QtDaq python analysis tools

Chloé Sole

Updated: April 5, 2022

This document covers the base workings of the `read_dat` class and `event` class developed for analysis of `dat` files produced by QtDaq software [ref].
To use these classes please ensure the associated python files are in the same folder as your script and that the dependencies are installed.

Contents

1 Class `read_dat`

Parameters: file_name, sample_rate, CFD, t_start, t_long, t_short, baseline_samples, output
Dependencies: scipy, event, matplotlib, csv
Public Attributes: headerSize, maxChannels, preambleSize, fileName, inputFile, header, eventCounter, eventTimeStamp, endFile, nsPerSample, CFD (array of length 3), tStart, tShort, tLong, chActive, fails, totFails, baselineSamples, selection, cuts
Public Methods: read_event, lst_out, get_fails, add_selections, select_events
Private Methods: onclick, press

This class provides an object representing a single file. The file is assumed to have an event structure as follows:

headerSize:	72 bytes
maxChannels:	64 ch
preambleSize:	4+20+4*maxChannels bytes

Each event is preceded by a preamble containing an event timestamp (preamble[5]*8×10^3 µs) and the size of each channel's acquisition (preamble[6:]).

The `read_dat` class needs to be imported into your script in order to use these tools.

```
from read_dat import *
```

To then create an object call `read_dat()`

```
file = read_dat("filename.dat",...)
```

1.1 Parameters

```
read_dat(self, file_name, sample_rate, CFD, t_start, t_long, t_short, baseline_samples,
output)
```

`file_name`: *str* of the file name must include file suffix ".dat"

`sample_rate`: float, ns per sample for the acquired data.
Default: 2 ns (DT5730 sample rate)

`CFD`: 1D array of length 3: [Fraction,Length (samples),Offset (samples)]
Refer to event.CFD for more information on what these parameters are
default: [0.75, 6, 6]

`t_start`: start time (ns) from the CFD defined t_0
default: -80 ns

`t_long`: long integral gate (ns) from the CFD defined t_0
default: 400 ns

`t_short`: short integral gate (ns) from the CFD defined t_0
default: 10 ns

`baseline_samples`: length of baseline calculation in samples
default: 200 samples

1.2 Attributes

All attributes are currently public. Please edit attributes directly with caution as doing this will not update events already processed in the file.

`headerSize`: *int*, header size of 72 bytes
`maxChannels`: *int*, maximum number of channels allowed 64
`preambleSize`: *int*, 4+20+4*maxChannels bytes
`fileName`: *str*, input file name
`inputFile`: *file*, the open binary file
`header`: *binary*, header for the open file

`eventCounter`: *int*, running event counter to keep track of where you are in the file

`eventTimeStamp`: *float*, time stamp in µs of the last event read

`endFile`: *boolean*, if end of file has been reached `endFile = True`

`nsPerSample`: *float*, the time in ns between samples

`CFD`: *1D array of length 3*, parameters required to determine a Constant Fraction Discriminator filter of the events [Fraction,Length (samples),Offset (samples)]

`tStart`: *int*, start time (ns) from the CFD defined t_0

`tShort`: *int*, short gate end time (ns) from the CFD defined t_0

`tLong`: *int*, long gate end time (ns) from the CFD defined t_0

`chActive`: *1D array*, an array of the active channels indices

`fails`: *n× 5D array*, fail tracker [start, long, short, integrals, zero] per ch, a description of the fail checks can be found in section 1.5.

`totFails`: *1D array*, running count of the number of failed events per channel

`baselineSamples`: *int*, number of samples used to calculate the baseline, taken from the first sample in the event

`selection`: *2D list*, list of `x` and `y` co-ordinates used for encolsed area selections

`cuts`: *1D array*, `len(cuts)` is equal to the number of selections available, with each element containing the number of co-ordinates in that selection

1.3 Public Methods

1.3.1 `read_event`

```
read_dat.read_event(self)
```

Method to read one event from the open file.

Parameters: No additional parameters

Returns: **ev**: `event` array
Array of `event` objects associated with that event just read for the active channels.
Returns `True` when the file end is reached

Raises:

1.3.2 `lst_out`

```
read_dat.lst_out(self,events=False, ch=True,output=True, traces=False, cuts=False, filename
```

Method to read multiple events or whole file from the open file with list mode output options. Produces a **params** and **trace** csv file per channel if requested.

Parameters: **events**: int or boolean, optional
The number of events to read and output the desired list mode information.
If `False` the full file is read.
Default value is `False`.

ch: int array or boolean, optional
If `True` all channels are read out.
If `int array` only the selected channels are read out (channel numbering from 0).
Default value is `True`.

output: int array or boolean, optional
If `True` L [ch], S [ch], $\mathrm{T}_{\mathrm{trigger}}$ [us], baseline [bits], pulse height [bits] are read out into a file per channel.
If a binary `int array` (len=5) only the selected parameters are read out.
The format of the array is [L,S,T,baseline,PH] where 1 is and indication to output that parameter and 0 is to ignore that parameter.
Default value is `True`.

traces: boolean, optional
If `False` no traces will be outputted.

If `True` traces will be outputted list mode in a csv file per channel.
Default value is `False`.

cuts: int array or boolean, optional
If `False` no cuts are taken into account in the outputted data.
If an `int array` (len = number of cuts added) where the index of the array is the cut id and the value being 1, 0 or -1 includes, ignores or excludes that cut.
Default value is `False`.

filename: str, optional
Desired output file name.
Default value is empty which produces *fileName*__**params**__*ch*.**csv**.

Output: *fileName*__**params**__*ch*.**csv**: a csv file per channel with the list mode output of the analysed parameters for the desired number of events.
*fileName*__**traces**__*ch*.**csv**: a csv file per channel with the list mode output of the traces (in bits) for the desired number of events.

Raises: No additional error codes.

1.3.3 `get_fails`

`read_dat.get_fails(`*`self`*`,display=False)`

Returns fail information for the processed events.

Parameters: **display**: boolean, optional
If `False`, nothing is printed out in the `cmd`,
If `True`, a summary of the fails per channel is printed to `cmd`
Default value is False.

Returns: **fails**: n×5D int array for n ch,
in the format of [start, long, short, integral, zero] for n channels
The value at the associated index indicates the number of fails out of the processed events that have failed that check. Refer to section 1.5 for a breakdown of the fail checks.
totFails: int array, length = number of active channels
Number of events failed per channel
eventCounter: int
Total number of events processed

Output: If **display** is `True`, writes the break down of fails per channel to `cmd`.

Raises:

1.3.4 `add_selections`

`read_dat.add_selections(`*`self`*`,L=[],S=[],mode="m",lims = [[0,50000],[0,1]],file=False)`

Method to add multiple cuts to the events. Can be run in manual `m` or predetermined `p` modes. Requires lists of L and S values in mode `m` for visual support to determine the selections. No lists of L and S values are required for mode `p` and no visual will be displayed.

Parameters: **L**: float array, optional
Array of *L* values for the processed events
S: float array, optional
Array of *S* values for the processed events
mode: char, optional
`m`: Manual mode, includes visual assistance when setting the enclosed area selections, refer to the key presses in table 1.
`p`: Predetermined mode, allows one to enter past manual selections without the visual assistance. The format of this file should match the output file as described below.
Default value is `m`.

lims: 2x2 float array, optional
Array of x and y limits for the visual aid SL plot.
Default value is [[0,50000],[0,1]].
file: False, or *str*, optional
if `false` no output file is set for the selections and they are not saved unless.
if mode is `p` then a file name is required *str*, this is taken as the file name for the input file for the predetermined selections.
Default value is `False`.

Output: If `file` is a *str* and mode is `p` then an output file is created *fileName*_**cuts_SL.csv**.
The output file has a header of one line, the cuts are then written to the rows in pairs. The first row is the x co-ordinates for the first cut and the second row in the y co-ordinates for the first cut.

Returns: Returns nothing, but sets the attributes `selection` and `cuts` for the file.

Raises: No additional errors.

1.3.5 `select_events`

```
read_dat.select_events(self,L,S, cut_id=[0],inc=[1],visual=False,lims = [[0,50000],[0,1]])
```

Method to pull the events which fall within the desired area. This area is defined by the inclusion or exclusion of the added selections. If no selections have been added to the `read_dat` object all events will be returned. There is a visual aid to ensure you are selecting for the correct events.

Parameters:
L: float array, required
Array of *L* values for the processed events
S: float array, required
Array of *S* values for the processed events
cut_id: int array, optional
array of cut id's which are either included or excluded.
Default value is `[0]`, referring to cut 0.
inc: int array, optional
array of 1's or -1's to indicate which cut is either included or excluded respectively, `len(inc) = len(cut_id)`.
Default value is `[1]`, including cut 0 as default.
visual: boolean, optional
If `True` a SL plot with the included and excluded events is displayed with the cut boundaries.
Default value is `False`.
lims: 2x2 float array, optional
Array of x and y limits for the visual aid SL plot.
Default value is [[0,50000],[0,1]].

Returns: Returns S and L values which fall within the defined area.

Raises: No additional errors. add check for validity of parameters

1.4 Private Methods

1.4.1 `onclick`

```
read_dat.__onclick(self)
```

Method used by `read_dat.add_selections` through `read_dat.__press`. Determines the clicked co-ordinates and stores them in attribute `selection`. Accessible within the `read_dat` class. Is called by a key press `a` or `A` while in `read_dat.add_selections`.

1.4.2 press

`read_dat.__press(`*`self`*`)`

While in manual mode of `read_dat.add_selections`, the commands for adding, editing or removing selections are given by the key presses in table 1. In an active selection, `a` or `A` has been pressed, use the left mouse click to select co-ordinates. The selected co-ordinates will be printed to screen and saved to attribute `selection` of the associated file.

Table 1: The relevant key presses while in manual mode for adding selections

Key	Action
a, A	Add a new selection
u, U	Undo previous point, only usable while in a selection
x, X	End current selection, can only end a selection if there are more than 2 co-ordinates in the selection
d, D	Delete previous completed selection
q, Q	Quit, ends visual guide and re-enters the main code segment
o, O	Outputs the selections added to a file (***fileName*__cuts_SL.csv**)

1.5 Fails

The array returned by `read_dat.get_fails` is a $n \times 5$D array for n channels. The five fail conditions presented per channel are indicated in table 2.

Table 2: Description of the fail checks returned by `read_dat.get_fails`.

Index	Fail Name	Fail Condition
0	start	The start time is set outside of the acquisition window
1	long	The long integral end gate is outside of the acquisition window
2	short	The short integral end gate is outside of the acquisition window
3	integral	The calculated short integral is negative or the calculated long integral has a smaller value than the calculated short integral
4	zero	The CFD calculation failed to return a reasonable t_0

2 Class event

Parameters: event_id, ch, t0, trace, CFD, integrals, baseline
Dependencies: numpy
Public Attributes: eventID, ch, triggerTime, baseline, trace, CFD, istart, ishort, ilong, longIntegral, shortIntegral, fails
Public Methods: get_event_id, get_ch, get_t0, get_trace, get_CFD, get_baseline, get_long_integral, get_short_integral, get_pulse_shape, get_pulse_height, get_fails
Private Methods: cfd, sum_integral, check_polarity

This class provides an object representing a single event.
The `event` class needs to be imported into your script in order to manipulate event objects directly and not only through the `read_dat` class. If you are using the `read_dat` class you do not need to import the `event` class additionally as it is inherited. To import the `event` class use the following command:

```
from event import *
```

To then create an event object call `event()`, refer to section 2.1 for a description of the parameters you can use to initiate an `event` object.

```
ev = event(...)
```

2.1 Parameters

`event(`*`self`*`, `**`event_id, ch, t0, trace, CFD, integrals, baseline`**`)`

`event_id`: *int*, event counter across all channels
`ch`: *int*, which channel the event occurred on
`t0`: *float*, trigger time in µs
`trace`: *1D array*, full y values of the trace in bits, the x values are the trace can be determined through the properties of the digitiser used to acquire the data
`CFD`: *1D array,* an array of length 3 containing the CFD analysis parameters required: [fraction,length,offset]. Refer to the private method `cfd` (sec. []) for a break down of these parameters.
`integrals`: *1D array* of length 3, containing the indices for the start and ends of the integrals in the format of
[start, short, long]
`baseline`: *int*, the number of samples used for the baseline calculation

2.2 Attributes

All attributes are currently public.

`eventID`: *int*, event ID
`ch`: *int*, channel the event was acquired on
`triggerTime`: *float*, the trigger time in µs
`baseline`: *float*, the baseline as calculated as the average of the number of samples from the start of the acquisition window as provided by the user
`trace`: *1D array*, array representing the y-values in bits for the sampled trace
`CFD`: [*1D array, float*], the array represents the event post CFD filter and the returned float (`izero`) is the determined weighted average for the index of the zero-crossing
`istart`: *int array*, the start index for the integrals, determined as
`istart = izero + integrals[0]`
`ishort`: *int array*, the end index for the short integral, defined as
`ishort = izero + integrals[1]`
`ilong`: *int array*, the end index for the long integral, defined as
`ilong = izero + integrals[2]`
`longIntegral`: *float array*, the value of the long integral, determined as the sum of samples within

the user defined gates
`shortIntegral`: *float array*, the value of the short integral, determined as the sum of samples within

the user defined gates
`fails`: *1D array, len = 5*, fail tracker [start, long, short, integrals, zero], a description of the fail conditions can be found in section 2.5

2.3 Public Methods

2.3.1 `get_event_id`

`event.get_event_id(`*`self`*`)`

Method to return the associated event ID.

Parameters: none
Returns: **eventID**: int
The event ID of the event used to call this method.

2.3.2 `get_ch`

`event.get_ch(`*`self`*`)`

Method to return the channel on which the associated event was acquired.

Parameters:	none
Returns:	**ch**: int The channel on which the event, used to call this method, was acquired.

2.3.3 `get_t0`

`event.get_t0(`*`self`*`)`

Method to return the trigger time in µs of the associated event.

Parameters:	none
Returns:	**triggerTime**: float The trigger time (µs) of the event used to call this method.

2.3.4 `get_trace`

`event.get_trace(`*`self`*`)`

Method to returns the y-values of the trace for the associated event.

Parameters:	none
Returns:	**trace**: 1D int array The y-values of the trace of the event.

2.3.5 `get_CFD`

`event.get_CFD(`*`self`*`)`

Method to return the CFD of the associated event.

Parameters:	none
Returns:	**cfdArr**: 1D float array, The array representing the CFD filtered trace of the event. **zero__cross**: float, The index value of the zero-crossing, determined as a weighted average of the indices on opposite sides of the zero-point.

2.3.6 `get_baseline`

`event.get_baseline(`*`self`*`)`

Method to return the baseline of the associated event.

Parameters:	none
Returns:	**baseline**: float Average of the first x, user defined, many samples of the trace.

2.3.7 `get_long_integral`

`event.get_long_integral(`*`self`*`)`

Method to return the long integral or integrals of the associated event.

Parameters:	none
Returns:	**longIntegral**: int

Sum of the samples within the user defined start and end gates for the long integral.

2.3.8 `get_short_integral`

`event.get_short_integral(`*`self`*`)`

Method to return the short integral or integrals of the associated event.

Parameters: none
Returns: **shortIntegral**: int
Sum of the samples within the user defined start and end gates for the short integral.

2.3.9 `get_pulse_shape`

`event.get_pulse_shape(`*`self`*`)`

Method to return the pulse shape value or values of the associated event determined through a charge comparison (CC) calculation. If there are multiple short and long gates pulse shape cannot be calculated.

The pulse shape parameter S, as determined through a CC, is defined as:

$$S = \frac{Q_s}{Q_l} \tag{1}$$

where, $Q_{s/l}$ represent the short and long integrals respectively.

Parameters: none
Returns: **S**: float
The pulse shape parameter S of the event as determined through the desired method.

2.3.10 `get_pulse_height`

`event.get_pulse_height(`*`self`*`)`

Method to return the pulse height of the associated event.

Parameters: none
Returns: **eventID**: float
The maximum of the measured trace.

2.3.11 `get_fails`

`event.get_fails(`*`self`*`,display=False)`

Returns fail information for the associated event.

Parameters: **display**: boolean, optional
If `False`, nothing is printed out in the `cmd`,
If `True`, a summary of the fails for the selected event is printed to `cmd`
Default value is False.
Returns: **fails**: n×5D int array for n ch,
in the format of [start, long, short, integral, zero] for n channels
The value at the associated index indicates the number of fails out of the processed events that have failed that check. Refer to section 1.5 for a breakdown of the fail checks.
Output: If **display** is `True`, writes the break down of fails per channel to `cmd`.

2.4 Private Methods

2.4.1 `cfd`

`event.__cfd(`*`self`*`,F,L,O)`

Method used when an event is created to assign values to the attribute `CFD`. This filter is defined in equation 2 where v_i is the CFD filtered event sample i, L is the filter length in samples, F is the filter fraction, O is the filter offset and V is the voltage of the raw unfiltered signal, N and O are in either ns or samples as long as the units used are consistent throughout the calculation.

$$v_i = \sum_{j=1}^{L} \left(F V_{i-j} - V_{i-j-O} \right) \tag{2}$$

Parameters: **F**: float, required
The fraction F which is used to scale the trace in equation 2.
L: int, required
The length of the filter L
O: int required
The offset of the filter O

Returns: **cfdArr**: 1D float array,
The trace array post CFD filter.
zero_cross: float,
The weighted sum of the indices of the samples closest to the zero-crossing.

2.4.2 `sum_integral`

`event.__sum_integral(`*`self`*`,end)`

Method used when an event is created to assign values to the long and short integrals.

Parameters: **end**: int, required
The end index for the sum integral.

Returns: **sum_int**: int,
The sum of the trace from the start index up to and not including the end index.

2.4.3 `check_polarity`

`event.__check_polarity(`*`self`*`)`

Method used when initiating an event to determine the polarity of the pulse and to switch it if necessary for consistency in the calculations.

2.5 Fails

The array returned by `event.get_fails` is a 1 × 5D array for a single event. The five fail conditions presented for the calculations a single event can fail are indicated in table 3.

Table 3: Description of the fail checks returned by `event.get_fails`.

Index	Fail Name	Fail Condition
0	start	The start time is set outside of the acquisition window
1	long	The long integral end gate is outside of the acquisition window
2	short	The short integral end gate is outside of the acquisition window
3	integral	The calculated short integral is negative or the calculated long integral has a smaller value than the calculated short integral
4	zero	The CFD calculation failed to return a reasonable t_0

3 Examples