Loading data from .bin files

In [ ]:
```
from sglxarray import load_trigger
from sglxarray.examples import example_data_path

lfp_bin_path = example_data_path()
print(f"Loading: {lfp_bin_path}")
lfp = load_trigger(lfp_bin_path)
# You could also load data specifying specific channels labels and/or segmen
```

```
Loading: /home/gfindlay/projects/sglxarray_dev/sglxarray/sglxarray/data/exam
ple-data.exported.imec0.lf.bin
nChan: 385, nFileSamp: 50000
```

In [ ]:
```
# View loaded data
# Note that x and y coordinates of each channel are automatically loaded fro
lfp
```

Out[ ]: xarray.DataArray (**time**: 50000, **channel**: 384)

```
array([[  28.125 ,   37.5   ,   -4.6875, ...,   98.4375,  192.1875,
         107.8125],
       [  32.8125,   32.8125,   -4.6875, ...,   60.9375,  173.4375,
         107.8125],
       [  32.8125,   14.0625,  -14.0625, ...,   70.3125,  182.8125,
         103.125 ],
       ...,
       [  98.4375,   89.0625,   75.    , ..., -107.8125,   23.4375,
          -4.6875],
       [  89.0625,   84.375 ,   60.9375, ...,  -98.4375,   28.125 ,
          14.0625],
       [  79.6875,   75.    ,   46.875 , ...,  -84.375 ,   32.8125,
           9.375 ]])
```

▼ Coordinates:

time	(time)	float64	0.0 0.0004 0.0008 ... 20.0 20.0
channel	(channel)	int64	0 3 4 7 8 ... 374 377 378 381 382
timedelta	(time)	timedelta64[ns]	00:00:00 ... 00:00:19.999579888
datetime	(time)	datetime64[ns]	2020-03-18T15:07:29 ... 2020-0...
x	(channel)	float64	27.0 43.0 27.0 ... 11.0 59.0 11.0
y	(channel)	float64	0.0 20.0 40.0 ... 7.64e+03 7.66e...

▼ Attributes:

units :	uV
fs :	2500.0025140000002
im :	CheckPat_1shank

In [ ]:
```
# Select data using index slices.
# For example, select all data from channels 3 to 8, from t=5 to 5=15
lfp.sel(channel=slice(3, 8), time=slice(5, 15))
```

Out[ ]: xarray.DataArray (**time**: 25000, **channel**: 4)

```
array([[-23.4375, -32.8125,  42.1875,  14.0625],
       [-23.4375, -42.1875,  42.1875,  14.0625],
       [-18.75  , -32.8125,  37.5   ,   0.    ],
       ...,
       [-60.9375, -70.3125,  14.0625,  14.0625],
       [-42.1875, -56.25  ,  18.75  ,  14.0625],
       [-46.875 , -51.5625,  28.125 ,  14.0625]])
```

▼ Coordinates:

time	(time)	float64	5.0 5.001 5.001 ... 15.0 15.0 15.0
channel	(channel)	int64	3 4 7 8
timedelta	(time)	timedelta64[ns]	00:00:05.000394972 ... 00:00:14....
datetime	(time)	datetime64[ns]	2020-03-18T15:07:34.00039497...
x	(channel)	float64	43.0 27.0 43.0 27.0
y	(channel)	float64	20.0 40.0 60.0 80.0

▼ Attributes:

units :	uV
fs :	2500.0025140000002
im :	CheckPat_1shank

In [ ]:
```
# Swap dimensions to select by non-index coordinates.
# Explicit swapping is necessary, because there are real performance implica
# same as in pandas where you have to set your indices before you do many op
lfp.swap_dims({'channel': 'y', 'time': 'datetime'}).sel(y=slice(3800, 3900),
```

Out[ ]: xarray.DataArray (**datetime**: 25000, **y**: 6)

```
array([[ -42.1875,  -70.3125,  -14.0625,  -79.6875,  -60.9375, -107.812
5],
       [ -65.625 , -107.8125,  -37.5   , -103.125 ,  -79.6875, -112.5
],
       [ -70.3125, -121.875 ,  -51.5625, -126.5625, -103.125 , -121.875
],
       ...,
       [-182.8125, -243.75  , -168.75  , -257.8125, -243.75  , -300.
],
       [-182.8125, -234.375 , -182.8125, -267.1875, -248.4375, -314.062
5],
       [-187.5   , -243.75  , -173.4375, -267.1875, -234.375 , -314.062
5]])
```

▼ Coordinates:

time	(datetime)	float64	5.0 5.001 5.001 ... 15.0 15.0 15.0
channel	(y)	int64	380 383 1 2 5 6
timedelta	(datetime)	timedelta64[ns]	00:00:05.000394972 ... 00:00:1...
datetime	(datetime)	datetime64[ns]	2020-03-18T15:07:34.0003949...
x	(y)	float64	27.0 43.0 59.0 11.0 59.0 11.0
y	(y)	float64	3.8e+03 3.82e+03 ... 3.9e+03

▼ Attributes:

units :	uV
fs :	2500.0025140000002
im :	CheckPat_1shank

ImecMap objects

In [ ]:
```
from IPython.display import display
# The "Imec Map" consists of the "Imec Readout Table (imro)" and "Imec Chann
display(lfp.im.imro)
display(lfp.im.cmp)
```

	chan_id	bank	ref_id	ap_gain	lf_gain	ap_highpass	site	x	y
0	0	0	0	500	250	1	0	27.0	0.0
1	1	1	0	500	250	1	385	59.0	3840.0
2	2	1	0	500	250	1	386	11.0	3860.0
3	3	0	0	500	250	1	3	43.0	20.0
4	4	0	0	500	250	1	4	27.0	40.0
...	...	...	...	...	...	...	...	...	...
379	379	0	0	500	250	1	379	43.0	3780.0
380	380	0	0	500	250	1	380	27.0	3800.0
381	381	1	0	500	250	1	765	59.0	7640.0
382	382	1	0	500	250	1	766	11.0	7660.0
383	383	0	0	500	250	1	383	43.0	3820.0

384 rows × 9 columns

	label	acq_order	usr_order	stream	chan_id
0	LF0	384	384	LF	0
1	LF1	385	576	LF	1
2	LF2	386	577	LF	2
3	LF3	387	385	LF	3
4	LF4	388	386	LF	4
...	...	...	...	...	...
380	LF380	764	574	LF	380
381	LF381	765	766	LF	381
382	LF382	766	767	LF	382
383	LF383	767	575	LF	383
384	SY0	768	768	SY	0

385 rows × 5 columns

In [ ]:
```
# With the IMRO and CMP, it is easy to get a complete representation of all
display(lfp.im.full)
```

	chan_id	bank	ref_id	ap_gain	lf_gain	ap_highpass	site	x	y	label	acq_order	us
0	0	0	0	500	250	1	0	27.0	0.0	LF0	384	
3	3	0	0	500	250	1	3	43.0	20.0	LF3	387	
4	4	0	0	500	250	1	4	27.0	40.0	LF4	388	
7	7	0	0	500	250	1	7	43.0	60.0	LF7	391	
8	8	0	0	500	250	1	8	27.0	80.0	LF8	392	
...	...	...	...	...	...	...	...	...	...	...	...	
374	374	1	0	500	250	1	758	11.0	7580.0	LF374	758	
377	377	1	0	500	250	1	761	59.0	7600.0	LF377	761	
378	378	1	0	500	250	1	762	11.0	7620.0	LF378	762	
381	381	1	0	500	250	1	765	59.0	7640.0	LF381	765	
382	382	1	0	500	250	1	766	11.0	7660.0	LF382	766	

384 rows × 13 columns

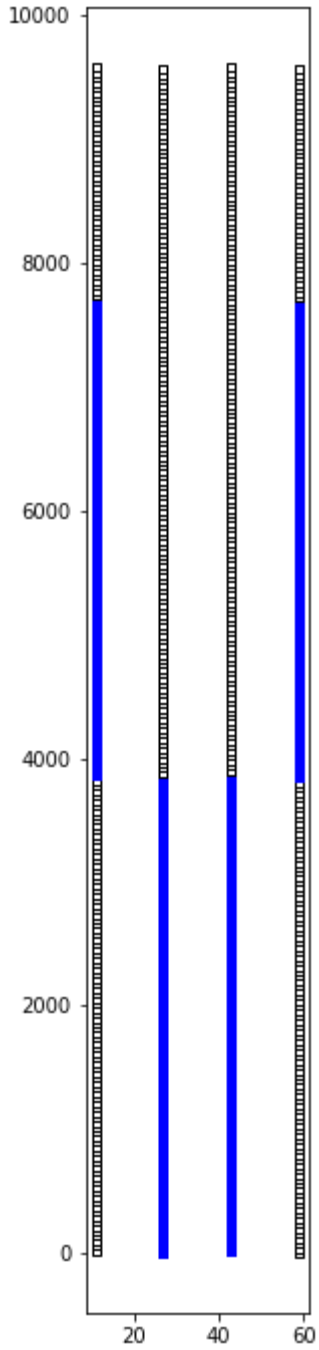
In [ ]:
```
# This can be useful for various things like plotting electrodes, converting
print(lfp.im.chans2coords([380, 383, 1, 5, 6]))
display(lfp.im.yrange2chans(y_lo=1280, y_hi=1380))
display(lfp.im.y2chans([1280, 1380]))
```

```
[[  27. 3800.]
 [  43. 3820.]
 [  59. 3840.]
 [  59. 3880.]
 [  11. 3900.]]
```

y	chan_id	bank	ref_id	ap_gain	lf_gain	ap_highpass	site	x	label	acq_order	usr_ord
1280.0	128	0	0	500	250	1	128	27.0	LF128	512	4
1300.0	131	0	0	500	250	1	131	43.0	LF131	515	4
1320.0	132	0	0	500	250	1	132	27.0	LF132	516	4
1340.0	135	0	0	500	250	1	135	43.0	LF135	519	4
1360.0	136	0	0	500	250	1	136	27.0	LF136	520	4
1380.0	139	0	0	500	250	1	139	43.0	LF139	523	4

y	chan_id	bank	ref_id	ap_gain	lf_gain	ap_highpass	site	x	label	acq_order	usr_ord
1280.0	128	0	0	500	250	1	128	27.0	LF128	512	4
1380.0	139	0	0	500	250	1	139	43.0	LF139	523	4

In [ ]:
```
lfp.im.plot_electrodes()
```



In [ ]:
```
# If you want an ImecMap object without having to load binary data, there ar
# All of the following are equivalent.
from sglxarray import ImecMap

im1 = ImecMap.from_bin(lfp_bin_path)
im2 = ImecMap.from_library('CheckPat_1shank', stream_type='LF')
im3 = ImecMap.CheckPat()
im3.stream_type = 'LF'
```

In [ ]:
```
# Data for both streams (LF and AP) are loaded, unless a single stream is sp
im4 = ImecMap.CheckPat()
display(im4.full)
```

	chan_id	bank	ref_id	ap_gain	lf_gain	ap_highpass	site	x	y	label	acq_order	us
0	0	0	0	500	250	1	0	27.0	0.0	AP0	0	
1	0	0	0	500	250	1	0	27.0	0.0	LF0	384	
6	3	0	0	500	250	1	3	43.0	20.0	AP3	3	
7	3	0	0	500	250	1	3	43.0	20.0	LF3	387	
8	4	0	0	500	250	1	4	27.0	40.0	AP4	4	
...	...	...	...	...	...	...	...	...	...	...	...	
757	378	1	0	500	250	1	762	11.0	7620.0	LF378	762	
762	381	1	0	500	250	1	765	59.0	7640.0	AP381	381	
763	381	1	0	500	250	1	765	59.0	7640.0	LF381	765	
764	382	1	0	500	250	1	766	11.0	7660.0	AP382	382	
765	382	1	0	500	250	1	766	11.0	7660.0	LF382	766	

768 rows × 13 columns

In [ ]: