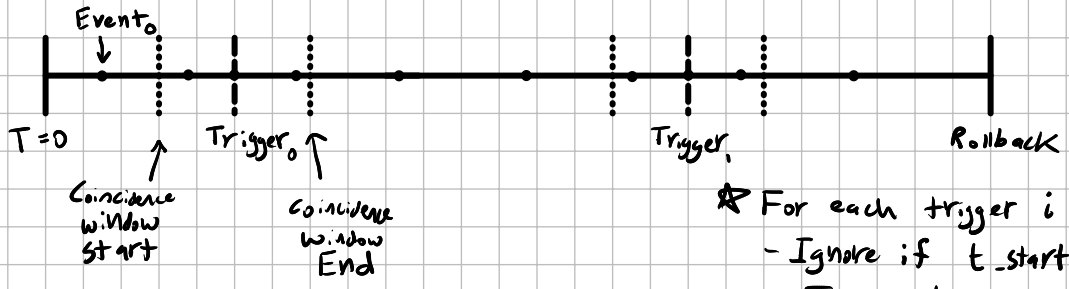


$t\_trg \equiv$ Trigger timetag

$t\_start \equiv$ Coincidence window Start timetag

$t\_end \equiv$ Coincidence window End timetag

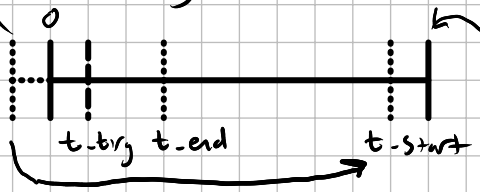
$t\_rollback \equiv$ Timetag rollback (INT_MAX)

☆ For each trigger i:
- Ignore if $t\_start_i \leq t\_end_{i-1}$
- Iterate backwards through each event until t_start.
- Iterate forwards through each event until t_end.

Taking rollback into Account:

1.) $t\_start = t\_trg - \frac{window}{2}$ (without rollback)

(If t_trg is within $0 \leq t\_trg < \frac{window}{2}$, take into account rollback)



if $(t\_start < 0)$ {

$t\_start = t\_rollback - \left(\frac{window}{2} - t\_trg\right)$

~~$t\_rollback = t\_start$~~

}

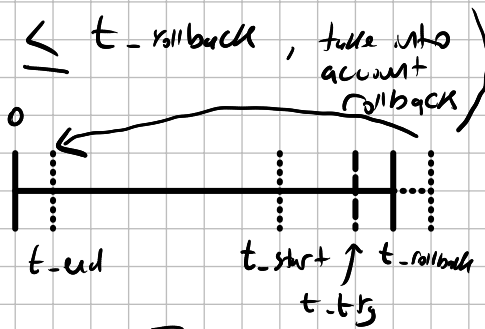
2.) $t\_end = t\_trg + \frac{window}{2}$ (without rollback)

(If t_trg is within $t\_rollback - \frac{window}{2} < t\_trg \leq t\_rollback$, take into account rollback)

if $(t\_end > t\_rollback)$ {

$t\_end = t\_trg + \frac{window}{2} - t\_rollback$

~~$t\_end = t\_rollback$~~

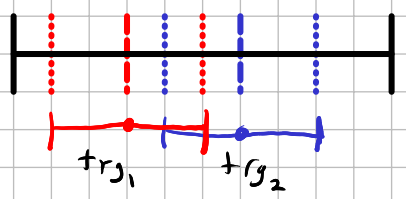


[Keep in mind that $t\_rollback \equiv$ INT_MAX, so t_end would initially be > INT_MAX. Must use uint32_t, so we can go beyond INT_MAX]

Checking if triggers are too close

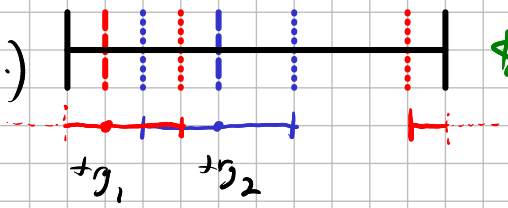
{ • Earlier trigger
• Later trigger

1.)



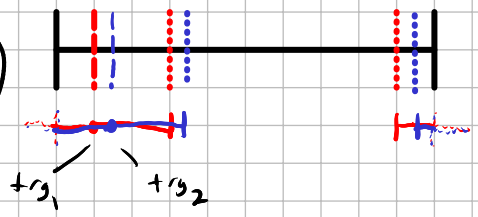
No rollback
No rollback

2.)



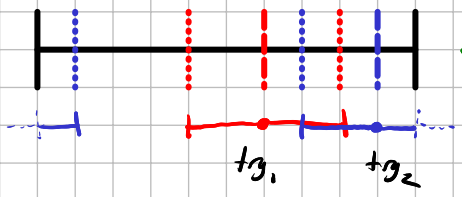
Below 0
No rollback

3.)



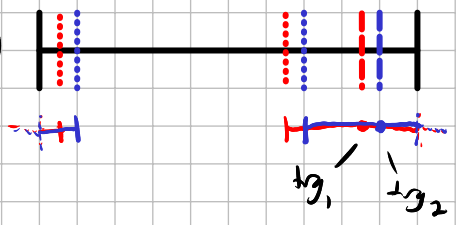
Below 0
Below 0

4.)



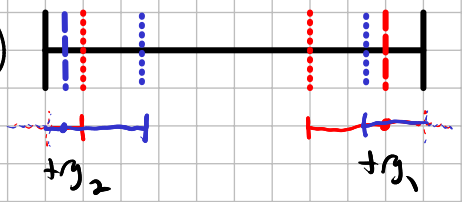
No rollback
Rollback

5.)



Rollback
Rollback

6.)



Rollback
Below 0

★ if $(t_{start_2} \leq t_{end_1})$
- ignore trigger$_2$

★ if $(t_{start_2} \geq t_{end_1})$
- ignore trigger$_2$

Between No rollback, below 0, and rollback, there are $3^2 = 9$ combinations for 2 triggers, but 1 must come before the other. So the remaining 3 combinations are forbidden:

{ No rollback, Below 0 | { Below 0, Rollback | { Rollback, No rollback

Observations:
- Only need to consider rollback if <u>both</u> triggers extend beyond either 0 or t_rollback.
- This can be checked with rollback bool.

★ if $(rollback_1$ && $rollback_2$ && $t_{start_2} \geq t_{end_1})$
- ignore trigger 2

★ if $(!(rollback_1$ && $rollback_2)$ && $t_{start_2} \leq t_{end_1})$
- ignore trigger 2

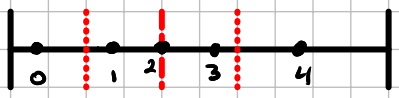
```
t_end_prev = 0;
extended_prev = true;
```

① `if (t_start ≤ t_end_prev && !(extended && extended_prev)) {`
`continue;`

② `}`
`else if (t_start ≥ t_end_prev && !extended && !extended_prev) {`
`continue;`
`}`

Check if events are within window, taking into account rollback

1.) Not extended beyond 0 or rollback:



Events 1 and 3 are within window

- checking below trigger timetag:

```
if (t ≥ t_start && t ≤ t_trg) {
  // Increment histograms, etc.
}
```

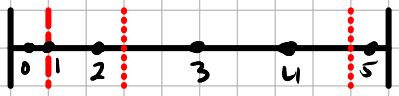
- checking above trigger timetag:

```
if (t < t_end && t > t_trg) {
  // Increment histograms, etc.
}
```

Not including t_end in case 2 adjacent triggers overlap perfectly with the start and end of their windows. In that case, the later trigger collects an event that is at the overlap.

If t == t_trg, the "below" check will cover it in order to avoid double counting.

2.) Extends below 0:



Events 0, 2, 5 within window

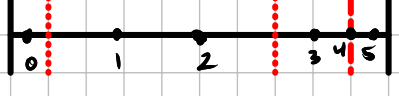
- checking below trigger:

```
if (t ≥ t_start || t ≤ t_trg) {
  // ...
}
```

- checking above trigger:

```
if (t < t_end && t > t_trg) {
  // ...
}
```

3.) Extends above t_rollback:



Events 0, 3, 5 within window

- checking below trigger:

```
if ( t ≥ t_start && t ≤ t_trg) {
  // ...
}
```

- checking above trigger:

```
if( t < t_end || t > t_trg) {
  // ...
}
```

Incrementing histograms efficiently for coincidences

What I would like to avoid is some complicated nested if statements that capture every possible combination of a coincidence scenario.

e.g.

```
if (ch == iE && !fE) {          // fE is true when E has already been collected during this coincidence window (avoids noise)
    if ( fDE) {
        // DE and E coincidence
    }
    if ( iFrontHE || iFrontLE){
        // Pos1 and E coincidence
    }
    :
}
:
```

1.) ~~Instead, one solution is to append the collected events to a vector for each trigger, and apply coincidences / increment histograms for only the elements in the vector.~~

2.) Or set energies and Pos1, Pos2 values to 0 by default and update if they are present in the window. Then increment everything in one go. the values that are still 0 will not actually be visualized if we add an if statement for values lower than the Histogram threshold.

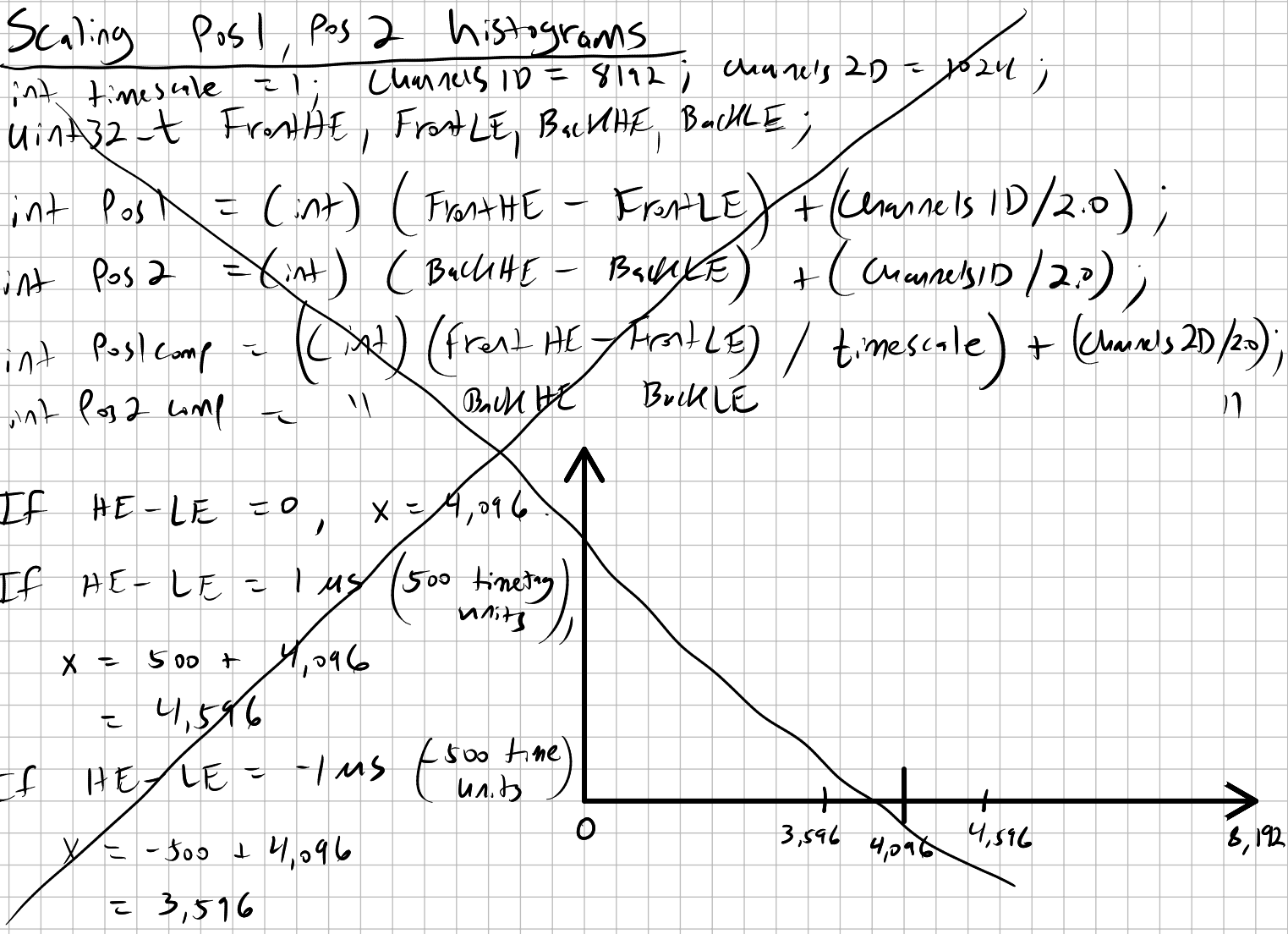
(this is basically what the old Sort routine did)

↑

Much easier!

Scaling Pos1, Pos2 histograms

```
int timescale = 1;  ChannelsID = 8192;  Channels2D = 1024;
Uint32_t FrontHE, FrontLE, BackHE, BackLE;
int Pos1 = (int) (FrontHE - FrontLE) + (ChannelsID/2.0);
int Pos2 = (int) (BackHE - BackLE) + (ChannelsID/2.0);
int Pos1comp = (int) (FrontHE - FrontLE) / timescale) + (Channels2D/2.0);
int Pos2comp = "     BackHE   BackLE                    "
```



If HE-LE = 0, x = 4,096.

If HE-LE = 1 ns (500 timetag units),

x = 500 + 4,096

= 4,596

If HE-LE = -1 ns (-500 time units)

x = -500 + 4,096

= 3,596

I need to scale it so that 0 and 8,196 correspond to -1 ns and +1 ns, respectively. (HE-LE = 1 ns ⇒ right side of spectrum)

$$Pos1 \in \underbrace{[-500 \text{ timetag units}}_{(-1\text{ ns})}, \underbrace{+500 \text{ timetag units}]}_{(+1\text{ ns})} \qquad Pos1 = FrontHE - FrontLE$$

$$Pos1^{1D}_{scaled} = \left(\frac{ChannelsID}{1{,}000}\right) Pos1 + \frac{ChannelsID}{2} \longrightarrow [0, 8192]$$

span of -500 to +500

$$Pos1^{2D} = (int)\ std::floor\left(Pos1^{1D}_{scaled} / 8.0\right);$$

$$\in [0, 1024]$$

Do (int) std::floor() here

Similarly for Pos2...

2 ns resolution ⇒ minimum res. for Pos1 spectrum covers 8,192 chs

→ 8 chs ⇔ 1 timetag unit

Is this a problem? There are only 1,000 unique FrontHE − FrontLE values in timetag units $(-500, \ldots, 0, \ldots, 499)$, so there can only be 1,000 bins max, right? Or can this be resolved by rebinning in EngSpec?

Go back to 4,096 chs? This would mean

4 chs $\Leftrightarrow$ 1 timetag unit

Might need CFD interpolation after all...

Implementing CFD interpolation

With the T_{fine} addition (10-bit number), the focal plane resolution becomes $-1\,\mu s$ to $+1\,\mu s$ $\Rightarrow$

$$-1\,\mu s \left(\frac{1000\,ns}{1\,\mu s}\right)\left(\frac{1024 \text{ timetag steps}}{2\,ns}\right)$$

$$= -512{,}000$$

$$+1\,\mu s \rightarrow +512{,}000$$

$\Rightarrow$ 1,024,000 bins ... plenty!

But we still need to scale this to 8,192 bins

$1{,}024{,}000 / 8{,}192 = 125$ exactly (or 250 for 4,096 bins)

$$Pos1^{ID}_{scaled} = (int)\ std::floor\left[\left(\frac{Channels1D}{1{,}024{,}000}\right)(FrontHE - FrontLE) + \frac{Channels1D}{2}\right]$$

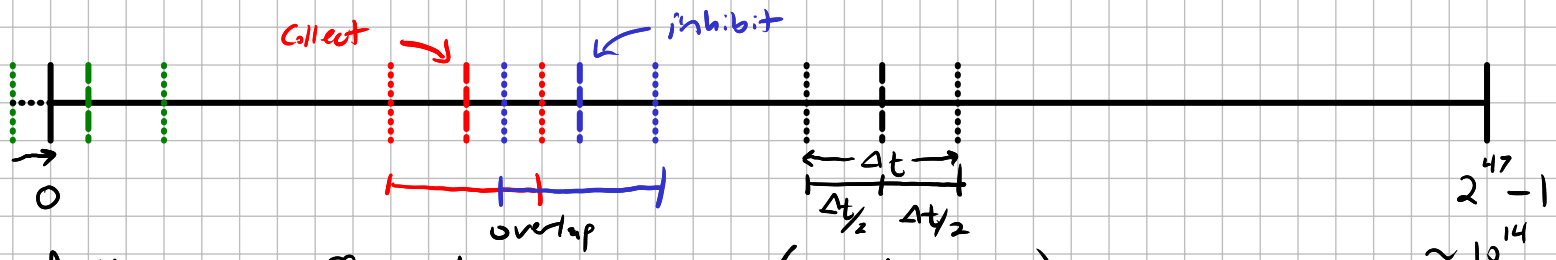
$$\in \left[0, Channels1D\right]$$

$FrontHE - FrontLE$: $-512{,}000$ to $+512{,}000$

64-bit signed int needed

Steps of $0.008 = \frac{1}{125}$ reduced to steps of 1 with std::floor.

Changes with EXTRAS enabled (rollback extension and fine timestamp)



- Rollback is effectively eliminated (78 hours...)
- Coincidence window can still extend below 0.
  Make the start time 0 in this case and extend stop time by Δt/2 like normal.
- Course time stamp (64 bits) ^ = EXTRAS[31:16] + TTT, where EXTRAS[31:16] are the most significant bits, i.e. $\left[\left[\left(\text{EXTRAS} \;\&\; \text{0xFFFF0000}\right) >> 16\right] << 31\right) + \text{TTT}$
- Fine time stamp (double) ^ = $\frac{\text{EXTRAS}[9:0]}{1024}$, fraction with 1024 steps

How to take the time difference for Pos sections

Course time stamp → uint64_t (47-bit max)
fine time stamp → double 10-bit number / 1024 $\in \left[0, \frac{1023}{1024}\right]$

- e.g. Let FrontHE_course = 10,000, FrontHE_fine = $\frac{634}{1024} = 0.619140625$
  FrontLE_course = 10,300, FrontLE_fine = $\frac{821}{1024} = 0.801757812\overline{5}$
  and assume they are within the same trigger window.

Then, the precise difference is

$$(\text{FrontHE\_course} + \text{FrontHE\_fine}) - (\text{FrontLE\_course} + \text{FrontLE\_fine})$$
$$= (\text{FrontHE\_course} - \text{FrontLE\_course}) + (\text{FrontHE\_fine} - \text{FrontLE\_fine})$$
$$\equiv \underbrace{\text{Pos1\_course}}_{\in \left[-2^{47}-1,\ 2^{47}-1\right]} + \underbrace{\text{Pos1\_fine}}_{\in \left[-\frac{1023}{1024},\ \frac{1023}{1024}\right]}$$

realistically $\in [-\Delta t, +\Delta t]$
where Δt = coincidence window

$$Pos1 = -300 + \left(\frac{634}{1024} - \frac{821}{1024}\right)$$

$$= -300 - \frac{187}{1024} = -300.1826171875 \text{ timing units}$$

How to convert this to 8,192 (or 4,096) bins?

$$Pos1^{ID}_{scaled} = (int)\ std::floor(\ X\ ) + \frac{channelsID}{2}$$ → offset so that HE-LE=0 is in the center

Where $X \in \left[-\frac{channelsID}{2}, +\frac{channelsID}{2}\right]$

and $Pos1^{ID}_{scaled} \in [0, channelsID]$.

Need $Pos1 \xrightarrow{map} [-500, +500]$ in steps of $1/1024$

$\rightarrow [-512{,}000, +512{,}000]$

e.g. $\left(-300 - \frac{187}{1024}\right) * 1024 = (-300 * 1024) - 187$

$= -307{,}387$

So $(Pos1 * 1{,}024) \in \left[-512{,}000 - \frac{1023}{1024},\ 512{,}000 + \frac{1023}{1024}\right]$

$$Pos1^{ID}_{scaled} = (int)\ std::floor\left[\left(\frac{channelsID}{1{,}024{,}000}\right)(Pos1 * 1024)\right] + \frac{channelsID}{2},$$

Where $Pos1 = (FHE_C - FLE_C) + (BHE_F - BLE_F)$

$= Pos1_{coarse} + Pos1_{fine}$

or equivalently

$$Pos^{ID}_{scaled} = (int)\ std::floor\left(\frac{channelsID}{1000} Pos1\right) + \frac{channelsID}{2}$$

Testing with debug.cpp (in FirmSpec)

Assuming ChannelsID = 8192
and Histogram_Threshold = 2

For $TTT_{FrontHE} = 4601$

$TTT_{FrontLE} = 5099$, $Pos1_{raw} = -498$, $Pos1_{scaled} = 21$] 8 ch jump

5098 → −497 → 29

5097 → −496 → 37

★ Course timestamp resolution is 8 ch (2 ns)

Pos1comp = 2

3

4] 1 ch jump

Fine timestamp resolution should be $\frac{2\ ns}{1024} = 0.00195\,3125$

$\Rightarrow 0.000\,78125$ ch

$= 1/128$ chs, but we scale it to nearest (floor) integer channel

For $TTT_{FrontHE} = 4601$, $TTT_{FrontLE} = 5099$

$T^{fine}_{FrontHE} = 657$, $T^{fine}_{FrontLE} = 75$, $\Rightarrow Pos1_{scaled} = 21$

76 → 21

(+128) 203 → crossed the gap [20

(+128) 331 → 18

260 → 19

Everything seems to be working!!