Luminis

configuration manual

Table of content

1 Hardware configuration

Hardware configuration is done through the “HARDWARE_CONFIG.xml” file.

1.1 Machine specification

The first thing to include in “HARDWARE_CONFIG.xml” is the definition of the machine. To define a machine create a **<Machine>** element. As all other devices, a machine is defined by its model. The model of a device is defined as an attribute of its element. In Snippet 1, the machine’s model is “MicroFab”. The **<Machine>** element contains a list of device elements, defining which model will be loaded for a given required device. For example, for the required device **<LargeStageZ>** the loaded model will be “SmartActMCS2” in line 6 of Snippet 1. All device model referenced in the machine configuration must be fully defined bellow the **<Machine>** element. If a device isn’t required for a machine it should be removed from the **<Machine>** element and its definition should be removed.

```xml
1 <Machine Model="MicroFab">
2         <Laser>NIDaqMX</Laser>
3         <Camera>IDS uEye</Camera>
4         <Stage>NIDaqMX</Stage>
5         <LargeStageXY>ASI MS2000</LargeStageXY>
6         <LargeStageZ>SmartActMCS2</LargeStageZ>
7         <LightLogo>NIDaqMX</LightLogo>
8 </Machine>
```

Snippet 1: XML machine element for µFab-3d with XYZ large stage option

1.2 Fabrication stage

1.2.1 Analog piezo controlled via NI Daq MX

To configure fabrication stage to NI Daq MX analog piezo controller, create a **<Stage>** element and set its model to “NIDaqMX”. The code Snippet 2 shows an example of such a configuration. In order to connect to the NI Daq board, its **<Name>** must be defined (line 4). The board name can be retrieved and set from NI MAX.

Each axis of the piezo is mapped to an analog output on the board. The outputs are defined by the **<XAxisPin>**, **<YaxisPin>** and **<ZaxisPin>** elements (lines 5,6,7). Pin names are found in the board’s documentation. See pin layout documentation for NI Daq model USB-4363 in appendix 1. At line 8 of Snippet 2, the **<MaxVelocity>** element allows to set a software limit on the maximum speed attainable by the stage in mm/s.

```xml
1 <Stage Model="NIDaqMX">
2       <Name Unit="none" Type="String"> Fabrication Stage </Name>
3       <Range Unit="mm" Type="Vector3"> (300;300;300) </Range>
4       <DaqName Unit="none" Type="String"> Dev1 </DaqName>
5       <XAxisPin Unit="none" Type="String"> ao0 </XAxisPin>
6       <YAxisPin Unit="none" Type="String"> ao1 </YAxisPin>
7       <ZAxisPin Unit="none" Type="String"> ao2 </ZAxisPin>
8       <MaxVelocity Unit="mm/s" Type="Single"> 0.7 </MaxVelocity>
9       <LaserDelay Unit="s" Type="Single"> (0.015;0.015;0.015) </LaserDelay>
10      <StabilisationDelay Unit="s" Type="Single"> (0.03;0.03;0.03) </StabilisationDelay>
11 </Stage>
```

Snippet 2: Analog piezo controlled via NI Daq MX XML configuration

Finally **<LaserDelay>** (line 9) and **<StabilisationDelay>** (line 10) are used to calibrate the piezo stage in relation to the laser. They are expressed in seconds (s). The laser delay is used to delay laser activation at the start of trajectories and keep it on longer at the end. It is used to compensate stage accelerations and deceleration. The stabilization delay is added to force the stage to stay in position at the end of trajectories. Laser and stabilization delays are defined are three dimensional vectors, defining delays for each axis separately.

In order to set these delays correctly, a simple grid task, shown in Figure 1, can be used. Each segment of the grid is fabricated starting from the inverse direction of the previous one. If the first one is fabricated starting at the bottom, the second one should be fabricated starting from the top. Figure 1 shows an example a fabrication result for a correctly calibrated stage.

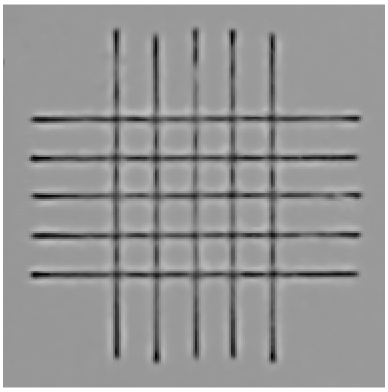


Figure 1: Fabrication result for correct laser and stabilization delays

If the laser delay is insufficient, segments starts will be elongated as seen in Figure 2. On the contrary, if the laser delay is excessive, segments starts will be too short and ends will be over polymerized (see Figure 3).

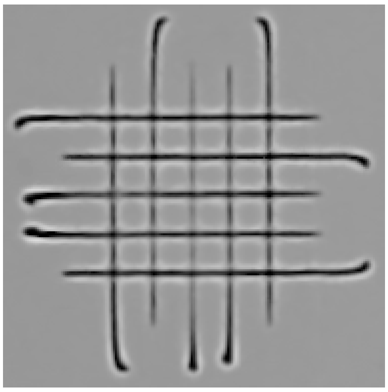


Figure 2: Fabrication result for correct stabilization delay and insufficient laser delay

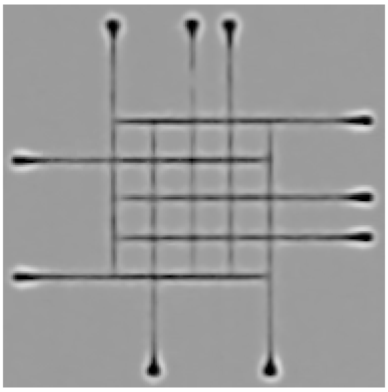


Figure 3: Fabrication result for correct stabilization delay and excessive laser delay

If stabilization delay is insufficient, segments starts will have the correct length but will be bent as show in Figure 4. Nothing notable is present on fabrication results of an excessive stabilization delay but it will increase fabrication times. The stabilization delay must therefor be set to it’s lowest correct value to optimize fabrication times.

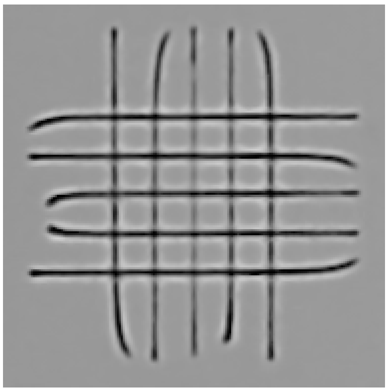


Figure 4: Fabrication result for correct laser delay and insufficient stabilization delay

1.3 Large stage

1.3.1 ASI XY stage with MS2000/Tiger controller

The ASI XY large stage communication is done over serial. A port name and a baud rate thus needs to be defined in the configuration. In the example configuration of Snippet 3, **<PortName>** and **<BaudRate>** define the port name and baud rate respectively. The baud rate needs to be defined to 9600 for MS2000 controllers and to 115200 for Tiger controllers. The port name can be found in the Windows device manager. Only the port number prefixed by “COM” is required for **<PortName>** configuration.

The range margin, defined at line 5 of code Snippet 3, is used to reduce the attainable positions range expressed in micro-meters (µm). This allows to reduce range inconsistencies introduced by physical stops. Stage ranges will be reduced on all axis by the amount defined in the **<RangeMargin>** element.

The position refresh delay defined at line 6 sets the time, in milliseconds (ms), between two fetch of the position from the controller. The smaller this value is, the smoother the user experience will be but the more computing resources it will require.

The next elements from line 7 to 12 in Snippet 3 are controller parameters used to fine tune stage performances balance:

- **<ProportionalError>** sets the servo proportional error term constant (more information at https://www.asiimaging.com/docs/commands/kp);
- **<DriftError>** sets the drift error in millimeters (mm) (more information at https://www.asiimaging.com/docs/commands/error);
- **<PositioningError>** Set the positionning error in millimeters (mm) (mode information at https://www.asiimaging.com/docs/commands/pcros);
- **<Backlash>** Set anti-backlash distance in millimeters (mm) (mode information at https://www.asiimaging.com/docs/commands/backlash);
- **<MaxVelocity>** Limits the stage’s maximum velocity in millimeters per seconds (mm/s);
- **<MaxAcceleration>** Limits the stage’s maximum acceleration in millimeters per seconds (mm/s);

```
1  <LargeStageXY Model="ASI MS2000">
2      <Name Unit="none" Type="String"> Large stage </Name>
3      <PortName Unit="none" Type="String">COM4</PortName>
4      <BaudRate Unit="none" Type="Int32"> 9600 </BaudRate>
5      <RangeMargin Unit="µm" Type="Single"> 1000 </RangeMargin>
6      <PositionRefreshDelay Unit="ms" Type="Int32"> 40 </PositionRefreshDelay>
7      <ProportionalError Unit="none" Type="Vector2"> (40;40) </ProportionalError>
8      <DriftError Unit="mm" Type="Vector2"> (0.0001;0.0001) </DriftError>
9      <PositioningError Unit="mm" Type="Vector2"> (0.000011;0.000011) </PositioningError>
10     <Backlash Unit="mm" Type="Vector2"> (0.004;0.004) </Backlash>
11     <MaxVelocity Unit="mm/s" Type="Single"> 3 </MaxVelocity>
12     <MaxAcceleration Unit="mm/s" Type="Single"> 3 </MaxAcceleration>
13 </LargeStageXY>
```

Snippet 3: ASI XY stage configuration for MS2000 controllers

As shown in Figure 5, the ASI MS-2000 controller displays 8 physical switches at its back. These switches control the activation of some controller options. These options are detailed here: “*https://www.asiimaging.com/docs/ms2000_operation?s[]=switch#dip_switch_settings*”. For µFAB-3D systems configuration these switches need to be, from left to right when facing the controller’s back panel: **00011100**. Where 0 represents a switch in “down” position and 1 one in “up” position.

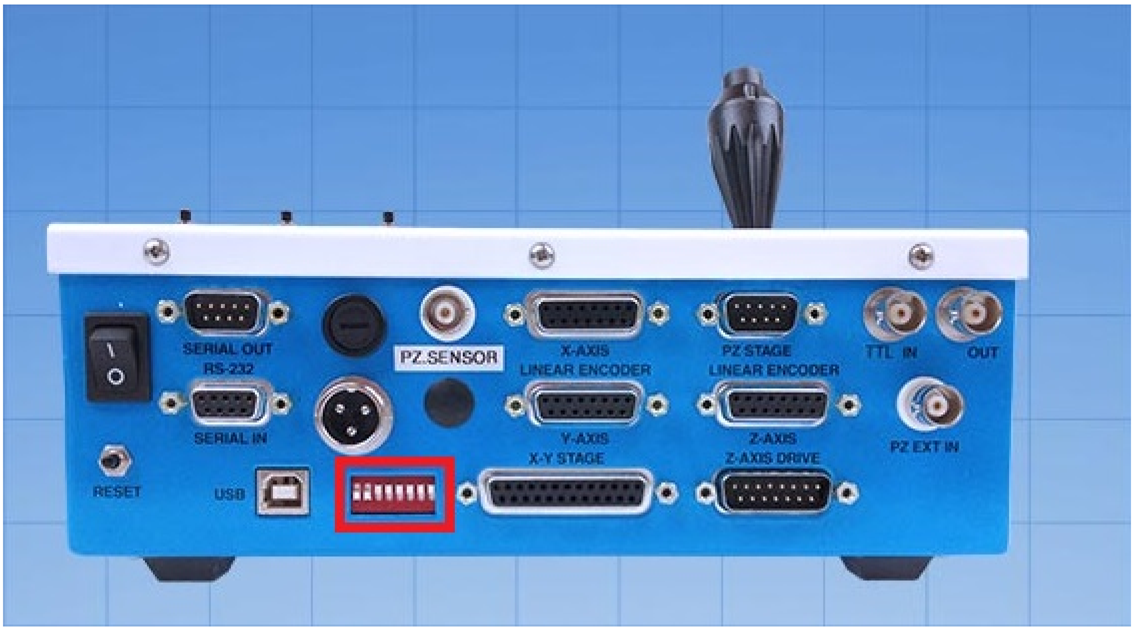


Figure 5: ASI MS-2000 controller switchs location

1.3.2 SmarAct Z stage using MCS2 controller

This stage's model must be defined as "SmartActMCS2". A shown in line 3 of code Snippet 4, it must contain a **<DeviceName>** element which is used for connection. MCS2 controllers can support multiple channels (i.e. axis). Similarly to selecting axis board pin in NI Daq piezo controller, the correct channel for the Z axis must be define in the **<Zchannel>** element (Snippet 4, line 4). Axis are defined by their index. Indices start from 0. Channel 1 is thus defined by index 0, channel 2 by index 1 and so on.

Range margin and position refresh delays are carried over from ASI stage configuration and thus serve the same purpose (see section 1.3.1). The **<MaxVelocity>** element at line 6 allows to set a software limit on the maximum speed attainable by the stage, in millimeters per second (mm/s).

```
1 <LargeStageZ Model="SmartActMCS2">
2         <Name Unit="none" Type="String"> Z Large stage </Name>
3         <DeviceName Unit="none" Type="String">usb:sn:MCS2-00005805</DeviceName>
4         <ZChannel Unit="none" Type="Int32"> 2 </ZChannel>
5         <RangeMargin Unit="µm" Type="Single"> 1000 </RangeMargin>
6         <MaxVelocity Unit="mm/s" Type="Single"> 1 </MaxVelocity>
7         <PositionRefreshDelay Unit="ms" Type="Int32"> 100 </PositionRefreshDelay>
8 </LargeStageZ>
```

Snippet 4: SmarAct Z stage using MCS2 controller configuration

If the SmartAct Z stage axis doesn't move in the correct direction in Luminis, invert it using the MCS2ServiceTool. The service tool can be generally found in **"C:\SmarAct\MCS2\Programs"**. As shown in Figure 6, connect to the SmartAct stage, go to the "Module Parameters" tab and change the "Logical Scale Inversion" to the unselect option (generally "Inverted").

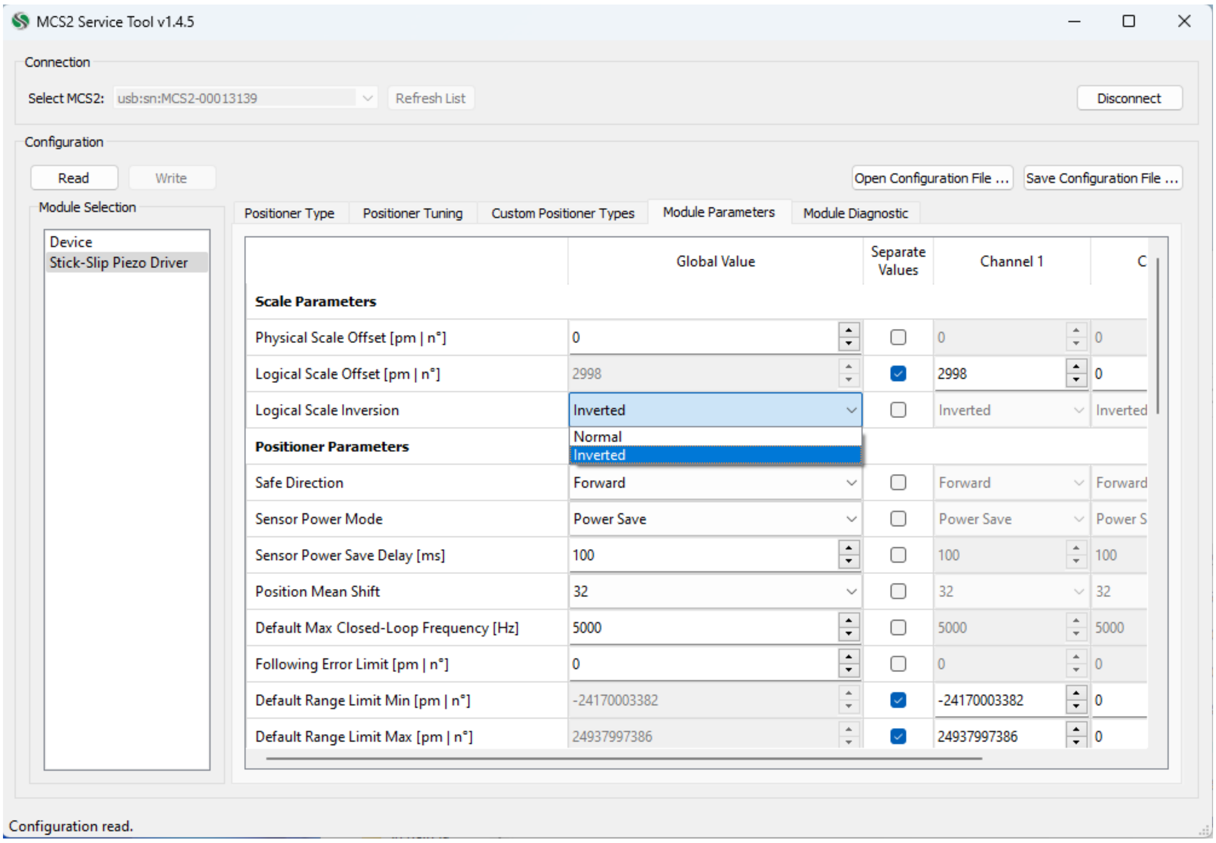


Figure 6: Logical scale inversion option in the MCS2 service tool

1.4 Laser

1.4.1 NI Daq MX controller for analog laser control

Similarly to the NI Daq piezo (see section 1.2.1), the NI Daq laser configuration requires the name of the NI Daq board for connection defined by the **<DaqName>** element at line 5 of Snippet 5 as well as an analog output pin defined by the **<Pin>** element at line 6 of Snippet 5. The configuration also includes a max power software limit in percent (%) and the wavelength of the laser in nanometer (nm).

```
1 <Laser Model="NIDaqMX">
2       <Name Unit="none" Type="String"> Green laser (λ=532nm) </Name>
3       <WaveLength Unit="nm" Type="Single"> 532 </WaveLength>
4       <MaxPower Unit="percent" Type="Single"> 100 </MaxPower>
5       <Precision Unit="percent" Type="Single"> 0 </Precision>
6       <DaqName Unit="none" Type="String"> Dev1 </DaqName>
7       <Pin Unit="none" Type="String"> ao3 </Pin>
8 </Laser>
```

Snippet 5: NI Daq MX controller for analog laser configuration

1.5 Camera

1.5.1 IDS Camera

IDS cameras can be controlled using two different IDS drivers: the legacy uEye driver and the GeniCam driver. They are differentiated in the configuration by their model: the legacy uEye driver's model is "IDS uEye" (line 1 of Snippet 6) whereas the GeniCam driver's is "IDS".

Regardless of the driver used, the camera name, defined in the <Name> element, is required to connect to the camera. It acts like an identifier in case there are multiple IDS cameras connected to the PC. Camera name can be retrieved in IDS peak Cockpit or IDS uEye software.

IDS camera configurations define the maximum exposure time supported by the camera in milliseconds. As in line 3 of code Snippet 6, maximum exposure time is defined using an **<ExposureMax>** element. Bellow the **<ExposureMax>** element in Snippet 6 is an **<IsColor>** element. This element defines if the camera supports color.

Elements **<MicronPerPixelObj>** on lines 5 through 9 of Snippet 6 define the number of microns per pixel for each objective. Microns per pixels are used to display the camera overlay. They need to be calibrated for each objective defined in "SLICING_PRESETS.xml" in the same order. See section 2.2 for details on objective configuration.

The next required elements of any camera configurations are **<TargetPosition>** and **<FocusModeZoom>**. The target position element defines where the laser target should be located on the camera view on application startup. Coordinates are in pixel, the origin being on the top-left of the camera image. The focus mode zoom is the amount of zoom applied when switching to focus mode.

Lastly, the four elements from line 12 to 15 of Snippet 6 define default exposures and gains in different camera modes:

- **<AlignmentModeExposure>** defines exposure value in percents (%) in alignment mode;
- **<AlignmentModeGain>** defines gain value in percents (%) in alignment mode;
- **<FocusModeExposure>** defines exposure value in percents (%) in focus mode;
- **<FocusModeGain>** defines gain value in percents (%) in focus mode;

```xml
<Camera Model="IDS uEye">
	<Name Unit="none" Type="String">UI154xLE-M</Name>
	<ExposureMax Unit="µs" Type="Single"> 38000 </ExposureMax>
	<IsColor Unit="none" Type="Boolean"> false </IsColor>
	<MicronsPerPixelObj1 Unit="µm" Type="Single"> 0.05172413793 </MicronsPerPixelObj1>
	<MicronsPerPixelObj2 Unit="µm" Type="Single"> 0.13054830287 </MicronsPerPixelObj2>
	<MicronsPerPixelObj3 Unit="µm" Type="Single"> 0.25862068965 </MicronsPerPixelObj3>
	<MicronsPerPixelObj4 Unit="µm" Type="Single"> 0.25862068965 </MicronsPerPixelObj4>
	<MicronsPerPixelObj5 Unit="µm" Type="Single"> 0.25862068965 </MicronsPerPixelObj5>
	<TargetPosition Unit="px" Type="Vector2"> (622;490) </TargetPosition>
	<FocusModeZoom Unit="%" Type="Single"> 300 </FocusModeZoom>
	<AlignmentModeExposure Unit="%" Type="Single"> 4 </AlignmentModeExposure>
	<AlignmentModeGain Unit="%" Type="Single"> 0 </AlignmentModeGain>
	<FocusModeExposure Unit="%" Type="Single"> 100 </FocusModeExposure>
	<FocusModeGain Unit="%" Type="Single"> 20 </FocusModeGain>
</Camera>
```

Snippet 6: Example configuration for an IDS UI154xLE-M camera using legacy "uEye" driver

2 Slicing configuration

Slicing configuration is done through the “SLICING_PRESETS.xml” file.

2.1 Resist thickness

Default resist thickness is used in fabrications in “UPSIDE DOWN” mode. It defines the expected resist thickness the laser will traverse before attaining the focus plane. It can be set from the <General> element using a <DefaultResistThickness> element (see Snippet 7).

```xml
<General>
	<DefaultResistThickness Unit="µm" Type="Int32"> 500 </DefaultResistThickness>
</General>
```

Snippet 7: Example of default resist thickness definition

2.2 Objective definition

To define an objective, it must be added to “SLICING_PRESETS.xml” using a <Objective> element (see Snippet 8). The objective name is defined in its “Model” attribute. The objective defined in Snippet 8 is named "Zeiss 100X/1.25".

Under the <Objective> element must be added three elements:

- <Aperture> defining the objective’s numerical aperture (Snippet 8 line 2);
- <IsImmersion> defining weather or not the objective is oil immersive (Snippet 8 line 3);
- <WorkingDistance> defining the working distance of the objective (Snippet 8 line 4).

```xml
<Objective Model="Zeiss 100X/1.25">
	<Aperture Unit="none" Type="Single"> 1.25 </Aperture>
	<IsImmersion Unit="none" Type="Boolean"> True </IsImmersion>
	<WorkingDistance Unit="µm" Type="Single"> 220 </WorkingDistance>
</Objective>
```

Snippet 8: Example of single objective definition

The default objective set on application start can be changed in the **<General>** element using a **<DefaultObjectiveIndex>** element, defining the default objective’s index. Indices are defined by the order of definition of the objectives and start from zero. In example Snippet 9, the first objective defined is set as the default startup objective.

```xml
<General>
	<DefaultObjectiveIndex Unit="none" Type="Int32"> 0 </DefaultObjectiveIndex>
</General>
```

Snippet 9: Example of default objective definition

2.3 Resist definition

To define a resist profile, a **<Resist>** element needs to be added to “SLICING_PRESETS.xml” as shown in Snippet 10. The resist name is defined in a “Model” attribute of the **<Resist>** element. Under this element must be defined a **<RefractiveIndex>** element (Snippet 10 line 2), containing the resist refractive index, and several **<LaserCurveElement>** elements (Snippet 10 line 4, 5).

These elements together define a laser power curve giving a power offset to be applied at a given elevation during fabrication. Laser curve elements contain two dimentional vectors acting as a point in the laser curve. Each vector X component represents a position in µm and its Y component represents a laser power offset. Points are interpolated and extrapolated on edges, creating a piece-wise affine function.

```xml
1 <Resist Model="Unknown">
2 	<RefractiveIndex Unit="none" Type="Single"> 1.49 </RefractiveIndex>
3 	<!-- Laser curve f(position (um)) = laser power (%) -->
4 	<LaserCurveElement Unit="none" Type="Vector2"> (0;0) </LaserCurveElement>
5 	<LaserCurveElement Unit="none" Type="Vector2"> (450;0) </LaserCurveElement>
6 </Resist>
```

Snippet 10: Example of default resist definition

The default resist set on application start can be changed in the **<General>** element using a **<DefaultResistIndex>** element, defining the default resist’s index. Indices are defined by the order of definition of the resists and start from zero. In example Snippet 11, the first resist defined is set as the default startup resist.

```
<General>
        <DefaultResistIndex Unit="none" Type="Int32"> 0 </DefaultResistIndex>
</General>
```

Snippet 11: Example of resist profil definition

2.4 Voxel size

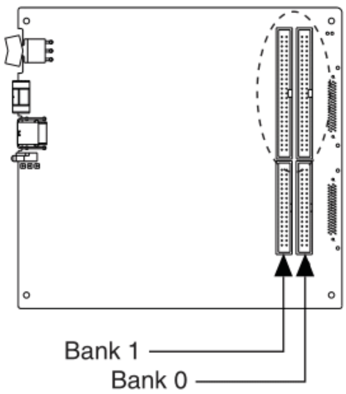
Empirical voxel size data doesn't need to be changed with every machine. That being said, if new empirical data is collected it can be updated in "SLICING_PRESETS.xml". This model links print qualities (highest, high, standard, low, lowest) and objective apertures to a voxel size. It is organized as follows: an element is created for each print quality: **<Highest>**, **<High>**, **<Standard>**, **<Low>** and **<Lowest>**. In each of these elements an element is created for each numerical aperture, currently: **<Na1.25>**, **<Na0.95>**, **<Na0.8>** and **<Na0.5>**. In these last elements is stored the voxel's horizontal V_x and vertical V_y sizes in the form "(V_x;V_y)". Voxels sizes are represented in micro-meter (µm).

```
<Highest>
        <Na1.25 Unit="micron" Type="Vector2"> (0.2;0.5) </Na1.25>
        <Na0.95 Unit="micron" Type="Vector2"> (0.22;0.97) </Na0.95>
        <Na0.8 Unit="micron" Type="Vector2"> (0.25;1.42) </Na0.8>
        <Na0.5 Unit="micron" Type="Vector2"> (0.4;4) </Na0.5>
</Highest>
<High>
        <Na1.25 Unit="micron" Type="Vector2"> (0.27;0.67) </Na1.25>
        <Na0.95 Unit="micron" Type="Vector2"> (0.28;1.25) </Na0.95>
        <Na0.8 Unit="micron" Type="Vector2"> (0.5;2.83) </Na0.8>
        <Na0.5 Unit="micron" Type="Vector2"> (0.83;8.3) </Na0.5>
</High>
<Standard>
        <Na1.25 Unit="micron" Type="Vector2"> (0.4;1) </Na1.25>
        <Na0.95 Unit="micron" Type="Vector2"> (0.43;1.93) </Na0.95>
        <Na0.8 Unit="micron" Type="Vector2"> (1.25;7) </Na0.8>
        <Na0.5 Unit="micron" Type="Vector2"> (1.67;16.67) </Na0.5>
</Standard>
<Low>
        <Na1.25 Unit="micron" Type="Vector2"> (0.53;1.33) </Na1.25>
        <Na0.95 Unit="micron" Type="Vector2"> (0.9;3.9) </Na0.95>
        <Na0.8 Unit="micron" Type="Vector2"> (2.5;14) </Na0.8>
        <Na0.5 Unit="micron" Type="Vector2"> (3;30) </Na0.5>
</Low>
<Lowest>
        <Na1.25 Unit="micron" Type="Vector2"> (0.8;2) </Na1.25>
        <Na0.95 Unit="micron" Type="Vector2"> (1.70;7.73) </Na0.95>
        <Na0.8 Unit="micron" Type="Vector2"> (5;28.3) </Na0.8>
        <Na0.5 Unit="micron" Type="Vector2"> (5;50) </Na0.5>
</Lowest>
```

Snippet 12: Slicing empirical data configuration

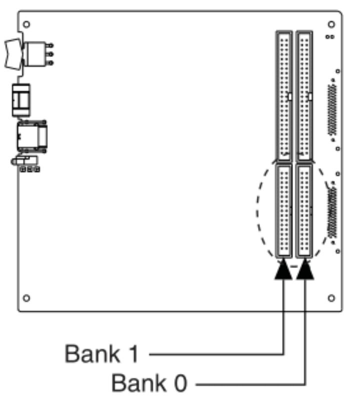
3 Annexe

50-Pin Digital Connectors



Bank 1				Bank 0			
+5 V	50	49	+5 V	+5 V	50	49	+5 V
D GND	48	47	P0.31	D GND	48	47	PFI 15
D GND	46	45	P0.30	D GND	46	45	PFI 14
D GND	44	43	P0.29	D GND	44	43	PFI 13
D GND	42	41	P0.28	D GND	42	41	PFI 12
D GND	40	39	P0.27	D GND	40	39	PFI 11
D GND	38	37	P0.26	D GND	38	37	PFI 10
D GND	36	35	P0.25	D GND	36	35	PFI 9
D GND	34	33	P0.24	D GND	34	33	PFI 8
D GND	32	31	P0.23	D GND	32	31	PFI 7
D GND	30	29	P0.22	D GND	30	29	PFI 6
D GND	28	27	P0.21	D GND	28	27	PFI 5
D GND	26	25	P0.20	D GND	26	25	PFI 4
D GND	24	23	P0.19	D GND	24	23	PFI 3
D GND	22	21	P0.18	D GND	22	21	PFI 2
D GND	20	19	P0.17	D GND	20	19	PFI 1
D GND	18	17	P0.16	D GND	18	17	PFI 0
D GND	16	15	P0.15	D GND	16	15	P0.7
D GND	14	13	P0.14	D GND	14	13	P0.6
D GND	12	11	P0.13	D GND	12	11	P0.5
D GND	10	9	P0.12	D GND	10	9	P0.4
D GND	8	7	P0.11	D GND	8	7	P0.3
D GND	6	5	P0.10	D GND	6	5	P0.2
D GND	4	3	P0.9	D GND	4	3	P0.1
D GND	2	1	P0.8	D GND	2	1	P0.0

34-Pin Analog Connectors



Bank 1				Bank 0			
AI GND	34	33	AI 31 (AI 23–)	AI GND	34	33	AI 15 (AI 7–)
AI 23 (AI 23+)	32	31	AI GND	AI 7 (AI 7+)	32	31	AI GND
AI 30 (AI 22–)	30	29	AI 22 (AI 22+)	AI 14 (AI 6–)	30	29	AI 6 (AI 6+)
AI GND	28	27	AI 29 (AI 21–)	AI GND	28	27	AI 13 (AI 5–)
AI 21 (AI 21+)	26	25	AI GND	AI 5 (AI 5+)	26	25	AI GND
AI 28 (AI 20–)	24	23	AI 20 (AI 20+)	AI 12 (AI 4–)	24	23	AI 4 (AI 4+)
AI GND	22	21	AI 27 (AI 19–)	AI GND	22	21	AI 11 (AI 3–)
AI 19 (AI 19+)	20	19	AI GND	AI 3 (AI 3+)	20	19	AI GND
AI 26 (AI 18–)	18	17	AI 18 (AI 18+)	AI 10 (AI 2–)	18	17	AI 2 (AI 2+)
AI GND	16	15	AI 25 (AI 17–)	AI GND	16	15	AI 9 (AI 1–)
AI 17 (AI 17+)	14	13	AI GND	AI 1 (AI 1+)	14	13	AI GND
AI 24 (AI 16–)	12	11	AI 16 (AI 16+)	AI 8 (AI 0–)	12	11	AI 0 (AI 0+)
AI GND	10	9	AI SENSE 2	AI GND	10	9	AI SENSE
AI GND	8	7	NC	AI GND	8	7	NC
AI GND	6	5	AO GND	AI GND	6	5	AO GND
AO 3	4	3	AO GND	AO 1	4	3	AO GND
AO 2	2	1	AO GND	AO 0	2	1	AO GND

NC = No Connect   NC = No Connect

Annexe 1: NI Daq USB-6343 model pin layout documentation