UHFQA
User Manual

600 MHz Quantum Analyzer

Zurich
Instruments

UHFQA User Manual

Zurich Instruments AG

Revision 24.04

Table of Contents

Declaration of Conformity
1. Change Log 2
2. Getting Started 5
2. 1. Quick Start Guide 5
2. 2. Inspect the Package Contents 6
2. 3. Handling and Safety Instructions 7
2. 4. Software Installation 9
2. 5. Connecting to the Instrument 16
2. 6. Software Update 32
2. 7. Troubleshooting 33
2. 8. UHFQA Firmware Upgrade Guide 37
3. Functional Overview 40
3. 1. Features 40
3. 2. Front Panel Tour 41
3. 3. Back Panel Tour 42
3. 4. Ordering Guide 43
3. 5. DIOLink cable set 43
4. Tutorials 45
4. 1. Arbitrary Waveform Generator 45
4. 2. Generate and Acquire a Test Signal 56
4. 3. Dual-phase Demodulation 58
4. 4. Weighted Integration and Result Logging 63
5. Functional Description LabOne User Interface 68
5. 1. User Interface Overview 68
5. 3. Architecture and Signalling 77
5. 4. Quantum Analyzer Setup Tab 78
5. 5. Quantum Analyzer Input Tab 81
5. 6. Quantum Analyzer Result Tab 83
5. 7. Scope Tab 86
5. 8. Auxiliary Tab 92
5. 9. Inputs/Outputs Tab 94
5. 10. DIO Tab 95
5. 11. Config Tab 98
5. 12. Device Tab 101
5. 13. File Manager Tab 105
5. 14. AWG Tab 106
5. 15. Multi Device Sync Tab 147
5. 16. ZI Labs Tab 149
5. 17. Upgrade Tab 149

Table of Contents

6. Specifications 150
6. 1. General Specifications 150
6. 2. Analog Interface Specifications 151
6. 3. Digital Interface Specifications 155
6. 4. Performance Diagrams 160
6. 5. Clock 10 MHz 162
6. 6. Device Self Calibration Procedure 162
7. Device Node Tree 164
7. 1. Introduction 164
7. 2. Reference Node Documentation 167

CE Declaration of Conformity

The manufacturer

Zurich Instruments
Technoparkstrasse 1
8005 Zurich
Switzerland

declares that the product

UHFQA 600 MHz Quantum Analyzer

is in conformity with the provisions of the relevant Directives and Regulations of the Council of the European Union:

Directive / Regulation	Conformity proven by compliance with the standards
2014/30/EU (Electromagnetic compatibility [EMC])	EN 61326-1:2013, EN 55011:2016, EN 55011:2016/A1:2017, EN 55011:2016/A11:2020 (Group 1, Class A and B equipment)
2014/35/EU (Low voltage equipment [LVD])	EN 61010-1:2010, EN 61010-1:2010/A1:2019, EN 61010-1:2010/A1:2019/AC:2019-04
2011/65/EU, as amended by 2015/863 and 2017/2102 (Restriction of the use of certain hazardous substances [RoHS])	EN IEC 63000:2018
(EC) 1907/2006 (Registration, Evaluation, Authorisation, and Restrictions of Chemicals [REACH])	-

Zurich, October 20th, 2022

Flavio Heer, CTO

UKCA Declaration of Conformity

The manufacturer

Zurich Instruments
Technoparkstrasse 1
8005 Zurich
Switzerland

declares that the product

UHFQA 600 MHz Quantum Analyzer

is in conformity with the provisions of the relevant UK Statutory Instruments:

Statutory Instruments	Conformity proven by compliance with the standards
S.I. 2016/1091 (Electromagnetic Compatibility Regulations)	EN 61326-1:2013, EN 55011:2016, EN 55011:2016/A1:2017, EN 55011:2016/A11:2020 (Group 1, Class A and B equipment)
S.I. 2016/1101 (Electrical Equipment (Safety) Regulations)	EN 61010-1:2010, EN 61010-1:2010/A1:2019, EN 61010-1:2010/A1:2019/AC:2019-04
S.I. 2012/3032 (Restriction of the Use of Certain Hazardous Substances Regulations)	EN IEC 63000:2018

Zurich, October 20th, 2022

Flavio Heer, CTO

1. Change Log

1.1. Release 24.04

Release date: 30-Apr-2024

- Improved the optimization pass of the AWG compiler that removes unused registers.

1.2. Release 24.01

Release date: 31-Jan-2024

- SeqC command `startQA` now has constant latency, no matter its arguments.
- SeqC command `setDIO` now has constant latency, no matter its arguments.

1.3. Release 23.10

Release date: 31-Oct-2023

- Device: Made it possible to connect and use the device from a different network.

1.4. Release 23.06

Release date: 30-Jun-2023

- FW: Fixed a sporadic firmware issue causing "[error] Exception: TCP receive timeout" in the data server log followed by a disconnect during vector data transfer (e.g. QA results) from the device to the data server. (patch release)
- AWG: Fixed a bug, which caused `ksn_dsp_awg_vec_read - ERROR: Can't read write-only ELF Data node` messages in the firmware log.

1.5. Release 23.02

Release date: 28-Feb-2023

- AWG: Fixed a bug in the AWG waveform playback for the auxiliary outputs using the SeqC command **playAuxWave**, where the voltage of the last sample was not held constant as specified.

1.6. Release 22.08

Release date: 31-Aug-2022

- AWG: The ELF data node `/DEV.../AWGS/0/ELF/DATA` accepts raw data as 8-, 16-, and 64-bit integer vectors in addition to 32-bit words.

1.7. Release 22.02

Release date: 28-Feb-2022

- QCCS: New Real-Time Logger that logs history of received ZSync and DIO messages facilitating setup of feedback data communication

1.8. Release 21.08

Release date: 30-Aug-2021

- AWG: Added support for variable arguments in the playZero() AWG instruction to improve performance of run-time delay sweeps
- AWG: Increased update rate of `/DEV.../AWGS/0/ENABLE`` node such that AWG state may be probed within 10 ms (previously >100 ms)
- LabOne: Reduced CPU load caused by the LabOne discovery service in networks with many Zurich Instruments devices

1.9. Release 21.02

Release date: 28-Feb-2021

- LabOne API: Added online Programming Manual and Documentation
- Added option to load factory defaults in Device tab
- More than 60 ns improvement in latency of real-time qubit readout by new bypass paths of deskew and rotation units
- AWG: New sequencer instruction startQA() to trigger integration and monitor simultaneously
- AWG: New instruction resetOscPhase() to initialize oscillator phase independently of AWG Integration Trigger
- QA Setup: Support of dual-phase demodulation on a single signal input by new options of Signal Input Mapping setting
- Oscillator signals can now be enabled independently of the AWG in the In / Out tab

1.10. Release 20.07

Release date: 20-Aug-2020

- NEW option UHF-DIG Digitizer: Scope enhancement with continuous scope streaming, increased memory, dual-channel operation, and segmented recording
- LabOne: Trends plots to track readings from the Math sub-tab over time
- LabOne: Introduced device error reporting
- QCCS: New DIO mode QA Results QCCS and new sequencer instruction setReadoutRegisterAddress for readout result feedback operation in combination with PQSC and HDAWG
- QCCS: New sequencer instruction waitZSyncTrigger for triggering over DIO link in combination with PQSC and HDAWG
- LabOne: Device Information report in Device tab
- LabOne: Improved colormaps available for 2D plots

1.11. Release 20.01

Release date: 28-Feb-2020

- AWG: added SeqC instructions startQAResult() and startQAMonitor()
- Result Logger: sequential averaging mode
- External triggering of input monitor and integration
- QA Result: added complex data plot with persistence, rotation, shift and scaling
- LabOne API: new quantum analyzer API module
- LabOne: histogram data can be saved in CSV format
- LabOne: improved importing of saved SVG figures to main vector graphics editors
- AWG: improved compilation stability

1.12. Release 19.05

Release date: 12-Aug-2019

- QAS: it's now possible to bypass of the crosstalk suppression matrix to reduce the latency
- QAS: the oscillator can now be directly output on Signal Outputs on Signal Output 1 (sine) and 2 (cosine) using controls in the Signal Outputs tab. Relative nodes are changed

- QAS: discriminated qubit state can now be output on the Trigger Out connectors of the instrument
- QAS: crosstalk suppression matrix can be edited in the LabOne UI
- QAS: the deskew parameters can now be specified by the gain and phase imbalance of the mixer in the LabOne UI
- AWG: Added getQAResult and waitQAResultTrigger instructions to read the result of the last qubit state discrimination
- AWG: improved compilation speed and stability
- AWG: the waveform viewer now supports waveforms up to 10 MSa length
- AWG: the sequencer program memory has been restricted to the cache memory
- LabOne: macOS support
- LabOne: plots can be saved in PNG or JPEG format
- LabOne: add context menus for plots, input fields and device connection dialog
- LabOne API: waveform update using vectorWrite replaced by faster and more robust method setVector. Waveforms are now ordered according to the sequence in which they are defined in the sequence program, rather than alphabetically.
- LabOne API: waveform update now uses integer format. It's advised to use the helper functions convert_awg_waveform and parse_awg_waveform to convert to the new format.
- Specifications: added performance diagram for Signal Output phase noise

1.13. Release 18.12

Release date: 20-Dec-2018

First edition of the UHFQA User Manual

2. Getting Started

This first chapter guides you through the initial set-up of your UHF Instrument in order to make your first measurements. This chapter comprises:

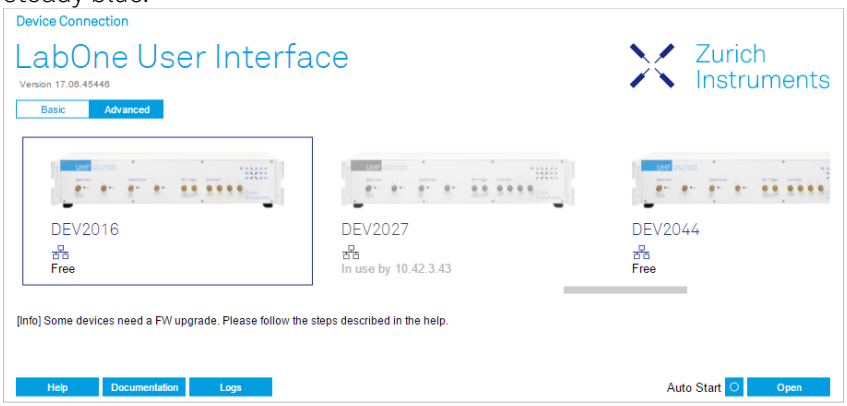
- Quick Start Guide for the impatient
- Inspecting the package content and accessories
- List of essential handling and safety instructions
- Installing LabOne, the UHF Instrument software, on your host computer
- Powering-on the device and connecting the device to a host computer
- Performing basic operation checks on the instrument

This chapter is delivered as a hard copy with the instrument upon delivery. It is also the first part of the UHF User Manual.

2.1. Quick Start Guide

This page addresses all the people who impatiently are awaiting their new gem to arrive and want to see it up and running quickly. Please proceed with the following steps:

1. Check the package content. Besides the Instrument there should be a country-specific power cable, a USB cable, an Ethernet cable and a hard copy of the user manual Getting Started.
2. Check the Handling and Safety Instructions in Handling and Safety Instructions.
3. Download and install the latest LabOne software from the Zurich Instruments Download Center. Choose the download file that fits your computer (e.g. Windows with 64-bit addressing). For more detailed information see Software Installation.
4. Connect the Instrument to the power line, turn it on and then connect in with the measurement PC by using the USB cable. The necessary drivers will now be installed automatically. The front panel LED will blink orange at this stage.
5. Start the LabOne User Interface from the Windows Start Menu. The default web browser will open and display your instrument in a start screen as shown below. Use Chrome, Edge, Firefox, or Opera for best user experience. The front panel LED turns from blinking orange to a steady blue.



6. Click the **Open** button on the lower right of the start screen. The default configuration is loaded and the first measurements can be taken. In cases the device could not be found or the user interface does not start at all, please be referred to LabOne Software Architecture.
7. The UHF User Manual is included in the LabOne installation. Under Windows 10 it can be found in[1] `Start Menu → Zurich Instruments → User Manual UHF`.

If any problems are encountered whilst setting up the instrument and software please see the Troubleshooting at the end of this chapter.

Once the Instrument is up and running we recommend to go through some of the tutorials given in Tutorials. Moreover, Functional Description LabOne User Interface provides a general introduction to the various tools and settings tabs with tables in each section providing a detailed description of every UI element as well. For specific application know-how the blog section of the Zurich Instruments website will serve as a valuable resource that is constantly updated and expanded.

Note

It's recommended to enable graphical hardware acceleration to ensure a high responsiveness of the web browser user interface. On most computers, hardware acceleration can be enabled by one of the following methods.

- Control panel: Control Panel\Appearance and Personalization\Display\Screen Resolution. Advanced settings. Trouble shoot. Change settings.
- NVIDIA control panel: select graphic processor. Apply.

Some computers have two graphic chip sets installed, an Intel and a NVIDIA chip set. Activating the NVIDIA along with the acceleration is recommended to achieve best possible performance. The only drawback changing these settings is a slightly increased power consumption.

---

1. Under Windows 7 and 8, the User Manual can be found in `Start Menu` → `All programs / All apps` → `Zurich Instruments` → `User Manual UHF` ↩

2.2. Inspect the Package Contents

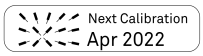
If the shipping container appears to be damaged, keep the container until you have inspected the contents of the shipment and have performed basic functional tests.

Please verify:

- You have received 1 Zurich Instruments UHFLI or UHFAWG Instrument
- You have received 1 power cord with a power plug suited to your country
- You have received 1 USB cable and/or 1 LAN cable (category 5/6 required)
- A printed version of the "Getting Started" section
- The "Next Calibration" sticker on the back panel of the Instrument indicates approximately 2 years ahead in time. Zurich Instruments recommends calibration intervals of 2 years
- The MAC address and serial number of the instrument are displayed on a sticker on the back panel

Table 2.1: Package contents for the UHFLI or UHFAWG

	
	the power cord (e.g. EU norm)
	USB cable
	Power inlet with power switch and fuse holder

	LAN / Ethernet cable (category 5/6 required)
	the "Next Calibration" sticker on the back panel of the instrument
	S/N and MAC address sticker on the back panel of the instrument

The UHF Instrument is equipped with a multi-mains switched power supply, and therefore can be connected to most power systems in the world. The fuse holder is integrated with the power inlet, and can be extracted by grabbing the holder with two finger nails (or small screwdrivers) at the top and at the bottom at the same time. A spare fuse is contained in the fuse holder. The fuse description is mentioned in the specification chapter.

Carefully inspect your Instrument. If there is mechanical damage or the amplifier does not pass the basic tests, then you should immediately notify the Zurich Instruments support team at support@zhinst.com.

2.3. Handling and Safety Instructions

The UHFLI Instrument is a sensitive piece of electronic equipment, and under no circumstances should its casing be opened, as there are high-voltage parts inside which may be harmful to human beings. There are no serviceable parts inside the instrument. Do not install substitute parts or perform any unauthorized modification to the product. Opening the instrument immediately voids the warranty provided by Zurich Instruments.

Do not use this product in any manner not specified by the manufacturer. The protective features of this product may be affected if it is used in a way not specified in the operating instructions.

The following general safety instructions must be observed during all phases of operation, service, and handling of the instrument. The disregard of these precautions and all specific warnings elsewhere in this manual may negatively affect the operation of the equipment and its lifetime.

Zurich Instruments assumes no liability for the user's failure to observe and comply with the instructions in this user manual.

Table 2.2: Safety Instructions

Ground the instrument	The instrument chassis must be correctly connected to earth ground by means of the supplied power cord. The ground pin of the power cord set plug must be firmly connected to the electrical ground (safety ground) terminal at the mains power outlet. Interruption of the protective earth conductor or disconnection of the protective earth terminal will cause a potential shock hazard that could result in personal injury and potential damage to the instrument.
Electromagnetic environment	This equipment has been certified to conform with industrial electromagnetic environment as defined in EN 61326-1. Emissions, that exceed the levels required by the document referenced above, can occur when connected to a test object.
Measurement category	This equipment is of measurement category I (CAT I). Do not use it for CAT II, III, or IV. Do not connect the measurement terminals to mains sockets.
Maximum ratings	The specified electrical ratings for the connectors of the instrument should not be exceeded at any time during operation. Please refer to the Specification for a comprehensive list of ratings.
Do not service or adjust anything yourself	There are no serviceable parts inside the instrument.

Software updates	Frequent software updates provide the user with many important improvements as well as new features. Only the last released software version is supported by Zurich Instruments.
Warnings	Instructions contained in any warning issued by the instrument, either by the software, the graphical user interface, the notes on the instrument or mentioned in this manual, must be followed.
Notes	Instructions contained in the notes of this user manual are of essential importance for correctly interpreting the acquired measurement data.
High voltage transients due to inductive loads	When measuring devices with high inductance, take adequate measures to protect the Signal Input connectors against the high voltages of inductive load switching transients. These voltages can exceed the maximum voltage ratings of the Signal Inputs and lead to damage.
Location and ventilation	This instrument or system is intended for indoor use in an installation category II and pollution degree 2 environment as per IEC 61010-1. Do not operate or store the instrument outside the ambient conditions specified in the Specification section. Do not block the ventilator opening on the back or the air intake on the chassis side and allow a reasonable space for the air to flow.
Cleaning	To prevent electrical shock, disconnect the instrument from AC mains power and disconnect all test leads before cleaning. Clean the outside of the instrument using a soft, lint-free cloth slightly dampened with water. Do not use detergent or solvents. Do not attempt to clean internally.
AC power connection and mains line fuse	For continued protection against fire, replace the line fuse only with a fuse of the specified type and rating. Use only the power cord specified for this product and certified for the country of use. Always position the device so that its power switch and the power cord are easily accessible during operation.
Main power disconnect	Unplug product from wall outlet and remove power cord before servicing. Only qualified, service-trained personnel should remove the cover from the instrument.
RJ45 sockets labeled ZCtrl	The two RJ45 sockets on the back panel labeled "Peripheral ZCtrl" are not intended for Ethernet LAN connection. Connecting an Ethernet device to these sockets may damage the instrument and/or the Ethernet device.
Operation and storage	Do not operate or store the instrument outside the ambient conditions specified in the Specification section.
Handling	Handle with care. Do not drop the instrument. Do not store liquids on the device, as there is a chance of spillage resulting in damage.
Safety critical systems	Do not use this equipment in systems whose failure could result in loss of life, significant property damage or damage to the environment.

If you notice any of the situations listed below, immediately stop the operation of the instrument, disconnect the power cord, and contact the support team at Zurich Instruments, either through the website form or through email.

Table 2.3: Unusual Conditions

Fan is not working properly or not at all	Switch off the instrument immediately to prevent overheating of sensitive electronic components.
Power cord or power plug on instrument is damaged	Switch off the instrument immediately to prevent overheating, electric shock, or fire. Please exchange the power cord only with one for this product and certified for the country of use.
Instrument emits abnormal noise, smell, or sparks	Switch off the instrument immediately to prevent further damage.
Instrument is damaged	Switch off the instrument immediately and ensure it is not used again until it has been repaired.

Table 2.4: Symbols

⏚	Earth ground

⏚	Chassis ground
⚠	Caution. Refer to accompanying documentation
⎓	DC (direct current)

2.4. Software Installation

The UHF Series Instrument is operated from a host computer with the LabOne software. To install the LabOne software on a computer, administrator rights may be required. In order to simply run the software later, a regular user account is sufficient. Instructions for downloading the correct version of the software packages from the Zurich Instruments website are described below in the platform-dependent sections. It is recommended to regularly update to the latest software version provided by Zurich Instruments. Thanks to the Automatic Update check feature, the update can be initiated with a single click from within the user interface, as shown in Software Update.

2.4.1. Installing LabOne on Windows

The installation packages for the Zurich Instruments LabOne software are available as Windows installer .msi packages. The software is available on the Zurich Instruments Download Center. Please ensure that you have administrator rights for the PC on which the software is to be installed. See LabOne compatibility for a comprehensive list of supported Windows systems.

Windows LabOne Installation

1. The UHF Series Instrument should not be connected to your computer during the LabOne software installation process.
2. Start the LabOne installer program with a name of the form `LabOne64-XX.XX.XXXXX.msi` by a double click and follow the instructions. Windows Administrator rights are required for installation. The installation proceeds as follows:
   - On the welcome screen click the **Next** button.

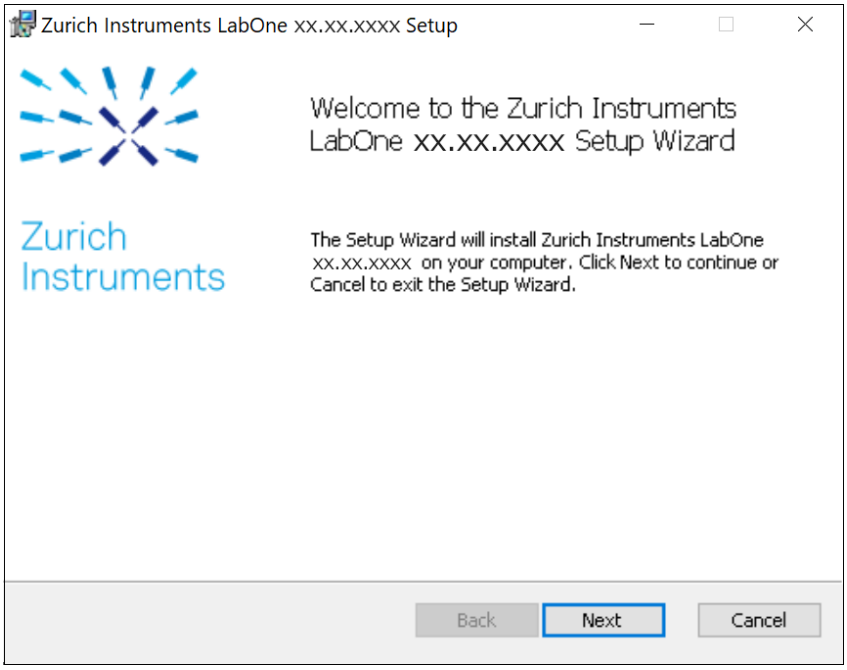


Figure 2.1: Installation welcome screen

   - After reading through the Zurich Instruments license agreement, check the "I accept the terms in the License Agreement" check box and click the **Next** button.
   - Review the features you want to have installed. For the UHF Series Instrument the "UHF Series Device", "LabOne User Interface" and "LabOne APIs" features are required. Please install the features for other device classes as well, if required. To proceed click the **Next** button.

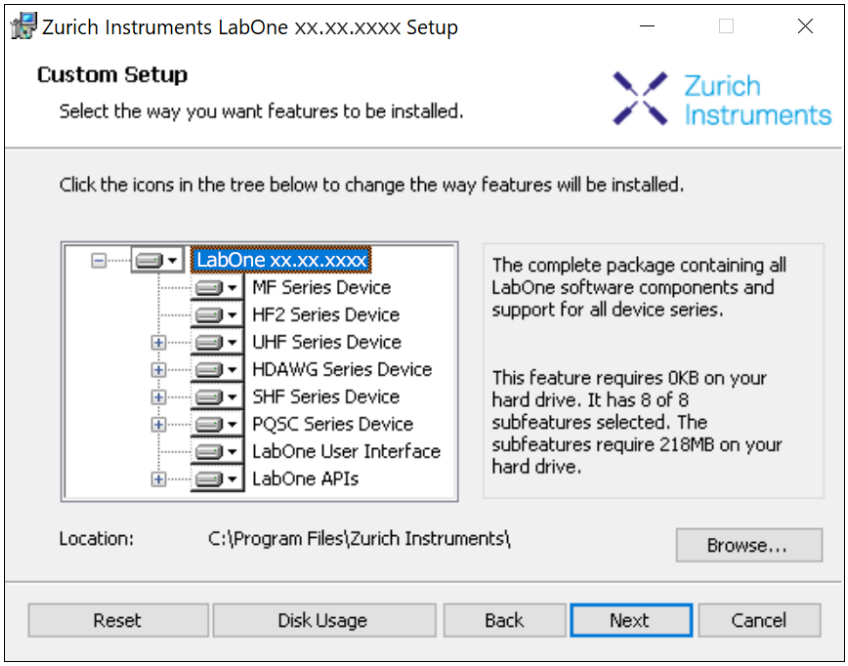


Figure 2.2: Custom setup screen

- Select whether the software should periodically check for updates. Note, the software will still not update automatically. This setting can later be changed in the user interface. If you would like to install shortcuts on your desktop area, select "Create a shortcut for this program on the desktop". To proceed click the **Next** button.

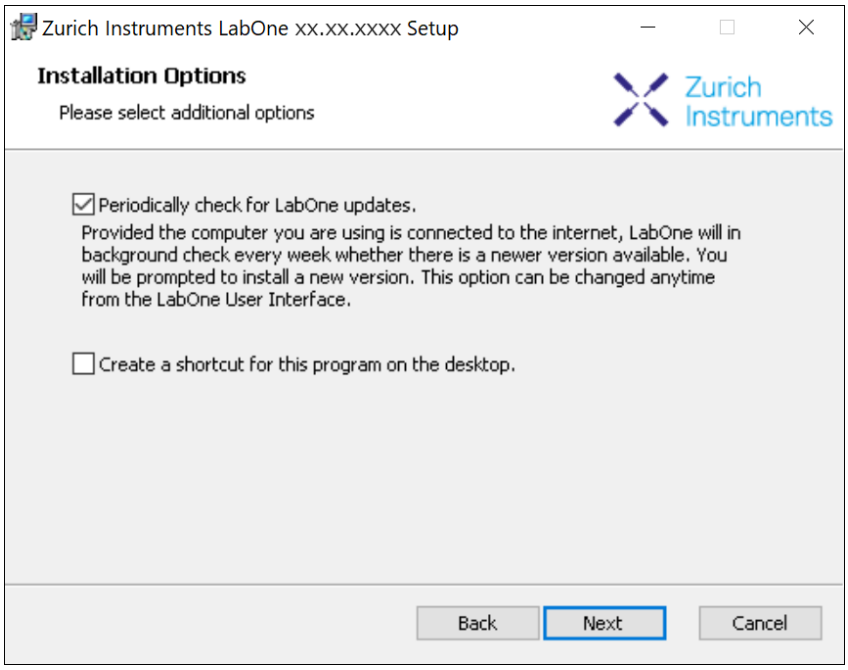


Figure 2.3: Automatic update check

- Click the **Install** button to start the installation process.
- Windows may ask up to two times to reboot the computer if you are upgrading. Make sure you have no unsaved work on your computer.

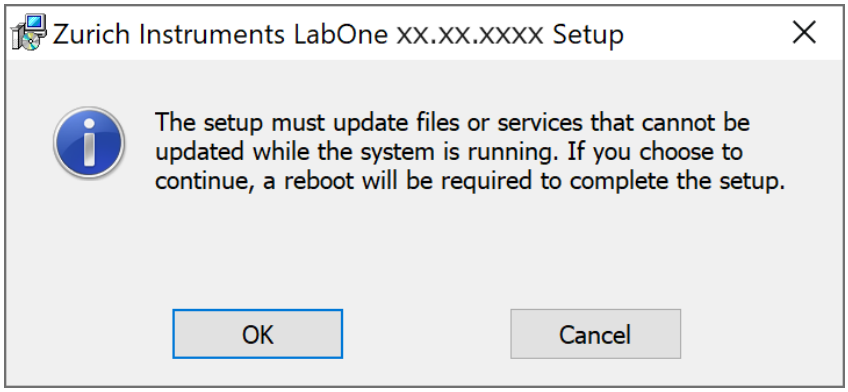


Figure 2.4: Installation reboot request

- During the first installation of LabOne, it is required to confirm the installation of some drivers from the trusted publisher Zurich Instruments. Click on **Install**.

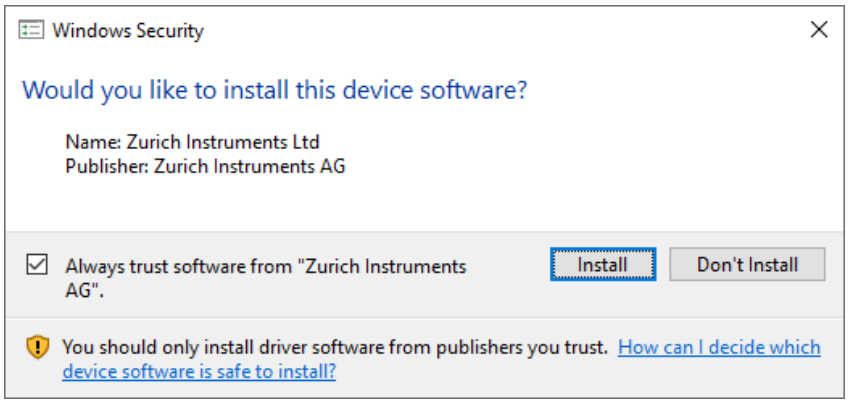


Figure 2.5: Installation driver acceptance

- Click **OK** on the following notification dialog.

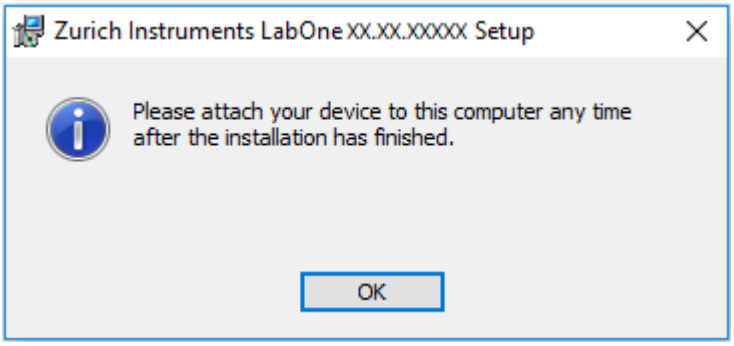


Figure 2.6: Installation completion screen

3. Click **Finish** to close the Zurich Instruments LabOne installer.
4. You can now start the LabOne User Interface as described in LabOne Software Start-up and choose an instrument to connect to via the Device Connection dialog shown in Device Connection dialog.

Warning

Do not install drivers from another source other than Zurich Instruments.

Start LabOne Manually on the Command Line

After installing the LabOne software, the Web Server and Data Server can be started manually using the command-line. The more common way to start LabOne under Windows is described in LabOne Software Start-up. The advantage of using the command line is being able to observe and change the behavior of the Web and Data Servers. To start the Servers manually, open a command-line

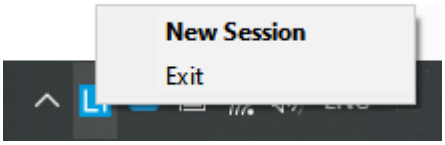
terminal (Command Prompt, PowerShell (Windows) or Bash (Linux)). For Windows, the current working directory needs to be the installation directory of the Web Server and Data Server. They are installed in the Program Files folder (usually: C:\Program Files) under \Zurich Instruments\LabOne in the WebServer and DataServer folders, respectively. The Web Server and Data Server (ziDataServer) are started by running the respective executable in each folder. Please be aware that only one instance of the Web Server can run at a time per computer. The behavior of the Servers can be changed by providing command line arguments. For a detailed list of all arguments see the command line help text:

```
$ ziWebServer --help
```

For the Data Server:

```
$ ziDataServer --help
```

One useful application of running the Webserver manually from a terminal window is to change the data directory from its default path in the user home directory. The data directory is a folder in which the LabOne Webserver saves all the measured data in the format specified by the user. Before running the Webserver from the terminal, the user needs to ensure there is no other instance of Webserver running in the background. This can be checked using the Tray Icon as shown below.



The corresponding command line argument to specify the data path is `--data-path` and the command to start the LabOne Webserver with a non-default directory path, e.g., `C:\data` is

```
C:\Program Files\Zurich Instruments\LabOne\WebServer> ziWebServer --data-path "C:
\data"
```

Windows LabOne Uninstallation

To uninstall the LabOne software package from a Windows computer, one can open the "Apps & features" page from the Windows start menu and search for LabOne. By selecting the LabOne item in the list of apps, the user has the option to "Uninstall" or "Modify" the software package as shown in Figure 2.7.

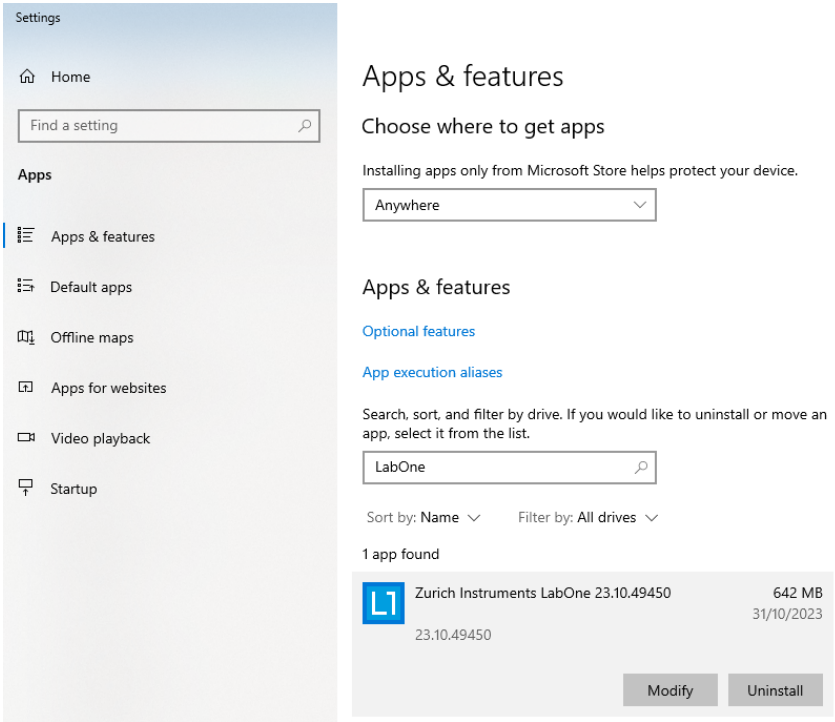


Figure 2.7: Uninstallation of LabOne on Windows computers

Warning

Although it is possible to install a new version of LabOne on a currently-installed version, it is highly recommended to first uninstall the older version of LabOne from the computer and then, install the new version. Otherwise, if the installation process fails, the current installation is damaged and cannot be uninstalled directly. The user will need to first repair the installation and then, uninstall it.

In case a current installation of LabOne is corrupted, one can simply repair it by selecting the option "Modify" in Figure 2.7. This will open the LabOne installation wizard with the option "Repair" as shown in Figure 2.8.

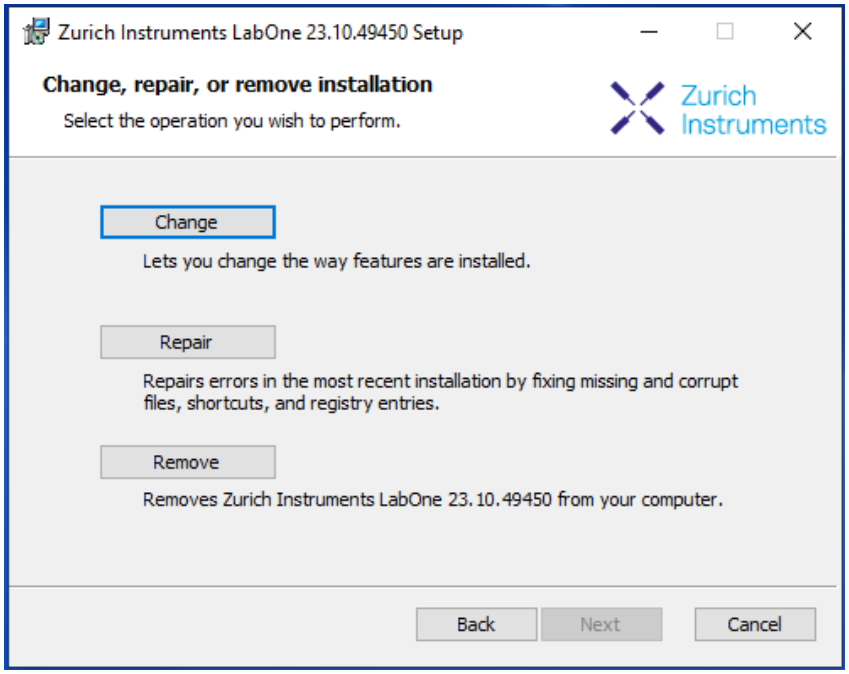


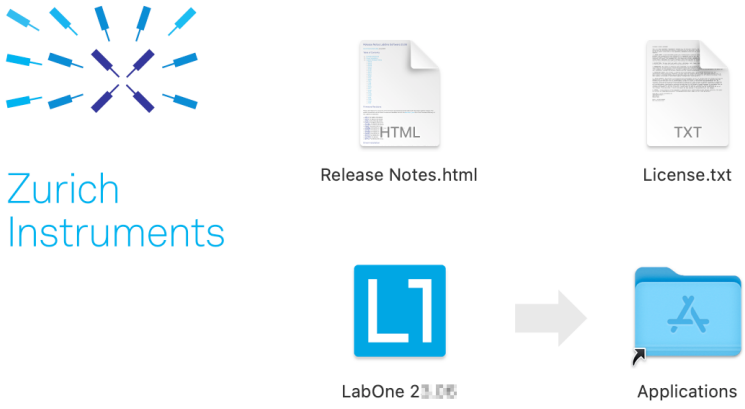
Figure 2.8: Repair of LabOne on Windows computers

After finishing the repair process, the normal uninstallation process described above can be triggered to uninstall LabOne.

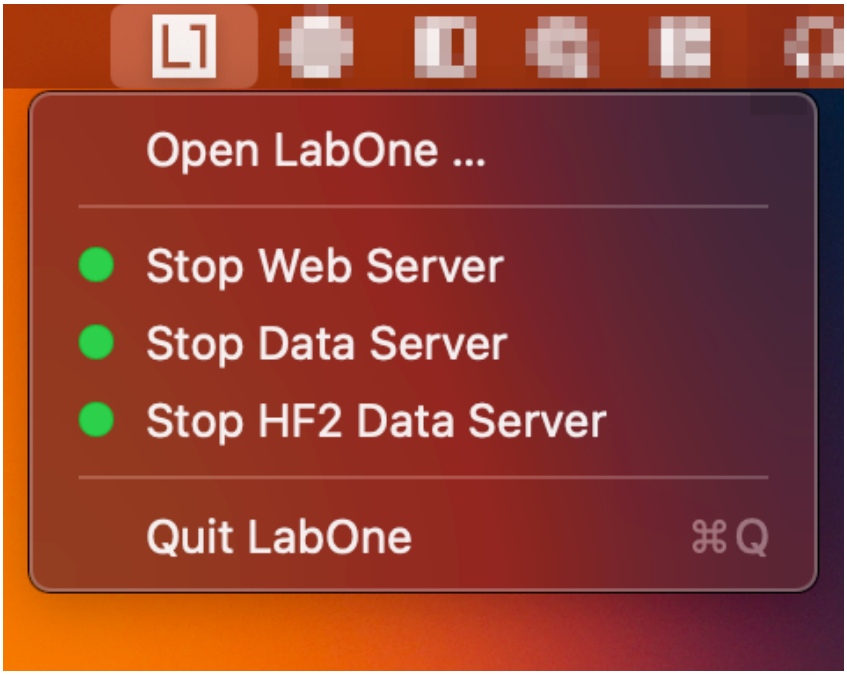
2.4.2. Installing LabOne on macOS

LabOne supports both Intel and ARM (M-series) architectures within a single universal disk image (DMG) file available in our Download Center.

- Download and double-click the DMG file to mount the image.



- The image contains a single LabOne application with all services needed.
- Once the application is started, a labone icon will appear in the menu bar. It allows the user to easily open a new session and shows the status of all services.



Uninstalling LabOne on macOS

To uninstall LabOne on macOS, simply drag the LabOne application to the trash bin.

Application Content

The LabOne application contains all resources available for macOS. This includes:

- The binaries for the Web Server and Data Servers.
- The binaries for the C, MATLAB, and LabVIEW APIs.
- An offline version of the user manuals.
- The latest firmware images for all instruments.

To access this content, right-click on the LabOne application and select "Show Package Contents". Then, go into Contents/Resources.

Note

Since the application name contains a space, one needs to escape it when using the command line to access the contents: `cd /Applications/LabOne\ 2X.XX.app/Contents/Resources`

Start LabOne Manually on the Command Line

To start the LabOne services like the data server and web server manually, one can use the command line.

The data server binary is called `ziDataServer` (`ziServer` for HF2 instruments) and is located at `Applications/LabOne\ 2X.XX.app/Contents/Resources/DataServer/`.

The web server binary is called `ziWebServer` and is located at `Applications/LabOne\ 2X.XX.app/Contents/Resources/WebServer/`.

Note

No special command line arguments are needed to start the LabOne services. Use the `--help` argument to see all available options.

2.4.3. Installing LabOne on Linux

Requirements

Ensure that the following requirements are fulfilled before trying to install the LabOne software package:

1. LabOne software supports typical modern GNU/Linux distributions (Ubuntu 14.04+, CentOS 7+, Debian 8+). The minimum requirements are glibc 2.17+ and kernel 3.10+.
2. You have administrator rights for the system.
3. The correct version of the LabOne installation package for your operating system and platform have been downloaded from the Zurich Instruments Download Center:

   ```
   LabOneLinux<arch>-<release>.<revision>.tar.gz,
   ```

Please ensure you download the correct architecture (x86-64 or arm64) of the LabOne installer. The `uname` command can be used in order to determine which architecture you are using, by running:

```
uname -m
```

in a command line terminal. If the command outputs `x86_64` the x86-64 version of the LabOne package is required, if it displays `aarch64` the ARM64 version is required.

Linux LabOne Installation

Proceed with the installation in a command line shell as follows:

1. Extract the LabOne tarball in a temporary directory:

   ```
   tar xzvf LabOneLinux<arch>-<release>-<revision>.tar.gz
   ```

2. Navigate into the extracted directory.

   ```
   cd LabOneLinux<arch>-<release>-<revision>
   ```

3. Run the install script with administrator rights and proceed through the guided installation, using the default installation path if possible:

   ```
   sudo bash install.sh
   ```

   The install script lets you choose between the following three modes:
   - Type "a" to install the Data Server program, the Web Server program, documentation and APIs.
   - Type "u" to install `udev` support (only necessary if HF2 Instruments will be used with this LabOne installation and not relevant for other instrument classes).
   - Type "ENTER" to install both options "a" and "u".
4. Test your installation by running the software as described in the next section.

Running the Software on Linux

The following steps describe how to start the LabOne software in order to access and use your instrument in the User Interface.

1. Start the LabOne Data Server program at a command prompt:

   ```
   $ ziDataServer
   ```

   You should be able to access your instrument. In case of problems please consult the Troubleshooting at the end of this chapter.
2. Start the Web Server program at a command prompt:

   ```
   $ ziWebServer
   ```

3. Start an up-to-date web browser and enter the `127.0.0.1:8006` in the browser's address bar to access the Web Server program and start the LabOne User Interface. The LabOne Web Server installed on the PC listens by default on port number 8006 instead of 80 to minimize the probability of conflicts.
4. You can now start the LabOne User Interface as described in LabOne Software Start-up and choose an instrument to connect to via the Device Connection dialog shown in Device Connection dialog.

Important

Do not use two Data Server instances running in parallel; only one instance may run at a time.

Uninstalling LabOne on Linux

The LabOne software package copies an uninstall script to the base installation path (the default installation directory is `/opt/zi/`). To uninstall the LabOne package please perform the following steps in a command line shell:

1. Navigate to the path where LabOne is installed, for example, if LabOne is installed in the default installation path:

```
$ cd /opt/zi/
```

2. Run the uninstall script with administrator rights and proceed through the guided steps:

```
$ sudo bash uninstall_LabOne<arch>-<release>-<revision>.sh
```

2.5. Connecting to the Instrument

The Zurich Instruments UHF Instrument is operated using the LabOne software. After installation of LabOne, the instrument can be connected to a PC by using either the Universal Serial Bus (USB) cable or the 1 Gbit/s Ethernet (1GbE) LAN cable supplied with the instrument. The LabOne software is controlled via a web browser once suitable physical and logical connections to the instrument have been made.

Note

The following web browsers are supported (latest versions)

- When using 1GbE, integrate the instrument physically into an existing local area network (LAN) by connecting the instrument to a switch in the LAN using an Ethernet cable. The instrument can then be accessed from a web browser running on any computer in the same LAN with LabOne installed. The Ethernet connection can also be point-to-point. This requires some adjustment of the network card settings of the host computer. Depending on the network configuration and the installed network card, one or the other connection scheme is better suited.
- Using the USB connection to physically connect to the instrument requires the installation of a USB driver on Windows computers. This driver is included in the LabOne software installer and will be installed on the host computer as part of the LabOne installation wizard.

2.5.1. LabOne Software Architecture

The Zurich Instruments LabOne software gives quick and easy access to the instrument from a host PC. LabOne also supports advanced configurations with simultaneous access by multiple software clients (i.e., LabOne User Interface clients and/or API clients), and even simultaneous access by

several users working on different computers. Here we give a brief overview of the architecture of the LabOne software. This will help to better understand the following chapters.

The software of Zurich Instruments equipment is server-based. The servers and other software components are organized in layers as shown in Figure 2.9.

- The lowest layer running on the PC is the LabOne Data Server, which is the interface to the connected instrument.
- The middle layer contains the LabOne Web Server, which is the server for the browser-based LabOne User Interface.
- The graphical user interface, together with the programming user interfaces, are contained in the top layer.

The architecture with one central Data Server allows multiple clients to access a device with synchronized settings. The following sections explain the different layers and their functionality in more detail.

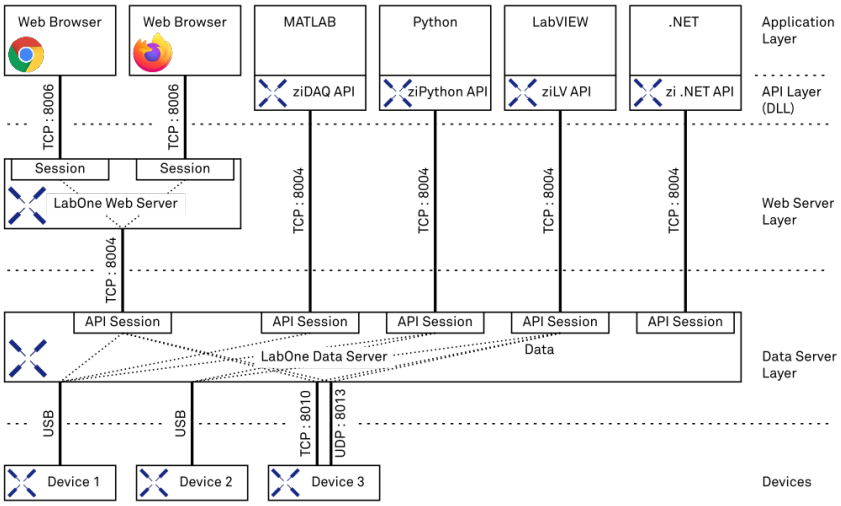


Figure 2.9: LabOne Software architecture

LabOne Data Server

The **LabOne Data Server** program is a dedicated server that is in charge of all communication to and from the device. The Data Server can control a single or also multiple instruments. It will distribute the measurement data from the instrument to all the clients that subscribe to it. It also ensures that settings changed by one client are communicated to other clients. The device settings are therefore synchronized on all clients. The Data Server is started automatically by a service when the PC is started. This service can be disabled if necessary, though the Data Server consumes only little resources when there is no active session. On a PC, only a single instance of a LabOne Data Server should be running.

LabOne Web Server

The LabOne Web Server is an application dedicated to serving up the web pages that constitute the LabOne user interface. The user interface can be opened with any device with a web browser. Since it is touch enabled, it is possible to work with the LabOne User Interface on a mobile device - like a tablet. The LabOne Web Server supports multiple clients simultaneously. This means that more than one session can be used to view data and to manipulate the instrument. A session could be running in a browser on the PC on which the LabOne software is installed. It could equally well be running in a browser on a remote machine.

With a LabOne Web Server running and accessing an instrument, a new session can be opened by typing in a network address and port number in a browser address bar. In case the Web Server runs on the **same** computer, the address is the localhost address (both are equivalent):

- `127.0.0.1:8006`
- `localhost:8006`

In case the Web Server runs on a **remote** computer, the address is the IP address or network name of the remote computer:

- `192.168.x.y:8006`
- `myPC.company.com:8006`

The most recent versions of the most popular browsers are supported: Chrome, Firefox, Edge, Safari and Opera.

LabOne API Layer

The instrument can also be controlled via the application program interfaces (APIs) provided by Zurich Instruments. APIs are provided in the form of DLLs for the following programming environments:

- MATLAB
- Python
- LabVIEW
- .NET
- C

The instrument can therefore be controlled by an external program, and the resulting data can be processed there. The device can be concurrently accessed via one or more of the APIs and via the user interface. This enables easy integration into larger laboratory setups. See the LabOne Programming Manual for further information. Using the APIs, the user has access to the same functionality that is available in the LabOne User Interface.

2.5.2. LabOne Software Start-up

This section describes the start-up of the LabOne User Interface which is used to control the UHF Series Instrument. If the LabOne software is not yet installed on the PC please follow the instructions in Software Installation. If the device is not yet connected please find more information in Visibility and Connection.

The LabOne User Interface start-up link can be found under the Windows 10 Start Menu (Under Windows 7 and 8, the LabOne User Interface start-up link can be found in `Start Menu → all programs / all apps → Zurich Instruments LabOne`). As shown in Figure 2.10, click on `Start Menu → Zurich Instruments LabOne`. This will open the User Interface in a new tab in your default web browser and start the LabOne Data Server and LabOne Web Server programs in the background. A detailed description of the software architecture is found in LabOne Software Architecture.

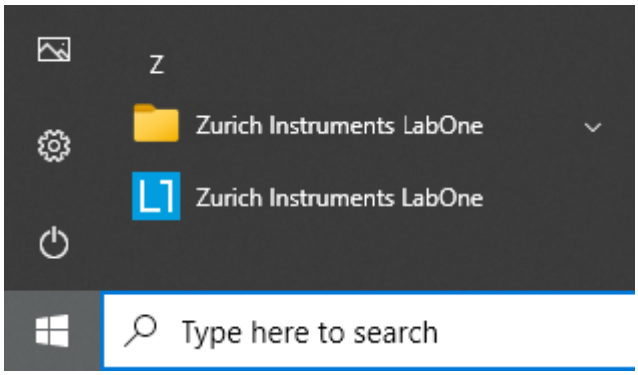


Figure 2.10: Link to the LabOne User Interface in the Windows 10 Start Menu

LabOne is an HTML5 browser-based program. This simply means that the user interface runs in a web browser and that a connection using a mobile device is also possible; simply specify the IP address (and port 8006) of the PC running the user interface.

Note

By creating a shortcut to Google Chrome on your desktop with the Target `path\to\chrome.exe -app=http://127.0.0.1:8006` set in Properties you can run the LabOne User Interface in Chrome in application mode, which improves the user experience by removing the unnecessary browser controls.

After starting LabOne, the Device Connection dialog Figure 2.11 is shown to select the device for the session. The term "session" is used for an active connection between the user interface and the device. Such a session is defined by device settings and user interface settings. Several sessions can be started in parallel. The sessions run on a shared LabOne Web Server. A detailed description of the software architecture can be found in the LabOne Software Architecture.

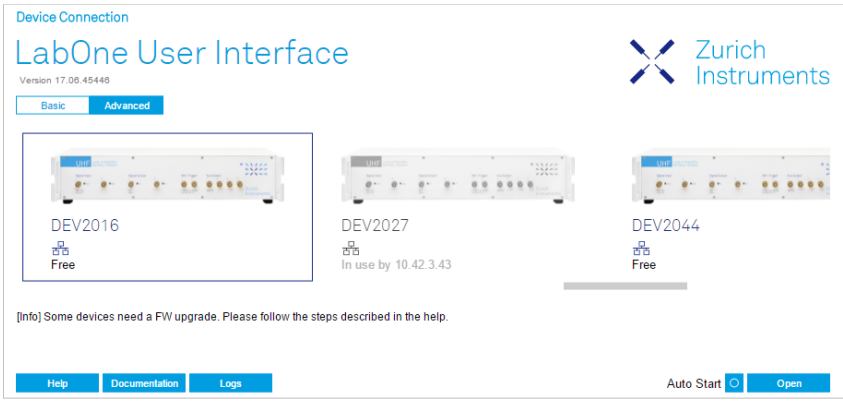


Figure 2.11: Device Connection dialog

The Device Connection dialog opens in the Basic view by default. In this view, all devices that are available for connection are represented by an icon with serial number and status information. If required, a button appears on the icon to perform a firmware upgrade. Otherwise, the device can be connected by a double click on the icon, or a click on the Open button at the bottom right of the dialog.

In some cases it's useful to switch to the Advanced view of the Device Connection dialog by clicking on the "Advanced" button. The Advanced view offers the possibility to select custom device and UI settings for the new session and gives further connectivity options that are particularly useful for multi-instrument setups.

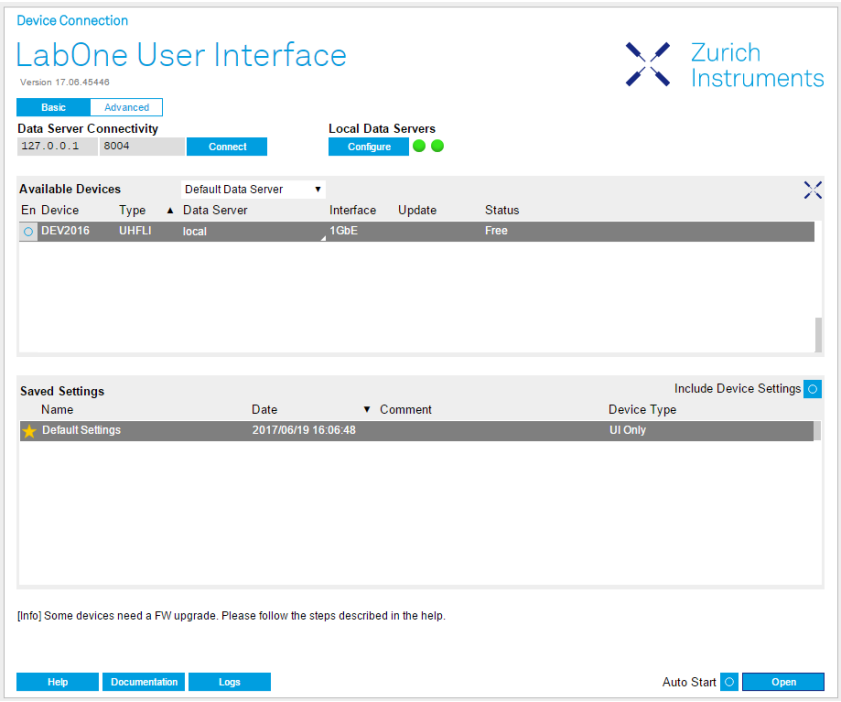


Figure 2.12: Device Connection dialog (Advanced view)

The Advanced view consists of three parts:

- Data Server Connectivity
- Available Devices
- Saved Settings

The Available Devices table has a display filter, usually set to **Default Data Server**, that is accessible by a drop-down menu in the header row of the table. When changing this to **Local Data Servers**, the Available Devices table will show only connections via the Data Server on the host PC and will contain all instruments directly connected to the host PC via USB or to the local network via 1GbE. When using the **All Data Servers** filter, connections via Data Servers running on other PCs in

the network also become accessible. Once your instrument appears in the Available Devices table, perform the following steps to start a new session:

1. Select an instrument in the **Available Devices** table.
2. Select a setting file in the **Saved Settings** list unless you would like to use the Default Settings.
3. Start the session by clicking on Open

Note

By default, opening a new session will only load the UI settings (such as plot ranges), but not the device settings (such as signal amplitude) from the saved settings file. In order to include the device settings, enable the **Include Device Settings** checkbox. Note that this can affect existing sessions since the device settings are shared between them.

Note

In case devices from other Zurich Instruments series (UHF, HF2, MF, HDAWG, PQSC, GHF, or SHF) are used in parallel, the list in **Available Devices** section can contain those as well.

The following sections describe the functionality of the **Device Connection** dialog in detail.

Data Server Connectivity

The Device Connection dialog represents a Web Server. However, on start-up the Web Server is not yet connected to a LabOne Data Server. With the **Connect/Disconnect** button the connection to a Data Server can be opened and closed.

This functionality can usually be ignored when working with a single UHF Series Instrument and a single host computer. Data Server Connectivity is important for users operating their instruments from a remote PC, i.e., from a PC different to the PC on which the Data Server is running or for users working with multiple instruments. The Data Server Connectivity function then gives the freedom to connect the Web Server to one of several accessible Data Servers. This includes Data Servers running on remote computers, and also Data Servers running on an MF Series instrument.

In order to work with a UHF, HF2, HDAWG, PQSC, GHF, or SHF instrument remotely, proceed as follows. On the computer directly connected to the instrument (Computer 1) open a User Interface session and change the Connectivity setting in the Config tab to "From Everywhere". On the remote computer (Computer 2), open the Device Connection dialog by starting up the LabOne User Interface and then go to the Advanced view by clicking on Advanced on the top left of the dialog. Change the display filter from Default Data Server to All Data Servers by opening the drop-down menu in the header row of the Available Devices table. This will make the Instrument connected to Computer 1 visible in the list. Select the device and connect to the remote Data Server by clicking on Connect. Then start the User Interface as described above.

Note

When using the filter "All Data Servers", take great care to connect to the right instrument, especially in larger local networks. Always identify your instrument based on its serial number in the form DEV0000, which can be found on the instrument back panel.

Available Devices

The Available Devices table gives an overview of the visible devices. A device is ready for use if either marked free or connected. The first column of the list holds the **Enable** button controlling the connection between the device and a Data Server. This button is greyed out until a Data Server is connected to the LabOne Web Server using the Connect button. If a device is connected to a Data Server, no other Data Server running on another PC can access this device.

The second column indicates the serial number and the third column shows the instrument type. The fourth column shows the host name of the LabOne Data Server controlling the device. The next

column shows the interface type. For UHF Instruments the interfaces USB or 1GbE are available and are listed if physically connected. The LabOne Data Server will scan for the available devices and interfaces every second. If a device has just been switched on or physically connected it may take up to 20 s before it becomes visible to the LabOne Data Server. If an interface is physically connected but not visible please read Visibility and Connection.

If a firmware update of the instrument is available, the second to last column will show a button, and a simple click on it will update the instrument. The last column indicates the status of the device. explains the meaning of some of the possible device statuses.

Table 2.5: Device Status Information

Connected	The device is connected to a LabOne Data Server, either on the same PC (indicated as local) or on a remote PC (indicated by its IP address). The user can start a session to work with that device.
Free	The device is not in use by any LabOne Data Server and can be connected by clicking the **Open** button.
In Use	The device is in use by a LabOne Data Server. As a consequence the device cannot be accessed by the specified interface. To access the device, a disconnect is needed.
Device FW upgrade required/available	The firmware of the device is out of date. Please first upgrade the firmware as described in Software Update.
Device not yet ready	The device is visible and starting up.

Saved Settings

Settings files can contain both UI and device settings. UI settings control the structure of the LabOne User Interface, e.g. the position and ordering of opened tabs. Device settings specify the set-up of a device. The device settings persist on the device until the next power cycle or until overwritten by loading another settings file.

The columns are described in Table 2.6. The table rows can be sorted by clicking on the column header that should be sorted. The default sorting is by time. Therefore, the most recent settings are found on top. Sorting by the favorite marker or setting file name may be useful as well.

Table 2.6: Column Descriptions

☆ ★	Allows favorite settings files to be grouped together. By activating the stars adjacent to a settings file and clicking on the column heading, the chosen files will be grouped together at the top or bottom of the list accordingly. The favorite marker is saved to the settings file. When the LabOne user interface is started next time, the row will be marked as favorite again.
Name	The name of the settings file. In the file system, the file name has the extension .md.
Date	The date and time the settings file was last written.
Comment	Allows a comment to be stored in the settings file. By clicking on the comment field a text can be typed in which is subsequently stored in the settings file. This comment is useful to describe the specific conditions of a measurement.
Device Type	The instrument type with which this settings file was saved.

Special Settings Files

Certain file names have the prefix "last_session_". Such files are created automatically by the LabOne Web Server when a session is terminated either explicitly by the user, or under critical error conditions, and save the current UI and device settings. The prefix is prepended to the name of the most recently used settings file. This allows any unsaved changes to be recovered upon starting a new session.

If a user loads such a last session settings file the "last_session_" prefix will be cut away from the file name. Otherwise, there is a risk that an auto-save will overwrite a setting which was saved explicitly by the user.

The settings file with the name "Default Settings" contains the default UI settings. See button description in Table 2.7.

Table 2.7: Button Descriptions

Open	The settings contained in the selected settings file will be loaded. The button "Include Device Settings" controls whether only UI settings are loaded, or if device settings are included.
Include Device Settings	Controls which part of the selected settings file is loaded upon clicking on Open. If enabled, both the device and the UI settings are loaded.
Auto Start	Skips the session dialog at start-up if selected device is available. The default UI settings will be loaded with unchanged device settings.

Note

The user setting files are saved to an application-specific folder in the directory structure. The best way to manage these files is using the File Manager tab.

Note

The factory default UI settings can be customized by saving a file with the name "default_ui" in the Config tab once the LabOne session has been started and the desired UI setup has been established. To use factory defaults again, the "default_ui" file must be removed from the user setting directory using the File Manager tab.

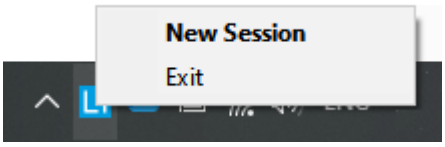
Note

Double clicking on a device row in the Available Devices table is a quick way of starting the default LabOne UI. This action is equivalent to selecting the desired device and clicking the **Open** button.

Double clicking on a row in the Saved Settings table is a quick way of loading the LabOne UI with those UI settings and, depending on the "Include Device Settings" checkbox, device settings. This action is equivalent to selecting the desired settings file and clicking the **Open** button.

Tray Icon

When LabOne is started, a tray icon appears by default in the bottom right corner of the screen, as shown in the figure below. By right-clicking on the icon, a new web server session can be opened quickly, or the LabOne Web and Data Servers can be stopped by clicking on Exit. Double-clicking the icon also opens a new web server session, which is useful when setting up a connection to multiple instruments, for example.

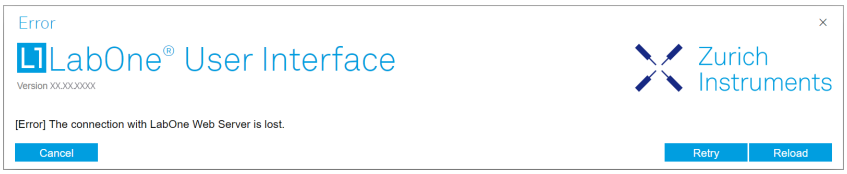


Messages

The LabOne Web Server will show additional messages in case of a missing component or a failure condition. These messages display information about the failure condition. The following paragraphs list these messages and give more information on the user actions needed to resolve the problem.

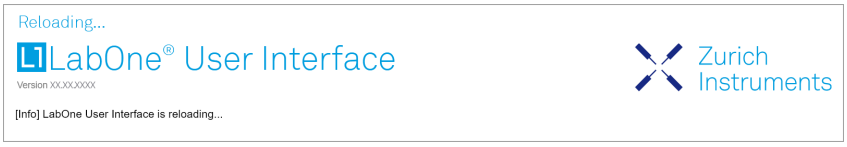
Lost Connection to the LabOne Web Server

In this case the browser is no longer able to connect to the LabOne Web Server. This can happen if the Web Server and Data Server run on different PCs and a network connection is interrupted. As long as the Web Server is running and the session did not yet time out, it is possible to just attach to the existing session and continue. Thus, within about 15 seconds it is possible with **Retry** to recover the old session connection. The **Reload** button opens the Device Connection dialog shown in Figure 2.11. The figure below shows an example of the Connection Lost dialog.



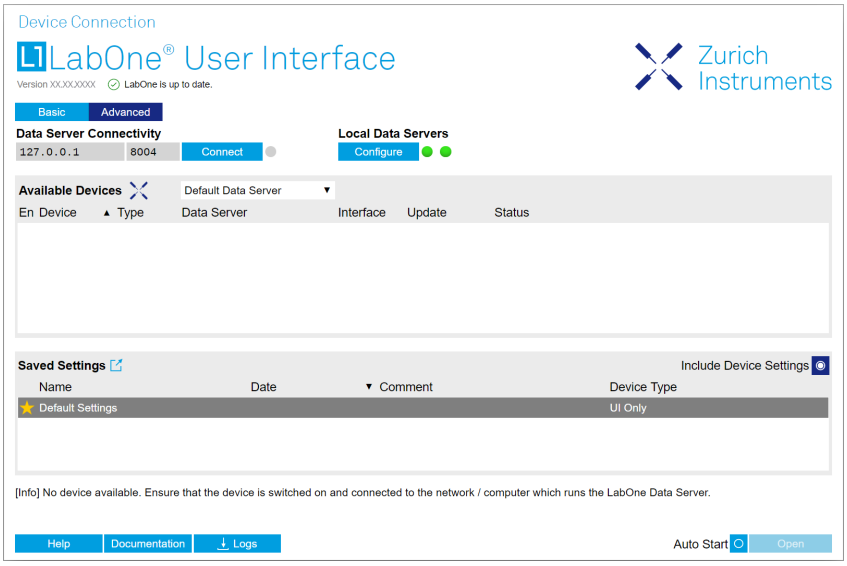
Reloading...

If a session error cannot be handled, the LabOne Web Server will restart to show a new Device Connection dialog as shown in Figure 2.11. During the restart a window is displayed indicating that the LabOne User Interface will reload. If reloading does not happen the same effect can be triggered by pressing F5 on the keyboard. The figure below shows an example of this dialog.



No Device Discovered

An empty "Available Devices" table means that no devices were discovered. This can mean that no LabOne Data Server is running, or that it is running but failed to detect any devices. The device may be switched off or the interface connection fails. For more information on the interface between device and PC see Visibility and Connection. The figure below shows an example of this dialog.



No Device Available

If all the devices in the "Available Devices" table are shown grayed, this indicates that they are either in use by another Data Server, or need a firmware upgrade. For firmware upgrade see Software Update. If all the devices are in use, access is not possible until a connection is relinquished by another Data Server.

2.5.3. Visibility and Connection

There are several ways to connect the Zurich Instruments lock-in amplifier to a host computer. The device can either be connected by Universal Serial Bus (USB) or by 1 Gbit/s Ethernet (1GbE). The USB connection is a point-to-point connection between the device and the PC on which the Data Server runs. The 1GbE connection can be a point-to-point connection or an integration of the device into the local network (LAN). Depending on the network configuration and the installed network card, one or the other connectivity is better suited.

If a device is connected to a network, it can be accessed from multiple PCs. To manage the access to the device, there are two different connectivity states: visible and connected. It is important to distinguish if a device is just physically connected over USB or 1GbE or actively controlled by the LabOne Data Server. In the first case the device is visible to the LabOne Data Server. In the second case the device is logically connected.

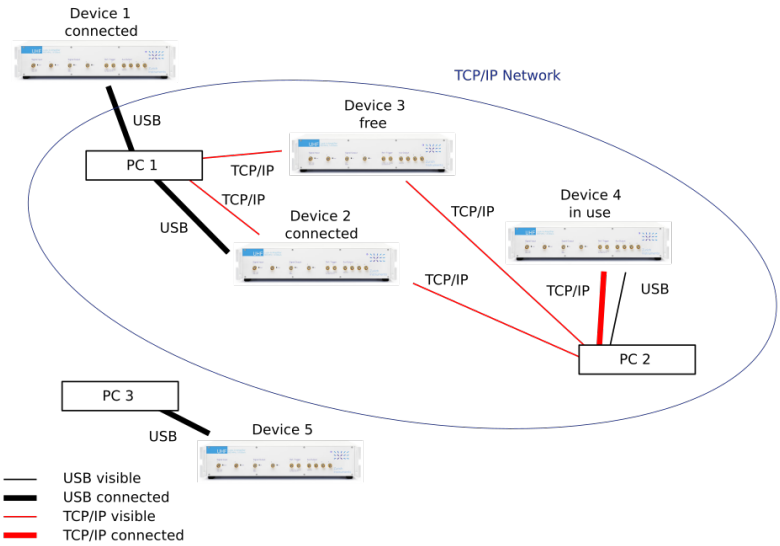


Figure 2.13: Connectivity

Figure 2.13 shows some examples of possible configurations of PC-to-device connectivity.

- Data Server on PC 1 is connected to device 1 (USB) and device 2 (USB).
- Data Server on PC 2 is connected to device 4 (TCP/IP).
- Data Server on PC 3 is connected to device 5.
- The device 3 is free and visible to PC 1 and PC 2 over TCP/IP.
- Devices 2 and 4 are physically connected by TCP/IP and USB interface. Only one interface is logically connected to the Data Server.

Visible Devices

A device is visible if the Data Server can identify it. On a TCP/IP network, several PCs running a Data Server will detect the same device as visible, i.e., discover it. If a device is discovered, the LabOne Data Server can initiate a connection to access the device and stream measurement data. Only a single Data Server can be connected to a device at a time.

Connected Device

Once connected to a device, the Data Server has exclusive access to data of that device. If another Data Server from another PC already has an active connection to the device, the device is still visible but cannot be connected.

Although a Data Server has exclusive access to a connected device, the Data Server can have multiple clients. Like this, multiple browser and API sessions can access the device simultaneously.

2.5.4. USB Connectivity

To control the device over USB, connect the instrument with the supplied USB cable to the PC on which the LabOne Software is installed. The USB driver needed for controlling the device is included in the LabOne Installer package. Ensure that the device uses the latest firmware. The software will automatically use the USB interface for controlling the device if available. If the USB connection is not available, the 1GbE connection may be selected. It is possible to enforce or exclude a specific interface connection.

Note

To use the device exclusively over the USB interface, modify the shortcut of the LabOne User Interface and LabOne Data Server in the Windows Start menu. Right-click and go to Properties, then add the following command line argument to the Target LabOne User Interface: `--interface-usb true --interface-ip false`

A device connected over USB can be automatically connected to by the Data Server because there is only a single host PC to which the device interface is physically connected.

auto-connect = on

If a device is attached via a USB cable, a connection will be established automatically by the Data Server. This is the default behavior.

auto-connect = off

To disable automatic connection via USB, add the following command line argument when starting the Data Server: `--auto-connect=off`

This is achieved by right clicking the LabOne Data Server shortcut in the Start menu, selecting "Properties" and adding the text to the Target field as shown in Figure 2.14.

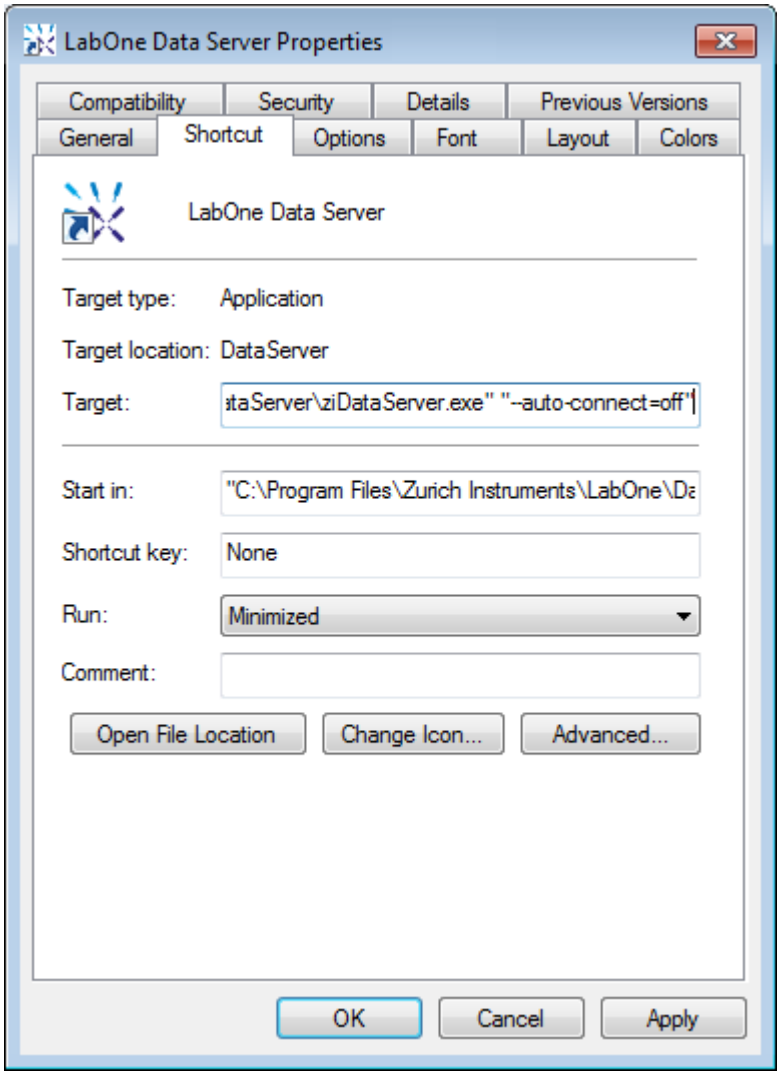


Figure 2.14: auto-connect

2.5.5. 1GbE Connectivity

There are three methods for connecting to the device via 1GbE:

- Multicast DHCP
- Multicast point-to-point (P2P)
- Static Device IP

Multicast DHCP is the simplest and preferred connection method. Other connection methods can become necessary when using network configurations that conflict with local policies. This particularly concerns the enabling of Jumbo frames, which is an essential setting for good performance when using high data transfer rates.

Note

To use the device exclusively over the Ethernet interface, modify the shortcut of the LabOne User Interface UHF and LabOne Data Server UHF in the Windows Start menu. Right-click and go to Properties, then add the following command line argument to the Target field: `--interface-usb false --interface-ip true`

Multicast DHCP

The most straightforward TCP/IP connection method is to rely on a network configuration to recognize the UHF Instrument. When connecting the instrument to a local area network (LAN), the DHCP server will assign an IP address to the instrument like to any PC in the network. In case of restricted networks, the network administrator may be required to register the device on the

network by means of the MAC address. The MAC address is indicated on the back panel of the instrument. The LabOne Data Server will detect the device in the network by means of a multicast.

If the network configuration does not support multicast, or if the host computer has other network cards installed, it is necessary to use a static IP setup as described below. The UHF Instrument is configured to accept the IP address from the DHCP server, or to fall back to the IP address `192.168.1.10` if it does not get the address from the DHCP server.

Requirements

- Network supports multicast

Multicast Point-to-Point

Setting up a point-to-point (P2P) network consisting only of the host computer and the UHF Instrument avoids problems related to special network policies. Since it is nonetheless necessary to stay connected to the internet, it is recommended to install two network cards in the computer, one of which is used for network connectivity (e.g. internet), the other can be used for connecting to the Instrument. Notebooks can generally profit from wireless LAN for internet connection.

In such a P2P network the IP address of the host computer needs to be set to a static value, whereas the IP address of the device can be left dynamic.

1. Connect the 1GbE port of the network card that is dedicated for device connectivity directly to the 1GbE port of the UHF Instrument
2. Set this network card to static IP in TCP/IPv4 using the address `192.168.1.n`, where n=[2..9] and the mask `255.255.255.0`, see Static IP configuration for the host computer (go to `Control Panel` → `Internet Options` → `Network and Internet` → `Network and Sharing Center` → `Local Area Connection` → `Properties`).
3. Start up the LabOne User Interface normally. If your instrument does not show in the list of Available Devices, the reason may be that your network card does not support multicast. In that case use a static device IP as described below.

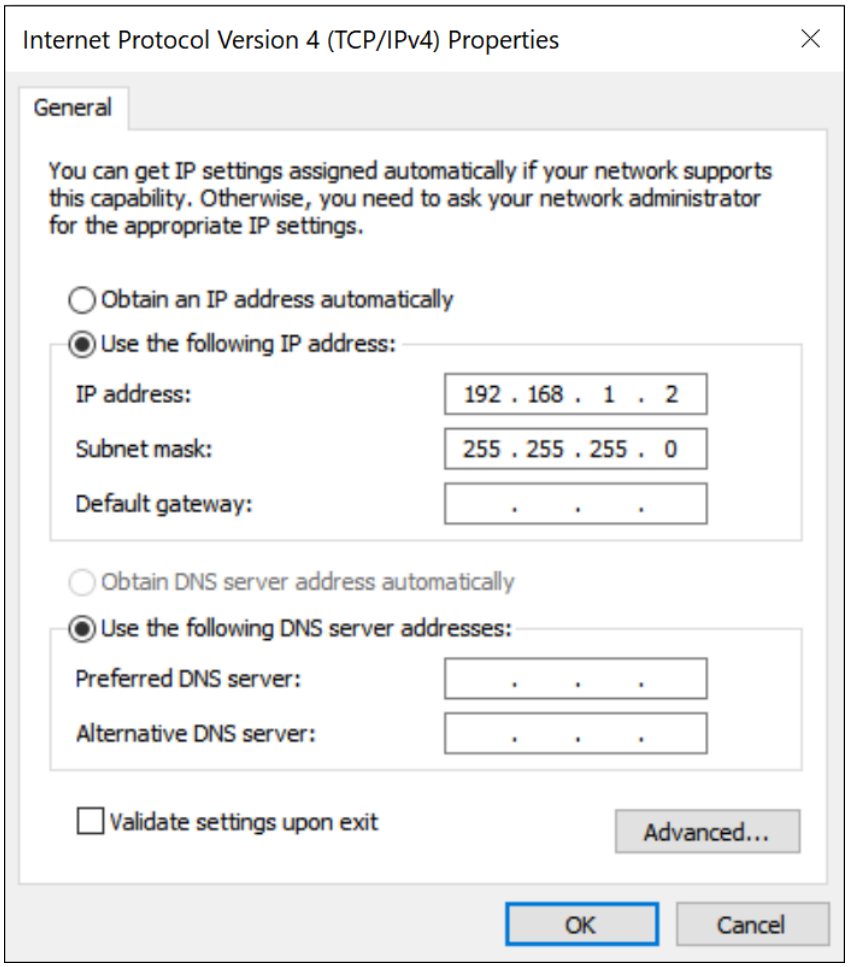


Figure 2.15: Static IP configuration for the host computer

Requirements

- Two networks cards needed for additional connection to internet
- Network card of PC supports multicast
- Network card connected to the device must be in static IP4 configuration

Note

A power cycle of the UHF Instrument is required if it was previously connected to a network that provided a IP address to the instrument.

Note

Only IP v4 is currently supported. There is no support for IP v6.

Note

If the instrument is detected by LabOne but the connection can not be established, the reason can be the firewall blocking the connection. It is then recommended to change the P2P connection from Public to Private. This is achieved by turning on network discovery in the Private tab of the network's advanced sharing settings as shown in the figure below.

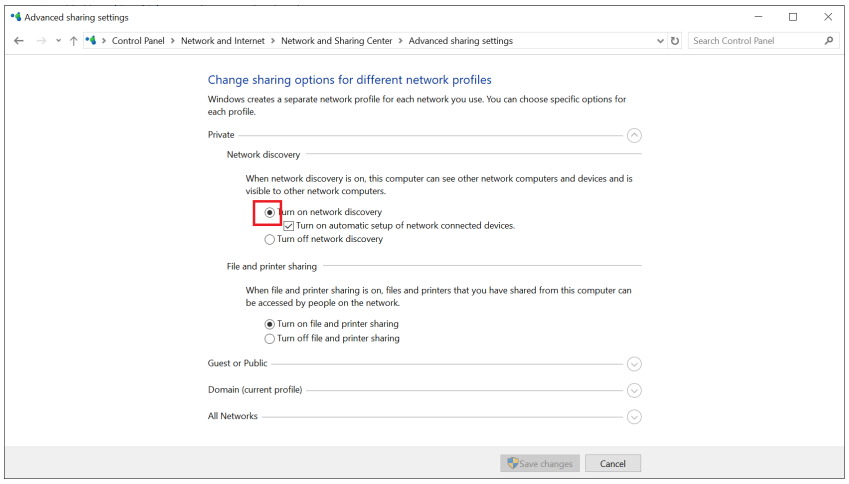


Figure 2.16: Turn on network discovery for Private P2P connection

Warning

Changing the IP settings of your network adapters manually can interfere with its later use, as it cannot be used anymore for network connectivity until it is configured again for dynamic IP.

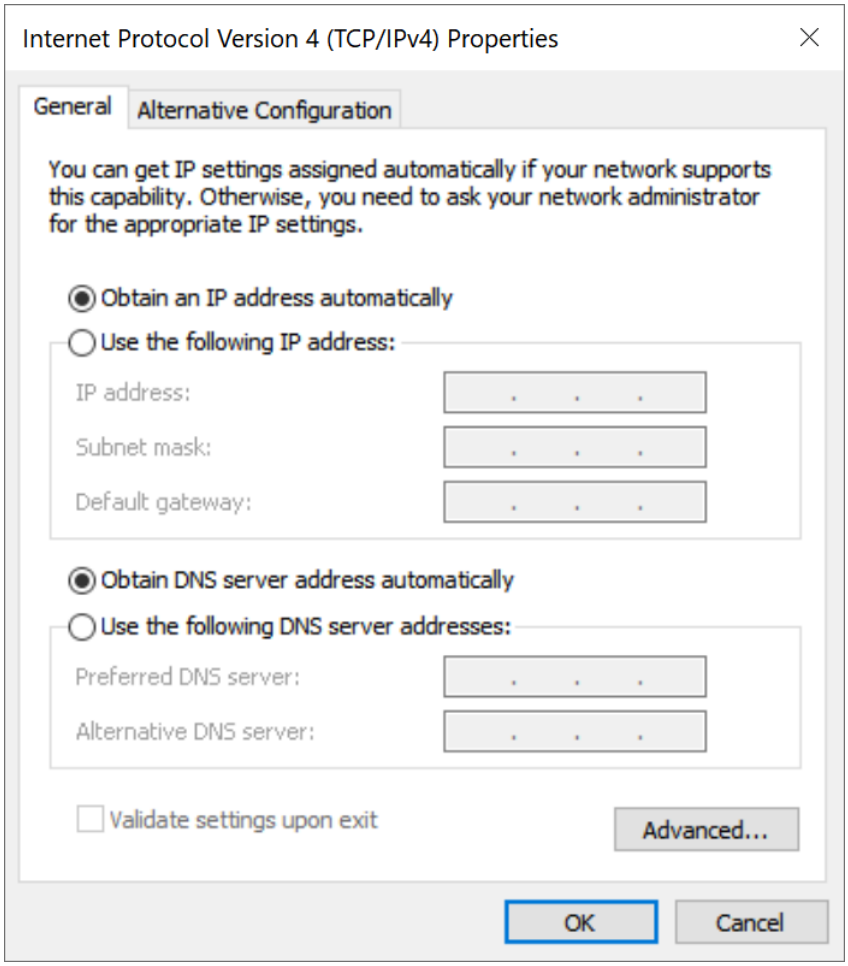
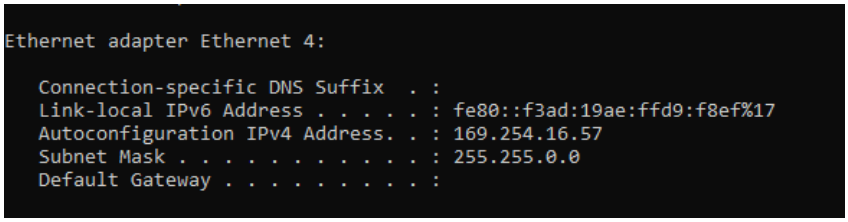


Figure 2.17: Dynamic IP configuration for the host computer

Static Device IP

Although it is highly recommended to use dynamic IP assignment method in the host network of the instrument, there may be cases where the user wants to assign a static IP to the instrument. For instance, when the host network only contains Ethernet switches and hubs but no Ethernet routers are included, there is no DHCP server to dynamically assign an IP to the instrument. It is still advised to add an Ethernet router to the network and benefit from dynamic IP assignment; however, if a router is not available, the instrument can be configured to work with a static IP.

Note that the static IP assigned to the instrument must be within the same range of the IP assigned to the host computer. Whether the host computer's IP is assigned statically or by a fallback mechanism, one can find this IP by running the command `ipconfig` or `ipconfig/all` in the operating system's terminal. As an example, Figure 2.18 shows the outcome of running `ipconfig` in the terminal.



```
Ethernet adapter Ethernet 4:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::f3ad:19ae:ffd9:f8ef%17
   Autoconfiguration IPv4 Address. . : 169.254.16.57
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Default Gateway . . . . . . . . . :
```

Figure 2.18: IP and subnet mask of host computer

It shows the network adapter of the host computer can be reached via the IP `169.254.16.57` and it uses a subnet mask of `255.255.0.0`. To make sure that the instrument is visible to this computer, one needs to assign a static IP of the form `169.254.x.x` and the same subnet mask to the instrument. To do so, the user should follow the instructions below.

1. Attach the instrument using an Ethernet cable to the network where the user's computer is hosted.
2. Attach the instrument via a USB cable to the host computer and switch it on.
3. Open the LabOne user interface (UI) and connect to the instrument via USB.
4. Open the "Device" tab of the LabOne UI and locate the "Communication" section as shown in Configuration of static IP in LabOne UI.
5. Write down the desired static IP, e.g. `169.254.16.20`, into the numeric field "IPv4 Address".
6. Add the same subnet mask as the host computer, e.g. `255.255.0.0` to the numeric field "IPv4 Mask".
7. You can leave the field "Gateway" as `0.0.0.0` or change to be similar to the IP address but ending with `1`, e.g. `169.254.16.1`.
8. Enable the radio button for "Static IP".
9. Press the button "Program" to save the new settings to the instruments.
10. Power cycle the instrument and remove the USB cable. The instrument should be visible to LabOne via Ethernet connection.

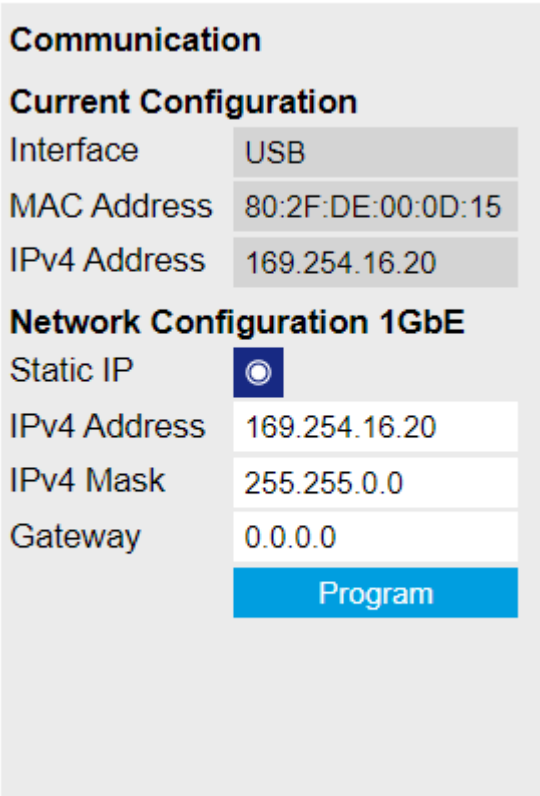
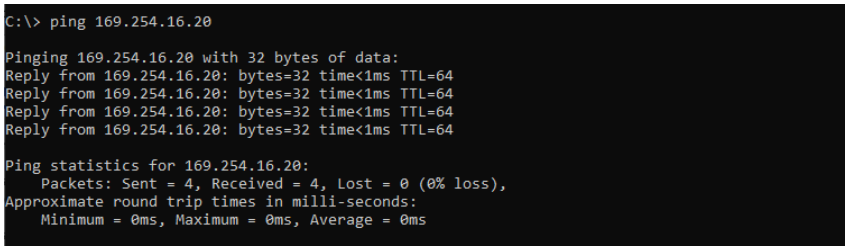


Figure 2.19: Configuration of static IP in LabOne UI

To make sure the IP assignment is done properly, one can use the command `ping` to check if the instrument can be reached through the network using its IP address. Figure 2.20 shows the outcome of `ping` when the instrument is visible via the IP `169.254.16.20`.



```
C:\> ping 169.254.16.20

Pinging 169.254.16.20 with 32 bytes of data:
Reply from 169.254.16.20: bytes=32 time<1ms TTL=64
Reply from 169.254.16.20: bytes=32 time<1ms TTL=64
Reply from 169.254.16.20: bytes=32 time<1ms TTL=64
Reply from 169.254.16.20: bytes=32 time<1ms TTL=64

Ping statistics for 169.254.16.20:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Figure 2.20: Instrument visible through pinging

If set properly according to the instructions above, the instrument will use the same static IP configurations after each power cycle.

Recommended Network Cards

Especially when working at high data transfer rates, it's recommended to use one of the network card models that have been tested by Zurich Instruments. In comparison, some older network cards either don't support performance features such as receive side scaling or jumbo packets, or have performance limitations that can lead to sample loss. In addition to the choice of the network card, a powerful processor on the host PC is essential for preventing data loss.

lists the network cards recommended by Zurich Instruments. The rightmost column lists the recommended settings where they differ from the default. Under Windows, the network card settings can be accessed from the Device Manager. In the Network Adapters group, right-click on the entry for the network card and select Properties to open a dialog such as the one shown in Figure 2.21

Recommended network cards

Model	Requirements	Settings
Intel I210-T1	PCIe x1	▬ Energy Efficient Ethernet: off ▬ Jumbo Packet: 9014 bytes ▬ Receive Side Scaling Queues: 4 queues
Intel I350-T2	PCIe x16	▬ Jumbo Packet: 9014 bytes ▬ Receive Side Scaling: enabled ▬ Maximum Number of RSS Queues: 4
Intel PRO/1000 PT	PCIe x4	▬ Jumbo Packet: 9014 bytes

Recommended network cards

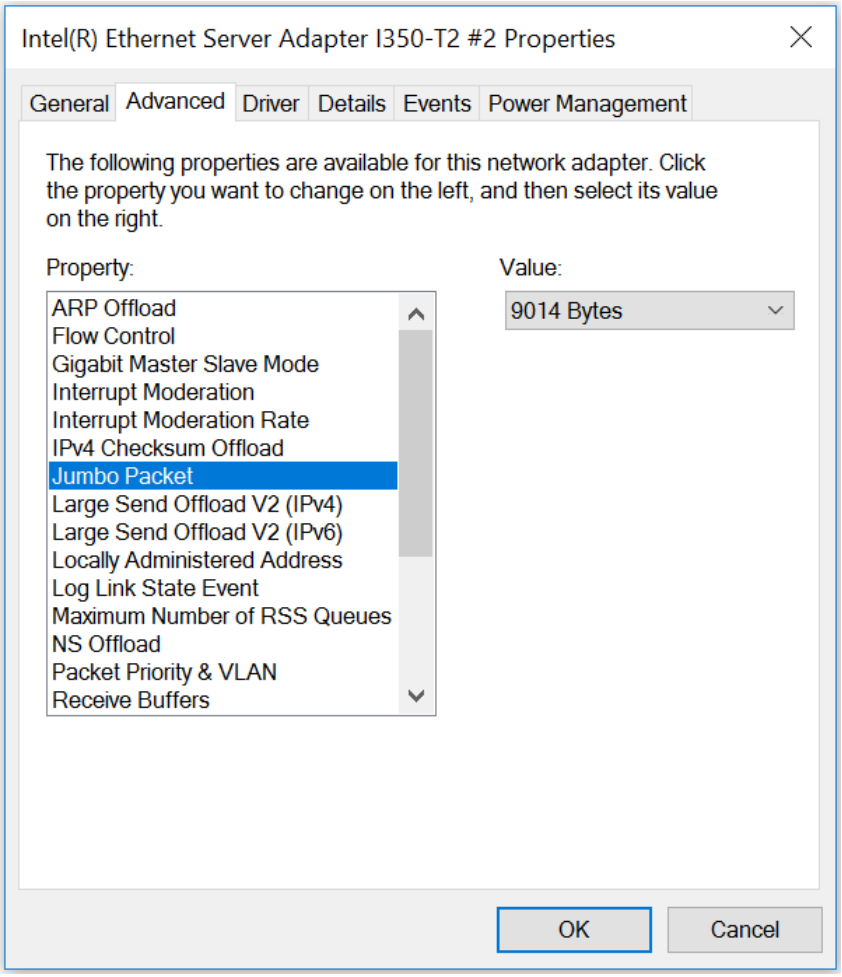


Figure 2.21: Network card properties dialog for the I350-T2 model. The dialog is accessible via the Windows Device Manager.

2.6. Software Update

2.6.1. Overview

It is recommended to regularly update the LabOne software on the UHFQA Instrument to the latest version. In case the Instrument has access to the internet, this is a very simple task and can be done with a single click in the software itself, as shown in Updating LabOne using Automatic Update Check. If you use one of the LabOne APIs with a separate installer, don't forget to update this part of the software, too.

2.6.2. Updating LabOne using Automatic Update Check

Updating the software is done in two steps. First, LabOne is updated on the PC by downloading and installing the LabOne software from the Zurich Instruments downloads page, as shown in Software Installation. Second, the instrument firmware needs to be updated from the Device Connection dialog after starting up LabOne. This is shown in Updating the Instrument Firmware . In case "Periodically check for updates" has been enabled during the LabOne installation and LabOne has access to the internet, a notification will appear on the Device Connection dialog whenever a new version of the software is available for download. This setting can later be changed in the Config tab of the LabOne user interface. In case automatic update check is disabled, the user can manually check for updates at any time by clicking on the button Check For Update in the Device Connection dialog. In case an update is found, clicking on the button "Update Available" shown in Figure 2.22 will start a download the latest LabOne installer for Windows or Linux, see Figure 2.23. After download, proceed as explained in Software Installation to update LabOne.

Figure 2.22: Device Connection dialog: LabOne update available

Figure 2.23: Download LabOne MSI using Automatic Update Check feature

2.6.3. Updating the Instrument Firmware

The LabOne software consists of both software that runs on your PC and software that runs on the instrument. In order to distinguish between the two, the latter will be called firmware for the rest of this document. When upgrading to a new software release, it's also necessary to update the instrument firmware.

If the firmware needs an update, this is indicated in the Device Connection dialog of the LabOne user interface under Windows.

In the Basic view of the dialog, there will be a button "Upgrade FW" appearing together with the instrument icon as shown in Figure 2.24. In the Advanced view, there will be a link "Upgrade FW" in the Update column of the Available Devices table. Click on **Upgrade FW** to open the firmware update start-up dialog shown in Figure 2.25. The firmware upgrade takes approximately 2 minutes.

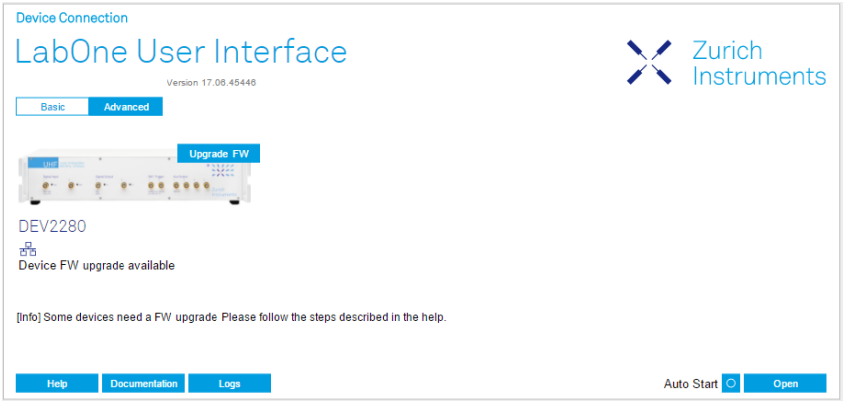


Figure 2.24: Device Connection dialog with available firmware update

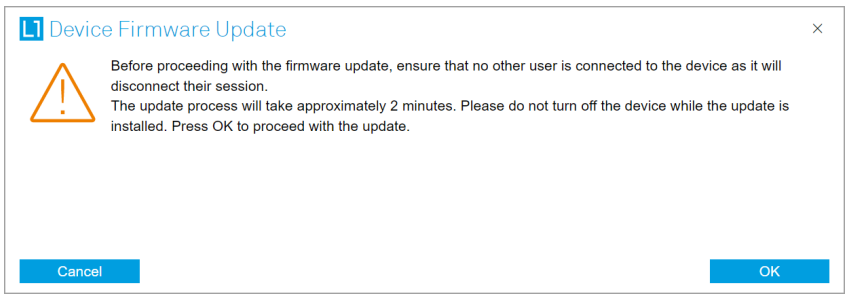


Figure 2.25: Device Firmware Update start-up dialog

Important

Do not disconnect the USB or 1GbE cable to the Instrument or power-cycle the Instrument during a firmware update.

If you encounter any issues while upgrading the instrument firmware, please contact Zurich Instruments at support@zhinst.com.

2.7. Troubleshooting

This section aims to help the user solve and avoid problems while using the software and operating the instrument.

2.7.1. Common Problems

Your UHF Series Instrument is an advanced piece of laboratory equipment which has many more features and capabilities than a traditional lock-in amplifier. In order to benefit from these, the user needs access to a large number of settings in the LabOne User Interface. The complexity of the settings might overwhelm a first-time user, and even expert users can get surprised by certain combinations of settings. To avoid problems, it's good to use the possibility to save and load settings in the Config Tab. This allows one to keep an overview by operating the instrument based on known configurations. This section provides an easy-to-follow checklist to solve the most common mishaps.

Table 2.8: Common Problems

Problem	Check item
The software cannot be installed or uninstalled	Please verify you have administrator/root rights.
The software cannot be updated	Please use the Modify option in Windows Apps & Features functionality. In the software installer select Repair, then uninstall the old software version, and install the new version.

Problem	Check item
The Instrument does not turn on	Please verify the power supply connection and inspect the fuse. The fuse holder is integrated in the power connector on the back panel of the instrument.
The Instrument has a high input noise floor (when connected to host computer by USB)	the USB cable connects the Instrument ground to computer ground, which might inject some unwanted noise to the measurements results. In this case it is recommended to use the Ethernet connection which is galvanically isolated using a UTP Cat 5 or 6 cable (UTP stands for "unshielded twisted pair").
The Instrument performs poorly at low frequencies (below 160 kHz with 50 Ω or below 100 Hz with 1 MΩ coupling) :	the signal inputs of the instrument might be set to AC operation. Please verify to turn off the AC switch in the Lock-in Tab or In / Out Tab.
The Instrument performs poorly during operation	the demodulator filters might be set too wide (too much noise) or too narrow (slow response) for your application. Please verify if the demodulator filter settings match your frequency versus noise plan.
The Instrument performs poorly during operation	clipping of the input signal may be occurring. This is detectable by monitoring the red LEDs on the front panel of the instrument or the Input Overflow (OVI) flags on the STATUS_TAB of the user interface. It can be avoided by adding enough margin on the input range setting (for instance 50% to 70% of the maximum signal peak).
The Instrument performs strangely when working with the UHF-MF Multi-frequency Option	it is easily possible to turn on more signal generators than intended. Check the generated Signal Output with the integrated oscilloscope and check the number of simultaneously activated oscillator voltages.
The Instrument performs close to specification, but higher performance is expected	After 2 years since the last calibration, a few analog parameters are subject to drift. This may cause inaccurate measurements. Zurich Instruments recommends re-calibration of the Instrument every 2 years.
The Instrument measurements are unpredictable	Please check the Status Tab to see if there is any active warning (red flag), or if one has occurred in the past (yellow flag).
The Instrument does not generate any output signal	verify that signal output switch has been activated in the Lock-in Tab or in the In / Out Tab.
The Instrument locks poorly using the digital I/O as reference	make sure that the digital input signal has a high slew rate and clean level crossings.
The Instrument locks poorly using the auxiliary analog inputs as reference	the input signal amplitude might be too small. Use proper gain setting of the input channel.
The sample stream from the Instrument to the host computer is not continuous	Check the communication (COM) flags in the status bar. The three flags indicate occasional sample loss, packet loss, or stall. Sample loss occurs when a sampling rate is set too high (the instrument sends more samples than the interface and the host computer can absorb). The packet loss indicates an important failure of the communications to the host computer and compromises the behavior of the instrument. Both problems are prevented by reducing the sample rate settings. The stall flag indicates that a setting was actively changed by the system to prevent UI crash.

Problem	Check item
The LabOne User Interface does not start	Verify that the LabOne Data Server (`ziDataServer.exe`) and the LabOne Web Server (`ziWebServer.exe`) are running via the Windows Task Manager. The Data Server should be started automatically by `ziService.exe` and the Web Server should be started upon clicking "Zurich Instruments LabOne" in the Windows Start Menu. If both are running, but clicking the Start Menu does not open a new User Interface session in a new tab of your default browser then try to create a new session manually by entering `127.0.0.1:8006` in the address bar of your browser.
The user interface does not start or starts but remains idle	Verify that the Data Server has been started and is running on your host computer.
The user interface is slow and the web browser process consumes a lot of CPU power	Make sure that the hardware acceleration is enabled for the web browser that is used for LabOne. For the Windows operating system, the hardware acceleration can be enabled in `Control Panel` → `Display` → `Screen Resolution`. Go to Advanced Settings and then Trouble Shoot. In case you use a NVIDIA graphics card, you have to use the NVIDIA control panel. Go to Manage 3D Settings, then Program Settings and select the program that you want to customize.

2.7.2. Location of the Log Files

The most recent log files of the LabOne Web and Data Server programs are most easily accessed by clicking on Logs in the LabOne Device Connection dialog of the user interface. The Device Connection dialog opens on software start-up or upon clicking on Session Manager in the Config tab of the user interface.

The location of the Web and Data Server log files on disk are given in the sections below.

Windows

The Web and Data Server log files on Windows can be found in the following directories.

- LabOne Data Server (`ziDataServer.exe`): C:\Windows\ServiceProfiles\LocalService\AppData\Local\Temp\Zurich Instruments\LabOne\ziDataServerLog
- LabOne Web Server (`ziWebServer.exe`): C:\Users[USER]\AppData\Local\Temp\Zurich Instruments\LabOne\ziWebServerLog

Note

The `C:\Users\[USER]\AppData` folder is hidden by default under Windows. A quick way of accessing it is to enter `%AppData%\..` in the address bar of the Windows File Explorer.

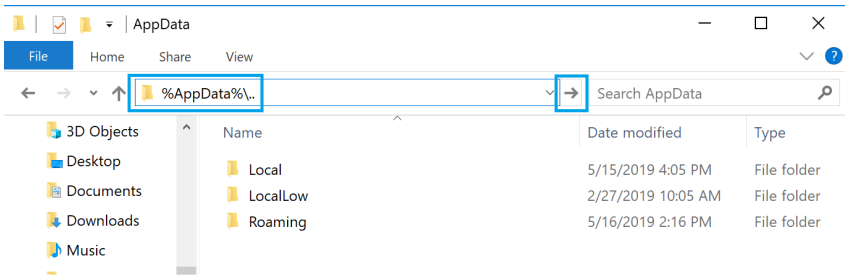


Figure 2.26: Using the

Linux and macOS

The Web and Data Server log files on Linux or macOS can be found in the following directories.

- LabOne Data Server (`ziDataServer`):
  /tmp/ziDataServerLog_[USER]
- LabOne Web Server (`ziWebServer`):
  /tmp/ziWebServerLog_[USER]

2.7.3. Prevent web browsers from sleep mode

It often occurs that an experiment requires a long-time signal acquisition; therefore, the setup including the measurement instrument and LabOne software are left unattended. By default, many web browsers go to a sleep mode after a certain idle time which results in the loss of acquired data when using the web-based user interface of LabOne for measurement. Although it is recommended to take advantage of LabOne APIs in these situations to automate the measurement process and avoid using web browsers for data recording, it is still possible to adjust the browser settings to prevent it from entering the sleep mode. Below, you will find how to modify the settings of your preferred browser to ensure a long-run data acquisition can be implemented properly.

Edge

1. Open **Settings** by typing `edge://settings` in the address bar
2. Select **System** from the icon bar.
3. Find the **Never put these sites to sleep** section of the **Optimized Performance** tab.
4. Add the IP address and the port of LabOne Webserver, e.g., `127.0.0.1:8006` or `192.168.73.98:80` to the list.

Chrome

1. While LabOne is running, open a tab in Chrome and type `chrome://discards` in the address bar.
2. In the shown table listing all the open tabs, find LabOne and disable its **Auto Discardable** feature.
3. This option avoids discarding and refreshing the LabOne tab as long as it is open. To disable this feature permanently, you can use an extension from the Chrome Webstore.

Firefox

1. Open **Advanced Preferences** by typing `about:config` in the address bar.
2. Look for `browser.tabs.unloadOnLowMemory` in the search bar.
3. Change it to **false** if it is **true**.

Opera

1. Open **Settings** by typing `opera://settings` in the address bar.
2. Locate the **User Interface** section in the **Advanced** view.
3. Disable the **Snooze inactive tabs to save memory** option and restart Opera.

Safari

1. Open **Debug** menu.
2. Go to **Miscellaneous Flags**.
3. Disable **Hidden Page Timer Throttling**.

2.8. UHFQA Firmware Upgrade Guide

Zurich Instruments LabOne consists of both software that runs on your PC and software (firmware) that runs on the UHFQA instrument. In order for the PC software to operate correctly it is necessary to update the UHFQA instrument's firmware when installing LabOne. This step only has to be performed once. The firmware upgrade process is integrated into the LabOne user interface. Alternatively, the firmware upgrade utility program can be used for this purpose. This guide explain how to upgrade the instrument firmware using this program.

2.8.1. Preparation

In order to upgrade the instrument firmware, you must first take the following steps:

1. Download and install the latest version of LabOne on your PC. Administrator rights are necessary for the installation. Please see the UHFQA User Manual.
2. Either start the UHF instrument or, if it was already running, switch off and restart the UHF instrument.
3. Connect the UHF to the PC with the LabOne installation via USB cable.

2.8.2. Starting the Firmware Upgrade UHFQA utility

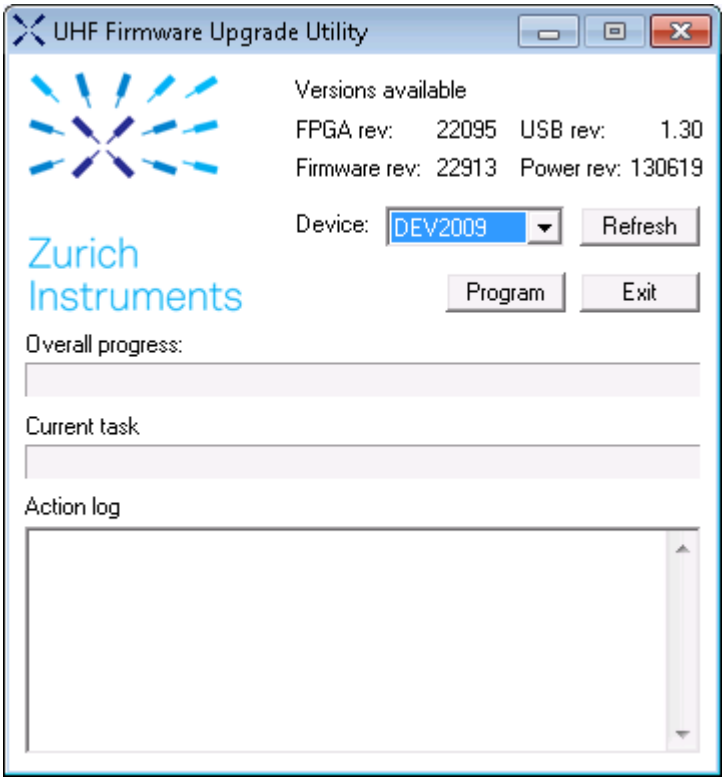
The Firmware Upgrade UHFQA utility is a program with graphical user interface used to perform a firmware upgrade and is included in the LabOne installation. Execute the file `uhfqa_flash.exe` in the LabOne installation folder (usually `C:\Program Files\Zurich Instruments\LabOne\DataServer\UHF_Flash`).

2.8.3. Using the Firmware Upgrade UHFQA utility

Important

Do not disconnect the USB cable to the UHF instrument or power-cycle the instrument whilst performing any of the following steps.

Upon starting the Firmware Upgrade UHFQA utility it should detect the device that is connected to the PC via USB. The device serial number is displayed next to `Device:`.



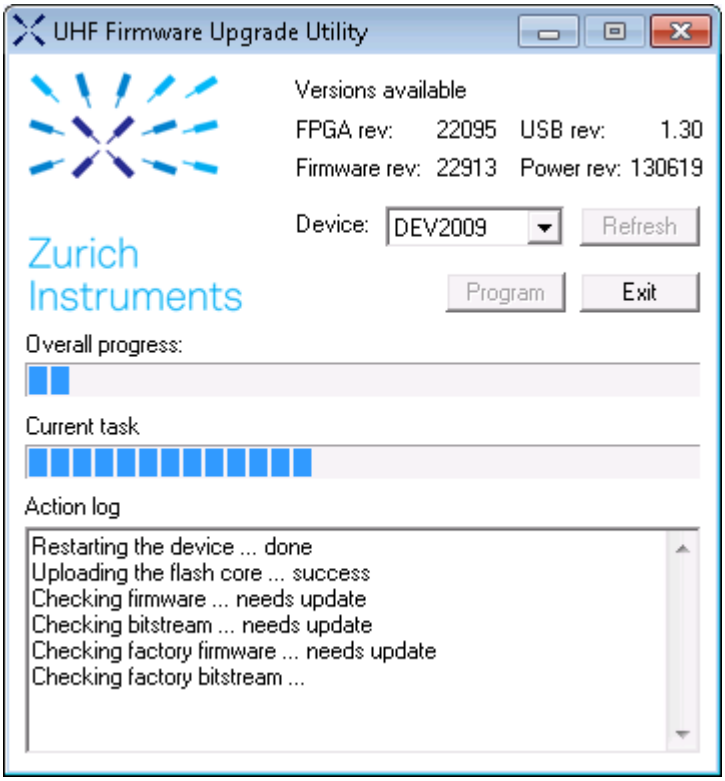
2.8.4. Select the device you would like to upgrade

Select which device you would like to upgrade via the pull-down menu. If no device is listed, please try the following steps:

1. Ensure that the USB cable is properly connected.
2. Try power-cycling the device.
3. Click the "Refresh" button.

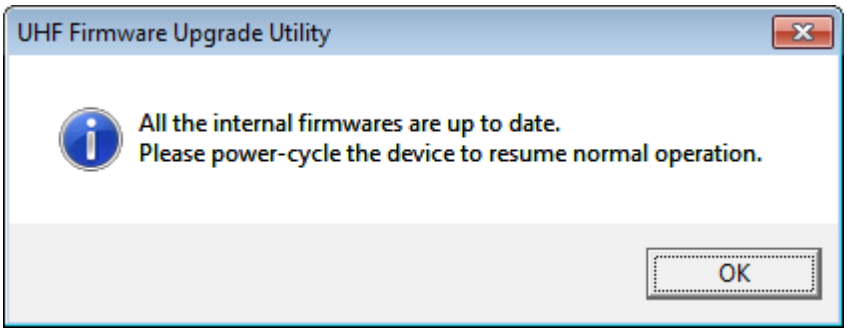
2.8.5. Program the firmware of the connected device

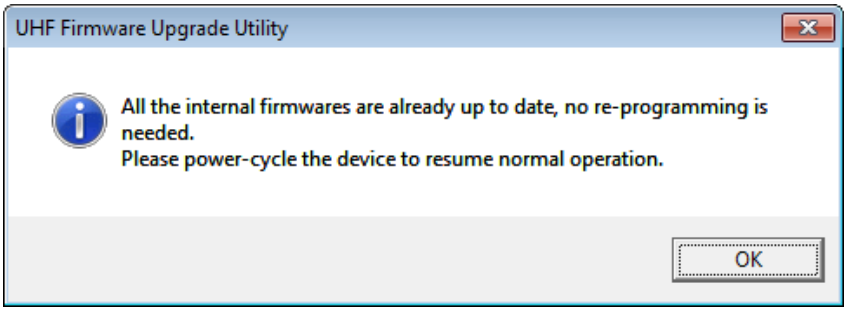
Click the "Program" button to check the version of the current firmware and install the new firmware on the device.



Important

After clicking "Program" and the update is finished it is always necessary to power-cycle the instrument to resume normal operation, even if the firmware was previously up-to-date.





2.8.6. Close the Firmware Upgrade UHFQA utility

Click the `Exit` button to close the Firmware Upgrade UHFQA utility.

2.8.7. Support

If you encounter any issues whilst upgrading the instrument firmware, please contact: support@zhinst.com.

3. Functional Overview

This chapter provides the overview of the features provided by the UHF Instrument. The first section contains the description of the graphical overview and the hardware and software feature list. The next section details the front panel and the back panel of the measurement instrument. The following section provides product selection and ordering support.

3.1. Features

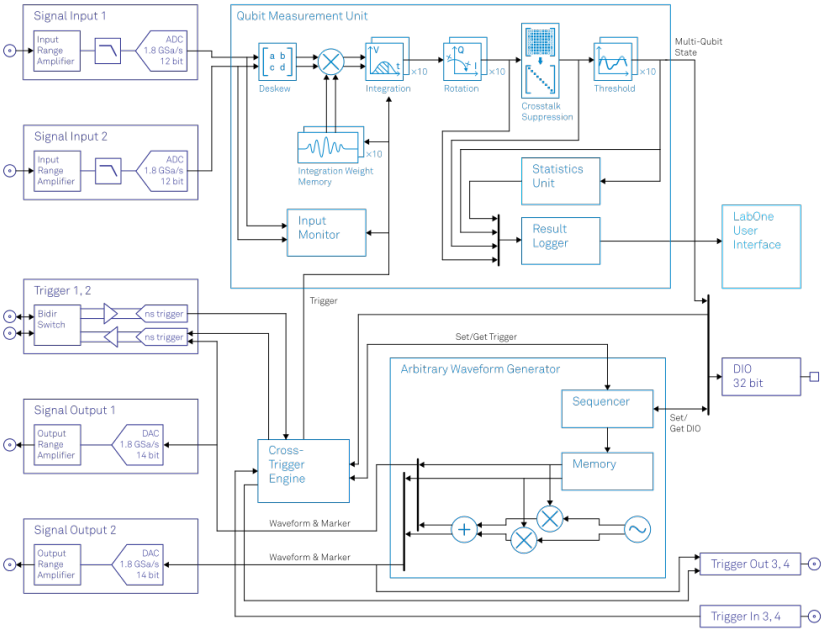


Figure 3.1: UHF instrument functional diagram

The UHF Instrument according to Figure 3.1 consists of several internal units (light blue color) surrounded by several interface units (dark blue color) and the front panel on the left-hand side and the back panel on the right-hand side. The arrows between the panels and the interface units indicates selected physical connections and the data direction flow.

The signal to be measured is usually connected to one of the two UHF signal inputs where it is amplified to a defined range and digitized at very high speed. The resulting samples are fed into the digital signal processor consisting of a qubit measurement unit with 10 parallel measurement channels . The output of the digital signal processor flow into a digital interface to be transferred to a host computer (LAN and USB interfaces) . Two low-distortion UHF signal outputs provide the signal generator functionality for arbitrary waveform generator output.

The Cross-Trigger Engine handles trigger exchange between the UHF-AWG Arbitrary Waveform Generator and the Qubit Measurement Unit. Internal triggers are used to trigger the Integration and Monitoring Scope.

Qubit Measurement Unit

- Dual-channel signal deskew
- 10 dual-channel integrators with programmable integration weight memory
- Complex rotation of integrated signals
- Crosstalk Suppression matrix
- Threshold operation
- Cross-correlation of integrated signals
- Cross-correlation of discretized measurement results
- Input Monitor averaging scope
- Statistics Unit for pattern analysis
- Result Logger

Ultra-high-frequency Signal Inputs

- 2 low-noise UHF inputs, single-ended, 600 MHz bandwidth
- Variable input range
- Switchable input impedance
- Selectable AC/DC coupling

Ultra-high-frequency Signal Outputs

- 2 low-distortion UHF outputs, single-ended, 600 MHz bandwidth
- Variable output range

AWG Features

- Sequencing and conditional branching
- Amplitude Modulation mode
- Parametric Sweeper
- High-level programming with LabOne AWG Sequencer
- 4-channel output mode

Auxiliary Input and Outputs

- 4 auxiliary outputs, user defined signals
- 2 auxiliary inputs, general purpose

High-speed Connectivity

- USB 2.0 high-speed 480 Mbit/s host interface
- LAN 1 Gbit/s controller interface
- DIO: 32-bit digital input-output port
- ZCtrl: 2 ports peripheral control
- Clock input connector (10 MHz)
- Clock output connector (10 MHz)

Software Features

- Web-based, high-speed user interface with multi-instrument control
- Data server with multi-client support
- API for C, LabVIEW, MATLAB, Python based instrument programming

3.2. Front Panel Tour

The front panel BNC connectors and control LEDs are arranged as shown in Figure 3.2 and listed in Table 3.1.

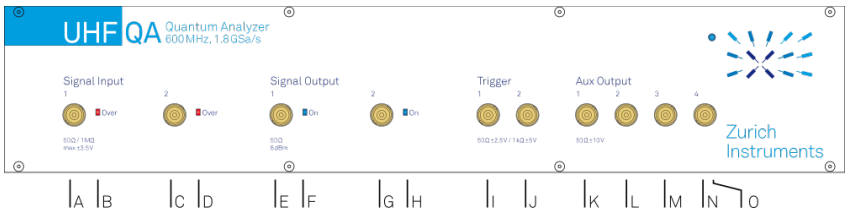


Figure 3.2: UHFQA Quantum Analyzer front panel

Table 3.1: UHF Instrument front panel description

Position	Label / Name	Description
A	Signal Input 1	single-ended UHF input
B	Signal Input 1 Over	this red LED indicates that the input signal saturates the A/D converter and therefore the input range must be increased or the signal must be attenuated
C	Signal Input 2	single-ended UHF input

Position	Label / Name	Description
D	Signal Input 2 Over	this red LED indicates that the input signal saturates the A/D converter and therefore the input range must be increased or the signal must be attenuated
E	Signal Output 1	single-ended UHF output
F	Signal Output 1 ON	this blue LED indicates that the signal output is actively driven by the instrument
G	Signal Output 2	single-ended UHF output
H	Signal Output 2 ON	this blue LED indicates that the signal output is actively driven by the instrument
I	Trigger 1	analog reference input, TTL reference output, AWG marker output, or bidirectional digital TTL trigger
J	Trigger 2	analog reference input, TTL reference output, AWG marker output, or bidirectional digital TTL trigger
K	Aux Output 1	this connector provides a user defined signal, often used to output lock-in signals X, Y, R, Θ, or to output AWG signals
L	Aux Output 2	this connector provides a user defined signal, often used to output lock-in signals X, Y, R, Θ, or to output AWG signals
M	Aux Output 3	this connector provides a user defined signal, often used to output lock-in signals X, Y, R, Θ, or to output AWG signals
N	Aux Output 4	this connector provides a user defined signal, often used to output lock-in signals X, Y, R, Θ, or to output AWG signals
O	Power	this LED indicates that the instrument is powered
		color blue: the device has an active connection over USB or Ethernet
		color orange: indicates ready to connect. The device is ready for connection over USB or Ethernet. The internal auto calibration process is also indicated by an orange LED
		color orange blinking: device is in start-up mode and waiting for an IP address. As long as the device does not have a dynamic IP address or does use its static default address a connection attempt over Ethernet will fail

3.3. Back Panel Tour

The back panel is the main interface for power, control, service and connectivity to other ZI instruments. Please refer to Figure 3.3 and Table 3.2 for the detailed description of the items.

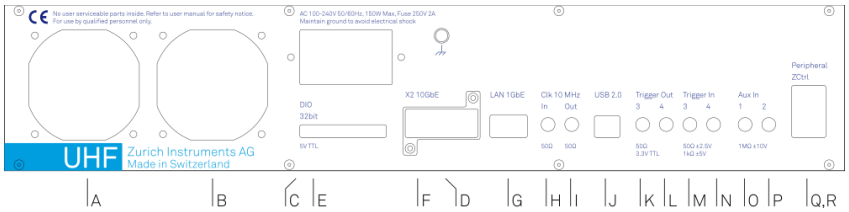


Figure 3.3: UHF Instrument back panel

Table 3.2: UHF Instrument back panel description

Position	Label / Name	Description
A	-	ventilator (important: keep clear from obstruction)
B	-	ventilator (important: keep clear from obstruction)

Position	Label / Name	Description
C	Power inlet	power inlet with ON/OFF switch
D	Earth ground	4 mm banana jack connector for earth ground, electrically connected to the chassis and the earth pin of the power inlet
E	DIO	32-bit digital input/output connector
F	X2 10GbE	10 Gbit LAN connector
G	LAN 1GbE	1 Gbit LAN connector
H	Clk 10 MHz In	clock input (10 MHz) for synchronization with other instruments
I	Clk 10 MHz Out	clock output (10 MHz) for synchronization with other instruments
J	USB	universal serial bus host computer connection
K	Trigger Out 3	digital TTL trigger and AWG marker output - note: some UHF Instruments indicate Trigger 1 on the back panel instead of Trigger 3
L	Trigger Out 4	digital TTL trigger and AWG marker output - note: some UHF Instruments indicate Trigger 2 on the back panel instead of Trigger 4
M	Trigger In 3	digital trigger input - note: some UHF Instruments indicate Trigger 1 on the back panel instead of Trigger 3
N	Trigger In 4	digital trigger input - note: some UHF Instruments indicate Trigger 2 on the back panel instead of Trigger 4
O	Aux In 1	auxiliary input
P	Aux In 2	auxiliary input
Q	ZCtrl 1	peripheral pre-amplifier power & control bus. Attention: this is not an Ethernet plug, connection to an Ethernet network might damage the instrument
R	ZCtrl 2	peripheral pre-amplifier power & control bus. Attention: this is not an Ethernet plug, connection to an Ethernet network might damage the instrument

3.4. Ordering Guide

Table 3.3 provides an overview of the available UHFQA products. Upgradeable features are options that can be purchased anytime without need to send the Instrument to Zurich Instruments.

Table 3.3: UHFQA product codes for ordering

Product code	Product name	Description	Field upgrade possible
UHFQA	UHFQA Quantum Analyzer	base product	-
UHF-AWG	UHF-AWG Arbitrary Waveform Generator	functionality included in base product	-
UHF-DIG	UHF-DIG Digitizer	option	yes
UHF-RUB	UHF-RUB Rubidium Atomic Clock	option	no

3.5. DIOLink cable set

When using the HDAWG and the UHFQA in a Zurich Instruments Quantum Computing Control System, the two instruments need to be connected with a VHDCI cable between their respective DIO ports. Due to the different voltage levels of the DIO connectors on the HDAWG and UHFQA, a passive voltage level shifter needs to be placed in series with the VHDCI connection. It is furthermore recommended to use a mechanical stabilization of the VHDCI connection on the HDAWG side in order to prevent damage to the DIO connector due to cable pull. A set of a VHDCI cable, a level

shifter, and a mechanical stabilization part are available as DIO Link Set from Zurich Instruments upon request under support@zhinst.com. The DIO Link Set is shown in the the figure below.

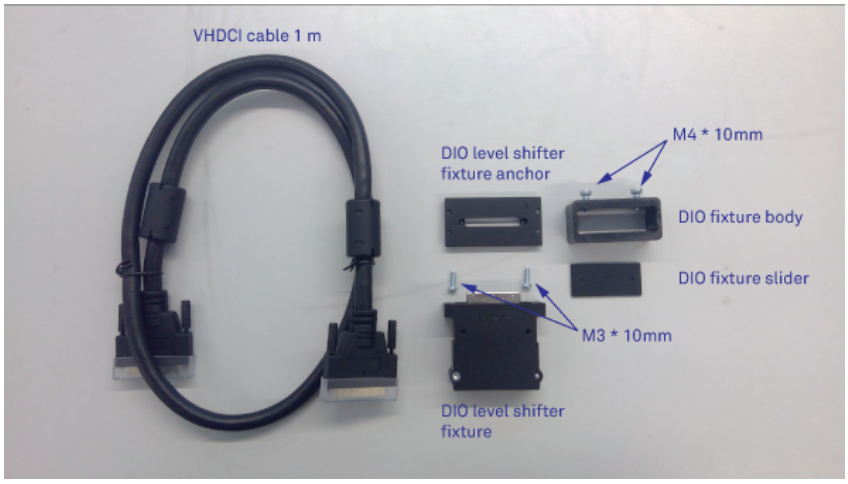


Figure 3.4: DIO link set

Please follow the instructions below in order to install the DIO Link Set between the HDAWG and the UHFQA.

1. Fix the DIO level shift fixture anchor to the DIO connector on the UHFQA using the screws already installed on the UHFQA DIO port.
2. Fix the DIO level shifter fixture to the anchor using the two M3*10 mm screws.
3. Clamp the DIO fixture (slider and body) lightly to one of the connectors of the VHDCI cable using the two M4*10 mm screws. This side of the cable is to be connected to the HDAWG.
4. Connect the VHDCI cable between the DIO level shifter fixture and the DIO port on the HDAWG and tighten the manual screws on both VHDCI connectors.
5. Push the DIO fixture (slider and body) firmly towards the HDAWG casing and tighten the two M4*10 mm screws to increase the mechanical stability of the VHDCI cable connection.

4. Tutorials

The tutorials in this chapter have been created to allow users to become more familiar with the basic technique of lock-in amplification, the operation of host-based lock-in amplifiers, the LabOne web browser based user interface, as well as some more advanced lock-in measurement techniques. In order to successfully carry out the tutorials, users are required to have certain laboratory equipment and basic equipment handling knowledge. The equipment list is given below.

Note

For all tutorials, you must have LabOne installed as described in the Getting Started.

- 1 USB 2.0 cable, 1 LAN cable (supplied with your UHF Instrument)
- 3 BNC cables
- SMA cable and adaptors
- 1 male BNC shorting cap (optional)
- 1 oscilloscope (optional)
- 1 BNC T-piece (optional)
- 1 resonator (for the PLL tutorial)

4.1. Arbitrary Waveform Generator

Note

This tutorial is applicable to UHFQA Instruments. Where indicated, the UHF-DIG option is required in addition.

4.1.1. Goals and Requirements

The goal of this tutorial is to demonstrate the basic use of the AWG. We demonstrate waveform generation and playback, triggering and synchronization, carrier modulation, and sequence branching. We conclude with a list of tips for operating the AWG. To perform the measurements in this tutorial, one will require a 3rd-party programmable arbitrary waveform/function generator for trigger generation.

4.1.2. Preparation

Connect the cables as illustrated below. Make sure that the UHF unit is powered on and connected by USB to your host computer or by Ethernet to your local area network (LAN) where the host computer resides. After starting LabOne, the default web browser opens with the LabOne graphical user interface.

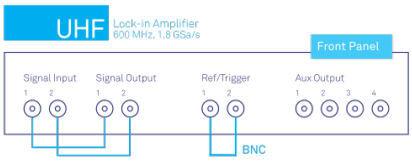


Figure 4.1: UHF connections for the arbitrary waveform generator tutorial

The tutorial can be started with the default instrument configuration (e.g. after a power cycle) and the default user interface settings (e.g. as is after pressing F5 in the browser).

4.1.3. Waveform Generation and Playback

In this tutorial we generate arbitrary signals with the AWG and visualize them with the Scope. In a first step we enable the Signal Outputs, but disable all sinusoidal signals generated by the lock-in

unit by default. We also configure the Scope signal input and triggering and arm it by clicking on Run/Stop in the Scope. The following table summarizes the necessary settings.

Table 4.1: Settings: enable the output and configure the Scope

Tab	Sub-tab	Section	#	Label	Setting / Value / State
In/Out		Signal Outputs	1	Enable	ON
In/Out		Signal Outputs	2	Enable	ON
Lock-in	All	Output Amplitudes	1-8	Amp 1 Enable	OFF
Lock-in	All	Output Amplitudes	1-8	Amp 2 Enable	OFF
Scope	Control	Vertical		Channel 1	Signal Input 1
Scope	Trigger	Trigger		Enable	ON
Scope	Trigger	Trigger		Signal	Signal Input 1
Scope	Trigger	Trigger		Level	0.1 V
Scope	Control			Run/Stop	ON

Figure 4.2: LabOne UI: AWG tab

In the AWG tab, we configure both channels to output signals at the full scale (FS) in plain output mode as summarized in the following table.

Table 4.2: Settings: configure the AWG output

Tab	Sub-tab	Section	#	Label	Setting / Value / State
AWG	Control			Rate (Sa/s)	1.8 GHz
AWG	Control			Rerun	OFF
AWG	Control	Output 1		Amplitude (FS)	1.0
AWG	Control	Output 1		Mode	Plain
AWG	Control	Output 2		Amplitude (FS)	1.0
AWG	Control	Output 2		Mode	Plain

Operating the AWG means first of all to specify a sequence program. This can be done interactively by typing the program in the Sequence Editor window. Let's start by typing the following code into the Sequence Editor.

```
wave w_gauss = 1.0*gauss(8000, 4000, 1000);
playWave(1, w_gauss);
```

In the first line of the program, we generate a waveform with a Gaussian shape with a length of 8000 samples and store the waveform under the name `w_gauss`. The peak center position 4000 and the standard deviation 1000 are both defined in units of samples. You can convert them into time by dividing by the chosen Rate (1.8 GSa/s by default). The waveform generated by the `gauss` function has a peak amplitude of 1. This amplitude is dimensionless and the physical signal amplitude is given by this number multiplied with the signal output range (e.g. 1.5 V). We put a scaling factor of 1.0 in place which can be replaced by any other value below 1. The code line is terminated by a semicolon according to C conventions. In the second line, the generated waveform `w_gauss` is played on AWG Output 1.

Note

For this tutorial, we will keep the description of the Sequencer commands short. You can find the full specification of the LabOne Sequencer language in LabOne Sequence Programming

Note

The AWG has a waveform granularity of 16 samples. It's recommended to use waveform lengths that are multiples of 16, e.g. 8000 like in this example, to avoid having ill-defined samples between successively played waveforms. Other waveform lengths are allowed, though.

If we now click on Save, the program gets compiled. This means the program is translated into instructions for the LabOne Sequencer on the UHF instrument, see AWG Tab. If no error occurs (due to wrong program syntax, for example), the Status LED lights up green, and the resulting program as well as the waveform data is written to the instrument memory. If an error or warning occurs, messages in the Status field will help in debugging the program. If we now have a look at the Waveform sub-tab, we see that our Gaussian waveform appeared in the list. The Memory Usage field at the bottom of the Waveform sub-tab shows what fraction of the instrument memory is filled by the waveform data. The Waveform viewer sub-tab allows you to graphically display the currently marked waveform in the list.

By clicking on Start with Rerun disabled, we have the AWG execute our program once. Since we have armed the Scope previously with a suitable trigger level, it has captured our Gaussian pulse with a FWHM of about 1.33 µs as shown in Figure 4.3.

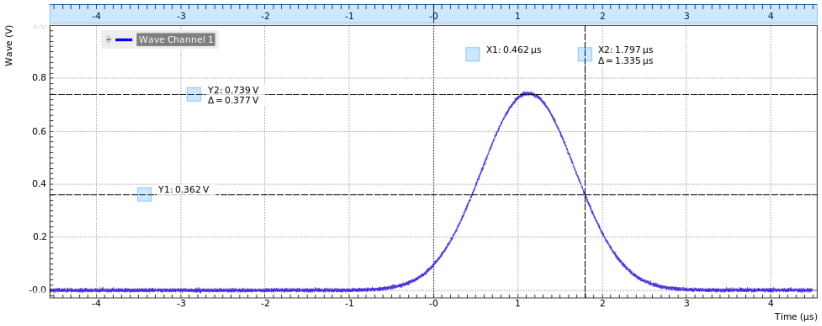


Figure 4.3: Gaussian pulse as generated by the AWG and captured by the LabOne Scope

The LabOne Sequencer language contains various control structures. The basic functionality is to play back a waveform several times. In the following example, all the code within the curly brackets '{...}' is repeated 5 times. Upon clicking Save and Start you should observe 5 short Gaussian pulses in a new scope shot, see Figure 4.4 .

```
wave w_gauss = 1.0 * gauss(640, 320, 50);

repeat (5) {
  playWave(1, w_gauss);
}
```

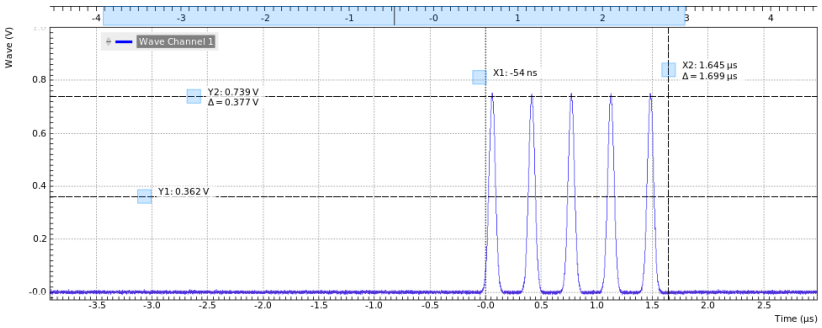


Figure 4.4: Burst of Gaussian pulses generated by the AWG and captured by the LabOne Scope

In order to generate more complex waveforms, the LabOne Sequencer programming language offers a rich toolset for waveform editing. On the basis of a selection of standard waveform generation functions, waveforms can be added, multiplied, scaled, concatenated, and truncated. It's also possible to use compile-time evaluated loops to generate pulse series with systematic parameter variations – see LabOne Sequence Programming for more precise information. In the following code example, we make use of these tools to generate a pulse with a smooth rising edge, a flat plateau, and a smooth falling edge. We use the `cut` function to cut a waveform at defined sample indices, the `rect` function to generate a waveform with constant level 1.0 and length 320, and the `join` function to concatenate three (or arbitrarily many) waveforms.

```
wave w_gauss = gauss(640, 320, 50);
wave w_rise = cut(w_gauss, 0, 319);
wave w_fall = cut(w_gauss, 320, 639);
wave w_flat = rect(320, 1.0);

wave w_pulse = join(w_rise, w_flat, w_fall);

while (true) {
  playWave(1, w_pulse);
}
```

Note that we replaced the finite repetition by an infinite repetition by using a `while` loop. Loops can be nested in order to generate complex playback routines. The Rerun button in the Control sub-tab will simply cause the entire AWG program to be restarted automatically. This is a simple alternative for creating loops, however, unlike the `while` loop the Rerun method does not allow back-to-back waveform playback and comes with a certain timing jitter. The output generated by the program above is shown in Figure 4.5.

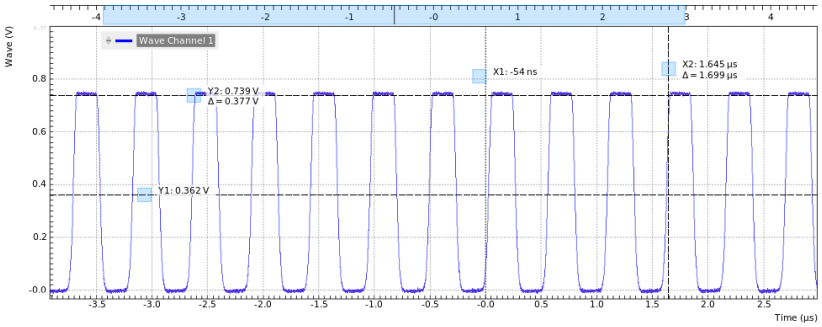


Figure 4.5: Infinite pulse series generated by the AWG and captured by the LabOne Scope

One pitfall when using loops has to do with the nature of the `playWave` and related commands. This command initiates the waveform playback, but during the playback the sequencer will start to execute the next command on his list. This is useful as it allows you to execute commands during playback. In a loop, it means the sequencer can jump back to the beginning of the loop while the waveform is still being played. You can easily change this behavior by adding `waitWave` as the last command in the loop.

As programs get longer, it becomes useful to store and recall them. Clicking on allows Save as... you to store the present program under a newly chosen file name. Clicking on Save then saves your program to the file name displayed at the top of the editor. As you begin to work on sequence programs more regularly, it's worth expanding your repertoire by some of the editor keyboard shortcuts listed in Sequence Editor Keyboard Shortcuts.

It's also possible to iterate over the samples of a waveform array and calculate each one of them in a loop over a compile-time variable `cvar`. This often allows to go beyond the possibilities of using the predefined waveform generation function, particularly when using nested formulas of elementary functions like in the following example. The waveform array needs to be pre-allocated e.g. using the instruction `zeros`.

```
const N = 1024;
const width = 100;
const position = N/2;
const f_start = 0.1;
const f_stop = 0.2;
cvar i;
wave w_array = zeros(N);
for (i = 0; i < N; i++) {
```

```
  w_array[i] = sin(10/(cosh((i-position)/width)));
}

playWave(w_array);
```

Should you require more customization than what is offered by the LabOne AWG Sequencer language, you can import any waveform from a comma-separated value (CSV) file. The CSV file should contain floating-point values in the range from –1.0 to +1.0 and contain one or several columns, corresponding to the number of channels. As an example, the following could be the contents of a file `wave_file.csv` specifying a dual-channel wave with a length of 16 samples:

```
  -1.0   0.0
  -0.8   0.0
  -0.7   0.1
  -0.5   0.2
  -0.2   0.3
  -0.1   0.2
 0.1   0.0
 0.2    -0.1
 0.7    -0.3
 1.0    -0.2
 0.9    -0.3
 0.8    -0.2
 0.4    -0.1
 0.0    -0.1
  -0.5    -0.1
  -0.8   0.0
```

Store the file in the location of `C:\Users\<user name>\Documents\Zurich Instruments\LabOne\WebServer\awg\waves\wave_file.csv` under Windows or `~/Zurich Instruments/LabOne/WebServer/awg/waves/wave_file.csv` under Linux. In the sequence program you can then play back the wave by referring to the file name without extension:

```
playWave("wave_file");
```

If you prefer, you can also store it in a `wave` data type first and give it a new name:

```
wave w = "wave_file";
playWave(w);
```

The external wave file can have arbitrary content, but consider that the final signal will pass through the 600 MHz low-pass filter of the instrument. This means that signal components exceeding the filter bandwidth are not reproduced exactly as suggested for example by looking at a plot of the waveform data. In particular, this concerns sharp transitions from one sample to the next.

In order to obtain digital marker data (see below) from a file, specify a second wave file with integer instead of floating-point values. The marker bits are encoded in the binary representation of the integer (i.e., integer 1 corresponds to the first marker high, 2 corresponds to the second marker high, and 3 corresponds to both bits high). Later in the program add up the analog and the marker waveforms. For instance, if the floating-point analog data are contained in `wave_file_analog.csv` and the integer marker data in `wave_file_digital.csv`, the following code can be used to combine and play them.

```
wave w_analog  = "wave_file_analog";
wave w_digital = "wave_file_digital";
wave w = w_analog + w_digital;
playWave(w);
```

As an alternative to specifying analog data as floating-point values in one file, and marker data as integer values in a second file, both may be combined into one file containing integer values that correspond to the raw data format of the instrument. The integer values in that file should be 16-bit unsigned integers with the two least significant bits being the markers. The values are mapped to 0 ⇒ -FS, 65535 ⇒ +FS, with FS equal to the full scale.

4.1.4. Triggering and Synchronization

Now we have a look at the triggering functionality of the AWG. In this section we will explain how to deal with the most important use cases:

- Triggering the AWG with an external TTL signal
- Generating a TTL signal with the AWG to trigger an external device
- Control the AWG repetition rate by an internal oscillator

We will simulate these situations with on-board means of the UHF instrument for the sake of simplicity, but the inclusion of external equipment is straightforward in practice.

The AWG's trigger channels can be freely linked to a variety of connectors, such as the bidirectional Trigger connectors on the front panel, and other functional units inside the instrument, such as the Scope or the Qubit Measurement Unit . This freedom of configuration is enabled by the Cross-Domain Trigger feature and enables triggering.

Triggering the AWG

In this section we show how to trigger the AWG with an external TTL signal. Please use a third-party instrument, such as a function generator, to generate a square wave with 300 kHz frequency and connect this signal to the Trigger Input 1 on the front panel.

The AWG has 4 trigger input channels. As discussed, these are not directly associated with physical device inputs but can be freely configured to probe a variety of internal or external signals. Here, we link the AWG Digital Trigger 1 to the physical Trigger 1 connector.

Table 4.3: Settings: configure the AWG analog trigger input

Tab	Sub-tab	Section	#	Label	Setting / Value / State
AWG	Trigger	Digital Trigger 1		Signal	Trig Input 1
AWG	Trigger	Digital Trigger 1		Slope	Rise

Finally, we modify our last AWG program by including a **`waitDigTrigger`** command just before the **`playWave`** command. The result is that upon every repetition inside the infinite **`while`** loop, the AWG will wait for a rising flank on Trigger input 1.

```
wave w_gauss = gauss(640, 320, 50);
wave w_rise  = cut(w_gauss, 0, 319);
wave w_fall  = cut(w_gauss, 320, 639);
wave w_flat  = rect(320, 1.0);

wave w_pulse = join(w_rise, w_flat, w_fall);

while (true) {
  waitDigTrigger(1, 1);
  playWave(1, w_pulse);
}
```

Compile and run the above program. Note that this and other programming examples are available directly from a drop-down menu on top of the Sequence Editor. Figure 4.6 shows the pulse series as seen in the Scope: the pulses are now spaced by the oscillator period of about 3.3 µs, unlike previously when the period was determined by the length of the waveform **`w_pulse`**. Try changing the frequency of the external trigger signal , or unplugging the trigger cable, to observe the immediate effect on the signal.

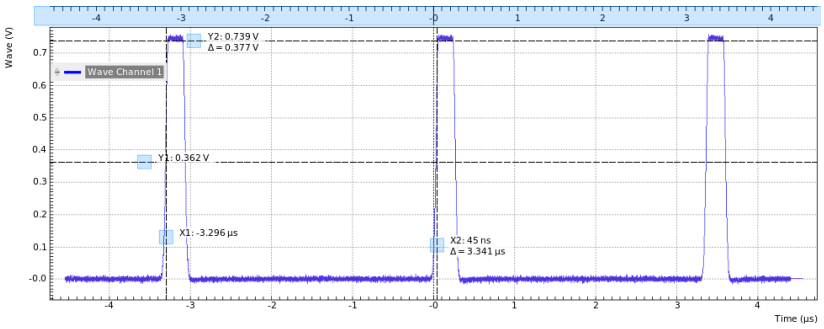


Figure 4.6: Externally triggered pulse series generated by the AWG and captured by the LabOne Scope

Generating Triggers with the AWG

There are two ways of generating trigger output signals with the AWG: as markers, or through sequencer commands.

The method using markers is recommended when precise timing is required, and/or complicated serial bit patterns need to be played on the trigger outputs. Marker bits are part of every waveform which is an array of 16-bit words: 14 bits of each word represent the analog waveform data, and the remaining 2 bits represent two digital marker channels. Upon playback, a digital signal with sample-precise alignment with the analog output is generated.

The method using a sequencer command is simpler, but the timing control is less flexible than when using markers. It is useful for instance to generate a single trigger signal at the start of an AWG program.

Table 4.4: Comparison: AWG markers and triggers

	Marker	Trigger
Implementation	Part of waveform	Sequencer command
Timing control	High	Low
Generation of serial bit patterns	Yes	No
Cross-device synchronization	Yes	Yes

Let us first demonstrate the use of markers. In the following code example we first generate a Gaussian pulse again. The so generated wave does include marker bits – they are simply set to zero by default. We use the `marker` function to assign the desired non-zero marker bits to the wave. The `marker` function takes two arguments, the first is the length of the wave, the second is the marker configuration in binary encoding: the value 0 stands for a both marker bits low, the values 1, 2, and 3 stand for the first, the second, and both marker bits high, respectively. We use this to construct the wave called `w_marker`.

```
const marker_pos = 3000;

wave w_gauss  = gauss(8000, 4000, 1000);
wave w_left   = marker(marker_pos, 0);
wave w_right  = marker(8000-marker_pos, 1);
wave w_marker = join(w_left, w_right);
wave w_gauss_marker = w_gauss + w_marker;

playWave(1, w_gauss_marker);
```

The waveform addition with the '+' operator adds up analog waveform data but also combines marker data. The wave `w_gauss` contains zero marker data, whereas the wave `w_marker` contains zero analog data. Consequentially the wave called `w_gauss_marker` contains the merged analog and marker data. We use the integer constant `marker_pos` to determine the point where the first marker bit flips from 0 to 1 somewhere in the middle of the Gaussian pulse.

Note

The **add** function and the '+' operator combine marker bits by a logical OR operation. This means combining 0 and 1 yields 1, and combining 1 and 1 yields 1 as well.

The following table summarizes the settings to apply in order to output marker 1 on Trigger 2, and to configure the scope to trigger on Trigger 1.

Table 4.5: Settings: configure the AWG marker output and scope trigger

Tab	Sub-tab	Section	#	Label	Setting / Value / State
DIO		Output	2	Signal	AWG Marker 1
DIO		Output	2	Drive	ON
Scope	Trigger	Trigger		Signal	Trig Input 1

Figure 4.7 shows the AWG signal captured by the Scope. The green curve shows the second Scope channel (requires UHF-DIG option) configured to display the Trigger Input 1 signal. Try changing the **marker_pos** constant and re-running the sequence program to observe the effect on the temporal alignment of the Gaussian pulse.

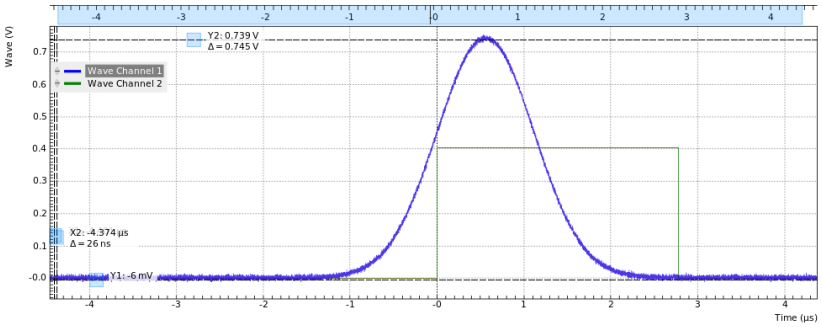


Figure 4.7: Pulse and marker signal generated by the AWG and captured by the LabOne Scope (dual-channel Scope operation requires UHF-DIG option)

Let us now demonstrate the use of sequencer commands to generate a trigger signal. Copy and paste the following code example into the Sequence Editor.

```
wave w_gauss = gauss(8000, 4000, 1000);

setTrigger(1);
playWave(1, w_gauss);
waitWave();
setTrigger(0);
```

The **setTrigger** function takes a single argument encoding the four AWG Trigger output states in binary form – the integer number 1 corresponds to a configuration of 0/0/0/1 for the trigger outputs 4/3/2/1. The binary integer notation of the form 0b0000 is useful for this purpose – e.g. **setTrigger(0b0011)** will set trigger outputs 1 and 2 to 1, and trigger outputs 3 and 4 to 0. We included a **waitWave** command after the **playWave** command. It ensures that the subsequent **setTrigger** command is executed only after the Gaussian wave has finished playing, and not during waveform playback.

We reconfigure the Trigger 2 connector such that it outputs the AWG Trigger 1, instead of the AWG Marker 1. The rest of the settings can stay unchanged.

Table 4.6: Settings: configure the AWG trigger output

Tab	Sub-tab	Section	#	Label	Setting / Value / State
DIO		Output	2	Signal	AWG Trigger 1

Figure 4.8 shows the AWG signal captured by the Scope. This looks very similar to Figure 4.7 in fact. With this method, we're less flexible in choosing the trigger time, as the rising trigger edge will always be at the beginning of the waveform. But we don't have to bother about assigning the marker bits to the waveform.

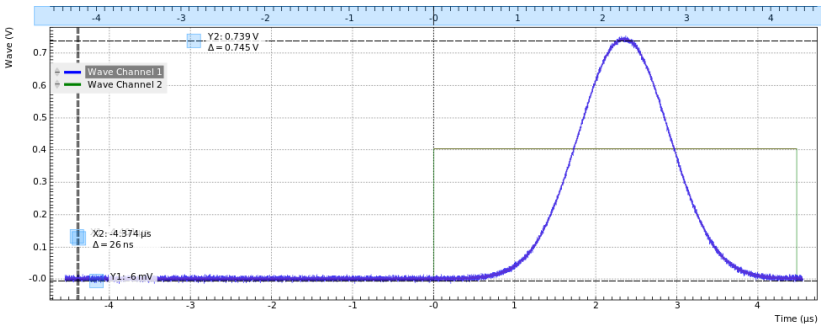


Figure 4.8: Pulse and trigger signal generated by the AWG and captured by the LabOne Scope (dual-channel Scope operation requires UHF-DIG option)

4.1.5. Signal Output Assignment

In addition to the single-channel and dual-channel playback used up to now, there are more options for the channel assignment. The `playWave` command can be used with different combinations of arguments: with one `wave` type argument or with two, with a `const` type integer number specifying the signal output or without. These different combinations of arguments allow the user to independently control the AWG outputs (the digital signal sources inside the instrument) and the place where their signal is routed to (the signal outputs on the front panel). The AWG outputs are represented in the AWG tab, whereas the signal outputs are represented in the In / Out tab .

The `playWave` command always assigns the first `wave` argument to the AWG output 1, and the second one (if it's provided) to the AWG output 2. Each of the `wave` arguments can optionally be preceded by an integer argument of type `const` which specifies the associated signal output. E.g., `playWave(2, w_gauss)` will play the wave `w_gauss` on Signal Output 2.

It's possible to route a single AWG Output to both Signal Outputs at the same time by specifying two integer arguments per wave argument as in `playWave(1, 2, w_gauss)`. This can for example be used to optimize waveform memory. Another option is to add up two AWG Outputs on one Signal Output by using twice the same integer argument as in `playWave(1, w_gauss, 1, w_drag)`. The following sequence program contains a number of examples for these configurations. Figure 4.9 shows the dual-channel signal generated with this program and measured with the LabOne Scope.

```
wave w_gauss = gauss(640, 320, 50);

while (true) {
  waitDemodOscPhase(8);
  playWave(1, w_gauss);
}
```

Note

Tricky examples are commands like `playWave(2, w_gauss)` that generate a signal on Signal Output 2, but use the AWG output 1. This means that the relevant Mode and Amplitude (FS) settings are in the section Output 1 of the AWG tab, not in the section Output 2.

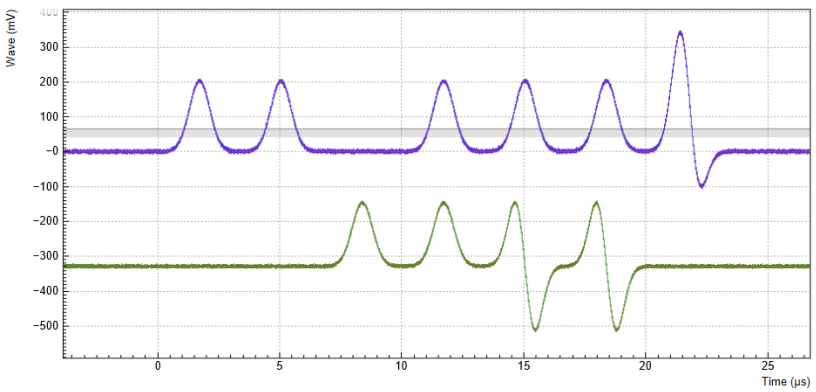


Figure 4.9: Dual-channel signal generated by the AWG and captured by the LabOne Scope (dual-channel Scope operation requires UHF-DIG option). The vertical scales of the two signals have been offset, see

4.1.6. Four-channel Aux Output

For applications requiring more channels and/or higher voltages, the UHF-AWG can generate signals on the auxiliary outputs of the UHF instrument. To this end the AWG resources for one fast channel (1.8 GSa/s) can be reallocated so as to generate four independent signals at 14 MSa/s and 16 bit resolution in a ±10 V range.

In the sequence program, the functionality is available through the `playAuxWave` function. The function requires four waveforms of equal length as arguments.

We configure the multi-purpose Auxiliary Outputs for AWG signal generation by setting the Auxiliary Output Signal to AWG in the Auxiliary tab. Each Auxiliary Output corresponds to one of the four rows in the tab. The Channel setting allows you to route one of the four AWG Outputs (the four waveforms of the `playAuxWave` command) to the given Auxiliary Output. Typically for the first row the Channel is set to 1, for the second row to 2, and so forth.

We intend to monitor the individual Auxiliary signals with the Scope on Signal Input 1. Before making the corresponding BNC connections, it's good practice to adjust the Auxiliary Output Lower and Upper Limits in order to prevent damage to the Signal Input. You can use the Scale and the Offset setting in order to modify the signal.

The Auxiliary Outputs have a much lower analog bandwidth than the Signal Outputs. It is therefore necessary to work at a sampling rate of 14 MSa/s or less. In order to combine slow Auxiliary Output signals with fast signals on the Signal Outputs, it's useful to set the sampling rate for every individual waveform play command. In the following example, we first play four waveforms in parallel on the Auxiliary Outputs at reduced sampling rate, and then one waveform on the Signal Output 2 at full sampling rate.

```
// Sampling rate of the system, adjust accordingly if the rate is reduced
const FS = 1800e6;
// Frequency of the 'sine' in the SINC waveform
const F_SINC = 42e6;

// Generate the four-channel auxiliary output waveform
wave aux_ch1 =  1.0*gauss(8000, 4000, 1000);
wave aux_ch2 =  0.5*gauss(8000, 4000, 1000);
wave aux_ch3 = -0.5*gauss(8000, 4000, 1000);
wave aux_ch4 = -1.0*gauss(8000, 4000, 1000);

// Generate a waveform to be played on Signal Output 2
wave w_sinc = sinc(8000, 4000, FS/F_SINC);

while (true) {
  // play the four Aux Output channels at reduced rate
  playAuxWave(aux_ch1, aux_ch2, aux_ch3, aux_ch4, AWG_RATE_14MHZ);
  // play a wave on Signal Output 2
  playWave(w_sinc, AWG_RATE_1800MHZ);
}
```

The following table summarizes the settings to be made for this example.

Table 4.7: Settings: configure the AWG for generating signals on the Auxiliary

Tab	Sub-tab	Section	#	Label	Setting / Value / State
AWG	Control	Output 1/2		Mode	Plain
Auxiliary		Aux Output	1-4	Lower/Upper Limit	–1.5 V/+1.5 V
Auxiliary		Aux Output	1-4	Signal	AWG
Auxiliary		Aux Output	1	Channel	1
Auxiliary		Aux Output	2	Channel	2
Auxiliary		Aux Output	3	Channel	3

Tab	Sub-tab	Section	#	Label	Setting / Value / State
Auxiliary		Aux Output	4	Channel	4

Outputs

Note

The four-channel AWG mode features a sample hold functionality: the output voltage of the last sample of a waveform remains fixed after the waveform playback is over. This can be used to control the output voltage between pulses.

4.1.7. Debugging Sequencer Programs

When generating fast signals and observing them with the LabOne Scope, in some configurations you may observe timing jitter or unexpected delays in the generated signal. There are two main reasons for that. The first reason is linked to the AWG's memory architecture, which is based on a main memory and a cache memory. Waveform data stored in the main memory (128 MSa per channel) must be copied to the cache memory (32 kSa per channel) prior to playback. The bandwidth available for this data transfer is less than that required by the AWG for dual-channel operation at 1.8 GSa/s. Therefore, if the AWG is configured to play waveforms longer than what fits in the cache memory in dual-channel mode at 1.8 GSa/s, interruptions in the generated signal may be observed. The second reason is connected to the AWG compiler concept explained in Description. When a program in the Sequence Editor is compiled into machine code that can be executed by the Sequencer hardware, single lines of code may be expanded into several machine instructions. Each instruction requires one clock cycle (4.44 ns) for execution. Therefore, the final timing of the generated waveform may not always be completely apart from looking solely at the high-level sequencer program. The compiled program, which defines the actual timing, is displayed in the Advanced sub-tab.

Please take the following tips into consideration when operating the UHF-AWG. They should help you prevent and solve timing problems.

- The Scope and the AWG share the same memory, which means that operating them together at high sampling rates affects the performance of both of them. Note that this is only a concern when the AWG is playing back waveforms that are too large to fit in the cache memory. If this is the case it may prove difficult to visualize the generated AWG signal using the LabOne Scope. One option for visualizing such long waveforms is to reduce the sampling rate of both the AWG and the Scope to 225 MHz, which allows both the AWG and the Scope to operate in dual-channel mode simultaneously. The overall shape of the generate AWG signal can then be visualized and evaluated. The sampling rate of the AWG can then be increased once you are satisfied with the shape of the generated signals.
- Minimize waveform memory (1): use the possibility to vary the sample rate during playback. The `playWave` command (and related commands) accept a sampling rate parameter, which means slow and fast signal components can be played at different rates.
- Minimize waveform memory (2): take advantage of the amplitude modulation mode in order to generate signals at the full bandwidth, but with reduced envelope sampling rate.
- Minimize waveform memory (3): in four-channel (Auxiliary Output) mode, the signal amplitude of the last sample after a waveform playback is held. This eliminates the need for long waveforms with constant amplitude, e.g. on a pulse plateau.
- Check the occupied waveform cache memory in the Waveform sub-tab. If you stay below 100%, the performance is best and there is no interference with the LabOne Scope.
- Take advantage of the AWG state signals available on the Trigger outputs. In the DIO tab you can select from a number of options for outputting the AWG state as TTL signals, such as "fetching", or "playing". Monitoring these signals on a scope can help in understanding the AWG timing.
- When possible, use the `repeat` loop instead of the `for` and `while` loops. The `for` and `while` loops evaluate and compare run-time variables, which makes them slower to execute in comparison to the `repeat` loop.
- Fill up sequencer waiting time with useful commands. Placing commands and run-time variable operations just before a `wait` command (and related commands) in the sequence program means they will be executed when the sequencer has time.
- When you need sample-precise timing between analog and digital output signals, use the AWG Markers rather than the AWG Triggers or the DIO outputs.
- When using the four-channel Auxiliary Output mode, be aware that the timing between Signal Output and Aux outputs is not well-defined. Use a scope to adjust inter-channel delays.
- Be aware that the sequencer instruction memory is also segmented into a cache memory and main memory. Very long sequence programs therefore require fetching operations, which costs some time. You can read the memory usage in the Advanced sub-tab.

4.2. Generate and Acquire a Test Signal

Note

This tutorial is applicable to all UHFQA Instruments.

4.2.1. Goals and Requirements

This tutorial explains how to generate and measure a simple pulsed signal with the AWG and the Monitoring Scope in the Quantum Analyzer Input tab.

The measurements in this tutorial can be performed using simple loop back connections.

4.2.2. Preparation

Connect the cables as illustrated below. Make sure that the UHF unit is powered on and connected by USB to your host computer or by Ethernet to your local area network (LAN) where the host computer resides. After starting LabOne the default web browser opens with the LabOne graphical user interface.

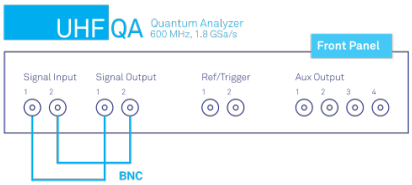


Figure 4.10: UHF connections for the Input Monitor tutorial

The tutorial can be started with the default instrument configuration (e.g. after a power cycle) and the default user interface settings (e.g. as is after pressing F5 in the browser).

4.2.3. Test Signal Generation

First, we enable both outputs of the UHF instrument.

Table 4.8: Settings: enable the Signal Outputs

Tab	Sub-tab	Section	#	Label	Setting / Value / State
In / Out		Signal Outputs	1	On	ON
In / Out		Signal Outputs	2	On	ON

Copy the following code into the Sequence Editor in the AWG tab.

```
const LENGTH = 4096;
wave w = gauss(LENGTH, LENGTH/2, LENGTH/8);
var loop_cnt = getUserReg(0);
var wait_time = 0;

repeat (loop_cnt) {
  playWave(w, -w);
  startQA(QA_INT_NONE, true);
  playZero(LENGTH);
}
```

Upload this sequence program to the AWG of the UHFQA by clicking on "Save" or "To Device". This program will generate a series of dual-channel Gaussian pulses. The repetition number is defined by the integer variable `loop_cnt`. To make it possible to control the repetition number through the user interface, we use one of the User Registers rather than to write this number into the program.

Apply the settings in the following table in order to configure the AWG output as well as the User Register.

Table 4.9: Settings: configure the AWG output

Tab	Sub-tab	Section	#	Label	Setting / Value / State
AWG	Control			Rerun	OFF
AWG	Control	Output 1		Amplitude (FS)	1.0
AWG	Control	Output 1		Mode	Plain
AWG	Control	Output 2		Amplitude (FS)	1.0
AWG	Control	Output 2		Mode	Plain
AWG	Control	User Registers		Register 1	64

4.2.4. Configure the QA setup tab

In the Quantum Analyzer Setup tab, apply the setting in the table below.

Table 4.10: Settings: configure the integration delay in the QA Setup tab

Tab	Sub-tab	Section	#	Label	Setting / Value / State
QA Setup		Deskew		Delay (sample)	200

This setting configures the trigger delay of the monitor, so that the generated signal is correctly aligned with the acquisition window.

4.2.5. Configure the Input Monitor

In the Quantum Analyzer Input tab, apply the settings in the table below.

Table 4.11: Settings: configure the Input Monitor

Tab	Sub-tab	Section	#	Label	Setting / Value / State
QA Input	Control	Input Monitor		Length	4096
QA Input	Control	Input Monitor		Averages	64
QA Input	Control			Run / Stop	ON

In the AWG tab, click on "Start/Stop" in order to run the AWG. The line `startQA(QA_INT_NONE, true);` triggers the Input Monitor acquisition. See Architecture and Signalling for an overview of the different functional blocks and internal trigger lines. 64 consecutive dual-channel pulses are acquired, averaged, and displayed. The figure below shows the signal as displayed in the QA Input tab. We denote the raw input signals (without averaging) as $V_1(t)$ and $V_2(t)$.

Note

The Input Monitor Averages setting must agree with the number of Input Monitor triggers generated by the AWG in one measurement burst (here this is determined by the User Register 1 equal to the sequencer variable `loop_cnt`. If the AWG generates more triggers than that, which is e.g. the case when AWG Rerun would be enabled, the Input Monitor may not be able to process and transmit all data in time, and may deliver corrupted data to the computer.

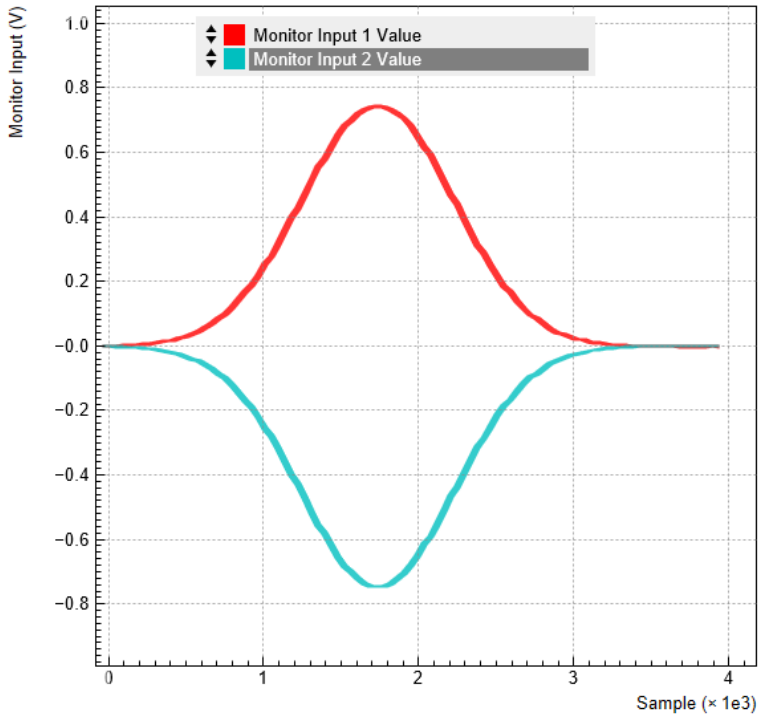


Figure 4.11: Test signal as measured in the Quantum Analyzer Input tab

4.3. Dual-phase Demodulation

Note

This tutorial is applicable to all UHFQA Instruments.

4.3.1. Goals and Requirements

This tutorial explains how to analyze a pulsed signal using the weighted integration units and record the result with the Result Logger in the Quantum Analyzer Result tab.

The measurements in this tutorial can be performed using simple loop back connections.

4.3.2. Preparation

Connect the cables as illustrated below. Make sure that the UHF unit is powered on and connected by USB to your host computer or by Ethernet to your local area network (LAN) where the host computer resides. After starting LabOne the default web browser opens with the LabOne graphical user interface.

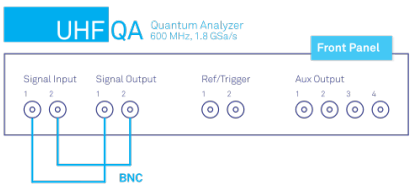


Figure 4.12: UHF connections for the Dual-phase Demodulation tutorial

The tutorial can be started with the default instrument configuration (e.g. after a power cycle) and the default user interface settings (e.g. as is after pressing F5 in the browser).

4.3.3. Continuous Wave Signal Generation

First, we enable both outputs of the UHF instrument and the digital oscillator. The first (second) output then generates a cosine (sine) waveform at the frequency defined in the oscillator section.

Table 4.12: Settings: observe the pulse waveform

Tab	Sub-tab	Section	#	Label	Setting / Value / State
In / Out		Signal Outputs	1	On	ON
In / Out		Signal Outputs	1	50 Ohm	ON
In / Out		Signal Outputs	1	Amp (Vpk)	0.5
In / Out		Signal Outputs	1	Amp (Vpk)	ON
In / Out		Signal Outputs	2	On	ON
In / Out		Signal Outputs	2	50 Ohm	ON
In / Out		Signal Outputs	2	Amp (Vpk)	0.5
In / Out		Signal Outputs	2	Amp (Vpk)	ON
In / Out		Oscillators	1	Frequency (Hz)	10M

Copy the following code into the Sequence Editor in the AWG tab.

```
var result_length = getUserReg(0);
var result_averages = getUserReg(1);

repeat (result_averages) {
  repeat (result_length) {
    startQA(QA_INT_0 | QA_INT_1, true);
    wait(512);
  }
}
```

Upload this program to the AWG by clicking on "Save" or "To Device". The outputs will generate a continuous waveform at the frequency as configured before, and the sequencer will control only the triggering of the readout and monitor unit. See Architecture and Signalling for an overview of the different functional blocks and internal trigger lines.

There are two nested loops in the sequence above: the number of repetitions in the inner loop is determined by User Register 1 and would correspond to the number of points in a pulsed measurement. The outer loop corresponds to the number of identical repetitions to be performed in order to allow averaging and is determined by User Register 2. For spectroscopy measurements, one is usually interested in a single averaged data point while stepping an external parameter, such as a signal generator frequency. This would correspond to a parameter `result_length` of 1. Here, we choose a `result_length` 2, for the practical reason that the final data is more clearly displayed in the QA Result tab. Apply the settings in the following table to configure the AWG output as well as the User Registers.

Table 4.13: Settings: configure the AWG output and User Registers

Tab	Sub-tab	Section	#	Label	Setting / Value / State
AWG	Control			Rerun	OFF
AWG	Control	User Registers		Register 1	2 (Length)
AWG	Control	User Registers		Register 2	32 (Averages)

To visualize the generated signal, open the QA Input tab, set Averages to 1, click "Run/Stop" in the QA Input tab, and "Start/Stop" in the AWG tab. The figure below shows the generated wave with a 10 MHz carrier. If the AWG is run multiple times, it can be observed that the phase changes across runs. This is because the AWG is triggered at random time, compared to the phase of the sine generator. See the AWG tutorial to learn how to control the phase.

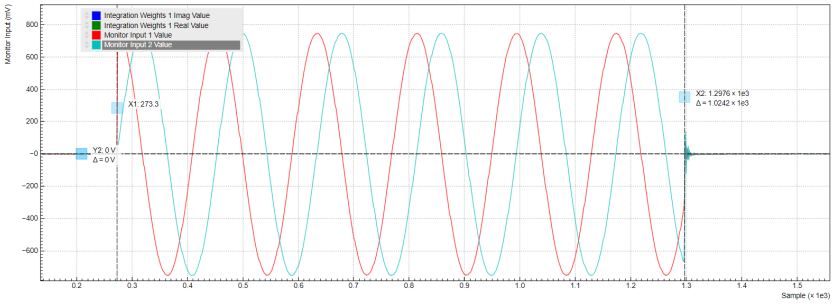


Figure 4.13: Dual-channel signal measured in the Quantum Analyzer Input tab

The two carrier signals correspond to the following configuration:

Signal Output 1 proportional to cos(ωt)

Signal Output 2 proportional to sin(ωt)

where ω = 2*pi*(10 MHz) is the carrier frequency. This pair of signals is then typically used as I and Q input signals of an IQ mixer with a local oscillator signal of frequency ω_{LO} applied to its L input. A perfect IQ mixer generates the following signal $V_{RF}(t)$ at its RF port:

$$V_{RF}(t)= V_I(t)\cos(\omega_{LO}t) + V_Q(t)\sin(\omega_{LO}t)$$

where $V_I(t)$ and $V_Q(t)$ are input signals at the mixer I and Q ports. A real mixer is characterized by a certain phase imbalance θ and amplitude imbalance α, so that it effectively performs the following operation:

$$V_{RF}(t)= V_I(t)\cos(\omega_{LO}t) + V_Q(t)\sin(\omega_{LO}t+\theta)\alpha$$

We need to apply the following baseband signals in order to generate a signal at the lower sideband $\omega_{LO}-\omega$:

$$V_I^-(t) = \cos(\omega t)$$

$$V_Q^-(t) = \sin(\omega t-\theta)/\alpha$$

In order to generate a signal at the upper sideband $\omega_{LO}+\omega$, we need:

$$V_I^+(t) = \cos(\omega t)$$

$$V_Q^+(t) = -\sin(\omega t+\theta)/\alpha$$

When disregarding the mixer imperfections (θ=0, α=1), we can generate the lower sideband by connecting Signal Output 1 to the I port, and Signal Output 2 to the Q port. We can generate the upper sideband by flipping the connections between I and Q ports. This is the recommended configuration. When using AWG digital modulation in spectroscopy mode, it is not possible to correct the carriers for mixer imbalance. However, in this case this is typically not needed, since the unwanted sideband usually does not interfere with the measurement. Furthermore, the mixer imbalance depends on frequency which is an issue when either ω_{LO} or ω is swept. When performing qubit readout in a frequency-multiplexed fashion, mixer imbalance correction is however important in order to avoid crosstalk. It can be accounted for by modifying the carrier amplitude and phase in the programmed waveform memory.

4.3.4. Configure the QA Setup tab

In the QA Setup tab, we define the method of demodulation of the generated signals. We use the Spectroscopy mode, in which the input signals are demodulated with the in-phase and quadrature components of the internal oscillator signal. Since each readout channel delivers only a single quadrature, we will set up 2 channels to obtain both quadratures. For a full dual-phase demodulation of the two input signals, we would like to obtain the following complex amplitude:

$$A = \int [V_1'(t) + iV_2'(t)]e^{-i\omega t}dt \qquad (1)$$

$V'_1(t)$ and $V'_2(t)$ are the input signals after application of the deskew matrix on the left side of the QA Setup tab. Since we didn't change the deskew matrix from its default value, which is the unit matrix, these signals are identical to the signals at the Signal Inputs, $V_1(t)$ and $V_2(t)$. When taking the real and imaginary part of this expression, we obtain:

$$\mathrm{Re}(A) = \int [V_1'(t)\cos(\omega t) + V_2'(t)\sin(\omega t)]dt \tag{2}$$

$$\mathrm{Im}(A) = \int [-V_1'(t)\sin(\omega t) + V_2'(t)\cos(\omega t)]dt \tag{3}$$

We configure the first two readout channels as follows such that their outputs are equal to Re(A) and Im(A), respectively:

Table 4.14: Settings: configure the integration

Tab	Sub-tab	Section	#	Label	Setting / Value / State
QA Setup		Integration		Mode	Spectroscopy
QA Setup		Integration		Length	1024
QA Setup		Integration	1	Signal Input Mapping	2 → Real, 1 → Imag
QA Setup		Integration	2	Signal Input Mapping	1 → Real, 2 → Imag
QA Setup		Integration	1	Rotation	1 - 1i
QA Setup		Integration	2	Rotation	-1 - 1i

When the UHFQA is configured to operate in spectroscopy mode, a readout channel will calculate the following quantities:

$$A_{INT} = \int [V_{REAL}'(t)\sin(\omega t) + V_{IMAG}'(t)\cos(\omega t)]dt, \tag{4}$$

$$A_{ROT} = \mathrm{Re}[\phi_{ROT} A_{INT}], \tag{5}$$

where A_{INT} and A_{ROT} are the results after integration and rotation, respectively.

Therefore, with these settings readout channel 1 will perform the following operation (note the interchanged input voltages):

$$\mathrm{Re}[(1 - 1i)\int [V_2'(t)\cos(\omega t) + iV_1'(t)\sin(\omega t)]dt] \tag{6}$$

Readout channel 2 will effectively perform the following operation:

$$\mathrm{Re}[(-1 - 1i)\int [V_1'(t)\cos(\omega t) + iV_2'(t)\sin(\omega t)]dt] \tag{7}$$

These expressions are identical to the desired quadratures Re(A) and Im(A) noted previously.

In the Quantum Analyzer Result tab, apply the settings in the table below in order to acquire an averaged measurement:

Table 4.15: Settings: configure the Result Logger

Tab	Sub-tab	Section	#	Label	Setting / Value / State
QA Result	Control	Result Wave		Source	Rotation
QA Result	Control	Result Wave		Length	2
QA Result	Control	Result Wave		Averages	32
QA Result	Control	Result Wave		Mode	Cyclic

Tab	Sub-tab	Section	#	Label	Setting / Value / State
QA Result	Control	Vertical Axis Groups		Result Wave 1 / Add Signal	click
QA Result	Control	Vertical Axis Groups		Result Wave 2 / Add Signal	click
QA Result	Control			Reset	click
QA Result	Control			Run / Stop	ON

Click on "Start/Stop" in the AWG tab in order to run the AWG. The line `startQA(QA_INT_0 | QA_INT_1, true);` in the AWG sequence program starts the weighted integration on the first two readout channels. 32 consecutive dual-channel pulses are acquired, integrated, averaged, and displayed.

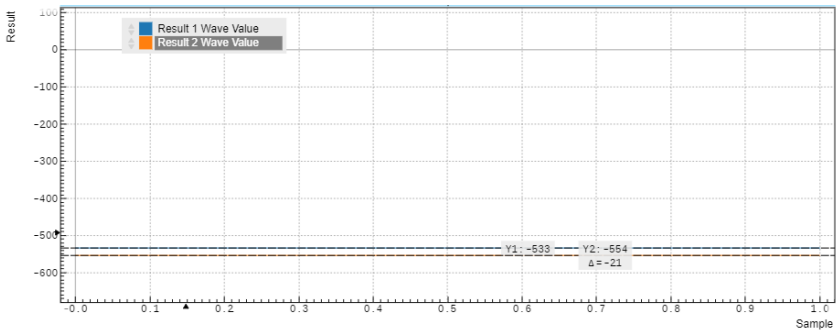


Figure 4.14: Measurement result at 10 MHz as displayed in the Quantum Analyzer Result tab

The data from channel 1 (blue curve) correspond to the in-phase component of the signal, the data from channel 2 (green curve) correspond to the quadrature component of the signal. For a signal with a negligible delay between generation and acquisition, we would expect this measurement to yield a zero phase, i.e., a zero quadrature component. Here, we measure an in-phase component of -533 V, and a quadrature component of -554 V. This would correspond to a normalized complex signal amplitude of (-533 - 554 i)/1024 V = 0.75 V @ -136 degrees. While the amplitude of 0.75 V corresponds well to the peak amplitude of our generated signal, but the phase is different from the naively expected value of 0 degrees. Here, this phase offset comes mainly from signal processing delays in the DAC and ADC stages, but in experiment, the length of the signal path between output and input contributes to the phase offset as well. This can normally be calibrated by changing the Rotation parameters of both measurement channels.

We can repeat the measurement using a much lower oscillator frequency of 10 kHz instead of 10 MHz. At this frequency, the phase offset due to signal processing delays is negligible. The following figure shows the result with an normalized in-phase component of 743/1024 V = 0.73 V, and a quadrature component of 11/1024 V = 0.01 V, corresponding well to the peak amplitude of 0.75 V at phase 0 degrees.

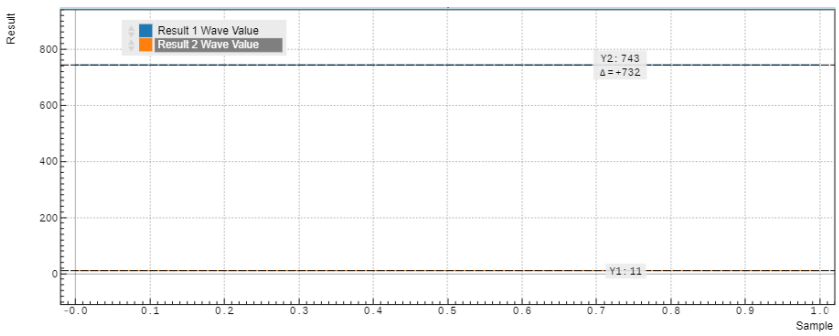


Figure 4.15: Measurement result at 10 kHz as displayed in the Quantum Analyzer Result tab

4.4. Weighted Integration and Result Logging

Note

This tutorial is applicable to all UHFQA Instruments.

4.4.1. Goals and Requirements

This tutorial explains how to analyze a pulsed signal using the weighted integration units and record the result with the Result Logger in the Quantum Analyzer Result tab.

The measurements in this tutorial can be performed using simple loop back connections.

4.4.2. Preparation

Connect the cables as illustrated below. Make sure that the UHF unit is powered on and connected by USB to your host computer or by Ethernet to your local area network (LAN) where the host computer resides. After starting LabOne the default web browser opens with the LabOne graphical user interface.

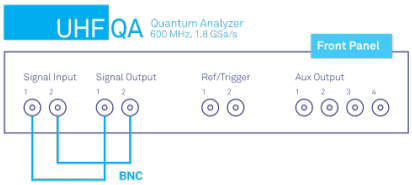


Figure 4.16: UHF connections for the Result Logger tutorial

The tutorial can be started with the default instrument configuration (e.g. after a power cycle) and the default user interface settings (e.g. as is after pressing F5 in the browser).

4.4.3. Test Signal Generation

First, we enable both outputs of the UHF instrument.

Table 4.16: Settings: observe the pulse waveform

Tab	Sub-tab	Section	#	Label	Setting / Value / State
In / Out		Signal Outputs	1	On	ON
In / Out		Signal Outputs	2	On	ON

Copy the following code into the Sequence Editor in the AWG tab.

```
const length = 4096;
const f_s = 1.8e9;
const f_readout = 10e6;
wave w_I = gauss(length, length/2, length/8)
             *cosine(length, 1, 0, f_readout*length/f_s);
wave w_Q = gauss(length, length/2, length/8)
             *sine(length, 1, 0, f_readout*length/f_s);

var result_length_by_2 = getUserReg(0)>>1; // We divide by two using the
                                           // bit shift operator because
                                           // each loop contains two pulses
var result_averages = getUserReg(1);

repeat (result_averages) {
  repeat (result_length_by_2) {
    playWave(w_I, w_Q);
    startQA(QA_INT_0, true);
```

```
    playZero(2*length);

    playWave(0.5*w_I, 0.5*w_Q);
    startQA(QA_INT_0, true);
    playZero(2*length);
  }
}
```

Upload this program to the AWG by clicking on "Save" or "To Device". The program generates a series of modulated Gaussian pulses with two alternating amplitudes. There are two nested loops: the number of pulses in the inner loop is determined by User Register 1 and would correspond to the number of points in a pulsed measurement (e.g., a Rabi flopping). The outer loop corresponds to the number of identical repetitions to be performed in order to allow averaging and is determined by User Register 2. Apply the settings in the following table to configure the AWG output as well as the User Registers.

Table 4.17: Settings: configure the AWG output and User Registers

Tab	Sub-tab	Section	#	Label	Setting / Value / State
AWG	Control			Rerun	OFF
AWG	Control	Output 1		Amplitude (FS)	1.0
AWG	Control	Output 1		Mode	Plain
AWG	Control	Output 2		Amplitude (FS)	1.0
AWG	Control	Output 2		Mode	Plain
AWG	Control	User Registers		Register 1	100 (Length)
AWG	Control	User Registers		Register 2	1 (Averages)

4.4.4. Configure the Integration Units

The AWG generates pulses with cosine and sine carrier waves at 10 MHz. In the QA Input tab, we have to define integration weights that enable demodulation of these signals at the right frequency. We choose channel 1 for this measurement.

Table 4.18: Settings: define the integration weights

Tab	Sub-tab	Section	#	Label	Setting / Value / State
QA Input	Control	Weights / Generate		Amplitude	1
QA Input	Control	Weights / Generate		Frequency	10 M
QA Input	Control	Weights / Generate		Window Length	4096
QA Input	Control	Weights / Generate		Channel	1
QA Input	Control	Weights / Generate		Set	click
QA Setup		Integration		Mode	Standard
QA Setup		Integration		Length	4096

Upon clicking on Set in the QA Input tab, the sinusoid integration weights are displayed in the plot area of this tab as shown in the figure below. We denote these dimensionless weights as $W_{I,q}(t)$ and $W_{Q,q}(t)$, where q = 1,...,10 denotes the measurement channel, and I and Q correspond to the two signal inputs.

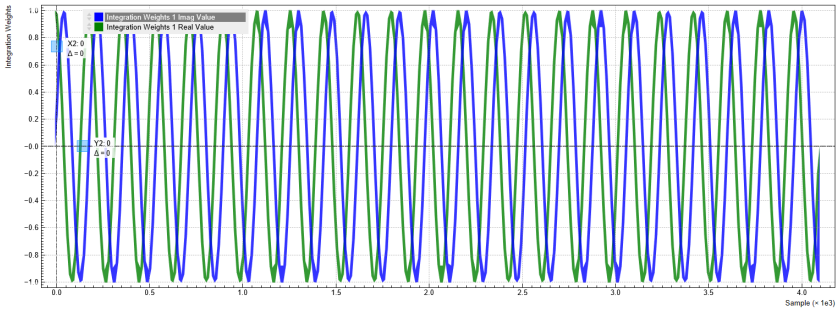


Figure 4.17: Integration weights as displayed in the Quantum Analyzer Input tab

We can check if the frequency of the sinusoids agree with the signal programmed in the AWG. The measurement is performed similarly as described in the previous tutorial. In the QA Input tab, set the Length to 4096 and the Averages to 64, and click on Run/Stop. Then, start the AWG by clicking on Run/Stop in the AWG tab. The peak amplitude of about 530 mV corresponds to the average between the high and low amplitude pulses generated by the AWG. We can see that the frequencies of the weights and the signal agree. An undesired delay between the signal and integration period can be compensated by using the Delay setting in the Deskew section of the QA Setup tab. Phase offsets can be compensated by directly changing the phase of the generated signal or the integration weights. Alternatively, the Rotation coefficients in the QA Setup tab allow you to rotate the signal in the complex plane after the integration. This can also be used to compensate a phase offset between signal and integration weight, but it doesn't have an effect on the raw signal displayed in the Input Monitor.

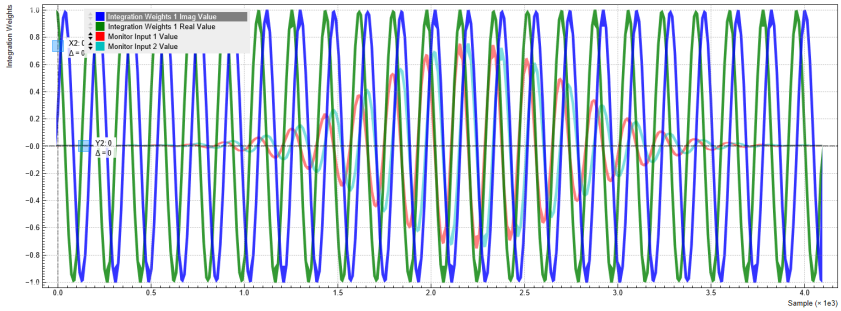


Figure 4.18: Integration weights and signal as displayed in the Quantum Analyzer Input tab

In the Quantum Analyzer Input tab, apply the settings in the table below in order to conduct a measurement without averaging.

Table 4.19: Settings: configure the Result Logger

Tab	Sub-tab	Section	#	Label	Setting / Value / State
QA Result	Control	Result Wave		Source	Rotation
QA Result	Control	Result Wave		Length	100
QA Result	Control	Result Wave		Averages	1
QA Result	Control	Result Wave		Run / Stop	ON

Click on "Start/Stop" in the AWG tab in order to run the AWG. The line `startQA(QA_INT_0, true);` in the AWG sequence program generates a trigger to start the weighted integration. 100 consecutive dual-channel pulses are acquired, integrated, and displayed. The figure below shows the signal as displayed in the QA result tab.

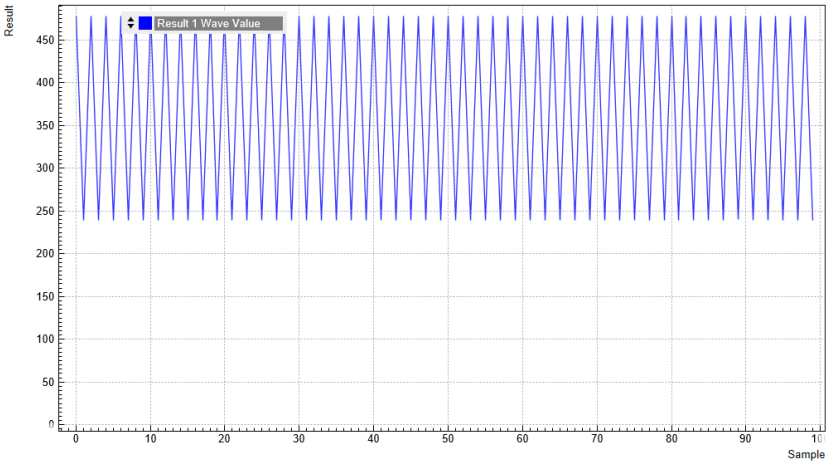


Figure 4.19: Measurement result as displayed in the Quantum Analyzer Result tab. The numerical values depends on the phase relation between generation and demodulation, which in turn depends on the cable length and on the Delay setting in the QA Setup tab.

These values in the graph are the integrated in-phase voltages $V'_{I,q}$ after application of the Rotation coefficient of channel q. Since we left the Rotation coefficient at its default, which is 1+0i, the voltage $V'_{I,q}$ is equal to the voltage $V_{I,q}$ before rotation. This voltage $V_{I,q}$, along with its quadrature counterpart $V_{Q,q}$, is obtained as

$$V_{I,q} = f_s \int_0^T W_{I,q}(t) V'_{1,q}(t) dt \tag{1}$$

$$V_{Q,q} = f_s \int_0^T W_{Q,q}(t) V'_{2,q}(t) dt \tag{2}$$

where T is the length (in units of time) of the integration weights $W_{I,q}(t)$ and $W_{Q,q}(t)$, and $V'_{1,q}(t)$ and $V'_{2,q}(t)$ are the input signals after application of the deskew matrix accessible in the QA Setup tab, and f_s is the input sampling rate 1.8 GSa/s. Since in our case we didn't change the deskew matrix from its default which is the unit matrix, these two signals are identical to the raw input signals on Signal Inputs 1 and 2, $V_{1,q}(t)$ and $V_{2,q}(t)$.

Expressed in terms of discrete-time samples n, the formula above become

$$V_{I,q} = f_s \sum_{n=0}^{N-1} W_{I,q}(t) V'_{1,q}(t) \tag{3}$$

$$V_{Q,q} = f_s \sum_{n=1}^{N-1} W_{Q,q}(t) V'_{2,q}(t) \tag{4}$$

where N is equal to the length (in units of samples) of the integration weights.

Note

The integrity of the data vector of the Result Logger depends on this unit receiving a number of AWG Integration triggers equal to the product of the Averages and Length settings of the Result Logger. If this condition is violated, the data of the Result Logger may get corrupted. Often, the first point of the Result vector may then contain an outlier value. Notably, the Result Logger may be driven into an invalid state ("Holdoff error") by generating an endless series of triggers, e.g. by enabling the AWG in Rerun mode.

In the given case, the integrated voltages in Figure 4.19 alternate between about 240 V and 480 V. This value lies in the range [– 4096×1.5 V, + 4096×1.5 V] determined by the maximum length of the integration window and the maximum input voltages. Note that the measured values depend on the phase relation between generated signal demodulation weights, which in turn depend on the cable length and on the Delay setting in the QA Setup tab.

In the next step, we can apply a voltage threshold in order to discriminate the analog signal. In practice, this allows us to convert the analog readout signal in a discrete qubit state readout result. In the Thresholds section of the QA Setup tab in line 1, enter a value of 360, which lies in between the high and low readout signals in the previous measurement. Change the Source signal in the QA Result tab to Threshold, click Reset, and run again the AWG. The following table summarizes these settings.

Table 4.20: Settings: apply a threshold

Tab	Sub-tab	Section	#	Label	Setting / Value / State
QA Setup		Thresholds	1	Level	360
QA Result	Control	Result Wave		Source	Threshold
QA Result	Control	Result Wave		Reset	click
QA Result	Control	Result Wave		Run / Stop	ON
AWG	Control			Start / Stop	ON

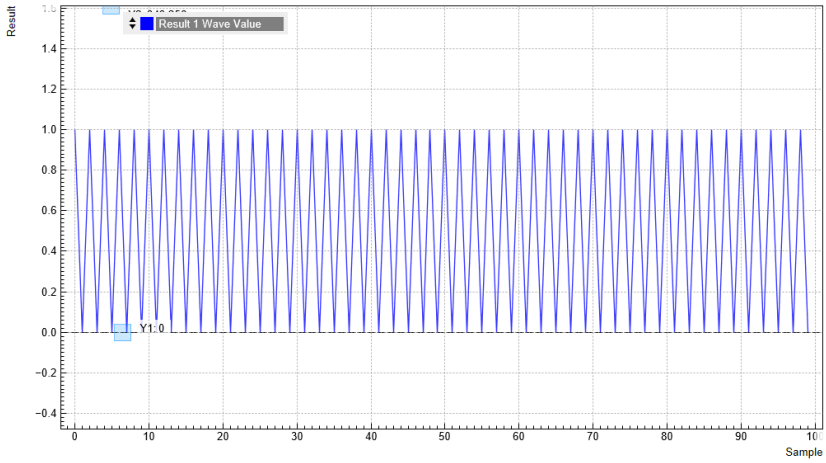


Figure 4.20: Measurement result as measured in the Quantum Analyzer Result tab

The discretized readout results can be output with low latency as digital signal on the DIO connector on the back of the instrument. To this end, set the Digital I/O Mode to QA Result in the DIO tab. Please refer to the Digital Interface Specifications for a specification of the pin assignment.

5. Functional Description LabOne User Interface

This chapter gives a detailed description of the functionality available in the LabOne User Interface (UI) for the Zurich Instruments UHFQA. LabOne provides a data server and a web server to control the Instrument with any of the most common web browsers (e.g. Firefox, Chrome, Edge, etc.). This platform-independent architecture supports interaction with the Instrument using various devices (PCs, tablets, smartphones, etc.) even at the same time if needed.

On top of standard functionality like acquiring and saving data points, this UI provides a wide variety of measurement tools for time and frequency domain analysis of measurement data as well as for convenient servo loop implementation.

5.1. User Interface Overview

5.2. UI Nomenclature

This section provides an overview of the LabOne User Interface, its main elements and naming conventions. The LabOne User Interface is a browser-based UI provided as the primary interface to the UHFQA instrument. Multiple browser sessions can access the instrument simultaneously and the user can have displays on multiple computer screens. Parallel to the UI, the instrument can be controlled and read out by custom programs written in any of the supported languages (e.g. LabVIEW, MATLAB, Python, C) connecting through the LabOne APIs.

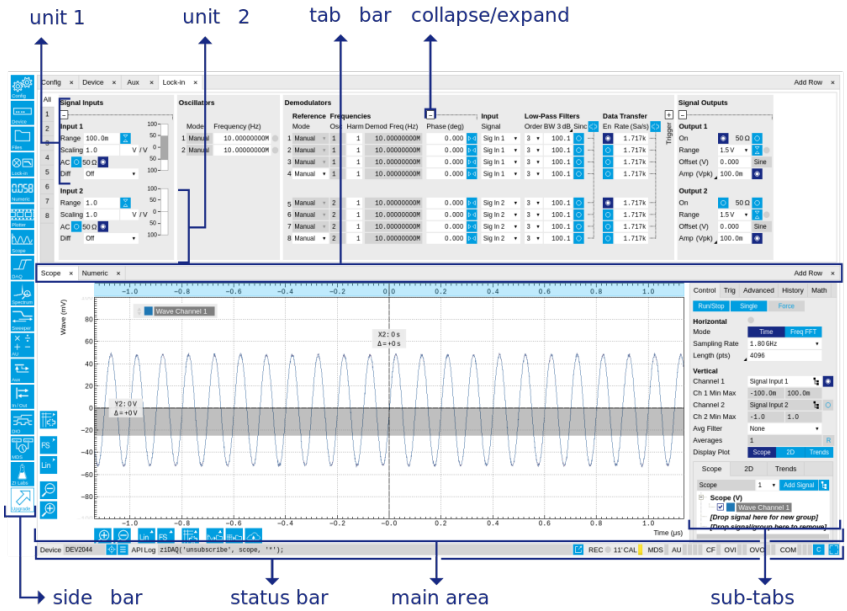


Figure 5.1: LabOne User Interface (default view)

The LabOne User Interface of the UHFLI Lock-in Amplifier automatically opens some tabs by default after a new UI session has been started. At start-up, the UI is divided into two tab rows, each containing a tab structure that gives access to the different LabOne tools. Depending on display size and application, tab rows can be freely added and deleted with the control elements on the right-hand side of each tab bar. Similarly, the individual tabs can be deleted or added by selecting app icons from the side bar on the left. A click on an icon adds the corresponding tab to the display, alternatively the icon can be dragged and dropped into one of the tab rows. Moreover, tabs can be moved by drag-and-drop within a row or across rows.

Table 5.1 gives a brief descriptions and naming conventions for the most important UI items.

Table 5.1: LabOne User Interface features

Item name	Position	Description	Contains
side bar	left-hand side of the UI	contains app icons for each of the available tabs - a click on an icon adds or activates the corresponding tab in the active tab row	app icons
status bar	bottom of the UI	contains important status and warning indicators, device and session information, and access to the command log	status indicators
main area	center of the UI	accommodates all active tabs – new rows can be added and removed by using the control elements in the top right corner of each tab row	tab rows, each consisting of tab bar and the active tab area
tab area	inside of each tab	provides the active part of each tab consisting of settings, controls and measurement tools	sections, plots, sub-tabs, unit selections

Further items are highlighted in Figure 5.2.

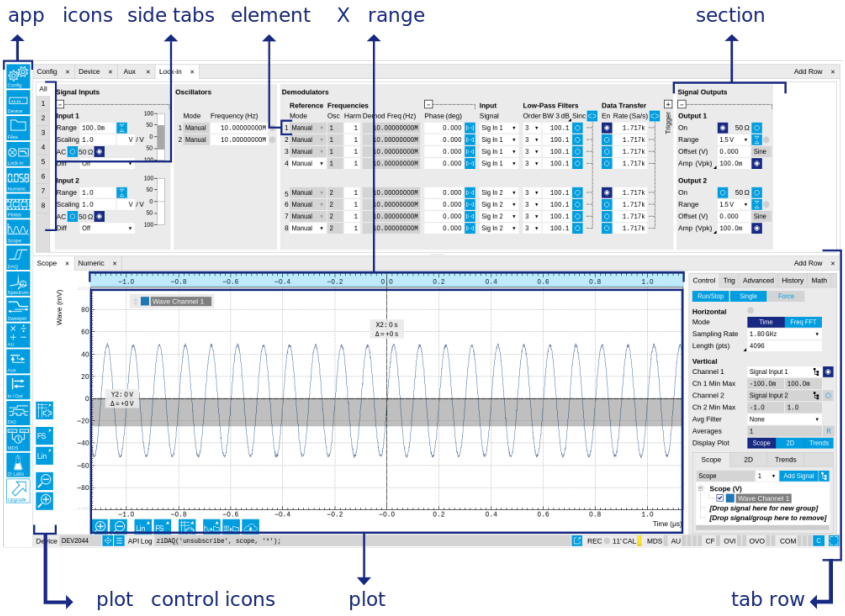


Figure 5.2: LabOne User Interface (more items)

5.2.1. Unique Set of Analysis Tools

All instruments feature a comprehensive tool set for time and frequency domain analysis for both raw and demodulated signals.

The following table gives the overview of all app icons. Note that the selection of app icons may depend on the upgrade options installed on a given instrument.

Table 5.2: Overview of app icons and short description

Control/ Tool	Option/ Range	Description
QA Setup		Configure the Qubit Measurement Unit
QA Input		Configure the Weighted Integration units and Monitoring Scope
QA Result		Configure the Result Logger.
Files		Access settings and measurement data files on the host computer.
Numeric		Access to all continuously streamed measurement data as numerical values.

Control/ Tool	Option/ Range	Description
Plotter		Displays various continuously streamed measurement data as traces over time (roll mode).
Scope		Displays shots of data samples in time and frequency domain (FFT) representation.
Aux		Controls all settings regarding the auxiliary inputs and auxiliary outputs.
In/Out		Gives access to all controls relevant for the Signal Inputs and Signal Outputs of each channel.
DIO		Gives access to all controls relevant for the digital inputs and outputs including DIO, Trigger Inputs, Trigger Outputs, and Marker Outputs.
Config		Provides access to software configuration.
Device		Provides instrument specific settings.
AWG		Generate arbitrary signals using sequencing and sample-by-sample definition of waveforms.
MDS		Synchronize multiple instruments.
ZI Labs		Experimental settings and controls.

Table 5.3 provides a quick overview over the different status bar elements along with a short description.

Table 5.3: Status bar description

Control/ Tool	Option/ Range	Description
Command log	last command	Shows the last command. A different formatting (MATLAB, Python, ..) can be set in the config tab. The log is also saved in [User] \Documents\Zurich Instruments\LabOne\WebServer\Log
Show Log		Show the command log history in a separate browser window.
Errors	Errors	Display system errors in separate browser tab.
Device	devXXX	Indicates the device serial number.
Identify Device		When active, device LED blinks
Next Calibration	Time or "M"	Remaining minutes until the first calibration is executed or a recalibration is requested. A time interval longer than 99 minutes is not displayed. Manual calibration mode is indicated by an "M".
CAL	grey/ yellow/red	State of device self calibration. Yellow: device is warming up and will automatically execute a self calibration after 16 minutes. Grey: device is warmed-up and self calibrated. Red: it is recommended to manually execute a self calibration to assure operation according to specifications.
MDS	grey/green/ red/yellow	Multiple device synchronization indicator. Grey: Nothing to synchronize - single device on the UI. Green: All devices on the UI are correctly synchronized. Yellow: MDS sync in progress or only a subset of the connected devices is synchronized. Red: Devices not synchronized or error during MDS sync.
REC	grey/red	A blinking red indicator shows ongoing data recording (related to global recording settings in the Config tab).
AWG	grey/green	Arbitrary Waveform Generator - Green: indicates that the AWG core is enabled.
CNT	grey/green	Pulse Counter - Green: indicates which of the pulse counter modules is enabled.
DAC Error	grey/green	Red indicates that the digital to analog converter at the output encountered an error during operation. An error leads to additional jitter in the output wave, scrambled output or no output at all. If an error is encountered, please contact Zurich Instruments for support.

Control/ Tool	Option/ Range	Description
AU	grey/green/ red	Arithmetic Unit - Green: indicates which of the arithmetic units is enabled. Red: indicates overflow.
CF	grey/ yellow/red	Clock Failure - Red: present malfunction of the external 10 MHz reference oscillator. Yellow: indicates a malfunction occurred in the past.
OVI	grey/ yellow/red	Signal Input Overload - Red: present overload condition on the signal input also shown by the red front panel LED. Yellow: indicates an overload occurred in the past.
OVO	grey/ yellow/red	Overload Signal Output - Red: present overload condition on the signal output. Yellow: indicates an overload occurred in the past.
COM	grey/ yellow/red	Packet Loss - Red: present loss of data between the device and the host PC. Yellow: indicates a loss occurred in the past.
COM	grey/ yellow/red	Sample Loss - Red: present loss of sample data between the device and the host PC. Yellow: indicates a loss occurred in the past.
COM	grey/red	Stall - Red: indicates that the sample transfer rates have been reset to default values to prevent severe communication failure. This is typically caused by high sample transfer rates on a slow host computer.
C	c	Reset status flags: Clear the current state of the status flags
RUB	grey/ yellow/ green	Rubidium Clock - Grey: no rubidium clock is installed. Yellow: Rubidium clock is warming up (takes approximately 300 s). Green: Rubidium clock is warmed up and locked.
BOX	grey/green	Boxcar - Green: indicates which of the boxcar units is enabled.
MOD	grey/green	MOD - Green: indicates which of the modulation kits is enabled.
PID	grey/green	PID - Green: indicates which of the PID units is enabled. Red: indicates PID unit is in PLL or ExtRef mode but is not locked. Yellow: indicates PID unit was not locked in the past.
PLL	grey/green	PLL - Green: indicates which of the PLLs is enabled.
Full Screen	⛶	Toggles the browser between full screen and normal mode.

5.2.2. Plot Functionality

Several tools provide a graphical display of measurement data in the form of plots. These are multi-functional tools with zooming, panning and cursor capability. This section introduces some of the highlights.

Plot Area Elements

Plots consist of the plot area, the X range and the range controls. The X range (above the plot area) indicates which section of the wave is displayed by means of the blue zoom region indicators. The two ranges show the full scale of the plot which does not change when the plot area displays a zoomed view. The two axes of the plot area instead do change when zoom is applied.

The mouse functionality inside of a plot greatly simplifies and speeds up data viewing and navigation.

Table 5.4: Mouse functionality inside plots

Name	Action	Description	Performed inside
Panning	left click on any location and move around	moves the waveforms	plot area
Zoom X axis	mouse wheel	zooms in and out the X axis	plot area
Zoom Y axis	shift + mouse wheel	zooms in and out the Y axis	plot area

Name	Action	Description	Performed inside
Window zoom	shift and left mouse area select	selects the area of the waveform to be zoomed in	plot area
Absolute jump of zoom area	left mouse click	moves the blue zoom range indicators	X and Y range, but outside of the blue zoom range indicators
Absolute move of zoom area	left mouse drag-and-drop	moves the blue zoom range indicators	X and Y range, inside of the blue range indicators
Full Scale	double click	set X and Y axis to full scale	plot area

Each plot area contains a legend that lists all the shown signals in the respective color. The legend can be moved to any desired position by means of drag-and-drop.

The X range and Y range plot controls are described in simpara_title.

Table 5.5: Plot control description

Control/ Tool	Option/ Range	Description
Axis scaling mode	Auto FS Man Auto	Selects between automatic, full scale and manual axis scaling.
Axis mapping mode	Lin Lin Log dB	Select between linear, logarithmic and decibel axis mapping.
Axis zoom in		Zooms the respective axis in by a factor of 2.
Axis zoom out		Zooms the respective axis out by a factor of 2.
Rescale axis to data		Rescale the foreground Y axis in the selected zoom area.
Save figure		Generates PNG, JPG or SVG of the plot area or areas for dual plots to the local download folder.
Save data		Generates a CSV file consisting of the displayed wave or histogram data (when histogram math operation is enabled). Select full scale to save the complete wave. The save data function only saves one shot at a time (the last displayed wave).
Cursor control	Off Off	Cursors can be switch On/Off and set to be moved both independently or one bound to the other one.
Net Link		Provides a LabOne Net Link to use displayed wave data in tools like Excel, MATLAB, etc.

Cursors and Math

The plot area provides two X and two Y cursors which appear as dashed lines inside of the plot area. The four cursors are selected and moved by means of the blue handles individually by means of drag-and-drop. For each axis, there is a primary cursor indicating its absolute position and a secondary cursor indicating both absolute and relative position to the primary cursor.

Cursors have an absolute position which does not change upon pan or zoom events. In case a cursor position moves out of the plot area, the corresponding handle is displayed at the edge of the plot area. Unless the handle is moved, the cursor keeps the current position. This functionality is very effective to measure large deltas with high precision (as the absolute position of the other cursors does not move).

The cursor data can also be used to define the input data for the mathematical operations performed on plotted data. This functionality is available in the Math sub-tab of each tool. The simpara_title gives an overview of all the elements and their functionality. The chosen Signals and Operations are applied to the currently active trace only.

Table 5.6: Plot math description

Control/ Tool	Option/Range	Description
Source Select		Select from a list of input sources for math operations.
	Cursor Loc	Cursor coordinates as input data.
	Cursor Area	Consider all data of the active trace inside the rectangle defined by the cursor positions as input for statistical functions (Min, Max, Avg, Std).
	Tracking	Display the value of the active trace at the position of the horizontal axis cursor X1 or X2.
	Plot Area	Consider all data of the active trace currently displayed in the plot as input for statistical functions (Min, Max, Avg, Std).
	Peak	Find positions and levels of up to 5 highest peaks in the data.
	Trough	Find positions and levels of up to 5 lowest troughs in the data.
	Histogram	Display a histogram of the active trace data within the x-axis range. The histogram is used as input to statistical functions (Avg, Std). Because of binning, the statistical functions typically yield different results than those under the selection Plot Area.
	Resonance	Display a curve fitted to a resonance.
	Linear Fit	Display a linear regression curve.
Operation Select		Select from a list of mathematical operations to be performed on the selected source. Choice offered depends on the selected source.
	Cursor Loc: X1, X2, X2-X1, Y1, Y2, Y2-Y1, Y2 / Y1	Cursors positions, their difference and ratio.
	Cursor Area: Min, Max, Avg, Std	Minimum, maximum value, average, and bias-corrected sample standard deviation for all samples between cursor X1 and X2. All values are shown in the plot as well.
	Tracking: Y(X1), Y(X2), ratioY, deltaY	Trace value at cursor positions X1 and X2, the ratio between these two Y values and their difference.
	Plot Area: Min, Max, Pk Pk, Avg, Std	Minimum, maximum value, difference between min and max, average, and bias-corrected sample standard deviation for all samples in the x axis range.
	Peak: Pos, Level	Position and level of the peak, starting with the highest one. The values are also shown in the plot to identify the peak.
	Histogram: Avg, Std, Bin Size, (Plotter tab only: SNR, Norm Fit, Rice Fit)	A histogram is generated from all samples within the x-axis range. The bin size is given by the resolution of the screen: 1 pixel = 1 bin. From this histogram, the average and bias-corrected sample standard deviation is calculated, essentially assuming all data points in a bin lie in the center of their respective bin. When used in the plotter tab with demodulator or boxcar signals, there additionally are the options of SNR estimation and fitting statistical distributions to the histogram (normal and rice distribution).
	Resonance: Q, BW, Center, Amp, Phase, Fit Error	A curve is fitted to a resonator. The fit boundaries are determined by the two cursors X1 and X2. Depending on the type of trace (Demod R or Demod Phase) either a Lorentzian or an inverse tangent function is fitted to the trace. The Q is the quality factor of the fitted curve. BW is the 3dB bandwidth (FWHM) of the fitted curve. Center is the center frequency. Amp gives the amplitude (Demod R only), whereas Phase returns the phase at the center frequency of the resonance (demod Phase only). The fit error is given by the normalized root-mean-square deviation. It is normalized by the range of the measured data.
	Linear Fit: Intercept, Slope, R^2	A simple linear least squares regression is performed using a QR decomposition routine. The fit boundaries are determined by the two cursors X1 and X2. The parameter outputs are the Y-axis intercept, slope and the R^2-value, which is the coefficient of determination to determine the goodness-of-fit.

Control/ Tool	Option/Range	Description
Add	Add	Add the selected math function to the result table below.
Add All	Add All	Add all operations for the selected signal to the result table below.
Clear Selected	Clear	Clear selected lines from the result table above.
Clear All	Clear All	Clear all lines from the result table above.
Copy	Copy	Copy selected row(s) to Clipboard as CSV
Unit Prefix		Adds a suitable prefix to the SI units to allow for better readability and increase of significant digits displayed.
CSV	CSV	Values of the current result table are saved as a text file into the download folder.
Net Link	Link	Provides a LabOne Net Link to use the data in tools like Excel, MATLAB, etc.
Help	Help	Opens the LabOne User Interface help.

Note

The standard deviation is calculated using the formula $\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2}$ for the unbiased estimator of the sample standard deviation with a total of N samples x_i and an arithmetic average $\bar{x}$. The formula above is used as-is to calculate the standard deviation for the Histogram Plot Math tool. For large number of points (Cursor Area and Plot Area tools), the more accurate pairwise algorithm is used (Chan et al., "Algorithms for Computing the Sample Variance: Analysis and Recommendations", The American Statistician 37 (1983), 242-247).

Tree Selector

The Tree selector allows one to access streamed measurement data in a hierarchical structure by checking the boxes of the signals that should be displayed. The tree selector also supports data selection from multiple instruments, where available. Depending on the tool, the Tree selector is either displayed in a separate Tree sub-tab, or it is accessible by a click on the button.

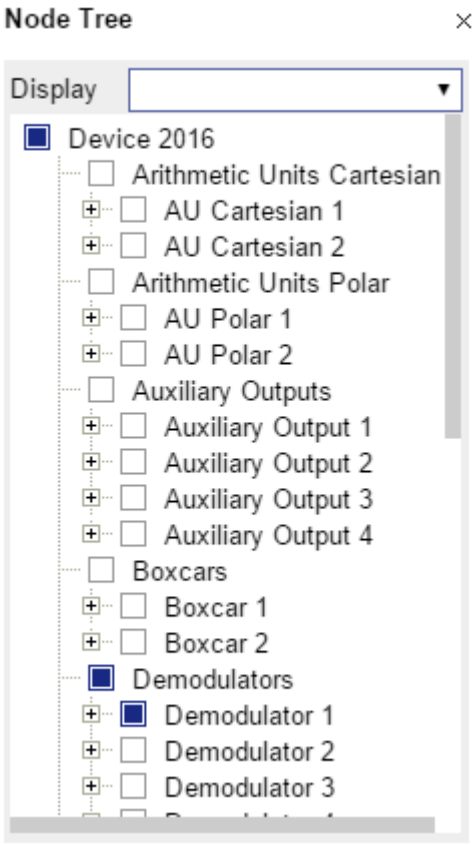


Figure 5.3: Tree selector with Display drop-down menu

Vertical Axis Groups

Vertical Axis groups are available as part of the plot functionality in many of the LabOne tools. Their purpose is to handle signals with different axis properties within the same plot. Signals with different units naturally have independent vertical scales even if they are displayed in the same plot. However, signals with the same unit should preferably share one scaling to enable quantitative comparison. To this end, the signals are assigned to specific axis group. Each axis group has its own axis system. This default behavior can be changed by moving one or more signals into a new group.

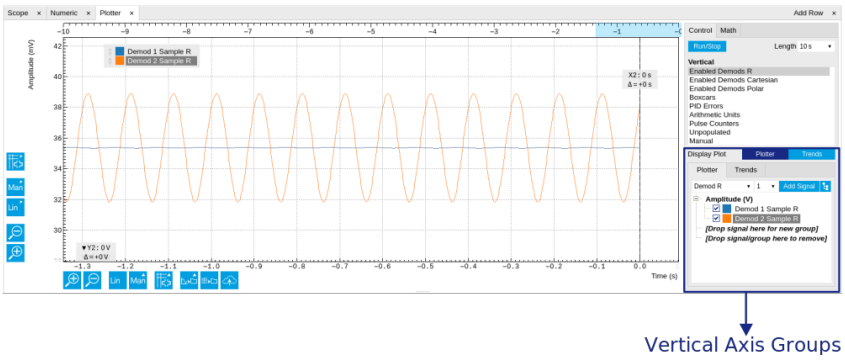


Figure 5.4: Vertical Axis Group in Plotter tool of the UHFLI Lock-in Amplifier

The tick labels of only one axis group can be shown at once. This is the foreground axis group. To define the foreground group click on one of the group names in the Vertical Axis Groups box. The current foreground group gets a high contrast color.

Select foreground group
Click on a signal name or group name inside the Vertical Axis Groups. If a group is empty the selection is not performed.

Split the default vertical axis group
Use drag-and-drop to move one signal on the field **[Drop signal here to add a new group]**. This signal will now have its own axis system.

Change vertical axis group of a signal
Use drag-and-drop to move a signal from one group into another group that has the same unit.

Group separation
In case a group hosts multiple signals and the unit of some of these signals changes, the group will be split in several groups according to the different new units.

Remove a signal from the group
In order to remove a signal from a group drag-and-drop the signal to a place outside of the Vertical Axis Groups box.

Remove a vertical axis group
A group is removed as soon as the last signal of a custom group is removed. Default groups will remain active until they are explicitly removed by drag-and-drop. If a new signal is added that match the group properties it will be added again to this default group. This ensures that settings of default groups are not lost, unless explicitly removed.

Rename a vertical axis group
New groups get a default name "Group of ...". This name can be changed by double-clicking on the group name.

Hide/show a signal
Uncheck/check the check box of the signal. This is faster than fetching a signal from a tree again.

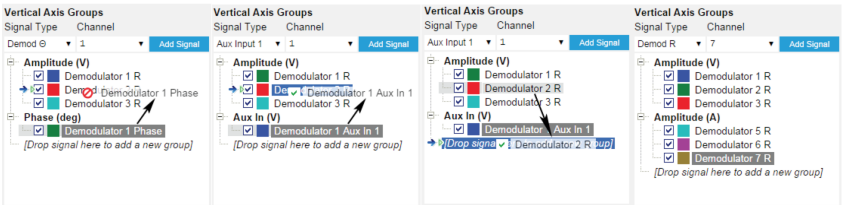


Figure 5.5: Vertical Axis Group typical drag and drop moves.

Table 5.7: Vertical Axis Groups description

Control/ Tool	Option/ Range	Description
Vertical Axis Group		Manages signal groups sharing a common vertical axis. Show or hide signals by changing the check box state. Split a group by dropping signals to the field [Drop signal here to add new group]. Remove signals by dragging them on a free area. Rename group names by editing the group label. Axis tick labels of the selected group are shown in the plot. Cursor elements of the active wave (selected) are added in the cursor math tab.
Signal Type	AU Polar	Select signal types for the Vertical Axis Group.
	Demod X, Y, R, Theta	
	Frequency	
	Aux Input 1, 2	
	HW Trigger	
	PID Error	
	PID Shift	
	PID Value	
	Boxcar	
	AU Cartesian	
Channel	integer value	Selects a channel to be added.
Signal	integer value	Selects signal to be added.

Control/ Tool	Option/ Range	Description
Add Signal	Add Signal	Adds a signal to the plot. The signal will be added to its default group. It may be moved by drag and drop to its own group. All signals within a group share a common y-axis. Select a group to bring its axis to the foreground and display its labels.
Window Length	2 s to 12 h	Window memory depth. Values larger than 10 s may cause excessive memory consumption for signals with high sampling rates. Auto scale or pan causes a refresh of the display for which only data within the defined window length are considered.

Trends

The Trends tool lets the user monitor the temporal evolution of signal features such as minimum and maximum values, or mean and standard deviation. This feature is available for the Scope tab. Using the Trends feature, one can monitor all the parameters obtained in the Math sub-tab of the corresponding tab.

The Trends tool allows the user to analyze recorded data on a different and adjustable time scale much longer than the fast acquisition of measured signals. It saves time by avoiding post-processing of recorded signals and it facilitates fine-tuning of experimental parameters as it extracts and shows the measurement outcome in real time.

To activate the Trends plot, enable the Trends button in the Control sub-tab of the corresponding main tab. Various signal features can be added to the plot from the Trends sub-tab in the Vertical Axis Groups . The vertical axis group of Trends has its own Run/Stop button and Length setting independent from the main plot of the tab. Since the Math quantities are derived from the raw signals in the main plot, the Trends plot is only shown together with the main plot. The Trends feature is only available in the LabOne user interface and not at the API level.

5.3. Architecture and Signalling

The UHFQA contains a number of functional blocks for signal generation and acquisition:

- The Arbitrary Waveform Generator represented in the AWG Tab
- The QA Input Monitor represented in the Quantum Analyzer Input Tab
- The QA Result Logger represented in the Quantum Analyzer Result Tab
- The Integration Units represented in the Quantum Analyzer Setup Tab
- The integration weight memory that can be configured and displayed in the Quantum Analyzer Input Tab
- The internal digital oscillator represented in the Inputs/Outputs Tab
- The internal trigger delay block represented in the Quantum Analyzer Setup Tab

Many of these blocks can operate independently and can be synchronized with external signals via the trigger inputs and outputs. In order to control the relative timing of the signal processing in these different blocks, internal trigger signals are used. Namely, the Arbitrary Waveform Generator can issue trigger signals in order to start the integration units, the input monitor, or the internal oscillator. The following block diagram gives an overview of the interconnection of these different blocks and illustrates the function of key user interface settings found in the user interface tabs listed above.

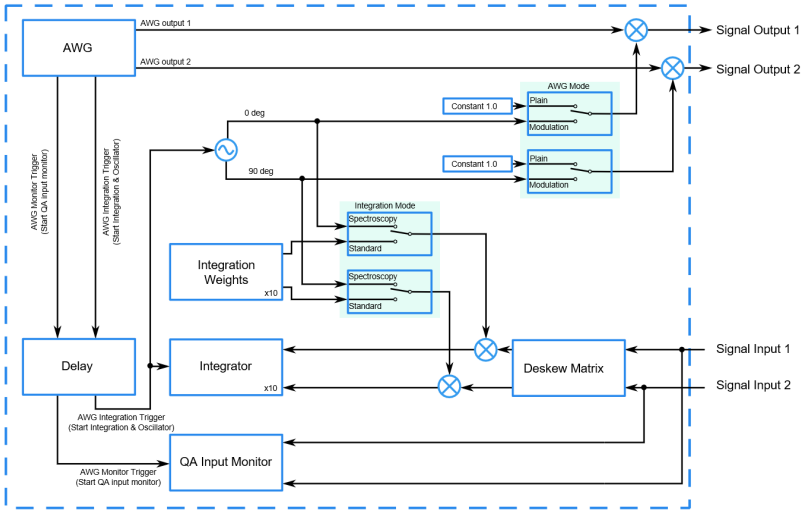


Figure 5.6: Block diagram of the core part of the UHFQA instrument

5.4. Quantum Analyzer Setup Tab

The Quantum Analyzer Setup is the main control panel for the qubit measurement unit on the instrument (see functional overview for an overview block diagram). It is available on all UHFQA instruments.

5.4.1. Features

- Raw signal deskew
- Signal rotation in I/Q plane
- 10×10 crosstalk subpression matrix
- Threshold operation
- Qubit-qubit correlation analysis

5.4.2. Description

Table 5.8: App icon and short description

Control/Tool	Option/Range	Description
QA Setup	|0⟩⟨0|	Configure the Qubit Measurement Unit

The Quantum Analyzer Setup tab (see LabOne UI: Quantum Analyzer Setup tab) is divided into different sections each representing a signal processing step starting from raw signal deskew to the final threshold operation that transforms an analog I/Q signal into a discrete qubit state. This tab represents the interface to the following functional blocks: the integration units, the internal oscillator, the crosstalk subpression matrix, the deskew matrix, the correlation unit, the statistics unit, and the thesholding unit. A block diagram representing the flow of data and trigger signals between the functional blocks is found in Architecture and Signalling.

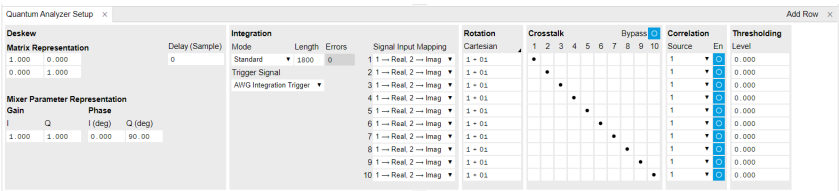


Figure 5.7: LabOne UI: Quantum Analyzer Setup tab

Two-element vectors of samples arrive at 1.8 GSa/s from Signal Inputs 1 and 2. In the **Deskew** section, at this rate, each sample vector is multiplied by a 2×2 Rotation/Gain matrix. The default value is the identity matrix [1, 0; 0, 1] which leaves both input signals unchanged. Changing this to a different value allows the user to compensate for signal imperfections such as analog linear crosstalk, or mixer amplitude imbalance.

In the **Integration** section, each of the two input signals is demodulated by multiplying with a reference signal and the product is integrated over a fixed time T after reception of a trigger. The user can choose the mode of operation of the weighted integration. In Standard mode, the reference signal is given by the Integration Weights programmed to the instrument memory using the Quantum Analyzer Input Tab. This mode offers the full flexibility to define a custom integration weight to realize a matched filter. In Spectroscopy mode, the reference signal is given by sine and cosine signals generated by the internal digital oscillator controlled from the Inputs/Outputs Tab, which allows for longer integration times and thus higher frequency resolution than the Standard mode. The input signals $V_{Sig\ In\ 1}(t)$ and $V_{Sig\ In\ 2}(t)$ of duration T are reduced to a single pair of voltages V_I and V_Q. Since there are 10 separate qubit measurement units, there can be up to 10 pairs ($V_{I,q}$, $V_{Q,q}$ for q=1...10), each corresponding to one frequency component of the signals $V_{\{Sig\ In\ 1\}}(t)$ and $V_{\{Sig\ In\ 2\}}(t)$.

The **Rotation** section rotates and scales the signal in the complex plane after integration. For each of the 10 channels, the rotation is characterized by a complex number $z_q = x_q + iy_q = r_q \times \exp(i\theta_q)$. The demodulated signal is multiplied with z_q: $V'_{I,q} + iV'_{\{Q,q\}} = (V_{\{I,q\}} + iV_{\{Q,q\}})_{\times z}q$. The user may specify z_q in polar coordinates in the form "r @ θ" or in Cartesian coordinates in the form "x + y i". Examples are "1@45" for a 45 degree rotation, or "0.0 + 1.0 i" for a 90 degree rotation. The purpose of the rotation step is to ensure that the readout contrast is shifted into the in-phase signal component, i.e., that the state of qubit q only affects $V'_{\{I,q\}}$ but not $V'_{\{Q,q\}}$.

The **Crosstalk** section is a graphical representation of the 10×10 crosstalk subpression matrix C that subports systematic minimization of the influence of one qubit's state on another qubit's readout signal. The signal processing up to after the rotation step can be abstracted as a 10×10 matrix M that transforms the vector of qubit states (s_1,...,s_{10}), with s_q = 0 or 1, into the vector of signals ($V'_{\{I,1\}}$, ... $V'_{\{I,10\}}$). This matrix can be measured systematically by preparing the qubit system in different states of the form (0,..., 0, 1, 0,..., 0), and measuring the resulting signal. Using the Crosstalk Suppression optimally relies on finding the matrix C such that C×M is diagonal. Due to the complexity of this method, setting the elements of the crosstalk subpression matrix C from the graphical UI would be impractical, and its elements can only be set from the API. We denote the signals after crosstalk subpression with a double prime as $(V''_{\{I,1\}}, \ldots V''_{\{I,10\}}) = C \times (V'_{\{I,1\}}, \ldots V'_{\{I,10\}})$.

The **Correlation** section optionally enables the outputs of two readout channels to be multiplied prior to averaging, logging, etc. When enabled, the corresponding channel is multiplied with another channel selected as the Source.

The **Thresholds** section allows one to define a voltage threshold to transform the in-phase quadrature $V''_{\{I,q\}}$ of the readout signal into a discrete qubit state, 0 or 1.

5.4.3. Functional Elements

Table 5.9: Quantum Analyzer Setup tab

Control/Tool	Option/ Range	Description
Rotation/Gain Matrix		Implements a 2×2 matrix multiplication. The two input signals are treated as a vector with two elements and the result is a vector as well.
In-Phase Gain		Gain of in-phase branch
Quadrature Gain		Gain of quadrature branch
In-Phase Phase		Phase of in-phase branch
Quadrature Phase		Phase of quadrature branch
Mode		Application mode.
	Standard	The integration weights are given by the user-programmed filter memory.
	Spectroscopy	The integration weights are generated by a digital oscillator. This mode offers enhanced frequency resolution.
Readout Trigger Selection		Select the source for triggering the readout.
	Trigger In 1	Use the Trigger In 1 as the trigger signal.

Control/Tool	Option/ Range	Description
	Trigger In 2	Use the Trigger In 2 as the trigger signal.
	Trigger In 3	Use the Trigger In 3 as the trigger signal.
	Trigger In 4	Use the Trigger In 4 as the trigger signal.
	AWG Integration Trigger	Use the AWG Integration Trigger as the trigger signal.
Spectroscopy Trigger Selection		Selects the source for triggering the spectroscopy.
Clear		Empty all Integration Weights memory slots.
Integration Length		The integration time of all weighted integration units specified in units of samples. In Standard mode, a maximum of 4096 samples can be integrated, which corresponds to 2.3 µs. In Spectroscopy mode, a maximum of 16.7 MSa can be integrated, which corresponds to ~10 ms.
Errors		Number of hold-off violations detected in the INTEGRATION unit since last reset.
Delay		A delay time in units of samples that adjusts the time at which the integration starts in relation to the trigger signal of the weighted integration units.
Source		Controls the routing of the input signals to the INTEGRATION units.
	1 -> Real, 2 -> Imag	Signal input 1 to real part, Signal input 2 to imaginary part.
	2 -> Real, 1 -> Imag	Signal input 2 to real part, Signal input 1 to imaginary part.
	1 -> Real, 1 -> Imag	Signal input 1 to real part, Signal input 1 to imaginary part.
	2 -> Real, 2 -> Imag	Signal input 2 to real part, Signal input 2 to imaginary part.
Rotation		Complex rotation coefficient applied to the signals after integration.
Representation		Select between Cartesian and polar representation of the complex rotation coefficient. Cartesian coordinates are entered in the format "x+iy", polar coordinates in the format "r@a" where x, y, r, and a are real numbers.
Crosstalk		Graphical representation of the 10×10 crosstalk suppression matrix. Positive values are black, negative values are red.
Bypass Crosstalk		Bypass the Crosstalk matrix in order to reduce the latency through the system.
Bypass Rotation		Bypass Rotation in order to reduce the latency through the system.
Bypass Deskew		Bypass Deskew in order to reduce the latency through the system.
En		Enable the correlation mode for the given channel.
Source		Controls the channel with which correlation should be made. Selecting the same channel as the readout channel number corresponds to self-correlation.
Level		The discretization level applied to the output of the Crosstalk Suppression matrix.
Length		Sets the integration length in spectroscopy mode in number of samples. A maximum of 33.5 MSa (2^25 samples) can be integrated, which corresponds to 16.7 ms.
Delay		Sets the delay of the start of the integration in spectroscopy mode with respect to the Trigger Signal.

Control/Tool	Option/ Range	Description
PSD Enable		Enable the power spectral density mode of the spectroscopy mode.
Offset Frequency		Sets the digital oscillator frequency. The sum of the Offset Frequency and the Center Frequency correspond to the frequency of the microwave tone at the Out connector.
Output Frequency		Displays the carrier frequency of the microwave signal at the Out connector. This frequency corresponds to the sum of the Center Frequency and the Offset Frequency.
Amplitude		Amplitude of the microwave signal relative to the range of the Output.
Set Mode		Set Generator Waveform by parametric generation or CSV Upload.
	Parametric	Generator Waveform are generated by defining sine wave parameters.
	Upload	Generator Waveform are uploaded by the user in a form of a CSV file.
Frequency		Frequency of the complex exponential function.
Phase		Phase of the complex exponential function.
Window Type		Window function to be applied to the complex exponential function.
Window Length		Length of the selected window in samples, starting from 0.
Waveform Memory		Selects the waveform memory for parametric or arbitrary waveform upload.
Set To device		Write the real and imaginary part of the Waveform to the selected Waveform Memory.
Set To Device		Write the real and imaginary part of the Waveform to the selected Waveform Memory.
CSV File		Drag and drop CSV file containing columns of Sequencer Waveform.
Amplitude		Amplitude of the complex exponential function.
Clear		Empty all Readout Waveform memory slots.
Sequencer		Runs the Sequencer.
Status	Running, Idle, Waiting	Displays the status of the sequencer on the instrument. Off: Ready, not running. Green: Running, not waiting for any trigger event. Yellow: Running, waiting for a trigger event. Red: Not ready (e.g., pending elf download, no elf downloaded)

5.5. Quantum Analyzer Input Tab

The Quantum Analyzer Input tab is the interface to the input monitor unit of the instrument (see functional overview for an overview block diagram). It is available on all UHFQA instruments.

5.5.1. Features

- Input Monitor with 4 kSa memory for raw signal display
- Raw signal averaging
- Display of integration weights
- Setting of basic integration weights with
- CSV upload of integration weights

5.5.2. Description

Table 5.10: App icon and short description

Control/Tool	Option/Range	Description
QA Input	|0⟩⟨0|	Configure the Weighted Integration units and Monitoring Scope

The Quantum Analyzer Setup tab (see LabOne UI: Quantum Analyzer Input tab) is divided into a display section on the left and a configuration section on the right. This tab represents the interface to the following functional blocks of the input monitor and the integration weight memory. A block diagram representing the flow of data and trigger signals between the functional blocks is found in Architecture and Signalling.

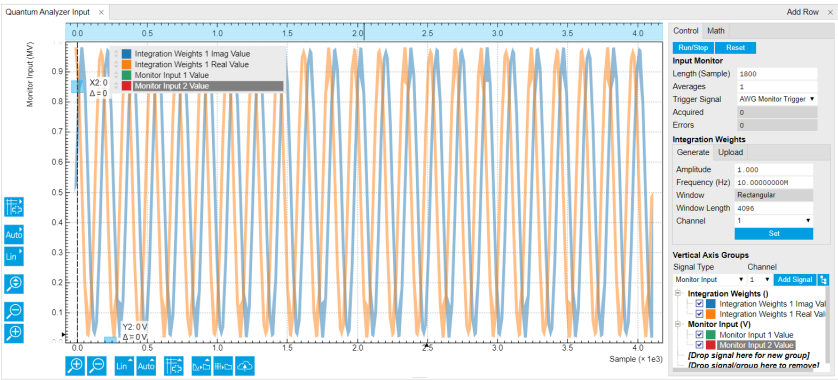


Figure 5.8: LabOne UI: Quantum Analyzer Input tab

The Input Monitor is an averaging scope with 4 kSa memory dedicated to measuring the raw signal response to a readout pulse containing a single or multiple carrier frequencies. The Input Monitor is triggered by the UHF-AWG via one dedicated line of the internal trigger. In an AWG Sequencer program (see AWG Tab), this trigger is activated by setting the corresponding bit to high and to low again using the following instructions:

```
setTrigger(AWG_MONITOR_TRIGGER);
setTrigger(0);
```

The Integration Weights section represents the integration weights used by the qubit measurement unit. In the typical case, the readout pulse is generated by the UHF-AWG on Signal Outputs 1 and 2, upconverted to a sideband of a local oscillator LO using an I/Q mixer, it passes through the device under test, gets downconverted using an I/Q mixer and the same local oscillator LO, and is acquired by the UHFQA on Signal Inputs 1 and 2. The simplest type of weight function are sinusoid functions at the sideband frequency generated by the AWG. These types of weight functions can easily be written to the instrument memory by selecting a carrier frequency (positive or negative), length in samples, channel, and then clicking on Set .This basic type corresponds to an unmatched filter with a sinc profile in frequency space. For more advanced measurements using matched filters, the weight function can be freely programmed using the API, or using the File Upload sub-tab.

Similarly to the Input Monitor, the weighted integration is controlled by dedicated internal trigger lines. There is one trigger line to arm the trigger of the weighted integration which can be accessed using the sequencer constant `AWG_INTEGRATION_ARM`. A second line is used to generate the actual trigger and can be accessed using the constant `AWG_INTEGRATION_TRIGGER`. The following instructions can be used as a template for use in an AWG sequence program:

```
setTrigger(AWG_INTEGRATION_ARM); // arm the integration unit
(...)
setTrigger(AWG_INTEGRATION_ARM + AWG_INTEGRATION_TRIGGER); // trigger the
integration unit
setTrigger(AWG_INTEGRATION_ARM); // reset the trigger
```

5.5.3. Functional Elements

Table 5.11: Quantum Analyzer Input tab

Control/Tool	Option/ Range	Description
Run/Stop		Run the Input Monitor scope.

Control/Tool	Option/ Range	Description
Length		The duration of each capture in samples. A maximum of 4096 samples can be captured, which corresponds to 2.3 µs.
Average Count		Number of averages to perform.
Monitor Trigger Selection		Select the source for triggering the input monitor.
	Trigger In 1	Use the Trigger In 1 as the trigger signal.
	Trigger In 2	Use the Trigger In 2 as the trigger signal.
	Trigger In 3	Use the Trigger In 3 as the trigger signal.
	Trigger In 4	Use the Trigger In 4 as the trigger signal.
	AWG Monitor Trigger	Use the AWG Monitor Trigger as the trigger signal.
Acquired		Indicates the index of the acquisition that will be performed on the next trigger.
Errors		Number of hold-off errors detected since last reset.
Reset		Clear the Input Monitor scope.
Amplitude		Amplitude of the complex exponential function.
Set Mode		Set Integration Weights by parametric generation or CSV Upload.
	Parametric	Integration Weights are generated by defining sine wave parameters.
	Upload	Integration Weights are uploaded by the user in a form of a CSV file.
Frequency		Frequency of the complex exponential function.
Phase		Phase of the complex exponential function.
Window Type		Window function to be applied to the complex exponential function.
Window Length		Length of the selected window starting from zero position.
Integration Weight		Selects the Integration Weight for parametric or arbitrary waveform upload.
Set To device		Write the real and imaginary part of the Waveform to the selected Integration Weight.
Set To Device		Write the real and imaginary part of the Waveform to the selected Integration Weight.
CSV File		Drag and drop CSV file containing columns of integration weights.

5.6. Quantum Analyzer Result Tab

The Quantum Analyzer Result tab is the interface to the Result Logger unit of the instrument and displays processed data after the qubit measurement unit (see functional overview for an overview block diagram). It is available on all UHFQA instruments.

5.6.1. Features

- Acquisition and display of measurement data
- Multiple probe points: after Rotation, Crosstalk Suppression, Threshold, Correlation
- Multi-qubit state statistical analysis

5.6.2. Description

Table 5.12: App icon and short description

Control/Tool	Option/Range	Description
QA Result		Configure the Result Logger.

The Quantum Analyzer Result tab (see LabOne UI: Quantum Analyzer Result tab) is divided into a display section on the left and a configuration section on the right.

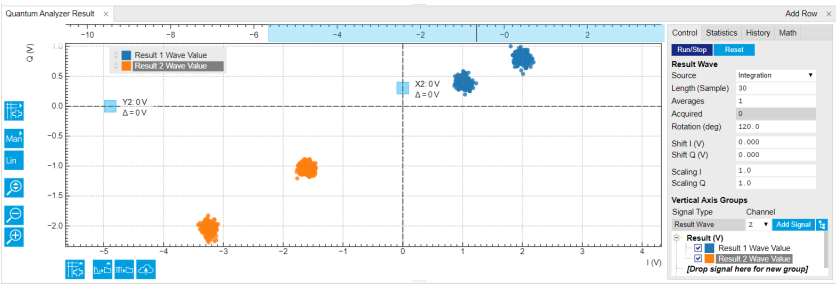


Figure 5.9: LabOne UI: Quantum Analyzer Result tab

This tool allows you to acquire, average, and analyze, large sets of data sourced at various points of the signal processing chain. The Source setting in the Control sub-tab lets you choose the probe point. The available sources are:

- After Rotation: voltages $V'_{I,q}$ after rotation in the complex plane
- After Crosstalk Suppression: voltages $V''_{I,q}$ after application of the Crosstalk Suppression matrix C
- After Voltage Correlation: product of voltages $V''_{I,q} \times V''_{I,q'}$ corresponding to two qubits q and q'
- After Threshold: discretized states s_q
- After State Correlation: product of discretized states $s_q \times s_{q'}$ corresponding to two qubits q and q'
- Statistics, before State Correlation: result of the Statistics Unit
- Statistics, after State Correlation: result of the Statistics Unit

The data are stored in a vector with a length of up to 10_6 points and displayed in the plot area on the left once the acquisition is complete. The Result Logger subports averaging of multiple vectors which is enabled by setting the Averages to any number higher than 1.

The Statistics sub-tab allows one to compress the information from many measurements into just a few numbers. The Statistics Unit measures the states s_q after the threshold operation and determines three values for each readout channel q:

1. The number of times the result of the measurement was a logic '1'. This value is called Ones.
2. The number of times the result of the measurement differed from the result of the previous measurement. This value is called Flips.
3. Number of state errors. A state error occurs when there is a discrepancy between the measured state of this channel and the state that is predicted based on the configured multi-qubit state table.

The Length setting determines the number of measurements s_q to take into account to determine these values.

The statistics values are best explained with an example. Consider that on channel 1 we read the sequence of results "0101001010", which contains four ones and eight flips, and for channel 2 we read the sequence of results "0011000110". Consider that we have set the state map to the array (1, 2, 3, 0), or (0b01, 0b10, 0b11, 0b00) in binary notation. The state map is set using the API. This state map defines what we expect for the next two-qubit state based on the current two-qubit state in the sequence according to the following table.

Table 5.13: Qubit state map example

Current two-qubit state	Expected next two-qubit state
00	01
01	10
10	11
11	00

The actual measured state sequence is 00, 01, 10, 11, 00, 00, 01, 10, 11, 00. From this sequence and the state map we can see that there is an error on channel 1 between the fifth and sixth measurement. Measurement five and six both result in the state 00 being measured. However, according to the state table, we expect the state 01 to follow the state 00. Therefore, there is a state error on channel 1. The user interface also reports the total number of state errors for the whole measurement, which is 1.

5.6.3. Functional Elements

Table 5.14: Quantum Analyzer Result tab

Control/ Tool	Option/ Range	Description
Run/Stop		Run the Result Logger.
Reset		Clear the Result Logger.
Acquired		Indicates when data point is recorded.
Display Source		Selects the signal source for the Result Logger.
	Crosstalk	Result after crosstalk unit.
	Threshold	Result after Threshold unit.
	Rotation	Result after Rotation unit.
	Crosstalk Correlation	Correlation unit after the Crosstalk unit.
	Threshold Correlation	Correlation unit after the Threshold unit.
Length		Number of data points to record. One data point corresponds to a single averaged output of the selected source.
Averages		Number of averages per recorded data point.
Acquired		Indicate the index of the data point that will be recorded next.
Mode		Defines the averaging mode of the Result Logger.
	Cyclic	The first point of the Result vector is the average of the results number 1, M+1, 2M+1, and so forth, where M is equal to the Length setting. The second point is the average of the results number 2, M+2, 2M+2, and so forth.
	Sequential	The first point of the Result vector is the average of the first N results, where N is equal to the Averages setting. The second point of the Result vector is the average of the following N results, and so forth.
Rotation		Rotation angle applied to the recorded complex values.
Shift I		Translation shift applied to the I component of the recorded data points.
Shift Q		Translation shift applied to the Q component of the recorded data points.
Scaling I		Scaling factor applied to the I component of the recorded data points.
Scaling Q		Scaling factor applied to the Q component of the recorded data points.
Enable		Enable the Statistics Unit.
Reset		Clear the Statistics Unit.
Length		Number of measurements over which the Statistics Unit should operate.
Total Errors		Total number of state errors. The total number of state errors is incremented if any of the measured bits show a state error. The value will be identical for all channels.
Ones		Number of logical ones measured for this channel.
Flips		Number of flips measured for this channel. A flip is defined as a change in qubit state from one measurement to the next.

Control/ Tool	Option/ Range	Description
Errors		Number of state errors measured for this channel. A state error occurs when there is a discrepancy between the measured state of this channel and the state that is predicted based on the configured state table.
Plot Type		Switch between complex wave dot plot and complex wave components (X, Y, Amplitude and Phase) plot.

5.7. Scope Tab

The Scope is a powerful time domain and frequency domain measurement tool as introduced in Unique Set of Analysis Tools and is available on all UHFQA instruments.

5.7.1. Features

- One input channel with 64 kSa of memory
- 12 bit nominal resolution
- Fast Fourier Transform (FFT): up to 900 MHz span, spectral density and power conversion, choice of window functions
- Sampling rates from 27 kSa/s to 1.8 GSa/s; up to 36 µs acquisition time at 1.8 GSa/s or 2.3 s at 27 kSa/s
- Up to 8 trigger sources and 2 trigger methods
- Independent hold-off, hysteresis, pre-trigger and trigger level settings `
- Support for Input Scaling and Input Units

5.7.2. Description

The Scope tab serves as the graphical display for time domain data. Whenever the tab is closed or an additional one of the same type is needed, clicking the following icon will open a new instance of the tab.

Table 5.15: App icon and short description

Control/ Tool	Option/ Range	Description
Scope		Displays shots of data samples in time and frequency domain (FFT) representation.

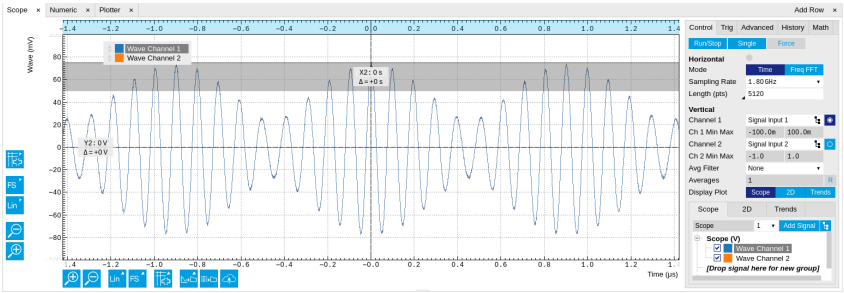


Figure 5.10: LabOne UI: Scope tab - Time domain

The Scope tab consists of a plot section on the left and a configuration section on the right. The configuration section is further divided into a number of sub-tabs. It gives access to a single-channel oscilloscope that can be used to monitor a choice of signals in the time or frequency domain. Hence the X axis of the plot area is time (for time domain display, Figure 5.10) or frequency (for frequency domain display, Figure 5.12). It is possible to display the time trace and the associated FFT simultaneously by opening a second instance of the Scope tab. The Y axis displays the selected signal that can be modified and scaled using the arbitrary input unit feature found in the Lock-in tab.

The Scope records data from a single channel at up to 1.8 GSa/s. The channel can be selected among the two Signal Inputs, Auxiliary Inputs, Trigger Inputs and AWG Trigger/Marker signals. The Scope records data sets of up to 64 kSa samples in the standard configuration, which corresponds to an acquisition time of 36 µs at the highest sampling rate.

The product of the inverse sampling rate and the number of acquired points (Length) determines the total recording time for each shot. Hence, longer time intervals can be captured by reducing the sampling rate. The Scope can perform sampling rate reduction either using decimation or BW Limitation as illustrated in Figure 5.11. BW Limitation is activated by default, but it can be deactivated in the Advanced sub-tab. The figure shows an example of an input signal at the top, followed by the Scope output when the highest sample rate of 1.8 GSa/s is used. The next signal shows the Scope output when a rate reduction by a factor of 4 (i.e. 450 MSa/s) is configured and the rate reduction method of decimation is used. For decimation, a rate reduction by a factor of N is performed by only keeping every N^{th} sample and discarding the rest. The advantage of this method is its simplicity, but the disadvantage is that the signal is undersampled because the input filter bandwidth of the UHFQA instrument is fixed at 600 MHz. As a consequence, the Nyquist sampling criterion is no longer satisfied and aliasing effects may be observed. The default rate reduction mechanism of BW Limitation is illustrated by the lowermost signal in the figure. BW Limitation means that for a rate reduction by a factor of N, each sample produced by the Scope is computed as the average of N samples acquired at the maximum sampling rate. The effective signal bandwidth is thereby reduced and aliasing effects are largely suppressed. As can be seen from the figure, with a rate reduction by a factor of 4, every output sample is simply computed as the average of 4 consecutive samples acquired at 1.8 GSa/s.

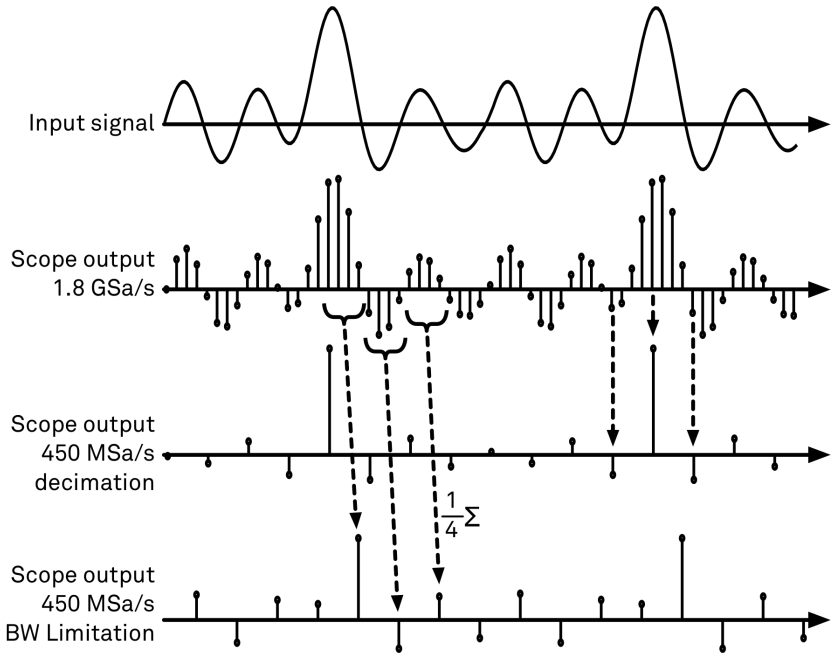


Figure 5.11: Illustration of how the Scope output is generated in BW Limitation and decimation mode when the sampling rate is reduced from the default of 1.8 GSa/s to 450 MSa/s

The frequency domain representation is activated in the Control sub-tab by selecting Freq Domain FFT as the Horizontal Mode. It allows the user to observe the spectrum of the acquired shots of samples. All controls and settings are shared between the time domain and frequency domain representations.

The Scope supports averaging over multiple shots. The functionality is implemented by means of an exponential moving average filter with configurable filter depth. Averaging helps to suppress noise components that are uncorrelated with the main signal. It is particularly useful in combination with the Frequency Domain FFT mode where it can help to reveal harmonic signals and disturbances that might otherwise be hidden below the noise floor.

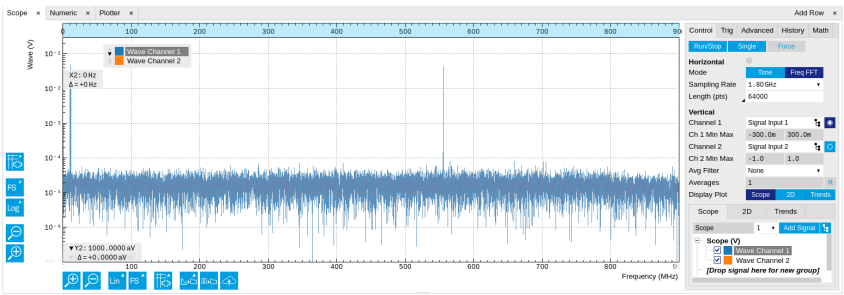


Figure 5.12: LabOne UI: Scope tab - Frequency domain

The Trigger sub-tab offers all the controls necessary for triggering on different signal sources. When the trigger is enabled, then oscilloscope shots are acquired whenever the trigger conditions are met. Trigger and Hysteresis levels can be indicated graphically in the plot. A disabled trigger is equivalent to continuous oscilloscope shot acquisition.

5.7.3. Functional Elements

Table 5.16: Scope tab: Control sub-tab

Control/ Tool	Option/ Range	Description
Run/Stop	Run/Stop	Runs the scope/FFT continuously.
Single	Single	Acquires a single shot of samples.
Force	Force	Force a trigger event.
Mode	Freq Domain (FFT)	Switches between time and frequency domain display.
	Time Domain	
Sampling Rate	27.5 kSa/s to 1.8 GSa/s	Defines the sampling rate of the scope. The numeric values are rounded for display purposes. The exact values are equal to the base sampling rate divided by 2^n, where n is an integer.
Sampling Rate		Defines the sampling rate of the scope. The numeric values are rounded for display purposes. The exact values are equal to the base sampling rate divided by 2^n, where n is an integer. Warning: Due to the lack of sample averaging feature, reduced sampling rates can cause aliasing and thus artifacts in the signal spectrum. Currently, the Scope tool only supports sample decimation, but in the future it will also offer sample averaging.
Length Mode		Switches between length and duration display.
	Length (pts)	The scope shot length is defined in number of samples. The duration is given by the number of samples divided by the sampling rate. The UHF-DIG option greatly increases the available length.
	Duration (s)	The scope shot length is defined as a duration. The number of samples is given by the duration times the sampling rate.
Length (pts) or Duration (s)	numeric value	Defines the length of the recorded scope shot. Use the Length Mode to switch between length and duration display.
Channel 1/2		Selects the signal source for the corresponding scope channel. Navigate through the tree view that appears and click on the required signal. Note: Channel 2 requires the DIG option.
Min	numeric value	Lower limit of the scope full scale range. For demodulator, PID, Boxcar, and AU signals the limit should be adjusted so that the signal covers the specified range to achieve optimal resolution.
Max	numeric value	Upper limit of the scope full scale range. For demodulator, PID, Boxcar, and AU signals the limit should be adjusted so that the signal covers the specified range to achieve optimal resolution.
Enable	ON / OFF	Activates the display of the corresponding scope channel. Note: Channel 2 requires the DIG option.
Average Filter		Enable Exponential Moving Average (EMA) filter that is applied when the average of several scope shots is computed and displayed. Depending on the mode, the source data for averaging is either the Time or the Freq FFT trace.
	Off	Averaging is turned off.
	On	Consecutive scope shots are averaged with an exponential weight.
Averages	integer value	The number of shots required to reach 63% settling. Twice the number of shots yields 86% settling.

Control/ Tool	Option/ Range	Description
Averages	integer value	The number of shots to average on the device before returning the data.
Reset	R	Resets the averaging filter.

For the Vertical Axis Groups, please see the table "Vertical Axis Groups description" in the section called "Vertical Axis Groups".

Table 5.17: Scope tab: Trigger sub-tab

Control/ Tool	Option/Range	Description
Trigger	grey/green/ yellow	When flashing, indicates that new scope shots are being captured and displayed in the plot area. The Trigger must not necessarily be enabled for this indicator to flash. A disabled trigger is equivalent to continuous acquisition. Scope shots with data loss are indicated by yellow. Such an invalid scope shot is not processed.
Enable	ON / OFF	When triggering is enabled scope data are acquired every time the defined trigger condition is met. If disabled, scope shots are acquired continuously.
Signal		Selects the trigger source signal. Navigate through the tree view that appears and click on the required signal.
Slope	Rising or falling edge trigger	Select the signal edge that should activate the trigger.
	None	
	Rising edge trigger	
	Falling edge trigger	
Level (V)	trigger signal range (negative values permitted)	Defines the trigger level.
Hysteresis Mode		Selects the mode to define the hysteresis strength. The relative mode will work best over the full input range as long as the analog input signal does not suffer from excessive noise.
	Hysteresis (V)	Selects absolute hysteresis.
	Hysteresis (%)	Selects a hysteresis relative to the adjusted full scale signal input range.
Hysteresis (V)	trigger signal range (positive values only)	Defines the voltage the source signal must deviate from the trigger level before the trigger is rearmed again. Set to 0 to turn it off. The sign is defined by the Edge setting.
Hysteresis (%)	numeric percentage value (positive values only)	Hysteresis relative to the adjusted full scale signal input range. A hysteresis value larger than 100% is allowed.
Show Level	ON / OFF	If enabled shows the trigger level as grey line in the plot. The hysteresis is indicated by a grey box. The trigger level can be adjusted by drag and drop of the grey line.
Trigger Gating		Select the signal source used for trigger gating if gating is enabled. This feature requires the UHF-DIG option.
	Trigger In 3 High	Only trigger if the Trigger Input 3 is at high level.
	Trigger In 3 Low	Only trigger if the Trigger Input 3 is at low level.
	Trigger In 4 High	Only trigger if the Trigger Input 4 is at high level.
	Trigger In 4 Low	Only trigger if the Trigger Input 4 is at low level.

Control/ Tool	Option/Range	Description
Trigger Gating Enable	ON / OFF	If enabled the trigger will be gated by the trigger gating input signal. This feature requires the UHF-DIG option.
Holdoff Mode		Selects the holdoff mode.
	Holdoff (s)	Holdoff is defined as time.
	Holdoff (events)	Holdoff is defined as number of events.
Holdoff (s)	numeric value	Defines the time before the trigger is rearmed after a recording event.
Holdoff (events)	1 to 1048575	Defines the trigger event number that will trigger the next recording after a recording event. A value one will start a recording for each trigger event.
Reference (%)	percent value	Trigger reference position relative to the plot window. Default is 50% which results in a reference point in the middle of the acquired data.
Delay (s)	numeric value	Trigger position relative to reference. A positive delay results in less data being acquired before the trigger point, a negative delay results in more data being acquired before the trigger point.
Enable	ON / OFF	Enable segmented scope recording. This allows for full bandwidth recording of scope shots with a minimum dead time between individual shots. This functionality requires the DIG option.
Segments	1 to 32768	Specifies the number of segments to be recorded in device memory. The maximum scope shot size is given by the available memory divided by the number of segments. This functionality requires the DIG option.
Shown Trigger	integer value	Displays the number of triggered events since last start.
Plot Type		Select the plot type.
	None	No plot displayed.
	2D	Display defined number of grid rows as one 2D plot.
	Row	Display only the trace of index defined in the Active Row field.
	2D + Row	Display 2D and row plots.
Active Row	integer value	Set the row index to be displayed in the Row plot.
Track Active Row	ON / OFF	If enabled, the active row marker will track with the last recorded row. The active row control field is read-only if enabled.
Palette	Solar	Select the colormap for the current plot.
	Viridis	
	Inferno	
	Balance	
	Turbo	
	Grey	
Colorscale	ON / OFF	Enable/disable the colorscale bar display in the 2D plot.
Mapping		Mapping of colorscale.
	Lin	Enable linear mapping.
	Log	Enable logarithmic mapping.
	dB	Enable logarithmic mapping in dB.
Scaling	Full Scale/ Manual/Auto	Scaling of colorscale.

Control/ Tool	Option/Range	Description
Clamp To Color	ON / OFF	When enabled, grid values that are outside of defined Min or Max region are painted with Min or Max color equivalents. When disabled, Grid values that are outside of defined Min or Max values are left transparent.
Start	numeric value	Lower limit of colorscale. Only visible for manual scaling.
Stop	numeric value	Upper limit of colorscale. Only visible for manual scaling.

Table 5.18: Scope tab: Advanced sub-tab

Control/ Tool	Option/ Range	Description
FFT Window	Cosine squared (ring-down)	Several different FFT windows to choose from. Each window function results in a different trade-off between amplitude accuracy and spectral leakage. Please check the literature to find the window function that best suits your needs.
	Rectangular	
	Hann	
	Hamming	
	Blackman Harris	
	Flat Top	
	Exponential (ring-down)	
	Cosine (ring-down)	
Resolution (Hz)	mHz to Hz	Spectral resolution defined by the reciprocal acquisition time (sample rate, number of samples recorded).
Absolute Frequency	ON / OFF	Shifts x-axis labeling to show the absolute frequency in the center as opposed to 0 Hz, when turned off.
Spectral Density	ON / OFF	Calculate and show the spectral density. If power is enabled the power spectral density value is calculated. The spectral density is used to analyze noise.
Power	ON / OFF	Calculate and show the power value. To extract power spectral density (PSD) this button should be enabled together with Spectral Density.
Persistence	ON / OFF	Keeps previous scope shots in the display. The color scheme visualizes the number of occurrences at certain positions in time and amplitude by a multi-color scheme.
BW Limit		Selects between sample decimation and sample averaging. Averaging avoids aliasing, but may conceal signal peaks. Channel 2 requires the DIG option.
	OFF	Selects sample decimation for sample rates lower than the maximal available sampling rate.
	ON	Selects sample averaging for sample rates lower than the maximal available sampling rate.
Rate	27.5 kHz to 28.1 MHz	Streaming rate of the scope channels. The streaming rate can be adjusted independent from the scope sampling rate. The maximum rate depends on the interface used for transfer. Note: scope streaming requires the DIG option.
Enable	ON / OFF	Enable scope streaming for the specified channel. This allows for continuous recording of scope data on the plotter and streaming to disk. Note: scope streaming requires the DIG option.

Control/ Tool	Option/ Range	Description
X Axis		Select the x-axis for xy-plot display mode.
	Time/Freq	The xy-plot mode is off. The x-axis is either time or frequency.
	Channel 1	The xy-plot is enabled with the first channel used for the x-axis.
	Channel 2	The xy-plot is enabled with the second channel used for the x-axis.

Table 5.19: Scope tab: History sub-tab

Control/ Tool	Option/ Range	Description
History	History	Each entry in the list corresponds to a single trace in the history. The number of traces displayed in the plot is limited to 20. Use the toggle buttons to hide or show individual traces. Use the color picker to change the color of a trace in the plot. Double click on a list entry to edit its name.
Length	integer value	Maximum number of records in the history. The number of entries displayed in the list is limited to the 100 most recent ones.
Clear All	Clear All	Remove all records from the history list.
Clear	Clear	Remove selected records from the history list.
Load file		Load data from a file into the history. Loading does not change the data type and range displayed in the plot, this has to be adapted manually if data is not shown.
Name		Enter a name which is used as a folder name to save the history into. An additional three digit counter is added to the folder name to identify consecutive saves into the same folder name.
Auto Save		Activate autosaving. When activated, any measurements already in the history are saved. Each subsequent measurement is then also saved. The autosave directory is identified by the text "autosave" in the name, e.g. "sweep_autosave_001". If autosave is active during continuous running of the module, each successive measurement is saved to the same directory. For single shot operation, a new directory is created containing all measurements in the history. Depending on the file format, the measurements are either appended to the same file, or saved in individual files. For HDF5 and ZView formats, measurements are appended to the same file. For MATLAB and SXM formats, each measurement is saved in a separate file.
File Format		Select the file format in which to save the data.
Save	Save	Save the traces in the history to a file accessible in the File Manager tab. The file contains the signals in the Vertical Axis Groups of the Control sub-tab. The data that is saved depends on the selection from the pull-down list. Save All: All traces are saved. Save Sel: The selected traces are saved.

For the Math sub-tab please see the table "Plot math description" in the section called "Cursors and Math".

5.8. Auxiliary Tab

The Auxiliary tab provides access to the settings of the Auxiliary Inputs and Auxiliary Outputs; it is available on all UHFQA instruments.

5.8.1. Features

- Monitor signal levels of auxiliary input connectors
- Monitor signal levels of auxiliary output connectors
- Auxiliary output signal sources: AWG and manual setting
- Define Offsets and Scaling for auxiliary output values
- Control auxiliary output range limitations

5.8.2. Description

The Auxiliary tab serves mainly to monitor and control the auxiliary inputs and outputs. Whenever the tab is closed or an additional one of the same type is needed, clicking the following icon will open a new instance of the tab.

Table 5.20: App icon and short description

Control/ Tool	Option/ Range	Description
Aux		Controls all settings regarding the auxiliary inputs and auxiliary outputs.

The Auxiliary tab (see LabOne UI: Auxiliary tab) is divided into three sections. The Aux Input section gives two graphical and two numerical monitors for the signal amplitude applied to the auxiliary inputs on the back panel. In the middle of the tab the Aux Output section allows to associate any of the measured signals to one of the 4 auxiliary outputs on the instrument front panel. With the action buttons next to the Preoffset and Offset values the effective voltage on the auxiliary outputs can be automatically set to zero. The analog output voltages can be limited to a certain range in order to avoid damaging the parts connected to the outputs.

Note

Please note the change of units of the scaling factor depending on what measurement signal is chosen.

Two Aux Output Levels on the right provides 4 graphical and 4 numerical indicators to monitor the voltages currently set on the auxiliary outputs.

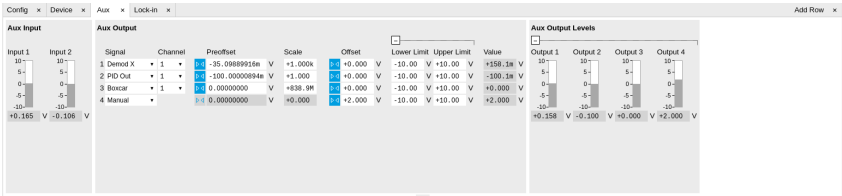


Figure 5.13: LabOne UI: Auxiliary tab

5.8.3. Functional Elements

Table 5.21: Auxiliary tab

Control/ Tool	Option/ Range	Description
Auxiliary Input Voltage	-10 V to 10 V	Voltage measured at the Auxiliary Input.
Signal		Select the signal source to be represented on the Auxiliary Output.
	X	Select the demodulator X component for auxiliary output.
	Y	Select the demodulator Y component for auxiliary output.
	R	Select the demodulator magnitude component for auxiliary output.
	Θ	Select the demodulator phase component for auxiliary output.
	PID Out	Select one of the PID controllers output. UHF-PID option needs to be installed.
	PID Shift	Select one of the PID controllers' shift signal. UHF-PID option needs to be installed.
	PID Error	Select one of the PID controllers' error signal. UHF-PID option needs to be installed.

Control/ Tool	Option/ Range	Description
	Boxcar	Select one of the two Boxcar units for auxiliary output. UHF-BOX option needs to be installed.
	AU Cartesian	Select one of the two Arithmetic Cartesian units for auxiliary output.
	AU Polar	Select one of the two Arithmetic Polar units for auxiliary output.
	AWG	Select one of the AWG Outputs for auxiliary output when running the AWG in four-channel mode. UHF-AWG option needs to be installed.
	CNT Out	Select one of the Pulse Counter signals for auxiliary output. UHF-CNT option needs to be installed.
	Manual	Manually define an auxiliary output voltage using the offset field.
Channel	index	Select the channel according to the selected signal source.
Preoffset	numerical value in signal units	Add a pre-offset to the signal before scaling is applied. Auxiliary Output Value = (Signal+Preoffset)*Scale + Offset
Auto-zero		Automatically adjusts the Pre-offset to set the Auxiliary Output Value to zero.
Scale	numerical value	Multiplication factor to scale the signal. Auxiliary Output Value = (Signal+Preoffset)*Scale + Offset
Auto-zero		Automatically adjusts the Offset to set the Auxiliary Output Value to zero.
Offset	numerical value in Volts	Add the specified offset voltage to the signal after scaling. Auxiliary Output Value = (Signal+Preoffset)*Scale + Offset
Lower Limit	-10 V to 10 V	Lower limit for the signal at the Auxiliary Output. A smaller value will be clipped.
Upper Limit	-10 V to 10 V	Upper limit for the signal at the Auxiliary Output. A larger value will be clipped.
Value	-10 V to 10 V	Voltage present on the Auxiliary Output port. The value is obtained by clamping the calculated signal (Signal+Preoffset)*Scale + Offset based on Lower and Upper Limits.

5.9. Inputs/Outputs Tab

The In / Out tab provides access to the settings of the Instrument's main Signal Inputs and Signal Outputs. It is available on all UHFQA instruments.

5.9.1. Features

- Signal input configuration
- Signal output configuration

5.9.2. Description

The In / Out tab gives access to the physical configuration of the signal inputs and outputs of the instrument, as well as to the oscillator settings.

Table 5.22: App icon and short description

Control/ Tool	Option/ Range	Description
In/Out		Gives access to all controls relevant for the Signal Inputs and Signal Outputs of each channel.

The In / Out tab contains one section for the signal inputs and one for the signal outputs. All of the corresponding connectors are placed on the instrument front panel. The

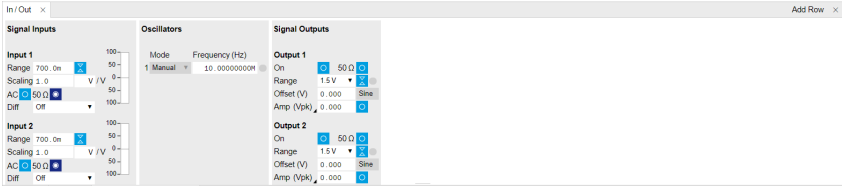


Figure 5.14: LabOne UI: Inputs/Outputs tab

5.10. DIO Tab

The DIO tab provides access to the settings and controls of the digital I/O as well as the Trigger channels and is available on all UHFQA instruments.

5.10.1. Features

- Monitor and control of digital I/O connectors
- Control settings for external reference and triggering

5.10.2. Description

The DIO tab is the main panel to control the digital inputs and outputs as well as the trigger levels and external reference channels . Whenever the tab is closed or an additional one of the same type is needed, clicking the following icon will open a new instance of the tab.

Table 5.23: App icon and short description

Control/ Tool	Option/ Range	Description
DIO		Gives access to all controls relevant for the digital inputs and outputs including DIO, Trigger Inputs, Trigger Outputs, and Marker Outputs.

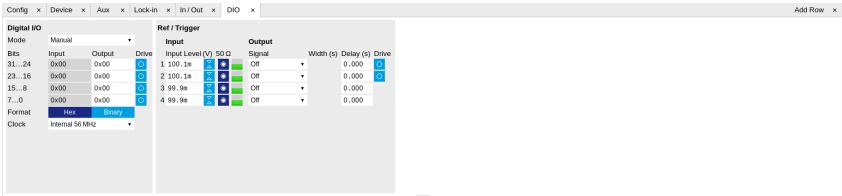


Figure 5.15: LabOne UI: DIO tab

The Digital I/O section provides numerical monitors to observe the states of the digital inputs and outputs. Moreover, with the values set in the Output column and the Drive button activated the states can also be actively set in different numerical formats.

The Ref/Trigger section shows the settings for the 6 reference and trigger inputs and outputs. The two BNC connectors on the front panel are numbered 1 and 2 and can act as inputs as well as outputs. The first two lines in this section are associated to these front panel connectors. On the back panel of the Instrument are 2 more trigger inputs (line 3 and 4, left columns) and 2 more trigger outputs (line 3 and 4, right columns). All four are SMA connectors.

Note

The Input Level determines the trigger threshold for trigger state discrimination. Also a 100 mV hysteresis is applied that cannot be adjusted such that a minimum amplitude of more than 100 mV is needed for the Trigger inputs to work reliably.

5.10.3. Functional Elements

Table 5.24: Digital input and output channels, reference and trigger

Control/ Tool	Option/Range	Description
DIO mode		Select DIO mode
	Manual	Enables manual control of the DIO output bits.
	AWG Sequencer	Enables setting of DIO output values by AWG sequencer commands.
	QA Results	Enables setting of DIO output values by QA results.
	QA Results QCCS	Enables setting of DIO output values by QA results compatible with the QCCS.
QA Result Overflow	grey/yellow/red	Red: present overflow condition on the DIO interface during readout. Yellow: indicates an overflow occurred in the past. An overflow can happen if readouts are triggered faster than the maximum possible data-rate of the DIO interface.
DIO bits	label	Partitioning of the 32 bits of the DIO into 4 buses of 8 bits each. Each bus can be used as an input or output.
DIO input	numeric value in either Hex or Binary format	Current digital values at the DIO input port.
DIO output	numeric value in either hexadecimal or binary format	Digital output values. Enable drive to apply the signals to the output.
DIO drive	ON / OFF	When on, the corresponding 8-bit bus is in output mode. When off, it is in input mode.
Format		Select DIO view format.
	Hexadecimal	DIO view format is hexadecimal.
	Binary	DIO view format is binary.
Clock		Select DIO internal or external clocking.
	Internal 56 MHz	The DIO is internally clocked with a frequency of 56.25 MHz.
	Clk Pin 68	The DIO is externally clocked with a clock signal connected to DIO Pin 68. Available frequency range 1 Hz to 56.25 MHz.
	Internal 50 MHz	The DIO is internally clocked with a frequency of 50 MHz.
Trigger level	-5 V to 5 V	Trigger voltage level at which the trigger input toggles between low and high. Use 50% amplitude for digital input and consider the trigger hysteresis.
Auto Threshold	Press once	Automatically adjust the trigger threshold. The level is adjusted to fall in the center of the applied transitions.
50 Ω	50 Ω/1 kΩ	Trigger input impedance: When on, the trigger input impedance is 50 Ω, when off 1 kΩ.
Trigger Input Low status		Indicates the current low level trigger state.
	Off	A low state is not being triggered.
	On	A low state is being triggered.
Trigger Input High status		Indicates the current high level trigger state.
	Off	A high state is not being triggered.
	On	A high state is being triggered.
Trigger output signal		Select the signal assigned to the trigger output.
	Off	The output trigger is disabled.
	Osc Phase Demod 4/8	Oscillator phase of demod 4 (trigger output channel 1) or demod 8 (trigger output channel 2).Trigger event is output for each zero crossing of the oscillator phase.

Control/ Tool	Option/Range	Description
	Scope Trigger	Trigger output is asserted when the scope trigger condition is satisfied.
	Scope /Trigger	Trigger output is deasserted when the scope trigger condition is satisfied.
	Scope Armed	Trigger output is asserted when the scope is waiting for the trigger condition to become satisfied.
	Scope /Armed	Trigger output is deasserted when the scope is waiting for the trigger condition to become satisfied.
	Scope Active	Trigger output is asserted when the scope has triggered and is recording data.
	Scope /Active	Trigger output is deasserted when the scope has triggered and is recording data.
	AWG Marker 1	Trigger output is assigned to one of the AWG Marker channels attached to AWG waveform data.
	AWG Marker 2	Trigger output is assigned to one of the AWG Marker channels attached to AWG waveform data.
	AWG Marker 3	Trigger output is assigned to one of the AWG Marker channels attached to AWG waveform data.
	AWG Marker 4	Trigger output is assigned to one of the AWG Marker channels attached to AWG waveform data.
	AWG Active	Trigger output is asserted when the AWG is enabled.
	AWG Waiting	Trigger output is asserted when the AWG is waiting for external triggers, for a clock timer, or for other events.
	AWG Fetching	Trigger output is asserted when the AWG is fetching data from the main waveform and instruction memory.
	AWG Playing	Trigger output is asserted when the AWG is playing waveforms.
	AWG Trigger 1	Trigger output is assigned to one of the AWG Trigger channels controlled by AWG sequencer commands.
	AWG Trigger 2	Trigger output is assigned to one of the AWG Trigger channels controlled by AWG sequencer commands.
	AWG Trigger 3	Trigger output is assigned to one of the AWG Trigger channels controlled by AWG sequencer commands.
	AWG Trigger 4	Trigger output is assigned to one of the AWG Trigger channels controlled by AWG sequencer commands.
	MDS Clock Out	Trigger output is driven by the multi-device synchronisation clock.
	MDS Sync Out	Trigger output is driven by the multi-device synchronisation signal.
Delay (s)		This delay adds an offset that acts only on the trigger/marker output. The total delay to the trigger/marker output is the sum of this value and the value of the output delay node.
Width	0 s to 0.149 s	Defines the minimal pulse width for the case of Scope and AWG Trigger/Active events written to the trigger outputs of the device.
Delay	0 ns to 2.4 ns	Controls the delay of the Ref / Trigger output. The resolution is 78 ps.
Trigger drive	ON / OFF	When on, the bidirectional trigger on the front panel is in output mode. When off, the trigger is in input mode.

5.11. Config Tab

The Config tab provides access to all major LabOne settings and is available on all UHFQA instruments.

5.11.1. Features

- define instrument connection parameters
- browser session control
- define UI appearance (grids, theme, etc.)
- store and load instrument settings and UI settings
- configure data recording

5.11.2. Description

The Config tab serves as a control panel for all general LabOne settings and is opened by default on start-up. Whenever the tab is closed or an additional one of the same type is needed, clicking the following icon will open a new instance of the tab.

Table 5.25: App icon and short description

Control/Tool	Option/Range	Description
Config		Provides access to software configuration.

The Config tab (see LabOne UI: Config tab) is divided into four sections to control connections, sessions, settings, user interface appearance and data recording.

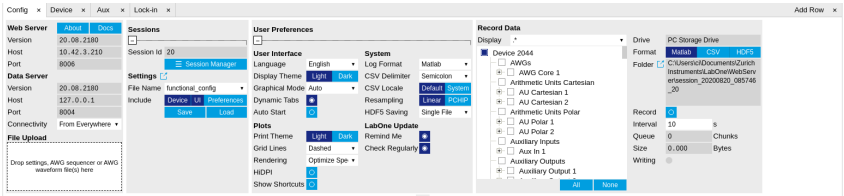


Figure 5.16: LabOne UI: Config tab

The **Connection** section provides information about connection and server versions. Access from remote locations can be restricted with the connectivity setting.

The **Session** section provides the session number which is also displayed in the status bar. Clicking on Session Dialog opens the session dialog window (same as start up screen) that allows one to load different settings files as well as to connect to other instruments.

The **Settings** section allows one to load and save instrument and UI settings. The saved settings are later available in the session dialog.

The **User Preferences** section contains the settings that are continuously stored and automatically reloaded the next time an UHFQA instrument is used from the same computer account.

For low ambient light conditions the use of the dark display theme is recommended (see Figure 5.17).

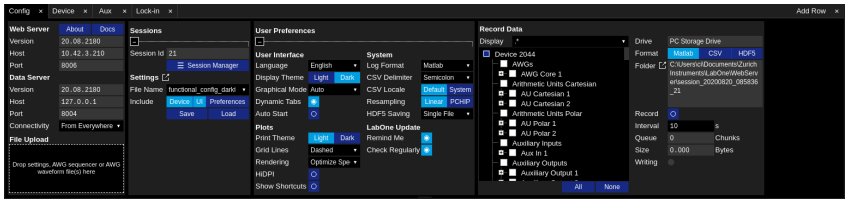


Figure 5.17: LabOne UI: Config tab - dark theme

5.11.3. Functional Elements

Table 5.26: Config tab

Control/ Tool	Option/Range	Description
About	About	Get information about LabOne software.
Web Server Version and Revision	string	Web Server version and revision number
Host	default is localhost: 127.0.0.1	IP-Address of the LabOne Web Server
Port	4 digit integer	LabOne Web Server TCP/IP port
Data Server Version and Revision	string	Data Server version and revision number
Host	default is localhost: 127.0.0.1	IP-Address of the LabOne Data Server
Port	default is 8004	TCP/IP port used to connect to the LabOne Data Server.
Connect/ Disconnect		Connect/disconnect the LabOne Data Server of the currently selected device. If a LabOne Data Server is connected only devices that are visible to that specific server are shown in the device list.
Status	grey/green	Indicates whether the LabOne User Interface is connected to the selected LabOne data server. Grey: no connection. Green: connected. Red: error while connecting.
Connectivity	From Everywhere	Forbid/Allow to connect to this Data Server from other computers.
	Localhost Only	
File Upload	drop area	Drag and drop files in this box to upload files. Clicking on the box opens a file dialog for file upload. Supported files: Settings (*.xml).
Session Id	integer number	Session identifier. A session is a connection between a client and LabOne Data Server.
Session Manager	Session Manager	Open the session manager dialog. This allows for device or session change. The current session can be continued by pressing cancel.
File Name	selection of available file names	Save/load the device and user interface settings to/from the selected file on the internal flash drive. The setting files can be downloaded/uploaded using the Files tab.
Include		Enable Save/Load of particular settings.
	No Include Settings	Please enable settings type to be included during Save/ Load.
	Include Device	Enable Save/Load of Device settings.
	Include UI	Enable Save/Load of User Interface settings.
	Include UI and Device	Enable Save/Load of User Interface and Device settings.
	Include Preferences	Enable loading of User Preferences from settings file.
	Include UI, Device and Preferences	Enable Save/Load of User Interface, Device and User Preferences.
Save	Save	Save the user interface and device setting to a file.
Load	Load	Load the user interface and device setting from a file.
Language		Choose the language for the tooltips.
Display Theme	Dark	Choose theme of the user interface.
	Light	
Plot Print Theme	Dark	Choose theme for printing SVG plots.
	Light	

Control/ Tool	Option/Range	Description
Plot Grid	None	Select active grid setting for all SVG plots.
	Dashed	
	Solid	
Plot Rendering		Select rendering hint about what tradeoffs to make as the browser renders SVG plots. The setting has impact on rendering speed and plot display for both displayed and saved plots.
	Auto	Indicates that the browser shall make appropriate tradeoffs to balance speed, crisp edges and geometric precision, but with geometric precision given more importance than speed and crisp edges.
	Optimize Speed	The browser shall emphasize rendering speed over geometric precision and crisp edges. This option will sometimes cause the browser to turn off shape anti-aliasing.
	Crisp Edges	Indicates that the browser shall attempt to emphasize the contrast between clean edges of artwork over rendering speed and geometric precision. To achieve crisp edges, the user agent might turn off anti-aliasing for all lines and curves or possibly just for straight lines which are close to vertical or horizontal.
	Geometric Precision	Indicates that the browser shall emphasize geometric precision over speed and crisp edges.
Resampling Method		Select the resampling interpolation method. Resampling corrects for sample misalignment in subsequent scope shots. This is important when using reduced sample rates with a time resolution below that of the trigger.
	Linear	Linear interpolation
	PCHIP	Piecewise Cubic Hermite Interpolating Polynomial
Show Shortcuts	ON / OFF	Displays a list of keyboard and mouse wheel shortcuts for manipulating plots.
Dynamic Tabs	ON / OFF	If enabled, sections inside the application tabs are collapsed automatically depending on the window width.
Graphical Mode	Collapsed	Select the display mode for the graphical elements. Auto format will select the format which fits best the current window width.
	Auto	
	Expanded	
Log Format	.NET	Choose the command log format. See status bar and [User] \Documents\Zurich Instruments\LabOne\WebServer\Log
	MATLAB	
	Python	
CSV Delimiter	Tab	Select which delimiter to insert for CSV files.
	Comma	
	Semicolon	
CSV Locale	System locale. Use the symbols set in the language and region settings of the computer	Select the locale used for defining the decimal point and digit grouping symbols in numeric values in CSV files. The default locale uses dot for the decimal point and no digit grouping, e.g. 1005.07. The system locale uses the symbols set in the language and region settings of the computer.
	Default locale. Dot for the decimal point and no digit grouping, e.g. 1005.07	
HDF5 Saving	Multiple files. Each measurement goes in a separate file	For HDF5 file format only: Select whether each measurement should be stored in a separate file, or whether all measurements should be saved in a single file.

Control/ Tool	Option/Range	Description
	Single file. All measurements go in one file	
Auto Start	ON / OFF	Skip session manager dialog at start-up if selected device is available. In case of an error or disconnected device the session manager will be reactivated.
Update Reminder	ON / OFF	Display a reminder on start-up if the LabOne software wasn't updated in 180 days.
Update Check	ON / OFF	Periodically check for new LabOne software over the internet.
Drive		Select the drive for data saving.
	PC Storage Drive	Storage of the PC on which the LabOne Web Server is running.
Format	HDF5	File format of recorded and saved data.
	MATLAB	
	CSV	
Open Folder		Open recorded data in the system File Explorer.
Folder	path indicating file location	Folder containing the recorded data.
Save Interval	Time in seconds	Time between saves to disk. A shorter interval means less system memory consumption, but for certain file formats (e.g. MATLAB) many small files on disk. A longer interval means more system memory consumption, but for certain file formats (e.g. MATLAB) fewer, larger files on disk.
Queue	integer number	Number of data chunks not yet written to disk.
Size	integer number	Accumulated size of saved data in the current session.
Record	ON / OFF	Start and stop saving data to disk as defined in the selection filter. Length of the files is determined by the Window Length setting in the Plotter tab.
Writing	grey/green	Indicates whether data is currently written to disk.
Display	filter or regular expression	Display specific tree branches using one of the preset view filters or a custom regular expression.
Tree	ON / OFF	Click on a tree node to activate it.
All		Select all tree elements.
None		Deselect all tree elements.

For more information on the tree functionality in the Record Data section, please see Tree Selector.

5.12. Device Tab

The Device tab is the main settings tab for the connected instrument and is available on all UHFQA instruments.

5.12.1. Features

- Option and upgrade management
- External clock referencing (10 MHz)
- Auto calibration settings
- Instrument connectivity parameters
- Device monitor

5.12.2. Description

The **Device tab** serves mainly as a control panel for all settings specific to the instrument that is controlled by LabOne in this particular session. Whenever the tab is closed or an additional one of the same type is needed, clicking the following icon will open a new instance of the tab.

Table 5.27: App icon and short description

Control/Tool	Option/Range	Description
Device		Provides instrument specific settings.

The Device tab (see LabOne UI: Device tab) is divided into five sections: general instrument information, configuration, communication parameters, device presets, and a device monitor.

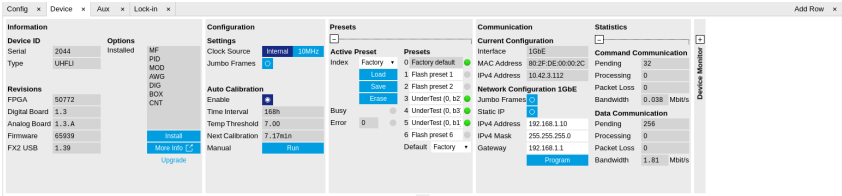


Figure 5.18: LabOne UI: Device tab

The **Information** section provides details about the instrument hardware and indicates the installed upgrade options. This is also the place where new options can be added by entering the provided option key.

The **Configuration** section allows one to change the reference from the internal clock to an external 10 MHz reference. The reference is to be connected to the Clock Input on the instrument back panel.

The **Presets** section allows you to define a custom instrument start-up configuration different from the factory default. This configuration is stored in the instrument itself and are applied independently of the control PC. This saves time in cases where the control PC is not routinely needed, for instance when using only analog interfaces the instrument configuration is fixed.

The **Communication** section offers access to the instruments TCP/IP settings.

Note

Activating Jumbo Frames is essential to achieve maximum data rates and also reduces load on the host PC.

The **Statistics** section gives an overview on communication statistics. In particular the current data rate (Bandwidth) that is consumed.

Note

Packet loss on data streaming over UDP or USB: data packets may be lost if total bandwidth exceeds the available physical interface bandwidth. Data may also be lost if the host computer is not able to handle high-bandwidth data. Network card setting optimization and Jumbo frame enabling may increase the maximal effective bandwidth.

Note

Packet loss on command streaming over TCP or USB: command packets should never be lost as it creates an invalid state.

The **Device Monitor** section is collapsed by default and generally only needed for servicing. It displays vitality signals of some of the instrument's hardware components.

Self Calibration

Note

The calibration routine takes about 200 ms for that time the transfer of measurement data is stopped. That will lead to the following visible effects on the UI: - missing data on the plotter - the UI will shortly freeze - the data loss flag will not report data loss (as the server intentionally trashed data) - Sweeper, Data Acquisition tool and Scope will behave as usual and wait until they get data again - The Spectrum tool will restart as it can only analyze continuously sampled data

Please see also additional remarks regarding calibration in specifications.

5.12.3. Functional Elements

Table 5.28: Device tab

Control/Tool	Option/Range	Description
Serial	1-4 digit number	Device serial number
Type	string	Device type
FPGA	integer number	HDL firmware revision.
Digital Board	version number	Hardware revision of the FPGA base board.
Analog Board	version indicator	Hardware revision of the analog board.
Firmware	integer number	Revision of the device internal controller software.
FX2 USB	version number	USB firmware revision.
Installed Options	short names for each option	Options that are installed on this device.
Install	Install	Click to install options on this device. Requires a unique feature code and a power cycle after entry.
More Information		Display additional device information in a separate browser tab.
Upgrade Device Options		Display available upgrade options.
Clock Source		10MHz reference clock source.
	Internal	The internal 10MHz clock is used as the frequency and time base reference.
	Clk 10MHz	An external 10MHz clock is intended to be used as the frequency and time base reference. Provide a clean and stable 10MHz reference to the appropriate back panel connector.
Jumbo Frames	ON / OFF	Enables jumbo frames (4k) on the TCP/IP interface. This will reduce the load on the PC and is required to achieve maximal throughput. Make sure that jumbo frames (4k) are enabled on the network card as well. If one of the devices on the network is not able to work with jumbo frames, the connection will fail.
Enabled	ON / OFF	Enables an automatic instrument self calibration about 16 min after start up. In order to guarantee the full specification, it is recommended to perform a self calibration after warm-up of the device.
Time interval	time	Time interval for which the self calibration is valid. After this time it is recommended to rerun the auto calibration. A LED indicator in the status bar indicates when another self calibration is recommended.
Calibration Temperature Threshold	temperature in °C	When the temperature changes by the specified amount, it is recommended to rerun the self calibration. A LED indicator in the status bar indicates when another self calibration is recommended.

Control/Tool	Option/Range	Description
Next calibration	time	Remaining time until the first calibration is executed or a recalibration is requested.
Manual self calibration	Run	Initiate self calibration to improve input digitizer linearity.
Index		Select between factory preset or presets stored in internal flash memory.
	Factory	Select factory preset.
	Flash 1-6	Select one of the presets stored in internal flash memory 1-6.
Load	Load	Load the selected preset.
Save	Save	Save the actual setting as preset.
Erase	Erase	Erase the selected preset.
Busy	grey/green	Indicates that the device is busy with either loading, saving or erasing a preset.
Error		Returns a 0 if the last preset operation was successfully completed or 1 if the last preset operation was illegal.
	0	Last preset operation was successfully completed.
	1	Last preset operation was illegal.
Error LED	grey/red	Turns red if the last operation was illegal.
Valid LED	grey/green	Turns green if a valid preset is stored at the respective location.
Presets		Shows a list of available presets including factory preset.
	0	Factory default preset. The name of the factory default preset is given and can not be edited.
	1	Flash preset 1. The name of this preset can be edited.
	2	Flash preset 2. The name of this preset can be edited.
	3	Flash preset 3. The name of this preset can be edited.
	4	Flash preset 4. The name of this preset can be edited.
	5	Flash preset 5. The name of this preset can be edited.
	6	Flash preset 6. The name of this preset can be edited.
Default		Indicates the preset which is used as default preset at start-up of the device.
	Factory	Select factory preset as default preset.
	Flash 1-6	Select one of the presets stored in internal flash memory 1-6 as default preset.
Interface	1. USB, 2. 1GbE	Active interface between device and data server. In case multiple options are available, the priority as indicated on the left applies.
MAC Address	80:2F:DE:xx:xx:xx	MAC address of the device. The MAC address is defined statically, cannot be changed and is unique for each device.
IPv4 Address	default 192.168.1.10	Current IP address of the device. This IP address is assigned dynamically by a DHCP server, defined statically, or is a fall-back IP address if the DHCP server could not be found (for point to point connections).
Jumbo Frames	ON / OFF	Enable jumbo frames for this device and interface as default.
Static IP	ON / OFF	Enable this flag if the device is used in a network with fixed IP assignment without a DHCP server.
IPv4 Address	default 192.168.1.10	Static IP address to be written to the device.

Control/Tool	Option/Range	Description
IPv4 Mask	default 255.255.255.0	Static IP mask to be written to the device.
Gateway	default 192.168.1.1	Static IP gateway
Program	Program	Click to program the specified IPv4 address, IPv4 Mask and Gateway to the device.
Pending	integer value	Number of buffers ready for receiving command packets from the device.
Processing	integer value	Number of buffers being processed for command packets. Small values indicate proper performance. For a TCP/IP interface, command packets are sent using the TCP protocol.
Packet Loss	integer value	Number of command packets lost since device start. Command packets contain device settings that are sent to and received from the device.
Bandwidth	numeric value	Command streaming bandwidth usage on the physical network connection between device and data server.
Pending	integer value	Number of buffers ready for receiving data packets from the device.
Processing	integer value	Number of buffers being processed for data packets. Small values indicate proper performance. For a TCP/IP interface, data packets are sent using the UDP protocol.
Packet Loss	integer value	Number of data packets lost since device start. Data packets contain measurement data.
Bandwidth	numeric value	Data streaming bandwidth usage on the physical network connection between device and data server.
FW Load	numeric value	Indicates the CPU load on the processor where the firmware is running.

5.13. File Manager Tab

The File Manager tab provides a quick access to measurement files, log files and setting files in the local file system.

5.13.1. Features

- Quick access to measurement files, log files and settings files
- File preview for settings files and log files

5.13.2. Description

The File Manager tab provides standard tools to see and organize the files relevant for the use of the instrument. Files can be conveniently copied, renamed and deleted. Whenever the tab is closed or an additional one of the same type is needed, clicking the following icon will open a new instance of the tab.

Table 5.29: App icon and short description

Control/ Tool	Option/ Range	Description
Files	🗀	Access settings and measurement data files on the host computer.

The Files tab (see LabOne UI: File Manager tab) provides three windows for exploring. The left window allows one to browse through the directory structure, the center window shows the files of the folder selected in the left window, and the right window displays the content of the file selected in the center window, e.g. a settings file or log file.

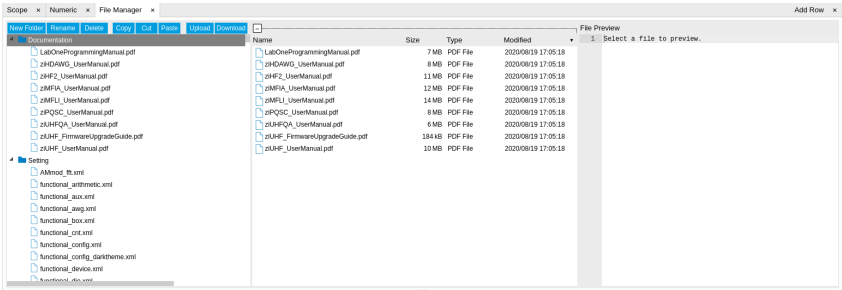


Figure 5.19: LabOne UI: File Manager tab

5.13.3. Functional Elements

Table 5.30: File tab

Control/ Tool	Option/ Range	Description
New Folder	New Folder	Create new folder at current location.
Rename	Rename	Rename selected file or folder.
Delete	Delete	Delete selected file(s) and/or folder(s).
Copy	Copy	Copy selected file(s) and/or folder(s) to Clipboard.
Cut	Cut	Cut selected file(s) and/or folder(s) to Clipboard.
Paste	Paste	Paste file(s) and/or folder(s) from Clipboard to the selected directory.
Upload	Upload	Upload file(s) and/or folder(s) to the selected directory.
Download	Download	Download selected file(s) and/or folder(s).

5.14. AWG Tab

The AWG tab is available on UHFAWG Arbitrary Waveform Generator instruments and on UHFLI Lock-in Amplifier instruments with installed UHF-AWG Arbitrary Waveform Generator option (see Information section in the Device tab).

5.14.1. Features

- Dual-channel arbitrary waveform generator
- 128 MSa waveform memory per channel
- Sequence branching
- Digital modulation
- Multi-instrument synchronization
- Sequence program distribution over multiple instruments
- Cross-domain trigger engine
- Sequence Editor with code highlighting and auto completion
- High-level programming language with waveform generation and editing toolset
- Waveform viewer

5.14.2. Description

The AWG tab gives access to the arbitrary waveform generator functionality. Whenever the tab is closed or an additional one of the same type is needed, clicking the following icon will open a new instance of the tab.

Table 5.31: App icon and short description

Control/ Tool	Option/ Range	Description
AWG		Generate arbitrary signals using sequencing and sample-by-sample definition of waveforms.

The AWG tab (see Figure 5.20) consists of a settings section on the right side and the Sequence and Waveform Viewer sub-tabs on the left side. The settings section is further divided into Control, Waveform, Trigger, and Advanced sub-tabs. The **Sequence** sub-tab is used for displaying, editing and compiling a LabOne sequence program. The sequence program defines which waveforms are played and in which order. The Sequence Editor is the main tool for operating the AWG.

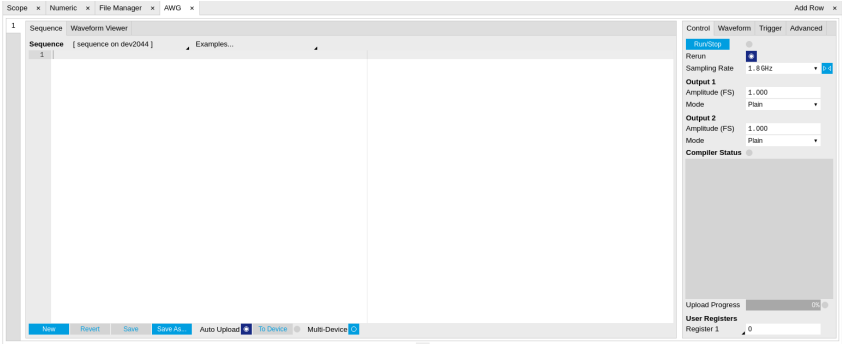


Figure 5.20: LabOne UI: AWG tab

A number of sequence programming examples are available through a drop-down menu at the top of the Sequence Editor, and additional ones can be found in Arbitrary Waveform Generator . The LabOne sequence programming language is specified in detail in LabOne Sequence Programming. The language comes with a number of predefined waveforms, such as Gaussian, Blackman, sine, or square functions. By combining those predefined waveforms using the waveform editing tools (add, multiply, cut, concatenate, etc), signals with a high level of complexity can be generated directly from the Sequence Editor window. Sample-by-sample definition of the output signal is possible by using comma-separated value (CSV) files specified by the user , see Waveform Generation and Playbackfor an example.

The AWG features a compiler which translates the high-level sequence program into machine instructions and waveform data to be stored in the instrument memory as shown in Figure 5.21. The sequence program is written using high-level control structures and syntax that are inspired by human language, whereas machine instructions reflect exactly what happens on the hardware level. Writing the sequence program using a high-level language represents a more natural and efficient way of working in comparison to writing lists of machine instructions, which is the traditional way of programming AWGs. Concretely, the improvements rely on features such as:

- combination of waveform generation, editing, and playback sequence in a single script
- easily readable syntax and naming for run-time variables and constants
- optimized waveform memory management, reduced transfers upon waveform changes
- definition of user functions and procedures for advanced structuring
- syntax validation

By design, there is no one-to-one link between the list of statements in the high-level language and the list of instructions executed by the Sequencer. There are cases in which a more detailed understanding of the Sequencer instruction list, and in particular its execution timing, is needed. Typically this is the case when observing delays or other signal timing properties that are unexpected from looking at the high-level script. Often such problems can be solved with a few adjustments to the program. Please see Debugging Sequencer Programs for practical advice.

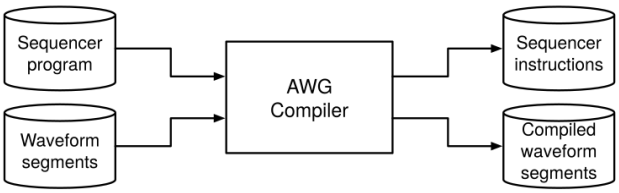


Figure 5.21: AWG sequence program compilation process

The **Sequence Editor** provides the editing, compilation, and transfer functionality for sequence programs. A program typed into the Editor is compiled upon clicking Save . If the compilation is successful and Automatic Upload is enabled, the program including all necessary waveform data is transferred to the device. If the compilation fails, the Status field will display debug messages. Clicking on Save as... allows you to choose a new name for the program. The name of the program that is currently edited is displayed at the top of the editor. External program files as well as waveform data files can be transferred to the right location easily using the file drag-and-drop zone in the Config Tab so they become accessible from the user interface. The files can be managed in the File Manager Tab and their location in the directory structure is shown in Table 5.32. The program name is displayed in a drop-down box. The box allows quick access to all programs in the

standard sequence program location. It is possible to quickly switch between programs using the box. Changes made in one program will be preserved when switching to a different program. The file name of a program will be postfixed by an asterisk in case there are unsaved changes in the source file. Note that switching programs in the editor is not sufficient to also update the program in the instrument. In order to send a newly selected program to the instrument, the To Device button must be clicked.

Table 5.32: Sequence program and waveform file location

File type	Location
Waveform files (Windows)	`C:\Users\<user name>\Documents\Zurich Instruments\LabOne\WebServer\awg\waves`
Sequence programs (Windows)	`C:\Users\<user name>\Documents\Zurich Instruments\LabOne\WebServer\awg\src`
Waveform files (Linux)	`~/Zurich Instruments/LabOne/WebServer/awg/waves`
Sequence programs (Linux)	`~/Zurich Instruments/LabOne/WebServer/awg/src`

In the **Control sub-tab** the user configures signal parameters and controls the execution of the AWG. The AWG can be started in by clicking on Start. When enabling the Rerun button, the Sequencer will be restarted automatically when its program completes. The continuous mode is a simple way to create an infinite loop, but it results in a considerable timing jitter. To avoid this jitter, it is recommended to specify infinite loops directly in the sequence program.

The Sampling Rate field is used to control the default playback sampling rate of the AWG. The sampling rate is dynamic, i.e., can be specified for each waveform by using an optional argument in the waveform playback instructions in the sequence program. This allows for considerably reducing waveform upload time for signals that contain both fast and slow components. The two Output sections are used to configure the AWG output mode and signal amplitude. The AWG output channels are not the same as the physical Signal Outputs of the instrument. The AWG output channels are routed to the Signal Outputs of the device. The Amplitude value is a gain parameter, 1.0 by default, that is applied to waveforms on the way from the AWG output channel to the Signal Output. The Amplitude value gives a means to rescale the signal independently of the programmed waveforms. The Mode control is used to enable or disable the modulation mode , or to enable advanced modulation mode . With enabled modulation, the signal of an AWG Output is multiplied with an oscillator signal prior to being sent to the Signal Output. This is useful for the case where the desired signal can be described as a sinusoidal carrier with a shaped envelope. The advanced modulation mode allows you to modulate multiple carriers (up to 4) with individual envelope waveforms. Please read more about use cases, advantages, and practical examples in Modulation Mode.

The **Waveform sub-tab** displays information about the waveforms that are used by the current sequence program, such as their length and channel number. Together with the **Waveform viewer sub-tab**, it is a useful tool to visualize the waveforms used in the sequence program.

On the **Trigger sub-tab** you can configure the trigger inputs of the AWG and control the Cross-Domain Trigger functionality of the instrument. The AWG has four trigger input channels which can be configured to probe a variety of signals coming both from internal (e.g. demodulator output data) or external (e.g. Ref/Trigger input) sources . This means that the AWG trigger input channels are not the same as physical device inputs. Two of the trigger input channels are called analog (meaning they can accept signals of continuous, analog-like character), and two are called digital (meaning they can accept binary signals). Trigger Level and Hysteresis may be configured for the Analog Triggers, and the user can select between rising and falling edge trigger functionality. The primary use of the triggers is to control the timing of the AWG signal relative to an external device. Another use of triggers is to implement sequence branching. See Triggering and Synchronization and Branching and Feed-Forward for practical examples on how to use the AWG trigger in- and outputs.

The **Advanced sub-tab** displays the compiled list of sequencer instructions and the current state of the sequencer on the instrument. This can help an advanced user in debugging a sequence program and understanding its execution.

Sequence Editor Keyboard Shortcuts

The tables below list a number of helpful keyboard shortcuts that are applicable in the LabOne Sequence Editor.

Table 5.33: Line Operations

Shortcut	Action
`Ctrl+D`	Remove line
`Alt+Shift+Down`	Copy lines down
`Alt+Shift+Up`	Copy lines up
`Alt+Down`	Move lines down
`Alt+Up`	Move lines up
`Alt+Del`	Remove to line end
`Alt+Backspace`	Remove to line start
`Ctrl+Backspace`	Remove word left
`Ctrl+Del`	Remove word right

Table 5.34: Selection

Shortcut	Action
`Ctrl+A`	Select all
`Shift+Left`	Select left
`Shift+Right`	Select right
`Ctrl+Shift+Left`	Select word left
`Ctrl+Shift+Right`	Select word right
`Shift+Home`	Select line start
`Shift+End`	Select line end
`Alt+Shift+Right`	Select to line end
`Alt+Shift+Left`	Select to line start
`Shift+Up`	Select up
`Shift+Down`	Select down
`Shift+Page Up`	Select page up
`Shift+Page Down`	Select page down
`Ctrl+Shift+Home`	Select to start
`Ctrl+Shift+End`	Select to end
`Ctrl+Shift+D`	Duplicate selection
`Ctrl+Shift+P`	Select to matching bracket

Table 5.35: Go to

Shortcut	Action
`Left`	Go to left
`Right`	Go to right
`Ctrl+Left`	Go to word left
`Ctrl+Right`	Go to word right
`Up`	Go line up
`Down`	Go line down
`Alt+Left`, `Home`	Go to line start
`Alt+Right`, `End`	Go to line end

Shortcut	Action
`Page Up`	Go to page up
`Page Down`	Go to page down
`Ctrl+Home`	Go to start
`Ctrl+End`	Go to end
`Ctrl+L`	Go to line
`Ctrl+Down`	Scroll line down
`Ctrl+Up`	Scroll line up
`Ctrl+P`	Go to matching bracket

Table 5.36: Find/Replace

Shortcut	Action
`Ctrl+F`	Find
`Ctrl+H`	Replace
`Ctrl+K`	Find next
`Ctrl+Shift+K`	Find previous

Table 5.37: Folding

Shortcut	Action
`Alt+L`	Fold selection
`Alt+Shift+L`	Unfold

Table 5.38: Other

Shortcut	Action
`Tab`	Indent
`Shift+Tab`	Outdent
`Ctrl+Z`	Undo
`Ctrl+Shift+Z`, `Ctrl+Y`	Redo
`Ctrl+/`	Toggle comment
`Ctrl+Shift+U`	Change to lower case
`Ctrl+U`	Change to upper case
`Ins`	Overwrite
`Ctrl+Shift+E`	Macros replay
`Ctrl+Alt+E`	Macros recording
`Del`	Delete

5.14.3. LabOne Sequence Programming

A Simple Example

The syntax of the LabOne AWG Sequencer programming language is based on C, but with a few simplifications. Each statement is concluded with a semicolon, several statements can be grouped with curly brackets, and comment lines are identified with a double slash. The following example shows some of the fundamental functionalities: waveform generation, repeated playback, triggering,

and single/dual-channel waveform playback. See Arbitrary Waveform Generator for a step-by-step introduction with more examples.

```
// Define an integer constant
const N = 4096;
// Create two Gaussian pulses with length N points,
// amplitude +1.0 (-1.0), center at N/2, and a width of N/8
wave gauss_pos = 1.0*gauss(N, N/2, N/8);
wave gauss_neg = -1.0*gauss(N, N/2, N/8);
// execute playback sequence 100 times
repeat (100) {
  // Play pulse on AWG channel 1
  playWave(gauss_pos);
  // Play pulses simultaneously on both AWG channels
  playWave(gauss_pos, gauss_neg);
  // Wait 8000 samples
  playZero(8000);
}
```

Multi-Instrument Support

The UHFLI supports multi-instrument operation by two important features

1. Automatic synchronization
2. Multi-instrument sequence program compilation

The first feature ensures that signals of multiple AWGs are precisely aligned in time and the user does not have to worry about cable delays, or about varying trigger delays after power cycles. The second feature greatly simplifies writing sequence program, as it allows to treat a setup with multiple AWGs conceptually like a single instrument.

Automatic synchronization can be set up using the Multi-Device Sync tab and is explained in detail in Multi Device Sync Tab. We assume that two UHFLI have been successfully synchronized according to the instructions in this section. Here we show an example of a sequence program to generate synchronized signals on two instruments

As part of the synchronization procedure using MDS, the LabOne Data Server running on the host PC is connected to both instruments. In the AWG tab, enable the Multi-Device button. The LabOne AWG Compiler is then able to distribute the high-level, multi-channel program the user enters in the AWG tab across all instruments. The Signal Output on which a given wave `w` is played is controlled by the integer argument `sig_out` in the instruction `playWave(sig_out, w)`. The numbering of the Signal Outputs is as follows:

Channel number in sequence program	Instrument number (according to order in MDS tab)	Signal Output number
1	Leader	1
2	Leader	2
3	Follower 1	1
4	Follower 1	2
5	Follower 2	1
...	...	...

The sequence program below contains three `playWave` instructions: the first instruction generates a pulse on instrument no. 1, the second one on instrument no. 2, and the third `playWave` instruction generates pulses simultaneously on both instruments.

```
wave w_gauss = gauss(8000, 4000, 1000);

while (true) {
  setTrigger(1);
  // Pulse on AWG 1 (Signal Output 1):
  playWave(1, w_gauss);
```

```
  // Pulse on AWG 2 (Signal Output 1):
  playWave(3, w_gauss);
  // Pulse on AWG 1 (Signal Outputs 1 & 2)
  //   and on AWG 2 (Signal Outputs 1 & 2):
  playWave(1, w_gauss, 2, w_gauss,
           3, w_gauss, 4, w_gauss);
  setTrigger(0);
  wait(100);
}
```

Keywords and Comments

The following table lists the keywords used in the LabOne AWG Sequencer language.

Table 5.39: Programming keywords

Keyword	Description
`const`	Constant declaration
`var`	Integer variable declaration
`cvar`	Compile-time variable declaration
`string`	Constant string declaration
`true`	Boolean true constant
`false`	Boolean false constant
`for`	For-loop declaration
`while`	While-loop declaration
`repeat`	Repeat-loop declaration
`if`	If-statement
`else`	Else-part of an if-statement
`switch`	Switch-statement
`case`	Case-statement within a switch
`default`	Default-statement within a switch
`return`	Return from function or procedure, optionally with a return value

The following code example shows how to use comments.

```
const a = 10; // This is a line comment. Everything between the double
              // slash and the end of the line will be ignored.

/* This is a block comment. Everything between the start-of-block-comment
and end-of-block-comment markers is ignored.

For example, the following statement will be ignored by the compiler.
const b = 100;
*/
```

Constants and Variables

Constants may be used to make the program more readable. They may be of integer or floating-point type. It must be possible for the compiler to compute the value of a constant at compile time, i.e., on the host computer. Constants are declared using the `const` keyword.

Compile-time variables may be used in computations and loop iterations during compile time, e.g. to create large numbers of waveforms in a loop. They may be of integer or floating-point type. They

are used in a similar way as constants, except that they can change their value during compile time operations. Compile-time variables are declared using the `cvar` keyword.

Variables may be used for making simple computations during run time, i.e., on the instrument. The Sequencer supports integer variables, addition, and subtraction. Not supported are floating-point variables, multiplication, and division. Typical uses of variables are to step waiting times , to output DIO values, or to tag digital measurement data with a numerical identifier. Variables are declared using the `var` keyword.

The following code example shows how to use variables.

```
var b = 100; // Create and initialize a variable

// Repeat the following block of statements 100 times
repeat (100) {
  b = b + 1; // Increment b
  wait(b);   // Wait 'b' cycles
}
```

The following table shows the predefined constants. These constants are intended to be used as arguments in certain run-time evaluated functions that encode device parameters with integer numbers. For example, the AWG Sampling Rate is specified as an integer exponent n in the expression (1.8 GSa/s)/2^n.

Table 5.40: Predefined Constants

Name	Value	Description
commandTableEntries	{4096}	
AWG_RATE_1800MHZ	0	Constant to set Sampling Rate to 1.8 GHz.
AWG_RATE_900MHZ	1	Constant to set Sampling Rate to 900 MHz.
AWG_RATE_450MHZ	2	Constant to set Sampling Rate to 450 MHz.
AWG_RATE_225MHZ	3	Constant to set Sampling Rate to 225 MHz.
AWG_RATE_112MHZ	4	Constant to set Sampling Rate to 112 MHz.
AWG_RATE_56MHZ	5	Constant to set Sampling Rate to 56 MHz.
AWG_RATE_28MHZ	6	Constant to set Sampling Rate to 28 MHz.
AWG_RATE_14MHZ	7	Constant to set Sampling Rate to 14 MHz.
AWG_RATE_7MHZ	8	Constant to set Sampling Rate to 7 MHz.
AWG_RATE_3P5MHZ	9	Constant to set Sampling Rate to 3.5 MHz.
AWG_RATE_1P8MHZ	10	Constant to set Sampling Rate to 1.8 MHz.
AWG_RATE_880KHZ	11	Constant to set Sampling Rate to 880 kHz.
AWG_RATE_440KHZ	12	Constant to set Sampling Rate to 440 kHz.
AWG_RATE_220KHZ	13	Constant to set Sampling Rate to 220 kHz.
AWG_MONITOR_TRIGGER	0x0000020	Constant to activate the trigger of the Monitor unit.
AWG_INTEGRATION_TRIGGER	0x0000010	Constant to activate the trigger of the Integration units.
AWG_INTEGRATION_ARM	0x3ff0000	Constant to arm the Integration units.
DEVICE_SAMPLE_RATE	<actual device sample rate>	
ZSYNC_DATA_RAW	0	Constant to use as argument to getZSyncData.
QA_INT_0	0b0000000001 << 16	Constant to enable Integration unit 0 in the Integration unit enable mask of the function startQA(). To construct more elaborate masks that enable multiple units, combine these predefined constants using the operator

Name	Value	Description
QA_INT_1	0b0000000010 << 16	Constant to enable Integration unit 1 in the Integration unit enable mask of the function startQA(). To construct more elaborate masks that enable multiple units, combine these predefined constants using the operator
QA_INT_2	0b0000000100 << 16	Constant to enable Integration unit 2 in the Integration unit enable mask of the function startQA(). To construct more elaborate masks that enable multiple units, combine these predefined constants using the operator
QA_INT_3	0b0000001000 << 16	Constant to enable Integration unit 3 in the Integration unit enable mask of the function startQA(). To construct more elaborate masks that enable multiple units, combine these predefined constants using the operator
QA_INT_4	0b0000010000 << 16	Constant to enable Integration unit 4 in the Integration unit enable mask of the function startQA(). To construct more elaborate masks that enable multiple units, combine these predefined constants using the operator
QA_INT_5	0b0000100000 << 16	Constant to enable Integration unit 5 in the Integration unit enable mask of the function startQA(). To construct more elaborate masks that enable multiple units, combine these predefined constants using the operator
QA_INT_6	0b0001000000 << 16	Constant to enable Integration unit 6 in the Integration unit enable mask of the function startQA(). To construct more elaborate masks that enable multiple units, combine these predefined constants using the operator
QA_INT_7	0b0010000000 << 16	Constant to enable Integration unit 7 in the Integration unit enable mask of the function startQA(). To construct more elaborate masks that enable multiple units, combine these predefined constants using the operator
QA_INT_8	0b0100000000 << 16	Constant to enable Integration unit 8 in the Integration unit enable mask of the function startQA(). To construct more elaborate masks that enable multiple units, combine these predefined constants using the operator
QA_INT_9	0b1000000000 << 16	Constant to enable Integration unit 9 in the Integration unit enable mask of the function startQA(). To construct more elaborate masks that enable multiple units, combine these predefined constants using the operator

Name	Value	Description
QA_INT_ALL	0b1111111111 << 16	Constant to enable all Integration units in the Integration unit enable mask of the function startQA().
AWG_CHAN1	1	Constant to select channel 1.
AWG_CHAN2	2	Constant to select channel 2.
AWG_MARKER1	1	Constant to select marker 1.
AWG_MARKER2	2	Constant to select marker 2.
AWG_OSC_PHASE_START	1	Constant to trigger the oscillator phase on the positive edge.
AWG_OSC_PHASE_MIDDLE	0	Constant to trigger the oscillator phase on the negative edge.
AWG_USERREG_SWEEP_COUNT0	35	Constant for the sweep count register 0.
AWG_USERREG_SWEEP_COUNT1	36	Constant for the sweep count register 1.

Numbers can be expressed using either of the following formatting.

```
const a = 10;            // Integer notation
const b = -10;           // Negative number
const h = 0xdeadbeef;    // Hexadecimal integer
const bin = 0b10101;     // Binary integer
const f = 0.1e-3;        // Floating point number.
const not_float = 10e3;  // Not a floating point number
```

Booleans are specified with the keywords `true` and `false`. Furthermore, all numbers that evaluate to a nonzero value are considered true. All numbers that evaluate to zero are considered false.

Strings are delimited using "" and are interpreted as constants. Strings may be concatenated using the + operator.

```
string AWG_PATH = "awgs/0/";
string AWG_GAIN_PATH = AWG_PATH + "gains/0";
```

Waveform Generation and Editing

The following table contains the definition of functions for waveform generation.

wave zeros(const samples)

Constant amplitude of 0 over the defined number of samples.

$$h(x) = 0$$

Args:

- `samples:` Number of samples in the waveform

Returns:

resulting waveform

wave ones(const samples)

Constant amplitude of 1 over the defined number of samples.

$$h(x) = 1$$

Args:

- `samples:` Number of samples in the waveform

Returns:

resulting waveform

wave sine(const samples, const amplitude=1.0, const phaseOffset, const nrOfPeriods)

Sine function with arbitrary amplitude (a), phase offset in radians (p), number of periods (f) and number of samples (N).

$$h(x) = a \cdot \sin(2\pi f \frac{x}{N} + p)$$

Args:

- `amplitude:` Amplitude of the signal (optional)
- `nrOfPeriods:` Number of Periods within the defined number of samples
- `phaseOffset:` Phase offset of the signal in radians
- `samples:` Number of samples in the waveform

Returns:

resulting waveform

wave cosine(const samples, const amplitude=1.0, const phaseOffset, const nrOfPeriods)

Cosine function with arbitrary amplitude (a), phase offset in radians (p), number of periods (f) and number of samples (N).

$$h(x) = a \cdot \cos(2\pi f \frac{x}{N} + p)$$

Args:

- `amplitude:` Amplitude of the signal (optional)
- `nrOfPeriods:` Number of Periods within the defined number of samples
- `phaseOffset:` Phase offset of the signal in radians
- `samples:` Number of samples in the waveform

Returns:

resulting waveform

wave sinc(const samples, const amplitude=1.0, const position, const beta)

Normalized sinc function with control of peak position (p), amplitude (a), width (\beta) and number of samples (N).

$$h(x) = \{a \quad \text{if } x = p \; a \cdot \frac{\sin(2\pi \cdot \text{beta} \cdot \frac{x-p}{N})}{2\pi \cdot \text{beta} \cdot \frac{x-p}{N}} \quad \text{else}$$

Args:

- `amplitude:` Amplitude of the signal (optional)
- `beta:` Width of the function
- `position:` Peak position of the function
- `samples:` Number of samples in the waveform

Returns:

resulting waveform

wave ramp(const samples, const startLevel, const endLevel)

Linear ramp from the start (s) to the end level (e) over the number of samples (N).

$$h(x) = s + \frac{x(e-s)}{N-1}$$

Args:

- `endLevel:` level at the last sample of the waveform
- `samples:` Number of samples in the waveform
- `startLevel:` level at the first sample of the waveform

Returns:

resulting waveform

wave sawtooth(const samples, const amplitude=1.0, const phaseOffset, const nrOfPeriods)

Sawtooth function with arbitrary amplitude, phase in radians and number of periods.

Args:

- `amplitude:` Amplitude of the signal
- `nrOfPeriods:` Number of Periods within the defined number of samples
- `phaseOffset:` Phase offset of the signal in radians
- `samples:` Number of samples in the waveform

Returns:

resulting waveform

wave triangle(const samples, const amplitude=1.0, const phaseOffset, const nrOfPeriods)

Triangle function with arbitrary amplitude, phase in radians and number of periods.

Args:

- `amplitude:` Amplitude of the signal
- `nrOfPeriods:` Number of Periods within the defined number of samples
- `phaseOffset:` Phase offset of the signal in radians
- `samples:` Number of samples in the waveform

Returns:

resulting waveform

wave gauss(const samples, const amplitude=1.0, const position, const width)

Gaussian pulse with arbitrary amplitude (a), position (p), width (w) and number of samples (N).

$$h(x) = a \cdot e^{-\frac{(x-p)^2}{2 \cdot w^2}}$$

Args:

- `amplitude:` Amplitude of the signal (optional)
- `position:` Peak position of the pulse
- `samples:` Number of samples in the waveform
- `width:` Width of the pulse

Returns:

resulting waveform

wave drag(const samples, const amplitude=1.0, const position, const width)

Derivative of Gaussian pulse with arbitrary amplitude (a), position (p), width (w) and number of samples (N) normalized to a maximum amplitude of 1.

$$h(x) = a \cdot \frac{\sqrt{e}(p - x)}{w} \cdot e^{-\frac{(x-p)^2}{2 \cdot w^2}}$$

Args:

- `amplitude:` Amplitude of the signal (optional)
- `position:` Center point position of the pulse
- `samples:` Number of samples in the waveform
- `width:` Width of the pulse

Returns:

resulting waveform

wave blackman(const samples, const amplitude=1.0, const alpha)

Blackman window function with arbitrary amplitude (a), alpha parameter and number of samples (N).

$$h(x) = a \cdot (\alpha_0 - \alpha_1 \cos(\frac{2\pi x}{N-1}) \quad + \alpha_2 \cos(\frac{4\pi x}{N-1})) \; \alpha_0 = \frac{1-\alpha}{2}; \quad \alpha_1 = \frac{1}{2}; \quad \alpha_2 = \frac{\alpha}{2};$$

Args:

- `alpha:` Width of the function
- `amplitude:` Amplitude of the signal (optional)
- `samples:` Number of samples in the waveform

Returns:

resulting waveform

wave hamming(const samples, const amplitude=1.0)

Hamming window function with arbitrary amplitude (a) and number of samples (N).

$$h(x) = a \cdot (\alpha - \beta \cos(\frac{2\pi x}{N-1})) \text{ with } \alpha = 0.54 \text{ and } \beta = 0.46$$

Args:

- `amplitude:` Amplitude of the signal (optional)
- `samples:` Number of samples in the waveform

Returns:

resulting waveform

wave hann(const samples, const amplitude=1.0)

Hann window function with arbitrary amplitude (a) and number of samples (N).

$$h(x) = a \cdot 0.5 \cdot (1 - \cos(\frac{2\pi x}{N-1}))$$

Args:

- `amplitude:` Amplitude of the signal
- `samples:` Number of samples in the waveform

Returns:

resulting waveform

wave rect(const samples, const amplitude)

Rectangle function, constants amplitude (a) over the defined number of samples.

$$h(x) = a$$

Args:

- `amplitude:` Amplitude of the signal
- `samples:` Number of samples in the waveform

Returns:

resulting waveform

wave marker(const samples, const markerValue)

Generate a waveform with marker bits set to the specified value. The analog part of the waveform is zero.

Args:

- `markerValue:` Value of the marker bits
- `samples:` Number of samples in the waveform

Returns:

resulting waveform

wave rand(const samples, const amplitude=1.0, const mean, const stdDev)

White noise with arbitrary amplitude, power and standard deviation.

Args:

- `amplitude:` Amplitude of the signal
- `mean:` Average signal level
- `samples:` Number of samples in the waveform
- `stdDev:` Standard deviation of the noise signal

Returns:

resulting waveform

wave randomGauss(const samples, const amplitude=1.0, const mean, const stdDev)

White noise with arbitrary amplitude, power and standard deviation.

Args:

- `amplitude:` Amplitude of the signal
- `mean:` Average signal level
- `samples:` Number of samples in the waveform
- `stdDev:` Standard deviation of the noise signal

Returns:

resulting waveform

wave randomUniform(const samples, const amplitude=1.0)

Random waveform with uniform distribution.

Args:

- `amplitude:` Amplitude of the signal
- `samples:` Number of samples in the waveform

Returns:

resulting waveform

wave lfsrGaloisMarker(const samples, const markerBit, const polynomial, const initial)

Generate a waveform with specified marker bit set to the Galois LFSR (linear-feedback shift register) generated sequence. The analog part of the waveform is zero. The LFSR characteristic polynomial is a member of the Galois Field of two elements and represented in binary form. See wikipedia entries for "Finite field arithmetic" and "Linear-feedback shift register (Galois LFSR)".

Args:

- `initial:` LFSR initial state, any nonzero value will work, usually 0x1
- `markerBit:` Marker bit to set (1 or 2)
- `polynomial:` LFSR characteristic polynomial in binary representation (max shift length 32), use 0x90000 for QRSS / PRBS-20
- `samples:` Number of samples in the waveform

Returns:

resulting waveform

wave chirp(const samples, const amplitude=1.0, const startFreq, const stopFreq, const phase=0)

Frequency chirp function with arbitrary amplitude, start and stop frequency, initial phase in radians and number of samples. Start and stop frequency are expressed in units of the AWG Sampling Rate. The amplitude can only be defined if the initial phase is defined as well.

Args:

- `amplitude:` Amplitude of the signal (optional)
- `phase:` Initial phase of the signal (optional)
- `samples:` Number of samples in the waveform
- `startFreq:` Start frequency of the signal
- `stopFreq:` Stop Frequency of the signal

Returns:

resulting waveform

wave rrc(const samples, const amplitude=1.0, const position, const beta, const width)

Root raised cosine function with arbitrary amplitude (a), position (p), roll-off factor (\beta) and width (w) and number of samples (N).

$$h(y) = a\frac{\sin(y\pi(1-\beta)) + 4y\beta\cos(y\pi(1+\beta))}{y\pi(1-(4y\beta)^2)} \text{ with } y(x) = 2w\frac{x-p}{N}$$

Args:

- `amplitude:` Amplitude of the signal
- `beta:` Roll-off factor
- `position:` Center point position of the pulse
- `samples:` Number of samples in the waveform
- `width:` Width of the pulse

Returns:

Resulting waveform

wave vect(const value,...)

Specify a waveform sample by sample. Each sample is defined by one of an arbitrary number of input arguments. Only recommended for short waveforms that consist of less than 100 samples. Larger waveforms may be defined in a CSV file.

Args:

- `value:` Waveform amplitude at the respective sample

Returns:

resulting waveform

wave placeholder(const samples, const marker0=false, const marker1=false)

Creates space for a single-channel waveform, optionally with markers, without actually generating any waveform data when compiling the sequence program. Actual waveform data needs to be uploaded separately via the "<dev>/AWGS/<n>/WAVEFORM/WAVES/<index>" API nodes after the sequence compilation and upload. The waveform index can be explicitly assigned to the generated placeholder wave using the assignWaveIndex instruction.

Args:

- `marker0:` true if marker bit 0 must be used (default false)
- `marker1:` true if marker bit 1 must be used (default false)
- `samples:` Number of samples in the waveform

Returns:

waveform object

The following table contains the definition of functions for waveform editing.

wave join(wave wave1, wave wave2, const interpolLength=0)

Connect two or more waveforms with optional linear interpolation between the waveforms.

Args:

- `interpolLength:` Number of samples to interpolate between waveforms (optional, default 0)
- `wave1:` Input waveform
- `wave2:` Input waveform

Returns:

joined waveform

wave join(wave wave1, wave wave2,...)

Connect two or more waveforms.

Args:

- `wave1:` Input waveform
- `wave2:` Input waveform

Returns:

joined waveform

wave interleave(wave wave1, wave wave2,...)

Interleave two or more waveforms sample by sample.

Args:

- `wave1:` Input waveform
- `wave2:` Input waveform

Returns:

interleaved waveform

wave add(wave wave1, wave wave2,...)

Add two or more waveforms sample by sample. Alternatively, the "+" operator may be used for waveform addition.

Args:

- `wave1:` Input waveform
- `wave2:` Input waveform

Returns:

sum waveform

wave multiply(wave wave1, wave wave2,...)

Multiply two or more waveforms sample by sample. Alternatively, the "*" operator may be used for waveform multiplication.

Args:

- `wave1:` Input waveform
- `wave2:` Input waveform

Returns:

product waveform

wave scale(wave waveform, const factor)

Scale the input waveform with the factor and return the scaled waveform. The input waveform remains unchanged.

Args:

- `factor:` Scaling factor
- `waveform:` Input waveform

Returns:

scaled waveform

wave flip(wave waveform)

Flip the input waveform back to front and return the flipped waveform. The input waveform remains unchanged.

Args:

- `waveform:` Input waveform

Returns:

flipped waveform

wave cut(wave waveform, const from, const to)

Cuts a segment out of the input waveform and returns it. The input waveform remains unchanged. The segment is flipped in case that "from" is larger than "to".

Args:

- `from:` First sample of the cut waveform
- `to:` Last sample of the cut waveform
- `waveform:` Input waveform

Returns:

cut waveform

wave filter(wave b, wave a, wave x)

Filter generates a rational transfer function with the waveforms a and b as numerator and denominator coefficients. The transfer function is normalized by the first element of a, which has to be non-zero. The filter is applied to the input waveform x and returns the filtered waveform.

$$y(n) = \frac{1}{a_0}\left(! \sum_{i=0}^{M} b_i x_{n-i} - \sum_{i=1}^{N} a_i y_{n-i} !\right) \text{ with } M = \text{length}(b) - 1 \text{ and } N = \text{length}(a) - 1$$

Args:

- `a:` Denominator coefficients
- `b:` Numerator coefficients
- `x:` Input waveform

Returns:

filtered waveform

wave circshift(wave a, const n)

Circularly shifts a 1D waveform and returns it.

Args:

- `n:` Number of elements to shift
- `waveform:` Input waveform

Returns:

circularly shifted waveform

Waveform Playback and Predefined Functions

The following table contains the definition of functions for waveform playback and other purposes.

void setDIO(var value)

Writes the value as a 32-bit value to the DIO bus.

The value can be either a const or a var value. Configure the Mode setting in the DIO tab when using this command. The DIO interface speed of 50 MHz limits the rate at which the DIO output value is updated.

Args:

- `value:` The value to write to the DIO (const or var)

var getDIO()

Reads a 32-bit value from the DIO bus.

Returns:

var containing the read value

var getDIOTriggered()

Reads a 32-bit value from the DIO bus as recorded at the last DIO trigger position.

Returns:

var containing the read value

void setTrigger(var value)

Sets the AWG Trigger output signals.

The state of all four AWG Trigger output signals is represented by the bits in the binary representation of the integer value. Binary notation of the form 0b0000 is recommended for readability.

Args:

- `value:` to be written to the trigger output lines

void wait(var cycles)

Waits for the given number of Sequencer clock cycles (4.44 ns per cycle).

Args:

- `cycles:` number of cycles to wait

void waitDIOTrigger()

Waits until the DIO interface trigger is active. The trigger is specified by the Strobe Index and Strobe Slope settings in the AWG Sequencer tab.

var getDigTrigger(const index)

Gets the state of the indexed Digital Trigger input (1 or 2 on UHF, 1-8 on HDAWG).

The physical signal connected to the AWG Digital Trigger input is to be configured in the Trigger sub-tab of the AWG tab.

Args:

- `index:` index of the Digital Trigger input to be read; can be either 1 or 2 on UHF, or 1-8 on HDAWG

Returns:

trigger state, either 0 or 1

void error(string msg,...)

Throws the given error message when reached.

Args:

- `msg:` Message to be displayed

void info(string msg,...)

Returns the specified message when reached.

Args:

- `msg:` Message to be displayed

void setInt(string path, var value)

Writes an integer value to one of the nodes in the device.

If the path does not start with a device identifier, then the current device is assumed.

Args:

- `path:` The node path to be written to
- `value:` The integer value to be written

void setDouble(string path, var value)

Writes a floating point value to one of the nodes in the device.

If the path does not start with a device identifier, then the current device is assumed.

Args:

- `path:` The node path to be written to
- `value:` The integer or floating point value to be written

void setDouble(string path, var value, const scale)

Writes a floating point value to one of the nodes in the device.

If the path does not start with a device identifier, then the current device is assumed.

Args:

- `path:` The node path to be written to
- `scale:` Scaling value to be applied to the value before writing to the node
- `value:` The integer or floating point value to be written

void setID(var id)

Sets the ID value that is attached to data streamed from the device to the host PC. The ID value is useful for synchronizing the data acquisition process in combination with the Sweeper or the Software Trigger. The ID value is denoted AWG Seq Index in the tree of tools like the plotter.

Args:

- `id:` The new ID to be attached to streaming data of the device

void setSeqIndex(var id)

Sets the ID value that is attached to data streamed from the device to the host PC. The ID value is useful for synchronizing the data acquisition process in combination with the Sweeper or the Software Trigger. The ID value is denoted AWG Seq Index in the tree of tools like the plotter. The setSeqIndex function is identical to the setID function.

Args:

- `id:` The new ID to be attached to streaming data of the device

void sync()

Perform Multi-Device synchronization command for all devices at this point. Leader/Follower assignment is automatic.

Only for programs running on multiply synchronized instruments.

void waitWave()

Waits until the AWG is done playing the current waveform.

void randomSeed()

Generate a new seed for the subsequent random vector commands.

void assignWaveIndex(const output, wave waveform, const index)

void assignWaveIndex(wave waveform, const index)

void playWave(const output, wave waveform, const rate=AWG_RATE_DEFAULT)

Starts to play the given waveforms on the defined output channels. The playback begins as soon as the previous waveform playback is finished.

Args:

- `output:` defines on which output the following waveform is played
- `rate:` sample rate with which the AWG plays the waveforms (default set in the user interface).
- `waveform:` waveform to be played

void playWave(const output, wave waveform,...)

Starts to play the given waveforms on the defined output channels. It can contain multiple waveforms with an output definition. The playback begins as soon as the previous waveform playback is finished.

Args:

- `output:` defines on which output the following waveform is played
- `waveform:` waveform to be played

void playWave(wave waveform, const rate=AWG_RATE_DEFAULT)

Starts to play the given waveforms, output channels are assigned automatically depending on the number of input waveforms. The playback begins as soon as the previous waveform playback is finished.

Args:

- `rate:` sample rate with which the AWG plays the waveforms (default set in the user interface).
- `waveform:` waveform to be played

void playWave(wave waveform,...)

Starts to play the given waveforms, output channels are assigned automatically depending on the number of input waveforms. The playback begins as soon as the previous waveform playback is finished.

Args:

- `waveform:` waveform to be played

void setUserReg(const register, var value)

Writes a value to one of the User Registers (indexed 0 to 15).

The User Registers may be used for communicating information to the LabOne User Interface or a running API program.

Args:

- `register:` The register index (0 to 15) to be written to
- `value:` The integer value to be written

var getUserReg(const register)

Reads the value from one of the User Registers (indexed 0 to 15). The User Registers may be used for communicating information to the LabOne User Interface or a running API program.

Args:

- `register:` The register to be read (0 to 15)

Returns:

current register value

void playZero(const samples)

void playZero(const samples, const rate)

void setRate(const rate)

Overwrites the default Sampling Rate for the following playWave commands.

Args:

- `rate:` New default sampling rate

void playWaveNow(const output, wave waveform, const rate=AWG_RATE_DEFAULT)

Starts to play the given waveforms on the defined output channels. It starts immediately even if the previous waveform playback is still in progress.

Args:

- `output:` defines on which output the following waveform is played
- `rate:` sample rate with which the AWG plays the waveforms (default set in the user interface).
- `waveform:` waveform to be played

void playWaveNow(const output, wave waveform,...)

Starts to play the given waveforms on the defined output channels. It can contain multiple waveforms with an output definition. It starts immediately even if the previous waveform playback is still in progress.

Args:

- `output:` defines on which output the following waveform is played
- `waveform:` waveform to be played

void playWaveNow(wave waveform, const rate=AWG_RATE_DEFAULT)

Starts to play the given waveforms, channels are assigned automatically depending on the number of input waveforms. It starts immediately even if the previous waveform playback is still in progress.

Args:

- `rate:` sample rate with which the AWG plays the waveforms (default set in the user interface).
- `waveform:` waveform to be played

void playWaveNow(wave waveform,...)

Starts to play the given waveforms, channels are assigned automatically depending on the number of input waveforms. It starts immediately even if the previous waveform playback is still in progress.

Args:

- `waveform:` waveform to be played

void playWaveIndexed(const output, wave waveform, var offset, const length, const rate=AWG_RATE_DEFAULT)

Starts to play the specified part of the given waveforms on the defined output channels. It can contain multiple waveforms with an output definition. The playback begins as soon as the previous waveform playback is finished.

Args:

- `length:` number of samples to be played from this waveform
- `offset:` offset in samples from the start of the waveform
- `output:` defines on which output the following waveform is played
- `rate:` sample rate with which the AWG plays the waveforms (default set in the user interface).
- `waveform:` waveform to be played

void playWaveIndexed(wave waveform, var offset, const length, const rate=AWG_RATE_DEFAULT)

Starts to play the specified part of the given waveforms, channels are assigned automatically depending on the number of input waveforms. The playback begins as soon as the previous waveform playback is finished.

Args:

- `length:` number of samples to be played from this waveform
- `offset:` offset in samples from the start of the waveform
- `rate:` sample rate with which the AWG plays the waveforms (default set in the user interface).
- `waveform:` waveform to be played

void playWaveIndexedNow(const output, wave waveform, var offset, const length, const rate=AWG_RATE_DEFAULT)

Starts to play the specified part of the given waveforms on the defined output channels. It can contain multiple waveforms with an output definition. It starts immediately even if the previous waveform playback is still in progress.

Args:

- `length:` number of samples to be played from this waveform
- `offset:` offset in samples from the start of the waveform
- `output:` defines on which output the following waveform is played
- `rate:` sample rate with which the AWG plays the waveforms (default set in the user interface).
- `waveform:` waveform to be played

void playWaveIndexedNow(wave waveform, var offset, const length, const rate=AWG_RATE_DEFAULT)

Starts to play the specified part of the given waveforms, channels are assigned automatically depending on the number of input waveforms. It starts immediately even if the previous waveform playback is still in progress.

Args:

- `length:` number of samples to be played from this waveform
- `offset:` offset in samples from the start of the waveform
- `rate:` sample rate with which the AWG plays the waveforms (default set in the user interface).
- `waveform:` waveform to be played

void playDIOWave(wave waveform, const rate=AWG_RATE_DEFAULT)

Starts to play the given waveforms, channels are assigned automatically depending on the number of input waveforms, with enabled 4-channel-mode. Configure the Signal and Channel settings in the Aux tab in combination with this function. The playback begins as soon as the previous waveform playback is finished.

Args:

- `rate:` sample rate with which the AWG plays the waveforms (default set in the user interface).
- `waveform:` waveform to be played

void playDIOWave(wave waveform,...)

Starts to play the given waveforms, channels are assigned automatically depending on the number of input waveforms, with enabled 4-channel-mode. Configure the Signal and Channel settings in the Aux tab in combination with this function. The playback begins as soon as the previous waveform playback is finished.

Args:

- `waveform:` waveform to be played

void playAuxWave(const output, wave waveform, const rate=AWG_RATE_DEFAULT)

Starts to play the given waveforms on the defined output channels with enabled 4-channel-mode. Configure the Signal and Channel settings in the Aux tab in combination with this function. The playback begins as soon as the previous waveform playback is finished.

Args:

- `output:` defines on which output the following waveform is played
- `rate:` sample rate with which the AWG plays the waveforms (default set in the user interface).
- `waveform:` waveform to be played

void playAuxWave(const output, wave waveform,...)

Starts to play the given waveforms on the defined output channels with enabled 4-channel-mode. It can contain multiple waveforms with an output definition. Configure the Signal and Channel settings in the Aux tab in combination with this function. The playback begins as soon as the previous waveform playback is finished.

Args:

- `output:` defines on which output the following waveform is played
- `waveform:` waveform to be played

void playAuxWave(wave waveform, const rate=AWG_RATE_DEFAULT)

Starts to play the given waveforms, channels are assigned automatically depending on the number of input waveforms, with enabled 4-channel-mode. Configure the Signal and Channel settings in the Aux tab in combination with this function. The playback begins as soon as the previous waveform playback is finished.

Args:

- `rate:` sample rate with which the AWG plays the waveforms (default set in the user interface).
- `waveform:` waveform to be played

void playAuxWave(wave waveform,...)

Starts to play the given waveforms, channels are assigned automatically depending on the number of input waveforms, with enabled 4-channel-mode. Configure the Signal and Channel settings in the Aux tab in combination with this function. The playback begins as soon as the previous waveform playback is finished.

Args:

- `waveform:` waveform to be played

void playAuxWaveIndexed(const output, wave waveform, var offset, const length, const rate=AWG_RATE_DEFAULT)

Starts to play the specified part of the given waveforms on the defined output channels with enabled 4-channel-mode. Configure the Signal and Channel settings in the Aux tab in combination with this function. The playback begins as soon as the previous waveform playback is finished.

Args:

- `length:` number of samples to be played from this waveform
- `offset:` offset in samples from the start of the waveform
- `output:` defines on which output the following waveform is played
- `rate:` sample rate with which the AWG plays the waveforms (default set in the user interface).
- `waveform:` waveform to be played

void playAuxWaveIndexed(wave waveform, var offset, const length, const rate=AWG_RATE_DEFAULT)

Starts to play the specified part of the given waveforms, channels are assigned automatically depending on the number of input waveforms, with enabled 4-channel-mode. Configure the Signal and Channel settings in the Aux tab in combination with this function. The playback begins as soon as the previous waveform playback is finished.

Args:

- `length:` number of samples to be played from this waveform
- `offset:` offset in samples from the start of the waveform
- `rate:` sample rate with which the AWG plays the waveforms (default set in the user interface).
- `waveform:` waveform to be played

void waitAnaTrigger(const index, const value)

Waits until the indexed Analog Trigger input (1 or 2) is equal to the given value (0 or 1). The physical signal connected to the AWG Analog Trigger inputs as well as the trigger level is to be configured in the Trigger sub-tab of the AWG tab.

Args:

- `index:` index of the analog trigger input to be waited on; can be either 1 or 2 on UHF, or 1 to 8 on HDAWG
- `value:` value to be compared with the Analog Trigger input, can be either 0 or 1

var getAnaTrigger(const index)

Gets the state of the indexed Analog Trigger input (1 or 2 on UHF, 1-8 on HDAWG).

The physical signal connected to the AWG Analog Trigger inputs as well as the trigger level is to be configured in the Trigger sub-tab of the AWG tab.

Args:

- `index:` index of the Analog Trigger input to be read; an be either 1 or 2 on UHF, or 1-8 on HDAWG

Returns:

trigger state, either 0 or 1

var getSweeperLength(const index)

Reads the sweep Length as configured in the Sweeper tab. The length is only valid when AWG Control is enabled in the Sweeper tab.

Args:

- `index:` The index of the Sweeper parameter. Currently only the value of 1 is accepted.

Returns:

length configured by the Sweeper

void now()

Resets the local timer.

void at(var time)

Waits until the local timer reaches the given value.

Args:

- `time:` value to wait for

void lock(wave waveform)

Ensures that the waveform is kept in the cache memory until the unlock command is used.

Args:

- `waveform:` The waveform to lock

void unlock(wave waveform)

Allow the compiler to use this memory block again to cache other waveforms.

Only valid after the waveform was previously locked using the lock command.

Args:

- `waveform:` The waveform to unlock

void waitTrigger(const mask, const value)

Waits until the masked trigger input is equal to the given value.

Args:

- `mask:` mask to be applied to the input signal
- `value:` value to be compared with the trigger input

void waitDigTrigger(const index, const value)

Waits until the indexed Digital Trigger (1 or 2) is equal to the given value (0 or 1). The physical signals connected to the two AWG Digital Triggers are to be configured in the Trigger sub-tab of the AWG tab.

Args:

- `index:` index of the digital trigger input to be waited on; can be either 1 or 2
- `value:` value to be compared with the digital trigger input, can be either 0 or 1

void startQA(const weighted_integrator_mask, const monitor, const result_address, const trigger)

Starts the Quantum Analysis Result and Input units by setting and clearing appropriate AWG trigger output signals. The choice of whether to start one or the other or both units can be controlled using the command argument. An bitmask may be used to select explicitly which of the ten possible qubit results should be read. If no qubit results are enabled, then the Quantum Analysis Result unit will not be triggered. An optional value may be used to set the normal trigger outputs of the AWG together with starting the Quantum Analysis Result and input units. If the value is not used, then the trigger signals will be cleared.

Args:

- `monitor:` Enable for QA monitor, default: false
- `result_address:` Set address associated with result, default: 0x0
- `trigger:` Trigger value, default: 0x0
- `weighted_integrator_mask:` Integration unit enable mask, default: QA_INT_ALL

var startQAResult(const mask, const trigger)

Starts the Quantum Analysis Result unit by setting and clearing appropriate AWG trigger output signals. An optional bitmask may be used to select explicitly which of the ten possible qubit results should be read. An optional value may be used to set the normal trigger outputs of the AWG together with starting the Quantum Analysis Result unit. If the value is not used, then the trigger signals will be cleared.

Args:

- `mask:` bitmask defining which of the ten qubit results should be read
- `trigger:` bitmask to apply to the normal trigger outputs, similar in use to the setTrigger command

var startQAMonitor(const trigger)

Starts the Quantum Analysis Monitor unit by setting and clearing appropriate AWG trigger output signals. An optional value may be used to set the normal trigger outputs of the AWG together with starting the Quantum Analysis Monitor unit. If the value is not used, then the trigger signals will be cleared.

Args:

- `trigger:` bitmask to apply to the normal trigger outputs, similar in use to the setTrigger command

void setReadoutRegisterAddress(var results_address)

var getQAResult()

Reads the value from Quantum Analysis Result unit.

Returns:

current state of all measured qubits

void waitQAResultTrigger()

Waits until the Quantum Analysis Result unit has produced a new qubit measurement result.

void resetOscPhase()

Reset the phase of the oscillator controllable by the AWG core.

var getFeedback(const data_type)

Read the last feedback message. The argument specify which data the function should return.

Args:

- **`data_type:`** Specifies which data the function should return: ZSYNC_DATA_RAW: Return the data received on the ZSync as-is without parsing. The structure of the message can change across different LabOne releases.

Returns:

var containing the read value

var getFeedback(const data_type, var wait_cycles)

void waitZSyncTrigger()

Waits for a trigger over ZSync.

void resetRTLoggerTimestamp()

Reset the timestamp counter of the Real-Time Logger.

Accessing Instrument Settings

Using the sequencer instructions **`setInt`** and **`setDouble`**, a large number of instrument settings may be accessed directly from the sequencer with a much shorter latency than when accessed via the LabOne API. The nodes accessible with **`setInt setDouble`** are a subset of the full list of device nodes accessible via the LabOne API (see Device Node Tree) and are listed in the table below together with their data type, latency class, and timing determinicity characteristics.

There are 3 levels of latency performance depending on how the node settings are processed on the instrument. Nodes that are classified with latency "medium" are processed with the instrument firmware by a non-realtime processor. These nodes are set typically within 100 microseconds, however not with deterministic timing. For some of the nodes, additional time is needed to take effect due to the time scale of the related hardware settings, e.g. changing the settings of the analog signal path. This needs to be taken into account by including sufficient waiting time by a **`wait`** sequencer instruction after setting the node.

Nodes classified with latency "low" are processed with deterministic timing on a dedicated bus inside the FPGA. The latency is of the order of 100 nanoseconds. Nodes classified with latency "ultra-low", such as timing-critical phase changes, are accessed via their dedicated signal path on the FPGA, and offer similarly low and deterministic latency as e.g. **`playWave`** instructions.

Providing the node as a character string is less flexible from the AWG sequencer than from the API: wildcards (*) are not supported, the node cannot start with a dash (/), and the device ID cannot be specified since it is excluded that the sequencer accesses other devices. This code example illustrates these restrictions:

```
setDouble("oscs/1/freq", 1e6); // permitted
setDouble("oscs/*/freq", 1e6); // not permitted
setDouble("dev8000/oscs/1/freq", 1e6); // not permitted
setDouble("/dev8000/oscs/1/freq", 1e6); // not permitted
setDouble("/oscs/1/freq", 1e6); // not permitted
```

In the table, the sequence [i-j] indicate a numeric range of valid indices from i to j

Node	Data type	Latency	Deterministic timing
demods/[0-7]/adcselect	Integer	medium	no
demods/[0-7]/order	Integer	medium	no
demods/[0-7]/rate	Float	medium	no
demods/[0-7]/oscselect	Integer	medium	no
demods/[0-7]/harmonic	Integer	medium	no
demods/[0-7]/phaseshift	Float	medium	no
demods/[0-7]/sinc	Integer	medium	no
demods/[0-7]/bypass	Integer	medium	no
demods/[0-7]/timeconstant	Float	medium	no
demods/[0-7]/enable	Integer	medium	no
scopes/0/enable	Integer	medium	no
scopes/0/time	Integer	medium	no
scopes/0/trigenable	Integer	medium	no
scopes/0/trigrising	Integer	medium	no
scopes/0/trigfalling	Integer	medium	no
scopes/0/triglevel	Float	medium	no
scopes/0/length	Integer	medium	no
scopes/0/trigholdoff	Float	medium	no
scopes/0/trigchannel	Integer	medium	no
scopes/0/channels/[0-1]/inputselect	Integer	medium	no
scopes/0/channels/[0-1]/bwlimit	Integer	medium	no
scopes/0/segments/count	Integer	medium	no
scopes/0/channel	Integer	medium	no
scopes/0/single	Integer	medium	no
scopes/0/trighysteresis/absolute	Float	medium	no
scopes/0/cont	Integer	medium	no
scopes/0/trighysteresis/relative	Float	medium	no
scopes/0/trighysteresis/mode	Integer	medium	no
scopes/0/trigforce	Integer	medium	no
scopes/0/segments/counts/enable	Integer	medium	no
scopes/0/trigstate	Integer	medium	no
scopes/0/triggate/enable	Integer	medium	no
scopes/0/triggate/inputselect	Integer	medium	no
scopes/0/trigreference	Float	medium	no
scopes/0/trigdelay	Float	medium	no
scopes/0/trigholdofftriggerenable	Integer	medium	no
scopes/0/trigholdofftrigger	Integer	medium	no
scopes/0/channels/[0-1]/limitlower	Float	medium	no
scopes/0/channels/[0-1]/limitupper	Float	medium	no
sigins/[0-1]/ac	Integer	medium	no
sigins/[0-1]/imp50	Integer	medium	no
sigins/[0-1]/bw	Integer	medium	no
sigins/[0-1]/on	Integer	medium	no

Node	Data type	Latency	Deterministic timing
sigins/[0-1]/range	Integer	medium	no
sigins/[0-1]/diff	Integer	medium	no
sigins/[0-1]/autorange	Integer	medium	no
sigins/[0-1]/scaling	Float	medium	no
sigouts/0/on	Integer	medium	no
sigouts/0/range	Float	medium	no
sigouts/0/imp50	Integer	medium	no
sigouts/0/autorange	Integer	medium	no
sigouts/[0-1]/on	Integer	medium	no
sigouts/[0-1]/range	Float	medium	no
sigouts/[0-1]/imp50	Integer	medium	no
sigouts/[0-1]/autorange	Integer	medium	no
sigouts/[0-1]/offset	Float	medium	no
sigouts/[0-1]/enables/[0-7]	Integer	medium	no
sigouts/[0-1]/amplitudes/[0-7]	Float	medium	no
mods/[0-1]/enable	Integer	medium	no
mods/[0-1]/output	Integer	medium	no
mods/[0-1]0/mode	Integer	medium	no
mods/[0-1]/sidebands/[0-1]/mode	Integer	medium	no
mods/[0-1]/freqdevenable	Integer	medium	no
mods/[0-1]/freqdev	Float	medium	no
mods/[0-1]/index	Float	medium	no
mods/[0-1]/sampleenable	Integer	medium	no
mods/[0-1]/operation	Integer	medium	no
mods/[0-1]/rate	Float	medium	no
mods/[0-1]/trigger	Integer	medium	no
mods/[0-1]/carrier/inputselect	Integer	medium	no
mods/[0-1]/sidebands/[0-1]/inputselect	Integer	medium	no
mods/[0-1]/carrier/order	Integer	medium	no
mods/[0-1]/sidebands/[0-1]/order	Integer	medium	no
mods/[0-1]/carrier/timeconstant	Float	medium	no
mods/[0-1]/sidebands/[0-1]/timeconstant	Float	medium	no
mods/[0-1]/carrier/oscselect	Integer	medium	no
mods/[0-1]/sidebands/[0-1]/oscselect	Integer	medium	no
mods/[0-1]/carrier/harmonic	Integer	medium	no
mods/[0-1]/sidebands/[0-1]/harmonic	Integer	medium	no
mods/[0-1]/carrier/phaseshift	Float	medium	no
mods/[0-1]/sidebands/[0-1]/phaseshift	Float	medium	no
mods/[0-1]/carrier_enable	Integer	medium	no
mods/[0-1]/sidebands/[0-1]/enable	Integer	medium	no
mods/[0-1]/carrier_gain	Float	medium	no
mods/[0-1]/sideband[0-1]_gain	Float	medium	no
boxcars/[0-1]/enable	Integer	medium	no

Node	Data type	Latency	Deterministic timing
boxcars/[0-1]/periods	Integer	medium	no
boxcars/[0-1]/windowstart	Float	medium	no
boxcars/[0-1]/insel	Integer	medium	no
boxcars/[0-1]/oscsel	Integer	medium	no
boxcars/[0-1]/limitrate	Float	medium	no
boxcars/[0-1]/baseline/enable	Integer	medium	no
boxcars/[0-1]/baseline/windowstart	Float	medium	no
boxcars/[0-1]/windowsize	Float	medium	no
outputpwas/[0-1]/enable	Integer	medium	no
outputpwas/[0-1]/single	Integer	medium	no
outputpwas/[0-1]/oscselect	Integer	medium	no
outputpwas/[0-1]/insel	Integer	medium	no
outputpwas/[0-1]/mode	Integer	medium	no
outputpwas/[0-1]/harmonic	Integer	medium	no
outputpwas/[0-1]/shift	Float	medium	no
outputpwas/[0-1]/termination	Integer	medium	no
outputpwas/[0-1]/samplecount	Long	medium	no
outputpwas/[0-1]/holdoff	Float	medium	no
outputpwas/[0-1]/progress	Float	medium	no
outputpwas/[0-1]/status	Integer	medium	no
inputpwas/[0-1]/enable	Integer	medium	no
inputpwas/[0-1]/single	Integer	medium	no
inputpwas/[0-1]/oscselect	Integer	medium	no
inputpwas/[0-1]/insel	Integer	medium	no
inputpwas/[0-1]/mode	Integer	medium	no
inputpwas/[0-1]/harmonic	Integer	medium	no
inputpwas/[0-1]/shift	Float	medium	no
inputpwas/[0-1]/termination	Integer	medium	no
inputpwas/[0-1]/samplecount	Long	medium	no
inputpwas/[0-1]/holdoff	Float	medium	no
pids/[0-3]/enable	Integer	medium	no
pids/[0-3]/demod/adcselect	Integer	medium	no
pids/[0-3]/demod/order	Integer	medium	no
pids/[0-3]/demod/timeconstant	Float	medium	no
pids/[0-3]/setpoint	Float	medium	no
pids/[0-3]/dlimittimeconstant	Float	medium	no
pids/[0-3]/limitupper	Float	medium	no
pids/[0-3]/limitlower	Float	medium	no
pids/[0-3]/p	Float	medium	no
pids/[0-3]/i	Float	medium	no
pids/[0-3]/d	Float	medium	no
pids/[0-3]/demod/harmonic	Integer	medium	no
pids/[0-3]/outputchannel	Integer	medium	no

Node	Data type	Latency	Deterministic timing
pids/[0-3]/output	Integer	medium	no
pids/[0-3]/phaseunwrap	Integer	medium	no
pids/[0-3]/stream/rate	Float	medium	no
pids/[0-3]/rate	Float	medium	no
pids/[0-3]/mode	Integer	medium	no
pids/[0-3]/inputsource	Integer	medium	no
pids/[0-3]/inputchannel	Integer	medium	no
pids/[0-3]/center	Double	medium	no
pids/[0-3]/pll/automode	Integer	medium	no
awgs/0/enable	Integer	medium	no
awgs/0/single	Integer	medium	no
awgs/0/time	Integer	medium	no
awgs/0/userregs/[0-15]]	Integer	medium	no
awgs/0/triggers/[0-1]/level	Float	medium	no
awgs/0/triggers/[0-1]/hysteresis/absolute	Float	medium	no
awgs/0/triggers/[0-1]/hysteresis/relative	Float	medium	no
awgs/0/triggers/[0-1]/hysteresis/mode	Integer	medium	no
awgs/0/triggers/[0-1]/rising	Integer	medium	no
awgs/0/triggers/[0-1]/falling	Integer	medium	no
awgs/0/triggers/[0-1]/channel	Integer	medium	no
awgs/0/triggers/[0-1]/force	Integer	medium	no
awgs/0/triggers/[0-1]/state	Integer	medium	no
awgs/0/triggers/[0-1]/gate/enable	Integer	medium	no
awgs/0/triggers/[0-1]/gate/inputselect	Integer	medium	no
awgs/0/auxtriggers/[0-1]/channel	Integer	medium	no
awgs/0/auxtriggers/[0-1]/rising	Integer	medium	no
awgs/0/auxtriggers/[0-1]/falling	Integer	medium	no
awgs/0/auxtriggers/[0-1]/state	Integer	medium	no
awgs/0/outputs/[0-1]/amplitude	Float	medium	no
awgs/0/outputs/[0-1]/mode	Integer	medium	no
auxouts/[0-3]/preoffset	Double	medium	no
auxouts/[0-3]/offset	Float	medium	no
auxouts/[0-3]/scale	Float	medium	no
auxouts/[0-3]/limitlower	Float	medium	no
auxouts/[0-3]/limitupper	Float	medium	no
auxouts/[0-3]/offset	Integer	medium	no
auxouts/[0-3]/outputselect	Integer	medium	no
auxouts/[0-3]/demodselect	Integer	medium	no
triggers/in/[0-3]/imp50	Integer	medium	no
triggers/in/[0-3]/level	Integer	medium	no
triggers/in/[0-3]/dcc_fedge	Integer	medium	no
triggers/out/[0-3]/srcsel	Integer	medium	no
triggers/out/[0-3]/dir	Integer	medium	no

Node	Data type	Latency	Deterministic timing
triggers/out/[0-3]/static_value	Integer	medium	no
triggers/out/[0-3]/hold	Integer	medium	no
triggers/out/[0-3]/delay	Float	medium	no
aucarts/[0-1]/mode	Integer	medium	no
aucarts/[0-1]/enable	Integer	medium	no
aucarts/[0-1]/rate	Float	medium	no
aucarts/[0-1]/ops/[0-1]/demodselect	Integer	medium	no
aucarts/[0-1]/ops/[0-1]/value	Integer	medium	no
aucarts/[0-1]/ops/[0-1]/coeff	Integer	medium	no
aucarts/[0-1]/ops/[0-1]/scale	Float	medium	no
aupolars/[0-1]/mode	Integer	medium	no
aupolars/[0-1]/enable	Integer	medium	no
aupolars/[0-1]/rate	Float	medium	no
aupolars/[0-1]/reserved1	Integer	medium	no
aupolars/[0-1]/ops/[0-1]/demodselect	Integer	medium	no
aupolars/[0-1]/ops/[0-1]/value	Integer	medium	no
aupolars/[0-1]/ops/[0-1]/coeff	Integer	medium	no
aupolars/[0-1]/ops/[0-1]/scale	Float	medium	no
aupolars/[0-1]/flags	Integer	medium	no
scopes/0/stream/enables/[0-1]	Integer	medium	no
scopes/0/stream/rate	Integer	medium	no
oscs/[0-7]/freq	Frequency	low	yes
demods/[0-7]/oscselect	Integer	ultra-low	yes
demods/[0-7]/phaseshift	Integer	ultra-low	yes

Nodes accessible with `setInt` and `setDouble`

Expressions

Expressions may be used for making computations based on mathematical functions and operators. There are two kinds of expressions: those evaluated at compile time (the moment of clicking "Save" or "Save as..." in the user interface), and those evaluated at run time (after clicking "Run/Stop" or "Start"). Compile-time evaluated expressions only involve constants (`const`) or compile-time variables (`cvar`) and can be computed at compile time by the host computer. Such expressions can make use of standard mathematical functions and floating point arithmetic. Run-time evaluated expressions involve variables (`var`) and are evaluated by the Sequencer on the instrument. Due to the limited computational capabilities of the Sequencer, these expressions may only operate on integer numbers and there are less operators available than at compile time.

The following table contains the list of mathematical functions supported at compile time.

Table 5.41: Mathematical Functions

Function	Description
const abs(const c)	absolute value
const acos(const c)	inverse cosine
const acosh(const c)	hyperbolic inverse cosine
const asin(const c)	inverse sine
const asinh(const c)	hyperbolic inverse sine

Function	Description
const atan(const c)	inverse tangent
const atanh(const c)	hyperbolic inverse tangent
const cos(const c)	cosine
const cosh(const c)	hyperbolic cosine
const exp(const c)	exponential function
const ln(const c)	logarithm to base e (2.71828...)
const log(const c)	logarithm to the base 10
const log2(const c)	logarithm to the base 2
const log10(const c)	logarithm to the base 10
const sign(const c)	sign function -1 if x<0; 1 if x>0
const sin(const c)	sine
const sinh(const c)	hyperbolic sine
const sqrt(const c)	square root
const tan(const c)	tangent
const tanh(const c)	hyperbolic tangent
const ceil(const c)	smallest integer value not less than the argument
const round(const c)	round to nearest integer
const floor(const c)	largest integer value not greater than the argument
const avg(const c1, const c2,...)	mean value of all arguments
const max(const c1, const c2,...)	maximum of all arguments
const min(const c1, const c2,...)	minimum of all arguments
const pow(const base, const exp)	first argument raised to the power of second argument
const sum(const c1, const c2,...)	sum of all arguments

The following table contains the list of predefined mathematical constants. These can be used for convenience in compile-time evaluated expressions.

Table 5.42: Predefined Constants

Name	Value	Description
M_E	2.7182818284590452353602874713526625 0	e
M_LOG2E	1.4426950408889634073599246810018921 4	log2(e)
M_LOG10E	0.4342944819032518276511289189166050 82	log10(e)
M_LN2	0.6931471805599453094172321214581765 68	loge(2)
M_LN10	2.3025850929940456840179914546843642 1	loge(10)
M_PI	3.1415926535897932384626433832795028 8	pi
M_PI_2	1.5707963267948966192313216916397514 4	pi/2
M_PI_4	0.7853981633974483096156608458198757 21	pi/4
M_1_PI	0.3183098861837906715377675267450287 24	1/pi
M_2_PI	0.6366197723675813430755350534900574 48	2/pi
M_2_SQRTPI	1.1283791670955125738961589031215451 7	2/sqrt(pi)
M_SQRT2	1.4142135623730950488016887242096980 8	sqrt(2)
M_SQRT1_2	0.7071067811865475244008443621048490 39	1/sqrt(2)

Table 5.43: Operators supported at compile time

Operator	Description	Priority
=	assignment	-1
+=, -=, *=, /=, %=, &=, |=, <<=, >>=	assignment by sum, difference, product, quotient, remainder, AND, OR, left shift, and right shift	-1
||	logical OR	1
&&	logical AND	2
|	bit-wise logical OR	3
&	bit-wise logical AND	4
!=	not equal	5
==	equal	5
<=	less or equal	6
>=	greater or equal	6
>	greater than	6
<	less than	6
<<	arithmetic left bit shift	7
>>	arithmetic right bit shift	7
+	addition	8
-	subtraction	8
*	multiplication	9
/	division	9
~	bit-wise logical negation	10

Table 5.44: Operators supported at run time

Operator	Description	Priority
=	assignment	-1
+=, -=, *=, /=, %=, &=, |=, <<=, >>=	assignment by sum, difference, product, quotient, remainder, AND, OR, left shift, and right shift	-1
||	logical OR	1
&&	logical AND	2
|	bit-wise logical OR	3
&	bit-wise logical AND	4
==	equal	5
!=	not equal	5
<=	less or equal	6
>=	greater or equal	6
>	greater than	6
<	less than	6
<<	left bit shift	7
>>	right bit shift	7
+	addition	8
-	subtraction	8
~	bit-wise logical negation	9

Control Structures

Functions may be declared using the `var` keyword. **Procedures** may be declared using the `void` keyword. Functions must return a value, which should be specified using the `return` keyword. Procedures can not return values. Functions and procedures may be declared with an arbitrary number of arguments. The `return` keyword may also be used without arguments to return from an arbitrary point within the function or procedure. Functions and procedures may contain variable and constant declarations. These declarations are local to the scope of the function or procedure.

```
var function_name(argument1, argument2, ...) {
  // Statements to be executed as part of the function.
  return constant-or-variable;
}

void procedure_name(argument1, argument2, ...) {
  // Statements to be executed as part of the procedure.
  // Optional return statement
  return;
}
```

An **if-then-else** structure is used to create a conditional branching point in a sequencer program.

```
// If-then-else statement syntax
if (expression) {
  // Statements to execute if 'expression' evaluates to 'true'.
} else {
  // Statements to execute if 'expression' evaluates to 'false'.
}

// If-then-else statement short syntax
(expression)?(statement if true):(statement if false)
```

```
// If-then-else statement example
const REQUEST_BIT     = 0x0001;
const ACKNOWLEDGE_BIT = 0x0002;
const IDLE_BIT        = 0x8000;
var dio = getDIO();
if (dio & REQUEST_BIT) {
  dio = dio | ACKNOWLEDGE_BIT;
  setDIO(dio);
} else {
  dio = dio | IDLE_BIT;
  setDIO(dio);
}
```

A **switch-case** structure serves to define a conditional branching point similarly to the `if-then-else` statement, but is used to split the sequencer thread into more than two branches. Unlike the `if-then-else` structure, the switch statement is synchronous, which means that the execution time is the same for all branches and determined by the execution time of the longest branch. If no default case is provided and no case matches the condition, all cases will be skipped. The case arguments need to be of type `const`.

```
// Switch-case statement syntax
switch (expression) {
  case const-expression:
    expression;
  ...
  default:
    expression;
}
```

```
// Switch-case statement example
switch (getDIO()) {
  case 0:
    playWave(gauss(1024,1.0,512,64));
```

```
  case 1:
    playWave(gauss(1024,1.0,512,128));
  case 2:
    playWave(drag(1024,1.0,512,64));
  default:
    playWave(drag(1024,1.0,512,128));
}
```

The for loop is used to iterate through a code block several times. The `initialization` statement is executed before the loop starts. The `end-expression` is evaluated at the start of each iteration and determines when the loop should stop. The loop is executed as long as this expression is true. The `iteration-expression` is executed at the end of each loop iteration.

Depending on how the `for` loop is set up, it can be either evaluated at compile time or at run time. Run-time evaluation is typically used to play series of waveforms. Compile-time evaluation is typically used for advanced waveform generation, e.g. to generate a series of waveforms with varying amplitude. For a run-time evaluated `for` loop, use the `var` data type as a loop index. To ensure that a loop is evaluated at compile time, use the `cvar` data type as a loop index. Furthermore, the compile-time `for` loop should only contain waveform generation/editing operations and it can't contain any variables of type `var`. The following code example shows both versions of the loop.

```
// For loop syntax
for (initialization; end-expression; iteration-expression) {
  // Statements to execute while end-expression evaluates to true
}
```

```
// FOR loop example to assemble a train of pulses into
// a single waveform (compile-time execution)
cvar gain_factor; // CVAR: integer or float values allowed
wave w_pulse_series;
for (gain_factor = 0; gain_factor < 1.0; gain_factor = gain_factor + 0.1) {
  w_pulse_series = join(w_pulse_series, gain_factor*gauss(1008, 504, 100));
}

// Playback of waveform defined using compile-time FOR loop
playWave(w_pulse_series);

// FOR loop example to vary waiting time between
// waveform playbacks (run-time execution)
var i; // VAR: integer values allowed
for (i = 0; i < 1000; i = i + 100) {
  playWave(gauss(1008, 504, 100));
  waitWave();
  wait(i);
}
```

The while loop is a simplified version of the `for` loop. The `end-expression` is evaluated at the start of each loop iteration. The contents of the loop are executed as long as this expression is true. Like the `for` loop, this loop comes in a compile-time version (if the end-expression involves only `cvar` and `const`) and in a run-time version (if the end-expression involves also `var` data types).

```
// While loop syntax
while (end-expression) {
 // Statements to execute while end-expression evaluates to true
}
```

```
// While loop example
const STOP_BIT = 0x8000;
var run = 1;
var i = 0;
var dio = 0;
while (run) {
  dio = getDIO();
  run = dio & STOP_BIT;

  dio = dio | (i & 0xff);
```

```
  setDIO(dio);
  i = i + 1;
}
```

The **repeat loop** is a simplified version of the `for` loop. It repeats the contents of the loop a fixed number of times. In contrast to the `for` loop, the repetition number of the `repeat` loop must be known at compile time, i.e., `const-expression` can only depend on constants and not on variables. Unlike the `for` and the `while` loop, this loop comes only in a run-time version. Thus, no `cvar` data types may be modified in the loop body.

```
// Repeat loop syntax
repeat (constant-expression) {
  // Statements to execute
}
```

```
// Repeat loop example
repeat (100) {
  setDIO(0x1);
  wait(10);
  setDIO(0x0);
  wait(10);
}
```

5.14.4. Functional Elements

Table 5.45: AWG tab: Control sub-tab

Control/Tool	Option/ Range	Description
Start		Runs the AWG.
Sampling Rate	220 kSa/s to 1.8 GSa/s	AWG sampling rate. This value is used by default and can be overridden in the Sequence program. The numeric values are rounded for display purposes. The exact values are equal to the base sampling rate divided by 2^n, where n is an integer between 0 and 13.
Round oscillator frequencies.		Round oscillator frequencies to nearest commensurable with 225 MHz.
Amplitude (FS)	0.0 to 1.0	Amplitude in units of full scale of the given AWG Output. The full scale corresponds to the Range voltage setting of the Signal Outputs.
Mode		Select between plain mode, amplitude modulation, and advanced mode.
	Plain	AWG Output goes directly to Signal Output.
	Modulation	AWG Output is multiplied with a sinusoid carrier signal. On UHFLI instruments, AWG Output 1 (2) is multiplied with oscillator signal of demodulator 4 (8). On UHFQA instruments, AWG Output 1 (2) is multiplied with the in-phase (quadrature) signal of the internal oscillator represented in the In/Out tab.
	Advanced	Output of AWG channel 1 (2) modulates demodulators 1-4 (5-8) with independent envelopes. Option not supported on UHFQA instruments.
Status		Display compiler errors and warnings.
Compile Status	grey/green/ yellow/red	Sequence program compilation status. Grey: No compilation started yet. Green: Compilation successful. Yellow: Compiler warnings (see status field). Red: Compilation failed (see status field).
Upload Progress	0% to 100%	The percentage of the sequencer program already uploaded to the device.
Upload Status	grey/yellow/ green	Indicates the upload status of the compiled AWG sequence. Grey: Nothing has been uploaded. Yellow: Upload in progress. Green: Compiled sequence has been uploaded.

Control/Tool	Option/ Range	Description
Register selector		Select the number of the user register value to be edited.
Register	0 to 2^32	Integer user register value. The sequencer has reading and writing access to the user register values during run time.
Input File		External source code file to be compiled.
Example File		Load pre-installed example sequence program.
New	New	Create a new sequence program.
Revert	Revert	Undo the changes made to the current program and go back to the contents of the original file.
Save (Ctrl+S)	Save	Compile and save the current program displayed in the Sequence Editor. Overwrites the original file.
Save as... (Ctrl+Shift+S)	Save as...	Compile and save the current program displayed in the Sequence Editor under a new name.
Automatic upload	ON / OFF	If enabled, the sequence program is automatically uploaded to the device after clicking Save and if the compilation was successful.
To Device	To Device	Sequence program will be compiled and, if the compilation was successful, uploaded to the device.
Multi-Device	ON / OFF	Compile the program for use with multiple devices. If enabled, the program will be compiled for and uploaded to the devices currently synchronized in the Multi-Device Sync tab.
Sync Status	grey/green/ yellow	Sequence program synchronization status. Grey: No program loaded on device. Green: Program in sync with device. Yellow: Sequence program in editor differs from the one running on the device.

Table 5.46: AWG tab: Waveform sub-tab

Control/ Tool	Option/ Range	Description
Wave Selection		Select wave for display in the waveform viewer. If greyed out, the corresponding wave is too long for display.
Waveforms		Lists all waveforms used by the current sequence program.
Mem Usage (%)	0 to 100	Amount of the used waveform data relative to the device cache memory. The cache memory provides space for 32 kSa of waveform data. Mem Usage > 100% means that waveforms must be loaded from the main memory (128 MSa per channel) during playback, which can lead to delays.

Table 5.47: AWG tab: Trigger sub-tab

Control/ Tool	Option/Range	Description
Force	Force	Enforce a trigger event.
Trigger State	grey/green	State of the Trigger. Grey: No trigger detected. Green: Trigger detected.
Signal		Selects the analog trigger source signal. Navigate through the tree view that appears and click on the required signal.
Slope		Select the signal edge that should activate the trigger. The trigger will be level sensitive when the Level option is selected.
	Level	Level sensitive trigger.
	Rise	Rising edge trigger.
	Fall	Falling edge trigger.

Control/ Tool	Option/Range	Description
	Both	Rising or falling edge trigger.
Level (V)	numeric value	Defines the analog trigger level.
Hysteresis Mode		Selects the mode to define the hysteresis size. The relative mode will work best over the full input range as long as the analog input signal does not suffer from excessive noise.
	Hysteresis (V)	Selects absolute hysteresis.
	Hysteresis (%)	Selects a hysteresis relative to the adjusted full scale signal input range.
Hysteresis (V)	trigger signal range (positive values only)	Defines the voltage the source signal must deviate from the trigger level before the trigger is rearmed again. Set to 0 to turn it off. The sign is defined by the Edge setting.
Hysteresis (%)	numeric percentage value (positive values only)	Hysteresis relative to the adjusted full scale signal input range. A hysteresis value larger than 100% is allowed.
Gating	Trigger In 4	Select the signal source used for trigger gating if gating is enabled.
	Trigger In 3	
Gating enable	ON / OFF	If enabled the trigger will be gated by the trigger gating input signal.
Auxiliary Trigger State	grey/green	State of the Auxiliary Trigger. Grey: No trigger detected. Green: Trigger detected.
Signal		Selects the digital trigger source signal.
DIO/Zsync Trigger state	grey/green	Indicates that triggers are generated from the DIO or ZSync interface to the AWG.
Strobe Index	16 to 31	Selects the index n of the DIO interface bit (notation DIO[n] in the Specification chapter of the User Manual) to be used as a STROBE signal input in connection with the waitDIOTrigger sequencer instruction.
Strobe Slope		Select the signal edge that activates the STROBE trigger in connection with the waitDIOTrigger sequencer instruction.
	None	Off
	Rise	Rising edge trigger
	Fall	Falling edge trigger
	Both	Rising or falling edge trigger
Valid Index	16 to 31	Selects the index n of the DIO interface bit (notation DIO[n] in the Specification chapter of the User Manual) to be used as a VALID signal input, i.e. a qualifier indicating that a valid codeword is available on the DIO interface.
Valid Polarity		Polarity of the VALID bit that indicates that a codeword is available on the DIO interface.
	None	VALID bit is ignored.
	Low	VALID bit must be logical low.
	High	VALID bit must be logical high.
	Both	VALID bit may be logical high or logical low.

Table 5.48: AWG tab: Advanced sub-tab

Control/ Tool	Option/ Range	Description
Sequence Editor		Display and edit the sequence program.

Control/ Tool	Option/ Range	Description
Assembly	Text display	Displays the current sequence program in compiled form. Every line corresponds to one hardware instruction.
AWG Core		Display assembly information.
Counter		Current position in the list of sequence instructions during execution.
Status	Running, Idle, Waiting	Displays the status of the sequencer on the instrument. Off: Ready, not running. Green: Running, not waiting for any trigger event. Yellow: Running, waiting for a trigger event. Red: Not ready (e.g., pending elf download, no elf downloaded)
Rerun	ON / OFF	Reruns the Sequencer program continuously. This way of looping a program results in timing jitter. For a jitter free signal implement a loop directly in the sequence program.
Mem Usage (%)	0 to 100	Size of the current sequence program relative to the device cache memory. The cache memory provides space for a maximum of 1024 instructions.
Status	grey/green/ red	Displays the status of the command table of the selected AWG Core. Grey: no table description uploaded, Green: table description successfully uploaded, Red: Error occurred during uploading of the table description.

5.15. Multi Device Sync Tab

The Multi Device Sync (MDS) tab gives access to the automatic timing synchronization of measurement data from multiple UHF instruments. This functionality and tab is available on all UHF instruments.

5.15.1. Features

- Automatic timing synchronization across instruments
- Periodic check of synchronization
- Selectable instrument subgroup
- Status display

5.15.2. Description

The Multi Device Sync tab contains the controls and status information for synchronized measurements on multiple instruments. Whenever the tab is closed or an additional one of the same type is needed, clicking the following icon will open a new instance of the tab.

Table 5.49: App icon and short description

Control/Tool	Option/Range	Description
MDS		Synchronize multiple instruments.

The Multi Device Sync tab shown in LabOne UI: Multi Device Sync tab consists of the Available Devices section, a Status section, and a wiring diagram.

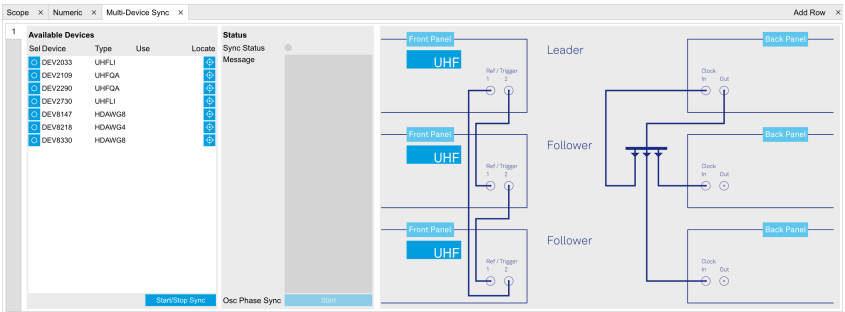


Figure 5.22: LabOne UI: Multi Device Sync tab

The Multi Device Synchronization feature provides an automated functionality to remove the clock offset of separate UHF instruments.

The first prerequisite for automatic synchronization is that all instruments are connected to the same LabOne Data Server (see Connecting to the Instrument).

To make these connections, click on Session Manager in the Config Tab to open the Device Connection dialog. Go to the Advanced view of this dialog and click on the Enable checkbox next to the corresponding entries in the Available Devices list. Once all instruments are connected, they are selectable in the Tree Selector of a newly opened Plotter tab allowing you to visualize their data simultaneously, though by default these data are not synchronized yet. The settings of multiple instruments can be accessed in parallel by opening a new Web Server session for each of them. This is done by opening a new browser tab and connecting to `localhost:8006` or `127.0.0.1:8006`, respectively, and then double-clicking the respective instrument entry in the Available Devices list. With multiple instruments connected to the same Data Server, tabs that are available for several instruments will feature a device selector as shown in Figure 5.23.

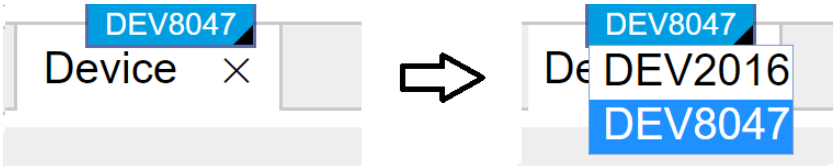


Figure 5.23: Example of the device selector for the Device tab

The second prerequisite for automatic synchronization is correct cabling of the instruments explained in the diagram in Figure 5.24. The instruments should share the same 10 MHz reference clock and communicate via the Ref / Trigger connectors for absolute timing synchronization. The 10 MHz clock signal is distributed in a star arrangement, where the signal of an external clock generator is sent to the Clk 10 MHz In connector of all instruments. On all instruments, the Clock Source in the Device Tab needs to be set to Clk 10 MHz rather than Internal. The bidirectional Ref / Trigger connectors are to be connected in a loop arrangement, where Ref / Trigger 1 of one instrument is connected to Ref / Trigger 2 of the next instrument, and so forth, until the loop is closed.

Note

Alternatively to using an external 10 MHz clock source, it's possible to distribute the Clk 10 MHz Out signal from the leading instrument to the Clk 10 MHz In connector of all following instruments and the leading instrument using a 1-to-N power divider and cables of equal length. For a large number of instruments, a clock distribution amplifier rather than a power divider is required.

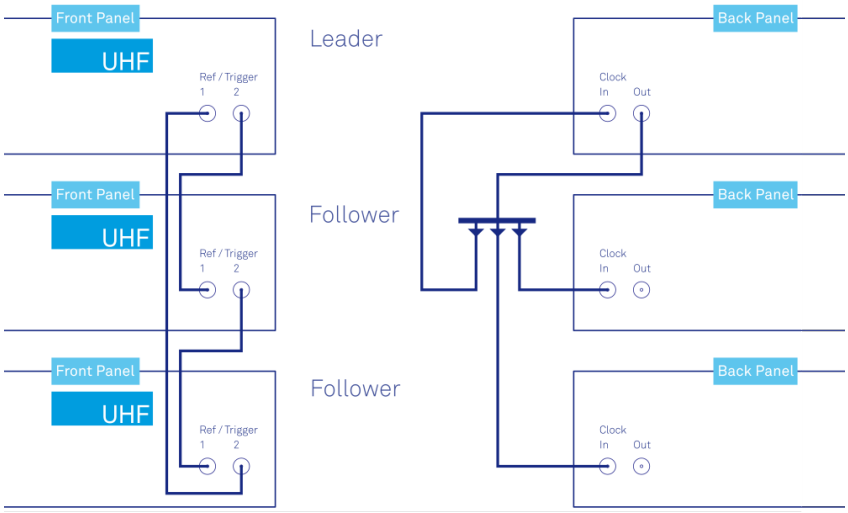


Figure 5.24: Cabling for automatic synchronization of multiple UHF instruments

Once the cabling and the connectivity is set up correctly, automatic synchronization is started in the Multi Device Sync tab by checking the Enable button on the instruments in the Available Devices list, and then clicking on Start/Stop Sync. The sequence assignment of the instruments (Leader, Follower 1, Follower 2,...) can be defined by the order in which the Enable button is clicked. This assignment has to agree with the way the cabling is made. The sequence is also relevant for addressing the instruments in an AWG sequence program. The Message display on the right will then report on the progress, and the Sync Status LED will turn green if the synchronization was successful. In that

case, visualizing a time-dependent measurement of multiple instruments in the Plotter will demonstrate the timing synchronization.

5.15.3. Functional Elements

Table 5.50: Multi Device Sync tab

Control/Tool	Option/ Range	Description
Start Sync	Start/Stop Sync	Start the automatic synchronization of the selected devices.
Sync Status		Indicates the status of the synchronization within this group. Green: synchronization successful. Yellow: synchronization in progress. Red: error (see message).
Message		Displays a status message of the synchronization group.
Cabling		This image shows how to connect the devices for device synchronization.
Phase Synchronization		Reset phases of all oscillators on all synchronized devices.
Identify Device		Make device's front LED blink

5.16. ZI Labs Tab

The ZI Labs tab contains experimental LabOne functionalities added by the ZI development team. The settings found here are often relevant to special applications, but have not yet found their definitive place in one of the other LabOne tabs. Naturally this tab is subject to frequent changes, and the documentation of the individual features would go beyond the scope of this user manual. Clicking the following icon will open a new instance of the tab.

Table 5.51: App Icon and short description

Control/Tool	Option/Range	Description
ZI Labs		Experimental settings and controls.

5.17. Upgrade Tab

The Upgrade tab serves as a source of information about the possible upgrade options for the instrument in use. The tab has no functional purpose but provides the user with a quick link to further information about the upgrade options online.

6. Specifications

Important

Unless otherwise stated, all specifications apply after 30 minutes of instrument warm-up.

Important

An internal calibration is performed 10 minutes after powering the instrument. This internal calibration is essential to achieve the specifications of the system. Further it is required to perform the internal calibration after 7 days of instrument use. This automatic calibration is turned on by default and can be configured in the Device tab.

Important

Important changes in the specification parameters are explicitly mentioned in the revision history of this document.

6.1. General Specifications

Table 6.1: General and storage

Parameter	min	typ	max
storage temperature	–25 °C	-	65 °C
storage relative humidity (non-condensing)	-	-	95%
operating temperature	5 °C	-	40 °C
operating relative humidity (non-condensing)	-	-	90%
specification temperature	18 °C	-	28 °C
power consumption	-	-	150 W
operating environment	IEC61010, indoor location, installation category II, pollution degree 2		
operating altitude	up to 2000 meters		
power inlet fuses	250 V, 2 A, fast acting, 5 x 20 mm		
power supply AC line	100-240 V (±10%), 50/60 Hz		
dimensions with handles and feet	45.0 x 34.5 x 10.0 cm, 17.7 x 13.6 x 3.9 inch, 19 inch rack compatible		
weight	6.4 kg		
recommended calibration interval	2 years		

Table 6.2: Maximum ratings

Parameter	min	typ	max
damage threshold Signal Input 1 and 2	–5 V	-	+5 V
damage threshold Signal Output 1 and 2	–2.5 V	-	+2.5 V
damage threshold Ref / Trigger 1 and 2	–6 V	-	+6 V
damage threshold Trigger Out 1 and 2	–1 V	-	+6 V

Parameter	min	typ	max
damage threshold Trigger In 1 and 2	−6 V	-	+6 V
damage threshold Aux Output 1, 2, 3, 4	−12 V	-	+12 V
damage threshold Aux In 1 and 2	−12 V	-	+12 V
damage threshold DIO (digital I/O)	−1 V	-	+6 V
damage threshold Clk In and Clk Out	−5 V	-	+5 V

Table 6.3: Host computer requirements

Parameter	Description
supported Windows operating systems	Windows 10, 11 on x86-64
supported macOS operating systems	macOS 10.11+ on x86-64 and ARMv8
supported Linux distributions	GNU/Linux (Ubuntu 14.04+, CentOS 7+, Debian 8+) on x86-64 and ARMv8
supported processors	x86-64 (Intel, AMD), ARMv8 (e.g., Raspberry Pi 4 and newer, Apple M-series)

6.2. Analog Interface Specifications

Table 6.4: UHF signal inputs

Parameter	Conditions	min	typ	max
connectors	-	BNC, front panel single-ended		
input impedance	low value	-	50 Ω	-
	high value	-	1 MΩ // 16 pF	-
input frequency range	50 Ω termination	DC	-	600 MHz
input frequency range	1 MΩ termination	DC	-	100 MHz
input A/D conversion	-	12 bit, 1.8 GSa/s		
input noise amplitude	> 100 kHz, 10 mV range, 50 Ω termination	-	4 nV/√Hz	-
input bias current	50 Ω termination	-	10 µA	-
	1 MΩ termination	-	-	1 nA
input full range sensitivity (10 V lock-in amplifier output)	-	1 nV	-	1.5 V
input AC ranges	-	10 mV	-	1.5 V
input range (AC + common mode)	DC coupling	−1.5 V	-	+1.5 V
	AC coupling	−3.5 V	-	+3.5 V
AC coupling cutoff frequency	50Ω termination	-	320 kHz	-
	1 MΩ termination	-	80 Hz	-
input amplitude accuracy	< 100 MHz	-	3 %	-
	> 100 MHz	-	10 %	-
input amplitude stability	-	-	0.1 %/°C	-

Parameter	Conditions	min	typ	max
input offset amplitude	with respect to range	-	-	5%
input harmonic distortion (HD2/HD3)	1 Vpp, 50 Ω termination, 10 minutes after manual input calibration < 1 MHz	-	–75 dB	-
	< 10 MHz	-	–70 dB	-
	< 100 MHz	-	–60 dB	-
	> 100 MHz	-	–50 dB	-
dynamic reserve		-	90 dB	100 dB

Table 6.5: UHF signal outputs

Parameter	Conditions	min	typ	max
connectors	-	BNC, front panel single-ended		
output impedance	-	-	50 Ω	-
output frequency range	-	DC	-	600 MHz
output rise time / fall time	10% to 90%	-	750 ps	-
output frequency resolution	-	-	6 µHz	-
output phase range	-	–180 °	-	180 °
output phase resolution	-	-	1.0 µ°	-
output D/A conversion	-	14 bit, 1.8 GSa/s		
output amplitude ranges	-	±150 mV, ±1.5 V		
output DC offset range	-	±150 mV or ±1.5 V, equal to the set output amplitude range		
output power	-	-	-	7.5 dBm
output amplitude accuracy	< 100 MHz	-	2%	-
	> 100 MHz	-	5%	-
output harmonic distortion (HD2/HD3)	1 Vpp, 50 Ω termination, < 1 MHz	-	–70 dB	-
	< 10 MHz	-	–70 dB	-
	< 100 MHz	-	–55 dB	-
	> 100 MHz	-	–42 dB	-
output noise amplitude	> 100 kHz	-	25 nV/√Hz	-
output phase noise	10 MHz, BW = 0.67 Hz, offset 100 Hz	-	–120 dBc/Hz	-
	10 MHz, BW = 0.67 Hz, offset 1 kHz	-	–130 dBc/Hz	-
output random jitter (RMS)	100 MHz, 6 dBm sine	-	4.5 ps	-
output offset amplitude	-	–5 mV	-	5 mV
output drive current	-	-	-	100 mA

Table 6.6: Trigger inputs and outputs

Parameter	Conditions	min	typ	max
connectors	-	BNC, front panel bidirectional SMA, back panel input SMA, back panel output		
input impedance (front and back panel)	low value	-	50 Ω	-

Parameter	Conditions	min	typ	max
	high value	-	1 kΩ	-
input level at Trigger (front panel) and Trigger In (back panel)	low input impedance	–2.5 V	-	+2.5 V
	high input impedance	–5 V	-	+5 V
output impedance (front and back panel)	-	-	50 Ω	-
output level (front and back panel)	-	-	-	3.3 V TTL
input trigger hysteresis	-	-	100 mV	-

Note

The UHF Instrument uses the same connectors for reference and trigger signals. This applies to input signals as well as output signals. Overall, the instrument features 2 output, 2 input, and 2 bidirectional connectors for reference and triggering purposes.

Table 6.7: Qubit Measurement Unit

Parameter	Details	min	typ	max
Deskew matrix	size	2 × 2		matrix element range
		–2	-	+2
	matrix element resolution	-	-	100e-6
	sampling rate	-	1.8 GSa/s	-
Input Monitor	number of scope channels	2		
	sampling rate	-	1.8 GSa/s	-
	memory	-	2 × 4 kSa	-
	averages	1	-	2^{15}
Integration units	number of dual-channel integrators	10		
	dual-channel integrator memory	-	2 × 4 kSa	-
	sampling rate	-	1.8 GSa/s	-
	trigger delay adjustment range	0 Sa	-	1020 Sa
	trigger delay resolution	-	4 Sa	-
Rotation	number of channels	10		
	data source	output of matched filters		
Crosstalk Suppression matrix	size	10 × 10		
	matrix element range	–1	-	+1
	matrix element resolution	-	-	20e-6
Signal Correlation units	number of units	10		

Parameter	Details	min	typ	max
	data source	output of Crosstalk Suppression matrix		
	operation	$V_{I,q} \times V_{I,q'}$ for 2 outputs of Crosstalk Suppression matrix q and q'		
Threshold units	number of units	10		
	data source	output of Crosstalk Suppression matrix		
	threshold range	–8,192 V	-	+8,192 V
	threshold resolution	-	-	1e-3 V
State Correlation units	number of units	10		
	data source	output of Threshold units		
	operation	$s_q \times s_{q'}$ for 2 outputs of Threshold units q and q'		
Result Logger	number of channels	10		
	data sources	output of Rotation, Crosstalk Suppression, Signal Correlation, Threshold, State Correlation, Statistics		
	memory	-	1 MSa	-
	averages	1	-	2^{17}
Statistics units	number of units	10		
	data source	outputs of Threshold units		
	operations	count number of flips in bit pattern, count number of logical '1' in bit pattern, count number of state errors based on multi-qubit state map		
Readout processing latency (delay between the end of a readout pulse at the Signal Inputs and the QA Result Trigger on any Trigger output)	Deskew, Rotation, and Crosstalk units bypassed	-	140 ns	-
	Deskew, Rotation, and Crosstalk units enabled	-	400 ns	-

Table 6.8: Auxiliary Inputs and Outputs

Parameter	Details	min	typ	max
auxiliary output	connectors	BNC, 4 outputs on front-panel		
	sampling	28 MSa/s, 16-bit		
	bandwidth	-	-	7 MHz
	impedance	-	50 Ω	-
	amplitude	–10 V	-	10 V
	resolution	0.3 mV	-	-
	drive current	-	-	100 mA
auxiliary input	connectors	SMA, 2 inputs on back-panel		
	sampling	400 kSa/s, 16-bit		
	bandwidth	-	-	100 kHz
	impedance	-	1 MΩ	-
	amplitude	–10 V	-	10 V
	resolution	0.3 mV	-	-

Table 6.9: Oscillator and clocks

Parameter	Details	min	typ	max
internal clock (ovenized crystal)	initial accuracy	-	±0.5 ppm	±1 ppm
	long term accuracy / aging	-	-	±0.4 ppm/ year
	short term stability (1 s)	0.00005 ppm	-	-
	short term stability (100 s)	0.0005 ppm	-	-
	temperature coefficient (23° ± 5°)	-	-	±0.03 ppm/°
	phase noise (at 100 Hz)	-	–130 dBc/Hz	-
	phase noise (at 1 kHz)	-	–140 dBc/Hz	-
	warm-up time	-	-	60 s
UHF-RUB Rubidium clock (option)	initial accuracy at 25°	-	-	±0.0005 ppm
	long term accuracy / aging	-	- a	±5e-6 ppm/ day ±0.0005 ppm/ year
	short term stability, AVAR (1 s)	0.00008 ppm	-	-
	short term stability, AVAR (100 s)	0.000008 ppm	-	-
	temperature coefficient (25° ± 25°)	-	-	±0.0005 ppm/ °
	phase noise (at 100 Hz)	-	-	-
	phase noise (at 1 kHz)	-	–140 dBc/Hz	-
	warm-up time	-	-	300 s @ 25°C
clock input	connector 3.+	SMA, on back-panel		
	impedance	-	50 Ω	-
	amplitude	200 mV	320 mV	1 V
	frequency	9.98 MHz	10 MHz	10.02 MHz
clock output	connector 3.+	SMA, on back-panel		
	impedance	-	50 Ω	-
	amplitude, 50 Ω	250 mV	500 mV	1 V
	frequency	-	10 MHz	-

6.3. Digital Interface Specifications

Table 6.10: Digital interfaces

Parameter	Description
host computer connection	USB 2.0 high-speed, 480 Mbit/s
	1GbE, LAN / Ethernet, 1 Gbit/s
DIO port	4 x 8 bit, general purpose digital input/output port, 5 V TTL specification
ZCtrl peripheral port	2 connectors for ZI proprietary bus to control external peripherals

6.3.1. DIO Port

The DIO port is a VHDCI 68 pin connector as introduced by the SPI-3 document of the SCSI-3 specification. It is a female connector that requires a 32 mm wide male connector. The DIO port features 32 bits that can be configured byte-wise as inputs or outputs.

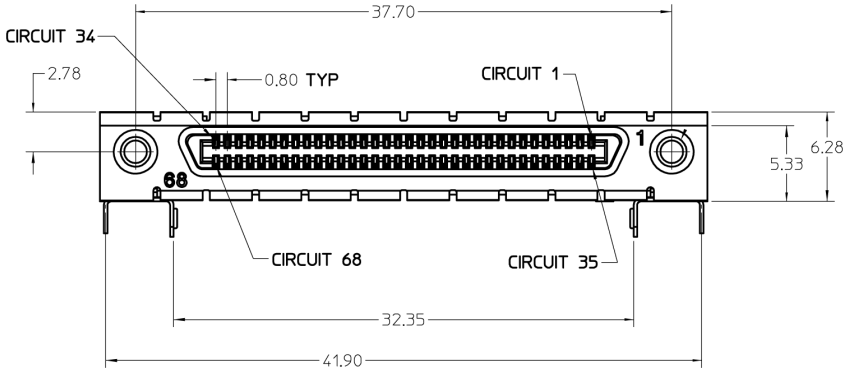


Figure 6.1: DIO HD 68 pin connector

Table 6.11: Electrical Specifications

Parameter	Details	Min	Typ	Max
output series termination	DO	low impedance (CMOS output)		
output series termination	DOL	33 Ω		
input termination	DI, CLKI	high impedance (CMOS input)		
high-level input voltage VIH	DI, CLKI	2.0 V	-	-
low-level input voltage VIL	DI, CLKI	-	-	0.8 V
high-level output voltage VOH	DO, at IOH < 24 mA	4.2 V	-	-
high-level output voltage VOH	DOL, at IOH < 32 mA	3.8 V	-	-
low-level output voltage VOL	DO, at IOL < 24 mA	-	-	0.55 V
low-level output voltage VOL	DOL, at IOL < 32 mA	-	-	0.55 V
high-level output current IOH (sourcing)	DO	-	-	24 mA
high-level output current IOH (sourcing)	DOL	-	-	32 mA
low-level output current IOL (sinking)	DO	-	-	24 mA
low-level output current IOL (sinking)	DOL	-	-	32 mA

Table 6.12: DIO pin assignment

Pin	Name	Description	Range specification
68	CLKI	clock input, used to latch signals at the digital input ports - can also be used to retrieve digital signals from the output port using an external sampling clock	5 V CMOS/TTL
67	DOL	DIO output latch, 56.25 MHz clock signal, the digital outputs are synchronized to the falling edge of this signal	5 V CMOS
66-59	DI[31:24]	digital input or output (set by user)	output CMOS 5 V, input is CMOS/TTL
58-51	DIO[23:16]	digital input or output (set by user)	output CMOS 5 V, input is CMOS/TTL
50-43	DIO[15:8]	digital input or output (set by user)	output CMOS 5 V, input is CMOS/TTL
42-35	DIO[7:0]	digital input or output (set by user)	output CMOS 5 V, input is CMOS/TTL
34-1	GND	digital ground	-

The table below shows the mapping between DIO pins and input/output signals available when using the DIO connector to output qubit state measurement results. The direction is as seen from the UHFQA instrument. In order to use these signals, the Digital I/O Mode and Drive setting have to be chosen accordingly. Please refer to DIO Tab for further documentation.

Table 6.13: DIO QA signal assignment

DIOLink signal	DIO pin	Direction	Description
VALID	DIO[0]	OUT	Codeword valid indicator
CW	DIO[10:1]	OUT	Quantized result of each of the 10 readout paths
reserved	DIO[13:11]	OUT	Reserved for future use
VALID	DIO[14]	OUT	Codeword valid indicator (same as DIO[0])
STROBE	DIO[15]	OUT	Toggle signal for timing alignment, 25 MHz

The figure below shows the architecture of the DIO input/output. The DIO port features 32 bits that can be configured byte-wise as inputs or outputs by means of a drive signal. The digital output data is latched synchronously with the falling edge of the internal clock, which is running at 56.25 MHz. The internal sampling clock is available at the DOL pin of the DIO connector. Digital input data can either be sampled by the internal clock or by an external clock provided through the CLKI pin. A decimated version of the input clock is used to sample the input data. The Decimation unit counts the clocks to decimation and then latches the input data. The default decimation is 5625000, corresponding to a digital input sampling rate of 1 sample per second.

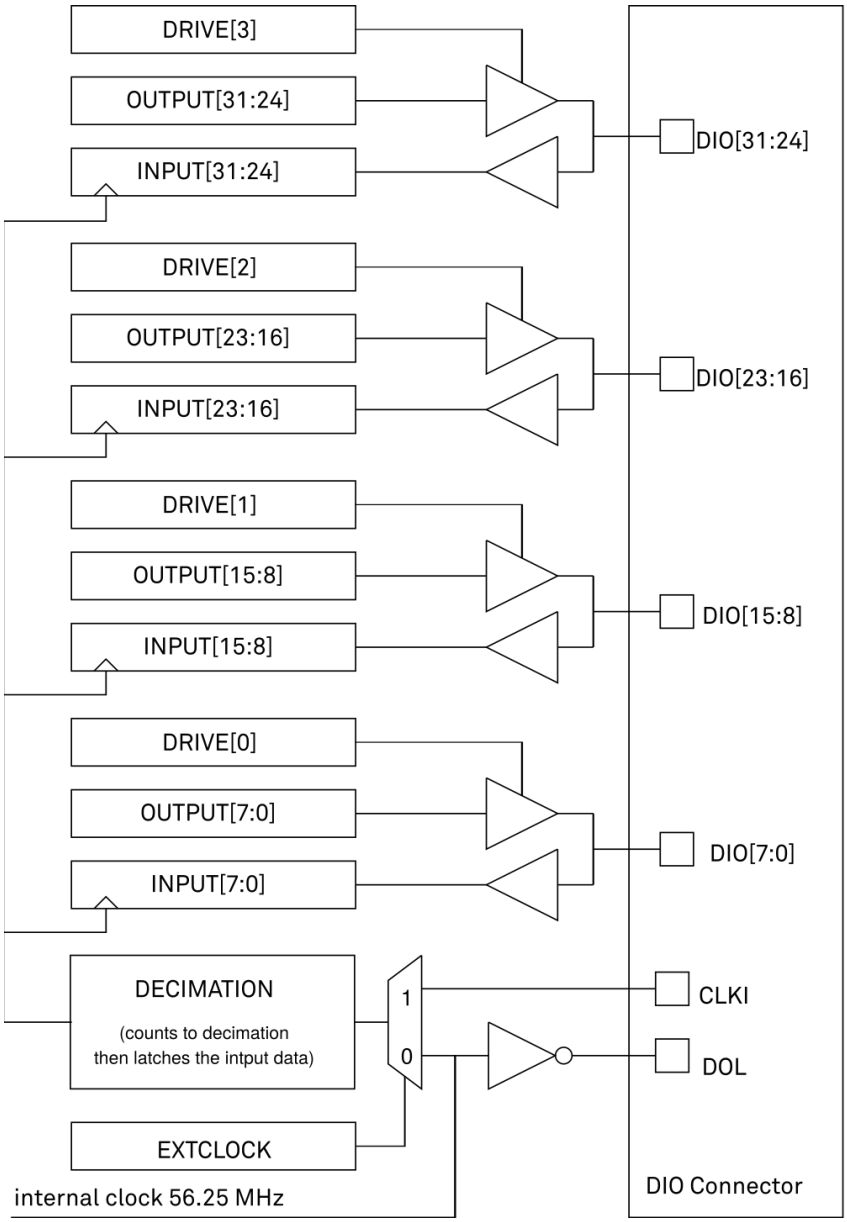


Figure 6.2: DIO input/output architecture

6.3.2. DIOLink Interface

Introduction

When operating the Zurich Instruments HDAWG and UHFQA in a multi-instrument setup for quantum computing, the DIO interface can be used to communicate data within the system. There are two modes supported:

1. Operation within a Zurich Instruments Quantum Computing Control System containing a PQSC Programmable Quantum System Controller
2. Operation in a control system controlled by a third-party central controller

In the first case, the DIO interface is used to embed the UHFQA into the QCCS by a connection to a HDAWG. This case is described in the PQSC User Manual. The second mode makes use of the DIOLink interface protocol documented in the following.

The DIOLink provides a digital interface to Zurich Instruments HDAWG and UHFQA instruments. It enables the user to trigger the AWGs in the respective instruments using a digital codeword sent from a central control unit. The codeword may be used for playing back a waveform from a table with low latency, triggering a qubit measurement, etc. In case of the UHFQA instrument, the interface may also be used for communicating the results of a measurement back to the controlling unit.

The following figure illustrates how such a measurement setup could be constructed. A host PC is responsible for controlling all the instruments in the setup. The instruments are synchronized by a shared 10 MHz reference clock. A central control unit controls the operation of the HDAWG and UHFQA instruments during experiments using the DIOLink interface. For the HDAWG, the DIOLink is unidirectional as there is rarely a need to communicate information back to the central control unit. In contrast, the DIOLink of the UHFQA is bidirectional such that measurement results can be reported back and acted upon.

For the purposes of the DIOLink, the DIO output latch (DOL) signal of the DIO connector can be ignored. The interface uses TTL signaling, which means it is sufficient to use 3.3 V for both HDAWG and UHFQA instruments in the direction from control unit to instrument. It is important to ensure that the DIOLink interface connected to a UHFQA instrument on the side of the central control unit is 5 V tolerant.

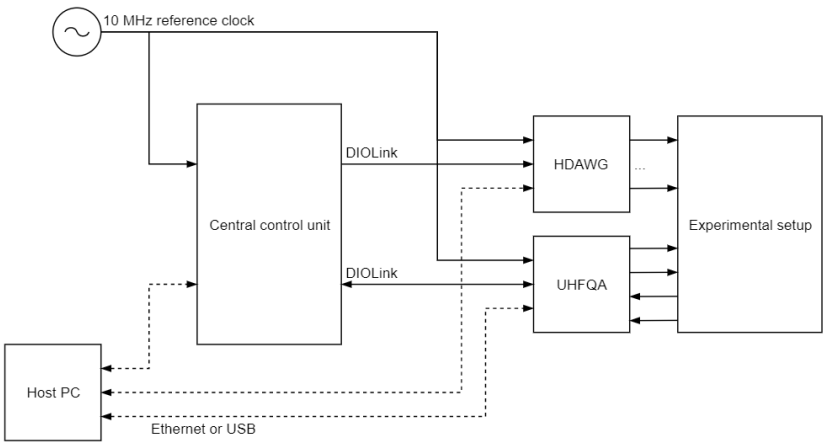


Figure 6.3: DIO Link connections between HDAWG, UHFQA, and central control unit in an experimental setup

Signal Protocol

The DIOLink interface makes use of the signals shown in the following table. The signals are all transmitted from the sender to the receiver of the link. The length of the codeword varies between instruments and can at least to some extend be configured by the user.

Table 6.14: DIOLink signals

Name	Description
STROBE	Toggle signal for timing alignment. The signal must toggle at a fixed rate, which defines the time grid on which codewords are transmitted. The signal is typically a 25 MHz square wave derived from the 10 MHz reference clock.
VALID	Codeword valid indicator. Must be asserted whenever a valid codeword is present on the DIOLink.
CW[N-1:0]	Codeword. The digital multi-bit codeword to be transmitted to the receiver.

The maximum rate at which codewords can be transmitted to the instruments is 50 MHz. It is allowed to use a lower rate than 50 MHz, but the rate must always be an integer division of 50 MHz. Therefore, 25 MHz and 12.5 MHz would be supported, but 40 MHz would not. The following figure shows a timing diagram of the signaling protocol on the interface.

The timing on the interface is described in more detail in the following table.

Table 6.15: DIOLink signals

Name	Range	Description
$T_{ref\text{-}clk}$	N/A	Delay from the reference clock to the clock that drives the DIOLink interface on the transmitter side. There are no specific requirements for this delay. However, it must always be the same delay every time the transmitter is activated (e.g. after powering up the instruments).
$T_{strb\text{-}vld}$	± 3.3 ns	Delay, or skew, between the STROBE and the VALID signals.
$T_{strb\text{-}cw}$	± 3.3 ns	Delay, or skew, between the STROBE and every bit of the CW signal.

Signal Assignment HDAWG

The DIOLink signal assignment on the DIO connector to an HDAWG instrument is freely configurable by the user. This is done using the corresponding settings in the AWG Sequencer tab in the LabOne User Interface. The DIO pin for the STROBE and VALID signals are selected using Strobe Index and Valid Index settings. The codeword is specified using the Codeword Mask and Codeword Shift settings. These two settings allow the user to select any range up to 10 bits wide to use as an index for playing back waveforms from a table using the `playWaveDIO` sequencer instruction.

Signal Assignment UHFQA

In case of the UHFQA, the assignment of DIOLink signals to DIO pins is static and specified in the following table for those pins that communicate data from the UHFQA to the central control unit. As such, the direction is as seen from the UHFQA instrument.

Table 6.16: Signal assignment UHFQA

DIOLink signal	DIO pin	Direction	Description
VALID	DIO[0]	OUT	Codeword valid indicator
CW	DIO[10:1]	OUT	Quantized result of each of the 10 readout paths
reserved	DIO[13:11]	OUT	Reserved for future use
VALID	DIO[14]	OUT	Codeword valid indicator (same as DIO[0])
STROBE	DIO[15]	OUT	Toggle signal for timing alignment, 25 MHz

6.3.3. ZCtrl Peripheral Port

The ZCtrl port serves to power and communicate to external equipment, such as pre-amplifiers: the port provides a floating power supply with ±14.5 V and 100 mA per port. After Instrument power-on,

the port is not active and must be switched on in order to be used. Two activation methods are supported:

- Manual switch in the user interface
- Manual switch by shorting the ZCtrl_Detect and Device_Ground - these pins should be floating against ZCtrl_GND and ZCtrl_PWR

Th ZCtrl port can be connected with an RJ45 connector, therefore non-crossed Ethernet cables can be used for convenient interfacing.

Warning

Connection to a Ethernet might damage the UHF Instrument.

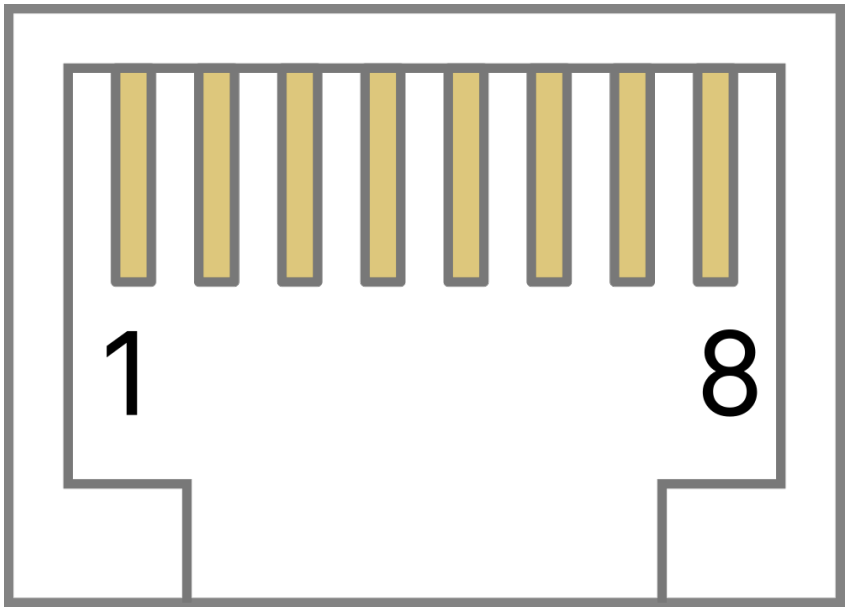


Figure 6.4: The pinout of the ZCtrl port

Table 6.17: DIO port pin assignment

Pin	Name	Description	Range specification
1	ZCtrl_Power+	power pin, for external use	14.5 V, 100 mA
2	ZCtrl_Detect	connection detection	-
3	Device_Ground	ground of UHF Instrument, connected to earth pin	-
4	ZCtrl_Power–	power pin, for external use	–14.5 V, 100 mA
5	ZCtrl_D	proprietary function	-
6	ZCtrl_C	proprietary function	-
7	ZCtrl_GND	floating input	-
8	ZCtrl_GND	reference ground pin for ZCtrl_Power+ and ZCtrl_Power–	-

6.4. Performance Diagrams

Many of the parameters mentioned in Analog Interface Specifications are valid without specific conditions. Other parameters instead are typical specifications, because they depend on several parameters, such as the input range setting, the input termination and/or the frequency. This section completes the previous chapters with detailed performance diagrams in order to support the validation of applications.

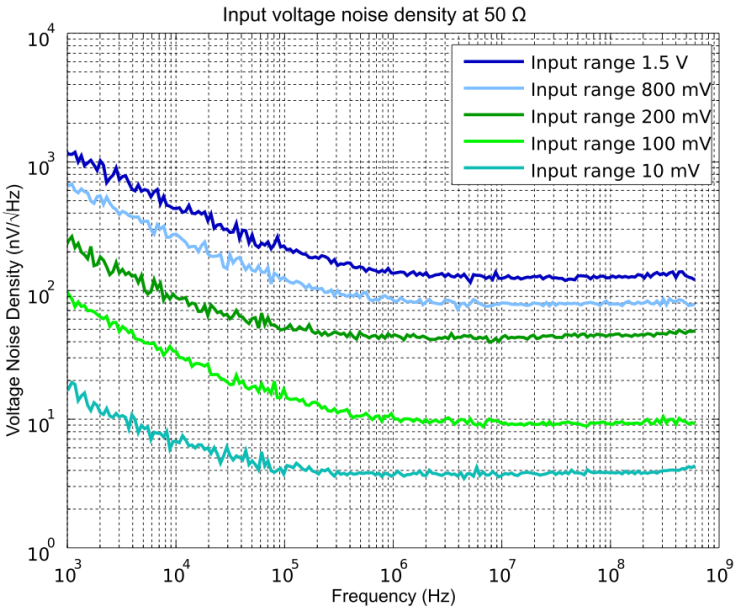


Figure 6.5: UHF Series input noise with 50Ω input impedance

Input noise amplitude depends on several parameters, and in particular on the frequency and the input range setting. The input noise is lower for smaller input ranges, and it is recommended to use small ranges especially for noise measurements. Only the noise with DC input coupling is shown here as the input noise with AC coupling is the same, as long as the frequency is above the AC cutoff frequency (see Table 6.4).

The input noise does depend on the input impedance setting, which can be 50 Ω or 1 MΩ. The performance diagrams for 50 Ω and 1 MΩ input impedance are shown in Figure 6.5 and in Figure 6.6, respectively. For both, the corner frequency of the 1/f noise is in the range of 100 kHz. For 50 Ω input impedance, the white noise floor is around 4 nV/√Hz for the smallest input range. For 1 MΩ input impedance, the white noise floor is below 8 nV/√Hz for the smallest input range.

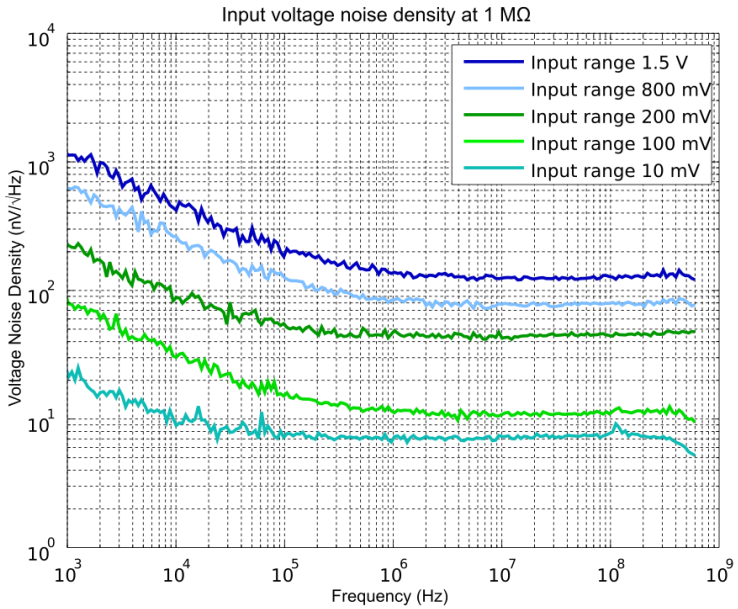


Figure 6.6: UHF Series input noise with 1 MΩ input impedance

Figure 6.7 shows a typical SSB phase noise measured at the signal output. For this measurement, the UHF instrument was connected to a phase noise analyzer and the signal output amplitude was set to 1.5 V. The phase noise at 10 MHz at 10 kHz offset is around -139 dBc/Hz. The phase noise at 100 MHz at 10 kHz offset is around -118 dBc/Hz.

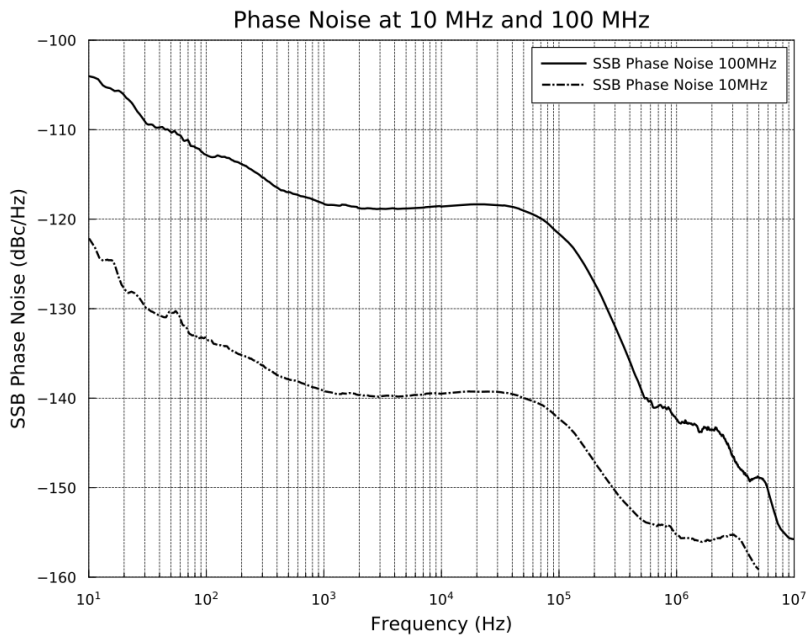


Figure 6.7: UHF phase noise

6.5. Clock 10 MHz

A 10 MHz clock input and output is provided for synchronization with other instruments. The figure explains the internal routing of the different clock signals. An internal clock generation unit receives a 10 MHz clock reference and generates all necessary internal sampling clocks. The clock reference either comes from the internal quartz/Rubidium oscillator or from an external clock source connected to the Clock 10 MHz In connector. The user can define if the clock is taken from the internal or external source. The Clock 10 MHz Out connector always provides the 10 MHz clock of the internal quartz/Rubidium oscillator.

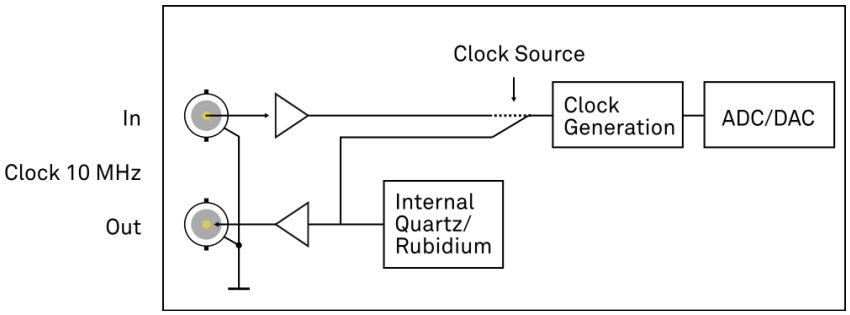


Figure 6.8: Clock routing

6.6. Device Self Calibration Procedure

The instrument requires a self calibration after a short warm-up period to ensure operation according to specifications. During this self calibration process, components of the sensitive analog front-end are calibrated to account for temperature variations and drift. The self calibration is not to be confused with the Instrument Calibration service by Zurich Instruments. The latter is performed at the manufacturer site. The self calibration lasts about one second and only applies a fine-tuning.

The first self calibration after warm-up is executed automatically. Any further self calibration needs to be manually executed by the user. The self calibration process can be executed by clicking the Run button of the Auto Calibration section in the Device tab of the user interface.

The user can disable the calibration procedure completely if necessary. This can be done by changing the Enable button of the Auto Calibration in the Device tab. If this flag is disabled, no calibration is executed after warm-up time.

The default self calibration procedure can be divided into three different states, which are also indicated by the CAL flag in the footer of the user interface. The CAL flag can be either yellow, gray/off, or red.

- Yellow: The yellow CAL flag indicates that the calibration has not been executed yet. After a warm-up and temperature settling period of approximately 16 minutes, a self calibration is executed and the CAL flag turns gray. If the self calibration is disabled, the CAL flag turns red after the warm-up period to indicate that no calibration was performed.
- Gray/off: The gray CAL flag indicates that the instrument is self calibrated. The CAL flag turns red when the temperature change is larger than a given threshold or the time since the last calibration is longer than a given time interval. The values of these thresholds are indicated in the Device tab.
- Red: The red CAL flag indicates that it is recommended to perform a self calibration. The self calibration is never executed automatically in this state. The CAL flag is red, either, when the instrument experienced a temperature change larger than a given threshold, or when the time since the last calibration is longer than a given time interval. By executing a self calibration, the CAL flag will turn gray.

7. Device Node Tree

This chapter contains reference documentation for the settings and measurement data available on UHFQA Instruments. Whilst Functional Description LabOne User Interface describes many of these settings in terms of the features available in the LabOne User Interface, this chapter describes them on the device level and provides a hierarchically organized and comprehensive list of device functionality.

Since these settings and data streams may be written and read using the LabOne APIs (Application Programming Interfaces) this chapter is of particular interest to users who would like to perform measurements programmatically via LabVIEW, Python, MATLAB, .NET or C.

Please see:

- Introduction for an introduction of how the instrument's settings and measurement data are organized hierarchically in the Data Server's so-called "Node Tree".
- Reference Node Documentation for a reference list of the settings and measurement data available on UHFQA Instruments, organized by branch in the Node Tree.

7.1. Introduction

This chapter provides an overview of how an instrument's configuration and output is organized by the Data Server.

All communication with an instrument occurs via the Data Server program the instrument is connected to (see LabOne Software Architecture for an overview of LabOne's software components). Although the instrument's settings are stored locally on the device, it is the Data Server's task to ensure it maintains the values of the current settings and makes these settings (and any subscribed data) available to all its current clients. A client may be the LabOne User Interface or a user's own program implemented using one of the LabOne Application Programming Interfaces, e.g., Python.

The instrument's settings and data are organized by the Data Server in a file-system-like hierarchical structure called the node tree. When an instrument is connected to a Data Server, its device ID becomes a top-level branch in the Data Server's node tree. The features of the instrument are organized as branches underneath the top-level device branch and the individual instrument settings are leaves of these branches.

For example, the auxiliary outputs of the instrument with device ID "dev2006" are located in the tree in the branch:

```
/dev1000/auxouts/
```

In turn, each individual auxiliary output channel has its own branch underneath the "AUXOUTS" branch.

```
/dev1000/auxouts/0/
/dev1000/auxouts/1/
/dev1000/auxouts/2/
/dev1000/auxouts/3/
```

Whilst the auxiliary outputs and other channels are labelled on the instrument's panels and the User Interface using 1-based indexing, the Data Server's node tree uses 0-based indexing. Individual settings (and data) of an auxiliary output are available as leaves underneath the corresponding channel's branch:

```
/dev1000/auxouts/0/demodselect
/dev1000/auxouts/0/limitlower
/dev1000/auxouts/0/limitupper
/dev1000/auxouts/0/offset
/dev1000/auxouts/0/outputselect
/dev1000/auxouts/0/preoffset
/dev1000/auxouts/0/scale
/dev1000/auxouts/0/value
```

These are all individual node paths in the node tree; the lowest-level nodes which represent a single instrument setting or data stream. Whether the node is an instrument setting or data-stream and which type of data it contains or provides is well-defined and documented on a per-node basis in the Reference Node Documentation section in the relevant instrument-specific user manual. The different properties and types are explained in Node Properties and Data Types .

For instrument settings, a Data Server client modifies the node's value by specifying the appropriate path and a value to the Data Server as a (path, value) pair. When an instrument's setting is changed in the LabOne User Interface, the path and the value of the node that was changed are displayed in the Status Bar in the bottom of the Window. This is described in more detail in Exploring the Node Tree.

Module Parameters

LabOne Core Modules, such as the Sweeper, also use a similar tree-like structure to organize their parameters. Please note, however, that module nodes are not visible in the Data Server's node tree; they are local to the instance of the module created in a LabOne client and are not synchronized between clients.

7.1.1. Node Properties and Data Types

A node may have one or more of the following properties:

Read	Data can be read from the node.
Write	Data can be written to the node.
Setting	The node corresponds to a writable instrument configuration. The data of these nodes are persisted in snapshots of the instrument and stored in the LabOne XML settings files.
Streaming	A node with the read attribute that provides instrument data, typically at a user-configured rate. The data is usually a more complex data type, for example demodulator data is returned as `ZIDemodSample`. A full list of streaming nodes is available in the Programming Manual in the Chapter Instrument Communication. Their availability depends on the device class (e.g. MF) and the option set installed on the device.

A node may contain data of the following types:

Integer	Integer data.
Double	Double precision floating point data.
String	A string array.
Integer (enumerated)	As for Integer, but the node only allows certain values.
Composite data type	For example, `ZIDemodSample`. These custom data types are structures whose fields contain the instrument output, a timestamp and other relevant instrument settings such as the demodulator oscillator frequency. Documentation of custom data types is available in

7.1.2. Exploring the Node Tree

In the LabOne User Interface

A convenient method to learn which node is responsible for a specific instrument setting is to check the Command Log history in the bottom of the LabOne User Interface. The command in the Status Bar gets updated every time a configuration change is made. Figure 7.1 shows how the equivalent MATLAB command is displayed after modifying the value of the auxiliary output 1's offset. The format of the LabOne UI's command history can be configured in the Config Tab (MATLAB, Python and .NET are available). The entire history generated in the current UI session can be viewed by clicking the "Show Log" button.

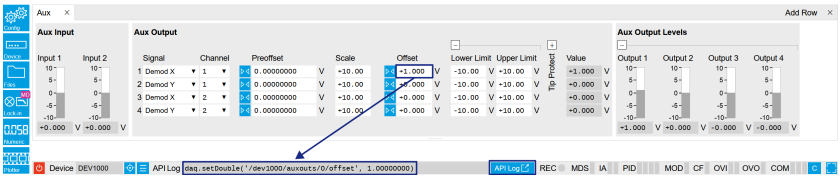


Figure 7.1: When a device's configuration is modified in the LabOne User Interface, the Status Bar displays the equivalent command to perform the same configuration via a LabOne programming interface. Here, the MATLAB code to modify auxiliary output 1's offset value is provided. When "Show Log" is clicked the entire configuration history is displayed in a new browser tab.

In a LabOne Programming Interface

A list of nodes (under a specific branch) can be requested from the Data Server in an API client using the **`listNodes`** command (MATLAB, Python, .NET) or **`ziAPIListNodes()`** function (C API). Please see each API's command reference for more help using the **`listNodes`** command. To obtain a list of all the nodes that provide data from an instrument at a high rate, so-called streaming nodes, the **`streamingonly`** flag can be provided to **`listNodes`**. More information on data streaming and streaming nodes is available in the LabOne Programming Manual.

The detailed descriptions of nodes that is provided in Reference Node Documentation is accessible directly in the LabOne MATLAB or Python programming interfaces using the "help" command. The **`help`** command is **`daq.help(path)`** in Python and **`ziDAQ('help', path)`** in MATLAB. The command returns a description of the instrument node including access properties, data type, units and available options. The "help" command also handles wildcards to return a detailed description of all nodes matching the path. An example is provided below.

```
    daq = zhinst.core.ziDAQServer('localhost', 8004, 6)
    daq.help('/dev2006/auxouts/0/offset')
    # Out:
    # /dev1000/auxouts/0/offset#
    # Add the specified offset voltage to the signal after scaling. Auxiliary
Output
    # Value = (Signal+Preoffset)*Scale + Offset
    # Properties: Read, Write, Setting
    # Type: Double
    # Unit: V
```

7.1.3. Data Server Nodes

The Data Server has nodes in the node tree available under the top-level /ZI/ branch. These nodes give information about the version and state of the Data Server the client is connected to. For example, the nodes:

- /ZI/ABOUT/VERSION
- /ZI/ABOUT/REVISION

are read-only nodes that contain information about the release version and revision of the Data Server. The nodes under the /ZI/DEVICES/ list which devices are connected, discoverable and visible to the Data Server.

The nodes:

- /ZI/CONFIG/OPEN
- /ZI/CONFIG/PORT

are settings nodes that can be used to configure which port the Data Server listens to for incoming client connections and whether it may accept connections from clients on hosts other than the localhost.

Nodes that are of particular use to programmers are:

- /ZI/DEBUG/LOGPATH - the location of the Data Server's log in the PC's file system,
- /ZI/DEBUG/LEVEL - the current log-level of the Data Server (configurable; has the Write attribute),
- /ZI/DEBUG/LOG - the last Data Server log entries as a string array.

The Global nodes of the LabOne Data Server are listed in the Instrument Communication chapter of the LabOne Programming Manual

7.2. Reference Node Documentation

This section describes all the nodes in the data server's node tree organized by branch.

7.2.1. AUXINS

/dev..../auxins/n/averaging

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Defines the number of samples on the input to average as a power of two. Possible values are in the range [0, 16]. A value of 0 corresponds to the sampling rate of the auxiliary input's ADC.

/dev..../auxins/n/sample

Properties: Read, Stream
Type: ZIAuxInSample
Unit: V

Voltage measured at the Auxiliary Input after averaging. The data rate depends on the averaging value. Note, if the instrument has demodulator functionality, the auxiliary input values are available as fields in a demodulator sample and are aligned by timestamp with the demodulator output.

/dev..../auxins/n/values/n

Properties: Read
Type: Double
Unit: V

Voltage measured at the Auxiliary Input after averaging. The value of this node is updated at a low rate (50 Hz); the streaming node auxins/n/sample is updated at a high rate defined by the averaging.

7.2.2. AUXOUTS

/dev..../auxouts/n/demodselect

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Select the channel number of the selected signal source.

/dev..../auxouts/n/limitlower

Properties: Read, Write, Setting
Type: Double
Unit: V

Lower limit for the signal at the Auxiliary Output. A smaller value will be clipped.

/dev..../auxouts/n/limitupper

Properties: Read, Write, Setting
Type: Double
Unit: V

Upper limit for the signal at the Auxiliary Output. A larger value will be clipped.

/dev..../auxouts/n/offset

Properties: Read, Write, Setting
Type: Double
Unit: V

Add the specified offset voltage to the signal after scaling. Auxiliary Output Value = (Signal+Preoffset)*Scale + Offset

/dev..../auxouts/n/outputselect

Properties: Read, Write, Setting
Type: Integer (enumerated)
Unit: None

Select the signal source to be represented on the Auxiliary Output.

-1	"manual": Select Manual as the output option.
0	"demod_x": Select Demod X as the output option.
1	"demod_y": Select Demod Y as the output option.
2	"demod_r": Select Demod R as the output option.
3	"demod_theta": Select Demod Theta as the output option.
4	"awg": Select one of the AWG Outputs for auxiliary output when running the AWG in four-channel mode. The AWG option needs to be installed.
5	"pid": Select PID Out as the output option.
6	"boxcar": Select Boxcar as the output option.
7	"au_cartesian": Select AU Cartesian as the output option.
8	"au_polar": Select AU Polar as the output option.
9	"pid_shift": Select PID Shift as the output option.
10	"pid_error": Select PID Error as the output option.
12	"pulse_counter": Select Pulse Counter as the output option.

/dev..../auxouts/n/preoffset

Properties: Read, Write, Setting
Type: Double
Unit: Dependent

Add a pre-offset to the signal before scaling is applied. Auxiliary Output Value = (Signal+Preoffset)*Scale + Offset

/dev..../auxouts/n/scale

Properties: Read, Write, Setting
Type: Double
Unit: None

Multiplication factor to scale the signal. Auxiliary Output Value = (Signal+Preoffset)*Scale + Offset

/dev..../auxouts/n/value

Properties: Read
Type: Double
Unit: V

Voltage present on the Auxiliary Output. Auxiliary Output Value = (Signal+Preoffset)*Scale + Offset

7.2.3. AWGS

/dev..../awgs/n/auxtriggers/n/channel

Properties: Read, Write, Setting
Type: Integer (enumerated)
Unit: None

Selects the digital trigger source signal.

0	"trigin0", "trigger_input0": Trigger In 1
1	"trigin1", "trigger_input1": Trigger In 2
2	"trigin2", "trigger_input2": Trigger In 3
3	"trigin3", "trigger_input3": Trigger In 4
4	"trigout0", "trigger_output0": Trigger Out 1
5	"trigout1", "trigger_output1": Trigger Out 2
6	"trigout2", "trigger_output2": Trigger Out 3
7	"trigout3", "trigger_output3": Trigger Out 4

/dev..../awgs/n/auxtriggers/n/slope

Properties: Read, Write, Setting
Type: Integer (enumerated)
Unit: None

Select the signal edge that should activate the trigger. The trigger will be level sensitive when the Level option is selected.

0	"level_sensitive": Level sensitive trigger
1	"rising_edge": Rising edge trigger
2	"falling_edge": Falling edge trigger
3	"both_edges": Rising or falling edge trigger

/dev..../awgs/n/auxtriggers/n/state

Properties: Read
Type: Integer (64 bit)
Unit: None

State of the Auxiliary Trigger: No trigger detected/trigger detected.

/dev..../awgs/n/dio/data

Properties: Read
Type: ZIVectorData
Unit: None

A vector of 32-bit integers representing the values on the DIO interface.

/dev..../awgs/n/dio/delay/index

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Index of the bit on the DIO interface for which the delay should be changed.

/dev..../awgs/n/dio/delay/value

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Corresponding delay value to apply to the given bit of the DIO interface in units of 450 MHz clock cycles. Valid values are 0 to 3.

/dev..../awgs/n/dio/strobe/index

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Select the DIO bit to use as the STROBE signal.

/dev..../awgs/n/dio/strobe/slope

Properties: Read, Write, Setting
Type: Integer (enumerated)
Unit: None

Select the signal edge of the STROBE signal for use in timing alignment.

0	"off": Off
1	"rising_edge": Rising edge trigger
2	"falling_edge": Falling edge trigger
3	"both_edges": Rising or falling edge trigger

/dev..../awgs/n/dio/valid/index

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Select the DIO bit to use as the VALID signal to indicate a valid input is available.

/dev..../awgs/n/dio/valid/polarity

Properties: Read, Write, Setting
Type: Integer (enumerated)
Unit: None

Polarity of the VALID bit that indicates that a valid input is available.

0	"none": None: VALID bit is ignored.
1	"low": Low: VALID bit must be logical zero.
2	"high": High: VALID bit must be logical high.
3	"both": Both: VALID bit may be logical high or zero.

/dev..../awgs/n/elf/checksum

Properties: Read
Type: Integer (64 bit)
Unit: None

Checksum of the uploaded ELF file.

/dev..../awgs/n/elf/data

Properties: Write
Type: ZIVectorData
Unit: None

Accepts the data of the sequencer ELF file.

/dev..../awgs/n/elf/length

Properties: Read
Type: Integer (64 bit)
Unit: None

Length of the compiled ELF file.

/dev..../awgs/n/elf/memoryusage

Properties: Read
Type: Double
Unit: None

Size of the uploaded ELF file relative to the size of the main memory.

/dev..../awgs/n/elf/name

Properties: Read
Type: ZIVectorData
Unit: None

The name of the uploaded ELF file.

/dev..../awgs/n/elf/progress

Properties:	Read
Type:	Double
Unit:	%

The percentage of the sequencer program already uploaded to the device.

/dev..../awgs/n/enable

Properties:	Read, Write, Setting
Type:	Integer (64 bit)
Unit:	None

Activates the AWG.

/dev..../awgs/n/outputs/n/amplitude

Properties:	Read, Write, Setting
Type:	Double
Unit:	None

Amplitude in units of full scale of the given AWG Output. The full scale corresponds to the Range voltage setting of the Signal Outputs.

/dev..../awgs/n/outputs/n/enables/n

Properties:	Read
Type:	Integer (64 bit)
Unit:	None

Indicates the routing of the AWG signal (k index) to the wave output or to the digital mixer input (m index).

/dev..../awgs/n/outputs/n/mode

Properties:	Read, Write, Setting
Type:	Integer (enumerated)
Unit:	None

Select between plain mode, amplitude modulation, and advanced mode.

0	"plain": Plain: AWG Output goes directly to Signal Output.
1	"modulation": Modulation: AWG Output 1 (2) is multiplied with oscillator signal of demodulator 4 (8).
2	"advanced": Advanced: Output of AWG channel 1 (2) modulates demodulators 1-4 (5-8) with independent envelopes.

/dev..../awgs/n/ready

Properties:	Read
Type:	Integer (64 bit)
Unit:	None

AWG has a compiled wave form and is ready to be enabled.

/dev..../awgs/n/rtlogger/clear

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Clears the logger data.

/dev..../awgs/n/rtlogger/data

Properties: Read
Type: ZIVectorData
Unit: None

Vector node with the logged events.

/dev..../awgs/n/rtlogger/enable

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Activates the Real-time Logger.

/dev..../awgs/n/rtlogger/mode

Properties: Read, Write, Setting
Type: Integer (enumerated)
Unit: None

Selects the operation mode.

0	"normal": Normal: Logger starts with the AWG and overwrites old values as soon as the memory limit of 1024 entries is reached.
1	"timestamp": Timestamp-triggered: Logger starts with the AWG, waits for the first valid trigger, and only starts recording data after the time specified by the starttimestamp. Recording stops as soon as the memory limit of 1024 entries is reached.

/dev..../awgs/n/rtlogger/starttimestamp

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Timestamp at which to start logging for timestamp-triggered mode.

/dev..../awgs/n/rtlogger/status

Properties: Read
Type: Integer (enumerated)
Unit: None

Operation state.

0	"idle": Idle: Logger is not running.
1	"normal": Normal: Logger is running in normal mode.
2	"ts_wait": Wait for timestamp: Logger is in timestamp-triggered mode and waits for start timestamp.
3	"ts_active": Active: Logger is in timestamp-triggered mode and logging.
4	"ts_full": Log Full: Logger is in timestamp-triggered mode and has stopped logging because log is full.

/dev..../awgs/n/rtlogger/timebase

Properties: Read
Type: Double
Unit: s

Minimal time difference between two timestamps. The value matches the AWG sequencer execution rate.

/dev..../awgs/n/sequencer/assembly

Properties: Read
Type: ZIVectorData
Unit: None

Displays the current sequence program in compiled form. Every line corresponds to one hardware instruction.

/dev..../awgs/n/sequencer/continue

Properties: Read, Write
Type: Integer (64 bit)
Unit: None

Reserved for future use.

/dev..../awgs/n/sequencer/memoryusage

Properties: Read
Type: Double
Unit: None

Size of the current Sequencer program relative to the available instruction memory of 1 kInstructions (1'024 instructions).

/dev..../awgs/n/sequencer/next

Properties: Read, Write
Type: Integer (64 bit)
Unit: None

Reserved for future use.

/dev..../awgs/n/sequencer/pc

Properties: Read
Type: Integer (64 bit)
Unit: None

Current position in the list of sequence instructions during execution.

/dev..../awgs/n/sequencer/program

Properties: Read
Type: ZIVectorData
Unit: None

Displays the source code of the current sequence program.

/dev..../awgs/n/sequencer/status

Properties:	Read
Type:	Integer (64 bit)
Unit:	None

Status of the sequencer on the instrument. Bit 0: sequencer is running; Bit 1: reserved; Bit 2: sequencer is waiting for a trigger to arrive; Bit 3: AWG has detected an error; Bit 4: sequencer is waiting for synchronization with other channels.

/dev..../awgs/n/single

Properties:	Read, Write, Setting
Type:	Integer (64 bit)
Unit:	None

Puts the AWG into single shot mode.

/dev..../awgs/n/sweep/awgtrigs/n

Properties:	Read, Write
Type:	Double
Unit:	Dependent

Node used by the sweeper module for fast index sweeps. When selected as sweep grid the sweeper module will switch into a fast index based scan mode. This mode can be up to 1000 times faster than conventional node sweeps. The sequencer program must support this functionality. See section 'AWG Index Sweep' of the UHF user manual for more information.

/dev..../awgs/n/time

Properties:	Read, Write, Setting
Type:	Integer (enumerated)
Unit:	None

AWG sampling rate. The numeric values are rounded for display purposes. The exact values are equal to the base sampling rate (1.8 GHz) divided by 2^n, where n is the node value. This value is used by default and can be overridden in the Sequence program.

0	"1.80_GHz": 1.80 GHz
1	"900_MHz": 900 MHz
2	"450_MHz": 450 MHz
3	"225_MHz": 225 MHz
4	"113_MHz": 112.5 MHz
5	"56.3_MHz": 56.25 MHz
6	"28.1_MHz": 28.12 MHz
7	"14.1_MHz": 14.06 MHz
8	"7.03_MHz": 7.03 MHz
9	"3.52_MHz": 3.52 MHz
10	"1.76M_Hz": 1.76 MHz
11	"878.91_kHz": 878.91 kHz
12	"439_kHz": 439.45 kHz
13	"220_kHz": 219.73 kHz

/dev..../awgs/n/triggers/n/channel

Properties: Read, Write, Setting
Type: Integer (enumerated)
Unit: None

Selects the signal source for the analogue trigger.

0	"sigin0", "signal_input0": Signal Input 1
1	"sigin1", "signal_input1": Signal Input 2
2	"trigin0", "trigger_input0": Trigger Input 1
3	"trigin1", "trigger_input1": Trigger Input 2
4	"auxout0", "auxiliary_output0": Aux Output 1. Requires an installed digitizer (DIG) option.
5	"auxout1", "auxiliary_output1": Aux Output 2. Requires an installed digitizer (DIG) option.
6	"auxout2", "auxiliary_output2": Aux Output 3. Requires an installed digitizer (DIG) option.
7	"auxout3", "auxiliary_output3": Aux Output 4. Requires an installed digitizer (DIG) option.
8	"auxin0", "auxiliary_input0": Aux Input 1
9	"auxin1", "auxiliary_input1": Aux Input 2
10	"demod3": Osc φ Demod 4
11	"demod7": Osc φ Demod 8
16	"demod0_x": Demod 1 X. Requires an installed digitizer (DIG) option.
17	"demod1_x": Demod 2 X. Requires an installed digitizer (DIG) option.
18	"demod2_x": Demod 3 X. Requires an installed digitizer (DIG) option.
19	"demod3_x": Demod 4 X. Requires an installed digitizer (DIG) option.
20	"demod4_x": Demod 5 X. Requires an installed digitizer (DIG) option.
21	"demod5_x": Demod 6 X. Requires an installed digitizer (DIG) option.
22	"demod6_x": Demod 7 X. Requires an installed digitizer (DIG) option.
23	"demod7_x": Demod 8 X. Requires an installed digitizer (DIG) option.
32	"demod0_y": Demod 1 Y. Requires an installed digitizer (DIG) option.
33	"demod1_y": Demod 2 Y. Requires an installed digitizer (DIG) option.
34	"demod2_y": Demod 3 Y. Requires an installed digitizer (DIG) option.
35	"demod3_y": Demod 4 Y. Requires an installed digitizer (DIG) option.
36	"demod4_y": Demod 5 Y. Requires an installed digitizer (DIG) option.
37	"demod5_y": Demod 6 Y. Requires an installed digitizer (DIG) option.
38	"demod6_y": Demod 7 Y. Requires an installed digitizer (DIG) option.
39	"demod7_y": Demod 8 Y. Requires an installed digitizer (DIG) option.
48	"demod0_r": Demod 1 R. Requires an installed digitizer (DIG) option.
49	"demod1_r": Demod 2 R. Requires an installed digitizer (DIG) option.
50	"demod2_r": Demod 3 R. Requires an installed digitizer (DIG) option.
51	"demod3_r": Demod 4 R. Requires an installed digitizer (DIG) option.
52	"demod4_r": Demod 5 R. Requires an installed digitizer (DIG) option.
53	"demod5_r": Demod 6 R. Requires an installed digitizer (DIG) option.
54	"demod6_r": Demod 7 R. Requires an installed digitizer (DIG) option.
55	"demod7_r": Demod 8 R. Requires an installed digitizer (DIG) option.
64	"demod0_theta": Demod 1 Θ. Requires an installed digitizer (DIG) option.
65	"demod1_theta": Demod 2 Θ. Requires an installed digitizer (DIG) option.
66	"demod2_theta": Demod 3 Θ. Requires an installed digitizer (DIG) option.
67	"demod3_theta": Demod 4 Θ. Requires an installed digitizer (DIG) option.
68	"demod4_theta": Demod 5 Θ. Requires an installed digitizer (DIG) option.
69	"demod5_theta": Demod 6 Θ. Requires an installed digitizer (DIG) option.
70	"demod6_theta": Demod 7 Θ. Requires an installed digitizer (DIG) option.
71	"demod7_theta": Demod 8 Θ. Requires an installed digitizer (DIG) option.
80	"pid0_value": PID 1 value. Requires an installed digitizer (DIG) option.
81	"pid1_value": PID 2 value. Requires an installed digitizer (DIG) option.
82	"pid2_value": PID 3 value. Requires an installed digitizer (DIG) option.
83	"pid3_value": PID 4 value. Requires an installed digitizer (DIG) option.
96	"boxcar0": Boxcar 1. Requires an installed digitizer (DIG) option.
97	"boxcar1": Boxcar 2. Requires an installed digitizer (DIG) option.
112	"au_cartesian0": AU Cartesian 1. Requires an installed digitizer (DIG) option.
113	"au_cartesian1": AU Cartesian 2. Requires an installed digitizer (DIG) option.
128	"au_polar1": Au Polar 2. Requires an installed digitizer (DIG) option.
144	"pid0_shift": PID 1 Shift. Requires an installed digitizer (DIG) option.
145	"pid1_shift": PID 2 Shift. Requires an installed digitizer (DIG) option.
146	"pid2_shift": PID 3 Shift. Requires an installed digitizer (DIG) option.
147	"pid3_shift": PID 4 Shift. Requires an installed digitizer (DIG) option.
176	"awg_marker0": AWG Marker 1. Requires an installed digitizer (DIG) option.
177	"awg_marker1": AWG Marker 2. Requires an installed digitizer (DIG) option.
178	"awg_marker2": AWG Marker 3. Requires an installed digitizer (DIG) option.

179	"awg_marker3": AWG Marker 4. Requires an installed digitizer (DIG) option.
192	"awg_trigger0": AWG Trigger 1. Requires an installed digitizer (DIG) option.
193	"awg_trigger1": AWG Trigger 2. Requires an installed digitizer (DIG) option.
194	"awg_trigger2": AWG Trigger 3. Requires an installed digitizer (DIG) option.
195	"awg_trigger3": AWG Trigger 4. Requires an installed digitizer (DIG) option.
208	"pid0_error": PID 1 Error. Requires an installed digitizer (DIG) option.
209	"pid1_error": PID 2 Error. Requires an installed digitizer (DIG) option.
210	"pid2_error": PID 3 Error. Requires an installed digitizer (DIG) option.
211	"pid3_error": PID 4 Error. Requires an installed digitizer (DIG) option.

/dev..../awgs/n/triggers/n/falling

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Sets a falling edge trigger.

/dev..../awgs/n/triggers/n/force

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Allows to manually force a trigger.

/dev..../awgs/n/triggers/n/gate/enable

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

If enabled the trigger will be gated by the trigger gating input signal.

/dev..../awgs/n/triggers/n/gate/inputselect

Properties: Read, Write, Setting
Type: Integer (enumerated)
Unit: None

Select the signal source used for trigger gating if gating is enabled.

0	"trigin2", "trigger_input2": Trigger In 3
1	"trigin3", "trigger_input3": Trigger In 4

/dev..../awgs/n/triggers/n/hysteresis/absolute

Properties: Read, Write, Setting
Type: Double
Unit: V

Defines the voltage the source signal must deviate from the trigger level before the trigger is rearmed again. Set to 0 to turn it off. The sign is defined by the Edge setting.

/dev..../awgs/n/triggers/n/hysteresis/mode

Properties: Read, Write, Setting
Type: Integer (enumerated)
Unit: None

Selects the mode to define the hysteresis strength. The relative mode will work best over the full input range as long as the analog input signal does not suffer from excessive noise.

0	"absolute": Selects absolute hysteresis (V).
1	"relative": Selects a hysteresis relative to the adjusted full scale signal input range (%).

/dev..../awgs/n/triggers/n/hysteresis/relative

Properties: Read, Write, Setting
Type: Double
Unit: %

Hysteresis relative to the adjusted full scale signal input range. A hysteresis value larger than 100% is allowed.

/dev..../awgs/n/triggers/n/level

Properties: Read, Write, Setting
Type: Double
Unit: V

Defines the analog trigger level.

/dev..../awgs/n/triggers/n/rising

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Sets a rising edge trigger.

/dev..../awgs/n/triggers/n/slope

Properties: Read, Write, Setting
Type: Integer (enumerated)
Unit: None

Select the signal edge that should activate the trigger. The trigger will be level sensitive when the Level option is selected.

0	"level_sensitive": Level sensitive trigger
1	"rising_edge": Rising edge trigger
2	"falling_edge": Falling edge trigger
3	"both_edegs": Rising or falling edge trigger

/dev..../awgs/n/triggers/n/state

Properties: Read
Type: Integer (64 bit)
Unit: None

State of the Trigger: No trigger detected/trigger detected.

/dev..../awgs/n/userregs/n

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Integer user register value. The sequencer has reading and writing access to the user register values during run time.

/dev..../awgs/n/waveform/descriptors

Properties: Read
Type: ZIVectorData
Unit: None

JSON-formatted string containing a dictionary of various properties of the current waveform: name, filename, function, channels, marker bits, length, timestamp.

/dev..../awgs/n/waveform/memoryusage

Properties: Read
Type: Double
Unit: %

Amount of the used waveform data relative to the device cache memory. The cache memory provides space for 32 kSa (32'768 Sa) per-channel of waveform data. Memory Usage over 100% means that waveforms must be loaded from the main memory of 64 MSa (67'108'864 Sa) per-channel during playback, which can lead to delays.

/dev..../awgs/n/waveform/waves/n

Properties: Read, Write
Type: ZIVectorData
Unit: None

The waveform data in the instrument's native format for the given playWave waveform index. This node will not work with subscribe as it does not push updates. For short vectors get may be used. For long vectors (causing get to time out) getAsEvent and poll can be used. The index of the waveform to be replaced can be determined using the Waveform sub-tab in the AWG tab of the LabOne User Interface.

7.2.4. CLOCKBASE

/dev..../clockbase

Properties: Read
Type: Double
Unit: Hz

Returns the internal clock frequency of the device.

7.2.5. DIOS

/dev..../dios/n/auxdrive

Properties: Read
Type: Integer (64 bit)
Unit: None

Not used. Reserved for future use.

/dev..../dios/n/decimation

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Sets the decimation factor for DIO data streamed to the host computer.

/dev..../dios/n/drive

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

When on (1), the corresponding 8-bit bus is in output mode. When off (0), it is in input mode. Bit 0 corresponds to the least significant byte. For example, the value 1 drives the least significant byte, the value 8 drives the most significant byte.

/dev..../dios/n/extclk

Properties: Read, Write, Setting
Type: Integer (enumerated)
Unit: None

Select DIO internal or external clocking.

0	The DIO is internally clocked with a frequency of 56.25 MHz.
1	"external": The DIO is externally clocked with a clock signal connected to DIO Pin 68. The available range is from 1 Hz up to the internal clock frequency.
2	"internal": The DIO is internally clocked with a frequency of 50 MHz.

/dev..../dios/n/input

Properties: Read, Stream
Type: ZIDIOSample
Unit: None

Gives the value of the DIO input for those bytes where drive is disabled.

/dev..../dios/n/mode

Properties: Read, Write, Setting
Type: Integer (enumerated)
Unit: None

Select DIO mode

0	"manual": Enables manual control of the DIO output bits.
1	"awg_sequencer_commands": Enables setting of DIO output values by AWG sequencer commands.
2	"qa_result": Enables setting of DIO output values by QA results.
4	"qa_result_qccs": Enables setting of DIO output values by QA results compatible with the QCCS.

/dev..../dios/n/output

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Sets the value of the DIO output for those bytes where 'drive' is enabled.

7.2.6. FEATURES

/dev..../features/code

Properties: Write
Type: String
Unit: None

Node providing a mechanism to write feature codes.

/dev..../features/devtype

Properties: Read
Type: String
Unit: None

Returns the device type.

/dev..../features/options

Properties: Read
Type: String
Unit: None

Returns enabled options.

/dev..../features/serial

Properties: Read
Type: String
Unit: None

Device serial number.

7.2.7. OSCS

/dev..../oscs/n/freq

Properties: Read, Write, Setting
Type: Double
Unit: Hz

Frequency control for each oscillator.

7.2.8. QAS

/dev..../qas/n/bypass/crosstalk

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Bypass the Crosstalk matrix in order to reduce the latency through the system.

/dev..../qas/n/bypass/deskew

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Bypass the Deskew matrix in order to reduce the latency through the system.

/dev..../qas/n/bypass/rotation

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Bypass the complex Rotation in order to reduce the latency through the system.

/dev..../qas/n/correlations/n/enable

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Enable correlation for a given channel.

/dev..../qas/n/correlations/n/source

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Controls the source with which correlation should be made. Selecting the same source as the readout channel number is effectively self-correlation.

/dev..../qas/n/crosstalk/bypass

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Bypass the Crosstalk matrix in order to reduce the latency through the system.

/dev..../qas/n/crosstalk/rows/n/cols/n

Properties: Read, Write, Setting
Type: Double
Unit: None

Transformation coefficients for each channel as a 10×10 matrix. The values are defined as floating point numbers. In hardware, the coefficients are implemented as 17-bit signed integers.

/dev..../qas/n/delay

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Delay in units of samples that adjusts the time at which the integration starts in relation to the trigger signal of the weighted integration units.

/dev..../qas/n/deskew/rows/n/cols/n

Properties: Read, Write, Setting
Type: Double
Unit: None

Implements a 2×2 matrix multiplication. The two input signals are treated as a vector with two elements and the result is a vector as well.

/dev..../qas/n/integration/errors

Properties: Read
Type: Integer (64 bit)
Unit: None

Number of hold-off violations detected in the INTEGRATION unit since last reset.

/dev..../qas/n/integration/length

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Integration time of all weighted integration units specified in units of samples. A maximum of 4096 samples can be integrated, which corresponds to 2.3 µs.

/dev..../qas/n/integration/mode

Properties:	Read, Write, Setting
Type:	Integer (enumerated)
Unit:	None

Operation mode of all weighted integration units.

0	"normal": Normal mode. The integration weights are given by the user-programmed filter memory 1
1	"spectroscopy": Spectroscopy mode. The integration weights are generated by a digital oscillator. This mode offers enhanced frequency resolution.

/dev..../qas/n/integration/sources/n

Properties:	Read, Write, Setting
Type:	Integer (enumerated)
Unit:	None

Controls the routing of the input signals to the INTEGRATION units.

0	"sigin0_real_sigin1_imag": Signal input 1 to real part, Signal input 2 to imaginary part.
1	"sigin1_real_sigin0_imag": Signal input 2 to real part, Signal input 1 to imaginary part

/dev..../qas/n/integration/trigger/channel

Properties:	Read, Write, Setting
Type:	Integer (enumerated)
Unit:	None

Selects the trigger channel for the weighted integration.

0	Trigger 1
1	Trigger 2
2	Trigger In 3
3	Trigger In 4
7	AWG Integration Trigger (default)

/dev..../qas/n/integration/weights/n/imag

Properties:	Read, Write, Setting
Type:	ZIVectorData
Unit:	None

The weight functions to be applied on the imaginary part of the input signal. The weights are written as vectors. In the hardware the weights are implemented as 17-bit integers.

/dev..../qas/n/integration/weights/n/real

Properties:	Read, Write, Setting
Type:	ZIVectorData
Unit:	None

The weight functions to be applied on the real part of the input signal. The weights are written as vectors. In the hardware the weights are implemented as 17-bit integers.

/dev..../qas/n/monitor/acquired

Properties: Read
Type: Integer (64 bit)
Unit: None

Number of acquired averages.

/dev..../qas/n/monitor/averages

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

log2 of the number of averages to perform, i.e. 0 means no averaging, 1 means 2 values are averaged, etc. Maximum setting is 15 meaning 2^15 values are averaged.

/dev..../qas/n/monitor/enable

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Enable the trigger inputs of the MONITOR from the AWG.

/dev..../qas/n/monitor/errors

Properties: Read
Type: Integer (64 bit)
Unit: None

Number of hold-off errors detected since last reset.

/dev..../qas/n/monitor/inputs/n/wave

Properties: Read
Type: ZIVectorData
Unit: V

The results of the data acquired for Signal Input in units of volts.

/dev..../qas/n/monitor/length

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

The time duration of each capture in samples. A maximum of 4096 samples can be captured, which corresponds to 2.3 µs.

/dev..../qas/n/monitor/reset

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Reset the Input Monitor and prepare it for a new measurement.

/dev..../qas/n/monitor/trigger/channel

Properties:	Read, Write, Setting
Type:	Integer (enumerated)
Unit:	None

Selects the trigger channel for the input averaging.

0	Trigger 1
1	Trigger 2
2	Trigger In 3
3	Trigger In 4
7	AWG Monitor Trigger (default)

/dev..../qas/n/result/acquired

Properties:	Read
Type:	Integer (64 bit)
Unit:	None

Indicates the index of the acquisition that will be performed on the next trigger.

/dev..../qas/n/result/averages

Properties:	Read, Write, Setting
Type:	Integer (64 bit)
Unit:	None

log2 of the number of averages to perform, i.e. 0 means no averaging, 1 means 2 values are averaged, etc. Maximum setting is 15 meaning 2^15 values are averaged.

/dev..../qas/n/result/data/n/wave

Properties:	Read
Type:	ZIVectorData
Unit:	V

The results of the data acquired for Signal Input in units of volts.

/dev..../qas/n/result/enable

Properties:	Read, Write, Setting
Type:	Integer (64 bit)
Unit:	None

Enable the trigger inputs of the result logger from the AWG.

/dev..../qas/n/result/errors

Properties:	Read
Type:	Integer (64 bit)
Unit:	None

Number of hold-off errors detected since last reset.

/dev..../qas/n/result/length

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Number of data points to record. One data point corresponds to a single averaged output of the selected source.

/dev..../qas/n/result/mode

Properties: Read, Write, Setting
Type: Integer (enumerated)
Unit: None

Selects the order of the result logger.

0	"experiment_first": Experiment first.
1	"average_first": Average first.

/dev..../qas/n/result/reset

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Reset the Result Logger and prepare it for a new measurement.

/dev..../qas/n/result/source

Properties: Read, Write, Setting
Type: Integer (enumerated)
Unit: None

Select the signal source of the Result Logger.

0	"result_after_crosstalk_unit": Result after crosstalk unit.
1	"result_after_threshold_unit": Result after Threshold unit.
2	"result_after_rotation_unit": Result after Rotation unit.
4	"correlation_after_crosstalk_unit": Correlation unit after the Crosstalk unit.
5	"correlation_after_threshold_unit": Correlation unit after the Threshold unit.

/dev..../qas/n/result/statistics/data/n/errors

Properties: Read
Type: Integer (64 bit)
Unit: None

Number of state errors measured for this channel. A state error occurs when there is a discrepancy between the measured state of this channel and the state that is predicted based on the configured state table.

/dev..../qas/n/result/statistics/data/n/flips

Properties: Read
Type: Integer (64 bit)
Unit: None

Number of flips measured for this channel. A flip is defined as a change in qubit state from one measurement to the next.

/dev..../qas/n/result/statistics/data/n/ones

Properties: Read
Type: Integer (64 bit)
Unit: None

Number of logical ones measured for this channel.

/dev..../qas/n/result/statistics/enable

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Enable automatic readout of the STATISTICS results.

/dev..../qas/n/result/statistics/length

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

The number of data points to record. One data point corresponds to a single averaged output of the selected source.

/dev..../qas/n/result/statistics/reset

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Reset the STATISTICS results in preparation for a new measurement.

/dev..../qas/n/result/statistics/stateerrors

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Number of STATEMAP violations detected during measurement.

/dev..../qas/n/result/statistics/statemap

Properties: Read, Write, Setting
Type: ZIVectorData
Unit: None

Defines the state table. The node has one entry for each possible state, i.e. 32 entries in case of five readout channels. The value of each entry defines the expected next state as a simple bit mask mapped to an unsigned integer.

/dev..../qas/n/rotations/n

Properties: Read, Write, Setting
Type: Complex Double
Unit: None

Rotation coefficient applied to the signals after integration. The values are defined as complex floating point numbers.

/dev..../qas/n/thresholds/n/correlation/enable

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Enables the correlation for the given channel.

/dev..../qas/n/thresholds/n/correlation/source

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Select the channel with which correlation should be made. Selecting the same channel as the readout channel number corresponds to self-correlation.

/dev..../qas/n/thresholds/n/level

Properties: Read, Write, Setting
Type: Double
Unit: None

The discretization level applied to the output of the Crosstalk Suppression matrix.

7.2.9. SCOPES

/dev..../scopes/n/channel

Properties: Read, Write, Setting
Type: Integer (enumerated)
Unit: None

Activates the scope channels.

1	Only channel 1 is active.
2	Only channel 2 is active.
3	"both": Both, channel 1 and 2 are active.

/dev..../scopes/n/channels/n/bwlimit

Properties: Read, Write, Setting
Type: Integer (enumerated)
Unit: None

Selects between sample decimation and sample averaging. Averaging avoids aliasing, but may conceal signal peaks.

0	"on": On: Selects sample averaging for sample rates lower than the maximal available sampling rate.
1	"off": OFF: Selects sample decimation for sample rates lower than the maximal available sampling rate.

/dev..../scopes/n/channels/n/fullscale

Properties: Read, Write, Setting
Type: Double
Unit: Dependent

Indicates the full scale value of the scope channel.

/dev..../scopes/n/channels/n/inputselect

Properties:	Read, Write, Setting
Type:	Integer (enumerated)
Unit:	None

Selects the scope input signal.

0	"sigin0", "signal_input0": Signal Input 1
1	"sigin1", "signal_input1": Signal Input 2
2	"trigin0", "trigger_input0": Trigger Input 1
3	"trigin1", "trigger_input1": Trigger Input 2
4	"auxout0", "auxiliary_output0": Aux Output 1. Requires an installed digitizer (DIG) option.
5	"auxout1", "auxiliary_output1": Aux Output 2. Requires an installed digitizer (DIG) option.
6	"auxout2", "auxiliary_output2": Aux Output 3. Requires an installed digitizer (DIG) option.
7	"auxout3", "auxiliary_output3": Aux Output 4. Requires an installed digitizer (DIG) option.
8	"auxin0", "auxiliary_input0": Aux Input 1
9	"auxin1", "auxiliary_input1": Aux Input 2
10	"demod3": Osc φ Demod 4
11	"demod7": Osc φ Demod 8
16	"demod0_x": Demod 1 X. Requires an installed digitizer (DIG) option.
17	"demod1_x": Demod 2 X. Requires an installed digitizer (DIG) option.
18	"demod2_x": Demod 3 X. Requires an installed digitizer (DIG) option.
19	"demod3_x": Demod 4 X. Requires an installed digitizer (DIG) option.
20	"demod4_x": Demod 5 X. Requires an installed digitizer (DIG) option.
21	"demod5_x": Demod 6 X. Requires an installed digitizer (DIG) option.
22	"demod6_x": Demod 7 X. Requires an installed digitizer (DIG) option.
23	"demod7_x": Demod 8 X. Requires an installed digitizer (DIG) option.
32	"demod0_y": Demod 1 Y. Requires an installed digitizer (DIG) option.
33	"demod1_y": Demod 2 Y. Requires an installed digitizer (DIG) option.
34	"demod2_y": Demod 3 Y. Requires an installed digitizer (DIG) option.
35	"demod3_y": Demod 4 Y. Requires an installed digitizer (DIG) option.
36	"demod4_y": Demod 5 Y. Requires an installed digitizer (DIG) option.
37	"demod5_y": Demod 6 Y. Requires an installed digitizer (DIG) option.
38	"demod6_y": Demod 7 Y. Requires an installed digitizer (DIG) option.
39	"demod7_y": Demod 8 Y. Requires an installed digitizer (DIG) option.
48	"demod0_r": Demod 1 R. Requires an installed digitizer (DIG) option.
49	"demod1_r": Demod 2 R. Requires an installed digitizer (DIG) option.
50	"demod2_r": Demod 3 R. Requires an installed digitizer (DIG) option.
51	"demod3_r": Demod 4 R. Requires an installed digitizer (DIG) option.
52	"demod4_r": Demod 5 R. Requires an installed digitizer (DIG) option.
53	"demod5_r": Demod 6 R. Requires an installed digitizer (DIG) option.
54	"demod6_r": Demod 7 R. Requires an installed digitizer (DIG) option.
55	"demod7_r": Demod 8 R. Requires an installed digitizer (DIG) option.
64	"demod0_theta": Demod 1 Θ. Requires an installed digitizer (DIG) option.
65	"demod1_theta": Demod 2 Θ. Requires an installed digitizer (DIG) option.
66	"demod2_theta": Demod 3 Θ. Requires an installed digitizer (DIG) option.
67	"demod3_theta": Demod 4 Θ. Requires an installed digitizer (DIG) option.
68	"demod4_theta": Demod 5 Θ. Requires an installed digitizer (DIG) option.
69	"demod5_theta": Demod 6 Θ. Requires an installed digitizer (DIG) option.
70	"demod6_theta": Demod 7 Θ. Requires an installed digitizer (DIG) option.
71	"demod7_theta": Demod 8 Θ. Requires an installed digitizer (DIG) option.
80	"pid0_value": PID 1 value. Requires an installed digitizer (DIG) option.
81	"pid1_value": PID 2 value. Requires an installed digitizer (DIG) option.
82	"pid2_value": PID 3 value. Requires an installed digitizer (DIG) option.
83	"pid3_value": PID 4 value. Requires an installed digitizer (DIG) option.
96	"boxcar0": Boxcar 1. Requires an installed digitizer (DIG) option.
97	"boxcar1": Boxcar 2. Requires an installed digitizer (DIG) option.
112	"au_cartesian0": AU Cartesian 1. Requires an installed digitizer (DIG) option.
113	"au_cartesian1": AU Cartesian 2. Requires an installed digitizer (DIG) option.
128	"au_polar1": Au Polar 2. Requires an installed digitizer (DIG) option.
144	"pid0_shift": PID 1 Shift. Requires an installed digitizer (DIG) option.
145	"pid1_shift": PID 2 Shift. Requires an installed digitizer (DIG) option.
146	"pid2_shift": PID 3 Shift. Requires an installed digitizer (DIG) option.
147	"pid3_shift": PID 4 Shift. Requires an installed digitizer (DIG) option.
160	Reserved for future use.
161	Reserved for future use.
176	"awg_marker0": AWG Marker 1. Requires an installed digitizer (DIG) option.

177	"awg_marker1": AWG Marker 2. Requires an installed digitizer (DIG) option.
178	"awg_marker2": AWG Marker 3. Requires an installed digitizer (DIG) option.
179	"awg_marker3": AWG Marker 4. Requires an installed digitizer (DIG) option.
192	"awg_trigger0": AWG Trigger 1. Requires an installed digitizer (DIG) option.
193	"awg_trigger1": AWG Trigger 2. Requires an installed digitizer (DIG) option.
194	"awg_trigger2": AWG Trigger 3. Requires an installed digitizer (DIG) option.
195	"awg_trigger3": AWG Trigger 4. Requires an installed digitizer (DIG) option.
208	"pid0_error": PID 1 Error. Requires an installed digitizer (DIG) option.
209	"pid1_error": PID 2 Error. Requires an installed digitizer (DIG) option.
210	"pid2_error": PID 3 Error. Requires an installed digitizer (DIG) option.
211	"pid3_error": PID 4 Error. Requires an installed digitizer (DIG) option.

/dev..../scopes/n/channels/n/limitlower

Properties: Read, Write, Setting
Type: Double
Unit: Dependent

Lower limit of the scope full scale range. For demodulator, PID, Boxcar, and AU signals the limit should be adjusted so that the signal covers the specified range to achieve optimal resolution.

/dev..../scopes/n/channels/n/limitupper

Properties: Read, Write, Setting
Type: Double
Unit: Dependent

Upper limit of the scope full scale range. For demodulator, PID, Boxcar, and AU signals the limit should be adjusted so that the signal covers the specified range to achieve optimal resolution.

/dev..../scopes/n/channels/n/offset

Properties: Read, Write, Setting
Type: Double
Unit: Dependent

Indicates the offset value of the scope channel.

/dev..../scopes/n/enable

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Enables the acquisition of scope shots.

/dev..../scopes/n/length

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Defines the length of the recorded Scope shot in number of samples.

/dev..../scopes/n/segments/count

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Specifies the number of segments to be recorded in device memory. The maximum scope shot size is given by the available memory divided by the number of segments. This functionality requires the DIG option.

/dev..../scopes/n/segments/enable

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Enable segmented scope recording. This allows for full bandwidth recording of scope shots with a minimum dead time between individual shots. This functionality requires the DIG option.

/dev..../scopes/n/single

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Puts the Scope into single shot mode.

/dev..../scopes/n/stream/enables/n

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Enable scope streaming for the specified channel. This allows for continuous recording of scope data on the plotter and streaming to disk. Note: scope streaming requires the DIG option.

/dev..../scopes/n/stream/rate

Properties: Read, Write, Setting
Type: Integer (enumerated)
Unit: Hz

Streaming Rate of the scope channels. The streaming rate can be adjusted independent from the scope sampling rate. The maximum rate depends on the interface used for transfer. Note: scope streaming requires the DIG option.

6	"28.1_MHz": 28.1 MHz
7	"14_MHz": 14.0 MHz
8	"7.03_MHz": 7.03 MHz
9	"3.5_MHz": 3.50 MHz
10	"1.75_MHz": 1.75 MHz
11	"880_kHz": 880 kHz
12	"440_kHz": 440 kHz
13	"220_kHz": 220 kHz
14	"110_kHz": 110 kHz
15	"54.9_kHz": 54.9 kHz
16	"27.5_kHz": 27.5 kHz

/dev..../scopes/n/stream/sample

Properties:	Read, Stream
Type:	ZIScopeWave
Unit:	None

Streaming node containing scope sample data. Note: scope streaming requires the DIG option.

/dev..../scopes/n/time

Properties:	Read, Write, Setting
Type:	Integer (enumerated)
Unit:	None

Defines the time base of the scope from the divider exponent of the instrument's clock base. The resulting sampling time is 2^n/clockbase.

0	1.80 GHz
1	900 MHz
2	450 MHz
3	225 MHz
4	113 MHz
5	56.2 MHz
6	28.1 MHz
7	14.0 MHz
8	7.03 MHz
9	3.50 MHz
10	1.75 MHz
11	880 kHz
12	440 kHz
13	220 kHz
14	110 kHz
15	54.9 kHz
16	27.5 kHz

/dev..../scopes/n/trigchannel

Properties: Read, Write, Setting
Type: Integer (enumerated)
Unit: None

Selects the trigger source signal.

0	"sigin0", "signal_input0": Signal Input 1
1	"sigin1", "signal_input1": Signal Input 2
2	"trigin0", "trigger_input0": Trigger Input 1
3	"trigin1", "trigger_input1": Trigger Input 2
4	"auxout0", "auxiliary_output0": Aux Output 1. Requires an installed digitizer (DIG) option.
5	"auxout1", "auxiliary_output1": Aux Output 2. Requires an installed digitizer (DIG) option.
6	"auxout2", "auxiliary_output2": Aux Output 3. Requires an installed digitizer (DIG) option.
7	"auxout3", "auxiliary_output3": Aux Output 4. Requires an installed digitizer (DIG) option.
8	"auxin0", "auxiliary_input0": Aux Input 1
9	"auxin1", "auxiliary_input1": Aux Input 2
10	"demod3": Osc φ Demod 4
11	"demod7": Osc φ Demod 8
16	"demod0_x": Demod 1 X. Requires an installed digitizer (DIG) option.
17	"demod1_x": Demod 2 X. Requires an installed digitizer (DIG) option.
18	"demod2_x": Demod 3 X. Requires an installed digitizer (DIG) option.
19	"demod3_x": Demod 4 X. Requires an installed digitizer (DIG) option.
20	"demod4_x": Demod 5 X. Requires an installed digitizer (DIG) option.
21	"demod5_x": Demod 6 X. Requires an installed digitizer (DIG) option.
22	"demod6_x": Demod 7 X. Requires an installed digitizer (DIG) option.
23	"demod7_x": Demod 8 X. Requires an installed digitizer (DIG) option.
32	"demod0_y": Demod 1 Y. Requires an installed digitizer (DIG) option.
33	"demod1_y": Demod 2 Y. Requires an installed digitizer (DIG) option.
34	"demod2_y": Demod 3 Y. Requires an installed digitizer (DIG) option.
35	"demod3_y": Demod 4 Y. Requires an installed digitizer (DIG) option.
36	"demod4_y": Demod 5 Y. Requires an installed digitizer (DIG) option.
37	"demod5_y": Demod 6 Y. Requires an installed digitizer (DIG) option.
38	"demod6_y": Demod 7 Y. Requires an installed digitizer (DIG) option.
39	"demod7_y": Demod 8 Y. Requires an installed digitizer (DIG) option.
48	"demod0_r": Demod 1 R. Requires an installed digitizer (DIG) option.
49	"demod1_r": Demod 2 R. Requires an installed digitizer (DIG) option.
50	"demod2_r": Demod 3 R. Requires an installed digitizer (DIG) option.
51	"demod3_r": Demod 4 R. Requires an installed digitizer (DIG) option.
52	"demod4_r": Demod 5 R. Requires an installed digitizer (DIG) option.
53	"demod5_r": Demod 6 R. Requires an installed digitizer (DIG) option.
54	"demod6_r": Demod 7 R. Requires an installed digitizer (DIG) option.
55	"demod7_r": Demod 8 R. Requires an installed digitizer (DIG) option.
64	"demod0_theta": Demod 1 Θ. Requires an installed digitizer (DIG) option.
65	"demod1_theta": Demod 2 Θ. Requires an installed digitizer (DIG) option.
66	"demod2_theta": Demod 3 Θ. Requires an installed digitizer (DIG) option.
67	"demod3_theta": Demod 4 Θ. Requires an installed digitizer (DIG) option.
68	"demod4_theta": Demod 5 Θ. Requires an installed digitizer (DIG) option.
69	"demod5_theta": Demod 6 Θ. Requires an installed digitizer (DIG) option.
70	"demod6_theta": Demod 7 Θ. Requires an installed digitizer (DIG) option.
71	"demod7_theta": Demod 8 Θ. Requires an installed digitizer (DIG) option.
80	"pid0_value": PID 1 value. Requires an installed digitizer (DIG) option.
81	"pid1_value": PID 2 value. Requires an installed digitizer (DIG) option.
82	"pid2_value": PID 3 value. Requires an installed digitizer (DIG) option.
83	"pid3_value": PID 4 value. Requires an installed digitizer (DIG) option.
96	"boxcar0": Boxcar 1. Requires an installed digitizer (DIG) option.
97	"boxcar1": Boxcar 2. Requires an installed digitizer (DIG) option.
112	"au_cartesian0": AU Cartesian 1. Requires an installed digitizer (DIG) option.
113	"au_cartesian1": AU Cartesian 2. Requires an installed digitizer (DIG) option.
128	"au_polar0": AU Polar 1. Requires an installed digitizer (DIG) option.
129	"au_polar1": Au Polar 2. Requires an installed digitizer (DIG) option.
144	"pid0_shift": PID 1 Shift. Requires an installed digitizer (DIG) option.
145	"pid1_shift": PID 2 Shift. Requires an installed digitizer (DIG) option.
146	"pid2_shift": PID 3 Shift. Requires an installed digitizer (DIG) option.
147	"pid3_shift": PID 4 Shift. Requires an installed digitizer (DIG) option.
176	"awg_marker0": AWG Marker 1. Requires an installed digitizer (DIG) option.
177	"awg_marker1": AWG Marker 2. Requires an installed digitizer (DIG) option.

178	"awg_marker2": AWG Marker 3. Requires an installed digitizer (DIG) option.
179	"awg_marker3": AWG Marker 4. Requires an installed digitizer (DIG) option.
192	"awg_trigger0": AWG Trigger 1. Requires an installed digitizer (DIG) option.
193	"awg_trigger1": AWG Trigger 2. Requires an installed digitizer (DIG) option.
194	"awg_trigger2": AWG Trigger 3. Requires an installed digitizer (DIG) option.
195	"awg_trigger3": AWG Trigger 4. Requires an installed digitizer (DIG) option.
208	"pid0_error": PID 1 Error. Requires an installed digitizer (DIG) option.
209	"pid1_error": PID 2 Error. Requires an installed digitizer (DIG) option.
210	"pid2_error": PID 3 Error. Requires an installed digitizer (DIG) option.
211	"pid3_error": PID 4 Error. Requires an installed digitizer (DIG) option.

/dev..../scopes/n/trigdelay

Properties: Read, Write, Setting
Type: Double
Unit: s

Trigger position relative to reference. A positive delay results in less data being acquired before the trigger point, a negative delay results in more data being acquired before the trigger point.

/dev..../scopes/n/trigenable

Properties: Read, Write, Setting
Type: Integer (enumerated)
Unit: None

When triggering is enabled scope data are acquired every time the defined trigger condition is met.

0	"off": OFF: Continuous scope shot acquisition
1	"on": ON: Trigger based scope shot acquisition

/dev..../scopes/n/trigfalling

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

When set (1), enables falling edge triggering. This settings is synchronized with the settings done in the /TRIGSLOPE node.

/dev..../scopes/n/trigforce

Properties: Read, Write
Type: Integer (64 bit)
Unit: None

Forces a trigger event.

/dev..../scopes/n/triggate/enable

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

If enabled the trigger will be gated by the trigger gating input signal. This feature requires the DIG option.

/dev..../scopes/n/triggate/inputselect

Properties: Read, Write, Setting
Type: Integer (enumerated)
Unit: None

Select the signal source used for trigger gating if gating is enabled. This feature requires the DIG option.

0	"trigin2_high", "trigger_input2_high": Only trigger if the Trigger Input 3 is at high level.
1	"trigin2_low", "trigger_input2_low": Only trigger if the Trigger Input 3 is at low level.
2	"trigin3_high", "trigger_input3_high": Only trigger if the Trigger Input 4 is at high level.
3	"trigin3_low", "trigger_input3_low": Only trigger if the Trigger Input 4 is at low level.

/dev..../scopes/n/trigholdoff

Properties: Read, Write, Setting
Type: Double
Unit: s

Defines the time before the trigger is rearmed after a recording event.

/dev..../scopes/n/trigholdoffcount

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Defines the trigger event number that will trigger the next recording after a recording event. A value of '1' will start a recording for each trigger event.

/dev..../scopes/n/trigholdoffmode

Properties: Read, Write, Setting
Type: Integer (enumerated)
Unit: None

Selects the holdoff mode.

0	"time": Holdoff is defined as time (s).
1	"number_of_events": Holdoff is defined as number of events.

/dev..../scopes/n/trighysteresis/absolute

Properties: Read, Write, Setting
Type: Double
Unit: V

Defines the voltage the source signal must deviate from the trigger level before the trigger is rearmed again. Set to 0 to turn it off. The sign is defined by the Edge setting.

/dev..../scopes/n/trighysteresis/mode

Properties: Read, Write, Setting
Type: Integer (enumerated)
Unit: None

Selects the mode to define the hysteresis strength. The relative mode will work best over the full input range as long as the analog input signal does not suffer from excessive noise.

0	"absolute": Selects absolute hysteresis.
1	"relative": Selects a hysteresis relative to the adjusted full scale signal input range.

/dev..../scopes/n/trighysteresis/relative

Properties: Read, Write, Setting
Type: Double
Unit: None

Hysteresis relative to the adjusted full scale signal input range. A hysteresis value larger than 1 (100%) is allowed.

/dev..../scopes/n/triglevel

Properties: Read, Write, Setting
Type: Double
Unit: V

Defines the trigger level.

/dev..../scopes/n/trigreference

Properties: Read, Write, Setting
Type: Double
Unit: None

Trigger reference position relative to the acquired data. Default is 50% (0.5) which results in a reference point in the middle of the acquired data.

/dev..../scopes/n/trigrising

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

When set (1), enables rising edge triggering. This settings is synchronized with the settings done in the /TRIGFALLING node.

/dev..../scopes/n/trigslope

Properties: Read, Write, Setting
Type: Integer (enumerated)
Unit: None

Sets on which slope of the trigger signal the scope should trigger. This setting is synchronized with the settings done in the /TRIGFALLING and /TRIGRISING nodes.

0	"none": None
1	"rising": Rising edge triggered.
2	"falling": Falling edge triggered.
3	"both": Triggers on both the rising and falling edge.

/dev..../scopes/n/trigstate

Properties: Read
Type: Integer (64 bit)
Unit: None

When 1, indicates that the trigger signal satisfies the conditions of the trigger.

/dev..../scopes/n/wave

Properties: Read, Stream
Type: ZIScopeWave
Unit: None

Contains the scope shot data.

7.2.10. SIGINS

/dev..../sigins/n/ac

Properties: Read, Write, Setting
Type: Integer (enumerated)
Unit: None

Defines the input coupling for the Signal Inputs. AC coupling inserts a high-pass filter.

0	"dc": OFF: DC coupling
1	"ac": ON: AC coupling

/dev..../sigins/n/autorange

Properties: Read, Write
Type: Integer (64 bit)
Unit: None

Automatic adjustment of the Range to about two times the maximum signal input amplitude measured over about 100 ms.

/dev..../sigins/n/bw

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Bandwidth of input aliasing filter at 600 MHz (ON) or 900 MHz (OFF) cut off frequency.

/dev..../sigins/n/diff

Properties: Read, Write, Setting
Type: Integer (enumerated)
Unit: None

Switch input mode between normal (OFF), inverted, and differential. The differential modes are implemented digitally and are not suited for analog common-mode rejection. When using the differential modes, the user is responsible for keeping the configuration (range, coupling, termination) of both channels in sync, the device provides no control mechanisms to force that.

0	"off": Off
1	"inverted": Inverted

/dev..../sigins/n/imp50

Properties: Read, Write, Setting
Type: Integer (enumerated)
Unit: None

Switches between 50 Ohm (ON) and 1 M Ohm (OFF).

0	"1_MOhm": OFF: 1 M Ohm
1	"50_Ohm": ON: 50 Ohm

/dev..../sigins/n/max

Properties: Read, Write
Type: Double
Unit: None

Indicates the maximum measured value at the input normalized to input range.

/dev..../sigins/n/min

Properties: Read, Write
Type: Double
Unit: None

Indicates the minimum measured value at the input normalized to input range.

/dev..../sigins/n/on

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Enables the signal input.

/dev..../sigins/n/range

Properties: Read, Write, Setting
Type: Double
Unit: V

Defines the gain of the analog input amplifier. The range should exceed the incoming signal by roughly a factor two including a potential DC offset. The instrument selects the next higher available range relative to a value inserted by the user. A suitable choice of this setting optimizes the accuracy and signal-to-noise ratio by ensuring that the full dynamic range of the input ADC is used.

/dev..../sigins/n/scaling

Properties: Read, Write, Setting
Type: Double
Unit: None

Applies the given scaling factor to the input signal.

7.2.11. SIGOUTS

/dev..../sigouts/n/amplitudes/n

Properties: Read, Write, Setting
Type: Double
Unit: V

Sets the peak amplitude that the oscillator assigned to the given demodulation channel contributes to the signal output.

/dev..../sigouts/n/autorange

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

If enabled, selects the most suited output range automatically.

/dev..../sigouts/n/enables/n

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: V

Enables individual output signal amplitude. When the MD option is used, it is possible to generate signals being the linear combination of the available demodulator frequencies.

/dev..../sigouts/n/imp50

Properties: Read, Write, Setting
Type: Integer (enumerated)
Unit: None

Select the load impedance between 50 Ohm and HiZ. The impedance of the output is always 50 Ohm. For a load impedance of 50 Ohm the displayed voltage is half the output voltage to reflect the voltage seen at the load.

0	"high_impedance": HiZ
1	"50_Ohm": 50 Ohm

/dev..../sigouts/n/offset

Properties: Read, Write, Setting
Type: Double
Unit: V

Defines the DC voltage that is added to the dynamic part of the output signal.

/dev..../sigouts/n/on

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Enabling/Disabling the Signal Output. Corresponds to the blue LED indicator on the instrument front panel.

/dev..../sigouts/n/over

Properties: Read
Type: Integer (64 bit)
Unit: None

Indicates that the signal output is overloaded.

/dev..../sigouts/n/range

Properties: Read, Write, Setting
Type: Double
Unit: V

Sets the output voltage range. The instrument selects the next higher available range.

7.2.12. STATS

/dev..../stats/cmdstream/bandwidth

Properties: Read
Type: Double
Unit: Mbit/s

Command streaming bandwidth usage on the physical network connection between device and data server.

/dev..../stats/cmdstream/bytesreceived

Properties: Read
Type: Integer (64 bit)
Unit: B

Number of bytes received on the command stream from the device since session start.

/dev..../stats/cmdstream/bytessent

Properties: Read
Type: Integer (64 bit)
Unit: B

Number of bytes sent on the command stream from the device since session start.

/dev..../stats/cmdstream/packetslost

Properties: Read
Type: Integer (64 bit)
Unit: None

Number of command packets lost since device start. Command packets contain device settings that are sent to and received from the device.

/dev..../stats/cmdstream/packetsreceived

Properties: Read
Type: Integer (64 bit)
Unit: None

Number of packets received on the command stream from the device since session start.

/dev..../stats/cmdstream/packetssent

Properties: Read
Type: Integer (64 bit)
Unit: None

Number of packets sent on the command stream to the device since session start.

/dev..../stats/cmdstream/pending

Properties: Read
Type: Integer (64 bit)
Unit: None

Number of buffers ready for receiving command packets from the device.

/dev..../stats/cmdstream/processing

Properties: Read
Type: Integer (64 bit)
Unit: None

Number of buffers being processed for command packets. Small values indicate proper performance. For a TCP/IP interface, command packets are sent using the TCP protocol.

/dev..../stats/datastream/bandwidth

Properties: Read
Type: Double
Unit: Mbit/s

Data streaming bandwidth usage on the physical network connection between device and data server.

/dev..../stats/datastream/bytesreceived

Properties: Read
Type: Integer (64 bit)
Unit: B

Number of bytes received on the data stream from the device since session start.

/dev..../stats/datastream/packetslost

Properties: Read
Type: Integer (64 bit)
Unit: None

Number of data packets lost since device start. Data packets contain measurement data.

/dev..../stats/datastream/packetsreceived

Properties: Read
Type: Integer (64 bit)
Unit: None

Number of packets received on the data stream from the device since session start.

/dev..../stats/datastream/pending

Properties:	Read
Type:	Integer (64 bit)
Unit:	None

Number of buffers ready for receiving data packets from the device.

/dev..../stats/datastream/processing

Properties:	Read
Type:	Integer (64 bit)
Unit:	None

Number of buffers being processed for data packets. Small values indicate proper performance. For a TCP/IP interface, data packets are sent using the UDP protocol.

/dev..../stats/physical/currents/n

Properties:	Read
Type:	Double
Unit:	mA

Internal current measurements.

/dev..../stats/physical/fanspeed

Properties:	Read
Type:	Double
Unit:	RPM

Speed of the internal cooling fan.

/dev..../stats/physical/fpga/aux

Properties:	Read
Type:	Double
Unit:	V

Supply voltage of the FPGA.

/dev..../stats/physical/fpga/core

Properties:	Read
Type:	Double
Unit:	V

Core voltage of the FPGA.

/dev..../stats/physical/fpga/temp

Properties:	Read
Type:	Double
Unit:	°C

Internal temperature of the FPGA.

/dev..../stats/physical/overtemperature

Properties: Read
Type: Integer (64 bit)
Unit: None

This flag is set to 1 if the temperature of the FPGA exceeds 85°C. It will be reset to 0 after a restart of the device.

/dev..../stats/physical/temperatures/n

Properties: Read
Type: Double
Unit: °C

Internal temperature measurements.

/dev..../stats/physical/voltages/n

Properties: Read
Type: Double
Unit: V

Internal voltage measurements.

7.2.13. STATUS

/dev..../status/adc0max

Properties: Read
Type: Integer (64 bit)
Unit: None

The maximum value on Signal Input 1 (ADC0) during 100 ms.

/dev..../status/adc0min

Properties: Read
Type: Integer (64 bit)
Unit: None

The minimum value on Signal Input 1 (ADC0) during 100 ms

/dev..../status/adc1max

Properties: Read
Type: Integer (64 bit)
Unit: None

The maximum value on Signal Input 2 (ADC1) during 100 ms.

/dev..../status/adc1min

Properties: Read
Type: Integer (64 bit)
Unit: None

The minimum value on Signal Input 2 (ADC1) during 100 ms

/dev..../status/fifolevel

Properties: Read
Type: Double
Unit: None

USB FIFO level: Indicates the USB FIFO fill level inside the device. When 100%, data is lost

/dev..../status/flags/binary

Properties: Read
Type: Integer (64 bit)
Unit: None

A set of binary flags giving an indication of the state of various parts of the device. Bit 0: Signal Input 1 overflow, Bit 1: Signal Input 2 overflow, Bit 2: Analog PLL fail, Bit 3: Output 1 DAC OK, Bit 4: Output 2 DAC OK, Bit 5: Signal Output 1 clipping, Bit 6: Signal Output 2 clipping, Bit 7: Ext Ref 1 Locked, Bit 8: Ext Ref 2 Locked, Bit 9:Ext Ref 3 Locked, Bit 10:Ext Ref 4 Locked, Bit 11: Sample Loss, Bits 12 - 13: Trigger In 1, Bits 14 - 15: Trigger In 2, Bits 16 - 17: Trigger In 3, Bits 18 - 19: Trigger In 4, Bit 20: PLL 1 locked, Bit 21: PLL 2 locked, Bit 22: PLL 3 locked, Bit 23: PLL 4 locked, Bit 24: Rubidium clock locked, Bit 25: AU Cartesian 1 Overflow, Bit 26: AU Cartesian 2 Overflow, Bit 27: AU Polar 1 Overflow, Bit 28: AU Polar 2 Overflow.

/dev..../status/flags/packetlosstcp

Properties: Read
Type: Integer (64 bit)
Unit: None

Flag indicating if tcp packages have been lost.

/dev..../status/flags/packetlossudp

Properties: Read
Type: Integer (64 bit)
Unit: None

Flag indicating if udp packages have been lost.

/dev..../status/time

Properties: Read
Type: Integer (64 bit)
Unit: None

The current timestamp.

7.2.14. SYSTEM

/dev..../system/activeinterface

Properties: Read
Type: String
Unit: None

Currently active interface of the device.

/dev..../system/awg/channelgrouping

Properties: Read, Write, Setting
Type: Integer (enumerated)
Unit: None

Sets the channel grouping mode of the device.

0	"groups_of_2": Use the outputs in groups of 2. One sequencer program controls 2 outputs (use /dev..../awgs/0..4/).
1	"groups_of_4": Use the outputs in groups of 4. One sequencer program controls 4 outputs (use /dev..../awgs/0/ and /dev..../awgs/2/)
2	"groups_of_8": Use the outputs in groups of 8. One sequencer program controls 8 outputs (use /dev..../awgs/0/). Requires 8 channel device.

/dev..../system/boardrevisions/n

Properties: Read
Type: String
Unit: None

Hardware revision of the FPGA base board

/dev..../system/calib/auto

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Enables an automatic instrument self calibration about 16 min after start up. In order to guarantee the full specification, it is recommended to perform a self calibration after warm-up of the device.

/dev..../system/calib/calibrate

Properties: Read, Write
Type: Integer (64 bit)
Unit: None

Initiates self calibration to improve input digitizer linearity.

/dev..../system/calib/next

Properties: Read
Type: Integer (64 bit)
Unit: s

Remaining seconds until the first calibration is executed or a recalibration is requested.

/dev..../system/calib/required

Properties: Read
Type: Integer (enumerated)
Unit: None

State of device self calibration.

0	Device is warmed-up and self calibrated.
1	It is recommended to manually execute a self calibration to assure operation according to specifications.
2	Device is warming up and will automatically execute a self calibration after 16 minutes.

/dev..../system/calib/tempthreshold

Properties: Read, Write, Setting
Type: Double
Unit: °C

When the temperature changes by the specified amount, it is recommended to rerun the self calibration.

/dev..../system/calib/timeinterval

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: s

Time interval for which the self calibration is valid. After this time it is recommended to rerun the auto calibration.

/dev..../system/compdelay/calibrate

Properties: Read, Write
Type: Integer (64 bit)
Unit: None

Perform automatic calibration of the input delay compensation.

/dev..../system/compdelay/delays/n

Properties: Read, Write
Type: Integer (64 bit)
Unit: None

Current values of the compensation delays. 0: Signal Input 0, 1: Signal Input 1, 2: Aux Inputs, 3: Trigger Inputs, 4: Loopbacks

/dev..../system/extclk

Properties: Read, Write, Setting
Type: Integer (enumerated)
Unit: None

10 MHz reference clock source.

0	"internal": Internal 10 MHz clock is used as the frequency and time base reference.
1	"external": An external 10 MHz clock is used as the frequency and time base reference. Provide a clean and stable 10 MHz reference to the appropriate back panel connector.

/dev..../system/fpgarevision

Properties: Read
Type: Integer (64 bit)
Unit: None

HDL firmware revision.

/dev..../system/fwlog

Properties: Read
Type: String
Unit: None

Returns log output of the firmware.

/dev..../system/fwlogenable

Properties: Read, Write
Type: Integer (64 bit)
Unit: None

Enables logging to the fwlog node.

/dev..../system/fwrevision

Properties: Read
Type: Integer (64 bit)
Unit: None

Revision of the device-internal controller software.

/dev..../system/fx2revision

Properties: Read
Type: String
Unit: None

USB firmware revision.

/dev..../system/identify

Properties: Read, Write
Type: Integer (64 bit)
Unit: None

Setting this node to 1 will cause the device to blink the power led for a few seconds.

/dev..../system/interfacespeed

Properties: Read
Type: String
Unit: None

Speed of the currently active interface (USB only).

/dev..../system/jumbo

Properties: Read, Write
Type: Integer (64 bit)
Unit: None

Enables jumbo frames (4k) on the TCP/IP interface. This will reduce the load on the PC and is required to achieve maximal throughput. Make sure that jumbo frames (4k) are enabled on the network card as well. If one of the devices on the network is not able to work with jumbo frames, the connection will fail.

/dev..../system/kerneltype

Properties:	Read
Type:	String
Unit:	None

Returns the type of the data server kernel (mdk or hpk).

/dev..../system/nics/n/defaultgateway

Properties:	Read, Write
Type:	String
Unit:	None

Default gateway configuration for the network connection.

/dev..../system/nics/n/defaultip4

Properties:	Read, Write
Type:	String
Unit:	None

IPv4 address of the device to use if static IP is enabled.

/dev..../system/nics/n/defaultmask

Properties:	Read, Write
Type:	String
Unit:	None

IPv4 mask in case of static IP.

/dev..../system/nics/n/gateway

Properties:	Read
Type:	String
Unit:	None

Current network gateway.

/dev..../system/nics/n/ip4

Properties:	Read
Type:	String
Unit:	None

Current IPv4 of the device.

/dev..../system/nics/n/jumbo

Properties:	Read, Write
Type:	Integer (64 bit)
Unit:	None

Stored setting for jumbo frames. Will be applied after reboot.

/dev..../system/nics/n/mac

Properties: Read
Type: String
Unit: None

Current MAC address of the device network interface.

/dev..../system/nics/n/mask

Properties: Read
Type: String
Unit: None

Current network mask.

/dev..../system/nics/n/saveip

Properties: Read, Write
Type: Integer (64 bit)
Unit: None

If written, this action will program the defined static IP address to the device.

/dev..../system/nics/n/static

Properties: Read, Write
Type: Integer (64 bit)
Unit: None

Enable this flag if the device is used in a network with fixed IP assignment without a DHCP server.

/dev..../system/owner

Properties: Read
Type: String
Unit: None

Returns the current owner of the device (IP).

/dev..../system/porttcp

Properties: Read, Write
Type: Integer (64 bit)
Unit: None

Returns the current TCP port used for communication to the dataserver.

/dev..../system/portudp

Properties: Read, Write
Type: Integer (64 bit)
Unit: None

Returns the current UDP port used for communication to the dataserver.

/dev..../system/powerconfigdate

Properties:	Read
Type:	Integer (64 bit)
Unit:	None

Contains the date of power configuration (format is: (year << 16) | (month << 8) | day)

/dev..../system/preampenable

Properties:	Read, Write, Setting
Type:	Integer (64 bit)
Unit:	None

Enables the preamplifier.

/dev..../system/preset/busy

Properties:	Read
Type:	Integer (64 bit)
Unit:	None

Indicates if presets are currently loaded.

/dev..../system/preset/default

Properties:	Read, Write
Type:	Integer (enumerated)
Unit:	None

Indicates the preset which is used as default preset at start-up of the device.

0	"factory": Select factory preset as default preset.
1	Select the preset stored in internal flash memory at position 1 as default preset.
2	Select the preset stored in internal flash memory at position 2 as default preset.
3	Select the preset stored in internal flash memory at position 3 as default preset.
4	Select the preset stored in internal flash memory at position 4 as default preset.
5	Select the preset stored in internal flash memory at position 5 as default preset.
6	Select the preset stored in internal flash memory at position 6 as default preset.

/dev..../system/preset/erase

Properties:	Read, Write
Type:	Integer (64 bit)
Unit:	None

Erase the selected preset.

/dev..../system/preset/error

Properties:	Read
Type:	Integer (64 bit)
Unit:	None

Indicates if the last operation was illegal. Successful: 0, Error: 1.

/dev..../system/preset/index

Properties:	Read, Write
Type:	Integer (enumerated)
Unit:	None

Select between factory preset or presets stored in internal flash memory.

0	"factory": Select factory preset.
1	Select the preset stored in internal flash memory at position 1.
2	Select the preset stored in internal flash memory at position 2.
3	Select the preset stored in internal flash memory at position 3.
4	Select the preset stored in internal flash memory at position 4.
5	Select the preset stored in internal flash memory at position 5.
6	Select the preset stored in internal flash memory at position 6.

/dev..../system/preset/load

Properties:	Read, Write
Type:	Integer (64 bit)
Unit:	None

Load the selected preset.

/dev..../system/preset/records/n/features

Properties:	Read
Type:	Integer (64 bit)
Unit:	None

Properties of the preset.

/dev..../system/preset/records/n/label

Properties:	Read, Write
Type:	String
Unit:	None

Name of this preset.

/dev..../system/preset/records/n/timestamp

Properties:	Read, Write
Type:	Integer (64 bit)
Unit:	None

Not used.

/dev..../system/preset/records/n/valid

Properties:	Read
Type:	Integer (64 bit)
Unit:	None

True if a valid preset is stored.

/dev..../system/preset/save

Properties:	Read, Write
Type:	Integer (64 bit)
Unit:	None

Save the actual setting as preset.

/dev..../system/properties/freqresolution

Properties:	Read
Type:	Integer (64 bit)
Unit:	None

The number of bits used to represent a frequency.

/dev..../system/properties/freqscaling

Properties:	Read
Type:	Double
Unit:	None

The scale factor to use to convert a frequency represented as a freqresolution-bit integer to a floating point value.

/dev..../system/properties/maxfreq

Properties:	Read
Type:	Double
Unit:	None

The maximum oscillator frequency that can be set.

/dev..../system/properties/maxtimeconstant

Properties:	Read
Type:	Double
Unit:	s

The maximum demodulator time constant that can be set. Only relevant for lock-in amplifiers.

/dev..../system/properties/minfreq

Properties:	Read
Type:	Double
Unit:	None

The minimum oscillator frequency that can be set.

/dev..../system/properties/mintimeconstant

Properties:	Read
Type:	Double
Unit:	s

The minimum demodulator time constant that can be set. Only relevant for lock-in amplifiers.

/dev..../system/properties/negativefreq

Properties: Read
Type: Integer (64 bit)
Unit: None

Indicates whether negative frequencies are supported.

/dev..../system/properties/timebase

Properties: Read
Type: Double
Unit: s

Minimal time difference between two timestamps. The value is equal to 1/(maximum sampling rate).

/dev..../system/saveports

Properties: Read, Write
Type: Integer (64 bit)
Unit: None

Flag indicating that the TCP and UDP ports should be saved.

/dev..../system/stall

Properties: Read, Write
Type: Integer (64 bit)
Unit: None

Indicates if the network connection is stalled.

/dev..../system/update

Properties: Read, Write
Type: Integer (64 bit)
Unit: None

Requests update of the device firmware and bitstream from the dataserver.

/dev..../system/xenpakenable

Properties: Read
Type: Integer (64 bit)
Unit: None

Indicates whether the 10 gigabit ethernet option is installed and enabled.

7.2.15. TRIGGERS

/dev..../triggers/in/n/autothreshold

Properties: Read, Write
Type: Integer (64 bit)
Unit: None

Automatically adjust the trigger threshold. The level is adjusted to fall in the center of the applied transitions.

/dev..../triggers/in/n/imp50

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

Trigger input impedance: When on, the trigger input impedance is 50 Ohm, when off 1 k Ohm.

/dev..../triggers/in/n/level

Properties: Read, Write, Setting
Type: Double
Unit: V

Trigger voltage level at which the trigger input toggles between low and high. Use 50% amplitude for digital input and consider the trigger hysteresis.

/dev..../triggers/out/n/delay

Properties: Read, Write, Setting
Type: Double
Unit: s

Trigger delay, controls the fine delay of the trigger output. The resolution is 78 ps.

/dev..../triggers/out/n/drive

Properties: Read, Write, Setting
Type: Integer (64 bit)
Unit: None

When on, the bidirectional trigger on the front panel is in output mode. When off, the trigger is in input mode.

/dev..../triggers/out/n/pulsewidth

Properties: Read, Write, Setting
Type: Double
Unit: s

Defines the minimal pulse width for the case of Scope and AWG Trigger/Active events written to the trigger outputs of the device.

/dev..../triggers/out/n/source

Properties:	Read, Write, Setting
Type:	Integer (enumerated)
Unit:	None

Select the signal assigned to the trigger output.

0	"disabled": The output trigger is disabled.
1	"demod3_phase": Oscillator phase of demod 4 (trigger output channel 1) or demod 8 (trigger output channel 2). Trigger event is output for each zero crossing of the oscillator phase.
2	"scope_trigger": Scope Trigger. Requires the DIG Option.
3	"scope_not_trigger": Scope /Trigger. Requires the DIG Option.
4	"scope_armed": Scope Armed. Requires the DIG Option.
5	"scope_not_armed": Scope /Armed. Requires the DIG Option.
6	"scope_active": Scope Active. Requires the DIG Option.
7	"scope_not_active": Scope /Active. Requires the DIG Option.
8	"awg_marker0": AWG Marker 1. Requires the AWG Option.
9	"awg_marker1": AWG Marker 2. Requires the AWG Option.
10	"awg_marker2": AWG Marker 3. Requires the AWG Option.
11	"awg_marker3": AWG Marker 4. Requires the AWG Option.
20	"awg_active": AWG Active. Requires the AWG Option.
21	"awg_waiting": AWG Waiting. Requires the AWG Option.
22	"awg_fetching": AWG Fetching. Requires the AWG Option.
23	"awg_playing": AWG Playing. Requires the AWG Option.
32	"awg_trigger0": AWG Trigger 1. Requires the AWG Option.
33	"awg_trigger1": AWG Trigger 2. Requires the AWG Option.
34	"awg_trigger2": AWG Trigger 3. Requires the AWG Option.
35	"awg_trigger3": AWG Trigger 4. Requires the AWG Option.
51	"mds_clock_out": MDS Clock Out.
52	"mds_sync_out": MDS Sync Out.