nvh group

Tamara

file specification

NVH Group AB

Stora Badhusgatan 18-20
411 21 Gothenburg
Sweden

Registered for corporation taxation by The Swedish Tax Agency
VAT. no: SE 559155918101 Company registration no: 559155-9181

www.nvh.group \\\ info@nvh.group \\\ +46 31-300 01 01

nvh group

Tamara file specification 1.3.9

In this document the variables used in a Tamara measurement file are described. Tamara files created prior to the initial release of the file specification are mostly fully compatible, but not guaranteed to comply with this specification.

See the *Changelog* chapter[i] for a overview of changes to the Tamara file specification.

i Page 30

Table of contents

Definition of variables used in this document

activemeasurementtype

The active measurement type without spaces for use in structs, e.g. *levelrecorder*, *slm* and *mechanicalmobility*.

B

As in 1/B fractional octave band, used here mostly as a character for fieldnames, e.g. *f.oct24.display*, where B = 24.

cansignalcount

Index indicating the k:th CAN signal. See *numcansignals*.

char cell

A cell matrix containing only *char* data. E.g. {'line 1 of text', 'line 2 of text'}.

gpssignalcount

Index indicating the k:th GNSS/GPS signal. See *numgpssignals*.

N

Number of processed blocks.

N_2

Number of full blocks processed, DFT overlap = 0 % → N = N2.

$N_2 = N \times (100 - \textit{DFT overlap in \%}) / 100$

numbands

Number of proportional frequency bands, as in number of 1/3-octave bands.

numcansignals

Number of activated CAN signals[ii].

numchs

Number of enabled input channels[iii] in a measurement.

numdftlines

Number of DFT lines. For single-sided variables it is related to Δf[iv], fs[v] and anti-aliasing factor. The anti-aliasing factor is found in *tamaraConfig.globals.ULfactor*, and defaults to 2.56 for Compactdaq and PXI devices, 2.4 for audio devices sampled at 48 kHz, and 2.205 for audio devices sampled at 44.1 kHz—unless *tamaraConfig.globals.ULfactor-Override* set to *true* (in which case the factor is completely determined by user).

ii E.g. TCMPropFr04\TrsmVehSpd

iii Enabled but hidden channels are excluded.

iv Frequency resolution, inversely proportional to sampling period *T*.

v Sampling rate, typically in Hz.

numedges
: Number of edge pairs/bins in a histogram.

numgpssignals
: Number of GNSS/GPS signals/variables, depending on device, but typically 11[vi].

numgpstimesteps
: Number of time steps for GPS/GNSS signal, time resolution vary with device—NVHG-GPS-1 and NVHG-GPS-2 have an update step of 60 ms.

numimpulses
: Number of triggered impulse blocks.

numinputchs
: Number of present input channels.

numlines
: Number of lines, e.g. in a *char cell*.

numlrcustombands
: Number of custom-bands[vii] used in Level Recorder.

numlrbands
: Number of 1/B octave bands used in Level Recorder.

numlrweights
: Number of weighting functions used in Level Recorder.

numoutputchs
: Number of present output channels.

numtachos
: Number of enabled tachometer signals.

numtimestepsfast
: Number blocks for fast-weighted time in SLM.

numtimestepsslow
: Number blocks for slow-weighted time in SLM.

scalar
: A matrix of size 1 x 1.

vector
: A matrix of size 1 x K, or K x 1.

vi When using NVHG-GPS-1 or NVHG-GPS-2.

vii *config.globals.LR.customFilterBank.f* when *config.globals.LR.useOctFilters = -1.*

Changelog

Data variables

Version 1.3.9

- *Info.ISO9614.dB_reference* added.

Version 1.3.7

- *LR.referenceUnitOffset* added.

Version 1.3.3

- *Info.note* added.

Version 1.3.2

- *LR.levels.scalars*, *LR.levels.bp(bandcount).scalars* and *LR.levels.oct(bandcount).scalars* added.
- *LR.levels.max.[wt]*, *LR.levels.min.[wt]* and *LR.levels.avg.[wt]* changed from *single* to *double*.

Version 1.3.1

- *Info.tacho* added.
- Datacontainer *tacho_<tachocount>* added.

Version 1.3.0

- *Info.frfReference*, *Info.frfEstimator* and *Info.frfMode* added.

Version 1.2.9

- *Info.ISO9614* added.
- *f.weights.[wt].narrow*, *f.weights.[wt].oct* and *f.weights.[wt].oct[B]* are now column vectors instead of row vectors.

Version 1.2.7

- *CAN.messages* removed.
- Added option to store all CAN frames in *CAN.frames*.

Version 1.2.6

- Added *LR.levels.max*, *LR.levels.min*, *LR.levels.avg* and *LR.levels.x_block*.
- Changed *Info.window*.

- Added *Info.index*.
- Added *Info.frequencyRange*.
- Added *Info.structs*.
- Info.timesignal changed to *Info.timeSignal*.
- Added *spectrogram.narrow*.
- *spectrogram.time* moved to *spectrogram.(<oct>).t*.
- *spectrogram.f* moved to *spectrogram.(<oct>).f.*
- spectrogram.oct[B].lin removed, only .dB will be available.
- Spectrograms are now stored every block N, instead of every full block, N_2.
- *SQ.AI.x* removed.
- Spectrograms are now always calculated from the instantaneous spectra, instead of using a moving average when enabled for the auto-spectra—*config.globals.expAverageTime*.

Version 1.2.5

- Clarified and changed *SLM.LSmax* and *SLM.LFmax*.

Version 1.2.2

- *Gxx_max* added.

Version 1.2.0

- *Info.LR* added if measurement type is *Level recorder*, a copy of *config.globals.LR*.
- *Info.duration* added.
- *Info.overload* structure added.
- *Info.df* added.
- *Info.window* added.
- *Info.SQ* structure added if measurement type is *Sound level meter*.

Prior to version 1.0.0[viii]

- *SLM.f* and *ISO9614.f* have been removed from stored data, since it was a subset of variable *f* and redundant respectively.
- *SLM.time* have been renamed to *SLM.duration*.
- *SLM.LEQ.percentile* have been moved to *SLM.Lavg.percentile*.
- SLM.LEQ.histogram have been moved to *SLM.Lavg.histogram*.
- Datatype of *LR.histogram.[wt].N*, *Info.N*, *Info.numberOfChannels* and *SLM.Lavg.histogram.[wt].N* changed from *double* to *uint32*.

viii Full list of changes prior to initial release 1.0.0 is not shown here, only changes made for the initial release.

- Datatype of data in *SLM.Lavg* changed from *double* to *single*.
- Datatype of *LR.percentile.array.P* and *LR.histogram.[wt].P* changed from *double* to *single*.
- Datatype for GPS signal data changed from mix of *double* and *single* to *single*.
- *LR.levels.oct.N* removed from stored data: redundant information.
- *LR.levels.bp.f.display* and *LR.levels.bp.f.exact* removed and replaced with *LR.levels.bp.f.center*.
- *LR.levels.bp.y* and *LR.levels.oct.y* changed from *double* to *single*.
- *Info.plus* removed: redundant information.

Tamara settings

Version 1.3.9

- Added *config.internal.filenaming.automatic.counter.width*.

Version 1.3.8

- Added *config.globals.input.lowpass*.

Version 1.3.7

- Added *config.globals.useReferenceUnitsForFRF*.

Version 1.3.6

- Renamed *config.chs(chcount).calibrationflag* to *config.chs(chcount).isCalibrated*.
- *config.chs(chcount).calibration* is now always available—but empty if no calibration has been performed and accepted.

Version 1.3.5

- Removed *config.chs(chcount).terminalConfigVal*, *config.chs(chcount).couplingvalue*, *config.chs(chcount).availableExcititationCurrent*, *config.chs_out(chcount).typeVal* and *config.chs_out(chcount).terminalConfigVal*: duplicated information.

Version 1.3.4

- *config.chs(chcount).secondaryReference* added.

Version 1.3.3

- *config.globals.note* added.
- *config.comment* removed. Now only available in *Info.comment*—previously this information was duplicated.

Version 1.3.2

- *config.globals.SLM.histogramResolutionPval* removed.
- *config.globals.SLM.useOctFilters* added.
- *config.internal.storing.plots.labelAngle* added.

Version 1.3.1

- *config.internal.storing.plots.backgroundColour* and *config.internal.storing.plots.overrideBackgroundColour* added.

Version 1.2.9

- *config.chs_out(chcount).dutycycle* and *config.chs_out(chcount).frequency* have been changed with regards to sweep types.
- *config.globals.LR.bandpassOrder* is removed.
- *config.globals.calibration.DFToverlap* is removed, instead 75% will always be used (the previous default value).
- *config.globals.acq.warmupTime* added.
- *config.globals.acq.store.automatic* is now documented.
- *config.internal.storing.plots.ratio* added.
- New default for *config.globals.LR.loggingRate*: -1, using the new automatic mode to determine a suitable logging rate depending on selected exponential time averaging.
- *config.globals.weighting* is removed.

Version 1.2.8

- *config.internal.filenaming.automatic.storeByDate* added.
- *config.globals.avg_slide* removed and fully replaced with *config.globals.expAverageTime*.
- Added *config.globals.expAverageUnit*.
- *config.globals.avg_slideUnit* removed and replaced with *config.globals.expAverageUnit*.
- *config.globals.ISO9614_1.weighting* removed, was barely used in the software, and only for what default weighting to show in plots.
- *config.globals.ISO9614_1.warning* added.

Version 1.2.7

- *config.internal.storing.plots.useColourbar* added.
- Removed option to store CAN signals as message objects: *config.internal.storing.variablesToSave.(<activemeasurementtype>).canMessages.enable*.
- Added option to store all CAN frames, option available here: *config.internal.storing.variablesToSave.(<activemeasurementtype>).canStoreAllFrames.enable.*

Version 1.2.6

- *config.globals.timeSignal.precision.flac* and *config.globals.timeSignal.scaleByRange removed*. Tamara no longer stores timesignal in any other form than in calibrated units, and filtered if enabled.
- *config.globals.spectrogram* added.
- *config.globals.octaveSpectrogram* changed to *config.globals.spectrogram.oct.enable*.
- *config.globals.timeSignal.mergeUNV* added.
- *config.globals.enable_time_log* renamed to *config.globals.storeTimeSignal*.
- *config.globals.output.filterTypeVal* have been removed, superfluous information.

Version 1.2.4

- *config.globals.LR.paperSpeed* have been renamed to *config.globals.LR.loggingRate*. In order to clarify its function. Default value changed from 20 to 40.
- Default value of *config.globals.useTwoStageStart* changed from 0 to 2 (software two-stage start).
- *config.globals.input.useIntegrationSuperSampling* have been added.
- *config.chs(chcount).trigger* have been removed, fully replaced by settings in *config.trigger*.
- *config.chs(chcount).internal.selfCalibrationSupported* and *config.chs_out(chcount).internal.selfCalibrationSupported* added.
- *config.chs(chcount).chassisNumber* and *config.chs_out(chcount).physicalChannel* added.
- *config.chs(chcount).nativeDataType* and *config.chs_out(chcount).nativeDataType* removed.

Version 1.2.2

- *config.internal.dataBasePath* have been removed—it was never used.
- *config.globals.calcDataStats* have been removed—it was not used in later versions.
- *config.globals.calcGxxMaxHold* have been added.
- *config.globals.input.highpass* have been added.

Version 1.2.1

- *config.globals.LR.integrateAcceleration* have been added.

Version 1.2.0

- *config.internal.virtual* have been added.
- The default value of *config.globals.timeSignal.appendData.mat* have been changed from *false* to *true*.
- *config.internal.deployed.tamaraVarPath* added.

- *config.internal.deployed.tamaraUserBackupPath* added, renamed from previous *config.internal.deployed.tamaraBackupPath*.
- *config.internal.deployed.tamaraBackupPath* changed from location of user backup files to location of Tamara backup files.
- *config.internal.checkSensorHealth* renamed to *config.internal.sensor.checkHealth.*
- *config.internal.sensor* added.
- The default value of *config.chs(chcount).rangevalue* is now the maximum number of available ranges, instead of 1.
- *config.internal.storing.plotsToSave* removed.
- Undocumented setting *config.internal.storing.plotsToSave.global.saveActivePlots.enable* replaced by *config.internal.storing.plots.autoSaveActive*.
- *config.internal.storing.plots.saveIndividualLines* have been added.

Version 1.1.0

- *config.globals.FRFwithoutReference* have been added.
- The default value of *config.globals.skip_nonreference_FRFS* have been changed from *true* to *false*.
- *config.internal.explorer.calc.store.psd* have been added.
- *config.internal.explorer.calc.store.timesignal* have been added.
- *config.internal.explorer.calc.store.includeOtherData* have been added.
- Default value of *config.globals.timeSignal.appendData.mat* changed from *true* to *false*.
- *config.internal.correction.nPoints* have been added.
- *config.internal.processor.useAutomaticPolarity* removed and replaced with new channel specific setting: *config.chs(chcount).invertPolarity*.
- Added module slot number: *config.chs(chcount).slot.*
- *config.internal.processor.exportLegends* have been added.
- Added *config.internal.hidden.disableCAN*. Set to *true* to fully disable the CAN GUI.
- Added *config.globals.SLM.enableTimeErase*.
- *config.chs(chcount).calibration.moduleSerialHex*, *config.chs(chcount).calibration.moduleSerialDec* and *config.chs(chcount).calibration.description* have been removed.

Prior to version 1.0.0

- *config.internal.freqULuser* have been deprecated.
- *config.location* renamed to *config.comment*.
- *config.internal.checkSensorHealth* has been added, default value is *true*.
- *config.internal.processor.exportLegends* has been added, default value is *true*.
- *config.fftOverlap* renamed to *config.DFToverlap*.
- *config.globals.fftOverlapVal* removed.

- *config.ISO9614_1* moved to *config.globals.ISO9614_1*.
- *config.chs.internal.sensitivityString* removed, use *config.chs.sensitivityvalue*[ix] instead.

ix See page 91.

File container for Tamara files

Tamara measurement files can be stored using three different containers/formats:

1) Matlab .mat file, version 6.0[x]

2) Matlab .mat file, version 7.0[xi] (default[xii])

3) Matlab .mat file, version 7.3[xiii] (HDF5 compatible)

Files are always stored without compression. See https://www.mathworks.com/help/pdf_doc/matlab/matfile_format.pdf[xiv] for information about the mat file specification.

See *config.internal.storing.matversion*.

x *config.internal.storing.matversion* = 6

xi *config.internal.storing.matversion* = 7

xii Starting in Tamara version 2022.1.4—previously version 6 was the default.

xiii *config.internal.storing.matversion* = 7.3

xiv If not available contact: support@tamara.app

Data variables

AGxx - averaged auto-spectra

AGxx

type: [struct] size: [scalar]

Struct container for averaged auto-spectra. Always available, unless *tamaraConfig.internal.storing.variablesToSave.(<activemeasurementtype>).AGxx.enable = 0*.

AGxx.label

type: [char cell] size: [numlines x 1]

Description of variable AGxx.

AGxx.narrow

type: [struct] size: [scalar]

Struct container for narrow-band AGxx data.

AGxx.narrow.dB

type: [single] size: [numdftlines x numchs]

Matrix containing averaged auto-spectrum data. Unit is dB re. units, e.g. dB re 1 Pa. Gxx(k) = 10*log10(conj(S(k)) .* S(k)) (in dB). NB: always re. 1.

AGxx.narrow.lin

type: [double] size: [numdftlines x numchs]

Matrix containing averaged auto-spectrum data. Linear units, e.g. Pa or V (NB: not Pa^2 etc). Gxx(k) = sqrt(conj(S(k)) .* S(k)).

AGxx.oct

type: [struct] size: [scalar]

Struct container for octave-band AGxx data. Which octave-band is defined in *tamaraConfig.band*. Not available if *tamaraConfig.band = narrow.*

AGxx.oct.dB

type: [single] size: [numbands x numchs]

Matrix containing averaged auto-spectrum data in octave-bands in dB. Units are same as for *AGxx.narrow.dB*.

AGxx.oct.lin

type: [double] size: [numbands x numchs]

Matrix containing averaged auto-spectrum data in octave-bands. Units are linear and are same as for *AGxx.narrow.lin*.

AGxx.oct[B]

type: [struct] size: [scalar]

Struct container for octave-band AGxx data. *B* is a number corresponding to 1/*B*-bands, e.g. *AGxx.oct3* for 1/3-octave band data. Available if *tamaraConfig.globals.octaveData-Storing* is set to *all*. Available numbers B: 1, 3, 6, 12 and 24; 48 is available if *tamaraConfig.band = '1/48'*.

AGxx.oct[B].dB

type: [single] size: [numbands x numchs]

Matrix containing averaged auto-spectrum data. Units are same as for *AGxx.narrow.dB*. See description for *AGxx.oct[B]*.

AGxx.oct[B].lin

type: [double] size: [numbands x numchs]

Matrix containing averaged auto-spectrum data in octave-band. Units are linear and are same as for *AGxx.narrow.lin*. See description for *AGxx.oct[B]*.

AGxy - averaged cross-spectra

AGxy

type: [struct] size: [scalar]

Struct container for averaged cross-spectra. Available if a reference channel has been defined, or if *tamaraConfig.skip_nonreference_FRFS = 0*; unless *tamaraConfig.internal.storing.variablesToSave.(<activemeasurementtype>).AGxy.enable = 0*.

AGxy.label

type: [char cell] size: [numlines x 1]

Description of variable AGxy.

AGxy.narrow

type: [struct] size: [scalar]

Struct container for narrow-band AGxy data.

AGxy.narrow.dB

type: [single] size: [numdftlines x numchs x numchs]

Matrix containing averaged auto-spectrum data. Unit is dB re. units, e.g. dB re 1 Pa. Gxy(k, l) = 10*log10(conj(S(k)) .* S(l)) (in dB). NB: always re. 1.

AGxy.narrow.lin

type: [double] size: [numdftlines x numchs x numchs]

Matrix containing averaged auto-spectrum data. Linear units, e.g. Pa or V (NB: not Pa^2 etc). Gxy(k, l) = sqrt(conj(S(k)) .* S(l)).

CAN - CAN bus data

CAN

type: [struct] size: [scalar]

Struct container for CAN. Available if CAN messages are enabled and detected.

CAN.frames

type: [struct] size: [scalar]

Struct container for CAN message objects. Available only if *config.internal.storing.variablesToSave.(<activemeasurementtype>).canStoreAllFrames.enable = true*. Undocumented.

CAN.signals

type: [struct] size: [1 x numcansignals]

Struct container for CAN signals.

CAN.signals(cansignalcount).info

type: [struct/double] size: [scalar]

Struct container for additional info for CAN signal. Is *double* and *empty* if no info is present for the *cansignalcount*:th CAN signal.

CAN.signals(cansignalcount).info.speedCorrectionEnabled

type: [logical] size: [scalar]

Available if *cansignalcount* CAN signal's physical quantity is velocity. Signifies whether a speed correction factor have been applied to *CAN.signals(cansignalcount).value*. Speed correction factor is found in *CAN.signals(cansignalcount).info.speedCorrectionFactor*.

CAN.signals(cansignalcount).messageType

type: [char] size: [1 x nchars]

String of the name of the message type of signalcount CAN signal.

CAN.signals(cansignalcount).name

type: [char] size: [1 x numchars]

String of the name of *cansignalcount* CAN signal.

CAN.signals(cansignalcount).relTimestamp

type: [single] size: [1 x nummessages]

Vector of relative time stamps for *cansignalcount* CAN signal data. Relative and absolute time pair matrix is found in *CAN.signals(cansignalcount).timestampPairs*.

CAN.signals(cansignalcount).timestampPairs

type: [single] size: [numcantimeblocks x 2]

Matrix of relative time stamps for *cansignalcount* CAN signal data, second row, and absolute time stamp, first row. Only last time stamps for each block stored.

CAN.signals(cansignalcount).unit

type: [char] size: [1 x numchars]

String for the unit of *cansignalcount* CAN signal.

CAN.signals(cansignalcount).value

type: [single] size: [1 x nummessages]

Vector of *cansignalcount* CAN signal data. See *CAN.signals(cansignalcount).relTimestamp* for relative time stamps, relative and absolute time pair matrix is found in *CAN.signals(cansignalcount).timestampPairs*.

f - frequency data

f

type: [struct] size: [scalar]

Struct container for frequency data.

f.label

type: [char cell] size: [numlines x 1]

Description of variable *f*.

f.narrow

type: [single] size: [1 x numdftlines]

Vector containing anti-alias protected[xv] single-sided frequencies.

f.oct

type: [struct] size: [scalar]

Struct container for octave-band frequency data. Which octave-band is defined in *tamaraConfig.band*. Not available if *tamaraConfig.band = narrow*. For K > 3 (as in 1/K octave-bands), the octave-band standard to use is defined in *tamaraConfig*: IEC 61672-1:2013 or IEC 61260-1:2014/ASA S1.11-2004.

f.oct.display

type: [single] size: [1 x numbands]

Vector containing centre-frequencies for octave-band in *tamaraConfig.band*. See *f.oct*. Preferred numbers in accordance with ISO 3:1973, ISO 266:1997 and ASA S1.6-1960 (where applicable).

f.oct.exact

type: [single] size: [1 x numbands]

Vector containing centre-frequencies for octave-band in *tamaraConfig.band*. See *f.oct*. Same as *f.oct.display*, but with exact values.

f.oct.matrix

type: [struct] size: [scalar]

Struct container for full octave-band frequencies in *tamaraConfig.band*. See *f.oct*.

xv Anti-alias factor can be user-modified.

f.oct.matrix.display

type: [single] size: [3 x numbands]

Matrix containing centre- and edge frequencies for octave-band in *tamaraConfig.band*. See *f.oct*. Preferred numbers in accordance with ISO 3:1973, ISO 266:1997 and ASA S1.6-1960 (where applicable). Order in first dimension: centre-, lower- and upper-frequency.

f.oct.matrix.exact

type: [single] size: [3 x numbands]

Matrix containing centre- and edge frequencies for octave-band in *tamaraConfig.band*. See *f.oct*. Same as *f.oct.matrix.display*, but with exact values.

f.octBandwidth

type: [single] size: [scalar]

Scalar *K* informing which *1/K* octave band is the default, as in *tamaraConfig.band*. Not available if *tamaraConfig.band = narrow.*

f.oct[B]

type: [struct] size: [scalar]

Struct container for 1/B octave-band frequency data, e.g. *f.oct3* for 1/3-octave band frequencies. Available if *tamaraConfig.globals.octaveDataStoring* is set to *all*. Available numbers B: 1, 3, 6, 12 and 24; 48 is available if *tamaraConfig.band = '1/48'*. See *f.oct*.

f.oct[B].display

type: [single] size: [1 x numbands]

Vector containing centre-frequencies for 1/B octave-band. See *f.oct[B]* and *f.oct.display*.

f.oct[B].exact

type: [single] size: [1 x numbands]

Vector containing centre-frequencies for 1/B octave-band. See *f.oct[B]* and *f.oct.exact*.

f.oct[B].matrix.display

type: [single] size: [3 x numbands]

Matrix containing centre- and edge frequencies for 1/B octave-band. See *f.oct[B]* and *f.oct.matrix.display*.

f.oct[B].matrix.exact

type: [single] size: [3 x numbands]

Matrix containing centre- and edge frequencies for 1/B octave-band. See *f.oct[B]* and *f.oct.matrix.exact*.

f.weights

type: [struct] size: [scalar]

Struct container for frequency weighting data.

f.weights.[wt]

type: [struct] size: [scalar]

Struct container for frequency weighting data for weighting function *wt*. Available weights: A[xvi], B[xvii], C[xviii] and D[xix].

f.weights.[wt].narrow

type: [single] size: [numdftlines x 1]

Vector containing correction factors, in dB, for weight [wt] for narrow-band frequencies. See *f.narrow* and *f.weights.[wt]*.

f.weights.[wt].oct

type: [single] size: [numbands x 1]

Vector containing correction factors, in dB, for weight [wt] for default octave-band. See *f.oct* and *f.weights.[wt]*.

f.weights.[wt].oct[B]

type: [single] size: [numbands x 1]

Vector containing correction factors, in dB, for weight [wt] for 1/B octave-band. See *f.oct[B]* and *f.weights.[wt]*.

gamma2xy - coherence function

gamma2xy

type: [struct] size: [scalar]

Struct container for coherence function data. Available if a reference[xx] channel has been defined, or if *tamaraConfig.skip_nonreference_FRFS = 0*; unless *tamaraConfig.internal.storing.variablesToSave.(<activemeasurementtype>)*.gamma2xy.*enable = 0*.

gamma2xy(n, k, l) = abs(AGxy(n, k, l)).^2./(AGxx(n, k).*AGxx(n, l))

gamma2xy.label

type: [char cell] size: [numlines x 1]

Description of variable *gamma2xy*.

gamma2xy.narrow

type: [double] size: [numdftlines x numchs x numchs]

Matrix containing narrow-band coherence data. See *gamma2xy*.

xvi IEC 61672-1:2013

xvii IEC 60651:2001

xviii IEC 61672-1:2013

xix IEC 537:1976

xx *Reference channel* is known as *Force channel* for *Mechanical Mobility* measurements, and *Inner channel* for *Sound Intensity* measurements.

gamma2xy.oct

type: [double] size: [numbands x numchs x numchs]

Matrix containing coherence data in octave-bands. Which octave-band is defined in *tamaraConfig.band*. Not available if *tamaraConfig.band = narrow.*

gamma2xy.oct[B]

type: [single] size: [numbands x numchs x numchs]

Matrix containing coherence data in octave-bands. *B* is a number corresponding to 1/*B*-bands, e.g. gamma2xy.oct3 for 1/3-octave band data. Available if *tamaraConfig.globals.octaveDataStoring* is set to *all*. Available numbers B: 1, 3, 6, 12 and 24; 48 is available if *tamaraConfig.band = '1/48'*.

GPS - GNSS/GPS data

GPS

type: [struct] size: [scalar]

Struct container for GNSS/GPS data. Available if a GNSS/GPS sensor is enabled.

GPS.info

type: [struct] size: [scalar]

Struct container for additional info.

GPS.info.positionformat

type: [char] size: [1 x numchars]

String giving information about the GNSS/GPS position format: order of latitude/longitude and reference system used.

GPS.info.timeformat

type: [char] size: [1 x numchars]

String giving information about the time format in use, typically: 'HH:MM:SS.FFF'. See *GPS.time*.

GPS.signals

type: [struct] size: [1 x numgpssignals]

Struct container for GNSS/GPS signals.

GPS.signals(gpssignalcount).messageType

type: [char] size: [1 x numchars]

String of the modifier name of *gpssignalcount* GNSS/GPS signal, e.g. 'horizontal' as in name of signal is 'accuracy' and modifier name is 'horizontal': 'accuracy: horizontal'. Is empty when no modifier needed/present. See *GPS.signals(gpssignalcount).name*. *Name* and *messageType* combination is unique.

GPS.signals(gpssignalcount).name

type: [char] size: [1 x numchars]

String of the name of *gpssignalcount* GNSS/GPS signal. Typically available names: accuracy[xxi], altitude, course, datenum, fix, latitude, longitude, nSatellites and speed.

GPS.signals(gpssignalcount).timestamp

type: [single] size: [1 x numgpstimestamps]

Vector of relative time stamps for *gpssignalcount* GNSS/GPS signal data. Absolute time is found in *GPS.time*.

GPS.signals(gpssignalcount).unit

type: [char] size: [1 x numchars]

String of the unit of *gpssignalcount* GNSS/GPS signal.

GPS.signals(gpssignalcount).value

type: [single] size: [1 x numgpstimestamps]

Vector of data for *gpssignalcount* GNSS/GPS signal.

GPS.time

type: [cell char] size: [1 x numgpstimestamps]

Cell with absolute time string for each time step. See *GPS.info.timeformat* for format information.

Gxx_max - max. hold auto-spectra

Gxx_max

type: [struct] size: [scalar]

Struct container for max. hold auto-spectra. Available if *config.globals.calcGxxMaxHold* is *true*, unless *tamaraConfig.internal.storing.variablesToSave.(<activemeasurement-type>).Gxx_max.enable = false*.

Gxx_max.label

type: [char cell] size: [numlines x 1]

Description of variable Gxx_max.

Gxx_max.narrow

type: [struct] size: [scalar]

Struct container for narrow-band Gxx_max data.

Gxx_max.narrow.dB

type: [single] size: [numdftlines x numchs]

Matrix containing max. hold auto-spectrum data. Unit is dB re. units, e.g. dB re 1 Pa. Gxx(k) = 10*log10(conj(S(k)) .* S(k)) (in dB). NB: always re. 1.

xxi Not a unique name, may have multiple entries, this applies to multiple names.See *GPS.signals(gpssignal-count).messageType*.

Gxx_max.narrow.lin

type: [double] size: [numdftlines x numchs]

Matrix containing max. hold auto-spectrum data. Linear units, e.g. Pa or V (NB: not Pa^2 etc). Gxx(k) = sqrt(conj(S(k)) .* S(k)).

HDxy - double-sided frequency response function

HDxy

type: [struct] size: [scalar]

Struct container for double-sided frequency response function.

Available if a reference channel has been defined, or if *config.globals.skip_nonreference_FRFS = false*; unless *tamaraConfig.internal.storing.variablesToSave.(<activemeasurementtype>).HDxy.enable = false*.

H-estimator used is defined in *config.globals.FRF_estimator*.

HDxy.label

type: [char cell] size: [numlines x 1]

Description of variable HDxy.

HDxy.narrow

type: [struct] size: [scalar]

Struct container for narrow-band HDxy data.

HDxy.narrow.dB

type: [single] size: [fs*T x numchs x numchs] or [fs*T x numchs]

Matrix containing double-sided FRF data in dB. See *HDxy.narrow.lin*

HDxy.narrow.lin

type: [double] size: [fs*T x numchs x numchs] or [fs*T x numchs]

Matrix containing double-sided linear FRF data. Number of dimensions depend on *config.globals.skip_nonreference_FRFS*, if fully populated (3D), second dimension is the reference channel.

HSxy - single-sided frequency response function

HSxy

type: [struct] size: [scalar]

Struct container for single-sided frequency response function.

Available if a reference channel has been defined, or if *config.globals.skip_nonreference_FRFS = false*; unless *tamaraConfig.internal.storing.variablesToSave.(<activemeasurementtype>).HSxy.enable = false.*

H-estimator used is defined in *config.globals.FRF_estimator*.

HSxy.label

type: [char cell] size: [numlines x 1]

Description of variable HSxy.

HSxy.narrow

type: [struct] size: [scalar]

Struct container for narrow-band HSxy data.

HSxy.narrow.dB

type: [single] size: [numdftlines x numchs x numchs] or [numdftlines x numchs]

Matrix containing single-sided FRF data in dB.

HSxy.narrow.lin

type: [double] size: [numdftlines x numchs x numchs] or [numdftlines x numchs]

Matrix containing single-sided linear FRF data. Number of dimensions depend on *config.globals.skip_nonreference_FRFS*, if fully populated (3D), second dimension is the reference channel.

HSxy.oct

type: [struct] size: [scalar]

Struct container for octave-band HSxy data. Which octave-band is defined in *tamaraConfig.band*. Not available if *tamaraConfig.band = narrow.*

HSxy.oct.dB

type: [single] size: [numbands x numchs x numchs]

Matrix containing FRF data in octave-bands in dB. Units are same as for *HSxy.narrow.dB*.

HSxy.oct.lin

type: [double] size: [numbands x numchs x numchs]

Matrix containing FRF data in octave-bands. Units are linear and are same as for *HSxy.narrow.lin*.

HSxy.oct[B]

type: [struct] size: [scalar]

Struct container for octave-band HSxy data. *B* is a number corresponding to 1/*B*-bands, e.g. HSxy.oct3 for 1/3-octave band data. Available if *tamaraConfig.globals.octaveDataStoring* is set to *all*. Available numbers B: 1, 3, 6, 12 and 24; 48 is available if *tamaraConfig.band = '1/48'.*

HSxy.oct[B].dB

type: [single] size: [numbands x numchs x numchs]

Matrix containing FRF data in octave-bands in dB Units are same as for HSxy.narrow.dB. See description for *HSxy.oct[B]*.

HSxy.oct[B].lin

type: [double] size: [numbands x numchs x numchs]

Matrix containing FRF data in octave-bands. Units are linear and are same as for HSxy.narrow.lin. See description for *HSxy.oct[B]*.

impulse - impulse noise measurement data

impulse

type: [struct] size: [scalar]

Struct container for impulse data. Available if t*amaraConfig.measurement = Impulse noise,* unless *tamaraConfig.internal.storing.variablesToSave.impulsenoise.impulse.enable = 0*.

impulse.avgTimeSignal

type: [struct] size: [1 x numimpulses]

Struct container for impulse data. Averaging time constant is defined in *tamaraConfig.globals.impulse.integrationTime* (ms)[xxii]. The type of time averaging is defined in *tamaraConfig.globals.impulse.integrationType*[xxiii].

impulse.avgTimeSignal(impulsecount)

type: [struct] size: [scalar]

Struct container for time-averaged data for block *impulsecount*.

impulse.avgTimeSignal(impulsecount).dB

type: [double] size: [N x numchs]

Matrix containing time-averaged data for block *impulsecount*, in dB. If only microphones are used the reference is 20 µPa[xxiv], otherwise the reference is always one—including for a mix of microphones and other sensors. See *impulse.time* for time-vector.

impulse.LEQsum

type: [struct] size: [1 x numimpulses]

Struct container for overall sound level. See *SLM.LEQ.narrow.[wt]* for available weightings. Available if only microphones are used—no sensor type mix allowed.

impulse.LEQsum(impulsecount)

type: [struct] size: [scalar]

Struct container for overall sound level for block *impulsecount*.

impulse.LEQsum(impulsecount).[wt]

type: [double] size: [1 x numchs]

Vector for overall sound level for each channel and *impulsecount* block for weighting *wt*. See *impulse.LEQsum*. Unit is in dB re 20 µPa.

xxii Also found in *tamaraConfig.globals.impulse.integrationTimeUnit*.

xxiii Available types: EMA, EMA (zero-phase), MA and MA (zero-phase)—EMA = Exponential Moving Average.

xxiv Unless tamaraConfig.globals.engineeringUnits.pressure.val have been modified.

impulse.Sxy

type: [struct] size: [scalar]

Struct container for impulse data.

impulse.Sxy.narrow

type: [struct] size: [scalar]

Struct container for block-wise auto-spectrum data in narrow-band.

impulse.Sxy.narrow.dB

type: [double] size: [numimpulses x numchs x numdftlines]

Matrix containing auto-spectrum data for each impulse block. Unit is dB re. units, e.g. dB re 1 Pa. Gxx(k) = 10*log10(conj(S(k)) .* S(k)) (in dB). NB: always re. 1. See *f.narrow* for frequency vector.

impulse.Sxy.narrow.lin

type: [double] size: [numimpulses x numchs x numdftlines]

Matrix containing auto-spectrum data for each impulse block. Linear units, e.g. Pa or V (NB: not Pa^2 etc). Gxx(k) = sqrt(conj(S(k)) .* S(k)). See *f.narrow* for frequency vector.

impulse.time

type: [double] size: [1 x N]

Vector containing the time-vector for one impulse block—relative time.

Info - general info

Info

type: [struct] size: [scalar]

Struct container for general measurement information.

Info.bandType

type: [char] size: [1 x numchars]

Selected bandtype, see *config.band*.

Info.calibrated

type: [cell char] size: [1 x numchs]

Cell containing a 'yes' or 'no' signifying whether the channel was calibrated in Tamara prior to start of measurement.

Info.chNames

type: [cell char] size: [1 x numchs]

Cell containing a string of channel name for each channel, *Info.chNames{chcount}*.

Info.chUnits

type: [cell char] size: [1 x numchs]

Cell containing a string of unit for each channel, *Info.chUnits{chcount}*.

Info.comment

type: [cell char] size: [1 x numcomments]

Cell containing a string for each time-stamped measurement comment. Only available when comments have been entered.

Info.df

type: [double] size: [scalar]

Frequency resolution in Hz.

Info.duration

type: [double] size: [scalar]

Duration of measurement in seconds.

Info.Filename

type: [char] size: [1 x numchars]

Name of file as of storing time.

Info.fileVersion

type: [char] size: [1 x numchars]

String describing the used Tamara file specification file version, e.g. "1.2.0".

Info.frequencyRange

type: [double] size: [1 x 2]

Selected frequency range: [lower, upper], in Hz.

Info.frfEstimator

type: [char] size: [1 x 2]

FRF-estimator selected, see *config.globals.FRF_estimator*.

Info.frfMode

type: [char] size: [1 x numchars]

FRF reference mode: *all*, *single* or *none*.

Info.frfReference

type: [uint32] size: [1 x 1 or 1 x 2]

Channel(s) used a FRF reference—if H4-estimator used two values are present. Optional field.

Info.fs

type: [double] size: [scalar]

Sampling rate in Hz.

Info.index

type: [struct] size: [scalar]

Struct container for channel index information, such as index of tacho channels.

Info.ISO9614

type: [struct] size: [scalar]

Struct container for ISO9614 settings, see *config.globals.ISO9614_1*, with an added field: *dB_reference*.

Info.LR

type: [struct] size: [scalar]

Struct container for Level recorder settings, see *config.globals.LR*.

Info.measurementType

type: [char] size: [1 x numchars]

Name of Tamara measurement type, e.g. 'Level recorder'.

Info.N

type: [uint32] size: [scalar]

Number of processed blocks, *N*.

Info.note

type: [char] size: [1 x numchars]

String for measurement note. Only available when a measurement note has been entered.

Info.numberOfChannels

type: [uint32] size: [scalar]

Number of channels in measurement, *numchs*.

Info.overlap

type: [char] size: [1 x numchars]

DFT overlap, e.g. '75 %'.

Info.overload

type: [struct] size: [scalar]

Struct container for overload information.

Info.overload.any

type: [logical] size: [scalar]

True if any any overloads detected during measurement, *false* otherwise.

Info.overload.N

type: [int64] size: [1 x numchs]

Number of overloaded block per channel.

Info.overload.P

type: [double] size: [1 x numchs]

Relative amount of overloaded blocks per channel, 0–1.

Info.SQ

type: [struct] size: [scalar]

Struct container for sound quality metrics information.

Info.SQ.AI

type: [struct] size: [scalar]

Struct container for articulation index information.

Info.SQ.AI.method

type: [char] size: [1 x numchars]

String of the AI method, 'open' or 'closed'.

Info.SQ.PR

type: [struct] size: [scalar]

Struct container for articulation index information.

Info.SQ.PR.numberOfTones

type: [uint16] size: [scalar]

Number of maximum prominence ratio tones.

Info.SQ.stationaryloudness

type: [struct] size: [scalar]

Struct container for stationary loudness information.

Info.SQ.stationaryloudness.field

type: [char] size: [1 x numchars]

Sound field used in stationary loudness calculation.

Info.SQ.stationaryloudness.method

type: [char] size: [1 x numchars]

Standard method used in stationary loudness method.

Info.SQ.timevaryingloudness

type: [struct] size: [scalar]

Struct container for timevarying loudness information.

Info.SQ.timevaryingloudness.field

type: [char] size: [1 x numchars]

Sound field used in timevarying loudness calculation.

Info.structs

type: [struct] size: [scalar]

Struct container mimicking the full dataset available in measurement file.

Info.tacho

type: [struct] size: [1 x numtachos]

Struct container for tacho data.

Info.tacho(tachocount).dir

type: [char] size: [1 x numchars]

Direction of speed signal, if sweep, typically *up*. If it is unknown it is empty.

Info.tacho(tachocount).label

type: [char] size: [1 x numchars]

Tacho channel name.

Info.tacho(tachocount).name

type: [char] size: [1 x numchars]

Fieldname of tacho channel, e.g. *tacho_1*, as stored in .mat-file.

Info.tacho(tachocount).pulses_per_rev

type: [double] size: [scalar]

Number of pulses per revolution.

Info.tacho(tachocount).unit

type: [char] size: [1 x numchars]

Unit for tacho channel, typically *rpm*.

Info.time

type: [char] size: [1 x numchars]

Absolute time at point of save. See *Info.timeFormat*.

Info.timeFormat

type: [char] size: [1 x numchars]

Description of timeformat used in Info.time, .e.g. 'yyyy mmm dd - HH:MM:SS'.

Info.timeSignal

type: [char] size: [1 x numchars]

Description of timeSignal variable. See *timeSignal*.

Info.version

type: [char] size: [1 x numchars]

Tamara version, same as *tamaraConfig.version{end}*.

Info.window

type: [struct] size: [scalar]

Struct container for DFT window information. If multiple DFT windows enabled, it will instead be a cell array of enabled DFT windows.

Info.window.name

type: [cell] size: [1 x numwindows]

Name of enabled DFT windows.

Info.window.correction

type: [cell] size: [1 x numwindows]

Correction type of enabled windows.

Info.window.correctionToEnergy

type: [cell] size: [1 x numwindows]

Correction factor to convert data to energy-scaling, equal to 1 if data already energy-scaled.

IR - impulse response function

IR

type: [struct] size: [scalar]

Struct container for impulse response function data. Available if a reference channel has been defined, or if *tamaraConfig.skip_nonreference_FRFS = 0*; unless *tamaraConfig.internal.storing.variablesToSave.(<activemeasurementtype>).IR.enable = 0*.

IR.label

type: [char cell] size: [numlines x 1]

Description of variable IR.

IR.lin

type: [double] size: [fs*T x numchs x numchs]

Matrix containing impulse response data.

ISO9614 - sound intensity measurement data

ISO9614

type: [struct] size: [scalar]

Struct container for ISO 9614[xxv] sound intensity measurement data.

ISO9614.LWsum

type: [struct] size: [scalar]

Struct container for ISO 9614 sound power, overall levels.

ISO9614.LWsum.Z

type: [double] size: [scalar]

Overall sound power, unweighted.

ISO9614.LWsum.A

type: [double] size: [scalar]

Overall sound power, A-weighted[xxvi].

xxv ISO 9614-1:1993

xxvi IEC 61672-1:2013

ISO9614.LWsum.D

type: [double] size: [scalar]

Overall sound power, D-weighted[xxvii].

ISO9614.LWtotal

type: [struct] size: [scalar]

Struct container for ISO 9614 total sound power spectrum.

ISO9614.LWtotal.Z

type: [double] size: [scalar]

Overall sound power, unweighted. Narrow-band data, frequency vector in *f.narrow*.

ISO9614.LWtotal.A

type: [double] size: [scalar]

Overall sound power, A-weighted[xxviii]. Narrow-band data, frequency vector in *f.narrow*.

ISO9614.LWtotal.D

type: [double] size: [scalar]

Total sound power, D-weighted[xxix]. Narrow-band data, frequency vector in *f.narrow*.

ISO9614.LWtotal.oct

type: [double] size: [scalar]

Struct container for ISO 9614 total sound power spectrum in default octave band. See *f.oct*.

ISO9614.LWtotal.oct.[wt]

type: [double] size: [1 x numbands]

Total sound power spectrum in default octave band for weight *wt*. See *f.oct*. For available weights, see *f.weights.[wt]*.

ISO9614.LWtotal.oct[B]

type: [struct] size: [scalar]

Struct container for ISO 9614 total sound power spectrum in 1/B octave band. See *f.oct[B]*.

ISO9614.LWtotal.oct[B].[wt]

type: [double] size: [1 x numbands]

Total sound power spectrum in 1/B octave band for weight *wt*. See *f.oct[B]*. For available weights, see *f.weights.[wt]*.

xxvii IEC 537:1976

xxviii IEC 61672-1:2013

xxix IEC 537:1976

ISO9614.pt

type: [struct] size: [numpoints x numsides]

Struct container for measurement data for each measurement point (dimension 1) and measurement side (dimension 2). Matrix is fully rectangular, e.g. if maximum number of points of side 1 is 3, but 5 for side 2, ISO9614.pt(4, 1) exists but is empty. Number of points and sides are found/defined in *tamaraConfig.ISO9614_1.segments*.

ISO9614.pt(pt, segment)

type: [struct] size: [scalar]

Struct container for measurement data for point *pt* and side *segment*.

ISO9614.pt(pt, segment).AGxy

type: [struct] size: [scalar]

Struct container for AGxy for point *pt* and side *segment*. See *AGxy*.

ISO9614.pt(pt, segment).HSxy

type: [struct] size: [scalar]

Struct container for HSxy for point *pt* and side *segment*. See *HSxy*.

ISO9614.pt(pt, segment).gamma2xy

type: [struct] size: [scalar]

Struct container for HSxy for point *pt* and side *segment*. See *gamma2xy*.

ISO9614.pt(pt, segment).LW

type: [struct] size: [scalar]

Struct container for sound power for point *pt* and side *segment*.

ISO9614.pt(pt, segment).LW.abs

type: [struct] size: [scalar]

Struct container for absolute sound power for point *pt* and side *segment*.

ISO9614.pt(pt, segment).LW.abs.values

type: [double] size: [numdftlines x 1]

Vector of narrow-band absolute sound power for point *pt* and side *segment*.

ISO9614.pt(pt, segment).LW.abs.sum

type: [double] size: [scalar]

Total absolute sound power for point *pt* and side *segment*.

ISO9614.pt(pt, segment).LW.neg

type: [struct] size: [scalar]

Struct container for negative going sound power for point *pt* and side *segment*.

ISO9614.pt(pt, segment).LW.neg.values

type: [double] size: [numdftlines x 1]

Vector of narrow-band negative going sound power for point *pt* and side *segment*.

ISO9614.pt(pt, segment).LW.neg.sum

type: [double] size: [scalar]

Total negative going sound power for point *pt* and side *segment*.

ISO9614.pt(pt, segment).LW.pos

type: [struct] size: [scalar]

Struct container for positive going sound power for point *pt* and side *segment*.

ISO9614.pt(pt, segment).LW.pos.values

type: [double] size: [numdftlines x 1]

Vector of narrow-band negative going sound power for point *pt* and side *segment*.

ISO9614.pt(pt, segment).LW.pos.sum

type: [double] size: [scalar]

Total positive going sound power for point *pt* and side *segment*.

ISO9614.pt(pt, segment).I

type: [double] size: [numdftlines x 1]

Vector of narrow-band sound intensity for point *pt* and side *segment*.

LR - level recorder data

LR

type: [struct] size: [scalar]

Struct container for level recorder data. Available if t*amaraConfig.measurement = Level recorder,* unless *tamaraConfig.internal.storing.variablesToSave.levelrecorder.LR.enable = 0*.

LR.histogram

type: [struct] size: [scalar]

Struct container for level histogram data. All levels are according to *LR.levels* (with regards to scaling and units).

LR.histogram.[wt]

type: [struct] size: [scalar]

Struct container for level histogram data for weighting *wt*.

LR.histogram.[wt].N

type: [uint32] size: [numedges x numchs]

Matrix containing number of occurances for each bin and channel, for weighting *wt*.

LR.histogram.[wt].P

type: [single] size: [numedges x numchs]

Matrix containing probability for each bin and channel, for weighting *wt*. Unit is %. Sum for each channel is 100 %.

LR.histogram.[wt].xCenter

type: [single] size: [numedges x 1]

Vector containing the histogram bin center values for weighting *wt*. See *LR.histogram.[wt].xEdges*.

LR.histogram.[wt].xEdges

type: [single] size: [numedges x 2]

Matrix containing the histogram bin edges for weighting *wt*. The step-size is defined in *tamaraConfig.globals.LR.histogramResolution*. Unit is dB. The number of edges is dependant on signal.

LR.levels

type: [struct] size: [scalar]

Struct container for level data.

LR.levels.avg

type: [struct] size: [scalar]

Struct container for block-wise average (rms) levels. See *LR.levels.y*.

LR.levels.avg.[wt]

type: [double] size: [numlrtimesteps x numchs]

Matrix containing block-wise average (rms) levels for weighting *wt*. See *LR.levels.y.[wt]*.

LR.levels.bp

type: [struct] size: [1 x numlrcustombands]

Struct container for custom filter bank band-passed level data. Available if *tamaraConfig.globals.LR.useOctFilters = -1*.

LR.levels.bp(bandcount)

type: [struct] size: [1 x numlrcustombands]

Struct container for band-passed level data.

LR.levels.bp(bandcount).[wt]

type: [struct] size: [1 x numlrcustombands]

Struct container for correction factors for band-passed level data for weight *wt*. Available weights are defined in *LR.weighting* plus weights A, B, C and D, see *f.weights.[wt]*.

LR.levels.bp(bandcount).[wt].dB

type: [single] size: [scalar]

Correction factors for band-passed (band *bandcount*) level data for weight *wt* in dB. See *LR.levels.bp(bandcount).[wt]*.

LR.levels.bp(bandcount).[wt].lin

type: [double] size: [scalar]

Correction factors for band-passed (band *bandcount*) level data for weight *wt* in linear unit. See *LR.levels.bp(bandcount).[wt]*.

LR.levels.bp(bandcount).f

type: [struct] size: [scalar]

Struct container for frequency information for the *bandcount*:th band.

LR.levels.bp(bandcount).f.center

type: [single] size: [scalar]

Centre frequency for the *bandcount*:th band.

LR.levels.bp(bandcount).f.edges

type: [single] size: [2 x 1]

Edge frequencies (-6 dB) for the *bandcount*:th band.

LR.levels.bp(bandcount).scalars

type: [single] size: [scalar]

Struct container for bandpass filter scalar levels.

LR.levels.bp(bandcount).y

type: [single] size: [numlrtimesteps x numchs]

Matrix containing time averaged unweighted level data for the *bandcount*:th band. Apply weightings from *LR.levels.bp(bandcount).[wt].dB*. Units according to *LR.levels.y*.

LR.levels.max

type: [struct] size: [scalar]

Struct container for block-wise maximum levels. See *LR.levels.y*.

LR.levels.max.[wt]

type: [double] size: [numlrtimesteps x numchs]

Matrix containing block-wise maximum levels for weighting *wt*. See *LR.levels.y.[wt]*.

LR.levels.min

type: [struct] size: [scalar]

Struct container for block-wise maximum levels. See *LR.levels.y*.

LR.levels.min.[wt]

type: [double] size: [numlrtimesteps x numchs]

Matrix containing block-wise minimum levels for weighting *wt*. See *LR.levels.y.[wt]*.

LR.levels.oct

type: [struct] size: [1 x numlrbands]

Struct container for *1/K*[xxx] band-passed level data. Available if *tamaraConfig.globals.LR.useOctFilters = 1*.

LR.levels.oct(bandcount)

type: [struct] size: [1 x numlrbands]

Struct container for band-passed level data.

xxx See *f.oct*.

LR.levels.oct(bandcount).[wt]

type: [struct] size: [1 x numlrbands]

Struct container for correction factors for band-passed level data for weight *wt*. Available weights are defined in *LR.weighting* plus weights A, B, C and D, see *f.weights.[wt]*.

LR.levels.oct(bandcount).[wt].dB

type: [single] size: [scalar]

Correction factors for band-passed (band *bandcount*) level data for weight *wt* in dB. See *LR.levels.oct(bandcount).[wt]*.

LR.levels.oct(bandcount).[wt].lin

type: [double] size: [scalar]

Correction factors for band-passed (band *bandcount*) level data for weight *wt* in linear unit. See *LR.levels.oct(bandcount).[wt]*.

LR.levels.oct(bandcount).f

type: [struct] size: [scalar]

Struct container for frequency information for the *bandcount*:th band.

LR.levels.oct(bandcount).f.display

type: [single] size: [scalar]

Centre frequency for the *bandcount*:th band, using preferred number, see *f.oct.display*.

LR.levels.oct(bandcount).f.exact

type: [single] size: [scalar]

Centre frequency for the *bandcount*:th band, using exact value, see *f.oct.exact*.

LR.levels.oct(bandcount).f.edges

type: [single] size: [2 x 1]

Edge frequencies (lower- and upper bound) for the *bandcount*:th band.

LR.levels.oct(bandcount).scalars

type: [single] size: [scalar]

Struct container for octave-band scalar levels.

LR.levels.oct(bandcount).scalars.avg.[wt]

type: [double] size: [1 x numchs]

Vector containing average levels for octave-band.

LR.levels.oct(bandcount).scalars.max.[wt]

type: [double] size: [1 x numchs]

Vector containing maximum levels for octave-band.

LR.levels.oct(bandcount).scalars.min

type: [double] size: [1 x numchs]

Vector containing minimum levels for octave-band.

LR.levels.oct(bandcount).y

type: [single] size: [numlrtimesteps x numchs]

Matrix containing time averaged unweighted level data for the *bandcount*:th band. Apply weightings from *LR.levels.oct(bandcount).[wt].dB*. Units according to *LR.levels.y*.

LR.levels.scalars

type: [struct] size: [scalar]

Struct container for global scalar levels.

LR.levels.scalars.avg.[wt]

type: [double] size: [1 x numchs]

Vector containing average levels for weighting *wt*. See *LR.levels.y.[wt]*.

LR.levels.scalars.max.[wt]

type: [double] size: [1 x numchs]

Vector containing maximum levels for weighting *wt*. See *LR.levels.y.[wt]*.

LR.levels.scalars.min.[wt]

type: [double] size: [1 x numchs]

Vector containing minimum levels for weighting *wt*. See *LR.levels.y.[wt]*.

LR.levels.x

type: [single] size: [numlrtimesteps x 1]

Time vector for level data. Unit is seconds. First timestep is $1/T_{LR}$[xxxi]. Interpolated if *tamaraConfig.globals.LR.interpolate = true*, otherwise it will be approximated to nearest integer time step (*1/fs*), and thus not always regularly spaced.

LR.levels.x_block

type: [single] size: [numlrtimesteps x 1]

Time vector for block-wise level data. Unit is seconds. Time of halfway sample of each block is used.

LR.levels.y

type: [struct] size: [scalar]

Struct container for levels. Units for all level data is according to *Info.chUnits*. All levels are in dB: $10 \times \log10(s^2/ref^2)$—ref is according to *tamaraConfig.globals.engineeringUnits.(<physicalquantity>).val*. See *LR.levels.x*.

LR.levels.y.[wt]

type: [single] size: [numlrtimesteps x numchs]

Matrix containing levels for weighting *wt*[xxxii]. Sampled at times found in *LR.levels.x*, using the exponential time constant in *tamaraConfig.globals.LR.writingSpeedTime*[xxxiii].

xxxi Defined in *tamaraConfig.globals.LR.paperSpeed*

xxxii Available field names are found in *LR.weighting*.

xxxiii If number of elements are more than one, and matching numlrweights: *LR.levels.y.(LR.weighting{k})* is using time constant *tamaraConfig.globals.LR.writingSpeedTime(k)*.

LR.percentile

type: [struct] size: [scalar]

Struct container for level percentile data. All levels are according to *LR.levels* (with regards to scaling and units).

LR.percentile.[wt]

type: [struct] size: [scalar]

Struct container for level percentile data for weighting *wt*.

LR.percentile.[wt].array

type: [struct] size: [scalar]

Struct container for full level percentile data for weighting *wt*. All levels are according to LR.levels (with regards to scaling and units).

LR.percentile.[wt].array.L

type: [single] size: [399 x numchs]

Matrix containing levels for percentiles in *LR.percentile.[wt].array.P* for each channel. All levels are according to LR.levels (with regards to scaling and units).

LR.percentile.[wt].array.P

type: [single] size: [399 x 1]

Vector containing evaluated percentiles: 0.25 to 99.75 in 0.25 steps.

LR.percentile.[wt].P5

type: [single] size: [1 x numchs]

Vector containing 5th percentiles for each channel for weighting *wt*.

LR.percentile.[wt].P10

type: [single] size: [1 x numchs]

Vector containing 10th percentiles for each channel for weighting *wt*.

LR.percentile.[wt].P25

type: [single] size: [1 x numchs]

Vector containing 25th percentiles for each channel for weighting *wt*.

LR.percentile.[wt].P50

type: [single] size: [1 x numchs]

Vector containing 50th percentiles for each channel for weighting *wt*.

LR.percentile.[wt].P75

type: [single] size: [1 x numchs]

Vector containing 75th percentiles for each channel for weighting *wt*.

LR.percentile.[wt].P90

type: [single] size: [1 x numchs]

Vector containing 90th percentiles for each channel for weighting *wt*.

LR.percentile.[wt].P95

type: [single] size: [1 x numchs]

Vector containing 95th percentiles for each channel for weighting *wt*.

LR.referenceUnitOffset

type: [double] size: [1 x numchs]

Vector of used dB reference unit values for levels. NB: in dB.

LR.weighting

type: [char cell] size: [numlrweights x 1]

Cell containing all enabled weighting functions used for LR.

mech - mechanical mobility[xxxiv] data

mech

type: [struct] size: [scalar]

Struct container for mechanical data. Available if *Mechanical mobility* is the selected measurement type[xxxv].

mech.accelerance

type: [struct] size: [scalar]

Struct container for mechanical accelerance. Available unless *tamaraConfig.internal.storing.variablesToSave.mechanicalmobility.accelerance.enable* is *false*.

mech.accelerance.label

type: [char cell] size: [numlines x 1]

Description of variable *mech.accelerance*.

mech.accelerance.narrow

type: [struct] size: [scalar]

Struct container for narrow-band accelerance data

mech.accelerance.narrow.dB

type: [single] size: [numdftlines x numchs-1]

Matrix containing accelerance data in dB. Unit is dB re. 1/kg. See *mech.accelerance.narrow.lin*.

mech.accelerance.narrow.lin

type: [double] size: [numdftlines x numchs-1]

Matrix containing accelerance data. Linear units, $(m/s^2)/N = 1/kg$. Force channel is excluded (numchs-1).

xxxiv Here mechanical mobility refers to all similar mechanical properties such as impedance, stiffness etc.

xxxv *tamaraConfig.measurement*

mech.accelerance.oct

type: [struct] size: [scalar]

Struct container for octave-band accelerance data. Which octave-band is defined in *tamaraConfig.band*. Not available if *tamaraConfig.band = narrow.*

mech.accelerance.oct.dB

type: [single] size: [numbands x numchs-1]

Matrix containing accelerance data in octave-bands in dB. See *mech.accelerance.narrow.dB*.

mech.accelerance.oct.lin

type: [double] size: [numbands x numchs-1]

Matrix containing accelerance data in octave-bands. See *mech.accelerance.narrow.lin*.

mech.accelerance.oct[B]

type: [struct] size: [scalar]

Struct container for octave-band accelerance data. *B* is a number corresponding to 1/*B*-bands, e.g. mech.accelerance.oct3 for 1/3-octave band data. Available if *tamaraConfig.globals.octaveDataStoring* is set to *all*. Available numbers B: 1, 3, 6, 12 and 24; 48 is available if *tamaraConfig.band = '1/48'.*

mech.accelerance.oct[B].dB

type: [single] size: [numbands x numchs-1]

Matrix containing accelerance data in octave-band *B*. See *mech.accelerance.narrow.dB* and *mech.accelerance.oct[B]*.

mech.accelerance.oct[B].lin

type: [double] size: [numbands x numchs-1]

Matrix containing accelerance data in octave-band *B*. See *mech.accelerance.narrow.lin* and *mech.accelerance.oct[B]*.

mech.compliance

type: [struct] size: [scalar]

Struct container for mechanical compliance. Available unless *tamaraConfig.internal.storing.variablesToSave.mechanicalmobility.compliance.enable* is *false*.

mech.compliance.label

type: [char cell] size: [numlines x 1]

Description of variable *mech.compliance*.

mech.compliance.narrow

type: [struct] size: [scalar]

Struct container for narrow-band compliance data

mech.compliance.narrow.dB

type: [single] size: [numdftlines x numchs-1]

Matrix containing compliance data in dB. Unit is dB re. 1 m/N. See *mech.compliance.narrow.lin*.

mech.compliance.narrow.lin

type: [double] size: [numdftlines x numchs-1]

Matrix containing compliance data. Linear units, m/N. Force channel is excluded (numchs-1).

mech.compliance.oct

type: [struct] size: [scalar]

Struct container for octave-band compliance data. Which octave-band is defined in *tamaraConfig.band*. Not available if *tamaraConfig.band = narrow*.

mech.compliance.oct.dB

type: [single] size: [numbands x numchs-1]

Matrix containing compliance data in octave-bands in dB. See *mech.compliance.narrow.dB*.

mech.compliance.oct.lin

type: [double] size: [numbands x numchs-1]

Matrix containing compliance data in octave-bands. See *mech.compliance.narrow.lin*.

mech.compliance.oct[B]

type: [struct] size: [scalar]

Struct container for octave-band compliance data. *B* is a number corresponding to 1/*B*-bands, e.g. mech.compliance.oct3 for 1/3-octave band data. Available if *tamaraConfig.globals.octaveDataStoring* is set to *all*. Available numbers B: 1, 3, 6, 12 and 24; 48 is available if *tamaraConfig.band = '1/48'*.

mech.compliance.oct[B].dB

type: [single] size: [numbands x numchs-1]

Matrix containing compliance data in octave-band *B*. See *mech.compliance.narrow.dB* and *mech.compliance.oct[B]*.

mech.compliance.oct[B].lin

type: [double] size: [numbands x numchs-1]

Matrix containing compliance data in octave-band *B*. See *mech.compliance.narrow.lin* and *mech.compliance.oct[B]*.

mech.impedance

type: [struct] size: [scalar]

Struct container for mechanical impedance. Available unless *tamaraConfig.internal.storing.variablesToSave.mechanicalmobility.impedance.enable* is *false*.

mech.impedance.label

type: [char cell] size: [numlines x 1]

Description of variable *mech.impedance*.

mech.impedance.narrow

type: [struct] size: [scalar]

Struct container for narrow-band impedance data

mech.impedance.narrow.dB

type: [single] size: [numdftlines x numchs-1]

Matrix containing impedance data in dB. Unit is dB re. 1 N/(m/s). See *mech.impedance.narrow.lin*.

mech.impedance.narrow.lin

type: [double] size: [numdftlines x numchs-1]

Matrix containing impedance data. Linear units, N/(m/s) = kg/s. Force channel is excluded (numchs-1).

mech.impedance.oct

type: [struct] size: [scalar]

Struct container for octave-band impedance data. Which octave-band is defined in *tamaraConfig.band*. Not available if *tamaraConfig.band = narrow.*

mech.impedance.oct.dB

type: [single] size: [numbands x numchs-1]

Matrix containing accelerance data in octave-bands in dB. See *mech.impedance.narrow.dB*.

mech.impedance.oct.lin

type: [double] size: [numbands x numchs-1]

Matrix containing accelerance data in octave-bands. See *mech.impedance.narrow.lin*.

mech.impedance.oct[B]

type: [struct] size: [scalar]

Struct container for octave-band impedance data. *B* is a number corresponding to 1/*B*-bands, e.g. mech.impedance.oct3 for 1/3-octave band data. Available if *tamaraConfig.globals.octaveDataStoring* is set to *all*. Available numbers B: 1, 3, 6, 12 and 24; 48 is available if *tamaraConfig.band = '1/48'.*

mech.impedance.oct[B].dB

type: [single] size: [numbands x numchs-1]

Matrix containing impedance data in octave-band *B*. See *mech.impedance.narrow.dB* and *mech.impedance.oct[B]*.

mech.impedance.oct[B].lin

type: [double] size: [numbands x numchs-1]

Matrix containing impedance data in octave-band *B*. See *mech.impedance.narrow.lin* and *mech.impedance.oct[B]*.

mech.inertia

type: [struct] size: [scalar]

Struct container for mechanical inertia. Available unless *tamaraConfig.internal.storing.variablesToSave.mechanicalmobility.inertia.enable* is *false*.

mech.inertia.label

type: [char cell] size: [numlines x 1]

Description of variable *mech.inertia*.

mech.inertia.narrow

type: [struct] size: [scalar]

Struct container for narrow-band inertia data

mech.inertia.narrow.dB

type: [single] size: [numdftlines x numchs-1]

Matrix containing inertia data in dB. Unit is dB re. 1 kg. See *mech.inertia.narrow.lin*.

mech.inertia.narrow.lin

type: [double] size: [numdftlines x numchs-1]

Matrix containing inertia data. Linear units, N/(m/s^2) = kg. Force channel is excluded (numchs-1).

mech.inertia.oct

type: [struct] size: [scalar]

Struct container for octave-band inertia data. Which octave-band is defined in *tamaraConfig.band*. Not available if *tamaraConfig.band = narrow.*

mech.inertia.oct.dB

type: [single] size: [numbands x numchs-1]

Matrix containing inertia data in octave-bands in dB. See *mech.inertia.narrow.dB*.

mech.inertia.oct.lin

type: [double] size: [numbands x numchs-1]

Matrix containing inertia data in octave-bands. See *mech.inertia.narrow.lin*.

mech.inertia.oct[B]

type: [struct] size: [scalar]

Struct container for octave-band inertia data. *B* is a number corresponding to 1/*B*-bands, e.g. mech.inertia.oct3 for 1/3-octave band data. Available if *tamaraConfig.globals.octaveDataStoring* is set to *all*. Available numbers B: 1, 3, 6, 12 and 24; 48 is available if *tamaraConfig.band = '1/48'.*

mech.inertia.oct[B].dB

type: [single] size: [numbands x numchs-1]

Matrix containing inertia data in octave-band *B*. See *mech.inertia.narrow.dB* and *mech.inertia.oct[B]*.

mech.inertia.oct[B].lin

type: [double] size: [numbands x numchs-1]

Matrix containing inertia data in octave-band *B*. See *mech.inertia.narrow.lin* and *mech.inertia.oct[B]*.

mech.mobility

type: [struct] size: [scalar]

Struct container for mechanical mobility. Available unless *tamaraConfig.internal.storing.variablesToSave.mechanicalmobility.mobility.enable* is *false*.

mech.mobility.label

type: [char cell] size: [numlines x 1]

Description of variable *mech.mobility*.

mech.mobility.narrow

type: [struct] size: [scalar]

Struct container for narrow-band mobility data

mech.mobility.narrow.dB

type: [single] size: [numdftlines x numchs-1]

Matrix containing mobility data in dB. Unit is dB re. 1 (m/s)/N. See *mech.mobility.narrow.lin*.

mech.mobility.narrow.lin

type: [double] size: [numdftlines x numchs-1]

Matrix containing mobility data. Linear units, (m/s)/N. Force channel is excluded (numchs-1).

mech.mobility.oct

type: [struct] size: [scalar]

Struct container for octave-band mobility data. Which octave-band is defined in *tamaraConfig.band*. Not available if *tamaraConfig.band = narrow.*

mech.mobility.oct.dB

type: [single] size: [numbands x numchs-1]

Matrix containing mobility data in octave-bands in dB. See *mech.mobility.narrow.dB*.

mech.mobility.oct.lin

type: [double] size: [numbands x numchs-1]

Matrix containing mobility data in octave-bands. See *mech.mobility.narrow.lin*.

mech.mobility.oct[B]

type: [struct] size: [scalar]

Struct container for octave-band mobility data. *B* is a number corresponding to 1/*B*-bands, e.g. mech.mobility.oct3 for 1/3-octave band data. Available if *tamaraConfig.globals.octaveDataStoring* is set to *all*. Available numbers B: 1, 3, 6, 12 and 24; 48 is available if *tamaraConfig.band = '1/48'.*

mech.mobility.oct[B].dB

type: [single] size: [numbands x numchs-1]

Matrix containing mobility data in octave-band *B*. See *mech.mobility.narrow.dB* and *mech.mobility.oct[B]*.

mech.mobility.oct[B].lin

type: [double] size: [numbands x numchs-1]

Matrix containing mobility data in octave-band *B*. See *mech.mobility.narrow.lin* and *mech.mobility.oct[B]*.

mech.stiffness

type: [struct] size: [scalar]

Struct container for mechanical stiffness. Available unless *tamaraConfig.internal.storing.variablesToSave.mechanicalmobility.stiffness.enable* is *false*.

mech.stiffness.label

type: [char cell] size: [numlines x 1]

Description of variable *mech.stiffness*.

mech.stiffness.narrow

type: [struct] size: [scalar]

Struct container for narrow-band stiffness data

mech.stiffness.narrow.dB

type: [single] size: [numdftlines x numchs-1]

Matrix containing stiffness data in dB. Unit is dB re. 1 N/m. See *mech.stiffness.narrow.lin*.

mech.stiffness.narrow.lin

type: [double] size: [numdftlines x numchs-1]

Matrix containing stiffness data. Linear units, N/m. Force channel is excluded (numchs-1).

mech.stiffness.oct

type: [struct] size: [scalar]

Struct container for octave-band stiffness data. Which octave-band is defined in *tamaraConfig.band*. Not available if *tamaraConfig.band = narrow.*

mech.stiffness.oct.dB

type: [single] size: [numbands x numchs-1]

Matrix containing stiffness data in octave-bands in dB. See *mech.stiffness.narrow.dB*.

mech.stiffness.oct.lin

type: [double] size: [numbands x numchs-1]

Matrix containing stiffness data in octave-bands. See *mech.stiffness.narrow.lin*.

mech.stiffness.oct[B]

type: [struct] size: [scalar]

Struct container for octave-band stiffness data. *B* is a number corresponding to 1/*B*-bands, e.g. mech.stiffness.oct3 for 1/3-octave band data. Available if *tamaraConfig.globals.octaveDataStoring* is set to *all*. Available numbers B: 1, 3, 6, 12 and 24; 48 is available if *tamaraConfig.band = '1/48'.*

mech.stiffness.oct[B].dB

type: [single] size: [numbands x numchs-1]

Matrix containing accelerance data in octave-band *B*. See *mech.stiffness.narrow.dB* and *mech.stiffness.oct[B]*.

mech.stiffness.oct[B].lin

type: [double] size: [numbands x numchs-1]

Matrix containing stiffness data in octave-band *B*. See *mech.stiffness.narrow.lin* and *mech.stiffness.oct[B]*.

PSD - averaged power spectral density

PSD

type: [struct] size: [scalar]

Struct container for PSD. Always available, unless *tamaraConfig.internal.storing.variablesToSave.(<activemeasurementtype>).PSD.enable = 0*.

PSD.label

type: [char cell] size: [numlines x 1]

Description of variable PSD.

PSD.narrow

type: [struct] size: [scalar]

Struct container for narrow-band PSD data

PSD.narrow.dB

type: [single] size: [numdftlines x numchs]

Matrix containing averaged PSD data. Unit is dB re. units, e.g. dB re 1 Pa. PSD(k) = 10*log10(conj(S(k)) .* S(k)) (in dB).

PSD.narrow.lin

type: [double] size: [numdftlines x numchs]

Matrix containing averaged PSD data. Linear units, e.g. Pa or V (NB: not Pa2 etc). PSD(k) = sqrt(conj(S(k)) .* S(k))

REV - reverberation time data

REV

type: [struct] size: [scalar]

Struct container for reverberation time measurement data.

REV.T60

type: [double] size: [numbands x numchs-1]

Matrix with T_{60} time for each band and band. NaN when no fit possible, or bad data.

REV.R2

type: [double] size: [numbands x numchs-1]

Matrix with R^2 correlation coefficients for each band and band. NaN when no fit possible, or bad data.

REV.Lrange

type: [double] size: [numbands x numchs-1]

Matrix with ΔL used for T_{60} for each band and band. NaN when no fit possible, or bad data.

SLM - sound level meter data

SLM

type: [struct] size: [scalar]

Struct container for sound level meter data. Available if *tamaraConfig.measurement = Sound level meter,* unless *tamaraConfig.internal.storing.variablesToSave*.soundlevelmeter.*SLM.enable = 0*.

SLM.duration

type: [double] size: [scalar]

Measurement duration in seconds.

SLM.Lavg

type: [struct] size: [scalar]

Struct container for averaged sound level data versus time. Unit is dB re 20 µPa[xxxvi].

SLM.Lavg.F

type: [struct] size: [scalar]

Struct container for fast-weighted sound level data versus time. According to IEC 61672-1:2013. Time-constant is 125 ms, and decay rate is approximately 34.7436 dB/s. Data is sampled every 125 ms. Maximum- and minimum values within that time block is stored in *SLM.Lavg.max.F.[wt]* and *SLM.Lavg.min.F.[wt]* respectively.

Timesteps are found in *SLM.Lavg.time.F*. Available if *tamaraConfig.globals.SLM.timeWeighting = "Slow+Fast"*. RMS of weighted data between time stamps in SLM.Lavg.time.F (0 should be implicitly added).

SLM.Lavg.F.[wt]

type: [single] size: [numtimestepsfast x numchs]

Matrix containing fast-weighted sound level data versus time for weighting *wt*. Available weightings: A, C and Z, according to IEC 61672-1:2013. See *SLM.Lavg.F*.

xxxvi Unless tamaraConfig.globals.engineeringUnits.pressure.val have been modified.

SLM.Lavg.histogram

type: [struct] size: [scalar]

Struct container for sound level histogram data. All levels are according to *SLM.Lavg*. *SLM.Lavg.F* is used if available, otherwise *SLM.Lavg.S*—refer to *SLM.Lavg.histogram.label*.

SLM.Lavg.histogram.label

type: [char cell] size: [numlines x 1]

Description of variable SLM.histogram.percentile.

SLM.Lavg.histogram.max.[wt]

type: [struct] size: [scalar]

Struct container for maximum level histogram data for weighting *wt*. See *SLM.Lavg.F.[wt]*, *SLM.Lavg* and *SLM.Lavg.max*.

SLM.Lavg.histogram.max.[wt].N

type: [uint32] size: [numedges x numchs]

Matrix containing number of occurances for each bin and channel, for weighting *wt*.

SLM.Lavg.histogram.max.[wt].P

type: [single] size: [numedges x numchs]

Matrix containing probability for each bin and channel, for weighting *wt*. Unit is %. Sum for each channel is 100 %.

SLM.Lavg.histogram.max.[wt].xCenter

type: [single] size: [numedges x 1]

Vector containing the histogram bin center values for weighting *wt*. See *SLM.Lavg.histogram.max.[wt].xEdges*.

SLM.Lavg.histogram.max.[wt].xEdges

type: [single] size: [numedges x 2]

Matrix containing the histogram bin edges for weighting *wt*. The step-size is defined in *tamaraConfig.globals.SLM.histogramResolution*. The number of edges is dependant on signal.

SLM.Lavg.histogram.min.[wt]

type: [struct] size: [scalar]

Struct container for maximum level histogram data for weighting *wt*. See *SLM.Lavg.F.[wt]*, *SLM.Lavg* and *SLM.Lavg.min*.

SLM.Lavg.histogram.min.[wt].N

type: [uint32] size: [numedges x numchs]

Matrix containing number of occurances for each bin and channel, for weighting *wt*.

SLM.Lavg.histogram.min.[wt].P

type: [single] size: [numedges x numchs]

Matrix containing probability for each bin and channel, for weighting *wt*. Unit is %. Sum for each channel is 100 %.

SLM.Lavg.histogram.min.[wt].xCenter

type: [single] size: [numedges x 1]

Vector containing the histogram bin center values for weighting *wt*. See *SLM.Lavg.histogram.min.[wt].xEdges*.

SLM.Lavg.histogram.min.[wt].xEdges

type: [single] size: [numedges x 2]

Matrix containing the histogram bin edges for weighting *wt*. The step-size is defined in *tamaraConfig.globals.SLM.histogramResolution*. The number of edges is dependant on signal.

SLM.Lavg.histogram.[wt]

type: [struct] size: [scalar]

Struct container for rms level histogram data for weighting *wt*. See *SLM.Lavg.F.[wt]* for weight info.

SLM.Lavg.histogram.[wt].N

type: [uint32] size: [numedges x numchs]

Matrix containing number of occurances for each bin and channel, for weighting *wt*.

SLM.Lavg.histogram.[wt].P

type: [single] size: [numedges x numchs]

Matrix containing probability for each bin and channel, for weighting *wt*. Unit is %. Sum for each channel is 100 %.

SLM.Lavg.histogram.[wt].xCenter

type: [single] size: [numedges x 1]

Vector containing the histogram bin center values for weighting *wt*. See *SLM.Lavg.histogram.[wt].xEdges*.

SLM.Lavg.histogram.[wt].xEdges

type: [single] size: [numedges x 2]

Matrix containing the histogram bin edges for weighting *wt*. The step-size is defined in *tamaraConfig.globals.SLM.histogramResolution*. The number of edges is dependant on signal.

SLM.Lavg.max

type: [struct] size: [scalar]

Struct container for maximum sound level data versus time.

SLM.Lavg.max.F

type: [struct] size: [scalar]

Struct container for maximum sound level data versus time for fast-weighted data. See *SLM.Lavg.F*.

SLM.Lavg.max.F.[wt]

type: [single] size: [numtimestepsfast x numchs]

Matrix containing maximum sound level data versus time for fast-weighted data. See *SLM.Lavg.F* and *SLM.Lavg.F.[wt]*. Between the time stamps in SLM.Lavg.time.F (0 should be implicitly added) the maximum value for the weighted sound level is recorded and stored here.

SLM.Lavg.max.S

type: [struct] size: [scalar]

Struct container for maximum sound level data versus time for slow-weighted data. See *SLM.Lavg.S*.

SLM.Lavg.max.S.[wt]

type: [single] size: [numtimestepsfast x numchs]

Matrix containing maximum sound level data versus time for slow-weighted data. See *SLM.Lavg.S, SLM.Lavg.F.[wt]*, *SLM.Lavg.time.S* and *SLM.Lavg.max.F.[wt]*.

SLM.Lavg.min.F

type: [struct] size: [scalar]

Struct container for minimum sound level data versus time for fast-weighted data. See *SLM.Lavg.F*.

SLM.Lavg.min.F.[wt]

type: [single] size: [numtimestepsfast x numchs]

Matrix containing minimum sound level data versus time for fast-weighted data. See *SLM.Lavg.F*, *SLM.Lavg.F.[wt]* and *SLM.Lavg.max.F.[wt]*.

SLM.Lavg.min.S

type: [struct] size: [scalar]

Struct container for minimum sound level data versus time for slow-weighted data. See *SLM.Lavg.S*.

SLM.Lavg.min.S.[wt]

type: [single] size: [numtimestepsfast x numchs]

Matrix containing minimum sound level data versus time for slow-weighted data. See *SLM.Lavg.F*, *SLM.Lavg.F.[wt]* and *SLM.Lavg.min.F.[wt]*.

SLM.Lavg.percentile

type: [struct] size: [scalar]

Struct container for sound level percentile data. All levels are according to *SLM.Lavg. SLM.Lavg.F* is used if available, otherwise *SLM.Lavg.S*—refer to *SLM.Lavg.percentile. label*. Percentiles on block RMS averaged data.

SLM.Lavg.percentile.label

type: [char cell] size: [numlines x 1]

Description of variable SLM.Lavg.percentile.

SLM.Lavg.percentile.max

type: [struct] size: [scalar]

Struct container for sound level percentile data. See *SLM.Lavg.percentile*. Percentiles on block maximum data.

SLM.Lavg.percentile.max.[wt]

type: [struct] size: [scalar]

Same as *SLM.Lavg.percentile.[wt]*, but on block max. data, instead of RMS.

SLM.Lavg.percentile.min

type: [struct] size: [scalar]

Struct container for sound level percentile data. See *SLM.Lavg.percentile*. Percentiles on block minimum data.

SLM.Lavg.percentile.min.[wt]

type: [struct] size: [scalar]

Same as *SLM.Lavg.percentile.[wt]*, but on block min. data, instead of RMS.

SLM.Lavg.percentile.[wt]

type: [struct] size: [scalar]

Struct container for sound level percentile data for weighting *wt*. See *SLM.Lavg.F.[wt]* for information about weightings.

SLM.Lavg.percentile.[wt].array

type: [struct] size: [scalar]

Struct container for full level percentile data for weighting *wt*.

SLM.Lavg.percentile.[wt].array.L

type: [single] size: [399 x numchs]

Matrix containing levels for percentiles in *SLM.Lavg.percentile.[wt].array.P* for each channel. All levels are according to *SLM.Lavg.F.[wt]* and *SLM.Lavg.F*, if fast weighting enabled[xxxvii], otherwise: *SLM.Lavg.S.[wt]* and *SLM.Lavg.S.*

SLM.Lavg.percentile.[wt].array.P

type: [single] size: [399 x 1]

Vector containing evaluated percentiles: 0.25 to 99.75 in 0.25 steps.

SLM.Lavg.percentile.[wt].P5

type: [single] size: [1 x numchs]

Vector containing 5th percentiles for each channel for weighting *wt*.

SLM.Lavg.percentile.[wt].P10

type: [single] size: [1 x numchs]

Vector containing 10th percentiles for each channel for weighting *wt*.

xxxvii tamaraConfig.globals.SLM.timeWeighting = "Slow+Fast"

SLM.Lavg.percentile.[wt].P25

type: [single] size: [1 x numchs]

Vector containing 25th percentiles for each channel for weighting *wt*.

SLM.Lavg.percentile.[wt].P50

type: [single] size: [1 x numchs]

Vector containing 50th percentiles for each channel for weighting *wt*.

SLM.Lavg.percentile.[wt].P75

type: [single] size: [1 x numchs]

Vector containing 75th percentiles for each channel for weighting *wt*.

SLM.Lavg.percentile.[wt].P90

type: [single] size: [1 x numchs]

Vector containing 90th percentiles for each channel for weighting *wt*.

SLM.Lavg.percentile.[wt].P95

type: [single] size: [1 x numchs]

Vector containing 95th percentiles for each channel for weighting *wt*.

SLM.Lavg.S

type: [struct] size: [scalar]

Struct container for slow-weighted sound level data versus time. According to IEC 61672-1:2013. Time-constant is 1 s, and decay rate is approximately 4.3429 dB/s. Data is sampled every second. Maximum- and minimum values within that time block is stored in *SLM.Lavg.max.S.[wt]* and *SLM.Lavg.min.S.[wt]* respectively. Timesteps are found in *SLM.Lavg.time.S*. RMS of weighted data between time stamps (0 should be implicitly added).

SLM.Lavg.S.[wt]

type: [single] size: [numtimestepsslow x numchs]

Struct container for slow-weighted sound level data versus time for weighting *wt*. See *SLM.Lavg.S* and *SLM.Lavg.F.[wt]*.

SLM.Lavg.time

type: [struct] size: [scalar]

Struct container for time vectors for SLM.Lavg.

SLM.LCpeak

type: [double] size: [1 x numchs]

Vector containing $L_{C_{peak}}$ values for each channel. According to IEC 61672-1:2013.

SLM.LEQ

type: [struct] size: [scalar]

Struct container for equivalent sound level data.

SLM.LEQ.narrow

type: [struct] size: [scalar]

Struct container for equivalent sound level narrow-band data.

SLM.LEQ.narrow.[wt]

type: [double] size: [numdftlines x numchs]

Matrix containing equivalent sound level narrow-band data for weight *wt*. Available weightings: Z[xxxviii], A[xxxix], B[xl], C[xli] and D[xlii]. Frequency vector is available at *f.narrow*. Unit is dB re. 20 µPa, unless *tamaraConfig.globals.engineeringUnits.pressure.val* have been changed.

SLM.LEQ.oct

type: [struct] size: [scalar]

Struct container for equivalent sound level octave-band data. Which octave-band is defined in *tamaraConfig.band*.

SLM.LEQ.oct.[wt]

type: [double] size: [numdftlines x numchs]

Matrix containing equivalent sound level octave-band data for weight *wt*. See *SLM.LEQ.narrow.[wt]*, *f.oct* and *SLM.LEQ.oct*.

SLM.LEQ.oct[B]

type: [struct] size: [scalar]

Struct container for equivalent sound level 1/B octave-band data. *B* is a number corresponding to 1/*B*-bands, e.g. SLM.LEQ.oct3 for 1/3-octave band data. Available if *tamaraConfig.globals.octaveDataStoring* is set to *all*. Available numbers B: 1, 3, 6, 12 and 24; 48 is available if *tamaraConfig.band = '1/48'*.

SLM.LEQ.oct[B].[wt]

type: [double] size: [numdftlines x numchs]

Matrix containing equivalent sound level 1/B octave-band data for weight *wt*. See *SLM.LEQ.oct.[wt]* and *SLM.LEQ[B].oct*.

SLM.LEQ.total

type: [struct] size: [scalar]

Struct container for overall equivalent sound level data.

SLM.LEQ.total.[wt]

type: [double] size: [1 x numchs]

Vector containing overall equivalent sound level data (single value) for weight *wt*. Unit is dB re. 20 µPa, unless *tamaraConfig.globals.engineeringUnits.pressure.val* have been changed. See *SLM.LEQ.narrow.[wt]*.

---

xxxviii IEC 61672-1:2013
xxxix IEC 61672-1:2013
xl IEC 60651:2001
xli IEC 61672-1:2013
xlii IEC 537:1976

SLM.LFmax

type: [struct] size: [scalar]

Struct container for maximum fast-weighted data. Available if *config.internal.storing.variablesToSave.soundlevelmeter.LFmax.enable* is *true*.

SLM.LFmax.A

type: [double] size: [1 x numchs]

Vector containing A-weighted maximum fast-weighted level (single value) for *each channel*. Unit is dB re. 20 µPa, unless *tamaraConfig.globals.engineeringUnits.pressure.val* have been changed.

SLM.LFmax.Z

type: [double] size: [1 x numchs]

Vector containing Z-weighted maximum fast-weighted level (single value) for *each channel*. Unit is dB re. 20 µPa, unless *tamaraConfig.globals.engineeringUnits.pressure.val* have been changed.

SLM.LSmax

type: [struct] size: [scalar]

Struct container for maximum slow-weighted data. Available if *config.internal.storing.variablesToSave.soundlevelmeter.LSmax.enable* is *true*.

SLM.LSmax.A

type: [double] size: [1 x numchs]

Vector containing A-weighted maximum slow-weighted level (single value) for *each channel*. Unit is dB re. 20 µPa, unless *tamaraConfig.globals.engineeringUnits.pressure.val* have been changed.

SLM.LSmax.Z

type: [double] size: [1 x numchs]

Vector containing Z-weighted maximum slow-weighted level (single value) for *each channel*. Unit is dB re. 20 µPa, unless *tamaraConfig.globals.engineeringUnits.pressure.val* have been changed.

spectogram - octave-band spectrogram data

spectrogram

type: [struct] size: [scalar]

Struct container for octave-band and narrow-band spectrograms. Available if *config.globals.spectrogram.oct.enable* not set to 0, or if config.globals.spectrogram.enable is *true*; unless *config.internal.storing.variablesToSave.(<activemeasurementtype>).spectrogram.enable* = 0.

spectrogram.label

type: [char cell] size: [numlines x 1]

Description of variable spectrogram.

spectrogram.narrow

type: [struct] size: [scalar]

Struct container for narrow-band spectrogram data.

spectrogram.narrow.dB

type: [single] size: [N x numbands x numchs]

Matrix containing spectrogram data in narrow-bands in dB. Units are same as for *spectrogram.oct.dB*.

spectrogram.narrow.f

type: [single] size: [1 x numfrequencies]

Frequency vector for narrow-band spectrogram, in Hz.

spectrogram.narrow.t

type: [single] size: [1 x N]

Time vector for narrow-band spectrogram, in seconds.

spectrogram.oct

type: [struct] size: [scalar]

Struct container for octave-band spectrogram data. Which octave-band is defined in *tamaraConfig.band*. Not available if *tamaraConfig.band = narrow.*

spectrogram.oct.dB

type: [single] size: [N x numbands x numchs]

Matrix containing spectrogram data in octave-bands in dB. Unit is dB re. units, e.g. dB re 1 Pa. Gxx(k) = 10*log10(conj(S(k)) .* S(k)) (in dB). NB: always re. 1.

spectrogram.oct.f

type: [struct] size: [scalar]

Struct container for frequency data for octave spectrogram for *oct* band. See *spectrogram.oct*.

spectrogram.oct.f.display

type: [single] size:[1 x numbands*3]

Same as *spectrogram.f.oct*.exact but in preferred numbers for center-frequencies, in accordance with ISO 3:1973, ISO 266:1997 and ASA S1.6-1960 (where applicable).

spectrogram.oct.f.exact

type: [single] size:[1 x numbands*3]

Vector containing frequencies for octave band *oct*[xliii] for octave spectrogram. Order of values: 1st lower bound, 1st centre frequency, 1st upper bound, 2nd lower bound, etc.

spectrogram.oct.t

type: [single] size: [1 x N]

Time vector for octave spectrogram, in seconds. NB: first edge of time blocks, not center values. Time resolution = *T*, where $T = 1/\Delta f$.

xliii See *spectrogram.oct*.

spectrogram.oct[B]

type: [struct] size: [scalar]

Struct container for octave-band spectrogram data. *B* is a number corresponding to 1/*B*-bands, e.g. spectrogram.oct3 for 1/3-octave band data. Available if *tamaraConfig.globals.octaveDataStoring* is set to *all*. Available numbers B: 1, 3, 6, 12 and 24; 48 is available if *tamaraConfig.band = '1/48'*.

spectrogram.oct[B].dB

type: [single] size: [N x numbands x numchs]

Matrix containing spectrogram data in 1/B octave-bands in dB. Units are same as for *spectrogram.oct.dB*.

spectrogram.oct[B].f

type: [struct] size: [scalar]

Struct container for frequency data for octave spectrogram for 1/B band. See *spectrogram.oct.f*.

spectrogram.oct[B].f.display

type: [single] size: [1 x numbands*3]

See *spectrogram.oct[B]* and spectrogram.f.oct.display. B = 48 does not have standardised values but are approximated[xliv] using rules from IEC 61260-1:2014.

spectrogram.oct[B].f.exact

type: [single] size: [1 x numbands*3]

See *spectrogram.oct[B]* and spectrogram.f.oct.exact.

spectrogram.oct[B].t

type: [single] size: [1 x N]

Time vector for 1/B octave spectrogram, in seconds. See *spectrogram.oct.t*.

SQ - sound quality metrics

SQ

type: [struct] size: [scalar]

Struct container for sound quality metrics. Available if any SQ metrics enabled, unless *tamaraConfig.internal.storing.variablesToSave.(<activemeasurementtype>).SQ = 0*.

SQ.AI

type: [struct] size: [scalar]

Struct container for Articulation Index data.

SQ.AI.y

type: [double] size: [1 x numchs]

Articulation index per channel ranging from 0 to 1.

xliv Deviation is approximately ±1.27 %

SQ.stationaryloudness

type: [struct] size: [scalar]

Container for all stationary loudness metrics.

SQ.stationaryloudness.label

type: [char] size: [1 x numchars]

Method used for loudness calculation, e.g. 'ISO 532-1' .

SQ.stationaryloudness.sone

type: [double] size: [1 x numchs]

Stationary loudness for each channel in sone.

SQ.stationaryloudness.spec

type: [double] size: [1 x numchs]

Stationary specific loudness for each channel in sone/bark.

SQ.stationaryloudness.field

type: [char] size: [1 x numchars]

Soundfield of measurement, default is 'free'.

SQ.toneToNoiseRatio

type: [struct] size: [scalar]

Struct container for all tone-to-noise ratio metrics (in development).

SQ.toneToNoiseRatio.label

type: [char] size: [1 x numchars]

Method used for TNR calculation.

SQ.toneToNoiseRatio.x

type: [double] size: [numTones x numchs x numchs]

M => numTone,N => TNR in dB, pages => frequency of tone in Hz,

SQ.toneToNoiseRatio.y

type: [double] size: [numdftlines x numchs]

Single sided comparison spectrum, consisting only of detected TNR and 0 dB background levels.

SQ.timevaryingloudness

type: [struct] size: [scalara]

Struct container for timevarying loudness metrics.

SQ.timevaryingloudness.label

type: [char] size: [1 x numchars]

Method used for timevarying loudness, default is 'Time-varying loudness according to ISO 532-1:2017'.

SQ.timevaryingloudness.x

type: [double] size: [numtimestepstimevaryingloudness x 1]

Time vector for timevarying loudness at 500 samples per second, in accordance to ISO 532-1:2017.

SQ.timevaryingloudness.y

type: [double] size: [numtimestepstimevaryingloudness x numchs]

Timevarying loudness in sones for each channel for each timestep.

SQ.timevaryingloudness.field

type: [char] size: [1 x numchars]

Sound field type, e.g. 'free'.

sweep - sweep measurement data

sweep

type: [struct] size: [scalar]

Struct container for sweep data.

sweep.AGxx

type: [struct] size: [scalar]

Struct container for auto-spectrum from sweep data.

sweep.AGxx.dB

type: [double] size: [numtones x numharmonics+1 x numchs]

Matrix of auto-spectrum for each tone, harmonic and channel, *harmoniccount = 1* is the fundamental. Unit is dB re 1.

sweep.AGxx.lin

type: [double] size: [numtones x numharmonics+1 x numchs]

Matrix of auto-spectrum for each tone, harmonic and channel. Unit is linear, see *AGxx.narrow*. See *sweep.AGxx.dB*.

sweep.config

type: [struct] size: [scalar]

Struct container for sweep settings.

sweep.config.frequencyStart

type: [double] size: [scalar]

First/start frequency, in Hz, for sweep.

sweep.config.frequencySteps

type: [double] size: [scalar]

Number of sweep frequencies.

sweep.config.frequencyStop

type: [double] size: [scalar]

Last/end frequency, in Hz, for sweep.

sweep.config.noHarmonics

type: [double] size: [scalar]

Number of harmonics to consider, fundamental excluded.

sweep.config.noiseBandwidth

type: [double] size: [scalar]

Not yet supported.

sweep.config.numberOfSweeps

type: [double] size: [scalar]

Number of sweeps.

sweep.config.rampType

type: [char] size: [1 x numchars]

Sweep type, 'logarithmic' or 'linear'.

sweep.config.singleStepDuration

type: [double] size: [scalara

Duration in seconds for each step.

sweep.config.singleSweepDuration

type: [double] size: [scalar]

Duration in seconds for a sweep.

sweep.config.type

type: [char] size: [1 x numchars]

Struct container for sweep settings.

sweep.f

type: [double] size: [1 x numtones]

Sweep frequencies.

sweep.HD

type: [struct] size: [scalar]

Struct container for harmonic distortion from sweep data.

sweep.HD.percent

type: [double] size: [numtones x numharmonics x numchs]

Matrix of distortion for each tone[xlv], harmonic and channel, *harmoniccount = 1* is the 1st harmonic. Unit is %.

xlv See sweep.f.

sweep.HD.total

type: [double] size: [numtones x numchs]

Matrix of total distortion for each tone and channel, unit is 1.

sweep.HSxy

type: [struct] size: [scalar]

Struct container for FRF from sweep data.

sweep.HSxy.dB

type: [double] size: [numtones x numchs]

Matrix of FRF for each tone and channel, referenced to the reference channel. Unit is dB re 1.

sweep.HSxy.lin

type: [double] size: [numtones x numchs]

Matrix of FRF for each tone and channel, referenced to the reference channel. Unit is linear.

sweep.SNR

type: [struct] size: [scalar]

Struct container for signal-to-noise-ratio from sweep data.

sweep.SNR.Awt

type: [struct] size: [scalar]

Struct container for A-weighted[xlvi] signal-to-noise-ratio from sweep data.

sweep.SNR.Awt.dB

type: [double] size: [numtones x numchs]

Matrix of A-weighted SNR for each tone and channel. Unit is dB. 20*log10(1/(1-sqrt(signal$_A$^2)/sqrt(total energy$_A$))).

sweep.SNR.Awt.lin

type: [double] size: [numtones x numchs]

Matrix of A-weighted SNR for each tone and channel. Unit is linear. See *sweep.SNR.Awt.dB*.

sweep.SNR.dB

type: [double] size: [numtones x numchs]

Matrix of SNR for each tone and channel. Unit is dB. 20*log10(1/(1-sqrt(signal^2)/sqrt(total energy))).

sweep.SNR.lin

type: [double] size: [numtones x numchs]

Matrix of SNR for each tone and channel. Unit is linear. See *sweep.SNR.dB*.

xlvi IEC 61672-1:2013

tacho - tachometer data

tacho_<tachocount>

type: [struct] size: [scalar]

Struct container for tachometer speed signal. E.g. named *tacho_1* and *tacho_2*.

tacho_<tachocount>.dir

type: [char] size: [1 x numchars]

See *Info.tacho(tachocount).dir*.

tacho_<tachocount>.label

type: [char] size: [1 x numchars]

See *Info.tacho(tachocount).label*.

tacho_<tachocount>.max

type: [char] size: [1 x numchars]

Maximum speed recorded.

tacho_<tachocount>.min

type: [single] size: [scalar]

Minimum speed recorded.

tacho_<tachocount>.n

type: [int64] size: [scalar]

Number of tacho pulse events.

tacho_<tachocount>.pulses_per_rev

type: [double] size: [scalar]

See *Info.tacho(tachocount).pulses_per_rev*.

tacho_<tachocount>.T

type: [single] size: [n x 1]

Array of derived speed at pulse events..

tacho_<tachocount>.unit

type: [char] size: [1 x numchars]

See *Info.tacho(tachocount).unit*.

tacho_<tachocount>.x

type: [double] size: [n x 1]

Time stamp array of pulse events.

timeSignal - measurement timesignal

timeSignal

type: [double/single] size: [numsamples x numchs]

Matrix containing the timesignal data. Data type according to *config.globals.timeSignal.precision.mat*. Units are according to Info.chUnits.

tamaraConfig - configuration/settings for tamara (stored variant)

tamaraConfig

type: [struct] size: [scalar]

Struct containing all settings from a measurement, not used in an active session. See *config* (page 87) for details on it specification, and *Different versions* on page 87 for an overview.

tamaraConfig.eventTime

type: [struct] size: [scalar]

Struct containing all information for each processed block. Only available in *tamaraConfig*.

tamaraConfig.eventTime.time

type: [double] size: [1 x N_2]

Serial time[xlvii] vector for each processed block.

tamaraConfig.eventTime.status

type: [logical] size: [1 x N_2]

Vector with trigger status for each processed block.

tamaraConfig.eventTime.overload

type: [logical] size: [1 x N_2]

Vector with overload status for each processed block, *true* = overload.

xlvii Number of days since January 0, 0000 in the proleptic ISO calendar.

Tamara Settings

Different versions

The Tamara settings variable come in different version, depending on context:

- *config*, is used in an active session, e.g. in the REPL or when defining settings in *runOnce.txt* and *runAlways.txt*.
- *tamaraConfig*, is a copy[xlviii] of the *config* used in a measurement and is stored in a measurement file[xlix]. Unless noted otherwise, see documentation for *config*.
- There are other versions but only used internally, and undocumented.

config specification

config

type: [struct] size: [scalar]

Struct container for all settings.

config.band

type: [char] size: [1 x numchars]

Default octave-band, 'narrow' if no octave-band selected.

config.chs

type: [struct] size: [1 x numinputchs]

Struct container for input channel settings.

config.chs(chcount)

type: [struct] size: [scalar]

Struct container for input channel settings for channel *chcount*.

config.chs(chcount).calibration

type: [struct] size: [1 x numcalibrations]

Struct container for calibration data. Empty if no calibration has been stored.

config.chs(chcount).channelorder

type: [uint16] size: [scalar]

Channel number within its module/device.

config.chs(chcount).chassisModel

type: [char] size: [1 x numchars]

Model name of device chassis. Empty string if device does not belong to a chassis[l].

xlviii GUI objects and handles are excluded.

xlix Unless disabled by *config.internal.storing.variablesToSave.(<activemeasurementtype>).config.enable = false*.

l PXI devices are housed in a chassis but does not report itself as belonging to a chassis.

config.chs(chcount).chassisName

type: [char] size: [1 x numchars]

Unique name[li] of device chassis. See *config.chs(chcount).chassisModel*.

config.chs(chcount).chassisNumber

type: [uint32] size: [scalar]

Unique number of PXI chassis, 1 to 255.

config.chs(chcount).chassisSerial

type: [char] size: [1 x numchars]

Serial number of device chassis. See *config.chs(chcount).chassisModel*.

config.chs(chcount).chassisSlotNumber

type: [uint16] size: [scalar]

Chassis slot number of module. Empty if device does not belong to a chassis, see *config.chs(chcount).chassisModel*.

config.chs(chcount).correction

type: [struct] size: [scalar]

Struct container for correction settings.

config.chs(chcount).correction.integrateFromAcc

type: [logical] size: [scalar]

Integrate to sensor unit channel from acceleration--e.g. when using an accelerometer as a velocity sensor. Default is *false*.

config.chs(chcount).correction.module

type: [struct] size: [scalar]

Struct container for module correction settings. Empty *double* if not present.

config.chs(chcount).correction.sensor

type: [struct/double] size: [scalar]

Struct container for sensor correction settings. Empty *double* if not present.

config.chs(chcount).correction.useModuleCorrection

type: [logical] size: [scalar]

Use module correction, see *config.chs(chcount).correction.module*. Default is *false*. Hidden option.

config.chs(chcount).coupling

type: [char] size: [1 x numchars]

Coupling type of channel, e.g. 'AC', 'DC' or 'IEPE 20 mA'.

li E.g. an unique inventory name.

config.chs(chcount).couplingsAvailable

type: [char cell] size: [1 x numcouplingsavailable]

Char cell of all available couplings available for channel *chcount*. See *config.chs(chcount).coupling*.

config.chs(chcount).daqDescription

type: [char] size: [1 x numchars]

Full name of input device.

config.chs(chcount).destination

type: [uint16] size: [1 x 2]

Internal indexes for device mapping.

config.chs(chcount).enable

type: [int8] size: [scalar]

State of channel, 0 = disable, 1 = enabled, 2 = enabled and reference channel.

config.chs(chcount).ID

type: [char] size: [1 x numchars]

Physical name of input channel, e.g. 'ai3'.

config.chs(chcount).internal

type: [struct] size: [scalar]

Struct container for internal input channel settings.

config.chs(chcount).internal.playRec

type: [struct] size: [scalar]

Struct container for channel *chcount*, if *config.chs(chcount).isPlayRecDevice = true*.

config.chs(chcount).internal.selfCalibrationSupported

type: [logical] size: [scalar]

Indicator for whether self-calibration is supported by device (true or false).

config.chs(chcount).invertPolarity

type: [logical] size: [scalar]

Invert polarity of timesignal, *true* or *false*. Default is *false*.

config.chs(chcount).isCalibrated

type: [logical] size: [scalar]

Flag for whether channel have been calibrated in current session.

config.chs(chcount).isHidden

type: [logical] size: [scalar]

Hidden state for channel *chcount*. If hidden and enabled it will be powered on and added to session as usual but its data will be ignored.

config.chs(chcount).isPlayRecDevice

type: [logical] size: [scalar]

Whether device is using the Playrec interface or not.

config.chs(chcount).isSimulated

type: [logical] size: [scalar]

Whether device is a virtual device or not.

config.chs(chcount).isTacho

type: [logical] size: [scalar]

Whether channel is of Tacho-type (*true*) or not (*false*). Default is *false*. Only valid for a subset of sensor types.

config.chs(chcount).model

type: [char] size: [1 x numchars]

Model name of input module.

config.chs(chcount).module

type: [char] size: [1 x numchars]

Inventory name of input module.

config.chs(chcount).moduleorder

type: [uint16] size: [scalar]

Module count for this module, e.g. 3:rd module as listed.

config.chs(chcount).name

type: [char] size: [1 x numchars]

Custom name for input channel *chcount*.

config.chs(chcount).physicalChannel

type: [char] size: [1 x numchars]

Full physical channel name of channel *chcount*, as used for instance by NI Daqmx, including unique module name.

config.chs(chcount).ranges

type: [struct] size: [scalar]

Struct container for available ranges for channel *chcount*.

config.chs(chcount).ranges.string

type: [char cell] size: [numranges x 1]

Cell containing the string for each available range for channel *chcount*.

config.chs(chcount).ranges.value

type: [double] size: [numranges x 2]

Matrix containing the minimum—first column—and maximum value—second column—for each available range for channel *chcount*.

config.chs(chcount).ranges.unit

type: [char cell] size: [numranges x 1]

Cell containing the unit string for each available range for channel *chcount*.

config.chs(chcount).rangevalue

type: [uint8] size: [scalar]

Index of selected input range count for channel *chcount*.

config.chs(chcount).secondaryReference

type: [logical] size: [scalar]

Indicates if channel is secondary H4 estimator reference. Default: *false*. Only one channel can have a *true* value. See *config.globals.FRF_estimator*.

config.chs(chcount).sensitivityunit

type: [char] size: [1 x numchars]

Sensitivity unit, e.g. 'mV/Pa'. Using friendly unit name[lii].

config.chs(chcount).sensitivityvalue

type: [double] size: [scalar]

Sensitivity of channel *chcount*, in SI-units, e.g. 50 mV/Pa -> 0.05 V/Pa.

config.chs(chcount).sensor

type: [char] size: [1 x numchars]

Name of selected sensor type, e.g. 'Microphone'.

config.chs(chcount).sensorvalue

type: [uint16] size: [scalar]

Index of selected sensor for channel *chcount*, e.g. 2.

config.chs(chcount).serial

type: [char] size: [1 x numchars]

Serial number of input module.

config.chs(chcount).slot

type: [uint32] size: [1 x numchars]

Slot number of module. If not applicable or unknown: 0.

config.chs(chcount).tacho

type: [struct] size: [scalar]

Struct container for tachometer settings for channel *chcount*.

config.chs(chcount).teds

type: [struct] size: [scalar]

Struct container for TEDS[liii] information for channel *chcount*.

lii NB: not always in SI.

liii Transducer Electronic Data Sheet, IEEE 1451.

config.chs(chcount).terminalConfig

type: [char] size: [1 x numchars]

Selected input configuration type for channel *chcount*, e.g. 'Differential'.

config.chs(chcount).terminalConfigAvailable

type: [char cell] size: [numterminalconfigs x 1]

Cell containing the string for each available input configurations for channel *chcount*.

config.chs(chcount).terminals

type: [char cell] size: [numterminals x 1]

Cell containing the string for each available terminal for input module.

config.chs(chcount).vendor

type: [char] size: [1 x numchars]

Internal name of manufacturer/device driver provider, e.g. 'ni'.

config.chs_out

type: [struct] size: [1 x numoutputchs]

Struct container for output channel settings.

config.chs_out(chcount).amplitude

type: [double] size: [scalar] or [1 x numtones]

Peak amplitude of signal for channel *chcount*, or peak amplitude for each tone in *config.chs_out(chcount).frequency*.

config.chs_out(chcount).channelorder

type: [uint16] size: [scalar]

Channel number within its module/device.

config.chs_out(chcount).chassisModel

type: [char] size: [1 x numchars]

Model name of device chassis. Empty string if device does not belong to a chassis[liv].

config.chs_out(chcount).chassisName

type: [char] size: [1 x numchars]

Unique name[lv] of device chassis. See *config.chs_out(chcount).chassisModel*.

config.chs_out(chcount).chassisSerial

type: [char] size: [1 x numchars]

Serial number of device chassis. See *config.chs_out(chcount).chassisModel*.

config.chs_out(chcount).chassisSlotNumber

type: [uint16] size: [scalar]

Chassis slot number of module. Empty if device does not belong to a chassis, see *config.chs_out(chcount).chassisModel*.

liv PXI devices are housed in a chassis but does not report itself as belonging to a chassis.

lv E.g. an unique inventory name.

config.chs_out(chcount).daqDescription

type: [char] size: [1 x numchars]

Full name of output device.

config.chs_out(chcount).destination

type: [uint16] size: [1 x 2]

Internal indexes for device mapping.

config.chs_out(chcount).dutycycle

type: [double real] or [double complex] size: [scalar] or [1 x 2]

Dutycycle for PWM signal, range: 0 – 1 (0 will output zeros, and 1 will output a DC-signal at defined peak amplitude).

If scalar:

PWM dutycycle is stationary and defined by this value.

If not scalar:

real(*dutycycle*(1)) = start dutycycle (1);

real(*dutycycle*(2)) = end dutycycle (1);

imag(*dutycycle*(1)) = sweep duration (s), for one cycle, if imag(*dutycycle*(2)) is negative a logarithmic- or linear stepped sweep is selected by using a negative imag(*dutycycle*(1)) (log) or positive (lin);

imag(*dutycycle*(2)) = sweep type:

0: *up-down-up* (linear);
1: *up* (linear);
2: *up-down-up* (logarithmic);
3: *up* (logarithmic);
<0: If sweep type is negative, instead the sweep is performed in fixed steps (rounded if not an integer), with number of steps = |imag(*dutycycle*(2))|, using the *up* sweep type.

All sweeps are repeating.

Examples:

[0.25, 0.75]+1i*[30, 0] → a sweep from 25 % to 75 % during 30 s using sweep type 0 (up-down-up: up to 75% down to 25% and up to 75%...). Sweep type 1 would reset after reaching the end and start over from the start value.

[0.1, 0.9]+1i*[45, -25]→ a stepped sweep from 10 % to 90 % during 45 s in 25 steps using *up* sweep type, linear.

config.chs_out(chcount).enable

type: [int8] size: [scalar]

State of channel, 0 = disable, 1 = enabled, -1 = monitor channel for Explorer.

config.chs_out(chcount).enableFilter

type: [logical] size: [scalar]

Enable (*true*) or disable (*false*) generator output filter[lvi].

config.chs_out(chcount).fileChannelToUse

type: [int16] size: [scalar/empty]

Channel number to use when using a *file* output type. A negative value indicate that automatic channel selection is used, e.g. -16 implies that channel 16 is used, but will change if needed, e.g. when loading a new file etc.

config.chs_out(chcount).frequency

type: [double real] or [double complex] size: [1 x numtones] or [1 x 2]

Frequency of signal for channel *chcount*, for types: Sinusoidal, Sine sweep and PWM, in Hz.

Sinusoidal type: If number of elements is more than one: multiple tones will be added to the signal, one for each element. The real part is the frequency and the imaginary part is the phase offset (in radians). The amplitude is either: 1/numtones * config.chs_out(chcount).amplitude, when amplitude is a scalar, or *config.chs_out(chcount).amplitude(tonecount)* for each tone *tonecount*, when *.amplitude* size matches *frequency* size. Maximum of number of tones: 100.

PWM/Sine sweep type: if number of elements is one, frequency is stationary and defined by this value, if not:

real(frequency(1)) = start frequency (Hz);

real(frequency(2)) = end frequency (Hz);

imag(frequency(1)) = sweep duration (s), if imag(frequency(2)) is negative a logarithmic- or linear stepped sweep is selected by using a negative imag(frequency(1)) (log) or positive (lin);

imag(frequency(2)) = sweep type:

0: *up-down-up* (linear);
1: *up* (linear);
2: *up-down-up* (logarithmic);
3: *up* (logarithmic);
<0: If sweep type is negative, instead the sweep is performed in fixed steps (rounded if not an integer), with number of steps = |imag(frequency(2))|, using the *up* sweep type.

All sweeps are repeating.

Examples:

[100, 6000]+1i*[200, 2]→ a sweep from 100 Hz to 6 kHz to 100 Hz during 200 s using sweep type 2 (*up-down-up*), logarithmic.

[1000, 3000]+1i*[-40, -10]→ a stepped sweep from 1 kHz to 3 kHz during 40 s in 10 steps using *up* sweep type, logarithmic.

lvi Defined in *config.globals.output*.

config.chs_out(chcount).ID

type: [char] size: [1 x numchars]

Physical name of output channel, e.g. 'ao2'.

config.chs_out(chcount).internal

type: [struct] size: [scalar]

Struct container for internal output channel settings.

config.chs_out(chcount).internal.playRec

type: [struct] size: [scalar]

Struct container for output channel *chcount*, if *config.chs_out(chcount).isPlayRecDevice = true*.

config.chs_out(chcount).internal.selfCalibrationSupported

type: [logical] size: [scalar]

Indicator for whether self-calibration is supported by device (true or false).

config.chs_out(chcount).isPlayRecDevice

type: [logical] size: [scalar]

Whether device is using the Playrec interface or not.

config.chs_out(chcount).isSimulated

type: [logical] size: [scalar]

Whether device is a virtual device or not.

config.chs_out(chcount).model

type: [char] size: [1 x numchars]

Model name of output module.

config.chs_out(chcount).module

type: [char] size: [1 x numchars]

Inventory name of output module.

config.chs_out(chcount).moduleorder

type: [uint16] size: [scalar]

Module count for this module, e.g. 4:th module as listed.

config.chs_out(chcount).physicalChannel

type: [char] size: [1 x numchars]

Full physical channel name of channel *chcount*, as used for instance by NI Daqmx, including unique module name.

config.chs_out(chcount).ranges

type: [struct] size: [scalar]

Struct container for available ranges for output channel *chcount*.

config.chs_out(chcount).ranges.string

type: [char cell] size: [numranges x 1]

Cell containing the string for each available range for channel *chcount*.

config.chs_out(chcount).ranges.value

type: [double] size: [numranges x 2]

Matrix containing the minimum—first column—and maximum value—second column—for each available range for channel *chcount*.

config.chs_out(chcount).ranges.unit

type: [char cell] size: [numranges x 1]

Cell containing the unit string for each available range for channel *chcount*.

config.chs_out(chcount).rangevalue

type: [uint8] size: [scalar]

Index of selected output range count for channel *chcount*.

config.chs_out(chcount).sensor

type: [char] size: [1 x numchars]

Name of output sensor type, e.g. 'Voltage'.

config.chs_out(chcount).sensorvalue

type: [uint8] size: [scalar]

Index of output sensor for channel *chcount*, e.g. 1.

config.chs_out(chcount).serial

type: [char] size: [1 x numchars]

Serial number of output module.

config.chs_out(chcount).terminalConfig

type: [char] size: [1 x numchars]

Selected output configuration type for channel *chcount*, e.g. 'Differential'.

config.chs_out(chcount).terminalConfigAvailable

type: [char cell] size: [numterminalconfigs x 1]

Cell containing the string for each available output configurations for channel *chcount*.

config.chs_out(chcount).terminals

type: [char cell] size: [numterminals x 1]

Cell containing the string for each available terminal for output module.

config.chs_out(chcount).type

type: [char] size: [1 x numchars]

Signal type for output channel *chcount*, e.g. 'Pink noise'.

config.chs_out(chcount).vendor

type: [char] size: [1 x numchars]

Internal name of manufacturer/device driver provider, e.g. 'ni'.

config.DFToverlap

type: [double] size: [scalar]

DFT overlap, range: 0 – 1.

config.fs

type: [double] size: [scalar]

Sampling rate in Hz.

config.globals

type: [struct] size: [scalar]

Struct container for settings such as Level recorder settings, DFT windows etc[lvii].

config.globals.acq

type: [struct] size: [scalar]

Struct container for some acquisition settings.

config.globals.acq.store

type: [struct] size: [scalar]

Struct container for some acquistion storing settings.

config.globals.acq.store.automatic

type: [logical] size: [logical]

Store measurement automatically when measurement finished. Default is *false*,

config.globals.acq.warmupTime

type: [double] size: [scalar]

Amount of time (in seconds) to wait additionally when using one stage start (*config.globals.useTwoStageStart*) before starting the acquisition. Default is 0 (s). Use when additional settling time, etc, is needed.

config.globals.autoArmMeasurement

type: [logical] size: [scalar]

Automatically arm measurement when measurement saved, *true* or *false*. Only applicable if *config.globals.useTwoStageStart* is 1 or 2.

config.globals.calcGxxMaxHold

type: [logical] size: [scalar]

Calculate the max hold of the instantenous auto-spectra. Default is *false*.

config.globals.calibration

type: [struct] size: [scalar]

Struct container for last used calibration settings. See details for each channel in *config.chs(chcount).calibration*.

lvii The distinctions between *config.internal* and *config.globals* are not always coherent and are in most cases an effect of backwards compatibility.

config.globals.calibration.amplitude

type: [struct] size: [scalar]

RMS amplitude of calibration signal, default: 1.

config.globals.calibration.calibratorType

type: [char] size: [1 x numchars]

Type of calibrator in use.

config.globals.calibration.f

type: [double] size: [scalar]

Calibration frequency, default: 1000 Hz.

config.globals.calibration.tol

type: [double] size: [scalar]

Acceptable relative standard deviation for calibration in dB. Default: – 65 dB.

config.globals.corrGroups

type: [struct] size: [scalar]

Struct for correlated output groups.

config.globals.corrGroups.nActive

type: [uint16] size: [scalar]

Number of grouped output channels with a correlated output.

config.globals.corrGroups.groups{n}

type: [cell] size: [1 x numcorrgroups]

Cell structure for correlated output channel group *n*.

config.globals.currentPath

type: [char] size: [1 x numchars]

Current Tamara installation path.

config.globals.disablePlots

type: [logical] size: [scalar]

Disable (*true*) or enable (*false*) plotting in acquisition mode.

config.globals.discardMeasWarning

type: [logical] size: [scalar]

Disable (*true*) or enable (*false*) the warning when starting a measurement or closing Tamara when previous measurement wasn't saved.

config.globals.discardMonitorWarning

type: [logical] size: [scalar]

Disable warning about a necessary Tamara restart if screen settings change.

config.globals.engineeringUnits

type: [struct] size: [scalar]

Struct container for settings for engineering units.

config.globals.engineeringUnits.acceleration

type: [struct] size: [scalar]

Struct container for acceleration unit settings.

config.globals.engineeringUnits.acceleration.exponent

type: [double] size: [scalar]

Exponent, default is 2.

config.globals.engineeringUnits.acceleration.symbol

type: [char] size: [1 x numchars]

Symbol, default is 'a'.

config.globals.engineeringUnits.acceleration.unit

type: [char] size: [1 x numchars]

Symbol, default is 'm/s^2'.

config.globals.engineeringUnits.acceleration.val

type: [double] size: [scalar]

dB reference, default is 1.

config.globals.engineeringUnits.current

type: [struct] size: [scalar]

Struct container for current unit settings.

config.globals.engineeringUnits.current.exponent

type: [double] size: [scalar]

Exponent, default is 2.

config.globals.engineeringUnits.current.symbol

type: [char] size: [1 x numchars]

Symbol, default is 'I'.

config.globals.engineeringUnits.current.unit

type: [char] size: [1 x numchars]

Symbol, default is 'A'.

config.globals.engineeringUnits.current.val

type: [double] size: [scalar]

dB reference, default is 1.

config.globals.engineeringUnits.displacement

type: [struct] size: [scalar]

Struct container for displacement unit settings.

config.globals.engineeringUnits.displacement.exponent

type: [double] size: [scalar]

Exponent, default is 2.

config.globals.engineeringUnits.displacement.symbol

type: [char] size: [1 x numchars]

Symbol, default is 'd'.

config.globals.engineeringUnits.displacement.unit

type: [char] size: [1 x numchars]

Symbol, default is 'm'.

config.globals.engineeringUnits.displacement.val

type: [double] size: [scalar]

dB reference, default is 1.

config.globals.engineeringUnits.force

type: [struct] size: [scalar]

Struct container for force unit settings.

config.globals.engineeringUnits.force.exponent

type: [double] size: [scalar]

Exponent, default is 2.

config.globals.engineeringUnits.force.symbol

type: [char] size: [1 x numchars]

Symbol, default is 'F'.

config.globals.engineeringUnits.force.unit

type: [char] size: [1 x numchars]

Symbol, default is 'N'.

config.globals.engineeringUnits.force.val

type: [double] size: [scalar]

dB reference, default is 1.

config.globals.engineeringUnits.power

type: [struct] size: [scalar]

Struct container for power unit settings.

config.globals.engineeringUnits.power.exponent

type: [double] size: [scalar]

Exponent, default is 1.

config.globals.engineeringUnits.power.symbol

type: [char] size: [1 x numchars]

Symbol, default is 'P'.

config.globals.engineeringUnits.power.unit

type: [char] size: [1 x numchars]

Symbol, default is 'W'.

config.globals.engineeringUnits.power.val
type: [double] size: [scalar]
dB reference, default is 1e-12.

config.globals.engineeringUnits.pressure
type: [struct] size: [scalar]
Struct container for pressure unit settings.

config.globals.engineeringUnits.pressure.exponent
type: [double] size: [scalar]
Exponent, default is 2.

config.globals.engineeringUnits.pressure.symbol
type: [char] size: [1 x numchars]
Symbol, default is 'p'.

config.globals.engineeringUnits.pressure.unit
type: [char] size: [1 x numchars]
Symbol, default is 'Pa'.

config.globals.engineeringUnits.pressure.val
type: [double] size: [scalar]
dB reference, default is 2e-5.

config.globals.engineeringUnits.velocity
type: [struct] size: [scalar]
Struct container for velocity unit settings.

config.globals.engineeringUnits.velocity.exponent
type: [double] size: [scalar]
Exponent, default is 2.

config.globals.engineeringUnits.velocity.symbol
type: [char] size: [1 x numchars]
Symbol, default is 'u'.

config.globals.engineeringUnits.velocity.unit
type: [char] size: [1 x numchars]
Symbol, default is 'm/s'.

config.globals.engineeringUnits.velocity.val
type: [double] size: [scalar]
dB reference, default is 1.

config.globals.engineeringUnits.voltage
type: [struct] size: [scalar]
Struct container for voltage unit settings.

config.globals.engineeringUnits.voltage.exponent

type: [double] size: [scalar]

Exponent, default is 2.

config.globals.engineeringUnits.voltage.symbol

type: [char] size: [1 x numchars]

Symbol, default is 'U'.

config.globals.engineeringUnits.voltage.unit

type: [char] size: [1 x numchars]

Symbol, default is 'V'.

config.globals.engineeringUnits.voltage.val

type: [double] size: [scalar]

dB reference, default is 1.

config.globals.expAverage

type: [logical] size: [scalar]

Use exponential averaging, *true* or *false*. See *config.globals.expAverageTime* and *config.globals.avg_slide*.

config.globals.expAverageTime

type: [double] size: [scalar]

DFT averaging time in seconds. See *config.globals.expAverage* and *config.globals.avg_slide*.

config.globals.expAverageUnit

type: [char] size: [1 x numchars]

Averaging unit to show for exponential averaging. Unit is *blocks* or *seconds*. Determined by measurement mode.

config.globals.FRF_estimator

type: [char] size: [1 x 2]

Possible values: 'H1', 'H2', 'H3' and 'H4'.

$H_1(k, l) = ADxy(k, l) / ADxx(k)$

$H_2(k, l) = ADxx(l) / ADxy(l, k)$

$H_3(k, l) = 0.5*H_1(k, l) + 0.5* H_2(k, l)$

$H_4(k, l) = ADxy(ref, l) / ADxy(ref, k)$ | ref = secondary reference channel[lviii]

config.globals.freqlimit

type: [char] size: [1 x numchars]

Allows to set the frequency range of interest in the format '[2/T FS/2.5]'. Frequency range can only be correctly applied as long as the sampling rate allows it.

lviii See *config.chs(chcount).secondaryReference*.

config.globals.FRFwithoutReference

type: [logical] size: [scalar]

Calculate FRF related data, when no reference selected. Default is *false*.

config.globals.input

type: [struct] size: [scalar]

Struct container for input channel options.

config.globals.input.highpass

type: [struct] size: [scalar]

Struct container for input channel high-pass filter settings.

config.globals.input.highpass.enable

type: [logical] size: [scalar]

Enable or disable Butterworth high-pass filter for all input channels, affects both Acquisition and Explorer quick analysis. Default is *false*.

config.globals.input.highpass.f

type: [double] size: [scalar]

Cut-off (-3 dB) frequency, in Hz, for high-pass filter, see *config.globals.input.highpass.order*. Default is 1 Hz, lower limit is 0.01 Hz and upper limit is (*sampling rate*/2.5) Hz.

config.globals.input.highpass.order

type: [double] size: [scalar]

Butterworth order of high-pass filter. Default is 4. Acceptable values: 1:6.

config.globals.input.lowpass

type: [struct] size: [scalar]

Struct container for input channel low-pass filter settings.

config.globals.input.lowpass.enable

type: [logical] size: [scalar]

Enable or disable Butterworth high-pass filter for all input channels, affects both Acquisition and Explorer quick analysis. Default is *false*.

config.globals.input.lowpass.f

type: [double] size: [scalar]

Cut-off (-3 dB) frequency, in Hz, for low-pass filter, see *config.globals.input.lowpass.order*. Default is 10 kHz, lower limit is 1 Hz and upper limit is (*sampling rate*/2.4) Hz.

config.globals.input.lowpass.order

type: [double] size: [scalar]

Butterworth order of low-pass filter. Default is 4. Acceptable values: 1:6.

config.globals.input.useIntegrationSuperSampling

type: [logical] size: [scalar]

Use supersampling for integration filters. Default is *true*. Factor is 16 for sampling rates below 3201 Hz, 8 for rates below 6401 Hz and 4 otherwise.

config.globals.ISO9614_1

type: [struct] size: [scalar]

Struct container for ISO 9614-1 settings.

config.globals.ISO9614_1.area

type: [double] size: [1 x numsides / scalar]

Area (m^2) of each measurement segment, per side. Default is [0.1 0.1 0.1 0.1 0.1]. If scalar and multiple points defined, all segments will use this value.

config.globals.ISO9614_1.automatic

type: [logical] size: [scalar]

Enable or disable automatic measurement mode, default is *true*.

config.globals.ISO9614_1.freq_lim

type: [double] size: [1 x 2]

Lower- and upper frequency limit (Hz), does not affect sampling rate, but intensity outside of this range will be set to zero and will not be considered by the warning function. Default is [65 5000].

config.globals.ISO9614_1.segments

type: [double] size: [1 x numsides / scalar]

Number of measurement points, per side. Default is [20 20 20 20 20]. If scalar and multiple points segment areas, all segments will use this value.

config.globals.ISO9614_1.unsigned

type: [logical] size: [scalar]

Enable (*false*) or disable (*true*) intensity separation, default is *false*.

config.globals.ISO9614_1.voice

type: [logical] size: [scalar]

Enable (*true*) or disable (*false*) vocal feedback for automatic measurements, default is *true*.

config.globals.ISO9614_1.warning

type: [logical] size: [scalar]

Enable/disable warning for negative intensity, when using automatic measurement. Default is *true*.

config.globals.LR

type: [struct] size: [scalar]

Struct container for Level recorder settings.

config.globals.LR.blockDC

type: [logical] size: [scalar]

Enable (true) or disable (false) 16 Hz[lix] high-pass filter, 4-th order Butterworth.

lix – 3 dB

config.globals.LR.customFilterBank

type: [struct] size: [scalar]

Struct container for settings for the custom filter bank. Only used if *config.globals.LR.useOctFilters* = -1.

config.globals.LR.customFilterBank.bandwidthValue

type: [uint8] size: [scalar]

Filter bandwidth:

1 = 1/1 (BW = 70.7%)

2 = 1/3 (BW = 23.2%)

3 = 1/6 (BW = 11.6%)

4 = 1/12 (BW = 5.8%)

5 = 1/24 (BW = 2.9%)

6 = 1/48 (BW = 1.4%)

Default is 3 (11.6 %).

config.globals.LR.customFilterBank.f

type: [double] size: [1 x numfilters]

Vector of centre frequencies for custom filter bank. Default is [100 400 1000].

config.globals.LR.customFilterBank.string

type: [char] size: [1 x numchars]

User input string describing *config.globals.LR.customFilterBank.f*, e.g. '100:100:1000'. Default is '[100 400 1000] '.

config.globals.LR.customWritingSpeedTime

type: [double] size: [scalar]

Writing speed time, in seconds, to use when custom writing speed is enabled[lx].

config.globals.LR.filter

type: [struct] size: [scalar]

Contains structs of all available weightings. Allows enabling weightings e.g. config.globals.LR.filter.A.enable = true;

config.globals.LR.filter.(filtername)

type: [struct] size: [scalar]

Struct container for filter <filtername> . Available filters:

Z: linear filter, i.e. unweighted, unless *config.globals.LR.blockDC* is true, in which case a high-pass filter is applied.

A: A-weigthing filter according to IEC 61672-1:2013.

C: C-weigthing filter according to IEC 61672-1:2013.

Wc, Wd, We, Wh, Wj, Wk and Wm: According to ISO 2631-1:1997.

lx config.globals.LR.useCustomWritingSpeed = true.

config.globals.LR.histogramResolution

type: [double] size: [scalar]

Resolution of histogram, in dB. Default is 1 dB.

config.globals.LR.integrateAcceleration

type: [struct] size: [scalar]

Struct container for acceleration integration settings.

config.globals.LR.integrateAcceleration.enable

type: [logical] size: [scalar]

Integrate acceleration signals, true or false (default).

config.globals.LR.integrateAcceleration.target

type: [char] size: [1 x numchars]

String for the selected acceleration integration target, '*velocity*' or '*displacement*' (default).

config.globals.LR.interpolate

type: [logical] size: [scalar]

Enable (*true*) or disable (*false*) linear interpolation when reducing data according to *config.globals.LR.paperSpeed*. If *false*, nearest neighbour is used. Default is *false*.

config.globals.LR.loggingRate

type: [double] size: [scalar]

Number of data points per seconds to store, for the level data. Default value is -1, implying automatic mode, depending on *config.globals.LR.writingSpeedTime*—it is set such that any maximum deviation is ≤ 0.4 dB (e.g. when the signal becomes zero right at one of the logged points) and the nearest primenumber above desired value (to avoid aliasing for some common tones). For a *fast* time weighting filter automatic mode is 87 points/s, and for *slow* time weighting: 11 points/s.

config.globals.LR.useCustomWritingSpeed

type: [logical] size: [scalar]

Enable (*true*) or disable (*false*) the custom writing speed time, see *config.globals.LR.customWritingSpeedTime*. Default is *false*.

config.globals.LR.useOctFilters

type: [int8] size: [scalar]

Enable (1) or disable (0) the octave filterbank. Requires that *config.band* is not 'narrow'. Default is 0. If *useOctFilters* = – 1, the *customFilterBank*[lxi] is used instead.

config.globals.LR.writingSpeed

type: [double] size: [scalar]

Writing speed of level filters, defined as decay in dB/s. Default is 34.7 dB/s (*fast* filter in accordance with IEC 61672-1:2013).

lxi See config.globals.LR.customFilterBank.

config.globals.LR.writingSpeedString

type: [char] size: [1 x numchars]

String for the selected writing speed/exponential averaging time. E.g. '125 ms' and 'Custom'.

config.globals.LR.writingSpeedTime

type: [double] size: [scalar]

Writing speed of level filters, defined as time constant in seconds. Default 0.125 s (fast filter in accordance with IEC 61672-1:2013).

config.globals.LR.writingSpeedTimeUnit

type: [char] size: [1 x numchars]

Unit of config.globals.LR.writingSpeedTime: 's'.

config.globals.LR.writingSpeedUnit

type: [char] size: [1 x numchars]

Unit of config.globals.LR.writingSpeed: 'dB/s'.

config.globals.LR.writingSpeedVal

type: [uint8] size: [scalar]

Index of selected writing speed option in GUI. Default: 4.

config.globals.mech

type: [struct] size: [scalar]

Struct container for settings related to *Mechanical mobility* and *Impact*.

config.globals.mech.autoRejectDoubleHit

type: [logical] size: [scalar]

Enable or disable double hit rejection.

config.globals.mech.autoSelectionBlocks

type: [logical] size: [scalar]

Enable or disable automatic rejection of block based on coherence optimisation.

config.globals.mech.channelToOptimise

type: [uint16] size: [scalar]

When *config.globals.mech.autoSelectionBlocks* is enabled, which channel to use. Default: 1.

config.globals.mech.doubleHit

type: [struct] size: [scalar]

Struct container for double hit rejection settings.

config.globals.mech.doubleHit.minPeakHeight

type: [double] size: [scalar]

Minimum relative peak height needed to trigger a double-hit.

config.globals.mech.doubleHit.minPeakProminence

type: [double] size: [scalar]

Minimum second peak prominence needed to trigger a double-hit.

config.globals.mech.doubleHit.T

type: [double] size: [scalar]

Time constant, in seconds, used for exponential smoothing for double hit detection.

config.globals.mech.doubleHit.useVocalFeedback

type: [uint8] size: [scalar]

Use audio feedback (text-to-speech) for rejection only (2), also for hits (1) or none (0).

config.globals.mech.minBlocks

type: [uint16] size: [scalar]

Not used/undocumented.

config.globals.mech.selectionMethod

type: [char] size: [1 x numchars]

Method used for *config.globals.mech.autoSelectionBlocks*.

config.globals.note

type: [char] size: [1 x numchars]

String for measurement note. Empty when no measurement note defined..

config.globals.octaveBase

type: [uint8] size: [scalar]

Use base-ten (10) or base-two (2) for octave-bands. Default is 10. Note that some sound quality standards will require, and use, base-two regardless of this setting.

config.globals.octaveDataStoring

type: [char] size: [1 x numchars]

Stores all ocatve bands with 'all' and only the selected ones with 'selected'. Default is 'selected'.

config.globals.octaveStandard

type: [char] size: [1 x numchars]

The octave band standard to use: IEC 61260-1:2014 or IEC 61672-1:2013 (default). IEC 61672-1:2013 is noted as 'IEC', and IEC 61260-1:2014 is noted as 'ASA', due to it compliance with ASA S1.11-2004.

config.globals.output

type: [struct] size: [scalar]

Struct container for some generator settings.

config.globals.output.filterCutOff

type: [double] size: [scalar, 1 x 2]

Cut-off frequency(ies), in Hz, for the selected generator filter-type, *config.globals.output.filterType*. Two values for the *band-pass* and *band-stop* types, and scalar for the others.

config.globals.output.fs

type: [double] size: [scalar]

Desired sampling rate for generator hardware.

config.globals.output.filterType

type: [char] size: [1 x numchars]

Kind of generator filter-type:

high-pass (default)

low-pass

band-pass

band-stop

Band-pass and band-stop filters use 8th-order Butterworth filters, while high- and low-pass filters use a 4th order Butterworth filter. See *config.globals.output.filterCutOff*.

config.globals.periodicAutosave

type: [uint8] size: [scalar]

Save measurements periodically without stopping or starting. Default is 0.

config.globals.REV

type: [struct] size: [scalar]

Struct container for reverberation time measurement settings

config.globals.REV.method

type: [char] size: [1 x numchars]

Method to use in reverberation time measurement. Only available option: 'backward integration'.

config.globals.skip_nonreference_FRFS

type: [logical] size: [scalar]

Skip calculating the cross-channel FRF data. If *true* only reference to receiver FRFs will be calculated. Default is *true*.

config.globals.SLM

type: [struct] size: [scalar]

Struct container for sound level meter settings.

config.globals.SLM.bargraph

type: [logical] size: [scalar]

Plot levels using bars, when only one channel is enabled, *true* or *false*.

config.globals.SLM.calcLCpeak

type: [logical] size: [scalar]

Enable or disable calculation of $L_{C_{peak}}$ level.

config.globals.SLM.enableTimeErase

type: [logical] size: [scalar]

Enable (*true*) or disable (*false*) the 15 second time erase function. Default is *false*.

config.globals.SLM.highpassMode

type: [uint8] size: [scalar]

Index indication highpass filter mode:

1 = no highpass filtering, i.e. the 0 Hz mode.

2 = highpass filter at 6 Hz, mimics a Brüel & Kjær 2260 SLM, (default).

3 = highpass filter at 16 Hz.

config.globals.SLM.histogramResolution

type: [double] size: [scalar]

Resolution of histogram in dB, default is 0.5 dB.

config.globals.SLM.timeWeighting

type: [char] size: [1 x numchars]

String specifying whether only slow weighting is used ('Slow') or both fast and slow weighting ('Slow+Fast').

config.globals.SLM.useOctFilters

type: [logical] size: [scalar]

Enable (*true*) or disable (*false*) the octave filterbank. Requires that *config.band* is not 'narrow'. Default is *false*.

config.globals.spectrogram

type: [struct] size: [scalar]

Struct container for spectrogram settings.

config.globals.spectrogram.narrow

type: [struct] size: [scalar]

Struct container for narrow-band spectrogram settings.

config.globals.spectrogram.narrow.enable

type: [logical] size: [scalar]

Enable or disable narrow-band spectrogram.

config.globals.spectrogram.oct

type: [struct] size: [scalar]

Struct container for octave-band spectrogram settings.

config.globals.spectrogram.oct.enable

type: [logical] size: [scalar]

Calculate and store octave spectrogram, *true* or *false* (default).

config.globals.SQ

type: [struct] size: [scalar]

Struct container for sound quality metrics settings.

config.globals.storeData

type: [struct] size: [scalar]

Struct container for data storing options. Not for timesignal storing settings.

config.globals.storeData.automatic

type: [logical] size: [scalar]

Use automatic filenames when storing data.. Default is *false*.

config.globals.storeData.format

type: [struct] size: [scalar]

Struct container for the data formats available for storing.

config.globals.storeTimeSignal

type: [logical] size: [scalar]

Enable (*true*) or disable (*false*), saving the timesignal/throughput.

config.globals.tamaraStarted

type: [double] size: [1 x 6]

Time vector of timestamp when Tamara started: [year, month, day, hour, minute, second].

config.globals.timeSignal

type: [struct] size: [scalar]

Struct container for timesignal settings.

config.globals.timeSignal.appendData

type: [struct] size: [scalar]

Struct container for timesignal data appending settings.

config.globals.timeSignal.appendData.mat

type: [logical] size: [scalar]

Append data to timesignal file for mat files, *true* (default) or *false*, or create separate files. Also used by .unv files.

config.globals.timeSignal.mergeUNV

type: [logical] size: [scalar]

Merge timesignals for all channels into a single file, *true* (default) or *false*. If *false*, data will not be appended, regardless of other settings.

config.globals.timeSignal.precision

type: [struct] size: [scalar]

Struct container for timesignal storing precision settings.

config.globals.timeSignal.precision.mat

type: [char] size: [1 x numchars]

Precision used when storing timesignal as *mat*. Available options: *single* (default) and *double*.

config.globals.timeSignal.precision.wav

type: [char] size: [1 x numchars]

Precision used when storing timesignal as *wav*. Available options: *int16*, *int24*, *single* (default) and *double*. Calibrated units are only available for floating point data.

config.globals.timeSignal.splitFile

type: [struct] size: [scalar]

Struct container for timesignal splitting settings.

config.globals.timeSignal.splitFile.fileSize

type: [double] size: [scalar]

Split file into files of size *fileSize* (MB), default is 1024 MB. Used if *config.globals.timeSignal.splitFile.enable* = 1.

config.globals.timeSignal.splitFile.time

type: [double] size: [scalar]

Split file into files with a certain duration (in minutes), default is 60 min. Used if *config.globals.timeSignal.splitFile.enable* = 2.

config.globals.timeSignal.splitFile.enable

type: [uint8] size: [scalar]

Split file into files according to file size (1), time (2), or don't split files (0). Default is 0.

config.globals.timeSignal.store

type: [struct] size: [scalar]

Struct container for timesignal storing settings.

config.globals.timeSignal.store.flac

type: [logical] size: [scalar]

Store timesignal as a *flac* file, *true* or *false*.

config.globals.timeSignal.store.mat

type: [logical] size: [scalar]

Store timesignal as a *mat* file, *true* or *false*.

config.globals.timeSignal.store.unv

type: [logical] size: [scalar]

Store timesignal as a universal file (unv-58), *true* or *false*.

config.globals.timeSignal.store.wav

type: [logical] size: [scalar]

Store timesignal as a *wave* file, *true* or *false*.

config.globals.ULfactorOverride

type: [logical] size: [scalar]

Override default upper-limit factor[lxii], *true* or *false*. See *config.globals.ULfactor*.

config.globals.useReferenceUnitsForFRF

type: [logical] size: [scalar]

Apply reference units to FRF derived data when showing dB-values. Default: *false*.

config.globals.useTwoStageStart

type: [uint8] size: [scalar]

Start measurement in two stages:

.useTwoStageStart = 0, start in one step

.useTwoStageStart = 1, start in two step, second stage is using a hardware start (hardware is not active before second start)

.useTwoStageStart = 2, start in two step, second stage is using a software start (hardware is fully active)

config.globals.window

type: [cell] size: [1 x numwindows]

Cell array of enabled DFT windows.

config.globals.window{num}.evenCorrection

type: [char] size: [1 x numchars]

The selected even-length correction for window number *num*. Default value: 'L+1 right'. Available options: 'L+1 Right', 'L+1 Left', 'L-1 Right', 'L-1 Left', 'Skewed Right', 'Skewed Left' and 'None'. Case-insensitive.

config.globals.window{num}.corrFactor

type: [char] size: [1 x numchars]

The correction factor for window number *num*. Available options: 'Energy' (default) and 'Amplitude'. NB: must be coupled to *config.globals.window{num}.corrFactorVal*. The correction factor for *Energy* = √(sum(window^2/windowlength), the correction factor for *Amplitude* = sum(window / windowlength)

config.globals.window{num}.corrFactorVal

type: [double] size: [scalar]

The correction factor index for window number *num*. Available options: 1 ('Energy') (default) and 2 ('Amplitude'). See *config.globals.window{num}.corrFactor*.

lxii Also known as anti-aliasing factor

config.globals.window{num}.type

type: [char] size: [1 x numchars]

The window type for window number *num*. E.g. 'Hanning'.

config.globals.window{num}.windowConfig

type: [struct] size: [scalar]

Struct container for window configuration options for window number *num*.

config.globals.window{num}.windowConfig.chebyshev

type: [struct] size: [scalar]

Struct container for window configuration options for window number *num* for a window of Chebyshev type.

config.globals.window{num}.windowConfig.chebyshev.sidelobeAtt

type: [double] size: [scalar]

Sidelobe attenuation in dB for a Chebyshev window. Default values is 120 (dB).

config.globals.window{num}.windowConfig.chebyshev.sidelobeAttUnit

type: [char] size: [1 x numchars]

'dB'. Not used, only for information purposes.

config.globals.window{num}.windowConfig.custom

type: [struct] size: [scalar]

Struct container for window configuration options for window number *num* for a window of custom type.

config.globals.window{num}.windowConfig.custom.script

type: [char] size: [1 x numchars]

The evaluated formulat for the custom window type. Important variables: *x* (valued 0–1). 'L+1 Right' even-length correction will be applied.

Default script: '0.5-0.5*cos(2*pi*x)', will provide a Hanning window.

config.globals.window{num}.windowConfig.dpss

type: [struct] size: [scalar]

Struct container for window configuration options for window number *num* for a window of DPSS type.

config.globals.window{num}.windowConfig.dpss.alpha

type: [double] size: [scalar]

Alpha used in the DPSS sequence. Default: 1.3143

config.globals.window{num}.windowConfig.dpss.alphaUnit

type: [char] size: [1]

'1', not used.

config.globals.window{num}.windowConfig.dpss.seq

type: [double] size: [scalar]

DPSS sequence number. Default: 1

config.globals.window{num}.windowConfig.exponential

type: [struct] size: [scalar]

Struct container for window configuration options for window number *num* for a window of exponential type.

config.globals.window{num}.windowConfig.exponential.peakOffset

type: [double] size: [scalar]

Offset in peak, from the left, in milliseconds. Default: 0.

config.globals.window{num}.windowConfig.exponential.peakOffsetUnit

type: [char] size: [1 x 2]

'ms', not used.

config.globals.window{num}.windowConfig.exponential.timeConstant

type: [double] size: [scalar]

Timeconstant of exponential window. Default: 100 (ms)

config.globals.window{num}.windowConfig.exponential.timeConstantUnit

type: [char] size: [1 x 2]

'ms', not used.

config.globals.window{num}.windowConfig.forceExponential

type: [struct] size: [scalar]

Struct container for window configuration options for window number *num* for a window of force/exponential type.

config.globals.window{num}.windowConfig.forceExponential.leadingTime

type: [double] size: [scalar]

Initial offset of force window. Window edges are tapered with the error function (*erf*). Default is 0 (ms).

config.globals.window{num}.windowConfig.forceExponential.leadingTimeUnit

type: [char] size: [1 x 2]

'ms', not used.

config.globals.window{num}.windowConfig.forceExponential.forceDuration

type: [double] size: [scalar]

Duration of force window, default: 200 (ms)

config.globals.window{num}.windowConfig.forceExponential.forceDurationUnit

type: [char] size: [1 x 2]

'ms', not used.

config.globals.window{num}.windowConfig.forceExponential.responseDuration

type: [double] size: [scalar]

Initial offset for exponential response window. Setting it to zero will generate a standard exponential window. Any time before the response duration the window amplitude will be set to one. Unit is ms. Default value: 40 (ms).

config.globals.window{num}.windowConfig.forceExponential.responseDurationUnit

type: [char] size: [1 x 2]

'ms', not used.

config.globals.window{num}.windowConfig.forceExponential.timeConstant

type: [double] size: [scalar]

Timeconstant of exponential window. Default: 250 (ms)

config.globals.window{num}.windowConfig.forceExponential.timeConstantUnit

type: [char] size: [1 x 2]

'ms', not used.

config.globals.window{num}.windowConfig.kaiser

type: [struct] size: [scalar]

Struct container for window configuration options for window number *num* for a window of Kaiser type.

config.globals.window{num}.windowConfig.kaiser.alpha

type: [double] size: [scalar]

Alpha used in the Kaiser window. Default: 2

config.globals.window{num}.windowConfig.kaiser.alphaUnit

type: [char] size: [1]

'1', not used.

config.globals.window{num}.windowConfig.transient

type: [struct] size: [scalar]

Struct container for window configuration options for window number *num* for a window of transient type.

config.globals.window{num}.windowConfig.transient.leadingTime

type: [double] size: [scalar]

Initial offset time of transient window. Window edges are tapered with the error function (*erf*).

config.globals.window{num}.windowConfig.transient.leadingTimeUnit

type: [char] size: [1 x 2]

'ms', not used.

config.globals.window{num}.windowConfig.transient.responseDuration

type: [double] size: [scalar]

Duration of the box part of the transient window. Default is 100 (ms).

config.globals.window{num}.windowConfig.transient.responseDurationUnit

type: [char] size: [1 x 2]

'ms', not used.

config.globals.windowSettings

type: [struct] size: [scalar]

Struct container for windows settings.

config.globals.windowSettings.activeWindow

type: [uint16] size: [scalar]

Index of active DFT window.

config.globals.windowSettings.default

type: [struct] size: [scalar]

Struct container for window default settings. Undocumented.

config.globals.windowSettings.internal

type: [struct] size: [scalar]

Struct container for window plotting settings. Undocumented.

config.globals.windowSettings.nActive

type: [uint16] size: [scalar]

Number of enabled DFT windows.

config.globals.windowsVersion

type: [char] size: [1 x numchars]

String containing version and build number of Windows.

config.GUI

type: [struct] size: [scalar]

Struct container for GUI settings. Undocumented.

config.internal

type: [struct] size: [scalar]

Struct container for settings of more Tamara internal nature. See *config.globals*.

config.internal.autoRemoveVirtualDevices

type: [logical] size: [scalar]

Automatically remove NI virtual hardware if non-virtual hardware connected, *true* (default) or *false*.

config.internal.autoStartGenerator

type: [logical] size: [scalar]

Automatically start the output generator (if any channel enabled) when starting an acquisition, *true* or *false*. If disabled the generator must be manually started and stopped.

config.internal.CAN

type: [struct] size: [scalar]

Struct container for CAN settings.

config.internal.correction

type: [struct] size: [scalar]

Struct container for correction related settings.

config.internal.correction.nPoints

type: [double] size: [scalar]

Number of points used for visualising correction filters. Default: 2560.

config.internal.default

type: [struct] size: [scalar]

Struct container for channel defaults.

config.internal.default.sensorType

type: [char] size: [1 x numchars]

Default sensor type for input channels, default is 'voltage'. Any available options in the input channel GUI can be applied, e.g. 'microphone'. Case insensitive.

config.internal.default.inputCoupling

type: [char] size: [1 x numchars]

Default coupling type for input channels, default is 'AC'. Any available options in the input channel GUI can be applied, e.g. 'IEPE'. Case insensitive.

config.internal.default.inputConfig

type: [char] size: [1 x numchars]

Default configuration type for input channels, default is 'PseudoDifferential'. Any available options in the input channel GUI can be applied, e.g. 'Differential'. Case insensitive.

config.internal.default.outputConfig

type: [char] size: [1 x numchars]

Default configuration type for output channels, default is 'PseudoDifferential'. Any available options in the output channel GUI can be applied, e.g. 'Differential'. Case insensitive.

config.internal.definedPath

type: [char] size: [1 x numchars]

Current project path.

config.internal.deployed

type: [struct] size: [scalar]

Struct container for parameters for the deployed Tamara application.

config.internal.deployed.tamaraBackupPath

type: [char] size: [1 x numchars]

Path to Tamara backup location.

config.internal.deployed.tamaraUserPath

type: [char] size: [1 x numchars]

Path to user profile data.

config.internal.deployed.tamaraUserBackupPath

type: [char] size: [1 x numchars]

Path to backup of user template files.

config.internal.deployed.tamaraVarPath

type: [char] size: [1 x numchars]

Path to Tamara variables folder location.

config.internal.disableCalcTimeCheck

type: [logical] size: [scalar]

Disable (true) or enable (false) the calculation time check when measuring. Default is false.

config.internal.excludeDeviceList

type: [cell char] size: [numexcludeddevices x 1]

List of all excluded devices.

config.internal.excludeDevicePath

type: [char] size: [1 x numchars]

Path to text-file with all excluded devices.

config.internal.explorer

type: [struct] size: [scalar]

Struct container for Explorer object and related settings.

config.internal.explorer

type: [struct] size: [scalar]

Struct container for Explorer settings.

config.internal.explorer.autoLoadMeasurements

type: [logical] size: [scalar]

Automatically load measurements when stored into Explorer. Default: *true*.

config.internal.explorer.calc

type: [struct] size: [scalar]

Struct container for Explorer Quick Analysis settings.

config.internal.explorer.calc.store

type: [struct] size: [scalar]

Struct container for Explorer Quick Analysis storing settings.

config.internal.explorer.calc.store.psd

type: [logical] size: [scalar]

Calculate and store PSD when performing a Quick Analysis in Explorer. Default: *false*.

config.internal.explorer.calc.store.timesignal

type: [logical] size: [scalar]

Include original timesignal when performing a Quick Analysis in Explorer. Default: *false*.

config.internal.explorer.calc.store.includeOtherData

type: [logical] size: [scalar]

Include non-viewable original data when performing a Quick Analysis in Explorer. Default: *true*.

config.internal.filenaming

type: [struct] size: [scalar]

Struct container for the filenaming settings.

config.internal.filenaming.automatic

type: [struct] size: [scalar]

Struct container for the automatic filenaming settings.

config.internal.filenaming.automatic.appendTimeStamp

type: [logical] size: [scalar]

Append timestamp of kind "YYYYMMDD*sep*HHMMSS", where *sep* is defined in *config.internal.filenaming.automatic.separator*.

config.internal.filenaming.automatic.counter

type: [struct] size: [scalar]

Struct container for the filenaming counter.

config.internal.filenaming.automatic.counter.end

type: [double] size: [1 x 3]

End values for the three counters, but only the first counters according to *config.internal.filenaming.automatic.noCounters* are used. Default is [inf 5 10].

config.internal.filenaming.automatic.counter.start

type: [double] size: [1 x 3]

Start values for the three counters, but only the first counters according to *config.internal.filenaming.automatic.noCounters* are used. Default is [1 1 1].

config.internal.filenaming.automatic.counter.width

type: [uint8] size: [scalar]

Minimum field-width used for counters, default is 1.

config.internal.filenaming.automatic.filePrefix

type: [struct] size: [scalar]

Struct container for the filenaming prefix.

config.internal.filenaming.automatic.filePrefix.custom

type: [char] size: [1 x numchars]

Filename prefix to use when fileprefix type is set to custom. Default is "meas".

config.internal.filenaming.automatic.filePrefix.type

type: [uint8] size: [scalar]

Use custom fileprefix (=2) or default fileprefix (=1). Default is 1.

config.internal.filenaming.automatic.noCounters

type: [uint8] size: [scalar]

Number of file counters to use, 1–3. Default is 1.

config.internal.filenaming.automatic.separator

type: [char] size: [1 x numchars]

String separator used e.g. for prefix and timestap. Default is "_".

config.internal.filenaming.automatic.storeByDate

type: [logical] size: [scalar]

Store file in subfolders of project path according to date, e.g. .\YYYY\MM_MMM\DD\.

Default is *false*.

config.internal.freqUL

type: [double] size: [scalar]

Approximate frequency upper limit for chosen settings/sampling rate.

config.internal.GPS

type: [struct] size: [scalar]

Struct container for GPS settings.

config.internal.GUI

type: [struct] size: [scalar]

Struct container for GUI objects and their related settings. Mostly undocumented.

config.internal.GUI.explorer.obj.legendType

type: [char] size: [1 x numchars]

Legend type used in Explorer. Default: 'auto'.

config.internal.GUI.explorer.obj.mapDPI

type: [double] size: [scalar]

DPI setting of maps in Explorer. Default: 140.

config.internal.GUI.explorer.obj.mapMarkerSize

type: [double] size: [scalar]

Marker size setting for maps in Explorer. Default: 140.

config.internal.GUI.explorer.obj.useOctStairs

type: [logical] size: [scalar]

Use octave stairs or straight lines for 1/N octave plots in Explorer. Default: *true*.

config.internal.hidden

type: [struct] size: [scalar]

Struct container for purely internal settings or parameters. Undocumented.

config.internal.output

type: [struct] size: [scalar]

Struct container for output related settings. See also *config.globals.output*.

config.internal.overloadMargin

type: [double] size: [scalar]

Threshold relative to input range that is considered an overload, default is 1 (exactly the input range). Not used for NI hardware with hardware detection of overloads.

config.internal.processor

type: [struct] size: [scalar]

Struct container for Acquisition related settings.

config.internal.processor.exportLegends

type: [logical] size: [scalar]

Add plot legend to exported plots, *true* (default) or *false*.

config.internal.processor.vuMeterEnabled

type: [uint8] size: [scalar]

Show channel VU meter in Acquisition. 0 = disabled, 1 = always, 2 (default) = enabled when channel count less than 100 channels.

config.internal.pxi

type: [struct] size: [scalar]

Struct container for PXI related settings.

config.internal.pxi.EnhancedAliasRejectionEnable

type: [logical] size: [scalar]

Enable (*true*) or disable (*false*) Enhanced Alias Rejection. Only applicable to certain PXI modules. Default: *true*.

config.internal.requireUserInteraction

type: [logical] size: [scalar]

Require user to provide input to certain dialogues. Needs to be disabled for autonomous operation. Mostly unused.

config.internal.sensor

type: [struct] size: [scalar]

Struct container for sensor related settings.

config.internal.sensor.checkHealth

type: [logical] size: [scalar]

Monitor IEPE sensors for short-circuits and open-loops, *true* (default) or *false*. Only applicable to supported hardware.

config.internal.sensor.automaticallyIncreaseRange

type: [logical] size: [scalar]

Automatically increase the input range after an overload during measurement, or calibration. Default is *false*.

config.internal.sensorGroups

type: [struct/double] size: [scalar/empty]

If no sensor groups defined this is an empty scalar value.

config.internal.settingBank

type: [struct] size: [scalar]

Undocumented.

config.internal.skipOverloadCheck

type: [logical] size: [scalar]

Ignore the overload check when measuring. Default: *false*.

config.internal.storing

type: [struct] size: [scalar]

Struct container for data storing settings. See *config.globals.storeData* as well.

config.internal.storing.CAN

type: [struct] size: [scalar]

Struct container for CAN storing settings.

config.internal.storing.matversion

type: [double] size: [scalar]

MAT-file version to use: 6, 7 or 7.3 (HDF5 compliant). Default is 6. Will revert to 7.3 if any variable is larger than 1.9 GB, and will revert from 6 to 7 if compression is needed.

config.internal.storing.plots

type: [struct] size: [scalar]

Struct container for plot storing settings.

config.internal.storing.plots.autoSaveActive

type: [logical] size: [scalar]

Export active plots when storing measurement. Default is *false*.

config.internal.storing.plots.backgroundColour

type: [double] size: [1 x 3]

Background colour to use when exporting plots. Default is [1, 1, 1]. Only used when *config.internal.storing.plots.overrideBackgroundColour* is *true*.

config.internal.storing.plots.fontSizeGrid

type: [uint16] size: [scalar]

Fontsize of grid for exported plots. Default is 14 pt.

config.internal.storing.plots.fontSizeLabel

type: [uint16] size: [scalar]

Fontsize of label for exported plots. Default is 17 pt.

config.internal.storing.plots.fontSizeLegend

type: [uint16] size: [scalar]

Fontsize of legend for exported plots. Default is 10 pt.

config.internal.storing.plots.fontSizeTitle

type: [uint16] size: [scalar]

Fontsize of title for exported plots. Default is 15 pt.

config.internal.storing.plots.labelAngle

type: [char] size: [1 x numchars]

Fonttype to use for labels, .e.g. *normal*, *bold* and *italic*. Default is *italic*.

config.internal.storing.plots.labelFont

type: [char] size: [1 x numchars]

Font name used for exported plots. Default is system dependant.

config.internal.storing.plots.overrideBackgroundColour

type: [logical] size: [scalar]

Use a custom background colour when exporting plot. Default is *false*. See *config.internal.storing.plots.backgroundColour*.

config.internal.storing.plots.ratio

type: [double] size: [scalar]

Height-to-width ratio of exported plots. Default is 0.6.

config.internal.storing.plots.resolution

type: [uint16] size: [scalar]

Width of exported plots. Default is 3000 pixels.

config.internal.storing.plots.saveIndividualLines

type: [logical] size: [scalar]

Export each individual line in separate plot when storing all plots. Default is *false*. Only effective when *config.internal.storing.plots.autoSaveActive* is *true*.

config.internal.storing.plots.useColourbar

type: [logical] size: [scalar]

Add a colourbar for 3D/spectrograms when exporting plots.

config.internal.storing.variablesToSave

type: [struct] size: [scalar]

Struct container for variable storing settings.

config.internal.storingFolder

type: [char] size: [1 x numchars]

Path to folder for temporary data files. Defaults to *config.internal.deployed.tamaraUserPath*.

config.internal.sync

type: [struct] size: [scalar]

Struct container for sync settings.

config.internal.sync.autoSyncPXI

type: [logical] size: [scalar]

Try to automatically sync PXI modules through the backplane. Default is *true*.

config.internal.sync.clock

type: [struct] size: [scalar]

Struct container for sync clock settings.

config.internal.sync.nActiveClock

type: [double] size: [scalar]

Number of added clock sync targets. Default is 0.

config.internal.sync.nActiveTrigger

type: [double] size: [scalar]

Number of added trigger sync targets. Default is 0.

config.internal.sync.trigger

type: [struct] size: [scalar]

Struct container for sync trigger settings.

config.internal.tellSpeed

type: [logical] size: [scalar]

Undocumented

config.internal.useADW

type: [logical] size: [scalar]

Enable or disable the Audio Device Writer output API type. Default is *false*.

config.internal.useCorrelatedNoise

type: [int8] size: [scalar]

Use correlated noise for all output channels (1), uncorrelated (0) or groupwise correlation (-1). Groups are defined in *config.globals.corrGroups*.

config.internal.useDsDaqForInputs

type: [logical] size: [scalar]

Enable or disable DirectSound devices for inputs. Default is *true*.

config.internal.useDsDaqForOutputs

type: [logical] size: [scalar]

Enable or disable DirectSound devices for outputs. Default is *true*.

config.internal.usePlayRec

type: [logical] size: [scalar]

Enable or disable the Playrec API for output channels. Default is *true*.

config.internal.usePlayRecForInputs

type: [logical] size: [scalar]

Not applicable, might be used in future releases.

config.internal.virtual

type: [struct] size: [scalar]

Struct container for virtual channel settings.

config.internal.virtual.input

type: [struct] size: [scalar]

Struct container for virtual input channel settings.

config.internal.virtual.input.enable

type: [uint8] size: [scalar]

Enable virtual input channels when no physical hardware connected (1), always (2) or never (0). Default is 1.

config.internal.virtual.input.numChannels

type: [uint16] size: [scalar]

Number of virtual input channels to use. Default is 8.

config.internal.virtual.output

type: [struct] size: [scalar]

Struct container for virtual output channel settings.

config.internal.virtual.output.enable

type: [uint8] size: [scalar]

Enable virtual output channels when no physical hardware connected (1), always (2) or never (0). Default is 1.

config.internal.virtual.output.numChannels

type: [uint16] size: [scalar]

Number of virtual output channels to use. Default is 8.

config.internal.virtual.realtimeFactor

type: [double] size: [scalar]

Time-factor used in Acquisition when using virtual channel, a value of 1 is actual time, and a value of 10 is a speed-up of factor 10. Default value is 1.

config.measurement

type: [char] size: [1 x numchars]

Active measurement type, e.g. 'Level recorder'.

config.time

type: [char] size: [1 x numchars]

Time when Tamara started, format: "YYYY mmm DD - HH:MM:SS".

config.timelimit

type: [struct] size: [scalar]

Struct container for time limit settings.

config.timelimit.enable

type: [logical] size: [scalar]

Enable time-limit for measurement.

config.timelimit.time

type: [double] size: [scalar]

Time-limit for measurement, in seconds. Only used if *config.timelimit.enable = true*.

config.trigger

type: [struct] size: [scalar]

Struct container for trigger settings.

config.trigger.autosave

type: [logical] size: [scalar]

Automatically save measurement when a stop is triggered. Default is *false*.

config.trigger.block

type: [logical] size: [scalar]

Struct container for block trigger settings.

config.trigger.block.can

type: [logical] size: [scalar]

Struct container for CAN trigger settings. See *config.trigger.(activetriggermode).can*.

config.trigger.block.gps

type: [logical] size: [scalar]

Struct container for gps trigger settings. See *config.trigger.(activetriggermode).gps*.

config.trigger.block.input

type: [logical] size: [scalar]

Struct container for input trigger settings. See *config.trigger.(activetriggermode).input*.

config.trigger.block.interval

type: [logical] size: [scalar]

Struct container for interval trigger settings. See *config.trigger.(activetriggermode).interval*.

config.trigger.block.preTriggerTime

type: [logical] size: [scalar]

How much of the timesignal to use prior to the trigger event, in seconds. Default is 0.04 s. Mostly applicable when trigger mode is set to *input* and threshold type is set to *'peak'*, since sample level trigger detection precision might not be achievable in other cases.

config.trigger.block.type

type: [char] size: [1 x numchars]

Type of trigger mode for the block trigger. Default is *'input'*.

config.trigger.session

type: [logical] size: [scalar]

Struct container for session trigger settings.

config.trigger.session.type

type: [char] size: [1 x numchars]

Type of trigger mode for the session trigger. Default is *'input'*.

config.trigger.start

type: [logical] size: [scalar]

Struct container for start trigger settings.

config.trigger.start.can

type: [logical] size: [scalar]

Struct container for CAN trigger settings. See *config.trigger.(activetriggermode).can*.

config.trigger.start.gps

type: [logical] size: [scalar]

Struct container for gps trigger settings. See *config.trigger.(activetriggermode).gps*.

config.trigger.start.input

type: [logical] size: [scalar]

Struct container for input trigger settings. See *config.trigger.(activetriggermode).input*.

config.trigger.start.time

type: [logical] size: [scalar]

Struct container for time trigger settings. See *config.trigger.(activetriggermode).time*.

config.trigger.start.type

type: [char] size: [1 x numchars]

Type of trigger mode for the start trigger. Default is *'input'*.

config.trigger.stop

type: [logical] size: [scalar]

Struct container for stop trigger settings.

config.trigger.stop.can

type: [logical] size: [scalar]

Struct container for CAN trigger settings. See *config.trigger.(activetriggermode).can*.

config.trigger.stop.gps

type: [logical] size: [scalar]

Struct container for gps trigger settings. See *config.trigger.(activetriggermode).gps*.

config.trigger.stop.input

type: [logical] size: [scalar]

Struct container for input trigger settings. See *config.trigger.(activetriggermode).input*.

config.trigger.stop.time

type: [logical] size: [scalar]

Struct container for time trigger settings. See *config.trigger.(activetriggermode).time*.

config.trigger.stop.type

type: [char] size: [1 x numchars]

Type of trigger mode for the stop trigger. Default is *'input'*.

config.trigger.saveUntriggeredData

type: [logical] size: [scalar]

Save timesignal also when measurement is awaiting the trigger. Default is *false*.

config.trigger.type

type: [char] size: [1 x numchars]

Type of active trigger mode. Default is *'none'*.

config.trigger.(activetriggermode).can

type: [struct] size: [scalar]

Struct container for can trigger settings.

config.trigger.(activetriggermode).can.ch

type: [double] size: [scalar]

Index number of selected CAN signal, as selected/seen in the CAN settings tab, and available as a structure here: *config.internal.CAN.signal*.

config.trigger.(activetriggermode).can.edge

type: [char] size: [1 x numchars]

Edge detection for trigger, available options: 'rising', 'falling' and 'none'. Default is *'none'*.

config.trigger.(activetriggermode).can.signal

type: [char] size: [1 x numchars]

Name of selected CAN signal.

config.trigger.(activetriggermode).can.threshold

type: [double] size: [scalar]

Trigger threshold. Default is 50.

config.trigger.(activetriggermode).can.thresholdType

type: [char] size: [1 x numchars]

Threshold type, available options: 'rms' and 'peak'. If 'rms' selected, a block rms will be applied and the index of the triggered sample will always be 1. Default: 'peak'.

config.trigger.(activetriggermode).can.timeConstant

type: [double] size: [scalar]

Time constant for exponential filter (in seconds), if *config.trigger.(activetriggermode).can.useFilter* is *true*. Default is 0.125 (s).

config.trigger.(activetriggermode).can.useFilter

type: [logical] size: [scalar]

Apply exponential filter to trigger CAN signal. Default is *false*. See *config.trigger.(activetriggermode).can.timeConstant*.

config.trigger.(activetriggermode).can.unit

type: [char] size: [1 x numchars]

Unit of trigger amplitude, unit is according to selected signal (from the dbc file).

config.trigger.(activetriggermode).gps

type: [struct] size: [scalar]

Struct container for gps trigger settings.

config.trigger.(activetriggermode).gps.ch

type: [char] size: [1 x numchars]

Selected GPS type for input channel trigger. Default is 'speed', and only available option.

config.trigger.(activetriggermode).gps.edge

type: [char] size: [1 x numchars]

Edge detection for trigger, available options: 'rising', 'falling' and 'none'. Default is *'none'*.

config.trigger.(activetriggermode).gps.threshold

type: [double] size: [scalar]

Trigger threshold. Default is 50.

config.trigger.(activetriggermode).gps.thresholdType

type: [char] size: [1 x numchars]

Threshold type, available options: 'rms' and 'peak'. If 'rms' selected, a block rms will be applied and the index of the triggered sample will always be 1. Default: 'peak'.

config.trigger.(activetriggermode).gps.timeConstant

type: [double] size: [scalar]

Time constant for exponential filter (in seconds), if *config.trigger.(activetriggermode).gps.useFilter* is *true*. Default is 0.125 (s).

config.trigger.(activetriggermode).gps.unit

type: [char] size: [1 x numchars]

Unit of trigger amplitude. Default: 'km/h'.

config.trigger.(activetriggermode).gps.useFilter

type: [logical] size: [scalar]

Apply exponential filter to trigger gps signal. Default is *false*. See *config.trigger.(activetriggermode).gps.timeConstant*.

config.trigger.(activetriggermode).input

type: [struct] size: [scalar]

Struct container for input trigger settings.

config.trigger.(activetriggermode).input.ch

type: [double] size: [scalar]

Selected channel for input channel trigger. Default is 1.

config.trigger.(activetriggermode).input.edge

type: [char] size: [1 x numchars]

Edge detection for trigger, available options: 'rising', 'falling' and 'none'. Default is *'none'*.

config.trigger.(activetriggermode).input.threshold

type: [double] size: [scalar]

Trigger threshold. Default is 1.

config.trigger.(activetriggermode).input.thresholdType

type: [char] size: [1 x numchars]

Threshold type, available options: 'rms' and 'peak'. If 'rms' selected, a block rms will be applied and the index of the triggered sample will always be 1. Default: 'peak'.

config.trigger.(activetriggermode).input.timeConstant

type: [double] size: [scalar]

Time constant for exponential filter (in seconds), if *config.trigger.(activetriggermode).input.useFilter* is *true*. Default is 0.125 (s).

config.trigger.(activetriggermode).input.unit

type: [char] size: [1 x numchars]

Unit of trigger amplitude.

config.trigger.(activetriggermode).input.useFilter

type: [logical] size: [scalar]

Apply exponential filter to trigger input channel signal. Default is *false*. See *config.trigger.(activetriggermode).input.timeConstant*.

config.trigger.(activetriggermode).interval

type: [struct] size: [scalar]

Struct container for interval trigger settings.

config.trigger.(activetriggermode).interval.startTime

type: [double] size: [scalar]

Initial start time threshold as a serial date number, the whole and fractional number of days starting from January 0, 0000. Default is 1 hour after the start of the Tamara application.

config.trigger.(activetriggermode).interval.startTimeFormat

type: [char] size: [1 x numchars]

The format of initial start time as represented in text. Default is 'yyyymmdd-HH:MM:SS'. Any partial parts can be used, e.g. 'MM:SS', to use a trigger of any hour at 'MM:SS'.

config.trigger.(activetriggermode).interval.startTimeStr

type: [char] size: [1 x numchars]

The initial start time as represented in text. Format is according to *config.trigger.(activetriggermode).interval.startTimeFormat*.

config.trigger.(activetriggermode).interval.threshold

type: [double] size: [scalar]

The interval time as represented in seconds. Default is 300.

config.trigger.(activetriggermode).interval.unit

type: [char] size: [scalar]

Unit of interval, which is "s".

config.trigger.(activetriggermode).interval.useStartTime

type: [logical] size: [scalar]

Use an initial start time trigger to trigger first block. Default is *false*.

config.trigger.(activetriggermode).time

type: [struct] size: [scalar]

Struct container for time trigger settings.

config.trigger.(activetriggermode).time.str

type: [char] size: [1 x numchars]

The threshold time as represented in text. Format is according to *config.trigger.(activetriggermode).time.unit*.

config.trigger.(activetriggermode).time.threshold

type: [double] size: [scalar]

Trigger threshold as a serial date number, the whole and fractional number of days starting from January 0, 0000 using the proleptic ISO 8601:2004 calendar. Default is 1 hour after the start of the Tamara application, but for the stop trigger type, it is 2 hour after application startup. For partial time thresholds, the serial date is not treated as an absolute time.

config.trigger.(activetriggermode).time.unit

type: [char] size: [1 x numchars]

The format of threshold time as represented in text. Default is 'yyyymmdd-HH:MM:SS'. Any partial parts can be used, e.g. 'MM:SS', to use a trigger every hour at 'MM:SS'.

config.t_scan

type: [double] size: [scalar]

DFT period in seconds, T.

config.version

type: [char cell] size: [1 x numversions]

Cell containing a list of Tamara versions that have used this settings file[lxiii], the last entry is the current version. Version is a string.

lxiii If the settings have been loaded, otherwise it will only contain one entry: the current version.