Moku Logic Analyzer

User Manual

LIQUID INSTRUMENTS

Table of Contents

Introduction

The Moku Logic Analyzer is a versatile instrument used for stimulating, monitoring, capturing, and debugging digital systems. It is especially beneficial for digital circuit design and analysis. The Logic Analyzer allows you to observe digital signals in the time domain. It features a multi-channel Pattern Generator for stimulating digital systems, and supports two independent protocol decoder channels.

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 Logic Analyzer datasheets.

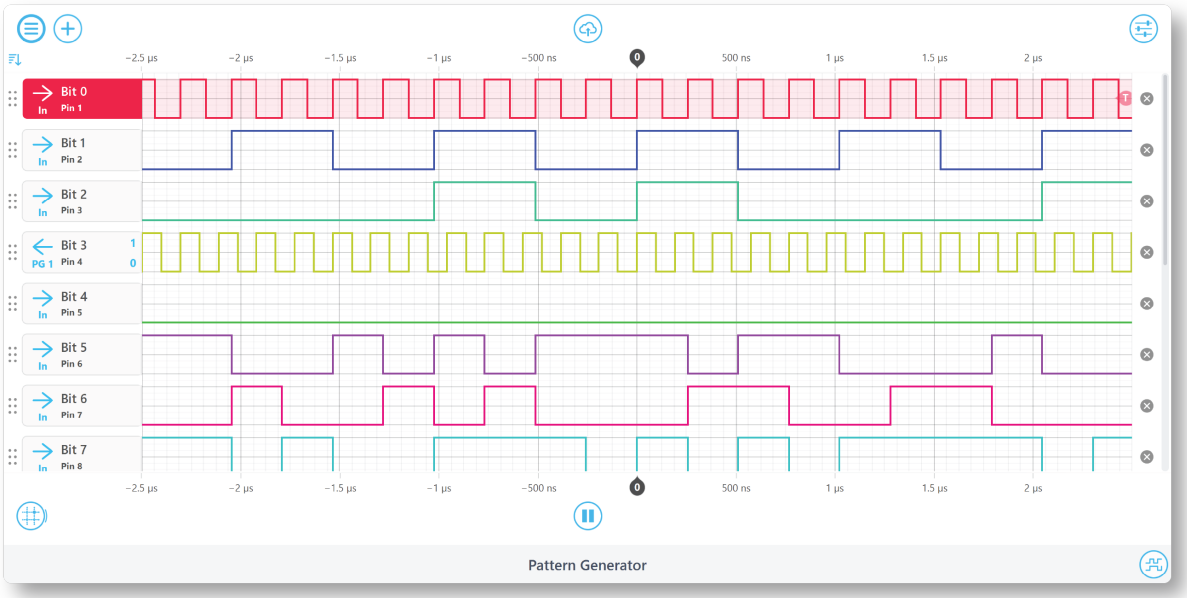


Figure 1. Moku Logic Analyzer

Quick start guide

Here we outline how to use the Moku Logic Analyzer to analyze digital signals, output signals from the Pattern Generator, and configure a protocol decoder.

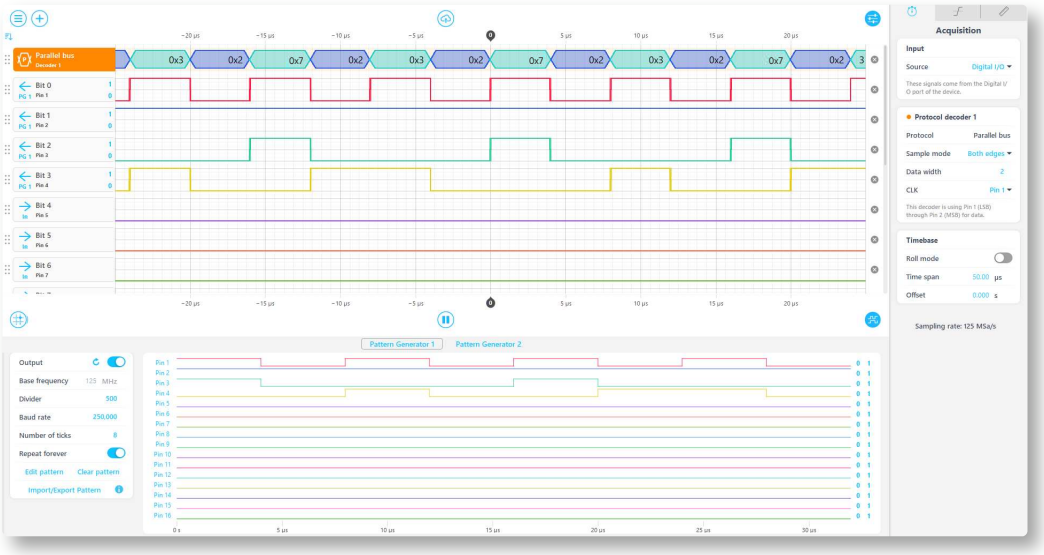


Figure 2. Quick start guide set-up of the Logic Analyzer

- **Step 1: Set up the acquisition**
  - In the acquisition panel, ensure the source is set to “Digital I/O” this will use the 16 bit digital I/O (DIO) if available on the hardware. The “Analog Inputs” option is available on all hardware platforms and uses each analog input as a pin or bit stream.
  - See input source selection, thresholding, and acquisition for more information.
- **Step 2: Set the time span**
  - Under the timebase sub-panel, set the time span to 50 µs and the offset to 0 s.
- **Step 3: Set the direction of the DIO pins**
  - Click the blue arrow labelled “In” where the Bit 0 / Pin 1 label is and select “Output - Pattern Generator 1”
  - Repeat this for the following 3 pins
- **Step 4: Generate a digital signal with the Pattern Generator**
  - Open the Pattern Generator from the icon
  - Set the baud rate to 250,000, this will automatically update the divider. Set the number of ticks to 8 and ensure repeat forever is toggled on; these are the defaults.
  - Click “Pin 1”, hover over “Fill Pin 1 with..”, and select “Clock” from the context menu. Enter “1” in “Ticks high/low” and click “Fill”.

- Click “Pin 2”, hover over “Fill Pin 2 with..”, and select “Ones” from the context menu.
- Click “Pin 3”, hover over “Fill Pin 3 with..”, and select “Pulse” from the context menu. Enter “1” in “Ticks high”, enter “3” in “Ticks low”, and click “Fill”.
- Click “Pin 4”, hover over “Fill Pin 4 with..”, and select “Random” from the context menu. Select “Manual” for the random number generator seed, enter “7”, and click “Fill”.
- See Pattern Generator for more ways to define your desired patterns.
- Toggle to enable the output of Pattern Generator 1.

- **Step 5: Configure the trigger**
  - In the trigger panel, select "Auto" for the trigger mode, and set the Nth event to 2
  - Change the trigger type from “Basic” to "Advanced”
  - Select “AND” for the combination (default) the set Pin 1 to “Rising edge” and Pin 3 to “Rising edge”
  - The display will now show a steady waveform
- **Step 6: Add a protocol decoder to convert the digital data into hexadecimal format**
  - To add a decoder, click the plus sign, in the top left of the screen.
  - Hover over "Add protocol decoder" and select "Parallel bus".
- **Step 7: Configure the parallel bus decoder**
  - Set the sample mode to “Both edges”, enter "3" for the data width, and select Pin 1 as the clock pin (CLK)
  - The parallel bus decoder will show a repeated series of hexadecimals following the pattern “0x7, 0x2, 0x3, 0x2”
- **Step 8: Characterize the digital signals using measurements and cursors**
  - In the measurements panel, click "Add" to add a measurement tile
  - Click on the tile, set the channel to "Pin 1" and set the measurement type to "Frequency"
  - Add another tile, set the channel to "Pin 1" and set the measurement type to "Period"
  - Add another tile, set the channel to "Pin 3" and set the measurement type to "Neg. width"
  - To add a time cursor, drag out from the cursor icon at the bottom right of the display.
  - Place a time cursor around -4 µs on a falling edge of Pin 1, place another cursor on the next falling edge of Pin 1, a time difference of 8 µs will appear between the cursors, this will match the period measurement tile we created earlier.
  - Place another cursor on the next rising edge of Pin 3, around 16 µs, the difference between the cursors will read 12 µs, closely matching the negative width measurement tile we created.
- **Step 9: Export your results**
  - Click the export icon to save and share your results.
  - Toggle on “Traces” and select “CSV”, toggle on “Protocol data”, and toggle on “Screenshot” and select “PNG”
  - Enter “QuickStartGuide” in the “Filename prefix” and “This is the user manual quick start guide” in the “Comments”
  - Either select “Share” or “My Files”, and then click “Export”

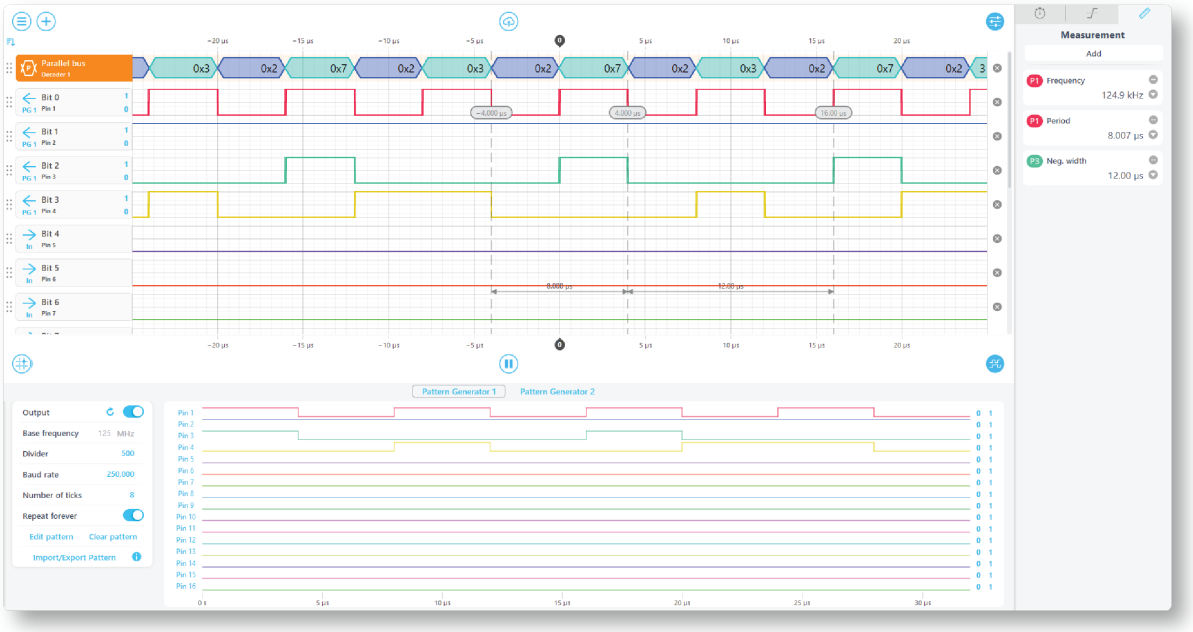


Figure 3. Screenshot of the Moku Logic Analyzer configured to output and readback the parallel bus pattern. Measurements are shown on the right.

System architecture

The Moku Logic Analyzer can be used to capture, analyze, and generate digital signals, the Digital Input/Output (DIO) pins or analog inputs and outputs can be used to achieve this.

The Logic Analyzer interprets digital information via the DIO pins (hardware dependent), or analog inputs. The Pattern Generators allow users to generate patterns through the DIO pins, or analog outputs. The Moku Logic Analyzer has an intuitive graphical user interface with extensive measurements readily available. Data, screenshots, and logs can be readily captured to email or cloud-based services for rapid sharing and evaluation.

The Moku Logic Analyzer supports two independent decoder channels that can be added to decode UART, I2C, I2S, CAN, parallel bus, USB, and SPI protocols. Moku is your solution for serial data decoding, custom digital design debugging, multi-protocol monitoring, automotive applications, industrial applications, and more.

An important distinction to note before reading further is that the bits follow 0-based indexing while pins follow 1-based indexing.

Input source selection

There are 3 options available for input source; "Digital I/O" (hardware dependent), "Analog Input", and "Slot Input".

For **"Slot Input"**, see Routing analog signals to the Slot Input of the Logic Analyzer. This source is only available in Multi-instrument Mode.

For the **"Analog Input"** source, each physical Analog Input corresponds to a displayed bit signal. The analog Input 1 maps to Bit 0, Input 2 maps to Bit 1 and so on. If the device has an external trigger available, this will appear as the last bit display and will have a fixed threshold. Refer to the Technical Specifications for the hardware dependent TTL threshold value liquidinstruments.com/resources/supporting-material/product-documentation/.

For the **"Digital I/O"** source, there is a simple and intuitive 1-to-1 mapping between Digital I/O pins and the displayed bit signals. The availability of the Digital I/O source is hardware dependent.

Physical interface

Moku:Go is equipped with a 20-pin digital I/O interface. 16 of the 20 pins are the bidirectional digital I/O. There are two ground pins, one 5 V output, and one 3.3 V output. Figure 4 shows the detailed DIO layout.

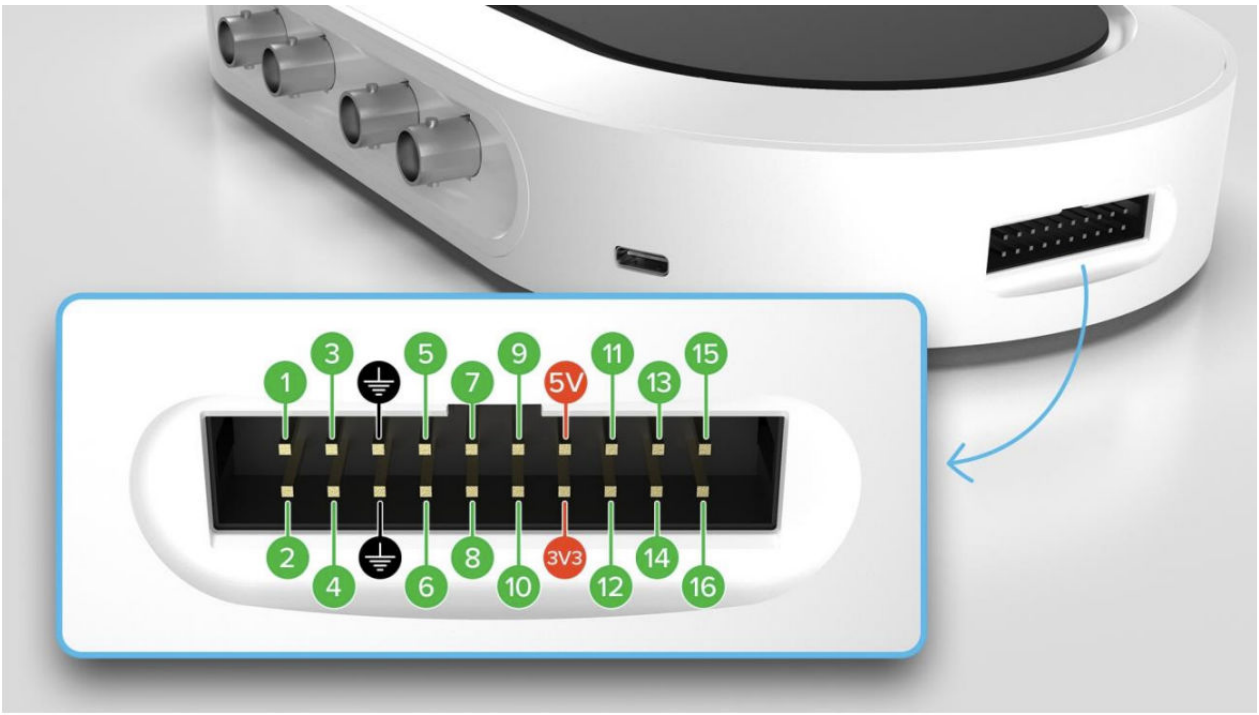


Figure 4. Physical DIO interface of the Moku:Go.

Moku:Delta is equipped with 32 Digital I/O pins, there are 2 ports of 16 bidirectional digital I/O pins, which run at LVDS. The supplied DIO module must be used to convert to LV CMOS ports, rated to output 3.3 V and receive 3.3 V or 5 V digital signals. Only one port of pins can be observed at a time, the sources will be **Digital I/O 1** and **Digital I/O 2**. A detailed layout can be found in Figure 5:

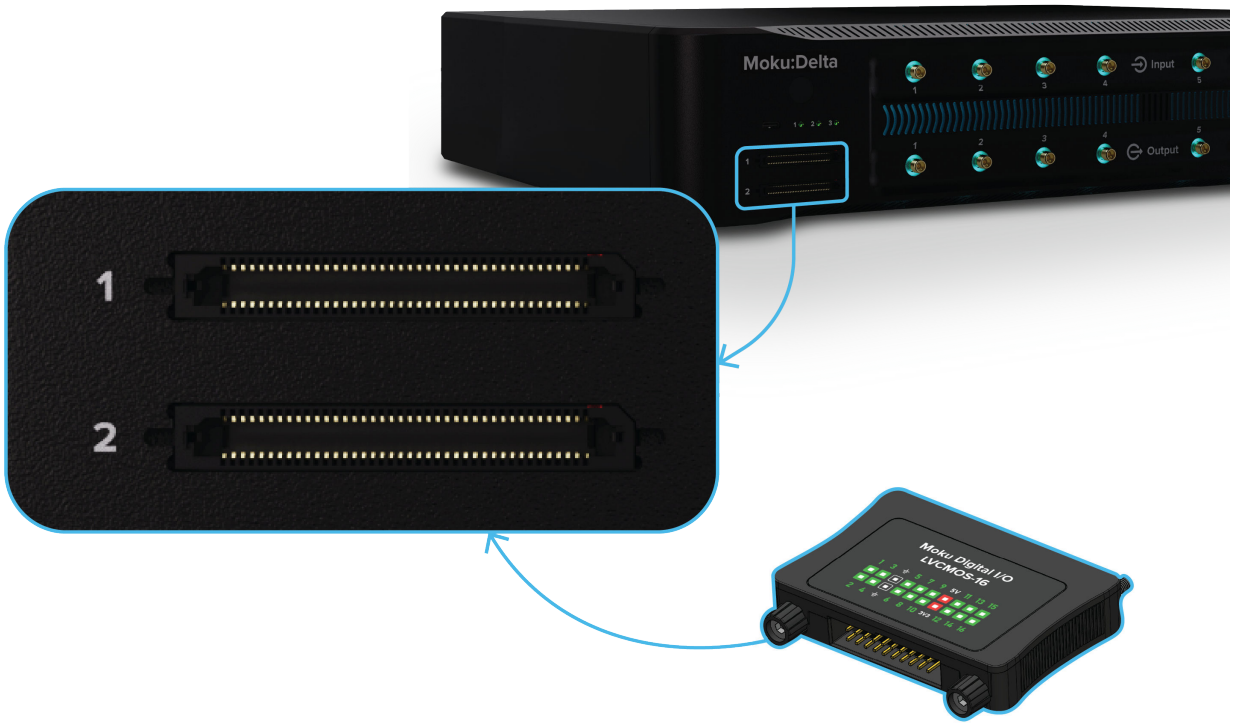


Figure 5. Moku:Delta Physical Digital I/O Interface

Pattern generation

The time scale of the pattern is determined by the base frequency, the baud rate, and the number of ticks. The divider or Baud rate can be adjusted to fix the resolution. These two values are inversely proportional, increasing one will cause the other to decrease.

Baud rate and divider

The baud rate represents the number of signal changes or symbols transmitted per second. In communications, it determines the rate at which information is transferred across a communication channel. The divider parameter is used to adjust the communication frequency by dividing the base frequency of the system. Using the divider offers more flexibility in setting the desired baud rate for communication.

Tick count

The number of ticks determines the total length of the sequence. 10 ticks of 10 microseconds will be 100 microseconds, 100 ticks of 10 microseconds will be 1 millisecond.

Thresholding

The thresholding options are single threshold or high/low thresholding.

Single thresholding

The threshold value set will determine the trigger-like functionality. When the signal moves above the threshold, the digital signal will display 1, when the signal moves below the threshold, the digital signal will display 0.

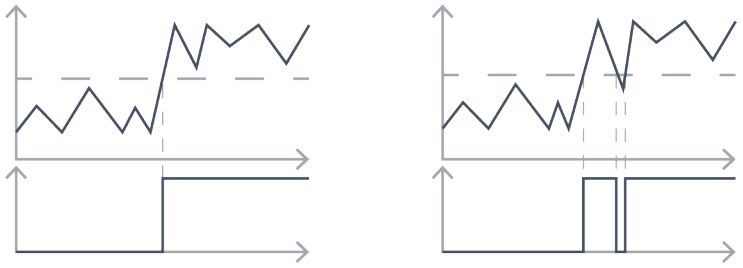


Figure 6. Single thresholding analog to digital examples

High/Low thresholding

The thresholding values will determine the hysteresis trigger-like functionality. The signal must pass between the low and high threshold for the bit value to change. If the signal crosses the lower threshold but does not cross the high threshold the signal will remain at 0, if the signal crosses the higher threshold but does not cross the low threshold, the signal will remain at 1. This is useful for minimizing transient errors in the signal.

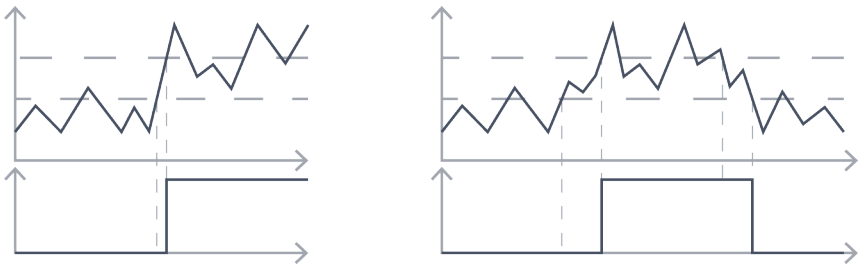


Figure 7. High/low thresholding analog to digital examples

Signal routing

When routing the signals in Multi-instrument Mode, there are a few things to consider. In Multi-instrument Mode, the signal from the DIO can be routed to an instrument that expects an analog voltage input. When this happens, the signal is interpreted as a 16-bit signed integer and converted into volts using a particular LSBs/volt scaling factor. The same theory goes for when the DIO is routed to an instrument that expects to output an analog voltage.

The Digital to Analog Converters (DACs) are the physical, analog outputs of the Moku device, and the Analog to Digital Converters (ADCs) are the physical, analog inputs of the Moku device. The theory above also applies when an ADC or DAC is routed to the input or output of the Logic Analyzer instrument slot in Multi-instrument Mode.

Routing the Pattern Generator or DIO to a Slot Input or DAC

Passing the Pattern Generator to the analog output will output a signal based on a parallel bus-like system. Bits 0 through 14 determine the absolute value of the output. Bit 15 determines the sign. The output is scaled to the output limits of the Moku. The maximum positive voltage will correspond to bits 0 to 14 all being one and bit 15 being zero. The maximum negative voltage will correspond to bits 0 through 14 all being zero and bit 15 being one.

Table 1. Output scaling

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Absolute value															Sign

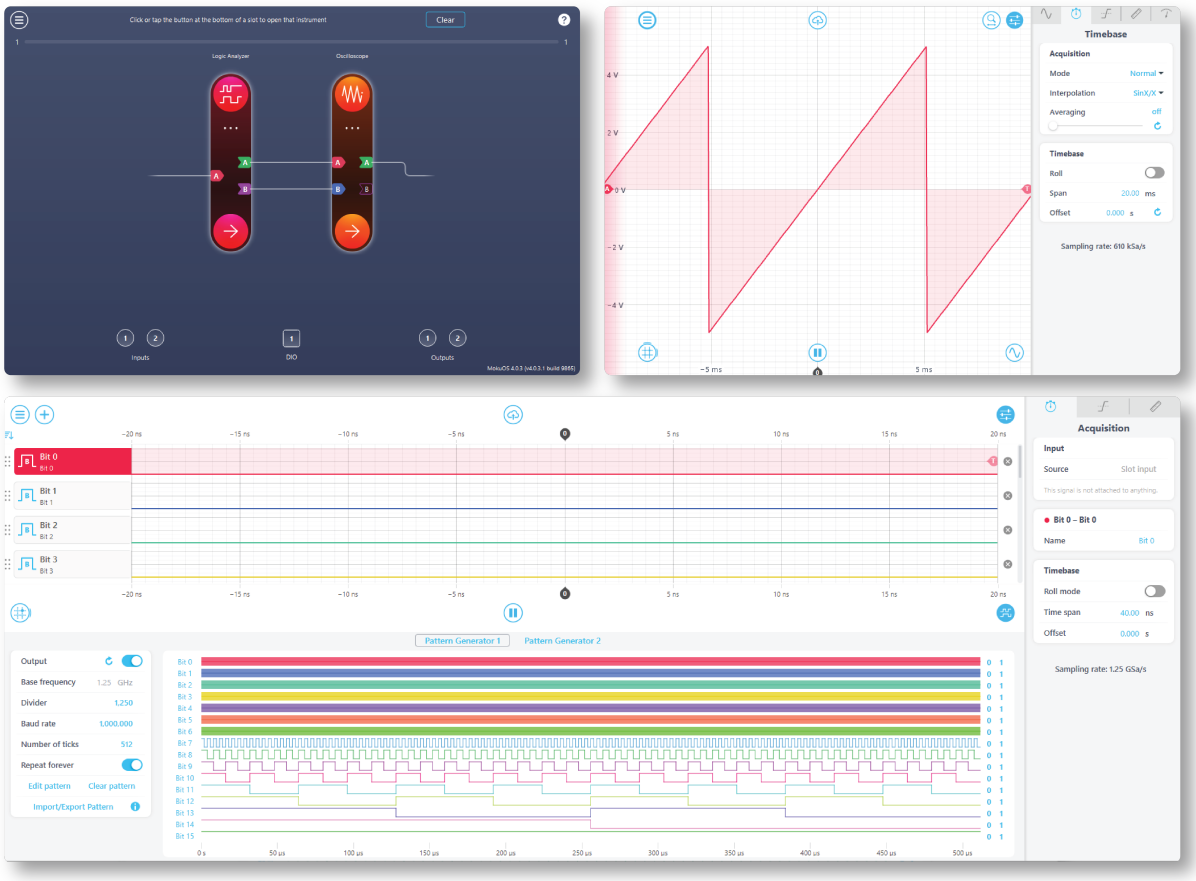


Figure 8. Routing the Pattern Generator to the analog outputs

Table 2. Example pin output

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Bit
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	Max
1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	
1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Zero
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Zero
1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	
1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	Min

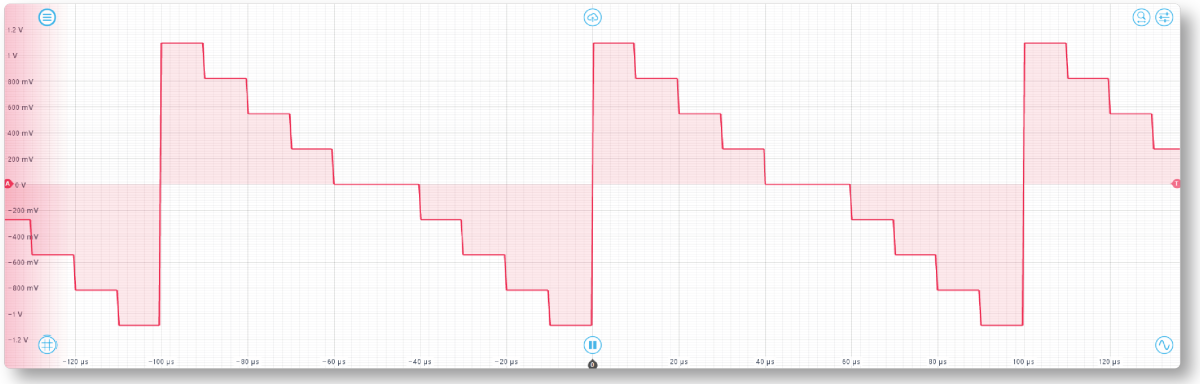


Figure 9. Example pin output

Routing a Slot Output or ADC to the Slot Input of the Logic Analyzer or DIO

The digital pins can be mapped to a 16 bit signal bus, ranking from the least significant bit (LSB) to the most significant bit (MSB), see Significant bits. For instance, Pin 1 corresponds to Bit 0 (LSB) in the 16 bit signal bus, while Pin 16 aligns with Bit 15 (MSB). In the depicted configuration shown in the screenshot below, the Waveform Generator produces a 16-bit ramp wave through the DIO; and simultaneously, Logic Analyzer visualizes the output waveforms on the 16 pins.

Figure 10. Routing analog signals to the Slot Input of the Logic Analyzer

The ramp waveform is set as 100 Hz repetition rate and the maximum amplitude. Waveforms on Logic Analyzer indicate that the higher-order bits experience fewer transitions, whereas the lower-order bits exhibit more transitions. This observation aligns with the anticipated behavior since the most significant bit (MSB) changes only twice within each cycle, while the least significant bit (LSB) changes with nearly every update of the waveform.

Significant bits

In digital systems, LSB (Least Significant Bit) and MSB (Most Significant Bit) are terms used to define the significance of bits in a binary number.

LSB (Least Significant Bit): This is the bit in a binary number that holds the smallest value. For example, it corresponds to Bit 0 in a 16-bit signal bus. It is the bit that changes most frequently when incrementing the value, as mentioned in the example of a Waveform Generator that produces a 16-bit ramp wave. In the context of Moku devices, the LSB of the digital output is significant due to the frequent transitions it exhibits with nearly every update of the waveform.

MSB (Most Significant Bit): This is the bit in a binary number that holds the highest value. In a 16-bit signal bus, it is Bit 15. The MSB changes less frequently as it represents the larger magnitude portion of the number. In the Waveform Generator example, the MSB changes only twice within each cycle of the waveform.

Bit order: MSB first and LSB first refers to the order the bits are sent or processed, this is often referred to as the bit order. MSB first means the most significant bit is sent or processed before the least significant bit. LSB first means the least significant bit is sent or processed first.

Some examples to visualize the complexity of where the `MSB` vs `LSB` are and what MSB first vs LSB first can look like:

0b01111 = 15	**0b11110 = 30**
`MSB 01111 LSB`	`MSB 11110 LSB`
`MSB First`	`MSB First`
`01111 = 15`	`11110 = 30`
`LSB First`	`LSB First`
`11110 = 15`	`01111 = 30`

Using the instrument

User interface

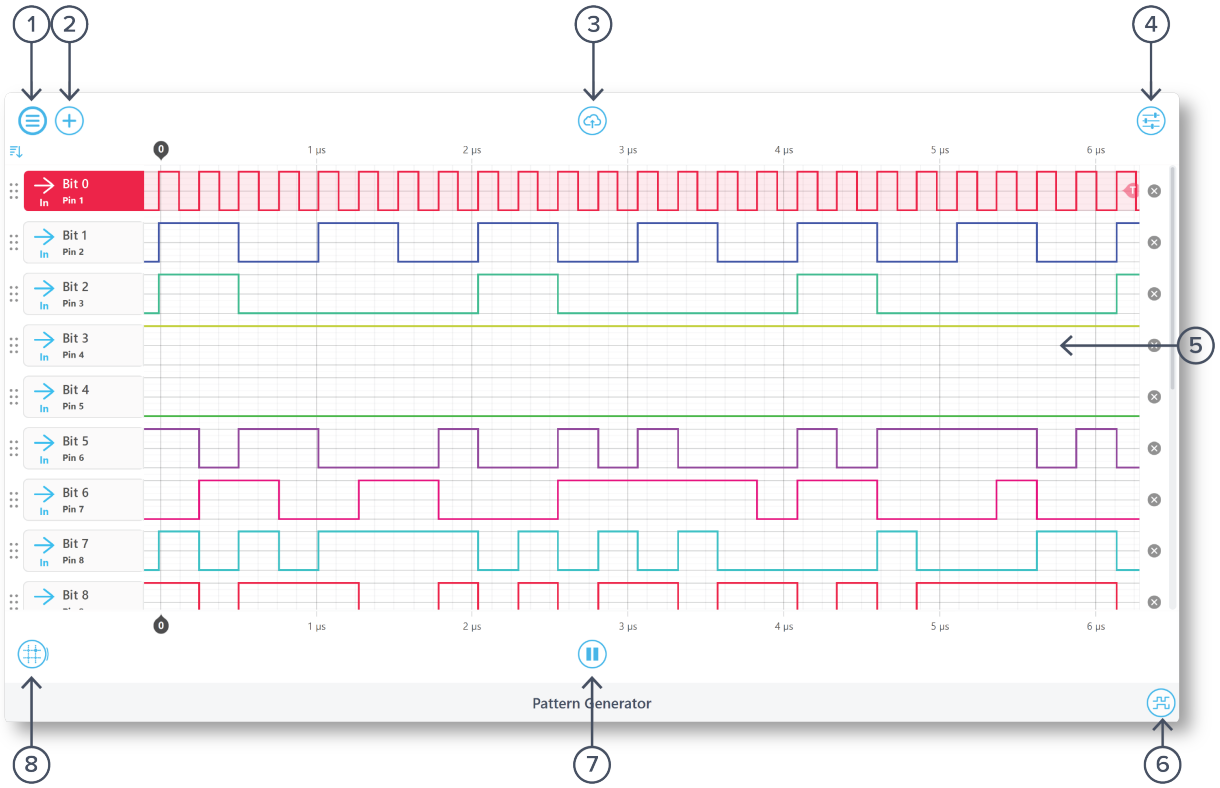


Figure 11. Logic Analyzer user interface

① Main menu
② Add channel
③ Save data
④ Settings
⑤ Signal display area
⑥ Output Pattern Generator
⑦ Input start/pause
⑧ Cursors

Display

The displayed signal can be moved around the screen by clicking anywhere on the signal display window and dragging to the new position.

Scrolling the mouse wheel zooms in and out along the time axis.

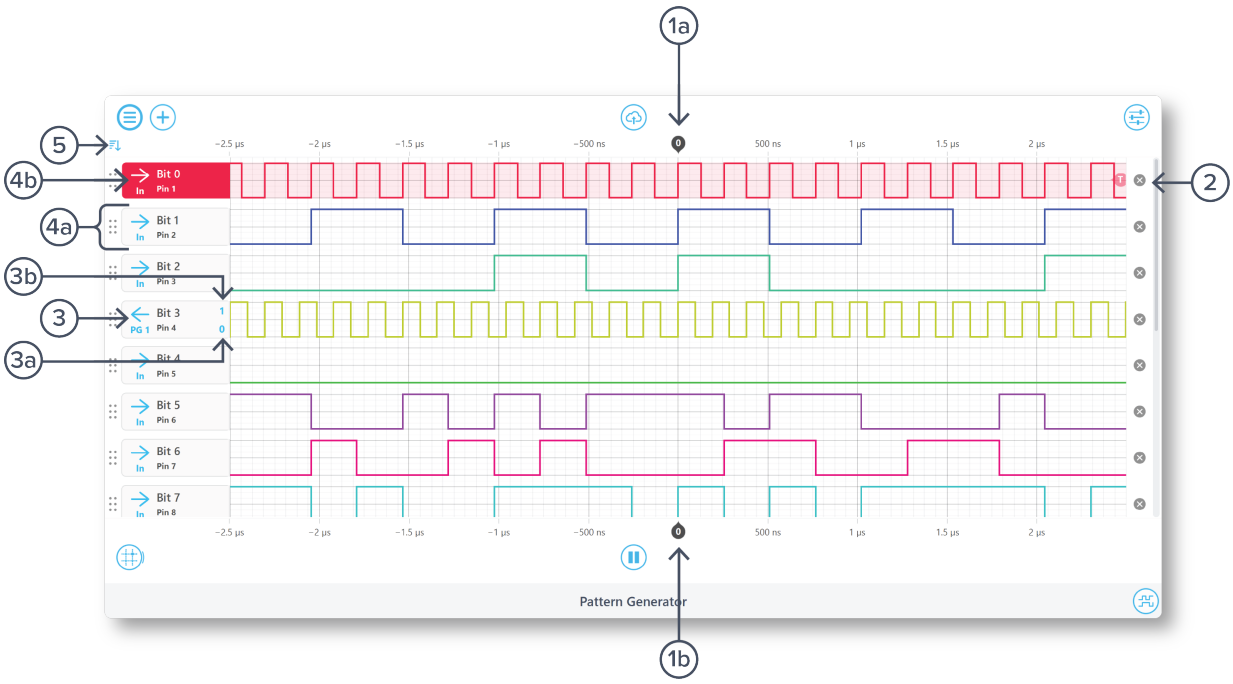


Figure 12. Logic Analyzer Display signal and trace

① **Top and bottom time origin mark**: Marks the “zero second” point on the time scale. This will be the trigger point if the Logic Analyzer has triggered.

② **Remove trace**: Click here to remove the trace ⊗.

③ **Output pin header**: Signal header for Pin 4. The left pointing arrow indicates it is currently set to be an output channel from Pattern Generator 1. Click the arrow to switch the pin between "Input" and the "Output - Pattern Generator" options.

③a **Low override**: Click to override this output to Low.

③b **High override**: Click to override this output to High.

④a **Input pin header**: Signal header for Pin 2. The right pointing arrow indicates it is currently set to be an input channel. Click the arrow to switch direction. Click-hold and drag the six dots to re-arrange the order of the signals.

④b **Active pin header**: Click the signal trace area or pin header for any pin to make it the active signal. This allows user to access settings and pattern editor for this pin. Click the header again to deselect the active pin.

⑤ **Sort channels**: Sort channels by channel number or channel type.

Add channel

Additional channels can be added to the screen by clicking the ⊕ icon.

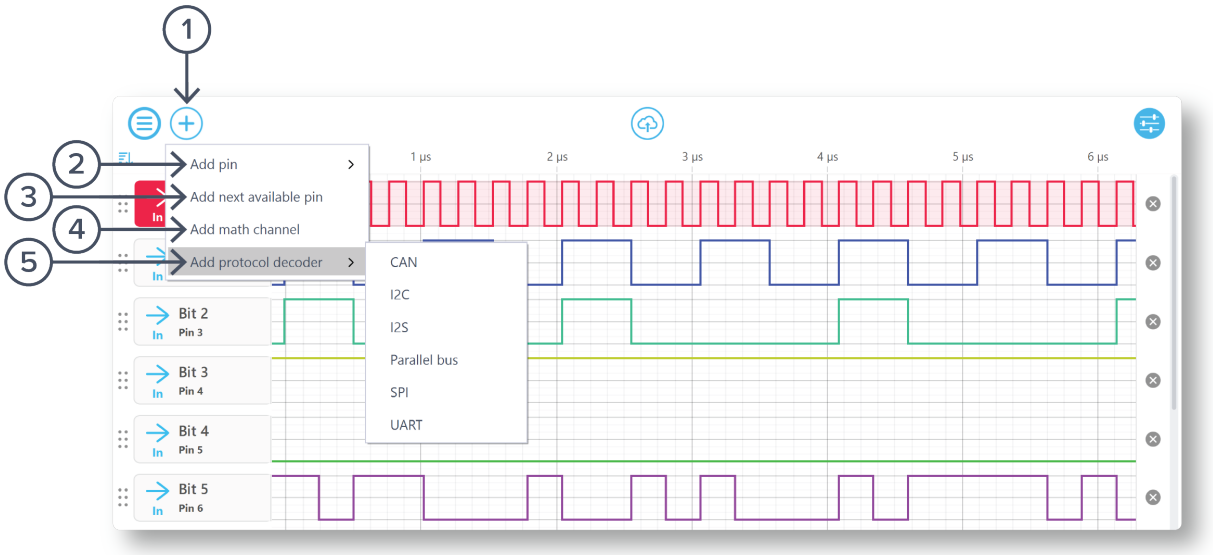


Figure 13. Add channels

① Add a pin or channel
② Select a specific pin to add
③ Add the next pin that is not currently in use, shortcut: Ctrl/Cmd+Shift+N
④ Add math channel
⑤ Add protocol decoder channel

Controls panel

The settings options can be accessed by clicking the ⚙ icon, allowing you to reveal or hide the controls panel, and giving you access to all instrument settings. The controls panel contains settings and measurements.

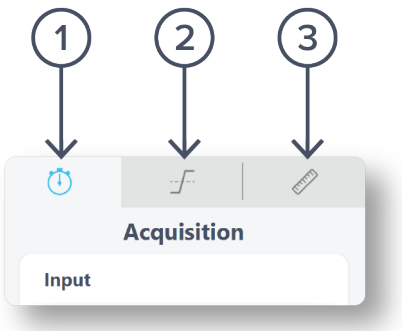


Figure 14. Controls panel

① Acquisition panel
② Trigger panel
③ Measurements panel

Acquisition

The acquisition pane allows you to configure the input source, active pins, math channels, protocol decoders, and timebase.

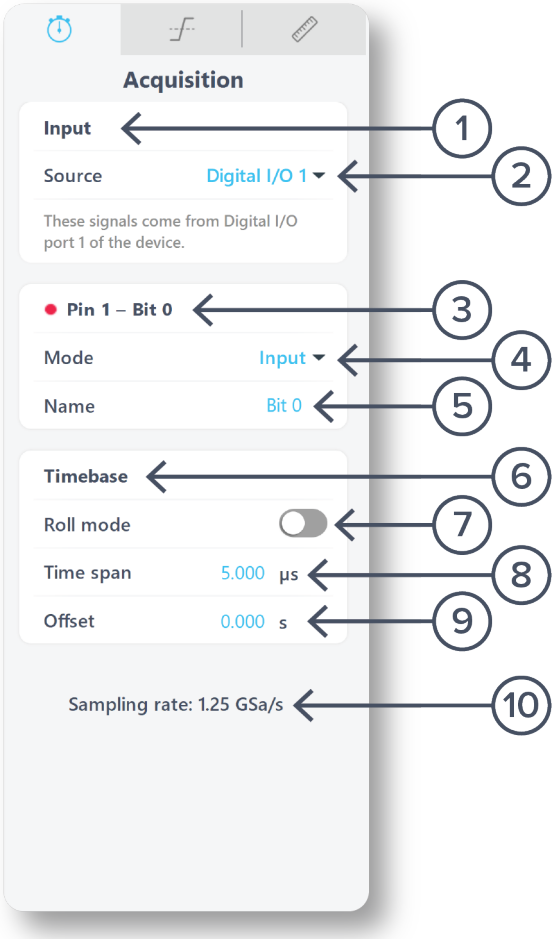


Figure 15. Acquisition side panel

① Input sub-panel
② Select the input source for all pins to use (Digital I/O, Analog Input, Slot Input)
③ Channel sub-panel
④ Set the mode of the selected pin to either an Input option or one of the Output Pattern Generators
⑤ Change the name of the selected pin
⑥ Timebase sub-panel
⑦ Toggle to enable roll mode
⑧ Enter the time span for the horizontal screen scale. Changes dynamically when zooming in and out of a trace, or can be entered manually.
⑨ Enter an offset for the horizontal trigger point offset. Changes dynamically when horizontally-dragging a trace or can be set manually.
⑩ Displayed sampling rate of the Logic Analyzer

Analog input

For further information about selecting analog input as the input source and thresholding, see Input source selection and Thresholding.

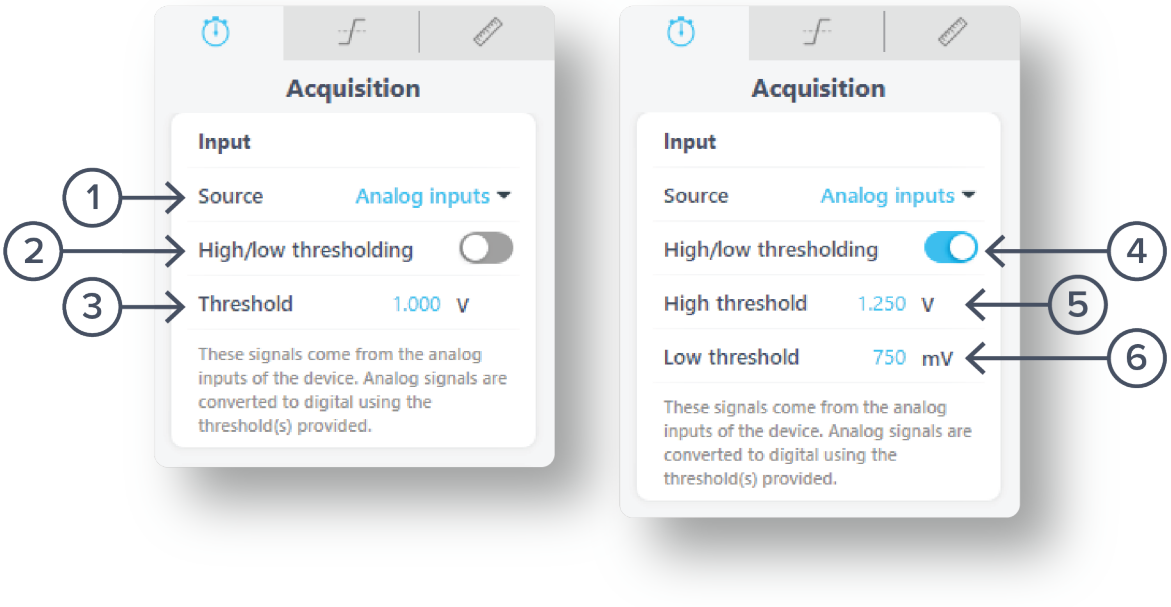


Figure 16. Acquisition Input subpanel for Analog input selection

① Input source selection set to Analog inputs
② High/low thresholding toggled OFF
③ Enter value for single threshold
④ High/low thresholding toggled ON
⑤ Enter value for High threshold
⑥ Enter a value for Low threshold

Math channel

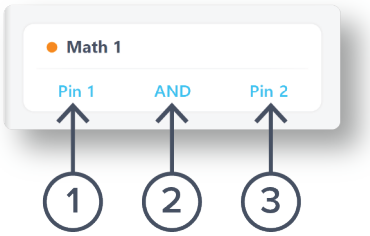


Figure 17. Logic Analyzer Math channel

① Select the first source for the math operation
② Select from AND, OR, XOR, NAND, NOR, XNOR operation
③ Select the second source for the math operation

Trigger

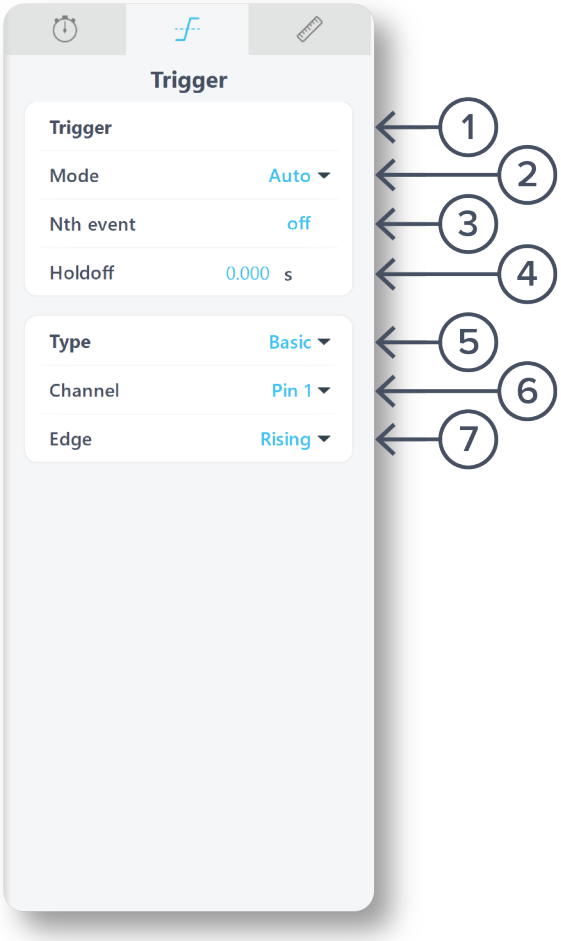


Figure 18. Logic Analyzer Trigger settings

① Trigger sub-panel
② Select between auto, normal and single trigger modes
③ Enter up to 65,535 trigger events before actually triggering, entering one or zero will set it back to "off"
④ Enter a time to holdoff trigger post trigger event
⑤ Select between basic or advanced trigger mode
⑥ Select the source for the trigger circuit
⑦ Select to trigger on rising, falling, or both edges

Advanced trigger mode

In the advanced trigger mode, the user can select to trigger from multiple channels, with OR or AND combinational logic.

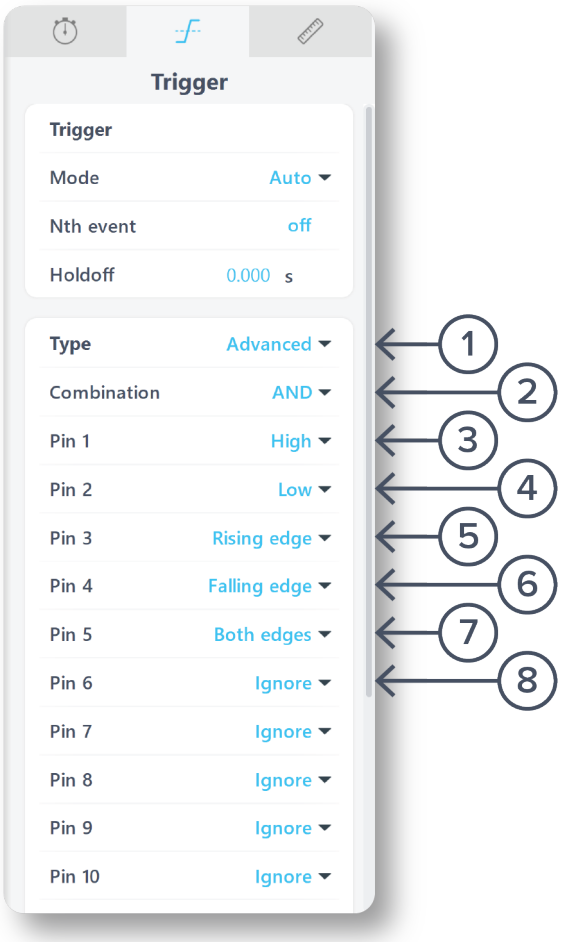


Figure 19. Advanced trigger mode

① Select advanced trigger mode
② Select the combinational logic (OR or AND)
③ Select High to trigger when the pin value is high/equal to one
④ Select Low to trigger when the pin value is low/equal to zero
⑤ Select Rising edge to trigger when the pin value moves from low to high
⑥ Select Falling edge to trigger when the pin value moves from high to low
⑦ Select Both edges to trigger whenever the pin value transitions
⑧ Select Ignore to ensure the pin isn't considered when triggering

Protocol decoders

The protocol decoder channels can be added via ⊕ the button. Detailed settings for each protocol can be configured under the acquisition pane when it is selected as the active channel.

UART

The Moku Logic Analyzer implements a UART decoder, a crucial tool for serial communication analysis. UART, or Universal Asynchronous Receiver/Transmitter, transmits data between devices by converting parallel data into serial form and vice versa. The Moku Logic Analyzer's UART decoder supports baud rates up to 2,000,000, with options for configuring data width, stop width, parity, and bit order. It allows users to monitor UART protocols, ensuring correct data transmission and debugging communication issues like mismatched baud rates or incorrect parity bits. Users can visualize decoded UART data in hexadecimal, aiding in efficient data interpretation and debugging.

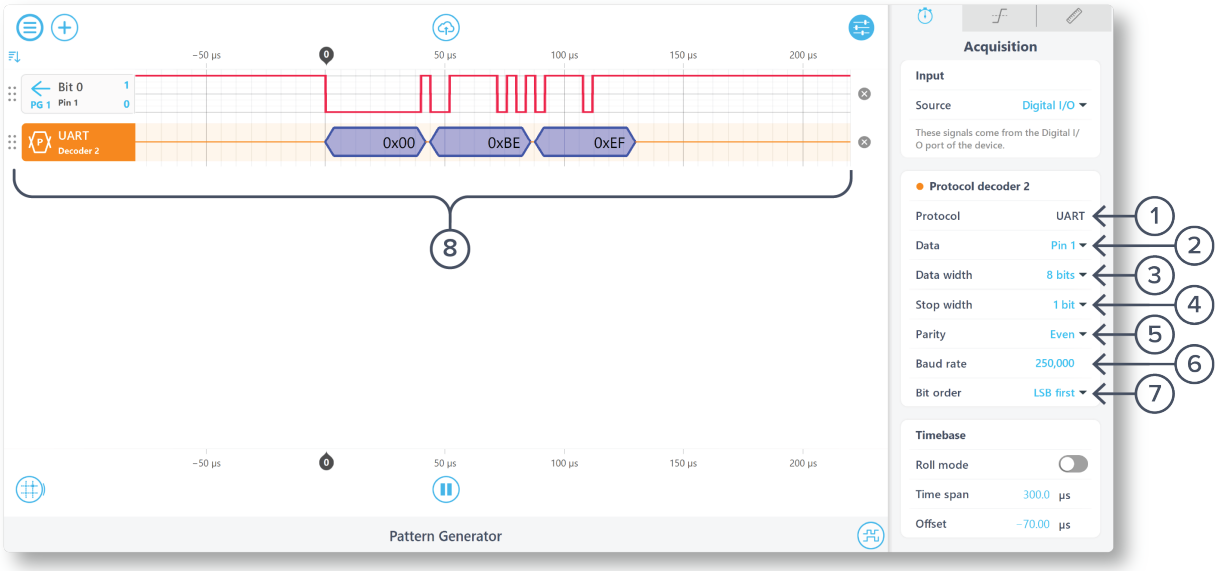


Figure 20. UART protocol decoder

① Protocol label
② Select a bit signal to be decoded for the Data pin
③ Select the Data width (5, 6, 7, 8, or 9 bits)
④ Select the Stop width (1 bit or 2 bits)
⑤ Select the Parity (Odd, Even, or None)
⑥ Enter the Baud rate
⑦ Select the Bit order (MSB or LSB)
⑧ Displayed decoded protocol

I2C

The Moku Logic Analyzer's I2C decoder supports decoding of I2C communications with address sizes of 7 bits and a maximum frequency exceeding 1 MHz. It facilitates the monitoring and analysis of I2C signals, ensuring accurate data exchange verification and troubleshooting in applications involving serial clock and data lines. An I2C protocol consists of a start condition, a 7 bit address, a read/write (R/W) bit, an acknowledgment (ACK) bit, the message bytes (each with a maximum length of 8 bits), and a stop condition.

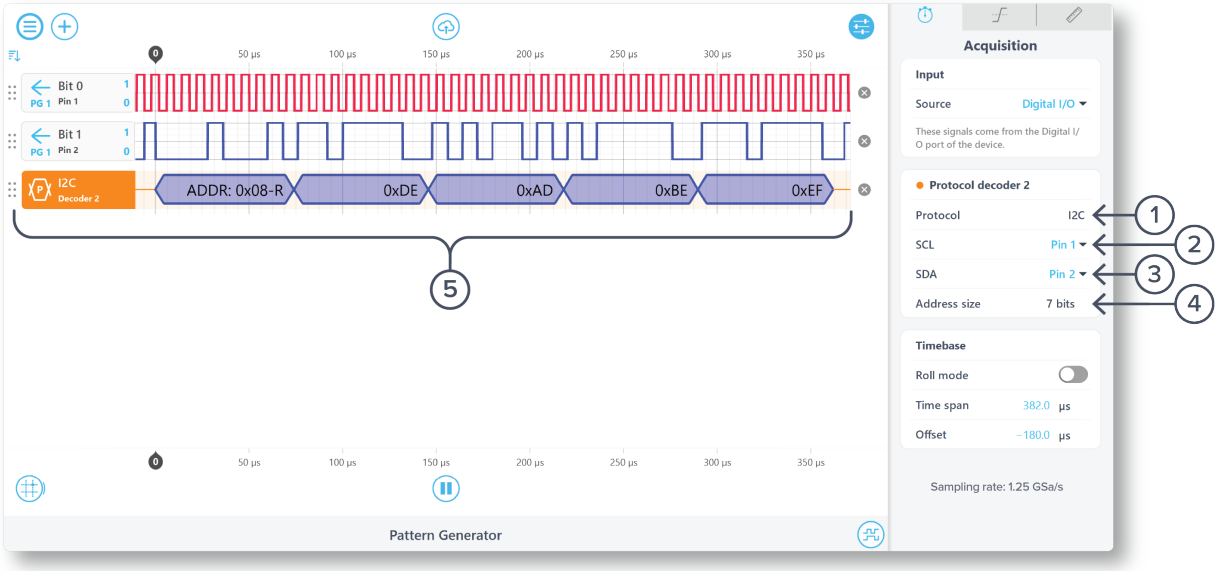


Figure 21. I2C protocol decoder

① Protocol label
② Select a bit signal for the serial clock pin SCL
③ Select a bit signal for the serial data pin SDA
④ Allowed address size (7 bits)
⑤ Displayed decoded protocol

I2S

The Moku Logic Analyzer's I2S decoder supports audio data transmission, featuring configurable bit order and data width from 2 to 32 bits. It enables analysis of digital audio interfaces, useful in audio tests and automotive systems, by decoding serial clock, word select, and serial data signals efficiently.

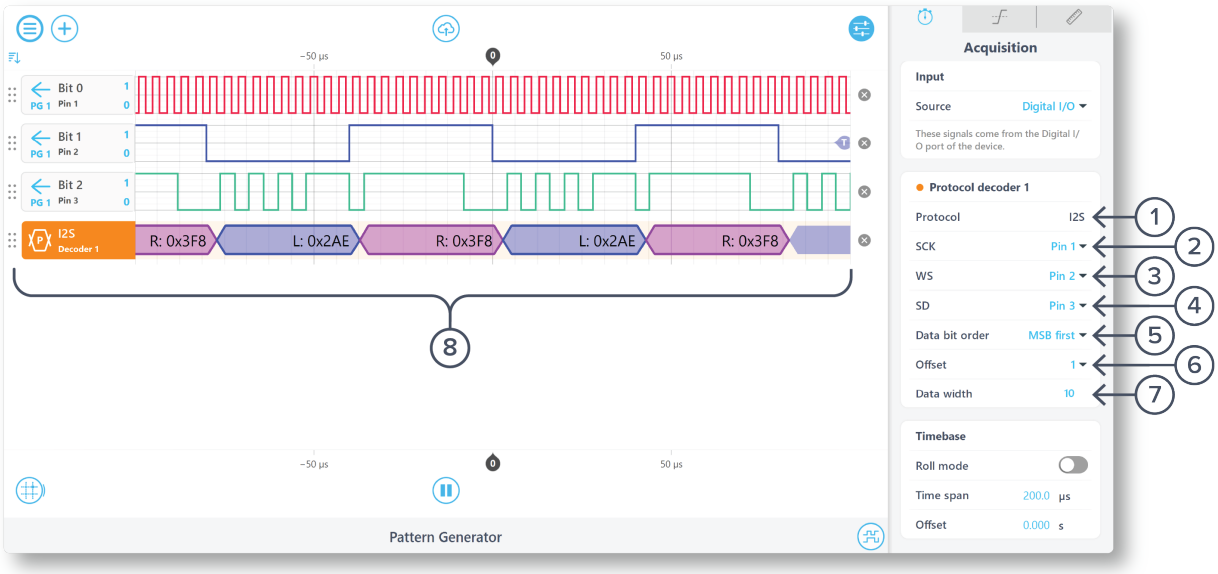


Figure 22. I2S protocol decoder

① Protocol label
② Select a bit signal for the serial clock pin SCK
③ Select a bit signal for the word select pin WS
④ Select a bit signal for the serial data pin SD
⑤ Select the Data Bit order (MSB or LSB)
⑥ Select the Offset (1 or 2 bits)
⑦ Set the Data width
⑧ Displayed decoded protocol

CAN

The Moku Logic Analyzer's CAN decoder analyzes Controller Area Network (CAN) communication. Supporting baud rates up to 1 Mbps, it decodes key protocol elements, including message IDs and data frames, assisting in automotive and industrial applications. The decoder ensures signal integrity and facilitates troubleshooting by providing comprehensive data insights.

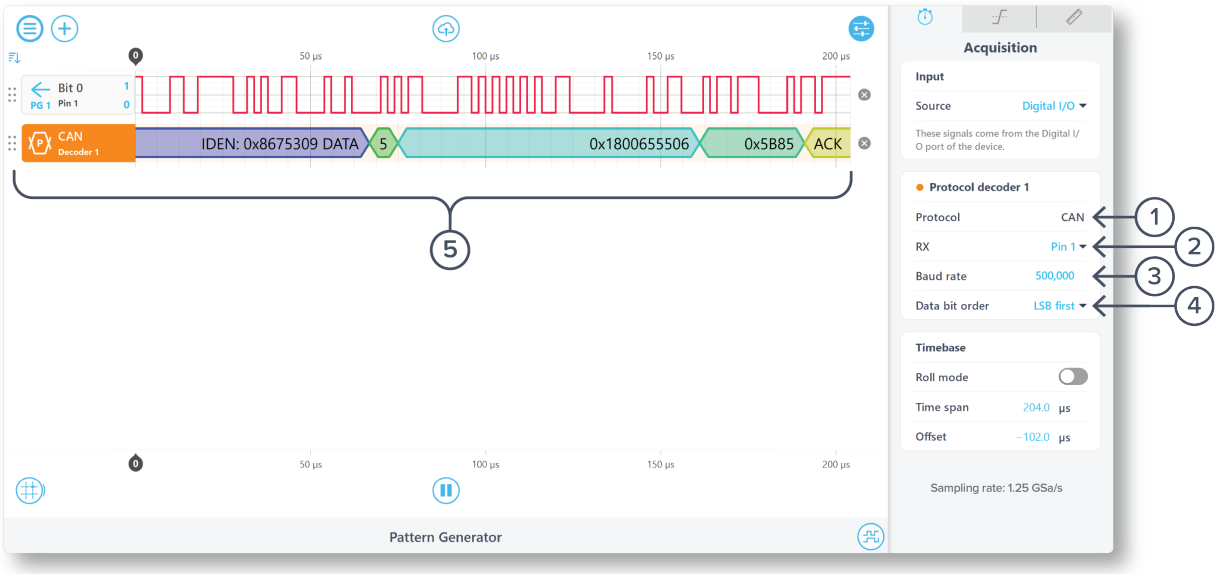


Figure 23. CAN protocol decoder

① Protocol label
② Select a bit signal to be decoded for the RX pin
③ Enter the Baud rate
④ Select the Data Bit order (MSB or LSB)
⑤ Displayed decoded protocol

Parallel

The Moku Logic Analyzer's parallel bus decoder facilitates protocol analysis by decoding data transmitted across multiple lines simultaneously. Configurable parameters include sample mode, data width, and clock bit, supporting rising, falling, or both edges. It's ideal for capturing wide data words efficiently, enabling quick debugging and verification of parallel communications.

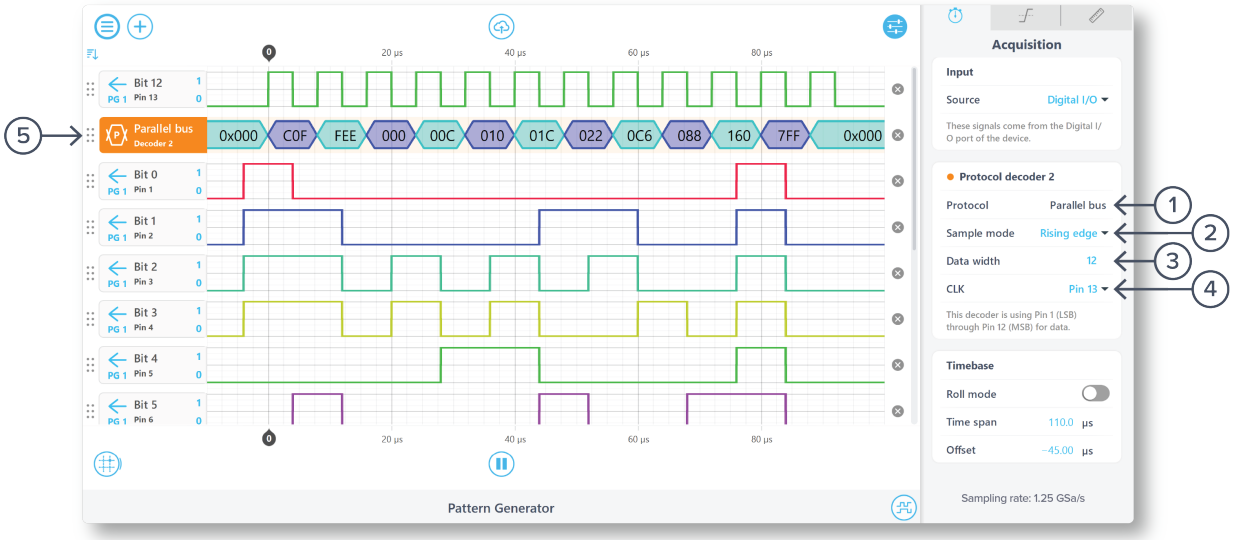


Figure 24. Parallel bus protocol decoder

① Protocol label
② Select the Sample mode (Rising edge, Falling edge, or Both edges)
③ Set the data width (up to 12)
④ Select a bit signal for the clock pin CLK
⑤ Displayed decoded protocol

SPI

The Moku Logic Analyzer provides an SPI decoder for analyzing Serial Peripheral Interface (SPI) communications. It supports clock polarity, phase adjustments, and a maximum decoder frequency of 5 MHz. This facilitates debugging and verification of SPI data exchanges in embedded systems by monitoring CLK, CS, and DATA lines.

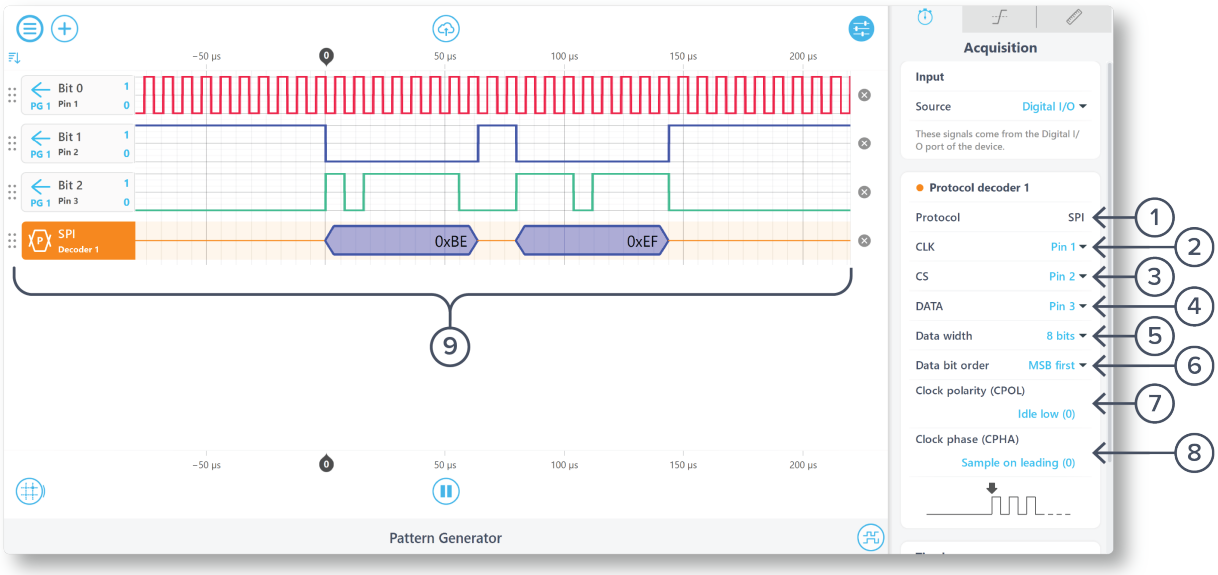


Figure 25. SPI protocol decoder

① Protocol label
② Select a bit signal for the clock pin CLK
③ Select a bit signal for the chip select pin CS
④ Select a bit signal for the data pin DATA
⑤ Select the Data width (5, 6, 7, 8, or 9 bits)
⑥ Select the Data Bit order (MSB or LSB)
⑦ Set the clock polarity CPOL (Idle low 0, or Idle high 1)
⑧ Set the clock phase CPHA (Sample on leading 0, or Sample on trailing 1)
⑨ Displayed decoded protocol

Clock polarity and phase

Configure the clock polarity and phase with respect to the data. The two options are CPOL and CPHA, for clock polarity and clock phase, respectively.

Clock polarity CPOL represents the polarity of the clock. Polarities can be converted with a simple inverter. When CPOL is 0, the clock idles at low voltage, when CPOL is 1, the clock idles at high voltage.

Figure 26. CPOL=0 - Idle low (left), CPOL=1 - Idle high (right)

Clock phase CPHA represents the phase of each data bit's transmission cycle relative to the clock, CLK. When CPHA is 0, sampling occurs when the clock transitions away from its idle

voltage level, meaning it will sample on the leading edge. When CPHA is 1, sampling occurs when the clock transitions to its idle voltage level, meaning it will sample on the trailing edge.

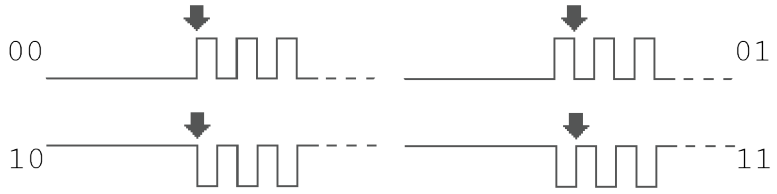


Figure 27. CPHA=0 - Sample on leading (left), CPHA=1 - Sample on trailing (right); CPOL=0 - Idle low (top), CPOL=1 - Idle high (bottom)

USB

The Moku Logic Analyzer provides a USB decoder for analyzing Universal Serial Bus (USB) communications. It supports a low speed baud rate (1,500,000) and custom baud rates from less than 100 to over 3,000,000. The USB protocol decoder supports the decoding of Low Speed USB 1.0 and 1.1 protocols. This facilitates debugging and verification of USB data exchanges in embedded systems by monitoring the D+ and D- data pins.

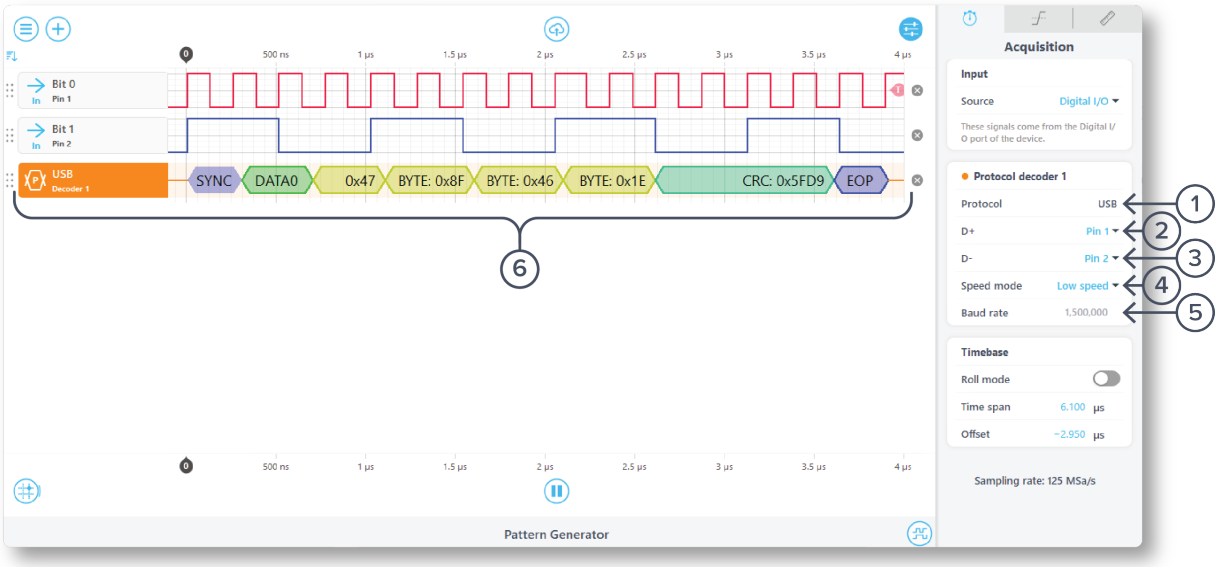


Figure 28. USB protocol decoder

① Protocol label
② Select a bit signal for the data pin D+
③ Select a bit signal for the data pin D-
④ Select a Speed mode (Low speed or Custom)
⑤ Enter the Baud rate (only in Custom speed mode)
⑥ Displayed decoded protocol

Pattern Generator

The output pattern can be accessed by clicking the icon. The Moku Logic Analyzer is equipped with two independent Pattern Generators. Each Pattern Generator can store a pattern for all 16 pins. User can select to output a pattern for a specific pin from Pattern Generator 1 or 2.

It is important to note that the Pattern Generator is available in Multi-instrument Mode for all hardware platforms and in Single-instrument Mode for hardware platforms that feature physical Digital I/O pins.

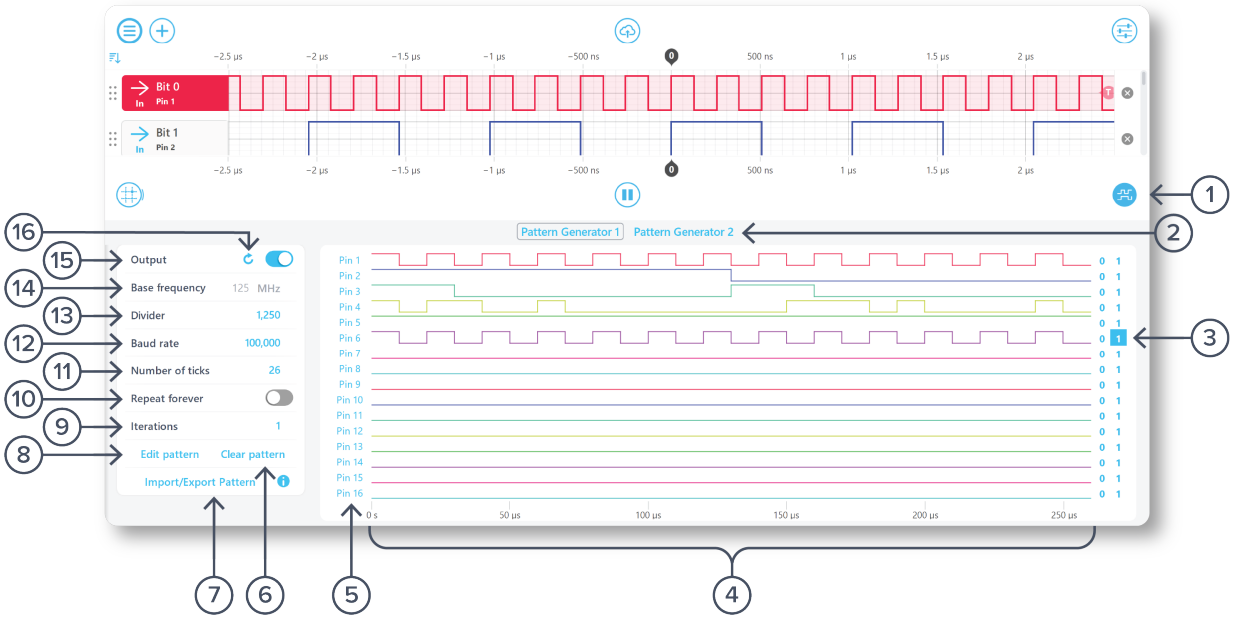


Figure 29. Logic Analyzer Pattern Generator

① Click to show or hide the Pattern Generator panel
② Switch to configure Pattern Generator 1 or 2
③ Click to override this output with "High" (1) or "Low" (0) without editing the pattern currently on the pin
④ Preview of the patterns and the associated pattern length
⑤ Click to edit the pattern and open a context menu
⑥ Click to clear all patterns
⑦ Click to import or export patterns *
⑧ Click Edit pattern to open the Pattern Editor pop-up
⑨ Enter the number of iterations for the Pattern Generator to perform
⑩ Toggle to repeat the pattern continuously
⑪ Enter tick count to adjust the length of the pattern
⑫ Enter the Baud rate, this will adjust dynamically when the divider is entered
⑬ Enter the Divider, this is the decimation factor from the base frequency, and will adjust dynamically when the Baud rate is entered
⑭ Base sampling frequency, this value is static and hardware dependent
⑮ Toggle to enable the Pattern Generator output
⑯ Click to restart the pattern playback if repeat forever is not enabled

* Import a CSV text representation of a pattern. Each column is a pin, each row is a tick (up to a maximum of 32,764 rows), and each value is 0 for low or 1 for high.

Pin numbers may be specified in an optional header row. e.g., "Pin 2, Pin 5". If no header row is present, pin numbers are assumed to start from 1 and increment each column.

Alternatively, a pattern may be imported into an individual pin by clicking a pin name in the pattern preview. In this case, if there's only one row then each column will be treated as a tick, otherwise each row will be treated as a tick and you will be asked to choose which column to import if there is more than one column.

The context menu options for importing or exporting patterns are:

- Load pattern from a file on your device
- Save the current pattern to a file on your device
- Paste pattern from your clipboard
- Copy the current pattern to your clipboard

Pattern editor

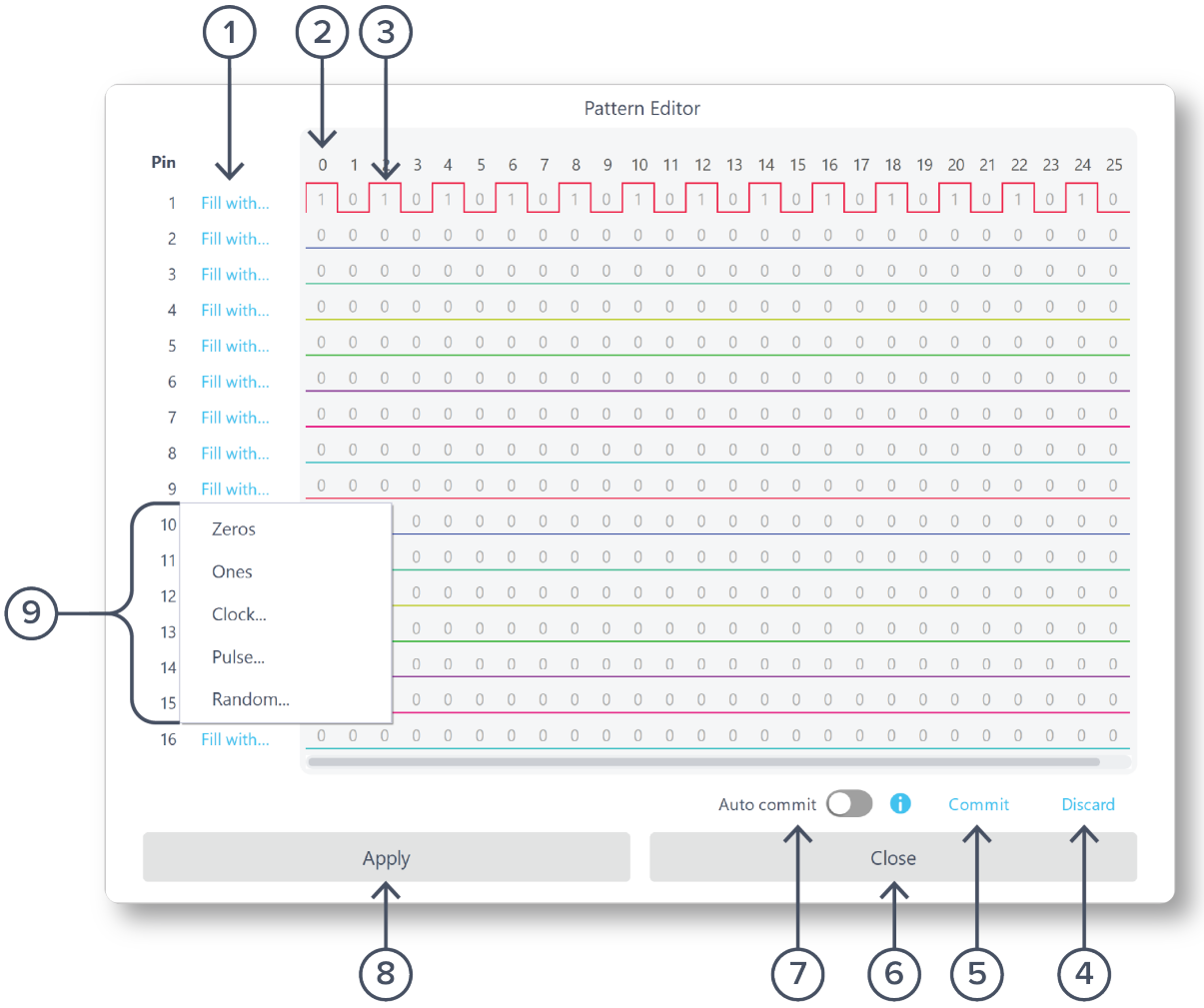


Figure 30. Pattern editor pop-up

① Click to open the Fill with.. context menu, to fill the pin with a specific pattern, see ⑨
② Tick number
③ Click to manually flip the bit
④ Discard the changes
⑤ Commit the changes
⑥ Click to close the editor
⑦ Toggle to enable or disable auto commit
⑧ Click to apply the changes and close the editor
⑨ Fill with.. context menu to fill the pin with a specific pattern, the fill options are: Zeros, Ones, Clock, Pulse, and Random

Clock fill

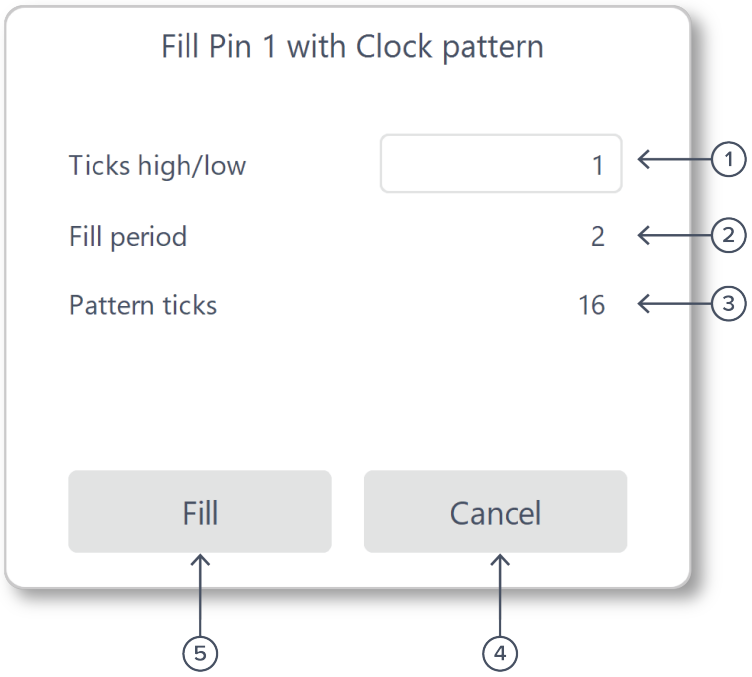


Figure 31. Pattern Editor Clock fill

① Enter the tick count for the high and low durations. In clock fill, the high tick count is equal to the low tick count
② Fill period will dynamically update based on the ticks
③ Pattern ticks is equal to the Number of ticks entered in the Pattern Generator
④ Click to discard the changes and close the Clock fill pop-up
⑤ Click to fill the pin with the clock pattern and close the pop-up

Ensure the fill period fits evenly within the pattern ticks to avoid truncation. A warning message will appear if this is the case:

Possible truncation ! Pattern ticks not evenly divisible by fill period. Some truncation may occur.

Pulse fill

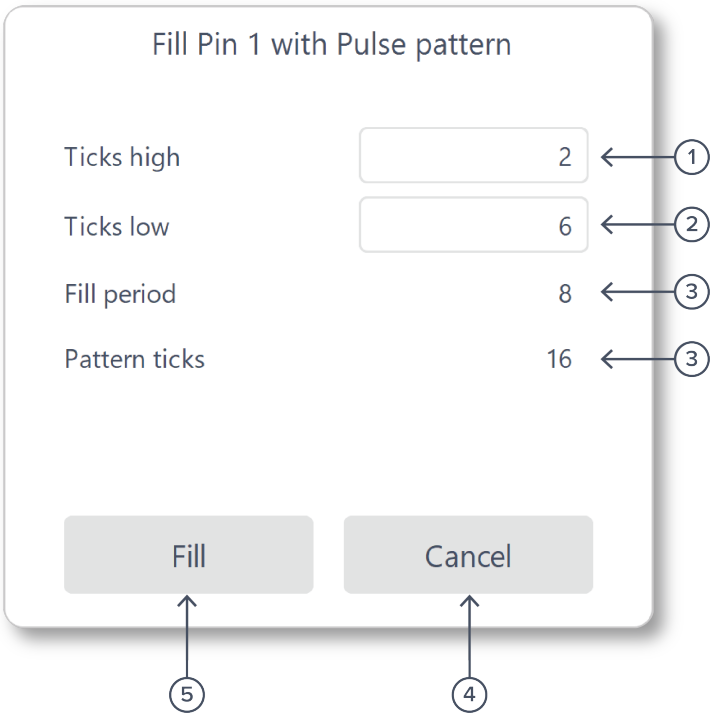


Figure 32. Pattern Editor Pulse fill

① Enter the tick count for the high duration of the pattern
② Enter the tick count for the low duration of the pattern
③ Fill period will dynamically update based on the ticks
④ Pattern ticks is equal to the Number of ticks entered in the Pattern Generator
⑤ Click to discard the changes and close the Pulse fill pop-up
⑥ Click to fill the pin with the pulse pattern and close the pop-up

Ensure the sum of the ticks high and low, or the fill period, fits evenly within the pattern ticks to avoid truncation. A warning message will appear if this is the case:

Possible truncation Pattern ticks not evenly divisible by fill period. Some truncation may occur.

Random fill

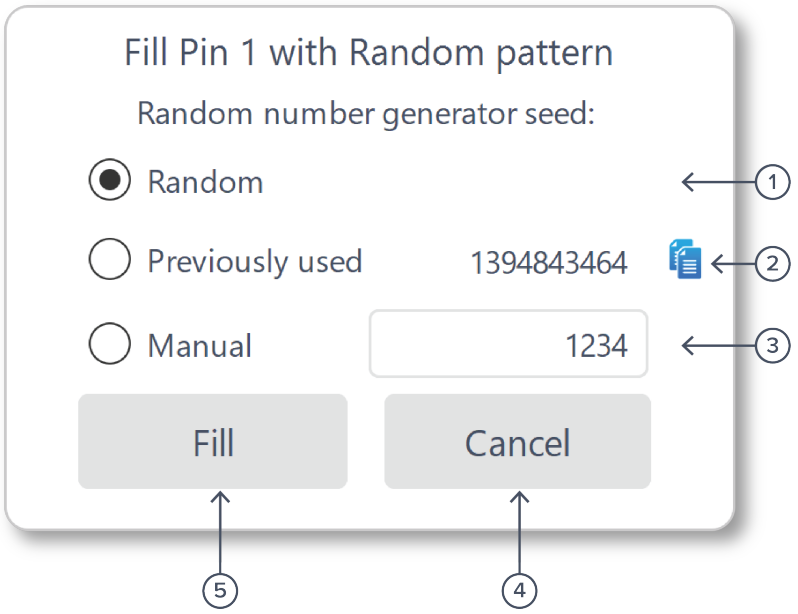


Figure 33. Pattern Editor Random fill

① Click to select a new random fill seed
② Click to select a previously used seed, click the icon to copy this value to your clipboard
③ Select to manually enter a seed, and enter the desired value
④ Click to discard the changes and close the Random fill pop-up
⑤ Click to fill the pin with the random pattern and close the pop-up

Measurements

The measurement pane allows you to add/remove measurements. A measurement can be assigned to a specific input, output, math channel, or difference between any two channels.

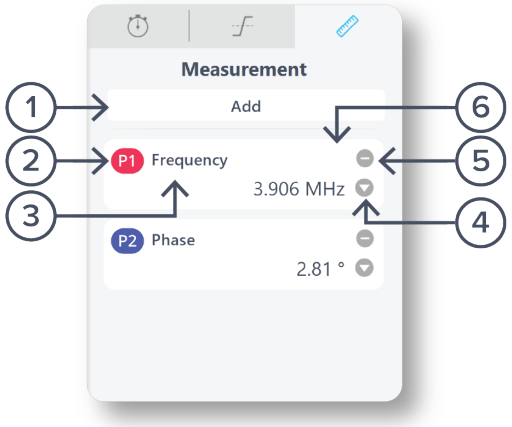


Figure 34. Measurement side panel

① Click to add additional measurement tile
② Measurement source. Click to loop through the measurement sources
③ Measurement type
④ Click to open the measurement's statistics
⑤ Click to remove the measurement tile
⑥ Measurement value

To adjust a measurement, click a measurement tile to open the menu, with the following options.

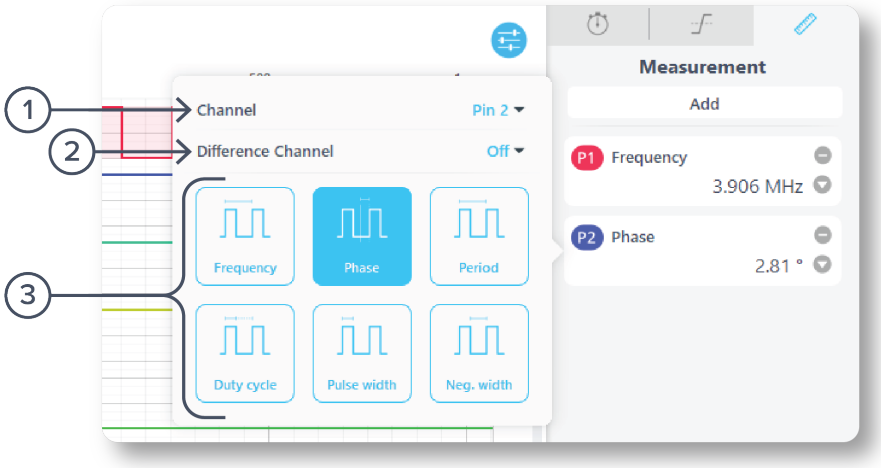


Figure 35. Measurement tile options

① Select measurement source
② Measure the difference between the measurement source to another channel
③ Select the measurement type (Frequency, Phase, Period, Duty cycle, Pulse width, Neg width)

Cursors

The cursors can be accessed by clicking the icon, allowing you to add time cursors or remove all cursors. You can right click on a cursor to set it as the reference or remove the cursor.

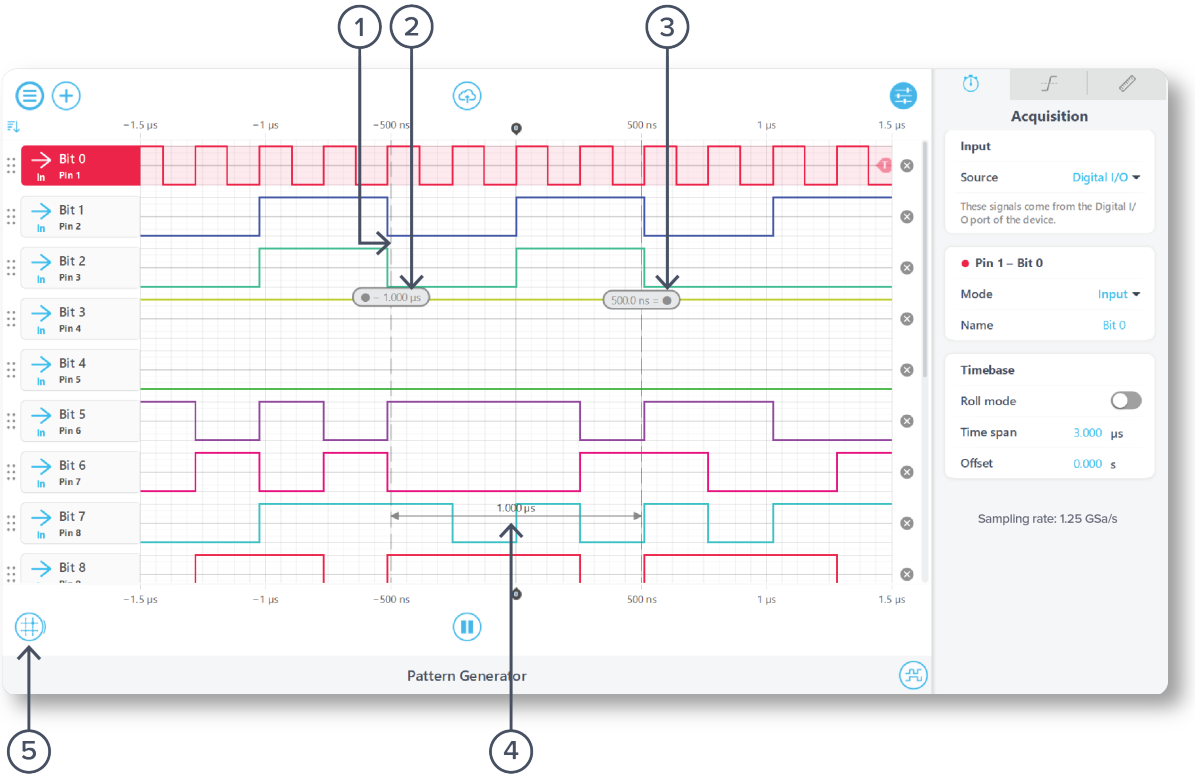


Figure 36. Logic Analyzer Cursors

① **Time cursor**: Drag left or right to set position
② **Time reading**: Click to enter time, right click to reveal time cursor options
③ **Reference indicator**: Indicates the cursor is set as reference
④ **Time difference**: Represents the time difference between two cursors. This will show up automatically when you have two or more time cursors placed.
⑤ **Cursors button**: Click to open cursor context menu
⑥ **Remove all cursors**: Click to remove all time cursors.
⑦ **Add time cursor**: Click to add a cursor measuring horizontal position, time.

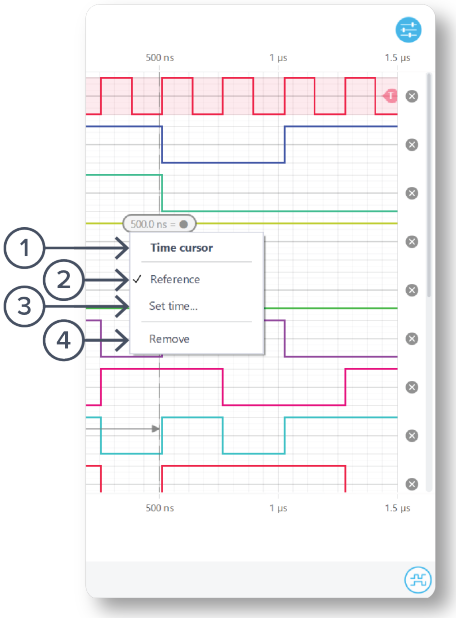


Figure 37. Time cursor context menu

① Cursor label
② Click to set the cursor to act as a horizontal reference value. When this option is selected all other cursors will display the difference between the cursor and the reference cursor's value
③ Click to position the cursor at the time entered
④ Click to remove the cursor from display

Examples

LiDAR example

Here we outline how to use the Moku Logic Analyzer to decode serial data from a LiDAR system.

In this example, we connected a LiDAR distance sensor to the input of the Moku with a transmission cable. The BNC end of the data transmission cable was plugged into Input 1 of the Moku whilst the other end is connected to the serial data pin on the range sensor. The programmable power supplies were used to power the LiDAR sensor with the desired DC voltage.

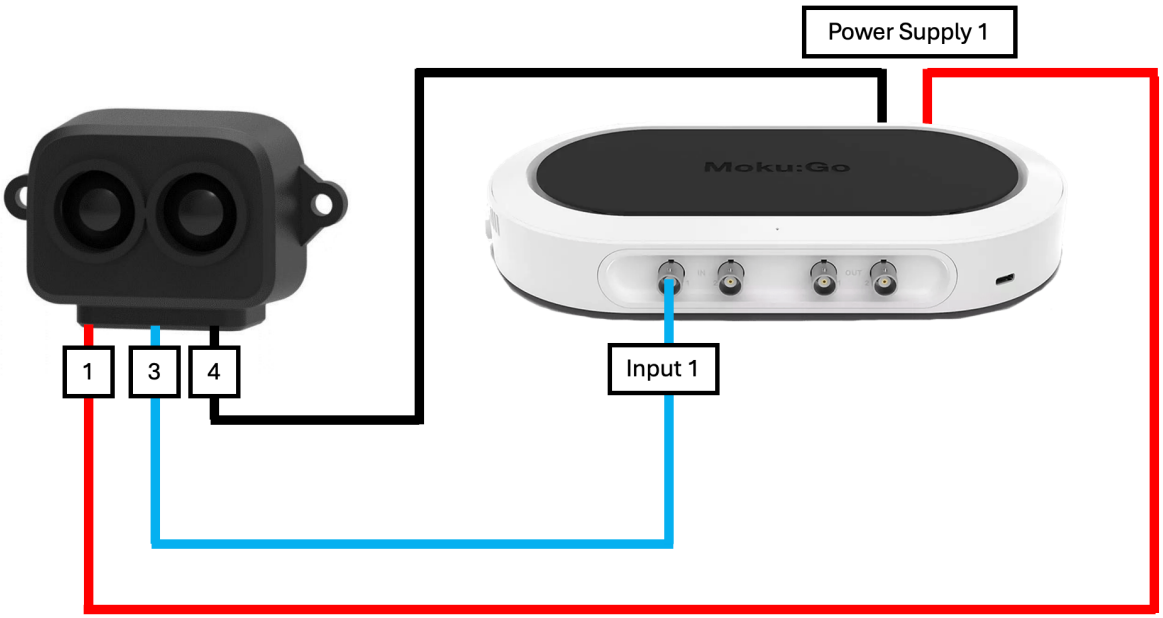


Figure 38. LiDAR example set-up

- **Step 1**: Set up the acquisition.
  - In the acquisition panel, change the source to “Analog inputs,” this will bypass the 16-bit digital I/O, for applicable hardware, and use the analog inputs as two-bit data.
  - Remove the extra input displays by clicking the “X” to the right of each plot.
- **Step 2**: Set the time span.
  - Under the timebase setting, set the time span to 1.5 ms and the offset to -300 µs. The serial pattern will appear on the Bit 0 plot
- **Step 3**: Add a protocol decoder to convert the serial data into hexadecimal format.
  - To add a decoder, click the plus sign, in the top left of the screen.
  - Hover over "Add protocol decoder" and select UART.
- **Step 4**: Configure the UART decoder
  - This will depend on the specifications of the range finder. For this example, the data width is 8 bits with 1 stop bit, there is no parity check, the default baud rate is 115200, and the default bit order is LSB first.
  - The hexadecimal numbers then appear in the UART decoder line

- **Step 5**: Interpret the data
  - Refer to the documentation of your range sensor to interpret the numerical data decoded by the Moku.
- **Step 6**: Test the range sensor.
  - Use your hand to cover the range sensor and observe the hexadecimal values of the UART decoder change accordingly.

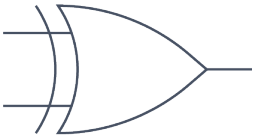
Figure 39. Screenshot of the Moku Logic Analyzer configured for a LiDAR measurement

XOR calculation with Cloud Compile

Here we outline how to use the Moku Logic Analyzer alongside Moku Cloud Compile to show how the Cloud Compile instrument interprets digital signals. In this example, we connected the Cloud Compile instrument to the Logic Analyzer in Multi-instrument Mode to output an XOR calculation. We then used the Math channel in the Logic Analyzer to compare the output of our system and validate the behaviour.

Table 3. XOR Logic Gate Results

A	B	XOR
0	0	0
1	0	1
0	1	1
1	1	0



- **Step 1: Set up Multi-instrument Mode**
  - Click Slot 1 to add the Cloud Compile instrument
  - Click Slot 2 to place the Logic Analyzer

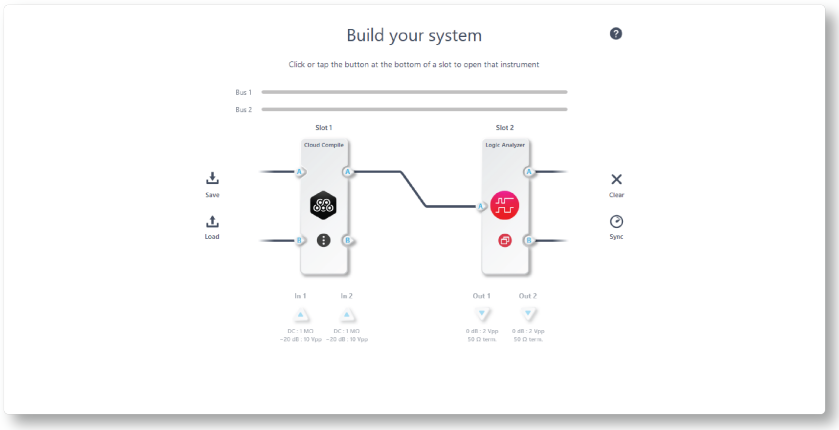


Figure 40. Multi-instrument mode set-up

- **Step 2: Write the VHDL code for outputting an XOR bitstream**
  - Open compile.liquidinstruments.com and create a project, to learn more, visit our API documentation at apis.liquidinstruments.com/mcc/
  - Add a file and name it "Top.vhd"
  - Paste in the following code:

```
architecture Behavioural of CustomWrapper is
begin
  OutputA(0) <= InputA(0);
  OutputA(1) <= InputA(1);
  OutputA(2) <= InputA(0) xor InputA(1);
end architecture;
```

  - What this code does:
    - Output A is a 16-bit output
    - Input A is a 16-bit input
    - Set the first bit of Output A to match the first bit of Input A
    - Set the second bit of Output A to match the second bit of Input A
    - Set the third bit of Output A to the result of an XOR operation between the first and second bits of Input A

- Enter the parameters for the build configuration. Enter the hardware, number of slots and firmware version of your Moku device.
- Save the project and click "Build"

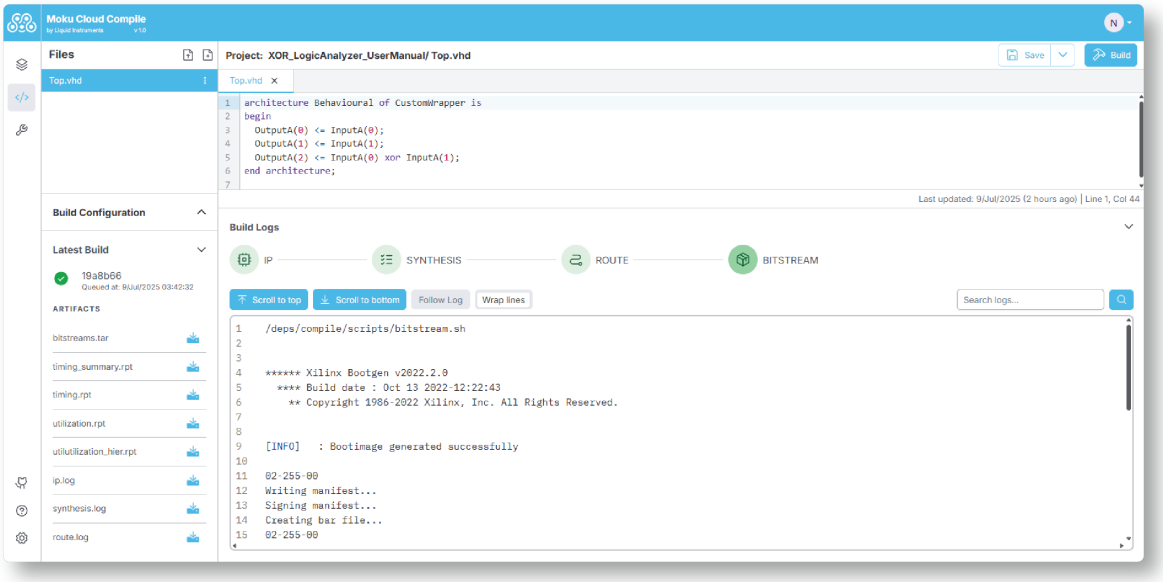


Figure 41. Building the instrument using Moku Cloud Compile

- **Step 3: Upload the compiled bitstreams**
  - After the build has completed, download the artifact, "bitstreams.tar"
  - Click the 3 dots on the Cloud Compile slot, and select "Upload bitstream"
  - Navigate to where you saved the bitstreams in the pop-up file explorer
  - Select the tar ball and click "Open"
- **Step 4: Generate a pattern to perform the XOR logic on**
  - Open the Logic Analyzer instrument
  - Open the Pattern Generator and set the number of ticks to 4
  - Set the baud rate to 1,000,000
  - Click "Edit pattern" to open the pattern editor pop-up
  - Set Bit 0 to follow the pattern 0, 1, 0, 1
  - Set Bit 1 to follow the pattern 0, 0, 1, 1
  - Toggle on the output
- **Step 5: Configure the Logic Analyzer**
  - In the acquisition panel, set the span to 8.5 µs
  - In the trigger panel, set the trigger type to "Advanced" and select "Falling edge" for Bit 0 and Bit 1
  - Remove the excess channels Bit 3 to Bit 15 by clicking the remove icon
- **Step 5: Validate the XOR calculation**
  - Click the icon to add a channel and select "Add math channel"
  - In the acquisition panel, configure the math channel
  - Click the operation, "AND", and select "XOR" from the drop down
  - The default will be for the operation to be between Bit 0 and Bit 1
  - Comparing Bit 2 and the Math channel in the display will show that they match for each combination of Bit 0 and Bit 1

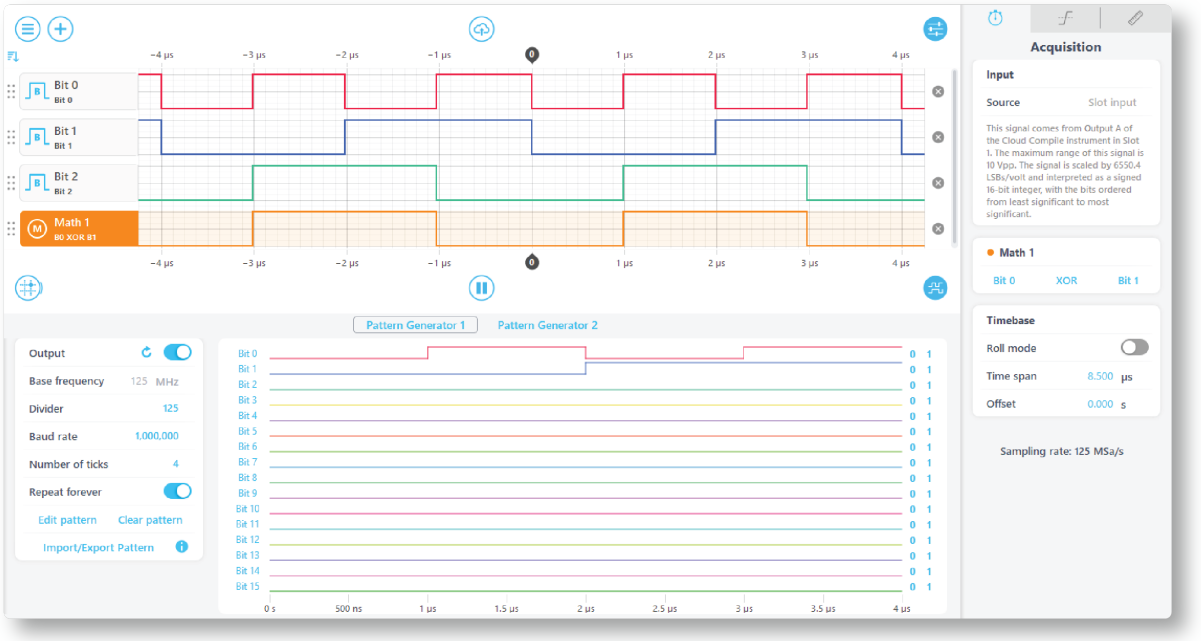


Figure 42. Screenshot of the Moku Logic Analyzer configured for an XOR calculation

Additional controls

Live data

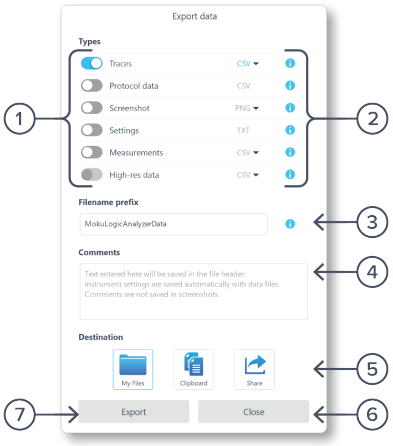


Figure 43. Live data exporting user interface and settings.

① Select the type of data to export:

- **Traces** Saves the trace data for all visible signal traces, in either a CSV or MAT-file format.
- **Protocol data** Saves the decoded protocol states and values in a CSV format.
- **Screenshots** Saves 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 MAT-file format.
- **High-res data** Saves the full memory depth of all visible channels 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 “MokuLogicAnalyzerData” 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: “MokuLogicAnalyzerData_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 chosen, 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.

Main menu

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

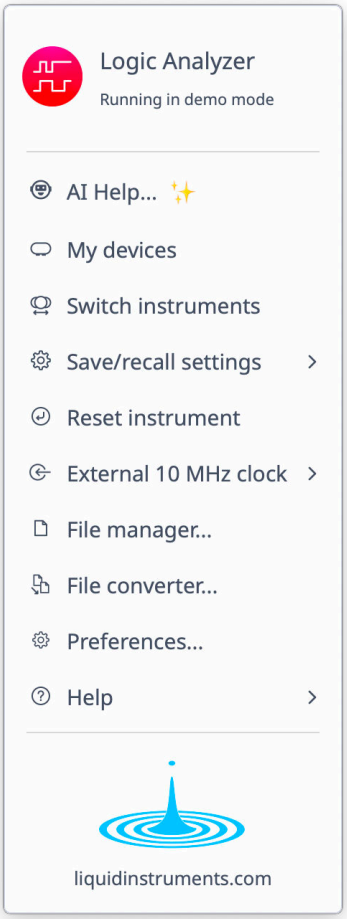


Figure 44. Main menu options for the Logic Analyzer.

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.

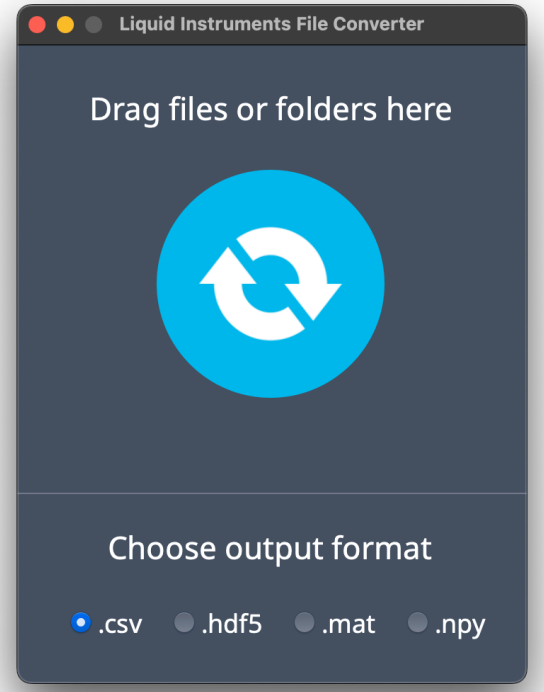


Figure 45. File Converter user interface.

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.

File manager

The File manager allows you to download the saved data from your Moku device to the local computer, with optional file format conversion. Once a file is transferred to the local computer, an icon appears next to the file.

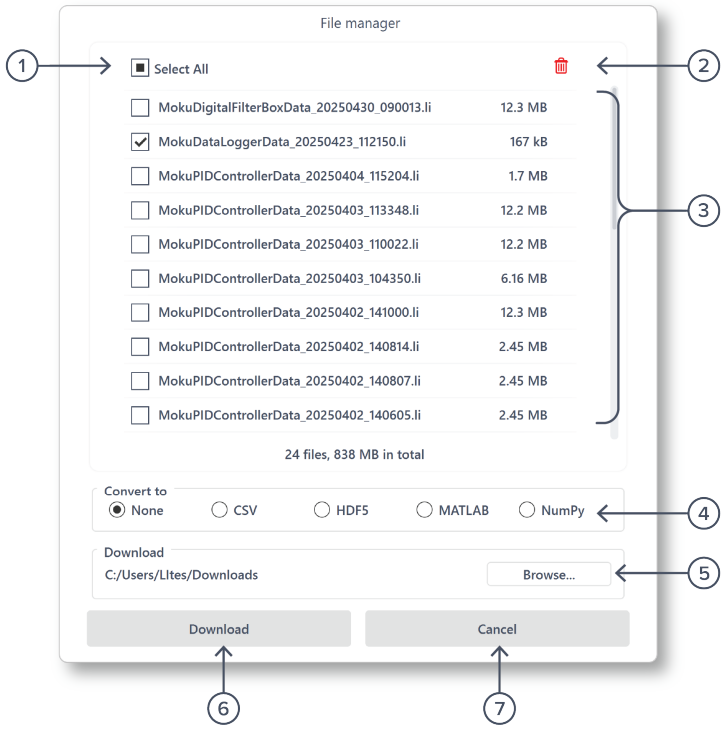


Figure 46. File exporting User Interface and settings.

To save logged data:

① **Select all** files logged to the device's memory, to download or convert.
② **Delete** all 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.

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.

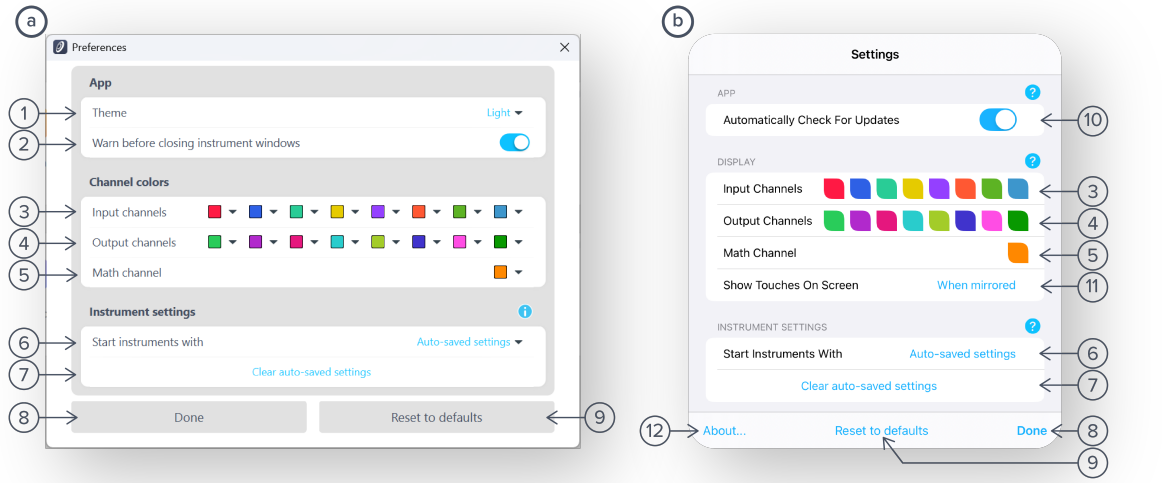


Figure 47. Preferences and settings for the Desktop (a) and for the iPad (b) App.

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.

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.

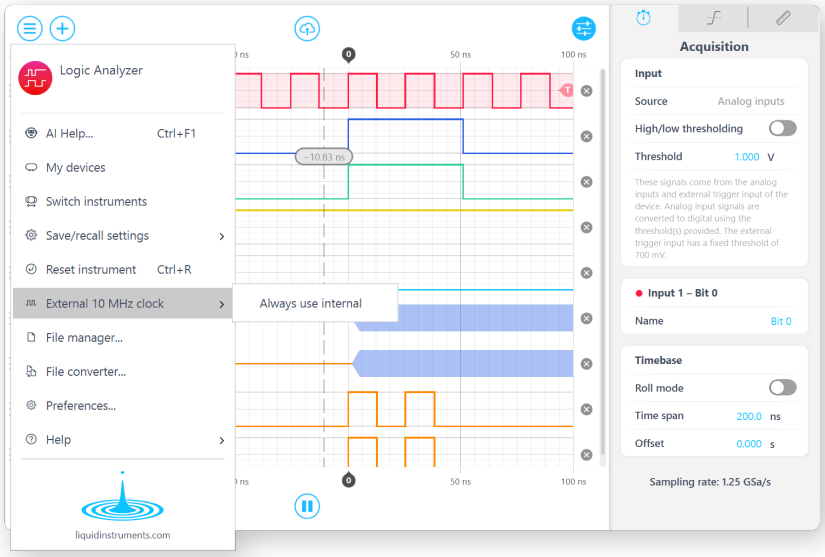


Figure 48. Moku main menu with "Always use internal" reference disabled and using an external reference.

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 49, 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.

Figure 49. Moku clock blending configuration dialog with an external 10 MHz frequency reference and GNSS 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).