Meas Module Reference

This document is the reference for the Meas sub module of the Python SLab system.

Index

Introduction

The “SLab System” is a software solution that provides a low cost access to circuit measurements by the use of low cost development boards.
The complete system has a natural tendency to feature creeping. In order to make the dimension of the project easy to manage, it has been divided in several modules.
The main module, associated to the file slab.py, contains all the code that directly accesses the Hardware Board and the most immediate low complexity commands.
The functionality of the main module is extended by the use of additional modules. This document describes the measurement Meas module, associated to the file slab_meas.py that includes a set of functions to work with complex time domain measurements of circuits.

If you want to use the Meas module you will also need to use the main slab module as it is the one that includes the ***connect*** command that starts the communication with the Hardware Board. That means that you will need to use two imports.

```
>>> import slab
>>> import slab_meas as meas
```

If you need to use other additional modules you can add more lines for other sub modules. Observe that we have used an alias name for the module namespace, so, a command with the name “command” will be accessed as “meas.command”.

Time Analysis Commands

Commands in this section perform time analysis over transient measurement data.

period(vector, time, ts, mode)

Arguments	vector	Numpy array of values	V1
	time	Nympy array of equal size to vector with sample times (Optional argument)	
	ts	Sample time (Optional argument)	
	mode	Edge to use in analysis (Defaults to *slab.tmodeRise*)	
Returns		Period of a wave	

Auxiliary

The only required parameter is ***vector*** the rest of parameters are optional.
The command obtains the ***halfRange*** of the vector and determine the locations where the *halfRange* is crossed in the mode direction. Period is computed averaging the differences of the obtained locations.
Mode can be ***slab.tmodeRise*** (option by default) or ***slab.tmodeFall***.
If a time array is provided, the command use two times in this array to compute the sample time, so uniform sampling is required to obtain proper results.
If no time array is provided, and a sampling time ***ts*** is provided, period is calculated from the provided sample time.
You can use the sample time from the last measurement using the internal variable ***slab.sampleTime*** if the module has been loaded with the default **slab** name.
If no ***time*** or ***ts*** parameter is given, period is returned in number of samples.

If less than two half range crosses are found, the command generates a **"Not enough edges for period"** exception.

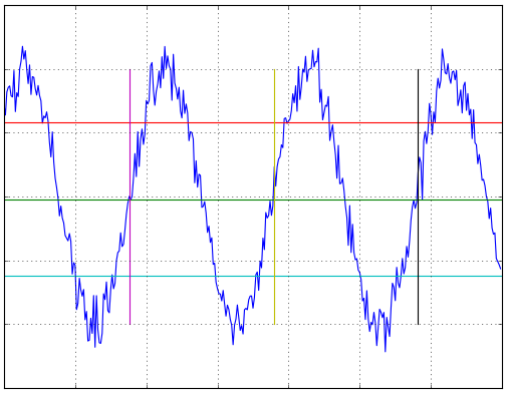
tcross(vector,value,mode,time,ts)

Arguments	vector	Numpy array of values	V1
	value	Value to cross	
	mode	Edge to use in analysis (Defaults to *slab.tmodeRise*)	
	time	Nympy array of equal size to vector with sample times (Optional argument)	
	ts	Sample time (Optional argument)	
Returns		List of cross locations	

Auxiliary

The required parameters are ***vector*** and ***value***. The rest of parameters are optional.
The command obtains a list of indexes $\mathbf{i_j}$ where the vector values cross ***value*** in the direction indicated by mode.
Direction set by mode can be ***slab.tmodeRise*** (option by default) or ***slab.tmodeFall***.
If a time array is provided, it will be used to give the time location of the cross.
If not, and a sampling time ***ts*** is provided, location will be given as $t_s * i_j$.
You can use the sample time from the last measurement using the internal variable ***slab.sampleTime*** if the module has been loaded with the default **slab** name.
If no ***time*** or ***ts*** parameter is given, the vector indices $\mathbf{i_j}$ are returned.

We will explain the operation of the command using the waveform shown below. The command first computes three values from the signal: The half range halfway between wave maximum and minimum values, the high threshold halfway between the half range and the maximum and the low treshold halfway between the half range and the minimum.



If the ***slab.tmodeRise*** mode has been selected, the command explores the wave, from left to right until its value is below the low treshold. Then, the cross time is found when the signal crosses the half range. At that point the algorith resets to found the next cross.
If the ***slab.tmodeFall*** mode has been selected, the algorithm is similar, but this time we check against the signal going over the high treshold.
In the example figure we see that we have found three crosses, indicated with purple, yellow and black vertical lines.

Global Analysis Commands

Commands in this section perform global analysis over data obtained from transient measurements.

analyze(data)

Arguments	data	Data to analyze (Default to get from transient async)	V1
Returns		Nothing	

Performs analysis on one or several signals. The signals are vectors that include a magnitude variation respect to time. The data to be analyzed can be supplied in three ways.
Data can be a list of floats o a numpy one dimension array. In that case it will represent one signal and time will be measured in samples with one sample for each data element.
Data can be list of vectors with each vector built as a list of floats or a one dimension numpy array. In that case, the first vector will be considered time and the rest will be considered signals numbered from 1 onwards. Units will be the ones of the vectors.
If data is not provided, a ***transientAsync*** command will be issued and its returned data will be used to provide as input data. Unit for time will be seconds and it will be volt for other vectors.

If a time vector is available, the command provides:

- Start time
- End time
- Total time

From each signal in the data, the command provides:

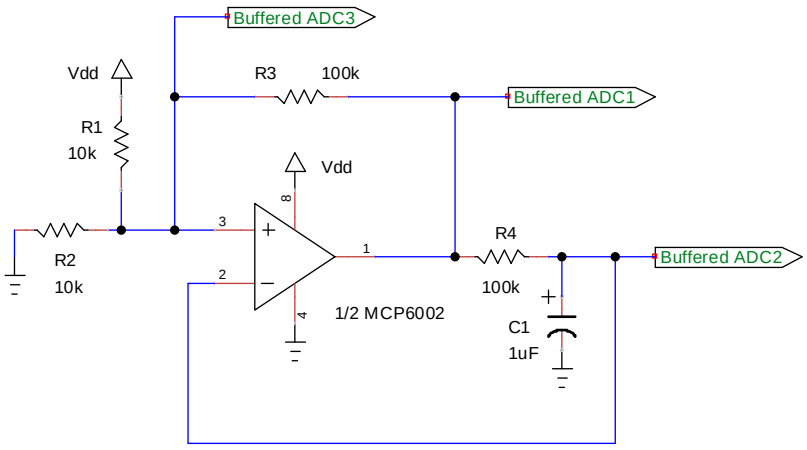
- Mean value
- Standard deviation value
- High Peak
- Low Peak
- Peak to peak value
- Halfrange value
- RMS value
- Mean period

Note that, if the signal is not periodic, the mean period value won't make sense.

Units only will be provided if parameter data is not provided and analysis is performed from a ***transientAsync*** command.

Example 25
Analyze an astable circuit

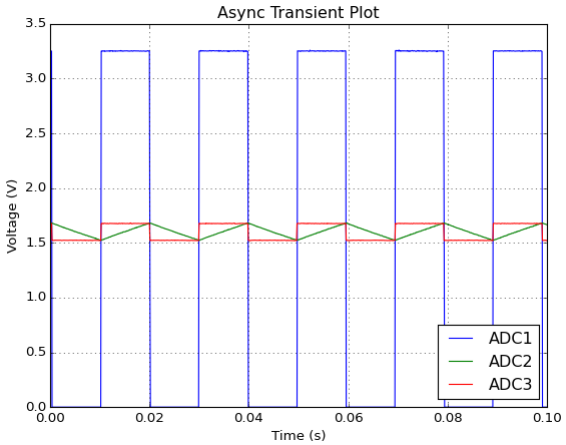
We use the same astable of Example 15.



We set the sample time and storage in the same way as in example 15. Using the ***setPlotReturnData*** command we make the ***tranAsyncPlot*** command return the measurement data. In the last line, we analyze the obtained data.

```
>>> slab.setSampleTime(0.0001)
>>> slab.setTransientStorage(1000,3)
>>> slab.setPlotReturnData(True)
>>> data = slab.tranAsyncPlot()
>>> slab.analyze(data)
```

When the ***tranAsyncPlot*** executes we obtain the ADC 1, 2 and 3 curves:



When we close the image, the data is sent to the analyze command that generates the following text information:

```
Min time: 0.0
Max time: 0.0999
Total time: 0.0999

Signal 1
   Mean: 1.61598390656
   Std Dev: 1.62459129868

   High Peak: 3.261277771
   Low Peak: 0.0
   Peak2peak: 3.261277771
   Half Range: 1.6306388855
   RMS: 2.29144078561

   Mean period: 0.019725
   Mean frequency: 50.6970849176

Signal 2
   Mean: 1.60310153961
   Std Dev: 0.0460045092635

   High Peak: 1.68262939453
   Low Peak: 1.52305755615
   Peak2peak: 0.159571838379
   Half Range: 1.60284347534
   RMS: 1.60376150383

   Mean period: 0.0197
   Mean frequency: 50.7614213198

Signal 3
   Mean: 1.59941683502
   Std Dev: 0.0768686539727

   High Peak: 1.68182373047
   Low Peak: 1.51902923584
   Peak2peak: 0.162794494629
   Half Range: 1.60042648315
   RMS: 1.60126293972

   Mean period: 0.019725
   Mean frequency: 50.6970849176
```

We see that we obtain data for all signals. Frequency is always 50.7 Hz. ADC 1 ranges from 0 V to 3.3 V and both ADC 2 and ADC 3 range from 1.52 V to 1.68 V.

References

STM32 Nucleo Page
http://www.st.com/en/evaluation-tools/stm32-mcu-nucleo.html

MBED Developer Page
https://developer.mbed.org/

Python page:
https://www.python.org/

Anaconda page:
https://www.continuum.io/downloads

TinyCad
https://sourceforge.net/projects/tinycad/
Circuit images on this document have been drawn using the free software TinyCad