TECHNICAL USER MANUAL

Gocator Measurement Tools

Algorithms and GDK development

Document revision: F

Copyright

Proprietary

Trademarks and Restrictions

Contact Information

LMI Technologies, Inc.
9200 Glenlyon Parkway
Burnaby BC V5J 5J8
Canada

Telephone: +1 604-636-1011
Fax: +1 604-516-8368

www.lmi3D.com

Table of Contents

Introduction

Several of LMI's measurement tools use complex algorithms to find features and then return measurements and decisions. This document describes the algorithms used by the following profile tools (only available on Gocator displacement sensors and line profile sensors):

- Panel
- Groove
- Strip

This document also describes the algorithms used by the following surface tools (only available on Gocator line profile sensors and snapshot sensors):

- Hole
- Stud
- Opening

This manual also provides information for getting started with the Gocator Development Kit (GDK).

Notational Conventions
This documentation uses the following notational conventions:

	Consider this information in order to make best use of the product.

Built-in Measurement Tools

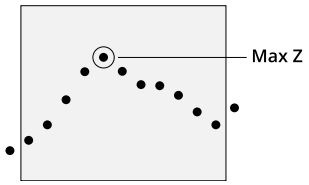
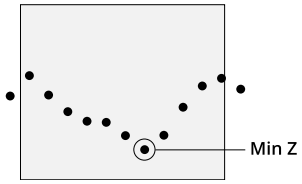
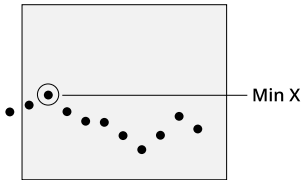
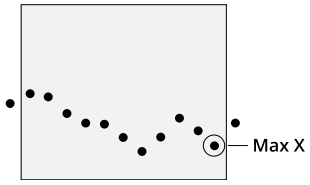
The following sections describe the algorithms and parameters used by some of Gocator's profile and surface measurement tools.

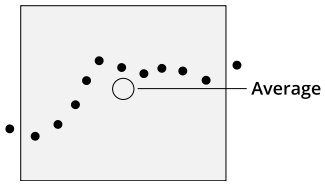
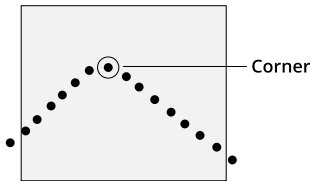
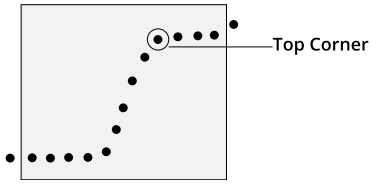
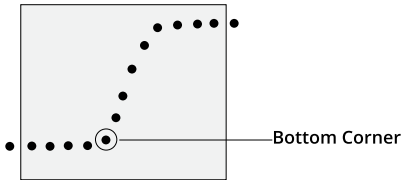
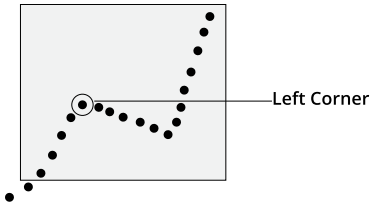
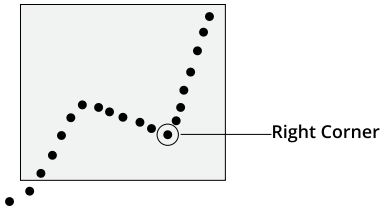
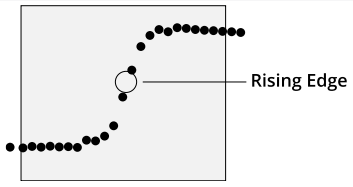
Profile Tools

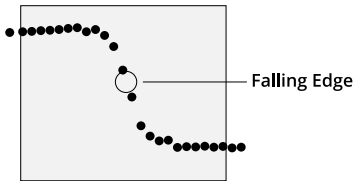
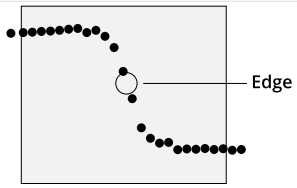
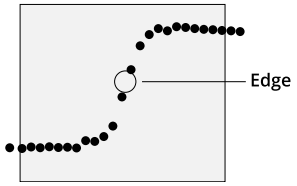
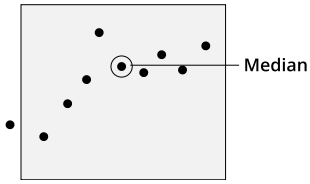
Feature Points

Most measurements detect and compare *feature points* or *lines* found within laser profile data. Measurement *values* are compared against minimum and maximum thresholds to yield *decisions*.

The following types of points can be identified.

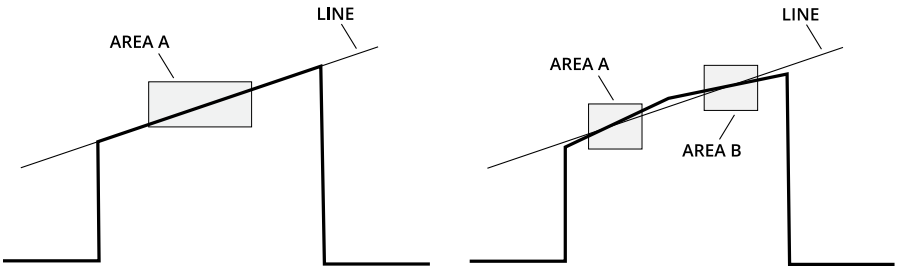
Point Type	Examples
Max Z Finds the point with the maximum Z value in the region of interest.	
Min Z Finds the point with the minimum Z value in the region of interest.	
Min X Finds the point with the minimum X value in the region of interest.	
Max X Finds the point with the maximum X value in the region of interest.	

Point Type	Examples
Average Determines the average location of points in the region of interest.	
Corner Finds a dominant corner in the region of interest, where corner is defined as a change in profile slope.	
Top Corner Finds the top-most corner in the region of interest, where corner is defined as a change in profile shape.	
Bottom Corner Finds the bottom-most corner in the region of interest, where corner is defined as a change in profile shape.	
Left Corner Finds the left-most corner in the region of interest, where corner is defined as a change in profile shape.	
Right Corner Finds the right-most corner in the region of interest, where corner is defined as a change in profile shape.	
Rising Edge Finds a rising edge in the region of interest.	

Point Type	Examples
Falling Edge Finds a falling edge in the region of interest.	
Any Edge Finds a rising or falling edge in the region of interest.	 
Median Determines the median location of points in the region of interest.	

Fit Lines

Some measurements involve estimating lines in order to measure angles or intersection points. A fit line can be calculated using data from either one or two fit areas.

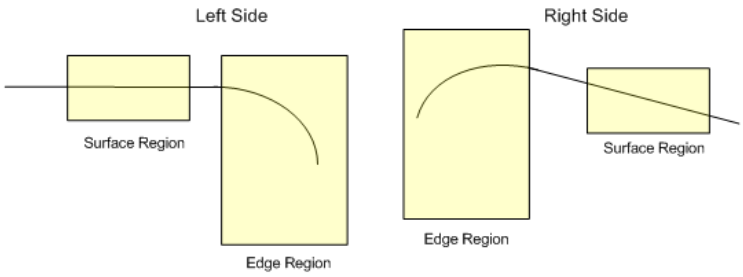


A line can be defined using one or two areas. Two areas can be used to bypass discontinuity in a line segment.

Panel (Gap and Flush) and Round Corner Algorithm

The Panel measurement tool uses the same algorithm to find a feature using either the Gap or the Flush measurement. The Round Corner tool uses the same algorithm, but applies it only to the left or the right; you must choose the side in the tool.

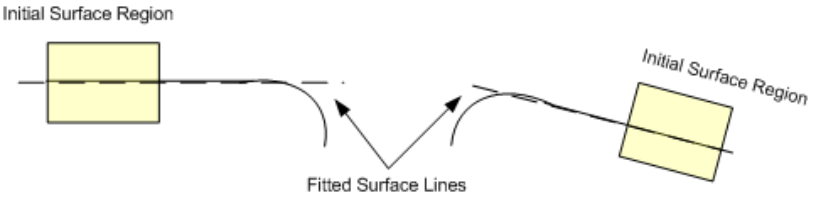
This algorithm first searches for two regions on a side: a surface region and an edge region. (See the tables below for the parameters used by the algorithm.)



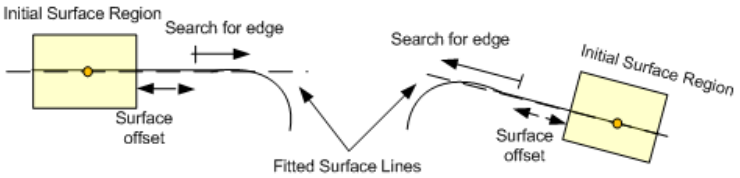
After the algorithm finds the regions, it places a feature point on the surface region based on a set of parameters. You can control the measurement regions, which contain the surface and the edge regions, for the left and the right side. A measurement region also defines the region in which the measurement tool will search for the feature points. Feature points are located on a side using the following algorithm.

1. On the left side, search from left to right to find a surface region with data that covers at least the value specified in the **Surface Width** setting. For the right side, do the same, searching from right to left.

2. If a surface region is found, fit a line, called the surface line, using the data within the area.

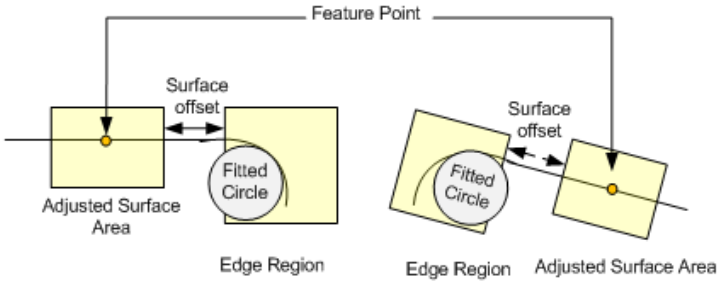


3. Search for a valid edge region that is located at least the distance specified in the **Surface Offset** setting from the end of the surface region. If a surface region is not found, move along the search direction and repeat step 1.

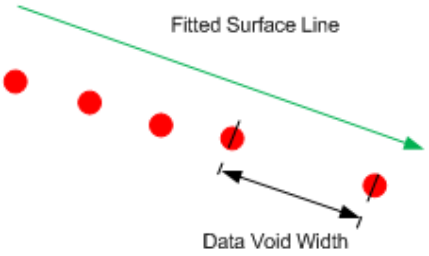
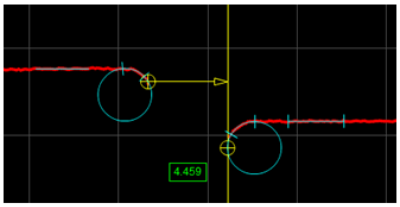
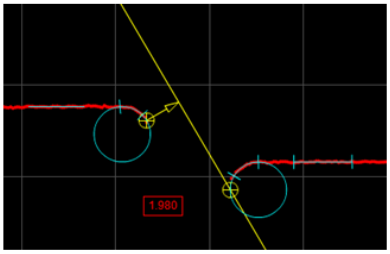
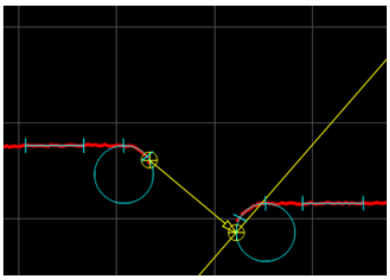


   A valid edge region is detected when an edge matches the value in the **Nominal Radius** setting or when the depth exceeds the value in the **Min Depth** setting. The search algorithm uses the **Max Void Width** setting to distinguish between an actual edge and an area of missing data.

4. If a valid edge region is detected, a model fit is applied to the surface and edge regions to accurately determine the region positions and feature point locations. The model fit takes into account the **Surface Width**, **Surface Offset**, **Edge Angle** and the **Edge Type** parameters.

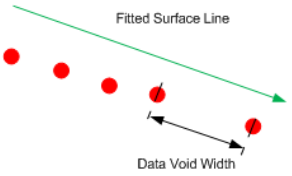
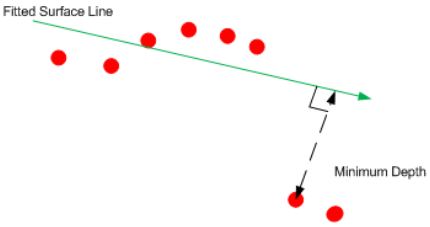
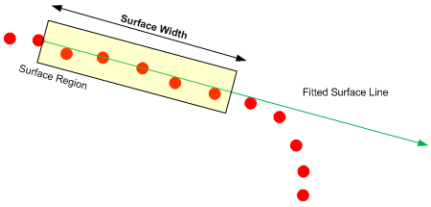
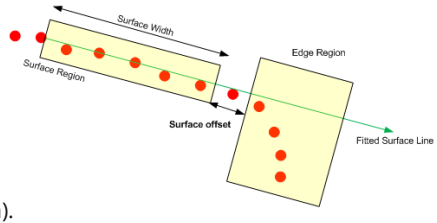
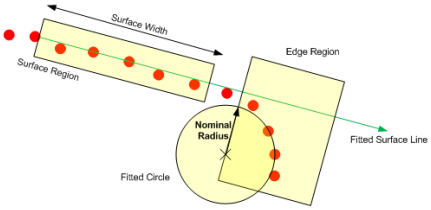


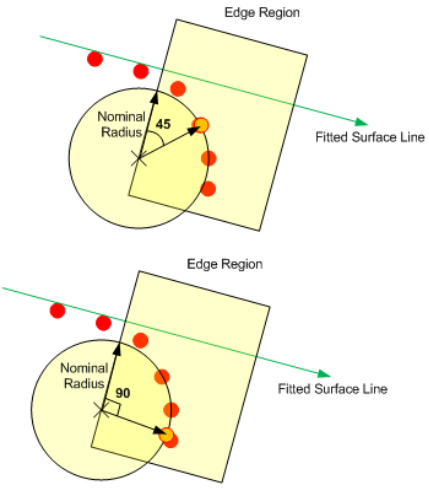
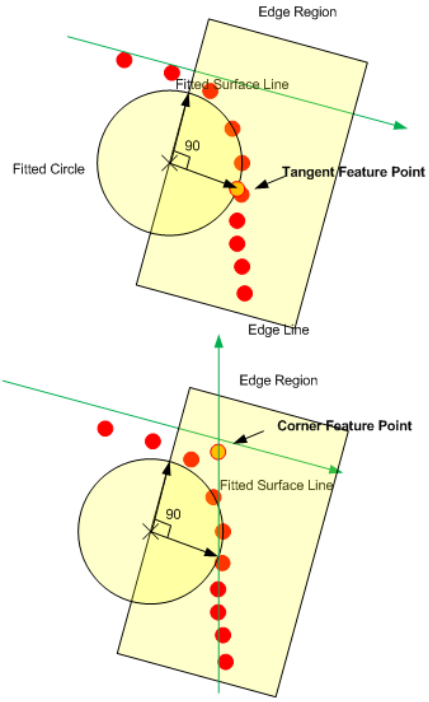
Parameters

Parameter	Description	Illustration
Max Gap Width	The maximum width of the gap. Allows the tool to filter gaps greater than the expected width. This can be used to single out the correct gap when there are multiple gaps in the field of view.	
Reference Side Direction	Defines the side used to calculate the measurement axis (see below) rounded corner.	
Measurement Axis *Panel Gap measurement only*	Defines the direction that the gap is calculated, in relation to the reference side (see above). **Surface**: In the direction of the fitted surface line of the reference surface. **Edge**: In the direction perpendicular to the edge of the reference surface. **Distance**: The Cartesian distance between the two feature locations.	**Surface Axis**  **Edge Axis**  **Distance Axis** 

Parameter	Description	Illustration
Absolute *Panel Flush measurement only*	When enabled, returns an absolute value rather than a signed value.	
Decision	See *Decisions* on page 27.	
Region	See *Regions* on page 29.	
Output	See *Filters* on page 28.	

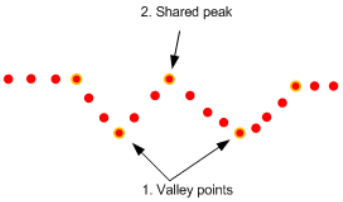
Left/Right SideEdge Parameters

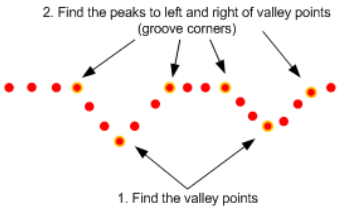
Parameter	Description	Illustration
Max Void Width	The maximum allowed width of missing data caused by occlusion or data dropout. A larger value prevents the algorithm from registering a section of missing data as an edge. Setting the value to 0 causes the algorithm to try to detect an edge in every missing data section.	
Min Depth	Defines the minimum depth before an opening could be considered to have a potential edge. The depth is the perpendicular distance from the fitted surface line.	
Surface Width	The width of the surface area in which laser data is used to form the fitted surface line. This value should be as large as the surface allows.	
Surface Offset	The distance between the edge region and the surface region. Setting a small value allows the edge within a tighter region to be detected. However, the measurement repeatability could be affected if the data from the edge are considered as part of the surface region (or vice versa). A rule of thumb is to set **Surface Offset** equal to **Nominal Radius**.	
Nominal Radius	The radius of the curve edge that the tool uses to locate the edge region. The algorithm searches for a start position in which the remaining data most resemble a circle of the specified nominal radius.	

Parameter	Description	Illustration
Edge Angle	A point on the best fit circle to be used to calculate the feature point. The selected point is on the circumference at the specified angle from the start of the edge region. The angle is measured from the axis perpendicular to the fitted surface line.	
Edge Type	Defines the type of feature point to use for the edge (Corner or Tangent). A tangent edge point is the point selected based on the defined Edge Angle. A corner edge point is the intersect point between the fitted surface line and a edge line formed by interpolating the points at and after the tangent within the edge region.	

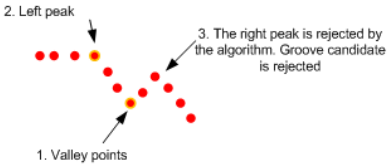
Groove Algorithm

The Groove measurement tool first locates valley along the profile line. The bottom point of a valley, the valley point, is the first estimation of the position of the groove bottom. For each valley, the algorithm searches for corner to the left and to the right to find the groove corners. A groove candidate is found when the groove corners are located on the left and right before the next valley is reached. Two groove candidates may share the same corner as shown in the right image below. (See the tables below for the parameters used by the algorithm.)

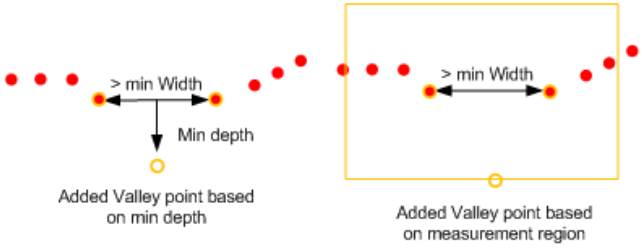




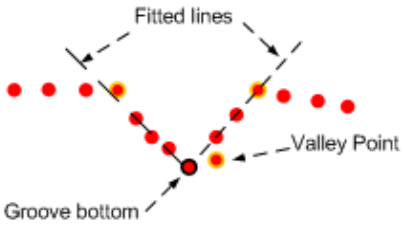
The algorithm derives search parameters from the user settings to prevent noise from triggering false detections. When detecting multiple grooves, an adaptive algorithm is used to ensure that candidate grooves are approximately the same scale.



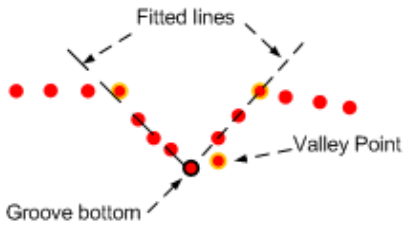
The valley points of open grooves may not be visible or may fall outside of the measurement region. Voids in the data (regions with no profile data) between pairs of valid points are detected. A valley point is added midway between the pair of valid points. The Z position of the valley point is either the minimum groove depth below the lower of the corners or the bottom edge of the measurement region. The algorithm then proceeds as if to find a U-shaped groove.



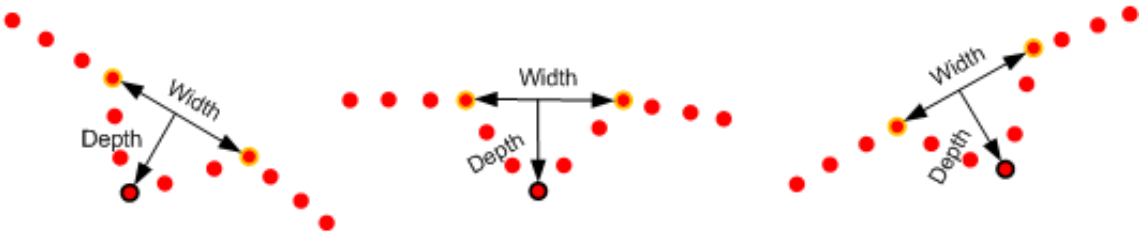
The actual groove bottom is calculated differently for different shapes. For a V-shaped groove, a line is fitted to the sides of the valley points starting from the corners, up to (but not including) the valley point. The groove bottom is the intersection of the left and right lines. Line fitting is used such that an accurate groove bottom can be found even when the real bottom is not visible (i.e., blocked by reflections).



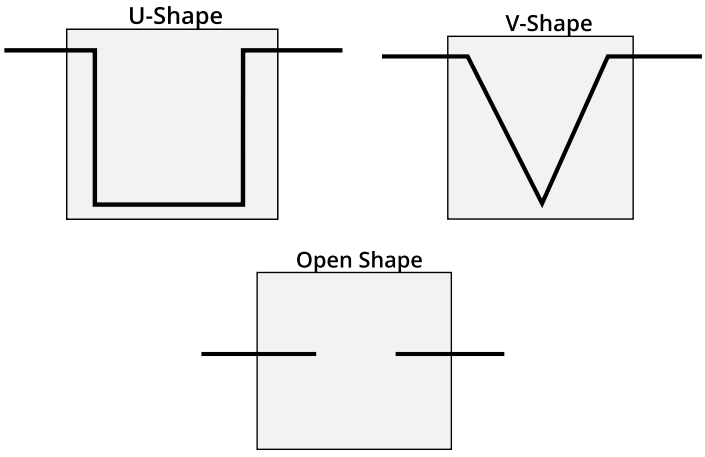
For U-shaped and open groove, the distance from each point within the groove (including the added point for open-shaped groove) is projected onto the width line. The groove bottom's X is at the centroid of the projected values along the width. The groove bottom's Z is at the maximum depth of the groove.

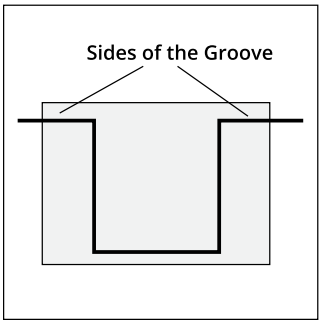


Groove candidates that do not meet the minimum and maximum width and depth settings are rejected. The width and depth measurements are invariant to the groove rotation. The width is the distance between the groove corners and the depth is perpendicular distance of the groove bottom from the groove width.



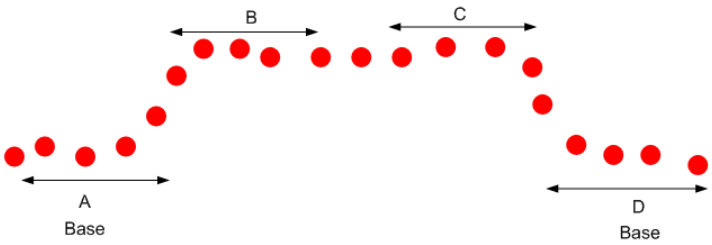
Parameters

Parameter	Description
Shape	Shape of the groove 

Parameter	Description
Location *(Groove X and Groove Z measurements only)*	Specifies the location type to return Bottom - Groove bottom. For a U-shape and open-shape groove, the X position is at the centroid of the groove. For a V-shape groove, the X position is at the intersection of lines fitted to the left and right sides of the groove. See algorithm section below for more details. Left - Groove's left corner. Right - Groove's right corner.
Select Type	Specifies how a groove is selected when there are multiple grooves within the measurement area. Maximum Depth - Groove with maximum depth. Index from The Left - 0-based groove index, counting from left to right Index from the Right - 0-based groove index, counting from right to left.
Index	0-based groove index.
Minimum Depth	Minimum depth for a groove to be considered valid.
Minimum Width	Minimum width for a groove to be considered valid. The width is the distance between the groove corners.
Maximum Width	Maximum width of a groove to be considered valid. If set to 0, the maximum is set to the width of the measurement area.
Decision	See *Decisions* on page 27.
Region	The measurement region defines the region in which to search for the groove. For a stable measurement, the measurement region should be made large enough to cover some laser data on the left and right sides of the groove. See *Regions* on page 29. 
Filters	See *Filters* on page 28.

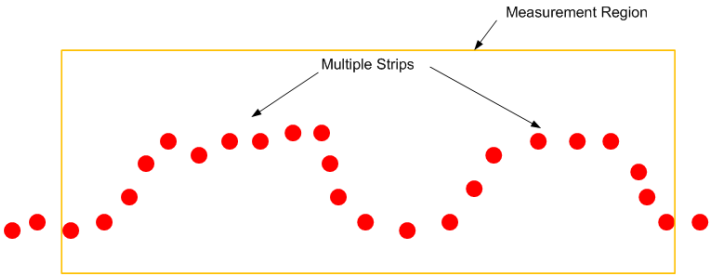
Strip Algorithm

A strip is a flat region bounded on the left and on the right by edges. The Strip tool can measure the edge positions, width, and height of a strip. The Strip tool assumes that regions outside the strip, referred to as the base regions (Region A and D below), deviate in height from the start and end parts of a strip (Region B and C). (See the tables below for the parameters used by the algorithm.)

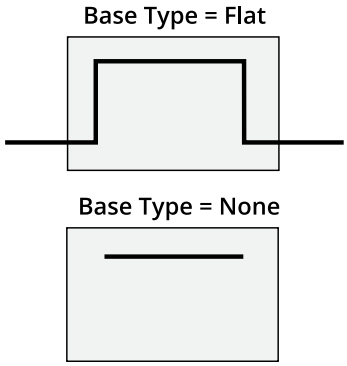


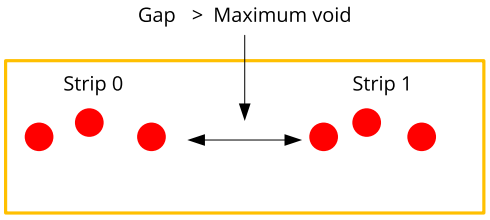
When the target is sitting on the surface, the base is lower than the strip (as shown above). Alternatively for a groove the base is above the strip surface. The base could be missing when the target is hanging in the air or the surface holding the target falls outside the sensor's active area. You can control the base type in the measurement panel.

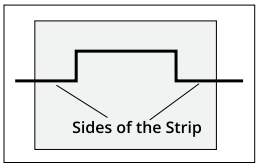
The Strip tool can detect multiple strips. You can select a region of interest, referred to as the measurement region, from which the algorithm search for multiple strips.



Parameters

Parameter	Description
Base Type	Affects detection of rising and falling edges.  When **Base Type** is set to **Flat**, both strip (raised area) and base support regions are needed. When set to **None**, only a point that deviates from a smooth strip support region is needed to find a rising or falling edge.
Location *(Strip Height, Strip X, and Strip Z measurements only)*	Specifies the strip position from which the measurements are performed. **Left** - Left edge of the strip. **Right** - Right edge of the strip. **Center** - Center of the strip.

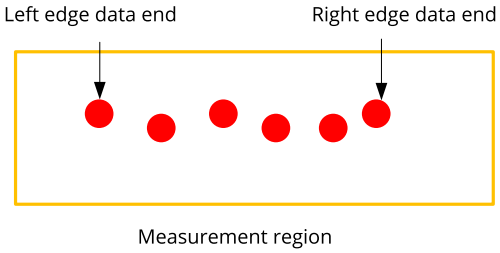
Parameter	Description
Left Edge Right Edge	Specifies the features that will be considered as the strip's left and right edges. You can select more than one condition. **Rising** - Rising edge detected based on the strip edge parameters. **Falling** - Falling edge detected based on the strip edge parameters. **Data end** - First valid profile data point in the measurement region. **Void** - Gap in the data that is larger than the maximum void threshold. Gaps connected to the measurement region's boundary are not considered as a void. See "Strip Start and Terminate Conditions" in the *Gocator Measurement Tool Technical Manual* for the definitions of these conditions.
Select Type	Specifies how a strip is selected when there are multiple strips within the measurement area. **Best** - The widest strip. **Index Left** - 0-based strip index, counting from left to right. **Index Right** - 0-based strip index, counting from right to left.
Index	0-based strip index.
Min Height	Specifies the minimum deviation from the strip base. See "Strip Step Edge Definitions" in the *Gocator Measurement Tool Technical Manual* on how this parameter is used for different base types.
Support Width	Specifies the width of the region around the edges from which the data is used to calculate the step change. See "Strip Step Edge Definitions" in the *Gocator Measurement Tool Technical Manual* on how this parameter is used by different base types.
Transition Width	Specifies the nominal width needed to make the transition from the base to the strip. See "Strip Step Edge Definitions" in the *Gocator Measurement Tool Technical Manual* on how this parameter is used by different base types.
Max Void Width	The maximum width of missing data allowed for the data to be considered as part of a strip when **Void** is selected in the **Left** or **Right** parameter. This value must be smaller than the edge **Support Width**.  Measurement region When occlusion and exposure causes data drops, users should use the gap filling function to fill the gaps.
Min Width	Specifies the minimum width for a strip to be considered valid.
Tilt Enabled	Enables/disables tile correction.
Decision	See *Decisions* on page 27.

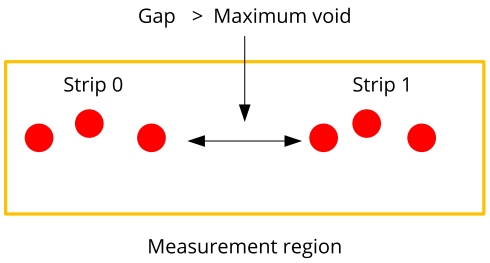
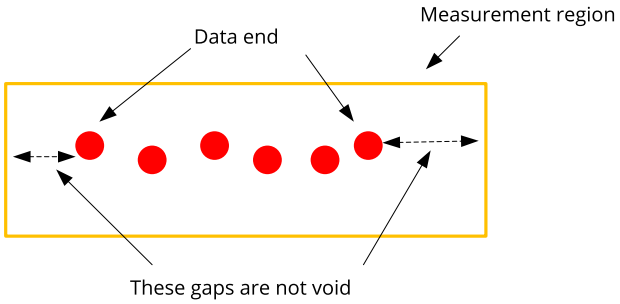
Parameter	Description
Region	The measurement region defines the region in which to search for the strip. If possible, the region should be made large enough to cover the base on the left and right sides of the strip.  See *Regions* on page 29 for more information.
Output	See *Filters* on page 28.

Strip Start and Terminate Conditions

The Strip tool allows you to define how a strip starts and ends. The Left Edge parameter controls how a strip starts and the Right Edge parameter controls how a strip ends.

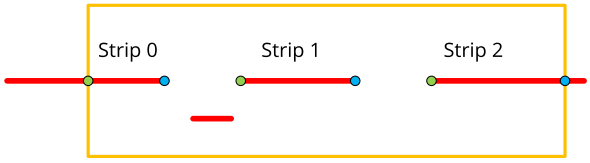
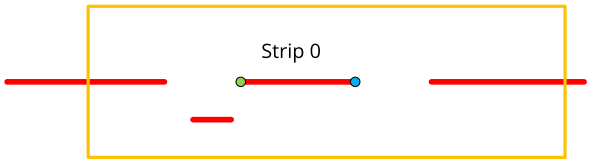
Start / terminate conditions

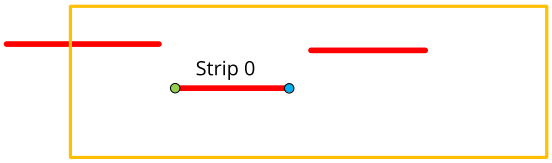
Condition	Description
Rising	Rising step edge detected based on the strip edge parameters. See *Strip Step Edge Definitions* on page 19 for details on how the step edge is detected.
Falling	Falling step edge detected based on the strip edge parameters. See *Strip Step Edge Definitions* on page 19 for details on how the step edge is measured.
Data end	The first (for the left edge) or the last (for the right edge) valid profile data point in the measurement region. 

Condition	Description
Void	Gaps in the data that are larger than the maximum void threshold.  Gaps at the ends of the measurement region's boundary are not considered as a void. 

The following examples show how the parameters affect the strip detection in different scenarios.

Left and Right Edge conditions

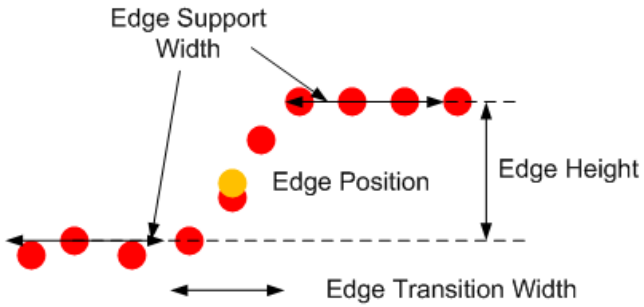
Condition	Example
Left: Rising, data end, void Right: Falling, data end, void	
Left: Rising, void Right: Falling, void	
Left: Rising Right: Data end, void	
Left: Data end, void Right: Falling	

Condition	Example
Left: Falling Right: Rising	

Strip Step Edge Definitions

The Strip tool detects step edges based on the parameters Base Type, Edge Transition Width, Edge Support Width, and Minimum Edge Height.

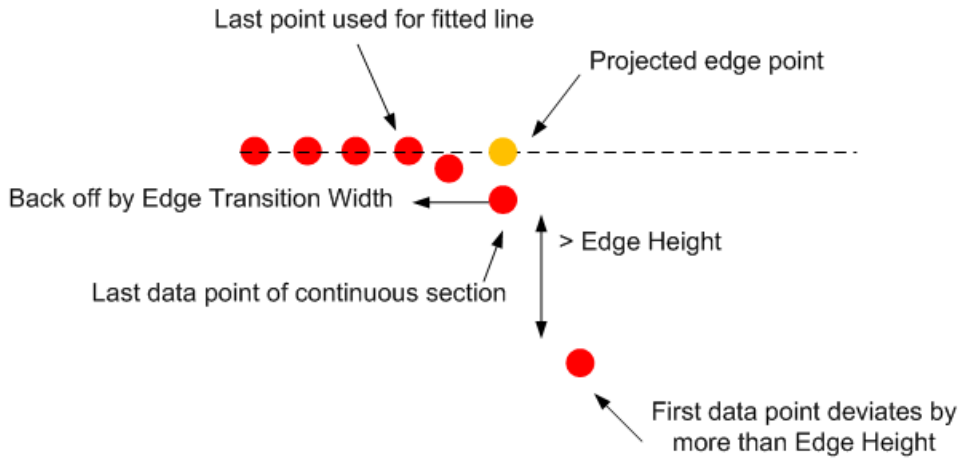
When Base Type is set to Flat, the regions around the edges are visible and the edge positions are between the base and the strip surface.



The Minimum Edge Height parameter defines the size of the step edge. The Edge Transition Width parameter specifies the nominal width of the transition, from the base to the strip surface.

The Edge Support Width parameter defines the width of the region around the edges from which the data is used to measure the step change. To improve noise immunity, the height level of the Edge Support Width parameter is calculated by averaging the data within the region.

When the base is set to None, the tool looks for continuous sections that are wider than the Edge Support Width parameter and have no data points that deviate positively or negatively more than the value of the Minimum Edge Height parameter. The data in the strip support region (the raised area) must be smooth. The height level of the continuous region is calculated based on the fitted line as shown below.



The algorithm then backs off by the value of the Edge Transition Width parameter and uses the data up to the back-off point to create the fitted line and projects the edge point on the line. This step prevents the points near the end of a rounded strip from affecting the height of the strip.

Surface Tools

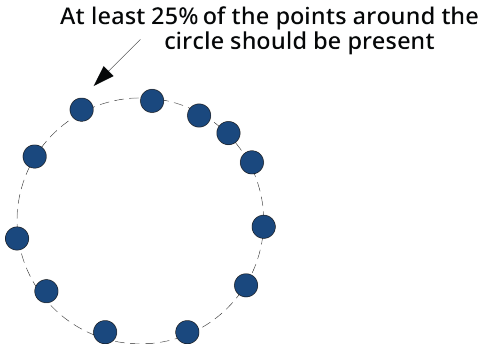
Hole Algorithm

The Hole tool processes the data in three phases: Search, Measure, and Filter. The algorithm can separate out background information that appears inside the hole. It can also detect holes that only partially appear in the data.

See the tables below for the parameters used by the algorithm.

Search phase - The tool searches for coarse data transitions (edge data) and performs a coarse fitting of the hole model (specified by the orientation angles and the nominal value) to determine the most likely candidate. If **Tilt Correction** is set to **Autoset**, the algorithm uses the data within the measurement region to estimate the orientation of the part.

Measure phase - A more rigorous edge detection algorithm is applied to precisely determine the edges around the feature. Edge detection at this stage will reject outliers and noise. The algorithm requires at least 25% of the data around the hole for the candidate to remain valid.

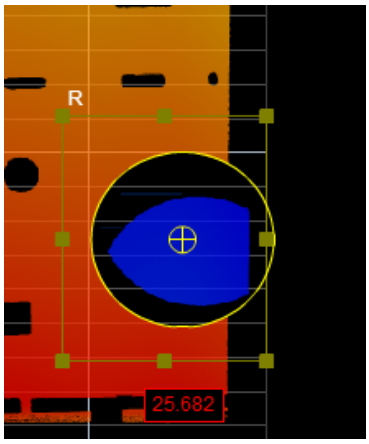


The accuracy of the algorithm improves when the points are spread more evenly along the hole's circumference.

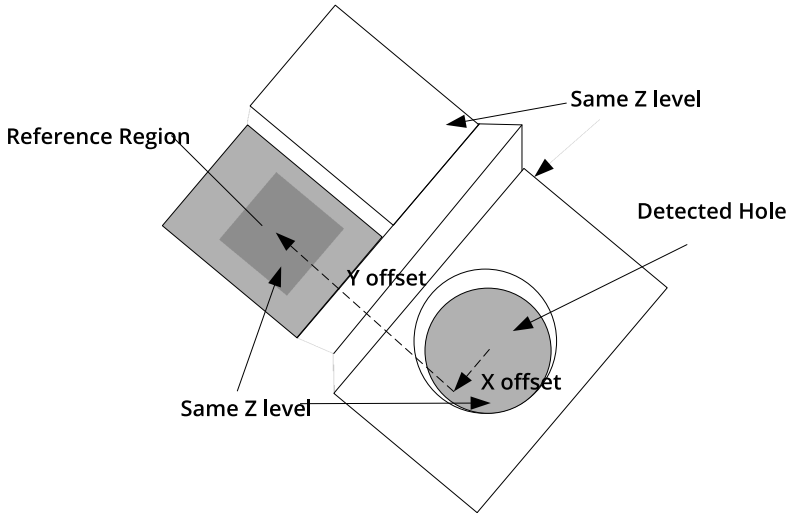
The set of refined edges is then used to locate and inspect the feature. If the Reference Regions option is enabled and set to AutoSet, the edges are also used to calculate the location of the reference regions.

Filter phase - The detected location and dimensions are then compared to the nominal and tolerance settings. If the refined feature falls within the measurement region and its measurements fit within the specified tolerance, the results are reported. If not, an invalid result is returned.

Parameters

Parameter	Description
Nominal Radius	Expected radius of the hole.
Radius Tolerance	The maximum variation from the nominal radius (+/- from the nominal radius).
Partial Detection	Enable if only part of the hole is within the measurement region. If disabled, the hole must be completely in the region of interest for results to be valid. 
Depth Limit	Data below this limit (relative to the surface) is excluded from the hole opening calculations.

Parameter	Description
Reference Regions	The algorithm uses the Reference Regions option to calculate the Z position of the hole. It is typically used in cases where the surface around the hole is not flat.



When this option is set to Autoset, the algorithm automatically determines the reference region. When the option is not set to Autoset, the user manually specifies the reference region. The location of the reference region is relative to the detected center of the hole and positioned on the nominal surface plane.

When the Reference Regions option is disabled, the tool measures the hole's Z position using all the data in the measurement region, except for a bounding rectangular region around the hole.

Tilt Correction	Tilt of the target with respect to the alignment plane. When this option is set to Autoset, the tool automatically detects the tilt. Otherwise, the user must enter the angles manually. Autoset requires the measurement region to cover more areas on the surface plane than other planes. The results from the Plane X and Y tool can be used for angles X and Y parameters.
Decision	See *Decisions* on page 27.
Region	See *Regions* on page 29.
Output	See *Filters* on page 28.

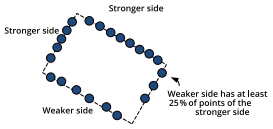
Opening Algorithm

The Opening tool processes the data in three phases: Search, Measure, and Filter.

See the tables below for the parameters used by the algorithm.

Search phase - The tool searches for coarse data transitions (edge data) and performs a coarse fitting of the opening shape (specified by the orientation angles and the nominal dimensions) to determine the most likely candidate. If **Tilt Correction** is enabled, the algorithm uses the flat surface in the measurement region to estimate the orientation of the part.

Measure phase - A more rigorous edge detection algorithm is applied to precisely determine the edges around the feature. Edge detection at this stage will reject outliers and noise. The algorithm requires opposite sides and ends to be associated with a comparable number of edge pixels, with the weaker side or end having at least 25% of the stronger.

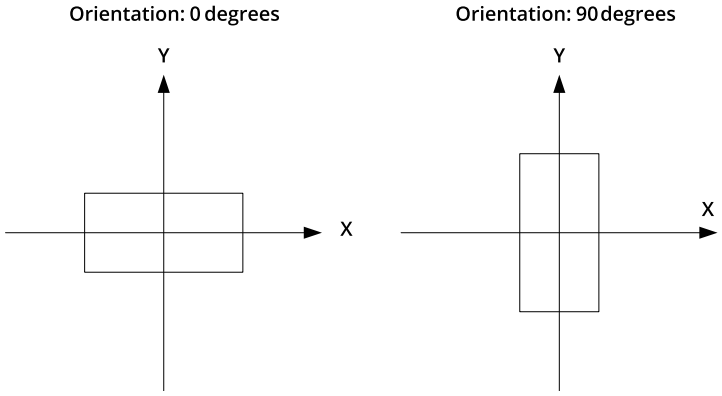


The set of refined edges is then used to locate and inspect the feature. If the **Reference Regions** setting is enabled, the edges are also used to calculate the location of the reference regions.

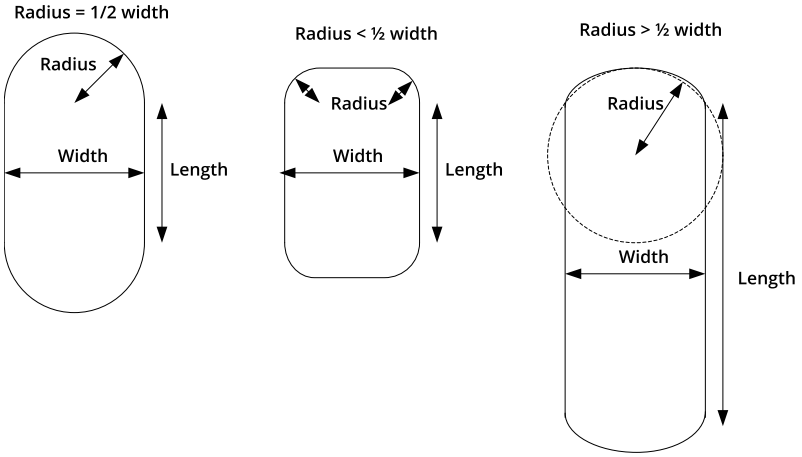
Filter phase - The detected location and dimensions are compared to the nominal and tolerance settings. If the refined feature falls within the measurement region and its measurements fit within the specified tolerance, the results are reported. If not, an invalid result is returned.

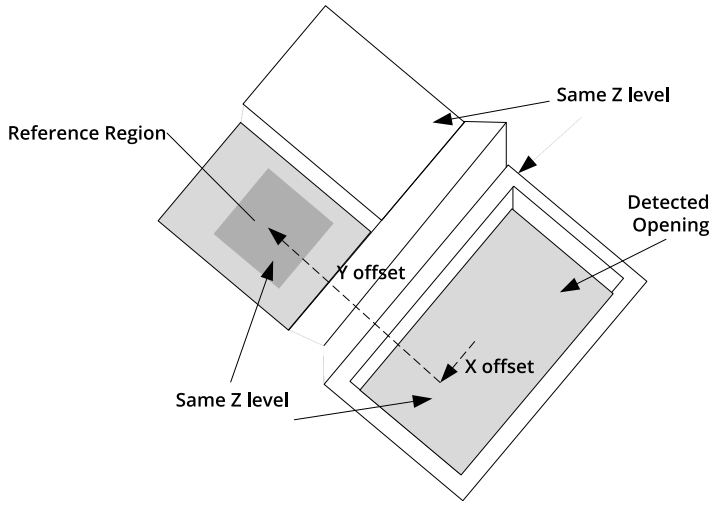
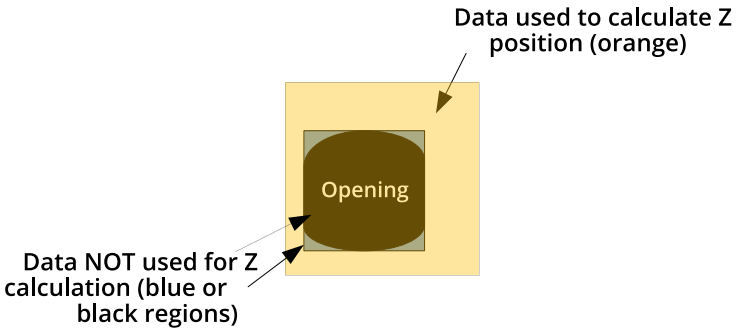
Parameters

Parameter	Description
Type	Rounded Slot, Rectangle.
Nominal Width	Nominal width of the opening.
Nominal length	Nominal length of the opening.
Nominal Angle	Nominal angle of the opening. The default orientation is the length of the opening along the X axis.



The diagram above illustrates the case where the surface is not tilted. When the surface is tilted, the orientation is defined with respect to the normal of the surface, not with respect to the X-Y plane

Parameter	Description
Nominal Radius	Nominal radius of the opening ends. If the opening type is set to rectangular, the radius setting is disabled. The opening has an oval shape if the radius is equal to ½ of the width. The opening is a rounded rectangle when the radius is less than ½ of the width. 
Width Tolerance	The maximum variation from the nominal width (+/- from the nominal value).
Length Tolerance	The maximum variation from the nominal length (+/- from the nominal value).
Angle Tolerance	The maximum variation from the nominal orientation (+/- from the nominal value).
Partial Detection	Enable if only part of the opening is within the measurement region. If disabled, the opening must be completely in the region of interest for results to be valid.
Depth Limit	Data below this limit (relative to the surface) is excluded from the hole opening calculations.

Parameter	Description
Reference Regions	The algorithm uses reference regions to calculate the Z position of the hole opening. Reference regions are relative to the center location of the feature. This option is typically used in cases where the surface around the opening is not flat.  When the Reference Regions setting is disabled, the tool measures the hole's opening's Z position using the all data in the measurement region, except for a bounding rectangular region around the opening.  With one or more reference region, the algorithm calculates the Z positions as the average values of the data within the regions. When the user places the reference region manually, all of the data is used, whether the data is inside or outside the opening. The user should place the reference region carefully.
Tilt Correction	Tilt of the target with respect to the alignment plane. Set to Auto-Set to have the tool automatically detect the target's tilt, or enter the angles manually. Auto-Set requires the measurement region to cover more areas on the surface plane than other planes. The results from the Plane X and Y tool can be used for angles X and Y parameters.
Decision	See *Decisions* on page 27.
Region	See *Regions* on page 29.
Output	See *Filters* on page 28.

Stud Algorithm

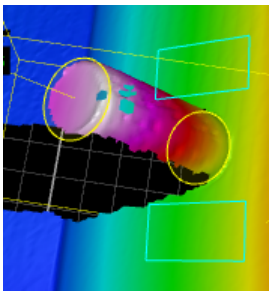
The Stud algorithm measures the stud in three steps: searching for the tip, finding the reference plane, and shaft fitting.

See the tables below for the parameters used by the algorithm.

Searching for the tip - The algorithm looks for the approximate location of the tip. If Auto-Tilt is enabled, the algorithm uses the flat surface around the tip to estimate the orientations of the part. The approximate tip is the location of the highest (maximum Z) pixel after correction for the nominal tilt angle.

Finding the reference plane - The reference regions are positioned using the approximate tip, the nominal angle values, and the nominal stud length. Compared to the hole/opening, misplaced stud reference regions are more likely to cause a failure to produce any measurement.

Shaft fitting - The shaft region is determined based on the approximate tip position, the nominal angles, the reference plane position, and the stud nominal size parameters. Shaft fitting is successful if the algorithm can fit at least three circles with the stud diameter along the shaft. Fitting each circle requires sufficient data along the top portion the shaft. Because of occlusions, the bottom of the shaft is often not visible to the sensor and the algorithm is designed to handle this situation.



Parameters

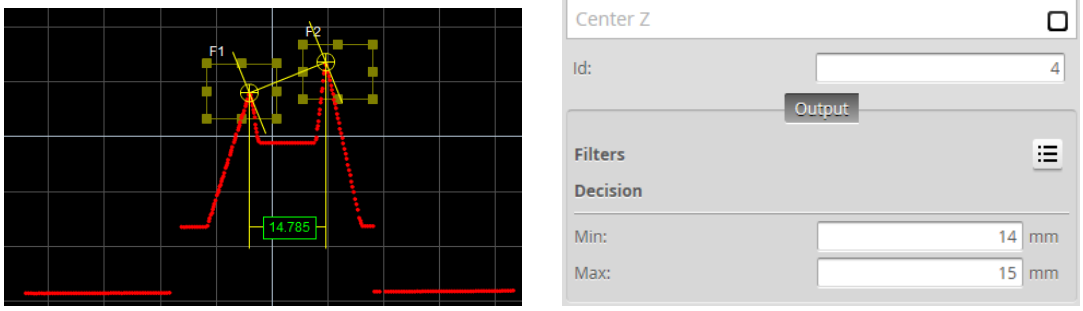
Parameter	Description
Nominal Stud Radius	Expected radius of the stud.
Nominal Stud Length	Expected length of the stud.
Base Height	The height above the base surface that will be ignored when the (truncated) cone is fit to the stud data.
Tip Height	The height from the top of the surface that will be ignored when the (truncated) cone is fit to the stud data.
Radius Offset *(Radius measurement only)*	The distance from the tip of the stud from which the radius is measured.
Reference Regions	The algorithm uses reference regions to calculate the base plane of the stud. Reference regions are relative to the base of the stud.
Tilt Correction	Tilt of the target with respect to the alignment plane. Set to Auto-Set to have the tool automatically detect the tilt, or enter the angles manually. Auto-Set requires the measurement region to cover more areas on the surface plane than other planes. The results from the Plane X and Y tool can be used for angles X and Y parameters.
Decision	See *Decisions* on the next page.
Region	See *Regions* on page 29.
Output	See *Filters* on page 28.

Common Parameters

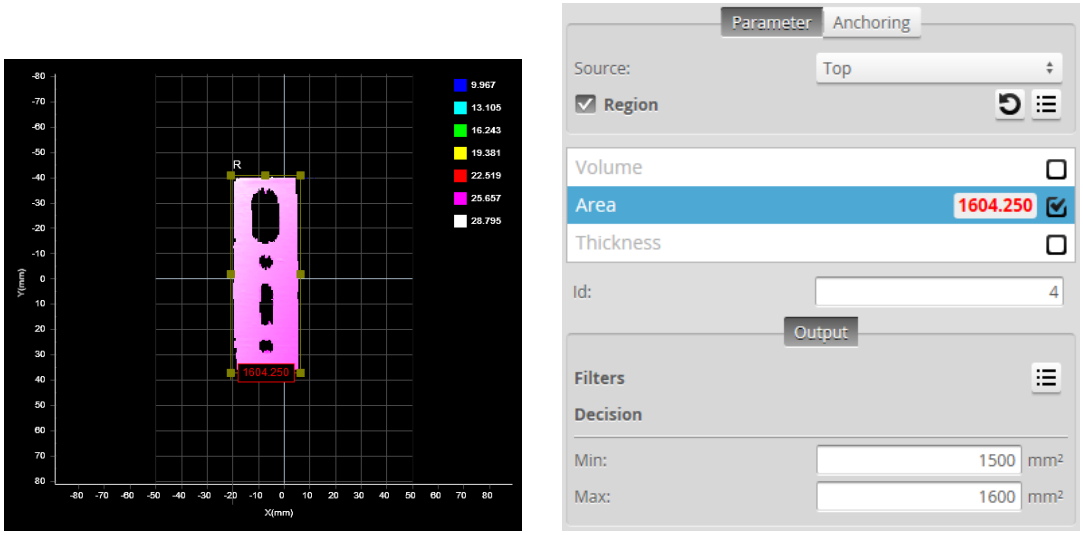
Decisions

Results from a measurement can be compared against minimum and maximum thresholds to generate *pass* / *fail* decisions. The decision state is *pass* if a measurement value is between the minimum and maximum threshold. In the data viewer and next to the measurement, these values are displayed in green. Otherwise, the decision state is *fail*. In the user interface, these values are displayed in red.

All measurements provide decision settings under the **Output** tab.



Value (14.786) within decision thresholds (Min: 14, Max: 15). Decision: Pass



Value (1604.250) outside decision thresholds (Min: 1500, Max: 1600). Decision: Fail

Along with measurement values, decisions can be sent to external programs and devices. In particular, decisions are often used with digital outputs to trigger an external event in response to a measurement.

To configure decisions:

1. Go to the **Measure** page by clicking on the **Measure** icon.

   > The scan mode must be set to the type of measurement you need to configure. Otherwise, the wrong tools, or no tools, will be listed on the **Measure** page.

2. In the **Tools** panel, click on a tool in the tool list.
3. In the measurement list, select a measurement.
   To select a measurement, it must be enabled.
4. Click on the **Output** tab.
   For some measurements, only the **Output** tab is displayed.
5. Enter values in the **Min** and **Max** fields.

Filters

Filters can be applied to measurement values before they are output from the Gocator sensors.

All measurements provide filter settings under the **Output** tab. The following settings are available.

Filter	Description
Scale and Offset	The Scale and Offset settings are applied to the measurement value according to the following formula: Scale * Value + Offset **Scale** and **Offset** can be used to transform the output without the need to write a script. For example, to convert the measurement value from millimeters to thousands of an inch, set **Scale** to 39.37. To convert from radius to diameter, set **Scale** to 2.
Hold Last Valid	Holds the last valid value when the measurement is invalid. Measurement is invalid if there is no valid value.
Smoothing	Applies moving window averaging to reduce random noise in a measurement output. The averaging window is configured in number of frames. If **Hold Last Valid** is enabled, smoothing uses the output of the **Hold Last Valid** filter.

To configure the filters:

1. Go to the **Measure** page by clicking on the **Measure** icon.

   > The scan mode must be set to the type of measurement you need to configure. Otherwise, the wrong tools, or no tools, will be listed on the **Measure** page.

2. In the **Tools** panel, click on a tool in the tool list.
3. In the measurement list, select a measurement.
   To select a measurement, it must be enabled.
4. Click on the **Output** tab.
   For some measurements, only the **Output** tab is displayed.
5. Expand the **Filters** panel by clicking on the panel header or the ⊕ button.
6. Configure the filters.
   Refer to the table above for a list of the filters.

Regions

Many measurement tools use region parameters to limit the area in which a measurement will occur or to help in the identification of a feature (see on page 5), a fit line (see on page 7), or left or right side of the Panel tool (see *Panel (Gap and Flush) and Round Corner Algorithm* on page 7). Unlike reducing the active area, reducing the region does not increase the maximum frame rate of the sensor.

> You can turn regions off by unchecking the checkbox next to the **Regions** setting. When you do this, the measurement tool uses the entire active area.

All tools provide region settings under the **Parameters** tab.

Region settings are often found within expandable feature sections in the tool's panel.

> The **Region** settings apply to all of a tool's measurements.

See the individual tools for any details on using this parameter with each tool.

This parameter is also referred to as a measurement region.

> In 2D mode, the tool region defaults to the center of the current data view, not the global field of view. In 3D mode, the region defaults to the global field of view.

To configure regions:

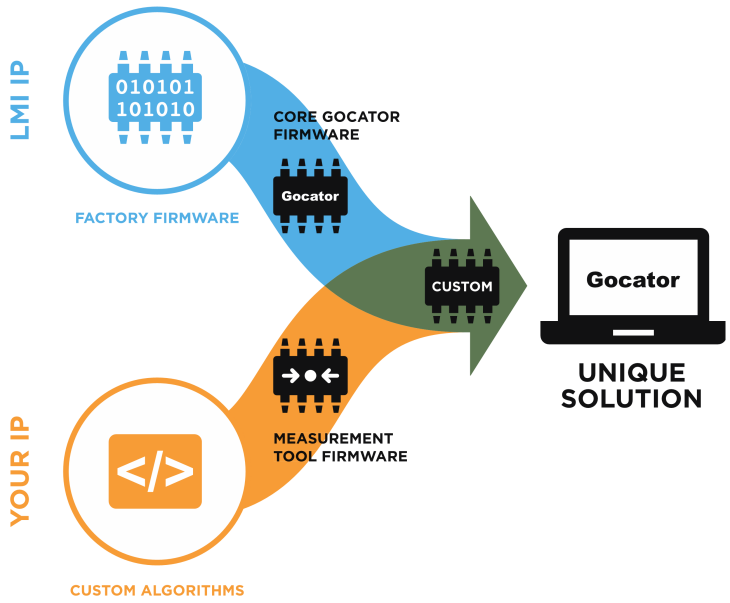
1. Go to the **Measure** page by clicking on the **Measure** icon.

   > The scan mode must be set to the type of measurement you need to configure. Otherwise, the wrong tools, or no tools, will be listed on the **Measure** page.

2. In the **Tools** panel, click on a tool in the tool list.
3. Expand the region section by clicking on the expand button ☰.
   Some region settings are found within other settings in this area.
4. Configure the region using the fields or graphically using the mouse in the data viewer.

Gocator Development Kit

The Gocator Development Kit (GDK) is a framework for developing and testing custom Gocator measurement tools containing your own algorithms, and then deploying them to Gocator sensors.



Custom tools created with the GDK act much like native Gocator measurement tools, running at native speeds and taking advantage of features such as anchoring. The GDK supports all data types (range, profile, and surface), and tools created with the GDK use the same data visualization as native tools.

> For the initial release, measurement tools created with the GDK only support Ethernet output. Also, these tools can't be used with Gocator Accelerator (for more information on the accelerator, see the main Gocator user manual). These capabilities will be added soon.

Supported Sensors

The GDK is available for free for the following Gocator sensors:

- Gocator 1300 series
- Gocator 2300 series
- Gocator 2400 series
- Gocator 3100 series

Typical Workflow

The following is the typical workflow for creating and deploying custom measurement tools:

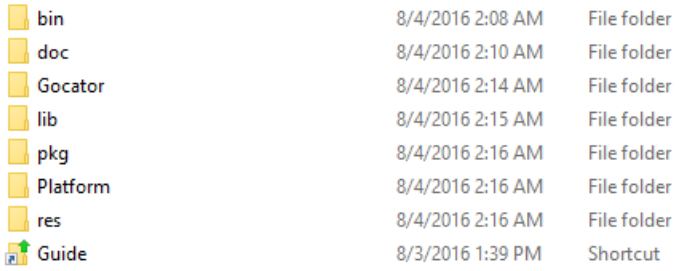
- Develop and build measurement tools using the GDK project files and libraries in Microsoft Visual Studio, targeting Win32.
- Debug the measurement tools using the emulator on a PC.
- Build the measurement tools into a custom firmware binary.
- Upload the custom firmware to a sensor.

Installation and Class Reference

The GDK project and library files are in the `14524-4.5.4.102_SOFTWARE_GDK.zip` GDK package. You can download the GDK package by going to http://lmi3d.com/support/downloads/, selecting a Gocator series, and clicking on the *Product User Area* link.

After downloading the package, extract the package to a directory.

You can access full installation and setup instructions, as well as the complete class reference documentation, by clicking the *Guide* shortcut under the root directory.



Required Tools

The GDK requires the following tools:

- Microsoft Visual Studio 2013
- Python 3.4 or later
- GCC ARM tools 4.9.3-p2 (for Gocator 2300C and 2400 series sensors)
- Texas Instruments C6000 Code Generation Tools 7.4.13 (for Gocator 1300, 2300A, 2300B, and 3100 series sensors)

You can download Microsoft Visual Studio 2013 from Microsoft's website.

The `14525_1.0.0.0_SOFTWARE_GDK_Prerequisites.zip` package contains the installation files for Python 5.2, the GCC ARM tools, and the Texas Instruments code generation tools. For information on installing the package, see *Installation and Class Reference* above.

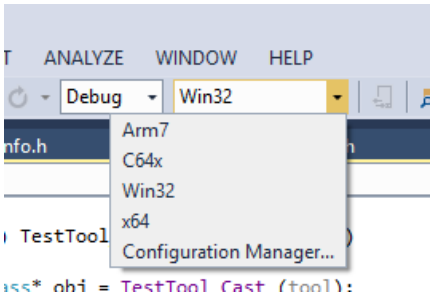
Getting Started with the Example Code

The best way to get started is with the GDK sample code. You can find the sample projects under `Gocator\GDKSampleApp`. This project is ready for you to build and use as a template for new projects.

Start by opening `GDK.sln` in Visual Studio 2013.

Building the Sample Code

You can build the sample code for working with either the emulator or a sensor. To do this, choose the target and then build the solution



The following targets are available:

- Win32/x64 for debugging code and emulating a sensor to test tools (on a PC)
- Arm7 for building for Gocator 2300C and 2400 series sensors
- C64x for Gocator 1300, 2300A, 2300B, and 3100 series sensors

The Win32 target supports Debug and Release builds. The Arm7 and C64x targets (Gocator sensors) only the support Release builds.

Tool Registration

For a tool to be available to a user in the Gocator web interface, you must add it to the project assembly in `Asm.c`.

```
#include <GdkSampleApp/Asm.h>
#include <GdkSampleApp/TestProfileSelect.h>
#include <GdkSampleApp/TestSurfaceSelect.h>
#include <GdkSampleApp/TestSurfaceConfiguration.h>
#include <GdkSampleApp/TestSurfaceGraphics.h>
#include <Gdk/GdkLib.h>
#include <GoSensor/Version.h>
#include <GoSensorAppLib/GsaDef.h>
#include <GoSensorAppLib/GsaAsm.h>

kBeginAssembly(Tool, ToolAsm, TOOL_VERSION, GOCATOR_VERSION)
    kAddDependency(GdkLib)
    kAddType(TestProfileSelect)
       kAddType(TestSurfaceSelect)
    kAddType(TestSurfaceConfiguration)
       kAddType(TestSurfaceGraphics)
```

```
kEndAssembly()
```

You can add multiple tools in a GDK project. As seen above, `TestProfileSelect`, `TestSurfaceSelect`, `TestSurfaceConfiguration` , etc. will be available for users from the drop-down menu in the **Tools** panel in sensor's web interface.

Tool Definitions

You must add standard entry functions (methods) for each tool. The class table declares the entry functions:

```
kBeginClass(Tool, TestTool, GdkTool)
    kAddVMethod(TestTool, kObject, VRelease)
    kAddVMethod(TestTool, GdkTool, VInit)
    kAddVMethod(TestTool, GdkTool, VDescribe)
    kAddVMethod(TestTool, GdkTool, VNewToolConfig)
    kAddVMethod(TestTool, GdkTool, VNewMeasurementConfig)
    kAddVMethod(TestTool, GdkTool, VUpdateConfig)
    kAddVMethod(TestTool, GdkTool, VStart)
    kAddVMethod(TestTool, GdkTool, VStop)
    kAddVMethod(TestTool, GdkTool, VProcess)
kEndClass()

ToolFx (kStatus) TestTool_VDescribe(GdkToolInfo toolInfo)
{
    GdkMeasurementInfo mmt;
    GdkParamsInfo params;
    GdkParamInfo paramInfo;

    kCheck(GdkToolInfo_SetTypeName(toolInfo, TEST_PROFILE_SELECT_TOOL_NAME));
    kCheck(GdkToolInfo_SetLabel(toolInfo, TEST_PROFILE_SELECT_TOOL_LABEL));

    kCheck(GdkToolInfo_SetSourceType(toolInfo, GDK_DATA_TYPE_UNIFORM_PROFILE));

    ...
```

The function `<Tool Name>_VDescribe` describes the tool and its basic configuration. This function is called during sensor start-up. For more information on entry functions, see *Entry Functions* below.

	The `VDescribe` function for each tool is properly formed. Significant issues with this function (for example, overwriting memory) could prevent the sensor from starting.

	You should use the emulator to debug tools *before* deploying tools to sensors.

Entry Functions

The following table describes the main entry functions.

Function	Description
VDescribe	Defines the tool's name, data types, acceptable source options, configuration parameters, and at least one measurement.
VStart	Called when the sensor starts running (that is, the user clicks the Run button). The

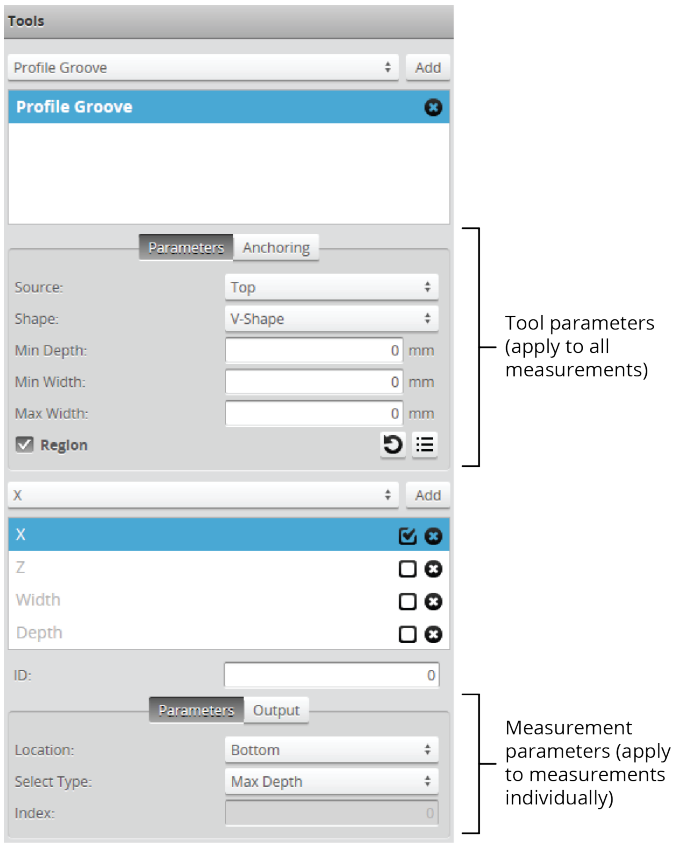
Function	Description
	function gets parameters from `GtTool`. You typically allocate memory in this function.
VProcess	Called every time data is received while the sensor is running.
VStop	Called when the user clicks the Stop button.

The `TestSurfaceConfiguration` example shows how to create and modify parameters based on other user settings.

For full descriptions of these functions, see the GDK class reference documentation (see *Installation and Class Reference* on page 32 for information on installing the documentation).

Parameter Configurations

Each tool has two levels of parameters: tool parameters and measurement parameters.



A tool can contain multiple measurements. In the image above, the Groove tool contains four measurements: X, Z, Width, and Depth. Each tool has one set of tool parameters and each measurement in a tool has one set of measurement parameters.

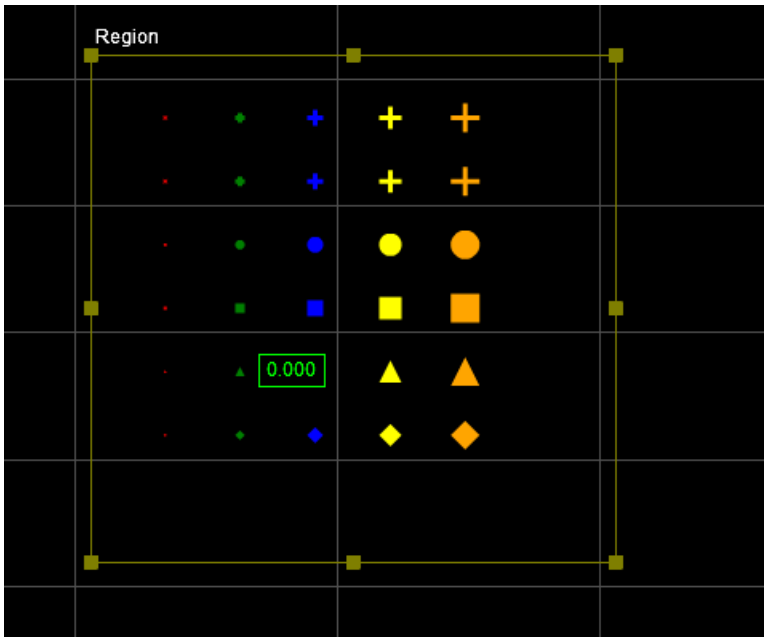
The following table lists the functions that provide advanced or interactive control for setting up tool and measurement parameters:

Function	Description
VNewToolConfig	Advanced method for setting default values of tool parameters based on the current sensor configuration (for example, active area). Called when a new tool is added in the interface.
VNewMeasurementConfig	Advanced method for setting default values of measurement parameters based on the current sensor configurations (for example, active area). Called when measurements in a tool is are added in the interface.
VUpdateConfig	Advanced method for updating the configuration based on parameters set by users.

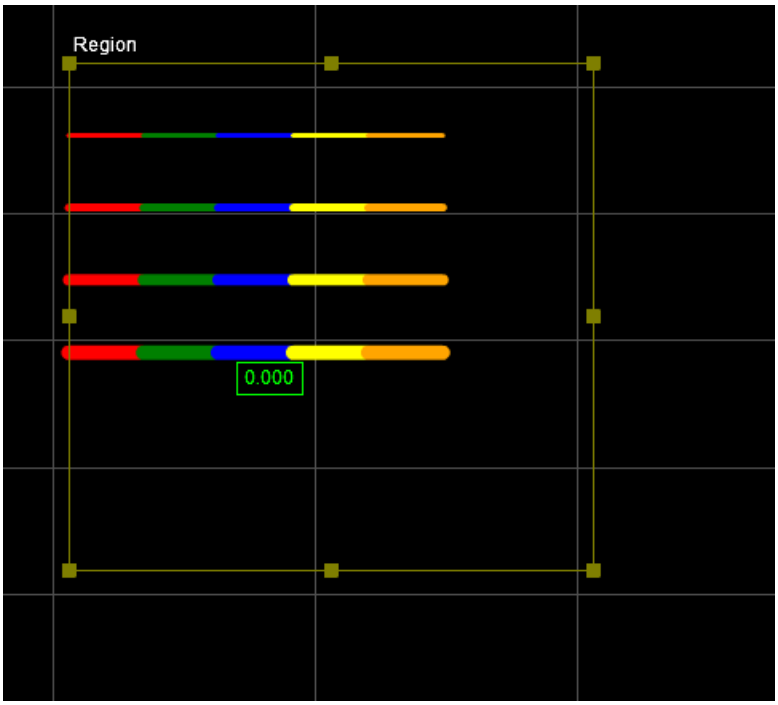
For full descriptions of these functions, see the GDK class reference documentation (see *Installation and Class Reference* on page 32 for information on installing the documentation).

Graphics Visualization

The `GdkGraphic` function supports points and lines.



Point graphics



Line graphics

To create graphics:

1. Use `GdkGraphic_Construct` to create a graphic object.
2. Use `GdkGraphicPointSet_Construct` to create points or `GdkGraphicLineSet_Construct` to create lines.
3. Add the points and lines to the graphic object using `GdkGraphic_AddPointSet` and `GdkGraphic_AddLineSet`.
4. Output using GdkToolOutput_`SetRendering`.

The following illustrates the process:

```
kTest(GdkGraphic_Construct(&graphic, kObject_Alloc(tool)));

kTest(GdkGraphicPointSet_Construct(&pointSet, 4.0, kMARKER_SHAPE_CROSS, kCOLOR_LIME,
&point32f, 1, kObject_Alloc(tool)));

kTest(GdkGraphic_AddPointSet(graphic, pointSet));

kTest(GdkToolOutput_SetRendering(output, measurementIndex, graphic));
```

The GDK example `TestSurfaceGraphics` shows how to use the graphics functions.

> Graphic functions take an array of `kPoint3d32f`. It does NOT accept `kPoint3d64f`.

Debugging Your Measurement Tools

We highly recommend using the emulator to debug measurement tools you create with the GDK. By using a Gocator support file and previously recorded scan data, downloaded from a physical sensor, you can completely simulate standalone and dual-sensor configurations on a PC to test your tools.

To debug your tools in the emulator:

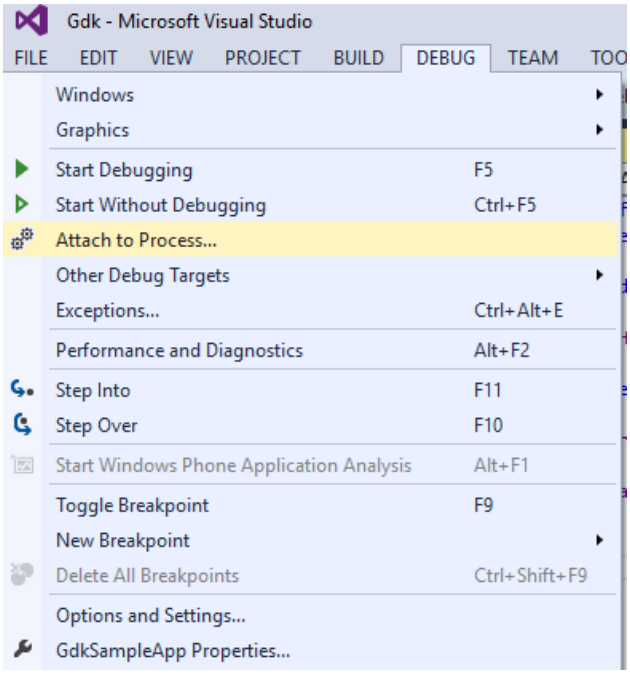
1. Compile your code using the Win32 target (Debug or Release).
2. In the output directory, rename the DLL with the same name as your project to GdkApp.dll.

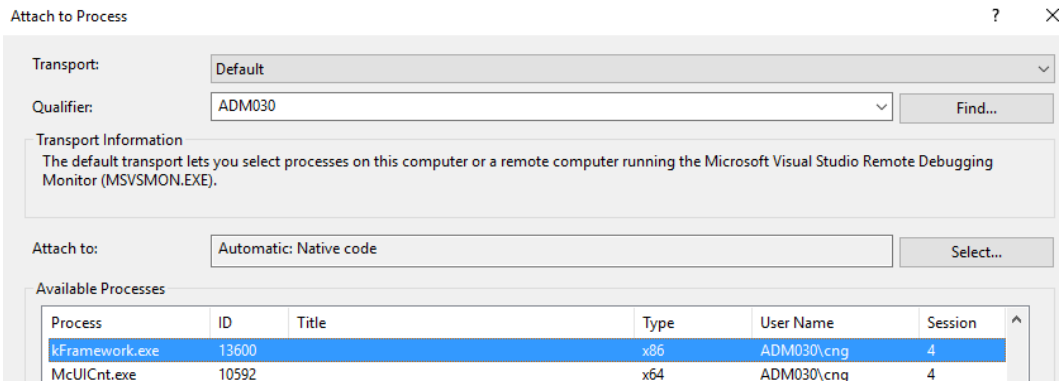
   For example, if your project is called `MyGDKTools`, the resulting DLL should be called `MyGDKTools.dll`. You rename this DLL to GdkApp.dll.

   The output directories are as follows:

   Release: `win32`

   Debug: `win32d`
3. Launch the emulator from same output directory as in step 2.
4. In the emulator, choose a scenario and start it.
5. In Visual Studio, attach the debugger to the `kFramework.exe` process.





> `kFramework.exe` is only loaded after a user selects a scenario and starts the emulator session.

Debugging Entry Functions

VStart, *VProcess*, and *VStop* are called whenever a data record is played back in the emulator (that is, when a user clicks on the Next button or types the frame number in the frame field) with at least one tool instance. For more information on playback controls, see *Recording, Playback, and Measurement Simulation* in the Gocator user manual.

VDescribe however is called when the DLL loads, before the debugger can attached to the `kFramework.exe` process. To debug *VDescribe*, we recommend testing the function calls by putting them in *VInit*.

> For information on building targets for testing in the emulator, see the GDK class reference documentation.

Tips

The following sections provide useful information for creating custom measurement tools.

Backward Compatibility with Older Versions of Tools

When loading a recording or job file that contains a custom measurement tool, the parameters in the loaded recording or job file must match those in the firmware.

By default, if declared parameters are missing from the configuration, a job file or a recording will fail to load.

There are two ways to provide backward compatibility with older parameter sets.

Define new parameters as optional

Mark a parameter as optional with the function `GdkParamInfo_SetIsOptional`. When a parameter is marked as optional, parameter parsing functions succeed even if the parameter is missing from the configuration. The missing parameter is initialized with default value.

Configuration Versioning

Over the lifetime of a tool, you may need to make changes to its interface (for example, changing or removing parameters). The user-defined aspects of a tool interface—its parameters and measurements—are captured by `GDKToolVersionInfo` objects.

By default, a tool has just one version (`GdkToolInfo_FirstVersion`), but more versions may be added using `GdkToolInfo_AddVersion`. Whenever the interface of a tool has changed, a new version can be registered so that the new interface can be correctly parsed by the framework.

When the configuration of a tool instance is saved, the version used at the time is also saved. This saved version is used by the framework to parse the configuration. If a version not defined by the firmware implementation, then that tool instance will not be inactive.

During run-time, you can query the version of the configuration of a tool instance by using `GdkToolCfg_Version`. You can then interpret the parameters depending on the version the configuration is saved in.

```
GdkFx(kStatus) GdkExampleTool_VDescribe(GdkToolInfo info)
{
	kCheck(GdkToolInfo_SetLabel(info, "Example"));

	kCheck(GdkToolInfo_SetSourceType(info, GDK_DATA_TYPE_UNIFORM_PROFILE));
	kCheck(GdkToolInfo_AddSourceOption(info, GDK_DATA_SOURCE_TOP));

	kCheck(GdkExampleTool_DescribeV0(info));
	kCheck(GdkExampleTool_DescribeV1(info));

	kCheck(GdkToolInfo_SetDefaultVersion(info, GdkToolInfo_VersionAt(info, 1)));

	return kOK;
}

GdkFx(kStatus) GdkExampleTool_DescribeV0(GdkToolInfo info)
{
	kCheck(GdkParamsInfo_Add(GdkToolInfo_Params(info), "RefRegion", GDK_PARAM_TYPE_PROFILE_
REGION, "Ref Region", kNULL));
	kCheck(GdkParamsInfo_Add(GdkToolInfo_Params(info), "Region", GDK_PARAM_TYPE_PROFILE_
REGION, "Region", kNULL));
	kCheck(GdkToolInfo_SetFirstVersionName(info, ""));

	return kOK;
}

GdkFx(kStatus) GdkExampleTool_DescribeV1(GdkToolInfo info)
{
	GdkToolVersionInfo versionInfo;

	// Auto-version

	kCheck(GdkToolInfo_AddVersion(info, kNULL, &versionInfo));
	kCheck(GdkToolVersionInfo_UseBase(versionInfo, GdkToolInfo_FirstVersion(info)));
	kCheck(GdkParamsInfo_AddFloat(GdkToolVersionInfo_Params(versionInfo), "BaseScale",
kNULL, 2.0, kNULL));
```

```
        return kOK;
}
```

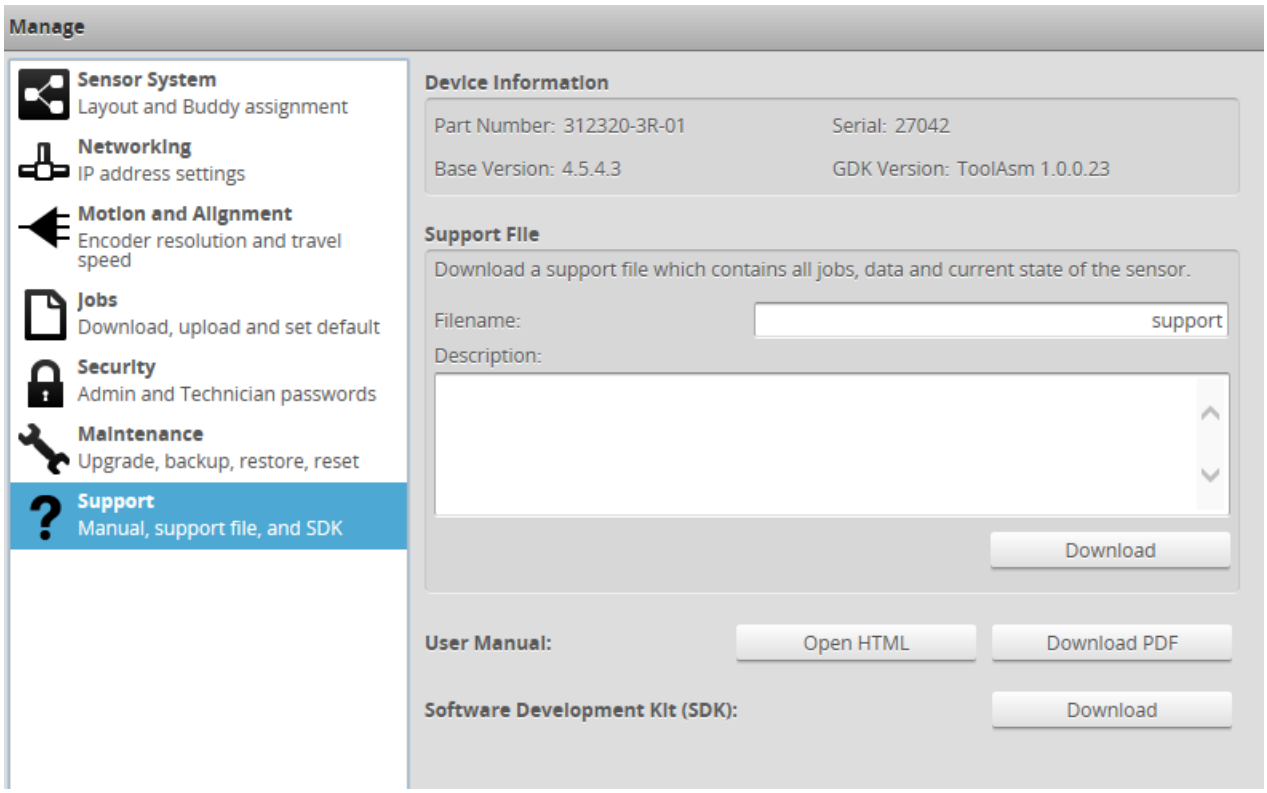
Adding a new measurement does not require special handling. The new measurement is just not instantiated in a previous configuration.

Version

You can define the version number of your tools in `Asm.x.h`.

```
#define TOOL_VERSION    kVersion_Stringify_(1, 0, 0, 23)
```

The version is displayed on the **Manage** page, in the **Support** category.



Common Programming Operations

The following sections describe common programming operations.

Input Data Objects

The `VProcess` function receives a `GdkToolInput` object as input. This object is a container where the information and actual data of the received input is stored.

```
GdkInputItem item = GdkToolInput_Find(input, obj->dataSource);
GdkDataInfo itemInfo = GdkInputItem_Info(item);
```

The `GdkToolInput_Find` and `GdkInputItem_Info` functions are used to extract the item and info objects. These objects can then be used to retrieve the input data and information (for example, offset and resolution) associated to the input. The following are some examples:

Retrieving offset and scales

```
const kPoint3d64f* scale = GdkDataInfo_Scale(GtInputItem_Info(item));
const kPoint3d64f* offset = GdkInputItem_Offset(item);
```

Retrieving a pointer to surface and intensity surface data

```
const k16s* rangeSrc = GdkSurfaceInput_RangeRowAt(item, 0);
const k8u* intensitySrc = GdkSurfaceInput_IntensityRowAt(item, 0);
```

Similar functions are available for retrieving information from range and profile data.

Computing actual height information using a offset and scales

```
k64f height = rangeSrc[index] * scale->z + offset->z;
```

Extracting height information from profiles and surfaces.

The `TestProfileSelect` and `TestSurfaceSelect` examples show how to perform these operations.

Setup and Region Info during Tool Initialization

Memory allocation is often done in the `VInit` or `VStart` function. To retrieve sensor and data information such as active area settings and data scale outside of `VProcess`, you can use the following function:

```
GdkDataInfo info = GdkSensorInfo_DataSource(GdkTool_SensorInfo(tool), GDK_DATA_SOURCE_
TOP);
```

Computing Region Based on the Offset from an Anchor Source

Just like built-in measurement tools, custom tools created with the GDK can be anchored to another tool (GDK-based tools or built-in tools).

To compute the offset region:

```
TestToolClass* obj = TestTool_Cast_(tool);
GdkParams params = GdkToolCfg_Parameters(config);
const kPoint3d64f* anchor = GdkToolInput_AnchorPosition(input);
GdkRegionXZ64f offsetRegion = { k64F_NULL, k64F_NULL, k64F_NULL, k64F_NULL };

param = GdkParams_Find(params, "Region");
obj->region = *GdkParam_AsProfileRegion(param);

offsetRegion = obj->region;
offsetRegion.x += anchor->x;
offsetRegion.z += anchor->z;
```

In the code above, we first retrieve the tool's region settings (before anchoring is applied), and then adjust the region based on the results from the anchored source in *VProcess*. If the anchored source fails, the tools will not be invoked.

The `TestProfileSelect` and `TestSurfaceSelect` examples show how to extract height information from anchored regions.

For more information on anchoring, see *Measurement Anchoring* in the Gocator user manual.

Part Matching

When part matching is enabled, the tool receives translated and corrected surface data. If part matching fails for the current scan (for example, the quality score is too low), the tools will not be invoked.

For more information on part matching, see *Part Matching* in the Gocator user manual.

Accessing Sensor Local Storage

You can access a sensor's local storage by using the kFile API.

For example, to read and write a file to a sensor's storage, you could use the following:

```
#include <kApi/Io/kFile.h>

…

ToolFx(kStatus) TestTool_VStart(TestTool tool)
{
       …

       kFile_Save("test.txt", stringBuf, (kSize) 1024);
       kFile_Load("test.txt", stringBuf, &bufLen, kNULL);
```

Print Output

In the emulator, you can send output to Visual Studio or to programs such as DebugView by using the `OutputDebugString` function.

```
GtsFx(kStatus) TestTool_Trace(const kChar* format, ...)
{
    kStatus status = kOK;
    kChar debugLine[256];

    kVarArgList argList;
    kVarArgList_Start_(argList, format);
    {
        status = kStrPrintvf(debugLine, 256, format, argList);
    }
    kVarArgList_End_(argList);
    OutputDebugStringA(debugLine);
    return status;
}
```

> OutputDebugString is NOT supported on sensor targets. Use #ifdef to comment out the code when compiling against sensor targets.

Software Licenses

Pico-C

Website:

http://code.google.com/p/picoc/

License:

picoc is published under the "New BSD License".

http://www.opensource.org/licenses/bsd-license.php

BlowFish

Website:

http://www.chiark.greenend.org.uk/~sgtatham/putty/licence.html

License:

PuTTY is copyright 1997-2011 Simon Tatham.

Portions copyright Robert de Bath, Joris van Rantwijk, Delian Delchev, Andreas Schultz, Jeroen Massar, Wez Furlong, Nicolas Barry, Justin Bradford, Ben Harris, Malcolm Smith, Ahmad Khalifa, Markus Kuhn, Colin Watson, and CORE SDI S.A.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL SIMON TATHAM BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

CodeMirror

Website:

http://codemirror.net

License:

Copyright (C) 2011 by Marijn Haverbeke <marijnh@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANT ABILITY,FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

jQuery

Website:

http://jquery.com/

License:

Copyright (c) 2011 John Resig, http://jquery.com/

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Closure Library

Website:

http://code.google.com/closure/library/index.html

License:

Copyright 2006 The Closure Library Authors. All Rights Reserved.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License.

You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS-IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

jQuery.CopyEvents

Website:

http://brandonaaron.net

License:

Copyright (c) 2006 Brandon Aaron

Licensed under the MIT License (http://www.opensource.org/licenses/mit-license.php)

jQuery.history

License:

jQuery history plugin

Copyright (c) 2006 Taku Sano (Mikage Sawatari)

Licensed under the MIT License (http://www.opensource.org/licenses/mit-license.php)

Modified by Lincoln Cooper to add Safari support and only call the callback once during initialization for msie when no initial hash supplied. API rewrite by Lauris Bukis-Haberkorns

jQuery.mouseWheel

Website:

http://brandonaaron.net

License:

Copyright (c) 2010 Brandon Aaron

Licensed under the MIT License (http://www.opensource.org/licenses/mit-license.php)

jQuery.scaling

Website:

http://eric.garside.name

License:

Scaling 1.0 - Scale any page element

Copyright (c) 2009 Eric Garside

Licensed under the MIT License (http://www.opensource.org/licenses/mit-license.php)

jQuery.scrollFollow

Website:

http://kitchen.net-perspective.com/

License:

Copyright (c) 2008 Net Perspective

Licensed under the MIT License (http://www.opensource.org/licenses/mit-license.php)

Flex SDK

Website:

http://opensource.adobe.com/wiki/display/flexsdk/Flex+SDK

License:

Copyright (c) 2010 Adobe Systems Incorporated

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.mozilla.org/MPL/

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

EtherNet/IP Communication Stack

Website:

sourceforge.net/projects/opener

License:

SOFTWARE DISTRIBUTION LICENSE FOR THE

ETHERNET/IP(TM) COMMUNICATION STACK (ADAPTED BSD STYLE LICENSE)

EtherNet/IP is a trademark of ODVA, Inc.

Support

For help with a component or product, please submit an online support ticket using LMI's Help Desk at http://support.lmi3d.com/newticket.php.

If you are unable to use the Help Desk or prefer to contact LMI by phone or email, use the contact information below.

> Response times for phone or email support requests will take longer than requests through the Help Desk.

North America

Phone	+1 604 636 1011
Fax	+1 604 516 8368
Email	support@lmi3d.com

Europe

Phone	+31 45 850 7000
Fax	+31 45 574 2500
Email	support@lmi3d.com

For more information on safety and laser classifications, please contact:

U.S. Food and Drug Administration
Center for Devices and Radiological Health
WO66-G609
10903 New Hampshire Avenue
Silver Spring MD 20993-0002
USA

Contact

Americas	EMEAR	ASIA PACIFIC
LMI Technologies (Head Office)	LMI Technologies GmbH	LMI (Shanghai) Trading Co., Ltd.
Burnaby, Canada	Berlin, Germany	Shanghai, China
+1 604 636 1011	+49 (0)3328 9360 0	+86 21 5441 0711

LMI Technologies has sales offices and distributors worldwide. All contact information is listed at lmi3D.com/contact/locations.