Moku FIR Filter Builder

User Manual

LIQUID
INSTRUMENTS

Table of Contents

Introduction

Moku FIR Filter Builder provides fine control over the filter's response in both the frequency and time domains, allowing you to match performance to the requirements of your application. Choose from a range of common frequency response shapes, impulse responses, and window functions. These options make it straightforward to control characteristics such as transition sharpness, sidelobe suppression, and phase linearity to optimize filters for each precision measurement or signal conditioning task.

This manual is intended to help users understand the user interface and underlying architecture of the instrument. It also includes a general example in the quick start guide and a small number of in-depth examples to provide a foundation for new users.

These user manuals are tailored to the graphical interfaces available on macOS, Windows, iPadOS, and visionOS. If you'd prefer to automate your application, you can use Moku API; available for Python, MATLAB, LabVIEW, and more. Refer to the API Reference to get started.

AI-powered help is available to aid both workflows. AI help is built into the Moku application, and provides fast, intelligent answers to your questions, whether you're configuring instruments or troubleshooting setups. It draws from Moku manuals, the Liquid Instruments Knowledge Base, and more, so you can skip the datasheets and get straight to the solution.

Access AI help from the main menu ☰.

For more information on the specifications for each Moku hardware, please refer to our product documentation, where you can find the specifications and the FIR Filter Builder datasheets.

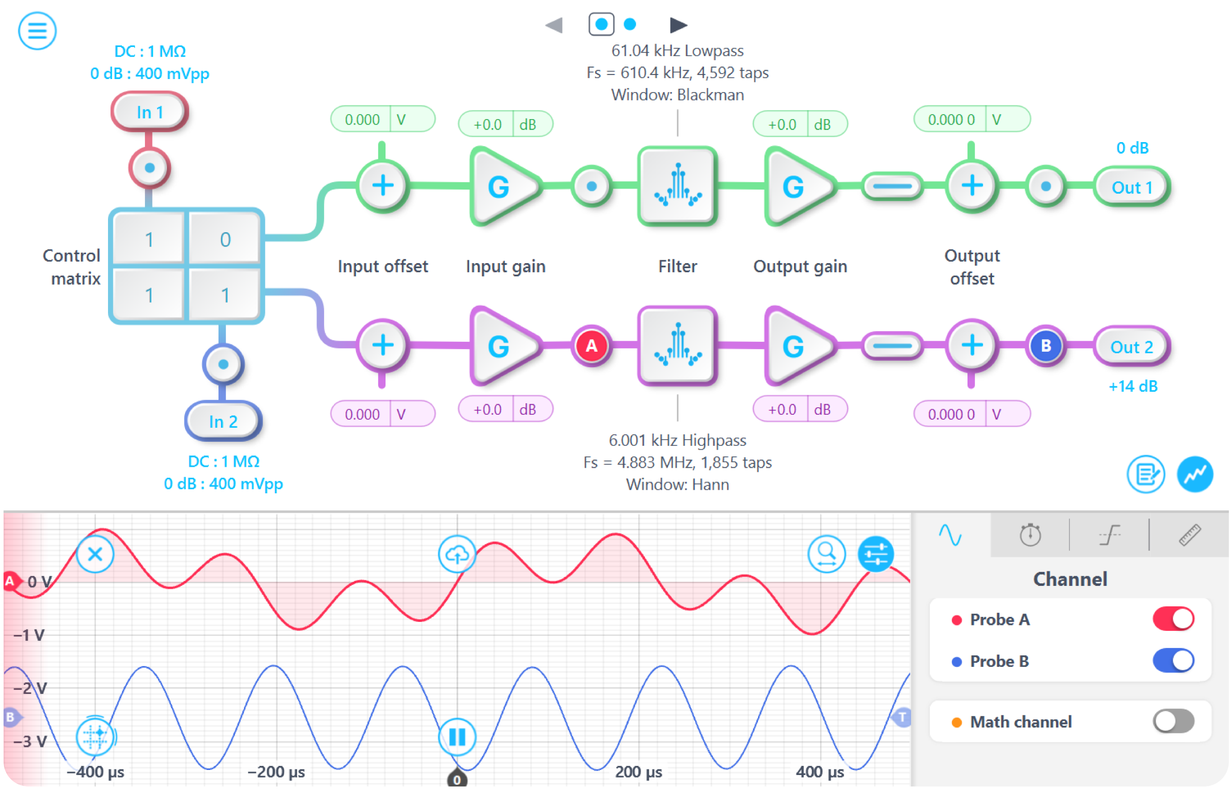
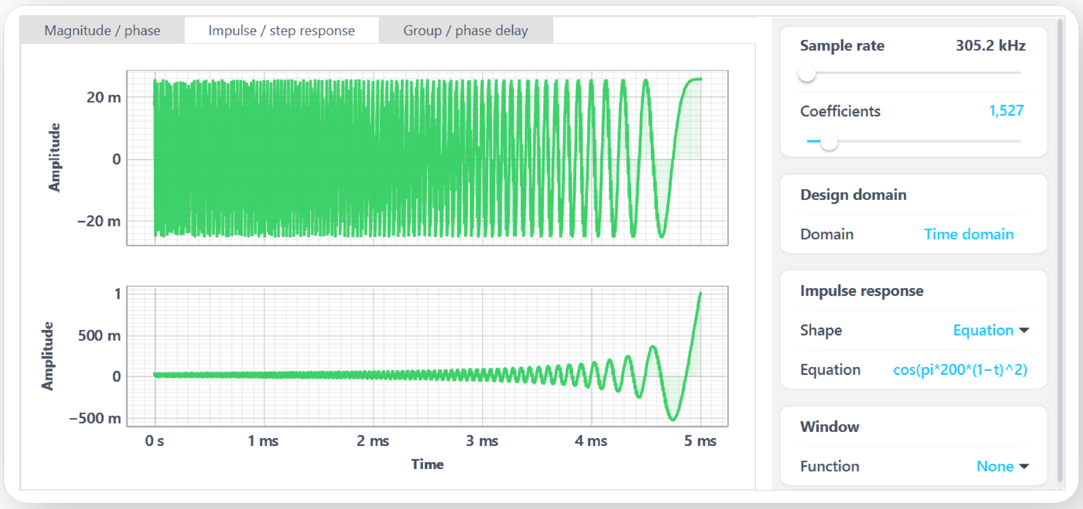


Figure 1. FIR Filter Builder user interface showing the instrument block diagram (top), embedded Oscilloscope panel (bottom) and the Oscilloscope settings panels (bottom right).

Quick start guide

This example shows how to use the Moku FIR Filter Builder to implement a matched filter before further measurement. In this case, the Moku FIR Filter Builder generates a matched filter to demodulate a noisy 5 ms duration chirp signal, which is applied to Input 1. The noisy signal has a 40 kHz bandwidth with a -40 dB Signal-to-Noise Ratio (SNR). The chirp signal is generated with $x[t] = \cos\left(\pi \frac{1}{0.005} t^2\right)$, therefore the matched filter is set to $h[t] = \cos\left(\pi \frac{1}{0.005} (1-t)^2\right)$

Matched filtering is a standard digital signal processing technique for detecting a signal of a known shape that is buried in noise. The custom impulse response design option in Moku FIR Filter Builder allows you to implement matched filters by uploading an impulse response that corresponds to a known reference signal. When the input signal contains that waveform, the matched filter produces a distinct correlation peak, enabling precise timing and reliable detection even in noisy environments.



Sample rate 305.2 kHz

Coefficients 1,527

Design domain

Domain Time domain

Impulse response

Shape Equation

Equation cos(pi*200*(1−t)^2)

Window

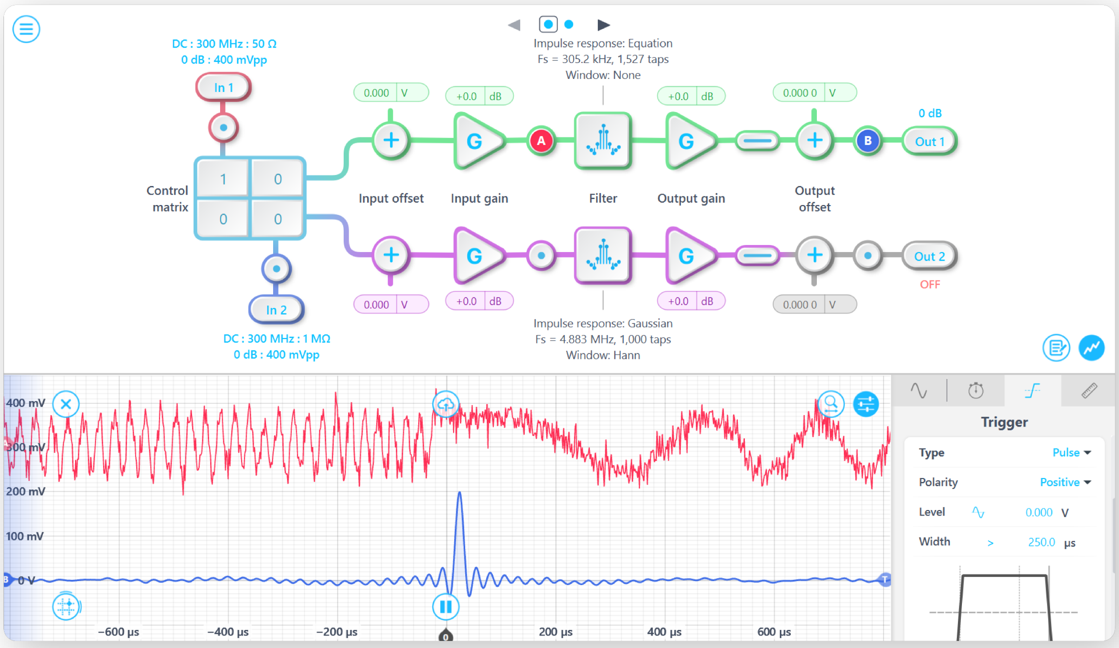
Function None

Figure 2. FIR Filter equation for the matched filter, using a time-reversed chirp signal.

- **Step 1:** Configure the analog front end settings for the signal inputs
  - Set In 1 to use 50 Ω input impedance, 300 MHz bandwidth, 0 dB attenuation, and DC coupling. This ensures the analog front end matches the source impedance and preserves both the AC signal and any DC bias present.
- **Step 2:** Configure the control matrix
  - Set the control matrix to [1, 0, 0, 0] to send the signal from Input 1 directly to the filter along the top signal path (in green). The lower filter path is not used, therefore the lower matrix coefficients are set to 0.
- **Step 3:** Configure the input and output offset and gain
  - There is no offset needed in this example, however in practical systems, it's common for signals to include small DC offsets. For instance, Input 1 may carry an error signal with a 10 mV baseline shift. If uncorrected, this offset propagates through the filter and into the output, biasing subsequent measurements or actuators. By setting an output offset, in this case, to -10 mV, you can re-center the signal around 0 V.
  - Both input and output gain can remain at 0 dB, since the signals are already within a good range. Gain adjustments are useful for optimizing measurement resolution, compensating

for low-level signals, or ensuring the output can properly drive subsequent stages, adjust this accordingly for your application.

- **Step 4:** Observe the signals in the Oscilloscope
  - Probe points can be used to observe and adjust the effects of the filter, while configuring the filter. Enable the probe points before the filter and at the filter output to check the filter behavior is as expected.
- **Step 5:** Configure the filter design and coefficients
  - Open the filter configuration panel and select to design in the time domain. Select the "Impulse / step response" filter view finder to check the matched filter looks correct.
  - Set the sampling rate first, as it affects the filter range. In this case, set it to 305.2 kHz, or at least twice the desired cutoff frequency to prevent aliasing and ensure the filter operates stably.
  - Select 1, 527 coefficients to match the filter length to the signal being demodulated.
  - Select "None" windowing. As the matched filter is designed to maximize the SNR by performing a correlation between the received signal and a known reference, we do not want to modify its shape in any way, including with windowing.
  - Select "Equation" as the filter shape and input the matched filter equation $\cos\left(\pi \cdot \frac{1}{0.005} \cdot (1-t)^2\right)$. Check the filter looks like what is shown in Figure 2.
- **Step 6:** Observe the correlation peak
  - In the Oscilloscope, the noisy input appears nearly indistinguishable from background. Figure 3 shows a sharp correlation peak (blue) at the filter output at the instant the chirp occurs.


DC : 300 MHz : 50 Ω
0 dB : 400 mVpp

Impulse response: Equation
Fs = 305.2 kHz, 1,527 taps
Window: None

Control matrix: 1 0 / 0 0

Input offset 0.000 V; Input gain +0.0 dB; Output gain +0.0 dB; Output offset 0.000 0 V; Out 1 0 dB

Impulse response: Gaussian
Fs = 4.883 MHz, 1,000 taps
Window: Hann

Out 2 OFF

DC : 300 MHz : 1 MΩ
0 dB : 400 mVpp

Trigger
Type: Pulse
Polarity: Positive
Level: 0.000 V
Width: > 250.0 µs

Figure 3. FIR Filter Builder configured for matched filter demodulation; the correlation peak (blue) as compared with the noisy chirp signal input (red).

Principles of operation

Moku FIR Filter Builder implements a finite impulse response (FIR) filter, where each output sample is calculated as a weighted sum of recent input samples. Since the output depends only on a finite number of past inputs, the impulse response always settles to zero in finite time. Unlike IIR filters, FIR filters are inherently stable and can have exactly linear phase, making them useful for applications requiring precise amplitude and phase control.

When designing a filter, you can work in either the frequency or time **design domains**, depending on your goals. To implement and optimize any filter, you must configure the **filter settings** correctly. This includes selecting an appropriate **design method** and **window function**, as well as balancing frequency resolution against sidelobe suppression. This section is an overview of how these components affect filter outputs and how to use them for your application. Read more about each setting and option in the linked sections.

Design domain

Regardless of the domain, the resulting filter operates by convolving the input signal with the designed impulse response on each sampling cycle. This process continuously produces a filtered output in real time, allowing you to refine both frequency selectivity and time-domain behavior through precise coefficient control.

Frequency domain is useful in shaping or constraining signal content within specific bandwidths, for example, isolating a passband or suppressing unwanted harmonics. Read more about frequency domain settings.

Time domain is preferred when needing to focus on how a filter modifies transient behavior, such as minimizing ringing or shaping pulse responses. Read more about time domain settings.

Filter settings

Configuring a filter to match the design requirements mandates that the filter sampling rate, coefficients, and windowing are all set up correctly for the application.

Sampling rate determines the frequency range over which the filter operates. It must be at least twice the highest input frequency to prevent aliasing. Selecting different sampling rates defines the filter's corner frequencies and operational limits, including resolution and processing bandwidth. Read more about sampling rate.

Number of filter coefficients (taps) determines filter sharpness and computational demand, more coefficients yield higher frequency selectivity and stopband attenuation, while fewer reduce latency and resource use. Symmetric coefficients ensure linear phase, preserving waveform integrity. Read more about filter coefficients and quantization.

Window function shapes the trade-off between transition sharpness and sidelobe suppression. Options such as Rectangular, Hann, Hamming, Blackman, Nuttall, Tukey, and Kaiser provide varying balances between frequency resolution and attenuation. Read more about window function options.

Design methods

Each of these methods can be implemented within Moku FIR Filter Builder through its design interface or by importing coefficients generated from external tools such as MATLAB or

SciPy. The choice of method depends on whether your priority is ease of implementation, computational efficiency, or precise frequency-domain control.

Window method is widely used for its simplicity and predictable results, well suited for applications that require smooth frequency roll-off without complex optimization. Select an ideal filter response, such as a sinc or rectangle function, then optionally apply a window function to control sidelobe levels and transition width. Appropriate window selection allows you to balance between frequency resolution and sidelobe suppression. See the different window function options and benefits.

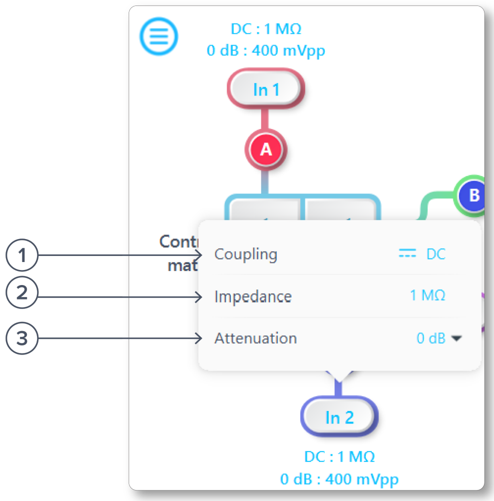
Frequency sampling method defines the filter by specifying desired amplitude and phase characteristics at discrete frequency points. This approach gives you precise control over the required frequency response, with selectable filter shape (lowpass, highpass, etc.) and corner frequencies. This is best for matching a target transfer function or compensating for known system characteristics, it is particularly effective when the filter needs to approximate arbitrary or measured frequency responses.

Custom and equation designs is best for specialized applications, you can define the impulse response directly or specify an analytical equation. This allows you to implement matched filters, equalizers, or other custom designs that reproduce a known waveform or system response. It provides the highest level of flexibility when standard filter shapes are insufficient.

Using the instrument

Signal inputs

The analog frontend settings for each input channel of the FIR Filter Builder can be individually configured. Click the In 1 icon to configure the input settings for the signal input.



Coupling DC
Impedance 1 MΩ
Attenuation 0 dB

Figure 4. Configuration of analog inputs on the FIR Filter Builder.

① Select between AC and DC input coupling.

② Select between 50 Ω and 1 MΩ input impedance (hardware dependent).

③ Select an input attenuation.

Control matrix

The control matrix combines, re-scales, and re-distributes the input signal to the two independent FIR Filter Builders. The output vector is the product of the control matrix multiplied by the input vector.

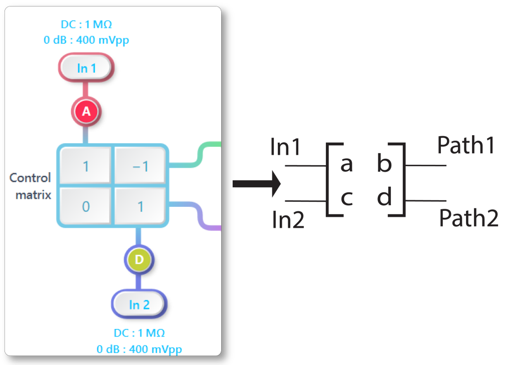


Figure 5. Control matrix in the block diagram and path schematic.

where $\mathrm{Path1} = a \times \mathrm{In1} + b \times \mathrm{In2}$ and $\mathrm{Path2} = c \times \mathrm{In1} + d \times \mathrm{In2}$.

The value of each element in the control matrix can be set between -20 to +20. The gain can be incremented by 0.1 when the absolute value is less than 10 and by 1 when the absolute value is between 10 and 20. Thus the matrix can be used to add or subtract two input signals to instead utilize a differential or common mode input for the FIR Filter Builder.

Filter design parameters

Moku FIR Filter Builder provides full control over filter parameters, allowing you to design with respect to the **magnitude/phase**, **impulse/step response**, and **group/phase delay** responses. For all filter response types, you will need to define the sampling rate, coefficients, and window type.

Table 1. Filter characteristic graphs and design domain options.

	Magnitude / phase		Impulse / step response		Group / phase delay	
	Plot 1	**Plot 2**	**Plot 1**	**Plot 2**	**Plot 1**	**Plot 2**
X-axis	Frequency		Time		Frequency	
Y-axis	Gain	Phase	Amplitude		Group / phase delay	

Sampling rate

Select different output sampling rates, based on the desired corner frequencies. The lower and upper bounds for each shape of pre-defined filters have different sampling rates, find the filter bounds for your device in its datasheet.

The sampling rate should be at least twice the frequency of the sampled signal to avoid aliasing.

Coefficients

The number of coefficients in a filter determines how closely the filter matches its design specifications. With more coefficients, the filter achieves sharper transition bands and stronger stopband attenuation. Filters with fewer coefficients are more efficient to implement, requiring less processing power and introducing less delay.

In practice, the best filter design uses the smallest number of taps that still meets the performance requirements of the application. This balances efficiency with accuracy, ensuring that the filter performs its intended role without consuming unnecessary resources.

Lower the number of coefficients in your filter for systems where real-time performance, low power consumption, or minimal latency is more critical than sharp frequency selectivity.

Increase the number of coefficients in your filter for applications that require precise control over frequency content. More coefficients can effectively isolate narrow frequency ranges or provide deep suppression of unwanted signals, at the cost of increased computation, memory usage, and processing delay.

To calculate the number of coefficients in your filter for a specific period, use Equation 1. It can be useful to make the filter symmetric by using an odd number of coefficients. Add one coefficient, or round up/down, to your filter to guarantee a linear phase response.

$$\text{Number of coefficients} = \text{Sampling rate} \cdot \text{Period} \tag{1}$$

For example, to generate a Gaussian filter over 25 ms at a sampling rate of 122.1 kHz (on Moku:Lab), you would calculate $n = 122.1 \times 10^3 \cdot 25 \times 10^{-3} = 3052.5$ taps. Therefore, you would use 3053 coefficients, rounded up.

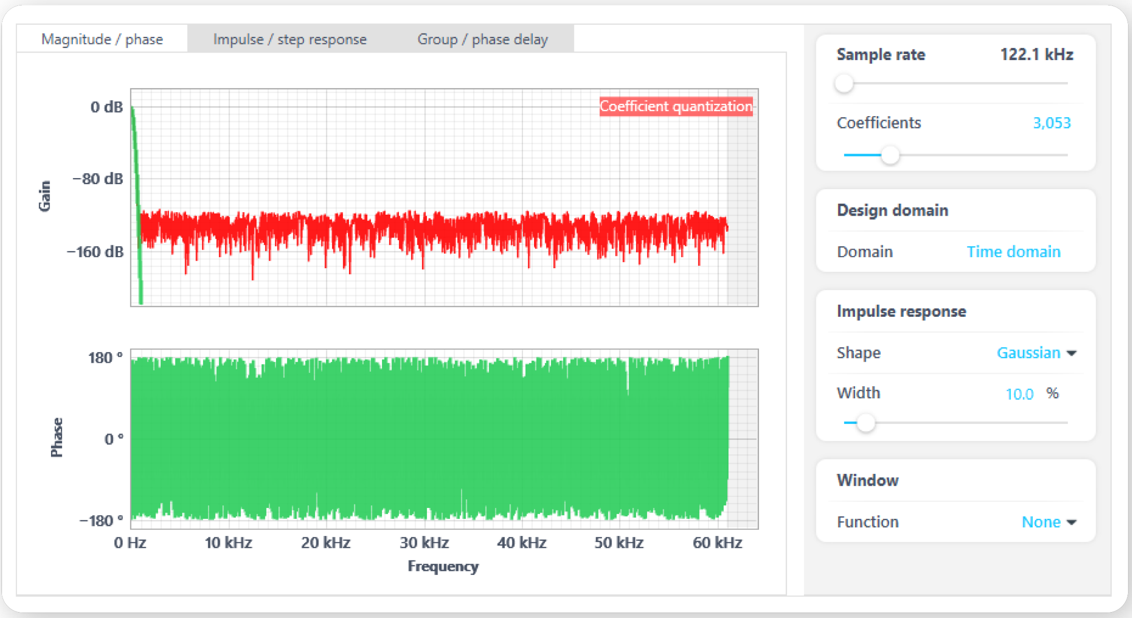
To achieve the same filter response on a different hardware platform, such as Moku:Pro, you could set the sampling rate to 305.2 kHz and the number of coefficients is therefore $305.2 \times 10^3 \cdot 25 \times 10^{-3} = 7630$. If the coefficients are not selected correctly for the sampling rate, the filter period will not match the desired temporal response, and the filter will not perform as designed.

Coefficient quantization

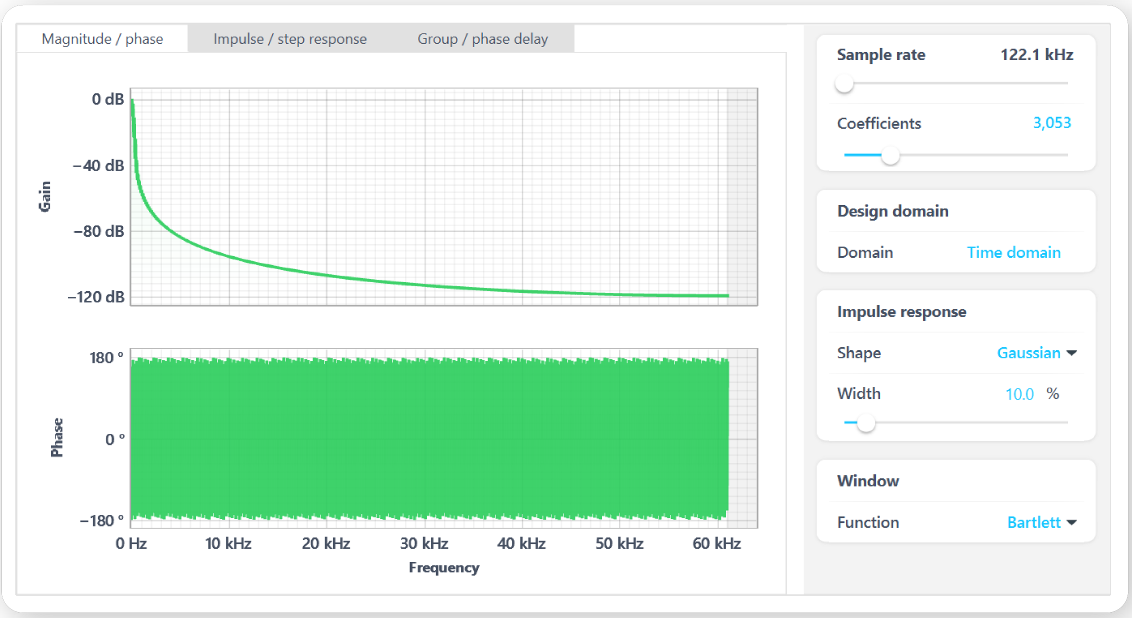
Due to the limited precision with which a coefficient can be digitally represented, quantization error is pronounced at certain filter settings. If this occurs, a coefficient quantization warning may appear on the response plot with a red trace in the transfer function. The red trace will show the closest achievable filter response to the ideal value, shown in the filter color.

This can be mitigated by applying windowing, which smooths the coefficient distribution, reduces dynamic range, and suppress truncation effects. This therefore preserves the intended filter response and minimizes quantization-induced distortion such that no coefficient falls below the quantizer’s LSB, removing coefficient quantization.

In the Gaussian filter example above, adding the Bartlett window changes the coefficient distribution so that no coefficient fell below the quantizer’s LSB, removing coefficient quantization.



Sample rate 122.1 kHz
Coefficients 3,053
Design domain
Domain Time domain
Impulse response
Shape Gaussian
Width 10.0 %
Window
Function None



Sample rate 122.1 kHz
Coefficients 3,053
Design domain
Domain Time domain
Impulse response
Shape Gaussian
Width 10.0 %
Window
Function Bartlett

Figure 6. A 25 ms Gaussian FIR filter designed with 10 % width, 3,053 coefficients, 122.1 kHz sample rate, and a Bartlett window (bottom) to reduce the dynamic range, eliminating quantization artifacts evident in the unwindowed design (top).

Selecting the right window

Window type	
None	No windowing gives the narrowest main lobe, which is the best resolution, but also has very high sidelobes, making it a good window type when the sharpest possible frequency resolution is needed and sidelobes aren't a concern.
Blackman	Provides excellent sidelobe suppression but has poorer resolution, with a wider main lobe, making it a good window type when very low sidelobes are critical, such as in audio or instrumentation filters where stopband leakage must be minimized.
Bartlett	This is a simple triangular window with reduced sidelobes compared to rectangular, but still has limited stopband attenuation, making it good for simple, low-cost implementations where modest sidelobe reduction is acceptable.
Hann	Good balance with moderate sidelobe suppression and main lobe width, this is commonly used as a general-purpose window, making it a good choice for spectral analysis, balancing resolution with reasonable sidelobe suppression.
Hamming	Similar to Hann but with slightly better sidelobe suppression at the expense of a slightly wider main lobe, making it a good window type when minimizing the first sidelobe is more important than overall sidelobe roll-off.
Nuttall	Very low sidelobes and smooth decay, but with a significantly wide main lobe and lower frequency resolution, making it an excellent window type precision spectral measurements where extremely low sidelobes are required.
Tukey	Adjustable tapering allows a trade off between rectangular (sharp) and Hann-like (smooth) characteristics, making it a good window type when you need adjustable trade-offs, e.g., in adaptive systems or exploratory analysis where you can tune between sharp resolution and sidelobe control.
Kaiser	Flexible window with a tunable parameter (order) that lets you trade off between main lobe width and sidelobe attenuation, making it a good window type when you want explicit control over the balance between transition bandwidth and stopband attenuation.

Impulse response

Frequency domain

Select the filter shape and the frequency corners (high and, or low) to design your FIR filter in the frequency domain. The frequency corners are where the system attenuates the signal to the half-power point, that is -3 dB of the nominal passband value.

Filter shape	Icon	Description
Lowpass		Passes signals above a cutoff frequency and attenuates higher frequencies. Used to remove high frequency noise and smoothing digital sensor data.
Highpass		Passes signals above a cutoff frequency and attenuates lower frequencies. Used to remove low frequency drift and noise to enhance accuracy of signal processing.
Bandpass		Passes signals within a specified frequency range and attenuates frequencies outside that range. Used for isolating a specific frequency band in radio channels, feature extraction or to tune instrumentation in a specific range.
Bandstop		Attenuates signals within a specific frequency range and passes those outside it. Used to remove specific interference and frequency components such as power lines or radio bands.

Time domain

Filter shape	Description
Rectangular	Produces an impulse response with constant amplitude across all taps within the filter length. The filter acts as a simple moving average, resulting in a lowpass characteristic that heavily smooths signals but provides poor frequency selectivity.
Sinc	The impulse response follows the mathematical sinc function (Sin(X)/X). In the time domain this represents the ideal response of a perfect lowpass filter. When truncated to finite length, it approximates an ideal lowpass behavior, and can be shifted or modulated for bandpass or highpass designs.
Triangular	The impulse response rises linearly to a midpoint and then symmetrically decreases, resembling a convolution of two rectangular pulses. It produces a smoother frequency response than a rectangular shape, reducing ringing while maintaining general lowpass characteristics.
Gaussian	The impulse response follows a Gaussian curve, symmetric and smooth, with no sharp discontinuities. Yields a frequency response that is also Gaussian, minimizing overshoot and ringing. Used in smoothing, pulse shaping, or feature extraction.
Equation	The impulse response is defined analytically using a mathematical formula, allowing for precise control of filter behavior, enabling custom designs like matched filters or equalizers. Read more about the equation editor.

Filter shape	Description
Custom	The impulse response is explicitly specified or imported by the user. Used when the desired time-domain behavior does not correspond to a standard analytical form. Read more about implimenting a custom filter.

Width controls how spread out the filter's impulse response is. Smaller values create narrower, stronger smoothing, while larger values produce wider responses that pass more frequencies.

Custom

You can define a custom filter by uploading filter coefficients via a text file or clipboard. The output of the FIR filter is a weighted sum of the most recent input values:

$$y[n] = \sum_{i=0}^{N-1} c_i x[n-1] \tag{2}$$

The maximum number of coefficients (or "taps") depends in the chosen sampling rate. Read about the sampling rate and coefficients that will best suit your application.

To specify a custom filter, you must supply a text file containing the filter coefficients. The file can contain up to 14,819 coefficients separated by commas or new lines. Each coefficient must be in the range of [-1.0, +1.0]. Internally, these are represented as signed 25-bit fixed-point numbers, with 24 fractional bits.

Some coefficients may result in overflow or underflow, which degrade filter performance. Filter responses should be checked prior to use.

Filter coefficients can be computed using signal processing toolboxes such as MATLAB and SciPy.

Equation

The Equation filter type enables you to design arbitrary filters using multiple piecewise mathematical functions. The filter can be configured in one or multiple segments, each with their own function and definable fractional time period.

Equation editor

The Equation editor allows you to define arbitrary mathematical functions for each segment in the filter. A large range of common mathematical expressions including trigonometric, quadratic, exponential, and logarithmic functions are available to generate a filter.

The variable 't' represents time in the range from 0 to 1 periods of the total filter, meaning that the input equation needs to be within the horizontal bounds of [0, 1] to remain undistorted. The filter equation should also be scaled to be within the vertical bounds of [-1, +1] to avoid distortion and for accurate amplitude scaling.

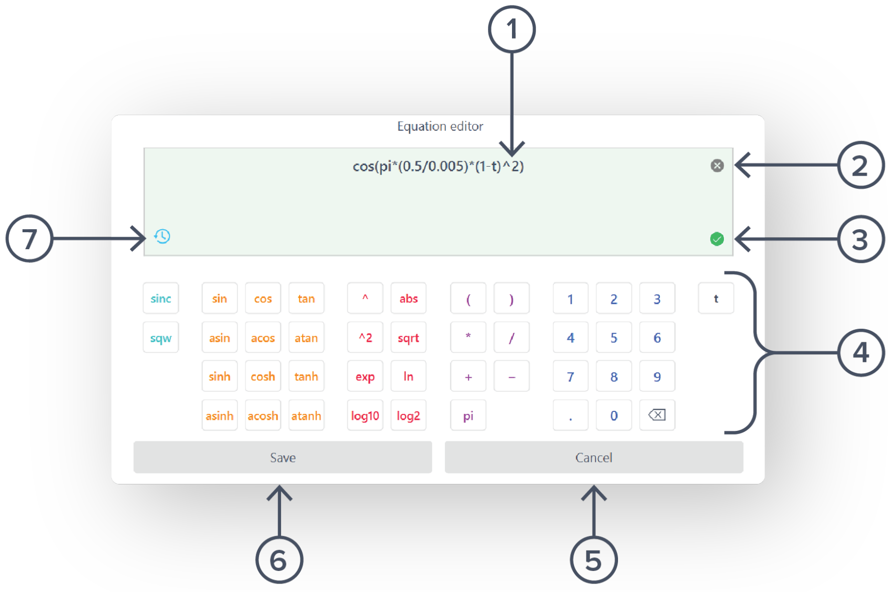

1
2
3
4
5
6
7
Equation editor
cos(pi*(0.5/0.005)*(1-t)^2)
sinc sin cos tan ^ abs ( ) 1 2 3 t
sqw asin acos atan ^2 sqrt * / 4 5 6
sinh cosh tanh exp ln + - 7 8 9
asinh acosh atanh log10 log2 pi . 0
Save
Cancel

Figure 7. Equation editor pop-up in the FIR Filter Builder.

① Current arbitrary mathematical equation entered
② Delete current equation
③ Equation validation success
④ Interactive keyboard of variables, operands, functions, parameters, and backspace
⑤ Cancel equation editing, exit out of the pop-up without applying any changes
⑥ Save the current equation, apply the filter, and exit the pop up
⑦ Open a history of recently used equations

Filter path settings

Other block diagram elements in the FIR Filter Builder include switches to enable/disable the signal in the processing output, offsets and gain that can be optionally applied to the input signal or the output path. These features contribute to the signal flow from input to the FIR Filter Builder and to the output.

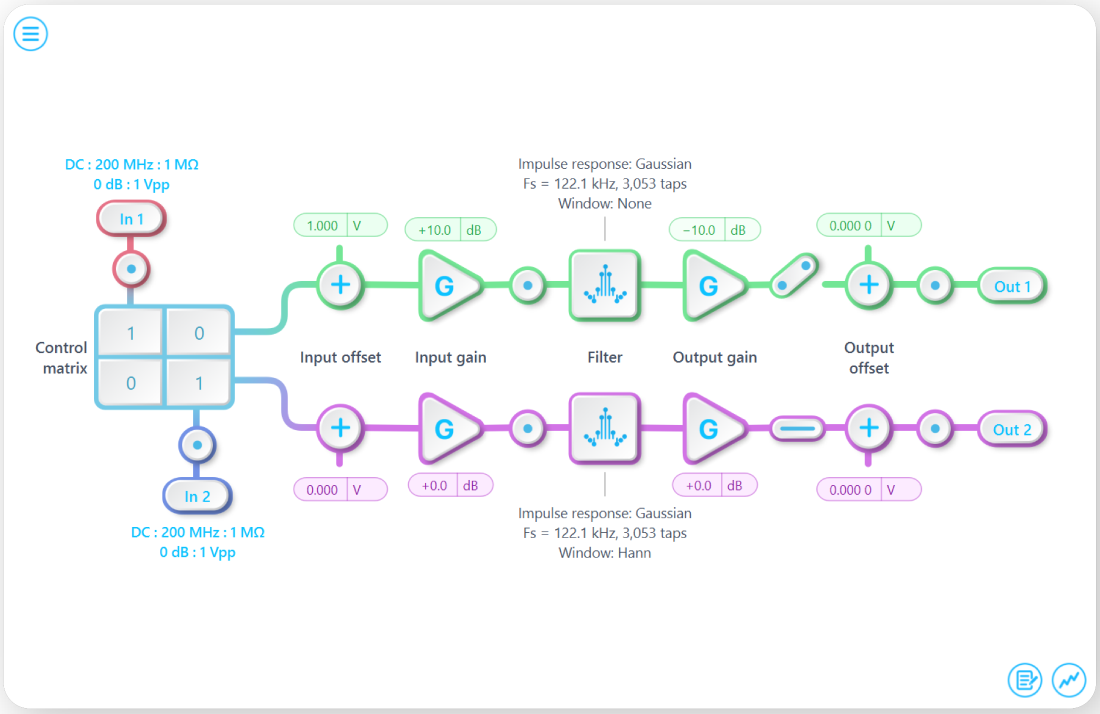


Figure 8. FIR Filter Builder signal paths shown in green and purple. Both filter paths feature input and output offsets, input and output gains and an output switch.

Offsets

A DC offset can be applied to both the signal before and after the filter. Input offsets can be added or subtracted from the measured process variable before it is fed to the filter.

Switches

The switches can be opened or closed to engage or disengage filter output. When the output switches are open, it sends zeros to the output. When the output switch is closed, the filtered signal and any offset is given to the output signal path.

Gain

The FIR Filter Builder processes signals internally with much higher precision than its physical outputs (for example, 32-bit internal processing vs. 16-bit Digital-to-Analog converter (DAC) output). Applying internal digital gain before the output stage increases the portion of the DAC's range used by the signal, improving effective resolution without distortion.

In Figure 9, the gray trace uses nearly the full 16-bit DAC range, preserving fine waveform detail. The orange trace represents the signal if an analog or post-DAC amplification, which only scales the coarse quantized steps upward without adding detail. To achieve the highest output fidelity, set the gain as high as possible without causing saturation or clipping, ensuring the output uses the full available bit range.

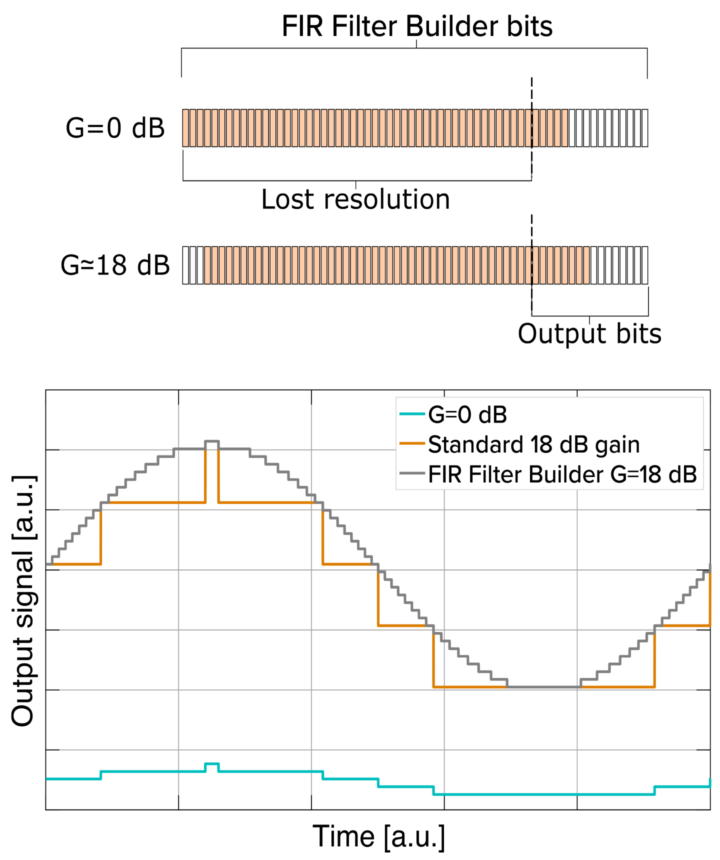


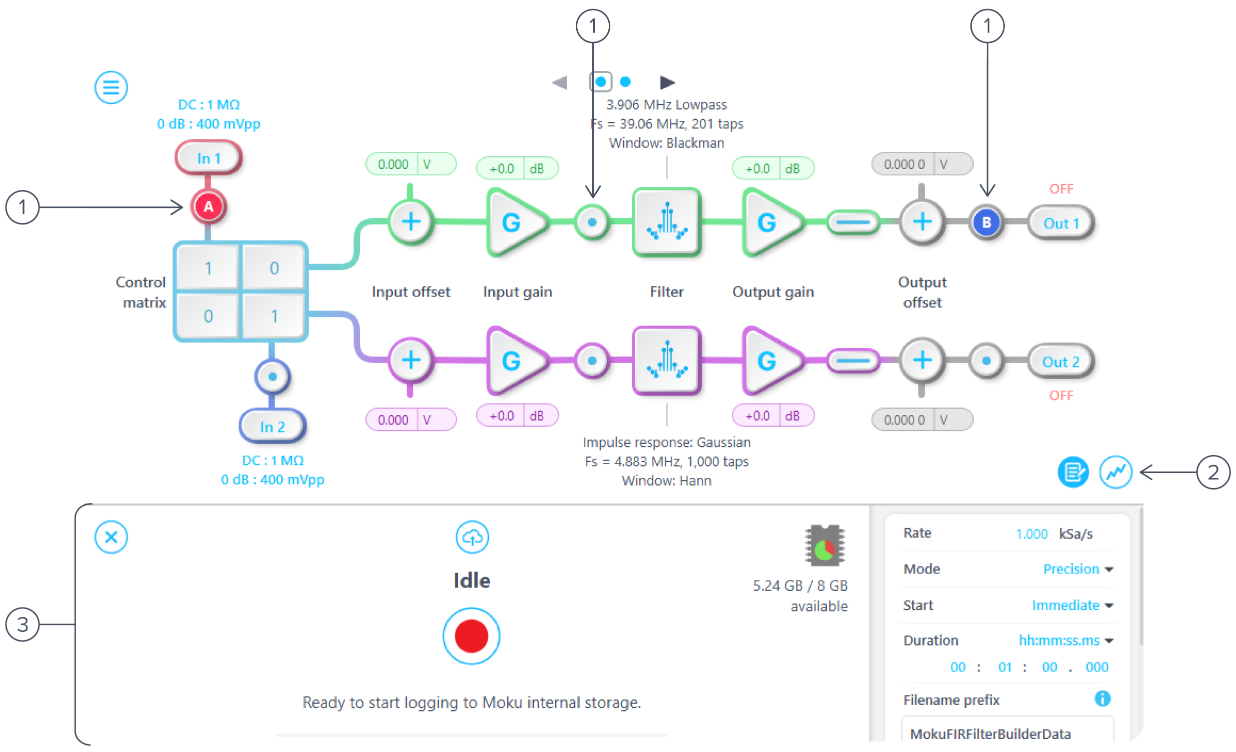
Figure 9. Flat 18 dB gain (orange) versus FIR Filter Builder with +18 dB gain (gray) block on the output, preserving finer waveform detail.

Observing the data

Embedded Oscilloscope

ID	Parameter	Description
1	Probe points	Click to place the probe point, the number available is device dependent.
2	Open embedded Oscilloscope and Data Logger	Open and close the embedded Oscilloscope and Data Logger.
3	Oscilloscope	Refer to the Oscilloscope user manual for the details.

Embedded Data Logger



Ready to start logging to Moku internal storage.

Rate 1.000 kSa/s
Mode Precision
Start Immediate
Duration hh:mm:ss.ms 00 : 01 : 00 . 000
Filename prefix MokuFIRFilterBuilderData

Figure 10. Embedded Data Logger in the FIR Filter Builder.

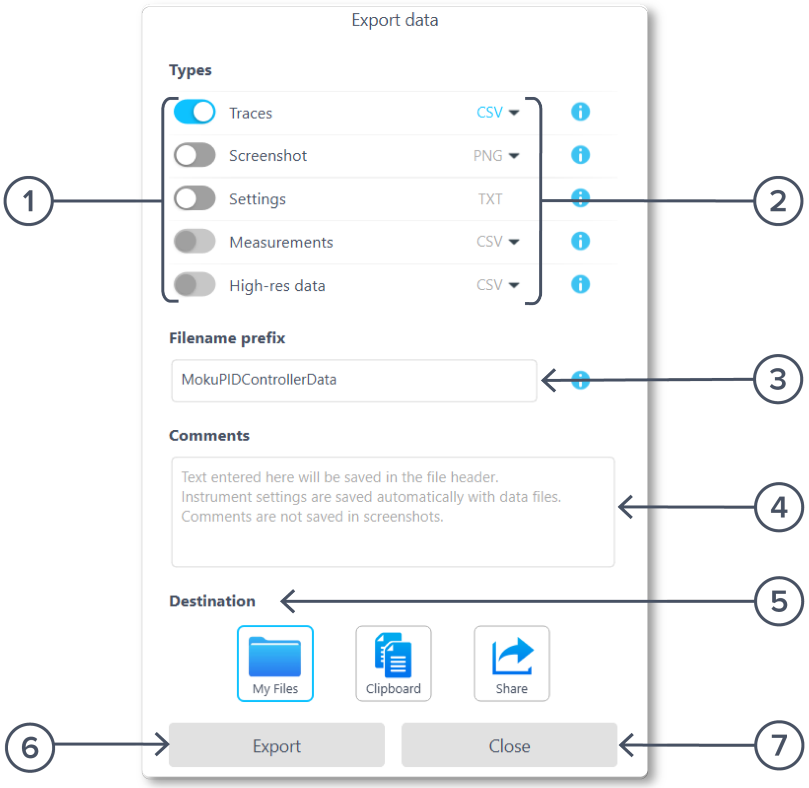
ID	Parameter	Description
1	Probe points	Click to place the probe point, the number available is device dependent.
2	Open the embedded Oscilloscope or Data Logger	Open and close the embedded Oscilloscope and Data Logger.
3	Data Logger	Refer to the Data Logger user manual for the details.

The embedded Data Logger can stream over a network or save data to the onboard storage of your Moku. For details, refer to the Data Logger user manual. More streaming information is in our API Reference.

Share data

Export data by clicking the share icon . Any active probe points will be captured in the live data export or logging. Open the embedded Oscilloscope or Data Logger to export live and logged data, respectively.

Live data



Export data

Types

Traces (on)	CSV	
Screenshot	PNG	
Settings	TXT	
Measurements	CSV	
High-res data	CSV	

Filename prefix

MokuPIDControllerData

Comments

Text entered here will be saved in the file header. Instrument settings are saved automatically with data files. Comments are not saved in screenshots.

Destination

Figure 11. Data exporting user interface and settings.

To save live data:

① Select the type of data to export:

- **Traces** Saves the trace data for all visible signal traces, in either a CSV or MATLAB format.
- **Screenshots** Save the app window as an image, in either a PNG or JPG format.
- **Settings** Saves the current instrument settings to a TXT file.
- **Measurements** Saves the active measurement values, in either a CSV or MATLAB format.
- **High-res data** Saves the full memory depth of statistic values for all visible channels, in LI, CSV, HDF5, MAT or NPY format.

② Select the **export format**.

③ Select the **Filename prefix** for your export. This is defaulted to "MokuFIRFilterBuilderData" and can be changed to any filename of alphanumeric characters and underscores. A timestamp and the data format will be appended to the prefix to ensure the filename is unique.

For example: "MokuFIRFilterBuilderData_YYYYMMDD_HHMMSS_Traces.csv"

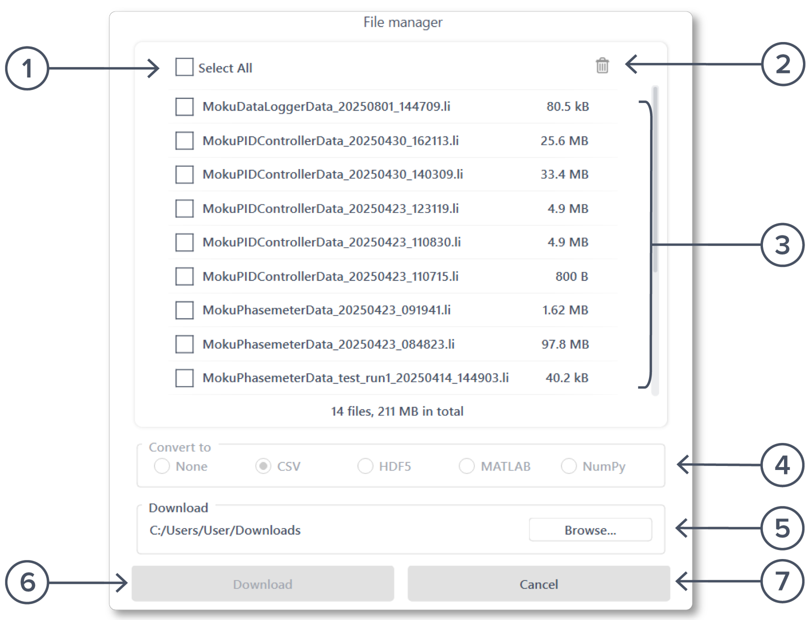
④ Enter additional **comments** to be saved in any text-based file header.

⑤ Select the export **destination** on your local computer. If "My Files" or "Share" is selected, the exact location is selected when the "Export" button is clicked. Multiple export types can be exported simultaneously using "My Files" and "Share", but only one export type can be exported to the clipboard at a time.

⑥ **Export** the data, or

⑦ **Close** the export data window, without exporting.

Logged data



File manager

Select All

MokuDataLoggerData_20250801_144709.li	80.5 kB
MokuPIDControllerData_20250430_162113.li	25.6 MB
MokuPIDControllerData_20250430_140309.li	33.4 MB
MokuPIDControllerData_20250423_123119.li	4.9 MB
MokuPIDControllerData_20250423_110830.li	4.9 MB
MokuPIDControllerData_20250423_110715.li	800 B
MokuPhasemeterData_20250423_091941.li	1.62 MB
MokuPhasemeterData_20250423_084823.li	97.8 MB
MokuPhasemeterData_test_run1_20250414_144903.li	40.2 kB

14 files, 211 MB in total

Convert to: None [✓ CSV] HDF5 MATLAB NumPy

Download: C:/Users/User/Downloads Browse...

Download Cancel

Figure 12. File exporting user interface and settings.

To save logged data:

① **Select all** files logged to the device's memory, to download or convert.

② **Delete** the selected file/s.

③ Browse and **select file/s** to download or convert.

④ Select an optional **file conversion format**.

⑤ Select a **location** to export your selected files to.

⑥ **Export** the data.

⑦ **Close** the export data window, without exporting.

Examples

Delay generator

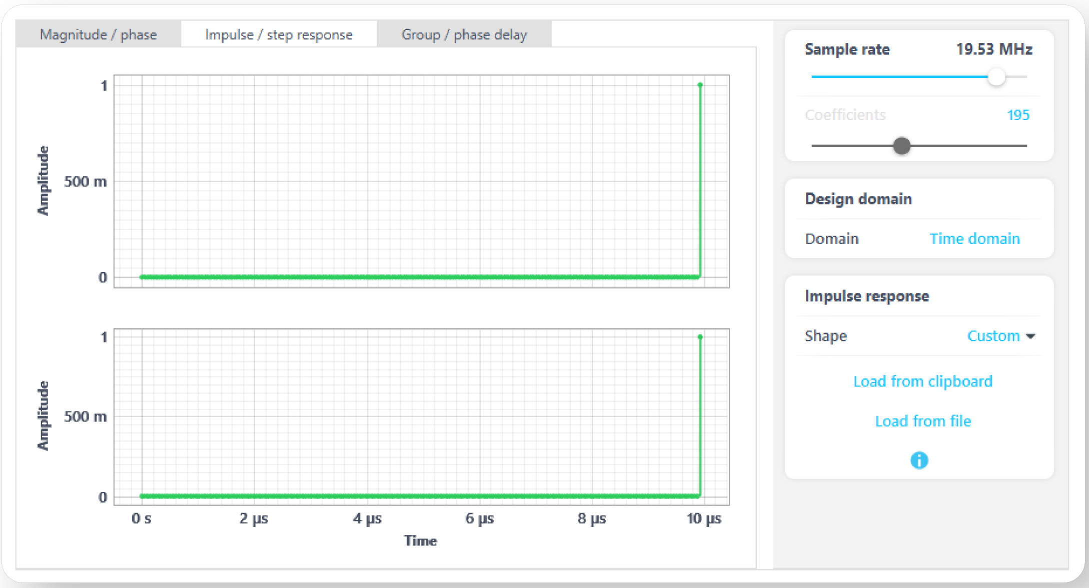
Precisely adjusting the delay of a signal can be important in many applications. For example, in high-speed signal transmission it is important to balance and compensate for the delays between different signal paths induced by cable length mismatches, differences in signal processing time, and many other factors. In this example we will use a custom filter to generate integer delays by adding zeros before the impulse response. This creates an allpass filter that shifts the signal in time, relative to the system clock, while preserving its waveform.

In this example, a 10 kHz sine wave is applied to Input 1. This input is routed to filter 1 (green), through the control matrix, to generate a 10 µs delay in the signal.

To generate a specific delay we need to know the sampling rate and the period of the filter, which in this case is 10 µs, to calculate the number of coefficients. We can select any sampling rate greater than 2× the frequency of the 10 kHz signal. We will select 19.53 MHz as the sampling rate in this example, as oversampling well above the Nyquist limit improves phase accuracy. Therefore,

$$n_{coeff} = 19.53 \times 10^{6} \cdot 10 \times 10^{-6} = 195.3 \tag{3}$$

- **Step 1:** Configure the analog front end settings for the signal inputs
  - Deploy the FIR Filter Builder on your Moku, connecting the input signal to Input 1 of the device.
  - Set In 1 to use 50 Ω input impedance, 0 dB attenuation, 300 MHz bandwidth, and DC coupling. This ensures the analog front end matches the source impedance and preserves both the AC signal and any DC bias present.
- **Step 2:** Configure the control matrix
  - Set the control matrix to [1, 0, 0, 0] to send the signal from input 1 to filter path 1 only, along the top, green signal paths.
- **Step 3:** Define impulse responses in the filter
  - Set the sampling rate to 19.53 MHz. This improves phase accuracy, prevents aliasing, and ensures the filter operates stably.
  - Select the 'Time domain' as the design domain and the 'Custom' filter shape.
  - Define the impulse response for the top (green) filter to have an impulse response with leading zeros, to delay the signal in time only. As calculated earlier (Equation 3), we will use 195 coefficients, rounded to the nearest odd number of coefficients, totaling 194 zeroes followed by a single 1. Load these values into the filter from clipboard or file, as shown in Figure 13.
  - 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1



Sample rate 19.53 MHz

Coefficients 195

Design domain

Domain: Time domain

Impulse response

Shape: Custom

Figure 13. FIR Filter with coefficients the delay filter loaded, showing a 10 µs delay.

- **Step 4:** Observe the delay in the oscilloscope
  - After the filter is set, probe points can be used to observe the signals. Enable the probe points before and after the filter and observe the 10 µs delay between the two signals.
  - Drag out cursors horizontally from the cursor menu to measure the delay between the zero-crossing points of Probe A and Probe B. Figure 14 shows the 10.82 µs delay in the signal.
- **Step 5:** Enable the outputs
  - Once the Oscilloscope is setup to observe the signals, the output can be enabled to send the signal to another instrument, in Multi-Instrument Mode, or output from your Moku. Click on the Output icon to select between Off and the available gain options. For this example, 0 dB is selected as the smallest range.

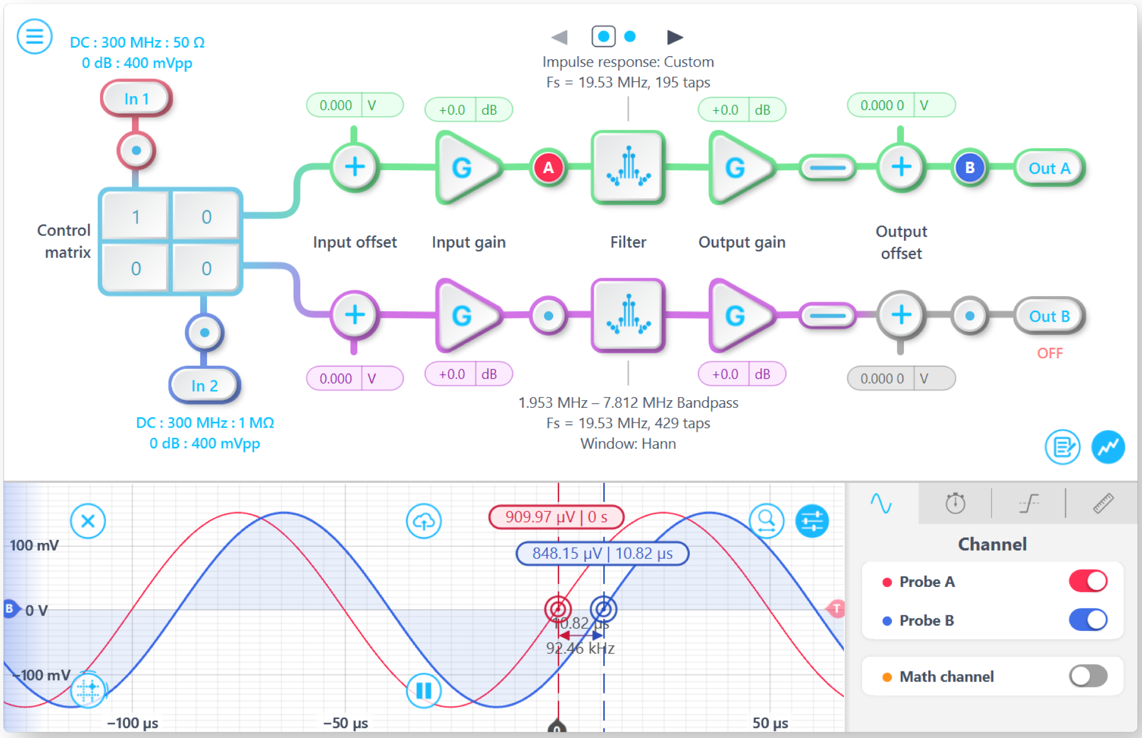


Figure 14. Moku embedded Oscilloscope showing the 10.82 µs delay generated in the signal.

Additional tools

Main menu

The main menu can be accessed by clicking the icon on the top-left corner.

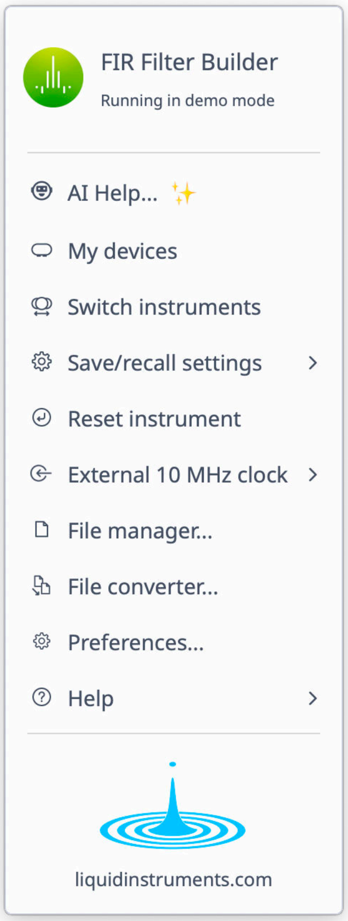


Figure 15. Main menu options for the FIR Filter Builder.

AI Help... Opens a window to chat to an AI trained to provide Moku-specific help (Ctrl/Cmd+F1)
My Devices returns to device selection screen
Switch instrument to another instrument
Save/recall settings

- Save current instrument state (Ctrl/Cmd+S)
- Load last saved instrument state (Ctrl/Cmd+O)
- Show the current instrument settings, with the option to export the settings

Reset instrument to its default state (Ctrl/Cmd+R)
Sync Instrument slots in Multi-Instrument Mode*
External 10 MHz clock selection determines whether the internal 10 MHz clock is used.
Clock blending configuration opens the clock blending configuration pop-up *
Power Supply access panel*
File Manager access tool
File Converter access tool
Preferences access tool
* If available using the current settings or device.

Help

- **Liquid Instruments website** opens in default browser
- **Shortcuts list** (Crtl/Cmd+H)
- **Manual** Open the user manual in your default browser (F1)
- **Report an issue** to the Liquid Instruments team
- **Privacy Policy** opens in default browser
- **Export diagnostics** exports a diagnostics file you can send to the Liquid Instruments team for support
- **About** Show app version, check for updates or licence information

File converter

The File converter can be accessed from the main menu .

The File converter converts a Moku binary (.li) format on the local computer to either .csv, .mat, .hdf5 or .npy format. The converted file is saved in the same folder as the original file.

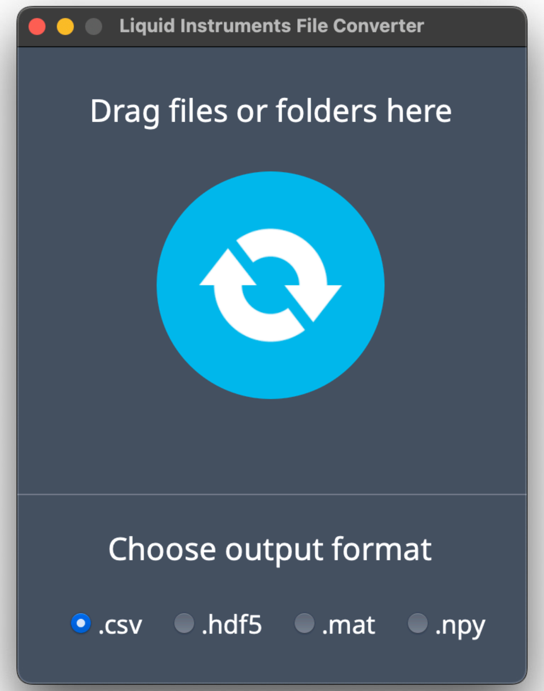


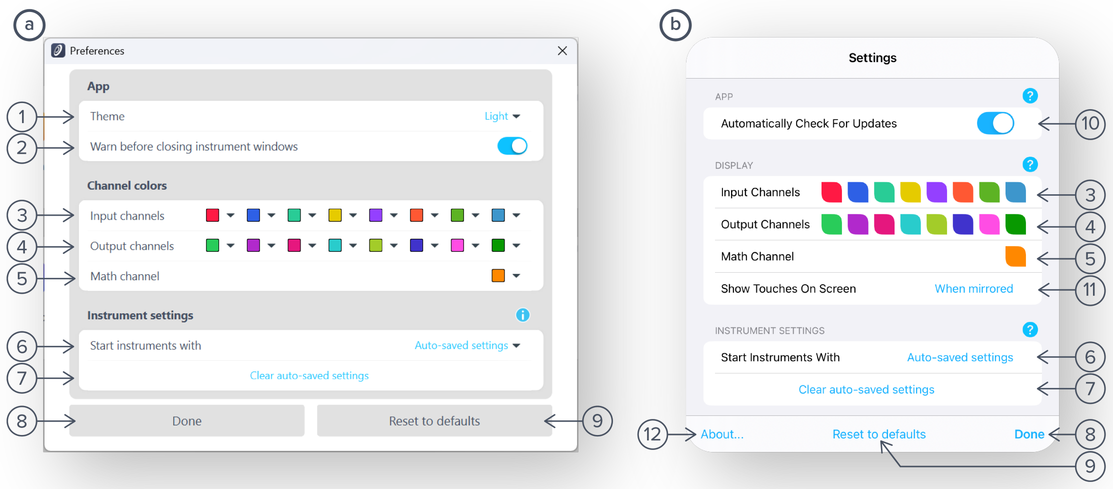
Figure 16. File Converter user interface.

To convert a file:

1. Select a file type.
2. Open a file (Ctrl/Cmd+O) or folder (Ctrl/Cmd+Shift+O) or drag and drop into the File converter to convert the file.

Preferences and settings

The preferences panel can be accessed via the Main Menu ☰. In here, you can reassign the color representations for each channel, switch between light and dark mode, etc. Throughout the manual, the default colors are used to present instrument features.



Preferences

App

Theme — Light

Warn before closing instrument windows

Channel colors

Input channels 🟥 🟦 🟩 🟨 🟪 🟧 🟩 🟦

Output channels 🟩 🟪 🟥 🟦 🟩 🟦 🟪 🟩

Math channel 🟧

Instrument settings

Start instruments with — Auto-saved settings

Clear auto-saved settings

Done — Reset to defaults

Settings

APP

Automatically Check For Updates

DISPLAY

Input Channels 🟥 🟦 🟩 🟨 🟪 🟧 🟩 🟦

Output Channels 🟩 🟪 🟥 🟦 🟩 🟦 🟪 🟩

Math Channel 🟧

Show Touches On Screen — When mirrored

INSTRUMENT SETTINGS

Start Instruments With — Auto-saved settings

Clear auto-saved settings

About... — Reset to defaults — Done

Figure 17. Preferences and settings for the Desktop (a) and for the iPad (b) App.

① Change the App theme, between dark and light mode.
② Choose if a warning opens before closing any instrument windows.
③ Tap to change the color associated with the input channels.
④ Tap to change the color associated with the output channels.
⑤ Tap to change the color associated with the math channel.
⑥ Select if instruments open with the last used settings, or default values each time.
⑦ Clear all auto-saved settings and reset them to their defaults.
⑧ Save and apply settings.
⑨ Reset all application preferences to their default state.
⑩ Notify when a new version of the app is available. Your device must be connected to the internet to check for updates.
⑪ Indicate touch points on the screen with circles. This can be useful for demonstrations.
⑫ Open information about the installed Moku application and license.

Shortcuts

Shortcuts are integrated to speed up your workflow on Moku. This includes graph zoom and scrolling, autoscaling, cursor, measurement, instrument, and general shortcuts. They are available from the main menu, as shown in Figure 18, or use the shortcut `Ctrl + H` to open the shortcuts dialog.

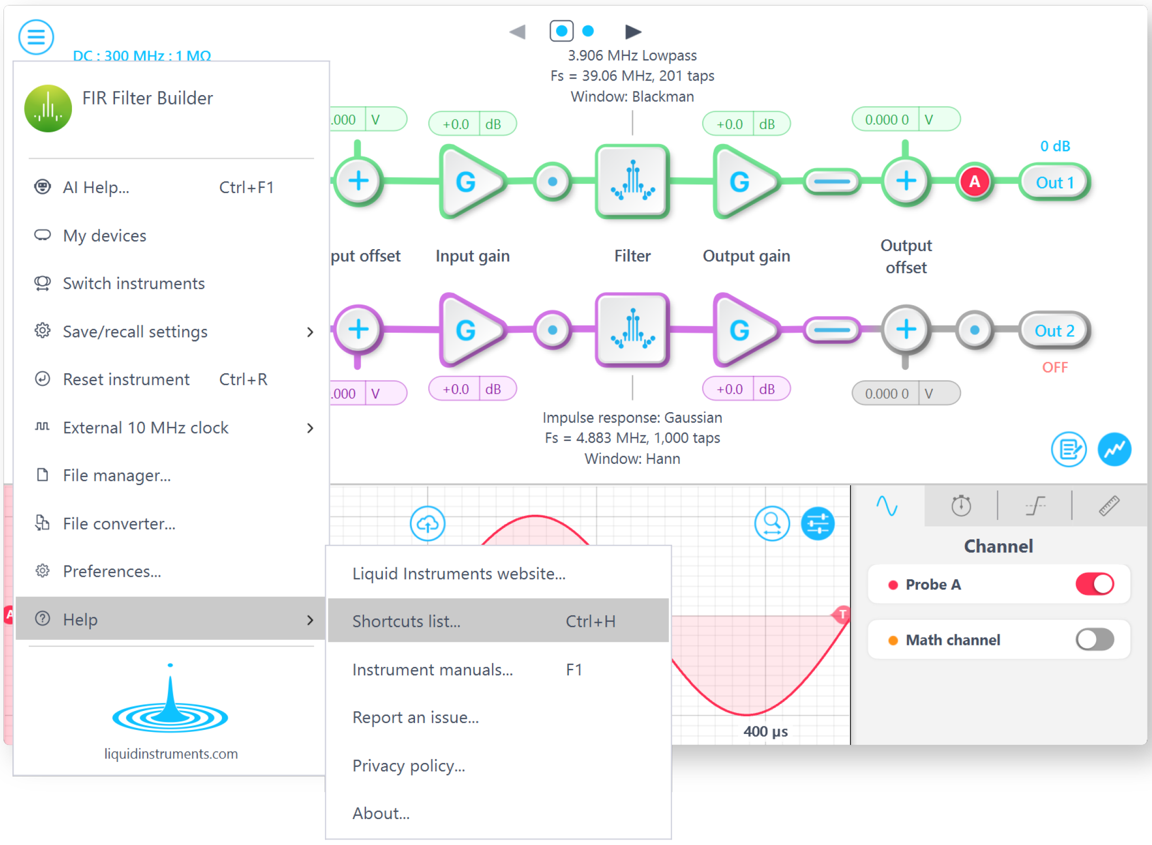


Figure 18. Main menu to access the help > shortcuts list.

External reference clock

Your Moku may support the use of an external reference clock, which allows Moku to synchronize with multiple Moku devices, other lab equipment, lock to a more stable timing reference, or integrate with laboratory standards. The reference clock input and output are on the rear panel of the device. Each external reference option is hardware dependent, review the available external reference options for your Moku.

Reference Input: Accepts a clock signal from an external source, such as another Moku, a laboratory frequency standard, or an atomic reference (for example, a rubidium clock or a GPS-disciplined oscillator).

Reference Output: Supplies the Moku internal reference clock to other equipment that require synchronization.

If your signal is lost, or is out of frequency, your Moku will revert to using its own internal clock until the reference signal returns. If this occurs, check the source is enabled, and that the correct impedance, amplitude, tolerance, frequency, and modulation are attached to the reference. Check the required specifications in the device specsheets.

When the reference returns within range, status changes to "validating" and then "valid" once lock is re-established.

10 MHz external reference

To use the 10 MHz external reference function, ensure "always use internal" is disabled in the Moku application, found in the main menu under "External 10 MHz clock". Then, when an external signal is applied to your Moku reference input and your Moku has locked to it, a pop up will show in the app. On some devices, the external reference information will be shown in the LED status as well, more information can be found in your Moku Quick Start Guide.

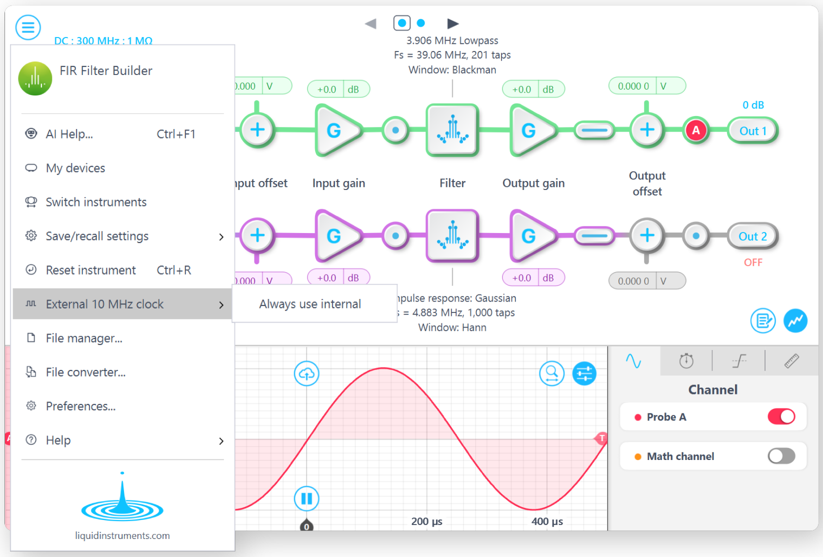


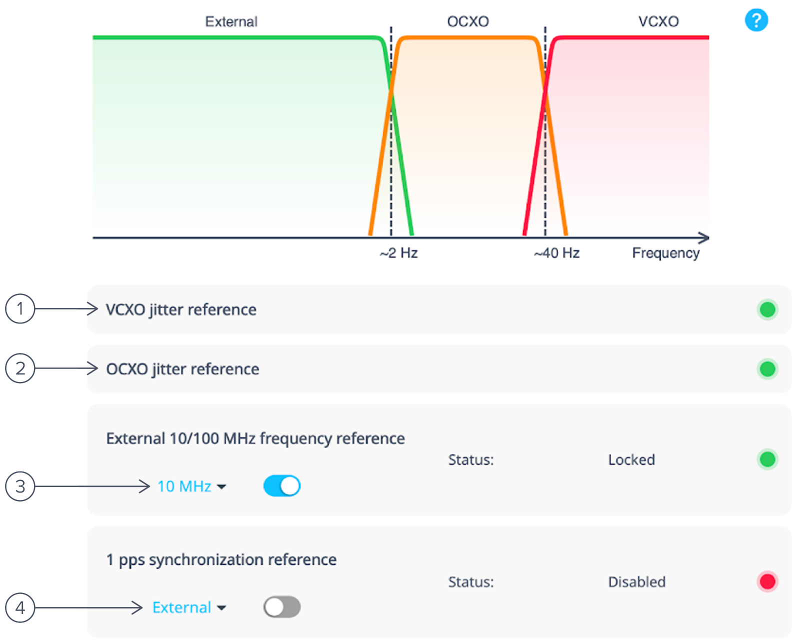
Figure 19. Moku main menu with "Always use internal" reference disabled and using an external reference.

Clock blending configuration

If available, Moku blends up to four clock sources simultaneously for more accurate phase, frequency, and interval measurements across all time scales. A low phase-noise Voltage-Controlled Crystal Oscillator (VCXO) is blended with a 1 ppb Oven-Controlled Crystal Oscillator (OCXO) for optimal wide-band phase noise and stability, which can be blended further with an external frequency reference and GPS disciplining to synchronize Moku with your lab and UTC.

The VCXO and OCXO will always be used for the clock generation signal. The external and 1 pps references are optional and can be enabled or disabled in the "Clock blending configuration..." settings from the main menu ☰. The loop bands are adjusted based on the different possible clock source configurations, shown in Figure 20, where the frequencies of the bands represent where each oscillator's phase noise dominates.

Read how the clock blending works on Moku:Delta for more details.



VCXO jitter reference ●

OCXO jitter reference ●

External 10/100 MHz frequency reference — 10 MHz (on) — Status: Locked ●

1 pps synchronization reference — External (off) — Status: Disabled ●

Figure 20. Moku clock blending configuration dialog with an external 10 MHz frequency reference enabled.

① **VCXO jitter reference** is always used for clock generation, handling high frequency jitter with the lowest noise.

② **OCXO jitter reference** is always used for clock generation, ensuring moderate term stability.

③ **External 10/100 MHz frequency reference** uses a "10 MHz" or "100 MHz" external reference to correct drift in the local oscillator, noting your Moku will have to be restarted after each change between a 10 MHz and 100 MHz source.

④ **1 pps synchronization reference** uses an "External" or "GNSS" reference to sync with UTC and correct drift in the local oscillator. The estimated clock stability is a measure of how much the reference performance deviates relative to the local OCXO/VCXO timebase (as currently blended and, if enabled, steered by the external 10 / 100 MHz External reference).