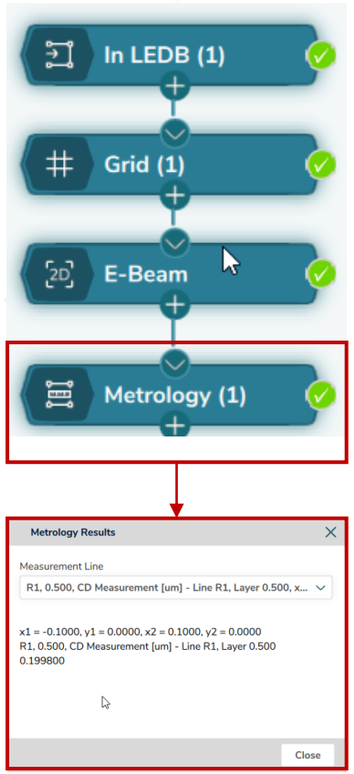
GenISys
Advancing the Standard

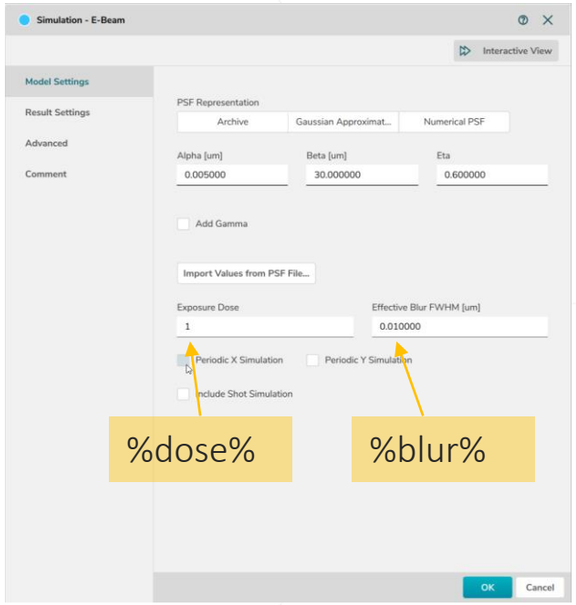
BEAMER

Loops, Variables, Functions

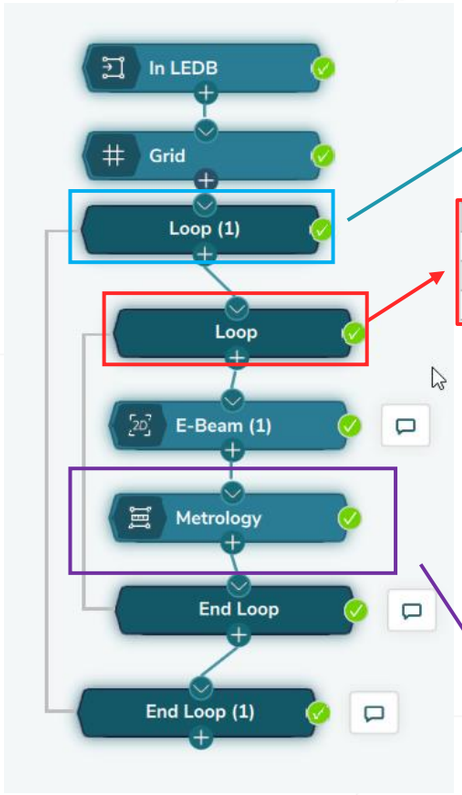
Variable Applications

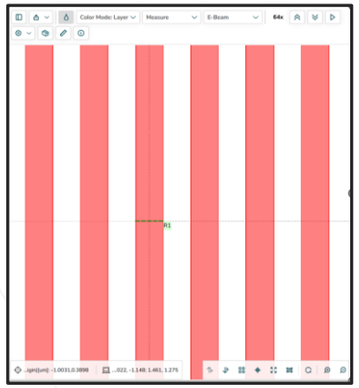
Variable handling is a key factor for effective and flexible working inside of flows for pattern processing.



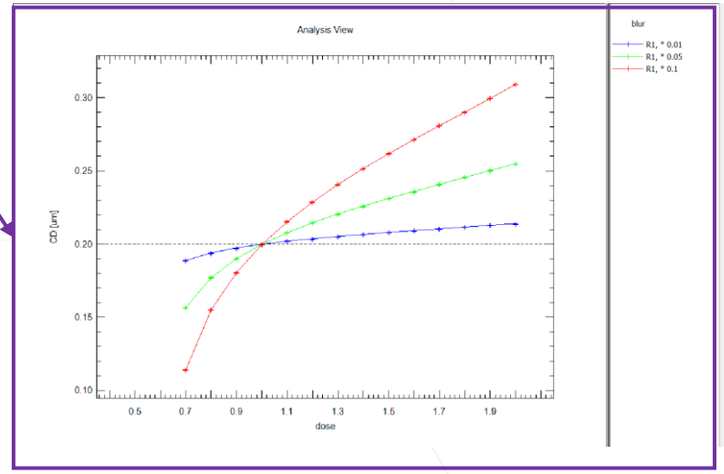


E-Beam simulation with metrology





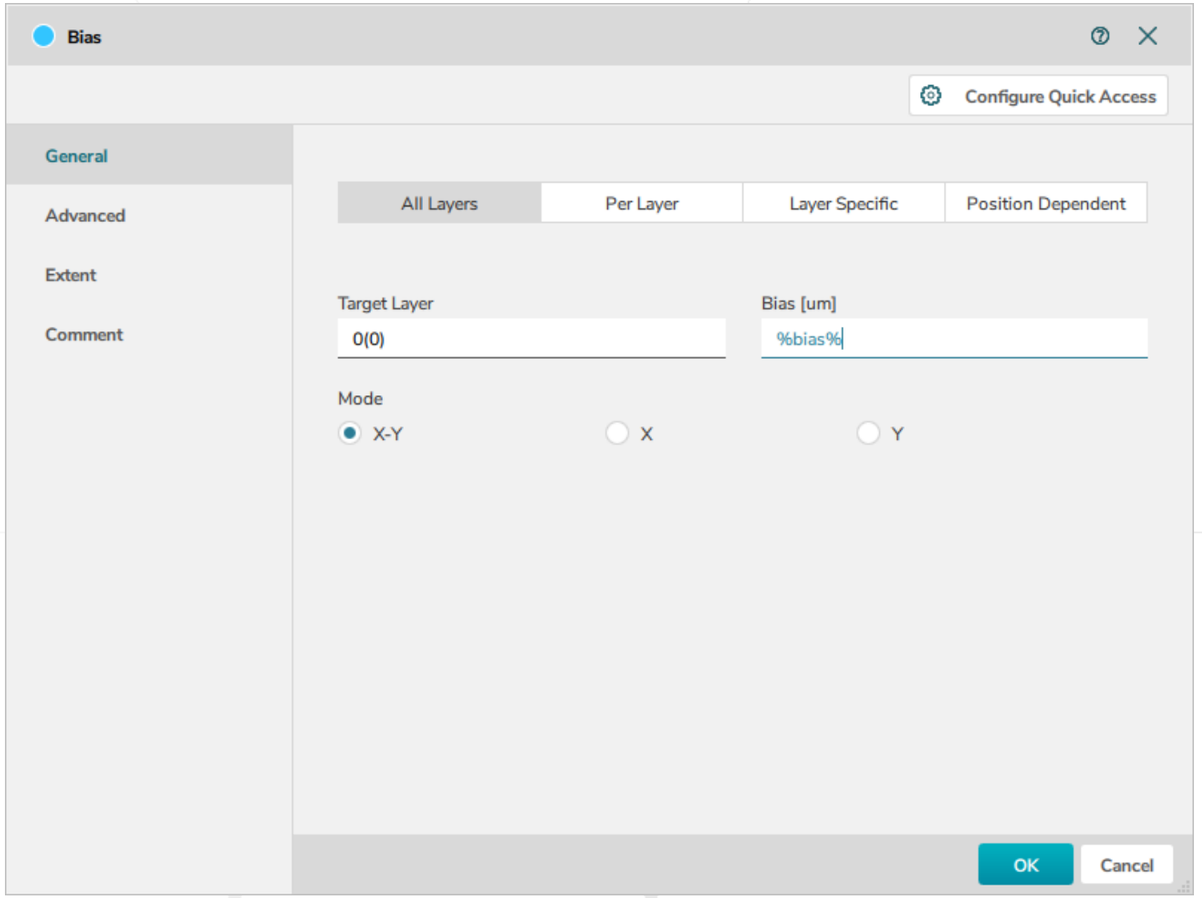
50% L/S pattern



Agenda

- Places for variables
- Definitions of variables
- Local and global variables
- Default values
- Math with a single variable
- Math with multiple variables
- Reading environmental variables
- Functions as inputs

- Variables can be used in the GUI at any location that allows text input. Examples:
  - Exposure dose
  - Effective blur
  - Bias
  - Target layer
  - Field size
  - Path for the IMPORT module or EXPORT module
  - …



Agenda

- Places for variables
- Definitions of variables
- Local and global variables
- Default values
- Math with a single variable
- Math with multiple variables
- Reading environmental variables
- Functions as inputs

- Variables are defined by a name enclosed within ‚**%**‘ symbols at at the start and end of the name. (%dose%, %x%...)
- Restricted names are:
  - %IMPORT%
    This holds the full path plus file name of the last IMPORT module in the flow.
  - %EXPORT%
    This holds the full path plus file name of the last EXPORT module in the flow.
  - %EnvVar_<name>%
    Allows the reading of a system environmental variable by the name of <name>.

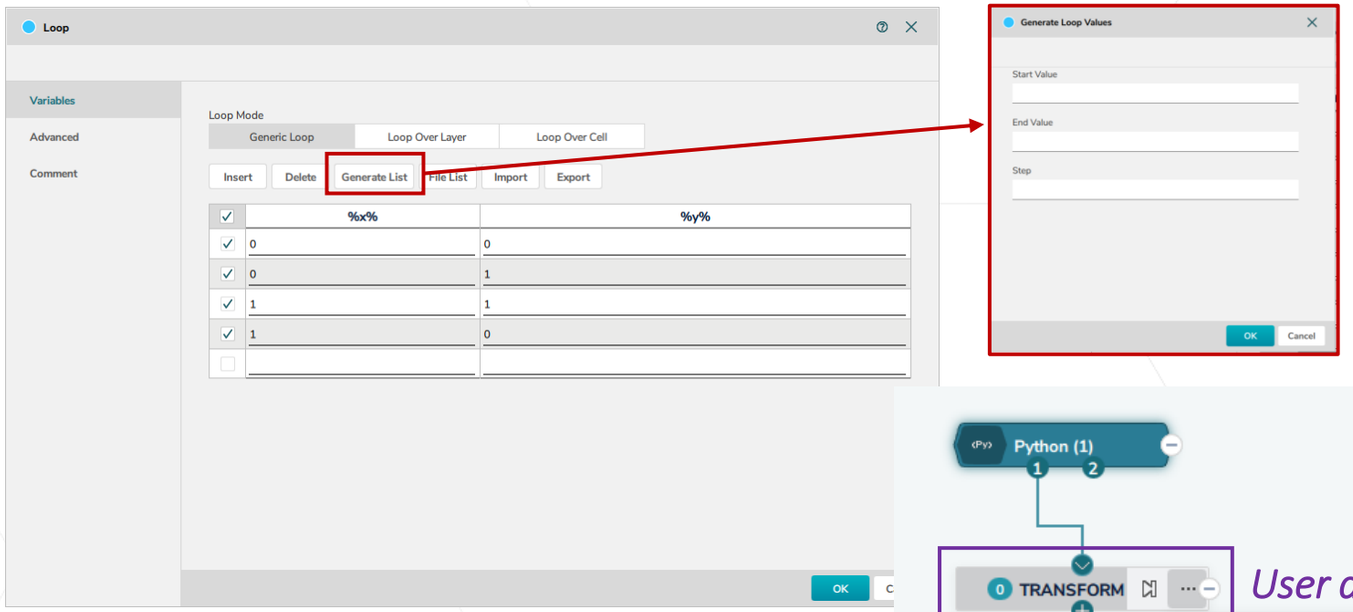
Agenda

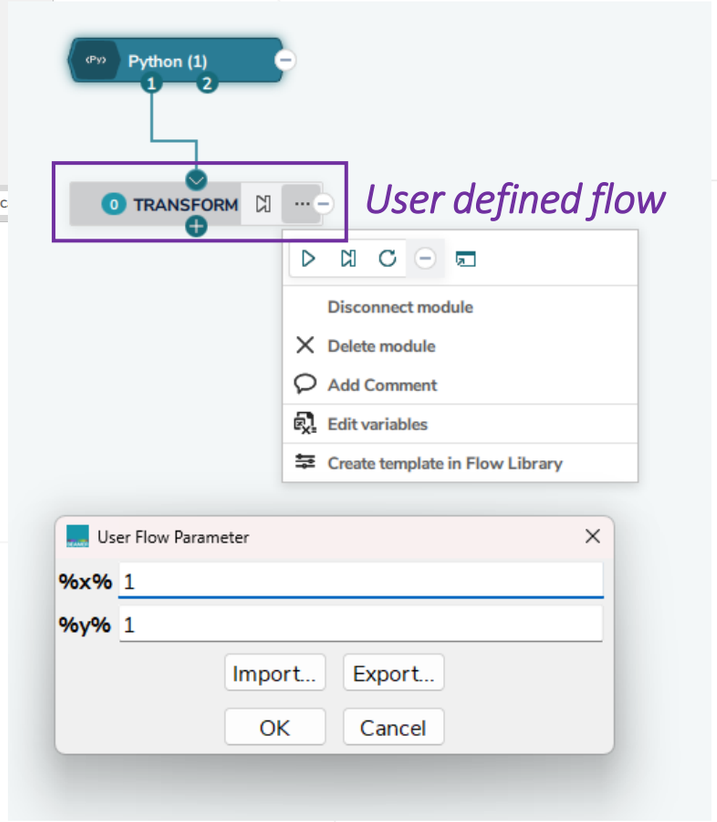
- Places for variables
- Definitions of variables
- Local and global variables
- Default values
- Math with a single variable
- Math with multiple variables
- Reading environmental variables
- Functions as inputs

- Local variables
  - Variables defined **within loops or a *User defined flow***.
  - Valid only within the modules of the loops or the User defined flow, and can‘t be used outside.
  - Local variable takes dominance over the globally defined one, meaning you can overwrite global variables with local variables.
- Global variables
  - Variables defined **outside of loops and *User defined flows***.
  - Can be used in the flow as often as desired.
  - Only change the value once and be applied in multiple places.

Local variables

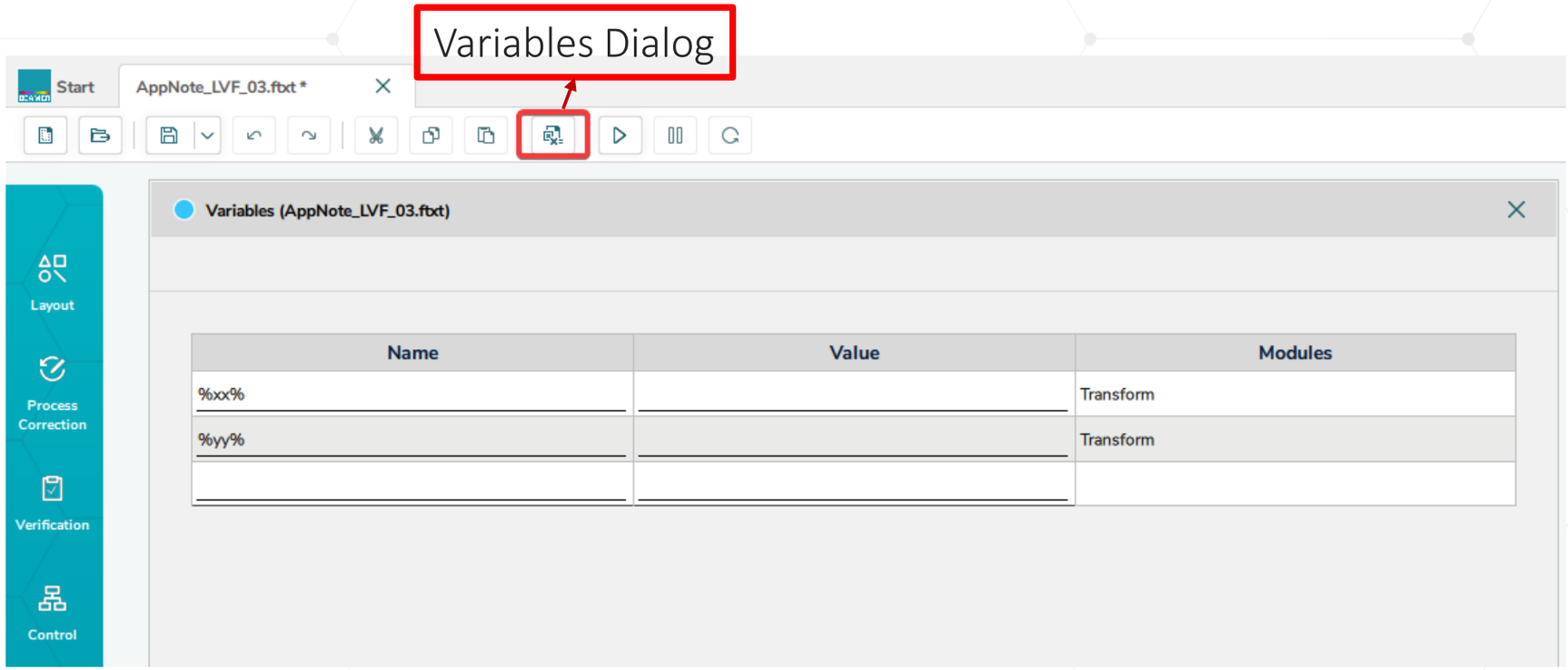
- Defined in loop:
  - A Loop module setup allows the definition of variables as single or coupled setup.
  - Each row will be executed in sequence.
- Defined in *User defined flow*
  - Get access to the variables via the ‚…‘ and the ‚Edit variables‘ menu.





Global variables

- A global variable interface displays all global variables used within the flow.



The values are easily set in the *Variables Dialog* which also displays the module name where the variable is used.

Agenda

- Places for variables
- Definitions of variables
- Local and global variables
- Default values
- Math with a single variable
- Math with multiple variables
- Reading environmental variables
- Functions as inputs

- Global variables contain by default no value. An input value is needed for execution of the flow, otherwise an error is shown.
- Defaults for variables allow to preload a variable with a specific value and therefore avoid error in the execution of the flow.
- Default definition:
  Simply insert a set of parentheses ‘()’ containing the desired value after the variable's name, but before the final '%'.

  Example: %name(20)%

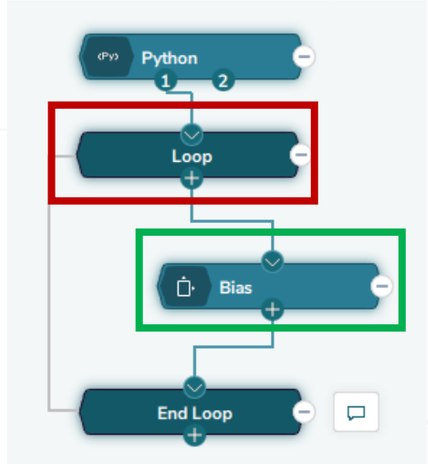
  Variable %name% is established with a default value of 20

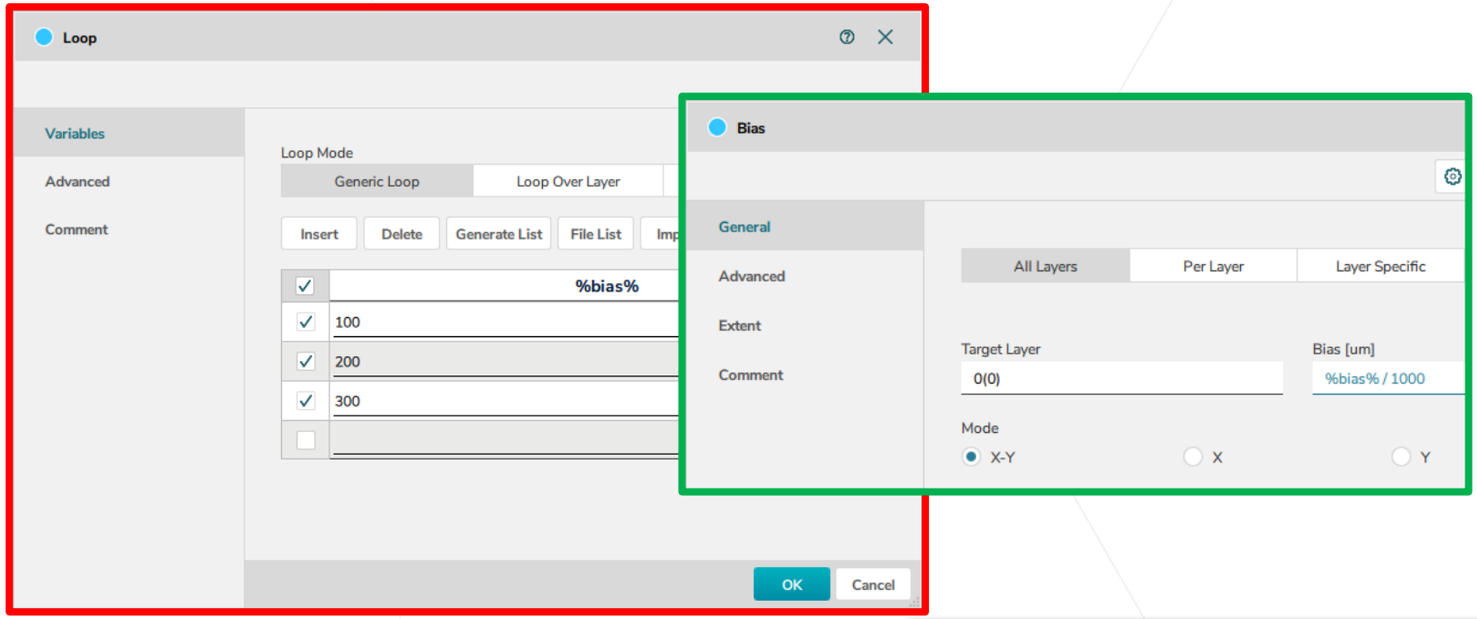
Agenda

- Places for variables
- Definitions of variables
- Local and global variables
- Default values
- Math with a single variable
- Math with multiple variables
- Reading environmental variables
- Functions as inputs

- Math operations with single variable are possible by including the desired formula in the field.
- For instance, if I would like to set my variable in units of nm but the field expects an input of µm, one can apply the conversion operation (division or multiplication).
- Example:

%bias% / 1000





Agenda

- Places for variables
- Definitions of variables
- Local and global variables
- Default values
- Math with a single variable
- Math with multiple variables
- Reading environmental variables
- Functions as inputs

- String creation can be a simple chain of text and variables:

Example: %x% and %y% indexing a matrix position

Export it as a file with matrix position information:

`Samplefile_posX%x%_posY%y%.gds`

Result: Samplefile_posX0_posY0.gds,
Samplefile_posX0_posY1.gds,
…

- Example: %x% and %y% indexing a matrix position

  Adding an offset to the position (100, 200…)

- In this case some math operation would be needed, the syntax is %( )% wrapping the formula.

Export it as a file with matrix position information:

`Samplefile_posX%(%bias%+%x%)%_posY%(%bias%+%y%)%.gds`

Result: Samplefile_posX100_posY100.gds,
Samplefile_posX100_posY101.gds,
…

Agenda

- Places for variables
- Definitions of variables
- Local and global variables
- Default values
- Math with a single variable
- Math with multiple variables
- Reading environmental variables
- Functions as inputs

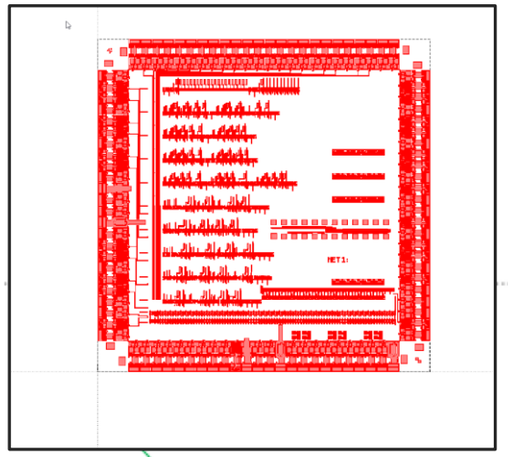
- In some cases, getting information from the environmental variables of the operating system can be helpful.
- Application case: Store a specific user or project name in a variable and append this to file names.
- **%EnvVar_<name>%:** Allows the reading of a system environmental variable by the name of <name>.
- Preset system variables:
  ProjectName: VariableDemo
  UserName: Ritter
- Sample:
  `%EnvVar_ProjectName%_%EnvVar_UserName%_myproject_01.gds`

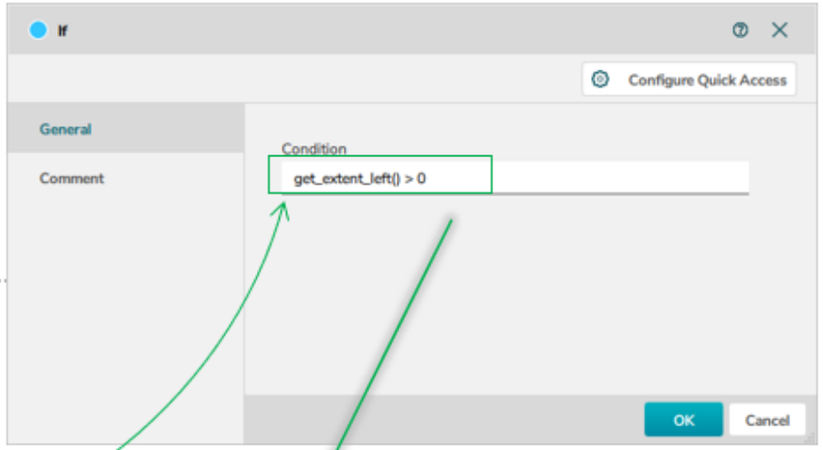
  VariableDemo_Ritter_myproject_01.gds

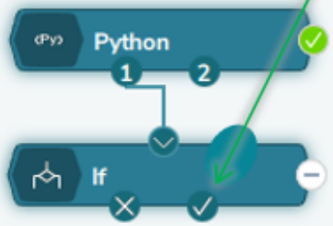
Agenda

- Places for variables
- Definitions of variables
- Local and global variables
- Default values
- Math with a single variable
- Math with multiple variables
- Reading environmental variables
- Functions as inputs

- The IF and SELECT modules support functions for decision-based flow.
- These functions use a Python-orientated syntax to read information from layout.
- Following function calls can be integrated into the formula.



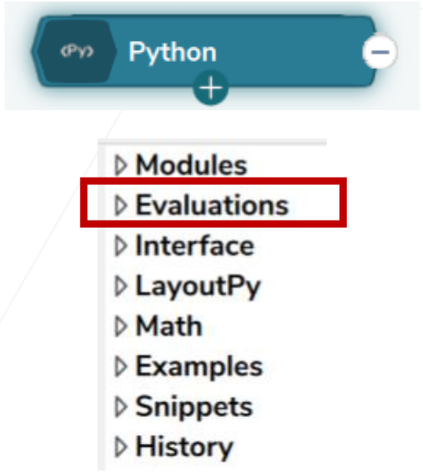


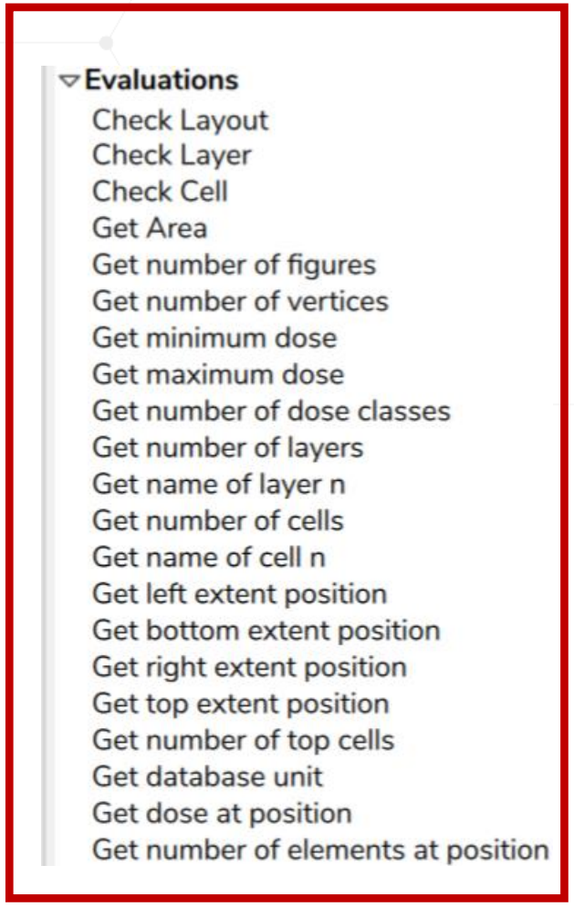


Information	Visual Flow Query (IF module)	Python	Example Output
Pattern area (um²)	get_area()	b.get_area(imp1)	23423.5
Figure count	get_number_figures()	b.get_	
Vertex count	get_number_vertices()	b.get_	
Dose Range	get_min_dose() get_max_dose()	b.get_ b.get_	
Assigned dose count	get_number_doseclasses()	b.get_	
Data Layer Count	get_number_layer()	beamer_	
Cell Count	get_number_cells()	beamer_	
Data Extents	get_extent_left() get_extent_right() get_extent_top() get_extent_bottom()	b.get_ b.get_ b.get_ b.get_	
Top Cell Count	get_number_topcells()	b.get_	
Database Unit (um)	get_database_units()	b.get_	

Information	Visual Flow Query (IF module)	Python	Example Output
Dose(s) assigned at coordinate (x,y)	get_dose(x,y)	b.get_dose(imp1,x,y)	1.156
Element (overlap) count at (x,y)	get_number_elements(x,y)	b.get_number_elements(imp1,x,y)	1
Layer Existence	check_layer_exist('7(0)')	b.check_layer_exist(imp1,'7(0)')	True
Layout Existence	check_layout_exist()	b.check_layout_exist(imp1)	False
Name of cell *n* *n < get_number_cells()*	get_cell(*n*)	b.get_cell(imp1,n)	"Dev3"
Layer Specification *n* *n < get_number_layer()*	get_layer(*n*)	b.get_layer(imp1,*n*)	"7(0)"

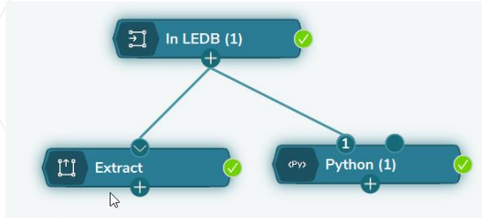
- *Python* module provides the tools for the perfect symbiosis with **BEAMER**



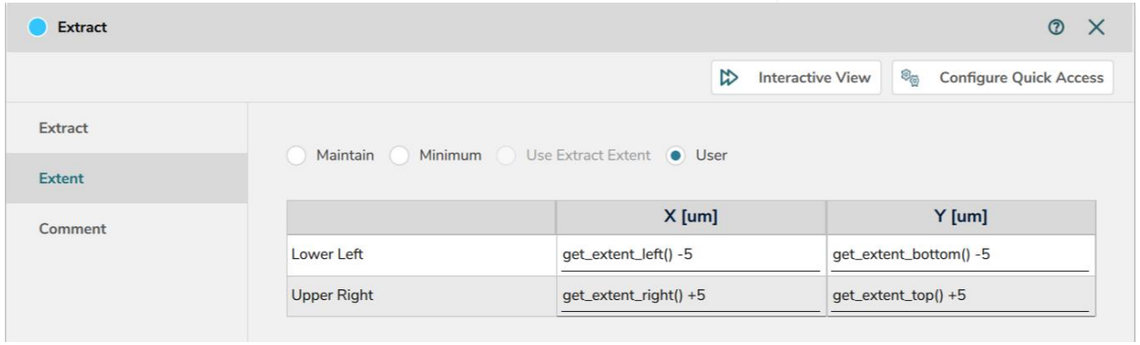


Commands that allow interaction with the layout

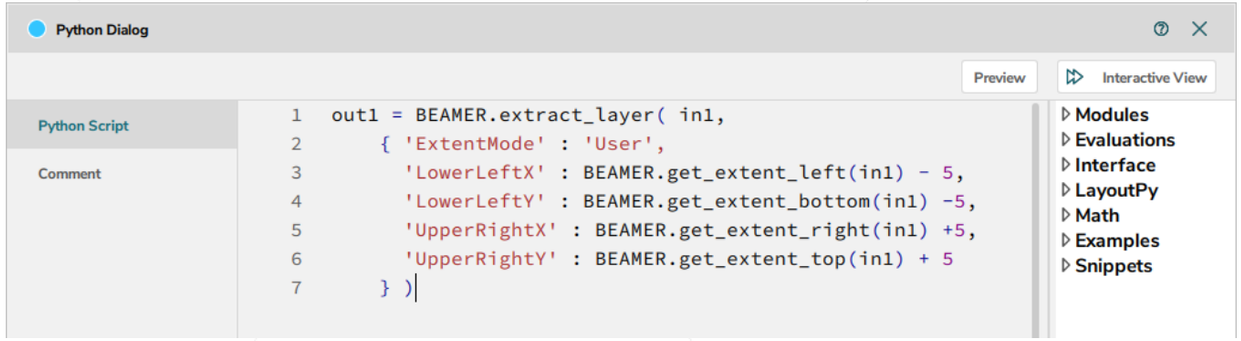
- Comparison Python with GUI
  - Grow the extent of the current pattern bounding box by 5µm in each direction. This should be independent of the pattern coming in and therefore should work for all layouts without the user needing to inspect the pattern extent and adjust.



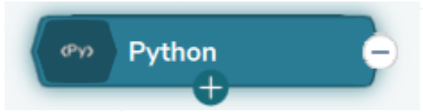
GUI

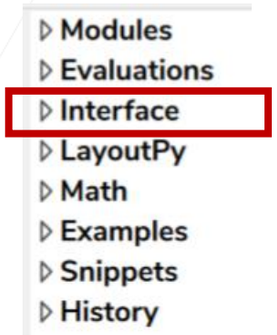


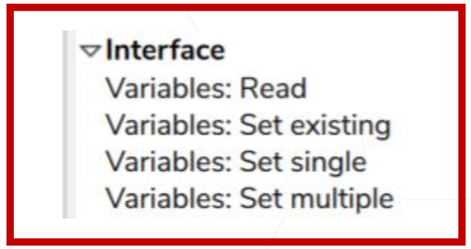
Python



- Python module provides the tools for the perfect symbiosis with **BEAMER**

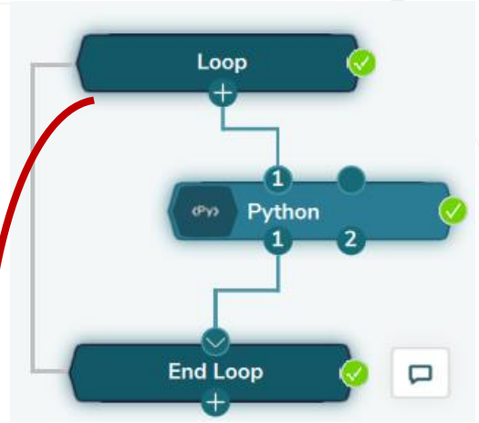


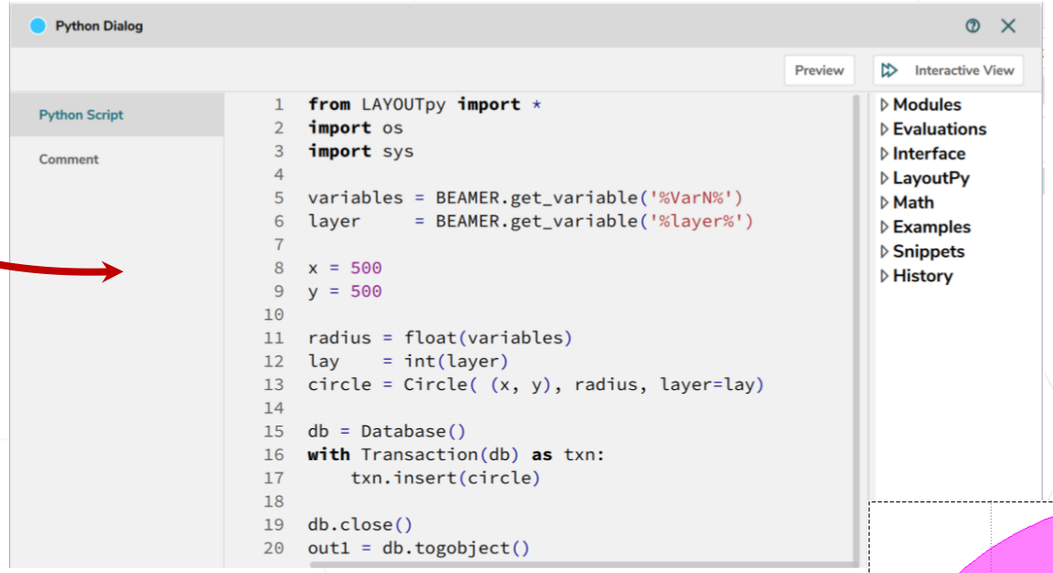


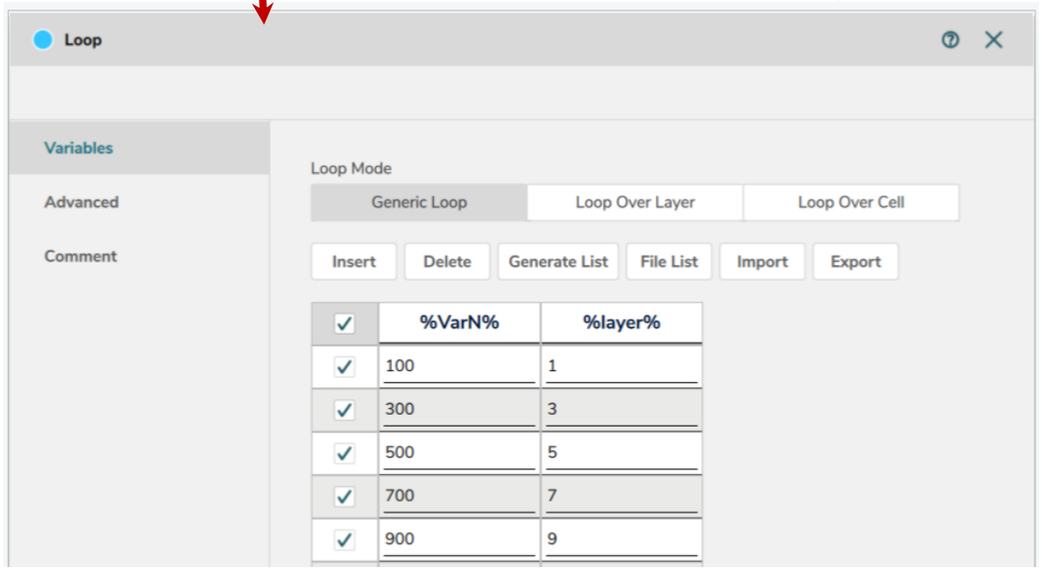


Commands that allow interaction with the layout and using variables as input in the Python environment. They allows you to read, create or change values of the variables which is defined in Beamer GUI.

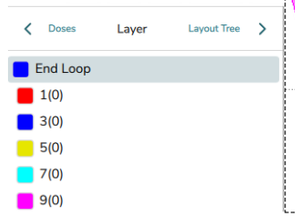
- Variables can be used within the Python module environment

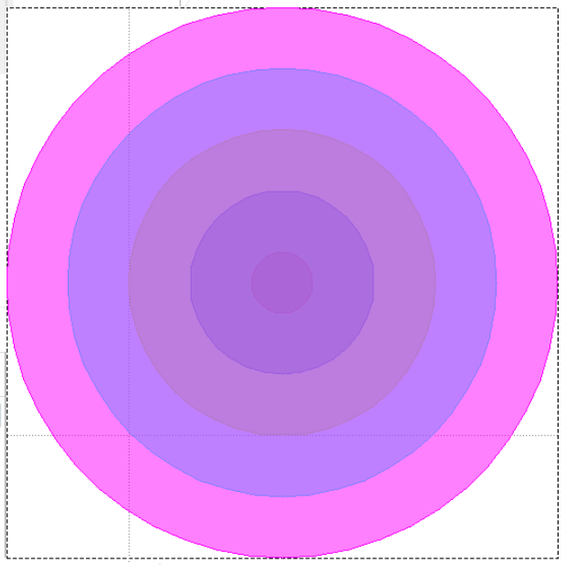






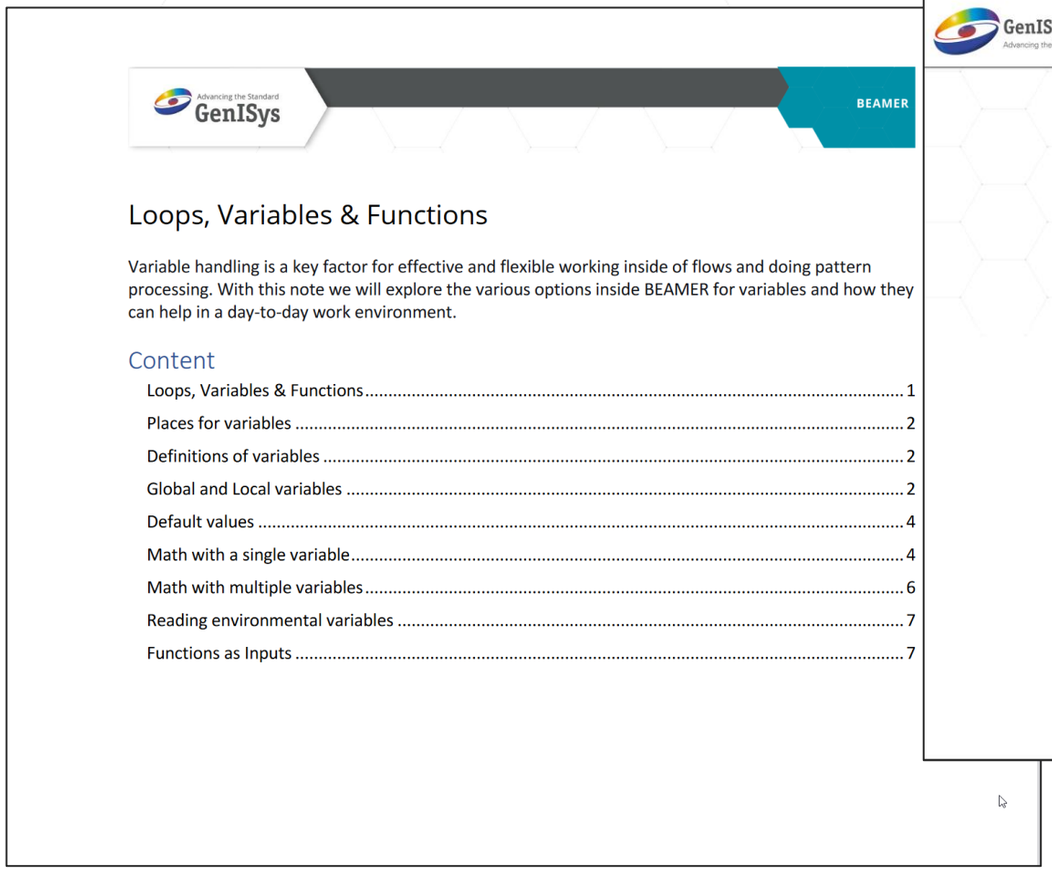
Variables are created in a *Loop* module and used as **radius** and **layer** values for a set of concentric circles





- You can find some useful information in the AppNote under:

https://www.genisys-gmbh.com/applications/loops-variables-and-functions.html



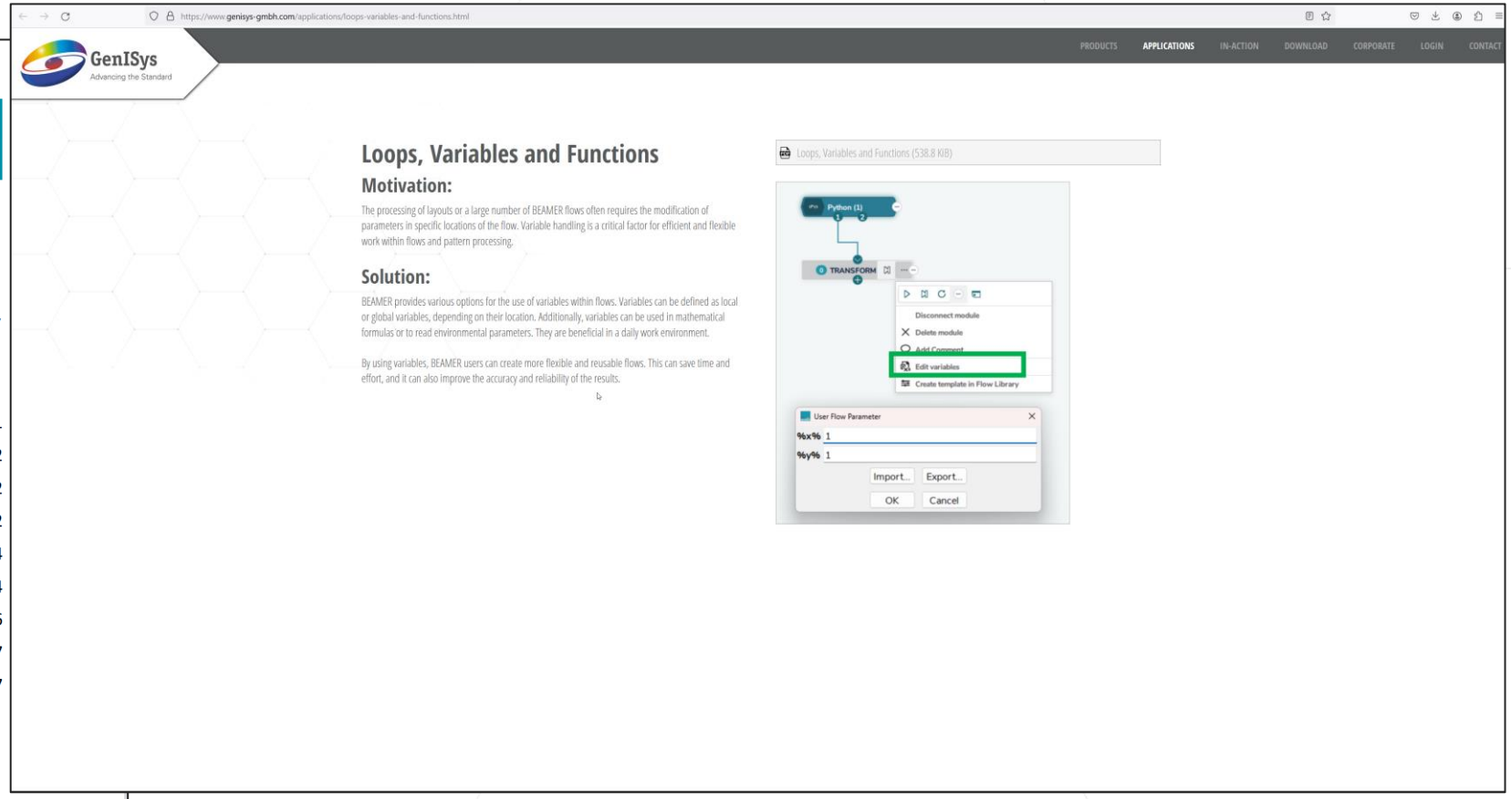
Advancing the Standard GenISys

BEAMER

Loops, Variables & Functions

Variable handling is a key factor for effective and flexible working inside of flows and doing pattern processing. With this note we will explore the various options inside BEAMER for variables and how they can help in a day-to-day work environment.

Content



Thank You!

support@genisys-gmbh.com

Headquarters
GenISys GmbH
Eschenstr. 66
D-82024 Taufkirchen (Munich)
GERMANY
+49-(0)89-3309197-60
+49-(0)89-3309197-61
info@genisys-gmbh.com

USA Office
GenISys Inc.
P.O. Box 410956
San Francisco, CA
94141-0956
USA
+1 (408) 353-3951
usa@genisys-gmbh.com

Japan / Asia Pacific Office
GenISys K.K.
German Industry Park
1-18-2 Hakusan Midori-ku
Yokohama 226-0006
JAPAN
+81 (0)45-530-3306
+81 (0)45-532-6933
apsales@genisys-gmbh.com

BEAMER LAB TRACER MASKER Pro SEM VIEWER