

---

# **TXLWriter Documentation**

***Release 1.0.0***

**Esteban Marin**

May 17, 2016



## CONTENTS

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	What does it do? . . . . .	3
1.2	Technical Information . . . . .	3
1.3	Example SVG Output . . . . .	3
<b>2</b>	<b>TXLWizard Example</b>	<b>5</b>
2.1	Code . . . . .	5
<b>3</b>	<b>TXLConverter</b>	<b>9</b>
<b>4</b>	<b>Python Module Reference</b>	<b>11</b>
<b>5</b>	<b>TXLWizard Advanced Example</b>	<b>13</b>
5.1	Code . . . . .	13
<b>6</b>	<b>Indices and tables</b>	<b>17</b>
	<b>Python Module Index</b>	<b>19</b>



Contents:



## INTRODUCTION

This document describes the usage and technical reference of the python program *TXLWizard* written by Esteban Marin ([estebanmarin@gmx.ch](mailto:estebanmarin@gmx.ch)).

### 1.1 What does it do?

The *TXLWizard* provides routines for generating TXL files (.txl) for the preparation of E-Beam lithography masks using python code. The TXL files can be processed with BEAMER. See the following links:

- <http://genisys-gmbh.com/web/products/beamer.html>
- [http://cad035.psi.ch/LB\\_index.html](http://cad035.psi.ch/LB_index.html)
- [http://cad035.psi.ch/LBDoc/BEAMER\\_Manual.pdf](http://cad035.psi.ch/LBDoc/BEAMER_Manual.pdf)

The generated TXL files are also converted to HTML / SVG for presentation in any modern browser or vector graphics application.

Moreover, a command line interface *TXLConverter* provides conversion of existing TXL files to HTML / SVG (See Section *TXLConverter*).

### 1.2 Technical Information

The “TXLWizard” is written in python and will run in Python version 2.7+ and 3.1+.

In order to use it, the *TXLWizard* package must be available as a python package, i.e. either it must be copied to

`Path_to_my_python_installation/site-packages/`

or to the path where your script is located.

Alternatively, you can also prepend the following command to your python script:

```
sys.path.append('path to the folder containing TXLWizard')
```

### 1.3 Example SVG Output

An example output can be seen here:

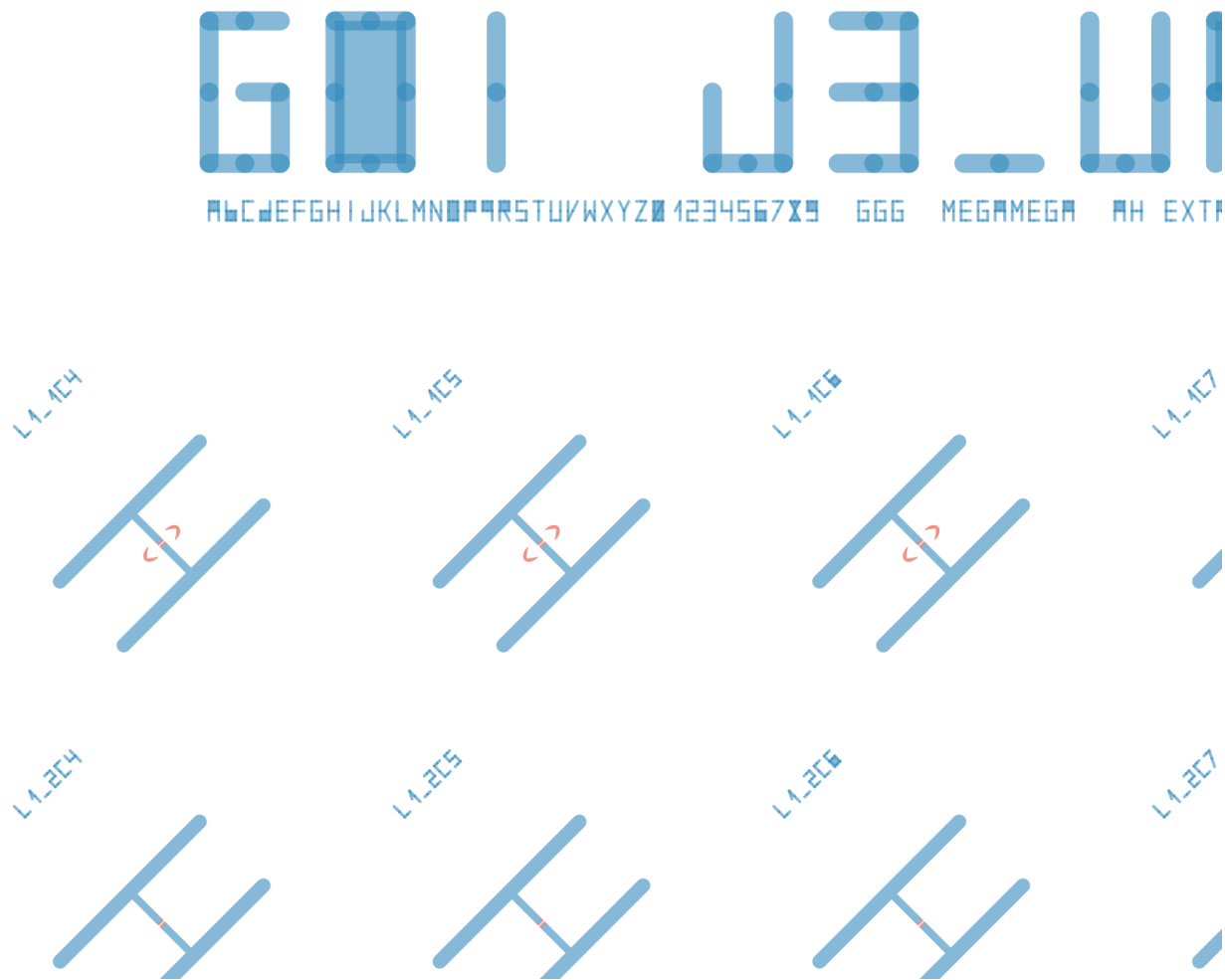


Fig. 1.1: Example SVG output for a mask



## TXLWIZARD EXAMPLE

The following code demonstrates a simple example usage of the *TXLWizard* for generating TXL files with python code.

The code can be found in the file `Content/Example_Simple.py`.

The resulting image is shown in Figure *Simple Example: Generated Mask*.

A more advanced example is shown in Section *TXLWizard Advanced Example*

### 2.1 Code

```
1 #####
2 # Import Libraries #
3 #####
4
5 # Import TXLWriter, the main class for generating TXL Output
6 import TXLWizard.TXLWriter
7
8 # Import Pre-Defined Shapes / Structures wrapped in functions
9 import TXLWizard.ShapeLibrary.EndpointDetectionWindows
10 import TXLWizard.ShapeLibrary.Label
11
12 # Import math module for calculations
13 import math
14
15
16 #####
17 # Sample / Structure Parameters #
18 #####
19
20 # Define all sample parameters
21 SampleParameters = {
22     'Width': 8e3,
23     'Height': 8e3,
24     'Label': 'Simple Demo',
25 }
26
27 # Define all structure parameters
28 StructureParameters = {
29     'Circle': {
30         'Radius': 50,
31         'Layer': 3
32     },
```

```
33     'CircleArray': {
34         'Columns': 6,
35         'Rows': 5,
36         'ArrayXOffset': 500,
37         'ArrayYOffset': -500,
38         'ArrayOrigin': [0.75e3, 3e3],
39         'Label': 'R{:d}C{:d}',
40     }
41 }
42
43
44 #####
45 # Initialize TXLWriter #
46 #####
47 TXLWriter = TXLWizard.TXLWriter.TXLWriter(
48     GridWidth=SampleParameters['Width'],
49     GridHeight=SampleParameters['Height']
50 )
51
52 #####
53 # Define Structures #
54 #####
55
56 ## Sample Label ##
57
58 # Give the sample a nice label
59 SampleLabelObject = TXLWizard.ShapeLibrary.Label.GetLabel(
60     TXLWriter,
61     SampleParameters['Label'],
62     OriginPoint=[
63         0.5e3, 1. * SampleParameters['Height'] / 2. - 500
64     ],
65     FontSize=150,
66     StrokeWidth=20,
67     RoundCaps=True, # Set to False to improve e-Beam performance
68     Layer=1
69 )
70
71
72 ## Endpoint Detection ##
73
74 # Use Pre-Defined Endpoint Detection Windows
75 TXLWizard.ShapeLibrary.EndpointDetectionWindows.GetEndpointDetectionWindows(
76     TXLWriter, Layer=1)
77
78 ## User Structure: Circle ##
79
80 # Create Definition Structure for Circle that will be reused
81 CircleStructure = TXLWriter.AddDefinitionStructure('Circle')
82 CircleStructure.AddPattern('Circle',
83     Center=[0, 0],
84     Radius=StructureParameters['Circle']['Radius'],
85     Layer=StructureParameters['Circle']['Layer']
86 )
87
88
89 # Create array of the definition structure above
90 CircleArray = TXLWriter.AddContentStructure('CircleArray')
```

```
91 CircleArray.AddPattern('Array',
92     ReferencedStructureID=CircleStructure.ID,
93     OriginPoint=StructureParameters['CircleArray']['ArrayOrigin'],
94     PositionDelta1=[
95         StructureParameters['CircleArray']['ArrayXOffset'], 0
96     ],
97     PositionDelta2=[
98         0, StructureParameters['CircleArray']['ArrayYOffset']
99     ],
100     Repetitions1=StructureParameters['CircleArray']['Columns'],
101     Repetitions2=StructureParameters['CircleArray']['Rows']
102 )
103
104
105
106 #####
107 # Generate Output Files #
108 #####
109
110 # Note: The suffix (.txl, .html, .svg) will be appended automatically
111 TXLWriter.GenerateFiles('Masks/Example_Simple')
112
```

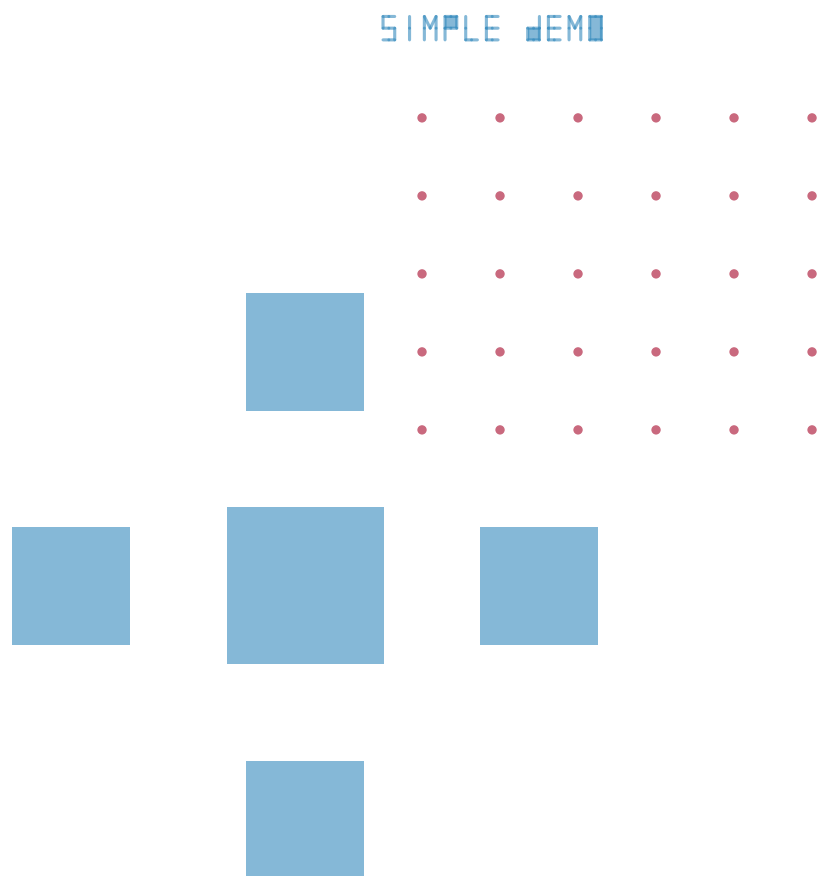


Fig. 2.1: Simple Example: Generated Mask

## TXLCONVERTER

blub



## PYTHON MODULE REFERENCE

Module *TXLWriter* contains the *TXLWizard*, *TXLWriter*, *TXLWriter* class

**class** *TXLWizard*.*TXLWriter*.**TXLWriter** (\*\*kwargs)  
Controller class for generating TXL / SVG / HTML output.

Here we can add structures (definitions and content) which will be rendered in the output. Optionally a coordinate system grid is drawn.

### Parameters

- **ShowGrid** (*bool*, *optional*) – Show the coordinate system grid or not.  
Defaults to True
- **GridWidth** (*int*, *optional*) – Full width of the coordinate system grid in um.  
Defaults to 800
- **GridHeight** (*int*, *optional*) – Full height of the coordinate system grid in um.  
Defaults to 800
- **GridSpacing** (*int*, *optional*) – Coordinate Sytem Grid Spacing in um.  
Defaults to 100
- **SubGridSpacing** (*int*, *optional*) – Coordinate System Sub-Grid Spacing in um.  
Defaults to 10

**AddContentStructure** (*Index*, \*\*kwargs)

Add content structure. A content structure can hold patterns that will render in the output.

A structure corresponds to the “STRUCT” command in the TXL file format.

### Parameters

- **Index** (*str*) – Unique identification of the structure. Must be used when referencing to this structure.
- **kwargs** (*dict*) – keyword arguments passed to the structure constructor

### Returns

**Return type** *TXLWizard*.*Patterns*.*Structure*.*Structure* structure instance

**AddDefinitionStructure** (*Index*, \*\*kwargs)

Add definition structure. A definition structure can be referenced by a content structure.

A structure corresponds to the “STRUCT” command in the TXL file format.

### Parameters

- **Index** (*str*) – Unique identification of the structure. Must be used when referencing to this structure.
- **kwargs** (*dict*) – keyword arguments passed to the structure constructor

#### Returns

**Return type** `TXLWizard.Patterns.Structure.Structure` structure instance

#### **AddHelperStructure** (*Index*, *\*\*kwargs*)

Add helper structure. Helper structures are only visible in the HTML / SVG Output.

A structure corresponds to the “STRUCT” command in the TXL file format.

#### Parameters

- **Index** (*str*) – Unique identification of the structure. Must be used when referencing to this structure.
- **kwargs** (*dict*) – keyword arguments passed to the structure constructor

#### Returns

**Return type** `TXLWizard.Patterns.Structure.Structure` structure instance

#### **GenerateFiles** (*Filename*, *TXL=True*, *SVG=True*, *HTML=True*)

Generate the output files (.txl, .svg, .html).

#### Parameters

- **Filename** (*str*) – Path / Filename without extension. The corresponding path will be created if it does not exist
- **TXL** (*Optional[bool]*) – Enable TXL Output
- **SVG** (*Optional[bool]*) – Enable SVG Output
- **HTML** (*Optional[bool]*) – Enable HTML Output



## TXLWIZARD ADVANCED EXAMPLE

The following code demonstrates an advanced example usage of the *TXLWizard* for generating TXL files with python code.

The code can be found in the file `Content/Example_Advanced.py`.

The resulting image is shown in Figure *Simple Example: Generated Mask*.

### 5.1 Code

```
1 #####
2 # Import Libraries #
3 #####
4
5 # Import TXLWriter, the main class for generating TXL Output
6 import TXLWizard.TXLWriter
7
8 # Import Pre-Defined Shapes / Structures wrapped in functions
9 import TXLWizard.ShapeLibrary.EndpointDetectionWindows
10 import TXLWizard.ShapeLibrary.Markers
11 import TXLWizard.ShapeLibrary.Label
12 import TXLWizard.ShapeLibrary.CornerCube
13
14 # Import math module for calculations
15 import math
16
17
18 #####
19 # Sample / Structure Parameters #
20 #####
21
22 # Define all sample parameters
23 SampleParameters = {
24     'Width': 8e3,
25     'Height': 8e3,
26     'Label': 'GOI Demo CornerCube',
27 }
28
29 # Define all structure parameters
30 StructureParameters = {
31     'CornerCube': {
32         'BridgeLength': 8,
33         'ParabolaFocus': 9,
34         'XCutoff': 9,
```

```
35     'AirGapX': 3,
36     'AirGapY': 1,
37     'LabelXOffset': 0,
38     'LabelYOffset': 50,
39     'Label': 'R{:d}C{:d}', # {:d} will be replaced
40                           # by str.format() with the corresponding row / column
41     'Layer': 2
42 },
43 'Circle': {
44     'Radius': 5,
45     'Layer': 3
46 },
47 'CornerCubeArray': {
48     'Columns': 6,
49     'Rows': 5,
50     'ArrayXOffset': 500,
51     'ArrayYOffset': -500,
52     'ArrayOrigin': [0.75e3, 3e3]
53 }
54 }
55
56 #####
57 # Initialize TXLWriter #
58 #####
59 TXLWriter = TXLWizard.TXLWriter.TXLWriter(
60     GridWidth=SampleParameters['Width'],
61     GridHeight=SampleParameters['Height']
62 )
63
64 #####
65 # Define Structures #
66 #####
67
68 ## Sample Label ##
69
70 # Give the sample a nice label...
71 SampleLabelObject = TXLWizard.ShapeLibrary.Label.GetLabel(
72     TXLWriter,
73     SampleParameters['Label'],
74     OriginPoint=[
75         0.5e3, 1. * SampleParameters['Height'] / 2. - 500
76     ],
77     FontSize=150,
78     StrokeWidth=20,
79     RoundCaps=True, # Set to False to improve e-Beam performance
80     Layer=1
81 )
82
83 # ...and some other information
84 Alphabet = TXLWizard.ShapeLibrary.Label.GetLabel(
85     TXLWriter,
86     'abcdefghijklmnopqrstuvwxyz0123456789 megamega ggg ah extraaaa rischaaar',
87     OriginPoint=[
88         0.5e3, 1. * SampleParameters['Height'] / 2. - 600
89     ],
90     FontSize=50,
91     StrokeWidth=3,
92     RoundCaps=True, # Set to False to improve e-Beam performance
```

```

93     Layer=1
94 )
95
96 ## Endpoint Detection ##
97
98 # Use Pre-Defined Endpoint Detection Windows
99 TXLWizard.ShapeLibrary.EndpointDetectionWindows.GetEndpointDetectionWindows (
100     TXLWriter, Layer=1)
101
102 ## Alignment Markers ##
103
104 # Use Pre-Defined Alignment Markers
105 TXLWizard.ShapeLibrary.Markers.GetMarkers (
106     TXLWriter, Layer=1)
107
108
109 ## User Structure: Corner Cube ##
110
111 # Create Definition Structure for Corner Cube that will be reused
112 CornerCubeDefinition = TXLWizard.ShapeLibrary.CornerCube.GetCornerCube (
113     TXLWriter,
114     ParabolaFocus=StructureParameters['CornerCube']['ParabolaFocus'],
115     XCutoff=StructureParameters['CornerCube']['XCutoff'],
116     AirGapX=StructureParameters['CornerCube']['AirGapX'],
117     AirGapY=StructureParameters['CornerCube']['AirGapY'],
118     Layer=StructureParameters['CornerCube']['Layer']
119 )
120
121 # Create Definition Structure for combination of cornercube and additional circle
122 FullCornerCubeNoRotation = TXLWriter.AddDefinitionStructure('FullCornerCubeNoRotation')
123 FullCornerCubeNoRotation.AddPattern('Reference',
124     ReferencedStructureID=CornerCubeDefinition.ID,
125     OriginPoint=[1. * StructureParameters['CornerCube']['BridgeLength'] / 2., 0]
126 )
127 FullCornerCubeNoRotation.AddPattern('Circle',
128     Center=[0, 0],
129     Radius=StructureParameters['Circle']['Radius'],
130     Layer=StructureParameters['Circle']['Layer']
131 )
132
133 # Create definition structure with rotation of entire referenced structure
134 FullCornerCube = TXLWriter.AddDefinitionStructure('FullCornerCube',
135     RotationAngle=45)
136 FullCornerCube.AddPattern('Reference',
137     ReferencedStructureID=FullCornerCubeNoRotation.ID,
138     OriginPoint=[0, 0]
139 )
140
141 # Create array of the definition structure above
142 CornerCubeArrayFine = TXLWriter.AddContentStructure('CornerCubeArrayFine')
143 CornerCubeArrayFine.AddPattern('Array',
144     ReferencedStructureID=FullCornerCube.ID,
145     OriginPoint=StructureParameters['CornerCubeArray']['ArrayOrigin'],
146     PositionDelta1=[
147         StructureParameters['CornerCubeArray']['ArrayXOffset'], 0
148     ],
149     PositionDelta2=[
150         0, StructureParameters['CornerCubeArray']['ArrayYOffset']

```

```

151 ],
152 Repetitions1=StructureParameters['CornerCubeArray']['Columns'],
153 Repetitions2=StructureParameters['CornerCubeArray']['Rows']
154 )
155
156
157 # Add Labels to each array element
158 for Row in range(1, StructureParameters['CornerCubeArray']['Rows'] + 1):
159     for Column in range(1, StructureParameters['CornerCubeArray']['Columns'] + 1):
160         RowColumnCountLabel = TXLWizard.ShapeLibrary.Label.GetLabel(
161             TXLWriter,
162             StructureParameters['CornerCube']['Label'].format(Row, Column),
163             OriginPoint=[
164                 StructureParameters['CornerCubeArray']['ArrayOrigin'][0]
165                 + StructureParameters['CornerCubeArray']['ArrayXOffset']
166                 * (Column - 1) + StructureParameters['CornerCube']['LabelXOffset'],
167                 StructureParameters['CornerCubeArray']['ArrayOrigin'][1]
168                 + StructureParameters['CornerCubeArray']['ArrayYOffset']
169                 * (Row - 1) + StructureParameters['CornerCube']['LabelYOffset']],
170             FontSize=16,
171             StrokeWidth=3,
172             RoundCaps=True, # Set to False to improve e-Beam performance
173             Layer=1,
174             RotationAngle=45
175         )
176
177
178 #####
179 # Generate Output Files #
180 #####
181
182 # Note: The suffix (.txl, .html, .svg) will be appended automatically
183 TXLWriter.GenerateFiles('Masks/Example_Advanced')
184

```

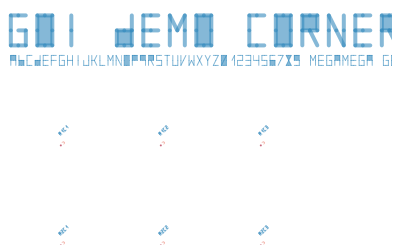


Fig. 5.1: Simple Example: Generated Mask

## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



**t**

TXLWizard.TXLWriter, [11](#)