**OTB+ FILE STRUCTURE**

This document has the purpose to explain the otb+ file structure, how it works and how it is generated. The otb+ file is a custom extension used by OTBiolab+ software, which is a computer program for the acquisition and elaboration of biomedical signals sampled through OT Bioelettronica s.r.l. systems (specialized in HDsEMG). The otb+ files are simple archives masked with the extension “.otb+” which include different files (raw data, metadata and so on…) necessaries for data visualization and interpretation. The archive content can be visualized by changing the extension from “otb+” to “zip” and can be opened through any decompression tool like 7zip. The following figure shows the archive contents of an otb+ file with the extension properly changed:

Immagine che contiene testo

Descrizione generata automaticamente

These files are automatically generated by software during the saving procedure. In the next sections all the files will be explained by referring to the archive showed in figure.

**DATA FILES**

Signals data are saved in binary format and need to be analysed to have a correct idea on how the data can be properly extracted and visualized. The metadata are obviously fundamentals for this type of issue, but these will be properly described in the next section. The first thing to do is to distinguish between the two only data files that are managed by the software: the raw data file and the processed data file.

* *Raw data (.sig):* the raw data are saved in a file with **“.sig”** extension, which it’s a simple binary file. The data acquired by the device are saved in binary format, but the number of bytes used for each sample can change depending on acquisition type. For hardware reasons there are just two possibilities, for EMG acquisition only 2 bytes are necessary, otherwise for EEG acquisition 3 bytes needs to be used. This information can be extrapolated from the correspondent metadata and details will be described in the next section. Another important characteristic is that raw data contains all the configured channels of the device, only a single file is used. This means that if the device is sampling 140 channels, only one .sig file will be created to save all the samples. The following figure shows how the data are saved inside the file.

In this description the following notation will be used: *Ch\_x – Smp\_y*, where Ch\_xmeans channel x, with x representing the channel number (from 0 to 139 in the example of 140 chanells), and *Smp\_y* means sample y, with y representing the sample number which depends on device frequency and on acquisition duration. For example, Ch\_10 – Smp\_2, means the sample number 2 of the channel 10.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Ch\_0-Smp\_0* | *Ch\_1-Smp\_0* | *Ch\_2-Smp\_0* | *Ch\_3-Smp\_0* | *………………* | *Ch\_139-Smp\_0* |
| *Ch\_0-Smp\_1* | *Ch\_1-Smp\_1* | *Ch\_2-Smp\_1* | *Ch\_3-Smp\_1* | *………………* | *Ch\_139-Smp\_1* |
| *Ch\_0-Smp\_2* | *Ch\_1-Smp\_2* | *Ch\_2-Smp\_2* | *Ch\_3-Smp\_2* | *………………* | *Ch\_139-Smp\_2* |
| *Ch\_0-Smp\_3* | *Ch\_1-Smp\_3* | *Ch\_2-Smp\_3* | *Ch\_3-Smp\_3* | *………………* | *Ch\_139-Smp\_3* |
| *Ch\_0-Smp\_4* | *Ch\_1-Smp\_4* | *Ch\_2-Smp\_4* | *Ch\_3-Smp\_4* | *………………* | *Ch\_139-Smp\_4* |
| *……………………………………* | | | | | |

The number of used bytes and the data format need to be considered to properly extract data and for the assignment to the correspondent channel.

* *Processed data (.sip):* every time that an elaboration is executed the data need to be saved in a dedicated file, in this situation the extension is **“.sip”** but it’s always a binary file with a specific format and with a specific data representation. This is slightly different from .sig file because the data are always codified with 8 bytes. Moreover there isn’t a unique file for all the processed data like before but a file for each channel will be created. For example, if a processing is used on a 64 channel track, 64 .sip files are created, each of this file contains the processed data of a single channel involved during the processing. This file is quite simpler to read compared to .sig file, because it’s just necessary read 8 byte as double from the file until the end is reached, and all of the samples are related to one channel of the input track. For each .sip file a **.pro** metadata file is created to extrapolate the necessary information for the correct visualization of the result.

**METADATA FILES**

The metadata contains all the necessary information to read properly the correspondent data and also to organise the visualization of the signals in OTBiolab+ software. These are organised in xml files which are read and parsed by software and can be easily visualized by any text editor in the pc. In our example archive the main metadata files are 20210316113819\_02.xml, DockPanel.config, form\_dock00.xml, trapezoidalpathXXXXXXXXXXXXXXX\_X.pro and patient.xml. The files with extension .config and .pro are always xml files with the extension modified just for readability.

* *20210316113819\_02.xml*: This file contains all the information related to acquisition device, included used adapter and the applied sensors. These types of information are used to estimate the correct conversion factor which allows to switch from binary format to the proper physical quantity (usually millivolts) of raw data, the acquisition mode, hardware filters and so on. Below a picture of the file is showed:

Immagine che contiene tavolo

Descrizione generata automaticamente

The configured device is quattrocento which enables a lot of different adapters and sensors to be used, for this reason the file is really long but the structure is quite repeated. From the section related to the “Device”, different fields can be analysed:

* + *Name:* the device title
  + *ID:* this field is normally unused, there is just the possibility to insert an identificative code to distinguish devices which share name.
  + *DeviceTotalChannels:* each device has a maximum number of channels that can be configured, for example Quattrocento device has 400 channels in the full configuration (see *Model* field for further information about this aspect), but when you create a setup not all the channels need to be configured or transmitted. For this reason, the number of channels is computed starting from the setup created through software and this number is inserted in this field. This information is important during reading step, because the number of channels is fundamental to unpack the raw data file and organise data between channels.
  + *Model:* this field is extremely related to Quattrocento device; indeed this system can be built with 4 different configurations related to the maximum number of available channels. The basic configuration is with 96 channels, the second one is 192 channels, the third one provides 288 channels and the last one with 384 (in all configurations the 16 auxiliary channels are included). The code contained in this field encode the configuration in the following way: 40096 => 96-configuration, 40192=> 192-configuration, 40288 => 288-configuration and 40384 => 394-configuration.
  + *Sources:* this field has been inserted to implement in the future the communication of multiple devices with the pc, it is used to understand what channels are related to what device when all the raw data are saved in a unique file. For the actual implementation the source is always set to 1.
  + *SampleFrequency:* in this field the sampling frequency used during acquisition is saved, obviously it is fundamental to properly extract the data from the file where are saved.
  + *Firmware:* the current firmware version of the device will be saved in this field, but it was not yet implemented.
  + *ad\_bits:* another important information to properly extract the data from the raw file, it represents the number of byte/bit used to represent a single value. For raw data there are just two possibilities: 16 bits (2 byte) for EMG acquisitions and 24 bits (3 byte) for EEG acquisitions. The processed data use 8 byte but they have a dedicated metadata file.
  + *AnalogOutConn, AnalogOutCh, AnalogOutGain:* these fields are explained in the same section because these are all releated to a feature of QUATTROCENTO called “Analog Out”. For a more detailed explanation of this functionality check the Quattrocento User Manual.

The device fields can be slightly different from one device to another one, but the general information common to all the systems should be contained in the previous description.

Each device can be configured with different adapters which contains other information about data conversion and acquisition mode. Let’s see specific field descriptions in the following section:

* + *ID:* quite similar to device “Name” field, used to understand what type of adapter has been connected to the device.
  + *Description:* a simple description of the connected adapter, used during visualization to provide some fast information on the utilised adapter.
  + *Gain:* this field is related to the amplification factor applied to the channels. This amplification is saved in the adapter section, but it’s related to the device connector where the adapter is connected. Since in the xml there are no sections related to the connectors, it was necessary to save this field in the adapter. The gain is fundamental for the conversion factor estimation and can be different not only between devices but also between the connectors of the same device. For example, in the quattrocento the gain change between front panel connectors and back panel connectors
  + *HighPassFilter/LowPassFilter:* these fields have only purpose description; these represent the cut frequencies used by hardware filters. These values are related to the connector where the adapter is connected (like for the previous field) and can be different for each connector in relation with the cut frequencies available for that device.
  + *DetectionMode:* this field is used to understand the acquisition mode that can be 0 for bipolar acquisitions, 1 for monopolar acquisitions and 2 for differential acquisitions.
  + *AdapterIndex:* this index represents the position of the adapter referred to all available adapter slots of the current device. Let’s suppose for example that with QUATTROCENTO, IN1 connector and IN3 connector have been filled with an adapter, and that no adapter has been inserted in the IN2 connector. The two *AdapterIndex* of IN1 and IN3 connectors will be respectively 0 and 2. But we have to pay attention because these indexes can change a lot compared to the *Model* field we saw previously, let’s see how by using as example the compare figure of 96-configuration model and 192-configuration of the QUATTROCENTO:

IN8

IN7

IN5

IN6

IN4

IN3

IN2

IN1

5

4

3

2

1

0

2

1

0

MULTIPLE 4

MULTIPLE 3

MULTIPLE 2

MULTIPLE 1

MULTIPLE 4

MULTIPLE 3

MULTIPLE 2

MULTIPLE 1

IN8

IN7

IN6

IN5

IN4

IN3

IN2

IN1

In red the AdapterIndex can be observed. Note how the MULTIPLE1 shows a different adapter index compared to the two configurations.

* + *ChannelStartIndex:* this field shows the index of the first channel of the current adapter inside the raw data file, this means that this number is related to *DeviceTotalChannels* field. This number is used when we want to obtain data for the specific adapter in a file that contains all the device channels.

There should be another section related to the sensors connected to the adapter, but these are mainly descriptive fields that can be used from the software for different tasks. One of this uses the ID field to discriminate the electrode type and consequently the acquisition mode, for example if a needle sensor is recognised through the ID field, the software knows that it’s an intramuscular acquisition, if there is a cuff it will be an eeg acquisition and so on.

* *form\_dock00.xml:* in OTBiolab+ software the visualization of signals uses a class called “Track” to manage and organise the plot of the data. The tracks can be divided in “grouped” and “single” when they contain multiple channels or just one channel respectively. In most cases each adapter should have its own track and contains all the configured signals of that adapter (except for bipolar acquisition where each signal has its own track). These tracks are organized in tab and multiple tabs can be managed by the software. This file is related to one tab of the visualization window (when there are multiple tabs, multiple dock xml files will be created) and inside of it there are all the information of the tracks contained in it, like the colour of the tracks, the duration, the total number of the signals contained in the tracks and so on. Below the figure showing the file:



A lot of these fields are old and not used anymore or others are exactly the same of the fields already explained in the previous section, so the following description will be related just to these actually used by the software and the new one never described before:

* + *Epoch:* the epoch is a time value used by the software to discretize time axis. It’s normally used to run specific processing elaboration.
  + *Plugin:* specific class used by the software to read samples from raw data file.
  + *Description:* just a descriptive field that it showed from the software when the mouse is over the current track header.
  + *Range\_max/range\_min:* these fields represent respectively the maximum and the minimum values of the range currently visualized during file saving. It’s related to the correspondent track, and it is expressed in the unity of measurement found in the previous field.
  + *Forecolor:* represent a code which express the color of the correspondent track.
  + *Signal\_path:* the name of raw data files where the sample of the correspondent track can be found.
  + *Track Index:* index related to the current track which represents the position in the list of tracks contained in the tab.
  + *Timestep:* it’s a time value which represents the time difference between two consecutive samples.
  + *Conversion\_factor:* it represents the value need to be multiplied to switch between binary data representation to voltage representation. When this value is equal to 1 means that the value needs to be estimated again by using the gain, ad bits and so on. But if it’s different from 1 it needs to be used exactly as it’s saved.
  + *Start\_list/Duration\_list:* these fields represent two list which contain the start times of the track and the durations of the tracks. Normally these lists will contain just one value, the track begin and the duration, but there are some specific cases, for example when some processing are used on different sections of the same tracks, where the tracks is composed by different pieces. In that case the list will contain more values related to the number of pieces of the track.

The opening tag of the file after the “*Epoch*” field is called “*Signal*”, this tag includes all the information we saw for a single track, so in this file will be contained a lot of “*Signal*” tag in a number equal to the number of tracks when this otb+ file has been saved.

* *DockPanel.config*: this file is automatically generated by a library used in the software to manage the docking. Docking is a simple tool which allows to the users to move the different GUI parts like independent components, so these can be opened like isolated window or place them in some specific position of a common interface.
* *Patient.xml*: this file contains some information about the acquisition protocol and the involved subject. These are simple descriptive information, and some can be inserted by a user through the software others are automatically inserted. All the fields are quite autodescriptive, for this reason just a figure of an example is showed below.Immagine che contiene testo

  Descrizione generata automaticamente
* *Files .pro*: this is the extension of file used to save the metadata of any elaboration executed on signals. Each .pro file contains the metadata necessary to open the correspondent .sip file, which contains the processed data in binary format. If we make a compare with the metadata and data of raw signals that we analysed before, there are some differences that need to be highlighted. As we saw previously, the raw data are saved in a unique file with a 2 bytes format, indeed the related metadata is a quite complex xml file which includes all the necessary information to extract data, convert them and assign them to the proper track (grouped or single). With the .pro and .sip files the thinks are slightly different, the data are formatted with 8 bytes and there’s no need of conversion factor like the raw data, because it’s just necessary to read the 8 byte as “double” to obtain the already converted data. Another difference is that for each processed channel we find the correspondent couple of .sip and .pro files. This means that if a grouped track with 64 channel is processed, 64 couples of .sip and .pro files will be find. This results in a metadata very simple, with just some few fields like “fsample” and “unity of measurement”. But to reconstruct the grouped elaborated track this is not enough, indeed this information are used together with the “form\_dock00.xml” which shows inside the “*signal”* tag. Each signal tag is related to a track, and inside of it there are all the files need to be included to reconstruct that. Below two images which show all the .sip and .pro files of a elaborated 64 channels grouped track and the correspondent form\_dock.xml which includes the .sip and .pro inside the same “*signal*”(track) tag.

Immagine che contiene tavolo

Descrizione generata automaticamenteImmagine che contiene testo

Descrizione generata automaticamente

Let’s give a look to the specific fields inserted inside the .pro file.

Immagine che contiene testo

Descrizione generata automaticamente

* + *Title:* it simply represents the title of the executed processing.
  + *Description:* it shows the description of the executed processing.
  + *Start:* important field which contains the beginning of the current processed channel
  + *End:* important field which contains the end of the elaboration for the correspondent channel. With *End* and *Start* fields the duration of all the elaboration can be estimated.
  + *FSample:* for this field it’s necessary to pay some attention because it’s strictly correlated to the epoch used for the elaboration. Some processing use the epoch for the signal elaboration, in some cases the result will provide a point for each epoch, so if you suppose to have for example a 0.5 epoch, the result will have just two point for second, so in this specific example the fsample will become “2 Hz”. In the image there is an fsample of 2048, this means that the epoch is not used, and the processing gives as result the exact number of points of input track.
  + *Ladder and positiveonly:* mainly unused fields.
  + *Unity\_of\_measurement:* it represents the unity of measurement of the processed data during file saving.