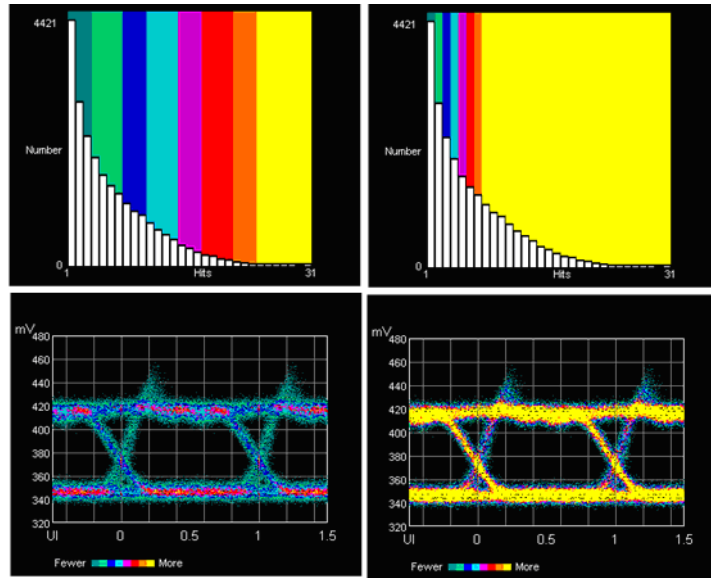
- The histogram on the left shows the finest grid, resulting in the fewest “coincident” samples. In this distribution, half of the samples have unique values, another fourth occur just twice.
- The histogram on the right shows the coarsest grid. This distribution has a substantially greater fraction of samples that occur multiple times.

***Assigning Colors to Sample Densities***

The DCA offers you eight display colors and a choice in assigning colors to sample bins:

- You can assign colors so that each color covers the same number of bins, resulting in every sample being displayed in the higher frequency of occurrence colors. (See the left histogram and eye diagram in the following figure.)
- You can assign each bin a separate color until you run out of colors, resulting in the highest sample densities being displayed very prominently. An intermediate setting gives results that are probably the most appealing. (See the

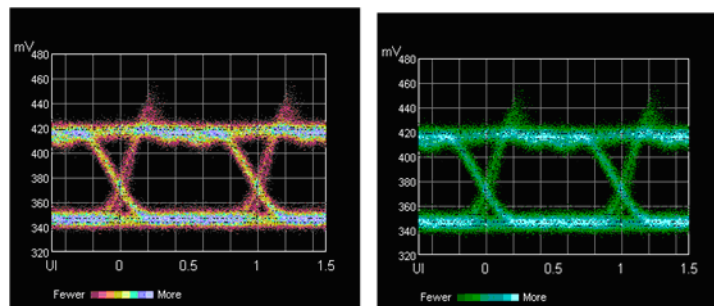
right histogram and eye diagram in the following figure.)



**Figure 5-4. Results of Assigning Colors to Sample Densities**

### ***Choosing a color scheme***

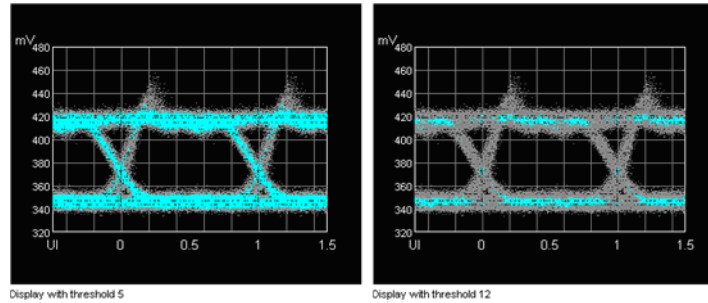
The DCA offers you three color schemes to distinguish between sample densities: two that use sets of colors, and one that uses intensity levels. In the previous figure, the right eye diagram uses a color-based scheme where orange and yellow are assigned to the highest sample densities. In the following figure, the left eye diagram uses a color-based scheme where blue and violet represent the highest sample densities. The right eye diagram uses an intensity-based scheme, displaying only green and blue.



**Figure 5-5. Examples of Color-Based and Intensity-Based Schemes**

***Adjusting the density threshold display***

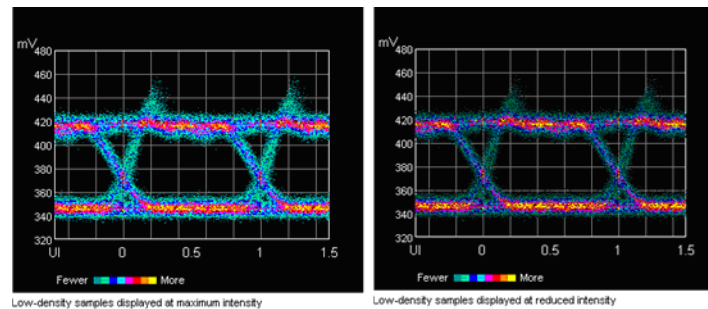
One further display variant uses two colors and a single, adjustable density threshold.



**Figure 5-6. Example of Adjustable Density Threshold**

***Adjusting the contrast of pixel intensity***

A Contrast adjustment controls the pixel intensity, allowing you to display the high-frequency-of-occurrence samples most prominently. This adjustment has the greatest effect on the low-density colors. The following examples show the effect of the contrast adjustment.



**Figure 5-7. Example of Contrast Adjustment**



## Copying and Saving Displayed Test Results

The DCA Control Panel provides a convenient mechanism to capture displayed measurement results, including acquisition data, statistics, visual traces, and test timestamp information for documentation purposes. Three Control Panel buttons are used for this purpose:

- **Data**
- **Copy Graph**
- **Copy Text**

### Data Button

Use this feature to view or copy the actual samples in a text format. Different data sorting, filtering, and display options are available. The two examples shown below are sampled point data and pattern data in hexadecimal format.

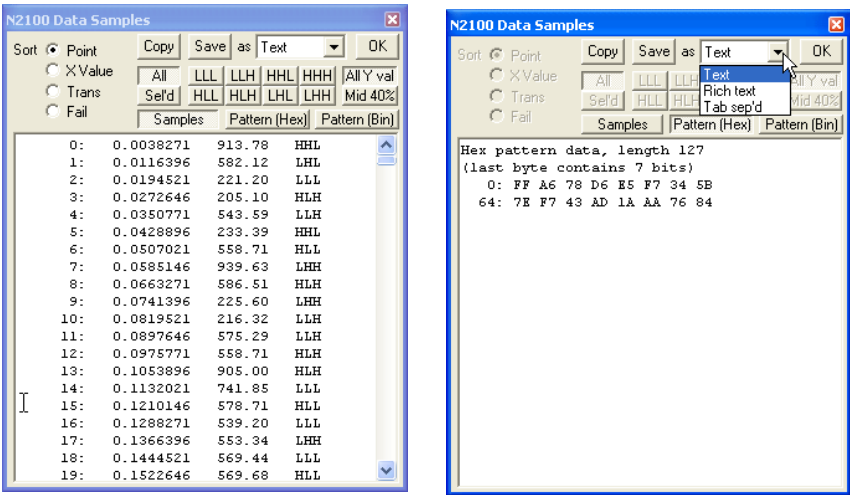


Figure 5-8. Examples of Two Data Samples

## Copying and Saving Displayed Test Results

### *Sorting Data*

- To sort the samples based on the order in which they were collected, select **Point**.
- To sort the samples based on the X value (their time position within the eye), select **X Val**.
- To sort the samples based on what type of transition they belong to, select **Trans**. Within transitions, samples are sorted based on X value.
- To put the mask fail points at the top of the list, select **Fail**. Within the pass and fail group, the samples are sorted based on X value.

### *Copying and Saving Data*

- To copy the data to the clipboard, click **Copy**.
- To save the data to a text (\*.txt) file, click the Save as drop-down arrow and select **Text**, then click **Save**.
- To save the data to a rich text format (\*.rtf) file, click the Save as drop-down arrow and select **Rich text**, then click **Save**.
- To save the data to a tab separated format (\*.txt) file that can easily be brought into a spreadsheet application, click the Save as drop-down arrow and select **Tab sep'd**, then click **Save**.

### *Displaying Transitions*

- To display all transitions, click the transition selection button, **All**.
- To only display transitions that you select using the transition buttons (LLL, LLH, LHH, HHH ....), click **Sel'd**. These transition buttons allow individual transitions to be switched on or off.

### *Displaying Qualified Samples*

- To display all qualifying samples in the data results, select **All Y val**.
- To only display samples that fall in the middle 40% of amplitude readings, select **Mid 40%**.
- To display the individual data samples, select **Samples**.
- To display the reconstructed pattern sequence in hexadecimal format (pattern acquisition only), select **Pattern Hex**.

- To display the reconstructed pattern sequence in binary format (pattern acquisition only), select **Pattern Bin**.

---

**NOTE** When data are acquired with a mask test and displayed in binary, pattern sequence bits containing samples that fail the mask test are underlined in the display.

---

**Copy Graph Button** The **Copy Graph** button copies the graph and all the text data from the most recent acquisition run as displayed on the screen. The image is available as an object on the PC clipboard so you may paste it into a word processing program for future use.

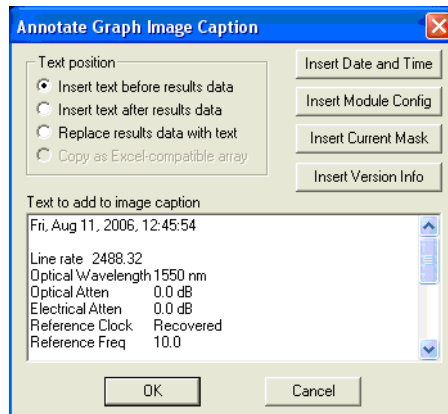
**Copy Text Button** The **Copy Text** button copies all the information from the results display window in text format. This information can be pasted into any program such as MS Word or Notepad as in this example:

Meas.	Average	Minimum	Maximum	Std. deviation
Amplitude	420.637	407.250	429.750	12.222
Peak to peak	514.087	491.250	528.563	15.871
RMS	197.393	190.520	201.963	6.251
Average	384.790	384.553	384.977	0.178
OMA	419.203	404.144	429.215	13.435

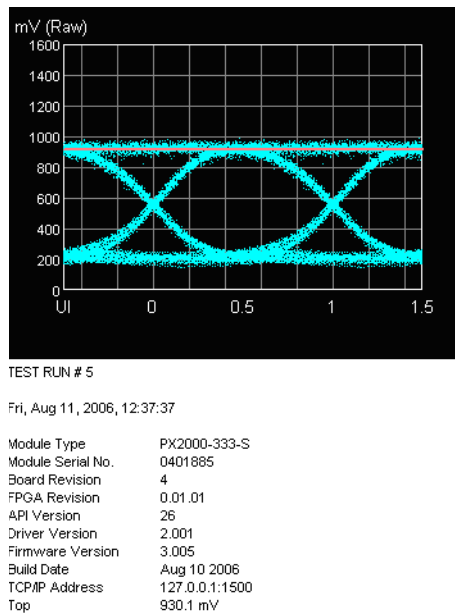
**Custom Annotations** You can annotate the graph image caption. Hold down the keyboard's CTRL key while clicking **Copy Graph** or **Copy Text**. This displays the dialog box shown in [Figure 5-9 on page 5-14](#). The buttons and text entry field provide a way for you to append additional text to the visual or textual data captured, as shown in [Figure 5-10 on page 5-14](#).

## Displaying and Saving Results

### Copying and Saving Displayed Test Results



**Figure 5-9. Annotate Graph and Image Caption Dialog Box**



**Figure 5-10. Annotated Graph Example**

The Copy as Excel-compatible array is available only when the following conditions exist:

- **Copy Text** option is selected when the Multiple Measurements tab is selected.
- **Repetitive Results Text** is set to '**Add**' or '**Add+Time**' option in Prog Settings panel.

This option reformats the list of results acquired during a repetitive measurements run into a tab-separated array of data. This data can then be directly copied into an Excel workbook or other spreadsheet.

The data are formatted as one column per measured parameter and one row per acquisition. The first row of the table has parameter labels. In 'Add+Time' mode, an additional column is populated with the timestamps of the measurement block at 30 second intervals.

Amplitude	Base	Top	Rise Time	Fall Time	Time
46.1	8.9	55.0	134.6	141.8	18:03:20
46.1	8.9	55.0	136.7	151.1	
46.1	8.9	55.0	128.6	142.0	
46.1	8.9	55.0	110.1	150.2	
50.7	6.6	57.3	159.7	163.5	
46.1	8.9	55.0	137.1	148.7	18:03:50
46.1	8.9	55.0	148.9	151.0	



---

## **Pattern Acquisition**

## Pattern Acquisition

The DCA normally samples its input signal over an interval of one bit time (unit interval, or UI) and reconstructs an eye diagram that is an aggregate of all the signal's states and transitions. In pattern sequence acquisition, the interval over which the DCA collects samples is equal to as many bit times as the full pattern sequence length. The reconstructed result is a representation of the entire pattern sequence. Although the displayed result is an aggregate of all the sequence periods over which the samples were taken, it is coherent with the sequence. Therefore, any individual transition in the reconstructed sequence consists only of samples from the corresponding transition of the input signal.

The reconstruction of the full pattern sequence allows you to do the following:

- Measure pattern-dependent (deterministic) jitter.
- Determine the effect on the shape and timing of a transition of the data values that precede it.
- Use digital filtering techniques to realize desired filter characteristics.

Filter functions can be applied that are specific for the signal line rate and desired filter bandwidth; these can also take into account the DCA's inherent frequency response so that the overall response is as close to optimum as possible. When the pattern acquisition feature is enabled, any specified filtering is applied to the reconstructed sequence and performed measurements.

### ***Guidelines for using pattern acquisition***

- The input pattern must be absolutely repetitive to allow a reconstruction from samples taken over many periods. The application of this technique is typically limited to unframed signals containing pure data.
- The sequence must be no longer than some upper limit. The sequence can range from a few bits in length up to PRBS  $2^{11} - 1$ .



- The signal reconstruction is an averaging process so measurements of random (non-periodic) jitter made on the reconstructed signal do not accurately reflect the jitter present on the input signal.
- The reconstruction process does not preserve individual data samples. It is possible (due to averaging) that an individual sample that may fail a mask test will not appear in the reconstructed output.
- Implementation of the filter with computational techniques requires significant processing. When this capability is enabled, instrument performance will be slower than when it is disabled.

**Operation**

Configuring the pattern sequence acquisition involves specifying the following characteristics:

- Pattern sequence length
- Filter to be applied, if any

**Control Panel Application**

The Pattern Acquisition control has selections for Off, Enabled (No filter), and a list of software filters (if any) that are available for the module.



The Pattern control specifies the length of the repeating pattern sequence (PRBS  $2^7-1$ , PRBS  $2^9-1$ , PRBS  $2^{11}-1$ , K28.5, K28.7), and Set Length for patterns of arbitrary length. When Set Length is selected, the field to the right is enabled so you can enter the pattern length. The length must be between 4 and 2047.

**NOTE**

The recovered clock with a sub-rate pattern of K28.7 is usable with the API, but not with the Control Panel interface.

**Acquisition**

With Pattern Acquisition enabled, any data acquisition operation acquires data over the sequence length, reconstructs the sequence, and applies the selected filter. The reconstructed eye diagram is displayed and any measurements invoked are made on the reconstructed eye.

---

**NOTE**

---

If the Pattern Length setting does not match the actual sequence length, or if the input signal is not absolutely repetitive, the sequence reconstruction will be incoherent and the eye display will appear distorted.

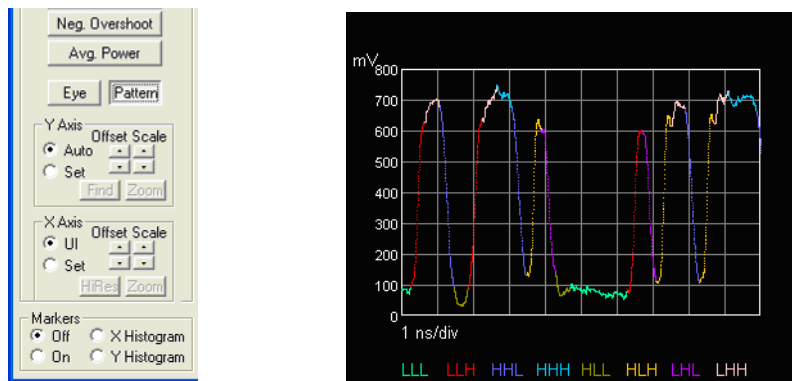
**Displaying Acquired Data in Text Format**

The acquired data pattern can be displayed and processed through the Data Samples dialog box described in [“Displaying and Saving Results” on page 5-1](#). Multiple display modes, sorting fields, and file manipulation functions are available.

**Graphical Sequence Display**

The Oscilloscope Measurements tab includes a selector for Eye or Pattern display. (The Pattern selection is enabled only when pattern sequence acquisition is active.)

If you select Pattern, the display changes to show the entire reconstructed sequence. You can use the Time Axis controls to scroll through the sequence and display it with suitable horizontal resolution.



**Figure 6-1. Pattern Display in Oscilloscope Mode**

**Saving Pattern Sequence Data for Use with the Simulator**

To ensure the simulator correctly interprets the data in a saved file of data collected with the pattern sequence acquisition, follow these steps:

- 1 Before saving the data, set the **Measurements** selector to **Oscilloscope Measurements**.
- 2 Set the **Display** selector to **Pattern**.

**Measurements  
(General)**

With pattern sequence acquisition enabled, measurements are made on the reconstructed sequence but otherwise function normally.

---

## **Software Filters with Pattern Acquisition**

The Pattern Acquisition feature offers the ability to apply software filters. A software filter is a digital filter that matches a specific desired frequency response and rise time when convolved digitally with the measured response of the DCA. Each filter is specified by a file containing an array of numerical data. The digital filter is calculated from the deconvolution of the measured known impulse response of the DCA and the impulse response of the desired filter.

---

**NOTE**

Software filters are only available for optical input. The filters are module specific and cannot be shared between separate modules.

### ***Guidelines for using digital filters***

- Agilent digital filters are designed for use when the DCA is configured with the unfiltered optical path and in the pattern acquisition mode. The Input Path and Filter Selection must be set to the unfiltered path. If the DCA is not set to the unfiltered path, the filters are calculated for use with the highest-bandwidth optical filtered path so that path should be selected.
- The filter file can only be applied to a signal whose line rate is the same as that of the filter. Applying a filter file to a different line rate will result in erroneous data
- Incorrectly applying digital filters can lead to unpredictable and erroneous results. When using Agilent supplied digital filters, ensure the configuration for capturing the data is set correctly.

**Software Filters with Pattern Acquisition****NOTE**

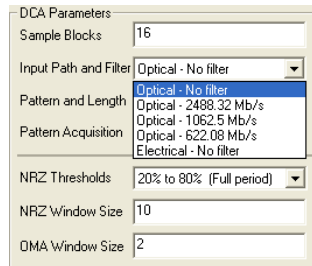
Because of the inherent averaging of the data when using the pattern acquisition mode, measurements of random jitter are not valid. It is recommended that you use hardware filters when performing jitter measurements.

**Filter Files**

Each DCA has filter files constructed to meet the target rates and bandwidths as the DCA's physical input filters. These files are stored on the module, and are automatically copied to the host for use by the filtering software. Available filter selections appear in the Pattern Acquisition control on the Configuration Settings page, following the <Enabled (No Filter)> choice.

**Filter List**

Filter selections are listed by bandwidth. If an optional 'optstring' field is present, that string appears in parentheses after the bandwidth specifier.

**Filter File Location**

Filter files are stored in the DCA Filter Files subdirectory of the N2100B installation directory. If the N2100B software is installed to the default location, this is C:\Program Files\Agilent\N2100B\DCA Filter Files.

**Filter File Format**

File names have the format: DCAxxxxxxBTstring@optstring.txt, where:

- xxxxxx is the serial number of the DCA to which the filter file applies.
- string is a numeric or mostly-numeric character string related to the filter bandwidth. For example:
  - 4250 suggests 4250 Mb/s.
  - 2488p32 suggests 2488.32 Mb/s, where p represents the decimal point.
- optstring is a character string used to distinguish two files that might otherwise have the same name. (It might be used to distinguish between filters of different length or order.)

Introduction	7-2
How the Mask and Mask Margin are defined	7-3
Agilent Mask Margin Rules	7-6
Tektronix Mask Margin Rules	7-8

---

## **Mask & Mask Margin Definitions**

---

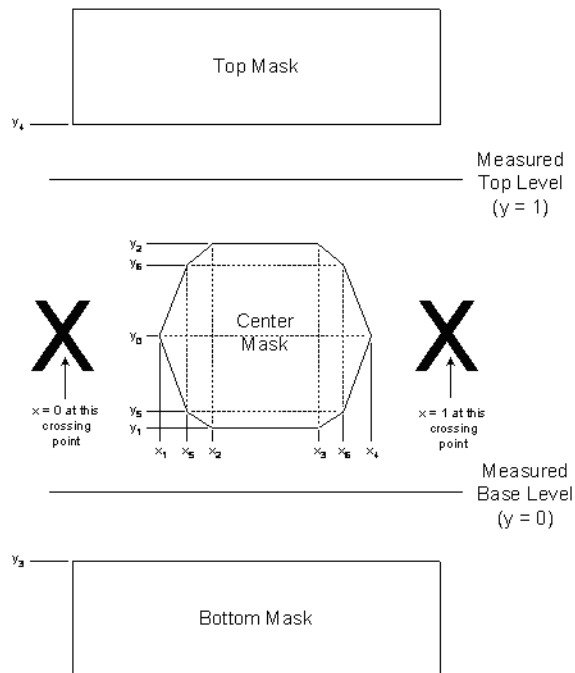
## Introduction

This chapter describes the structure of masks used for measurements that determine if acquired sample points pass or fail a defined test.

- [“How the Mask and Mask Margin are defined” on page 7-3.](#)
- [“Agilent Mask Margin Rules” on page 7-6.](#)
- [“Tektronix Mask Margin Rules” on page 7-8.](#)

## How the Mask and Mask Margin are defined

A mask defines an area in the reconstructed image of the eye where no sample points should exist. It always consists of an area in the center of the eye and can optionally have areas above and below the eye.



**Figure 7-1. Mask Definition**

**How the Mask and Mask Margin are defined**

All y-axis values are defined relative to the measured base level ( $y = 0$ ) with the measured top level being  $y = 1$ . The Y level can be determined based on either of the following:

- Top and bottom of the entire eye
- High and low based on a certain percentage around the middle of the eye

All x-axis values are defined relative to the left side crossing point ( $x = 0$ ) with the right side crossing point being  $x = 1$ .

The center mask can have 4, 6, or 10 sides depending on the checkbox settings in the setup form below. This form also allows you to setup the position of all the vertices of the mask.

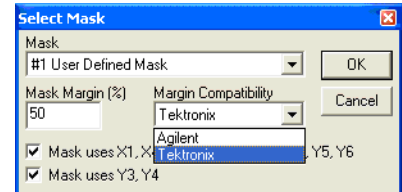
**Figure 7-2. Select Mask Dialog Box**

- Selecting none of the checkboxes sets the mask to center only and the shape to rectangular based on  $x2$ ,  $x3$ ,  $y1$ ,  $y2$ .
- Checking **Mask uses  $x1$ ,  $x4$**  adds two additional points, allowing you to define the center mask as a hexagon.
- Checking **Mask used  $x5$ ,  $x6$ ,  $y5$ ,  $y6$**  adds four additional points allowing you to define the center mask to a decagon.
- Checking **Mask uses  $y3$ ,  $y4$**  adds the top and bottom mask definitions.



A validity check is performed on the coordinates before values are accepted. One or more pop-ups appear if there is an inconsistency in specifying the mask parameters. You must acknowledge this test before proceeding with such values.

The Margin Compatibility option allows you to choose whether a mask that applies the margin is based on the Agilent rule or the Tektronix rule.



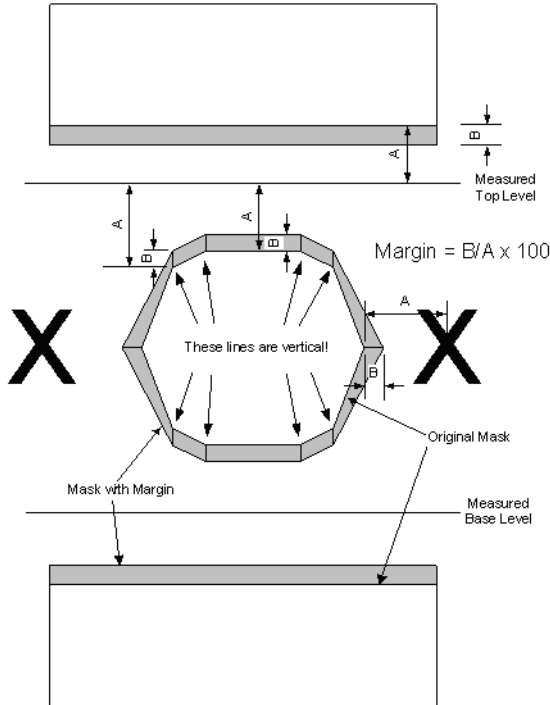
**Figure 7-3. Margin Compatibility Option**

Refer to “Agilent Mask Margin Rules” on page 7-6 and “Tektronix Mask Margin Rules” on page 7-8.

## Agilent Mask Margin Rules

### Positive Margin

When a positive margin is applied, vertices of the inner mask are moved by the requested proportion towards the left and right crossing points and towards the top and base levels. However, the parameters x2 and x4 remain constant. The top mask region (y4) moves down toward the top level, and the bottom mask region (y3) moves up towards the base level in proportion to the margin.

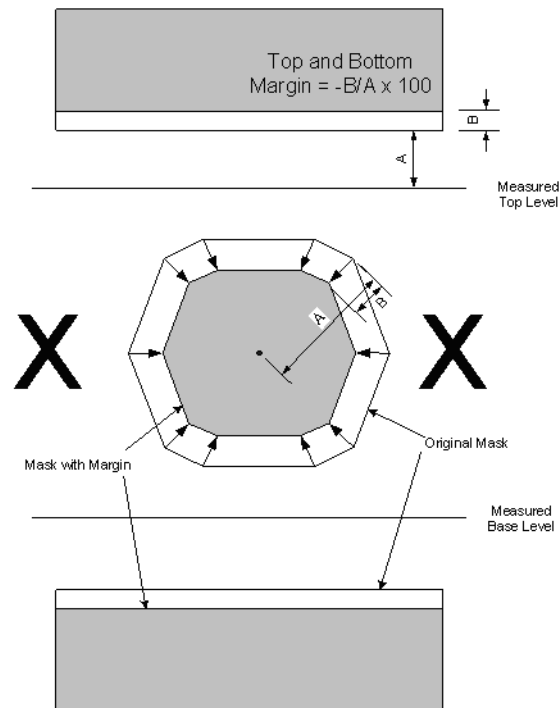


**Figure 7-4. Agilent Mask - Positive Margin**

### Negative Margin

When a negative mask margin is applied, all points in the center mask are moved in towards the center by the requested proportion.

The Top and Bottom masks are moved away from the top and base levels by the requested proportion.



**Figure 7-5. Agilent Mask - Negative Margin**

## Tektronix Mask Margin Rules

### Available Margin in the X Dimension

In the Tektronix mask margin conventions, the available margin in the X (time) dimension ( $\Delta X_{100}$ ) is the smaller of the following values:

- Distance from the left eye crossing to the leftmost mask vertex
- Distance from the rightmost mask vertex and the right eye crossing

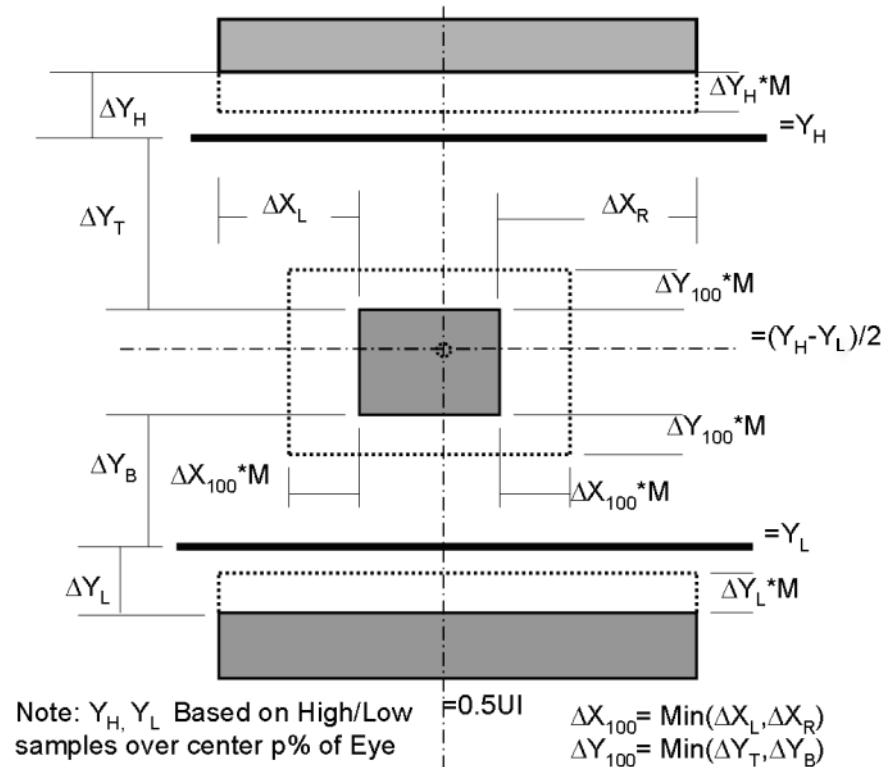


Figure 7-6. Tektronix Rectangle Mask Margin Illustration

The leftmost and rightmost vertices of the center mask are moved outward by the specified percentage (M) of this available margin or inwards if the margin is negative. In [Figure 7-6](#), this is represented as  $\Delta X_{100} = \text{Min}(\Delta X_L, \Delta X_R)$ .

### **Available Margin in the Y Dimension**

For the center mask, the available margin in the Y (amplitude) dimension,  $\Delta Y_{100}$  is the smaller of the following values:

- Distance from the signal base level ( $Y_B$ ) to the lower mask edge
- Distance from the upper mask edge to the signal top level ( $Y_T$ )

$$\Delta Y_{100} = \text{Min}(\Delta Y_T, \Delta Y_B)$$

You can define the  $Y_T$  and  $Y_B$  values relative to the full-UI values or a specified window (p%) about the center of the eye.

The upper and lower edges of the center mask are moved outwards by the specified percentage of this available margin, or inwards if the margin is negative.

The dotted lines represent the mask margin while the full straight mask lines represent the original mask.

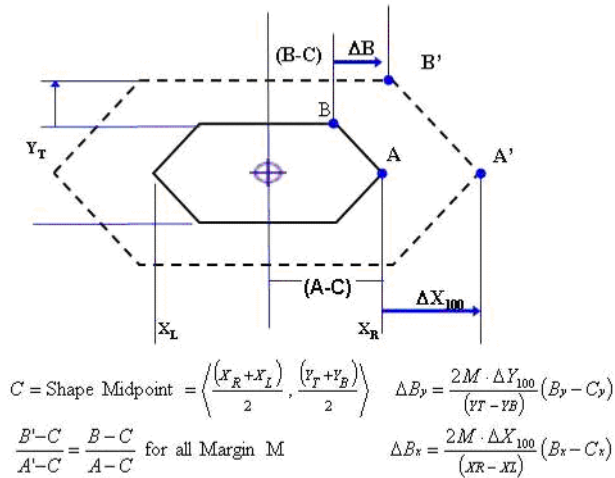
### **Margin for Upper and Lower Mask Regions**

For the upper (lower) masks, the available margin in the Y dimension is the distance from the mask level to the top (base) level. These masks are moved by the specified percentage of this available margin, toward the center mask if the margin is positive or away if the margin is negative.

### **Tektronix Margin Definition**

The Tektronix margin definition requires that the upper and lower mask regions be present and that the center region be defined to be between the base and top levels, the upper and lower regions outside of these levels.

For the 6- or 10-point mask case, all intermediate points are moved in proportion to their distance from the mask center to the corresponding outermost vertices. This rule applies independently in the horizontal and, for the 10 point shape, in the vertical axes. This concept is illustrated in [Figure 7-7](#).



### Figure 7-7. Tektronix Margin Rules for 6- and 10-Point Masks

Introduction	8-2
Configuring the Simulator	8-5
Controlling with User-written Applications	8-7

---

## Using the Simulator

---

## Introduction

When the N2100B software is installed in the DCA, a simulator is also installed. The simulator reads previously captured data from a file on the system and presents it to the control panel as if it were from an instrument. The simulator is also a very powerful development tool as it enables you to create your own code to control the instrument without having any hardware in place.

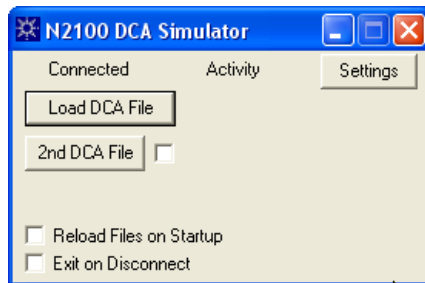
This chapter describes how to do the following tasks with the simulator:

- “To run the simulator” on page 8-2.
- “Configuring the Simulator” on page 8-5.
- “Controlling with User-written Applications” on page 8-7.

---

## To run the simulator

- 1 Start the simulator from the Windows Start menu by clicking **Start > All Programs > Agilent > N2100B > N2100B Simulator**.

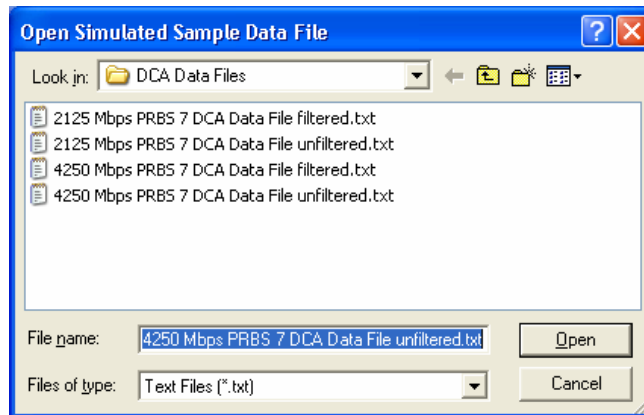


The simulator and the control application communicate using TCP/IP, even when both run on the same computer. If you are running a firewall program, the simulator will probably be prevented from communicating with the control panel application. Either disable the firewall or configure it to allow these programs to communicate as they wish.

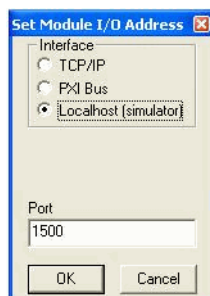


- 2 Click **Load DCA File** and select a data file.

You could also use a file of data written by the DCA Control Panel application's Data/Save to File function. When no files of actual data are available, you can use one of the sample data files in the DCA data files directory, (Program Files/Agilent/N2100B/dca data files).



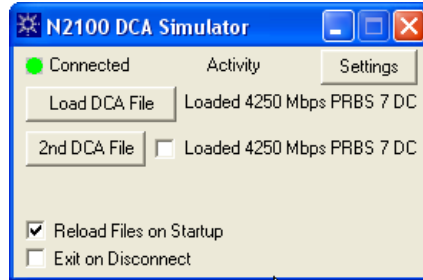
- 3 Start the DCA Control Panel application.
- 4 Click **I/O Config** and select the simulator option. The Port is 1500 by default. It is possible to have up to four simulators running on ports 1500-1503.



Using the Simulator

**To run the simulator**

- 5 Click **Connect**. The connected indicator on the simulator GUI will activate, as shown below.



- 6 Click **Start Acquisition**.

The Activity Indicator flashes on the Simulator panel, as the control application reads data from the simulator and displays the data as an eye pattern. Use the Module Config controls to change the simulator settings, and use the various measurement and display controls to interact with the simulator.

---

**NOTE**

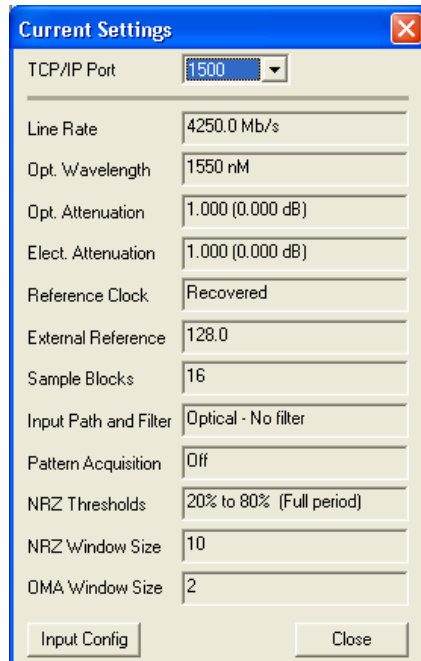
---

The **2nd DCA File** button allows you to load a second data file. When a second file is loaded and the checkbox is checked, successive acquisitions alternate the data from the two files.

---

## Configuring the Simulator

By clicking **Settings** on the Simulator, the following panel appears. This panel allows you to view the settings that a user program has applied to the simulated DCA, or verify consistency with the Module Configuration information on the DCA Control Panel.

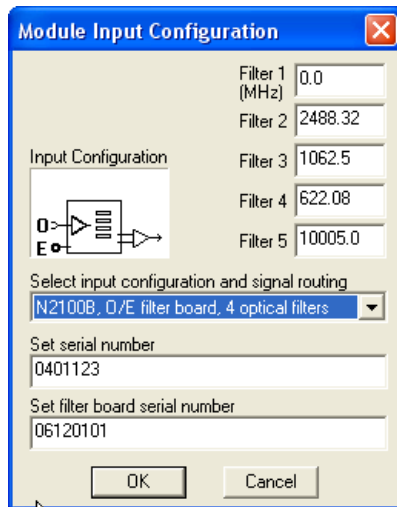


The image shows a 'Current Settings' dialog box with a blue title bar and a close button (X) in the top right corner. The dialog contains a list of settings, each with a label and a text field or dropdown menu. The settings are: TCP/IP Port (1500), Line Rate (4250.0 Mb/s), Opt. Wavelength (1550 nM), Opt. Attenuation (1.000 (0.000 dB)), Elect. Attenuation (1.000 (0.000 dB)), Reference Clock (Recovered), External Reference (128.0), Sample Blocks (16), Input Path and Filter (Optical - No filter), Pattern Acquisition (Off), NRZ Thresholds (20% to 80% (Full period)), NRZ Window Size (10), and DMA Window Size (2). At the bottom of the dialog, there are two buttons: 'Input Config' and 'Close'.

Setting	Value
TCP/IP Port	1500
Line Rate	4250.0 Mb/s
Opt. Wavelength	1550 nM
Opt. Attenuation	1.000 (0.000 dB)
Elect. Attenuation	1.000 (0.000 dB)
Reference Clock	Recovered
External Reference	128.0
Sample Blocks	16
Input Path and Filter	Optical - No filter
Pattern Acquisition	Off
NRZ Thresholds	20% to 80% (Full period)
NRZ Window Size	10
DMA Window Size	2

Clicking **Input Config** on the panel opens the Module Input Configuration dialog box, allowing you to match the DCA being simulated. Setting parameters such as serial number, input configuration, signal routing, and filter frequencies allows you to load filter and data files specific to the given DCA.

## Configuring the Simulator



---

### NOTE

Measurement results from the simulator may not exactly match those from the DCA module. The simulator is primarily intended as a qualitative simulation of the DCA.

---

---

## Controlling with User-written Applications

Since the simulator presents the same TCP/IP (network) interface as the DCA module, it is accessible by any application that can control the DCA. To control the simulator through another application:

- 1 Start the simulator and load the desired simulator data file before the application attempts to connect to it.
- 2 Within the application, select TCP/IP as the communication link type and localhost as the module IP address.

The simulator must run on the same computer as the application that attempts to connect to it.

### Running More Than One Simulator

To run more than one instance of the simulator simultaneously, use the TCP/IP Port selector on the simulator's Settings panel to select a different port for each simulator instance. Connect each client application to the intended simulator by specifying that port when connecting. Four ports are currently available.

### Simulator Data File Format

The DCA simulator reads data files written by the control panel application's Data panel. To create suitable data files by another means, the files must follow this format:

- Data are in fixed-format ASCII, with one data point (sample) per line.
- Columns are formatted as listed below.
- Intervening positions are filled with space characters.

0:	0.5000000	9.30	HLL
1:	0.0527776	9.56	HLL
2:	0.6055553	9.50	LLL

**Table 8-1. Simulator Data File Format**

Column	Field
1	The sequence number. The simulator does not use this information.
8	A colon that follows the sequence number. It must be present and properly positioned.
12	The sample X value, a floating-point number between 0 and 1.
22	The sample Y value, a floating-point number.
32	The transition type, one of the following strings.
LLL	Sustained low state following a sustained low state.
LLH	Low-to-high transition following a sustained low state.
LHL	High-to-low transition following a low-to-high transition.
LHH	Sustained high state following a low-to-high transition.
HLL	Sustained low state following a high-to-low transition.
HLH	Low-to-high transition following a high-to-low transition.
HHL	High-to-low transition following a sustained high state.
HHH	Sustained high state following a sustained high state.

Introduction	9-2
DLL API Reference	9-6
Module Information	9-6
Module Connection	9-10
Configuration	9-11
Data Acquisition	9-17
Measurement	9-19
Calibration	9-26
Status	9-74
Logging	9-27
Returned Status, Warning, and Error Values	9-30
Data Structures	9-33
Enum Values	9-39
Module Information	9-42
Module Connection	9-45
Mask Configuration	9-47
DCA Sample Data Acquisition	9-50
Measurement	9-54
Pattern Sequence Acquisition	9-65
Calibration	9-69
Control Display	9-70
DCA Configuration	9-75
ActiveX API Properties	9-75
DCA Configuration	9-75
Pattern Sequence Acquisition	9-78

---

## Programming

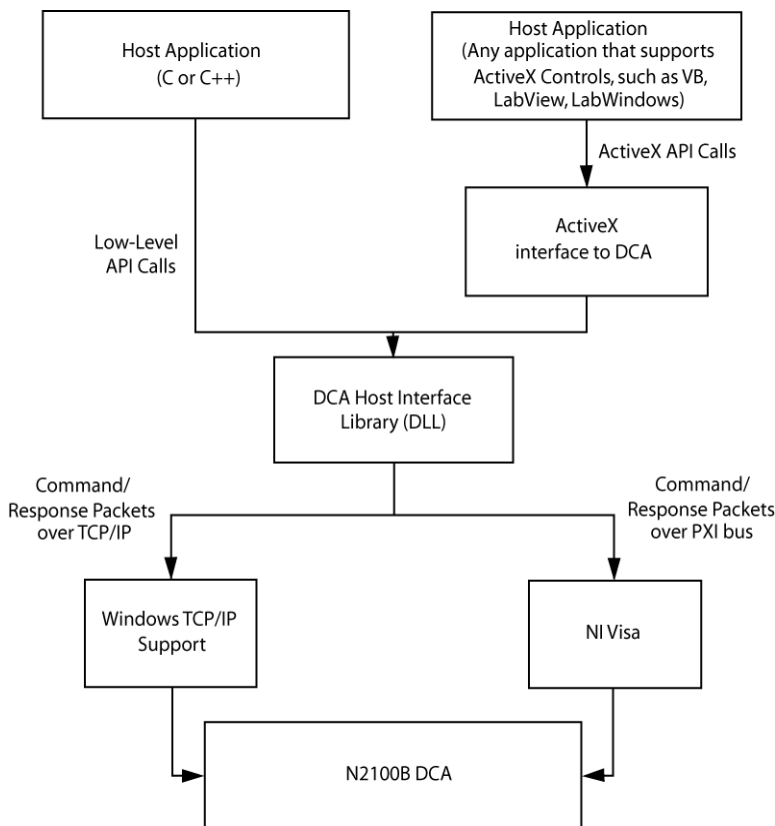
## Introduction

In addition to using the Control Panel, the DCA can be controlled automatically through the following two methods:

- Application Programming Interface (API)
- ActiveX Control

The API for N2100B control is provided in a Windows based DLL and an ActiveX control. The API allows the host application to create and delete communication channels to instruments, configure the N2100B, perform data acquisitions, measurements, and eye mask testing. [Refer to “Module Configuration Settings” on page 3-4](#) and Chapter 4, “Measurements” for details of the configurable parameters.





**Figure 9-1. Host Library Overview**

The N2100B Host Library consists of the following software components. A program that uses the DCA API calls must include the files `fgapidefs.h` and `fghostapi.h`. These are found in the installed software, in the include directory.

- fghostapi.dll** This file is the Windows based DLL that implements the Host Library API.
- fghostapi.lib** This file contains all the DLL exported functions that are linked against the Host Application.
- fghostapi.h** This header file contains the Host library function prototypes and definitions.
- fgtypes.h** This header contains data type definitions common between the embedded firmware and the host library.

**fgstatus.h** This header file contains all the status codes used by the N2100B.

**fgapidefs.h** This header file contains all the data structures that make up the API between the Host Application, Host Library, and embedded firmware. The interface between the OEM application and the N2100B is packet based.  
A folder of API examples is included with the downloaded software and drivers.

## **Establishing Communications**

The host system communicates with the N2100B module over the PXI Bus or through a TCP/IP interface. The host controls the N2100B module in the following ways:

- Sets operating modes and parameter values
- Initiates data acquisition operations
- Reads module status and measurement result values

Before any communications can take place, the host system must create a connection to the N2100B. [Refer to “Module Connection” on page 9-10](#) for more details. Connections must be closed at the end of the session.

## **Active X**

ActiveX allows developers to create code and applications from a multitude of languages. This also allows you to build a defined interface code, making it accessible to other applications.

The DCA software interface is implemented as an ActiveX control which is a software component with a standard interface that allows it to be included (hosted) by a variety of programming and test environments such as LabView, LabWindows/CVI, Visual C++, Visual Basic, and Visual C#. Its properties and methods make the DCA functionality available to the controlling program, and its optional display presents the same eye diagrams and measurement results as the GUI control application.

Rules for using the DCA ActiveX control differ depending on the environment that hosts the control. Visual Basic provides direct access to the control. LabWindows/CVI automatically generates a wrapper with callable functions. Visual C++ generates a wrapper class when the ActiveX control is added to a project. Refer to the documentation for your language/environment for the details of using ActiveX controls.

Use the following steps to add the DCA to a test application and acquire data:

- 1** Add an instance of the activeX control to your project, using the procedures appropriate to your programming environment.

- 2** Call one of the module connection functions `OpenTcp`, `OpenTcpByVal`, `OpenPxi`, or `OpenPxiByVal` to establish communications with the DCA module.
- 3** Set the `DisplayMode` property to select whether the eye display window is visible, and if so, what information it displays.
- 4** Call the various configuration functions to set module parameters such as line rate, reference clock select, and number of sample data points.
- 5** Call one of the data acquisition functions `AcquireDCA` or `AcquireDCAWithMask` to acquire data.
- 6** Call the various measurement functions to make measurements on the acquired data and retrieve the results. Or, call one of the data acquisition functions `GetDCASampleData` or `GetDCASampleDataVariant` to retrieve the data sample values.
- 7** Update the module parameter settings and/or make additional measurements as your application requires.
- 8** At the end, call the `Close` function to release the communication link to the module.

## DLL API Reference

A program that uses the DCA API calls must include the files `fgapidefs.h` and `fghostapi.h`. You can find these files in the installed software's `/include` folder.

- “Module Information” on page 9-6.
- “Module Connection” on page 9-10.
- “Configuration” on page 9-11.
- “Data Acquisition” on page 9-17.
- “Measurement” on page 9-19.
- “Calibration” on page 9-26.
- “Status” on page 9-74.
- “Logging” on page 9-27.
- “Returned Status, Warning, and Error Values” on page 9-30.
- “Data Structures” on page 9-33.
- “Enum Values” on page 9-39.

---

## Module Information

- “FGGetBoardInfoAPI” on page 9-6.
- “FGGetBoardNameAPI” on page 9-7.
- “FGGetSerialNumberAPI” on page 9-7.
- “FGGetBoardRevAPI” on page 9-8.
- “FGGetInputConfigAPI” on page 9-8.
- “FGGetFilterConfigAPI” on page 9-9.

---

### FGGetBoardInfoAPI

Returns the module type string, serial number string, and revision level.

**Call** `FGGetBoardInfoAPI(FG_HANDLE hModule, char *pszSerialNumber, char *pszBoardName, ULONG *pnBoardRev)`

### Parameters

- The module connection handle returned by `fg_connect`.
- A pointer to a character string buffer that receives the module serial number.
- A pointer to a character string buffer that receives the module type string.
- A pointer to an unsigned long integer that receives the module version number.

---

**NOTE**

The type string buffer must be at least `FG_MAXBOARDNAME_SIZE` bytes long and the serial number buffer must be at least `FG_MAXSERIALNUM_SIZE`. These values are defined in `fgapidefs.h`.

---

**Returns** A status value of type `FG_STATUS` reflecting the result of the operation.

---

### **FGGetBoardNameAPI**

Returns the module type string.

**Call** `FGGetBoardNameAPI(FG_HANDLE hModule, char *pszBoardName)`

### Parameters

- The module connection handle returned by `fg_connect`.
- A pointer to a character string buffer that receives the module type string.

---

**NOTE**

This buffer must be at least `FG_MAXBOARDNAME_SIZE` bytes long, as defined in `fgapidefs.h`.

---

**Returns** A status value of type `FG_STATUS` reflecting the result of the operation.

---

### **FGGetSerialNumberAPI**

Returns the module serial number string.

**Call** `FGGetBoardSerialNumberAPI(FG_HANDLE hModule, char *pszSerialNumber)`

### Parameters

- The module connection handle returned by `fg_connect`.
- A pointer to a character string buffer that receives the module serial number string.

---

**NOTE**

---

This buffer must be at least FG\_MAXSERIALNUM\_SIZE bytes long, as defined in fgapidefs.h.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

**FGGetBoardRevAPI**

Returns the module revision level as an integer.

**Call** FGGetBoardRevAPI(FG\_HANDLE hModule, unsigned long \*pnBoardRev)

**Parameters**

- The module connection handle returned by fg\_connect.
- A pointer to an unsigned long variable that receives the returned value.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

**FGGetInputConfigAPI**

Returns an integer value that specifies the available inputs and the signal routing configuration.

**Call** FGGetBoardInputConfigAPI(FG\_HANDLE hModule, unsigned long \*pnInputConfig)

**Parameters**

- The module connection handle returned by fg\_connect.
- A pointer to an unsigned long variable that receives the returned value.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

**Table 9-1. Returned Values (1 of 2)**

Value <sup>a</sup>	Available Input Selections	Module Input Configuration
1	5	Optical input to internal filter bank, electrical input direct
2	2	Optical input direct to high-speed path, electrical input direct
3	2	Optical input to high-speed path with filter, electrical input direct
4	5	Optical input to external high-performance filter bank, electrical input direct

**Table 9-1. Returned Values (2 of 2)**

Value <sup>a</sup>	Available Input Selections	Module Input Configuration
5	1	Electrical only, direct
6	1	Electrical only, to high-speed path with filter
7	4	Electrical only, to internal filter bank
8	4	Electrical only, to external high-performance filter bank
9	4	Optical only, to internal filter bank
10	1	Optical only, to high-speed path
11	1	Optical only, to high-speed path with filter
12	4	Optical only, to external high-performance filter bank
13	3	Electrical with differential input
14	5	Optical input to external high-performance filter bank through high-speed path, electrical input direct
15	4	Optical only, to external high-performance filter bank through high-speed path
16	5	N2100B with four optical filter selections, includes electrical input..
17	4	N2100B with three optical filter selections.
18	3	N2100B with two optical filter selections.
19	2	N2100B with a single optical filter selection.
20	2	N2100B with no filter assembly, electrical and optical input selections.

a. Values 1 through 15 apply to the N2100A. Values 16 through 20 apply the the N2100B.

---

## FGGetFilterConfigAPI

Returns the frequencies of the installed input filters.

**Call** FGGetBoardFilterConfigAPI(FG\_HANDLE hModule, FG\_FILTERCONFIG \*pFilterConfig)

### Parameters

- The module connection handle returned by fg\_connect.
- A pointer to a structure of type FG\_FILTERCONFIG that receives the returned

values.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

## Module Connection

---

### **fg\_connect**

Sets up a connection to the module.

**Call** FG\_HANDLE fg\_connect(FG\_COMM\_PARAMS \* pParams)

**Parameter** A pointer to a structure of type FG\_COMM\_PARAMS that specifies the communications link (TCP/IP or PXI bus) to be used and the address on that link of the target DCA module.

**Returns** A handle to the module connection. This handle must be passed as a parameter to subsequent calls to the module API functions. If the operation fails, the call returns NULL (0).

---

### **fg\_disconnect**

Closes a currently open module connection.

**Call** FG\_STATUS fg\_disconnect(FG\_HANDLE hModule)

**Parameters** The handle to the connection to be closed, as returned by the call to fg\_connect.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.



---

## Configuration

- “FGCfgGetAPI” on page 9-11.
- “FGCfgSetAllAPI” on page 9-11.
- “FGCfgSetCustomIntfRateAPI” on page 9-12.
- “FGCfgSetWaveLength” on page 9-12.
- “FGCfgSetOptAttenRatioAPI” on page 9-12.
- “FGCfgSetElectAttenRatioAPI” on page 9-13.
- “FGCfgSetRefClkSlrAPI” on page 9-14.
- “FGCfgSetRefClkAPI” on page 9-14.
- “FGCfgSetRecoveredClockRatioAPI” on page 9-14.
- “FGCfgSetDCAAcqSizeAPI” on page 9-15.
- “FGCfgSetInputSelectAPI” on page 9-15.
- “FGCfgSetHWFILTERAPI” on page 9-16.
- “FGCfgSetNRZThresholdAPI” on page 9-16.
- “FGCfgSetNRZWinSizeAPI” on page 9-16.
- “FGCfgSetNRZOMAWinSizeAPI” on page 9-17.

---

### FGCfgGetAPI

Reads the module’s current configuration parameter values.

**Call** FG\_STATUS FGCfgGetAPI(FG\_HANDLE hModule, FG\_CONFIGPKT \* pData)

#### Parameters

- The module connection handle returned by fg\_connect.
- A pointer to a structure of type FG\_CONFIGPKT that receives the configuration settings.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### FGCfgSetAllAPI

Sets all DCA configuration parameters.

**Call** FG\_STATUS FGCfgGetAPI(FG\_HANDLE hModule, FG\_CONFIGPKT \* pData)

#### Parameters

- The module connection handle returned by fg\_connect.
- A pointer to a structure of type FG\_CONFIGPKT that specifies the new config-

uration settings.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### **FGCfgSetCustomIntfRateAPI**

Sets the line data rate.

**Call** FG\_STATUS FGCfgSeCustomIntfRateAPI(FG\_HANDLE hModule, double dRate)

#### **Parameters**

- The module connection handle returned by fg\_connect.
- A value of type double that specifies the rate in MHz.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### **FGCfgSetWaveLength**

Selects the optical input wavelength.

**Call** FG\_STATUS FGCfgSetWaveLength(FG\_HANDLE hModule, FG\_WAVELENGTH\_TYPE nWavelength)

#### **Parameters**

- The module connection handle returned by fg\_connect.
- An enum value of type FG\_WAVELENGTH\_TYPE that specifies the wavelength.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### **FGCfgSetOptAttenRatioAPI**

Sets the value of any external optical input attenuation.

**Call** FG\_STATUS FGCfgSetOptAttenRatioAPI(FG\_HANDLE hModule, float fAttenRatio)

Informs the DCA of an external attenuator in the optical input path. The DCA measurements use this value so the measured values reflect the signal at the input of this attenuator.

#### **Parameters**

- The module connection handle returned by fg\_connect.
- A value of type float that specifies the ratio by which the attenuator reduces

the optical power.

Attenuation specified in dB may be converted to power ratio according to the following equation:

$$\text{Ratio} = 10^{\text{dB}/10}$$

Power ratio may be converted to dB according to the following equation:

$$\text{dB} = 10 \times \log_{10}(\text{ratio})$$

For example, a 3 dB optical attenuator reduces the power by a factor of 2:1; it would be entered as an attenuation ratio of 2.0. If no attenuator is used, this value should be set to 1.0.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### FGCfgSetElectAttenRatioAPI

Sets the value of any external electrical input attenuation.

**Call** FG\_STATUS FGCfgSetElectAttenRatioAPI(FG\_HANDLE hModule, float fAttenRatio)

Informs the DCA of an external attenuator in the electrical input path. The DCA measurements use this value so the measured values reflect the signal at the input of this attenuator.

### Parameters

- The module connection handle returned by fg\_connect.
- A value of type float that specifies the ratio by which the attenuator reduces the input voltage.

For example, a 6 dB electrical attenuator reduces the voltage by a factor of 2:1; it would be entered as an attenuation ratio of 2.0.

Attenuation specified in dB may be converted to voltage ratio according to the following equation:

$$\text{Ratio} = 10^{\text{dB}/20}$$

Voltage ratio may be converted to dB according to the following equation:

$$\text{dB} = 20 \times \log_{10}(\text{ratio})$$

For example, a 6 dB optical attenuator reduces the power by a factor of 2:1; it would be entered as an attenuation ratio of 2.0. If no attenuator is used, this value should be set to 1.0.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### **FGCfgSetRefClkSlrAPI**

Specifies the timing source (reference clock) for the DCA's data acquisition.

**Call** FG\_STATUS FGCfgSetRefClkSlrAPI(FG\_HANDLE hModule, FG\_REFCLKSLT nRefSelect)

#### **Parameters**

- The module connection handle returned by fg\_connect.
- An enum value of type FG\_REFCLKSLT that specifies the timing source.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### **FGCfgSetRefClkAPI**

Specifies the clock frequency to be used when the reference clock source selection is FG\_REFCLK\_EXTERNAL or FG\_REFCLK\_EXTERNAL\_AUTO.

**Call** FG\_STATUS FGCfgSetRefClkAPI(FG\_HANDLE hModule, double dFreq)

#### **Parameters**

- The module connection handle returned by fg\_connect.
- A value of type double that specifies the frequency in MHz.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### **FGCfgSetRecoveredClockRatioAPI**

Specifies the ratio of the line rate to the recovered clock rate when the reference clock source selection is FG\_REFCLK\_RECOVERED\_RATIO.

**Call** FG\_STATUS FGCfgSetRecoveredClockRatioAPI(FG\_HANDLE hModule, double dFreq)

#### **Parameters**

- The module connection handle returned by fg\_connect.
- A value of type float that specifies the ratio.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### **FGCfgSetWanderCompEnableAPI**

Enables or disables time base wander correction when acquiring data. Wander correction is always used when acquiring data with a mask.

**Call** FG\_STATUS FGCfgSetWanderCompEnableAPI(FG\_HANDLE hModule, unsigned long nEnable)

**Parameters**

- The module connection handle returned by fg\_connect.
- A value of type unsigned long that controls wander correction. Parameter values are as follows:  
0: Disable wander correction  
1: Enable wander correction

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

**FGCfgSetDCAcqSizeAPI**

Sets the number of samples to be taken during data acquisition.

**Call** FG\_STATUS FGCfgSetDCAcqSizeAPI(FG\_HANDLE hModule, unsigned long nSamples)

**Parameters**

- The module connection handle returned by fg\_connect.
- A value of type unsigned long that specifies the number of samples.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

**FGCfgSetInputSelectAPI**

Selects the optical or electrical signal input. For the N2100A, the unselected input must have no signal applied.

**Call** FG\_STATUS FGCfgSetInputSelectAPI(FG\_HANDLE hModule, unsigned long nInputSelect)

**Parameters**

- The module connection handle returned by fg\_connect.
- A value of type unsigned long that specifies the input. Parameter values are as follows:  
1: Electrical input  
2: Optical input

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### **FGCfgSetHWFilterAPI**

Selects the hardware input filter.

**Call** FG\_STATUS FGCfgSetHWFilterAPI(FG\_HANDLE hModule, FG\_FILTER\_TYPE nFilterType)

#### **Parameters**

- The module connection handle returned by fg\_connect.
- An enum value of type FG\_FILTER\_TYPE that specifies the filter.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### **FGCfgSetNRZThresholdAPI**

Selects the low and high amplitude threshold values used for NRZ measurements.

**Call** FG\_STATUS FGCfgSetNRZThresholdAPI(FG\_HANDLE hModule, FG\_HILOW\_VLTREF nThresholdSel)

#### **Parameters**

- The module connection handle returned by fg\_connect.
- An enum value of type FG\_HILOW\_VLTREF that specifies the thresholds.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### **FGCfgSetNRZWinSizeAPI**

Sets the start and end of the time window, in percent of 1 UI, over which NRZ measurements are made.

**Call** FG\_STATUS FGCfgSetNRZWinSizeAPI(FG\_HANDLE hModule, FG\_WINSIZE \* pWindow)

#### **Parameters**

- The module connection handle returned by fg\_connect.
- A pointer to a structure of type FG\_WINSIZE that specifies the window.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### FGCfgSetNRZOMAWinSizeAPI

Sets the start and end of the time window, in percent of 1 UI, over which NRZ OMA measurements are made. Because the OMA window spans the eye crossing, the start time of the window must be greater than the end time.

**Call** FG\_STATUS FGCfgSetNRZOMAWinSizeAPI(FG\_HANDLE hModule, FG\_WINSIZE \* pWindow)

#### Parameters

- The module connection handle returned by fg\_connect.
- A pointer to a structure of type FG\_WINSIZE that specifies the window.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

## Data Acquisition

- [“FGDCAAcquireAPI” on page 9-17.](#)
- [“FGDCAAcquireWithMaskAPI” on page 9-17.](#)
- [“FGDCAGetNextAPI” on page 9-18.](#)

---

### FGDCAAcquireAPI

Acquires DCA data using the current configuration settings, and optionally returns a structure containing the first block of sample values.

**Call** FG\_STATUS FGDCAAcquireAPI(FG\_HANDLE hModule, FG\_EYE\_PLOTDATA \* PlotData)

#### Parameters

- The module connection handle returned by fg\_connect.
- A pointer to a structure of type FG\_EYE\_PLOTDATA in which to return the sample data. If the pointer is NULL, no sample data are returned.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### FGDCAAcquireWithMaskAPI

Acquires DCA data using the current configuration settings, and optionally returns a structure containing the first block of sample values.

**Call** FG\_STATUS FGDCAAcquireWithMaskAPI(FG\_HANDLE hModule, FG\_MASK\_DATA \*pMask, UINT16

nMaskMarginType, UINT16 nMaskMargin, FG\_EYE\_PLOTDATA \* PlotData)

### Parameters

- The module connection handle returned by fg\_connect.
- A pointer to a structure of type FG\_MASK\_DATA that specifies the mask.
- An integer value that specifies the mask margin convention:
  - 0: Mask margins are compatible with Agilent 86100 mask margins.
  - 1: Mask margins are compatible with Tektronix CSA8000 mask margins.
- An integer value that specifies the mask margin in percent.
- A pointer to a structure of type FG\_EYE\_PLOTDATA in which to return the sample data. If the pointer is NULL, no sample data are returned.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### FGDCAGetNextAPI

Returns the next block of data from the most recent FGDCAAcquireAPI or FGDCAAcquireWithMaskAPI call.

**Call** FG\_STATUS FGDCAGetNextAPI(FG\_HANDLE hModule, FG\_EYE\_PLOTDATA \* PlotData)

### Parameters

- The module connection handle returned by fg\_connect.
- A pointer to a structure of type FG\_EYE\_PLOTDATA in which to return the sample data.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation. If no more sample data are available, the return value is FG\_NOMORE.



---

## Measurement

- “FGDCAAvgPowerAPI” on page 9-19.
- “FGDCAOscilloscopeAPI” on page 9-19.
- “FGDCAMaskTestAPI” on page 9-20.
- “FGDCAMaskTestAPI” on page 9-20.
- “FGNRZExtinctionRatioAPI” on page 9-20.
- “FGNRZEyeHeightAPI” on page 9-21.
- “FGNRZEyeAmplitudeAPI” on page 9-21.
- “FGNRZEyeWidthAPI” on page 9-21.
- “FGNRZOMAAPI” on page 9-22.
- “FGNRZLowAPI” on page 9-22.
- “FGNRZHighAPI” on page 9-22.
- “FGNRZSNRAPI” on page 9-22.
- “FGNRZBitRateAPI” on page 9-23.
- “FGNRZRiseTimeAPI” on page 9-23.
- “FGNRZFallTimeAPI” on page 9-23.
- “FGNRZCrossingPercentAPI” on page 9-24.
- “FGNRZDutyCyDistortionAPI” on page 9-24.
- “FGNRZJitterAPI” on page 9-24.
- “FGNRZRiseJitterAPI” on page 9-25.
- “FGNRZFallJitterAPI” on page 9-25.
- “FGNRZDeterministicJitterAPI” on page 9-25.

---

### FGDCAAvgPowerAPI

Returns the average optical power at the optical input.

**Call** FG\_STATUS FGDCAAvgPowerAPI(FG\_HANDLE hModule, FG\_RSP\_DCA\_AVGPWR \* pData)

#### Parameters

- The module connection handle returned by fg\_connect.
- A pointer to a structure of type FG\_RSP\_DCA\_AVGPWR to receive the results.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### FGDCAOscilloscopeAPI

Measures the basic oscilloscope measurement values and statistics on the acquired data.

**Call** FG\_STATUS FGDCAOscilloscopeAPI(FG\_HANDLE hModule, FG\_RSP\_DCA\_OSCILLOSCOPE \* pData)

**Parameters**

- The module connection handle returned by fg\_connect.
- A pointer to a structure of type FG\_RSP\_DCA\_OSCILLOSCOPE to receive the results.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

**FGDCAMaskTestAPI**

Applies the specified mask and margin to the acquired samples and returns the number of sample points that fail.

**Call** FG\_STATUS FGDCAMaskTestAPI(FG\_HANDLE hModule, FG\_MASK\_DATA \* pMask, UINT16 nMaskMarginType, UINT16 nMaskMargin, FG\_RSP\_DCA\_MASK \* pNumFailed)

**Parameters**

- The module connection handle returned by fg\_connect.
- A pointer to a structure of type FG\_MASK\_DATA that specifies the mask.
- An integer value that specifies the mask margin convention:
  - 0: Mask margins are compatible with Agilent 86100 mask margins.
  - 1: Mask margins are compatible with Tektronix CSA8000 mask margins.
- An integer value that specifies the mask margin in percent.
- A pointer to a structure of type FG\_RSP\_DCA\_MASKTEST that receives the number of failing points.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

**FGNRZExtinctionRatioAPI**

Measures extinction ratio on the acquired data.

**Call** FG\_STATUS FGNRZExtinctionRatioAPI(FG\_HANDLE hModule, FG\_RSP\_DCA\_EXTINCTION \* pData)

**Parameters**

- The module connection handle returned by fg\_connect.
- A pointer to a structure of type FG\_RSP\_DCA\_EXTINCTION to receive the results.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### **FGNRZEyeHeightAPI**

Measures eye height on the acquired data.

**Call** FG\_STATUS FGNRZEyeHeightAPI(FG\_HANDLE hModule, FG\_RSP\_DCA\_EYEHEIGHT \* pData)

#### **Parameters**

- The module connection handle returned by fg\_connect.
- A pointer to a structure of type FG\_RSP\_DCA\_EYEHEIGHT to receive the results.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### **FGNRZEyeAmplitudeAPI**

Measures eye amplitude on the acquired data.

**Call** FG\_STATUS FGNRZEyeAmplitudeAPI(FG\_HANDLE hModule, FG\_RSP\_DCA\_EYEAMPLITUDE \* pData)

#### **Parameters**

- The module connection handle returned by fg\_connect.
- A pointer to a structure of type FG\_RSP\_DCA\_EYEAMPLITUDE to receive the results.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### **FGNRZEyeWidthAPI**

Measures eye width on the acquired data.

**Call** FG\_STATUS FGNRZEyeWidthAPI(FG\_HANDLE hModule, FG\_RSP\_DCA\_EYEWIDTH \* pData)

#### **Parameters**

- The module connection handle returned by fg\_connect.
- A pointer to a structure of type FG\_RSP\_DCA\_EYEWIDTH to receive the results.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### **FGNRZOMAAPI**

Measures optical modulation amplitude on the acquired data.

**Call** FG\_STATUS FGNRZOMAAPI(FG\_HANDLE handle, FG\_RSP\_DCA\_OMA \* pData)

#### **Parameters**

- The module connection handle returned by fg\_connect.
- A pointer to a structure of type FG\_RSP\_DCA\_OMA to receive the results.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### **FGNRZLowAPI**

Measures the low level in the NRZ window on the acquired data.

**Call** FG\_STATUS FGNRZLowAPI(FG\_HANDLE hModule, FLOAT \* pData)

#### **Parameters**

- The module connection handle returned by fg\_connect.
- A pointer to a variable of type float to receive the results.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### **FGNRZHighAPI**

Measures the average high level within the NRZ window on the acquired data.

**Call** FG\_STATUS FGNRZHighAPI(FG\_HANDLE hModule, FLOAT \* pData)

#### **Parameters**

- The module connection handle returned by fg\_connect.
- A pointer to a variable of type float to receive the results.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### **FGNRZSNRAPI**

Measures signal-to-noise ratio on the acquired data.

**Call** FG\_STATUS FGNRZSNRAPI(FG\_HANDLE hModule, FLOAT \* pData)

### Parameters

- The module connection handle returned by fg\_connect.
- A pointer to a variable of type float to receive the results.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### FGNRZBitRateAPI

Returns the signal bit rate, or the reference clock frequency. The value returned depends on the reference clock selection. For internal clock, this call returns the line rate set by the user. For external clock, this call returns the frequency at the reference clock input. For recovered clock, this call returns the measured line rate. See “FGCfSetRefClkSlitAPI” on page 14.

**Call** FG\_STATUS FGNRZBitRateAPI(FG\_HANDLE hModule, FLOAT \* pData)

### Parameters

- The module connection handle returned by fg\_connect.
- A pointer to a variable of type float to receive the results.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### FGNRZRiseTimeAPI

Measures NRZ signal rise time between the programmed NRZ thresholds on the acquired data.

**Call** FG\_STATUS FGNRZRiseTimeAPI(FG\_HANDLE hModule, FG\_RSP\_DCA\_RISEFALL \* pData)

### Parameters

- The module connection handle returned by fg\_connect.
- A pointer to a variable of type FG\_RSP\_DCA\_RISEFALL to receive the results.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### FGNRZFallTimeAPI

Measures NRZ signal fall time between the programmed NRZ thresholds on the acquired data.

**Call** FG\_STATUS FGNRZFallTimeAPI(FG\_HANDLE hModule, FG\_RSP\_DCA\_RISEFALL \* pData)

### Parameters

- The module connection handle returned by fg\_connect.
- A pointer to a variable of type FG\_RSP\_DCA\_RISEFALL to receive the results.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### **FGNRZCrossingPercentAPI**

Measures the signal level at the eye crossing on the acquired data, expressed as a percentage of signal amplitude.

**Call** FG\_STATUS FGNRZCrossingPercentAPI(FG\_HANDLE hModule, FLOAT \* pData)

### Parameters

- The module connection handle returned by fg\_connect.
- A pointer to a variable of type float to receive the results.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### **FGNRZDutyCyDistortionAPI**

Measures the duty cycle distortion in percent on the acquired data, that is, the asymmetry between the time spent in the low state and the time spent in the high state.

**Call** FG\_STATUS FGNRZDutyCyDistortionAPI(FG\_HANDLE hModule, FG\_RSP\_DCA\_DCD \* pData)

### Parameters

- The module connection handle returned by fg\_connect.
- A pointer to a variable of type FG\_RSP\_DCA\_DCD to receive the results.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### **FGNRZJitterAPI**

Measures the signal edge jitter at the eye crossing level on the acquired data.

**Call** FG\_STATUS FGNRZJitterAPI(FG\_HANDLE hModule, FG\_RSP\_DCA\_JITTER \* pData)

### Parameters

- The module connection handle returned by fg\_connect.
- A pointer to a variable of type FG\_RSP\_DCA\_JITTER to receive the results.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### FGNRZRiseJitterAPI

Measures the rising edge jitter at the eye crossing level on the acquired data.

**Call** FG\_STATUS FGNRZRiseJitterAPI(FG\_HANDLE hModule, FG\_RSP\_DCA\_JITTER \* pData)

### Parameters

- The module connection handle returned by fg\_connect.
- A pointer to a variable of type FG\_RSP\_DCA\_JITTER to receive the results.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### FGNRZFallJitterAPI

Measures the falling edge jitter at the eye crossing level on the acquired data.

**Call** FG\_STATUS FGNRZFallJitterAPI(FG\_HANDLE hModule, FG\_RSP\_DCA\_JITTER \* pData)

### Parameters

- The module connection handle returned by fg\_connect.
- A pointer to a variable of type FG\_RSP\_DCA\_JITTER to receive the results.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### FGNRZDeterministicJitterAPI

Measures deterministic (data-dependent) edge jitter by measuring the average time position difference between LLH and HLH transitions, and between HHL and LHL transitions, at the eye crossing level.

**Call** FG\_STATUS FGNRZDeterministicJitterAPI(FG\_HANDLE hModule, FG\_RSP\_DCA\_DETJITTER \* pJitter);

### Parameters

- The module connection handle returned by fg\_connect.
- A pointer to a variable of type FG\_RSP\_DCA\_DETJITTER to receive the re-

sults.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

## Calibration

- [“FGCalDarkCalAPI” on page 9-26.](#)
- [“FGCalibrateDCAPI” on page 9-26.](#)
- [“FGCalCancelAPI” on page 9-26.](#)
- [“FGCalGetCalStatus” on page 9-26.](#)

---

### FGCalDarkCalAPI

Measures and saves the optical dark current level.

**Call** FG\_STATUS FGCalDarkCalAPI(FG\_HANDLE hModule)

**Parameter** The module connection handle returned by fg\_connect.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### FGCalibrateDCAPI

Starts delay calibration for the currently selected input and filter.

**Call** FG\_STATUS FGCalibrateDCAPI(FG\_HANDLE hModule)

**Parameter** The module connection handle returned by fg\_connect.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### FGCalCancelAPI

Stops any delay calibration operation in progress.

**Call** FG\_STATUS FGCalCancelAPI(FG\_HANDLE hModule)

**Parameter** The module connection handle returned by fg\_connect.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### FGCalGetCalStatus

Returns the state of the current delay calibration operation.



**Call** FG\_STATUS FGCalGetCalStatus(FG\_HANDLE hModule, unsigned long \*pProgressValue)

**Parameter**

- The module connection handle returned by fg\_connect.
- A pointer to an unsigned long integer that receives a value indicating the progress of the delay calibration operation.

This value starts at 0 and increments as calibration progresses; the exact final value depends on the delay calibration values found.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation. In particular, the function returns FG\_BUSY if calibration is running, or FG\_SUCCESS if it has finished.

---

## Logging

- “FGSyslogSyncTimestampAPI” on page 9-27.
- “FGSyslogSetLevelAPI” on page 9-27.
- “FGSyslogGetLevelAPI” on page 9-28.
- “FGSyslogReadAPI” on page 9-28.
- “FGSyslogClearAPI” on page 9-28.

The following logging functions are not available through the ActiveX control.

---

### FGSyslogSyncTimestampAPI

Sets the module timestamp equal to the host computer’s current time.

**Call** FGSyslogSyncTimestampAPI(FG\_HANDLE hModule)

**Parameter** The module connection handle returned by fg\_connect.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### FGSyslogSetLevelAPI

Sets the log level that specifies the types of information to be logged.

**Call** FGSyslogSetLevelAPI(FG\_HANDLE hModule, FG\_SYSLOG\_LEVEL level)

### Parameters

- The module connection handle returned by `fg_connect`.
- An enum value of type `FG_SYSLOG_LEVEL` that specifies the log level.

**Returns** A status value of type `FG_STATUS` reflecting the result of the operation.

---

### **FGSyslogGetLevelAPI**

Reads the log level. Set the log level that specifies the types of information to be logged.

**Call** `FGSyslogGetLevelAPI(FG_HANDLE hModule, FG_SYSLOG_LEVEL * pLevel)`

### Parameters

- The module connection handle returned by `fg_connect`.
- A pointer to an enum value of type `FG_SYSLOG_LEVEL` that receives the current log level.

**Returns** A status value of type `FG_STATUS` reflecting the result of the operation.

---

### **FGSyslogReadAPI**

Creates a file and copies the specified logged data to that file.

**Call** `FGSyslogReadAPI(FG_HANDLE hModule, char *filename, FG_SYSLOG_TYPE LogSelect)`

### Parameters

- The module connection handle returned by `fg_connect`.
- A pointer to a buffer that receives the system-generated file name.
- An enum value of type `FG_SYSLOG_TYPE` that specifies whether the operation is to read the application log or the system (kernel) log.

**Returns** A status value of type `FG_STATUS` reflecting the result of the operation.

---

### **FGSyslogClearAPI**

Clears the specified log.

**Call** `FGSyslogClearAPI(FG_HANDLE hModule, FG_SYSLOG_TYPE LogSelect)`

**Parameters**

- The module connection handle returned by `fg_connect`.
- An enum value of type `FG_SYSLOG_TYPE` that specifies whether the operation is to read the application log or the system (kernel) log.

**Returns** A status value of type `FG_STATUS` reflecting the result of the operation.

---

## Returned Status, Warning, and Error Values

This table lists some of the more common status, warning, and error values (FG\_STATUS) returned by module functions. A full list of status values appears in the API header file fgapidefs.h.

**Table 9-2. Status, Warning, and Error Values (1 of 3)**

FG_STATUS Values	Description
FG_SUCCESS	The operation was successful.
FG_BUSY	The calibration operation has not yet finished.
FG_WARN_GENERIC	Generic warning state.
FG_NOT_SUPPORTED	The specified operation is not supported by the module.
FG_CANCELLED	The calibration operation was cancelled by the user.
FG_NOMORE	There are no more data sample values available.
FG_DELAYCAL_REQD	Calibration of the delay lines is recommended.
FG_ERR_GENERIC	Generic error message.
FG_INVALID_COMMINTF	The specified module communication interface is not valid.
FG_MSG_SEND	An error occurred sending packet data to the module.
FG_MSG_RCV	An error occurred receiving the packet response from the module.
FG_INVALID_PKTTYPE	The packet type sent to the module was not valid.
FG_PKTID_UNEXPECTED	The response received from the module had a packet ID different from that of the command packet that prompted the response.
FG_BADCMD	The command sent to the module was invalid.
FG_INVALID_LOG_CMD	The status/error logging command sent to the module was invalid.
FG_NOTSUPPORTED	The specified operation is not supported by the module.
FG_TIMEOUT	A timeout occurred while performing the specified operation.
FG_INVALIDCONFIG	The specified module configuration is not valid.

**Table 9-2. Status, Warning, and Error Values (2 of 3)**

<b>FG_STATUS Values</b>	<b>Description</b>
FG_INVALID_CFG_ITEM	The configuration operation specifies an invalid parameter.
FG_INVALID_WAVELENGTH	The optical wavelength specified is invalid.
FG_INVALID_REFCLKSLT	The reference clock source specified is invalid.
FG_INVALID_REFCLK	The external reference clock frequency specified is invalid.
FG_INVALID_FILTER	The filter selection specified is invalid.
FG_INVALID_INTF	The interface type and line rate specified is invalid.
FG_INVALID_HILOREF	The voltage threshold selection specified is invalid.
FG_INVALID_WINSIZE	The window size specified is invalid.
FG_INVALID_TIAWINDOW	The TIA window specified is invalid.
FG_INVALID_CLOCKEDGE	The trigger edge specified is invalid.
FG_INVALID_MASKMARGIN	The specified mask margin is out of range.
FG_INVALID_CUSTOMRATE	The specified line rate is invalid.
FG_INVALID_INPUTSELECT	The specified input selection is invalid.
FG_ACQUIRE_FAILED	The module was unable to acquire sample data.
FG_NOSAMPLES	There is no acquired data on which to perform the measurement.
FG_NO_BUFFERS	The module was unable to allocate a data buffer.
FG_CONFIG_FAILED	An operation to update the module configuration failed.
FG_NVRREAD_FAILED	A failure occurred reading from the non-volatile memory (NVR).
FG_NVRWRITE_FAILED	A failure occurred writing to the NVR.
FG_CALFAILED	The calibration operation failed.
FG_BOARDINFO_INIT_FAILED	A failure occurred reading the module type and revision data.
FG_SETLOGLVLFAILED	The operation to set the status/error logging level failed.
FG_INTERNAL_ERROR	An unspecified internal error occurred.
FG_TIMESETFAILED	An attempt to set the module's time stamp failed.

**Table 9-2. Status, Warning, and Error Values (3 of 3)**

<b>FG_STATUS Values</b>	<b>Description</b>
FG_FILEOPEN	A file open operation failed or no file is currently open.
FG_NOSUCHFILE	The specified file was not found.
FG_NOSUCHDIR	The specified directory was not found.

---

## Data Structures

- “FG\_COMM\_PARAMS structure” on page 9-33.
- “FG\_TCPIP\_PARAMS structure” on page 9-33.
- “FG\_BUS\_PARAMS structure” on page 9-34.
- “FG\_FILTERCONFIG structure” on page 9-34.
- “FG\_CONFIGPKT structure” on page 9-34.
- “FG\_WINSIZE structure” on page 9-34.
- “FG\_XY\_DATA structure” on page 9-35.
- “FG\_EYE\_PLOTDATA structure” on page 9-35.
- “FG\_MASK\_DATA structure” on page 9-36.
- “FG\_RSP\_DCA\_AVGPWR structure” on page 9-36.
- “FG\_RSP\_DCA\_OSCILLOSCOPE structure” on page 9-36.
- “FG\_RSP\_DCA\_MASKTEST structure” on page 9-36.
- “FG\_RSP\_DCA\_VAMP structure” on page 9-37.
- “FG\_STATISTICS structure” on page 9-37.
- “FG\_RSP\_DCA\_EXTINCTION structure” on page 9-37.
- “FG\_RSP\_DCA\_EYEHEIGHT structure” on page 9-37.
- “FG\_RSP\_DCA\_EYEAMPLITUDE structure” on page 9-37.
- “FG\_RSP\_DCA\_EYEWIDTH structure” on page 9-37.
- “FG\_RSP\_DCA\_OMA structure” on page 9-38.
- “FG\_RSP\_DCA\_RISEFALL structure” on page 9-38.
- “FG\_RSP\_DCA\_DCD structure” on page 9-38.
- “FG\_RSP\_DCA\_JITTER structure” on page 9-38.
- “FG\_RSP\_DCA\_DETJITTER structure” on page 9-38.

---

### FG\_COMM\_PARAMS structure

```
typedef struct _FG_COMM_PARAMS {
    LONG pktid;

    FG_COMM_TYPES type;
    unsigned long rcvtimeout;
    void *pReserved;
    union {
        FG_TCPIP_PARAMS tcpip;
        FG_BUS_PARAMS bus;
    } comm_type;
} FG_COMM_PARAMS;
```

An integer to identify the packet; sequential calls typically use sequential numbers.

Specifies a PXI bus connection or a TCP/IP connection.

Timeout value (TCP/IP only).

For system use.

Parameters specific to a TCP/IP connection.

Parameters specific to a PXI bus connection.

---

### FG\_TCPIP\_PARAMS structure

```
typedef struct _FG_TCPIP_PARAMS {
    int fd;
    char ip_address[FG_IPMAXSIZE];

    struct sockaddr_in saddr;
```

File descriptor, for system use.

The IP address, a string of the form “192.100.40.1” The constant FG\_IPMAXSIZE is defined in the file fghostapi.h.

Socket address, for system use.

```
USHORT port;  
} FG_TCPIP_PARAMS;
```

The port number for the connection.

---

### FG\_BUS\_PARAMS structure

```
typedef struct _FG_BUS_PARAMS {  
    int fd;  
    char szInstrumentString[256];  
    UINT32 local_dma_mem_addr;  
    UINT32 local_dma_fifo_addr[8];  
} FG_BUS_PARAMS;
```

File descriptor, for system use.  
The PXI target module address string, eg. "PXI2::17::INSTR".  
PXI bus address values, for system use.

---

### FG\_FILTERCONFIG structure

```
typedef struct _FG_FILTERCONFIG {  
    float Filter[6];  
  
} FG_FILTERCONFIG;
```

Entries 0 through 4 reflect the frequencies of up to five installed filters. The number of input options and filters available for use depends on the module's input configuration. Entry 5 is unused.

---

### FG\_CONFIGPKT structure

```
typedef struct _FG_CONFIGPKT {  
    FG_INTF_TYPE    intftype;  
    FG_REFCLKSLT    refclkslt;  
    ULONG PatternAcqEnable;  
  
    double refclk;  
    double CustomIntfLineRate;  
    ULONG PatternAcqType;  
  
    ULONG PatternAcqLength;  
  
    FG_HILOW_VLTREF hi_low_volt_ref;  
    FG_TIA_WINDOW TIAWindow;  
    ULONG DCASampleSize;  
    FG_WINSIZE NRZWinWidth;  
    FG_WINSIZE NRZOMAWinWidth;  
    FG_HILOW_VLTREF NRZThreshold;  
    FG_WINSIZE RZWinWidth;  
    FG_HILOW_VLTREF RZThreshold;  
    FLOATRecoveredClockRatio;  
    ULONG Unused2;  
    FG_HILOW_VLTREF OscThreshold;  
    FG_FILTER_TYPE HWFilterType;  
    ULONG Unused1;  
    ULONG WanderCompEnable;  
    FLOAT OptAttenRatio;  
  
    FLOAT ElectAttenRatio;  
  
    FG_WAVELENGTH_TYPE WaveLength;  
    FG_INPUTSELECT InputSelect;  
} FG_CONFIGPKT;
```

(not used)  
Reference clock source selection.  
Enable pattern sequence acquisition (must be set to 0 when controlling DCA through API).  
External reference clock frequency.  
Line rate.  
Control parameter for pattern sequence acquisition (not used when controlling DCA through API).  
Control parameter for pattern sequence acquisition (not used when controlling DCA through API).  
(Not used).  
(Not used).  
Acquisition sample count.  
NRZ time window boundaries.  
OMA time window boundaries.  
NRZ amplitude threshold selection.  
(Not used)  
(Not used)  
RecoveredClockRatio  
(Not used)  
(Not used)  
DCA input filter selection.  
(Not used)  
Time base wander compensation enable.  
Ratio by which optical measurements are multiplied to compensate for an external optical attenuator.  
Ratio by which electrical measurements are multiplied to compensate for an external electrical attenuator.  
Optical wavelength selection.  
Optical or electrical input selection.

---

### FG\_WINSIZE structure

```
typedef struct _FG_WINSIZE {  
    ULONG start;  
    ULONG end;  
} FG_WINSIZE;
```

The position, in percent (0.01 UI), of the start of the window.  
The position, in percent, of the end of the window.



---

### FG\_XY\_DATA structure

```
typedef struct _FG_XY_DATA {  
    Used as an element of FG_EYE_PLOTDATA, below  
    FLOAT XCoordinate;           The X coordinate of the sample.  
    FLOAT YCoordinate;           The Y coordinate of the sample.  
} FG_XY_DATA;
```

---

### FG\_EYE\_PLOTDATA structure

```
typedef struct _FG_EYE_PLOTDATA {  
    USHORT nSamples;             The number of samples contained in this block.  
    FG_XY_DATA Sample[FG_EYE_MAXSEG_SIZE]; Array of sample X, Y coordinate values.  
    unsigned char transition[FG_EYE_MAXSEG_SIZE]; Refer to Table 9-3 for values.  
} FG_EYE_PLOTDATA;
```

**Table 9-3. Transition Field Values for FG\_EYE\_PLOTDATA**

Bits	Description	Value
Transition (bits 2..0)		
LLL	Steady-state low following a steady-state low.	0
LLH	Low-to-high transition following a steady-state low.	1
HHL	High-to-low transition following a steady-state high.	2
HHH	Steady-state high following a steady-state high.	3
HLL	Steady-state low following a high-to-low transition.	4
HLH	Low-to-high transition following a high-to-low transition.	5
LHL	High-to-low transition following a low-to-high transition.	6
LHH	Steady-state high following a low-to-high transition.	7
Invalid (bit 3)	Sample is not valid (see MeasPatternAcqDetJitterFall and MeasPatternAcqDetJitterRise).	8
Fail bits (bits 7..6)	Fail bits are applicable only for the AcquireWithMask acquisition functions.	
Fail	A ‘1’ in this bit indicates a point that failed the mask test (as modified by the mask margin, if a margin is used).	128 (0x80)
Fail, no margin	A ‘1’ in this bit indicates a point that would fail the mask without a margin. This bit is not used when the margin is 0.	64 (0x40)
Fail-region bits (bits 5..4)		
Fail center	This point failed the mask’s center region.	16 (0x10)

**Table 9-3. Transition Field Values for FG\_EYE\_PLOTDATA**

Bits	Description	Value
Fail upper	This point failed the mask's upper (overshoot) region.	32 (0x20)
Fail lower	This point failed the mask's lower (undershoot) region.	48 (0x30)

---

### FG\_MASK\_DATA structure

X values have the range 0 to 1 UI; Y values are expressed as a fraction of the signal amplitude, where 0.0 is the base level and 1.0 is the top level.

```
typedef struct FG_MASK_DATA {
    FLOAT X1;           Left extent of mask.
    FLOAT X2;           Left edge of rectangular region.
    FLOAT X3;           Right edge of rectangular region.
    FLOAT X4;           Right extent of mask.
    FLOAT X5;           Left intermediate X value.
    FLOAT X6;           Right intermediate X value.
    FLOAT Y0;           Center mask Y center line.
    FLOAT Y1;           Center mask lower limit.
    FLOAT Y2;           Center mask upper limit.
    FLOAT Y3;           Lower overshoot mask limit.
    FLOAT Y4;           Upper overshoot mask limit.
    FLOAT Y5;           Lower intermediate Y value.
    FLOAT Y6;           Upper intermediate Y value.
    UINT16 nMaskShape;  0: Rectangular mask. 1: 6-point mask with triangular end regions. 2: 10-point
                        mask with end regions and intermediate points.
    UINT16 bHasy3y4;    0: Overshoot masks are not present. 1: Overshoot masks are present.
    UINT16 nWindow;     0: Reference levels are based on full interval. > 0: Reference levels are based on
                        specified percentage of unit interval, centered in eye.
} FG_MASK_DATA;
```

---

### FG\_RSP\_DCA\_AVGPWR structure

```
typedef struct FG_RSP_DCA_AVGPWR {
    FLOAT dB;           Power in dBm.
    FLOAT watts;        Power in watts.
    FLOAT ratio;         The ratio of the average power to 0 dBm, equivalent to the average power in
                        mW.
} FG_RSP_DCA_AVGPWR;
```

---

### FG\_RSP\_DCA\_OSCILLOSCOPE structure

This structure is a wrapper for a FG\_RSP\_DCA\_VAMP structure.

```
typedef struct FG_RSP_DCA_OSCILLOSCOPE {
    FG_RSP_DCA_VAMP amplitude;
} FG_RSP_DCA_OSCILLOSCOPE;
```

---

### FG\_RSP\_DCA\_MASKTEST structure

```
typedef struct FG_RSP_DCA_MASKTEST {
    UINT32 nFailTotal;   Total count of failing points.
    UINT32 nFailCenter;  Count of points that fail the center region.
    UINT32 nFailUpper;   Count of points that fail the upper region.
}
```

---

```
    UINT32 nFailLower;           Count of points that fail the lower region.
} FG_RSP_DCA_MASKTEST;
```

---

### FG\_RSP\_DCA\_VAMP structure

This structure contains a FG\_STATISTICS structure and a number of individual values.

```
typedef struct _FG_RSP_DCA_VAMP {
    FG_STATISTICS stats;
    FLOAT posovershoot;           Positive overshoot, in percent of amplitude
    FLOAT negovershoot;           Negative overshoot, in percent of amplitude
    FLOAT vbase;                  Signal base level, in uW (optical) or mV (electrical)
    FLOAT vtop;                   Signal top level
    FLOAT vamplitude;             Signal amplitude
} FG_RSP_DCA_VAMP;
```

---

### FG\_STATISTICS structure

```
typedef struct FG_STATISTICS {
    float mean;                   Average value
    float stddev;                 Standard deviation or RMS value
    float maximum;                Maximum value
    float minimum;                Minimum value
    float pkpk;                   Peak-to-peak amplitude
    float max_pos_deviation;       Maximum positive deviation
    float max_neg_deviation;       Maximum negative deviation
} FG_STATISTICS;
```

---

### FG\_RSP\_DCA\_EXTINCTION structure

```
typedef struct _FG_RSP_DCA_EXTINCTION {
    float dB;                     Extinction ratio in dB
    float percent;                Extinction ratio in percent
    float ratio;                  Extinction ratio
} FG_RSP_DCA_EXTINCTION;
```

---

### FG\_RSP\_DCA\_EYEHEIGHT structure

```
typedef struct _FG_RSP_DCA_EYEHEIGHT {
    float dB;                     (Not used)
    float amplitude;              Eye height
    float min;                    Lower extent of eye height range
    float max;                    Upper extent of eye height range
} FG_RSP_DCA_EYEHEIGHT;
```

---

### FG\_RSP\_DCA\_EYEAMPLITUDE structure

```
typedef struct _FG_RSP_DCA_EYEAMPLITUDE {
    float amplitude;              Eye amplitude
    float min;                    Lower extent of eye amplitude range
    float max;                    Upper extent of eye amplitude range
} FG_RSP_DCA_EYEAMPLITUDE;
```

---

### FG\_RSP\_DCA\_EYEWIDTH structure

```
typedef struct _FG_RSP_DCA_EYEWIDTH {
    float ratio;                  Eye width in unit intervals (UI)
    float time;                   Eye width in picoseconds
    float start;                  Left extent of eye, in unit intervals (UI)
}
```

float end;	Right extent of eye
float ymid;	Signal amplitude at eye crossing
} FG_RSP_DCA_EYEWIDTH;	

---

### **FG\_RSP\_DCA\_OMA structure**

typedef struct _FG_RSP_DCA_OMA {	
float time;	Transition time in picoseconds
float start;	Start position of transition, in unit intervals (UI)
float end;	End position of transition
} FG_RSP_DCA_OMA;	

---

### **FG\_RSP\_DCA\_RISEFALL structure**

typedef struct _FG_RSP_DCA_RISEFALL {	
float time;	Transition time in picoseconds
float start;	Start position of transition, in unit intervals (UI)
float end;	End position of transition
float thresholdL;	Low threshold used for measurement
float thresholdH;	High threshold used for measurement
} FG_RSP_DCA_RISEFALL;	

---

### **FG\_RSP\_DCA\_DCD structure**

typedef struct _FG_RSP_DCA_DCD {	
float time;	Duty cycle distortion in picoseconds
float percent;	Duty cycle distortion, in percent
float risemid;	Rising edge midpoint, in unit intervals (UI)
float fallmid;	Falling edge midpoint, in UI
} FG_RSP_DCA_DCD;	

---

### **FG\_RSP\_DCA\_JITTER structure**

typedef struct _FG_RSP_DCA_JITTER {	
float pkpk;	Peak-to-peak jitter amplitude in unit intervals (UI)
float rms;	RMS jitter amplitude in UI
} FG_RSP_DCA_JITTER;	

---

### **FG\_RSP\_DCA\_DETJITTER structure**

typedef struct _FG_RSP_DCA_DETJITTER {	
float fRiseJitter;	Time difference, in picoseconds, between the average time position of a LLH transition and the average time position of a HLH transition.
float fFallJitter;	Time difference, in picoseconds, between the average time position of a HHL transition and the average time position of a LHL transition.
float fRiseLLH;	Average time position, in unit intervals (UI), of a LLH transition.
float fRiseHLH;	Average time position, in UI, of a HLH transition.
float fFallHHL;	Average time position, in UI, of a HHL transition.
float fFallLHL;	Average time position, in UI, of a LHL transition.
} FG_RSP_DCA_DETJITTER;	

## Enum Values

**Table 9-4. Enum Values (1 of 3)**

Integer Value	Enum Value	Description
<b>Module communications link FG_COMM_TYPES enum values</b>		
0	FG_COMM_TCPIP	TCP/IP
1	FG_COMM_BUS	PXI bus
<b>Input FG_INPUTSELECT enum values</b>		
1	FG_INPUTSELECT_ELECTRICAL	Electrical input
2	FG_INPUTSELECT_OPTICAL	Optical input
<b>Line rate FG_WAVELENGTH_TYPE enum values</b>		
1	FG_WAVELENGTH_850	850 nm
2	FG_WAVELENGTH_1310	1310 nm
3	FG_WAVELENGTH_1550	1550 nm
<b>Reference clock source FG_REFCLK_SLT enum values</b>		
1	FG_REFCLK_STRATUM3	The internal reference clock.
2	FG_REFCLK_EXTERNAL	A user-provided clock signal connected to the External Clock input. The frequency of this clock must be between 10 MHz and 8.5 GHz and must match the value of the External Reference Clock Frequency parameter.
3	FG_REFCLK_RECOVERED	The clock derived from the received optical or electrical data.
4	FG_REFCLK_STRATUM3_AUTO	The internal reference clock. After acquisition, the data are analyzed and corrected, if possible, for small deviations from the stated line rate.
5	FG_REFCLK_EXTERNAL_AUTO	A user-provided clock signal, as in FG_REFCLK_EXTERNAL. After acquisition, the data are analyzed and corrected, if possible, for small deviations from the stated line rate or reference frequency.

**Table 9-4. Enum Values (2 of 3)**

Integer Value	Enum Value	Description
6	FG_REFCLK_RECOVERED_RATIO	The clock derived from the received optical or electrical signal. This option allows specifying the ratio of the line rate to the recovered clock signal, for example, when the N2100B's wideband clock recovery detects a K28.7 pattern at one-fifth the actual rate. See the Note on page 9-41.
<b>Hardware filter (API) and input path/filter (ActiveX) selection values FG_FILTER_TYPE enum values</b>		
1	FG_HWFILTER_1	First selection, as returned by the FGGetInputConfigAPI and FGGetFilterConfigAPI API calls and the GetModuleInformation ActiveX call.
2	FG_HWFILTER_2	Second selection, if present.
3	FG_HWFILTER_3	Third selection, if present.
4	FG_HWFILTER_4	Fourth selection, if present.
5	FG_HWFILTER_HS	Fifth selection, if present.
<b>Thresholds FG_HILOW_VLTREF enum values</b>		
1	FG_VLTREF_9010	10% to 90% of amplitude over full UI period.
2	FG_VLTREF_8020	20% to 80% of amplitude over full UI period.
3	FG_VLTREF_7030	30% to 70% of amplitude over full UI period.
4	FG_VLTREF_9010_WIN	10% to 90% of amplitude within NRZ window.
5	FG_VLTREF_8020_WIN	20% to 80% of amplitude within NRZ window.
6	FG_VLTREF_7030_WIN	30% to 70% of amplitude within NRZ window.
<b>Information logged FG_SYSLOG_LEVEL enum values</b>		
5	FG_SYSLOG_NONE	Logging is disabled.
4	FG_SYSLOG_FATAL	Fatal errors only.
3	FG_SYSLOG_ERROR	All errors.
2	FG_SYSLOG_WARNING	All errors and warnings.
1	FG_SYSLOG_INFO	All errors and warning, as well as other significant information.

Table 9-4. Enum Values (3 of 3)

Integer Value	Enum Value	Description
0	FG_SYSLOG_DEBUG	All of the above, plus internal information chosen for debugging purposes.
	<b>Log selected FG_SYSLOG_TYPE enum values</b>	
1	FG_SYSLOG_APP	Application (user-level) log.
2	FG_SYSLOG_SYSTEM	System (kernel) log.

**NOTE**

The N2100B clock recovery does not function well with signals that can be interpreted as lower than the actual bit rate (for example, the K28.7 pattern is a repeating pattern of five ones followed by five zeroes; this looks the same as a pattern of alternating ones and zeroes at one-fifth of the actual bit rate). The N2100A does not have this restriction. To deal with this restriction, you must select enum 6 or FG\_REFCLK\_RECOVERED\_RATIO as the reference clock source (RefClockSel). This sets the clock recovery mode to expect a signal that will be detected as a lower rate than it should. You then enter a multiplier using RecoveredClockRatio; the detected clock frequency is multiplied by this number to obtain the actual line rate. For example, for a K28.7 signal, the multiplier is 5.

---

## ActiveX API Methods

- [“Module Information” on page 9-42.](#)
- [“Module Connection” on page 9-45.](#)
- [“Mask Configuration” on page 9-47.](#)
- [“DCA Sample Data Acquisition” on page 9-50.](#)
- [“Measurement” on page 9-54.](#)
- [“Pattern Sequence Acquisition” on page 9-65.](#)
- [“Calibration” on page 9-69.](#)
- [“Control Display” on page 9-70.](#)

---

## Module Information

---

### GetModuleInformation

**Call** BSTR GetModuleInformation(long nIndex)

Returns the specified module information (module type, serial number...) as a string.

**Parameters** An integer value that specifies the information to be returned. Refer to the table for the list of valid values and the information returned.

**Returns** A string containing the specified module information. Note that all values, including numeric values, are returned as character strings. Conversion to numeric values or other types is the responsibility of the user.

**Table 9-5. GetModuleInformation Index Values And Information Returned**

nIndex value	Information returned	Description
0	Module type string	Alphanumeric string, for example, N2100B.



**Table 9-5. GetModuleInformation Index Values And Information Returned**

nIndex value	Information returned	Description
1	Module serial number	String of digits, for example, 04101234.
2	Module revision number	String of one or more digits.
3	FPGA revision number	String of the form a.bb or aa.bb, where aa and bb are the major and minor FPGA revision numbers, respectively.
4	API revision number	String of digits, for example, 20.
5	Calibration API revision	String of digits, for example, 20.
6	Maintenance API revision	String of digits, for example, 20.
7	Driver version	String of the form a.bbb or aa.bbb, where aa and bbb are the major and minor driver revision numbers, respectively.
8	Firmware version	String of the form a.bbb or aa.bbb, where aa and bbb are the major and minor firmware revision numbers, respectively.
9	Firmware build date	String of the form Nov 18 2004.
10	ActiveX control build date	String of the form Nov 18 2004.
11	Input configuration and filter specifiers	See Input configuration string, below.
12	Input / filter #1 descriptor	See Input / Filter descriptor, below.
through	through	
16	Input / Filter #5 descriptor	
17	Pattern acquisition filter files resident on the module	A comma-separated list of file names for the pattern acquisition filter files that reside in the DCA's flash memory.
18	All pattern acquisition filter files available for use	A comma-separated list of file names for the pattern acquisition filter files in the local DCA Data Files directory that match the DCA's serial number.
19	ActiveX control version	Alphanumeric string, for example, 2.4.1.0.

- Input configuration string in the following form:  
1,5,O:4250.0000,O:2488.3200,O:2125.0000,O:1062.5000,E:0.0000

Where each of the fields is separated by a comma:

- A number that identifies the module's input configuration, as returned by FGGetInputConfigAPI.
- The number of input selections available. Refer to [Table 9-6](#).
- One or more Input / Filter descriptors (see below), matching the number of available input selections. The values in [Table 9-6](#) reflect the module's signal routing configuration.
- Input / Filter Descriptor string in the following form:

O:4250.0000

Where the content is:

The letter O or E, specifies Optical or Electrical input, respectively, or the strings ED, E+, or E-, specify the differential, positive, or negative inputs of the electrical differential DCA.

A colon.

A string of digits representing a floating-point number that specifies the frequency of the filter for that input selection. A numeric value of 0 indicates no filter.

**Table 9-6. Returned Values**

Value <sup>a</sup>	Available Input Selections	Module Input Configuration
1	5	Optical input to internal filter bank, electrical input direct.
2	2	Optical input direct to high-speed path, electrical input direct.
3	2	Optical input to high-speed path with filter, electrical input direct.
4	5	Optical input to external high-performance filter bank, electrical input direct.
5	1	Electrical only, direct.
6	1	Electrical only, to high-speed path with filter.
7	4	Electrical only, to internal filter bank.
8	4	Electrical only, to external high-performance filter bank.
9	4	Optical only, to internal filter bank.

**Table 9-6. Returned Values**

Value <sup>a</sup>	Available Input Selections	Module Input Configuration
10	1	Optical only, to high-speed path.
11	1	Optical only, to high-speed path with filter.
12	4	Optical only, to external high-performance filter bank.
13	3	Electrical with differential input.
14	5	Optical input to external high-performance filter bank through high-speed path, electrical input direct.
15	4	Optical only, to external high-performance filter bank through high-speed path.
16	5	N2100B with four optical filter selections, includes electrical input..
17	4	N2100B with three optical filter selections.
18	3	N2100B with two optical filter selections.
19	2	N2100B with a single optical filter selection.
20	2	N2100B with no filter assembly, electrical and optical input selections.

a. Values 1 through 15 apply to the N2100A. Values 16 through 20 apply to the N2100B.

---

## Module Connection

- [“OpenPxi” on page 9-45.](#)
- [“OpenPxiByVal” on page 9-46.](#)
- [“OpenTcp” on page 9-46.](#)
- [“OpenTcpByVal” on page 9-46.](#)
- [“Close” on page 9-47.](#)

---

### OpenPxi

Establishes a connection between the DCA ActiveX control and the module via the PXI, specified by the parameter.

**Call** FG\_STATUS OpenPxi(BSTR \* plnstrumentString)

**Parameter** An appropriately formatted character string, of the form PXI2::17::INSTR, that specifies the target DCA module.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### **OpenPxiByVal**

Sets up a connection to a DCA module through the PXI bus, specified by the parameter. This is an alternative form of OpenPxi, for applications that have difficulty passing the required BSTR pointer to OpenPxi.

**Call** FG\_STATUS OpenPxiByVal(\_TCHAR \* pszInstrString)

**Parameter** An appropriately formatted character string, of the form PXI2::17::INSTR, that specifies the target DCA module.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### **OpenTcp**

Establishes a connection between the DCA ActiveX control and the module via TCP/IP (network), specified by the parameter.

**Call** FG\_STATUS OpenTcp(BSTR \* plpAddr)

**Parameter** An appropriately formatted character string, of the form 192.168.22.40, that specifies the network IP address of the target DCA module.

The connection port may be specified explicitly by including it in the address string, separated from the IP address by a colon, for example, 192.168.22.40:1500.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### **OpenTcpByVal**

Set up a connection to a DCA module through TCP/IP (network). This is an alternative form of OpenTcp, for applications that have difficulty passing the required BSTR pointer to OpenTcp.

**Call** FG\_STATUS OpenTcpByVal(\_TCHAR \* pszInstrString)

**Parameter** An appropriately formatted character string, of the form 192.168.22.40, that specifies the network IP address of the target DCA module.

The connection port may be specified explicitly by including it in the address string, separated from the IP address by a colon, e.g., 192.168.22.40:1500.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### Close

Disconnects the connection between the ActiveX control and the DCA ActiveX control and the DCA module.

**Call** FG\_STATUS Close( )

**Parameters** None

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

## Mask Configuration

- [“SetDCAMask” on page 9-47.](#)
- [“SetDCAMaskVariant” on page 9-48.](#)
- [“SetDCAMaskWindowed” on page 9-49.](#)
- [“SetDCAMaskWindowedVariant” on page 9-49.](#)

---

### SetDCAMask

**Call** void SetDCAMask(FLOAT \* fMaskArray)

Sets the mask definition to be used for mask tests. (See also [“SetDCAMaskVariant” on page 9-48.](#))

**Parameters** An array of floating-point values for the mask definition.

**Parameter values** The mask definition is an array of 13 (for SetDCAMask) or 14 (for SetDCAMaskWindowed) values of type FLOAT.

Mask shape

Rectangular: Set X1 and X4 both 0

6-point: Set X5 and X6 both 0

10-point: Use all X, Y values for center mask

Overshoot masks

Not present: Set Y3, Y4 both 0

Present: Set Y3, Y4 to desired limit values

**Table 9-7. Mask Definition**

Offset	Function
0	X1: Left extent of mask
1	X2: Left edge of rectangular region
2	X3: Right edge of rectangular region
3	X4: Right extent of mask
4	X5: Left intermediate X value
5	X6: Right intermediate X value
6	Y0: Center mask Y center line
7	Y1: Center mask lower limit
8	Y2: Center mask upper limit
9	Y3: Lower overshoot mask limit
10	Y4: Upper overshoot mask limit
11	Y5: Lower intermediate Y value
12	Y6: Upper intermediate Y value
13	For SetDCAMaskWindowed and SetDCAMaskWindowedVariant, this parameter specifies the window width in percent over which the low and high Y reference levels are measured. For SetDCAMask and SetDCAMaskVariant, this parameter is not used and need not be present.

---

### **SetDCAMaskVariant**

Sets the mask parameters used for mask tests. This is an alternative form of SetDCAMask, for applications that have difficulty directly passing array parameters.

**Call** void SetDCAMaskVariant(VARIANT \* vMaskArray)

Sets the mask definition to be used for mask tests.

**Parameters** A VARIANT containing an array of floating-point values for the mask definition.

**Parameter values** [Refer to “Mask Definition” on page 9-48.](#)

---

### **SetDCAMaskWindowed**

Sets the mask parameters used for mask tests (Y levels based on windowed low and high measurements).

**Call** void SetDCAMask(FLOAT \* fMaskArray)

Sets the mask definition to be used for mask tests.

(See also [“SetDCAMaskVariant” on page 9-48.](#))

**Parameters** An array of floating-point values for the mask definition.

**Parameter values** [Refer to “Mask Definition” on page 9-48.](#)

---

### **SetDCAMaskWindowedVariant**

Sets the mask parameters used for mask tests (Y levels based on windowed low and high measurements).

This is an alternative form of SetDCAMaskWindowed, for applications that have difficulty directly passing array parameters.

**Call** void SetDCAMask(VARIANT \* vMaskArray)

Sets the mask definition to be used for mask tests.

**Parameters** A VARIANT containing an array of floating-point values for the mask definition.

**Parameter values** [Refer to “Mask Definition” on page 9-48.](#)

---

## DCA Sample Data Acquisition

- [“AcquireDCAWithMask” on page 9-50.](#)
- [“GetDCASampleCount” on page 9-50.](#)
- [“GetDCASampleData” on page 9-51.](#)
- [“GetDCASampleDataVariant” on page 9-52.](#)

---

### AcquireDCA

Acquires DCA sample data.

**Call** long Acquire( )

Acquires DCA data using the current configuration settings.

**Parameters** None

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### AcquireDCAWithMask

Acquires DCA sample data and performs a mask test.

**Call** long AcquireWithMask(int nMaskMarginType, int nMaskMargin)

Acquires DCA data using the current configuration settings and performs a mask test on it using the current mask.

**Parameters**

- An integer value that specifies the mask margin convention:
  - 0: Mask margins are compatible with Agilent 86100 mask margins.
  - 1: Mask margins are compatible with Tektronix CSA8000 mask margins.
- An integer value that specifies the mask margin in percent.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### GetDCASampleCount

Returns the number of samples acquired. This call is applicable only when the ActiveX control window is displayed. Otherwise, the sample data are never transferred to the ActiveX control and is therefore not available.



**Call** long GetDCASampleCount( )

Returns the number of valid samples acquired.

**Parameters** None

---

### GetDCASampleData

Return the sample values acquired. This call is not applicable when the DisplayMode property is 0; in that case the sample data are never transferred to the ActiveX control and is therefore not available.

**Call** long GetDCASampleData(long nCount, float \* fXArray, float \* fYArray, short \* nTransArray)

Returns the X / Y values and transition types of the acquired samples. See also [“GetDCASampleDataVariant” on page 9-52.](#)

**Parameters**

- An integer value that specifies the number of sample points to return. The number of points returned will be this number or the total number acquired, whichever is less.
- A pointer to an array of type float that receives the sample X coordinates.
- A pointer to an array of type float that receives the sample Y coordinates.
- A pointer to an array of type unsigned short that receives the sample transition-type codes. See [Table 9-8](#) for transition type values for these codes.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

**Table 9-8. Transition Field Values (1 of 2)**

Bits	Description	Value
Transition (bits 2..0)		
LLL	Steady-state low following a steady-state low.	0
LLH	Low-to-high transition following a steady-state low.	1
HHL	High-to-low transition following a steady-state high.	2
HHH	Steady-state high following a steady-state high.	3
HLL	Steady-state low following a high-to-low transition.	4

**Table 9-8. Transition Field Values (2 of 2)**

Bits	Description	Value
HLH	Low-to-high transition following a high-to-low transition.	5
LHL	High-to-low transition following a low-to-high transition.	6
LHH	Steady-state high following a low-to-high transition.	7
Invalid (bit 3)	Sample is not valid (see MeasPatternAcqDetJitterFall and MeasPatternAcqDetJitterRise).	8
Fail bits (bits 7..6)	Fail bits are applicable only for the AcquireWithMask acquisition functions.	
Fail	A '1' in this bit indicates a point that failed the mask test (as modified by the mask margin, if a margin is used).	128 (0x80)
Fail, no margin	A '1' in this bit indicates a point that would fail the mask without a margin. This bit is not used when the margin is 0.	64 (0x40)
Fail-region bits (bits 5..4)		
Fail center	This point failed the mask's center region.	16 (0x10)
Fail upper	This point failed the mask's upper (overshoot) region.	32 (0x20)
Fail lower	This point failed the mask's lower (undershoot) region.	48 (0x30)

---

### GetDCASampleDataVariant

Returns the sample values acquired. This call is not applicable when the DisplayMode property is 0; in that case the sample data are never transferred to the ActiveX control and is thus not available.

This is an alternative form of GetDCASampleData, for applications that have difficulty directly passing array parameters.

**Call** long GetDCASampleData(long nCount, VARIANT \* vXArray, VARIANT \* vYArray, VARIANT \* vTransArray)

Returns the X / Y values and transition types of the acquired samples.

### Parameters

- An integer value that specifies the number of sample points to return. The number of points returned will be this number or the total number acquired,

whichever is less.

- A pointer to a variant containing an array of type float that receives the sample X coordinates.
- A pointer to a variant containing an array of type float that receives the sample Y coordinates.
- A pointer to a variant containing an array of type unsigned short that receives the sample transition-type codes. See [Table 9-8](#) for transition type values for these codes.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

## Measurement

- [“MeasAvgPowerUW” on page 9-55.](#)
- [“MeasAvgPowerDBM” on page 9-55.](#)
- [“GetMaskTestFailPoints” on page 9-55.](#)
- [“GetMaskTestFailPointsAll” on page 9-55.](#)
- [“GetMaskTestFailPointsAllVariant” on page 9-56.](#)
- [“FindThresholdMaskMargin” on page 9-56.](#)
- [“ExpandDCAMaskMargin” on page 9-57.](#)
- [“ExpandDCAMaskMarginVariant” on page 9-57.](#)
- [“GetBitRate” on page 9-58.](#)
- [“MeasOscAmplitude” on page 9-58.](#)
- [“MeasOscRMS” on page 9-58.](#)
- [“MeasOscPeakToPeak” on page 9-58.](#)
- [“MeasOscMinLevel” on page 9-59.](#)
- [“MeasOscMaxLevel” on page 9-59.](#)
- [“MeasOscMaxLevel” on page 9-59.](#)
- [“MeasOscBaseLevel” on page 9-59.](#)
- [“MeasOscTopLevel” on page 9-59.](#)
- [“MeasOscPosOvershoot” on page 9-60.](#)
- [“MeasOscNegOvershoot” on page 9-60.](#)
- [“MeasNRZExtinctionRatio” on page 9-60.](#)
- [“MeasNRZExtinctionRatioDB” on page 9-60.](#)
- [“MeasNRZExtinctionRatioHist” on page 9-60.](#)
- [“MeasNRZExtinctionRatioDBHist” on page 9-61.](#)
- [“MeasNRZEyeHeight” on page 9-61.](#)
- [“MeasNRZEyeAmplitude” on page 9-61.](#)
- [“MeasNRZEyeWidth” on page 9-61.](#)
- [“MeasNRZOMA” on page 9-62.](#)
- [“MeasNRZLowLevel” on page 9-62.](#)
- [“MeasNRZHighLevel” on page 9-62.](#)
- [“MeasNRZSNR” on page 9-62.](#)
- [“MeasNRZCrossingPercent” on page 9-62.](#)
- [“MeasNRZRiseTime” on page 9-63.](#)
- [“MeasNRZFallTime” on page 9-63.](#)
- [“MeasNRZDutyCycleDistortion” on page 9-63.](#)
- [“MeasNRZJitterPeakToPeak” on page 9-63.](#)
- [“MeasNRZJitterRMS” on page 9-64.](#)
- [“MeasNRZSingleEdgeJitterPP” on page 9-64.](#)
- [“MeasNRZSingleEdgeJitterRMS” on page 9-64.](#)
- [“MeasNRZDeterministicJitterRise” on page 9-64.](#)

- [“MeasNRZDeterministicJitterFall” on page 9-65.](#)

---

### **MeasAvgPowerUW**

Measures average optical power in uW.

**Call** float MeasAvgPowerUW( )  
Measures the average optical power.

**Parameters** NA

**Returns** A float value representing the optical power in microwatts.

---

### **MeasAvgPowerDBM**

Measures average optical power in dBm.

**Call** float MeasAvgPowerDBM( )  
Measures the average optical power.

**Parameters** NA

**Returns** A float value representing the optical power in dBm.

---

### **GetMaskTestFailPoints**

Gets the count of failing points from a mask test.

**Call** long GetMaskTestFailPoints( )  
Gets the results of the most recent AcquireDCAWithMask operation.

**Parameters** NA

**Returns** A long value representing the count of samples that failed the mask test.

---

### **GetMaskTestFailPointsAll**

Gets the count of failing points and separate counts for each mask region.

**Call** long GetMaskTestFailPointsAll(long \*pResult )  
Gets the results of the most recent AcquireDCAWithMask operation.

**Parameters** A pointer to an array of four long values that receive the mask test results, as follows:

- [0] The total count of failing sample points.
- [1] The count of points that fail the center mask region.
- [2] The count of points that fail the upper mask region.
- [3] The count of points that fail the lower mask region.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### **GetMaskTestFailPointsAllVariant**

Gets the count of failing points and separate counts for each mask region. This is an alternative form of GetMaskTestFailPointsAll, for use with applications that have difficulty passing an array as a parameter.

**Call** VARIANT GetMaskTestFailPointsAllVariant( )

Gets the results of the most recent AcquireDCAWithMask operation.

**Parameters** NA

**Returns** A VARIANT, containing an array of four long values that receive the mask test results as follows:

- [0] The total count of failing sample points.
- [1] The count of points that fail the center mask region.
- [2] The count of points that fail the upper mask region.
- [3] The count of points that fail the lower mask region.

---

### **FindThresholdMaskMargin**

Determines the mask margin that yields a specified number of failing points.

**Call** long FindThresholdMaskMargin(long nFailingPoints)

Determines the maximum mask margin value that results in no more than the specified number of points that fail the mask test.

**Parameters** A long value that specifies the maximum number of failing points allowed.

**Returns** A long value representing the mask margin.

---

## ExpandDCAMaskMargin

Returns the coordinates of the mask vertices with a mask margin applied.

**Call** long ExpandDCAMaskMargin(float \*fArray, long nMarginType, long nMargin)

Applies the specified margin to the current mask and returns the vertices of the mask that results. Use SetDCAMask to specify the current mask.

### Parameters

- An array of floating-point values, in the format used by SetDCAMask, that receives the result. [Refer to “Mask Definition” on page 9-48](#) for the format of this array.
- An integer value that specifies the mask margin convention:  
0: Mask margins are compatible with Agilent 86100 mask margins.  
1: Mask margins are compatible with Tektronix CSA8000 mask margins.
- An integer value that specifies the mask margin in percent.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

## ExpandDCAMaskMarginVariant

Returns the coordinates of the mask vertices with a mask margin applied.

This is an alternative form of ExpandDCAMaskMargin, for applications that have difficulty directly passing array parameters.

**Call** long ExpandDCAMaskMargin(VARIANT \* pvArray, long nMarginType, long nMargin)

Applies the specified margin to the current mask and returns the vertices of the mask that results. Use SetDCAMask to specify the current mask.

### Parameters

- A pointer to a VARIANT. On return from this function, the VARIANT contains an array of floating-point values, in the format used by SetDCAMask, that receives the result. [Refer to “Mask Definition” on page 9-48](#) for the format of this array.
- An integer value that specifies the mask margin convention:  
0: Mask margins are compatible with Agilent 86100 mask margins.

1: Mask margins are compatible with Tektronix CSA8000 mask margins.

- An integer value that specifies the mask margin in percent.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### **GetBitRate**

Returns the current bit rate or equivalent bit rate, depending on the reference clock selected:

- Internal clock returns the line rate set by the user.
- External clock returns the frequency at the reference clock input.
- Recovered clock returns the measured line rate.

**Call** float GetBitRate( )

**Parameters** NA

**Returns** A float value representing the line rate in Mb/s.

---

### **MeasOscAmplitude**

Measures base-to-top amplitude.

**Call** float MeasOscAmplitude( )

**Parameters** NA

**Returns** A float value representing the base-to-top amplitude.

---

### **MeasOscRMS**

Measures AC RMS amplitude.

**Call** float MeasOscRMS( )

**Parameters** NA

**Returns** A float value representing the RMS amplitude.

---

### **MeasOscPeakToPeak**

Measures peak-to-peak amplitude.

**Call** float MeasOscPeakToPeak( )



**Parameters** NA

**Returns** A float value representing the peak-to-peak amplitude.

---

### **MeasOscMinLevel**

Measures minimum level.

**Call** float MeasOscMinLevel( )

**Parameters** NA

**Returns** A float value representing the minimum level.

---

### **MeasOscMaxLevel**

Measures maximum level.

**Call** float MeasOscMaxLevel( )

**Parameters** NA

**Returns** A float value representing the maximum level.

---

### **MeasOscBaseLevel**

Measures base level.

**Call** float MeasOscBaseLevel( )

**Parameters** NA

**Returns** A float value representing the base level.

---

### **MeasOscTopLevel**

Measures top level.

**Call** float MeasOscTopLevel( )

**Parameters** NA

**Returns** A float value representing the top level.

---

### **MeasOscPosOvershoot**

Measures the positive overshoot, in percent, of the acquired signal. Positive overshoot is defined as  $100 * (\text{max} - \text{top}) / \text{amplitude}$ .

**Call** float MeasOscPosOvershoot

**Parameters** NA

**Returns** A float value representing the positive overshoot.

---

### **MeasOscNegOvershoot**

Measures the negative overshoot, in percent, of the acquired signal. Negative overshoot is defined as  $100 * (\text{base} - \text{min}) / \text{amplitude}$ .

**Call** float MeasOscNegOvershoot

**Parameters** NA

**Returns** A float value representing the negative overshoot.

---

### **MeasNRZExtinctionRatio**

Measures the signal's extinction ratio within the NRZ window.

**Call** float MeasNRZExtinctionRatio

**Parameters** NA

**Returns** A float value representing the extinction ratio, expressed as a ratio.

---

### **MeasNRZExtinctionRatioDB**

Measures the signal's extinction ratio within the NRZ window in dB.

**Call** float MeasNRZExtinctionRatioDB

**Parameters** NA

**Returns** A float value representing the extinction ratio, expressed in dB.

---

### **MeasNRZExtinctionRatioHist**

Measures the signal's extinction ratio within the NRZ window and display a histogram of signal amplitude within the window.

**Call** float MeasNRZExtinctionRatioHist

**Parameters** NA

**Returns** A float value representing the extinction ratio, expressed as a ratio.

---

### **MeasNRZExtinctionRatioDBHist**

Measures the signal's extinction ratio within the NRZ window and display a histogram of signal amplitude within the window.

**Call** long Acquire( )

**Parameters** NA

**Returns** A float value representing the extinction ratio, expressed in dB.

---

### **MeasNRZEyeHeight**

Measures eye height.

**Call** float MeasNRZEyeHeight

**Parameters** NA

**Returns** A float value representing the eye height.

---

### **MeasNRZEyeAmplitude**

Measures eye amplitude.

**Call** float MeasNRZEyeAmplitude

**Parameters** NA

**Returns** A float value representing the eye amplitude.

---

### **MeasNRZEyeWidth**

Measures eye width.

**Call** float MeasNRZEyeWidth

**Parameters** NA

**Returns** A float value representing the eye width.

---

### **MeasNRZOMA**

Measures the OMA, which is the signal amplitude at the crossing, measured within the NRZ OMA window.

**Call** float MeasNRZOMA

**Parameters** NA

**Returns** A float value representing the optical modulation amplitude.

---

### **MeasNRZLowLevel**

Measures the average low level within the NRZ window.

**Call** float MeasNRZLowLevel

**Parameters** NA

**Returns** A float value representing the low level.

---

### **MeasNRZHighLevel**

Measures the average high level within the NRZ window.

**Call** float MeasNRZHighLevel

**Parameters** NA

**Returns** A float value representing the high level.

---

### **MeasNRZSNR**

Measures the signal-to-noise ratio (SNR) within the NRZ window.

**Call** float MeasNRZSNR

**Parameters** NA

**Returns** A float value representing the signal-to-noise ratio.

---

### **MeasNRZCrossingPercent**

Measures the level at the eye crossing.

**Call** float MeasNRZCrossingPercent( )

Measures the signal level at the eye crossing, expressed as a percentage of the signal amplitude.

**Parameters** NA

**Returns** A float value representing the eye crossing level in percent.

---

### MeasNRZRiseTime

Measures the signal rise time between the selected NRZ thresholds.

**Call** float MeasNRZRiseTime

**Parameters** NA

**Returns** A float value representing the rise time in picoseconds.

---

### MeasNRZFallTime

Measures the signal fall time between the selected NRZ thresholds.

**Call** float MeasNRZFallTime

**Parameters** NA

**Returns** A float value representing the fall time in picoseconds.

---

### MeasNRZDutyCycleDistortion

Measures the duty cycle distortion, that is, the asymmetry between the time the signal is in the high state and the time the signal is in the low state.

**Call** float MeasNRZDutyCycleDistortion( )

**Parameters** NA

**Returns** A float value representing the duty cycle distortion in percent.

---

### MeasNRZJitterPeakToPeak

Measures the peak-to-peak edge jitter at the eye crossing.

**Call** float MeasNRZJitterPeakToPeak( )

**Parameters** NA

**Returns** A float value representing the peak-to-peak jitter in picoseconds.

---

### **MeasNRZJitterRMS**

Measures the RMS edge jitter at the eye crossing.

**Call** float MeasNRZJitterRMS( )

**Parameters** NA

**Returns** A float value representing the RMS jitter in picoseconds.

---

### **MeasNRZSingleEdgeJitterPP**

Measures the peak-to-peak edge jitter for the selected edge at the eye crossing.

**Call** float MeasNRZJitterPeakToPeak(int nEdgeSelect )

**Parameters** An integer value that selects the edge to be measured:

0: falling edge

1: rising edge

**Returns** A float value representing the peak-to-peak jitter in picoseconds.

---

### **MeasNRZSingleEdgeJitterRMS**

Measures the RMS edge jitter for the selected edge at the eye crossing.

**Call** float MeasNRZJitterRMS(int nEdgeSelect )

**Parameters** An integer value that selects the edge to be measured:

0: falling edge

1: rising edge

**Returns** A float value representing the RMS jitter in picoseconds.

---

### **MeasNRZDeterministicJitterRise**

Measures the time difference between the average time position of a LLH transition and the average time position of a HLH transition.

**Call** float MeasNRZDeterministicJitterRise( )

**Parameters** NA

**Returns** A float value representing the time difference in picoseconds.

---

### MeasNRZDeterministicJitterFall

Measures the time difference between the average time position of a HHL transition and the average time position of a LHL transition.

**Call** float MeasNRZDeterministicJitterFall( )

**Parameters** NA

**Returns** A float value representing the time difference in picoseconds.

---

## Pattern Sequence Acquisition

- “LoadPatternAcqFilterFile” on page 9-65.
- “LoadPatternAcqFilterFileByValue” on page 9-65.
- “MeasPatternAcqDetJitterRise” on page 9-66.
- “MeasPatternAcqDetJitterFall” on page 9-67.
- “MeasPatternAcqStdDetJitterRise” on page 9-67.
- “MeasPatternAcqStdDetJitterFall” on page 9-68.
- “MeasPatternSpecificRiseTime” on page 9-68.

---

### LoadPatternAcqFilterFile

Loads a filter specification file for use with pattern sequence acquisition.

**Call** FG\_STATUS LoadPatternAcqFilterFile(BSTR \* FileString)

**Parameter** A character string that is part of the file name and which specifies the particular file to load.

Filter file names have the format DCAnnnnnnBTsssss.txt, where nnnnnn is the module’s serial number and sssss is a user-specified string that gives the file a unique name. The string parameter specifies this second field; the remainder of the file name is filled in automatically.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### LoadPatternAcqFilterFileByValue

Loads the specified filter file for use with pattern sequence acquisition.

This is an alternative form of LoadPatternAcqFilterFile, for applications that have difficulty passing the required BSTR pointer to LoadPatternAcqFilterFile.

**Call** FG\_STATUS LoadPatternAcqFilterFileByValue(char \* FileString)

**Parameter** A character string that is part of the file name and which specifies the particular file to load.

Filter file names have the format DCAnnnnnnBTsssss.txt, where nnnnnn is the module's serial number and sssss is a user-specified string that gives the file a unique name. The string parameter specifies this second field; the remainder of the file name is filled in automatically.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### **MeasPatternAcqDetJitterRise**

Measures the rising-edge deterministic (data-dependent) jitter between the specified bit sequences.

**Call** float MeasPatternAcqDetJitterRise(long nPatternFast, long nLengthFast, long nPatternSlow, long nLengthSlow)

- Parameters**
- 1: The bit sequence to be used for the “fast,” or “high-frequency” transition.
  - 2: The length (number of bits) of the “fast” sequence.
  - 3: The bit sequence to be used for the “slow,” or “low-frequency” transition.
  - 4: The length (number of bits) of the “slow” sequence.

Transitions may be up to 16 bits in length. The bit sequences must be right-aligned in the data word, for example, the bit sequence 101 is represented as a pattern value of 5 with length 3. Bits are matched left to right with the acquired data sequence.

**Returns** A value of type float representing the deterministic jitter in picoseconds.



---

**NOTE**

---

The functions `MeasPatternAcqDetJitterRise` and `MeasPatternAcqDetJitterFall` select specific transitions by marking other transitions as invalid. After these measurements, sample data returned by `GetDCASampleData` or `GetDCASampleDataVariant` will contain samples with type code 0x08 (invalid) for such transitions.

---

**MeasPatternAcqDetJitterFall**

Measures the falling-edge deterministic (data-dependent) jitter between the specified bit sequences. The functions `MeasPatternAcqDetJitterRise` and `MeasPatternAcqDetJitterFall`.

Select specific transitions by marking other transitions as invalid. After these measurements, sample data returned by `GetDCASampleData` or `GetDCASampleDataVariant` will contain samples with type code 0x08 (invalid) for such transitions.

**Call** `float MeasPatternAcqDetJitterFall(long nPatternFast, long nLengthFast, long nPatternSlow, long nLengthSlow)`

- Parameters**
- 1: The bit sequence to be used for the “fast,” or “high-frequency” transition.
  - 2: The length (number of bits) of the “fast” sequence.
  - 3: The bit sequence to be used for the “slow,” or “low-frequency” transition.
  - 4: The length (number of bits) of the “slow” sequence.

Transitions may be up to 16 bits in length. The bit sequences must be right-aligned in the data word. For example, the bit sequence 101 is represented as a pattern value of 5 with length 3. Bits are matched left to right with the acquired data sequence.

**Returns** A value of type `float` representing the deterministic jitter in picoseconds.

---

**MeasPatternAcqStdDetJitterRise**

Measures the maximum rising-edge deterministic (data-dependent) jitter between any bit sequences in the acquired pattern. This function selects specific transitions by marking other transitions as invalid. After this measurement, sample data returned by [“GetDCASampleData” on page 9-51](#), [“GetDCASampleDataVariant” on page 9-52](#) will contain samples with type code 0x08 (invalid) for such transitions.

**Call** `float MeasPatternAcqStdDetJitterRise`

**Parameters** None

**Returns** A value of type float representing the deterministic jitter in picoseconds.

---

### **MeasPatternAcqStdDetJitterFall**

Measures the maximum falling-edge deterministic (data-dependent) jitter between any bit sequences in the acquired pattern. This function selects specific transitions by marking other transitions as invalid. After this measurement, sample data returned by "[“GetDCASampleData” on page 9-51](#)," "[“GetDCASampleDataVariant” on page 9-52](#) will contain samples with type code 0x08 (invalid) for such transitions.

**Call** float MeasPatternAcqStdDetJitterFall

**Parameters** None

**Returns** A value of type float representing the deterministic jitter in picoseconds.

---

### **MeasPatternSpecificRiseTime**

Measures the rise time of transitions described by the length and pattern parameters.

**Call** float MeasPatternSpecificRiseTime(long nLength, long nPattern).

- Parameter** 1: The length (number of bits) of the sequence  
 2: The bit state sequence, as a right-justified integer

Transitions may be up to 16 bits in length. The bit sequences must be right-aligned in the data word, for example, the bit sequence 101 is represented as a pattern value of 5 with length 3. Bits are matched left to right with the acquired data sequence.

For a valid rise time measurement, the transition specified must end with the states 01. Invalid transition specifiers will return a value of 0 and will set the last error flag.

**Returns** A value of type float representing the rise time in picoseconds.

---

### **MeasPatternSpecificFallTime**

Measures the rise time of transitions described by the length and pattern parameters.

**Call** float MeasPatternSpecificFallTime(long nLength, long nPattern).

- Parameter** 1: The length (number of bits) of the sequence  
2: The bit state sequence, as a right-justified integer

Transitions may be up to 16 bits in length. The bit sequences must be right-aligned in the data word, for example, the bit sequence 101 is represented as a pattern value of 5 with length 3. Bits are matched left to right with the acquired data sequence.

For a valid rise time measurement, the transition specified must end with the states 01. Invalid transition specifiers will return a value of 0 and will set the last error flag.

**Returns** A value of type float representing the rise time in picoseconds.

---

### GetPatternAcqData

Gets the acquired pattern data sequence as a string of binary or hex digits.

**Call** BSTR GetPatternAcqData(int nFormatSel)

---

### NOTE

The data sequence is aligned so that the longest string of consecutive identical bits (1s or 0s) comes first. If the longest string of 1s and the longest string of 0s are the same length, the string of 0s comes first. If there is no unique longest substring, the sequence begins with one of the longest substrings, chosen randomly; in this case, successive acquisitions may not begin with the same substring.

---

**Parameter** An integer value that specifies the data format, as follows:

- 0: Binary data  
1: Hex data

**Returns** A string containing the data as digits in the selected data format.

---

## Calibration

- [“DarkCal” on page 9-69.](#)
- [“DelayCalStart” on page 9-70.](#)
- [“DelayCalGetState” on page 9-70.](#)

---

### DarkCal

Measures and saves the optical dark current level.

---

**Call** void DarkCal( )

**Parameters** NA

---

### **DelayCalStart**

Start or stop delay calibration for the currently selected input and filter.

**Call** void DelayCalStart(int nFunc)

**Parameters** An integer value that specifies the Parameter values:

- 0: Stop delay calibration
- 1: Start delay calibration

---

### **DelayCalGetState**

Return the state of the delay calibration operation in progress.

**Call** long DelayCalGetState( )

**Parameters** NA

**Returns** 0: Delay calibration is not running (has finished).  
1: Delay calibration is running.

---

## **Control Display**

- “ClearDisplay” on page 9-70.
- “CopyGraphToClipboard” on page 9-71.
- “CopyGraphToFile” on page 9-71.
- “DisplayMode” on page 9-72.
- “EyeDisplayColorSel” on page 9-72.
- “EyeDisplayTraceSel” on page 9-72.
- “EyeDisplayPixelSizeSel” on page 9-73.
- “EyeDisplayDeferredUpdate” on page 9-73.
- “EyeDisplayCursorX0 EyeDisplayCursorX1 EyeDisplayCursorY0 EyeDisplayCursorY1” on page 9-73.
- “JpegQuality” on page 9-74.

---

### **ClearDisplay**

Clear the eye window display.

**Call** void ClearDisplay( )

**Parameters** NA

---

### **CopyGraphToClipboard**

Place a bitmap copy of the eye display window image on the Windows clipboard.

**Call** long CopyGraphToClipboard(BSTR szCaption )

**Parameters** A character string specifying a caption for the image. If the parameter value is NULL or the caption string is empty, no caption is included.

**Returns** 0: The operation succeeded.  
-1: The operation failed because the eye window is not displayed.  
-2: The operation failed because of insufficient memory or system resources.  
-3: The operation failed because the Windows clipboard was not accessible.

---

### **CopyGraphToFile**

Copy the image on the display window to an image file.

**Call** long CopyGraphToFile(int nFileType, BSTR szFilePath, BSTR szCaption )

**Parameters**

- An integer value that specifies the file format:
  - 0: Bitmap (.bmp) file format
  - 1: JPEG (.jpg) file format
- A character string specifying the file name and path.
- A character string specifying a caption for the image. If the parameter value is NULL or the caption string is empty, no caption is included.

---

### **NOTE**

Portions of this software make use of the CxImage image-processing library, [www.codeproject.com/bitmap/cximage.asp](http://www.codeproject.com/bitmap/cximage.asp).

**Returns** 0: The operation succeeded.  
-1: The operation failed because the eye window is not displayed.  
-2: The operation failed because of insufficient memory or system resources.

- 4: The operation failed because the file format specifier is invalid.
- 5: An internal error occurred.
- 6: An internal error occurred.
- 7: An error occurred saving the file.

---

### **DisplayMode**

Selects whether the ActiveX control displays the eye window or is invisible, and, if visible, selects the information displayed.

**Parameters** An integer value that specifies the display mode.

- 0: Window is not visible; acquired sample data are not transferred to the host.
- 1: Display the acquired data as an eye diagram.
- 2: Display the eye with cursors indicating the measurement results.
- 3: Display the eye, cursors, and the measured values as text.
- 4: Window is not visible; acquired sample data are transferred to the host and may be read using the GetDCASampleData and GetDCASampleDataVariant methods.

---

### **EyeDisplayColorSel**

Selects display color mode.

**Parameters** An integer value that specifies the color mode.

- 0. Disable trace coloring.
- 1. Display traces in four different colors based on only the current transition (LL, LH, HL, HH).
- 2. Display traces in eight different colors based on the current and previous transitions (LLL, LLH, LHL, LHH, HLL, HLH, HHL, HHH).

---

### **EyeDisplayTraceSel**

Enables the selective display of different transition types.

**Parameters** An integer value whose bits enable display of the different transition types.

- 1 LLL
- 2 LLH
- 4 HHL

8 HHH  
16 HLL  
32 HLH  
64 LHL  
128 LHH

Traces are displayed whose corresponding bits are set. Note that a parameter value of 0 enables the display of all traces.

---

### **EyeDisplayPixelSizeSel**

Selects pixel size used for drawing the eye display.

**Parameters** An integer value that specifies the pixel size.

0 Small pixels (1 x 1)  
1 Large pixels (2 x 2)

---

### **EyeDisplayDeferredUpdate**

Select whether the display is updated incrementally, as points are read from the DCA module, or only after all points have been read.

**Parameters** An integer value that specifies the update mode.

Zero (0) updates incrementally.

One (1) updates in a single action, after all points have been read.

---

### **EyeDisplayCursorX0** **EyeDisplayCursorX1** **EyeDisplayCursorY0** **EyeDisplayCursorY1**

Measurement cursor position values.

Read-only properties that return the positions of the cursors that show measurement results. The values returned are floats; the units are unit intervals (UI) for the X cursors, uW or mV for the Y cursors. The values returned pertain to the most recently performed measurement; for cursors that are not displayed, the value 0 is returned.

**Parameters** NA

---

### **JpegQuality**

Sets the image quality and file compression for JPEG image files created by the method, [“CopyGraphToFile” on page 9-71](#).

**Parameters** An integer value in the range 0 to 100. Greater values result in higher-quality images and larger image files.

---

#### **NOTE**

Portions of this software make use of the CxImage image-processing library, [www.codeproject.com/bitmap/cximage.asp](http://www.codeproject.com/bitmap/cximage.asp).

---

---

### **Status**

---

#### **GetLastError**

Returns the result of the last operation performed.

**Call** long GetLastError( )

**Parameters** NA

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.



---

## ActiveX API Properties

- “DCA Configuration” on page 9-75.
- “Pattern Sequence Acquisition” on page 9-78.

---

### DCA Configuration

- “LineRate” on page 9-75.
- “Wavelength” on page 9-75.
- “OptAttenRatio” on page 9-76.
- “ElectAttenRatio” on page 9-76.
- “RefClockSel” on page 9-76.
- “RefClockFreq” on page 9-77.
- “WanderCorrection” on page 9-77.
- “DCASampleBlockCount” on page 9-77.
- “DCADisplayBlockCount” on page 9-77.
- “DCAInputPathSelect” on page 9-78.
- “NRZThresholdSel” on page 9-78.
- “NRZWindowSize” on page 9-78.
- “NRZOMASize” on page 9-78.

---

#### LineRate

Specifies the line data rate.

**Type** Value of type double that specifies the rate in MHz.

---

#### Wavelength

Configures the DCA for the specified optical input wavelength.

**Type** Enum value of type FG\_WAVELENGTH\_TYPE that specifies the wavelength.

---

### **OptAttenRatio**

Informs the DCA of an external attenuator in the optical input path. DCA measurements will use this value so that the measured values reflect the signal at the input of this attenuator.

**Type** Value of type float that specifies the ratio by which the attenuator reduces the optical power. For example, a 3 dB optical attenuator reduces the power by a factor of 2:1; it would be entered as an attenuation ratio of 2.0

Attenuation specified in dB may be converted to power ratio according to the following equation:

$$\text{Ratio} = 10^{\text{dB}/10}$$

The power ratio may be converted to dB according to the following equation:

$$\text{dB} = 10 \times \log_{10}(\text{ratio})$$

For example, a 3 dB optical attenuator reduces the power by a factor of 2:1; it would be entered as an attenuation ratio of 2.0.

If no attenuator is used, this value should be set to 1.0.

---

### **ElectAttenRatio**

Specifies an external attenuator in the electrical input path. DCA measurements will use this value so that the measured values reflect the signal at the input of this attenuator.

**Type** Value of type float that specifies the ratio by which the attenuator reduces the input voltage. For example, a 6 dB electrical attenuator reduces the voltage by a factor of 2:1; it would be entered as an attenuation ratio of 2.0.

Attenuation specified in dB may be converted to voltage ratio according to the following equation:

$$\text{Ratio} = 10^{\text{dB}/20}$$

The power ratio may be converted to dB according to the following equation:

$$\text{dB} = 20 \times \log_{10}(\text{ratio})$$

For example, a 6 dB electrical attenuator reduces the voltage by a factor of 2:1; it would be entered as an attenuation ratio of 2.0.

If no attenuator is used, this value should be set to 1.0.

---

### **RefClockSel**

Specifies the timing source (reference clock) for the DCA's data acquisition.

**Type** Enum value of type FG\_REFCLKSLT that specifies the timing source.

---

### RefClockFreq

Specifies the clock frequency to be used when the reference clock source selection is FG\_REFCLK\_EXTERNAL.

**Type** Value of type double that specifies the frequency in MHz.

---

### RecoveredClockRatio

Specifies the ratio of the line rate to the recovered clock rate when the reference clock source selection is FG\_REFCLK\_RECOVERED\_RATIO. See a note on page 9-41 for further details.

**Call** FG\_STATUS FGcfgSetRecoveredClockRatioAPI(FG\_HANDLE hModule, double dFreq)

#### Parameters

- The module connection handle returned by fg\_connect.
- A value of type float that specifies the ratio.

**Returns** A status value of type FG\_STATUS reflecting the result of the operation.

---

### WanderCorrection

Enables or disables correction for wander of the DCA's time base during data acquisition. (Wander correction is always used when acquiring data for mask tests, or when making edge jitter measurements.)

**Type** Value of type unsigned long that controls wander correction.

0: Disable wander correction.

1: Enable wander correction.

---

### DCASampleBlockCount

Specifies the number of 1024-sample blocks to be taken when the DCA acquires data.

**Type** Integer value that specifies the number of 1024-sample blocks.

---

### DCADisplayBlockCount

Specifies the number of sample blocks to be displayed, and available for transfer to the user's buffer. A value of 0 specifies "all."

**Type** Integer value

---

**DCAInputPathSelect**

Selects the optical or electrical input and the input filter, if applicable. For proper DCA operation, the input not selected must have no signal applied.

**Parameters** Enum value of type FG\_FILTER\_TYPE that selects the input and filter. The available selections are returned by the queries, [“GetModuleInformation” on page 9-42](#) for input configuration and available filters.

---

**NRZThresholdSel**

Selects the low and high amplitude threshold values used for NRZ measurements.

**Parameters** A value of type FG\_HILOW\_VLTREF that specifies the thresholds.

---

**NRZWindowSize**

Sets the start and end of the time window, in percent of 1 UI, over which NRZ measurements are made. The window is positioned in the center of the eye.

**Parameters** An integer value between 2 and 98 that specifies the window size.

---

**NRZOMASize**

Sets the start and end of the time window, in percent of 1 UI, over which NRZ OMA measurements are made. The window is centered on the eye crossing.

**Parameters** An integer value between 2 and 98 that specifies the window size.

---

## Pattern Sequence Acquisition

- [“PatternAcqLength” on page 9-78.](#)
- [“PatternAcqFilterSel” on page 9-79.](#)
- [“PatternAcqDisplayBits” on page 9-79.](#)
- [“PatternAcqFileLoaded \(read-only\)” on page 9-79.](#)

---

**PatternAcqLength**

Enables pattern sequence acquisition and specifies the sequence length.

**Parameters** An integer value.

- 0. Disable pattern sequence acquisition.
  - $4 \leq N \leq 2047$ : Acquire data with the specified sequence length.
- 

### **PatternAcqFilterSel**

Enables or disables the software filtering.

**Parameters** An integer value.

- 0: Unfiltered
  - 1: Bessel-Thompson filter
- 

### **PatternAcqDisplayBits**

Selects the number of bits of the reconstructed pattern sequence data to be displayed.

**Parameters** An integer value.

- 0: Display all bits of the reconstructed pattern sequence.
  - $> 0$ : Display the specified number of bits.
- 

### **PatternAcqFileLoaded (read-only)**

Returns the status of the most recent filter file load operation.

**Parameters** An integer value.

- 0: The specified file was not found or no file has been loaded.
- 1: The specified file was successfully loaded.



---

## Specifications

---

# Specifications

The distinction between specifications and characteristics is described as follows:

- Specifications describe warranted performance. All specifications apply after the instrument is turned on for one (1) hour and while self-calibration is valid. Many performance parameters are enhanced through frequent, simple user calibrations.
- *Characteristics* provide useful information by giving functional, but nonwarranted, performance parameters. *Characteristics are printed in italics.*

**Table 10-1. General Specifications**

<b>Sample rate</b>	160 Ms/s
<b>Points per sample block</b>	1024
<b>Maximum number of sample blocks</b>	1024
<b>Clock recovery</b>	< 2.7 Gb/s
<b>Pattern acquisition maximum length</b>	2047 bits
<b>Fixed number of points per bit in pattern acquisition mode</b>	128

**Table 10-2. Display Specifications: Optical Vertical Scale Factor Per Division**

<b>Minimum vertical scale factor</b>	1 $\mu$ W
<b>Maximum vertical scale factor</b>	500 $\mu$ W
<b>Number of vertical divisions</b>	8



**Table 10-3. Display Specifications: Electrical Vertical Scale Factor Per Division**

<b>Minimum vertical scale factor</b>	1 mV/div
<b>Maximum vertical scale factor</b>	100 mV/div
<b>Maximum number of divisions</b>	8
<b>Vertical scale settings</b>	1, 2, 5, 10, 20, 50, 100, 200, and 450

**Table 10-4. Display Specifications: Horizontal Specifications**

<b>Minimum X-scale setting</b>	1 ps/div
<b>Maximum X-scale setting</b>	20 ns/div
<b>Number of divisions (fixed)</b>	10

**Table 10-5. Electrical Specifications**

<b>Number of channels</b>	1 single ended
<b>BW of electrical input</b>	9.5 GHz ( <i>characteristic</i> )
<b>Transition time (10% to 90%, calculated from <math>TR = 0.35/BW</math>)</b>	37 ps ( <i>characteristic</i> )
<b>Connection type</b>	AC Coupled
<b>AC input voltage range</b>	1 V pp (Max)
<b>Connector type</b>	SMA
<b>Nominal impedance</b>	50 ohm ( <i>characteristic</i> )
<b>Absolute maximum input voltage without damage</b>	2 V pp
<b>Line rate coverage</b>	155 Mb/s to 11.318 Gb/s <sup>a</sup>

**Table 10-5. Electrical Specifications**

<b>Electrical return loss</b>	<i>–12 dB (Return Loss on electrical path when an optical signal is applied) (characteristic)</i>
-------------------------------	---

a. Maximum line rate based on an input signal with 100 mVpp minimum signal.

**Table 10-6. Clock Specifications**

<b>Clock recovery</b>	<i>155 Mb/s to 2.7 Gb/s (characteristic)</i>
<b>Clock input frequency range</b>	<i>10 MHz to 11.318 GHz (characteristic)</i>
<b>Clock input impedance</b>	50 ohm
<b>Reflection</b>	<i>10% for 100 ps RT (characteristic)</i>
<b>Maximum clock input signal</b>	2 Vpp
<b>Clock input voltage range</b>	0.5 to 1 V pp

**Table 10-7. Optical Specifications**

<b>Number of optical channels</b>	1
<b>Unfiltered Optical BW</b>	<i>7.5 GHz (characteristic)</i> Unfiltered option only available if selected.
<b>Optical connector</b>	FC/PC
<b>Fiber input</b>	62/125 $\mu$ m
<b>Optical input return loss</b>	<i>850 nm &gt; 13 dB (characteristic) 1550 nm &gt; 24 dB (characteristic)</i>
<b>Maximum non-destruct average</b>	–3 dBm at 1310 nm
<b>Maximum non-destruct peak</b>	+7 dBm at 1310 nm
<b>Average power monitor (specified operating range)</b>	–30 dBm to –2 dBm at 850 nm

Table 10-7. Optical Specifications

Wavelength range	750 to 1650 nm
Number of optical filters	4
Available filter choices	1.062 Gb/s, 1.250 Gb/s, 2.125 Gb/s, 4.25 Gb/s, 5.0 Gb/s, 6.25 Gb/s, 8.5 Gb/s, and unfiltered
Optical channel RMS noise at 850 nm	6.5 $\mu$ W (filtered) <sup>a</sup>
Optical channel RMS noise at 1310/1550 nm	4.5 $\mu$ W (filtered) <sup>a</sup>
Average optical power meter accuracy (single-mode)	5% + uncertainty of reference source + connector uncertainty + 200 nW <sup>b</sup>
Average optical power meter accuracy (multi-mode)	10% + uncertainty of reference source + connector uncertainty + 200 nW (characteristic) <sup>b</sup>
Line rate coverage	155 Mb/s to 11.318 Gb/s <sup>c</sup>
Calibrated wavelengths	850 nm 1310 nm, 1550 nm (Characteristic)

- a. The measurement was made with no input signal; this test condition returns the worst case optical noise level.
- b. For -2 dBm to -25 dBm (at 850 nm).
- c. Maximum line rate based on an input signal with 200  $\mu$ W minimum OMA.

Table 10-8. Environmental Specifications

Use	Indoor
Power consumption	40 W
Dimensions	Width: Four-slot PXI module Height: 3U high PXI module

Specifications  
**Specifications**

## A

- Acquiring Data, 3–10
- Active-X, 9–2
- ActiveX, 9–4
- air flow, 2–3
- Amplitude, 4–9
- Annotations, 5–13
- API, 9–2
- Average Optical Power, 4–15

## B

- Base, 4–9
- Bit Rate, 4–27

## C

- Calc Stats, 4–45
- cleaning
  - fiber-optic connections, 1–13
  - non-lensed connectors, 1–13
- Color Intensity Display, 5–6
- Configuration Settings, 3–4
- Connect, 2–5, 3–3
- Control Panel, 3–2
- Copy Another Mask, 4–40
- Count, 3–10
- Crossing Hist., 4–35
- Crossing Histogram Window, 5–5
- Crossing Percentage, 4–30
- custom user masks, 4–40

## D

- DCA Parameters, 3–6
- Deterministic Jitter, 4–34
- dimensions, 10–5
- Disconnect, 2–8
- Display Background, 5–5
- Display Colors, 5–3
- Display Traces, 5–4
- DLL, 9–3
- Duty Cycle Distortion, 4–30

## E

- Edge Jitter, 4–32
- electrostatic discharge, 1–5
- Environmental Specifications, 10–5
- Extinction Ratio, 4–17

- Extinction Ratio with histogram, 4–20
- Eye Amplitude, 4–21
- Eye Diagram, 4–3
- Eye Height, 4–20
- Eye Mask Test Mode Measurements, 4–37
- Eye Width, 4–22
- eye-diagram analysis, 4–2

## F

- Failing Points, 4–38
- Fall Time, 4–29
- Filter File Format, 6–6
- Filter Files, 6–6
- Find Max Margin, 4–39
- Find Max. Margin, 4–43
- Firmware, 2–9
- floor mat, 1–5

## G

- Global Parameters, 3–5

## H

- heel strap, 1–5
- High, 4–25

## I

- ID String, 2–5, 3–3
- industry standard masks, 4–39
- Inspect, 2–2
- instrument
  - returning for service, 1–14
- ISI Histogram, 4–35

## L

- Load DCA File, 8–3
- Low, 4–24

## M

- Margin Compatibility option, 7–5
- Markers, 4–7
- Mask Definitions, 4–41
- Mask Display, 5–3
- Mask Margin, 7–3

- Mask Margins Rules, Agilent, 7–6
- Mask Margins Rules, Tektronix, 7–8
- Mask Test mode, 4–37
- Max, 4–13
- Min, 4–12
- Module Config, 2–5
- Multiple Measurements Mode, 4–44

## N

- Negative Overshoot, 4–14
- NRZ Measurement Mode, 4–16
- NRZ Thresholds, 3–7
- NRZ Window Size, 3–8

## O

- OMA, 4–23
- OMA Window Size, 3–8
- Optical modulation amplitude, 4–23
- Oscilloscope Measurement mode, 4–6
- Oscilloscope Mode Measurements, 4–5

## P

- Pass/Fail, 4–42
- Pattern Acquisition, 3–7, 4–49, 6–2
- Pattern Sequence D. J. Mode Measurements, 4–46
- Pattern Sequence D.J. mode, 4–46
- PC, 2–3
- Peak to Peak, 4–10
- Positive Overshoot, 4–13
- Presets, 3–9
- Program Settings, 5–3

## Q

- quick confidence check, 2–5

## R

- remote bridge, 2–3
- Repetitive, 3–10
- Resolution, 5–6
- returning to Agilent, 1–14
- Rise Time, 4–27
- RMS, 4–11

## Index

### S

Sample Density Color Scheme, 5–6  
Sample Density Display Information,  
5–7  
Scan for Instruments, 2–5, 3–3  
Select Mask, 4–38, 4–39  
service, 1–14  
Set Defaults, 3–9  
Set IP Addr, 3–9  
shipping  
    procedure, 1–14  
Show Version, 3–9  
Show Y axis scale, 5–6  
Signal to Noise Ratio, 4–26  
Simulator data file format, 8–7  
Single Edge Jitter, 4–33  
software digital filters, 6–5  
Specifications, 10–2, 10–4, 10–5  
Start Acquisition, 2–6, 3–10, 4–38  
static-safe accessories, 1–6  
Statistics, 4–45

### T

table mat, 1–5

### U

Units, 5–5  
User-written Applications, 8–7

### W

wrist strap, 1–5

### X

X Axis Control, 4–6

### Y

Y Axis Control, 4–7  
Y-Scale, 5–5