The class u3BaseFilter extents the class u3Base and adds filters to analog inputs for LabJack device.

Carl Friis-Hansen November 2009 , http://computingconfidence.com/u3 , http://fiskeryd.com/svn/u3base/ , http://labjack.com/support/u3

```
class u3BaseFilter : public u3Base
  public:
    friend class u3Base;

                    //  Constructor
                    u3BaseFilter(      int devicePath );       //  If devicePath is 0 then U3_DEVICE_PATH is used.

                    //  Destructor
         virtual ~u3BaseFilter(        void );

                    //  Filter is disabled for all analog inputs by default (0), but can be enabled by
                    //  individually setting a time constant to a positive value larger than 1.
                    //  By setting the timeConst to 1200, the time constant for the filter will be 1 minute
                    //  assuming a sample rate of 20 Hz. (60sec*20)=1200
         int        setTimeConst(      const char      *sFECio,       //  FIO0..7, FIOT, EIO0..7
                                       int             timeConst );   //  0...any positive value over 1
```

timeConst = time * sample
  Where time is the wanted filter time in seconds and sample is the sample rate in Hz.

Example:

```
  u3b->setTimeConst( "FIO0", 60*20 );  // Create a 1min filter assuming 20 Hz sample rate.
```

The above line creates an input filter for analog input "FIO0" with a filter time of 60sec.
The sample rate is determined by you main loop. On modern computers the 20 Hz rate is very far from any limit, thus it should be okay to increase sample the rate by at least 10 fold, which would give 60 * 200 for a filter time of 1 minute.

How does the filter work?
The filter is obviously a running filter type. It simply adds to current value a fraction of the difference compared to previous value. The fraction is the difference divided by filter time in second and divided by number of samples per second. It is the filtered value that is presented to the rest of the system.

When to use filters.
In principle you should always use filters. The reason being that A/D converters are generally designed in a way that causes them to give fluctuating results on several of the las binary digits. Even more important is the environment surrounding the A/D converter and the LabJack. An environment that is mostly far from pretty, filled with transients from computers, relays, mains and other electronic equipment.

How fast or how slow should the filter be?
The most trustworthy results are obtained with a precisely calculated input filter. Setting the filter too slow and you will not experience important events and if an alarm is associated with the analog input, then this alarm might not be triggered. Let me note here that alarm for min and/or max voltage can be delayed so that short analog spikes do not trigger any event. If you make the analog filter too fast, than you might end up with meaningless glitches that does not come from the measured object.

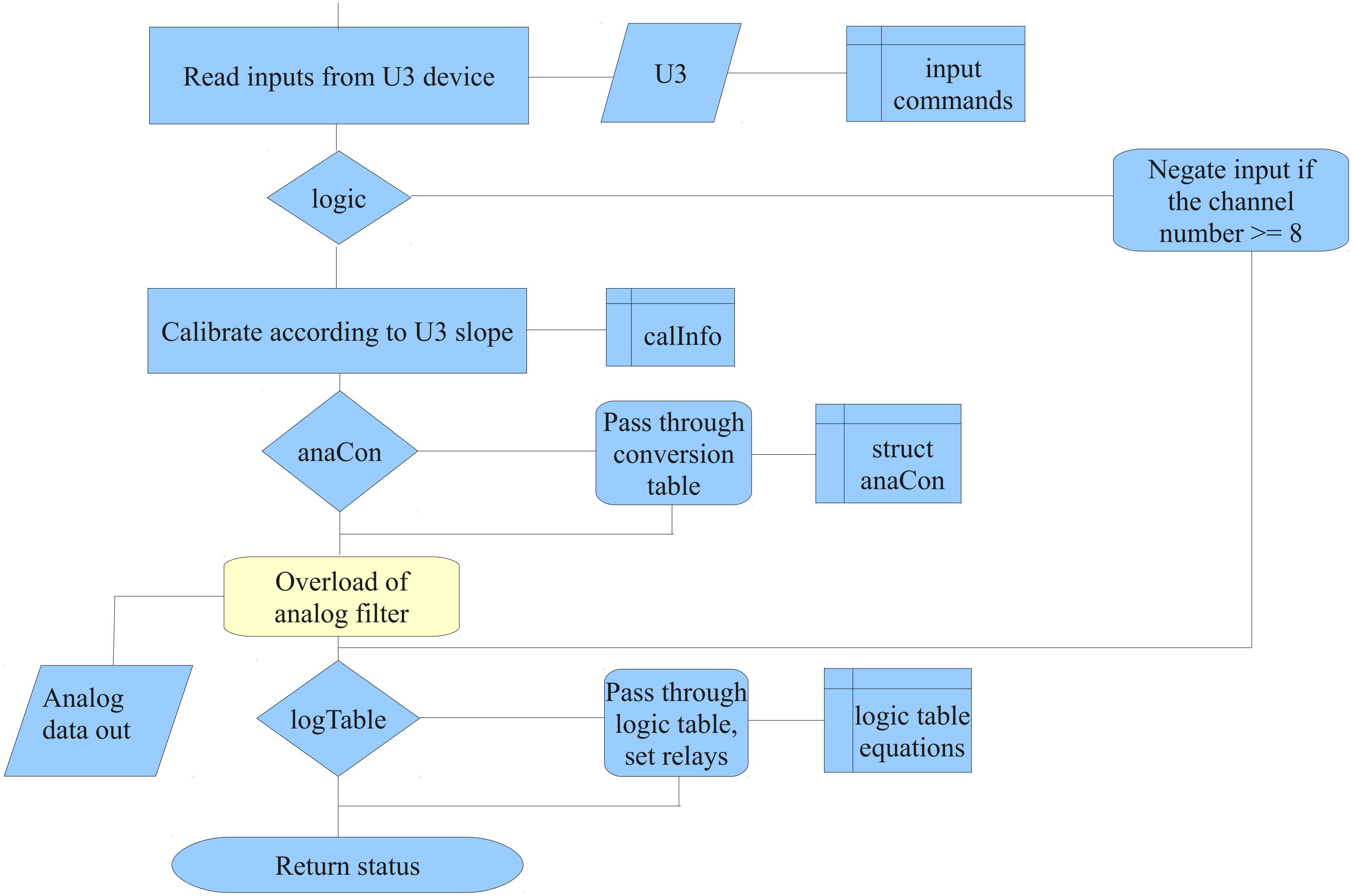
Why is the filter class not just a part of the main class?
This filter class is separate for several reasons. Firstly the filters reduces the max sampling rate, so if one wants to do extremely fast measurements, it is best to have the possibility to totally exclude the filter section. Another reason to make the filter as an extension, is that you might want to use an active filter instead with much sharper cutoff.

Things you need to put in main:

```
 u3BaseFilter  u3b( 1 );       //  Construct the class to interface to U3 device number 1
```

The above function automatically calls the parent class u3Base.

From the block diagram below, the filter is identified as the pale yellow block named “Overload of analog filter”. The filter is applied after the calibration and converter units, but before going through the logic tables and therefore before any action or recording can be taken.


Read inputs from U3 device
U3
input commands
logic
Negate input if the channel number >= 8
Calibrate according to U3 slope
calInfo
anaCon
Pass through conversion table
struct anaCon
Overload of analog filter
Analog data out
logTable
Pass through logic table, set relays
logic table equations
Return status