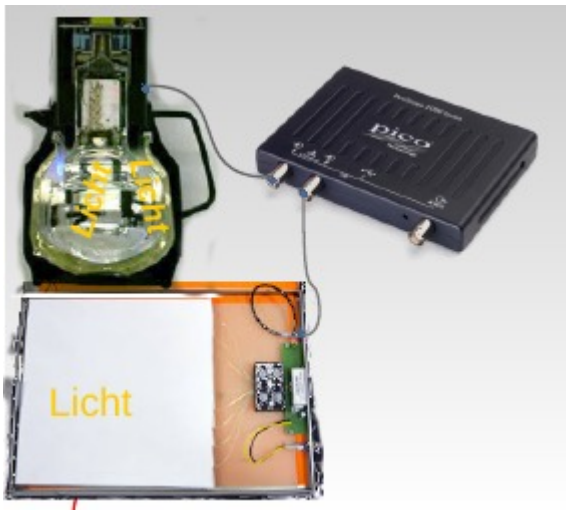
picoCosmo

python script to analyze data from CosMO detectors and Kamiokanne by Netzwerk Teilchenwelt with picoScope USB device



This is the **English Version** of the documentation.

For German readers:
Die **deutsche Version** dieses Dokuments findet sich unter dem Link README_de.md.

The software is tailored to identify short pulses from muon detectors (the scintillator panels of the *CosMO*-experiment by "Netzwerk Teilchenwelt", http://www.teilchenwelt.de, or the *Kamiokanne*-Experiment (a water-Cherenkov detector with photomultiplier readout) with a PicoScope USB oscilloscope with two or four channels.

Data is read from the PicoScope device via a Buffer Manager, see project *picoDAQ* (https://github.com/GuenterQuast/picoDAQ), which records waveforms and distributes them to consumer processes. The consumers either provide real-time displays of a sub-set of the data or perform data analysis. *PiocoCosmo* is a specialised and extended version of the script *runDAQ.py* from project *picoDAQ*.

The analysis proceeds in three steps. First, the trigger is validated by cross-correlation with a signal template located around the trigger time. Next, coincidences near a validated triggering pulse are searched for in all connected channels. The optional third step performs a search for additional pulses after the triggering event, indicating the decay of a stopped muon in or near the detector.

The software provides real-time displays of waveforms, detector signals and rates. Optionally, parameters of identified pulses or of double-pulses are written to files in CSV format. In addition, raw waveforms or pictures in *.png* format of identified double pulses can optionally be stored for off-line analysis or for an instructive analysis "by hand" based on the waveform pictures. From this information, the mean muon lifetime in the muon rest frame (2.2 µs) can be derived.

Dependence on other packages

This code relies on

- the data acquisition and analysis package picoDAQ, https://github.com/GuenterQuast/picoDAQ, which must be installed on your system, together with
- the *python* bindings of the *pico-python* project by Colin O'Flynn, see https://github.com/colinoflynn/pico-python and
- the low-level drivers and C-libraries contained in the Pico Technology Software Development Kit, see https://www.picotech.com/downloads

The code also runs on a Raspberry Pi.

Program Execution

To run *picoCosmo* from the graphical interface, create the subdirecotry *picoCosmo* in your home directory, where modified configurations and output are stored:

```
cd
mkdir picoCosmo
```

Now you are ready to execute the graphical interface of *picoCosmo*:

```
<picoCosmo install directory>/CosmoGui.py
```

Alternatively you may change to the *picoCosmo* installation directory

```
cd  <picoCosmo install directory>
```

and start *./runCosmo.py* from the command line.

Both scripts are controlled by an input file in *.daq*` format, which specifies the configuration:

- Cosmo.daq : configuration for Cosmo-Panels
- Kanne.daq : configuration for Kamiokanne

These files contain the names of files for the configurations of the PicoScope device, the Buffer Manager and the Pulse Filter.

The graphical interface allows to inspect and modify the configurations, to select a working directory for the output files and to start a new run. Configuration and output files are stored in a newly created sub-directory *_/* in the selected working directory, where a specific can be defined by the user.

Installation

- Install the PicoTech Software Development Kit from https://www.picotech.com/library/oscilloscopes/picoscope-software-development-kit-sdk.
- Install the *pico-pyhton* package from https://github.com/colinoflynn/pico-python.
- Install the picoDAQ package, vers. >= 0.7.2 from https://github.com/GuenterQuast/picoDAQ.
- Download all files from this project https://github.com/GuenterQuast/picoCosmo.

For your convenience, the sub-directory *whl/* contains compatible versions of *picoscope* from package *pico-python* and *picodaqa* from package *picoDAQ* as python-wheels, which you may install via `pip install package-<vers\>-<tags\>.whl`.

Configuration and program execution

As stated above, data Acquisition and Analysis is started using the graphical interface by typing `<picoCosmo installation directroy>/CosmoGui.py xxx.daq`. If no configuration is given, the file *default.daq* is used. Optionally, a run can also be started by directly executing `./runCosmo xxx.daq` from the *picocCosmo* installation directory.

After run start, control is performed via the main window of the BufferManager, which contains the options *Pause*, *Resume*, *Stop* and *EndRun*. In stopped state, all windows remain open and graphs may be saved and log-output inspected. In End-state, all processes are stopped, and consequently all windows disappear. Resume running from Stop-state is presently not foreseen.

The helper scripts, *plotDoublePulses.py* and *makeFigs.py*, allow to read stored raw waveforms in text or zip format from the double-pulse search and display resp. store them as images in *.png* format.

The configuration for *runCosmo.py* is defined in several *.yaml* files contained in sub-directory *config/**. The files used for a specific configuration are listed in files of type *.daq* , e. g. *Kanne.daq* for Kamiokanne and *Cosmo.daq* for a run with the CosMo panels.

The *.yaml* files specify configurations for the oscilloscope, the BufferManager and the Pulse Filter. Here is the example to run with Kamiokanne:

```
# file Kanne.daq
# --------------------
# configuration files for Kamiokanne

DeviceFile: config/PMpulse.yaml   # Oscilloscope configuration file
BMfile:     config/BMconfig.yaml  # Buffer Manager configuration
PFfile:     config/PFKanne.yaml   # Pulse Filter Configuration
```

The oscilloscope configuration specifies the oscilloscope model, the active channels and the trigger conditions:

```
# file PMpulse.yaml
# -----------------
# configuration file for PicoScope 2000 Series connected to a PM tube

PSmodel: 2000a      # model type here (2000a is default)

picoChannels:      [A]
ChanRanges:        [0.5, 0.2]
ChanOffsets:       [0.4, 0.45]

sampleTime:   16.E-6  # scientific format with '.' and signed exponent
Nsamples:     3500

trgChan:    A
trgThr:     -45.E-3
trgTyp:     Falling
trgTO:      5000    #   time-out after which read-out occurs
pretrig:    0.05
ChanColors: [darkblue, sienna, indigo]
```

The configuration for the Buffer manager allows to specify the number of buffers, the display modules for raw data and the logging level:

```
# file BMconfig.yaml
# ------------------
# configuration of picoDAQ Buffer Manager

NBuffers: 16         # number of buffers to store raw waveforms
BMmodules: [mpOsci]  # BufferMan modules to start
verbose: 1   # set verbosity level
LogFile: BMsum # write log-file entries with current statistics
```

The configuration for running with the CosMO detectors or Kamiokanne are specified in the PulseFilter configuration file. It contains the specification of desired output files, the pulse shapes and pulse heights for every connected channel, the display modules to be started and and the definition of real-time histograms for pulse heights, muon rate and muon life-time. An example is shown here:

```
# file PFKanne.yaml
# -------------------
# Configuration file for PulseFilter with Kamiokanne

#logFile: pFilt     # store all pulses, put Null if no output wanted
logFile: Null      # store all pulses, put Null if no output wanted
logFile2: dpFilt   # store double-pulses only, put Null if not
rawFile:  rawDP    # store raw wave forms, put Null if not wanted
pictFile: pictDP   # save pictures of double-pulse wave forms

# pulse parameters
#         ______
#        /      \
#     _ /_ _ _ _ \_
#        r  on   f
#

# pulse shape(s) for channels
#    if only one given, it is used for all channels
pulseShape:
 - pheight: -0.035
   taur   : 20.E-9
   tauon  : 12.E-9
   tauf   : 128.E-9

# pulse shape for trigger signal
#   optional - if not given, uses pulseShape
trgPulseShape:
 - pheight: -0.045
   taur   : 20.E-9
   tauon  : 12.E-9
   tauf   : 128.E-9

# Display Modules to be started
```

```
modules: [RMeter, Display, Hists]

# Definition of Histograms
histograms:
 # min  max Nbins ymax    title              lin/log
 - [0., 0.4, 50, 20., "noise Trg. Pulse (V)", 0]
 - [0., 0.8, 50, 15., "valid Trg. Pulse (V)", 0]
 - [0., 15.,  45, 7.5, "Tau (µs)", 1]
 - [0., 0.8, 50, 15., "Pulse Height (V)", 0]

doublePulse: True  # switch for double-pulse search
```

Example output

The directory *./output* contains the results from a long run of almost 20 days with the Kamiokanne detector and a one-day run with the CosMO panels. The compressed file *rawDP_.dat.zip* contains the raw wave forms of identified double-pulses in *yaml*-format. The scripts *plotDoublePulses.py* and *makeFigs.py* can be used to read the zipped file and to produce graphical displays or images in *.png* format of the waveforms.

The parameters of events containing double-pulses are stored in file *dpKanne2_180403.dat*. An unbinned log-likelihood fit of measured lifetimes between 1.5 µs and 15. µs with the script *fit_dpData.py* yields the result shown in figure *life-ofMU_.png*.

Running on Raspberry Pi

picoCosmo also runs on the very popular Rasbperry Pi single board computer. After setting up your Raspberry Pi, the following steps should be taken to update and install all necessary packages:

```
sudo apt-get update
sudo apt-get upgrade

sudo apt-get install python3-scipy
sudo apt-get install python3-matplotlib
sudo apt-get install python3-pyqt5

sudo pip3 install pyyaml

# PicoTech base drivers for picoScope usb devices
#   see https://www.picotech.com/support/topic14649.html
# after inclusion of the picotech raspbian repository:
sudo apt-get install libps2000a
# allow access of user pi to usb port
sudo usermod -a -G tty pi

# get picoCosmo code and dependencies
mkdir git
cd git
git pull https://github.com/GuenterQuast/picoCosmo
cd picoCosmo/whl
sudo pip3 install *.whl
```

Create the subdirectory `picoCosmo` in the home directory, where all the output and modified configuration files will be stored:

```
cd
mkdir picoCosmo
```

Now you are ready to execute the graphical interface of `picoCosmo`:

```
<picoCosmo install directory >/CosmoGui.py
```