## HPLC

Setting the communication with USB AD Converter

At the start phase the ADC was communicating with the Gemynix software, however it only handles one HPLC setup (which is way beyond of the possibilities of the ADC), the software is hard to reinstall and expensive. However the ADC communicates with the computer through USB connection, which is not as simple as Serial communication.

In the first step the communication was logged with Device Monitoring Studio and USBcap+Wireshark software. The communication was several pages long, the data transmission came in the following way:

**A10002000004681`A20001FFFFC972A30001FFFFDB84A40001FFFFFD88A5000200000161`2A6000200004D88A10001FFFFFD88A20001FFFFBE8AA30001FFFFD1`56AA40001FFFFEB84A5000200002264A600020000466CA10001FFF1`FFE8AA20001FFFFBF8CA30001FFFFDD88A40001FFFFFE8AA5000201`0000468A6000200003F8CA1000200000162A20001FFFFBF8CA30001`1FFFFE366A40001FFFFF468A5000200000972A6000200003B84A101`001FFFFFD88A20001FFFFC468A30001FFFFE56AA40001FFFFEB84A1`5000200001870A6000200003366A1000200000972A20001FFFFC66C1`A30001FFFFDB84A40001FFFFEA82A5000200001B84A6000200003871`0A100020000076EA20001FFFFCE8AA30001FFFFE56AA40001FFFFE1`76EA5000200000162A6000200003C86A10001FFFFFC86A20001FFF1`FC972A30001FFFFED88A40001FFFFF56AA5000200001870A6000201`0003C86A1000200000366A20001FFFFC56AA30001FFFFDD88A40001`1FFFFEF8CA5000200000F8CA6000200003F8CA**

This code contains all the necessary information. The character string was analyzed in MATLAB and it showed that the 1` signs were unnecessary, the first two code identifies the channel, the next three code the injection, and the remaining characters the measured signal intensity in Hexadecimal. After this point a code was built to extract information from the string set. However direct communication was still not possible, as the ADC had to be initialized.

After the device and the computer set up the communication protocol, the following codes were sent to the ADC:

|  |  |
| --- | --- |
| 02 43 31 31 03 | .C11. |

|  |  |
| --- | --- |
| 02 43 32 31 03 | .C21. |

|  |  |
| --- | --- |
| 02 43 33 31 03 | .C31. |

|  |  |
| --- | --- |
| 02 43 34 31 03 | .C41. |

|  |  |
| --- | --- |
| 02 43 35 31 03 | .C51. |

|  |  |
| --- | --- |
| 02 43 36 31 03 | .C61. |

|  |  |
| --- | --- |
| 02 47 03 | .G. |
| 02 41 30 03 | .A0. |

|  |  |
| --- | --- |
| 02 41 31 03 | .A1. |

Further investigation in Wireshark showed that the first six lines were necessary (channel initialization?), the seventh line could be discarded, and the last two lines start the transmission. In the last step the Labview VI’s were applied directly to set up the USB communication, available at <https://forums.ni.com/t5/LabVIEW/Nugget-2-of-n-USB-Control-transfers-using-VISA/td-p/757011>.

For the integration part the following link was used as base, and parts of it directly applied:

<http://www.cs.wcupa.edu/~tstarn/LabVIEW.html>.

The software is functioning as it is. With basic Labview knowledge it can be further developed to control six parallel HPLC/GC or other measurement device (a software from an HPLC company would easily cost 4000 € each unit).

### Initializing communication

Each time the ADC is detached from the power source it has to be reinitialized. This is done by starting the ‘**Initialize, has to run once**’ VI file. There is a ‘Visa Refnum in’ pop down menu, where the current USB slot has to be chosen, usually there is only one option. Once it runs, it sends measurement data continuously.

The main part is the **‘Initialize USB detector’** Sub VI: Which opens the communication through USB connection, sends the codes ‘.C11.’; ‘.C21.’; ‘.C31.’; ‘.C41.’; ‘.C51.’; ‘.C61.’ (most probably channel initialization); ‘.A0.’ (stop communication); ‘.A1.’ (start communication). After this the ADC send the measured data in the usual format continuously.

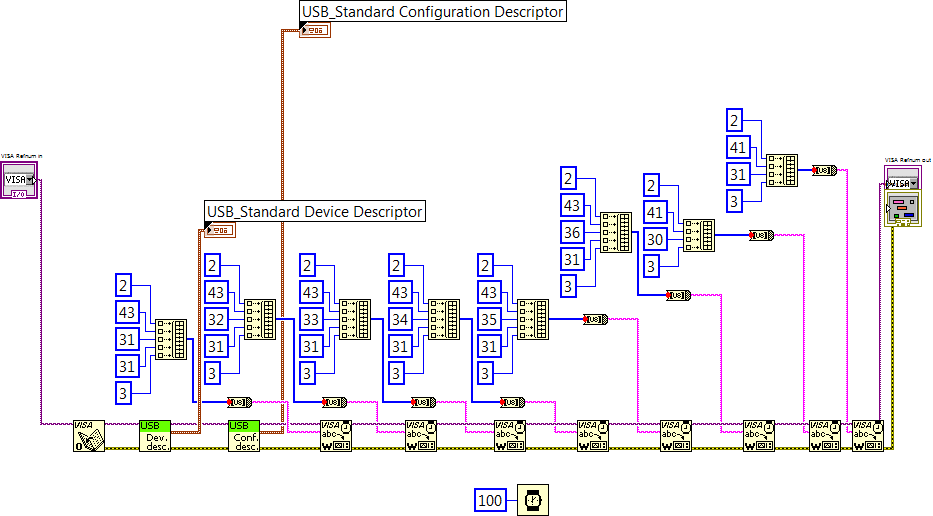


Figure X: Initialize USB detector diagram



The USB Analog-Digital Converter (ADC)

### Users manual

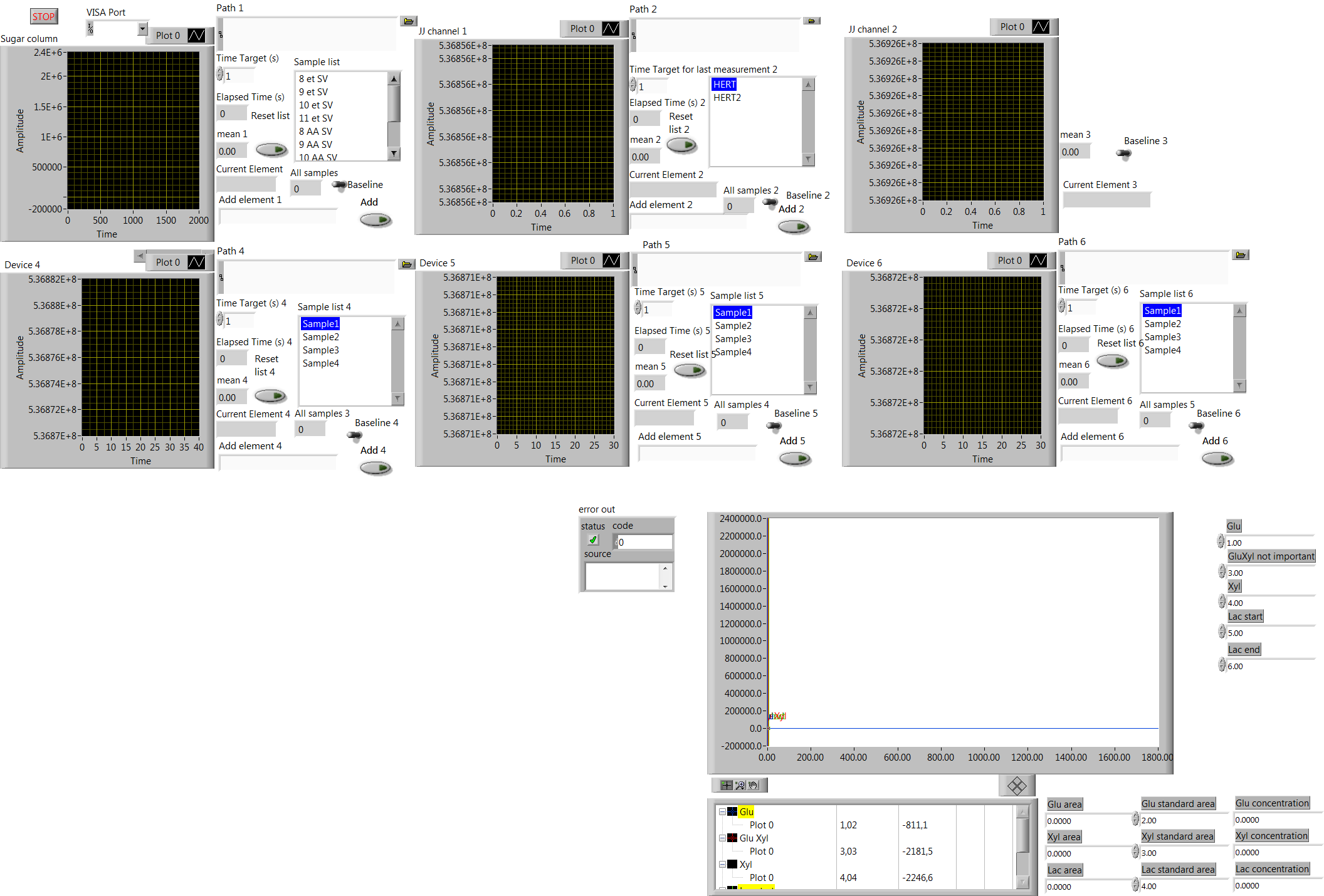
After the HPLC setup is connected, the column and the detector equilibrated, the ADC initialized, the program can be started. Immediately after the start the mean value values will appear, else the communication has to be initialized again (see previous section).

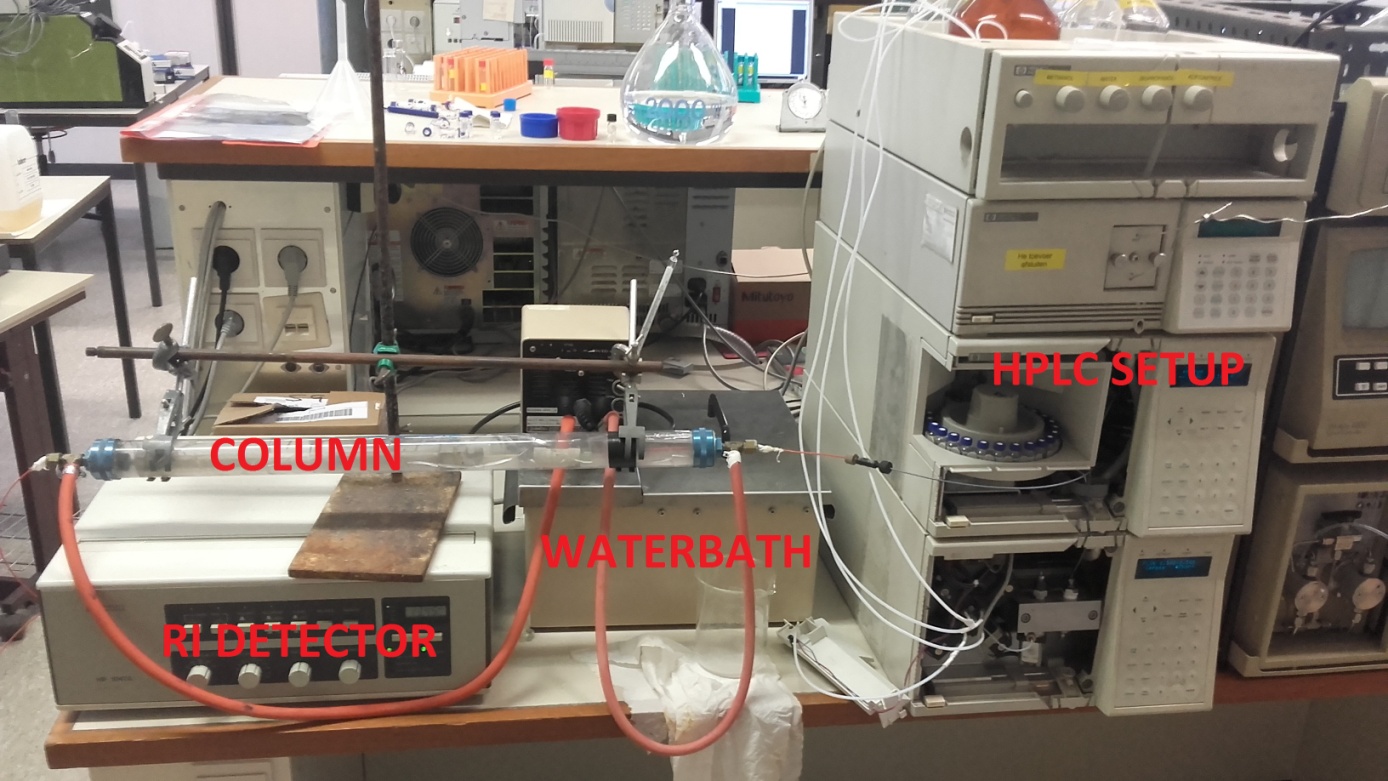
The front panel consists of six almost identical sections, each representing one ADC input from the available six. The second and third plots are both connected to one HPLC setup, measuring on two UV wavelengths, hence only panel 2 has input values, panel 3 uses these also. The rest of the plots and ADC inputs are free, anything can be connected what has analog output (pH meters, conductometers, UV-VIS detectors). More complex detectors, which more signals per time (DAD-diode array detectors), could not be connected. Each channel has inject signal input also. Currently the first panel is attached to one HPLC with RI detector, and panel 2 and 3 to another HPLC setup with double wavelength UV detector.

In the next step the folder has to be created where the measurement files will be saved (the software will not create it automatically), the path of the folder has to be chosen via the **Path box**, the **Sample list** filled with the names of the samples via the **Add element** text box and the **Add** button. In the first panel the **Target time** set in seconds is the measurement time for each sample, this has to be set also on the autosampler. The software records data until this time has passed then saves it out and waits for the next inject signal. The autosampler can only handle one measurement length. In case of panel 2 and 3 the **Time targe**t is the measurement time of the **last sample**, as this autosampler can handle different measurement methods (also different measurement times), hence the software records data between two injections, only stops measurement after the set time has passed after the injection of the **last sample**. Right before the measurement the baseline has to be zeroed down with the **Baseline** switch (not essential but the ADC works in the widest range this way). After this the autosampler can be started and the rest will be automatic.

The front panel has several outputs and indicators: The **Plot** shows the recorded data in time, **Elapsed time** the time passed since the last injection signal, **Current element** the name of the current sample, **All samples** the number of all of the samples.

The integration part works for panel 1 and the connected HPLC as it is optimized for a single measurement. For other measurements the Matlab integration program has to be used. After each measurement the integrator will activate automatically, the peak time details are filled in already as it is optimized to our method (start and end of the peak, standard peak area). The peaks are integrated, the cursor marks the start and end points, the chromatogram is visualized, the concentration calculated and the values remain in the boxes until the next sample is not ready for integration.





### HPLC block diagram

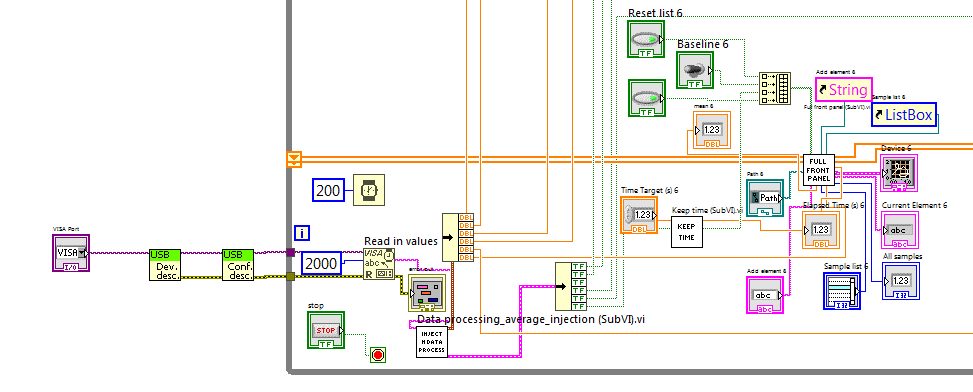
The **‘HPLC software’** VI consists of the following sub VI’s, discussed in hierarchy:

1. USB Device Descriptor (Readily available on the internet)
2. USB Configuration Descriptor (Readily available on the internet)
3. Inject N Data Process
4. Keep time
   1. Keep Time
   2. Keep Time between injections
5. Full Front Panel
   1. Add Element
   2. Save Values At End

Discarded, because of the separate Matlab integration tool

1. Get Cursor Data
2. Integrate One Peak
3. Integrate Two Peaks

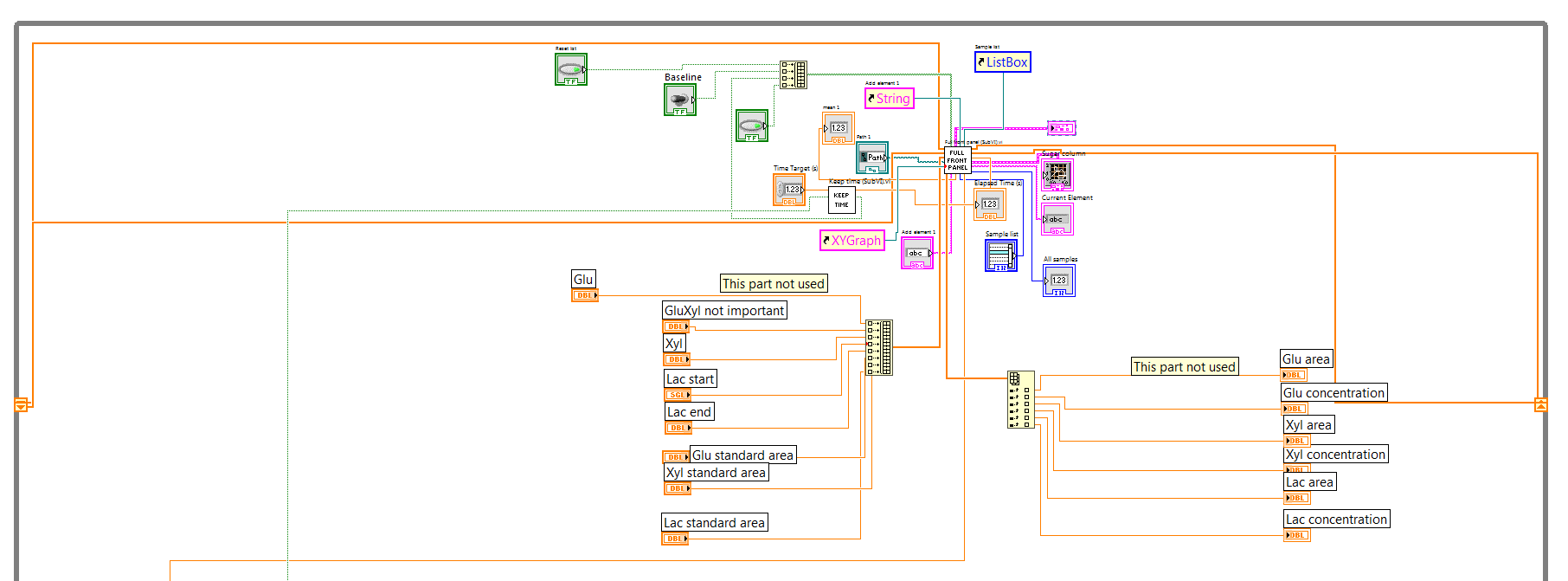
**The Block**

****

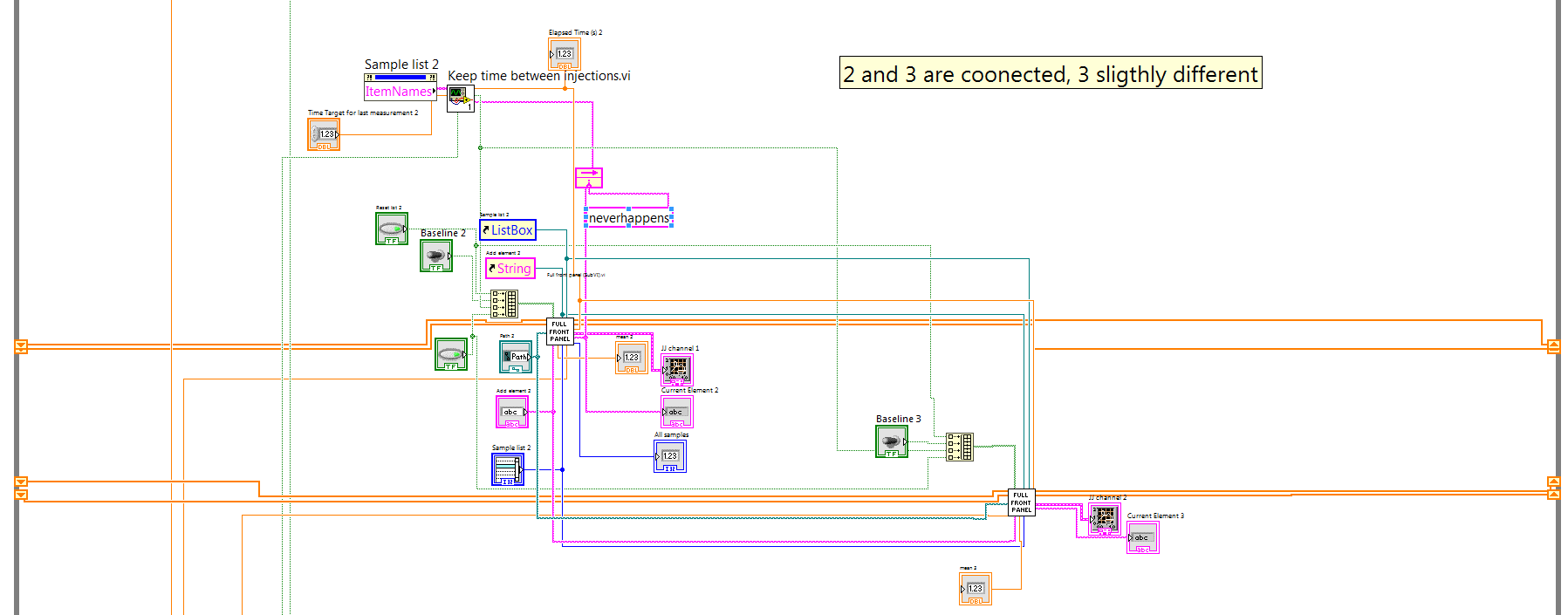
Picture X: The common reading for all the panels and Panel 6. Panel 4 and 5 are identical, Panel 1, 2 and 3 are similar

Used Sub VI’s are always explained by either indicating their source or by separate descriptions in later sections in case of self written Sub VI’s. The Diagram starts in the bottom left corner (Picture X), where the Visa port of the detector can be set (Usually there is only one possibility in the drop down manual, and it appears when the ADC is connected). The following two elements are parts of the USB communication package available freely on the internet; they were applied without any change. Briefly the software asks for communication details and sets the packages accordingly. If the ADC was initialized it is already sending the data and it can be directly read in (‘Read in values element’, each point is calculated from 2000 input characters). The data string goes in **Inject N Data Process** Sub VI, where it is processed, the six channel data separated, the inject signs analyzed, and the measurement data converted in double precision. The two outputs (inject, data) of the six channels go to six **Keep Time** and **Full Front Panel** Sub VI’s, correspondingly. The **Keep Time** module keeps the ‘Measurement running?’ output true after the inject signal (input) until Elapsed time is smaller/equal than Time target (input control) and also gives an Elapsed time after injection (output) value. There are two different **Keep time** Sub VI’s for HPLC 1 and HPLC 2.

It has to be noted that three slightly different **Full Front Panel** Sub VI’s exist; Panel 6 will be discussed here for the others the differences will be highlighted. The **Full Front Panel** Sub VIcontains the majority of the code. Its inputs are the ‘Mean in’ (measured data), the ‘Measurement running?’ boolean from **Keep time** Sub VI, the Path where the data needs to be saved (control), the Baseline Boolean for baseline setting (control), Reset list Boolean (control), Add element Boolean (control), Add element string (control), and Sample list. The outputs are the chromatogram plot, the name of the current sample (Current Element), and the number of samples on the list (All samples). The **Full Front Panel** Sub VI, plots the measured data on a chromatogram until the measurement is active and clears the plot before the next measurement, it sets the baseline 0 when the Baseline button is pressed, clears the sample list when the Reset List button is pressed, adds the element written in the Add element box when the Add element button I pressed, saves data after every run to the user specified place under the sample name in excel format, outputs the current sample name and how many samples are on the list all together. The six graphs on the control panel, together with the buttons, lists and other controls belong to these Sub VI’s.



Picture X: Panel 1



Picture X: Panel 2 and 3

Panel 1 also contains an integration part, this means an extra input array containing the values needed for integration (start and end of peak, standard peak area), an output array containing concentration values, and an output for the graph which visualizes the peaks on the chromatogram. Panel 3 receives all its inputs from Panel 2, except Zero baseline. Keep time is slightly different, having two extra inputs: the Sample list and the Current sample string.

**Sub VI’s:**

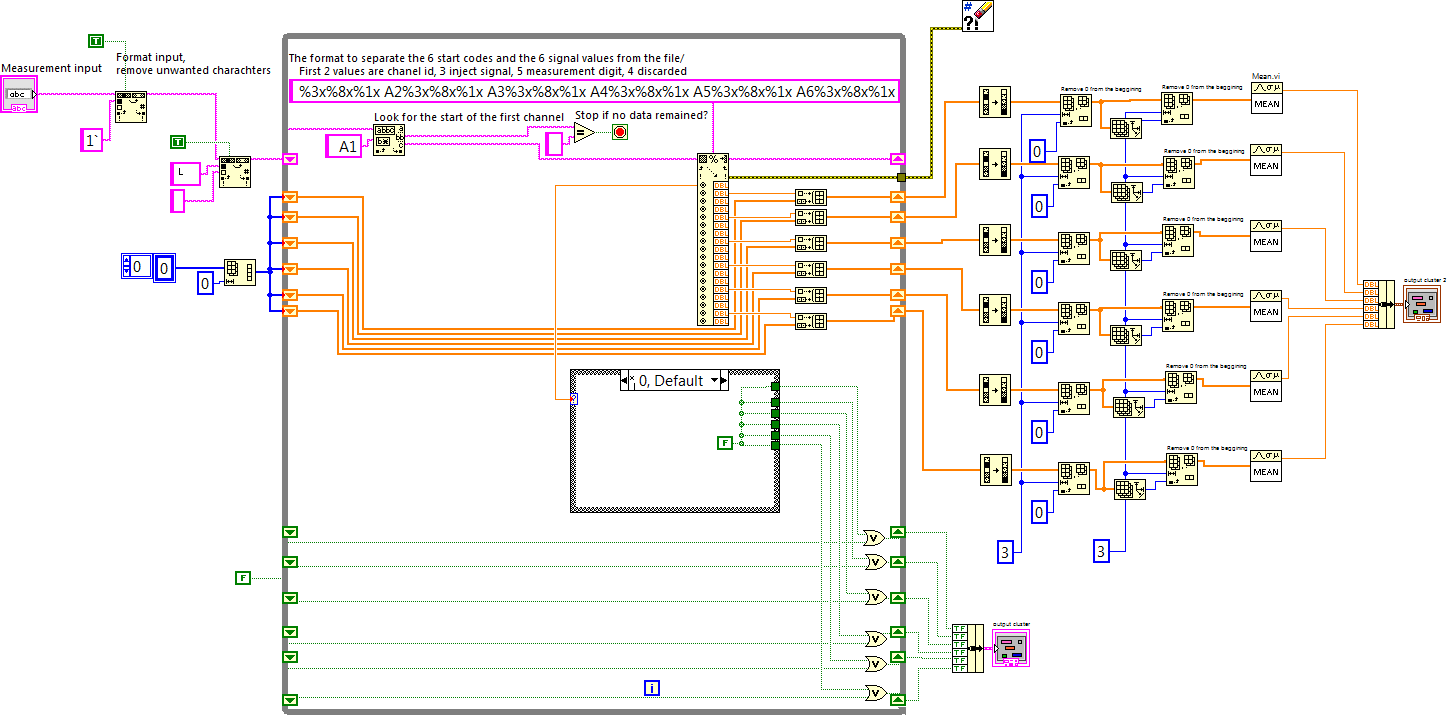
#### Inject N Data Process

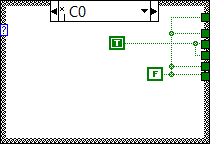
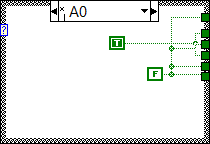
The Sub VI receives the 2000 character long data string, removes the ‘1`’ characters and substitutes the ‘’ characters with simple space. As a result the data will have the following format: Axyyyzzzzzzzzz, where A indicates the starting of the string, x is the channel id (1-6), the three y values contain the inject code, and the nine z values are the actual measurements in hexadeca. In the next step a Match pattern element searches for the start of the dataset (A1) and passes the rest of the string to the Scan from string element. If it does not find A1, the whole dataset has been processed and the Sub VI terminates, sending the results to the main VI.

The Scan from string element separates the channel id/injection mark/measurement values. The measurement values are collected in one array per channel until all data are processed, afterwards sorted in a row, the first and last three elements are discarded (remove outliers), the average is taken from the elements and it is passed out from the Sub VI. A Clear errors element is needed, because if the string does not contain elements for every channel (happens often at the end of the 2000 characters), the program returns an error and stops.

The injection code is the same for every channel (all yyy are identical). Hexadeca 10, 20, 40, 80, 100, 200 values represent injection at 1, 2, 3, 4, 5, 6 channel. Simultaneous injections are marked by the addition of the numbers: 30 is 1-2, 300 is 5-6 etc. Start codes were placed in Case structure, and coded until two simultaneous injections (6 single, 15 double), a signal is about 1s long, it is almost impossible that 3 occur simultaneously in a several minute’s long HPLC method. The OR structure-Feedback combination keeps the value true, once a true signal gets in the 2000 characters.

The outputs are grouped and passed forward.

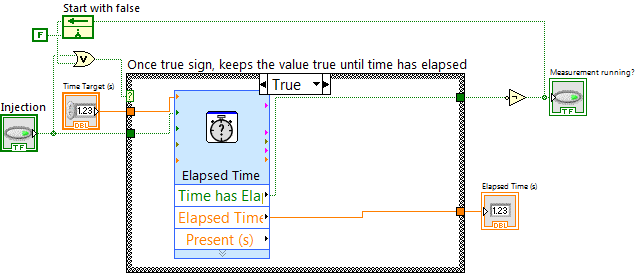


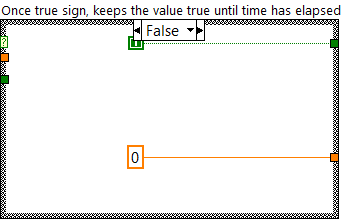


#### Keep Time

##### Keep time

The **Keep Time** Sub VI keeps the output signal true after an inject signal until the Time target is not reached. At the start the inject time is false and the Feedback node gives a default false value, the output of the OR element is false, the output of the Sub VI is also false and 0 time has passed. If the signal turns to true the OR element gives a true sign, it starts the Elapsed time element, which gives the actual Elapsed time value as constant and gives out a false signal (Time target not exceeded yet), which is converted to a true signal (output) and redirected to a Feedback node to the OR element. The Feedback keeps the Case structure true until the time has elapsed output does not turn true, and returns the system in initial state until the next inject sign.

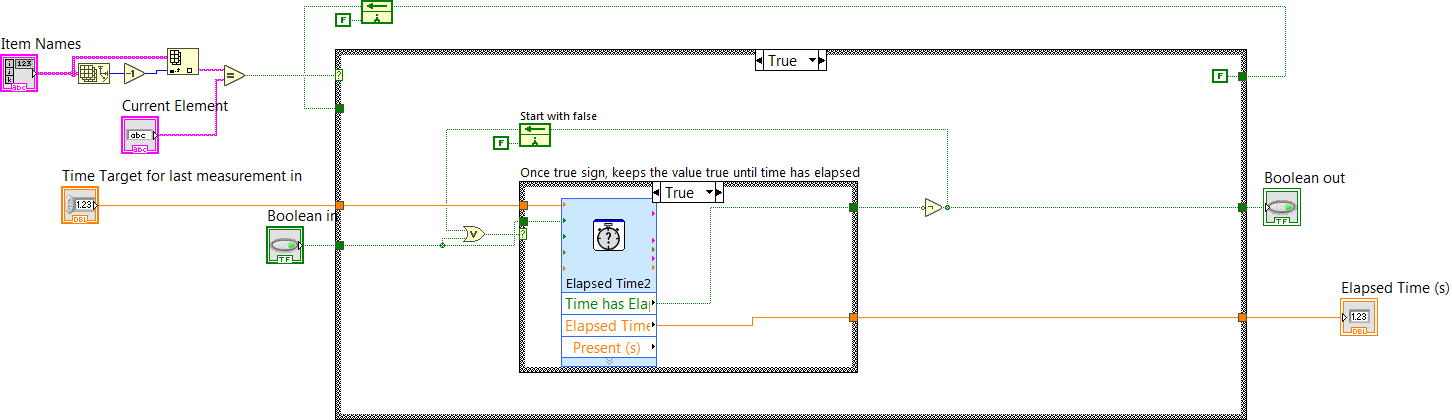




##### Keep time between injections

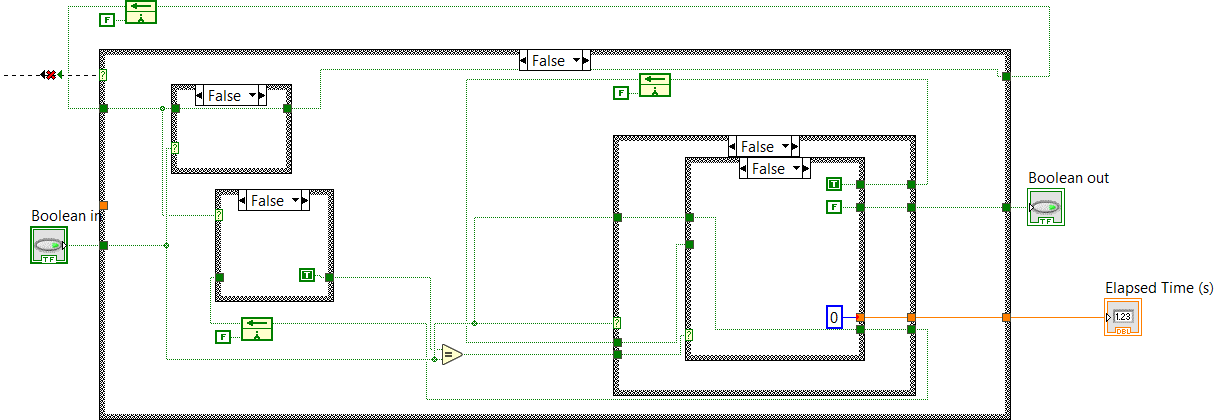
HPLC 2 can work with different measurement methods, hence it is needed to measure between two injection signals, and only stop the last measurement after the time has passed. The Sub VI has two extra inputs: the

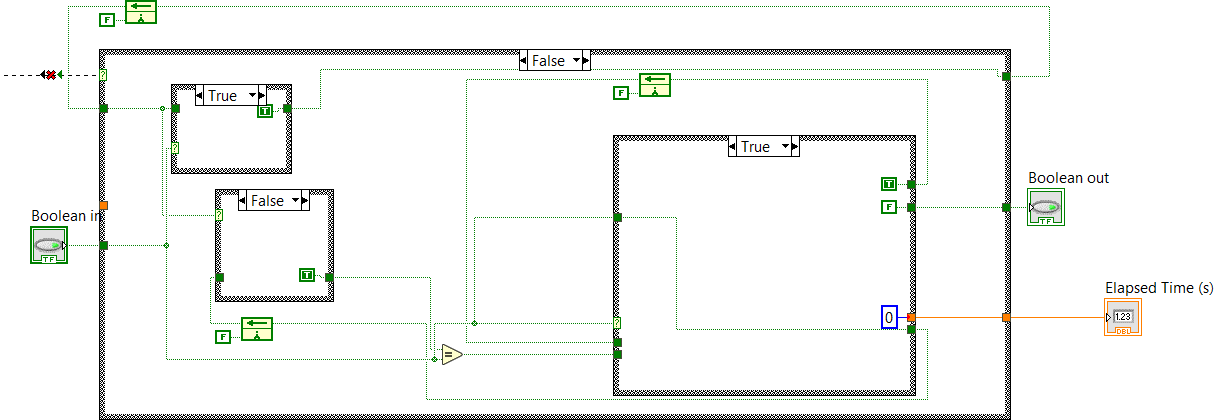
Sample list and the Current element Feedback node (it has to be a feedback node as the Full Front Panel will run only after the Keep time Sub VI has finished). The default value of the Feedback node is ‘neverhappens’, and it will work fine until someone does not measure a sample named neverhappens. I hope it will never happen.

****

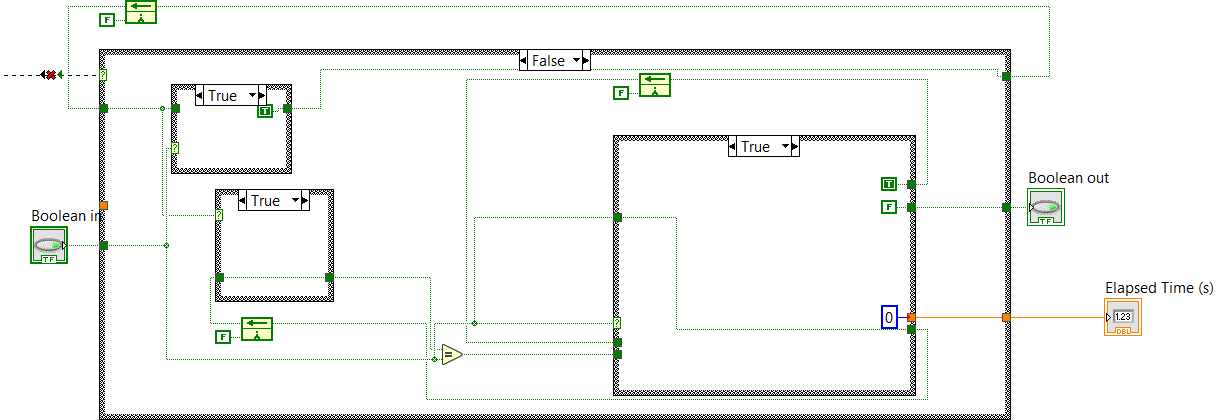
In the upper left corner of the Sub VI compares the Current sample with the last name in the List box, if the last sample runs it will work the same way as the simple **Keep time** Sub VI. In case of the first measurement after the startup the upper left case structure is set to false. at the start the inject Boolean is c

The Measurement Boolean out is false and the Elapsed time value is zero.

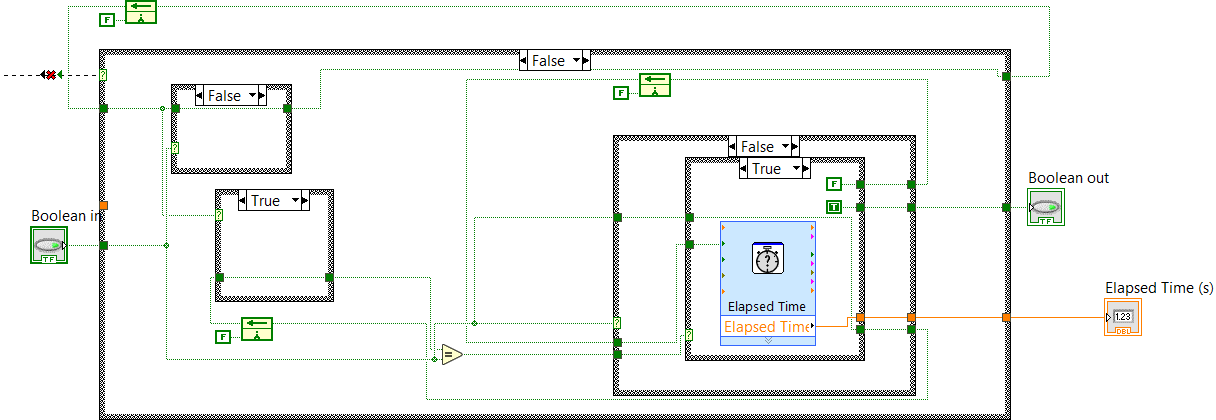
When the first start signal arrives it will change the upper left case structure to true, which will feed a true value to the upper Feedback node, this node will be kept at this value until the main HPLC program is restarted. The value of the Feedback node on the right is also changed to true, this feedback node is responsible for timer resetting, only active if inject signal goes from true to false. Also the third Feedback node is changed to true, which is responsible to mark the change of the inject signal with a comparison between the current and previous inject signal value (for every two changes of this node the right Feedback node changes once).



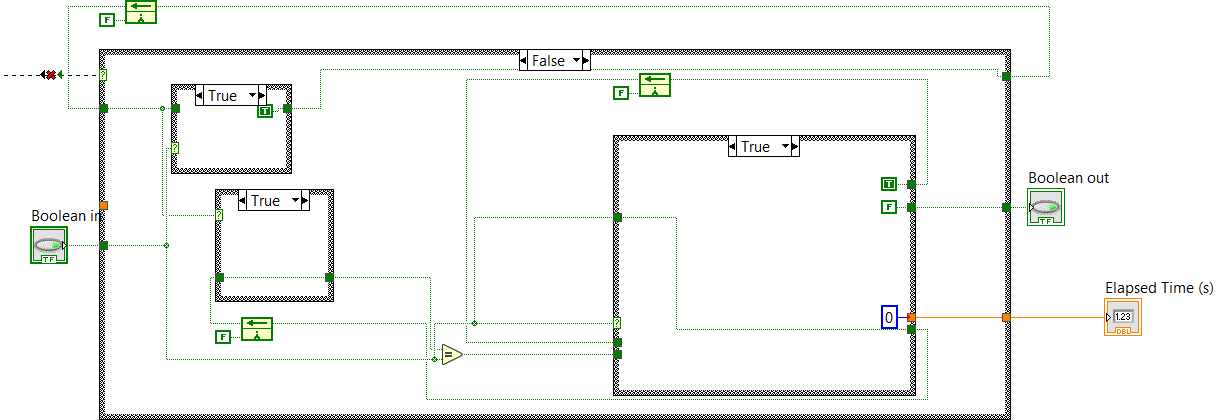
After the first execution the left upper and lower case structure changes value because of the feedback node, the inject signal is still true ( remember the HPLC’s inject signal lasts for one second the sample taking time is around 5 Hz).



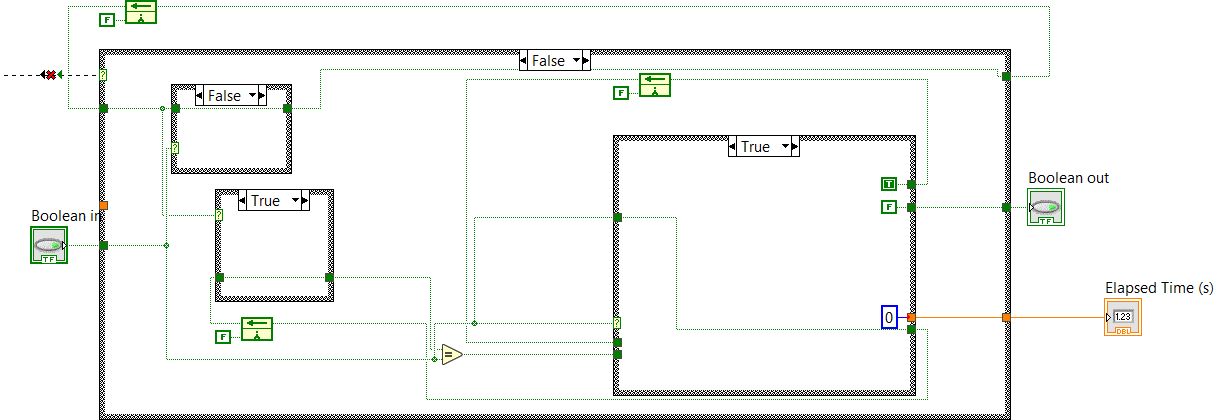
After the inject signal is set to false (measurement round started) it will change the case of the Case structure to the right, and the right Feedback node will give a start signal to the timer, in the next cycle its value is set back to false and it will not restart the timer. The value of the lower Feedback node changes to false, in this way the right inner case structure will be kept at True case until the inject signal does not go to true again (next injection). A true Bollean out marks that the measurement is running and the Elapsed time is sent to the main VI.



When the next injection signal appears it will set the upper Case structure to True, but this does not affect the upper Feedback node anymore. The other two Feedback nodes are changed to true, the one on the right side marks that is ready to set the timer, while the lower left Feedback node changes marking the current change in the inject signal. Measurement Boolean changed to false, because measurement run is off and the Elapsed time set to zero.



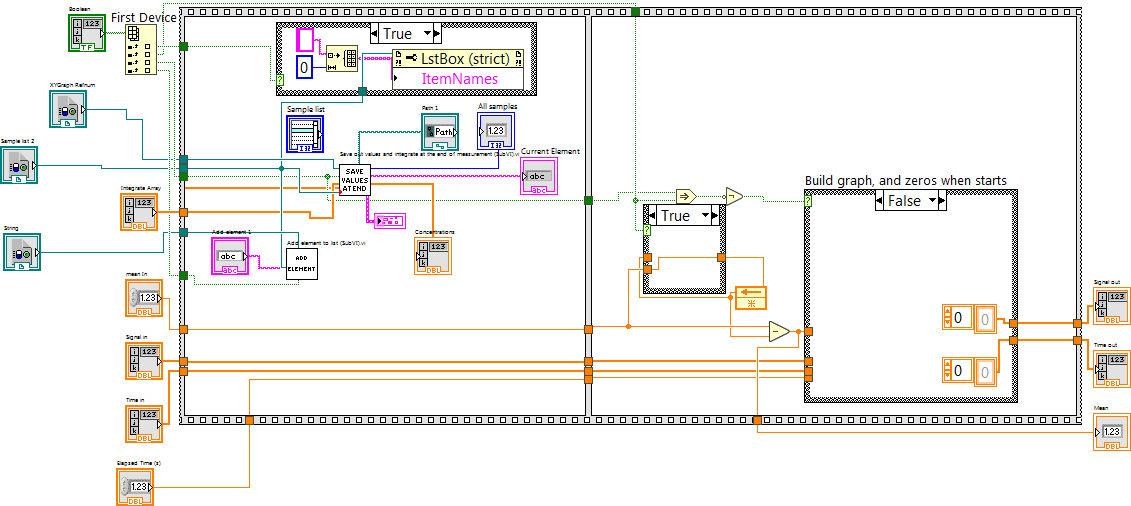
After the inject signal stays at false for two rounds the measurement time will be started and the cycle continues.

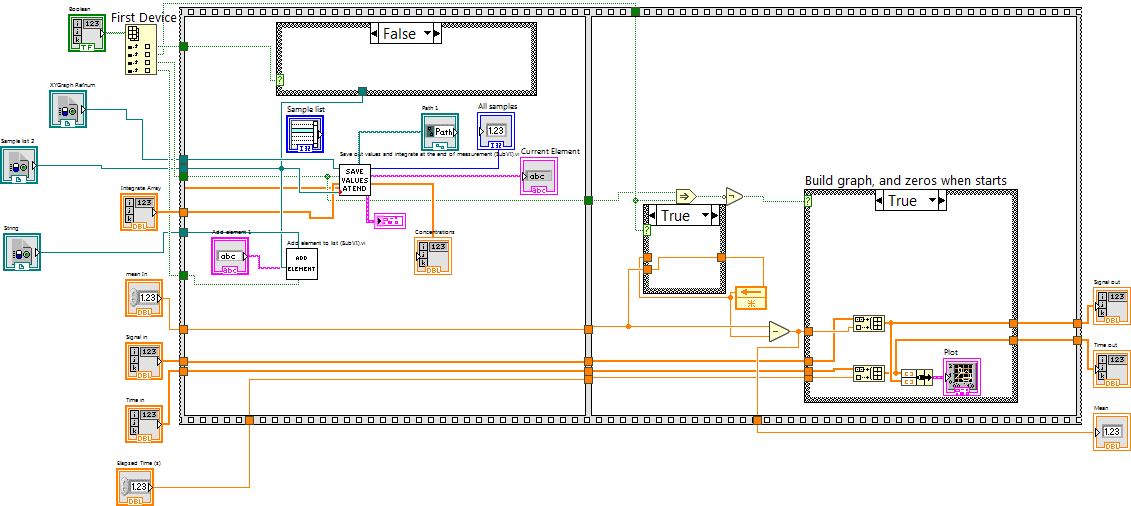


#### Full Front Panel

The inputs are all the Booleans packed in one array, the current Mean and its previous values in an array, the Current time and its previous values in an array, the Refnums of the plot, the Sample list and the Add element text box, plus the integration parameters (the last one is only used in the case of Panel 1). The Boolean values represent the state of ‘Measurement running?’, the Reset list, Add element and Zero baseline Booleans. If Reset list is activated the Itemnames Property node of the Sample list is set to ‘ ’, which erases all the entries in the list (left Case structure). Add elements Sub VI will be discussed later. Zero baseline button sets the Feedback node to the current mean value, which is always subtracted from the measured mean value (central Case structure).

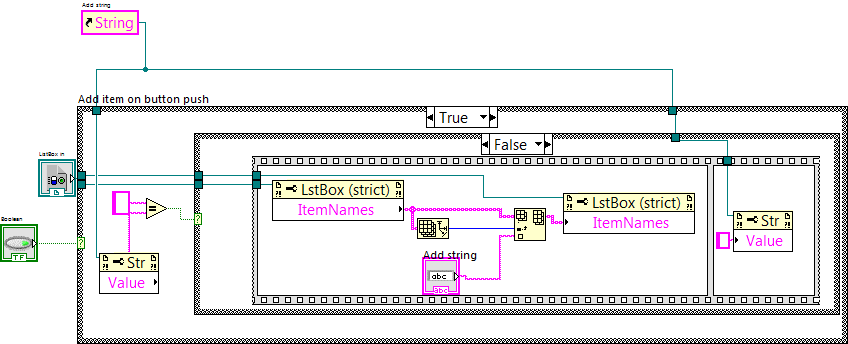
The measurement data is only collected if measurement is running and the baseline is not pressed (right Case structure, negation of measurement implies zero baseline), in other cases the Time and Mean arrays are set to zero. If the ‘Measurement running?’ value goes to true it triggers the Case structure and collects the time elapsed and Mean values in two arrays, visualizing them in the mean time.

****

****

##### Add Element

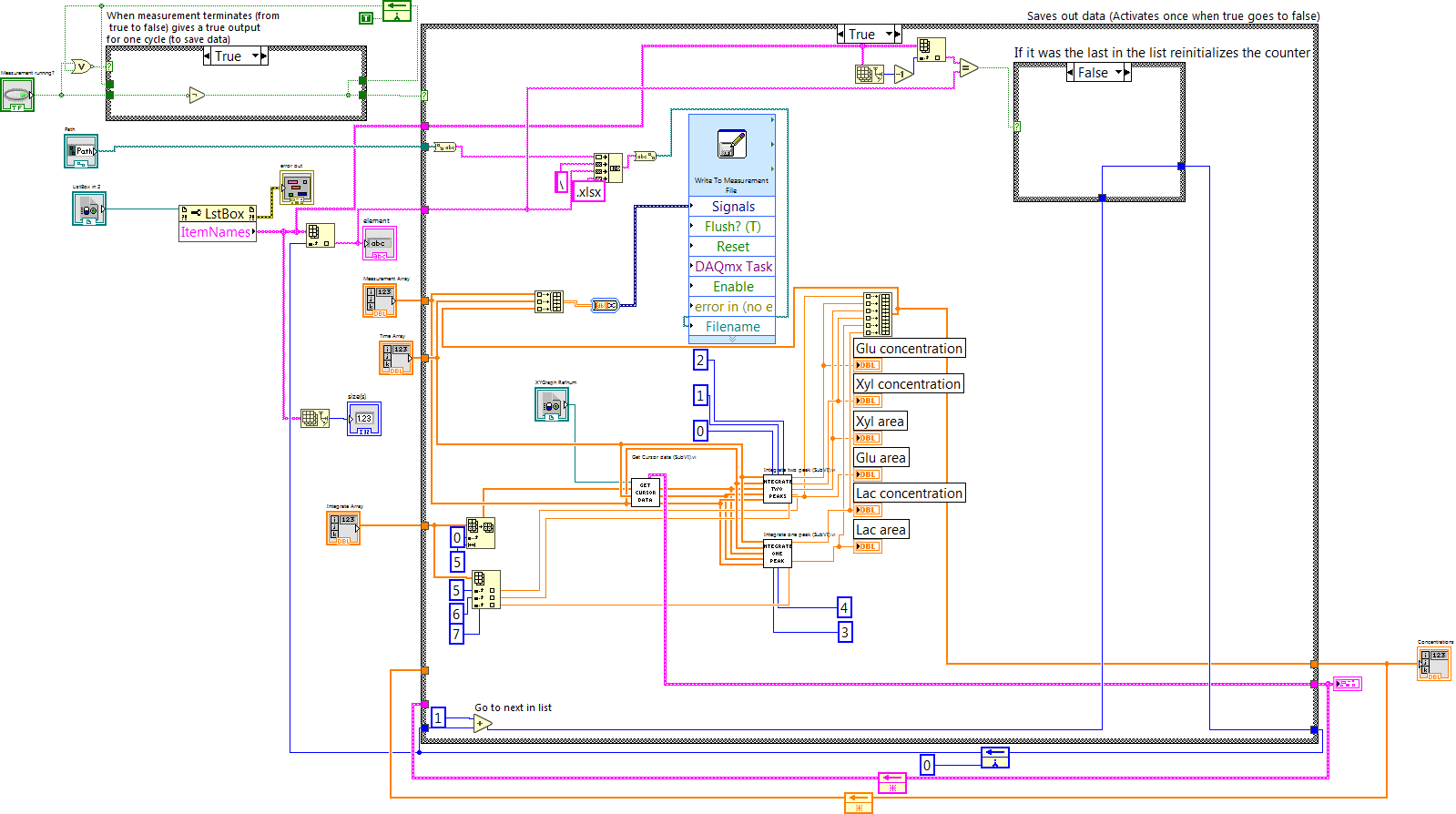
If Add element button is pressed, the following code is executed: A comparator checks if the Value of the string is other than empty (this is to avoid adding empty spaces to the list). If the value is other, it executes a case structure, which adds the string to the last place of the Itemnames property of the list, and sets the string to empty afterwards (the name disappears from the box after it has been added to the list).

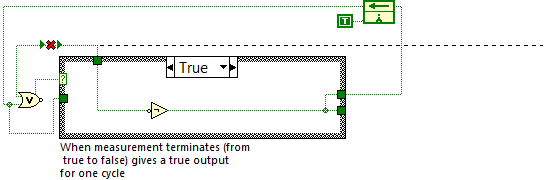


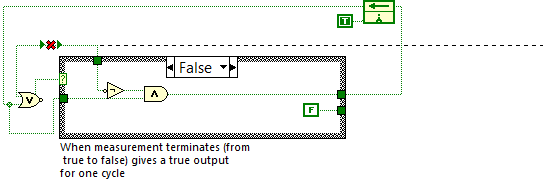
##### Save Values At End

In the upper left corner the same trigger structure is visible. At the start or the measurement input is false and the Feedback node is true, the Not Or (NOR) gives false output on true inputs (only gives true output on two false inputs). This sets the Case structure to false, the two inputs of the And are true and true so the output is true, which keeps up the current state. The output of the whole structure is the false sign in the case structure. When the signal turns to true, the And receives a false and true input, sends a false sign, which sets the Feedback node to false (Trigger). However the Case structure does not change until the input sign does not turn to false, in this case the NOR gives a true output on two false inputs, triggers the Case structure, which gives a true signal to the output strictly once, it sets the Feedback node to true, which is the starting state. This enables a code to be executed strictly once every time the measurement ended. The lower half of the Case structure is responsible for the integration part; the rest is to save out the measurement data.

The Sample lists Itemnames property node’s size is the number of samples on the list (visible at front panel as All samples indicator), the Feedback node starts from 0 (first element) and increases by one every time the Case structure is executed. The element (String) defined by the value of Feedback node is visualized on the Front panel as Current sample. Once the Case structure is executed it creates a file using the Path supplied by the user (Front Panel), the name of the Current sample, with xlsx extension. The recorded Elapsed time and Mean values arrays are saved out. If the current value was the last value, the counter is reset to 0 in the Case structure to the right.



The



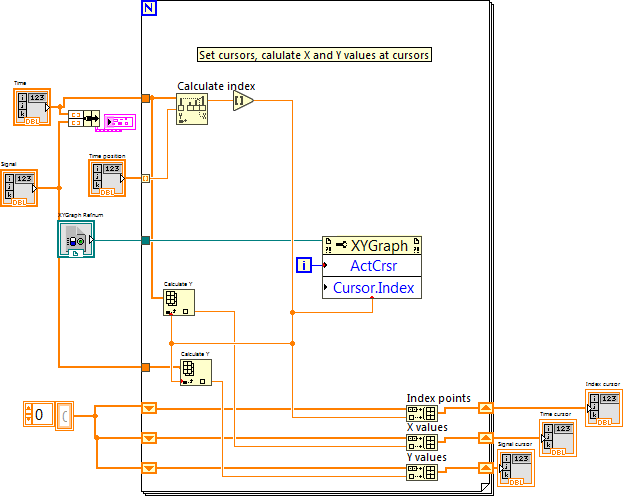
The trigger starts the integration also. This section is strictly written for the fermentation to determine xylose, glucose and lactic acid. The control inputs are the time details for the double xyl-glu peak (start-separator –end), the start and end point of the lactic acid peak, and the standard areas of all three compounds. First the **Get Cursor Data** Sub VI is started (Time-Measurement array inputs and the five peak-time parameters), the output is the graph with the chromatogram, where the cursors are placed in the time moments defined by the five peak-time parameters, and three arrays: the Index of the cursors (this represents the index of elements in the Time-Measurement array where the cursors are), the Time points belonging to the cursors (closely the five peak-time parameters, but not exactly as they are placed on the curve defined by input Time-Measurement array), and the Signal of cursor which is the signal value at the cursors place.

The two **Integrate** Sub VI’s work similarly**:** the inputs are the Time-Signal measured dataset, the three cursor arrays, the numbers indicating the index of the cursors corresponding to the peaks and the standard’s area. The outputs are the concentrations. First the Sub VI calculates the baseline by inserting a linear between the start and end point of the peak, subtracting the corresponding baseline intensity from every measurement point and summing them (numerical integral). It divides the sum with the standard’s area and outputs the concentration.

It has to be noted that for good results the time parameters have to be inputted really precisely, unlike with the Matlab integration tool.

##### Get Cursor Data

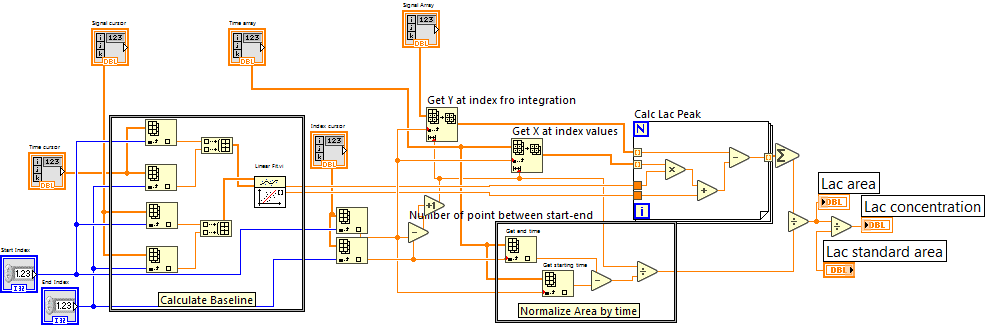
In the first step the signal (S) values are paired with the time (T) values and exported to a graph (this is the measured chromatogram). Afterwards for each of the five integration time value’s (X) the following code is executed in a for loop: the threshold 1-D array element searches for index of the biggest T value smaller than X, this value is rounded, as it does not need to be an integer (Ix) (see Labview help for details). The cursor is set to this time value. An Index array element searches for the Tx and Sx values at Ix. This is the reason why X and TX are not the same, as X can be any value, but Tx has to be a member of T array. All Ix, Tx and Sx values are saved out in three arrays.



##### Integrate One Peak

The Calculate baseline part searches for the Time (T) and Signal (S) values at the points defined by the user (4-5 in this case, which are the start and end points of the lactic acid peak). The values are paired (Tstart, Sstart; Tend, Send) and fed to a Linear fit element, which returns the slope and intercept values.

In the next step a pair of Index array elements search for the index belonging to point four and five, and calculates how much datapoints (D) are between them (subtracting them). Two Array subset elements form a smaller T and S array, starting from the smaller index (point 4) and in the length of D. These arrays are fed to a Calc Lac Peak part, where it calculates the baseline for every T datapoint with the help of slope and intercept, subtract them from the measured corresponding S value, and summarize them for all D datapoints.

The Normalize area by time part calculates the datapoints/s ratio ((Tend-Tstart)/D), and corrects the area with it, this way the measurements are not affected if the sampling time is manipulated. Finally the Lac concentration is calculated with the Lac standard area and the Lac area and concentration values are exported. 

##### Integrate Two Peaks

This is similar to the previous so only the differences are discussed. The baseline is calculated from the start of the first peak and the end of the second peak. The peaks are calculated based on the same idea, by summarizing the differences of the measured signals and the calculated baseline values from the intercept and slope of the linear fit. The Normalization by time is done calculating the datapoints/time between the Start and End (Tend-Tstart/D1+D2). The concentrations are calculated by comparison to a standard area.

In practice there more than two peak overlapping are rare; however based on this idea any number can be solved analogously.

