The most important settings can (and have to) be made via the SetConditions.C and the SetParameters.C files, so that there is no need to recompile the analysis code. C script files can be loaded/executed any time in the Analysis Terminal window if the corresponding Objects (Parameter, Condition etc.) are already registered to the Go4 framework (via a unique name string of the Object).

88	Analysis Terminal [pid:9494]						-ox	
	GO4-*> Reading library: libGo4UserAnalysis.so						<u> </u>	
	GO4-*> Find user analysis class TLdaAnalysis							
	GO4-*> !!! Find constructor with prototype TLdaAnalysis::TLdaAnalysis(int, char**) GO4-*> Process: new TLdaAnalysis(1, (char**)0x7ffd7a045c58) GO4-*> Welcome to Go4 Analysis Framework Release v6.1.4 (build 60104) ! **** TLdaAnalysis: Create GO4-*> Analysis: Added analysis step Unpack GO4-*> Analysis: Added analysis step Analysis ===================================							
	Old Parameters Parameter TimeParameters: TimeAxisForLaserPulse:		1				-	
	Press enter to execute.	.x SetParamet	ers.C				- 🔍	
	*	9	~			R		

In our case both script files with the settings are loaded automatically when starting Go4 but at different times (see below).

Loading the analysis via the connect symbol



Use the . (dot) for directory which guarantees that the analysis shared library ".so" is loaded from the current directory !

88	Launcl	h analysis ? 🗸 🤉 🤇			
Operation mode:		As client (default)			
Host localhost	Port 5	000 ÷ Name MyAnalysis			
Dir .					
Lib 🔽 libGo4UserAnalysis.so					
Args:					
Starting mode		Shell mode			
exec Cirsh	C ssh	C Qt window C xterm C konsole			
		✓ ×			

Parameters are loaded already here since they are first registered to the Go4 framework in the constructor of the main analysis class TldaAnalysis.cxx. The SetParameters.C file loaded subsequently also in the constructor with a system call to load the script.

This also fires up the Analysis Configuration window. "Submit" registers all (and only those) analysis steps that are "enabled" and executes the corresponding constructors of the classes. (Please leave Auto Save Disabled because AutoSave files can turn into nightmares...). The window condition objects are registered in the constructor of step 2 since they are (only) needed in the scope of step 2, i.e., after pressing the submit (or submit + start). Also here the SetConditions.C script is executed via a system call.

Stream Analysis Configuration							
Unpack xxo Analysis xoo							
Step Control							
Enable Step 🔽 Source 🔽 Store							
Event source							
MBS File (*.Imd *.LMD *.Iml) + AMS							
Name: /mnt/Z/e128/daq1/208Bi8145.Imd							
Auto Save File							
MyAnalysisASF.root							
Enabled 500 s ÷ 5 ÷ ♥ Overwrite							
Analysis Configuration File							
Go4AnalysisPrefs.root							
Submit Submit+Start Science							

The start button (or in a combined action submit+start) executes first the TldaAnalysis::UserPreLoop() in TldaAnalysis.cxx and then the loop over the event data



The PreLoop can thus also be used to reset some variables that are defined as parameters (or that have global scope otherwise) if you Stop – Start between two "online injections" – recommended. I use it to reset piLaserStep = 0 (which is the variable that counts laser steps within one scan, see below).

Similarly, the stop button executes the UserPostLoop() function.

Please, don't mix up the start **and stop** button (single green arrow) with "monitor" button ***** (three small arrows) and the stop monitor button here.

Important spectra and how to determine the parameters

Parameters and conditions and their meanings

a) Parameters

```
fPar->pfRevolutionFrequency = 1.98777e6;
```

The value is needed for the correct folding/accumulation of the TDC spectra.

fPar->piPhaseOffset = 30;

Offset/shift of ALL TDC hit/bunch spectra in Bins of the TDC bunch spectra. The value can e used to center the peaks in software.

```
fPar->pfSetWavelengthLow = 601.7;
fPar->pfSetWavelengthHigh = 601.9;
```

Value will be used to determine PMTOverWL Spectra Range. But not the exact values will be used but they serve as input for slight adjustments to minimize binning effects. Please use the values here as you give them to the laser via the LabView control window.

fPar->piNOfLaserSteps = 40;

The value is needed for the correct folding/accumulation of the step spectra. It corresponds to the number of laser steps per scan and defines the axis of all the step spectra (see section on Determining / diagnosing the number of laser steps below).

fPar->pusFoldStepSpectra = 1;

Used to fold the step spectra, i.e. add up the data or don't (0: contiuous, 1: add up several scans). (see section on Determining / diagnosing the number of laser steps below).

fPar->piCSReductionFactor = 90;

Determines the frequency/time of the Common Stop (CS) Signal. 1 CS every piCSReductionFactor * 1/pfRevolutionFrequency (with the numbers above: 1 CS about every 45,277 us) ...

fPar->piNOfCSForValidTimeRange = 12;

Used for the soft gating of the fluorescense light. Determines how many common stop intervals are included after one QSwitch signal are included in the analysis. In the example, 12 CS Intervals are used which approximates to 12 * 45,277 us ~ 543us. Please

ote that this time is now sharp since the actual TDC time of the Qswitch and the other TDC signals are not used.

```
fPar->piLaserStep = 0 ;
```

LaserStepCounter, leave at = 0, value changes during the analysis, LaserStep as a parameter allows for accessing and resetting the step counter e.g. every time the start button is pressed or via a script.

b) Conditions

Window Conditions starting with wins give the range of allowed Scaler values

(or: TDC size). **Use the spectra under 3_TDC/size to determine these values**. As a rule of thumb: For the RIB beams (208Bi) or low intensity 209Bi for detectors MIDDLE and NORTH only the values 1 and 2 should be allowed, for higher intensities 10 is reasonable. The value for XUV depends if it is hardware gated (not necessary at low intensities) or not.

Window Conditions starting with **winc** give the **range of allowed TDC values** ("Time peak")

that are allowed. **Use the spectra under 3_TDC/accu to fix these values.** The peaks can be shifted with Offset Parameter which gives a time / phase offset for all (!) spectra. If you change the offset parameter the window conditions need to be adjusted correspondingly. Keep in mind that due to the additional hardware used to provide the 1ms hardware coincidence window the peaks with "GATED" in the name are at different positions than the ungated ones.

c) Resulting spectra

Filtered TDC spectra can be found under **3_TDC/accusig**. Filters: max numbers of counts (wins conditions), TDC peak (winc conditions) and soft gating(number of CS included) for ANODE, MIDDLE, NORTH.

Filtered STEP/Resonance Spectra

In order to get proper STEP spectra use the LS0_N to determine the number of steps (see corresponding section).

4_LaserSteps/Signal/LSS3_MIDDLE (here, files 8150, 8153, 8154 combined)



4_LaserSteps/Signal/LSS3_NORTH



4_LaserSteps/Signal/LSS3_ALL (signals from both detectors added up).



4_LaserSteps/Signal/LSS3_MIDDLE_GATED



4_LaserSteps/Signal/LSS3_NORTH_GATED



Please note that the hardware gating and the software gating do not work together so that there is no filtering for the hardware gated (1ms) spectra.

Unfiltered STEP/Resonance Spectra

All spectra under 4_LaserSteps/Counts are unfiltered (raw data). Of course, the LS10_NORTH_GATED and LS11_MIDDLE_GATED show only the gated 1ms, all others the full 30Hz (33ms).



PMToverWave (5_Laserscan)

In the SetParameters.C the two values of the actual set san range (as given in the LabView control of the laser) and the number of steps need to be given. From these parameters usefiul parameters for the borders of the spectra are calculated to minimize binning effects. The PMT/Anode spectra are filtered, MIDDLE, NORTand ANODE in addition have the soft gating.

5_Laserscan/TDC_PMToverWave_PMT_middle



5_Laserscan/TDC_PMToverWave_PMT_North



5_Laserscan/TDC_PMToverWave_PMT_middle_gated



5_Laserscan/TDC_PMToverWave_PMT_North_gated



Determining the laser steps

For the laser steps, the next laser step pulse in the scalers (presently ch 30 when counting 0 to 31 and TDC scalers in channels 0 to 15). The steps are counted in the parameter **fPar** \rightarrow **piLaserStep.** The variable is reset to zero by modulo with

fPar→piNOfLaserSteps. So it is important to set this value correctly and check in the step

is

spectra. In addition, **fPar**→**piLaserStep =0** (i.e. reset) every time the start button

pressed. This is to force starting the step spectra with the first step if the analysis is paused (stop-start) for a new injection. In order to check/measure/varify the correct value $fPar \rightarrow piNOfLaserSteps$. With the parameter $fPar \rightarrow pusFoldStepSpectra$ the modulo devision ("the folding" / "resetting" after $fPar \rightarrow piNOfLaserSteps$ can be switched off to determine the proper value for $fPar \rightarrow piNOfLaserSteps$. $fPar \rightarrow pusFoldStepSpectra = 1$ folds the spectra (adds up several scans) $fPar \rightarrow pusFoldStepSpectra = 0$ leads to a long sequence of all new steps.

For this 4_LaserSteps-Counts-LS0_N is very helpful as in the unfolded spectrum one can immediately see if everything is working correctly and which value to chose for **fPar** \rightarrow **piNOfLaserSteps**. As an example, in the unfolded spectrum below (file 8150) we see 6 scans with exactly 40 steps, i.e., the peaks have to be at exact multiplies of 40 (40/80/120/...).



In the folded form where data are added up the spectra looks like below.

