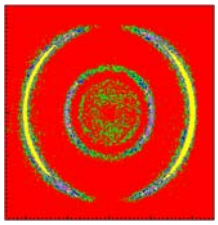
RoentDek

Handels GmbH

Supersonic Gas Jets
Detection Techniques
Data Acquisition Systems
Multifragment Imaging Systems

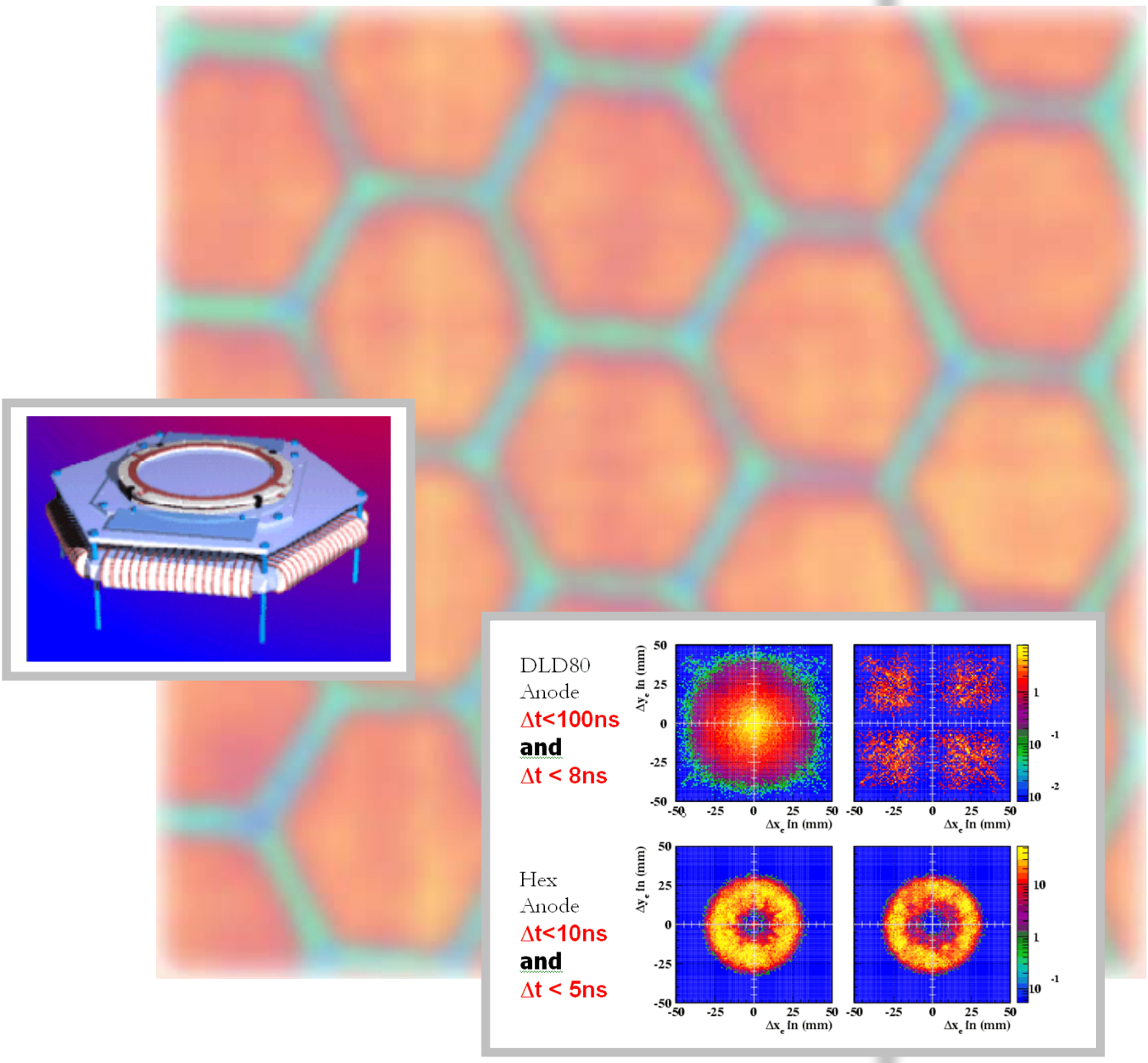


Software for the

RoentDek Hexanode

Version (2.0.3.4) for CoboldPC 2011 R5

May 2015



Mail Addresses:

Headquarter

RoentDek Handels GmbH
Im Vogelshaag 8
D-65779 Kelkheim-Ruppertshain
Germany

Frankfurt subsidiary

RoentDek Handels GmbH
c/o Institut für Kernphysik
Max-von-Laue Str. 1
D-60438 Frankfurt am Main
Germany

Author of this manual:
Achim Czasch <czasch@roentdek.com>

Web-Site:

www.roentdek.com

Table of Contents

1 General Description

You have purchased the very latest **RoentDek** delay-line detector product. This product is being further developed and your version may be a custom-made prototype adjusted to your special requirements.

The Hexagonal anode requires considerable more effort concerning the data analysis. This is rewarded with a much better multi hit performance / dead time characteristic compared to the DLD.

If you need additional info please contact **service@roentdek.com**.

This manual does not replace the main detector manual. Please read the main detector manual first.

The **RoentDek Hexanode** has a third delay-line layer that gives redundant detection opportunities either to improve the multi-hit performance or to allow the construction of a MCP setup with central hole and minimized blind detection area. In its usual version it has about 75mm redundant detection area (hexagonal) and covers about 80mm total detection area with at least two layers.
Our data acquisition software CoboldPC needs a special plug-in "Pro_DanDAq.dll" in order to handle the data which comes from the hexagonal detector. Please note: There are several DAn dll files in the distribution package of CoboldPC. Please make sure that the correct "Pro_DanDaq.dll" (or "Coltrims_DanDaq.dll" if a complete Coltrims-system was purchased) is present in the main directory of CoboldPC.
DAn is the short form of "Data Analysis". This is the part in CoboldPC where calculations are done with the data.
DAq is the short form of "Data Acquisition". This is the part which handles the communication with the hardware (usually a TDC or fast digitizing ADC). In older versions of CoboldPC the DAn and DAq was put in two separate DLLs. But in the Pro- and Coltrims-version both parts are integrated into one DLL.
This manual covers the "Pro_DanDAq.dll". The "Pro_DanDaq.dll" is almost identical compared to the "Coltrims_DanDaq.dll" but with a reduced functionality. The "Coltrims-DanDaq.dll" provides additional data analysis tools for COLTRIMS type momentum spectrometers. The 2 DLLs "Pro_DanDaq.dll" and "Coltrims_DanDaq.dll" are not included on the installation-CD of CoboldPC because special personal customer support is required for a successful application of these DLLs.
Instead CoboldPC is shipped with the standard "Dan_Standard.dll". With the help of this simple DLL CoboldPC can handle the data coming from the hexagonal detector. However the capabilities of this standard "DAn_Standard.dll" are limited to single hit measurements and it offers no functionality for the calibration of the hexagonal anode.
This manual will only cover the "Pro_DanDaq.dll" and the calibration process. The additional functionality of "Coltrims-DanDaq.dll" is discussed in a different manual. Even if you are planning to use the "Coltrims_DanDaq.dll" later you should read this manual because all functions of the Pro_DanDAq.DLL are included in Coltrims_DanDaq.DLL.
If you are planning multi hit measurements, e.g. where several particles hit the detector within a short time range of a few 100 ns an advanced reconstruction algorithm must be used which is in the file "resort64c_VS2010.dll". The DLLs "Pro_DanDaq.dll" and "Coltrims_DanDaq.dll" use this sorting algorithm. Since some customers want to use their own software the "eesort64c_Vs2010.dll" is made in such a way that it can be easily combined with existing C++ programs. In the appendix of this manual you will find a short description of the parameters of the sorting algorithm.
The advanced reconstruction routine requires a set of parameter which are unique for each hexagonal anode. The ceramic half rods which are spanning the wires of the anode have a certain ε which influences the overall propagation speed of the signals across the anode. The exact value of ε varies due to deviations during the production process of the ceramic parts. This is the reason why every anode must be calibrated individually.

The main problems related **to multi-hit measurements**:

- If several particles hit the detector within a short time range then the anode signals might be mixed up in time order.
- Additionally it can happen that some signals are not detected due to the dead time of the data acquisition system.
- Sometimes signals are not detected because they are too weak.
- At the ends of the anode wires reflections can occur. These reflections might be detected. They must be sorted out.

The reconstruction/sorting routine "resort64c_VS2010.dll" was written to solve these problems.

It is most important to understand that parameters which are not 100% correct will lead to unexpected/wrong results. The reconstruction routine tries to make sense out of any data. If the parameters are wrong then it might produce wrong results. However these wrong results might look like good data. Therefore the parameters should be chosen carefully.

2 Road map

If you have never installed and used a delay line detector from RoentDek before then the installation of the detector and the fine tuning the electronics will take some time. It is important to follow certain steps in a well defined order:

Important for Laser scientists:
For the first tests of the detector it is important **not to use laser light**. In the first test measurements only dark counts should be used. Dark counts are always generated inside the MCPs (typical rate less than 100 Hz). Dark counts have a smaller signal amplitude. Therefore they can not be used for the fine tuning of the CFD-thresholds. But they are sufficient to confirm that the detectors are operational. Do not use laser light in the first tests because this can destroy the MCPs very quickly. The first test with laser light should only be done after confirmation from RoentDek.

1) Install the detector
2) Follow the start-up procedure for new MCPs. (Please refer to the main detector manual.)
3) Check the detector signals with an oscilloscope (using dark counts only!).
4) Install the standard version of CoboldPC.
5) Use the standard version or the Pro/Coltrims-version to collect some mega bytes of LMF data with dark counts. Zip the data and send it to czasch@roentdek.de.
   Please also refer to our tutorial movies on our website http://www.roentdek.com/info/movies
6) Then we will analyze the data and recommend new parameter values for your CCF. We may also recommend other settings for the amplifier/CFD-modules.
7) Then new data must be collected and sent again.

3 Adjusting the CFD-thresholds

For the correct operation of the detector it is essential that the threshold levels of the CFDs are correctly set. If the thresholds are too low (too close to ground level) then there is an increased risk that the CFDs will trigger on noise or on secondary signals (e.g. reflections or ringing). If the thresholds are set too high then signals with small amplitudes may not get recorded. Therefore the thresholds of the CFDs must be set as low as possible but not too low. This is especially important if several particles shall be detected within a short time range (multihit).

Step 1: Signal quality:

Check the signals from the anode with an oscilloscope. It is recommended to use the following settings on the oscilloscope:

- 50 Ω input impedance
- DC coupling
- trigger on falling edge (if the signal is negative)
- 10 or 20 or 50 ns per time division
- 50 mV or 100 mV per division
- trigger mode: normal (do not use auto-level or auto-trigger)
- trigger level: start with -70 mV

The signals should have an average amplitude of several 100 mV. If the signals appear to be small try to raise the MCP voltage (but do not go beyond 2400V MCP voltage if the MCPs are new). Old MCPs can be operated at slightly higher voltages. The anode signals should have a negative polarity (if the monitor output of the ATR19 or if the non-inverting output of the FAMP is used). If the polarity seems to be inverted then the cables inside the vacuum were not correctly connected. In some cases this can be compensated by interchange the two output cables of the BA3-box (voltage for signal and reference).

After the anode signals the MCP signal must be checked. Usually the signal quality of anode signals is significantly better than the quality of MCP signals. The detector signals should have only one primary peak. For the anode signals this will be most probably the case. But the MCP signal might show ringing or reflections or a component with inverted polarity. Try to eliminate these structures by turning the potis on the signal decouplers of the two MCPs and of the holder.

Step 2: Adjusting the CFD-thresholds using the oscilloscope:

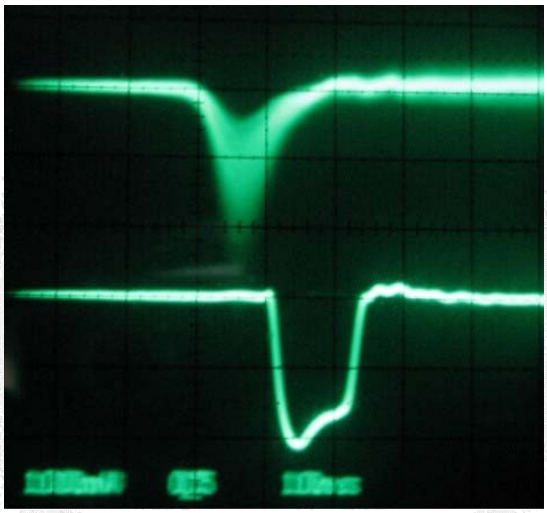
Feed the analog signal into channel 1 of the oscilloscope.

If an ATR19 is used:
Take the signals from the monitor output. The MCP signal and the anode signals should have a negative polarity.

If a FAMP is used:
Take the MCP-signal from the non-inverting and the anode signals from the inverting outputs (because the CFDs can only work with negative signals it is necessary to feed the positive signals into the oscilloscope). Therefore the signals that are visible on the oscilloscope will have a positive polarity.

Mode A: Trigger the oscilloscope with the NIM-signal:

Put the signals from the NIM-outputs into channel 2 of the oscilloscope. Set the oscilloscope trigger source to channel 2 and the trigger level to -400mV. Now the oscilloscope will trigger on the NIM-signals. That means that you will see all analog signals in channel 1 that have triggered a signal in the CFD. There will be analog signals with large amplitudes and with smaller amplitudes. If you raise the CFD-threshold you will see that there is gap between the ground level and the smallest analog signals.



The gap is visible. Here the CFD-threshold is well chosen. The lower trace shows the NIM-signal. The upper trace shows the analog signal.

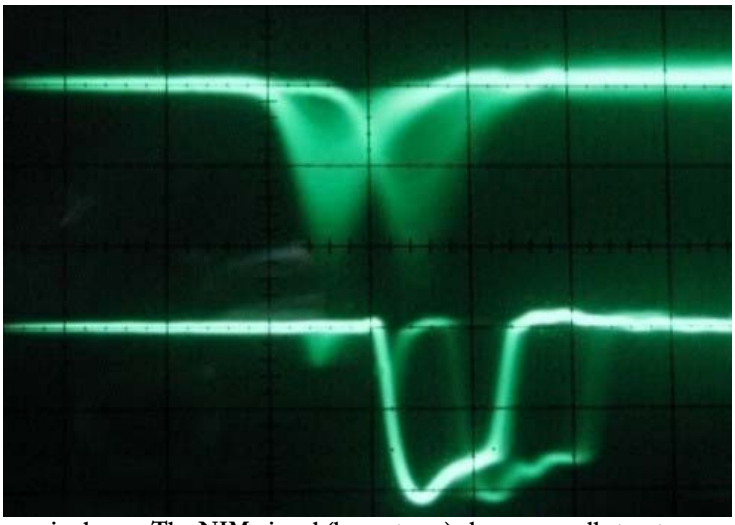
By raising the CFD-threshold more and more smaller anode signals get filtered out. If you lower the threshold again you will see that the gap becomes smaller. If the threshold is lowered too far then the CFD will start to trigger on noise. In this moment an additional line close to the ground level will appear. You might also notice that additional NIM-signals begin to appear behind the primary NIM-signal. This is a clear sign that the CFD-threshold is too low.

Mode B: Trigger the oscilloscope with the analog-signal:

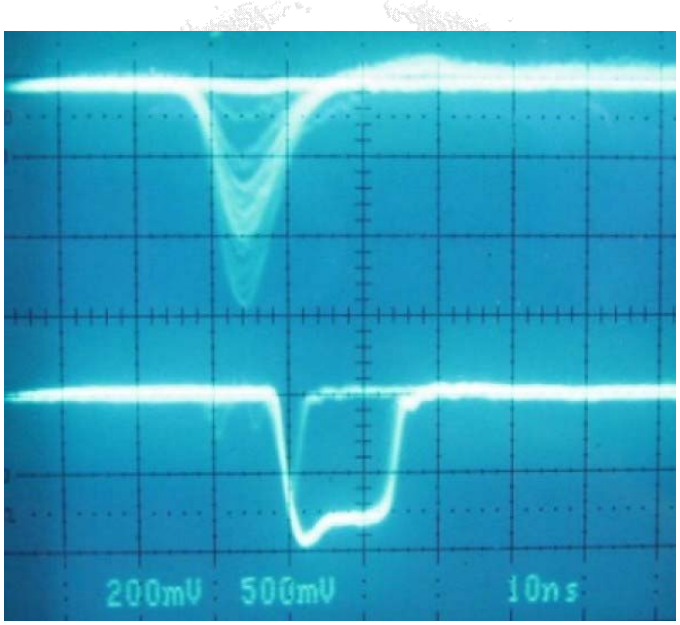
Now configure the oscilloscope so that it triggers on channel 1. Start by setting the threshold level of the oscilloscope closer to ground (e.g. 50 mV). Now tune the threshold level of the oscilloscope up and down and watch the NIM-signal. This way you can get another view on the threshold level of the CFD.

It is recommended to switch back and forth between mode A and B until a good CFD-threshold is set.

There is another reason why the CFD-thresholds should not be set too low: To avoid "pre-triggers" Please trigger the oscilloscope on the NIM-signal. Now lower the threshold of the oscilloscope to about -60 mV and watch the analog signal. If the image looks as in the screen shot below then the CFD-threshold was set too low. These pre-triggers can also be seen in the time sum plots as a small additional peak on the left of the main time sum peak.



Here the effect of pre-triggers is shown. The NIM-signal (lower trace) shows a small structure on the left of the main NIM-signal. This effect can occur if the CFD-threshold is set too low.



For this image the CFD-threshold was set too low. Pre-triggers in the NIM-signal are visible and the noise in the analog signal shows up as intense trace at ground potential.

Using the software CoboldPC for the fine adjustment of the CFD-thresholds:

The following procedure will only work if the signal quality is good (no ringing, no reflections). In this section it is assumed that the TDC8HP-card is used and that the MCP signal is plugged into channel 8 of the TDC.

Set the group range start to -300 ns (usually parameter 69). Set the group range end to +300 ns (usually parameter 70) and the trigger dead time to +310 ns (usually parameter 68). The trigger channel (usually parameter 64) should be set to the TDC channel which is connected to the MCP signal (usually TDC channel 8).

In the standard CCF of CoboldPC you will find the histogram "consistency indicator" and the histograms "n01", "n02", …, "n08".

First check the histogram "n08". This histogram shows a statistic about the number of signals that were recorded in channel 8. Under normal circumstances this histogram should show most entries in bin 1 and only very few in bin 2.

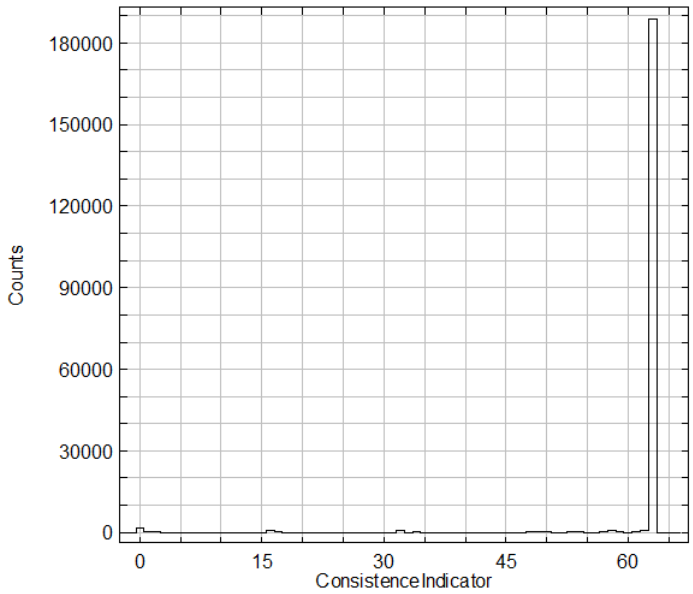
(Only if the particles are produced in Coulomb explosions then there might be a fraction of real double events in the data). If the CFD-threshold is set too low then the CFD might trigger on ringing or reflections which come after the main peak. This would result in 2 NIM-signals. Please raise the CFD-threshold until the entries in bin 2 almost disappear. But the number of entries in bin 1 should not decrease.

In the next step check the histogram "consistency indicator".
The histogram "consistency indicator" shows a variable which is calculated anew for each event.
First it is set to zero.
If a signal is detected in the TDC channel for x1 then this variable is incremented by +1.
If a signal is detected in the TDC channel for x2 then this variable is incremented by +2.
If a signal is detected in the TDC channel for y1 then this variable is incremented by +4.
If a signal is detected in the TDC channel for y2 then this variable is incremented by +8.

For HEX-detectors:
If a signal is detected in the TDC channel for z1 then this variable is incremented by +16.
If a signal is detected in the TDC channel for z2 then this variable is incremented by +32.

For a DLD-detector the result is 15 if the event is complete. "Complete" means that all anode signals were recorded. For a HEX-detector the result is 63.

If all CFD-thresholds are correctly set then this histogram will show most entries in bin 15 (63 for HEX-detector). A few counts can be in bin zero. This is ok. But there should be almost no counts in the bins ranging from 1 to 15 (or 63). If there are no counts in bin zero then the CFD-threshold of the MCP-signals may be set too high. In this case go back to histogram "n08" and lower the threshold until the counts in bin do not increase any longer and counts in bin 2 begin to appear instead.



In this data set the CFD-thresholds were correctly set.

However, it is important to understand that the histogram "consistency indicator" can only show that some CFD-thresholds are set too high.
Example: If the CFD-thresholds are set so low so that the CFDs trigger on noise then the TDC will always record signals in the channels. This would also result in 15 (or 63). But the data would contain mostly noise triggers.
To make sure that the thresholds are not set too low please check the histograms "n01", "n02" , …
These histograms should look as the histogram "n08": The number of entries in bin 2 should.

After this procedure it is important to check the time sum histograms. If there is a small peak to the left of the main peak then pre-triggers have occurred. In this case the CFD-threshold levels in the anode signal must be raised a bit. If a small peak appears to the right of the main peak then the CFD-threshold of the MCP signal must be raised.

4 The Pro_DanDaq.dll for the Hexagonal Anode

The "Pro_DanDaq.dll" for the Hexanode is as a plug-in for CoboldPC. It must be present in the same directory as CoboldPC.exe. A special CoboldPC Command File (CCF) must be used: "start_Cobold2011R5_TDC8HP-DAq-part1.ccf" if the TDC8HP TDC-card is used. This files will cause CoboldPC to execute a second file "DAN-part2.ccf". All parameters which are discussed in this manual must be defined in these "part2"-CCF-file.s

All functions of the "Pro_DanDaq.dll" are also included in "Coltrims_DanDaq.dll". Therefore this chapter is also valid for the Coltrims-version.

The Pro_DanDaq.dll can be operated in 3 different ways: (Please check the CCF for further comments on the parameters)

Simple mode: (Parameter 1123=0, parameter 1128=0)
In this mode the DAN calculates the positions in a very simple way. It does not sort signals which might be mixed up in their order in case of multi hit. This mode should be used in order to get used to the detector system. Even without calibrating any parameters you will see a raw position picture of the detector surface.

Auto-Calibration mode: (Parameter 1123=0, parameter 1128=1)
The advanced reconstruction routine needs several parameters in order to operate. Some of these parameters can be extracted automatically from the data. After the list mode file was read a pop-up window will appear with the values of some parameters. In addition a file "correction_table_i_det_part3.ccf" will be written. This procedure and the meaning of the new ccf-file will be described further below.

Advanced reconstruction/sorting mode: (Parameter 1123=1, parameter 1128=0)
In this mode the DLL will make use of the advanced sorting routine "resort64c_VS2010.dll".

It is very important to understand that all input parameters must be very accurate. Otherwise the routine might produce "ghost" hits that did not really exist. It will be explained further below how these parameters must be obtained.

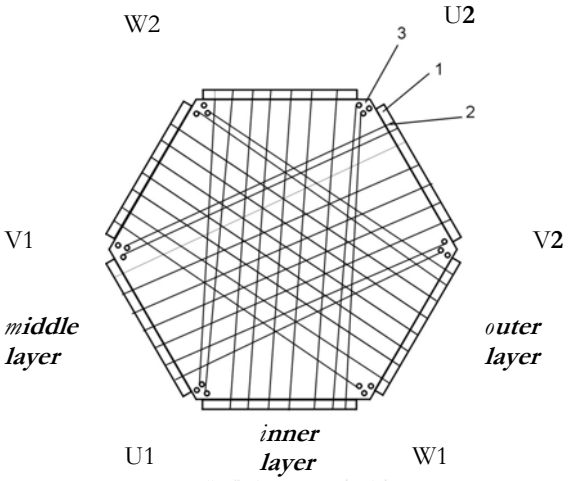
Also, the correct adjustment of the threshold levels of the constant fraction discriminators (CFD) is very important. If the threshold levels are too low then the CFD will trigger on echoes, reflections or ringing and the handling of real signals might be disturbed by internal cross-talk from other channels inside the CFD. This will decrease the quality of the acquired data and the risk of wrong reconstructed hits will be increased.

If several particles are expected to hit the detector within a short time (approx. 100ns time range) then the MCP signal should be measured in addition to the 6 anode signals. This MCP-information is important for the advanced reconstruction routine. However, the reconstruction routine can work without this additional information the MCPs but this is not recommended.

For programmers:
The source code of the Pro- or Coltrims DAN might be a bit too complicated if one tries to understand how the advanced reconstruction routine is imbedded in the DAN (in ProcessDetector() in DanUtilities.cpp). Therefore we provide simple example code which contains only the most important parts. Please contact RoentDek or Mr. Achim Czasch <czasch@roentdek.com> if you ware planning to integrate the sorting algorithm into your own data analysis software.

5 Simple Position Calculation

The software expects the anode to be connected in the following way:



Rear view of the Hexanode with recommended cable connections

The position in the hexagonal coordinate frame is represented by the differences of the arriving times from the signals in opposite corners of the anode.

$u = 0.5 \cdot (u1 - u2) \cdot f_u$ (u1,u2,v1,v2,w1 and w2 in nanoseconds)

$v = 0.5 \cdot (v1 - v2) \cdot f_v$

$w = 0.5 \cdot ((w1 - w2) + w\_offset) \cdot f_w$

If v_i is the inverse single path signal propagation speeds for a given delay line layer (they are different for each layer) then f_i is $v_i/2$. f_i must be precisely known to make the images obtained via different layer combination coherent. w_offset is an offset value that shall unify the "time difference zero center lines" of all three layers, i.e. it must be chosen so that geometrically the position lines for calculated u, v, w have a common crossing point.

Approximate values (for a HEX75) for f_i are 0.737, 0.706 and 0.684 mm/ns, from inner to outer layer (u, v, w).

"w_offset" differs from anode to anode and lies between -3 ns and +3 ns or more if the connection cables have varying length. To achieve optimal results from the software the parameters f_u, f_v, f_w and w_offset must be calibrated for each delay-line detector using acquired data. This procedure will be described further below.

The hexagonal frame can be transformed into a Cartesian coordinate system by the following equations using only two of the hexagonal coordinates respectively:

$Xuv = u \qquad Yuv = (u - 2v)/\sqrt{3}$

$Xuw = u \qquad Yuw = (2w - u)/\sqrt{3}$

$Xvw = v + w \qquad Yvw = (w - v)/\sqrt{3}$

The X and Y position have to be assembled from any combination of the above.

6 Working principles of the sorting algorithm

The algorithm searches for good combinations of anode signals and MCP signals. At the beginning it tries to find 6 anode signals and a corresponding MCP signal. In the next step it tries to find combinations of 5 anode signals and a MCP signal. After this it tries to find 6 anode signals without a MCP signal. These steps are repeated down to a combination of 3 anode signal without a MCP signal.

The following three criteria must be fulfilled by each combination:

a) the time sum for each layer must be within the tolerance window:
e.g. (u1+u2 – 2 mcp) must be close to zero

b) the position which is calculated using layer u and v must be close to the position which is calculated with layers u and w.

c) the resulting position on the detector must be within the active area of the MCP.

Missing signals are reconstructed using the equation for the time sums: 0 = u1+u2 - 2 mcp
and using the equations for the position encoding (see chapter "Simple Position calculation").
Signals which can not be used in any combination are discarded.

If a signal can be assigned to two or more combinations then the combinations begin to compete for this signal. Taking the dead time margins into account the algorithm calculates a ranking for each combination which corresponds to a "degree of plausibility". Using this ranking it tries to resolve ambiguous situations.

7 The Spectra defined in “Dan-part2.ccf”

Note: “Pro_DanDaq.dll” and “Coltrims_DanDaq.dll” can handle two detectors: The first one is always called “i_det” and the second one “e_det” (for “ion”- and “electron”-detection). However, this scheme is arbitrary and “i_det” could aswell be used for electron detection and “e_det” for ion-detection. In the spectrum-names or coordinate-names the prefix i_det or i_ indicates the first detector. The prefixes e_det or e_ indicate the second detector.

In the following sections some definitions are needed:

Cu1, Cu2, Cv1, Cv2, Cw1, Cw2 and Cmcp are the channel numbers of the TDC.

Usually they are: Cu1 = 0, Cu2 = 1, Cv1 = 2, Cv2 = 3, Cw1 = 4, Cw2 = 5, Cmcp = 6 (or Cmcp = -1 if the MCP signal is not used.)
Note: The MCP-signal here should not be mistaken as the trigger signal for the TDC.

`count[Cu1]` is the number of hits detected in channel Cu1
`count[Cu2]` is the number of hits detected in channel Cu2 …

`tdc_ns[channel][hit_number]` is the data array of the TDC.

```
time_sum_U   = (tdc_ns[Cu1][0] – tdc_ns[Cmcp][0]) + (tdc_ns[Cu2][0] – tdc_ns[Cmcp][0])
             =  tdc_ns[Cu1][0] + tdc_ns[Cu2][0] – 2 * tdc_ns[Cmcp][0]
time_sum_V   =  tdc_ns[Cv1][0] + tdc_ns[Cv2][0] – 2 * tdc_ns[Cmcp][0]
time_sum_W   =  tdc_ns[Cw1][0] + tdc_ns[Cw2][0] – 2 * tdc_ns[Cmcp][0]

u = (tdc_ns[Cu1][0] – tdc_ns[Cu2][0]) * f_u
v = (tdc_ns[Cv1][0] – tdc_ns[Cv2][0]) * f_v
w = (tdc_ns[Cw1][0] – tdc_ns[Cw2][0] + w_offset) * f_w
```

Spectrum "i_ConsistenceIndicator":
This value is calculated in the following way:

```
ConsistenceIndicator = 0;
if  (count[Cu1] > 0)  ConsistenceIndicator = ConsistenceIndicator + 1;
if  (count[Cu2] > 0)  ConsistenceIndicator = ConsistenceIndicator + 2;
if  (count[Cv1] > 0)  ConsistenceIndicator = ConsistenceIndicator + 4;
if  (count[Cv2] > 0)  ConsistenceIndicator = ConsistenceIndicator + 8;
if  (count[Cw1] > 0)  ConsistenceIndicator = ConsistenceIndicator + 16;
if  (count[Cw2] > 0)  ConsistenceIndicator = ConsistenceIndicator + 32;
```

If all signals are present then the result is 63. If some threshold levels of the constant fraction discriminators (CFD) are set too strict (too high) then other values between 0 and 63 will emerge in this spectrum.
This spectrum should show most entries at 63 (or 31 in case of a DLD) and a few counts in bin zero.

Spectrum "i_1 xy [mm]":
This spectrum is a xy-position picture of the first particle of each event.

Spectrum "i_2 xy [mm]":
This spectrum is a xy-position picture of the second particle of each event.

Spectra "i_XYuv [mm]" (and 2 more)
These spectra show the results of the three different position calculations.

```
1) Xuv   = u           Yuv   = (u -2v)/ √3
2) Xuw   = u           Yuw   = (2w -u)/ √3
3) Xvw   = v + w       Yvw   = (w -v) / √3
```

These spectra show raw data only. The reconstruction algorithm does not affect these spectra.

Spectrum "time i_sum U [ns]" (and 2 more)
These spectra show the time sums of each layer.

```
example: sum_u = tdc_ns[Cu1][0] + tdc_ns [Cu2][0] - 2*tdc_ns [Cmcp][0];
```

They must be shifted to zero by adjusting the parameters 1108, 1109 and 1110. Only one strong peak should be visible in each of these spectra. Otherwise the settings for the CFDs are wrong.

Spectrum "i_det deviation map"
This is a very useful spectrum. It is a position picture of the detector. But the colors do not encode the number of hits in each bin. Instead the colors represent the deviation between two different methods of calculating the position.

Deviation = sqrt($(Xuv-Xvw)^2 + (Yuv-Yvw)^2$)

If the scale factors (parameters 1102, 1103 and 1104) and the w_offset (parameter 1105) are correctly chosen then the picture should show an average deviation of not more than 0.6 mm.
If the scale factors are only slightly off then it will have a visible effect in this spectrum.

This spectrum can not be found in the CCF. It is automatically generated by the Dan.

Spectrum "i_det resolution map"

This spectrum shows the position resolution in mm as a function of the position.
This spectrum can not be found in the CCF. It is automatically generated by the Dan.

Spectra "i_u [ns]" (and 2 more)
In these spectra the raw values of

```
u = tdc_ns[Cu1][0] - tdc_ns[Cu2][0]
v = tdc_ns[Cv1][0] - tdc_ns[Cv2][0]
v = tdc_ns[Cw1][0] - tdc_ns[Cw2][0]
```

are displayed in logarithmic Y-scale. These spectra are needed later to determine the "runtime"-parameters.

Spectra "i_sumu vs. i_u" (and 2 more)

Here, the time sums are displayed versus the position on each layer. It is clearly visible that the time sums are a function of the positions. The advanced reconstruction routine can correct this. This will be explained in the following chapters.

Spectra "corrected i_sumu vs. i_u" (and 2 more)
These spectra are only filled if the advanced reconstruction routine is used. They are control spectra to check whether the position dependent correction of the time sums works. The distributions in these spectra should be horizontal.

Spectrum "used reconstruction method i_(Hit 1)"
This spectrum is filled if the advanced reconstruction routine is used. The reconstruction routine uses 19 different methods for the reconstruction of missing signals. Here, the method which was used to reconstruct the first hit is displayed. Please refer to chapter 10 in which the advanced reconstruction method is discussed.

Spectrum "i_reflection_u1" (and 5 more)
These spectra show reflections at the ends of the anode layers. If you see sharp spikes in this spectrum then you must raise the CFD-threshold until the spike disappears. This is not important in "single hit condition" when the particles are separated in time by more than a few 100 nanoseconds. But in typical "multi hit measurements" it is important not to have any reflections.

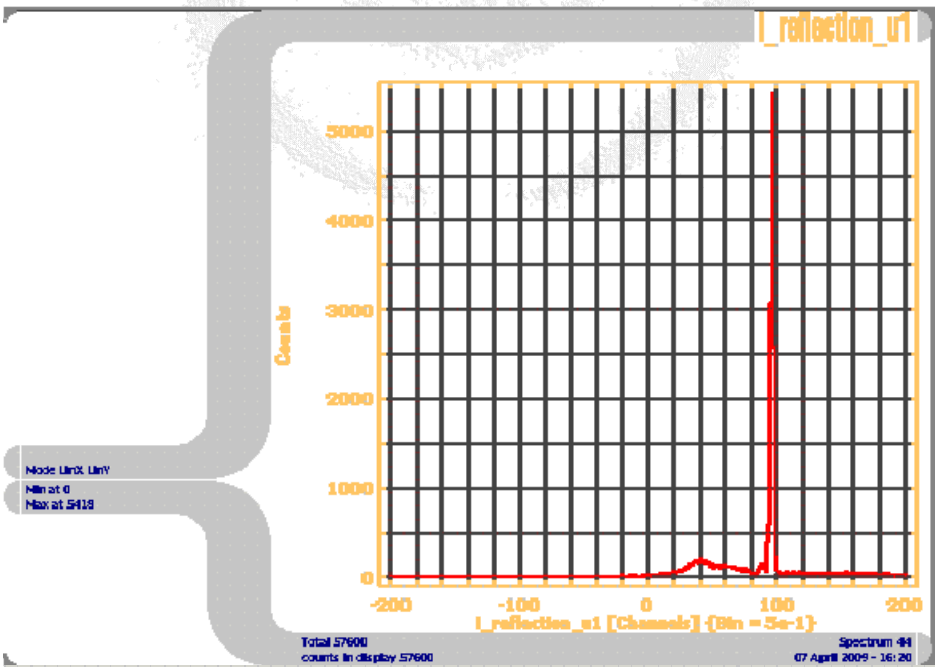


Figure 1: A typical histogram showing a intense reflection at x=94 ns.
This indicates that the CFD-threshold for channel u1 is set too low.

8 The Parameters defined in "Pro-DAN-part2.ccf"

Only a selection of parameters will be explained in the following section.

```
Parameter 1102,0.344   ; fu: Time to mm calibration factor for u (mm/ns)
Parameter 1103,0.34375 ; fv: Time to mm calibration factor for v (mm/ns)
Parameter 1104,0.39025 ; fw: Time to mm calibration factor for w (mm/ns)
```

These scale factors define the conversion from nanoseconds to mm for each layer. If the advanced reconstruction routine shall be used it is important to consider the following: The reconstruction algorithm works with some internal parameters which are in the unit of millimeters. Therefore this algorithm will not work correctly if the conversion from nanoseconds does not result in mm. The calibration of these parameters will be described in the following chapter.

```
Parameter 1105,-1.975  ; Offset for w layer (units: nanoseconds)
```

This offset shifts layer W in such a way that the centre lines of all 3 layers meet in one point. This parameter changes if different cables between the detector and the PC are used. If the cables are changed then this parameter must be calibrated anew. The calibration of this parameter will be described in the following chapter.

```
Parameter 1106,0    ; Position offset X (so that the MCP is centered around x=0
Parameter 1107,0    ; Position offset Y (so that the MCP is centered around y=0)
```

With these 2 parameters the position image can be centered around x=y=0. These parameters must be set so that the detector surface is centered – independent of how the image looks. The parameters should not be used to center the image around the zero origin of the distribution of the particles! Please see the next chapter for a detailed description.

```
Parameter 1108,-108.9      ; Offset for time sum of U
Parameter 1109,-108        ; Offset for time sum of V
Parameter 1110,-96         ; Offset for time sum of W
```

Under normal conditions the time sums of the three layers will not peak at zero. However, the auto-calibration algorithm and also the advanced reconstruction routine expect these time sums to peak at zero. This correction is done inside the Pro_DanDaq.dll before these algorithms are used. The calibration of these parameters will be described in the following chapter.

```
Parameter 1111,40.  ; radius of active MCP area in mm
```

This parameter is important if the advanced reconstruction routine is used. It must be set carefully. This will be explained in the following chapter.

```
Parameter 1115,5.   ; half width of time sum gate for layer u (units: nanoseconds)
Parameter 1116,5.   ; half width of time sum gate for layer v (units: nanoseconds)
Parameter 1117,5.   ; half width of time sum gate for layer w (units: nanoseconds)
```

Parameter 1117 is ignored if a detector with only 2 delay line layers is used (e.g. DLD40 or DLD80).
The meaning of these parameters is described in the next chapter.

```
Parameter 1118,110.  ; Maximum runtime (nanosecs) of signals across layer u
Parameter 1119,110.  ; Maximum runtime (nanosecs) of signals across layer v
Parameter 1120,110.  ; Maximum runtime (nanosecs) of signals across layer w
```

This parameter is important to the advanced reconstruction routine. It will be explained in the following chapter.

```
Parameter 1121,25.  ; Dead time of anode electronics (nanosecs) (Always use 25 ns)
Parameter 1122,25.  ; Dead time of MCP electronics (nanosecs) (Always use 25 ns)
```

These values are important for the advanced reconstruction routine.

```
Parameter 1123,0  ; Use advanced reconstruction routine (0=no, 1=yes)
```

This parameter turns on/off the advanced reconstruction routine.

```
Parameter 1124,0  ; Use position dependent correction of time sums (0=no, 1=yes)
                  ; (only needed if the advanced reconstruction routine is used)
```
The advanced reconstruction routine can reconstruct missing MCP signals by using the information from the time sums. In theory the time sums are constant. However, in reality there is a minimal dependency on the position. One must correct for this effect. Otherwise the reconstructed MCP-signals will have a time inaccuracy of up to 3 ns.

```
Parameter 1125,0  ; Use position non-linearity correction 0=no, 1=yes)
```
With a HEX anode it is possible to correct non-linearity effects in the image. This does not increase the local Position resolution. But it increases the momentum resolution because here the linearity of the position is important. The correction table that is required for this type of correction is also created during the auto-calibration procedure.

```
Parameter 1126 and 1127
These 2 corrections will be available in future versions.
```

```
Parameter 1128,0  ; do auto calibration (0=no, 1=yes) (not possible if parameter 1123=1)
```
This parameters activates the auto-calibration algorithm which calculates good values for the scale factors f_v, f_w and w_offset. The calibration procedure will be explained in the following chapter.

```
Parameter 1008,0  ; Write data back to hard drive (0=no, 1=yes ASCII, 2=yes LMF,
3=positions in text file)
```

1) Sometimes it is necessary to convert the LMF data file to a text file. This can be done if this parameter is set to 1.
2) The advanced reconstruction routine is very slow. Therefore it is possible to write a new LMF file which contains the resorted/reconstructed event data.

9 Calibration of the Hexanode

For the calibration of the Hexanode one needs a LMF data file from the detector. The events on the detector must be uniformly distributed across the entire surface of the detector. Do not put a mask in front of the detector. Each mm² of the surface must have been hit at least 3 times. A calibration of a detector with a MCP of 50 mm diameter needs a LMF file of at least 150.000 events (80mm MCP needs about 400.000 events and 120mm MCP needs about 900.000 events).

In the following sections the calibration procedure will be explained using the example file "sample_TDC8HP_HEX80_8ch2hit.lmf". Please contact RoentDek if you want to use this file for training.

It is important to follow these steps in exactly the same order. Otherwise some parameters will not be calibrated correctly.

Step 1)
Set the following parameters in the CCF-file:

```
Parameter 1106,0.0  ; Position offset X
Parameter 1107,0.0  ; Position offset Y
Parameter 1112,0.0  ; Rotation Offset Center for x
Parameter 1113,0.0  ; Rotation Offset Center for y
Parameter 1114,0.0  ; Rotation Angle (in mathematical direction) (unit: RAD)
Parameter 1105,0.0  ; Offset for w layer (units: nanoseconds)
Parameter 1108,0.0  ; Offset for time sum of U
Parameter 1109,0.0  ; Offset for time sum of V
Parameter 1110,0.0  ; Offset for time sum of W
Parameter 1123,0  ; Use advanced reconstruction routine (0=no, 1=yes)
Parameter 1128,0  ; do auto calibration (0=no, 1=yes)
Parameter 1008,0  ; Write data back to hard drive (0=no, 1=yes ASCII, 2=yes LMF)
```

Step 2) adjusting the time sums
Process the LMF-file.

Go to the spectra for "time i_sum U [ns]", "time i_sum V [ns]" and "time i_sum W [ns]". Determine the position of the time sums. (use the CoboldPC commands "expand" and "fit g"). Now change the sign of these value because the time sums are corrected by adding the following parameters.

Result:

```
Parameter 1108,-108.9      ; Offset for time sum of U
Parameter 1109,-108        ; Offset for time sum of V
Parameter 1110,-96         ; Offset for time sum of W
```

Step 3) adjusting the "runtime"-parameters
Go to the spectra "i_u [ns]", "i_v [ns]" and "i_w [ns]". Make sure that the Y-axis is set to logarithmic scaling. Determine the left and right edges of the main distributions. (use the CoboldPC command "cursor")

Example: u_ns: -107 and +108, v_ns: -110 and +109, w_ns: -99 and +94

Chose the maximum absolute values without sign. (this example: 108, 110 and 99)
This value is called the "runtime" for each layer.

Now set in the CCF-file:

```
Parameter 1118,108.; Maximum runtime (nanoseconds) of signals across layer u
Parameter 1119,110.; Maximum runtime (nanoseconds) of signals across layer v
Parameter 1120, 99.; Maximum runtime (nanoseconds) of signals across layer w
```

For simplification you can set runtime_v and runtime_w to zero. In this case the value of runtime_u will be used for all delay line layers.

Step 4) setting of preliminary scale factors
Set in the CCF-file:

```
Parameter 1102,0.3  ; Time to mm calibration factor for u (mm/ns)
Parameter 1103,0.3  ; Time to mm calibration factor for v (mm/ns)
Parameter 1104,0.3  ; Time to mm calibration factor for w (mm/ns)
Parameter 1105,0.   ; Offset for w layer (units: nanoseconds)
```

Then process the data file again. Look at the spectra XYuv, XYuw. These are position pictures of the detector calculated with different methods. Now the 3 scale factors must be adjusted in such a way that the size of these pictures is close to the size of the active area of the MCP. In this example the diameter of the active are is 38 mm. Therefore the scale factors must be set to

```
Parameter 1102,0.344  ; Time to mm calibration factor for u (mm/ns)
Parameter 1103,0.344  ; Time to mm calibration factor for v (mm/ns)
Parameter 1104,0.344  ; Time to mm calibration factor for w (mm/ns)
```

At this stage there is no need to do this adjustment very precisely.

Step 5) auto calibration of the scale factors and of w_offset
Set in the CCF-file:

```
Parameter 1128,1  ; do auto calibration (0=no, 1=yes)
```

Process the data file again.

A pop-up window will appear. It contains the results for the scale factors f_v and f_w and for w_offset.

These values must be written into the CCF-file. Then the data file must be processed again.
This procedure must be repeated until the values do not change any more.
In this example this happens after the third run.

Final result:

```
Parameter 1102,0.344   ; Time to mm calibration factor for U (mm/ns)
Parameter 1103,0.34375 ; Time to mm calibration factor for v (mm/ns)
Parameter 1104,0.39025 ; Time to mm calibration factor for w (mm/ns)
Parameter 1105,-1.975  ; Offset for w layer (units: nanoseconds)
```

Step 6) turn off auto the calibration
Set in the CCF-file:

```
Parameter 1128,0  ; do auto calibration (0=no, 1=yes)
```

Step 7) final correction of the scale factors
Process the LMF data file again.

Since the scale factors have changed during step 5 we have to check whether the size of the position pictures XYuv and XYuw still corresponds to the real size of the active MCP area.

In our example no change is needed.

Step 8) Adjustment of the center position of the position picture Xuv
Process the LMF data file again.

Please look at spectrum XYuv. Determine the center of the distribution. (use the CoboldPC commands "expand" and "cursor")
In this example it appears to be at X = -1 and Y = 0. Therefore an offset of +1 must be applied in X-direction.

Set the parameters in the CCF-file:

```
Parameter 1106,1.0  ; Position offset X
Parameter 1107,0.0  ; Position offset Y
```

Important: Do not use these parameters to center the measured particle distributions of the measurement (e.g. center of an ion beam). The parameter 1106 and 1107 should only be used to center the central point of the MCPs.

Step 9) rotating the position pictures
Process the data file again.

The position pictures 4 and 5 can be rotated. With parameters 1112, 1113 and 1114 the rotation angle and the center of the rotation can be set.

The following instructions must be followed only if the advanced reconstruction routine will be used:

Step 10) check the value "runtime" again (parameters 1118, 1119, 1120)
Since the parameters 1106 and 1107 have been changed in step 8 the value for "runtime" (parameters 1118, 1119, 1120) might have been changed too.

Process the data file again.

Repeat step 3.

Step 11) adjust the scale factors again
Repeat the auto-calibration (steps 5 and 6). Of course these parameters can not have changed much.

Step 12) set the parameter "radius" (parameter 1111)
Look at spectrum XYuv. Determine the minimum radius of an imaginary circle that includes the distribution. This value should be chosen slightly larger to ensure that the entire MCP area is inside this circle.

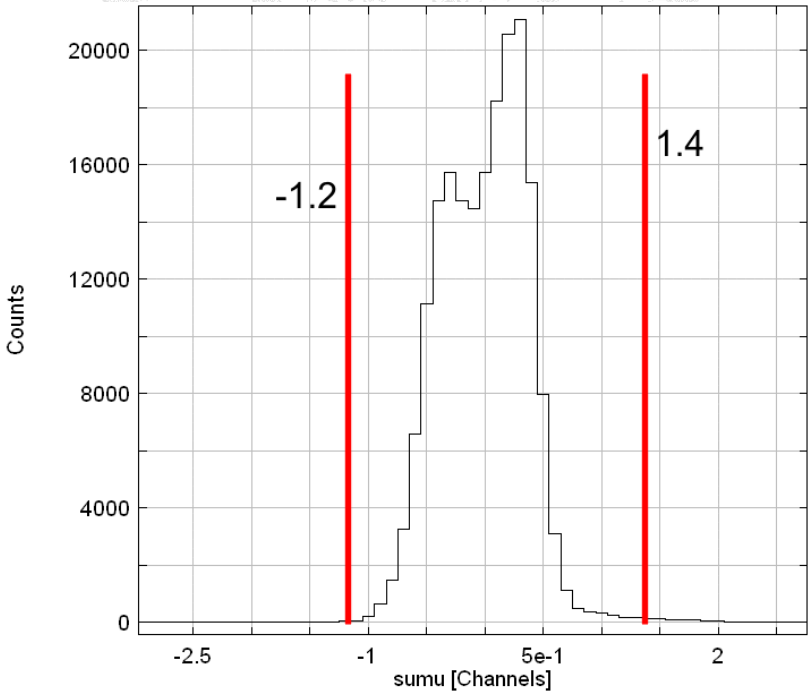
In this example this would be 40mm.

In the CCF-file set:

```
Parameter 1111,40 ; radius of active MCP area in mm
```

It is important not to choose a too small value. The advanced reconstruction routine needs this parameter to be exact. If a hit is reconstructed outside this circle then the advanced reconstruction routine will know that this set of anode signals is not a valid group. In this case it will try to find better combinations. Therefore this parameter is not simple a cut-off radius.

Step 13) the widths of the time sums
The advanced reconstruction routine needs more information about the time sums.
Got to the spectra "time sum U [ns]", "time sum V [ns]" and "time sum W [ns]" and determine the boundaries of the peaks at the base.



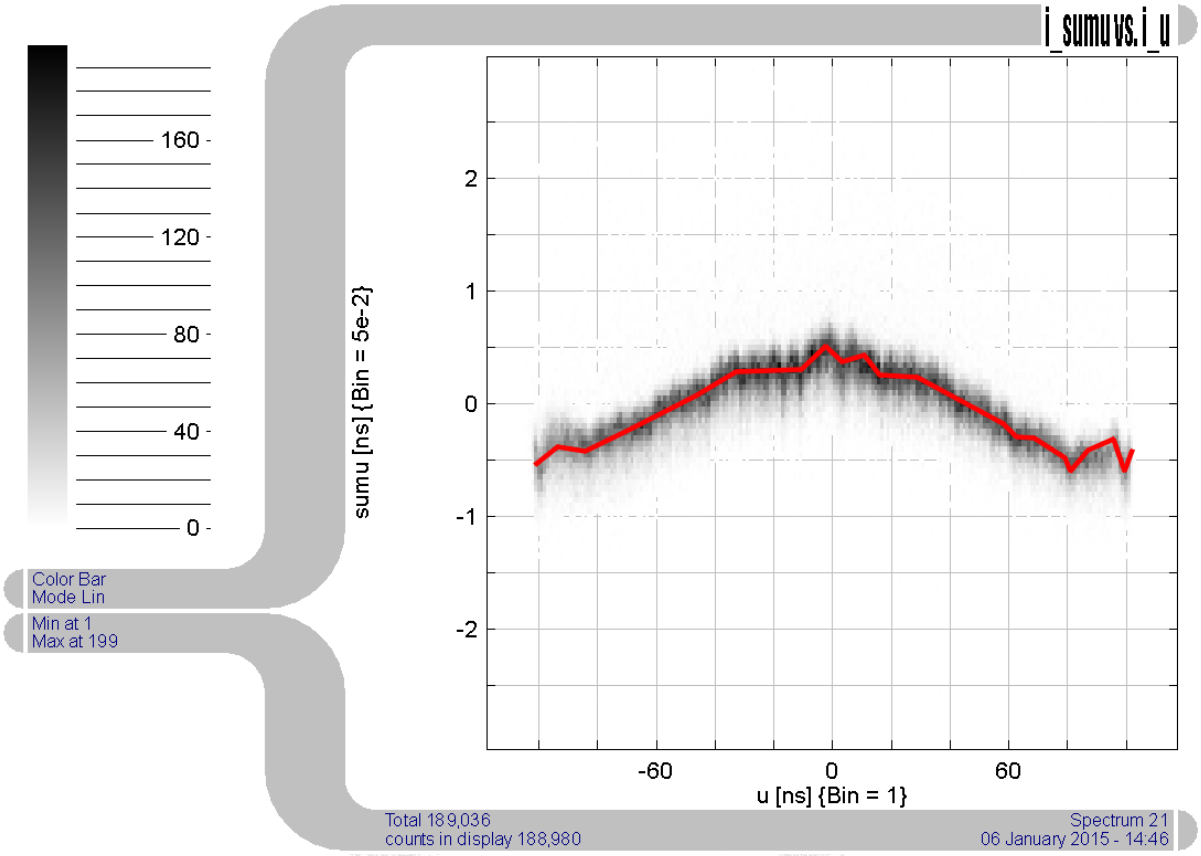
Example: time sum U -1.2 to +1.4, time sum V -1.1 to +1.2 and time sum W -0.8 to +1.3

Now the largest ones of the absolute values must be chosen on each layer.

Set in the CCF-file:

```
Parameter 1115,1.4 ; half width of time sum gate for layer u (units: nanoseconds)
Parameter 1116,1.2 ; half width of time sum gate for layer v (units: nanoseconds)
Parameter 1117,1.3 ; half width of time sum gate for layer w (units: nanoseconds)
```

Step 14) settings for the position dependent correction of the time sums
Look at the spectra "i_sumu vs. i_u", "i_sumv vs. i_v" and "i_sumw vs. i_w".



Example for layer U

The time sum on each layer is a function of the position on this layer. The advanced reconstruction routine can correct this. By doing this the program can reconstruct missing MCP signals with a better timing precision. In order to do this correction the DAn must be given a list of interpolation points. Between these points the program will do a linear interpolation (indicated as a red line in the picture above). The coordinates of these points must be included in the CCF-file using the following syntax:

First the correction table must be initialized using the command

```
DanModule,reset_i_det_correction_table
```

```
Then the correction points are set using this syntax:
```

```
DanModule,set_correction_point,sum,i_det,0,-90,-0.05
…
…
```

Description: After "set_correction_point" there are 4 arguments.
- "sum" means: This is a correction point for the sum correction. "pos" would cause a correction of the position calculation. This is used by the build-in non-linearity correction which can be used with the detectors HEX40, HEX75 and HEX120.
- "i_det" This argument is explained in the following paragraphs

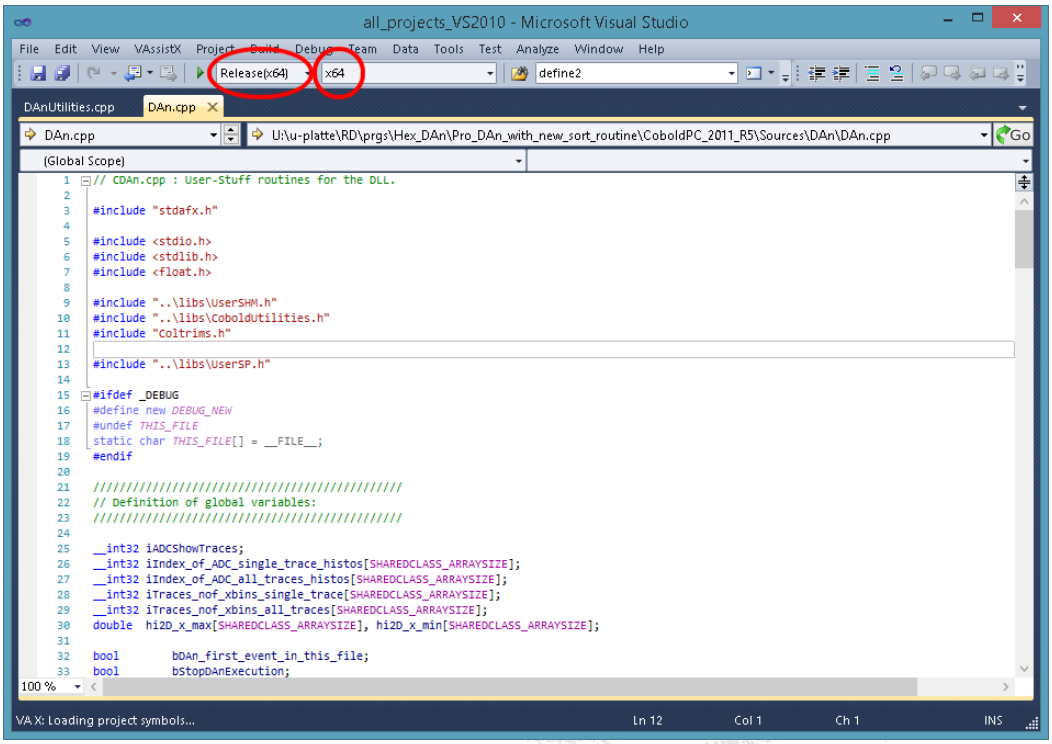
- the new 2 arguments are the position (in units of nanoseconds) and the correction value (also in ns)

But this has not to be done manully. This list of parameters should be placed into the file “correction_table_i_det_part3.ccf”.
Note: This file will be automatically generated if the auto-calibration feature of the DAN is used.
It will be placed into the same folder where the LMF data file is located. Then it must be manually copied into the CCF-folder.

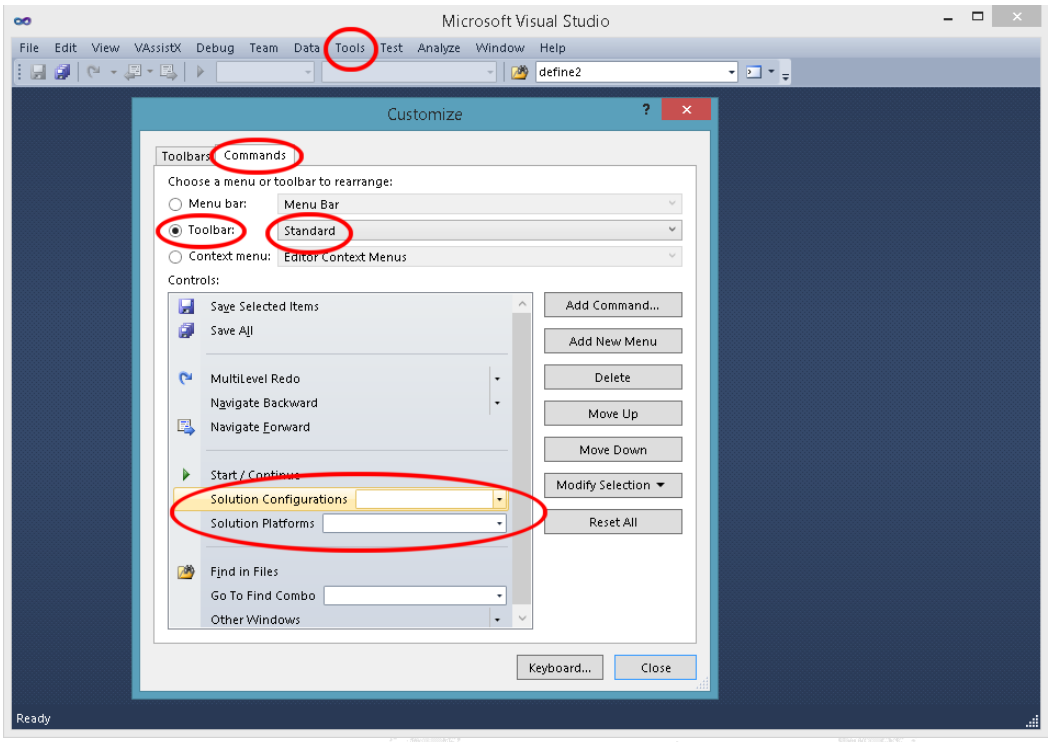
The spectra "corrected i_sumu vs. i_u", "corrected i_sumv vs. i_v" and "corrected i_sumw vs. i_w" are control spectra. Here the result of the time sum correction is shown. These spectra are filled by the reconstruction routine. Therefore parameter 1123 must be set to 1 (this activates the reconstruction routine).

10 Modifying the source code of the data analysis (DAn)

The data analysis in CoboldPC is not script based because this would be too slow. Instead it is directly programmed using the C++ programming language. For this the compiler Visual Studio VS2010 (or any more recent version) is needed. The free "express edition" does not contain some required libraries and can not be used.
Start the compiler and open the solution file "\sources\all_projects_VS2010.sln".
This will open all compiler projects. Each project will produce one DLL.
(Do not open the .vcxproj-files which are located in the sub-folders. Always open the file "all_projects_VS2010.sln")
All functions of the data analysis are contained in the file "Dan.cpp" in the project "Pro_DanDAq" or "Coltrims_DanDaq".
Please only do modifications in this file.
If you are working on a 32bit Windows system then please make sure that the configuration "Release" and the platform "Win32" are selected.
On a 64bit system it must be set to "Release (x64)" and platform "x64". (see the image below for a 64bit system)



Sometimes these fields are not visible in the toolbar of the compiler. In this case they must be included in the toolbar by using the menu Tools/Customize/Commands/Toolbar/Standard. From there add the commands "Solution Configuration" and "Solution Platforms" to the compiler toolbar. (see the image below)



After the compile/build process the DLLs will be automatically placed at the correct position (in the same folder where CoboldPC.exe is in).

All calculations of momenta/angles/energies should be placed inside the function "UserAnalysis()" in the file "DAn.cpp". In this function you will find more information in the comments. Simple example code shows how the x,y,t-information for each particle can be accessed and how the results can be transferred to the list of coordinates so that they can be displayed in the histograms in CoboldPC.

Data flow inside CoboldPC:

The following steps are repeated for each trigger event on the TDC or ADC card:

1) CoboldPC enters the DAq.

2) The DAq communicates with the TDC (or ADC) and retrieves a new data block.

3) The DAq writes the data into the LMF data file (if this was specified at the beginning of the data acquisition) and the DAq writes the raw time values into the upper section of the Coordinates list in the CCF. The list of Coordinates is divided into 2 parts: The upper part contains the raw time values. The lower part will contain the results (positions, TOFs, momenta, …). You will notice that only the lower half of the list of Coodinates is explicitly defined in the file "DAn-part2.ccf". The upper part of the Coordinates is define in the file "start_Cobold2011R5_TDC8HP-DAq-part1.ccf" with this line:
UserFCall,SetDAqCoordinates,ch??n,ch??s??;
This line will create the exact number of required Coordinates for the raw data. (For this it uses the information from Parameters 32 and 33).

4) CoboldPC steps out of the DAq and enters the DAn.

5) The DAn does some raw data processing inside the function AnalysisProcessEvent() (in DanUtilities.cpp) and in the function ProcessDetector() (also in DanUtilities.cpp). Here it calculates the x,y,t-information for each particle and writes this information into the lower half of the list of Coordinates (which is defined in the file "Dan-part2.ccf").

6) Then CoboldPC enters the function UserAnalysis() (which located in the file Dan.cpp). Here you can do calculations with the x,y,t-information of the particles. The results are then exported into the list of Coordinates. See the 2 variables example_value1 and example_value2 for details.

7) CoboldPC steps out of the DAn.

8) CoboldPC fills all histograms with the values from the list of Coordinates.

9) CoboldPC starts again for the next event with step 1.

Notes:

- The advanced sorting algorithm is used inside the function ProcessDetector().

- The functions ProcessDetector() and AnalysisProcessEvent() should not be modified. Please only modify the functions in the file "DAn.cpp". This makes it easier to upgrade the analysis code to newer version of CoboldPC.

- The Coltrims-Version of the DAn differs mostly from the Pro-version in the function UserAnalysis(). These functions will be explained during direct email communication with the customer.

11 Trigger modes

For the first detector tests the TDC should be triggered with the MCP signal.

Then later in the real measurements there are two choices:

Choice A) Trigger the TDC with the laser signal:

You can trigger the TDC with the laser signal. But this must be a NIM-signal.

If the laser signal comes from a photo diode:
The signal from the photo diode must be converted to a NIM-signal.
NIM is defined as LOW = 0V and HIGH = -1000mV at 50 Ohms.

The ATR is not made for this task. But if the signal from the photo diode is small enough (and not too narrow) then it could work with the ATR.
The ATR expects signals that have an amplitude of less than 30 mV. If the signals from the photo diode have an amplitude of several 100 mV then you could use one free channel of the CFD8 directly. (The analog signal must be negative.)
You must use a very short delay cable because the signals from photo diodes are usually very narrow.

If the signal is a TTL-signal:
TTL-signals are 0V/+5V or 0V/3.3V high. The TDC can not work with TTL signals. Therefore you must convert it to a NIM-signal. NIM is defined as LOW = 0V and HIGH = -1000mV at 50 Ohms.
Triggering the TDC with the laser signal has one advantage and one disadvantage:
advantage: It is easy to understand. Therefore this is the "beginners mode".
disadvantage: The LMF data files get a bit bigger than necessary because the file will also contain all the laser shots in which no ions were made.

It is recommended to tune the laser intensity and/or the target density so that only 10% of all laser shots make a reaction. Then the probability of random double reactions is only 1%. Random double events are bad because you can not tell which electrons belonged to the ions. This means that only 10% of all events in the data file will contain real data. But this does not

mean that the LMF will be 10 times larger than necessary. Empty events don't use much space. Therefore the difference is not very large.

For this mode you must edit the following parameters in the CCF:

```
Parameter 68,30010      ; TriggerDeadTime, time in ns
Parameter 69,-300       ; GroupRangeStart, time in ns
Parameter 70,30000      ; GroupRangeEnd, time in ns
```

This means: The TDC will collect data starting 300 ns before the laser signal. And it will collect data for up to 30 µs after the trigger. If you think that the ions can have a longer TOF then you must use a larger value. Parameter 68 must always be 10 ns larger than parameter 70.

Choice B) Trigger the TDC with the ion MCP signal:

This is the more advanced mode. It will make smaller LMF data files because no empty events will be written to the LMF data file. The TDC will be triggered with the first ion signal. This mode has no disadvantages. But it is less intuitive to understand. For this mode you must edit the following parameters in the CCF:

```
Parameter 68,20010      ; TriggerDeadTime, time in ns
Parameter 69,-30000     ; GroupRangeStart, time in ns
Parameter 70,20000      ; GroupRangeEnd, time in ns
```

This means: The TDC will collect data starting 30000 ns before the 1st ion. And it will collect data for up to 20 µs after the 1st ion. Then the TDC will look back in time (in a short term memory) and record the laser signal and the signals from the electron detector.

In the first measurements you should use mode A (trigger with laser signal)

12 Appendix:

13 Description of the advanced reconstruction routine (sort_class)

This chapter is a reference for programmers who want to integrate the sorting algorithm into their own software. User of CoboldPC can skip this chapter.
The advanced reconstruction routine is provided as a dynamic link library (resort64c_VS2010.dll). It can be linked to any C++ program. This library is also available for the GNU-Compiler (gcc) for Linux. The sorting algorithm can be used for hexagonal detectors (e.g. HEX75) and the conventional detectors with only 2 layers (e.g. DLD40). RoentDek can provide example source code which serves as a good starting point for the understanding/integration of this routine.
First an instance of the sort_class must be constructed:

```
sort_class * hex_sorter = new sort_class();
```

then a list of parameters must be initialized:

Most of the parameters are discussed in chapter 6.

```
sorter->fu              [type double]
sorter->fv              [type double]
sorter->fw              [type double]
```

These are the 3 scalefactors

sorter->common_start_mode [type bool]
Either "true" or "false". If the data was acquired with the TDC8HP always use "true".

sorter->use_HEX [type bool]
"true" if a hexagonal anode is used (e.g. HEX75). "false" if a squared anode with only 2 delay line layers is used (e.g. DLD80)

sorter->use_MCP [type bool]
"true" if the sorting algorithm should use information of the MCP channel.

sorter->MCP_radius [type double]
The radius of the MCP in mm. Always chose a number that is about 3 mm larger.

sorter->uncorrected_time_sum_half_width_u [type double]
sorter->uncorrected_time_sum_half_width_v [type double]
sorter->uncorrected_time_sum_half_width_w [type double]
The half width at the base of the time sum peak

sorter->Cu1 channel number of signal u1 (the first channel has index 0)
sorter->Cu2
sorter->Cv1
sorter->Cv2
sorter->Cw1
sorter->Cw2
sorter->Cmcp

sorter->runtime_u [type double]
sorter->runtime_v [type double]
sorter->runtime_w [type double]
These parameters are explained in chapter 6 step 3.

sorter->dead_time_anode [type double]
sorter->dead_time_mcp [type double]
The dead time of the CFDs. Always positive numbers.

sorter->use_sum_correction [type bool]
"true" if the sorting algorithm should use the position dependent correction of the time sums.
With these functions you can set interpolation points for the sum correction:
sorter->sum_corrector_U->set_point(x,y);
sorter->sum_corrector_V->set_point(x,y);
sorter->sum_corrector_W->set_point(x,y);

sorter->tdc_array_row_length [type int]
The size of the second dimension of the TDC array.

The raw data from the TDC must be stored in 2 arrays:

double tdc_ns[number_of_channels][maximum_number_of_hits]
and
int count[number_of_channels]

tdc_ns[][] contains the measured TDC times already converted to nanoseconds.
count[] contains the number of measured signals for each channel.

sorter->tdc_pointer = (double*)tdc_ns; [type double *]
Pointer to the first entry of the TCD array. The values in this array must be in units of nanoseconds.

sorter->count [type int*]

Pointer to the array that contains the number of hits for each channel.

After all parameters have been set you must call the function
sorter->init();
once. Now everything is prepared for the sorting procedure

For each event you must process the sorting algorithm with this command:
n = sorter->sort() (this sorts the data. "n" is the number of reconstructed particles.)

Preparation of the raw data

Important: Before the command n = sorter->sort() can be executed the raw data in tdc_ns[][] must be prepared.
This preparation is done in 3 steps:

step a)
The 6 anode signals must be shifted so that the time sums peak at zero.
The time sums are defined as sum_u = tdc_ns[Cu1][0] + tdc_ns[Cu2][0] – 2 * tdc_ns[Cmcp][0];
Please see chapter 7 step 2.

The shifting of the time sums is done with the function:
sorter->shift_sums(direction, Sumu_offset, Sumv_offset, Sumw_offset); // in case of a HEX-detector
or
sorter->shift_sums(direction, Sumu_offset, Sumv_offset); // in case of a DLD-detector

the parameter direction should always be +1.

Example:
The time sums in the example data file peak at 108.9 ns, 108.0 ns and 96.0 ns.
The sums can be shifted with the command:
sorter->shift_sums(+1, -108.9, -108., -96.);

step b)
In chapter 4 the parameter w_offset was discussed. The function
sorter->shift_layer_w(direction, w_offset); // shifts layer w so that all center lines of the layers meet in one point
w_offset is in units of nanoseconds
direction should always be +1;

step c)
Please read step 8 in chapter 7.
With the function
sorter->shift_position_origin(direction, x_pos_offset, y_pos_offset);
the position picture must be centered to zero/zero.
direction should always be +1.

After the command n = sorter->sort() has been executed the results can be found in
the arrays tdc_ns[][] and count[] which now contain sorted/reconstructed/filtered data
and in the arrays
sorter->output_hit_array[]->x (x position in millimeters)
sorter->output_hit_array[]->y (y position in millimeters)
sorter->output_hit_array[]->time (MCP time in nanoseconds)
sorter->output_hit_array[]->used_method (the reconstruction method that was used for this hit)

The reconstruction routine uses 19 different methods of reconstruction.

Examples: (please refer to list below)

Method 0: All 6 anode signals and a corresponding MCP-signal were found.

Method 4: One signal on layer U is missing. All other signals for this event were found.

Both methods are very safe because several redundancy checks can be applied: The time sums must be correct and the position Xuv/Yuv and Xvw/Yvw must be identical.

Method 18: Only one anode signal on each layer was found and the MCP signal is missing. There is no way how the program could check whether this combination of signals is valid. Therefore these cases are marked red.

reconstruction method N°.	U V W MCP
0	2+2+2 + 1
1	0+2+2 + 1
2	2+0+2 + 1
3	2+2+0 + 1
4	1+2+2 + 1 (2 permutations)
5	2+1+2 + 1 (2 permutations)
6	2+2+1 + 1 (2 permutations)
7	2+2+2 + 0
8	0+2+2 + 0
9	2+0+2 + 0
10	2+2+0 + 0
11	1+2+2 + 0 (2 permutations)
12	2+1+2 + 0 (2 permutations)
13	2+2+1 + 0 (2 permutations)
14	2+1+1 + 1 1+2+1 + 1 1+1+2 + 1 (12 permutations)
15	2+1+0 + 1 2+0+1 + 1 1+2+0 + 1 (12 permutations) 1+0+2 + 1 0+2+1 + 1 0+1+2 + 1
16	1+1+1 + 1 (8 permutations)
17	2+1+1 + 0 1+2+1 + 0 1+1+2 + 0 (12 permutations)
18	1+1+1 + 0 (8 permutations)
19	2+1+0 + 0 2+0+1 + 0 1+2+0 + 0 (12 permutations) 1+0+2 + 0 0+1+2 + 0 0+2+1 + 0

The red methods are risky because no time sums can be checked and also the position check can not be used. If the scale factors and/or w_offset are not correct then the number of events which were reconstructed with the risky methods will increase. They will be most likely "ghost hits" which do not correspond to real impacts on the detector.

14 Description of the calibration functions in resort64c.dll

This chapter is a reference for programmers who want to integrate the sorting algorithm into their own software. User of CoboldPC can skip this chapter.

a) Calibrating the parameters f_u, f_v, f_w and w_offset (only for HEX-detectors)
Most of the parameters which are required by the sorting algorithm can be extracted from measured data simply by looking at the correct histograms (e.g. sum offsets, runtime, MCP radius ...)
But the parameters f_u, f_v, f_w and w_offset can not be made visible directly. For this reason there is the class "scalefactors_calibration_class" embedded in resort64c.dll. This class requires that the anode signals are already shifted so that the sums peak at zero and the detector is centered (layer w can not be shifted because w_offset is not yet know in this stage of the calibration process).

The construction of this class must happen right before sorter->init() is accessed.
The construction is done by accessing a function in the sorter-instance:

```
sorter->create_scalefactors_calibrator(BuildStack, sorter->runtime_u, sorter->runtime_v, sorter->runtime_w, 0.78, sorter->fu, sorter->fv, sorter->fw);
```

BuildStack This bool-parameter should always be "true"

The following is an example for the calibration process of a typical HEX75 detector.

Step a.1)
An instance of sort_class "sorter" must exist.

```
sorter->fu = 0.344;
sorter->fv = 0.344;
sorter->fw = 0.344;
double w_offset  = 0.;
sorter->runtime_u = 108.;
sorter->runtime_v = 110.;
sorter->runtime_w = 99.;
double sumu_offset = -108.9;
double sumv_offset = -108.;
double sumw_offset = -96.;
double pos_offset_x = 1.;
double pos_offset_y = 0.;

sorter->create_scalefactors_calibrator(true, sorter->runtime_u, sorter->runtime_v, sorter->runtime_w, 0.78, sorter->fu, sorter->fv, sorter->fw);

sorter->init();

// Now for each event the following code must be executed:

while (true) {
       get new event data into tdc_ns[][] and count[]

       sorter->shift_sums(+1, sumu_offset, sumv_offset, sumw_offset);
       sorter->shift_layer_w(+1,w_offset);
       sorter->shift_position_origin(+1, pos_offset_x, pos_offset_y);
       double sumu_ns = tdc_ns[Cu1][0] + tdc_ns[Cu2][0] - 2. * tdc_ns[Cmcp][0];
       double sumv_ns = tdc_ns[Cv1][0] + tdc_ns[Cv2][0] - 2. * tdc_ns[Cmcp][0];
       double sumw_ns = tdc_ns[Cw1][0] + tdc_ns[Cw2][0] - 2. * tdc_ns[Cmcp][0];
       double u_ns = tdc_ns[Cu1][0] - tdc_ns[Cu2][0];
       double v_ns = tdc_ns[Cv1][0] - tdc_ns[Cv2][0];
       double w_ns = tdc_ns[Cw1][0] - tdc_ns[Cw2][0];
       if (count[Cmcp]>0) {
       if (count[Cu1]>0 && count[Cu2]>0) {
       if (count[Cv1]>0 && count[Cv2]>0) {
```

```
        if (count[Cw1]>0 && count[Cw2]>0) {
                if (fabs(sumu_ns)<10. && fabs(sumv_ns)<10. && fabs(sumw_ns)<10.) {
                        calib->feed_calibration_data(true, w_offset);
                }
        }}}}

        if (this_was_the_last_event) break;
}


// After all events have been processed the following commands must be executed:

calib->do_auto_calibration();

printf("Good scalefactors (f_U is fixed) are:\nf_V =%lf\nf_W =%lf\nOffset on layer W =%lf\n",
                calib->best_fv,
                calib->best_fw,
                calib->best_w_offset);
```

Step a.2)
Now the same code must be processed again with the same data. But the new values for the parameters f_v, f_w and w_offset must be used:

The final result is:

```
sorter->fv = 0.4375;
sorter->fw = 0.39025;
double w_offset  = -1.975;
```