Moku:Lab
MATLAB API Migration Guide

Upgrading Moku:Lab to software version 3.0 unlocks a host of new features. When updating, API users must take extra steps to migrate their scripts to the new Moku API package. This migration guide outlines API changes, new features available in the version 3.0 update, and any backward compatibility limitations.

Table of Contents

Overview

Moku:Lab software version 3.0 is a major update that brings new firmware, user interface, and APIs to Moku:Lab hardware. The update brings Moku:Lab in line with Moku:Pro and Moku:Go, making it easy to share scripts across all Moku platforms. The update unlocks a host of new features to many of the existing instruments. It also adds two new features: Multi-instrument Mode and Moku Cloud Compile. There are some subtle behavioral differences as well, outlined in the Backward compatibility section.

This is a major update that affects the API architecture, and therefore the new MATLAB API v3.0 package will not be backward compatible with existing MATLAB scripts. API users will need to port their scripts to the new Moku API package if they upgrade their Moku:Lab to version 3.0. API users with significant custom software development should carefully consider the level of effort required to port their existing code. Moku:Lab 1.9 is not recommended for new deployments and all customers are encouraged to upgrade. If issues arise after upgrading, users will have the option to downgrade to software version 1.9.

This migration guide outlines advantages of updating and potential complications to Moku:Lab version 3.0. It also outlines the process to upgrade the MATLAB API and how to downgrade your Moku:Lab if necessary.

Version 3.0 new features

New features

Software version 3.0 brings Multi-Instrument Mode and Moku Cloud Compile to Moku:Lab for the first time, as well as many performance and usability upgrades across the suite of instruments.

Multi-instrument Mode

Multi-instrument Mode on Moku:Lab allows users to deploy two instruments simultaneously to create a custom test station. Each instrument has full access to the analog inputs and outputs along with interconnections between instrument slots. The interconnections between instruments support high-speed, low-latency, real-time digital communication up to 2 Gb/s, so instruments can run independently or be connected to build advanced signal processing pipelines. Instruments can be dynamically swapped in and out without interrupting the other instrument. Advanced users can also deploy their own custom algorithms in Multi-instrument Mode using Moku Cloud Compile.

Moku Cloud Compile

Moku Cloud Compile allows you to deploy custom DSP directly onto the Moku:Lab FPGA in Multi-instrument Mode. Write code using a web browser and compile it in the cloud; Moku Cloud Compile deploys the bitstream to one or more target Moku devices.

Oscilloscope

- Deep memory mode: save up to 4M samples per channel at full sampling rate (500 MSa/s)

Spectrum Analyzer

- Improved noise floor
- Logarithmic Vrms and Vpp scale
- Five new window functions (Bartlett, Hamming, Nuttall, Gaussian, Kaiser)

Phasemeter

- Frequency offset, phase, and amplitude can now be output as analog voltage signals
- Users can now add DC offset to output signals
- The phase-locked sine wave output can now be frequency multiplied up to 250x or divided down to 0.125x
- Improved bandwidth range (1 Hz to 100 kHz)
- Advanced phase wrapping and auto-reset functions

Waveform Generator

- Noise output
- Pulse width modulation (PWM)

Lock-In Amplifier

- Improved performance of low-frequency PLL locking
- The minimum PLL frequency has been decreased to 10 Hz
- The internal PLL signal can now be frequency multiplied up to 250x or divided down to 0.125x for use in demodulation
- 6-digit precision for phase values

Frequency Response Analyzer

- Increased maximum frequency from 120 MHz to 200 MHz
- Increase maximum sweep points from 512 to 8192
- New Dynamic Amplitude feature optimizes output signal automatically for best measurement dynamic range
- New In/In1 measurement mode
- Input saturation warnings
- The math channel now supports arbitrary complex-valued equations involving the channel signals, enabling new types of complex transfer function measurements
- Input signals can now be measured in dBVpp and dBVrms in addition to dBm
- The progress of the sweep is now displayed on the graph
- The frequency axis can now be locked to prevent accidental changes during a long sweep

Laser Lock Box

- Improved block diagram shows scan and modulation signal paths
- New locking stages feature allows customizing lock procedure
- Improved performance of low-frequency PLL locking
- 6-digit precision for phase values
- Improved performance of low-frequency PLL locking
- The minimum PLL frequency has been decreased to 10 Hz

- The PLL signal can now be frequency multiplied up to 250x or divided down to 0.125x for use in demodulation

Other

- Added support for the sinc function to the equation editor which can be used to generate custom waveforms in the Arbitrary Waveform Generator
- Convert binary LI files to CSV, MATLAB, or NumPy formats when downloading from the device

Upgraded API Support

The new Moku MATLAB API v3.0 package provides enhanced functionality and stability. It will receive regular updates to improve performance and introduce new features.

Backward compatibility limitations

API

The new Moku MATLAB API v3.0 package is not backwards compatible with the previous Moku:Lab MATLAB v1.9 package. The MATLAB scripting arguments and return values are wholly different. If you have extensive custom software development utilizing the Moku:Lab MATLAB, consider the impact of migrating all your software to be compatible with the new API.

While the Moku:Lab MATLAB package will no longer receive updates, Liquid Instruments will still continue to provide support for users who are unable to migrate to the new API package.

Find detailed examples for each instrument in the new Moku MATLAB API v3.0 package to serve as a base line for converting prior MATLAB development to the new API package.

Regressions

RAM disk for data logging

Version 1.9 had a 512 MB filesystem in the device's RAM, which could be used to log data at high sampling rates. In version 3.0, logging to RAM is no longer available. To enable data logging, an SD card is required. Accordingly, the maximum acquisition speed changes as well. Version 1.9 supported up to 1 MSa/s, whereas version 3.0 supports up to 250 kSa/s at 1 channel and 125 kSa/s at 2 channels. Even at lower speeds and with an SD card, workflows which included saving multiple high-speed logs to RAM and then later copying them to the SD card or the client will no longer be supported.

Data logging to CSV

Version 1.9 had the ability to save data directly to a CSV file while logging. This feature is not directly available on version 3.0. Users whose workflow included saving CSV files directly to an SD card or the client will now need to first convert the binary file to CSV, either using the client app or by installing the standalone Liquid Instruments File Converter onto the computer that they use for data processing.

Non-backwards-compatible changes

Data scaling in LIA

In version 1.9, we implemented data scaling such that multiplying two 0.1 V DC signals resulted in a 0.02 V DC output. In version 3.0, we changed this such that the result was 0.01 V DC, which is more in line with customers' intuitive expectations.

Waveform Generator output must be enabled to use as modulation source/trigger

In version 1.9, a different channel's waveform could be used as a modulation or trigger source in the Waveform Generator, even if that channel's output was disabled. This was removed in version 3.0. Users who want to do cross-modulation without needing to unplug the outputs of their device would need to adjust their workflow.

Moku MATLAB API

The Moku MATLAB API v3.0 package is intended to provide MATLAB developers the resources needed to control any Moku device and, ultimately, the ability to incorporate these controls into larger end-user applications.

The new Moku MATLAB API v3.0 package provides the following:

- Fully functional example MATLAB scripts for each instrument.
- All MATLAB scripts are provided with comments, which are easy to understand and can serve as an end user's starting point for customization and adaptation.
- A set of functions providing full control over the Moku device.

Currently supported instruments

1. Arbitrary Waveform Generator
2. Data Logger
3. Digital Filter Box
4. FIR Filter Builder
5. Frequency Response Analyzer
6. Laser Lock Box
7. Lock-in Amplifier
8. Oscilloscope
9. Phasemeter
10. PID Controller
11. Spectrum Analyzer
12. Waveform Generator
13. Multi-instrument Mode
14. Moku Cloud Compile

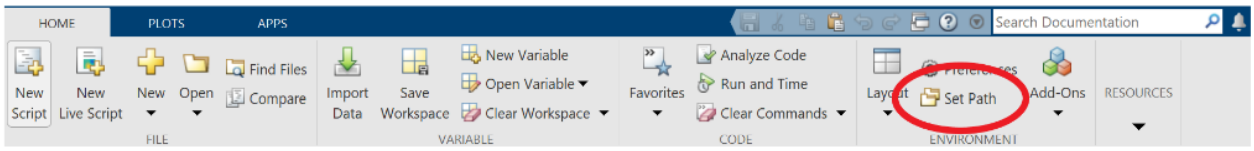
Installation

Requirements

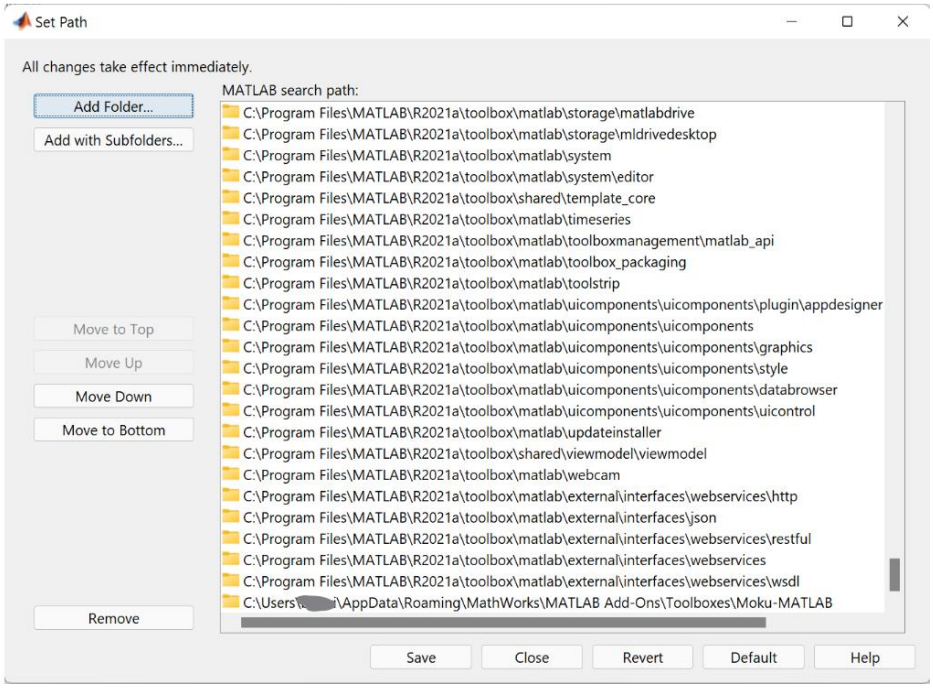
- MATLAB version 2015 or later

If you already have a previous version of the Moku MATLAB API installed, please uninstall it before proceeding. You can uninstall the package from the Add-on Manager.

1. Open the Add-on Manager through the Home > Environment tab.
2. Search for Moku in the Add-on Manager and click 'Add'. The toolbox will show up as Moku-MATLAB.
3. Alternatively, you can download the toolbox directly from the Liquid Instruments website at https://www.liquidinstruments.com/products/apis/matlab-api/. You will have to set the search path manually if you do this.
4. Check that the correct path has been added to the toolbox by selecting 'Set Path' from the Home > Environment tab.



5. Ensure there is an entry pointing to the toolbox installation location. A typical path might be C:\Users\<username>\AppData\Roaming\Mathworks\MATLAB Add-Ons\Toolboxes\Moku-MATLAB.



6. Download the instrument data files by typing ‘moku_download(###)’ into the MATLAB Command Window. The ### should be replaced with your current firmware version. You can find your current firmware version through the Moku: desktop app by right clicking on your Moku and hovering ‘Device info’, or in the iPad app by long pressing on your Moku.
7. Confirm your toolbox is set up correctly by typing ‘help Moku’ into the MATLAB Command Window. If this command succeeds, then the toolbox has been successfully installed.

Moku API changes

The new Moku MATLAB API architecture is sufficiently different from its predecessor and therefore not backwards compatible with existing API scripts. The following simplified Oscilloscope example shows the differences between the legacy and new API packages and serves as a road map for porting existing code.

Oscilloscope example

Moku MATLAB 1.9

```
%% Plotting Oscilloscope Example
%[...]

%% Connect to your Moku
% Connect to your Moku by its IP address.
i = MokuOscilloscope('192.168.###.###');

try

    %% Configure the instrument

    % Configure the frontend
    % Channel 1 DC coupled, 10Vpp range
    i.set_frontend(1, '1MOhm', 'DC', '10Vpp');
    % Channel 2 DC coupled, 50Vpp range
    i.set_frontend(2, '1MOhm', 'DC', '10Vpp');

    % Configure the trigger conditions
    % Trigger on input Channel 1, rising edge, 0V
    i.set_trigger('type',"Edge", 'source',"Input1", 'level',0);

    % View +- 1 ms i.e. trigger in the centre
    i.set_timebase(-1e-3,1e-3);

    % Generate a sine wave on Output 1
    % 0.5Vpp, 10kHz, 0V offset
    i.gen_sinewave(1, 'amplitude',0.5, 'frequency', 10e3, 'offset', 0);

    % Generate a square wave on Output 2
    % 1Vpp, 20kHz, 0V offset, 50% duty cycle
    i.gen_squarewave(2, 'amplitude',1, 'frequency',20e3, 'duty', 50);


    % Set the data source of Channel 1 to be Input 1
    i.set_source(1,'in1');
    % Set the data source of Channel 2 to Input 2
    i.set_source(2,'in2');

    %% Retrieve data
    % Get one frame of dual-channel voltage data
    data = i.get_data();

catch ME
    % End the current connection session with your Moku
    i.relinquish_ownership();
    rethrow(ME)
end

i.relinquish_ownership();
```

Moku MATLAB 3.0

```
%% Plotting Oscilloscope Example
%[...]

%% Connect to your Moku
% Connect to your Moku by its IP address.
i = MokuOscilloscope('192.168.###.###');

try

    %% Configure the instrument

    % Configure the frontend
    % Channel 1 DC coupled, 10Vpp range
    i.set_frontend(1, '1MOhm', 'DC', '10Vpp');
    % Channel 2 DC coupled, 50Vpp range
    i.set_frontend(2, '1MOhm', 'DC', '10Vpp');

    % Configure the trigger conditions
    % Trigger on input Channel 1, rising edge, 0V
    i.set_trigger('type',"Edge", 'source',"Input1", 'level',0);

    % View +- 1 ms i.e. trigger in the centre
    i.set_timebase(-1e-3,1e-3);

    % Generate a sine wave on Output 1
    % 0.5Vpp, 10kHz, 0V offset
    i.generate_waveform(1, 'Sine', 'amplitude',0.5, 'frequency', 10e3);

    % Generate a square wave on Output 2
    % 1Vpp, 20kHz, 0V offset, 50% duty cycle
    i.generate_waveform(2, 'Square', 'amplitude',1, 'frequency',20e3, 'duty', 50);

    % Set the data source of Channel 1 to be Input 1
    i.set_source(1,'Input1');
    % Set the data source of Channel 2 to Input 2
    i.set_source(2,'Input2');

    %% Retrieve data
    % Get one frame of dual-channel voltage data
    data = i.get_data();

catch ME
    % End the current connection session with your Moku
    i.relinquish_ownership();
    rethrow(ME)
end

i.relinquish_ownership();
```

Table 1: Oscilloscope API comparison example

Sequence steps

1. Import the Moku MATLAB API 3.0 packages.
2. Claim the Moku ownership and upload Oscilloscope bitstream to Moku.
3. Set time base and set the left- and right-hand span for the time axis.
4. Get data, acquire a single frame of the data from the Oscilloscope.
5. End client session by relinquishing the Moku ownership.

The sequence described above is a simplified example to illustrate the differences between the legacy and new API packages. Aside from beginning a client session, uploading an instrument bitstream to Moku, and ending the client session, an end user can exercise any number of functions in various order to meet the needs of their application.

Differences

Here, we look at the differences between the two APIs for each step in the sequence.

1. Claim Moku ownership and upload the Oscilloscope bitstream to the device. Compared with Moku MATLAB 1.9, the new API has completely different functions:

	Moku MATLAB 1.9		Moku MATLAB 3.0
Function	*get_by_name()*	*deploy_or_connect()*	*Oscilloscope()*
Allowed fields and values	name: string timeout: float force: bool	instrument: the class of the instrument wish to deploy set_default: bool use_external: bool	ip: string serial: string force_connect: bool ignore_busy: bool persist_state: bool connect_timeout: float read_timeout: float

1. Set time base. The function is the same, but the allowed arguments are slightly different:

	Moku MATLAB 1.9	Moku MATLAB 3.0
Function	*set_timebase()*	*set_timebase()*
Allowed fields and values	t1: float t2: float	t1: float t2: float strict: bool

2. Get data. The functions and the allowed arguments are the same, but the returned data type and length are different:

	Moku MATLAB 1.9	Moku MATLAB 3.0
Function	*get_data()*	*get_data()*
Allowed fields and values	timeout: float wait: bool	timeout: float wait_reacquire: bool
Return length	16383 points per frame	1024 points per frame

3. Release the Moku ownership:

	Moku MATLAB 1.9	Moku API v3.0
Function	*close()*	*relinquish_ownership()*

Oscilloscope functions list

Moku MATLAB 1.9	Moku MATLAB 3.0
set_source()	*set_sources()*
set_trigger()	*set_trigger()*
get_data()	*get_data()*
set_frontend()	*set_frontend()*
set_defaults()	*set_defaults()*
set_timebase()	*set_timebase()*
set_xmode()	*disable_input()*
set_precision_mode()	*enable_rollmode()*
sync_phase()	*set_acquisition_mode()*
get_frontend()	*sync_output_phase()*
get_samplerate()	*get_frontend()*
get_realtime_data()	*get_samplerate()*
gen_rampwave()	*save_high_res_buffer()*
gen_sinewave()	*generate_waveform()*
gen_squarewave()	*get_acquisition_mode()*
gen_off()	*get_sources()*
set_samplerate()	*get_timebase()*
set_framerate()	*get_output_load()*
	get_interpolation()
	set_output_load()
	set_hysteresis()
	set_interpolation()
	set_input_attenuation()
	set_source()
	osc_measurement()
	summary()

The Moku MATLAB API is based upon Moku API. For full Moku API documentation, refer to the Moku API Reference found here https://apis.liquidinstruments.com/reference/.

Additional details for getting started with Moku MATLAB API can be found at https://apis.liquidinstruments.com/starting-matlab.html

Downgrade process

If the upgrade to version 3.0 has proven to limit, or otherwise adversely affect, something critical to your application, you can downgrade to the previous version 1.9. This can be done through a web browser.

Steps

1. Contact Liquid Instruments and obtain the file for firmware version 1.9.
2. Type your Moku:Lab IP address into a web browser (see screen shot).
3. Under Update Firmware, browse and select the firmware file provided by Liquid Instruments.
4. Select Upload & Update. The update process can take more than 10 minutes to complete.

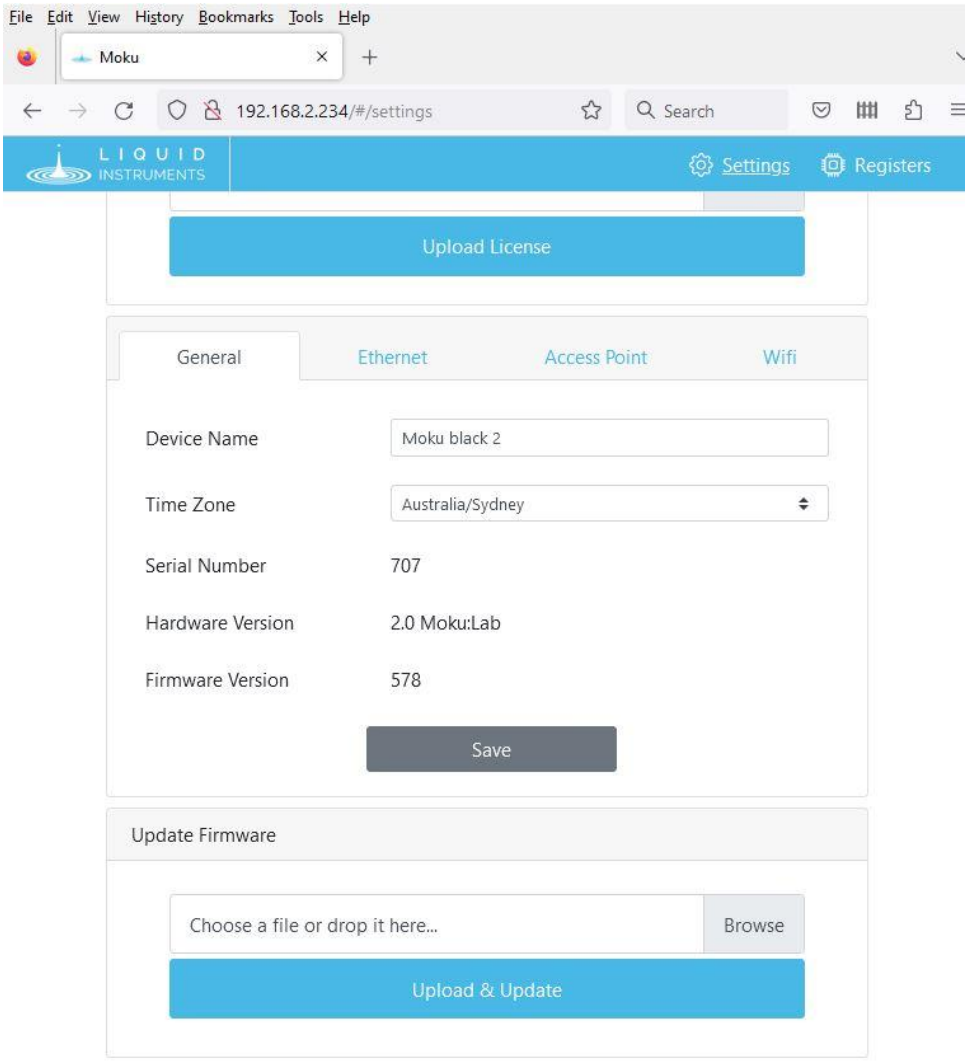


Figure 11: Downgrade procedure

