

**GUIDE FOR SAMPLING POWER AND POWER QUALITY MEASUREMENTS**

|  |
| --- |
|  |

CHAPTER 3

USER’S GUIDE FOR THE OPEN SOFTWARE TOOL

BLANK PAGE

**GLOSSARY**

**PQ** : Power Quality

To be completed…

BLANK PAGE

**OVERVIEW**

This document produces a user’s guide for the open software tool which will help end-users to acquaint themselves with the tool. The software is described in terms of the user interface and the configuration of the digitizers. The guide describes the integration of new digitizers into the system and the format of the calibration datasets for components used in the modular setup. It also provides information about power and PQ tests included in the initial database of calculation algorithms and about the implementation of new algorithms.

BLANK PAGE

**CONTENTS** To be updated !

[A INTRODUCTION 9](#_Toc6233695)

[A.1 TWM manual 9](#_Toc6233696)

[A.1.1 Installation 9](#_Toc6233697)

[A.1.1.a Installing TWM tool 9](#_Toc6233698)

[A.1.1.b Installing TWM prerequisites 10](#_Toc6233699)

[A.1.1.c Installing GNU Octave 10](#_Toc6233700)

[A.1.1.d Installing Matlab 11](#_Toc6233701)

[A.1.2 Before starting TWM – Set decimal separator 12](#_Toc6233702)

[A.1.3 Startup 12](#_Toc6233703)

[A.1.4 User guide 12](#_Toc6233704)

[A.1.4.a Configuring the processing environment 13](#_Toc6233705)

[A.1.4.b Configuring the digitizer 15](#_Toc6233706)

[A.1.4.c Configuring setup and corrections 18](#_Toc6233707)

[A.1.4.c.1 Transducer configuration 18](#_Toc6233708)

[A.1.4.c.2 Digitizer corrections 21](#_Toc6233709)

[A.1.4.d Configuring new measurement 23](#_Toc6233710)

[A.1.4.d.1 Sampling setup assistant 24](#_Toc6233711)

[A.1.4.d.2 Configuring data processing 26](#_Toc6233712)

[A.1.4.e Initiating measurement 29](#_Toc6233713)

[A.1.4.f Viewing the results 29](#_Toc6233714)

[A.1.4.g Batch processing 32](#_Toc6233715)

[A.1.5 Resources 32](#_Toc6233716)

[B SOFTWARE CONFIGURATION OF THE BUILT-IN DIGITIZERS 34](#_Toc6233717)

[C INTEGRATION OF NEW TYPES OF DIGITIZERS 35](#_Toc6233718)

[C.1 LabVIEW environment 35](#_Toc6233719)

[C.2 LabWindowsTM/CVI environment 35](#_Toc6233720)

[D POWER AND PQ TESTS 36](#_Toc6233721)

[D.1 Algorithms 36](#_Toc6233722)

[D.1.1 TWM-PSFE - Phase Sensitive Frequency Estimator 36](#_Toc6233723)

[D.1.2 TWM-MODTDPS - Modulation analyzer in Time Domain, by quadrature Phase Shifting 37](#_Toc6233724)

[D.1.3 TWM-FPNLSF - Four Parameter Non Linear Sine Fit 37](#_Toc6233725)

[D.1.4 TWM-HCRMS - Half Cycle RMS algorithm 37](#_Toc6233726)

[D.1.5 TWM-InDiSwell - Interruption, Dip, Swell event detector 38](#_Toc6233727)

[D.1.6 TWM-THDWFFT - THD from Windowed FFT 38](#_Toc6233728)

[D.1.7 TWM-PWRTDI - Power by Time Domain Integration 38](#_Toc6233729)

[D.1.8 TWM-WRMS - RMS value by Windowed Time Domain Integration 39](#_Toc6233730)

[D.1.9 TWM-WFFT - Windowed FFT spectrum analysis 39](#_Toc6233731)

[D.1.10 TWM-Flicker - Flicker algorithm 39](#_Toc6233732)

[D.1.11 TWM-MFSF - Multi-Frequency Sine Fit 40](#_Toc6233733)

[D.1.12 TWM-PWRFFT - Power by FFT 40](#_Toc6233734)

[D.1.13 TWM-InpZ – Estimator of Digitizer Input Impedance 40](#_Toc6233735)

[D.2 Integration of new algorithms 40](#_Toc6233736)

[E TITLE 1 43](#_Toc6233737)

[E.1 Title 2 43](#_Toc6233738)

[E.1.1 Title 3 43](#_Toc6233739)

[E.1.1.a Title 4 43](#_Toc6233740)

[E.1.1.a.1 Title 5 43](#_Toc6233741)

[E.1.1.a.1.1 Title 6 43](#_Toc6233742)

**FIGURES** To be updated !

[Figure A‑1: Location of TWM download on the TWM GitHub webpage. 10](#_Toc6233743)

[Figure A‑2: Downloading development version of TWM from GitHub webpage. 10](#_Toc6233744)

[Figure A‑3: Front panel of TWM. 13](#_Toc6233745)

[Figure A‑4: TWM processing environment configuration. 14](#_Toc6233746)

[Figure A‑5: TWM GNU Octave package assistant. 15](#_Toc6233747)

[Figure A‑6: TWM configuration panel for the niScope digitizers (for NI 5922). 16](#_Toc6233748)

[Figure A‑7: TWM configuration panel for Agilent 3458A sampling multimeters. 17](#_Toc6233749)

[Figure A‑8: TWM transducer configuration panel. 19](#_Toc6233750)

[Figure A‑9: TWM transducer correction editor. 20](#_Toc6233751)

[Figure A‑10: TWM CSV correction matrix editor. 21](#_Toc6233752)

[Figure A‑11: TWM digitizer corrections configuration panel. 22](#_Toc6233753)

[Figure A‑12: TWM digitizer correction editor. 23](#_Toc6233754)

[Figure A‑13: TWM new measurement configuration panel. 24](#_Toc6233755)

[Figure A‑14: TWM sampling setup assistant. 25](#_Toc6233756)

[Figure A‑15: TWM sampling setup assistant - fundamental frequency measurement panel. 25](#_Toc6233757)

[Figure A‑16: TWM processing configuration panel (raw Matlab commands mode). 27](#_Toc6233758)

[Figure A‑17: TWM processing configuration panel. 27](#_Toc6233759)

[Figure A‑18: TWM multicore processing setup panel. 28](#_Toc6233760)

[Figure A‑19: TWM multicore processing GNU Octave servers. 29](#_Toc6233761)

[Figure A‑20: TWM record viewer panel. 30](#_Toc6233762)

[Figure A‑21: TWM spectrum viewer panel. 30](#_Toc6233763)

[Figure A‑22: TWM processing results viewer. 31](#_Toc6233764)

[Figure A‑23: TWM QWTB batch processing panel. 32](#_Toc6233765)

[Figure E‑1 : Blue Exemple 43](#_Toc6233766)

[Figure E‑2 : Red Exemple 43](#_Toc6233767)

**TABLES** To be updated !

[Table A‑1: Typical GNU Octave packages required for TWM. 11](#_Toc6233768)

[Table E‑1 : Six\*Three exemple 43](#_Toc6233769)

[Table E‑2 : Three\*Two example 43](#_Toc6233770)

# INTRODUCTION

***WP3 - A3.3.1***

***CMI****, INRIM will produce guidance documentation on the start-up, including installation of the software, and the user interface for both LabVIEW and LabWindowsTM/CVI environments (the 2 GUIs) based on information from the report on the software tool from A2.4.5.*

*CMI will produce the guidance for the LabVIEW environment and INRIM for the LabWindowsTM/CVI environment, respectively.*

## TWM manual

Following chapters will describe installation and basic usage of the TWM tool [1].

### Installation

Installation of TWM consists of several components:

1. Download and unpacking of the TWM tool itself from GitHub [1].
2. Installation of the prerequisites for the TWM run, which are:
   1. LabVIEW 2013 Runtime Engine. This is needed to run any LabVIEW application.
   2. Drivers of all integrated instruments, which are currently:
      1. NI VISA drivers for handling GPIB bus.
      2. niScope drivers for handling NI digitizers.
3. Installation of the GNU Octave or Matlab for data processing (optional).

Steps 1) and 2) are mandatory. Current version of TWM is built with several digitizers integrated. Unfortunately it means the TWM application will not run unless there are drivers for all integrated. So it is necessary to install VISA drivers even for use with NI 5922 cards and “niScope” drivers even for use only with Agilent 3458A multimeters. Note this unpleasant “feature” will be fixed in the future by building several version of TWM having support only for selected instrument.

Step 3) is needed when processing of the recorder waveforms is to be performed in place. Without GNU Octave or Matlab, the TWM can only record the waveforms.

#### Installing TWM tool

TWM tool requires no installation. Its files just must be unpacked from the ZIP to any user folder, e.g.:

c:\Program Files (x86)\TWM

The build releases of TWM are relatively rare and usually contains version that was at least partially tested. It can be downloaded from GitHub webpage (see Figure A‑1).

However, user may also download development version which will run only with development version of LabVIEW installed on the computer. Note the TWM was intentionally developed in LabVIEW 2013 Base version. The development version can be obtained either by cloning the TWM Git or manually by downloading the Git as a ZIP file (see Figure A‑2).

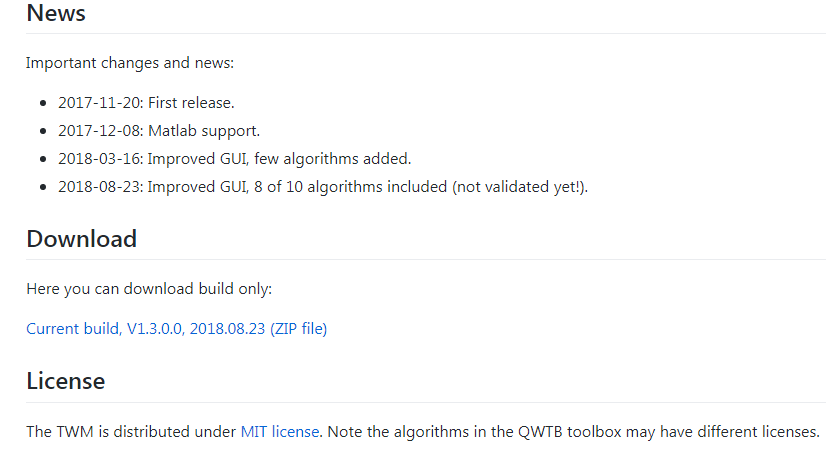


Figure A‑1: Location of TWM download on the TWM GitHub webpage.

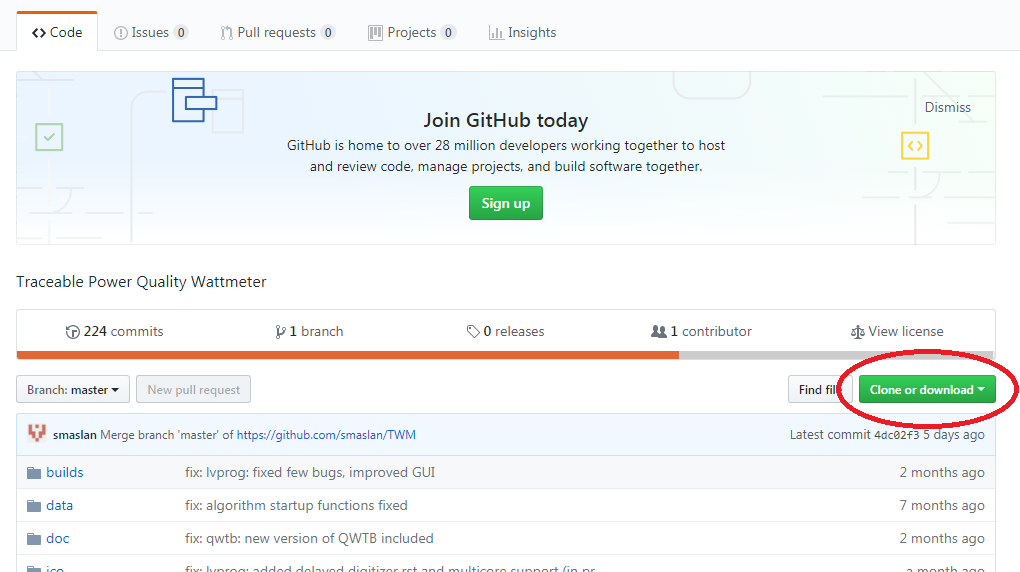


Figure A‑2: Downloading development version of TWM from GitHub webpage.

#### Installing TWM prerequisites

LabVIEW applications in general requires large amount of external libraries and drivers to run. First required is LabVIEW 2013 Runtime Engine 32bit [4]. The version must be 32bit as the TWM is built as a 32bit application.

Next component are the VISA drivers, which are needed for communication via GPIB. At least version 5.4 should be installed. The newer versions are partially compatible, see [5] for selection.

Last needed component are niScope drivers, which are used to communicate with NI 5922 digitizers. These supported versions are 4.0.5 to 16.1 (see [6] for details). *Note there is a also a build version made without niScope support, so this part can be optionally skipped.*

#### Installing GNU Octave

In order to enable calculations of the PQ parameters from the waveforms, user must install GNU Octave (or Matlab). The TWM was tested with Octave version 4.0.0 and later. Note the preferred version of GNU Octave is always 64bit, as it has much higher memory limit. That may be useful for some memory demanding algorithms.

The download and guidance to the GNU Octave can be found at [2]. Some of the algorithms do require additional packages to be installed and loaded to the GNU Octave after its startup. This is typically ensured by creating a startup file in the user folder, *but see section A.1.4.a Configuring processing environment for alternative approach!* GNU Octave will always execute the sequence of the command on the startup. E.g. in Windows, for the user named “user”, the file location will be:

c:\Documents and Settings\user\.octaverc

Note the file name actually starts with the dot! Example of the content of the file may be following:

more off;

pkg load io;

pkg load optim;

pkg load signal;

pkg load statistics;

pkg load outliers;

graphics\_toolkit('gnuplot');

One of the main uses of this file is usually to load the packages to the memory on startup. Note the packages must be installed first. Some distributions of the GNU Octave ask user to select the packages to install during the installation. Other distributions are plain without packages and those must be downloaded and installed separately following the documentation [2]. Typical packages needed for the currently implemented algorithms are listed in Table A‑1.

Table A‑1: Typical GNU Octave packages required for TWM.

|  |  |
| --- | --- |
| **Package** | **Description** |
| io | Input/output to external formats (CSV files, etc.) |
| optim | Non-linear optimization toolkit. |
| outliers | Outlier tests and removal. |
| signal | Signal processing routines (e.g. filters). |
| statistics | Additional statistical functions. |
| multicore | Multicore calculations. |
| golpi | More effective data exchange between LabVIEW and GNU Octave. |

Package “multicore” is not necessary, however it can be used to significantly speedup Monte Carlo uncertainty evaluation of some of the algorithms. Package “golpi” is designed to speed up the data interchange between GNU Octave and LabVIEW GOLPI library [7]. Note the GOLPI library should offer automatic installation of the “golpi” package to the user when the mode of communication is selected (see below).

#### Installing Matlab

Matlab [3] is alternative processing environment for the TWM, however it should be 100% compatible. TWM was tested with versions 2008b and later. Typical Matlab installation should contain all required packages and they are loaded automatically, so no additional actions after installation should be required.

### Before starting TWM – Set decimal separator

TWM internally uses GNU Ocatve or Matlab for processing and result formatting. LabVIEW can switch decimal separator to either dot or comma, whereas Octave or Matlab cannot. This may result in problems when trying to display or copy results when decimal separator is set to comma. Although some changes in TWM were made to prevent these problems, it may still fail in some case, so only safe solutions is to **change Windows setup to decimal dot** (Windows Control Panel | Region and Language | Formats | Additional settings).

### Startup

When all required components are installed, the TWM can be started by its executable “TWM.exe”. When no component is missing, the front panel should appear with no error messages. If some driver is missing, LabVIEW will throw an error with explanation which component cannot be located. Typical missing components are “niScope.dll”, “niTclk.dll” (part of “niScope” drivers) or VISA drivers. If TWM requests the libraries, follow the installation guidance in section A.1.1. TWM requires no other configurations prior starting the application itself. In some cases TWM may start, but error field will contain an error for a first start. This may be caused by residue of setting from a build computer. The problem should stop once the TWM is restarted.

### User guide

Main panel of TWM is shown in Figure A‑3. All subpanels with particular configurations can be invoked from the main panel. Error indicator at the bottom will show eventual error message of the TWM. User must configure the system before any measurement can be taken. This is done by buttons Digitizer for digitizer selection and configuration, button Octave/Matlab for processing environment selection and button HW corrections for selection of the transducer and digitizer connection and correction files.

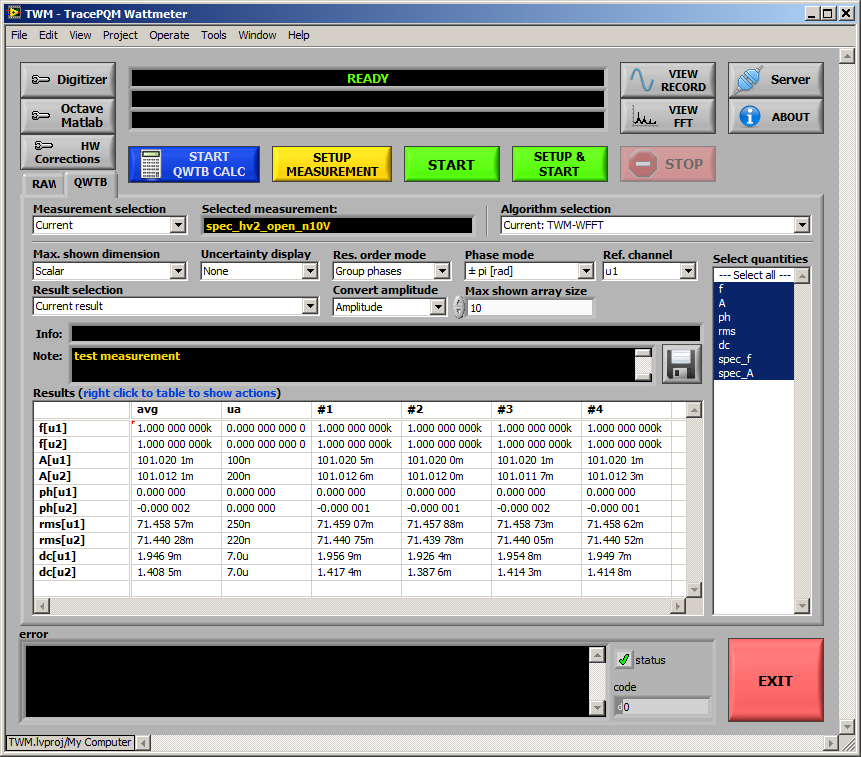


Figure A‑3: Front panel of TWM.

#### Configuring the processing environment

First step before any processing can be done is to setup the processing environment, i.e. Matlab or GNU Octave by pressing button Octave/Matlab on the front panel. This will invoke panel shown in Figure A‑4. The main option is checkbox Enable Matlab script mode. This will select Matlab for processing. If unchecked, the GNU Octave is selected. Matlab has no other options in the top frame. For GNU Octave user must manually list the GNU Octave binary path of the selected GNU Octave installation. Example for version up to 4.x.x is following:

C:\Octave\Octave-4.2.2\bin

Note the new version starting with Octave 5.x.x changed the internal location of the binary to either of following subfolders depending on 32/64-bit version selection:

C:\Octave\Octave-5.1.0.0\mingw32\bin

C:\Octave\Octave-5.1.0.0\mingw64\bin

Next, user may enable option Always clear function cache, which has effect only for debugging. This will force GNU Octave to reload the scripts before every command to ensure the changes in them are recognized (do not use for normal operation – it will only slow down operation). Another option is Use bitstream mode, which will enable faster communication between LabVIEW and GNU Octave. This requires the package “golpi” to be installed. TWM should offer user automatic installation. However, it is not necessary as there is mostly only small amount of data transferred between the environments. Last option is Show console window which will show GNU Octave console window for debugging. Note the Matlab will always show its console window. Do not close neither the Octave’s or Matlab’s console during TWM usage as it will terminate the application! Validity of the setup can be tested by button Restart.

Another option to be done is selection of the QWTB toolbox root folder path. Although TWM should work with main QWTB distribution, the TWM algorithms are distributed along with the TWM, so always set path to the local copy of QWTB in your TWM installation subfolder:

.\octprog\qwtb

This is essential as the downloaded TWM will contain the path that was valid at the compute that was used for building the application! Validity of the setup can be tested by TEST QWTB.

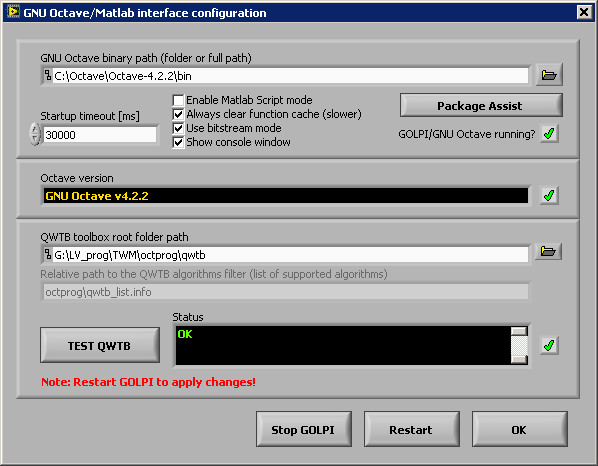


Figure A‑4: TWM processing environment configuration.

A special package assistant shown in Figure A‑5 was made to reduce complexity of initial GNU Octave setup. The panel can be invoked by pressing Package Assist button in GNU Octave/Matlab panel. The left list should show the installed packages in the selected installation of GNU Octave. The right panel shows which packages are already loaded after startup (caused by the startup file) or are selected to load by TWM. Select and shift packages you want to load on startup by right or left buttons. Note the right list will always contain at least the packages that are loaded by startup file “.octaverc”. Also note to apply the changes you must restart Octave in panel Figure A‑4. The selected packages to load should be automatically loaded after the TWM is restarted.

The assistant panel also contains button Install packages. This button will display open dialog which can be used to select downloaded package (extension \*.tar.gz) and TWM will try to install the package to the Octave. The success can be tested by pressing Reload Packages. The newly installed packed should appear and must be moved to right panel to be loaded after Octave restart.

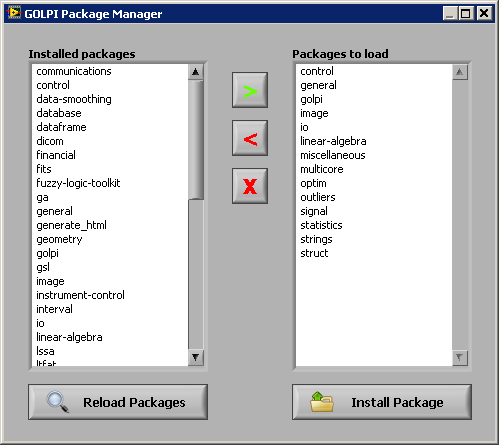


Figure A‑5: TWM GNU Octave package assistant.

#### Configuring the digitizer

The button Digitizer on the main TWM panel must be pressed in order to invoke Digitizer configuration panel. Example of the panels for the niScope and 3458A digitizers are shown in Figure A‑6 and Figure A‑7. The panel contains Digitizer type selector which is used to select the HW to be used for digitizing. Below is a text frame with brief description of the selected digitizer.

Next option On device close at the bottom-left of the panel is used to determine behavior of the TWM after the digitizers were used. Selection of Immediate reset will reset all HW related to the digitizer to the safe state. Selection of Delayed reset will do the same when the HW is not used for preset timeout. This second option is preferred way especially for the sampling multimeters 3458A as it saves the relays for repeated measurements. Option Force reset on open will issue reset command to the instruments related to the digitizer every time the digitizer is initialized. Normally this should be disabled.

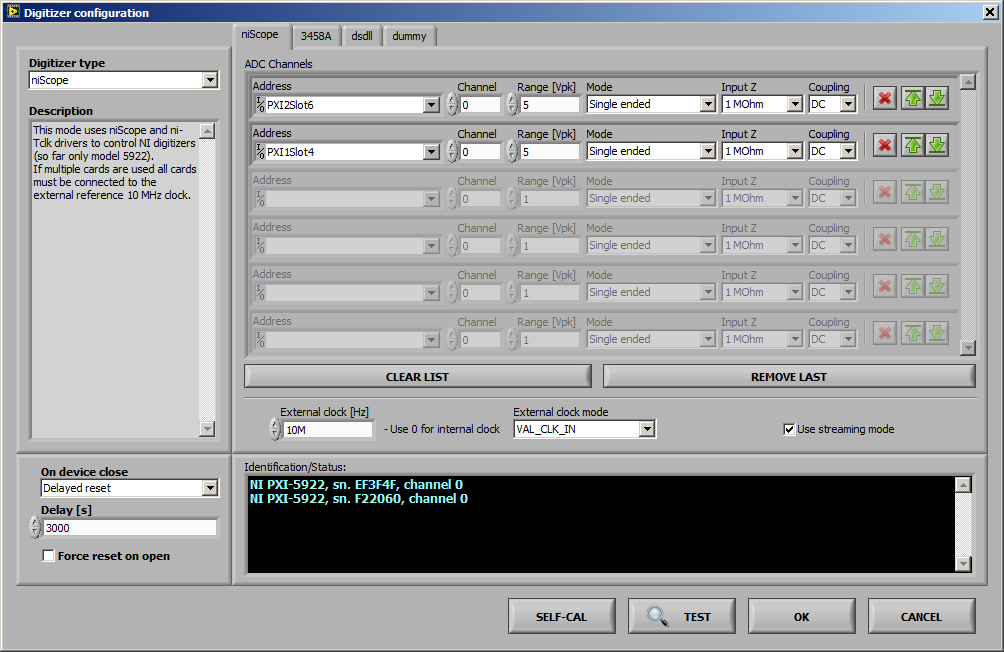


Figure A‑6: TWM configuration panel for the niScope digitizers (for NI 5922).

Each digitizer has one private page with specific options. The page for niScope digitizers is shown in Figure A‑6. User can select any number of physical channels of any number of digitizing cards by adding or removing rows to the table ADC Channels. Each virtual channel (row) contains Channel ID (note niScope counts channels from zero), input Range [Vpk], which is maximum applied voltage (e.g., 5 for range ±5 V), Input Z is input impedance of the channel and Coupling is mode of input coupling. The icons on the right of each row enables reordering or removal of row(s). Note the physical channels of the card(s) which are not selected are automatically set to ground to prevent buildup of noise at near channels. The driver supports PLL synchronization to the external 10 MHz source, which is needed for coherent sampling or when synchronizing multiple cards. Type in the nominal sync. frequency to the External clock [Hz] entry to enable the PLL. Use value of 0 to switch to the internal clock. Source of the synchronization clock can be selected by External clock mode. Selection VAL\_CLK\_IN refers to front “CLK IN” SMB of 5922. Options VAL\_PXI\_CLOCK should select internal chassis clock distribution. Last option is Use streaming mode, which will enable runtime streaming of the sample data from the cards to the disk storage. This option still uses the internal card memory, but it reads the sample data along with the sampling, so unless the disk storage is too slow, user may digitize indefinitely. Validity of the setup can be tested by the button TEST. Eventual problems will be shown in the black area below. The SELF-CAL button will initiate self-calibration routine of all selected digitizer cards.

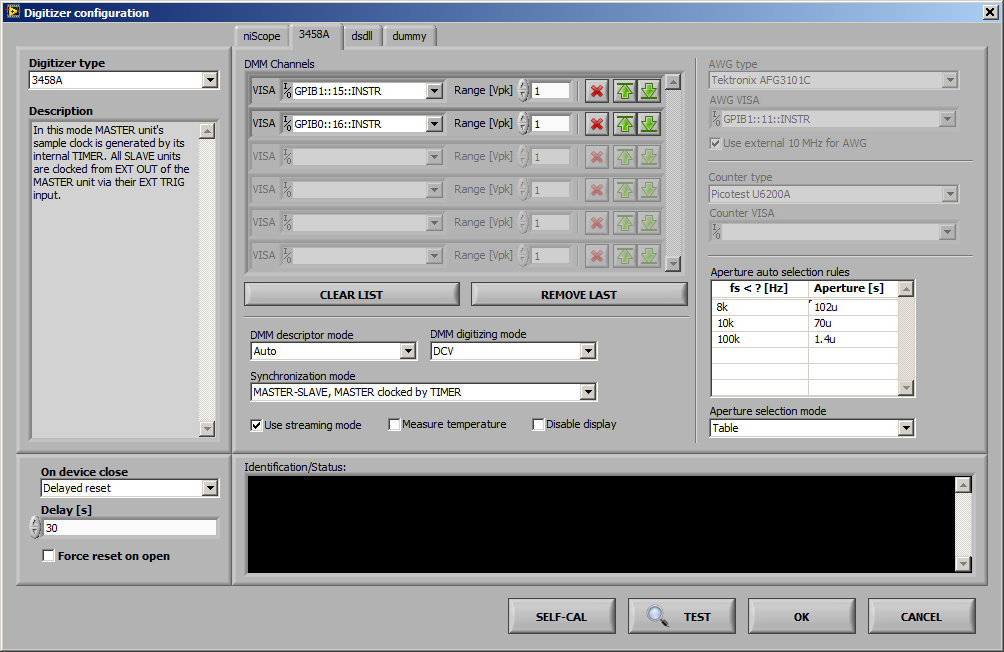


Figure A‑7: TWM configuration panel for Agilent 3458A sampling multimeters.

Configuration panel for the digitizers 3458A is shown in Figure A‑7. User may create any number of virtual channels by adding or removing rows to the table DMM Channels. VISA is visa address of the 3458A instruments. Range [Vpk] is nominal range of the DMM. Typical value for digitizing is 1 V. Color icons on the right enable reordering or removal of the row(s). Driver has no internal limitation of the channels count. Control DMM descriptor mode allows to change the way the multimeters are identified. Selection of Sequential will generate fake identifiers “HP3458A, channel 1”, “HP3458A, channel 2”, etc. These are identifiers that must match channel names in the digitizer correction files. Another option From calibration string means the TWM will scan the calibration string of each DMM (command “CALSTR?”) and if it discovers serial number in there, it will use it as an identifier. Using CALSTR for serial numbers is common practice for distinguishing the DMMs via GPIB. Selection of Auto will choose automatically with priority of calibration strings. Next option is DMM digitizing mode, which selects the digitizing method. Available modes are DCV, DSDC and DSAC (refer to 3458A manual for meaning). Option Synchronization mode determines the way how multiple DMMs are synchronized. Four selections are possible:

1. MASTER-SLAVE, MASTER clocked by TIMER: Master (first DMM in the list) clocks itself from internal TIMER. All other DMMs takes sample clock (EXT TRIG) from master’s EXT OUT. This option means sampling rate of the whole virtual digitizer is determined by the MASTER’s internal TIMER and thus by MASTER’s reference clock.
2. MASTER-SLAVE, MASTER clocked by AWG: Master EXT TRIG is clocked from the arbitrary waveform generator (AWG). All slaves are clocked via EXT TRIG from MASTER’s EXT OUT. The sampling clock is determined by the AWG.
3. All clocked by AWG: EXT TRIG of all DMMs are clocked from the AWG.
4. All clocked by TIMER: All DMM’s are clocked from their internal TIMERs. This special mode is dedicated for DMMs with modified internal reference clocks, which can be locked to the external 10 MHz source. The absolute time shift between the channels is determined by additional counter connected between EXT OUT outputs of the DMMs. Note this is intended for two DMMs only.

Typical selection when non-coherent sampling is acceptable is (i). For coherent sampling use modes (ii) or (iii). Note in modes (ii) and (iii) user must define AWG type and AWG VISA address and in mode (iv) user must define Counter type and Counter VISA address.

Note the modes (ii) and (iii) can be used with an external non-controlled source of sampling pulses, which may be e.g. some PLL synchronizing unit or other instrument’s output. In that case the AWG type has to be set to dummy, which means TWM will ignore it. The sample clock generator will be connected to the EXT TRIG of master (ii) or EXT TRIG of all DMMs (iii). Typical example of such operation is calibration of Fluke 6100 calibrator series, which are equipped by a “Sample Ref Output” connector. This output can be enabled and it outputs fundamental frequency multiplied by certain factor. It is convenient e.g. for achieving coherent sampling.

Next option is Use streaming mode, which will enable direct readout of the samples via GPIB without the limitation of the internal DMM memory. This will enable up to 16 MSamples per channel to be captured in one record. However, this mode requires one GPIB controller for each DMM and it may fail for the highest sampling rates. The fail will be indicated by beep and message “trigger too fast” while sampling. The streaming method is faster, as the data are read continuously. Another option is Measure temperature, which will read the internal DMM temperature once per record. Note it takes additional time as the reading is made via the main ADC. Last option is Disable display which will turn the CFL displays off while digitizing. It is recommended to reduce the noise.

Last option is selection mode of the DMM integration time (aperture) made by control Aperture selection mode. Three options are available:

1. Manual: user selects the aperture in the new measurement configuration.
2. Table: TWM selects the aperture based on the rules in table Aperture auto selection rules depending on the sampling frequency. This option is helpful to limit possible apertures to few calibrated values. The example shown in Figure A‑7 defines to select aperture of 102 µs up to 8 kSa/s sampling rate, 70 µs up to 10 kSa/s and 1.4 µs up to 100 kSa/s.
3. Maximum: TWM selects the maximum possible aperture based on empirical formula. Note each aperture needs separate calibration of DMMs, because residual errors of TWM changes e.g. when crossing 100 µs aperture.

Configuration of the virtual digitizer can be confirmed and tested by pressing TEST. Any error detected should appear in red color in the black area. Autocalibration of the DMMs can be initiated by the SELF-CAL button.

#### Configuring setup and corrections

TWM was designed to be able to correct each component of the measurement setup: (i) Digitizer; (ii) Transducer; (iii) Loading effects (e.g. cables). Corrections panel shown in Figure A‑8 can be invoked by pressing button HW Corrections on the main panel. The panel contains two pages. One for configuration of Transducers and another one for Digitizers corrections.

##### Transducer configuration

User must define connection of the transducers and their correction files before any meaningful measurement can be done. The format of the correction files is described in [13].

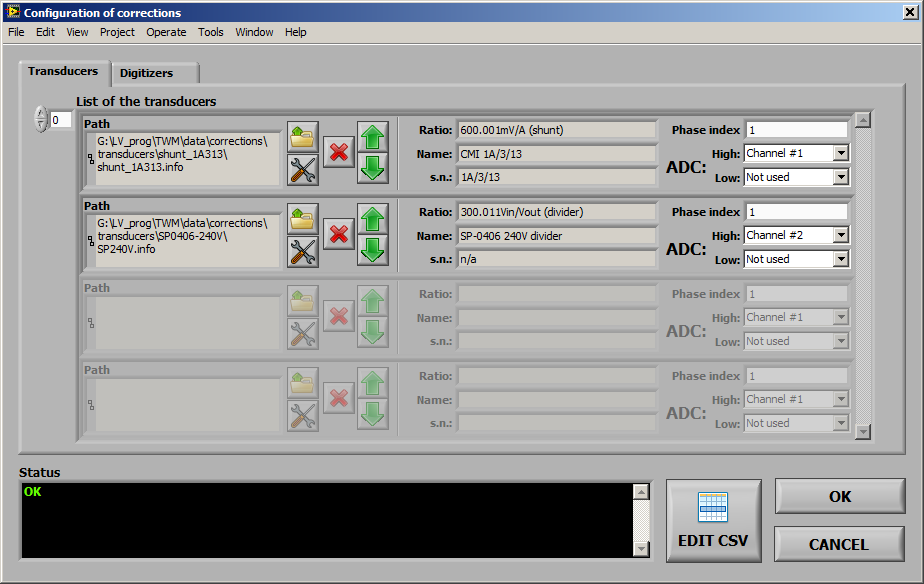


Figure A‑8: TWM transducer configuration panel.

The page Transducers allows user to create list of transducers. User may create multiple transducers by adding or removing the rows to the List of the transducers. Each row designates one transducer, which may be either for voltage or current channel. Open icon is used to select the transducer correction file (voltage divider or shunt). When the load is successful, the type and nominal ratio will be shown in the middle of the row. User must then assign Phase index to each transducer. This is integer number that is used by TWM to pair the voltage and current transducers for algorithms such as power. Selectors High and Low are used to map the transducer output(s) to particular digitizer virtual channels. High channel must be always assigned. For differentially connected transducers, user must also assign the Low side digitizer channel. For single ended mode assign Not used to the Low. Every time the change is made, TWM will check the validity and show either OK or error message in the black area. Note the TWM does not require the transducers to be defined for single input algorithms, such as PSFE. It will load default unity transfer transducer. But it is required for multi-input algorithms such as power, because TWM must know which is voltage and current and to which phase it belongs. Also note the TWM installation contains examples including “dummy” divider and shunt correction with unity transfers in the TWM folder:

./data/corrections/transducers

These should be used if no input correction is needed.

The transducer items in the List of the Transducers can be also edited directly from TWM. Pressing the Edit button will invoke a dialog shonw in Figure A‑9. The panel starts with selector of Correction file path. Change the path to load another file to the editor. Type defines type string of the transducer divider („shunt“ or „divider“). Name and Serial number are user descriptions of the correction. Digitizer channel identifier is optional name of a digitizer channel to which the transducer belongs. TWM will throw an error if non-empty and digitizer channel identifier does not match. Next, user can select Correction item to edit. Either numeric entry or CSV editor button EDIT CSV will be shown. The numeric values can be typed in directly to the Value and Uncertainty edits. CSV style correction is edited via EDIT CSV button or the relative link to the CSV file can be written directly to CSV table path (relative) edit. Note TWM is designed so it prefers CSV files related to the correction to be placed to the same folder (or subfolder) as the correction INFO file. Any changes done must be saved by SAVE CHANGES button before leaving.

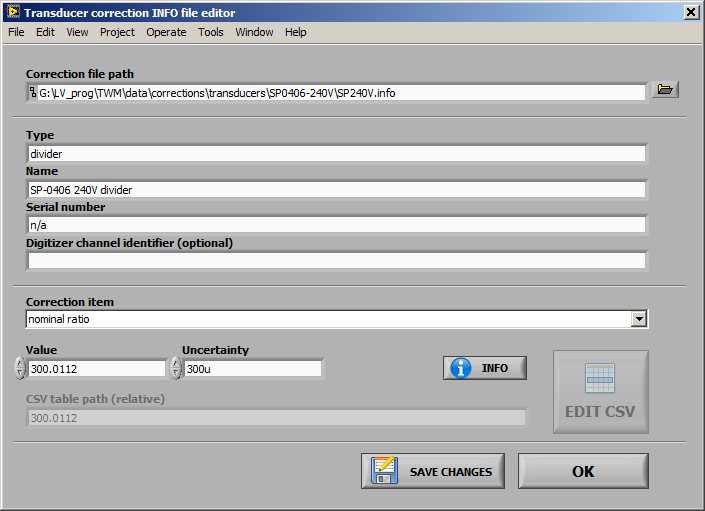


Figure A‑9: TWM transducer correction editor.

The CSV file editor example is shown in Figure A‑10. The path of selected CSV file is shown in the File path selector which can be also used to select another file directly in the panel. Editing the values is possible by typing into the cells or via the pop-up menu. Copy Data and Paste Data is possible between Excel style editors and the TWM CSV editor. Note the TWM will automatically try to convert all numeric decimal separators to a dot which is only accepted by TWM. In case of a new CSV creation, user may select pop-up item Create Template which should generate expected header of the required CSV table according to the correction manual [13]. Note any changes must be saved by SAVE button before leaving. If the CSV editor was called from a correction item editor, it will overwrite relative CSV path in the selected correction item by the saved CSV file path.

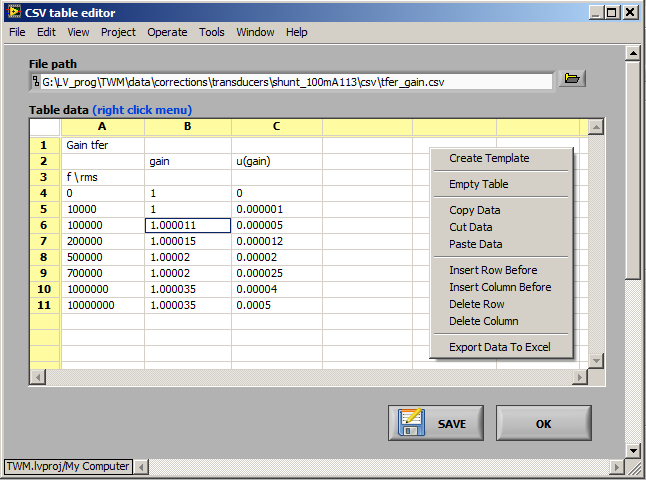


Figure A‑10: TWM CSV correction matrix editor.

##### Digitizer corrections

Digitizers page of corrections panel shown in Figure A‑11 is used for loading the digitizer correction file. The correction file is selected by pressing LOAD NEW CORRECTIONS. This is verified by pressing CHECK HW CONSISTENCY. The content of the correction file must match the configured digitizer, i.e. the channel names and order must be the same. RELOAD CORRECTIONS will just reload the same correction file in case it was modified. CLEAR CORRECTIONS is used to clear digitizer corrections. TWM will work without digitizer corrections selected normally, except no correction will be applied.

Last option is EDIT CORRECTION, which will invoke panel shown in Figure A‑12 for direct editing (or creation) of digitizer correction files. Correction file path entry is used for selection of correction INFO file. Change it to load another file if required. Channel identifiers is a list of digitizer channels for which the correction will apply. Optionally use pop-up menu to load channel names from currently connected digitizer. Channels list must contain one row for each channel. The items are relative paths to the INFO correction files of particular channels. Use pop-up menu to Insert, Remove or Edit Channel Correction items. Edit will invoke identical editor panel for a channel correction file. In channel editing mode the Channel identifiers will be disabled and Channel name filter will appear. This is optional item that may be used to restrict use of the correction file for particular digitizer. Leave it empty in not needed.

Main editor starts with the Correction item selector. The list should contain all available corrections for given correction type (digitizer or channel). Selecting the desired correction should show Value and Uncertainty matrices. For direct numeric corrections the numeric values are placed directly to the Value and Uncertainty matrices. For CSV style table corrections the Uncertainty matrix is ignored and Value matrix is used to define relative CSV matrix path. Use pop-up menu to Edit/Create CSV matrix. Note the Value and Uncertainty matrices allows Copy Data and Paste Data between Excel style editor and TWM. Also note TWM will automatically convert the decimal separators to dot. User can additionally restrict the correction for given attributes of the digitizer and make it dependent on Primary or Secondary parameter. Please refer to the corrections manual [13] for details. Pressing the INFO button should show brief description of the correction meaning. If you want to inhibit given correction without removing its data, use Disable correction item. Use button CLEAR to clear given correction data.

Any changes must be saved by pressing SAVE CHANGES before leaving the panel. Note when valid INFO path was selected during save, TWM will override caller’s correction path by the newly created/edited file automatically.

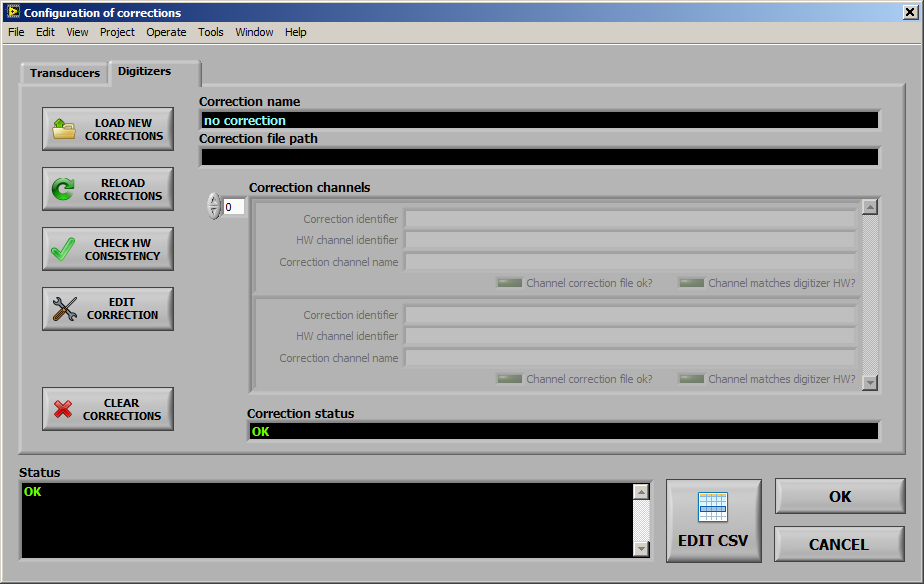


Figure A‑11: TWM digitizer corrections configuration panel.

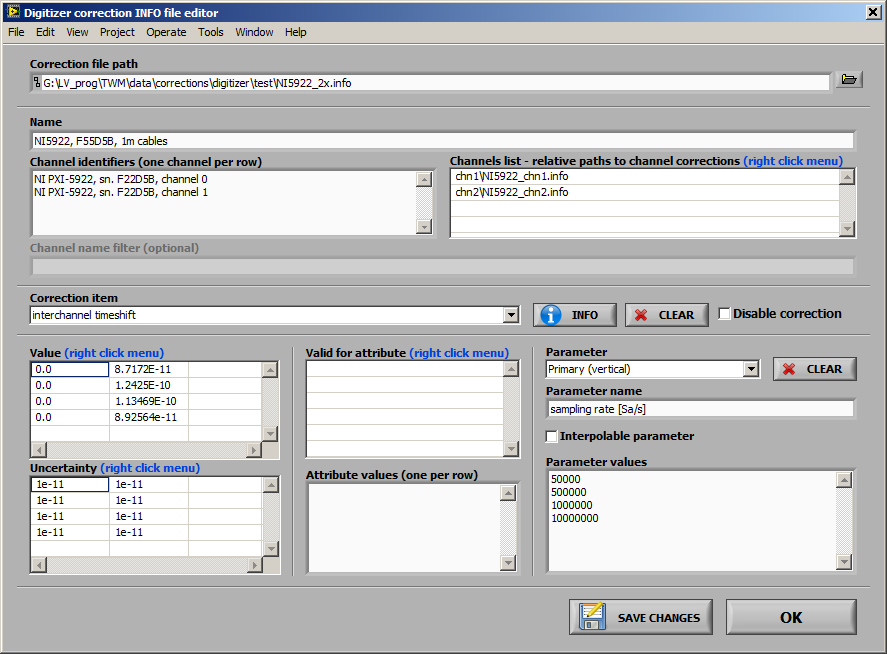


Figure A‑12: TWM digitizer correction editor.

#### Configuring new measurement

New measurement configuration can be invoked by button SETUP NEW MEASUREMENT or button SETUP&START. These will open panel shown in Figure A‑13. The panel is used to setup Sampling rate, Aperture time of the digitizer ADC, Samples count to record or Measurement time, measurement Repetition cycles, additional Repetition settling for extra delay between repetitions. Next group of options is Trigger mode and its parameters. Mostly it is not needed, so Immediate can be used. Note the mode Level is always derived from the first channel of the digitizer. This decision was made because of the DMM synchronization modes MASTER-SLAVE which do not allow principally different operation. The mode External is some form of external trigger input dependent on the selected digitizer. For DMMs it is EXT TRIG input of the MASTER DMM (if not used for AWG input). For NI 5922 it is the TRIG input on the first card in the digitizer channels list. The panel also contains Voltage range selection. Normally this is not needed, because the ranges can be set in the digitizer configuration, but the default ranges can be overridden from here, when other than Keep defaults option is selected. The TWM will allow user to set the same range to all virtual channels, or separately to odd and even (intended for voltage and current channels).

The mandatory part of this panel is setup of the destination folder for the measurement data. User must select some Folder for measurement data files to which the TWM will store the data and results. User must also choose a Measurement name base, which is subfolder to be created in the Folder for measurement data files. This name may contain variables as shown in the panel which will be replaced. Preview of the final path to the measurement is shown in the indicator Full measurement base path. Normally, if user does not require to archive the measured data for future processing, the Measurement name base may be still the same and TWM will simply overwrite the previous data by new ones (it will always ask to delete old data). By adding the %id% variable to the name, TWM will auto increment the Measurement number every time the measurement is taken, so the data will be archived in the Folder for measurement data files. User can also define Measurement comment to be stored with the data. Use Shift+Enter to break line in the comment editor.

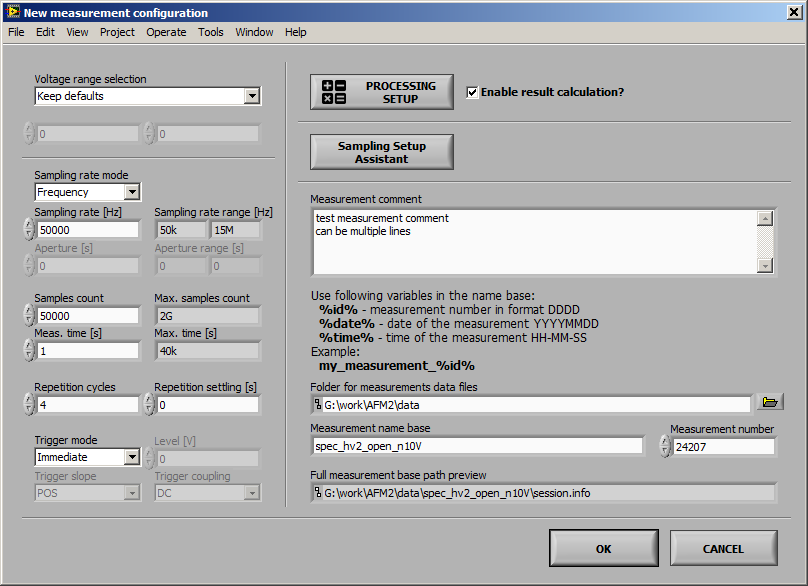


Figure A‑13: TWM new measurement configuration panel.

Processing setup and Sampling Setup Assistant are described in the following chapters. Option Enable result calculation must be checked to perform the data processing immediately after digitizing. Otherwise TWM will only record the data to the measurement folder(s) and the processing can be initiated afterwards.

When the setups are done, user may confirm by pressing OK or CANCEL to return back. If the panel was invoked by SETUP&START, the measurement will start immediately after OK.

##### Sampling setup assistant

In some cases user may need to setup coherent sampling. The New measurement configuration panel has integrated Sampling Setup Assistant for easing this task. Panel of the assistant is shown in Figure A‑14. Purpose of this panel is to measure or manually enter frequency of the fundamental component of the signal and calculate combination of the sampling rate and samples count which will result in coherent sampling. The frequency can be measured using subpanel shown in Figure A‑15 that can be invoked by pressing MEASURE. Alternatively, if it is known, it may be entered directly to the Reference frequency fr. The frequency can be multiplied by a factor P/Q and shifted by given offset to resulting Fundamental frequency f0. At this point user must enter search range of the sampling rate Minimum fs and fs max tolerance and total measurement time range Min integration t., Max integration t. or Min periods, Max periods counts. By pressing FIND ROUGH, the TWM will select nearest possible sampling rate and time, but not coherent. This is just to show user rough calculated parameters of the sampling such as DFT bin spacing, periods count, etc. By pressing FIND COHERENT, TWM will call experimental m-function that will try to find coherent setup in the set ranges. The function is quite complicated and may take a while, so Max timeout [s] control is used to set maximum time to spend on the calculation. Control Max setup error defines how accurately TWM searches the coherent setup. Note the digitizers may have very limited step of sampling period or frequency, so there may be no coherent setup for irrational fundamental frequencies f0. This may be prevented by selecting higher tolerance.

The subpanel for automatic frequency measurement of the reference frequency shown in Figure A‑15 is basically a reading loop for a selected counter. It will perform up to Max averages readings and calculates average frequency. When OK is pressed, the measured value will be transferred to the Sampling setup assistant panel.

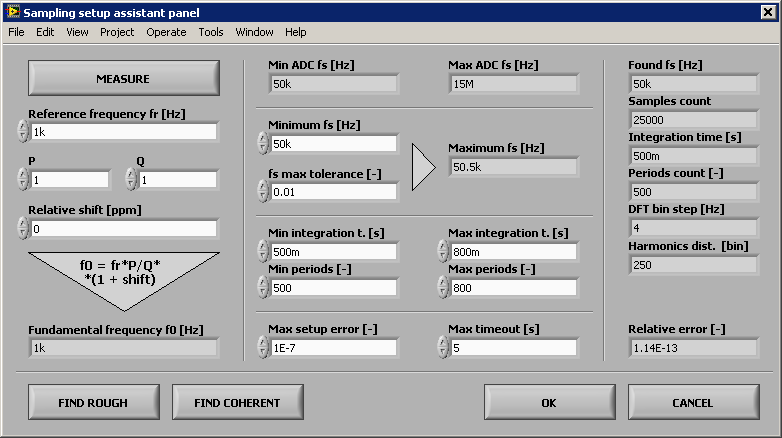


Figure A‑14: TWM sampling setup assistant.

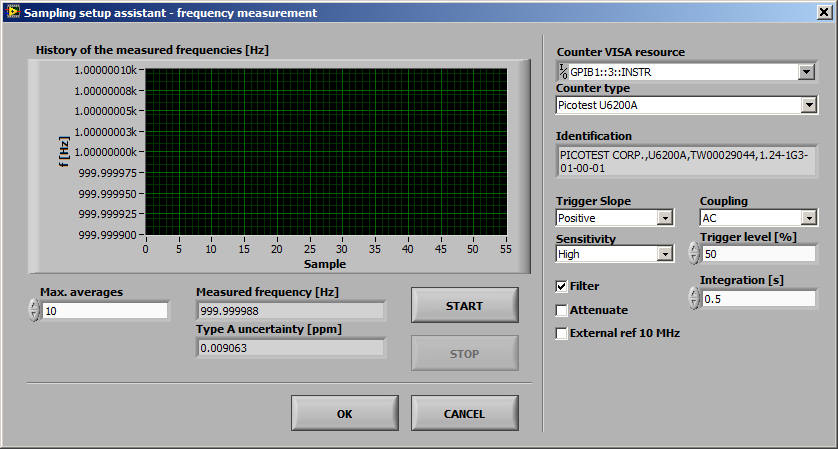


Figure A‑15: TWM sampling setup assistant - fundamental frequency measurement panel.

##### Configuring data processing

Panel for configuration of the processing is accessible from the New measurement configuration panel shown in Figure A‑13 or from batch processing panel Figure A‑23 (see section A.1.4.g). There are two main modes of processing which are selected by the page selector. The RAW command is a simple debugging mode which enables user to simply type in or load m-code to execute on the raw measured sample data (see panel shown in Figure A‑16).

The other and preferred option is the QWTB mode, which is shown in Figure A‑17. When the panel is opened, TWM will fetch the available List of algorithms from QWTB distribution linked in the panel GNU Octave/Matlab configuration panel in Figure A‑4. Note this operation may take some time during which the controls will be grayed. The algorithms may be reloaded manually by pressing the RELOAD ALGs button. Note the actual displayed list is not full list of QWTB algorithms. It is intentionally limited by the filtering file (see [12] for details):

.\ocprog\qwtb\_list.info

This decision was made to remove algorithms which are not compatible with TWM. When no error occurred and correct version of QWTB was selected, the list should contain some 10+ algorithms. By selecting one, TWM will fetch the algorithm info, which may also take some time. The description should appear in the indicators on the right side. The LED indicators will indicate features of the algorithm. The table Parameters of the algorithm will be filled by the user parameters of the algorithm. User may enter the numeric or text into the table following the algorithm manual [8]. The table can be cleared by the button CLEAR PARAMETERS. Note each row name may contain a suffix in the parentheses. Opt. means the parameter is optional. Alt. means it has alternative parameters (details can be found in QWTB guide [9]). Description of the selected parameter (row) is shown in the Variable description indicator.

Bottom part of the panel contains several specialized options. First, user can segment the recorded waveform by selecting Start offset of the first sample and by limiting the amount of input data to process by Input data length. Another option is Calculate result just once…, which means to send multiple records to the algorithm at once. This is currently supported only by the TWM-THDWFFT algorithm. Last option is Calculate uncertainty just once, which means the uncertainty is calculated only for the first repetition cycle (record) and it id disabled in following cycles to save processing time.

One more special processing option Time-stamp handling was added to the processing panel. This option will alter the way TWM processing module handles the time-stamps coming from the digitizers. Many algorithms that involves phase angle perform phase correction based on the time-stamp *ts* and frequency component *f* following formula:

.

As long as sampling is coherent, this correction will ensure the measured phase remains constant independent of a trigger point of a digitizer, which may be valuable for averaging multiple record results. It works fine for algorithms such as FFT, where frequency is given exactly by the DFT bin selection. But for fitting algorithms the frequency *f* has its own uncertainty and for large values of *ts* the correction will produce large uncertainty of a phase angle. Therefore, TWM processing module was extended by option to Disable the time-stamp (algorithm will receive only the channel time-shift correction in this case) or to the Relative to 1.record, which means the time-stamps of particular records are subtracted from a first record. This will still allow the phase correction within the repeated records, but minimizes the absolute value of the correction and thus the additional phase uncertainty.

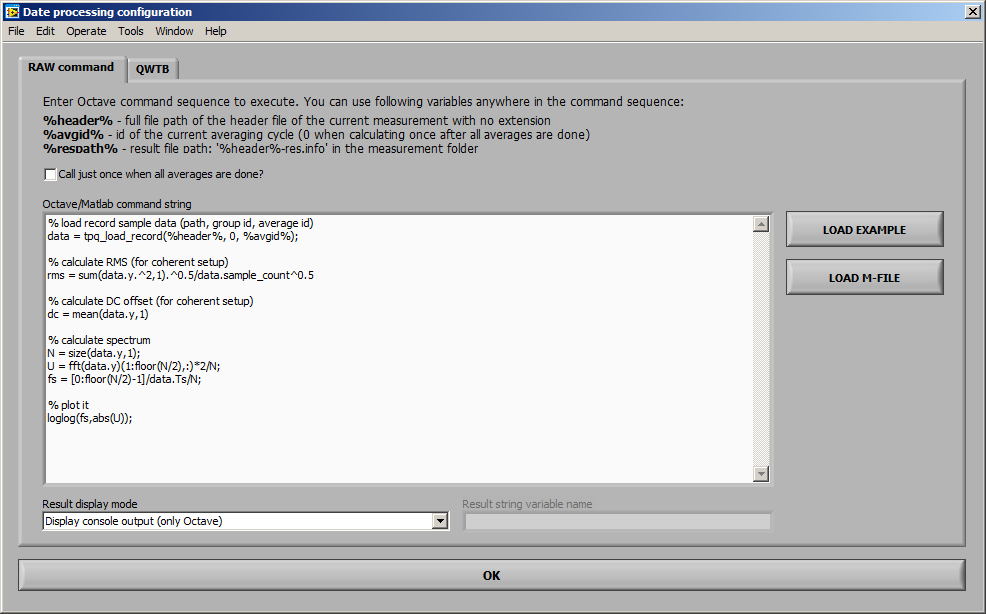


Figure A‑16: TWM processing configuration panel (raw Matlab commands mode).

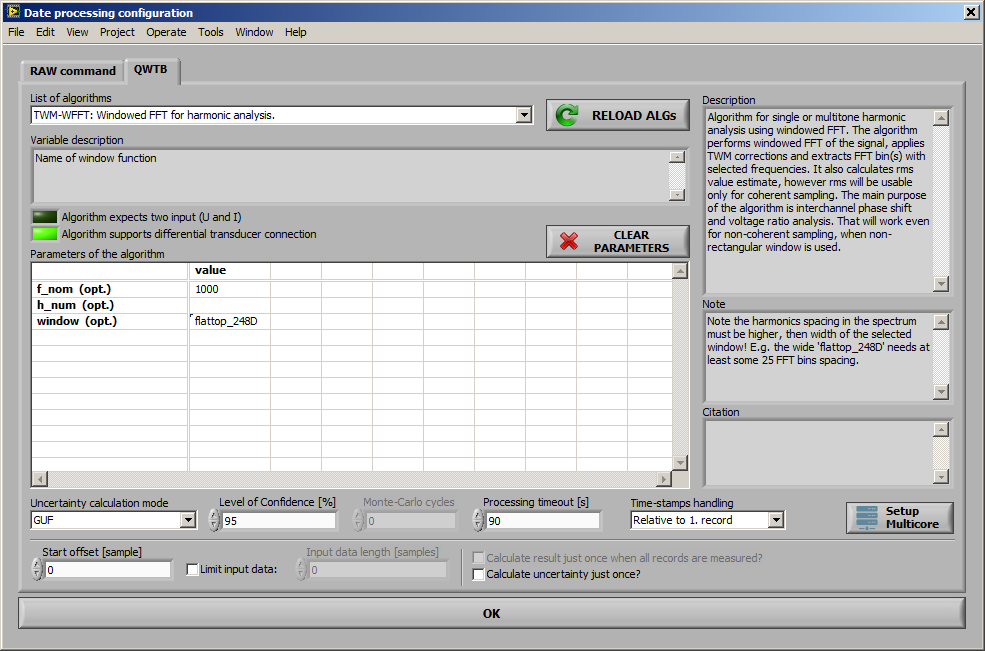


Figure A‑17: TWM processing configuration panel.

Next, user should select Uncertainty calculation mode. Depending on the algorithm, there may be None (no uncertainty calculation), GUF (GF method or fast estimator) or Monte Carlo (numeric Monte Carlo calculation). User should also enter desired Level of Confidence for the calculation. For Monte Carlo there is also entry Monte Carlo cycles which should be always set to at least 1000 iterations (refer to algorithms manual [8] for optimal setup). Control Processing timeout [s] is maximum allowed calculation timeout. Note this will work only for GNU Octave as Matlab interface has no abort capability from LabVIEW. Also note if the timeout is reached, the TWM will throw an error, but the processing still runs in the GNU Octave, so Octave must be eventually restarted using the Octave/Matlab configuration panel.

The Monte Carlo calculation has one more advanced option – the Setup Multicore. By pressing the button, the panel in Figure A‑18 will be shown. The default Execution mode is Singlecore, which means all iterations of the Monte Carlo will be performed on a single core. Option Multicore uses “parfor” command for Matlab or “pracellfun” command from “optim” package [11] for GNU Octave. Note the “parcellfun” will not work in Windows in current version of GNU Octave. Last option is Multistation, which will work only for GNU Octave and it uses package “multicore” [10] which must be installed and loaded. This package is using shared Multistation jobs folder to distribute so called job files, which are then processed by any number of GNU Octave servers that can be started by pressing Start Servers. If the operation was successful, the panel shown in Figure A‑19 will appear. Note all the servers print to the same console so the content will be most likely unreadable. The only purpose is to see the processing is happening and to enable easy termination. The servers can be terminated by simply closing their console window. This option has significant effect on the performance of Monte Carlo calculation, however it should be used only after the Singlecore was tested and working properly.

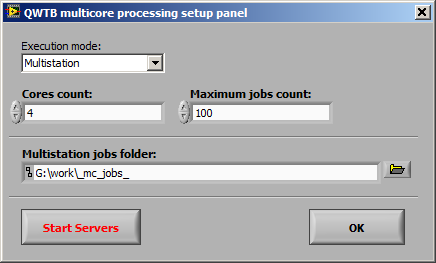


Figure A‑18: TWM multicore processing setup panel.

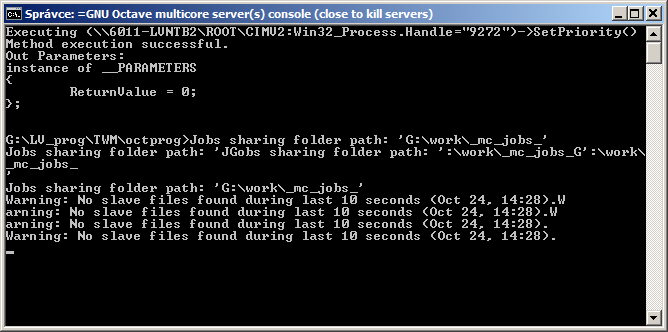


Figure A‑19: TWM multicore processing GNU Octave servers.

#### Initiating measurement

When everything is configured, the measurement can be initiated by pressing the START or SETUP&START buttons on the main panel. Note pressing the START will still ask for new setup if the HW configuration was changed. The ongoing measurement can be terminated by pressing the STOP button at any time.

#### Viewing the results

User may inspect the digitized waveforms using the button VIEW RECORD at any time. The panel shown in Figure A‑20 will be displayed. This panel rereads the data from saved record on runtime, so it may be a little slow for the long records. That is why it is equipped by entry Max samples to display to limit the viewed data amount. The panel can be displayed permanently along with the TWM.

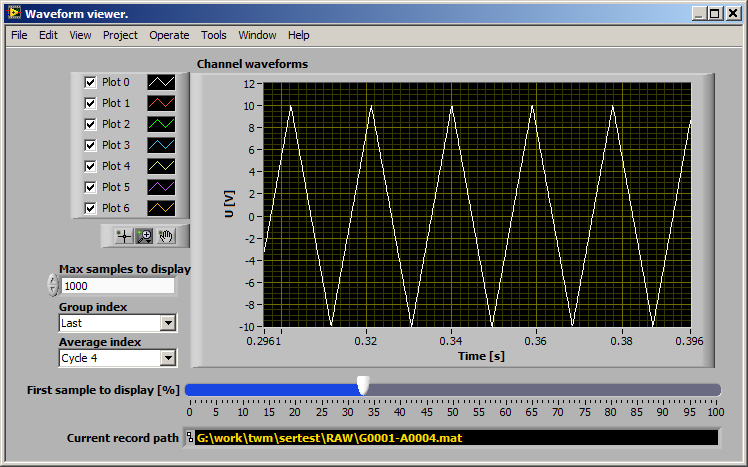


Figure A‑20: TWM record viewer panel.

If the algorithm contains the spectra of defined in the “qwtb\_list.info”, they can be displayed in dedicated panel shown in Figure A‑21. The panel can be shown by pressing button VIEW FFT on main panel Figure A‑3. The panel shows the spectra calculated by selected algorithm. So it will not display spectrum if processing is disabled or selected algorithms does not return it as one of the output quantities.

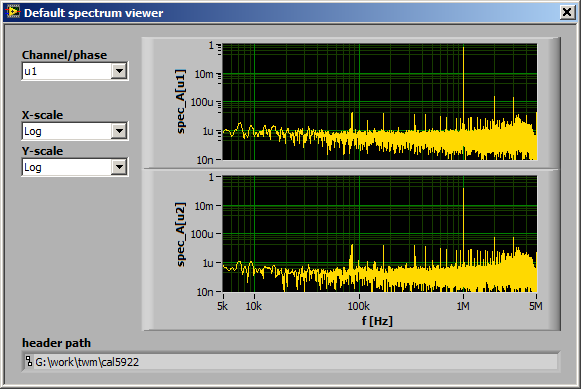


Figure A‑21: TWM spectrum viewer panel.

The results of the processing are displayed in the main panel of TWM. Depending on the processing mode, TWM will display either the RAW panel or QWTB results panel. The RAW panel is just a plain text. The QWTB is shown in Figure A‑3. First, it is important to mention the result viewer operates independently to the measurement. It can be used to view the results while TWM is digitizing and calculating new results. However, all the results formatting and querying is performed in GNU Octave/Matlab, so when the data are being processed, the display of results will be halted.

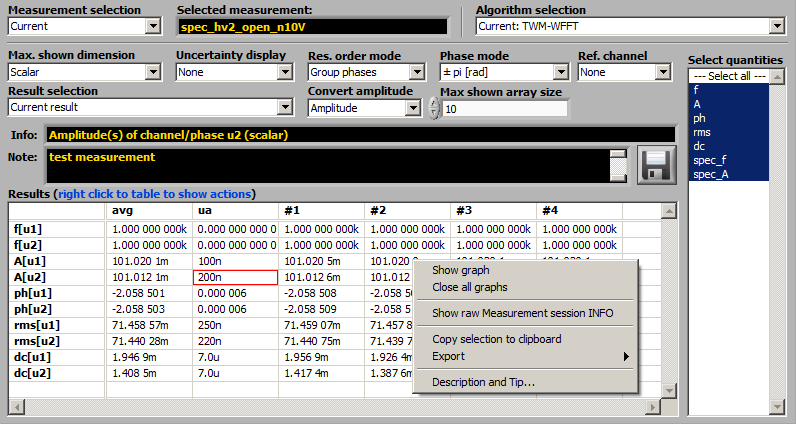


Figure A‑22: TWM processing results viewer.

First control to use is Measurement selection. To speed up the TWM operation, user may choose None to stop result viewer. Next option is Current, which will show currently measured results. Last option is Load from file to select previously processed results. Next control is Algorithm selection, which will contain names of all algorithms that were applied to the particular measurement. Next, user must select Max. shown dimension entry to define how the results should be displayed. Option Scalar means TWM will show only scalar quantities and it will show the results from all repetition cycles at once. Option Vector or Matrix will show scalars and vectors (and matrices), so it can show only one result at once. User may select desired result by entry Result selection. TWM can display Current result, selected one or Average of all available. Note the averaging for large results count may take considerable time. Next, user may Select quantities to display only needed. Next option is Uncertainty display, Res. order mode to regroup the quantities and Phase mode to select phase display format. Max. shown array size limits the size of vector and matrix quantities to be displayed, because TWM was not designed for showing large amounts of data as a table. If the size is exceeded, TWM will display “only graph” instead. Ref. channel is used to display phase differences between channels. This is useful e.g. for measuring inter-channel phase shift by PSFE or similar algorithm. Note the channels must have unique tags, e.g. either different phase indices or u/I transducer type, so TWM can pick the reference channel. Next option is Convert amplitude, which will convert the quantities marked by “is amplitude” in qwtb\_list.info to the desired format, e.g. effective value.

The result quantities and their uncertainties are shown in the table Results. By selecting the row, TWM will display description from the QWTB toolbox. Also note there are several options available by right click to the table content (see Figure A‑22). First, the data can be exported to MS Excel (it must be installed first). Next, any row can be shown as a graph. This is useful for spectra or viewing the history of given quantity. Note the graphs are generated by GNU Octave/Matlab so they remain opened until closed manually or via Close all graphs option but it may take some time especially for GNU Octave for large datasets. Any selection can be copied to a clipboard and paste to Excel style editor. Note the exporting and copying to Excel will always try to convert data to numeric format.

#### Batch processing

TWM can be used as a sampling tool only and the processing can be done later. Also user may need to calculate additional parameters from already digitized signals. TWM is equipped by the batch processing tool for this case. The tool shown in Figure A‑23 can be invoked by pressing START QWTB CALC. User must select the Measurement session. Next, the tool will show available measurement groups and records, which must be selected first. Next, the algorithm must be selected by pressing the PROCESSING SETUP button. After the selection, the processing can be started by pressing START and eventually terminated by pressing STOP. The new measurement results will be shown in the results viewer in the main panel. Note calculating repeatedly the same algorithm will always override past results for the same algorithm. However, the results from the other algorithms will be untouched and available for displaying.

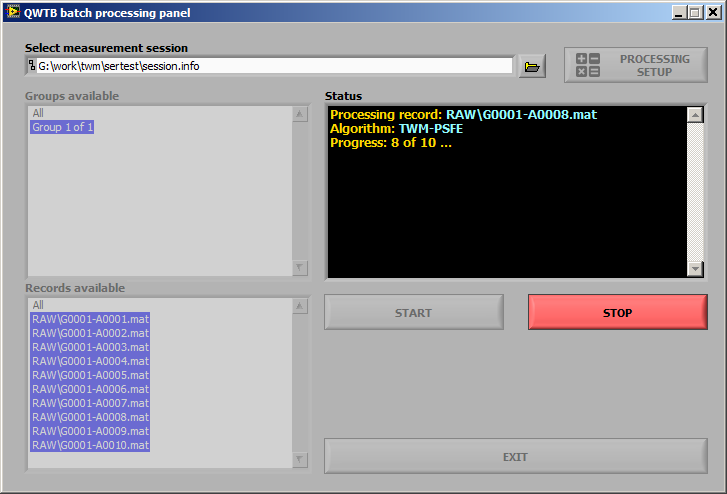


Figure A‑23: TWM QWTB batch processing panel.

### Resources

1. Stanislav Mašláň, et al. *TWM tool*, url:   
   <https://github.com/smaslan/TWM>
2. John W. Eaton, *GNU Octave - Scientific Programming Language*, url:   
   <https://www.gnu.org/software/octave/>
3. MathWorks, *Matlab*, url: <https://www.mathworks.com/products/matlab.html>
4. National Instruments, *LabVIEW Run-Time Engine 2013 (32-bit)*, url:   
   <http://www.ni.com/download/labview-run-time-engine-2013/4059/en/>
5. National Instruments, *LabVIEW and VISA drivers compatibility*, url:   
   <http://www.ni.com/product-documentation/53413/en/>
6. National Instruments. *LabVIEW and niScope drivers compatibility*, url:   
   <http://www.ni.com/product-documentation/53540/en/>
7. *GOLPI library – GNU Octave to LabVIEW Pipe Interface*, url:   
   <https://github.com/KaeroDot/GOLPI>
8. *A244 TWM algorithms description*, Available at “./TWM/doc” or url:   
   <https://github.com/smaslan/TWM/blob/master/doc/A244%20Algorithms%20description.pdf>
9. *QWTB – Q-Wave toolbox*, url:   
   <https://qwtb.github.io/qwtb/>
10. Stanislav Mašláň, Martin Šíra, Markus Buehren, *Octave-Multicore package*, url:  
    <https://gitlab.com/KaeroDot/octave-multicore>
11. Hayato Fujiwara, Jaroslav Hajek, Olaf Till, *GNU Octave package parallel*, url:   
    <https://octave.sourceforge.io/parallel/index.html>
12. Stanislav Mašláň, *A232 TWM file formats and concept*, available at “./TWM/doc”, url:  
    <https://github.com/smaslan/TWM/blob/master/doc/A231%20Data%20exchange%20format%20and%20file%20formats.docx>
13. Stanislav Mašláň, *A231 Corrections Files Reference Manual*, available at “./TWM/doc”, url:   
    <https://github.com/smaslan/TWM/blob/master/doc/A231%20Correction%20Files%20Reference%20Manual.docx>.
14. Stanislav Mašláň, *A245 TWM Structure* available at “./TWM/doc”, url:   
    <https://github.com/smaslan/TWM/blob/master/doc/A245%20TWM%20structure.docx>
15. Stanislav Mašláň, *A244 TWM Algorithms Description* available at “./TWM/doc”, url:  
    <https://github.com/smaslan/TWM/blob/master/doc/A244%20Algorithms%20description.pdf>.
16. Stanislav Mašláň, *A232 Algorithms exchange format* available at “./TWM/doc”, url:  
    <https://github.com/smaslan/TWM/blob/master/doc/A232%20Algorithm%20Exchange%20Format.docx>.

# SOFTWARE CONFIGURATION OF THE BUILT-IN DIGITIZERS

***WP3 - A3.3.2***

***INRIM*** *with support from Metrosert will produce guidance documentation on the software configuration of the built-in digitizers (bandwidth, aperture time, sampling rate etc.), taking into account information from A3.1.2, A3.1.3 and A2.4.3.*

**To be completed…**

# INTEGRATION OF NEW TYPES OF DIGITIZERS

***WP3 - A3.3.3***

***INRIM*** *and CMI will produce guidance documentation on the integration of new types of digitizers (in addition to the NI 5922 digitizer and the sampling DMM 3458A) including integration of calibration datasets of all the components used into the open software tool, based on information from A2.4.3 and A2.4.4.*

*CMI will produce the guidance for the LabVIEW environment and INRIM for the LabWindowsTM/CVI environment.*

## LabVIEW environment

TWM was designed as a modular setup capable of extension by new digitizers without affecting the rest of the SW tool or algorithms. The addition of support for new digitizer requires modification of the LabVIEW part of TWM tool. The programming details are beyond scope of this guide. Therefore, the details are available in a standalone document A2.4.5 TWM Structure [14] which describes internal structure of TWM and steps needed to include new digitizer.

## LabWindowsTM/CVI environment

**To be completed…**

# POWER AND PQ TESTS

***WP3 - A3.3.4***

***TUBITAK*** *and CMI will jointly produce a document briefly describing the power and PQ tests included in the initial database of ~10 algorithms from A2.3.2-A2.3.6 and will produce guidance on the integration of new algorithms into the open software tool using input from A2.4.4.*

## Algorithms

Total of 12 algorithms were developed in scope of TracePQM project. Each of the algorithms was developed, equipped by an uncertainty estimator or Monte Carlo calculator and validated using numeric simulation. The list of algorithms is shown in following table.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Uncertainty** | **Verification** | **Description** |
| TWM-PSFE | GUF | Yes | Single-harmonic estimation (amplitude, frequency and phase) |
| TWM-FPNLSF | GUF | Yes | Single-harmonic estimation (offset, amplitude, frequency and phase) |
| TWM-MFSF | GUF, MCM | Yes | Multi-harmonic estimation (offset, amplitudes, phases, frequency) |
| TWM-WRMS | GUF, MCM | Yes | RMS level calculation in time-domain |
| TWM-WFFT | GUF | Yes | Multi-harmonic estimation (offset, amplitudes, phases) |
| TWM-PWRTDI | GUF, MCM | Yes | Power parameters estimation in time-domain |
| TWM-PWRFFT | GUF | Yes | Power parameters estimation in frequency domain |
| TWM-Flicker | GUF | Yes | Flicker measurement following IEC 61000-4-15 |
| TWM-MODTDPS | GUF | Yes | Amplitude modulation estimator |
| TWM-HCRMS | GUF | Yes | Half-cycle RMS detector following IEC 62586 |
| TWM-InDiSwell | GUF | Yes | Events detector IEC 61000-4-30 |
| TWM-THDWFFT | GUF | Yes | Harmonics and THD estimator |
| TWM-InpZ | None | No | Estimation of digitizer input impedance. |

The algorithms are briefly described in following sections. Up to date details of each algorithm can be found online [15].

### TWM-PSFE - Phase Sensitive Frequency Estimator

TWM-PSFE is a TWM wrapper for the Phase Sensitive Frequency Estimator algorithm (PSFE). PSFE is an algorithm for estimating the frequency, amplitude, and phase of the fundamental component in harmonically distorted waveforms. The algorithm minimizes the phase difference between the sine model and the sampled waveform by effectively minimizing the influence of the harmonic components. It uses a three-parameter sine-fitting algorithm for all phase calculations. The resulting estimates show up to two orders of magnitude smaller sensitivity to harmonic distortions than the results of the four-parameter sine fitting algorithm.

The TWM wrapper TWM-PSFE is designed for single-ended transducers only. It will only estimate frequency in the differential input transducer mode. The algorithm is equipped by a fast uncertainty estimator for the frequency quantity only.

### TWM-MODTDPS - Modulation analyzer in Time Domain, by quadrature Phase Shifting

TWM-MODTDPS is algorithm for calculation of the amplitude modulation parameters of non-coherently sampled signal in time domain. It was designed for basic estimation of the modulation parameters of a sinusoidal carrier modulated by sine wave or rectangular wave with duty cycle 50 %. It is intended as an alternative to the Flicker algorithm if it is required to measure only modulating parameters.

The algorithm operates in time domain and it is based on the analytical signal. It is capable to estimate the parameters up to modulating-to-carrier frequency ratio of 33 %. The record must contain at least 3 periods of the modulating signal and it also requires at least 10 samples per period of carrier.

It is capable to use the differential transducer connection, however the uncertainty is not calculated for the differential mode. The algorithm is equipped by an uncertainty estimator, which covers most of the operating range. The estimator parameter space contains a few gaps where the algorithm may fail, which will be always indicated as an error message. These gaps problems may be prevented by changing the sampling parameters, e.g. by changing the samples count or a sampling rate.

### TWM-FPNLSF - Four Parameter Non Linear Sine Fit

This algorithm fits a sine wave to the recorded data by means of non-linear least squares fitting method using 4 parameter (frequency, amplitude, phase and offset) model. Due to non-linear characteristic, convergence is not always achieved. When run in Matlab, function “lsqnonlin” in Optimization toolbox is used. When run in GNU Octave, function “leasqr” in GNU Octave Forge package optim is used. Therefore results can differ.

This algorithm, in general, is not suitable for distorted signals. It offers good results for signals with low harmonic content if at least 10 periods of signal are recorded with preferably at least 50 samples per period. The algorithm also requires initial estimate of the frequency accurate to 500 ppm.

The algorithm supports differential transducer connection. The integrated uncertainty estimator was developed only for the GNU Octave version. This should be still kept in mind when using the algorithm with Matlab despite the Matlab version seems to give always more accurate results than GNU Octave.

### TWM-HCRMS - Half Cycle RMS algorithm

Algorithm for calculation of the so called half cycle RMS values or sliding window RMS values of a single phase waveform. It calculates RMS value of signal in length of one period with window step defined by the method of calculation. That is, according to the IEC 61000-3-40: (i) Class A - half-cycle step; (ii) Class S – “sliding window” step (20 windows per period for this implementation).

The algorithm is designed so it can handle non-coherent sampling and also it is capable to compensate slow frequency drifts. It uses PSFE and resampling technique to ensure coherent sampling internally. The user can enter signal frequency manually if coherent sampling was ensured by the digitizer.

In general, the algorithm will work better with higher sampling rates. At least 100 samples should be recorded per period of the fundamental component (= sampling rate 5 kSa/s for 50 Hz networks). The higher is better, because the RMS algorithm will better suppress the harmonic and inter-harmonic content.

The algorithm is for single-ended input only and it is equipped with fast uncertainty estimator.

### TWM-InDiSwell - Interruption, Dip, Swell event detector

This algorithm detects power quality events “dip”, “swell” and “interruption” for a single phase systems according to the IEC 61000-3-40, “class A” (half-cycle step) or “class S” (sliding window). It returns relative event time, duration and its residual RMS value in percents relative to the entered nominal level. Note the result provided for the classes A and S should be identical as long as the event is synchronised with the nominal frequency zero-cross. However that is rarely the case of real life situations, so the selection must be made depending on the prescription for the given PQ meter test or PQ event calibrator.

The algorithm internally uses RMS envelope detector TWM-HCRMS, so the accuracy of the detection depends on its properties. In general, the algorithm will work better with higher sampling rates. At least 100 samples should be recorded per period of the fundamental component (= sampling rate 5 kSa/s for 50 Hz networks). The higher is better, because the RMS algorithm will better suppress the harmonic and inter-harmonic content.

The algorithm is for single-ended input only and it is equipped with fast uncertainty estimator.

### TWM-THDWFFT - THD from Windowed FFT

This algorithm is designed for calculation of the harmonics and Total Harmonic Distortion (THD) of the non-coherently sampled signal. It uses windowed FFT to detect the harmonic amplitudes, which limits the achievable accuracy of the harmonics detection due to the window scalloping effect. However, the algorithm was initially designed for THD calculation of the low-distortion signals, where the accuracy was not critical. The relative expanded uncertainty of the harmonics is at least 0.015 % (or 0.005 % after highly experimental correction method). On the other hand, the algorithm was designed to compensate the spectral leakage of the noise to the harmonics near noise level, so it offers decent accuracy for the very low distortions near self-THD of the digitizer itself.

The algorithm supports direct processing of a multiple records which are used to produce averaged spectrum before the main calculation. This possibility should be preferred instead of repeated call of the algorithm for each record as it reduces the noise. The algorithm supports only single-ended transducer connection.

The algorithm returns: (i) Full spectrum; (ii) Identified harmonics; (iii) THD coefficients according various definitions; (iv) RMS noise estimate; (v) THD+Noise estimate.

Note the uncertainty is evaluated only for some of the returned parameters.

### TWM-PWRTDI - Power by Time Domain Integration

TWM-PWRTDI is an algorithm for calculation of power parameters using a time domain integration of *u*(*t*)\**i*(*t*) product. It is based on the use of window function to eliminate effects of non-coherent sampling. Therefore, it does work even for non-coherently sampled waveforms. The algorithm itself without correction effect can easily reach errors below 1 µW/VA with proper selection of a sampling rate and windows size.

The algorithm can calculate all basic parameters: active power *P*, reactive power *Q*, apparent power *S*, RMS voltage *U*, RMS current *I* and power factor *PF*. It also returns DC components separately: *U*dc, *I*dc and *P*dc. User may choose optional AC coupling mode by setting parameter *ac\_coupling* = 1 in which case the *U*, *I*, *P*, *Q*, *S* and *PF* will be calculated without the AC component.

Note the windowed RMS method itself can calculate power in any quadrant, however it is not able to distinguish all four quadrants. The quadrant identification (proper signs for *P* and *Q*) is obtained from a complementary windowed FFT algorithm which is running along the main RMS calculation. Note the quadrant selection may fail around *PF* = *0* (the absolute values will be correct). The sign of *Q* is calculated using harmonic components method according Budenau definition.

The TWM-PWRTDI algorithm is able to use single-ended or differential input sensors for voltage channel, current channel or both. The algorithm is also equipped by a fast uncertainty estimator and the Monte Carlo uncertainty calculation method for more accurate but slower uncertainty evaluation.

### TWM-WRMS - RMS value by Windowed Time Domain Integration

TWM-WRMS is an algorithm for calculation RMS value and DC component of signal a time domain integration of windowed signal *y*(*t*). The windowing function eliminates effects of non-coherent sampling. Therefore, it does work even for non-coherently sampled waveforms. The algorithm itself without contribution of corrections can easily reach errors below 1 µV/V with proper selection of a sampling rate and windows size.

The TWM-WRMS algorithm wrapper is able to use single-ended or differential input sensors. The algorithm is also equipped by a fast uncertainty estimator and the Monte Carlo uncertainty calculation method for more accurate but slower uncertainty evaluation.

### TWM-WFFT - Windowed FFT spectrum analysis

Algorithm for single or multi-tone harmonic analysis using windowed FFT. The algorithm performs windowed FFT of the signal, applies TWM corrections and extracts FFT bin(s) with selected frequencies. It also calculates *rms* value estimate, however *rms* will be usable only for coherent sampling. The main purpose of the algorithm is inter-channel phase shift and voltage ratio analysis. That will work even for non-coherent sampling, when non-rectangular window is used.

Note the harmonics spacing in the spectrum must be higher, then width of the selected window! E.g. the wide “flattop\_248D” needs at least some 25 FFT bins spacing. Also note the wider windows have higher equivalent noise bandwidth, so the noise in the analysed harmonic is amplified.

The TWM-WFFT algorithm wrapper is able to use single-ended or differential input sensors. The algorithm is also equipped with a fast uncertainty estimator for the harmonic components.

### TWM-Flicker - Flicker algorithm

The TWM wrapper TWM-Flicker is an algorithm for evaluation of the short term flicker parameters. It calculates instantaneous flicker sensation *P*inst and short-term flicker severity *P*st. Sampling rate has to be higher than 7 kHz. If sampling rate is higher than 23 kHz, signal will be down sampled by algorithm. More than 600 s of signal is required as the algorithm needs at least a minute to settle the filters. Typical sampling time value is above 660 s.

The algorithm requires either Signal Processing Toolbox when run in MATLAB or a signal package when run in GNU Octave. Frequency of line (carrier frequency) *f\_line* can be only 50 or 60 Hz.

The algorithm was implemented according IEC 61000-4-15.

The algorithm wrapper is equipped by a simple uncertainty estimator based on the worst observed error of the algorithm on the tabulated *P*st values for various sampling rates.

Note the algorithm output slightly differ for Matlab and GNU Octave implementation. The cause of this difference was not yet identified. Also the observed performance in the Matlab 2017b was about five times higher then in GNU Octave 4.2.2 on the same computer.

### TWM-MFSF - Multi-Frequency Sine Fit

TWM-MFSF is an algorithm for estimating the frequency, amplitude, and phase of the fundamental and harmonic components in a waveform. Amplitudes and phases of harmonic components are adjusted to find minimal sum of squared differences between sampled signal and multi-harmonic model. When all sampled signal harmonics are included in the model, the algorithm is efficient and produces no bias. It can even handle aliased harmonics, if they are not aliased back exactly at frequencies where other harmonics are already present. Further, it can also handle non harmonic components, when their frequency ratio to the fundamental frequency is exactly known a-priori.

The TWM wrapper TWM-MFSF is equipped with a Monte Carlo uncertainty calculator and also a fast uncertainty estimator limited for certain types of signal and algorithm setup.

### TWM-PWRFFT - Power by FFT

Algorithm for calculation of power parameters from FFT spectra of voltage and current channels. It calculates the power in full bandwidth. It designed for coherent sampling.

The algorithm can calculate all basic parameters: active power *P*, reactive power *Q*, apparent power *S*, RMS voltage *U*, RMS current *I* and power factor *PF*. It also returns DC components separately: *U*dc, *I*dc and *P*dc. User may choose optional AC coupling mode by setting parameter *ac\_coupling* = 1 in which case the *U*, *I*, *P*, *Q*, *S* and *PF* will be calculated without the AC component. The definition of components is identical to the TWM-PWRTDI.

The TWM-PWRFFT algorithm wrapper is able to use single-ended or differential input sensors for voltage channel, current channel or both. The algorithm is also equipped by a fast uncertainty estimator.

### TWM-InpZ – Estimator of Digitizer Input Impedance

This simple algorithm was designed to measure input impedance of digitizer channel. It is experimental algorithm designed as an addition to the main algorithms developed for PQ measurement. It is not equipped by any uncertainty evaluation.

## Integration of new algorithms

TWM and TPQA tools are designed so they can both use TWM processing module that runs in Matlab or GNU Octave. Thus following section applies for both tools.

The processing module described in [14] is and extension for QWTB toolbox [9]. The toolbox was designed for simple addition of new algorithms by simply creating new folder with specified files. However, to make TWM work properly, the rules described in [16] for the naming of the input and output quantities in the newly developed algorithm must be followed, so all the data received from TWM processing module are recognized and processed correctly.

Each QWTB algorithm compatible with TWM processing module must contain at least following files in following file structure:

* **.\octprog\QWTB** QWTB toolbox root folder in the TWM ‘octprog’ folder.
  + **alg\_TWM**-my\_algorithm\_name Algorithm ‘**TWM-**my\_algorithm\_name’ root folder .
    - **alg\_info.m** Algorithm info definition file (see QWTB doc [9]).
    - **alg\_wrapper.m** Algorithm wrapper (call to low level functions).
    - \*.\* Auxiliary function of the wrapper.

The **red** parts are mandatory names. The **bold** prefix in the algorithm name ‘**TWM-**’ was chosen to distinguish the original QWTB algorithms that usually do not apply any corrections from the TWM algorithm extensions that do calculate corrections and their uncertainties. So it is recommended to follow this rule.

TWM always expect at least ‘alg\_info.m’ file, which contains name, short description, references and definitions of input and output quantities. TWM then only requires main algorithm wrapper ‘alg\_wrapper.m’, which may either call some low level functions, call another QWTB algorithms or contain the algorithm code directly. It must process all the received quantities with the sample data and corrections and return quantities to be saved and displayed by TWM. It also contains estimator or Monte Carlo calculator of uncertainty.

In order to make the newly added algorithm visible in TWM, one must first add its name to the filter of supported algorithms which is located in the file:

.\octprog\qwtb\_list.info

This file is described to details in [12]. It must at least contain name of the algorithm in the list of supported algorithms:

// 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

…

TWM-my\_algorithm\_name

…

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

…

The newly integrated algorithm should automatically appear in the list of available algorithms in the QWTB processing panel of TWM show in Figure A‑17.

When implementing the algorithms, user should keep in mind the algorithm should be compatible with both GNU Octave and Matlab environments, so it is recommended to not use the extended Octave syntax and when possible also not the commercial packages where no equivalent for the other environment is available. Where needed, use may use if/else statement to distinguish Matlab and Octave code to reflect the differences or eventually throw and error with information the algorithm cannot run in given environment. Example of environment dependent code:

%% Check if running in Matlab or GNU Octave:

IS\_OCTAVE = (**exist**('OCTAVE\_VERSION') ~= 0);

%% Check available packages

**if** IS\_OCTAVE

**if** not(exist('butter'))

pkg load signal

**if** not(exist('butter'))

**error**('This algorithm requires package `signal` from Octave Forge');

**end**

**end**

**else**

**if** not(exist('butter'))

**error**('This algorithm requires `Signal Processing Toolbox`');

**end**

**end**

More details on implementation of the algorithm are given in up to date version of document “A232 Algorithms Exchange Format.docx” available online [16].

# TITLE 1

Title 1 title 1 title 1 title 1 title 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Table E‑1 : Six\*Three exemple

## Title 2

Title 2 title 2 title 2 title 2 title 2 title 2 title 2 title 2 title 2 title 2 title 2 title 2 title 2 title 2 title 2 title 2 title 2 title 2 title 2 title 2 title 2 title 2 title 2 title 2 title 2 title 2 title 2 title 2 title 2 title 2 title 2 title 2.



Figure E‑1 : Blue Exemple

### Title 3

Title 3 title 3 title 3 title 3 title 3 title 3 title 3 title 3 title 3 title 3 title 3 title 3 title 3 title 3 title 3 title 3 title 3.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

Table E‑2 : Three\*Two example

#### Title 4

Title 4 title 4 title 4 title 4 title 4 title 4 title 4 title 4 title 4 title 4 title 4 title 4 title 4 title 4 title 4 title 4 title 4 title 4.

##### Title 5

Title 5 title 5 title 5 title 5 title 5 title 5 title 5 title 5 title 5 title 5 title 5 title 5 title 5 title 5 title 5 title 5 title 5 title 5 title 5 title 5 title 5 title 5 title 5 title 5 title 5 title 5 title 5.



Figure E‑2 : Red Exemple

###### Title 6

Title 6 title 6 title 6 title 6 title 6 title6 title 6 title 6 title6 title 6 title 6 title 6 title6 title 6 title 6 title6 title 6 title 6 title 6 title 6 title 6 title 6 title 6 title 6 title 6 title 6.