admesy
ADVANCED MEASUREMENT SYSTEMS

How to measure flicker

Measuring flicker

Flicker can occur in all sources that emit light.
Flicker measurements are commonly used in the LCD and lighting industry.

In and LCD Flicker is a phenomenon that is caused by Vcom offset resulting usually in a frequency of half the frame rate and usually is visible by the human eye.

In lighting we can differentiate high frequency flicker or stroboscopic effects and low frequency flicker often caused by power supplies.

Flicker outside the human visible range is not well defined by the current standard because since it's not visible, humans can not see flicker. However, some of the methods shown in this document treat high frequency flicker equally to low (<60Hz) flicker.

Flicker needs to be measured using a human vision corrected sensor (CIE1931 luminance response). The signal is captured using a high sample rate and usually a low pass filter is applied in order to not count high frequencies to the measured flicker value.

Currently a number of methods to calculate flicker are defined. All have their advantages and disadvantages, which are discussed in this document.

Flicker images

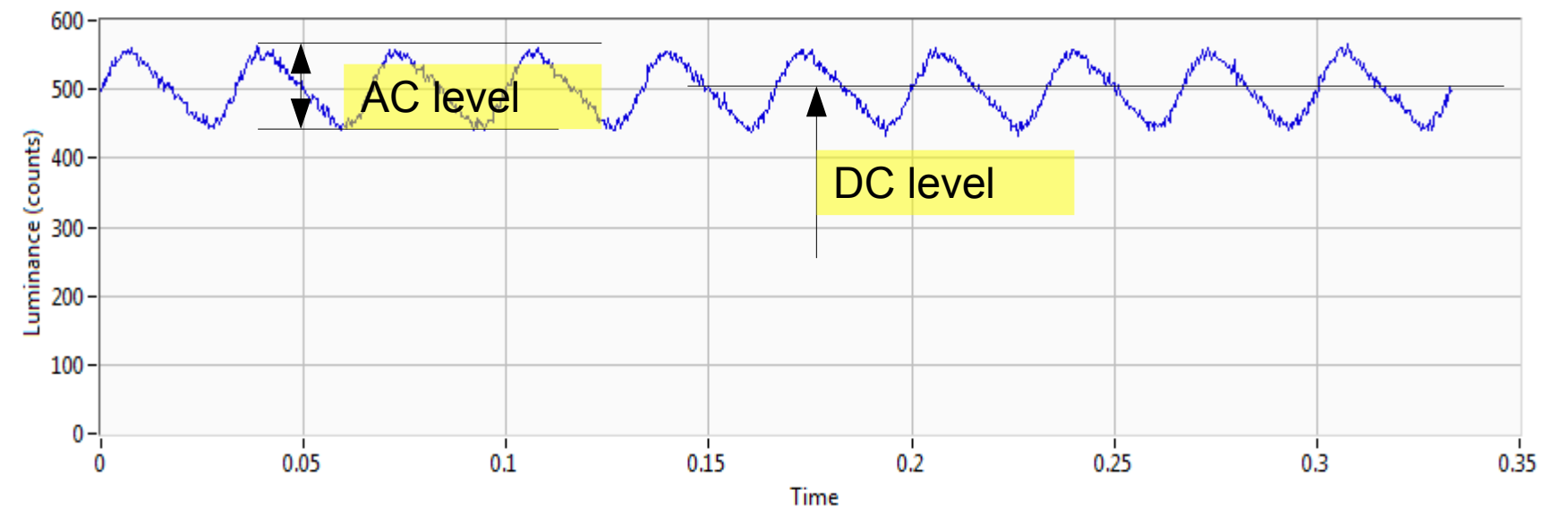
In an LCD flicker depends also on the driving technique of the display. This means some images will show flicker, while other images do not.

Think about things like dot or line inversion and the different result an image will give. There are many image possible and these all depend on needs the display driving technique.

Admesy provides some images in their LCD demo software.

Flicker calculation in general

In general, flicker can be defined as the relation between the AC level of the signal divided by the DC level of the signal.

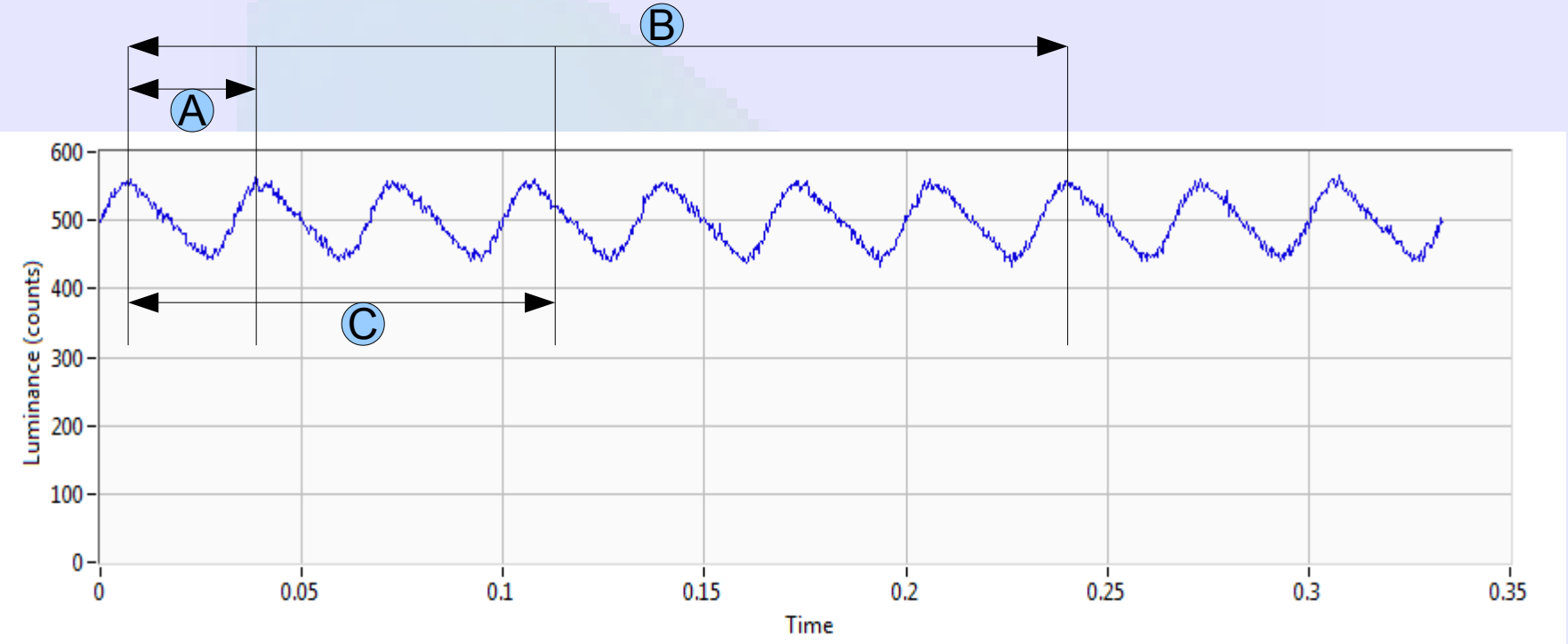


$$Flicker = \frac{AC\ level}{DC\ level}$$

This is a simplified model of course. The AC component can be calculated in various ways, therefore multiple definitions exist for flicker measurement.

Flicker measurement stability

In general, you should try to measure an exact number of frames !!!



A: just 1 frame, but maybe stable, depending on display low frequency behavior !
B: More frames (exact frame lengths) is more stable
C : Not matched to exact frame length, the DC component therefore is not stable !

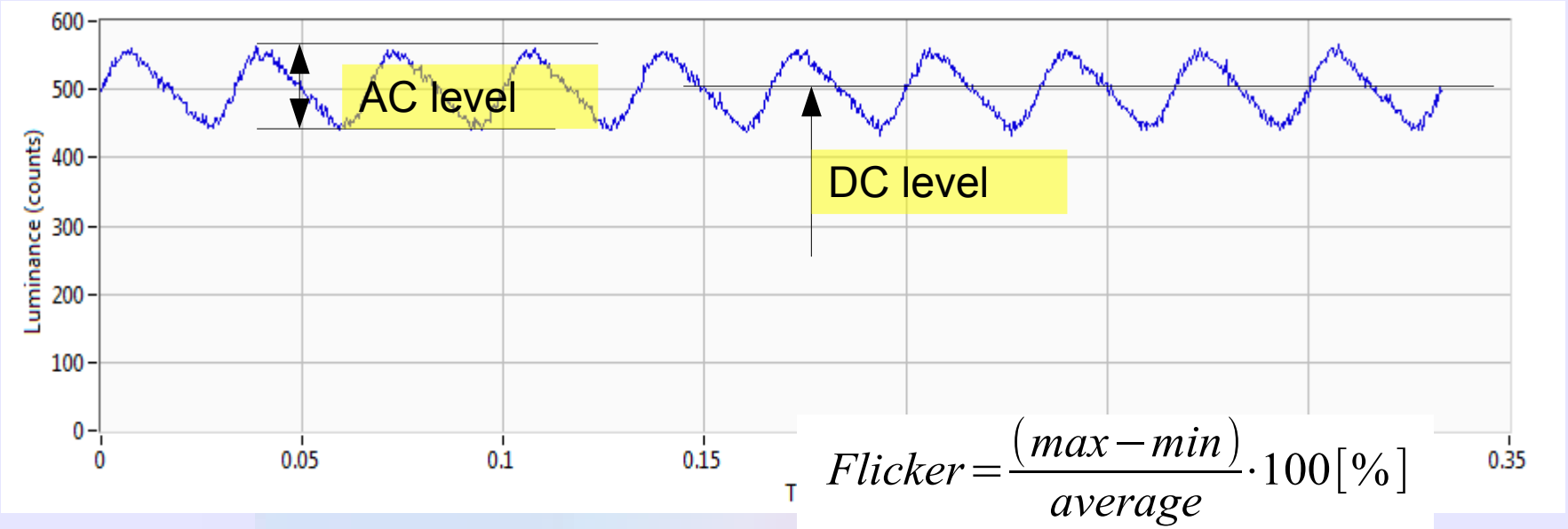
Flicker calculation methods

Admesy implements four different methods :

1) Contrast method (Min/Max)
2) Contrast method (RMS)
3) JEITA method
4) VESA method
5) Flicker percentage (IES method for lighting)
6) Flicker index (IES method for lighting)

These methods are explained in the next pages.

Flicker calculation methods : Contrast Min/Max



$$Flicker = \frac{(max - min)}{average} \cdot 100[\%]$$

$$Flicker = 10\log_{10}\frac{(max - min)}{average}\, dB$$

This is the most simple method :

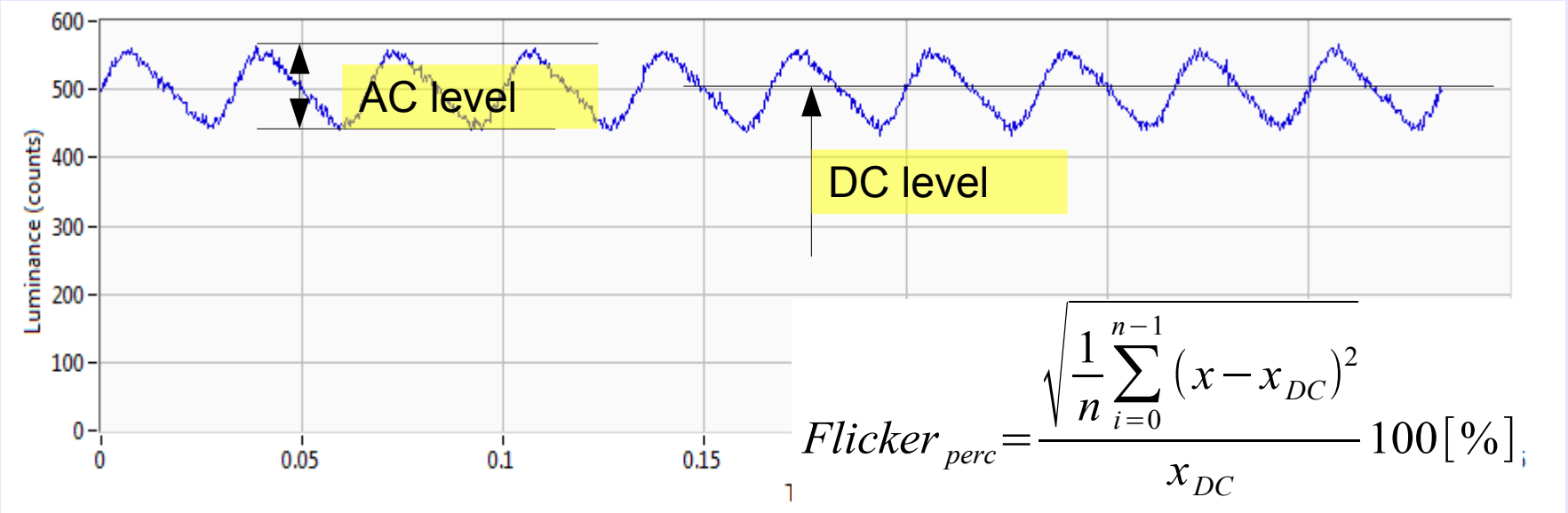
Advantages :

- Fast
- Simple : Always possible to embed in firmware of a device.

Disadvantage :

- It depends on the maximum and minimum and is therefore not the most stable method.
- Needs a low pass filter either in hardware or software to filter higher frequencies.

Flicker calculation methods : Contrast RMS



$$Flicker_{perc} = \frac{\sqrt{\frac{1}{n}\sum_{i=0}^{n-1}(x - x_{DC})^2}}{x_{DC}} 100[\%]$$

$$Flicker_{dB} = 10\mathrm{Log}_{10}\left(\frac{Flicker_{perc}}{100}\right)$$

x_{DC} : *average value of the signal.*

Based on Root Mean Square calculation of the AC component.

Advantages :

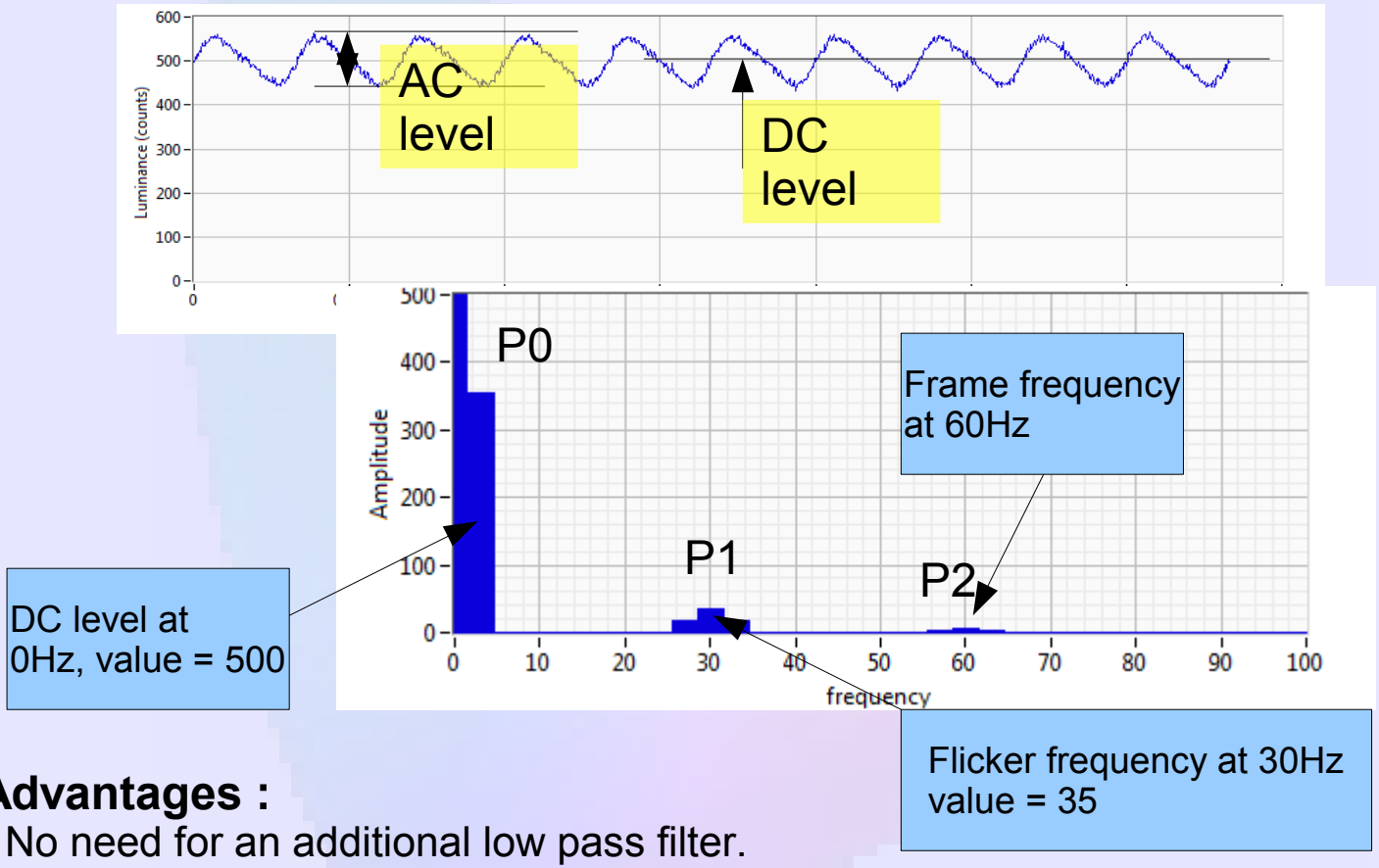
- Reasonably Fast (but slower than min/max method)
- Reasonably Simple : Always possible to embed in firmware of a device.

Disadvantage :

Needs a low pass filter either in hardware or software to filter higher frequencies.

Flicker calculation methods : JEITA

A method based on the FFT (Fourier transformation) of the signal.



Advantages :

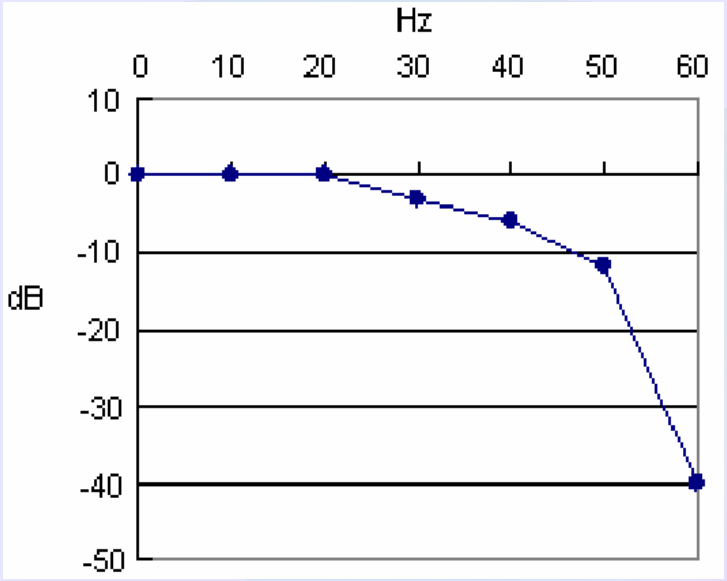
- No need for an additional low pass filter.
- Accurate and well defined

Disadvantage :

Slowest method, on embedded systems sometimes too slow to be useful.

Flicker calculation methods : JEITA

The JEITA method uses a human eye sensitivity function. This is mostly defined like the graph below :



Frequency (Hz)	Factor	
	dB	Ratio
0	0	1.000
10	0	1.000
20	0	1.000
30	-3	0.708
40	-6	0.501
50	-12	0.251
60	-40	0.010

So, the level as measured previously at 30Hz should be reduced by -3dB.The 60Hz component is lower than the 30Hz and is not used.
The amplitudes found in the previous page are reduced by these factors giving Pr1 = P1*factor.

$$Flicker_{JEITA} = 10\mathrm{Log}_{10}\left(\frac{Pr1}{Pr0}\right) dB$$

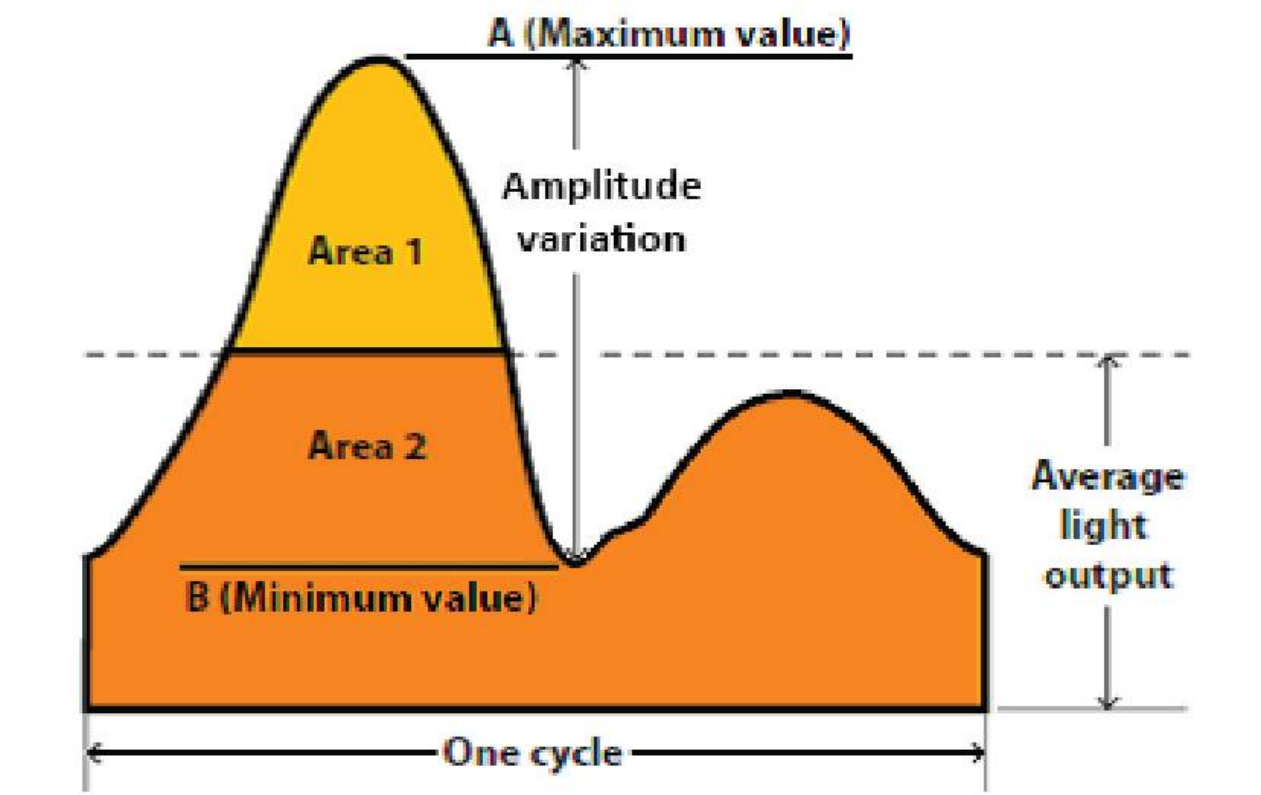
Flicker calculation methods :VESA

The VESA flicker method is equal to JEITA with a small difference in the result calculation. The difference is approximately 3.01dB which is the result of squaring the amplitudes found in the FFT :

$$Flicker_{VESA} = \left(Flicker_{JEITA} + 20\log\sqrt{2} \right) dB$$

Flicker percentage and flicker index defined by IES

This method has been defined for lighting measurements, but seems not an official standard yet.



$$\text{Percent Flicker} = 100\% \times \frac{A - B}{A + B} \qquad \text{Flicker Index} = \frac{\text{Area 1}}{\text{Area 1} + \text{Area 2}}$$

Flicker percentage and flicker index defined by IES

If we look closer to the formulas and graph the following can be noted :

1) Flicker percentage is basically equal to the Contrast max/min method except that the (A+B) factor is (A+B)/2 in the contrast max/min method.
   It is thought that the IES method meant to take the average of the signal, so we believe the formula is wrong as the graph clearly indicates the Average light output.

2) As mentioned earlier, taking the maximum and minimum is not a great idea regarding measurement stability. The RMS method performs usually better.

3) Flicker index seems to almost give equal value to the Contrast RMS method.

In short we feel that not much is added by these new methods for lighting as compared to the existing flicker calculations for LCD.

What is the best choice ?

If you're on a slow system (like pattern generator/ embedded system)

1) Take the raw signal and apply a simple low pass filter
2) Calculate the the RMS flicker value

If you're on a PC

Use the JEITA or VESA method

It makes sense to first compare the results on a PC. If a more simple method can be chosen, than it can save time in production !

Implementation in Admesy Instruments

NOT E : this does not include a low pass filter, so frame rate is included.

Contrast RMS method

Command: “**:MEASure:FLICker samples**”

Contrast Min/Max method

Command: “**:MEASure:FLICker:CONtrast samples**”

“samples” is an integer value with a maximum of 24000. Decent result are usually found between 4000-8000 samples.

Implementation in external software

Admesy provides a DLL for calculating flicker based on the raw signal that is measured by the instrument.

Get the raw data from the instrument by using the following command :

Command: “**:sample:Y samples, delay**”

Sample = number of samples, delay = delay in sample time (1 means skip 1 sample).

Next, the signal can be passed through a low pass filter (in case contrast method is used !!!) by using the following function in the DLL :

```
/* Low pass filter */
void bwlp_filter(double* samples, int count, int rate, int freq)
```

Count : number of data points
Rate : sample rate in Hz (1/dt)
Freq : cut of frequency in Hz.

Implementation in external software

Admesy provides a DLL for calculating flicker based on the raw signal that is measured by the instrument.

Get the raw data from the instrument by using the following command :

Command: “**:sample:Y samples, delay**”

Sample = number of samples, delay = delay in sample time (1 means skip 1 sample).

Next, the signal can be passed through a low pass filter (in case contrast method is used !!!) by using the following function in the DLL :

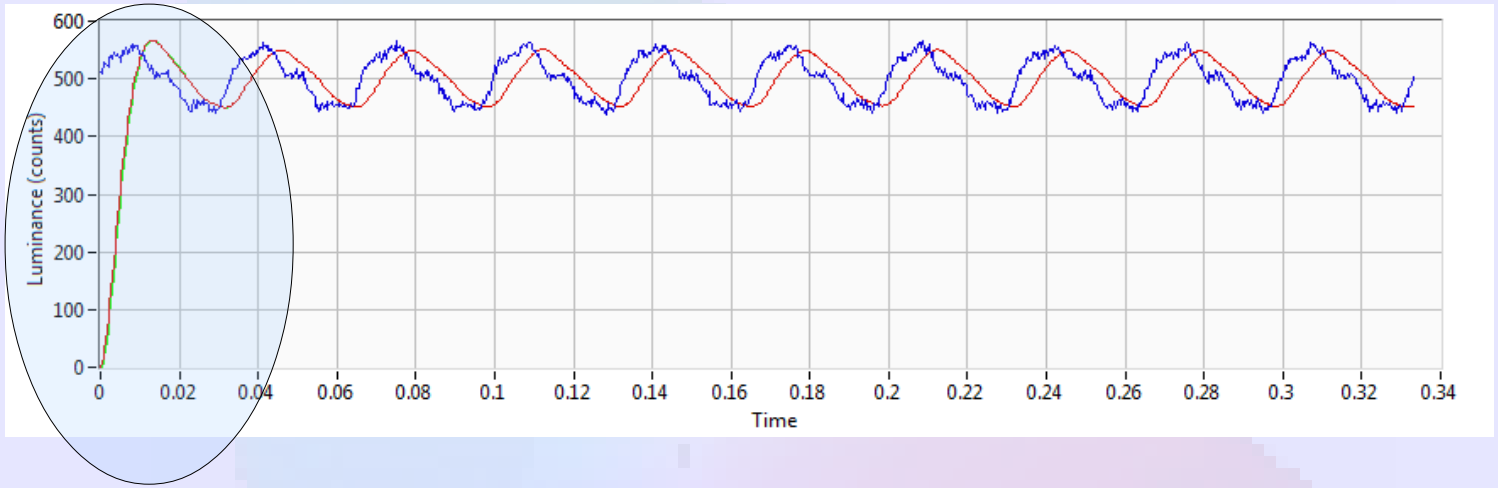
```
/* Low pass filter */
void bwlp_filter(double* samples, int count, int rate, int freq)
```

Count : number of data points
Rate : sample rate in Hz (1/dt)
Freq : cut of frequency in Hz. Usually 40-60Hz.

Implementation in external software

Using the low pass filter (bwlp_filter function) :

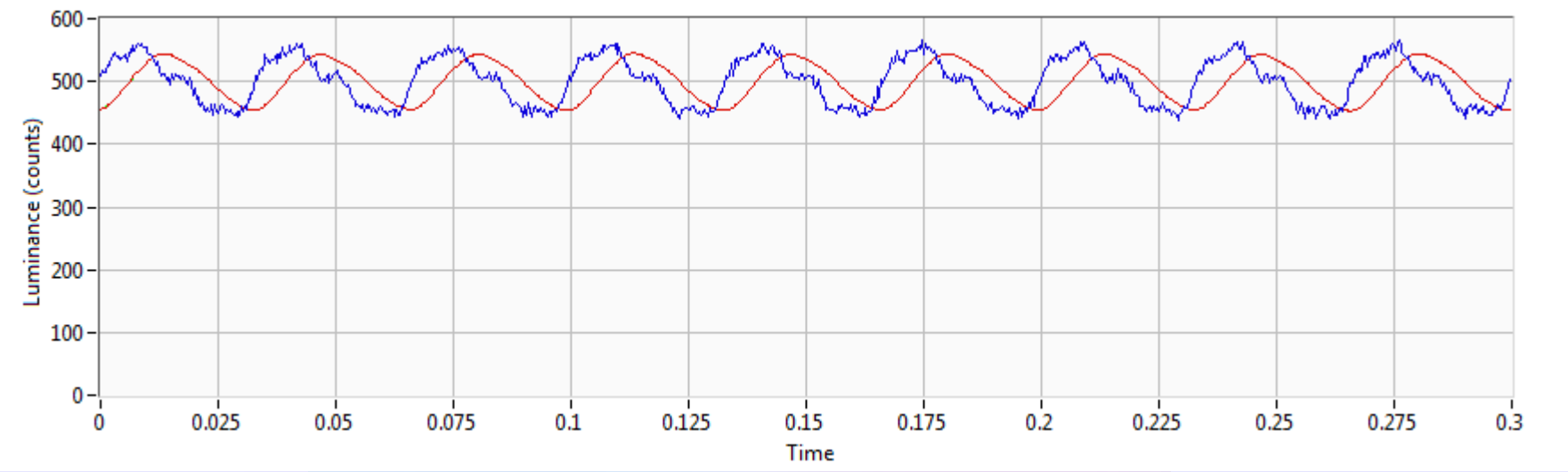
When using the filter function, the full output is returned. This will look like this :



The initial data must be skipped before passing the data to the contrast flicker measurement functions.
This can be done by simply skipping the a fixed number of samples, but it's better to skip a number of periods.

Implementation in external software

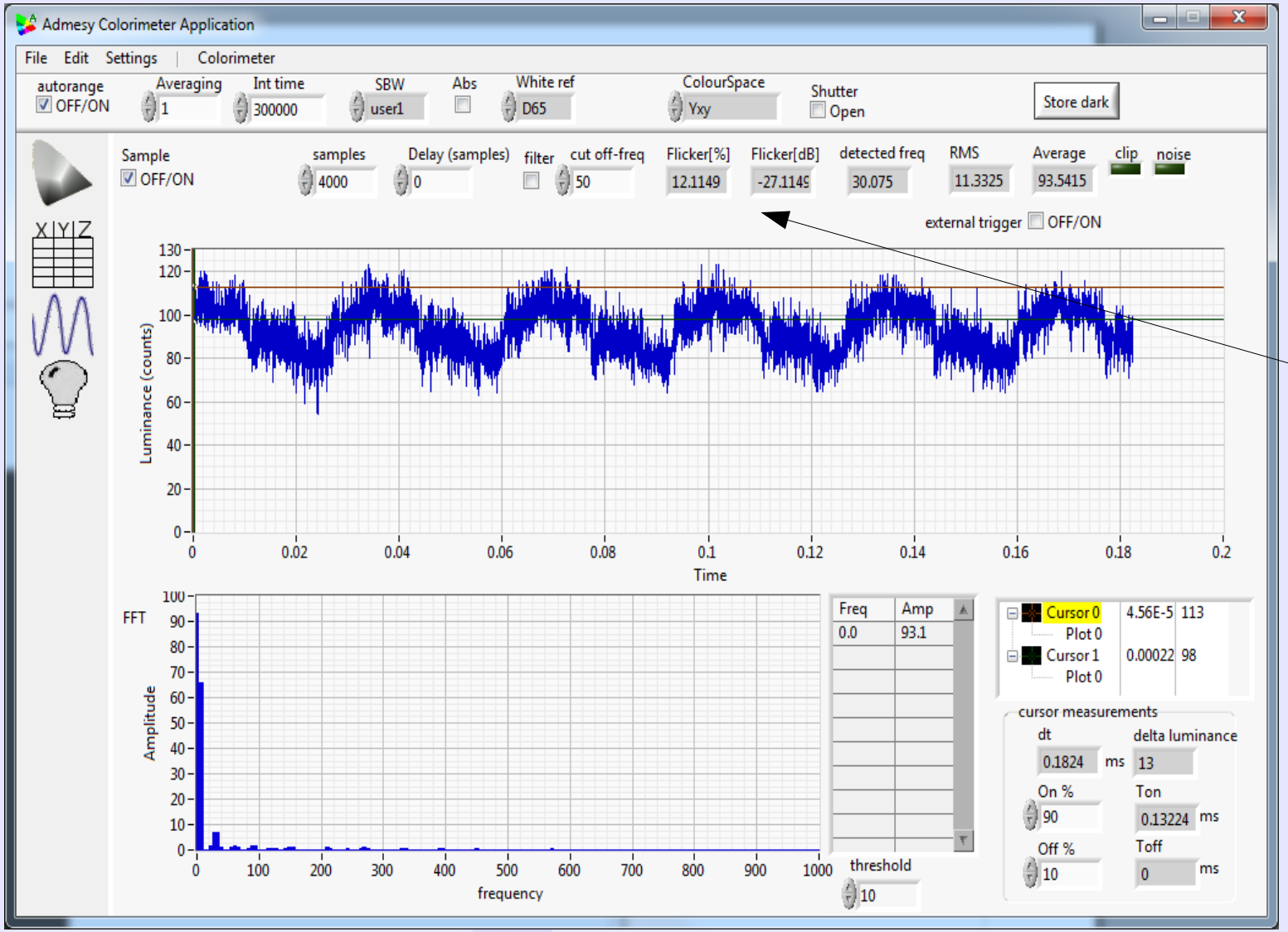
The signal after skipping the initial data should look like this (red is filtered data):



After this, you can pass the filtered data to the contrast flicker functions.
Note : The filter is not necessary when using JEITA or VESA flicker method because that is FFT based.

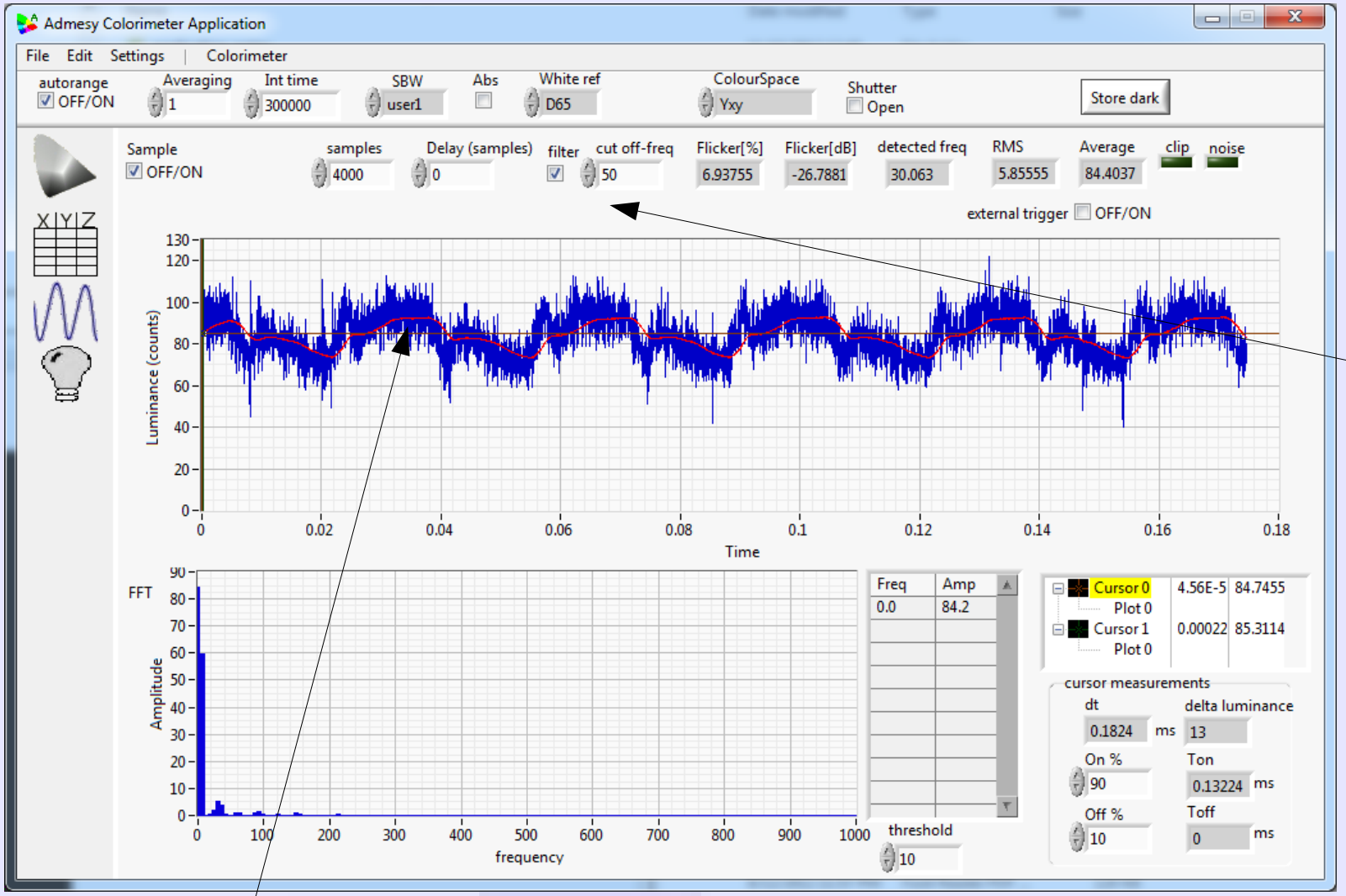
The cut off frequency is normally around 50-60Hz, but in case the frame frequency is still of large influence after filtering, a lower frequency may be chosen.

Implementation in Standard software



Flicker in %
And dB

Implementation in Standard software



Low pass filter on at 50Hz

See the flicker value get a little lower

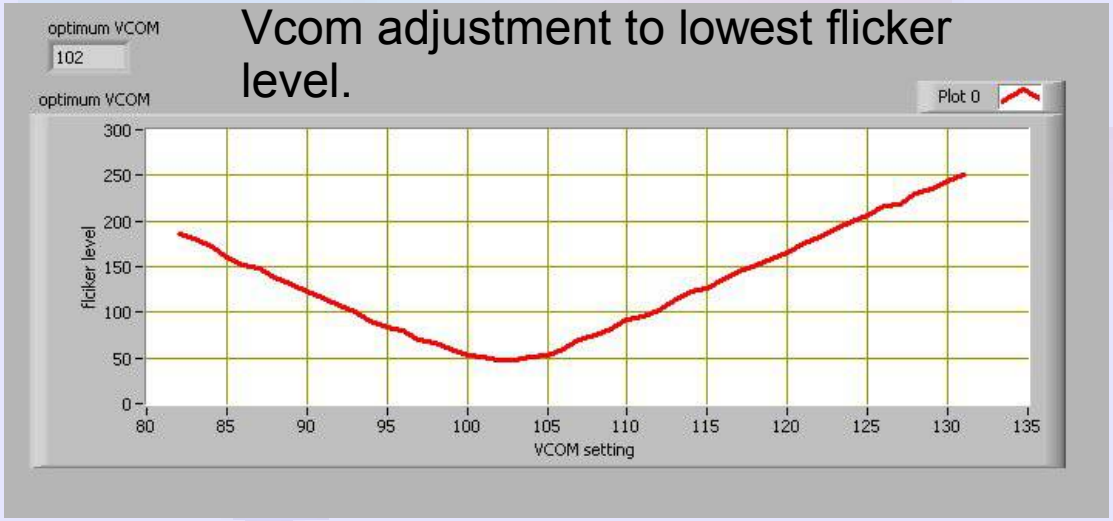
Red line is the filtered signal

Where to go from here :

Admesy can help with further implementation. If help is needed, let us know.

The Admesy results correlate with other brands of equipment but values may be slightly different due to differences in sample frequency rate and sensor sensitivity.

Usually, for flicker measurement the result to look for is the lowest value.
A graph showing Vcom on the X-axis and flicker on the Y-axis should resemble the following example graph :



Vcom adjustment to lowest flicker level.