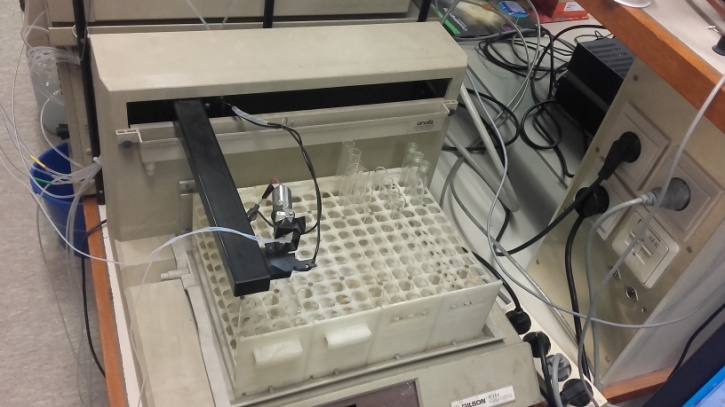
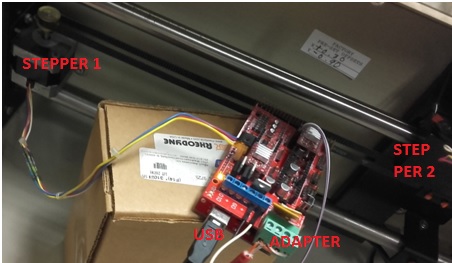
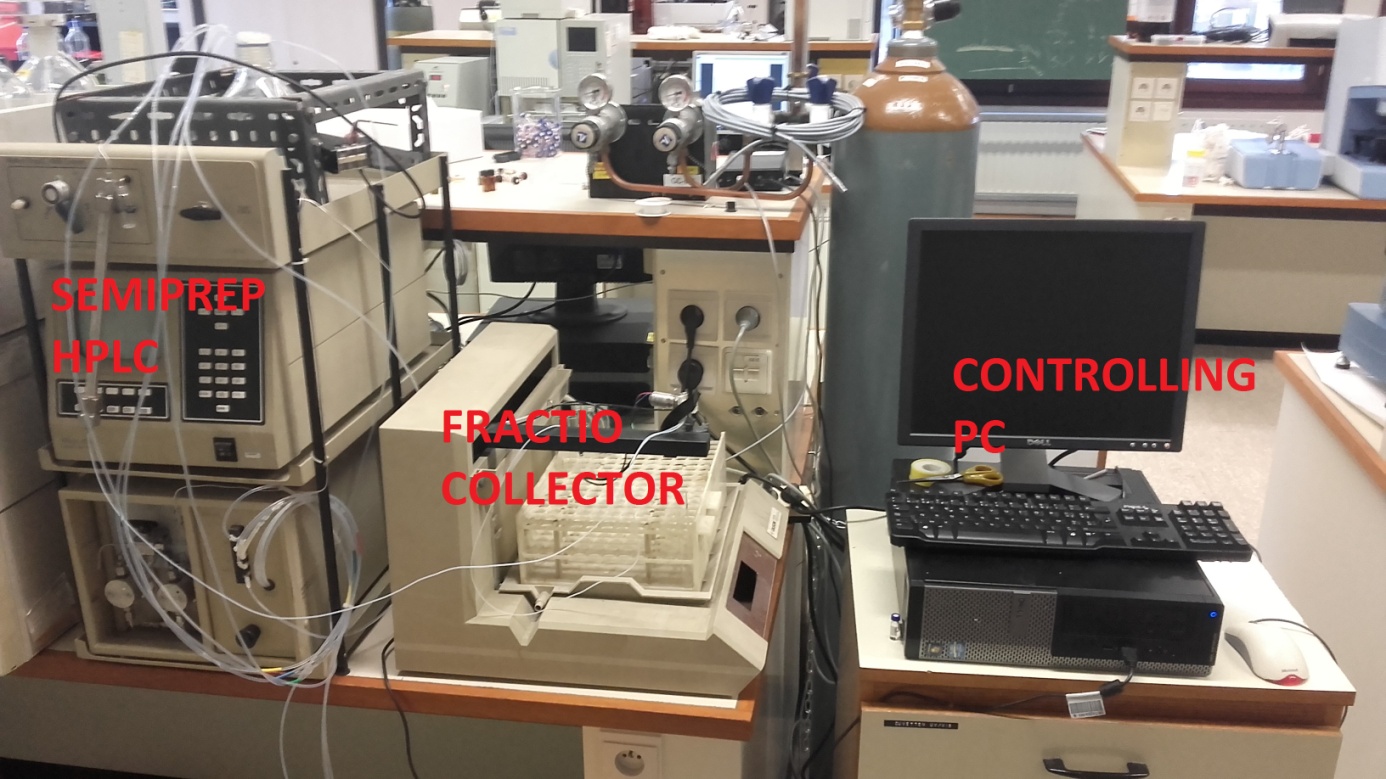
## Fraction collector

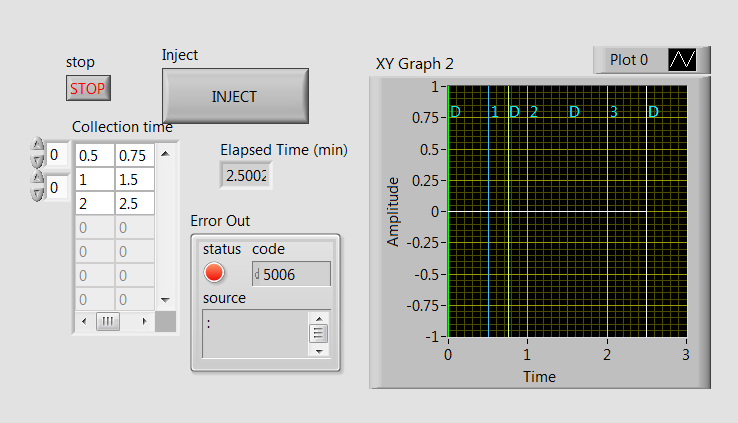
A semi preparative Waters HPLC was connected with a Gilson fraction collector (FC). The FC was a standalone unit itself, with peak recognition software. The electronic circuitry of the FC was damaged and could not be repaired so it was substituted with an Arduino mega board with a Ramps 1.4 board (used for 3d printers). This setup would be capable to control up to five stepper motors and several 12 V on/off outputs, the FC only has two steppers and a 12 V valve. The power is fed to the system by a laptop charger set on 12 V, the communication is via USB cable. The software on the PC is written in Labview with Linx addon (freeware needed for Arduino communication). The Linx upgrade has an .ino file, which has to be uploaded on the board to enable Arduino-Labview communication. Two different programs were made, one with peak recognition ready to be coupled with an HPLC software, and another as standalone unit.

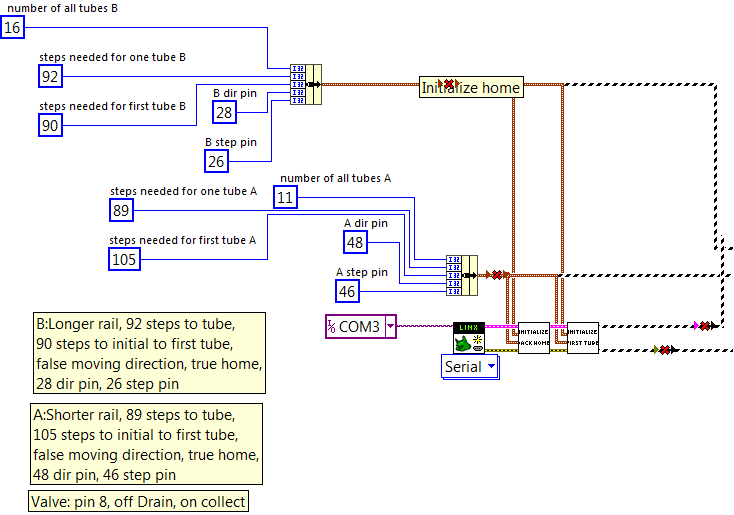




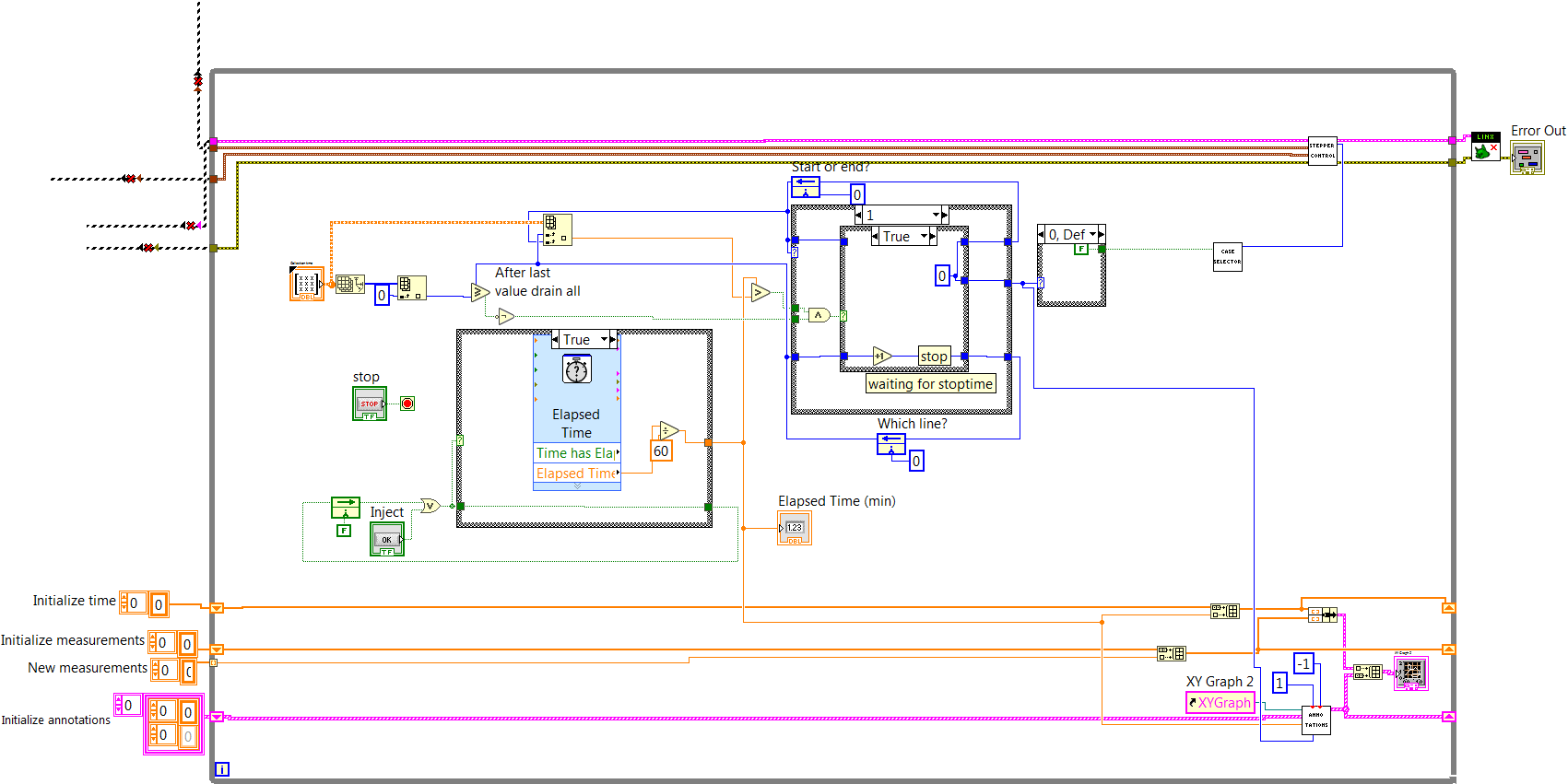
### Standalone unit

The standalone unit program is capable to only collect fractions at set time intervals, and cannot be connected with a detector. The setup is much simpler, but exact previous knowledge is necessary about retention times. After the inject button is pressed the timer starts and the graph will visualize the fractions, marking the index of tubes where the sample is collected in or ‘D’ marking drain periods.

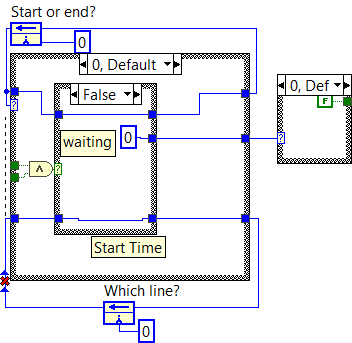
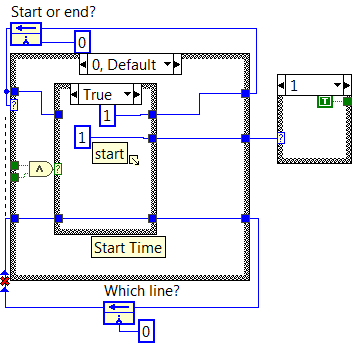


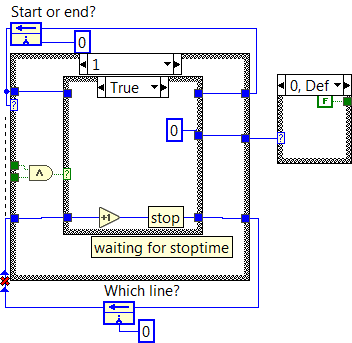
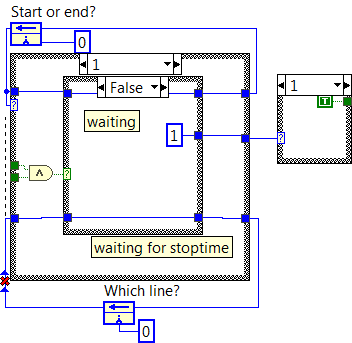


In the first step the pins are assigned for Arduino boards (direction and stepping pins), the number of tubes and steps needed for each tube. The USB port of the Arduino is set and fed into the Linx element to set up the communication. The **Initialize Backhome** Sub VI steps the valve back to the corner closest to the first tube (the head does not have displacement sensors, it has to go to a known point), while the **Initialize first tube** Sub VI sends the head above the first tube.



The next part of the program is the loop. By this time the variables are initialized (Time, Measurement and Annotation), and the valve head is in a known position. The case structure on the left starts the timer when the inject button is pressed and keeps it on afterwards (the Feedback node keeps the true value). The lower part of the program conserves the measurement data (Time and Measurement Arrays) and conserves/calculates the annotation details (the visual marks on the graph). The part containing the double case structure in the middle is responsible for reading in the values from the Collection time array on the Front panel, containing start-stop pairs in lines. The values in the array are chosen by the number of the line they are (Feedback node under the bottom of case structure) and place in the line (0-start, 1-stop value, Feedback node at the top) in an Index array element. The chosen value is compared with the Elapsed time, and if it is passed a true signal is sent inside the Case structure. The case selector for the outer Case structure is the upper Feedback node (start/stop time) as the two require different actions (open/close valve). The lower Feedback node is compared to the number of all lines (Size array measures the dimensions of the Collection time array, an Index array chooses the number of lines in the array), if the lines executed exceeds or equals the number of lines in the array the desired collections are finished, the output will be true. Before that the output of the comparator is false which is negated before it is sent to the Case structure. The two Case structures define four cases altogether: ‘Currently draining and waiting for start time’, ‘Starting collection’, ‘Currently collecting waiting for stop’, and ‘Stop’. The case structure to the right controls the valve; true output means collection, false drain.



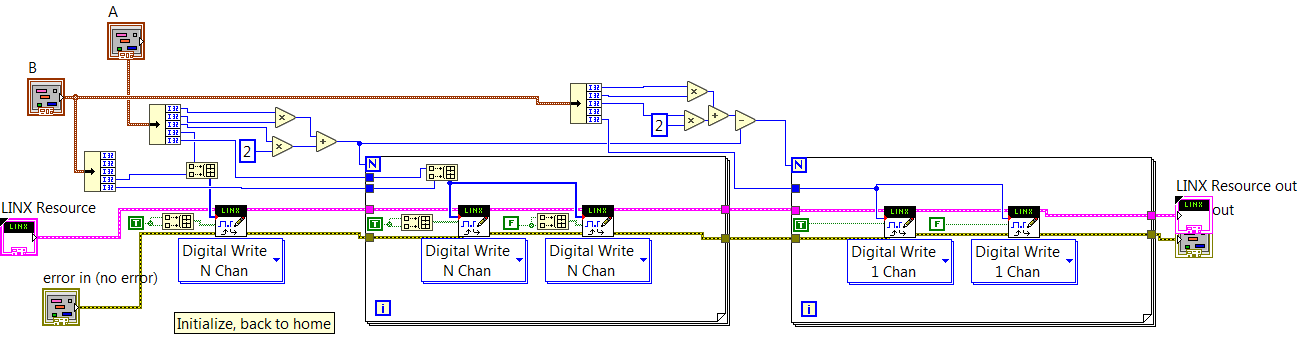
In case of ‘Currently draining and waiting for start time’ the upper Feedback node is zero, the sample is currently drained (false signal in Case structure to the right), the output of the And comparator is false until the collection time is exceeded. When this happens the ‘Starting collection’ case appears, runs only once, sets the valve to collection mode, sets the upper Feedback node to 1, which will change the Case structure to ‘Currently collecting waiting for stop’. In this mode the valve is set to collection, and the structure is waiting for the stop signal from the comparator outside. When this arrives the ‘Stop’ case runs only once, the valve set to drain, the upper Feedback node set to 1, and the value of the lower Feedback node increased by one. The next is ‘Currently draining and waiting for start time’ and the cycle starts again. It has to be seen that the number of executed lines (lower Feedback node) will exceed the number of all lines in the array in ‘Currently draining and waiting for start time’ case, as the Feedback node’s value was increased in the ‘Stop’ case just before. This will send a false signal in the And comparator and keep its value false leaving the ‘Currently draining and waiting for start time’ structure active until the program is not stopped with the stop button.

The valve controlling Case structure to the right sends its signal to the **Case selector** Sub VI, which gives signal to the **Stepper control** and **Annotations** Sub VI’s, these will be discussed under.

#### Initialize backhome

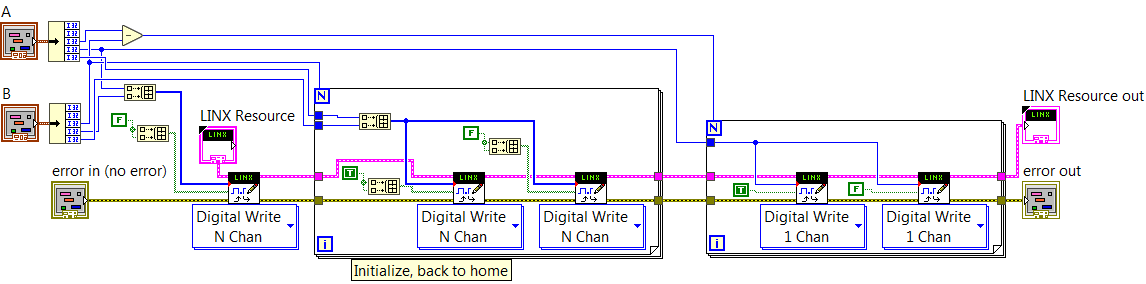
This Sub VI steps back the valve head to the corner closest to the first tube. Since there are no placement sensors in the head this is done by assuming that the head is in the opposite corner and stepping it back all the way. The left side of the algorithm sets the two direction pins (A and B) to the right direction (true-digital on- for both).

The number of overall steps needed for both steppers is calculated by multiplying the number of all tubes with the steps needed for one tube, adding twice the steps needed from the corner to the first tube (the multiplication with two is just to be sure it gets there). This value is the turns needed for each stepper to reach home-corner; hence they are connected as loop counters for the for-loops. The first loop steps both steppers, the second only steps B, as it needs overall more turns than A (counter is set by the difference between overall turns A and B). Programmatically it would be easier to step both steppers separately, but this decreases the speed of movement significantly.



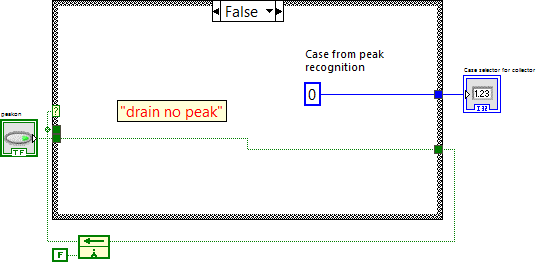
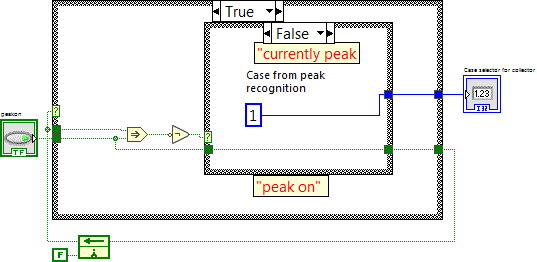
#### Initialize first tube

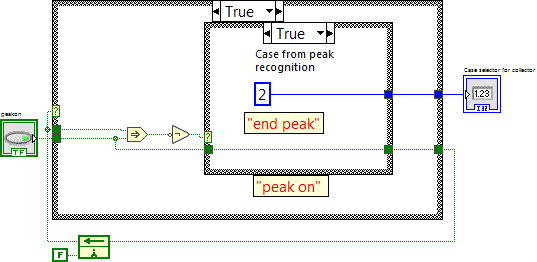
This Sub VI is very similar to the previous one, the only difference is the value of the direction pins (now set to false), and the number of steps (for both steppers this is the steppes needed for first tube).



#### Case selector

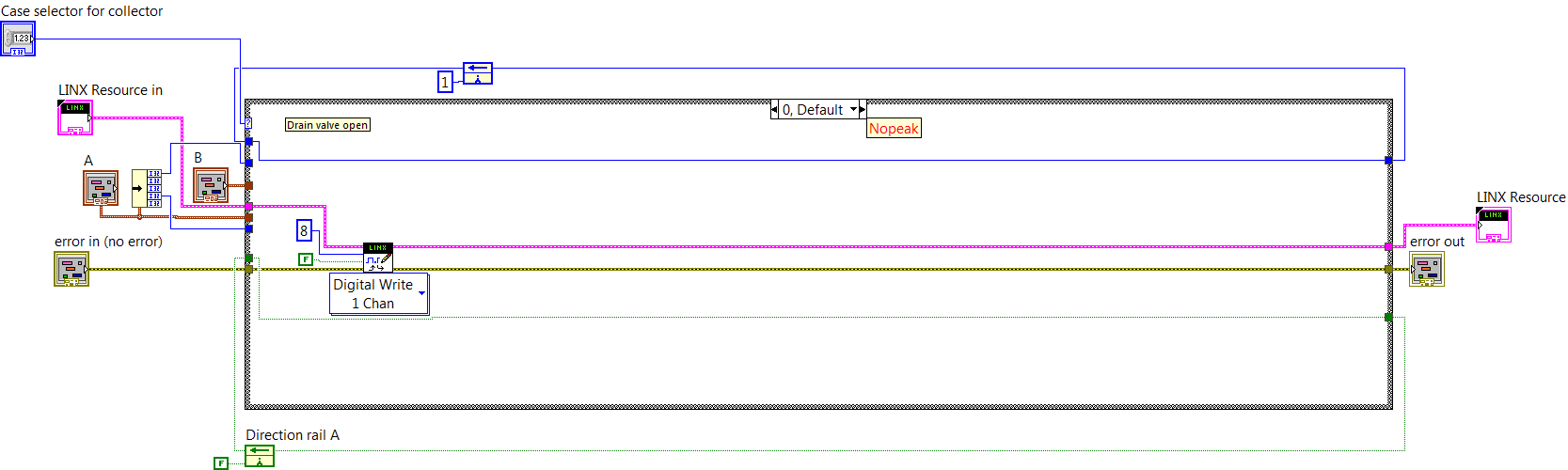
This Sub VI is to choose between three cases: ‘Drain’, ‘Currently peak, collect’ and ‘End of peak’. There is a Feedback node preserving the previous value of the valve switch. The firs two cases are straightforward, the ‘End of peak’ case activates when the previous state of the valve switch was true and now it is false. The differences between the three cases are significant in the **Stepper control** Sub VI.

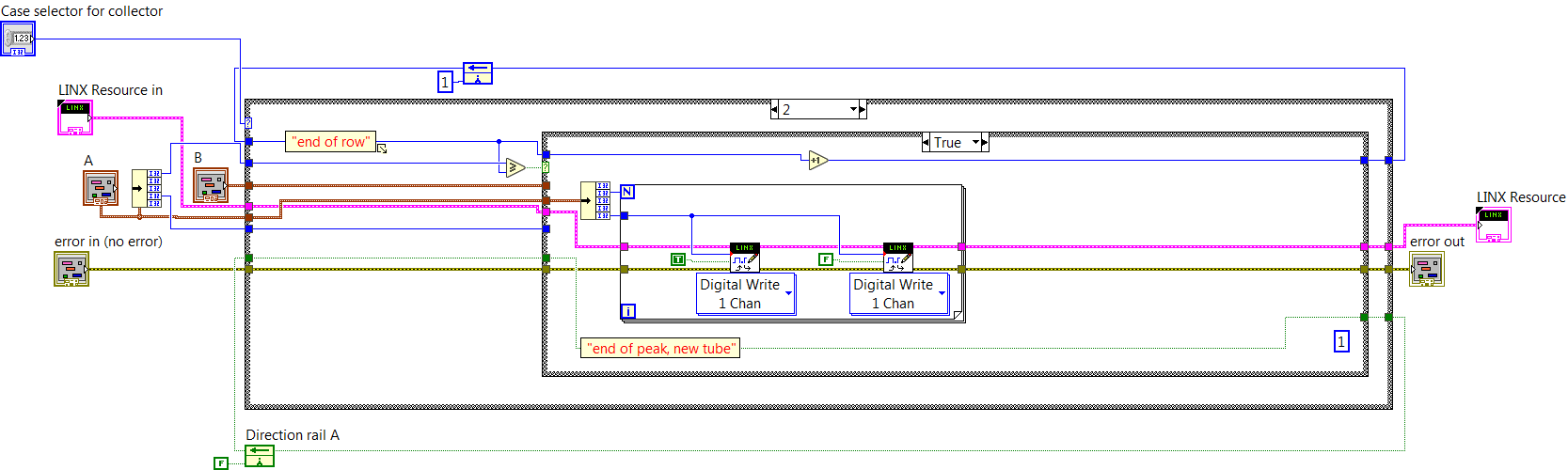
 



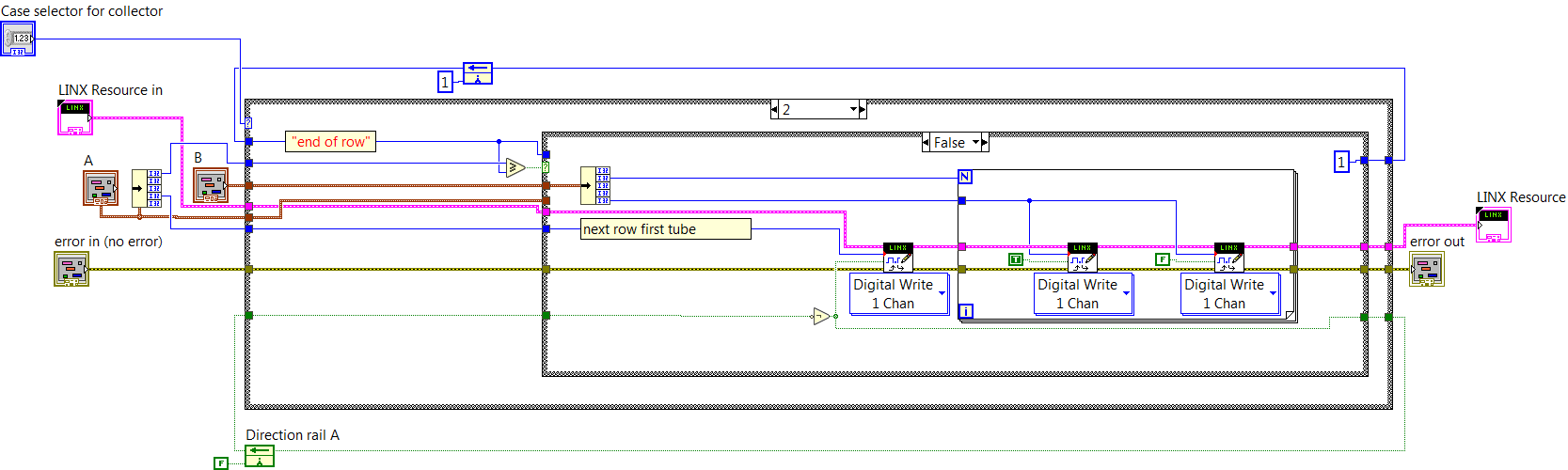
#### Stepper control

Four different cases are coded in this Sub VI: ‘Drain’, ‘Peak’, ‘End of peak move to next tube’, ‘End of peak move to next row’. In the first two cases only the valve pins value is set on or off. In the other two cases the valve head has to be moved. The upper Feedback node shows which tube is currently in use, the lower Feedback node shows the direction of the stepper movement.

If the number of all tubes in the line is bigger or equal to the tube index currently in use, it will move it to the next tube in line (steps needed to next tube with A) and increase the index of used tube with one.



If the number of all tubes in the line is smaller than the tube index currently in use, it will move it to the next row (steps needed to next tube with B), set the index of used tube to one and change the direction of stepper A.



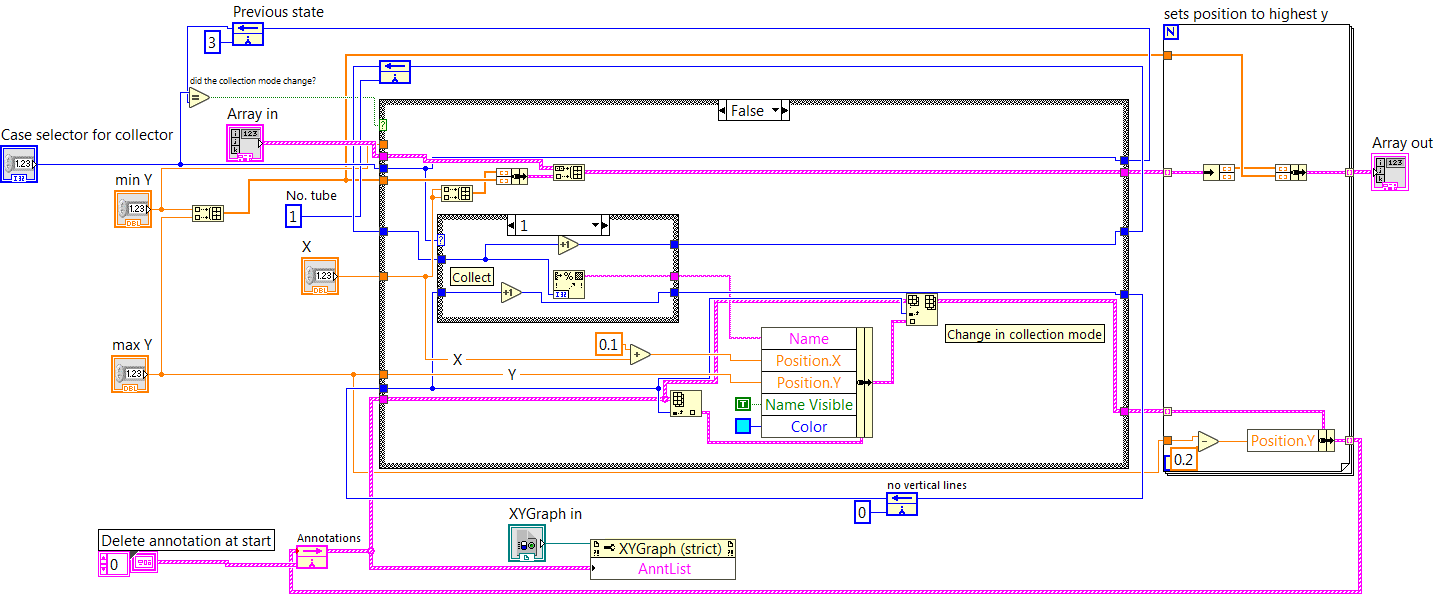
#### Annotations

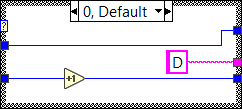
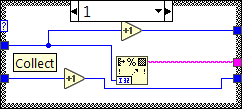
This Sub VI creates vertical lines on the graphs at the time moments when the collection starts or stops and adds a ‘D’ letter to the drain periods and the index number of the tube where the sample is collected. The inputs are the min and max value for Y (detector signal, set to -1 and 1 as the detector is not coupled and the signal is constant zero), the case selector value (0,1 or 2), X (Elapsed time), Array in (Array of annotations) and the Refnum of the XY graph. The output is the renewed annotation array.

Three Feedback nodes remember the previous Case selector value (starting value 3), the number of previous tube used for collection and the number of lines already on the graph.the structure is run when the previous case selector value is not the same as the current. Because of the Feedback nodes default value it will run once it is called and afterwards every time the value changes. The Case selector on the left will choose to write D or the number of the tube, depending on the value of the Case selector input. If it is collect cycle, it converts the number of tube integer to a string and sends it further, increasing both the number of lines and number of tubes Feedback node with one. In drain cycle only the number of lines is increased.

The X point of the annotation is increased by 0.1, so it will not be directly on the vertical line. The Y point is also set, but this is not important, as it will be changed latter. The Bundle by name element in the bottom sets the details of the last annotation (X, Y position, Visibility, Color and Name). The vertical lines are set above the left case structure; each line is defined by four parameters: start and end X and Y. Since the line is vertical, the two X values are identical, Y are set to min and max value (-1 and 1).

The for-loop on the right resets all the Y values to the current min and max values, as if the detector reaches a peak the increasing Y values would visually shrink the previous vertical lines. This is done by looping through all the values in the array, and resetting the Y pairs to the current values. The place of annotations is set to a bit less than Ymax, so the letters will be visualized a bit under the end of the lines.

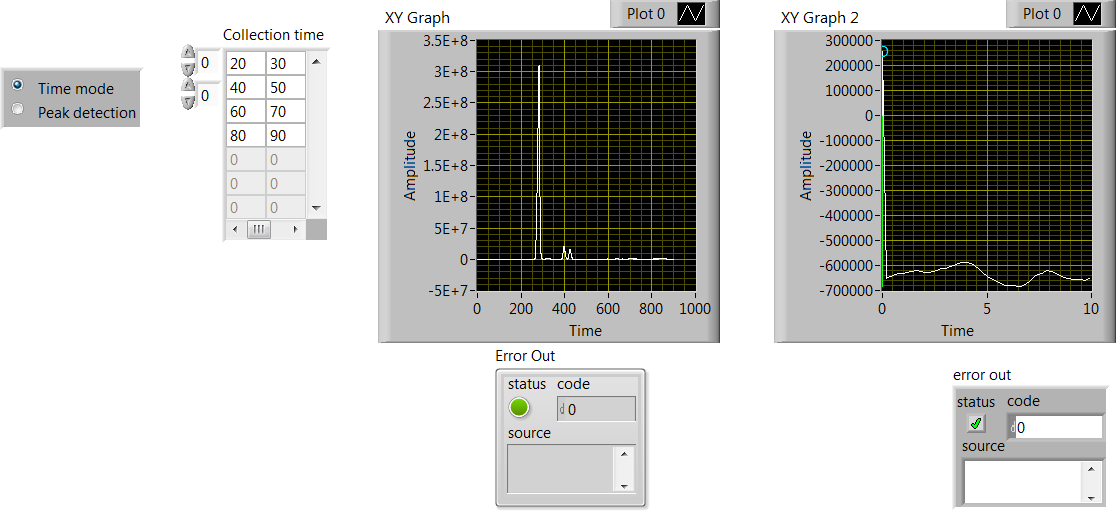


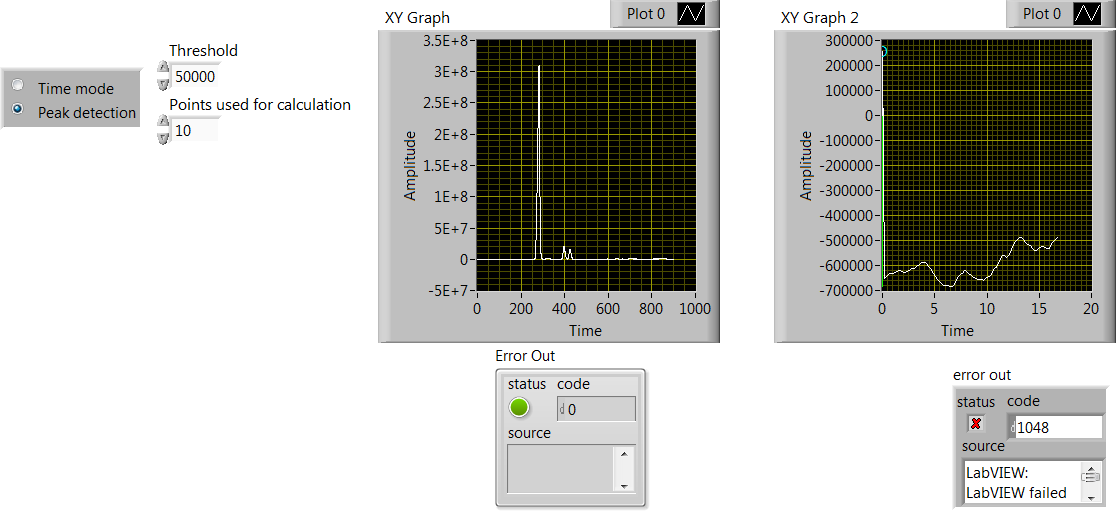
 

### Coupled with detector

The FC is also capable to work coupled to a detector, inserted in the HPLC software for example. It can be run in either peak recognition or time collect mode. Since the FC was planned to work in only time collect mode, this program was not integrated in the software, just an offline version was delivered, where one recorded measurement was fed time point by time point to the system. The program needs Matlab to be installed on the PC (peak detection uses and embedded algorithm).

Two slightly different front panels are visible based on which radio button is pressed down. In case ‘Time mode’ is active, a similar array is visible as discussed in the Standalone part. In case of ‘Peak detection’ two input boxes are visible: Threshold sets the sensitivity of detection and Points used for calculation defines how many points are used for slope calculation. The rest of the front panel is the same: one graph is showing the whole chromatogram, while the second one the current measurement, marking the Dran and Collect phases with lines and letters. The two boxes at the bottom show error codes for the Arduino board and the Matlab algorithm.

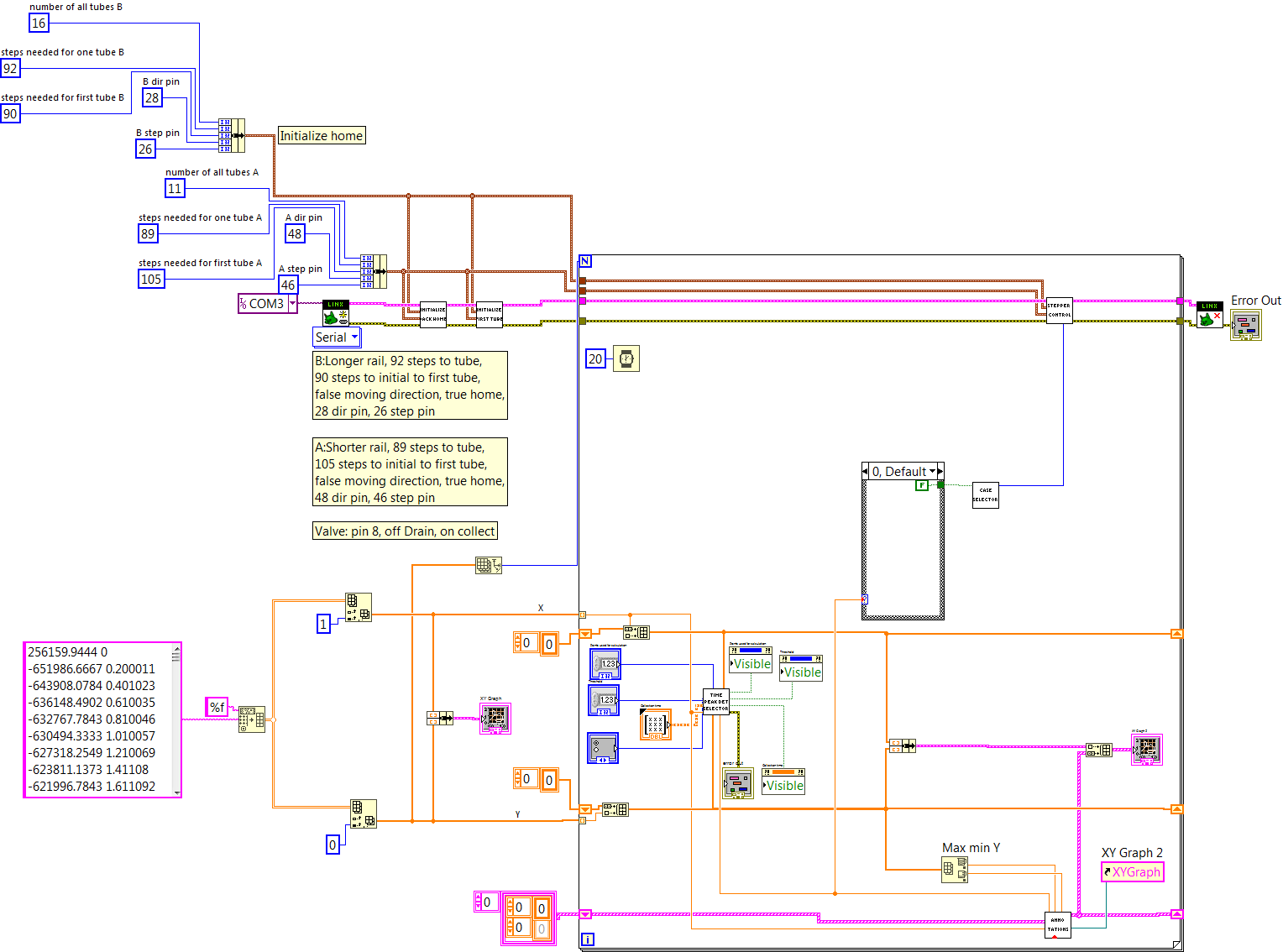




Since the Standalone unit was discussed in details, only the differences will be highlighted here. The upper part, responsible for the Arduino board communication is exactly the same as in the previous case, same is true for **Case selector** and **Annotations** Sub VI’s. The lower left side of the algorithm outside the loop feeds the recorded measurement values one-by-one to the rest of the code. First a Spreadsheet string to array element converts the txt file to array, next two Index array elements separate X (time) and Y (measurement) vectors. An Array size element measures the elements in Y vector and feeds it to the for-loop as index (the for-loop has to run once for each measurement point). Next the two vectors are united and visualized on an XY graph (the whole chromatogram). The values are fed one-by-one via an Auto indexed tunnel. When the algorithm will be connected to a detector the outside of the loop has to be discarded and the inputs directly connected to the loop.

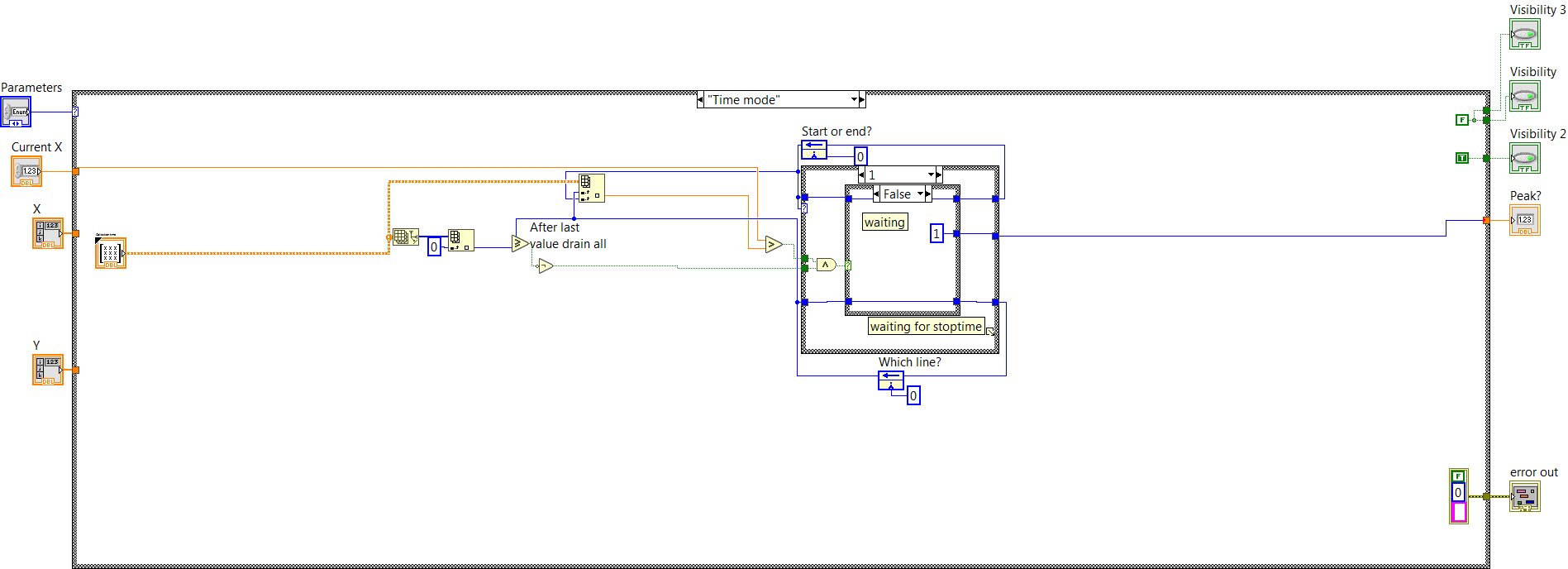
The loop runtime is set to 20 ms, this way it runs relatively fast, but details are still visible. Inside the loop the measurement values are united in an array, and visualized on the second XY graph. The current X value is fed to the **Annotations** Sub VI, as this will define where the lines will be drawn. The **Annotations** Sub VI’s other inputs and outputs are the same, difference is that the minimal and maximal value of Y is taken from the current dataset via an Array max & min element.

The heart of the code is **The Peak Detector** Sub VI. Inputs are the X, Y arrays, X current, radio buttons (which mode is active?), parameters (Threshold and Points used for calculation) and the Collection time array. Outputs are the Peak? (peak indicator), three visibility Property nodes (Collection time array and the two parameters) and an error handler for the Matlab part.



#### The peak detector

If the Time mode radio button is pushed the Sub VI’s is very similar to the one discussed in Standalone unit part. The difference is that the visibility of the Parameters on the Front panel is set to false and the visibility of the Collection array is set to true.



If the Peak detection radiobutton is pressed the embedded Matlab structure appears, the visibility of the Parameters will be turned on and the Collection array off. Two vectors, length set by Point used for calculation, are fed to the script, together with the Threshold value. Three Feedback nodes memorize the current peak state (Peak on or off) and two inner counters (k and trigger). The calculations are based on the first derivatives (slope of linear fitted on the points), if the derivative is higher than threshold for 5 consecutive values the peakmode is turned on, if it is smaller for 10 samples it is turned off.

