

# EDA - Import and export data

## Revision History

| Date       | Version | Description   |
|------------|---------|---|
| 14/06/2024 | 1.2     | Added information to the description of the .dat format |
| 09/04/2021 | 1.1     | Updated contacts  |
| 06/11/2019 | 1.0     | First version of document                               |



---

## Supported data formats

EDA is meant to load and navigate data recorded using EDR3. For this reason all data files must be accompanied by an Elements Data Header file (.edh) in order to be loadable by EDA. For this reason, all exported data files will be accompanied by .edh files as well.

Supported formats for data files are .abf, .dat and .csv.

### Abf format

Axon Binary Files (.abf) version 2 can be loaded by EDA.

### Dat format

Binary files in .dat format consist of a continuous series of IEEE 754 single-precision (32-bit) binary floating-point values expressed in little endian (least significant bytes first), with no header. The data is arranged in groups of CurrentChannelsNumber+1 values. Each group consists of CurrentChannelsNumber current samples (ordered by channel number) followed by 1 voltage sample corresponding to a given time instant. Groups are arranged in temporal order.

E.g. for a 4 channels device the values will be:

IChan1Sample1, IChan2Sample1, IChan3Sample1, IChan4Sample1, VChanSample1,  
IChan1Sample2, IChan2Sample2, IChan3Sample2, IChan4Sample2, VChanSample2,  
and so on.

Every value has no scaling and is expressed in the same unit as the current range selected during the recording, e.g. currents might be expressed in pA or nA. Voltage is always expressed in mV.

Please note that this format does not store intrinsically the current range and the sampling rate information, which are stored in the .edh file, so the user will have to recover those pieces of information from there.

Below is shown a simple python script that loads the data of a 4 channels device and plots it:

```
import struct
import numpy as np
import matplotlib.pyplot as plt

# Parameters
CurrentChannelsNumber = 4
data_file = 'file_000.dat'
samplingRate = 100000 # in Hz, assuming a sampling rate of 100kHz

# Function to read binary data
def read_binary_file(file_name, channels):
```

ELEMENTS srl - ITALY - C.F./P.IVA/VAT 04113900403 - tel: +39 0547 482983 - [www.elements-ic.com](http://www.elements-ic.com)

commercial info: [info@elements-ic.com](mailto:info@elements-ic.com) - technical support: [support@elements-ic.com](mailto:support@elements-ic.com)



```
data = []
with open(file_name, 'rb') as f:
    while True:
        try:
            # Read a group of channels + 1 (voltage) samples
            group = struct.unpack('<' + 'f' * (channels + 1), f.read(4 *
(channels + 1)))
            data.append(group)
        except struct.error:
            break
    return np.array(data)

# Read the data from the binary file
data = read_binary_file(data_file, CurrentChannelsNumber)

# Separate the current and voltage data
currents = data[:, :CurrentChannelsNumber]
voltage = data[:, CurrentChannelsNumber]

# Generate the time axis based on the sampling rate
time = np.arange(0, len(voltage)) / samplingRate

# Plotting
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(15, 10))

# Plot current traces
for i in range(CurrentChannelsNumber):
    ax1.plot(time, currents[:, i], label=f'Current Channel {i+1}')
ax1.set_title('Current Traces')
ax1.set_xlabel('Time (s)')
ax1.set_ylabel('Current (nA)')
ax1.legend()

# Plot voltage trace
ax2.plot(time, voltage, label='Voltage', color='red')
ax2.set_title('Voltage Trace')
ax2.set_xlabel('Time (s)')
ax2.set_ylabel('Voltage (mV)')
ax2.legend()

# Show the plot
plt.tight_layout()
plt.show()
```

## Csv format

Comma Separated Value files consist of CurrentChannelsNumber+1 columns of numbers expressed in ASCII code separated by commas. Each row consists of CurrentChannelsNumber



---

current samples (ordered by channel number) followed by 1 voltage sample corresponding to a given time instant. Rows are arranged in temporal order.

E.g. for a 4 channels device the file will look like this:

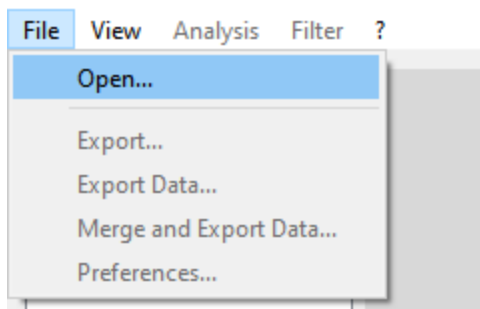
```
IChan1Sample1, IChan2Sample1, IChan3Sample1, IChan4Sample1, VChanSample1  
IChan1Sample2, IChan2Sample2, IChan3Sample2, IChan4Sample2, VChanSample2  
and so on.
```

Every value has no scaling and is expressed in the same unit as the current range selected during the recording, e.g. currents might be expressed in pA or nA. Voltage is always expressed in mV.

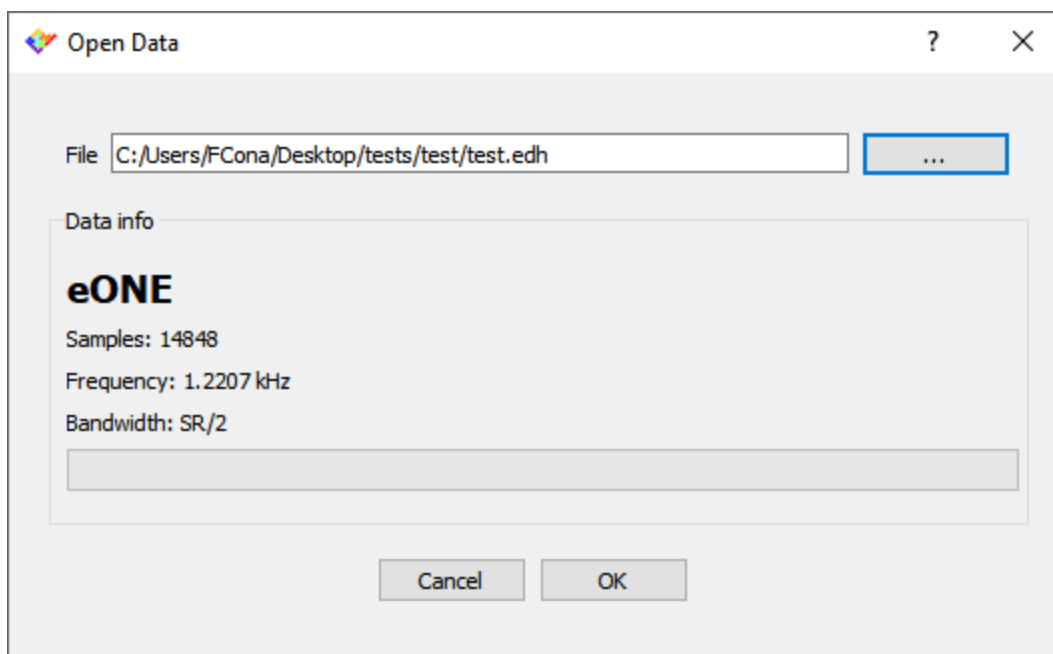
Please note that this format does not store intrinsically the current range and the sampling rate information, which are stored in the .edh file, so the user will have to recover those pieces of information from there.

## Import data

In order to import data select the Open option from the File menu.

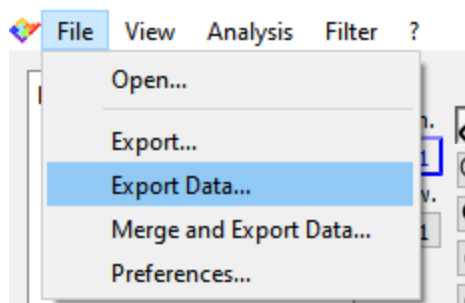


In the window that will appear click the button with 3 dots to browse to the .edh file of the dataset you want to load and then click ok to load the data.

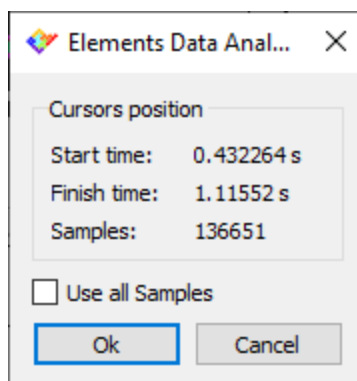


## Export Data

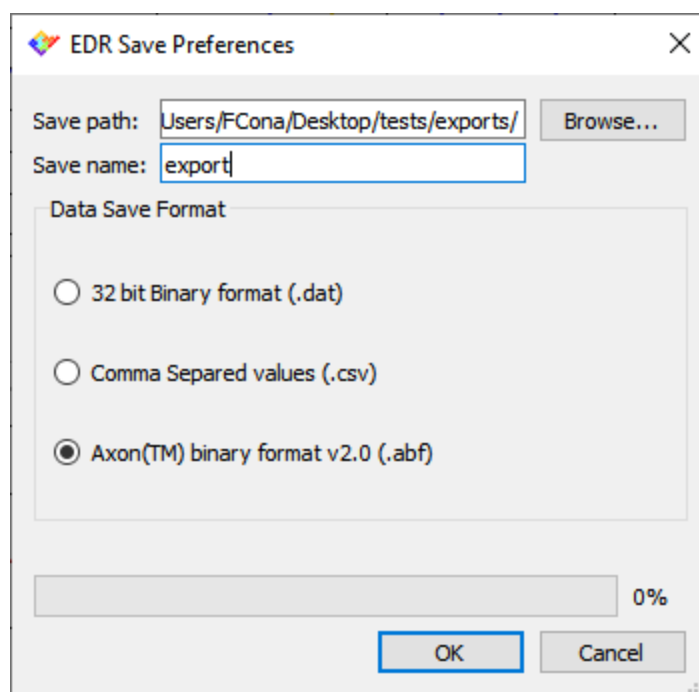
EDA allows to export the loaded data to any of the compatible formats described above. In order to do so select the Export Data from the File menu.



A popup will ask if the user wants to export the whole dataset or only the time window enclosed by the 2 cursors.



Make your choice and click OK. In the window that will appear select the path where you want to export the dataset, the filename of the exported dataset and its file format. Then click Ok to export the data.




## Merge and export data

EDA offers a further export option that allows to merge several imported datasets and to export them as a single dataset. In this case, the merged dataset will have a number of current channels equal to the sum of the current channels of all of the source datasets, and it will inherit a single voltage channel from only one of the source datasets.

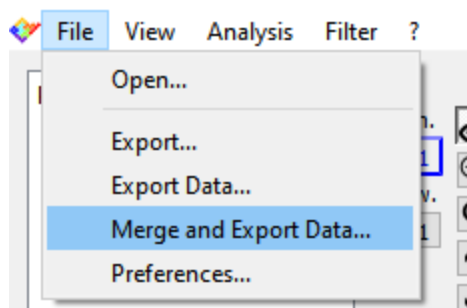
In order to use this feature a preliminary step is required to provide a common time axis since the source datasets might not be synchronized or even have the same length: for each source



---

dataset place the leftmost cursor to an easily recognizable mark point (for example the start of a voltage protocol) and click the save button  on the right of the cursors (see EDA GUI and basic commands).

After this procedure has been repeated for all the source datasets click the Merge and Export Data from the File menu.



In the window that will appear select the source datasets by dragging them from the Available Data panel to the right panels. Doing this remember that you have to select exactly one dataset to inherit voltage data from. Then, as for the regular export select the path, the filename, the file format and click ok.



EDR Save Preferences

| Available Data | Current Only Data | Current + Voltage Data<br>(exactly 1 item) |
|----------------|-------------------|--|
|                | test_01           | test                                       |

Save path:

Save name:

Data Save Format

32 bit Binary format (.dat)

Comma Separated values (.csv)

Axon(TM) binary format v2.0 (.abf)

0%

The document at this [link \(website page\)](#) provides a practical example of how to use this feature.