# Data exchange format and file formats

**V0.1, 1.11.2017, Stanislav Mašláň, CMI**

Following text describe formats and structure of the files used for (i) data exchange between LabVIEW and Octave/Matlab and (ii) data formats of corrections (iii) data formats of the transfer of data between GUI and processing module (Matlab/Octave). Note in current version it is not full documentation of the formats, but more an explanation what decisions were made and why.

Abbreviations:

LV – LabVEIW

CVI – LabWindows CVI

EOS – End of string

DWORD – unsigned 32bit variable

INT16 – signed 16bit integer

INT32 – signed 32bit integer

Float32 – 32-bit real number

BYTE – unsigned 8bit variable

HDD – Hard drive

TWM – The LV program developed in scope of TracePQM project

GUI – Graphical User Interface

HW – HardWare

QWTB – Q-Wave toolbox

INFO – Brain-dead structured, human readable text file

Matlab – Matlab SW (Mathworks)?

GNU Octave – Open source equivalent of Matlab that happens to be almost 100% comatible

m-script – Matlab/Octave’s function file

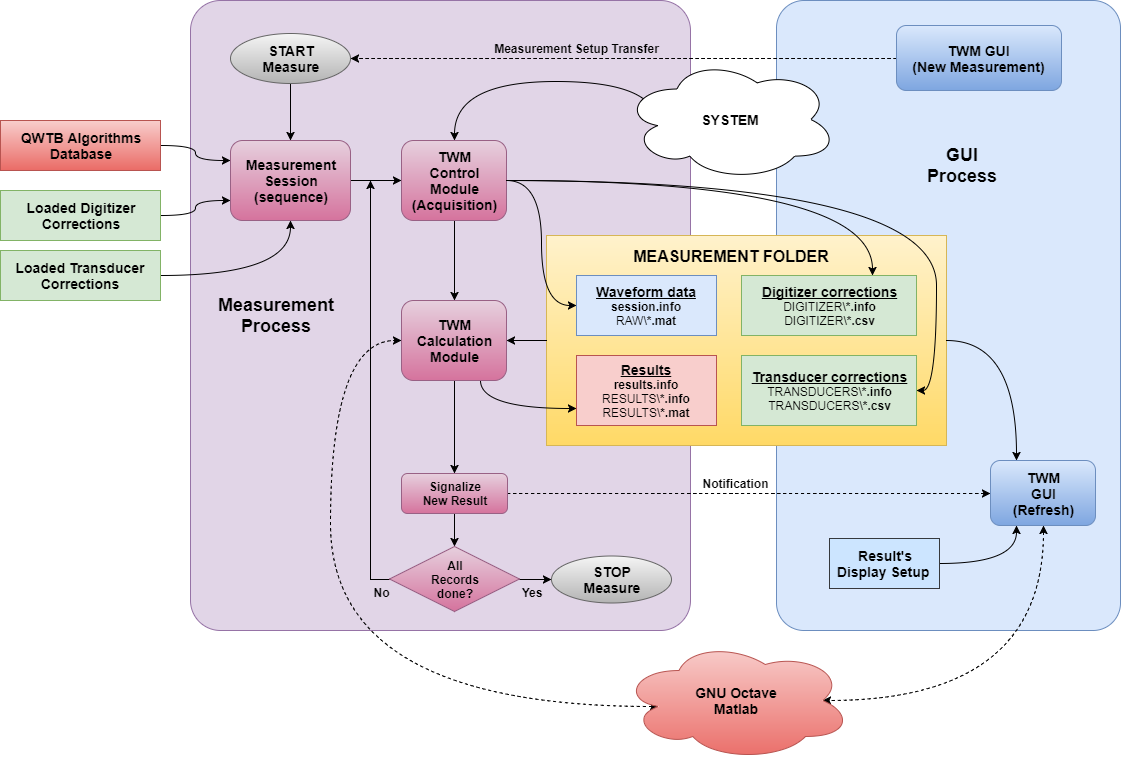
## Date Flow and Data Interchange Structure

The TWM is organized according to the diagram below. The whole TWM application is split into two separate processes that run in parallel. Main process is ‘**GUI Process**’. It contains configuration panels of the HW, configuration panels of the measurement, configurations of the result display and selector of the correction files (not loading, just selection).

When the user wants to initiate a new measurement the ‘**GUI process**’ will create ‘**Measurement Process**’ which will do following: (i) Loads correction files, (ii) Loads selected QWTB algorithm’s configuration from QWTB alg. database file, (iii) Builds measurement sequence, (iv) Initiates acquisition, (v) Stores acquired data and full copy of the Corrections and QWTB alg. setup, (vi) When requested by user, initiates calculation of the stored waveforms (vii) Signalizes ‘new result available’ to the GUI process and (viii) Repeats from (iv) until all acquisitions are done.

When ‘**GUI Process**’ receives notification of the new result or user requires refresh of the results view, it will look into the current measurement folder and will read, format and display the results.

Key feature of the proposed system is, the LV workload is minimized to acquisition of the data, storage of the measurement data and displaying of the results. However the actual work related to the processing the data, loading corrections, reading and formatting the results for displaying are done in Octave/Matlab! This way both LV and CVI implementation can share ALL processing and file handling m-scripts. Sharing of the data between the LV/CVI and Matlab/Octave is made via files that remains archived in the measurement folder (unique folder for each new measurement). Details on the selected file formats and files/folder hierarchy are shown in the following chapters.



## Storage of the measured data

Main requirements for storage of the captured records are following:

1. Must be easy to handle in LV, CVI, Octave and Matlab or plain C/C++.
2. Must have human readable and editable header (text file).
3. Must be memory-saving because of streaming modes from fast digitizers.

After analyzing possibilities it was decided to use combination of two files. First, the **raw** binary data are stored in the **Matlab MAT version 4** format. Second, the **header** will be stored as text file in INFO format.

Organization of the files in the measurement folder is following:

**Measurement folder:**

**session.info** *- measurement header*

**results.info** *- calculated results header*

RESULTS/**\*.info** *- results data headers*

RESULTS/**\*.mat** *- results data (large objects)*

DIGITIZER/**\*.\*** *- digitizer correction files*

TRANSDUCERS/**\*.\*** *- transducer’s correction files*

RESULTS/ *- calculate results*

#### Raw Binary Data Format

**MAT-v4** file format is very primitive and easy to handle format having following file structure:

|  |  |
| --- | --- |
| **Item type** | **Description** |
| DWORD | ID if the variable data type. |
| DWORD | Rows count M. |
| DWORD | Columns count N. |
| DWORD | Is complex flag. |
| DWORD | Length Q of the name. |
| [BYTE\*Q] | Name of the variable including ‘\0’ EOS. |
| [M\*N\*item\_size] | Array of the items organized per columns [column\_1, column\_2, …, column\_M]. |
| … next variable … |  |

The limitation of the format is the data cannot have more than 4 GSamples as the matrix dimensions are store in 32-bit variables (Matlab actually state only 100 MItems are allowed). However the format may be in future replaced by plain binary if the limitation became important.

The sample data from all channels are merged and stored into the 2D matrix variable called ‘**y**’, one row per channel. Traditional order one column per channel is not possible due to internal structure of MAT format – during streaming of data to the file it is easy to add columns, however to add rows whole file have to be reordered. In order to minimize HDD usage and maximize the data throughput, the sample data are stored directly in the integer format generated by the digitizers. So far, only two formats are considered (i) INT32 and when possible in terms of resolution (ii) INT16. If the selected HW supports logging of the temperature, the MAT file will also contain two variables with temperatures. Two variables are related to the temperature:

*temp\_sample* – 1D array of the sample indices when the temperature was measured (float32)

*temp\_data* – 2D array of measured temperatures in float32 (rows: channels, columns: readings)

Note the ‘**temp\_sample**’ values are indices of the sample where the temperature was measured, i.e. value 100 means hundredth sample, 1000 means thousandth sample, etc… The sampling rate for the temperature is set to 10 seconds so there is not unnecessarily lot of values.

The file naming rules for the record data are show in the following table:

**RAW records data (./RAW/):**

G0001-A0001.mat *- record for 1. average of 1. group*

G0001-A0002.mat *- record for 2. average of 1. group*

G0002-A0001.mat *- record for 1. average of 2. group*

G0002-A0002.mat *- record for 2. average of 2. group*

…

#### Data header format

Second file related to the raw records is human readable header. Many formats can be used here. However, as the file structure must support subsections in order to made it versatile enough. It was decided to use INFO format developed at CMI. This is very simple ‘braindead’ text format which can be generated by any program or can be written manually and it is also very easy to read. Libraries are available for LV, Octave and Matlab and can be implemented even for C/C++. Each header of the measurement (= one measurement session) is structured into following levels: (i) Session, (ii) Averaging group, (iii) Record. Each session (i) contains one or more averaging groups (ii) defined by item ‘**groups count**’. The session (i) always contains setup of the HW, which is common for all averaging groups (ii), such as HW identifiers, capabilities of HW, etc. Next, it contains ‘**averaging group G**’ sections (ii), where **G** is index of the averaging group. Each averaging group (ii) contains setup that is unique for each group, such as number of samples, sampling rate, etc. Finally, each averaging group also contains information about particular records (iii) within the group.

The example of the header of the record that contains one measurement group is shown in the following text:

*// ====== COMMON SETUP ======*

*// Unique identifiers of each channel:*

**#startmatrix**:: channel descriptors

HP3458A, sn. MY45053095;

HP3458A, sn. MY45053104;

**#endmatrix**:: channel descriptors

*// unique identifiers of auxiliary HW (AWG, Counter, …):*

**#startmatrix**:: auxiliary HW descriptors

**#endmatrix**:: auxiliary HW descriptors

*// number of virtual channel of the digitizer:*

channels count:: 2

*// file format of the sample data:*

sample data format:: mat-v4

*// name of the variable with the sample data:*

sample data variable name:: y

*// number of measurement groups:*

groups count:: 1

*// digitizer has temperature measurement capability?:*

temperature available:: 0

*// digitizer has temperature logging during sampling?:*

temperature log available:: 0

*// DMM sampling mode (HW specific attribute):*

sampling mode:: DCV

*// DMM synchronization mode (HW specific attribute):*

synchronization mode:: MASTER-SLAVE, MASTER clocked by TIMER

**#startsection**:: averaging group 1

*// ====== AVERAGING GROUP #1 ======*

*// number of averaging cycles (repeated records):*

averages count:: 3

*// number of desired samples per record:*

samples count:: 10000

*// set sampling rate:*

sampling rate [Sa/s]:: 48000.0000000000

*// voltage ranges for each channel:*

**#startmatrix**:: voltage ranges [V]

1.00; 1.00;

**#endmatrix**:: voltage ranges [V]

*// DMM aperture time (HW specific attribute):*

aperture [s]:: 1e-6

*// trigger setup:*

trigger mode:: Immediate

*// ====== RECORDS ======*

*// relative file paths to the files with sample data:*

**#startmatrix**:: record sample data files

RAW\G0001-A0001.mat;

RAW\G0001-A0002.mat;

RAW\G0001-A0003.mat;

**#endmatrix**:: record sample data files

*// actual samples counts for each record file:*

**#startmatrix**:: record samples counts

10000;

10000;

10000;

**#endmatrix**:: record samples counts

*// time increment (sampling period) for each record:*

**#startmatrix**:: record time increments [s]

2.08333333333333E-5;

2.08333333333333E-5;

2.08333333333333E-5;

**#endmatrix**:: record time increments [s]

*// gain factors for scaling of the sample data for each channel and record:*

**#startmatrix**:: record sample data gains [V]

9.9999997E-10; 9.9999997E-10;

9.9999997E-10; 9.9999997E-10;

9.9999997E-10; 9.9999997E-10;

**#endmatrix**:: record sample data gains [V]

*// offset for scaling of the sample data for each channel and record:*

**#startmatrix**:: record sample data offsets [V]

0.0000000; 0.0000000;

0.0000000; 0.0000000;

0.0000000; 0.0000000;

**#endmatrix**:: record sample data offsets [V]

*// relative timestamp for each channel and record (initial time of first sample):*

**#startmatrix**:: record relative timestamps [s]

0.0312291666666667; 0.0312291666666667;

0.406229166666667; 0.406229166666667;

0.687479166666667; 0.687479166666667;

**#endmatrix**:: record relative timestamps [s]

*// absolute timestamps of each record start (using low. res system time):*

**#startmatrix**:: record absolute timestamps

2014-03-03T22:18:53.77343750000000000001;

2014-03-03T22:18:54.16308593749999999997;

2014-03-03T22:18:54.47265625000000000002;

**#endmatrix**:: record absolute timestamps

**#endsection**:: averaging group 1

*// ====== CORRECTIONS DATA ======*

**#startsection**:: corrections

*// relative paths for correction file of each channel:*

**#startmatrix**:: transducer paths

T01\dummy.info;

T02\dummy.info;

**#endmatrix**:: transducer paths

*// phase index for each virtual channel of the digitizer:*

**#startmatrix**:: channel phase indexes

1;

1;

**#endmatrix**:: channel phase indexes

*// relative path for correction file of the digitizer:*

digitizer path:: DIGITIZER\HP3458A\_awg.info

**#endsection**:: corrections

Meaning of the particular items of the header file should be obvious from the attached comments. Note the comments introduced by ‘**//**’ are not required. It is just for documentation.

## Correction files

One of the key concerns are the correction files. The corrections loaded in the TWM at the time of the measurement must be somehow stored together with the measurement sample data and header (in the same folder). The requirement is the measured data can be easily copied, therefore some link from measured data to data with corrections is not suitable, because these corrections would be missing in the copy. When the raw data are later (re)processed, all necessary information must be available together with the measured data. After considering possibilities it was found the only reasonable way is to not modify the correction data before the storage. I.e. the correction data loaded into the TWM are identical to the correction data attached to the each measurement. Therefore, if the corrections in the measurement folder are somehow modified during manual processing of the data, e.g. if some mistake is found in the correction files, the corrections from the measurement folder can be easily copied back to the location of TWM and loaded into the TWM and used for next measurements. This, of course, leads to the problem of format choice because the corrections are relatively complex in content.

The correction data format must be versatile enough in order to enable storage of any calibrated dependency (frequency, voltage, aperture, temperature, …) and must enable filtering the correction file based on the setup of the HW which is also very complex. Furthermore the dependencies of the correction parameters on the attributes of the digitizers themselves are not known in advance as the TWM may be extended by another digitizer with another attributes. Several choices were considered:

1. XLS file with one sheet per correction parameter. This solution was discarded because XLS files are not directly readable in all required systems. Only reliable way to access them is via ActiveX. First of all, that requires installed MS Office and secondly, it would not be possible to handle such files when batch-processing the data or performing uncertainty analysis on the supercomputers which are typically using Linux OS. Furthermore the sheet organization of the data is not sufficient for the purpose.
2. Storing all the data in the something like INFO file or XML file. Such solution is possible because these formats allow to store anything in structured form however editing of large number of dependencies in such formats is not easy for non-programmer persons.
3. Combination of minimalistic human readable header such as INFO file and CSV tables with correction data (frequency/voltage/… dependence). This solution has advantage it requires minimum (or none) editing of the headers and all correction data can be stored as simple CSV tables which are editable in many programs and also readable in Excel, LV, CVI, Octave and Matlab.

For means of the TWM the third (iii) option was chosen. Three types of corrections are supported by the TWM: Transducer correction, Digitizer correction and Channel correction.

#### Transducer corrections

The transducer corrections are relatively simple as they do not contain any links between two transducers or between transducer and another HW. Therefore the header looks like in the following example:

*// type of the correction:*

type:: shunt

*// name of the transducer:*

name:: Current shunt 1A

*// serial number of the transducer:*

serial number:: CMI/1A/1/13

*// identifier of the channel of the digitizer if the transducer was calibrated together with the digitizer:*

*// note: leave empty or remove if not needed*

linked to digitizer channel:: HP3458A, sn. MY45053095

*// nominal/DC ratio [V/A]:*

nominal ratio:: 0.6

*// combined amplitude and frequency dependence of the shunt*

*// note: one file CSV for each 'nominal amplitude'*

*// note: if just one file is entered, correction loader assumes no amplitude dependence*

**#startmatrix**:: frequency amplitude dependence paths

freq\_dep\_100mA.csv;

freq\_dep\_300mA.csv;

freq\_dep\_1A.csv;

**#endmatrix**:: frequency amplitude dependence paths

*// list of nominal amplitdues for each frequency dependence file*

**#startmatrix**:: nominal amplitude

0.1;

0.3;

1;

**#endmatrix**:: nominal amplitude

*// output dependence of impedance on frequency:*

*// note: used for loading corrections, needed at least for HF measurements*

*// note: the output impedance includes impedance of the output cable*

*// note: leave empty or remove if not used*

output impedance path::

The item ‘**type**’ is present in each correction file in order to identify to what this file applies. Current shunt must contain value ‘**shunt**’, voltage divider must contain value ‘**divider**’. Rest of the file has identical format for both divider and shunt. Each transducer file must contain item ‘**nominal ratio**’ which defines frequency independent nominal ratio of transducer (nominal resistance for a shunt and intput-to-output nominal ratio for a divider). Next, a combined frequency and amplitude dependence of the transducer in form of the relative transfer is entered as a list of simple CSV table files (**frequency amplitude dependence paths**). One file for each ‘**nominal amplitude**’. Format of each of the CSV files is following:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | | | | |
|  |  |  |  |  |
| f | Gain | arg | u(gain) | u(arg) |
| Hz | V/V | rad | V/V | rad |
| 0 | 1 | 0 | 1e-5 | 0 |
| 100 | 1.000001 | 5e-6 | 1e-5 | 2e-6 |
| 1000 | 1.000011 | 50e-6 | 1e-5 | 20e-6 |
| … | … | … | … | … |

First row of the CSV dependence is always description of the file. Next row is empty for better readability. Following two rows are defining names of the columns. Next the dependence data comes. For frequency/amplitude dependence the file format is in form of a gain correction ‘**gain**’ and a phase correction ‘**arg**’. Additionally two more columns with the absolute uncertainties of the parameters are added. Note this table may be in future split into separate magnitude and phase transfers having different frequency/amplitude steps. The correction loader will process it into a single dependence.

Last component of the correction is output impedance of the transducer. This is parameter that should be included for high frequency measurements. If the output impedance of the transducer and the input impedance of the digitizer channel are known, it is possible to calculate gain and phase correction for the loading effect. This parameter may be omitted if the transducer was calibrated together with the digitizer channel. In such case the correction file should contain optional channel identifier ‘**linked to digitizer channel**’ in order to limit the correction file use for a particular digitizer channel. Format of the impedance file is as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | | | | |
|  |  |  |  |  |
| f | Rs | Ls | u(Rs) | u(Ls) |
| Hz | Ohm | H | Ohm | H |
| 0 | 1e6 | 180e-9 | 100 | 20e-9 |
| 100 | 999e3 | 179e-9 | 500 | 20e-9 |
| 1000 | 993e3 | 183e-9 | 500 | 20e-9 |
| … | … | … | … | … |

The files structure of the transducer corrections in the measurement folder is following:

**Transducer corrections (./TRANSDUCER/):**

T01/**transducer\_1.info** *- transducer 1 header*

T01/tables/**\*.csv** *- transducer 1 CSV files with dependencies*

T02/**transducer\_2.info** *- transducer 2 header*

T02/tables/**\*.csv** *- transducer 2 CSV files with dependencies*

… …

The transducer correction folders will be always renamed in to the ‘**Txx’** format, when copied from the calibration data folder because multiple channels can share the same correction file so there may be a folder name collision.

#### Digitizer corrections

The correction of the digitizer and its channels is more complicated. Special care has been taken in order to make it both versatile and also simple enough to enable editing for less skilled users. It consists of the two parts: (i) Definition of the whole digitizer (interchannel corrections), (ii) definition of the particular channels (corrections that are independent to another channel or HW). Example of the digitizer correction header file is show in the following text:

*// correction type:*

type:: digitizer

*// description of the digitizer corrections:*

name:: Demonstration corrections for setup with two 3458A digitizers

*// names of the channels as they appear in the digitizer identification:*

*// these are exact unique names of the channels in the order that will be loaded to the SW*

**#startmatrix**:: channel identifiers

HP3458A, sn. MY45053095;

HP3458A, sn. MY45053104;

**#endmatrix**:: channel identifiers

*// relative links to the files with channel corrections for each channel:*

**#startmatrix**:: channel correction paths

..\channel\_MY45053095\HP3458\_MY45053095.info;

..\channel\_MY45053104\HP3458\_MY45053104.info;

**#endmatrix**:: channel correction paths

*// definition of ANY correction, in this case interchannel timeshift:*

**#startsection**:: interchannel timeshift

*// 2D array with the list of values of the correction:*

*// rows: vectors of correction values, eg. relative timeshifts to the first channel*

*// columns: primary parameter (synchronization mode)*

**#startmatrix**:: value

**0.0; 0.1; 0.2;**

**0.0; 0.1; 0.2;**

**0.0; 0.01; 0.02;**

**#endmatrix**:: value

*// uncertainty (same rules as 'value'):*

**#startmatrix**:: uncertainty

**0.0; 0.001; 0.001;**

**0.0; 0.001; 0.001;**

**0.0; 0.001; 0.001;**

**#endmatrix**:: uncertainty

*// --- Filtering of the correction by HW attributes: ---*

*// this is the list of channel specific attributes for which the correction is valid*

*// anything put here will be checked with the digitizer setup stored in the header file*

*// of the measurement and if it does not match, the loader will return an error*

**#startmatrix**:: valid for attributes

sampling mode;

**#endmatrix**:: valid for attributes

*// list of allowed values of attribute 1 (eg.: sampling mode):*

**#startmatrix**:: sampling mode

DCV;

**#endmatrix**:: sampling mode

*// --- List of parameters on which the correction values depends: ---*

*// primary parameter (remove if not used):*

**#startsection**:: primary parameter

*// name of the HW parameter:*

*// note: it must be exact name of the parameter that appears in measurement header*

name:: synchronization mode

*// is this parameter interpolable?*

*// note: set to 0 or remove if not interpolable*

interpolable:: 0

*// list of supported values of a primary parameter on which the correction depends:*

*// eg.: synchronization modes of the 3458A multimeters*

**#startmatrix**:: value

MASTER-SLAVE, MASTER clocked by TIMER;

MASTER-SLAVE, MASTER clocked by AWG;

ALL clocked by AWG;

**#endmatrix**:: value

**#endsection**:: primary parameter

**#endsection**:: interchannel timeshift

First item is ‘**type**’ which must be ‘**digitizer**’ for digitizer correction. Next there is a ‘**name**’ which is just a title that will be displayed in the TWM when the file is loaded. Next there is a list of a digitizer channel identifiers ‘**channel identifiers**’ exactly as they are returned during the instrument identification in the TWM. These are used to filter the correction file only for particular instruments. Next item is ‘**channel correction paths**’ which are relative links to the files with the channel corrections, one for each channel.

Next the file may contain any number of corrections. The corrections structure is designed so it allows following: (i) filtering the correction file by attributes of the digitizer, (ii) automatic selection or interpolation of the correction data by the configuration (parameters) of the digitizer.

Filtering (i) is used to inhibit the use the correction data when the digitizer has different configuration than was used for the calibration. The filter is defined by optional matrix ‘**valid for attributes**’. This matrix can contain any number of attributes from the measurement header. In the example above the filter is set to ‘**sampling mode**’ and the valid values of this attribute are listed in the matrix ‘**sampling mode**’, i.e. in this case only ‘**DCV**’ mode. If the correction loader finds out the attribute in the measurement header is not present or have different value, the loader returns error.

Automatic selection/interpolation (ii) of the correction data is used to select or interpolate multiple calibration values based on the value(s) of up to two parameters from the measurement header. The parameters, if defined, are in the optional sections ‘**primary parameter**’ and ‘**secondary parameter**’. If there is no parameter defined, the correction loader returns whole matrix ‘**value**’. If primary parameter is defined, it selects (or interpolates) a row of the matrix ‘**value**’ so a vector of values is returned. If the ‘**secondary parameter**’ is also defined, it will be used to select (or interpolate) a column of the matrix ‘**value**’. Each correction section can also contain matrix ‘**uncertainty**’ which is complementary to the ‘**value**’ so it will be selected or interpolated in the same way. Each parameter contains ‘**name**’ of the parameter from the measurement header. Then it contains matrix ‘**value**’ with nominal values of the parameter. These are used to select the row/column of the correction or are used as independent variable for interpolation, when the ‘**interpolable:: 1**’ is present in the parameter section. The values of the correction itself can be entered directly as numeric values as in the example above, or can be relative paths to the CSV files. In case of the CSV files, each CSV file contains a 1D dependence (frequency, amplitude, …). The correction loader will select/interpolate the CSV dependencies in the same manner as numeric values except is will always return 1D dependence. The uncertainty in case of the CSV files is located in the CSV file directly exactly as in the transducer correction.

Note the only values that needs to be modified when performing recalibration of the digitizer are marked in **red** color. The rest will stay unchanged and will come from a prepared template so the user don not need to write the whole structure.

Files structure of the digitizer correction is following:

**Digitizer corrections (./DIGITIZER/):**

correction\_name/**correction\_name.info** *- digitizer correction header*

correction\_name/tables/**\*.csv** *- CSV tables with correction dependencies*

channel\_1/**channel\_1.info** *- channel 1 correction*

channel\_1/tables/**\*.csv** *- channel 1 CSV tables with correction dependencies*

channel\_2/**channel\_2.info** *- channel 1 correction*

channel\_2/tables/**\*.csv** *- channel 1 CSV tables with correction dependencies*

…

The **red** elements only are related to the digitizer correction itself. The rest of elements are the channel corrections.

#### Channel corrections

Channel corrections define corrections that apply only to a single channel. Example of the channel correction file:

*// type of the correction*

type:: channel

*// correction name string*

name:: Channel correction HP3458A, sn. MY45053095

*// device/channel identification as it appears in the digitizer identification*

*// note: leave empty or remove if this correction should be independent of the instrument/channel*

channel identifier:: HP3458A, sn. MY45053095

**#startsection**:: nominal gain

*// 2D array with the list of values of the correction:*

*// rows: primary parameter (range)*

*// columns: secondary parameter (unused)*

**#startmatrix**:: value

**1.000005;**

**1.000003;**

**1.000006;**

**#endmatrix**:: value

*// uncertainty (same rules as 'value'):*

**#startmatrix**:: uncertainty

**0.000003;**

**0.000003;**

**0.000003;**

**#endmatrix**:: uncertainty

*// --- List of parameters on which the correction values depends: ---*

*// primary parameter (remove if not used):*

**#startsection**:: primary parameter

*// name of the HW parameter:*

*// note: it must be exact name of the parameter that appears in measurement header*

name:: voltage ranges [V]

*// list of supported values of a primary parameter on which the correction depends*

*// eg.: voltage range of the DMM*

**#startmatrix**:: value

**1;**

**10;**

**100;**

**#endmatrix**:: value

**#endsection**:: primary parameter

**#endsection**:: nominal gain

**#startsection**:: frequency dependence

*// 2D array with the list of values of the correction*

*// rows: primary parameter (aperture)*

*// columns: secondary parameter (range)*

*// note: in this case the values of the correction are stored in the CSV files*

**#startmatrix**:: value

**tables/fdep\_rng1V\_aper1u.csv; tables/fdep\_rng10V\_aper1u.csv;**

**tables/fdep\_rng1V\_aper10u.csv; tables/fdep\_rng10V\_aper10u.csv;**

**tables/fdep\_rng1V\_aper100u.csv; tables/fdep\_rng10V\_aper100u.csv;**

**#endmatrix**:: value

*// --- List of parameters on which the correction values depends: ---*

*// primary parameter (remove if not used):*

**#startsection**:: primary parameter

*// name of the HW parameter:*

*// note: it must be exact name of the parameter that appears in measurement header*

name:: aperture [s]

*// is this parameter interpolable?*

*// note: set to 0 or remove if not interpolable*

interpolable:: 1

*// list of supported values of a primary parameter on which the correction depends*

*// eg.: voltage range of the DMM*

**#startmatrix**:: value

**1e-6;**

**1e-5;**

**1e-4;**

**#endmatrix**:: value

**#endsection**:: primary parameter

*// secondary parameter (remove if not used):*

**#startsection**:: secondary parameter

*// name of the HW parameter:*

*// note: it must be exact name of the parameter that appears in measurement header*

name:: voltage ranges [V]

*// list of supported values of a primary parameter on which the correction depends*

*// eg.: voltage range of the DMM*

**#startmatrix**:: value

**1;**

**10;**

**#endmatrix**:: value

**#endsection**:: secondary parameter

**#endsection**:: frequency dependence

**#startsection**:: input admittance

*// 2D array with the list of values of the correction:*

*// rows: primary parameter (unused)*

*// columns: secondary parameter (unused)*

*// note: in this case the correction is independent of any parameters but the on in the CSV*

**#startmatrix**:: value

**tables\_input\_Y.csv;**

**#endmatrix**:: value

**#endsection**:: input admittance

The example above shows channel correction for DMM 3458A. First item is ‘**type**’ set to ‘**channel**’ to identify channel correction. Next there is a ‘**name**’ item which holds the name of the correction. Next, there is optional parameter ‘**channel identifier**’ which says to what channel the correction applies. It is optional so if it is left empty or not present, the loader of the correction won’t check the digitizer identification. Then the file contains any number of corrections in the same format as the digitizer correction. Note the filtering of the correction by attributes was left out in order to make the example more readable.

The file structure of the channel corrections:

**Digitizer corrections (./DIGITIZER/):**

correction\_name/**correction\_name.info** *- digitizer correction header*

correction\_name/tables/**\*.csv** *- CSV tables with correction dependencies*

channel\_1/**channel\_1.info** *- channel 1 correction*

channel\_1/tables/**\*.csv** *- channel 1 CSV tables with correction dependencies*

channel\_2/**channel\_2.info** *- channel 1 correction*

channel\_2/tables/**\*.csv** *- channel 1 CSV tables with correction dependencies*

…

Note only the **red** items are related to the channel correction, rest is the digitizer correction.

## Data interchange format between GUI and Calculation module

Data exchange between the GUI of TWM and Processing Module is realized via the INFO files. All calculations will be performed via the QWTB toolbox. The QWTB toolbox contains a lot of algorithms. Most of them are not suitable for the TWM so the TWM will contain a database of supported algorithms and their configurations. The database is stored in the INFO format. Format of the database file is following:

*// filter of the algorithms*

type:: qwtb list

*// === list of the supported algorithms ===*

*// note: enter algorithm ID's, e.g.: PSFE, SFDR, ...*

**#startmatrix**:: list of supported algorithms

PSFE;

SFDR;

SP-FFT;

**#endmatrix**:: list of supported algorithms

*// === setup for the particular algorithms ===*

*// These are optional sections, one for each algorithm. Name of the section must*

*// be equal to the value in the 'list of supported algorithms'. These are used*

*// to configurate the algorithm behaviour.*

*//*

*// parameters:*

*// exclude outputs: matrix of output quantities that will be excluded*

*// from display (usually time vector or frequency)*

*// graphs: 2D matrix of graph-like outputs (frequency dependence, ...), one row per graph,*

*// column: x; y*

*// example: f; A;*

*// spectrum: output quantity that will be displayed as default frequency spectrum*

*// number formats: 2D matrix of configurations for particular variables,*

*// one row per variable, columns:*

*// variable name; format specifier; minimum abs. uncertainty; minimum rel. uncertainty;*

*//*

*// variable name - name of the output variable*

*// format specifier - number format:*

*// 'f': float (no exponent)*

*// 'si': SI prefix*

*// minimum abs. uncertainty - minimum absolute uncertainty of the quantity*

*// - this will have effect in case no uncertainty is available*

*// minimum rel. uncertainty - minimum relative uncertainty of the quantity (unit-less)*

*// - this will have effect in case no uncertainty is available*

*//*

*// example: f; si; 1e-6; 0.0001;*

*//*

**#startsection**:: SP-FFT

**#startmatrix**:: exclude outputs

f;

**#endmatrix**:: exclude outputs

**#startmatrix**:: graphs

f; A;

f; ph;

**#endmatrix**:: graphs

spectrum:: A

**#startmatrix**:: number formats

f; si; 1e-6; 1e-6;

A; si; 1e-6; 1e-6;

ph; f; 1e-6; 1e-6;

**#endmatrix**:: number formats

**#endsection**:: SP-FFT

The ‘**list of supported algorithms**’ contains IDs of the algorithms from the QWTB toolbox. Only these will be visible in the TWM GUI. Optionally, each algorithm can contain configuration placed in the section named according the algorithm’s ID. The configuration is used only for the displaying of the results in GUI, not for the calculation itself. Its format is preliminary and will be most likely modified along with development of the TWM GUI.

The TWM will define the algorithm to be used for the calculation and its parameters by a following section of an INFO file:

*// --- copy of the QWTB algorithm setup:*

**#startsection**:: QWTB calculation

*// ID of the QWTB algorithm:*

algorithm id:: SP-FFT

*// calculate result for each averaging cycle (0) or calculate all averaging cycles at once (1):*

calculate whole average at once:: 0

*// list of algorithm parameters:*

**#startmatrix**:: list of parameter names

window

**#endmatrix**:: list of parameter names

*// parameter(s) data:*

**#startmatrix**:: window

hann;

**#endmatrix**:: window

*// algorithm configuration:*

**#startsection**:: algorithm configuration

**#startmatrix**:: exclude outputs

f;

**#endmatrix**:: exclude outputs

**#startmatrix**:: graphs

f; A;

f; ph;

**#endmatrix**:: graphs

spectrum:: A

**#startmatrix**:: number formats

f; si; 1e-6; 1e-6;

A; si; 1e-6; 1e-6;

ph; f; 1e-6; 1e-6;

**#endmatrix**:: number formats

**#endsection**:: algorithm configuration

**#endsection**:: QWTB calculation

It is not yet decided where this section will be placed. Possibly it will part of the measurement header or separate temporary file stored in the measurement folder. The section contains QWTB algorithm’s id, parameters entered by used in the TWM GUI and it will always contain a copy of the algorithm configuration from the algorithms database, as shown in the example.

The caller function of the QWTB algorithm will load the measurement header and data (see above), the ‘**QWTB calculation**’ section and will execute the calculation. After the execution of the QWTB algorithm, it will store the results of the calculation into the folder ‘**RESULTS**’ in the measurement folder:

**Results (./RESULTS/):**

**\*.info** *- algorithm calculated results*

**\*.mat** *- algorithm calculated results*

The caller of the QWTB toolbox will store the calculated variables into the INFO file and complementary MAT file of the same name. Naming rules are derived from the names of the records:

**QWTB toolbox result:**

**ALGID-G0001-A0001.info** *- algorithm calculated results*

**ALGID-G0001-A0001.mat** *- algorithm calculated results*

Note the MAT file is optional and will be created automatically if the results are too large for INFO text format. That may happen if the algorithm returns e.g. spectrum which may contain several millions of values. In such case the INFO file will contain just a link to the MAT file and a name of the variable inside MAT file which holds the data. Example of the result data (**\*.info**):

*// --- copy of the QWTB algorithm setup:*

**#startsection**:: QWTB parameters

*// ID of the QWTB algorithm:*

algorithm id:: SP-FFT

*// calculate result for each averaging cycle (0) or calculate all averaging cycles at once (1):*

calculate whole average at once:: 0

*// list of algorithm parameters:*

**#startmatrix**:: list of parameter names

window

**#endmatrix**:: list of parameter names

*// parameter(s) data:*

**#startmatrix**:: window

hann;

**#endmatrix**:: window

**#endsection**:: QWTB parameters

*// --- list of phases/channels for which the QWTB algorithm was executed:*

**#startmatrix**:: list

u1;

u2;

**#endmatrix**:: list

*// --- calculated data of the phase/channel 'u1':*

**#startsection**:: u1

*// index of the digitizer's phase/channel:*

phase index:: 1

*// tag(s) of the channels related to the phase/channel (e.g.: u1; i1 for phase L1):*

**#startmatrix**:: channel tag

u1;

**#endmatrix**:: channel tag

*// names of the output varibles of the QWTB algorithm:*

**#startmatrix**:: variable names

f;

A;

rms;

**#endmatrix**:: variable names

*// data for the variable 'f':*

**#startsection**:: f

*// name:*

name:: f

*// description:*

description:: Frequency series

*// dimensions of the variable (Matlab's size() command):*

**#startmatrix**:: dimensions

1; 5000;

**#endmatrix**:: dimensions

*// name of the MAT file's variable with the data:*

MAT file variable - value:: f\_v

**#endsection**:: f

*// data for variable 'A':*

**#startsection**:: A

*// name:*

name:: A

*// description:*

description:: Amplitude spectrum

*// dimensions of the variable (Matlab's size() command):*

**#startmatrix**:: dimensions

1; 5000;

**#endmatrix**:: dimensions

*// name of the MAT file's variable with the data:*

MAT file variable - value:: A\_v

*// name of the MAT file's variable with the uncertainty:*

MAT file variable - uncertainty:: A\_u

**#endsection**:: A

*// data for variable 'rms':*

**#startsection**:: rms

*// name:*

name:: rms

*// description:*

description:: RMS value

*// dimensions of the variable (Matlab's size() command):*

**#startmatrix**:: dimensions

1; 1;

**#endmatrix**:: dimensions

*// matrix with values of the variable:*

**#startmatrix**:: value

1.00053;

**#endmatrix**:: value

*// matrix with uncertainties of the variable (optional):*

**#startmatrix**:: uncertainty

0.00028;

**#endmatrix**:: uncertainty

**#endsection**:: rms

**#endsection**:: u1

*// --- calculated data of the phase/channel 'u2':*

**#startsection**:: u2

*// identical format to 'u1' ...*

**#endsection**:: u2

This file starts with a copy of the setup of the QWTB algorithm. It contains ID of the QWTB algorithm and a list and values of the algorithm’s parameters. Next the file contains a list of channels/phases. When the QWTB algorithm has just one input, it will be called for each channel of the digitizer and the **‘list’** will contain values such as: **u1; i1; u2; i2; …** If it has multiple inputs, such as for power calculation, the algorithm will be called for each group of the digitizer channels (one phase), such as **u1+i1** for phase one, **u2+i2** for phase two, etc. So the **‘list’** will contain values: **L1; L2; …** Assigning of the virtual digitizer’s channels to the phases is defined in the measurement header, section **‘corrections’**, subsection **‘channel phase indexes’**. Next the file contains section with calculated data for each phase/channel. For details, see the comments in the example.

Note the caller of the QWTB algorithm will also always create (or update) file ‘**results.info**’ in the measurement folder. Example of the results header file:

*// ID of the last calculated QWTB algorithm:*

last algorithm:: SP-FFT

*// ID of the last result for selected QWTB algorithm:*

last result id:: 3

*// List of calculated algorithms:*

**#startmatrix**:: algorithms

SP-FFT;

**#endmatrix**:: algorithms

*// List(s) of relative paths to the result files for each QWTB algorithm:*

**#startmatrix**:: SP-FFT

RESULTS\SP-FFT-G0001-A0001;

RESULTS\SP-FFT-G0001-A0002;

RESULTS\SP-FFT-G0001-A0003;

**#endmatrix**:: SP-FFT

The file contain list of ‘**algorithms**’ with all the calculated QWTB algorithms for the measurement. Next it contains a list(s) of calculated results for each algorithm. The file also contains information about last calculated algorithm and the last calculated result. This file will be used by the GUI of the TWM to identify available results and their locations in the measurement folder. When GUI needs to display the result, it will just call the loader of the result(s). The loader function will look into this list, select the results(s) for displaying and load the data. Next, it will format the result data and will return table of the formatted strings which will be displayed in the GUI. This way the workload of the GUI will be significantly reduced, as the GUI will just display table. Furthermore it can be shared for both LV and CVI implementation.