Resonance Scanner in Scope

Using the software

Main page

Resolution: The possible resolution values in X direction are different from the ones in Y direction, because a resonance scanner is used only for the X direction.
The X resolution is calculated as

$$\frac{1}{t_p \cdot 2 \cdot f_{resonance}}$$

where

$f_{resonance}$	resonance frequency of the resonance scanner (can be set in the XML file)
t_p	pixeltime (can be set in the GUI and in the XML file)

When changing the X resolution, Scope automatically calculates the next possible value (always rounded up). Afterwards, the pixeltime is adjusted.

Common X resolutions for $f_{resonance} = 7910\ \text{Hz}$ and an FPGA sampling frequency of $40\ \text{MHz}$:

X resolution	Pixeltime [μs]	samples per pixel
1264	0.05	2
843	0.075	3
632	0.1	4
506	0.125	5

The Y resolution needs to be even, because of the way Scope currently handles forward and backward lines.

Aspect Ratio: If the aspect ratio value for Y is smaller than the one for X, then the scanner amplitude of the X scanner is set to its maximum. For example, with aspect ratio 5x4, the X amplitude is at the maximum and the Y amplitude is $\frac{4}{5}$ of the X amplitude. Compared with aspect ratio 1x1, the top and bottom portions of the image are cut away.

After setting the aspect ratio, Scope tries to change the resolution. Because only certain X resolutions are possible, Scope corrects the aspect ratio. If the new resolution is not possible (e.g. the highest resolution is already set and the aspect ratio should be changed to 2x1), trying to change the aspect ratio will have no effect. In this case, it is necessary to reduce the Y resolution first.

Because the resonance scanner can only zoom by a factor of 1, 2, 3, and 4, the X amplitude is calculated first and rounded. Afterwards, the Y amplitude is calculated according to the aspect ratio. If the aspect ratio value for Y is larger than the one for X, this means that the Y amplitude is no longer exactly at its maximum.

Changing the resolution has no effect on the aspect ratio, because otherwise, there would be no possibility to set the number of pixels per micron in X and Y direction independently.

Points in the sourcecode which are responsible for the aspect ratio:

- `scope::parameters::Area::CalculateResolutionX()`
- `scope::parameters::Area::CalculateResolutionY()`
- `scope::ScannerVectorFrameResonance::FillY()`
- `scope::DaqController::DaqControllerImpl::Start()` (in DaqController_p.h)
- `scope::DaqController::DaqControllerImpl:: WorkerOnlineParameterUpdate ()` (in DaqController_p.h)

X turn fraction: The X turn fraction cannot be set on the GUI, but it can be set in the XML file.

Wait Storage/Wait Display: After the images are passed to the queues to the StorageController/DisplayController, the program waits some milliseconds. This helps against a problem where some chunks appear in the wrong frames or sometimes are repeated and averaging does not work correctly. This should be solved in another fashion, because the time necessary to wait will vary between computers.

Pixeltime: Since the FPGA needs to have an integer value for the number of samples per pixel, only certain pixeltimes are allowed. The number of samples per pixel can be calculated from the pixeltime by

$$n_s = t_p \cdot f_{sampling}$$

where

$f_{sampling}$	sampling frequency of the FPGA (currently fixed in the FPGA VI to 40 MHz)
t_p	pixeltime (can be set in the GUI and in the XML file)

The smallest pixeltime is currently set to be $t_p = 0.05$ for $f_{sampling} = 40\ \mathrm{MHz}$, which means 2 samples per pixel. Having $t_p = 0.025$ and only 1 sample per pixel would cause performance issues in the software.

Zoom: Only 1, 2, 3 and 4 are allowed.

Scannerdelay: Can be set to try to align the forward and backward lines (but this does not work currently).

Storage page

Save path: Use an SSD for image data storage. Otherwise writing is too slow.

Compress TIFF: When storing the images acquired with the resonance scanner to the disk, the TIFF compression should be turned off. Otherwise, the data processing in Scope will be too slow and make the program crash.

Images

Displayed images

Only the forward lines are shown on the screen. Each line is repeated, such that the image dimensions correspond to the resolution settings on the GUI.

Saved TIFF images

The left half of the saved TIFF image contains the forward lines, the right part the backward lines.

The first pixel on the left of a forward line is acquired after the synchronization signal from the driver board of the resonance scanner has switched from high to low. A forward line has as many pixels as the X resolution is set to in the GUI.

The last pixel on the right of a backward line is the last pixel acquired before the synchronization signal has switched from high to low. Because there is some variation in the period of the synchronization signal, the remaining pixels which do not fit into the image are just omitted. This results in a discontinuity in the middle of the image. In the case that there are not enough pixels to fill an entire line, some pixels in the middle of the image (on the right-hand side) are left black. In the XML file, there is a parameter for setting the resonance frequency that Scope uses for calculating the resolution. Slightly varying this value will allow to change how many pixels are omitted and to get rid of the black pixels, respectively.

Postprocessing: The TIFF images can be processed (backward line flipping and correction for sinusoidal scan trajectory) with the MATLAB script `UCLA_ICI_2PLSM_PostProcess`. The resulting corrected images are then saved in the folders “Downsampled”, “DownsampledRelative” and “PreProcess”. Example:

```
output =
UCLA_ICI_2PLSM_PostProcess('C:\\Users\\lombriser\\Desktop\\Adrien
Setup\\');

figure(1);
imagesc(output.data_flipped(:,:,5));

figure(2);
imagesc(output.data_sined(:,:,5));
```

Parameters in XML file

The following parameters are new:

- area0 -> daq -> inputs -> AcquisitionClockRate_Hz
  Sampling frequency of the FPGA
- area0 -> daq -> outputs -> ZoomChannelsString
  Two digital signals used for the discrete zoom (1, 2, 3, 4) of the resonance scanner
- area0 -> daq -> outputs -> ExternalClockSource
  Input for the resonance scanner synchronization signal
- area0 -> daq -> SwitchResonanceLine
  Output for switching the resonance scanner on and off (not needed)
- area0 -> daq -> ResonanceFrequency_Hz
  Resonance frequency of the resonance scanner used for calculations in the software
- area0 -> frameresonance -> SquarePixels -> XAspectRatio
  area0 -> frameresonance -> SquarePixels -> YAspectRatio
  Settings for image dimensions and pixel dimensions, respectively
- area0 -> frameresonance -> WaitAfterEnqueueStorage_Millisecond
  area0 -> frameresonance -> WaitAfterEnqueueDisplay_Millisecond
  Time to wait, quick fix for a problem

Hardware

The current setup is built from the following components:

- Power Supply: 2 x Agilent U8031A
- Outputs: NI PXI 6259 with BNC-2110
- Inputs: NI 5751 with SMB-2148 and SMB-2147
- Resonance scanner: 8 kHz
- Galvo: 6215H
- Driver board resonance scanner: Cambridge Technology 311-149881, Rev. P
- Driver board resonance scanner: Cambridge Technology 71215H05LJ-1HP, Rev. A

NI PXI 6259 with BNC-2110:

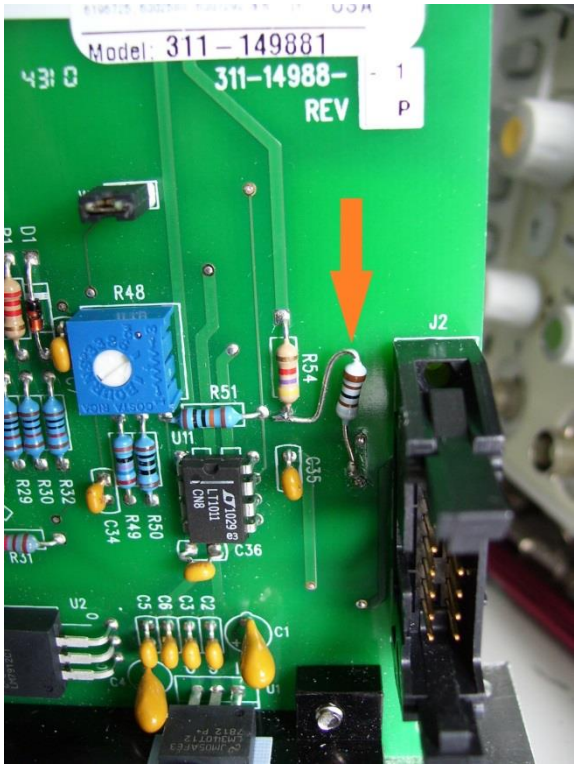
- PO.0 and D GND: Shutter (two single wires connected to a BNC cable)
- AO 0: Scanner position for y direction (BNC cable)
- PFI 1 and D GND: Synchronization signal of resonance scanner coming from NI 5751 (two single wires)
- PO.1: Discrete zoom (gray wire of D-SUB cable)
- PO.2: Discrete zoom (red wire of D-SUB cable)
- PO.3: Not necessary; turns off the resonance scanner when not scanning (orange wire of D-SUB cable)
- D GND: Digital ground (green wire of D-SUB cable)

NI 5751 with SMB-2148 and SMB-2147:

- DI 0: Synchronization signal of resonance scanner coming from the driver board (BNC cable)
- DO 2: Synchronization signal of resonance scanner (BNC cable)
- AI 0: Image signal coming from the Preamplifier (BNC cable)

Synchronization

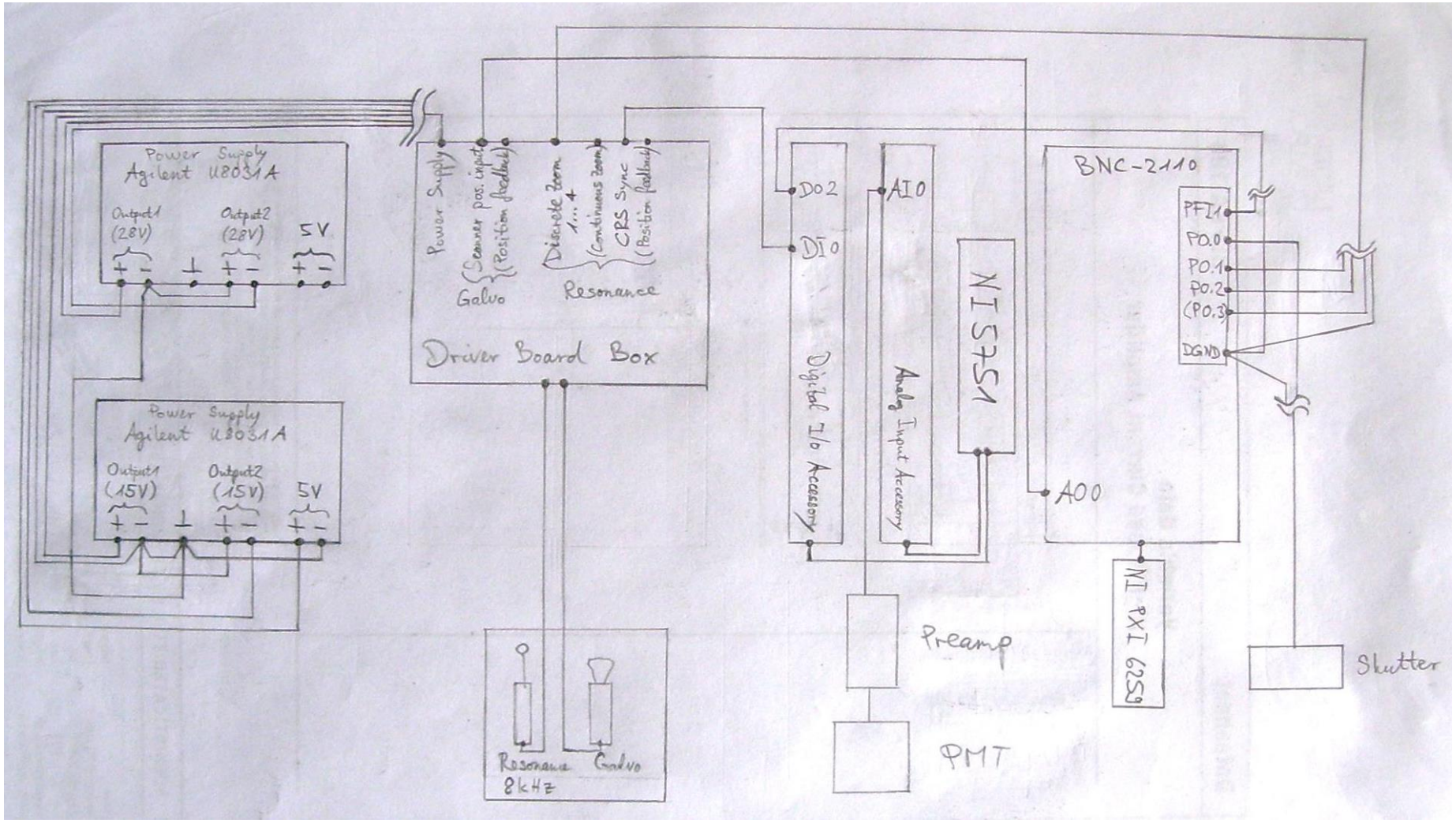
The driver board of the resonance scanner generates a square signal (“CRS Sync”) with the resonance frequency of the scanner. It is used inside the FPGA VI to identify forward and backward lines. The FPGA also provides this signal on its output DO 2, but only when it is acquiring image data. The NI PXI 6259 then uses this signal to generate the scanner signal for the Y scanner. It gets a double information out of the signal: The beginning of a frame (the Y signal just starts when the first edge arrives) and the sampling clock for the Y signal (high-low edge).



Signal voltage adjustment

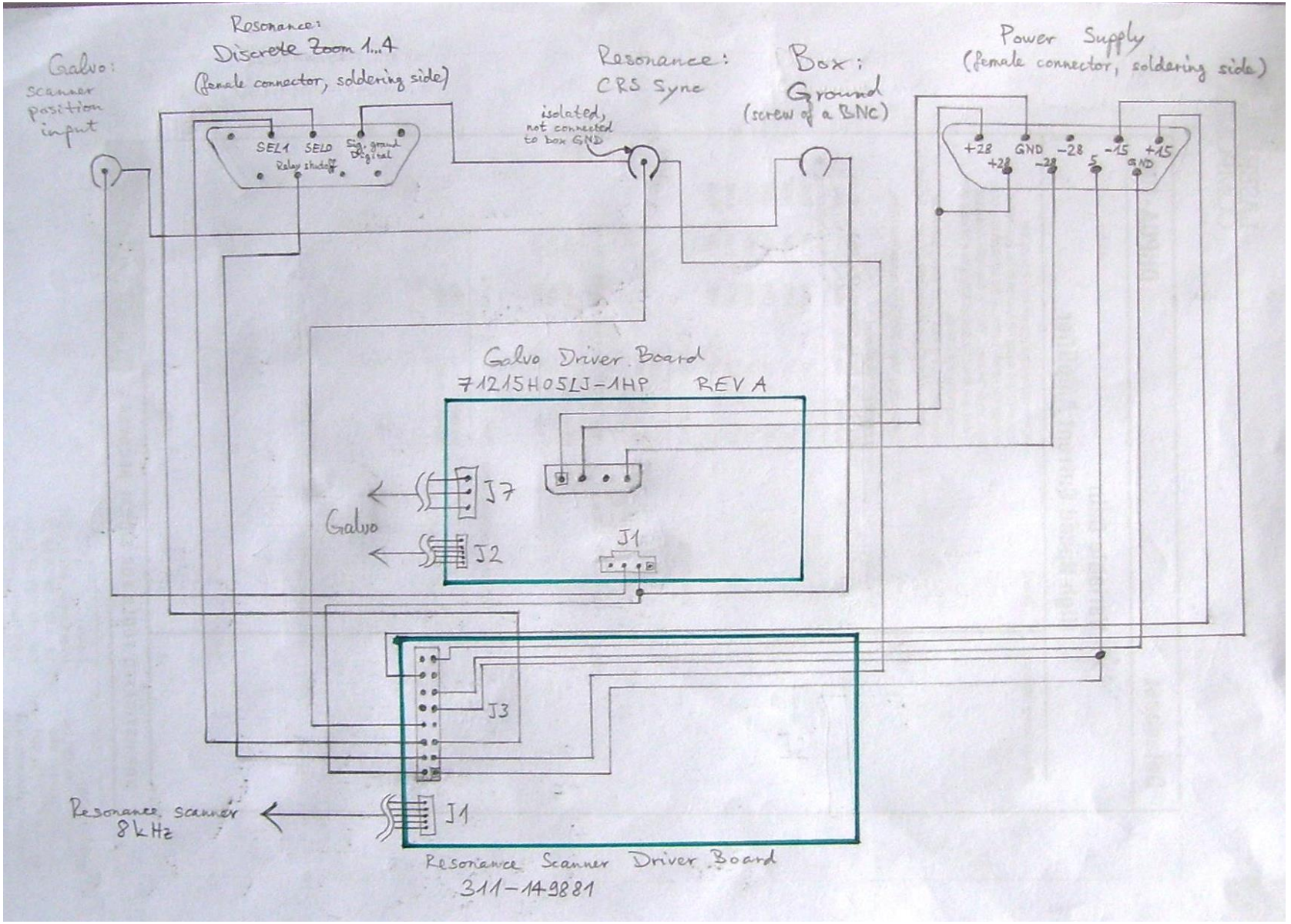
The amplitude of the synchronization signal of the resonance scanner is too high. Therefore, an additional resistor needs to be soldered on the driver board of the resonance scanner. It will lower the amplitude of the signal.

28.05.2014



Connections between the devices

28.05.2014



Connections inside the box with the driver boards

Troubleshooting

Problem: Scanning does not start when clicking on “Live Scan”. After stopping, the error “A STL exception happened in scope::BaseController<1,0>::BaseControllerImpl::WaitForAll: invalid vector<T> subscript” is shown.
Possible solution: The power source for the resonance scanner is not turned on, i.e. the synchronization/triggering signal is not generated.

Problem: Scope displays messages about “CreateFolder” and about memory.
Possible solution: The disk where the images get stored is full. Remove some of the images.

Problem: The number of MB is rising.
Possible solutions:

- Do not use the TIFF image that is currently being written. For example, do not refresh the file properties, like the space needed on the disk, too often.
- Make sure no other time- or memory-consuming applications are running and no other users are logged on.

Problem: The slider for the pixeltime does not go to the next position.
Solution: Type the number into the text field.

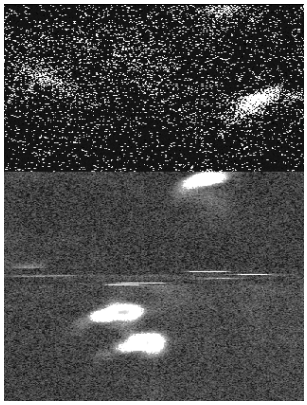
Problem: The error message “A Scope exception happened in scope::BaseController<1,0>::BaseControllerImpl::WaitForAll: WaitForOne did not succeed, async thread did not return” appears after stopping the scan.
Possible solution: Increase the values for WaitForAll() in the method `StopAllControllers()` in ScopeController_p.h (needs recompilation).

Problem: Writing the images to the harddisk is slow when using TIFF compression.
Possible solution: Change the TIFF compression algorithm used. This is done in scope/helpers/ScopeMultiImageEncoder.cpp in the method ScopeMultiImageEncoder::NewFrame(). Exchange there WICTiffCompressionZIP with one of these options (see also http://msdn.microsoft.com/en-us/library/windows/desktop/ee719867%28v=vs.85%29.aspx):

```
WICTiffCompressionDontCare
WICTiffCompressionNone
WICTiffCompressionCCITT3
WICTiffCompressionCCITT4
WICTiffCompressionLZW
WICTiffCompressionRLE
WICTiffCompressionZIP
WICTiffCompressionLZWHDifferencing
```

Another possibility would be to try LibTIFF (www.libtiff.org) or BigTIFF (bigtiff.org).

Possible Improvements

- Replace
  `std::this_thread::sleep_for(std::chrono::milliseconds(parameters.areas[_area]->frameresonance.waitafterenqueuestorage()));`
  and
  `std::this_thread::sleep_for(std::chrono::milliseconds(parameters.areas[_area]->frameresonance.waitafterenqueuedisplay()));`
  in PipelineController_p.h.
  These lines help against the problem that chunks appear in the wrong images or are repeated sometimes and also that averaging does not work correctly. Another way of solving this problem should be found.
  When both parameters `waitafterenqueuestorage` and `waitafterenqueuedisplay` are zero, the images look like the one below (averaged image).
  
- Generate only one image (passed both to the Display Controller and the Storage Controller) when the number of averages is set to 1. Currently, two images are generated in resonance scanner mode, also if no averaging occurs.
- Pockels cell signal: changes probably necessary in scope/devices/OutputsDAQmxResonance.cpp and scope/scanmodes/ScannerVectorFrameResonance.cpp (in the method `ScannerVectorFrameResonance::FillP()`)