HDAWG User Manual

Zurich
Instruments

HDAWG User Manual
Zurich Instruments AG

Revision 20.07.0

Revision History

Revision 20.07.0, 20-Aug-2020:

The document was updated to comply with the changes of the 20.07 LabOne release.

Highlights of the changes and additions to the HDAWG product are:

- NEW accessory HDIQ IQ Modulator: 4 RF channels, RF frequency range 4-8 GHz, switchable signal routing for automated mixer calibration
- AWG: New FIFO memory architecture for reliable playback of long waveforms from main memory
- AWG: New command table feature for memory-efficient sequencing with real-time amplitude/phase changes
- AWG: Real-time pseudo-random number generation feature
- AWG: New sequencer instruction `playZero` improving the generation of gaps in signals
- AWG: New sequencer instructions `placeholder` and `assignWaveIndex` removing the need to upload unneccessary waveform data when using direct waveform upload from API
- AWG: Discontinued `playWaveIndexed` sequencer instruction due to incompatibility with FIFO memory architecture
- 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` and `getZSyncData` for triggering and data communication with PQSC over ZSync
- Specifications: doubled sequencer instruction memory to 16,384 instructions per AWG core

A more detailed list of all technical changes can be found in the LabOne release notes.

Revision 20.01.0, 28-Feb-2020:

The document was updated to comply with the changes of the 20.01 LabOne release.

Highlights of the changes and additions to the HDAWG product are:

- AWG: new dual-phase digital IQ modulation at full sampling rate
- AWG Sequencer: new instructions `resetOscPhase`, `setSinePhase`, and `incrementSinePhase` for real-time control of digital oscillators
- AWG: improved stability and predictability of the sequencer
- AWG: consistent alignment of waveform outputs in grouped channel mode in combination with with external triggering
- Precompensation Simulator: CSV import of step response measurement signal
- LabOne: improved stability of AWG Compiler
- LabOne: reduced memory footprint to improve responsiveness when multiple instruments are connected
- Math: added linear fit to the Sweeper and DAQ tabs

- LabOne: saving of histogram data
- LabOne: improved compatibility of saved SVG figures with popular vector graphics editors
- NEW option HDAWG-SKW Output Skew Control: 10 ps resolution skew control of Wave signal outputs with 10 ps resolution. HDAWG instruments purchased in 2019 or earlier are equivalent to the HDAWG with the HDAWG-SKW option purchased from 2020 onwards.
- Specifications: added Wave output level accuracy and resolution, offset voltage accuracy, RMS voltage noise, overshoot.
- Specifications: added Marker output rise time, jitter, skew control range/resolution.
- Specifications: added reference clock output jitter and clock initial accuarcy.
- Specifications: added trigger input threshold hysteresis, min pulse width, input impedance.
- Specifications: direct marker output on DIO connector not supported and functionality replaced by `setDIO` instruction with improved timing control when issued at high rate.

A more detailed list of all technical changes can be found in the LabOne release notes.

Revision 64000, 12-Aug-2019:

The document was updated to comply with the changes of the 19.05 LabOne release.

Highlights of the changes and additions to the HDAWG product are:

- NEW option HDAWG-PC Real-time Precompensation: configurable digital filter for each Wave output to minimize effects of external wiring on the signal
- AWG Sequencer: new instruction `waitSineOscPhase` to align waveform playback with oscillators
- AWG Sequencer: improved compilation and uplodad speed
- AWG Sequencer: the waveform viewer now supports waveforms up to 10 MSa length.
- AWG Sequencer: the sequencer program memory has been increased to 8192 hardware instructions and restricted to the cache memory
- Outputs: Marker output delay is now adjustable
- Reference clock: faster external clock locking
- LabOne: macOS support
- LabOne: plots can be saved in PNG or JPEG format
- LabOne: support of HDF5 file 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: minimum base sample clock changed to 100 MHz (previously 50 MHz). Lower sampling rates can be achieved using sampling rate division.
- Specifications: added documentation of DIOLink as high-speed interface to external controllers for dynamic sequencing
- Specifications: added Wave output worst harmonic component, second and third and harmonic distortion
- Specifications: added output period jitter
- Specifications: added performance diagram for Wave output phase noise

A more detailed list of all technical changes can be found in the LabOne release notes.

Revision 58300, 20-Dec-2018:

The document was updated to comply with the changes of the 18.12 LabOne release.

Highlights of the changes and additions to the HDAWG product are:

- USB 3.0 interface support
- Improved alignment of marker and waveform output signals

- AWG Sequencer: new instruction `playWaveDigTrigger` to achieve 50 ns trigger delay to output
- AWG Sequencer: new instructions `prefetch` and `prefetchIndexed` for improved control over waveform cache memory management
- AWG Sequencer: expanded random waveform generation with new instructions `randomSeed`, `randomGauss`, and `randomUniform`
- AWG Sequencer: improved speed and stability for large pattern upload
- MDS: support of multiple instruments in the same LabOne UI session
- MDS: visual device identification via front panel LEDs
- Manual check for LabOne update in startup screen
- Chinese LabOne tooltips
- Specifications: added damage thresholds for all connectors, reduced damage threshold of Reference and Sample Clk In/Out to ±4 V (previously ±5 V)
- Specifications: Wave output phase noise (configuration 100 MHz, offset 10 kHz) changed to −135 dBc/Hz (previously −130 dBc/Hz), added phase noise −148 dBc/Hz (configuration 100 MHz, offset 1 MHz)
- Specifications: added Wave output spurious free dynamic range 80 dBc
- Specifications: added bandwidth 80 MHz of switchable Wave output filter
- Specifications: changed Wave output rise time for 1 V step to 800 ps (previously 550 ps), added rise time 450 ps for 0.2 V, and 1100 ps for 5 V step
- Specifications: changed maximum trigger delay to 180 ns (previously 270 ns), minimum trigger delay using `playWaveDigTrigger` instruction is 50 ns

A more detailed list of all technical changes can be found in the LabOne release notes.

Revision 54600, 14-Aug-2018:

Second revision related to LabOne release 18.05 with minor corrections.

Revision 53400, 17-Jun-2018:

First revision of the HDAWG user manual related to LabOne release 18.05.

Table of Contents

Declaration of Conformity

The manufacturer

Zurich Instruments
Technoparkstrasse 1
8005 Zurich
Switzerland

declares that the product

HDAWG Arbitrary Waveform Generator, 750 MHz, 2.4 GSa/s

fulfills the requirements of the European guidelines

- 2004/108/EC Electromagnetic Compatibility
- 2006/95/EC Low Voltage
- 2011/65/EU Restriction of Hazardous Substances (RoHS)
- 1907/2006/EC Registration, Evaluation, Authorization, and Restriction of Chemicals (REACH)

The assessment was performed using the directives according to Table 1.

Table 1. Conformity table

EN 61326-1:2012 (ed.2.0)	Electrical equipment for measurement, control and laboratory use - EMC requirements - Part 1: General requirements
CISPR 11:2016-06 (EN 55011)	Industrial, scientific and medical (ISM) radio-frequency equipment – Electromagnetic disturbance characteristics – Limits and methods of measurement
EN 61010-1:2010 IEC61010-1:2010 + AMD1:2016 IEC61010-1 National Deviations for EU, US, CA	Safety requirements for electrical equipment for measurement, control and laboratory use

CE

Chapter 1. Getting Started

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

- A Quick Start Guide for the impatient
- Inspecting the package content and accessories
- List of essential handling and safety instructions
- Connecting to the HDAWG Instrument
- Handy list of troubleshooting guidelines

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

1.1. Quick Start Guide

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

1. Inspect 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 Chapter 1.

2. Check the Handling and Safety Instructions in Section 1.3.

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 Section 1.4.

4. Connect the Instrument to the power line. Turn it on and connect it to a switch in the LAN using the Ethernet cable.

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.
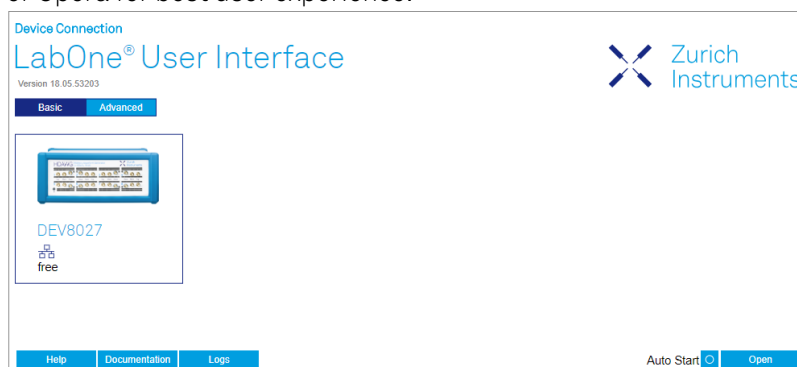
6. The LabOne User Interface start-up screen will appear. Click the **Open** button on the lower right of the page. The default configuration will be loaded and the first signals can be generated. If the user interface does not start up successfully, please refer to Section 1.5.

If any problems occur whilst setting up the instrument and software please see the troubleshooting section at the end of this chapter. After use it is recommended to shut down the instrument using the soft power button on the front panel instrument or by clicking on the ⏻ button at the bottom left of the user interface screen before turning the power switch on the back panel of the instrument.

Once the Instrument is up and running we recommend going through some of the tutorials given in Chapter 3. Chapter 4 provides a general introduction to the various tools and settings tabs with tables in each section describing every UI element. For specific application know-how, the Blog section of the Zurich Instruments web page will serve as a valuable resource that is constantly updated and expanded.

1.2. 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 HDAWG Instrument

- You have received 1 power cord with a power plug suited to your country

- You have received 1 USB 3.0 cable and/or 1 LAN cable (category 5/6 required)

- You have received a printed version of the "Getting Started" section

- The "Next Calibration"   sticker on the rear panel of the Instrument indicates approximately 2 years ahead in time. Zurich Instruments recommends calibration intervals of 2 years

- The MAC address of the instrument is displayed on a sticker on the back panel

Table 1.1. Package contents for the HDAWG Instrument

the power cord (e.g. EU norm)	the USB 3.0 cable
the power inlet, with power switch and fuse holder	the LAN / Ethernet cable (category 5/6 required)
Next Calibration Apr 2022 the "Next Calibration" sticker on the back panel of the instrument	S/N HD-DEV8024 MAC 80:2F:DE:00:07:EA the MAC address sticker on the back panel of the instrument

The HDAWG 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 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 found in the specifications chapter.

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

1.3. Handling and Safety Instructions

The HDAWG Instrument is a sensitive piece of electronic equipment, which under no circumstances should 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 cancels 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 affect correct 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 1.2. Safety Instructions

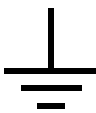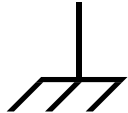|	
Ground the instrument	The instrument chassis must be correctly connected to earth ground by means of the supplied power cord. The ground pin of the power cord set plug must be firmly connected to the electrical ground (safety ground) terminal at the mains power outlet. Interruption of the protective 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.
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 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, 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 the correct interpretation of 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 . Do not block the ventilator opening on the back or the air intake on the chassis side and front 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 accessed 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	The RJ45 socket on the back panel labeled "ZSync" 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 at the instrument outside the ambient conditions specified in .
Handling	Do not drop the Instrument, handle with due care, do not store liquids on the device as there is a chance of spilling and 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.

When 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 by email at `<support@zhinst.com>`.

Table 1.3. 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 only with a power cord specified for this product and certified for the country of use.
Instrument emits abnormal noise, smell, or sparks	Switch off the Instrument immediately to prevent large damage.
Instrument is damaged	Switch off the Instrument immediately and secure it against unintended operation.

Table 1.4. Symbols

	Earth ground
	Chassis ground
	Caution. Refer to accompanying documentation

1.4. Software Installation

The HDAWG Instrument is operated from a host computer with the LabOne software. To install the LabOne software on a PC, administrator rights are 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 Section 1.6.

1.4.1. Installing LabOne on Windows

The installation packages for Zurich Instruments LabOne software are available as Windows installer .msi packages. The software is available on the Zurich Instruments download page, www.zhinst.com/downloads. Please ensure that you have administrator rights for the PC where the software is to be installed and that you download the correct software installer for the PC's processor architecture (32-bit or 64-bit), for help see the section called "Determining PC Architecture on Microsoft Windows". See www.zhinst.com/labone/compatibility for a comprehensive list of supported Windows systems.

Determining PC Architecture on Microsoft Windows

In case you are unsure which Windows architecture you are using, it can be checked as follows:

- Windows 7: `Control panel → System and Security → System/System type`
- Windows 8: `Control panel → System → System/System type`
- Windows 10: `Settings → System → About/System type`
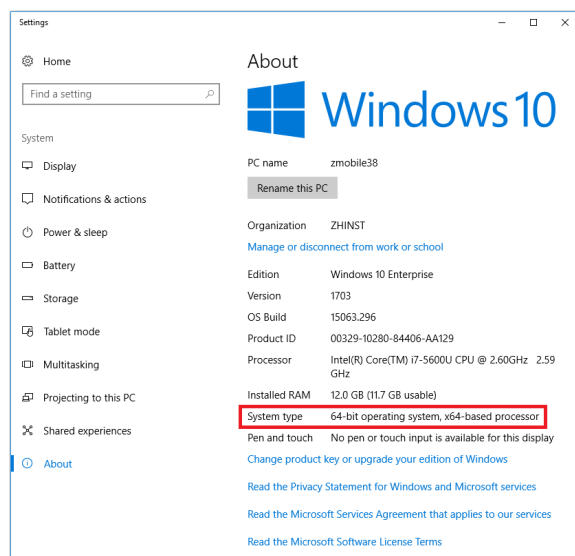
Figure 1.1. Find out the OS addressing architecture (32-bit or 64-bit)

Windows LabOne Installation

1. The HDAWG 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 `LabOne32/64-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.
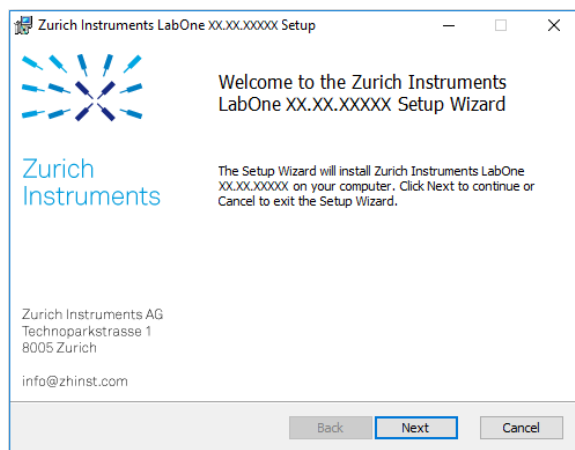
Figure 1.2. Installation welcome screen

- 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 HDAWG Instrument the "HDAWG Series Device, Web Server" and "API" features are required. Please install the features for other device classes as well as required. If you would like to install shortcuts on your desktop area enable the feature "Desktop Shortcuts". To proceed click the **Next** button.
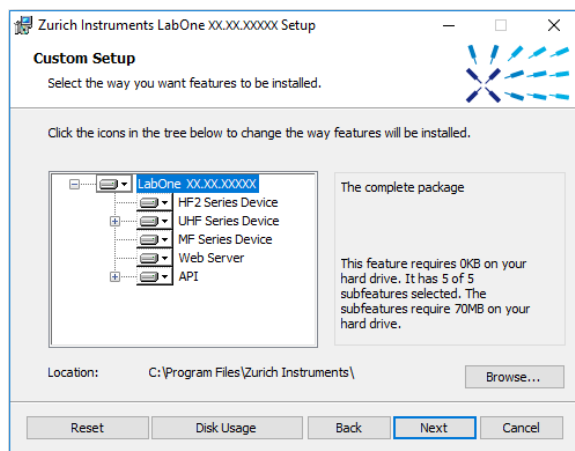
Figure 1.3. Custom setup screen

- Select whether the software should periodically check for updates. The software will not update automatically even with enabled periodic check for updates. This setting can later be changed in the user interface. To proceed click the **Next** button.
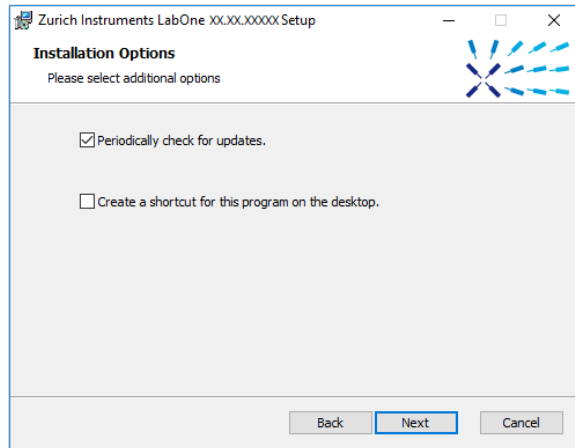
Figure 1.4. Automatic update check

- Click the **Install** button to start the installation process.

- Windows will ask up to two times to reboot the computer. Make sure you have no unsaved work on your computer. Actually a reboot is practically never required, so that one may safely click **OK**.
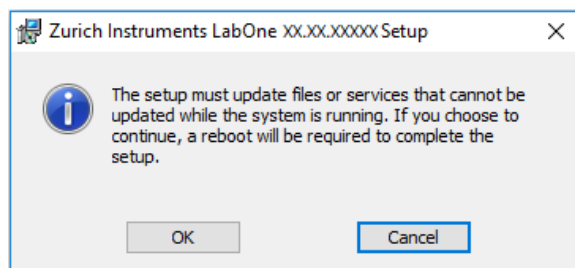
Figure 1.5. Installation reboot request

- On Windows Server 2008 and Windows 7 it is required to confirm the installation of up to 2 drivers from the trusted publisher Zurich Instruments. Click on **Install**.
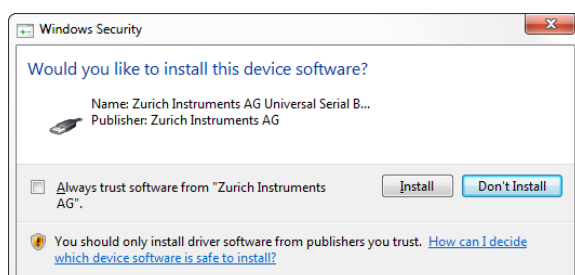
Figure 1.6. Installation driver acceptance

- Click **OK** on the following notification dialog.
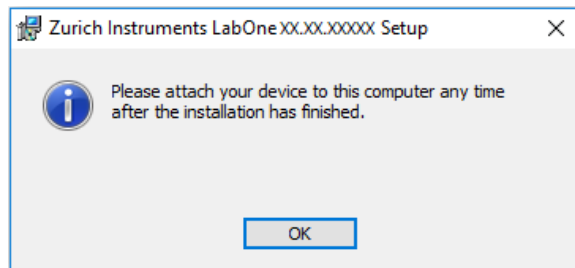
Figure 1.7. Installation completion screen

3. Click **Finish** to close the Zurich Instruments LabOne installer.

4. You can now start the LabOne User Interface as described in Section 1.5.2 and choose an instrument to connect to via the Device Connection dialog shown in Figure 1.10.

Warning

Do not install drivers from another source and therefore not trusted as originating from Zurich Instruments.

1.4.2. Installing LabOne on Linux

Requirements

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

1. Officially, Ubuntu 14.04 LTS and 16.04 LTS (amd64) are supported although in practice LabOne software may work on other platforms. Please ensure that you are using a Linux distribution that is compatible with Ubuntu/Debian.

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 downloads page:

    - LabOneLinux<arch>-<release>.<revision>.tar.gz, for example:

        ```
        LabOneLinux32/64-16.12.41721.tar.gz
        ```

    Please ensure you download the correct architecture (32-bit/64-bit) 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 " `x686`" the 32-bit version of the LabOne package is required, if it displays " `x86_64`" the 64-bit 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 Web Server program at a command prompt:

```
$ ziWebServer
```

2. 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.

3. You can now start the LabOne User Interface as described in Section 1.5.2 and choose an instrument to connect to via the Device Connection dialog shown in Figure 1.10.

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
```

1.4.3. Start LabOne Manually on the Command Line

After installing the LabOne software, the Web and Data Server can be started manually using the command-line. The more common way to start LabOne under windows is described in Section 1.5.2. The advantage of using the command line is being able to observe and change the behaviour of the Web and Data Server. 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 and Data Server. They are installed in the Program Files Folder (usually: C:\Program Files) under \Zurich Instruments \LabOne in the WebServer or DataServer folder respectively. The Web and Data Server (ziDataServer) is started by running the respective executable in each folder. Please be aware that only one instance of each Server may run at a time per computer. The behaviour of the Server can be changed by providing command line arguments. For the Web Server the following arguments are possible:

```
Options:
  -h [ --help ]              Produce help message
  --ip arg (=0.0.0.0)        IP address to which the webserver should listen
  --port arg                 The ports on which the webserver should listen,
                             default is 8006 if none is specified
  -w [ --websocket ] arg (=1)  Enable WebSocket use, if available, for data
                             streaming to the browser. Default is on, will
                             use AJAX if switched off.
  --discovery arg (=1)       Enable discovery based session dialog. Default
                             is on.
  --multicast-hops arg (=1)  Set number of IP multicast hops
  --server-ip arg (=127.0.0.1) IP address of the ziDataServer
  --server-port arg (=8004)  The port on of the ziDataServer
  -r [ --resource-path ] arg  Home directory for the web server (read access
                             rights)
  -d [ --data-path ] arg     Data directory for the web server (write access
                             rights)
  -s [ --setting-path ] arg  Setting directory for the web server (write
                             access rights)
  -l [ --log-path ] arg      Log directory for the web server (write access
                             rights)
  -D [ --doc-path ] arg      Documentation directory for the web server (read
                             access rights)
  --firmware-path arg        Directory containing device firmware (read
                             access rights)
  --api-log arg (=268435711)  API command log mask
  -a [ --auto-start ] arg (=0) Start browser page automatically
  --debug arg (=3)           Set the debug level (trace:0, info:1, debug:2,
                             warning:3, error:4, fatal:5, status:6)
  --api-level arg (=6)       Stick to the given ziAPI version: 1 = ziAPI v1;
                             4 = ziAPI v4; 5 = ziAPI v5; 6 = ziAPI v6
                             (default)
  --hide-console arg (=0)    Start process without console window.
  --wait-on-exit arg (=0)    Wait for key press on exit of the server
  -i [ --info ] arg          Output requested information and exit, available
                             arguments are: product, version, revision
  --dir-watching arg (=1)    If true (default), enable monitoring of changes
                             to directories and files used by LabOne. Affects
                             the Config and File Manager tabs.
```

For the Data Server the following arguments are possible:

```
Will log to directory '/tmp/ziDataServerLog_danielw'
Options:
  -h [ --help ]              Produce help message
  --port arg (=8004)         The port on which the server should listen
  --open-override arg (=0)   Start the server listening on connections from
                             outside
  --debug arg (=3)           Set the debug level (trace:0, info:1, debug:2,
                             warning:3, error:4, fatal:5, status:6)
  --device-ip arg            Set static IP for the device, or specify 'usb' to
                             force connection via USB
  --device-port arg (=8010)  Set TCP/IP port for the device
  --interface-ip arg (=1)    Enables devices search on ip via multicast
  --interface-usb arg (=1)   Enables devices search on usb
  --interface-pcie arg (=0)  Enables devices search on PCIe
  --auto-connect arg (=1)    Auto-connects a device if on USB or PCIe.
```

```
--discovery arg (=1)        Enable UDP multicast data server discovery.
                            Default is on.
--multicast-hops arg (=1)   Set number of IP multicast hops
--device-serial arg         Connects to a given device serial (devnnn). Will
                            automatically detect if not specified.
--buffer-size arg (=65536)  Session buffer size (>=32768)
--firmware-path arg         Directory containing device firmware (read access
                            rights)
-l [ --log-path ] arg       Log directory for the server (write access rights)
--wait-on-exit arg (=0)     Wait for key press on exit of the server
--hide-console arg (=0)     Start process without console window.
-i [ --info ] arg           Output requested information and exit, available
                            arguments are: product, version, revision
```

1.5. Connecting to the Instrument

The Zurich Instruments HDAWG 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)

Chrome	Firefox	Opera	Edge	Safari

In order to physically connect to the instrument, integrate the instrument 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 device in the same LAN. 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 an RNDIS driver on the host computer. For PC users, this driver included in the LabOne software installer. The driver is available online for Mac users.

1.5.1. 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 1.8. 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. This 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.
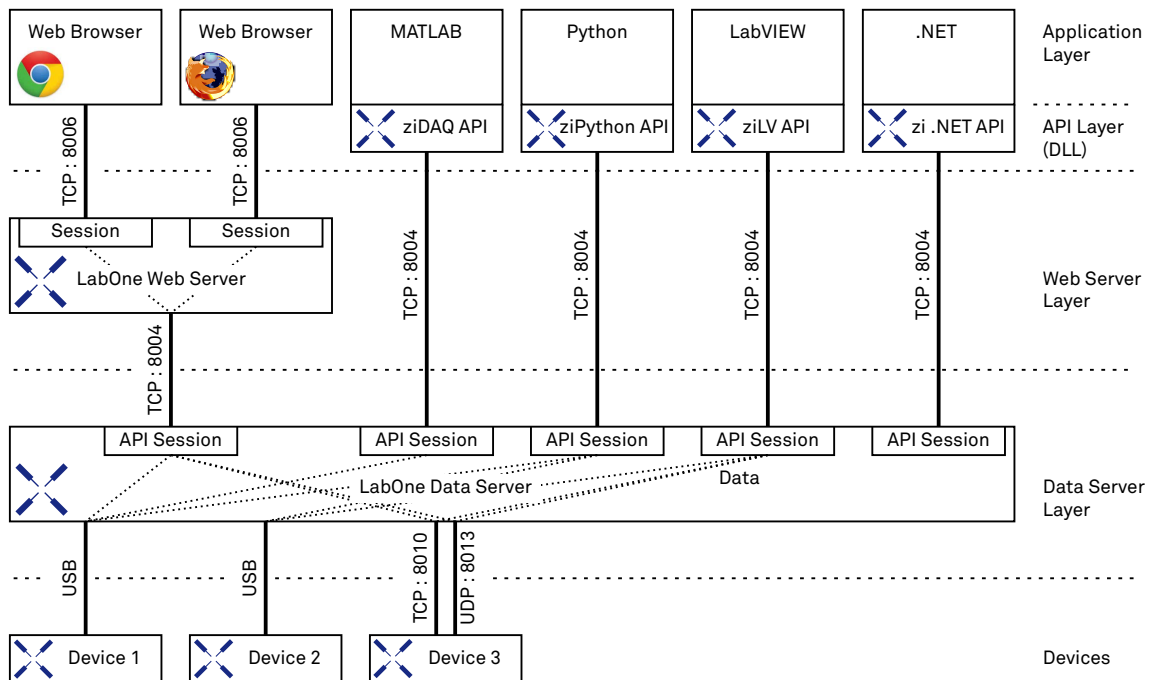
Figure 1.8. Software architecture

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. 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. That is to say 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.

1.5.2. LabOne Software Start-up

This section describes the start-up of the LabOne User Interface which is used to control the HDAWG Instrument. If the LabOne software is not yet installed on the PC please follow the instructions in Section 1.4 Software Installation. If the device is not yet connected please find more information in Section 1.5 Device Connectivity.

The LabOne User Interface start-up link can be found under the Windows 10 Start Menu[1] as shown in Figure 1.9: 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 Section 1.5.1.
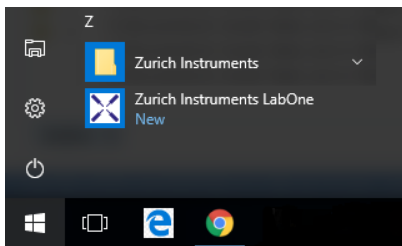
Figure 1.9. Link to the LabOne User Interface in the Windows 10 Start Menu

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 run the LabOne User Interface in Chrome

---

[1]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`.

in application mode which improves the user experience by removing the unnecessary browser controls.

After starting LabOne, the Device Connection dialog in Figure 1.10 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 Section 1.5.1.
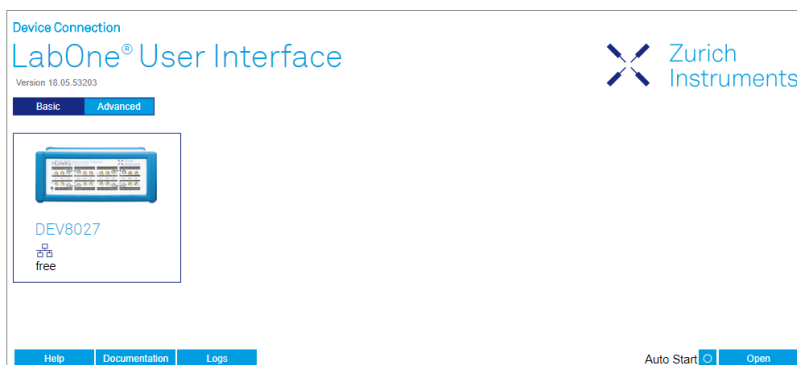
Figure 1.10. Device Connection dialog

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 which are particularly useful for multi-instrument setups.

The Advanced view consists of three parts: Data Server Connectivity, Available Devices, and 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, also connections via Data Servers running on other PCs in the network 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**
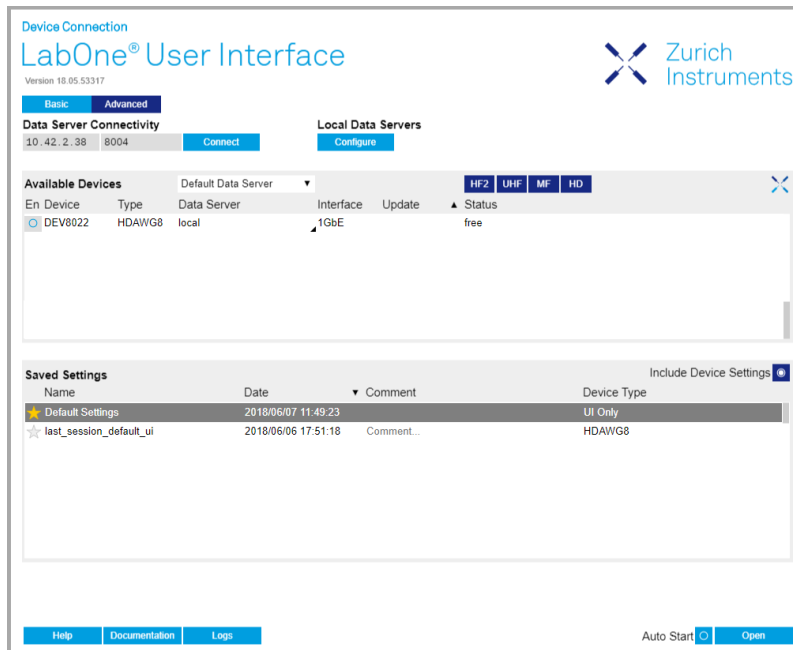
Figure 1.11. Device Connection dialog (Advanced view)

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, HD) are used in parallel, the list of 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 HDAWG 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 where 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, or HDAWG 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", cf. Section 4.11.

On the remote computer (Computer 2), open the Device Connection dialog by starting up the LabOne User Interface. 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 of 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. The interface is listed if it is physically connected. For MF series instruments the interface is indicated as PCIe in case the Data Server is running on the instrument, even if the physical connection between PC and instrument is USB or 1GbE. PCIe corresponds to the interface between the embedded PC and the measurement unit inside the MF instrument. 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.

The last column indicates the status of the device. Table 1.5 explains the meaning of some of the possible device statuses.

Table 1.5. 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 Section 1.6.2.
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 1.6. 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 1.6. 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 .xml.
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 1.7.

Table 1.7. 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 the 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. Via a right click 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 e.g. useful when setting up a connection to multiple instruments.
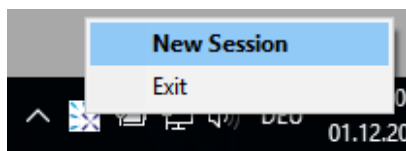
Figure 1.12. LabOne Tray Icon in Windows 10

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 1.10. The figure below shows an example of the Connection Lost dialog.
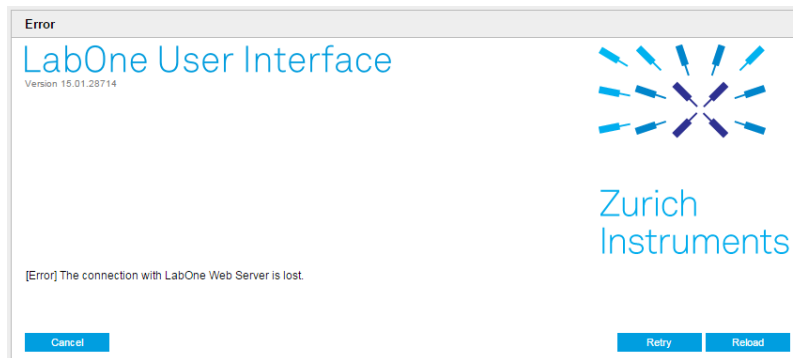
Figure 1.13. Dialog: Connection Lost

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 1.10. 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.
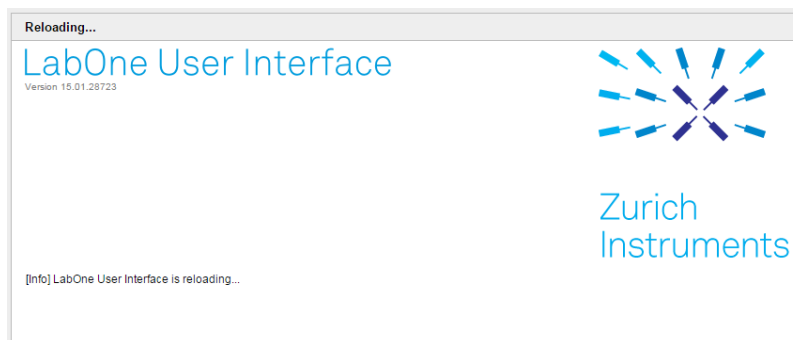
Figure 1.14. Dialog: Reloading

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 Section 1.5. The figure below shows an example of this dialog.
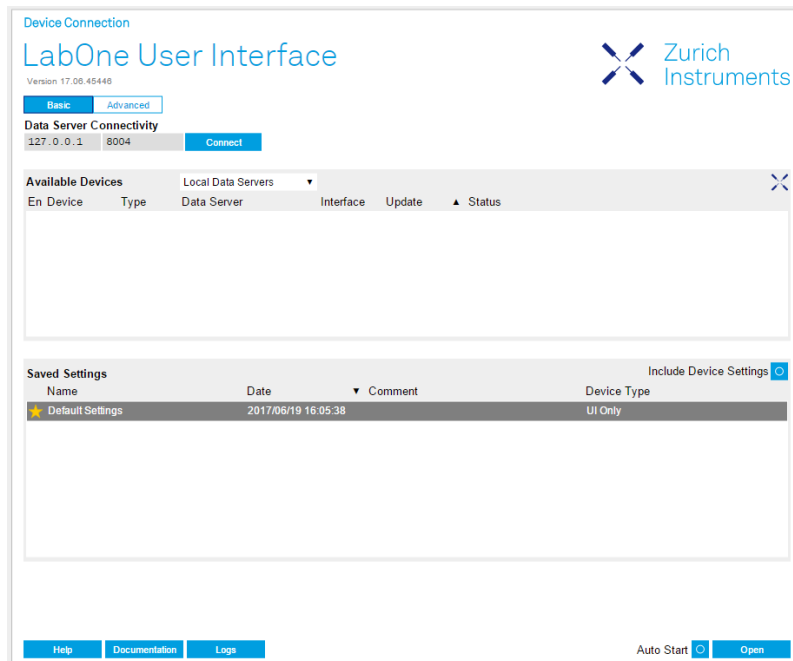
Figure 1.15. No Device Discovered

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 Section 1.6.2. If all the devices are in use, access is not possible until a connection is relinquished by the another Data Server.

1.5.3. Visibility and Connection

There are several ways to connect the instrument 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 an instrument is connected to a network, it can be accessed from multiple host computers. To manage the access to the instrument, there are two different connectivity states: visible and connected. It is important to distinguish if an instrument is just physically connected over 1GbE or actively controlled by the LabOne Data Server. In the first case the instrument is visible to the LabOne Data Server. In the second case the instrument is logically connected.
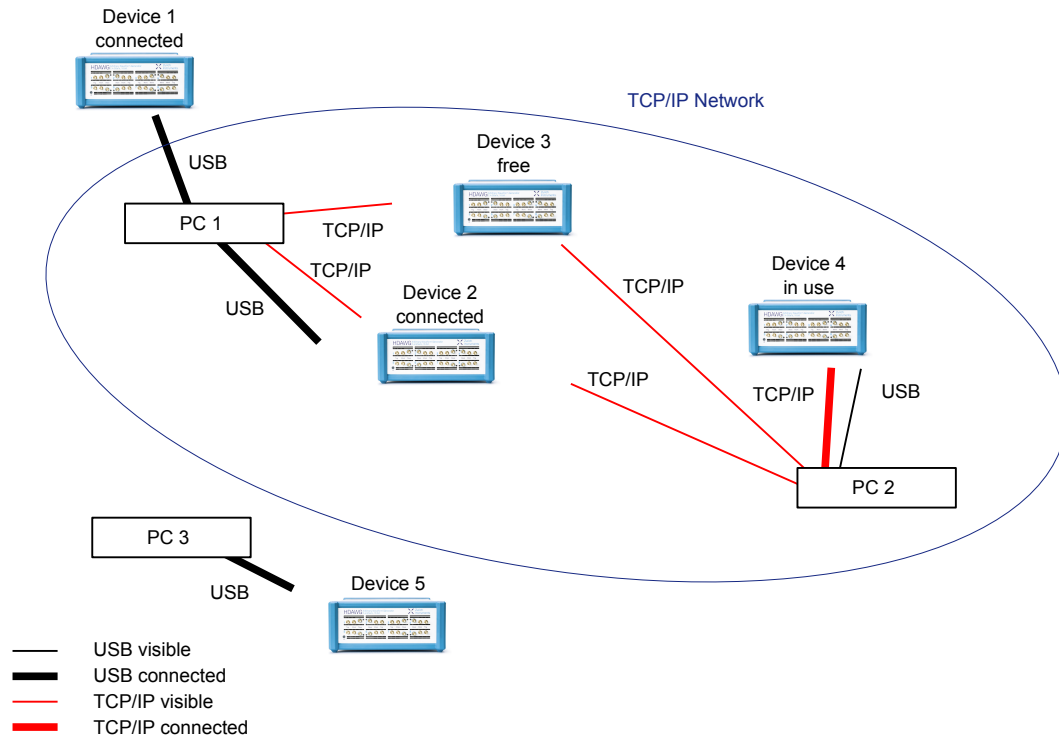
Figure 1.16. Connectivity

Figure 1.16 shows some examples of possible configurations of computer-to-instrument 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 Instruments

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

Connected Instrument

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

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

1.5.4. 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 instrument is included in the LabOne Installer package (LabOne 18.12 and later). Ensure that the instrument 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`

An instrument 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 1.17.
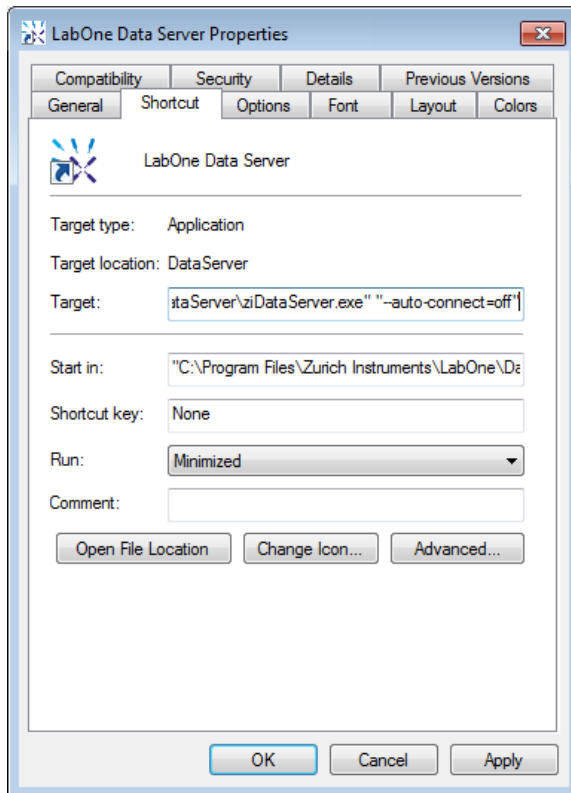
Figure 1.17. auto-connect

1.5.5. 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.

Multicast DHCP

The most straightforward TCP/IP connection method is to rely on a network configuration to recognize the 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 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 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 internet connectivity, the other can be used for connecting to the instrument. Alternatively, internet connectivity can be established via wireless LAN.

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 instrument connectivity directly to the 1GbE port of the 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 Figure 1.18 (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 in the following section.
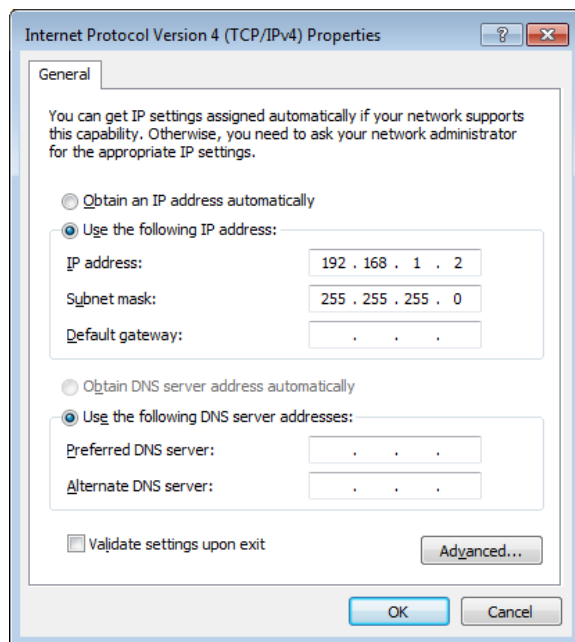
Figure 1.18. Static IP configuration for the host computer

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 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.
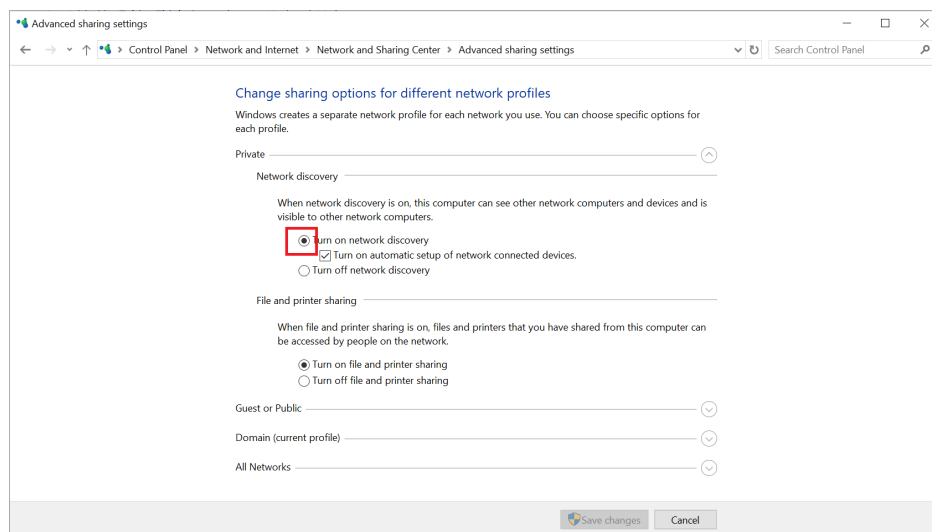
Figure 1.19. Turn on network discovery for Private P2P connection

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.
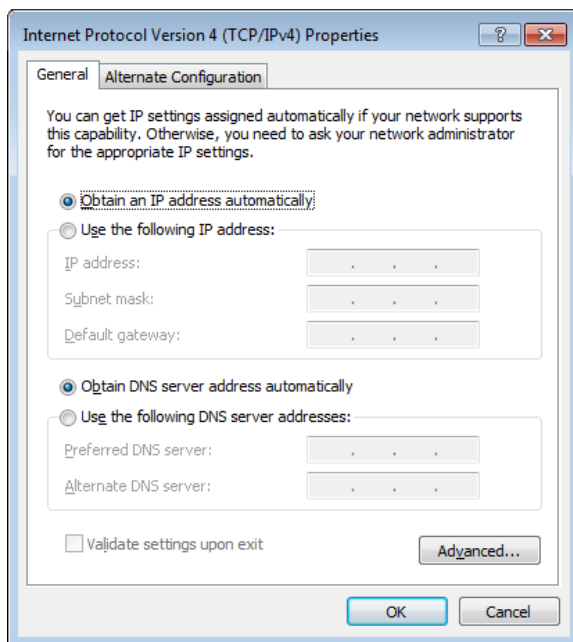
Figure 1.20. Dynamic IP configuration for the host computer

Static Device IP

Using a static IP address for the host computer is necessary to set up a point-to-point network. On top of that, a static device IP configuration can be necessary in the rare cases in which the network card does not support multicast.

1. Connect the 1GbE port of the network card that is dedicated for device connectivity directly to the 1GbE port of the instrument.

2. In the Device tab of the LabOne user interface, enable the setting "Static IP" with the IP Address 192.168.1.10, click on "Program", and restart the instrument using the soft power button

3. 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 field: `--device-ip 192.168.1.10`.

   The LabOne User Interface shortcut Target field should look like this:

   ```
   "C:\Program Files\Zurich Instruments\LabOne\WebServer\ziWebServer.exe"
   --auto-start=1  --server-port=8004  --resource-path "C:\Program  Files
   \Zurich Instruments\LabOne\WebServer\html\\" --device-ip 192.168.1.10
   ```

   The LabOne Data Server shortcut Target field should look like this:

   ```
   "C:\Program       Files\Zurich       Instruments\LabOne\DataServer
   \ziDataServer.exe" --device-ip 192.168.1.10
   ```
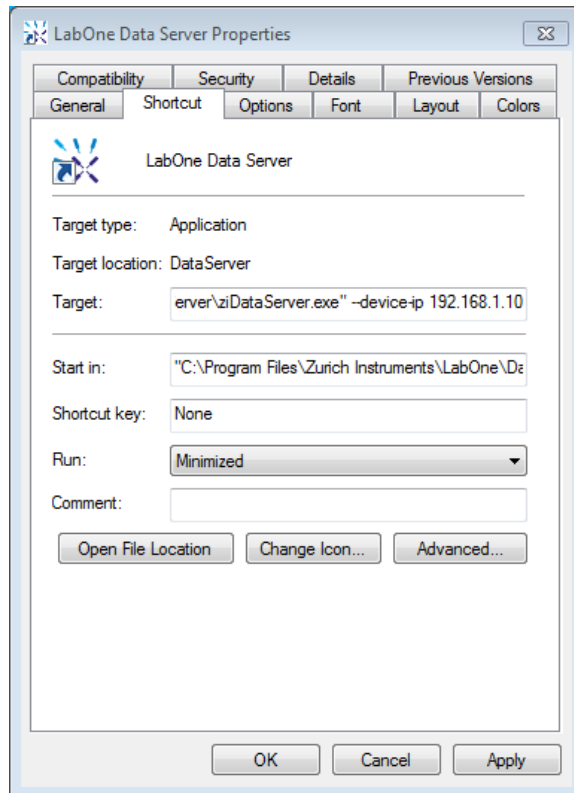
Figure 1.21. Static IP shortcut modification

4. (Optional) To verify the connection between the host computer and the UHF Instrument, open a DOS command window and ping the IP address entered above

Requirements

- Device IP must be known
- Needs network administrator support on networks with dynamic IP configuration

1.6. Software Update

1.6.1. Updating LabOne using Automatic Update Check

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 1.22 will start a download the latest LabOne installer for Windows or Linux, see Figure 1.23. After download, proceed as explained in Section 1.4 to update LabOne.
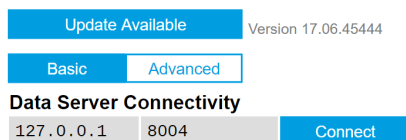
Figure 1.22. Device Connection dialog: LabOne update available
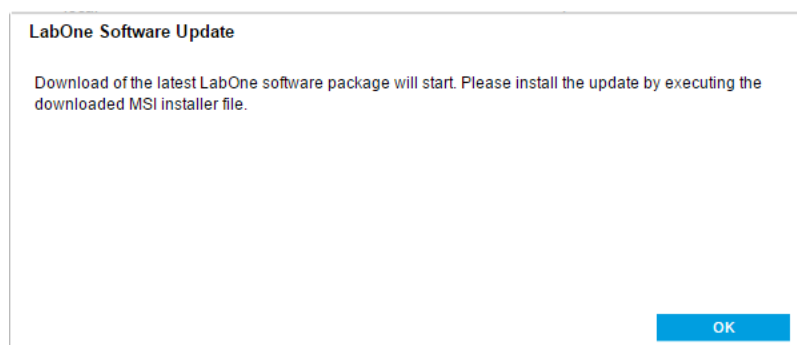
Figure 1.23. Download LabOne MSI using Automatic Update Check feature

1.6.2. 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 1.24. In the Advanced view, there will be a link "Upgrade FW" in the Update column of the Available Devices table. Click on Upgrade FW or Upgrade FW , respectively, to open the firmware update start-up dialog shown in Figure 1.25. The firmware upgrade takes approximately 2 minutes.
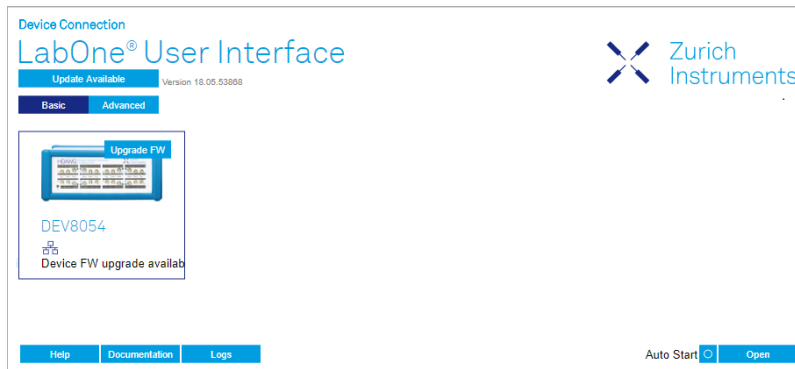
Figure 1.24. Device Connection dialog with available firmware update
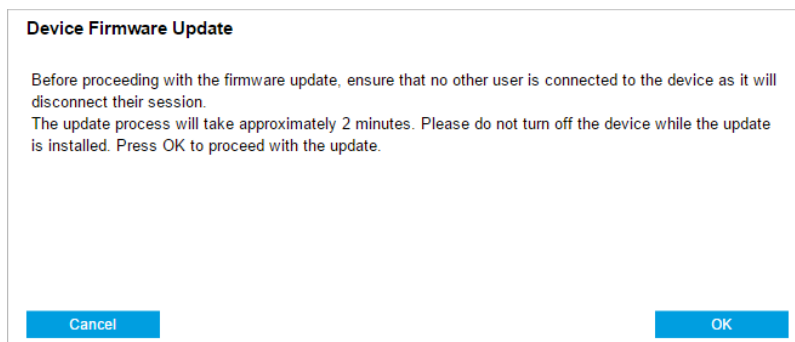
Figure 1.25. Device Firmware Update start-up dialog

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 whilst upgrading the instrument firmware, please contact Zurich Instruments at support@zhinst.com.

1.7. Troubleshooting

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

1.7.1. Common Problems

Your HDAWG Instrument is an advanced piece of laboratory equipment which has many more features and capabilities than a traditional arbitrary waveform generator. 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.

The software cannot be installed or uninstalled: please verify you have Windows administrator 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.

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 does not generate any output signal:** verify that signal output switch has been activated in the Output tab.

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 instruments 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.

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.

1.7.2. 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 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.
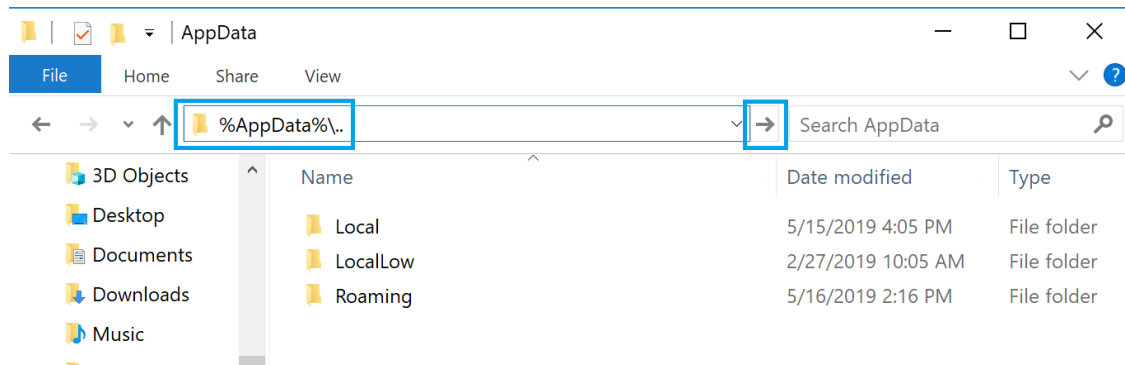
Figure 1.26. Using the `%AppData%\..` shortcut in Windows Explorer to access the hidden folder.

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.exe`):

  `/tmp/ziWebServerLog_[USER]`

Chapter 2. Functional Overview

This chapter provides the overview of the features provided by the HDAWG 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.

2.1. Features

The HDAWG Instrument according to Figure 2.1 consists of several internal units (light blue color) surrounded by several interface units (dark blue color), 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 flow. The orange blocks are optional units that can be either ordered at the beginning or upgraded later. The HDAWG Arbitrary Waveform Generator comes in two model variants, the HDAWG4 (4 channels) and the HDAWG8 (8 channels). Section 2.4 details the available upgrade options for each instrument type and whether the corresponding option can be upgraded directly in the field. The arbitrary waveform generator output is generated on 4 or 8 Wave signal outputs.
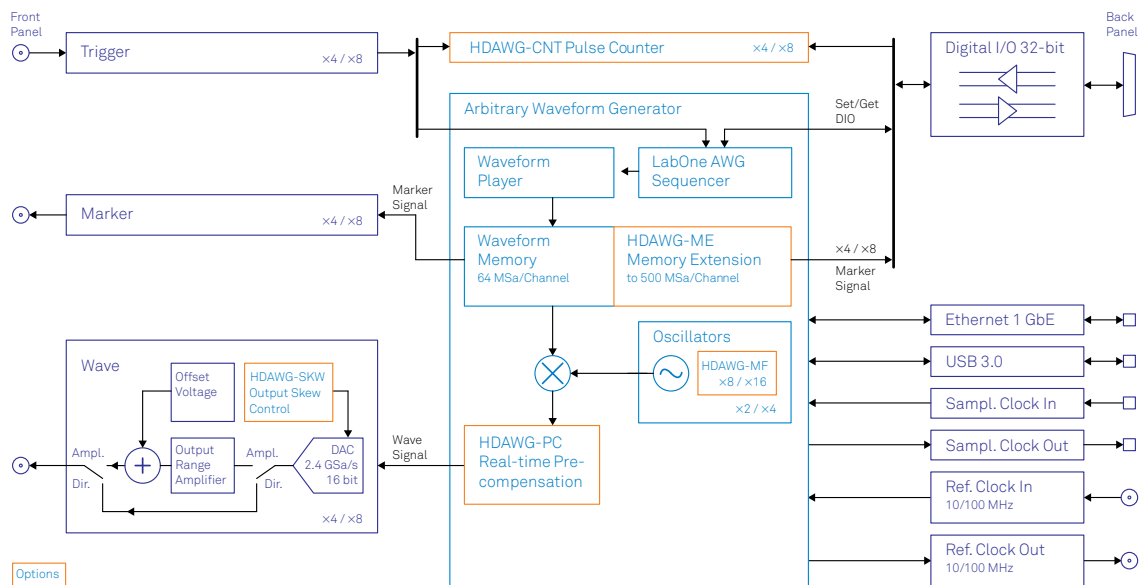
Figure 2.1. HDAWG instrument functional diagram

The numerical oscillators generate sinusoid signals that are used for digital modulation of the AWG signal. Hardware trigger signals are used for timing synchronization within the instrument and with external equipment, such as digitizers, or pattern generators.

Arbitrary Waveform Generator

- 2.4 GSa, 16 bit
- Up to 500 MSa waveform memory per channel
- 2 marker outputs per channel (1 on front panel)
- Digital modulation at multiple frequencies
- Channel grouping 1x8, 2x4, 4x2
- Dynamic sequencing
- Direct and amplified output
- LabOne® AWG Sequencer and Compiler
- Last sample hold
- Dynamic sampling rate
- Switchable output filter

High-speed Connectivity

- LAN 1 Gbit/s interface
- USB 3.0 high-speed host interface
- DIO: 32-bit digital input-output port
- ZSync: multi-instrument synchronization
- Reference clock input/output connector (10/100 MHz)
- Sample clock input/output connector (50 MHz to 2.4 GHz)

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

2.2. Front Panel Tour

The front panel BNC connectors and control LEDs are arranged as shown in Figure 2.2 and Figure 2.3 and listed in Table 2.1.
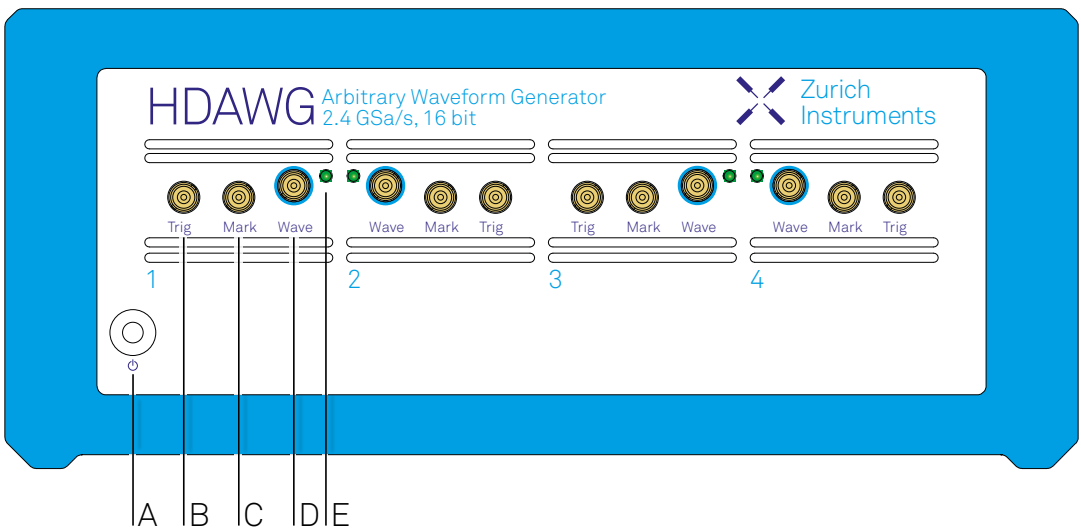
Figure 2.2. HDAWG4 Arbitrary Waveform Generator front panel
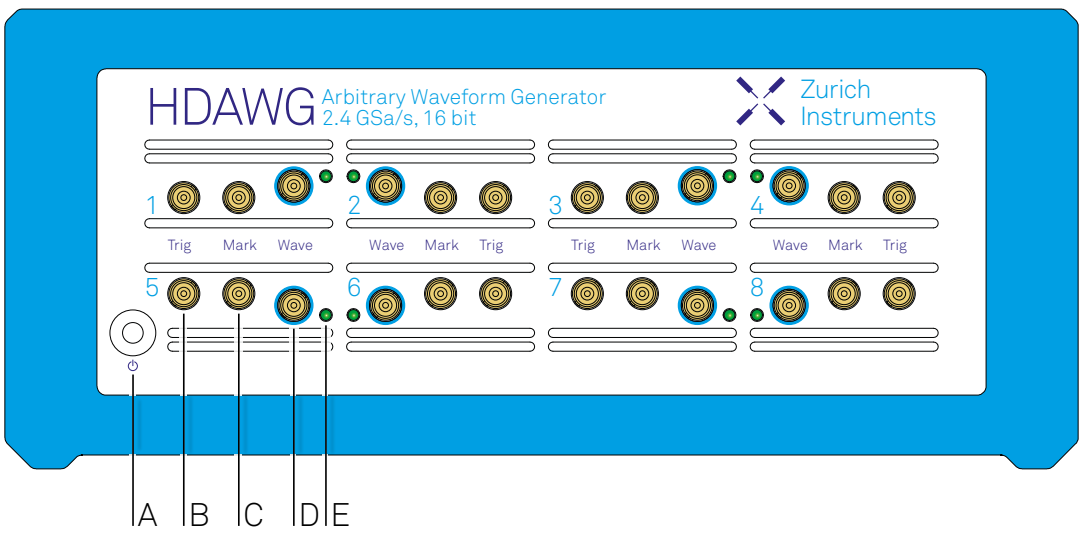
Figure 2.3. HDAWG8 Arbitrary Waveform Generator front panel

Table 2.1. HDAWG Instrument front panel description

Position	Label / Name	Description
A	⏻	soft power button with incorporated status LED
		steady glow, blue: the instrument has an active connection over USB or Ethernet
		blinking rapidly (>1/sec), blue: the instrument firmware is starting up or a firmware upgrade is in progress
		blinking slowly (<1/sec), red[1]: the instrument is ready to connect

Position	Label / Name	Description
		dimmed glow, red[1]: the instrument is shut down and can be powered off using the rear panel switch, or restarted using the soft power button
B	Trig	TTL trigger input
C	Mark	TTL marker output
D	Wave	single-ended waveform signal output
E	n/a	this multi-color LED indicates that the associated Wave output is actively driven by the instrument.
		color blue: output enabled, amplified output
		color green: output enabled, direct output
		color red, running pattern: the instrument firmware has started up correctly
		color red, blinking pattern: indicator used by the Identify Device functionality, or to signal a device error

[1]On some instruments, the color of the soft power button LED is blue in these states.

2.3. Back Panel Tour

The back panel is the main interface for power, control, service and connectivity to other ZI instruments. Please refer to Figure 2.4 and Table 2.2 for the detailed description of the items.
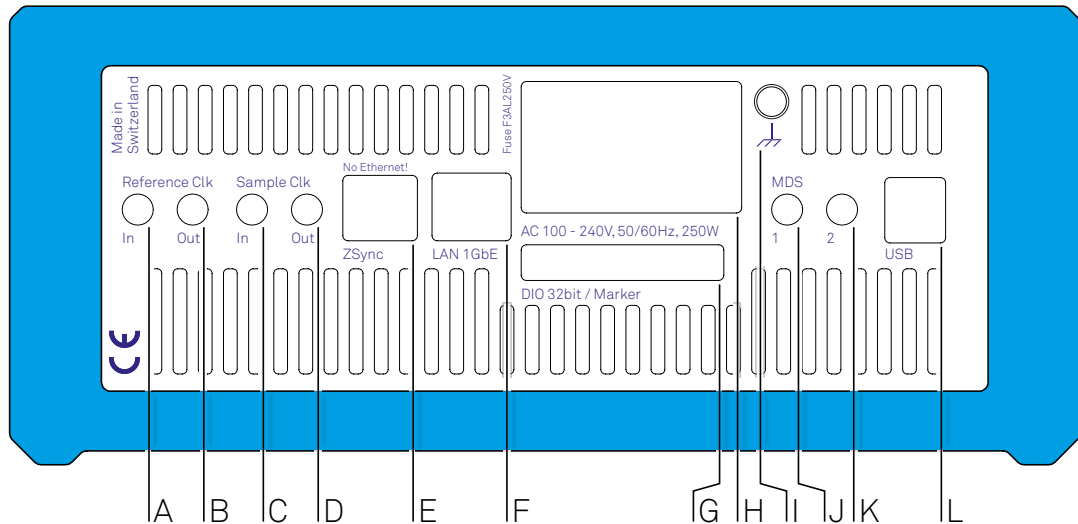
Figure 2.4. HDAWG Instrument back panel

Table 2.2. HDAWG Instrument back panel description

Position	Label / Name	Description
A	Reference Clk In	reference clock input (10 MHz) for synchronization with other instruments
B	Reference Clk Out	reference clock output (10 MHz) for synchronization with other instruments
C	Sample Clk In	sample clock input for external control of the DAC sample clock
D	Sample Clk Out	sample clock output
E	ZSync	inter-instrument synchronization bus connector - attention: this is not an Ethernet plug, connection to an Ethernet network might damage the instrument. Attention: this is not an Ethernet plug, connection to an Ethernet network might damage the instrument
F	LAN 1GbE	1 Gbit LAN connector for connection to the host computer
G	DIO 32bit / Marker	32-bit digital input/output connector
H	AC 100 - 240 V	power inlet, fuse holder, and power switch
I	⏚	4 mm banana jack connector for earth ground, electrically connected to the chassis and the earth pin of the power inlet
J	MDS 1	bidirectional TTL port for multi-device synchronization
K	MDS 2	bidirectional TTL port for multi-device synchronization
L	USB	Universal Serial Bus (USB) 3.0 port for connection to the host computer

2.4. Ordering Guide

Table 2.3 provides an overview of the available HDAWG products. Upgradeable features are options that can be purchased anytime without need to send the Instrument to Zurich Instruments.

Table 2.3. HDAWG Instrument product codes for ordering

Product code	Product name	Description	Field upgrade possible
HDAWG4	HDAWG4 Arbitrary Waveform Generator	base 4-channel arbitrary waveform generator	-
HDAWG4-CNT	HDAWG4-CNT Pulse Counter	option for HDAWG4	yes
HDAWG4-MF	HDAWG4-MF Multi-frequency	option for HDAWG4	yes
HDAWG4-ME	HDAWG4-ME Memory Extension	option for HDAWG4	yes
HDAWG4-PC	HDAWG4-PC Real-time Precompensation	option for HDAWG4	yes
HDAWG4-SKW	HDAWG4-SKW Output Skew Control	option for HDAWG4	no[1]
HDAWG8	HDAWG8 Arbitrary Waveform Generator	base 8-channel arbitrary waveform generator	-
HDAWG8-CNT	HDAWG8-CNT Pulse Counter	option for HDAWG8	yes
HDAWG8-MF	HDAWG8-MF Multi-frequency	option for HDAWG8	yes
HDAWG8-ME	HDAWG8-ME Memory Extension	option for HDAWG8	yes
HDAWG8-PC	HDAWG8-PC Real-time Precompensation	option for HDAWG8	yes
HDAWG8-SKW	HDAWG8-SKW Output Skew Control	option for HDAWG8	no[1]
HDIQ	IQ Modulator	4 channel IQ modulator, accessory for HDAWG4 and HDAWG8	yes

[1]The upgrade requires the instrument to be returned and calibrated at Zurich Instruments.

Table 2.4. Product selector HDAWG

Feature	HDAWG8	HDAWG8 + HDAWG8-MF	HDAWG8 + HDAWG8-MF + HDAWG8-ME	HDAWG8 + HDAWG8-MF + HDAWG8-ME + HDAWG8-CNT
Number of AWG channels	8	8	8	8
Amplitude Modulation mode	yes	yes	yes	yes
Sequencing	yes	yes	yes	yes
Direct and amplified output modes	yes	yes	yes	yes
Variable sampling rate	yes	yes	yes	yes

Feature	HDAWG8	HDAWG8 + HDAWG8-MF	HDAWG8 + HDAWG8-MF + HDAWG8-ME	HDAWG8 + HDAWG8-MF + HDAWG8-ME + HDAWG8-CNT
Sine generators[1]	8	32	32	32
Oscillators[1]	4	16	16	16
Multi-device synchronization	yes	yes	yes	yes
Multi-frequency modulation	no	yes	yes	yes
Digital IQ modulation	yes	yes	yes	yes
Waveform memory per channel	64 MSa	64 MSa	500 MSa	500 MSa
Number of pulse counters[1]	-	-	-	8
Sampling rate 2.4 GSa/s	yes	yes	yes	yes
Marker bits[1]	16	16	16	16
Vertical resolution 16 bit	yes	yes	yes	yes
Frequency range	750 MHz	750 MHz	750 MHz	750 MHz
USB 3.0	yes	yes	yes	yes
LAN 1 Gbit/s	yes	yes	yes	yes

[1]Numbers in these lines are divided by 2 in for the HDAWG4 4-channel model variant.

Chapter 3. Tutorials

The tutorials in this chapter have been created to allow users to become more familiar with the operation of the arbitrary waveform generator. In order to successfully carry out the tutorials, it's assumed that users 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 Chapter.

- 1 Ethernet cable (supplied with your HDAWG)
- 1 oscilloscope (min. 2 channels, recommended 4, bandwidth 500 MHz or more)
- 4 BNC or SMA coaxial cables
- 4 adaptors BNC male to SMA female

3.1. Basic Waveform Playback

Note

This tutorial is applicable to all HDAWG Instruments.

3.1.1. Goals and Requirements

The goal of this tutorial is to demonstrate the basic use of the HDAWG. We demonstrate waveform generation and playback, triggering and synchronization. In order to visualize the multi-channel signals, an oscilloscope with sufficient bandwidth and channel number is required.

3.1.2. Preparation

Connect the cables as illustrated below. Make sure that the instrument is powered on and connected 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.
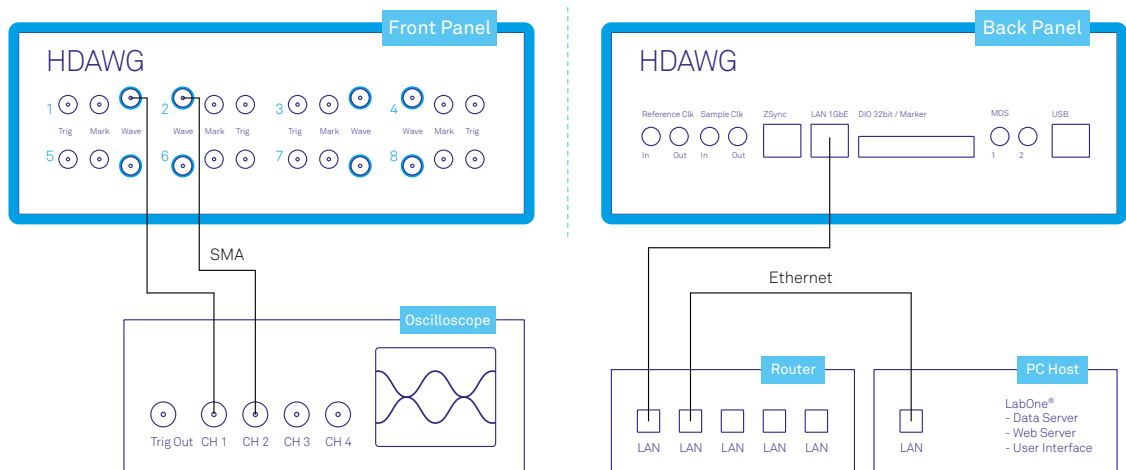
Figure 3.1. Connections for the arbitrary waveform generator basic playback tutorial

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).

3.1.3. Waveform Generation and Playback

In this tutorial we generate signals with the AWG and visualize them with the scope. In a first step we enable the Wave outputs, but disable all sinusoidal signals generated by the sine generators by default. We also configure the scope with a suitable time base (e.g. 500 ns per division) and range (e.g. 0.2 V per division) The following table summarizes the necessary settings.

Table 3.1. Settings: enable the output

Tab	Sub-tab	Section	#	Label	Setting / Value / State
Output		Wave Outputs	1	Enable	ON

Tab	Sub-tab	Section	#	Label	Setting / Value / State
Output		Wave Outputs	2	Enable	ON
Output		Sine Generators	1&2	Wave 1 Enable	OFF
Output		Sine Generators	1&2	Wave 2 Enable	OFF

Table 3.2. Settings: configure the external scope

Scope Setting	Value / State
Ch1 enable	ON
Ch1 range	0.2 V/div
Timebase	500 ns/div
Trigger source	Ch1
Trigger level	100 mV
Run / Stop	ON
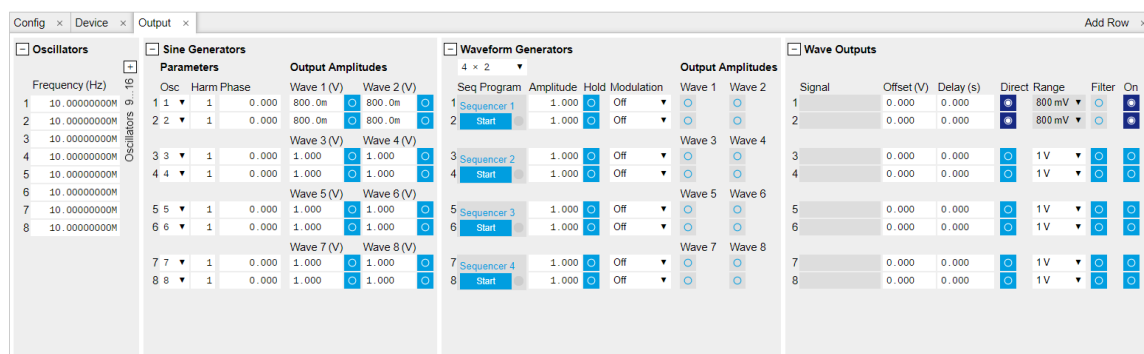
Figure 3.2. LabOne UI: Output tab

In the Output tab, we configure the first output channel. The final signal amplitude is given by the product of the full scale output range of 1 V, the dimensionless amplitude scaling factor 1.0, and the actual dimensionless signal amplitude stored in the waveform memory. The necessary settings are summarized in the following table.

Table 3.3. Settings: configure the AWG output

Tab	Sub-tab	Section	#	Label	Setting / Value / State
Seq	Control			Sampling Rate	2.4 GHz
Output		Waveform Generators	1	Amplitude	1.0
Output		Waveform Generators	1	Modulation	OFF
Output		Wave Outputs	1	Range	1 V
Output		Wave Outputs	1	Enable	ON

To operate the AWG we need to specify a sequence program. This can be done interactively by typing the program in the Sequence 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 (2.4 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 Wave output range (here we chose 1.0 V). We put a scaling factor of 1.0 in place which can be replaced by any other value. The code line is terminated by a semicolon according to C conventions. With the second line of the program, the generated waveform `w_gauss` is played on AWG Output 1.

Note

For this tutorial, we will keep the description of the Sequencer instructions short. You can find the full specification of the LabOne Sequencer language in Section 4.14.3

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. Waveforms with other lengths are allowed, but they get padded with zeros by the compiler. This may be undesired in some cases, specifically when using the Hold feature of the AWG.

If we now click on Save, the program gets compiled. This means the program is translated into instructions for the LabOne Sequencer on the instrument, see Section 4.14.2. If no error occurs (due to wrong program syntax, for example), the Compiler 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 Compiler 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 Single, 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 1.0 μs as shown in Figure 3.3 .
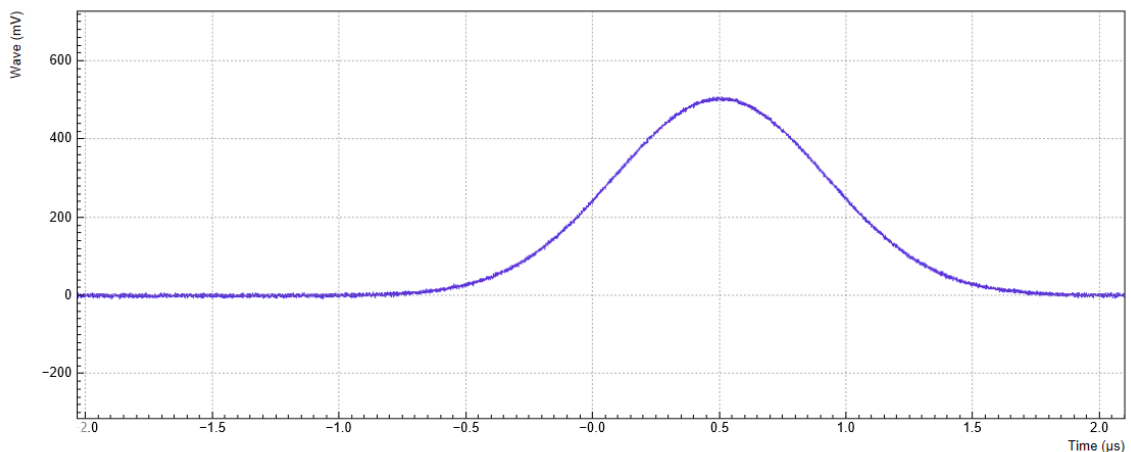
Figure 3.3. Gaussian pulse as generated by the AWG and captured by the scope

The LabOne Sequencer language offers a lot of run-time control. The basic functionality is to repeat a waveform several times. In the following example, all the code within the curly brackets {…} is repeated 5 times. Upon clicking `Save` and `Single`, you should observe 5 short Gaussian pulses in a new scope shot, see Figure 3.4 .

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

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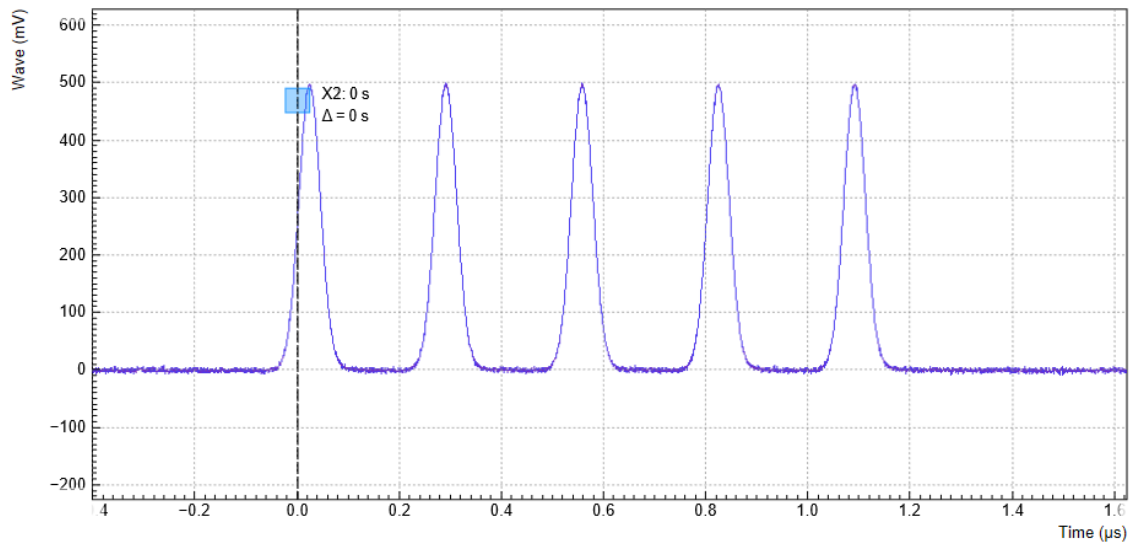

Figure 3.4. Burst of Gaussian pulses generated by the AWG and captured by the scope

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 Section 4.14.3 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 output generated by the program above is shown in Figure 3.5.
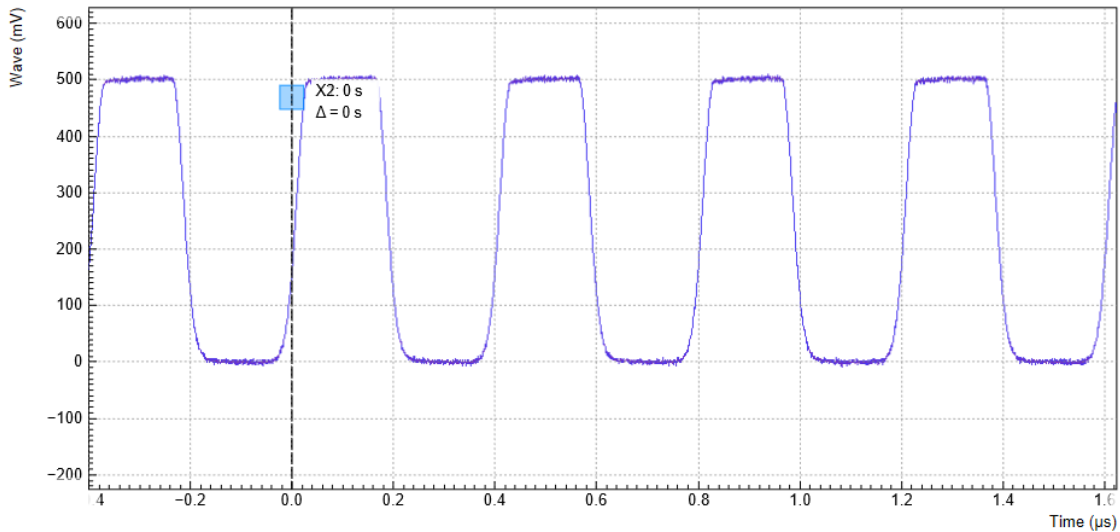
Figure 3.5. Infinite pulse series generated by the AWG and captured by the scope

As programs get longer, it becomes useful to store and recall them. Clicking on `Save as...` allows you to store the present program under a new 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 using some of the editor keyboard shortcuts listed in the section called "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
```

If the file is stored in the location (`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), you can then play back the wave by referring to the file name without extension in the sequence program:

```
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 output stage of the instrument. This means that signal components exceeding the output stage 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 on the device. The integer values in that file should be 18-bit unsigned integers with the two least significant bits being the markers. The values are mapped to 0 => -FS, 262143 => +FS, with FS equal to the full scale.

3.1.4. Triggering and Synchronization

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

- Triggering the AWG with an external TTL signal
- Generating a TTL signal with the AWG to trigger another piece of equipment

Triggering the AWG

In this section we show how to trigger the AWG with an external TTL signal. We start by generating a periodic TTL signal using the external scope (on the trigger, sync, or auxiliary output), if it has this capability, or by using another source of such a signal, e.g., a function generator. Generate a signal with the parameters listed in the following table.

Table 3.4. Settings: generate a 300 kHz TTL signal

Setting	Value / State
Signal type	square wave
Signal level	0 V / 3.3 V
Frequency	300 kHz

We then connect this square signal to the HDAWG Trig input 1, and separately monitor the signal on the scope using a power splitter or a T-piece. The figure below shows the connection used.
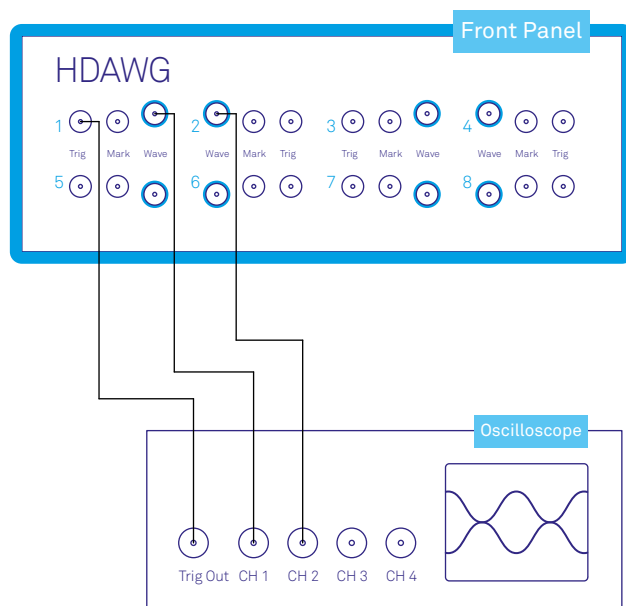
Figure 3.6. Connections for triggering the arbitrary waveform generator

The AWG internally has 2 digital trigger input channels. 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 Trig 1 connector.

Table 3.5. Settings: configure the AWG analog trigger input

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

Finally, we modify our last AWG program by including a `waitDigTrigger` instruction just before the `playWave` instruction. The result is that upon every repetition inside the infinite `while` loop, the AWG will wait for a rising edge 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);
```

```
    playWave(1, w_pulse);
}
```

Compile and run the above program. Figure 3.7 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 trigger signal frequency or unplugging the trigger cable to observe the immediate effect on the signal.
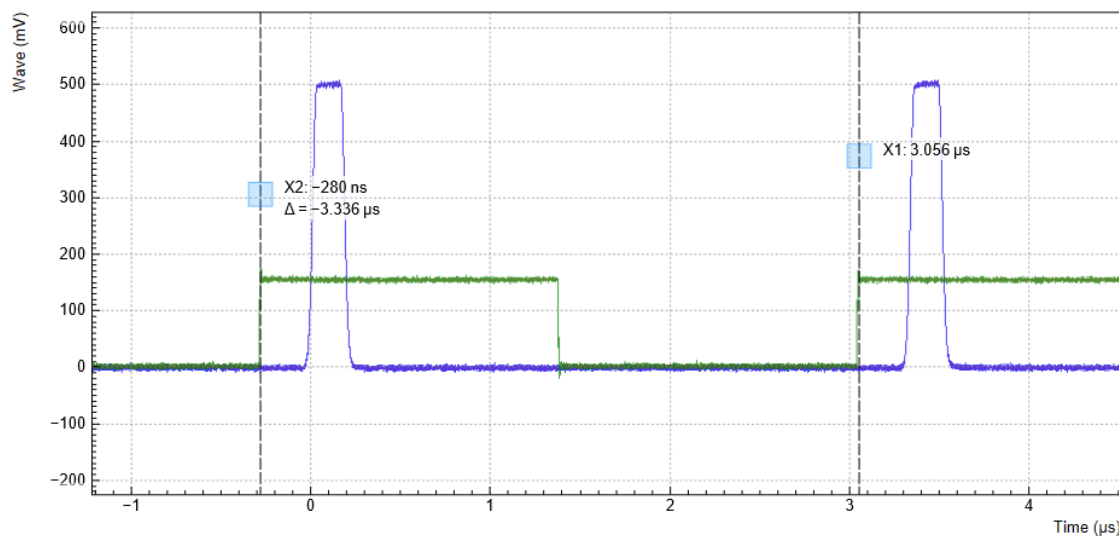
Figure 3.7. Externally triggered pulse series generated by the AWG and captured by the scope. The AWG signal is shown in blue, the external trigger signal (attenuated by 20 dB) is shown in green.

Generating Markers with the AWG

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

The method using markers is recommended when precise timing is required, and/or complicated serial bit patterns need to be played on the Marker outputs. Marker bits are part of every waveform which is an array of 18-bit words: 16 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 instruction is simpler, but the timing control is less powerful than when using markers. It is useful for instance to generate a single trigger signal at the start of an AWG program.

Table 3.6. Comparison: AWG markers and triggers

	Marker	Trigger
Implementation	Part of waveform	Sequencer instruction
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 generated wave actually 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.

There is a certain freedom to assign different marker bits to different Mark outputs. The following table summarizes the settings to apply in order to output marker bit 1 on Mark 1.

Table 3.7. Settings: configure the AWG marker output and scope trigger

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

We connect the Mark 1 signal to one of the scope channels as shown below.
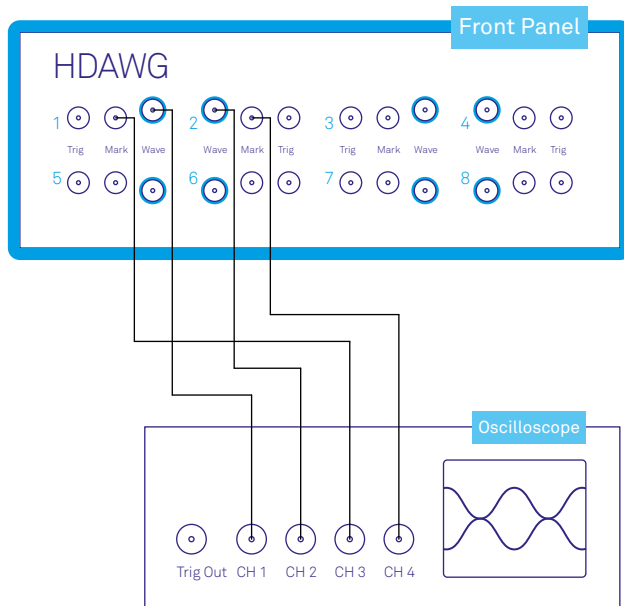
Figure 3.8. Connections for generating marker signals with the arbitrary waveform generator

Figure 3.9 shows the AWG signal captured by the scope as a blue curve. The green curve shows the second scope channel displaying the marker 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.
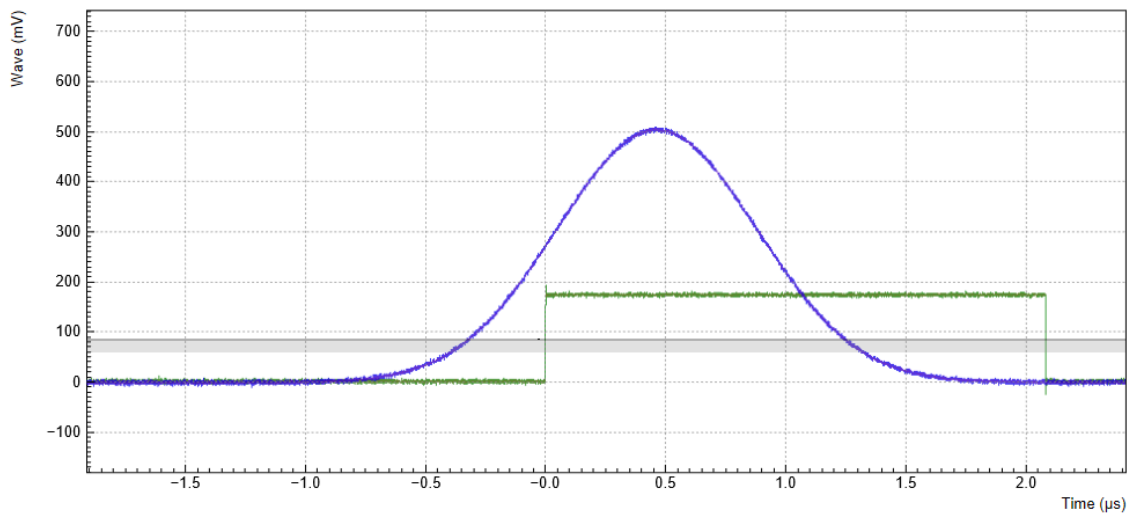
Figure 3.9. Gaussian pulse and square marker signal generated by the AWG and captured by the scope

Let us now demonstrate the use of **sequencer instructions** 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 trigger output states in binary manner – 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` instruction after the `playWave` instruction. It ensures that the subsequent `setTrigger` instruction is executed only after the Gaussian wave has finished playing, and not during waveform playback.

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

Table 3.8. Settings: configure the AWG trigger output

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

Figure 3.10 shows the AWG signal captured by the scope. This looks very similar to Figure 3.9 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.
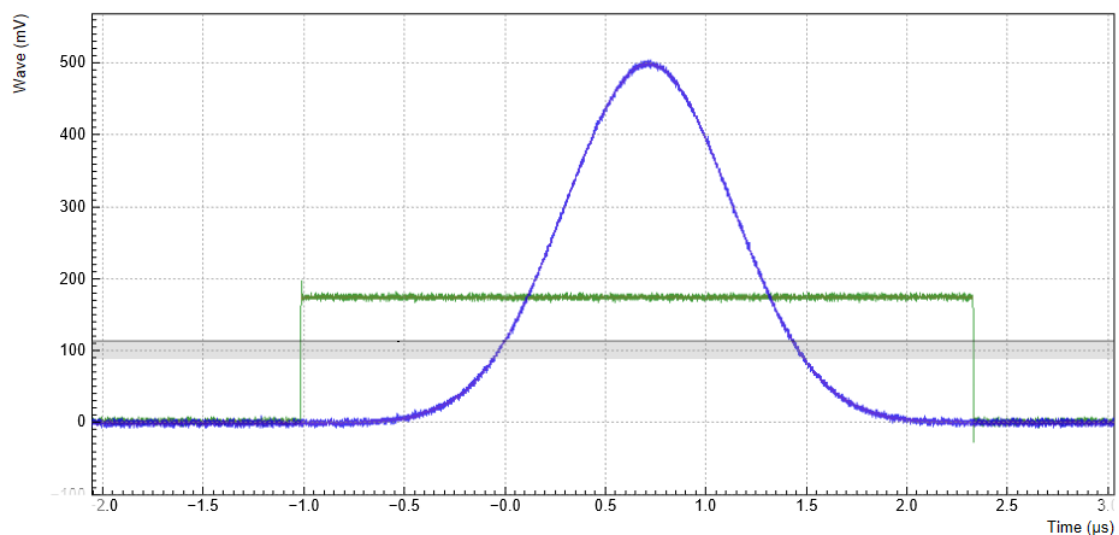
Figure 3.10. Gaussian pulse and trigger signal generated by the AWG and captured by the scope

3.2. Multi-Channel Playback

Note

This tutorial is applicable to all HDAWG Instruments. The number of available signal channels depends on the model (HDAWG4 or HDAWG8).

3.2.1. Goals and Requirements

The goal of this tutorial is to demonstrate multi-channel waveform playback with the AWG. In order to visualize the multi-channel signals, an oscilloscope with sufficient bandwidth and channel number is required.

3.2.2. Preparation

Connect the cables as illustrated below. Make sure that the instrument is powered on and connected 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.
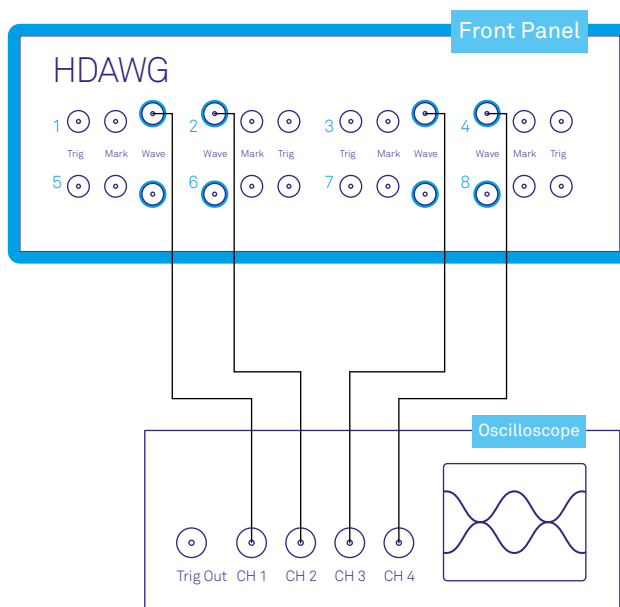
Figure 3.11. Connections for the arbitrary waveform generator multi-channel playback tutorial

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).

3.2.3. Channel Grouping

Using the channel grouping feature of the HDAWG, we can generate multi-channel signals in different configurations: using one sequence program for all 8 Wave outputs, using 2 sequence programs for Wave outputs 1-4 and 5-8 respectively, or using 4 sequence programs for Wave

outputs 1&2, 3&4, etc. This is enabled by the multi-core architecture of the HDAWG, see Section 4.3. Channel grouping enables independent timing and triggering on different outputs and gives the flexibility to use 1 instrument for multiple experiments.

In the previous tutorial, we used the default channel grouping mode, which is 4x2 on 8-channel instruments, or 2x2 on 4-channel instruments, and we wrote sequence programs for dual-channel playback. Here we show an example of a 4-channel sequence program. To this end, we change the Channel Grouping setting in the AWG Sequencer tab to 2x4 (1x4, respectively) which changes the number of sequence editor side-tabs in that tab. We monitor the AWG signal using four channels of an external scope. The following tables summarize the settings to enable the HDAWG Wave outputs, to change the channel grouping, and to configure the external scope.

Table 3.9. Settings: enable the output

Tab	Sub-tab	Section	#	Label	Setting / Value / State
Output		Wave Outputs	1-4	Enable	ON
Output		Sine Generators	1&2	Wave 1 Enable	OFF
Output		Sine Generators	1&2	Wave 2 Enable	OFF
Output		Sine Generators	3&4	Wave 3 Enable	OFF
Output		Sine Generators	3&4	Wave 4 Enable	OFF
AWG Sequencer	Control			Channel Grouping	2x4 or 1x4

Table 3.10. Settings: Configure the external scope

Scope Setting	Value / State
Ch1-4 enable	ON
Ch1-4 range	0.2 V/div
Timebase	500 ns/div
Trigger source	Ch1
Trigger level	100 mV
Run / Stop	ON

The step from a dual-channel to a four-channel sequence is achieved by simple extension of the number of `wave` arguments of the `playWave` instruction to 4. The following sequence program generates a signal with 4 simultaneous pulses with different amplitudes. The `setTrigger` instructions can optionally be used to trigger the scope if the latter has an auxiliary trigger input as shown in the section called "Generating Markers with the AWG". Upload and run this program. Figure 3.12 shows the signal generated with this program and measured with the scope.

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

while (true) {
  setTrigger(1); setTrigger(0);
  playWave(1.0*w, 0.5*w, -0.5*w, -1.0*w);
  wait(10000);
}
```
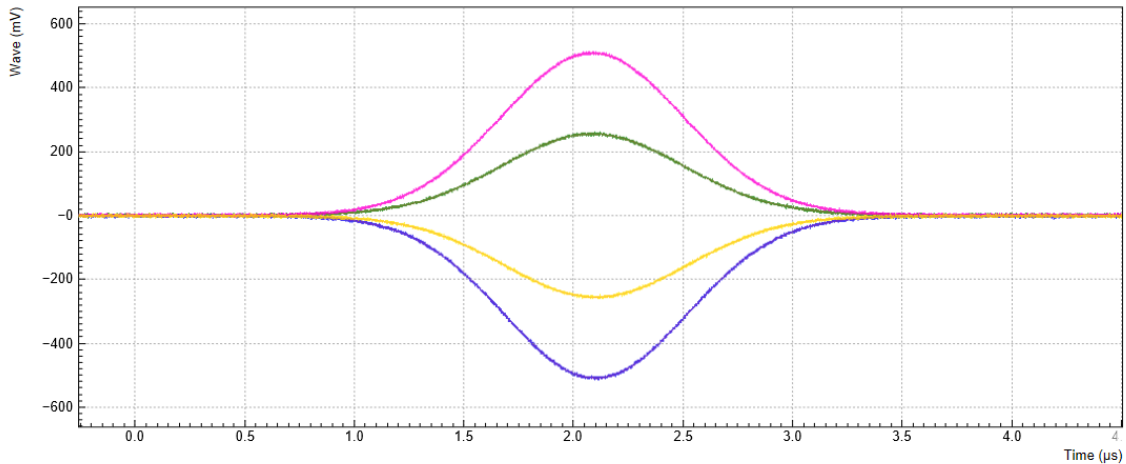
Figure 3.12. Four-channel signal generated by the AWG and captured by the scope.

3.2.4. Output Assignment

In addition to the single-channel, dual-, and quad-channel playback used up to now, there are more options for the channel assignment. The `playWave` instruction can be used with different combinations of arguments: with one `wave` type argument or with several, with a `const` type integer number specifying the Wave output or without. These different combinations of arguments allow the user to independently control the AWG core outputs (the digital signal sources inside the instrument) and the place where their signal is routed to (the Wave outputs on the front panel). See Section 4.3 for an overview of the internal architecture and the terminology. Inside the Output tab, the AWG core outputs are represented in the Waveform Generators section, whereas the signal outputs are represented in the Wave Outputs section.

The `playWave` instruction always assigns the first `wave` argument to the AWG core output 1, and the second one (if it's provided) to the AWG core 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 Wave output 2.

Note

A common pitfall is the instruction `playWave(2, w_gauss)` which generates a signal on Wave output 2, but uses the AWG core output 1. This means that the relevant Mode and Amplitude (FS) settings are on line 1 of the Waveform Generators section (Output tab), even though the signal is routed to Wave output 2.

It's possible to route a single AWG core output to two Wave 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 Wave output by using twice the same integer argument as in `playWave(1, w_gauss, 1, w_drag)`. This is e.g. useful in combination with the Amplitude Modulation mode as it enables digital IQ modulation of an internal oscillator which gives full freedom in controlling the amplitude and phase of a carrier with the AWG. The following sequence program contains a number of examples for these configurations. Figure 3.13 shows the dual-channel signal generated with this program and measured with the scope.

```
wave w_gauss = 0.5*gauss(8000, 4000, 1000);
wave w_drag  = 0.5*drag(8000, 4000, 1000);
```

```
while (true) {
  setTrigger(1);
  // play wave on Signal Output 1 with AWG Output 1 (two equivalent commands):
  playWave(w_gauss);
  playWave(1, w_gauss);
  // play wave on Signal Output 2 with AWG Output 1:
  playWave(2, w_gauss);
  // play identical Wave on Signal Output 1 and 2 generated with AWG Output 1:
  playWave(1, 2,  w_gauss);
  // play independent Waves on Signal Output 1 and 2 generated
  // with AWG Output 1 and 2 (two equivalent commands):
  playWave(w_gauss, w_drag);
  playWave(1, w_gauss, 2, w_drag);
  // add up two independent Waves on Signal Output 1
  // generated with AWG Output 1 and 2:
  playWave(1, w_gauss, 1, w_drag);

  waitWave();
  setTrigger(0);
  wait(10000);
}
```

Note

Limitations to the concept of crossed assignments between AWG core outputs and Wave outputs come from the fact that one AWG core can only send signals to its associated Wave output pair (see Section 4.3)
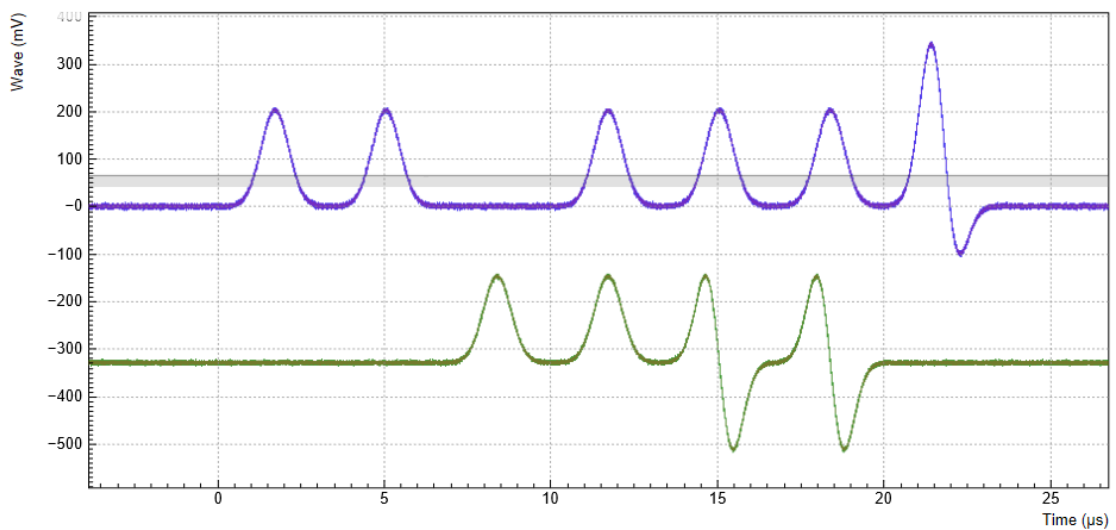
Figure 3.13. Dual-channel signal generated by the AWG and captured by the scope.

3.3. Digital Modulation

Note

The first part of this tutorial is applicable to all HDAWG Instruments. However, Section 3.3.5 requires the HDAWG-MF option.

3.3.1. Goals and Requirements

The goal of this tutorial is to demonstrate the use of the digital modulation feature of the AWG. We demonstrate amplitude modulation and multi-frequency modulation. In order to visualize the generated multi-channel signals, an oscilloscope with sufficient bandwidth and at least 3 channels is required.

3.3.2. Preparation

Connect the cables as illustrated below. Make sure that the instrument is powered on and connected by 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.
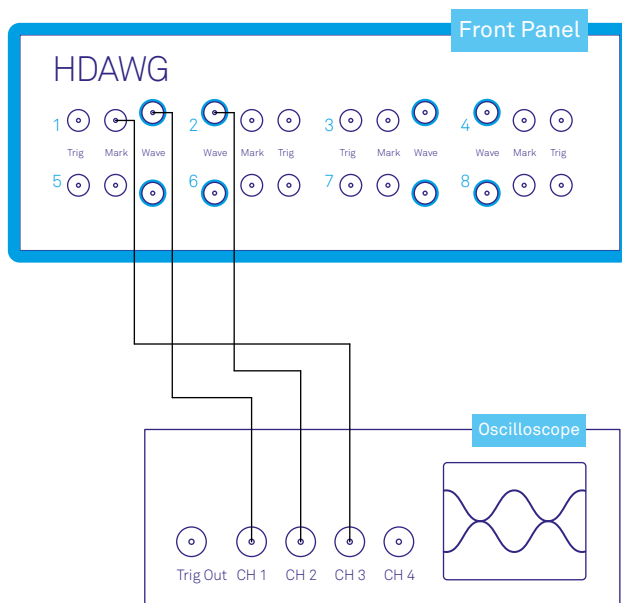
Figure 3.14. Connections for the arbitrary waveform generator digital modulation tutorial

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).

3.3.3. Generating an I/Q Baseband Signal

In digital modulation mode, the output of the AWG is multiplied with the amplitude of one or more of the internal sine generator signals of the instrument. There are numerous advantages to using

modulation mode in comparison to simply generating the sinusoidal signal directly using the AWG, such as the ability to change the frequency without uploading a new waveform, extremely high frequency resolution independent of AWG waveform length, phase-coherent generation of signals (because the oscillator keeps running even when the AWG is off), the ability to analyze an input signal at the exact frequency of the generated signal using demodulators, Boxcar and PWA, and more. The goal of this section is to demonstrate how to use the modulation mode.

We monitor the AWG signal using two channels of an external scope and use the third scope channel for triggering purposes. The following tables summarize the settings to enable the HDAWG Wave outputs and to configure the external scope.

Table 3.11. Settings: enable the output

Tab	Sub-tab	Section	#	Label	Setting / Value / State
Output		Wave Outputs	1	Enable	ON
Output		Wave Outputs	2	Enable	ON
Output		Sine Generators	1&2	Wave 1 Enable	OFF
Output		Sine Generators	1&2	Wave 2 Enable	OFF

Table 3.12. Settings: Configure the external scope

Scope Setting	Value / State
Ch1-3 enable	ON
Ch1-3 range	0.2 V/div
Timebase	500 ns/div
Trigger source	Ch3
Trigger level	100 mV
Run / Stop	ON

We design this example around a common use case, which is the generation of dual-channel quadrature (I/Q) modulation signals to feed into a microwave mixer. Such signals require independent control of two envelope waveforms multiplied by a carrier that is shifted by 90° between the two channels. The program below generates two independent waveforms and plays them repeatedly on both channels. For dual-channel playback we can use the same `playWave` function that we used up to now, and simply pass to it two waveforms as arguments. We include the previously used trigger instructions for the scope, and include a `wait` instruction whose argument is in units of the sequencer clock period of about 3.33 ns.

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

while (true) {
  setTrigger(1);
  playWave(w_gauss, w_drag);
  waitWave();
  setTrigger(0);
  wait(100);
}
```

By setting the Modulation control of AWG channel 1, we can select either sine generator 1 or 2 for the carrier generation. The sine generators are represented in the Output tab to the left of the Waveform Generators section. A sine generator is a direct digital synthesis (DDS) unit that converts a digital oscillator signal (essentially just an incrementing phase) to a sinusoid with a certain phase offset and harmonic multiplier using a look-up table containing one period

of the sinusoid signal. The digital oscillator in turn is a phase accumulator with a very precise frequency derived from the instrument's master clock. The digital oscillators on the instrument are represented in the Oscillators section of the Output tab. If the HDAWG-MF Multi-frequency option is installed, we have the freedom to freely assign oscillators to sine generators using the Osc selector in the Sine Generators section. This allows us to generate multiple signals with the same frequency but different phases, or to generate several harmonics of one base frequency with well-controlled phase relations.

The following parameter settings apply to the case with installed HDAWG-MF option.

Table 3.13. Settings: enable the output and configure the AWG marker output and scope trigger

Tab	Sub-tab	Section	#	Label	Setting / Value / State
DIO		Marker Output	1	Signal	AWG Trigger 1
Output		Sine			
Output		Oscillators	1	Frequency	5 MHz
Output		Sine Generators	1	Osc	1
Output		Sine Generators	2	Osc	1
Output		Sine Generators	1	Phase	0°
Output		Sine Generators	2	Phase	90°
Output		Waveform Generators	1	Modulation	Off
Output		Waveform Generators	2	Modulation	Off

Save and play the Sequencer program with the above settings. The upper plot in Figure 3.15 shows the AWG signals captured by the scope. We see the expected Gaussian pulse on Wave output 1 (green) and the DRAG pulse, which corresponds to the derivative of a Gaussian function, on Wave output 2.
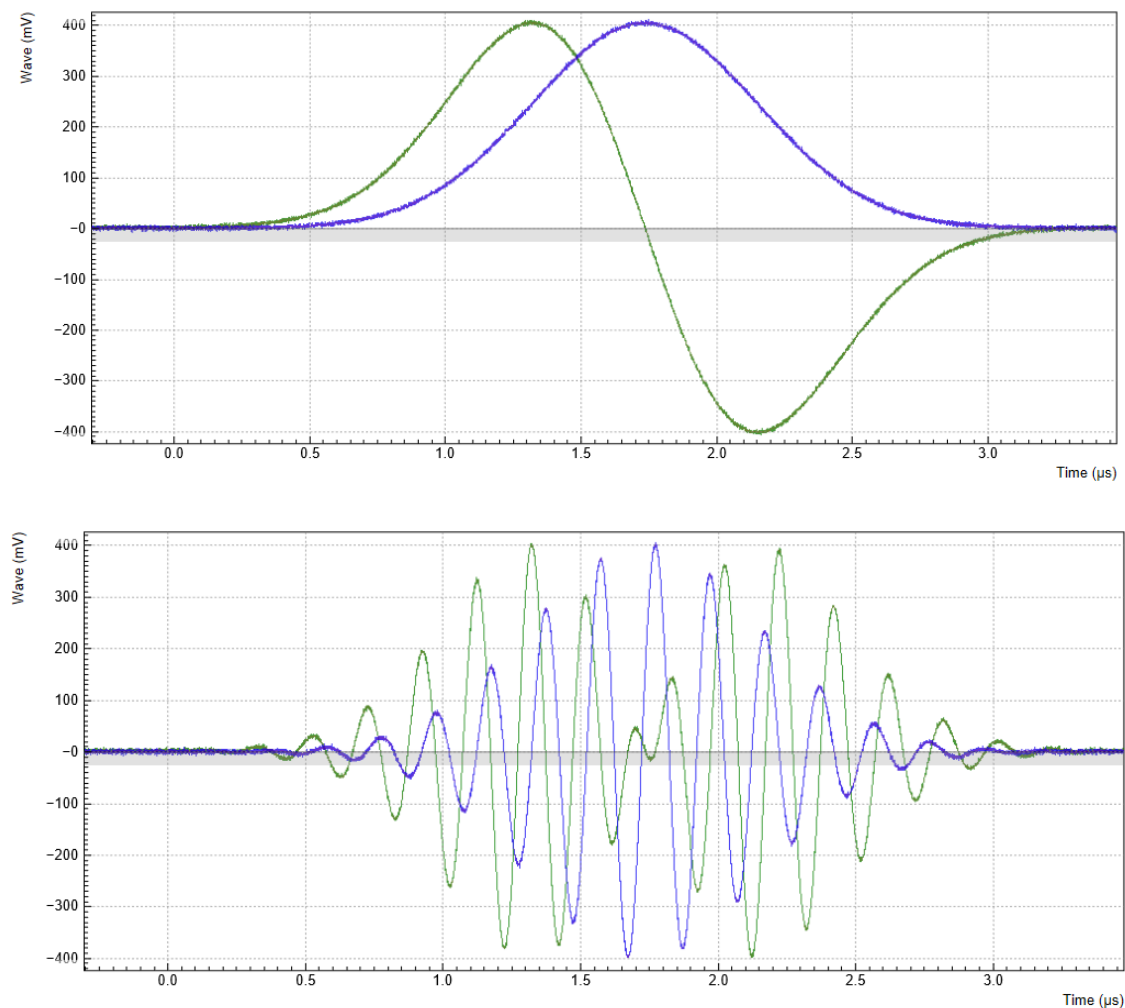
Figure 3.15. Dual-channel signal generated by the AWG and captured by the scope. The top figure shows two envelope waveforms played without modulation, the bottom figure shows the same envelope waveforms played with enabled modulation.

While the AWG is running, you can enable modulation by setting the AWG channel 1 Modulation to Sine 11, and AWG channel 2 Modulation to Sine 22. The notation Sine NM with N, M = 1, 2,... signifies the following: the given AWG channel is multiplied with the signal of Sine Generator N when routed to the first Wave output of the given AWG core, and with the signal of Sine Generator M when routed to the second Wave output of the given AWG core. Here, we route the AWG signals only along the diagonal path (AWG channel 1 to Wave 1, and AWG channel 2 to Wave 2), so we make use of only 2 of the 4 signal multipliers of core 1. The resulting signal on Wave 1 is AWG1×Sine1, and the resulting signal on Wave 2 is AWG2×Sine2. Here, AWG1 denotes the signal of AWG channel 1 (the Gaussian pulse) AWG2 the signal of AWG channel 2 (the DRAG pulse), and Sine1 (Sine2) the signal of Sine Generator 1 (2).

The lower plot in Figure 3.15 shows the resulting signals, which are the Gaussian and DRAG pulses multiplied by a 5 MHz carrier with phase shift 0° and 90°, respectively.

Table 3.14. Settings: enable modulation on two Wave outputs

Tab	Sub-tab	Section	#	Label	Setting / Value / State
Output		Waveform Generators	1	Modulation	Sine 11

Tab	Sub-tab	Section	#	Label	Setting / Value / State
Output		Waveform Generators	2	Modulation	Sine 22

In this practical case of I/Q modulation, the two Wave outputs typically require further adjustments of the pulse amplitude, DC offset, and inter-channel phase offset in order to compensate for analog mixer imperfections. All these adjustments can now be done on the fly using the AWG Amplitude, the Wave Output Offset, and the Sine Generator Phase settings without having to make any changes to the programmed AWG waveforms.

For certain cases of I/Q modulation, it's required to generate linear combinations of 2 modulated signal on each Wave output. To realize this, we can make use of all 4 signal multipliers of the AWG core. We can modify the sequence program such that each AWG channel is routed to both Wave outputs by passing additional arguments to the `playWave` instruction (see Section 3.2.4):

```
wave w_gauss = gauss(8000, 4000, 1000);
wave w_drag = drag(8000, 4000, 1000);
while (true) {
setTrigger(1);
playWave(1, 2, w_gauss, 1, 2, w_drag);
waitWave();
setTrigger(0);
wait(100);
}
```

In a typical case, we want to use crossed assignments between AWG channels and Sine generators by selecting the settings in the following table. In order to prevent an overflow condition of the summed-up signals, we set the gain parameters at each signal multiplier to 0.5.

Table 3.15. Settings: enable crossed dual-phase modulation on two Wave outputs

Tab	Sub-tab	Section	#	Label	Setting / Value / State
Output		Waveform Generators	1	Modulation	Sine 12
Output		Waveform Generators	2	Modulation	Sine 21
Output		Waveform Generators	1	Output Amplitude Wave 1	0.5
Output		Waveform Generators	1	Output Amplitude Wave 2	0.5
Output		Waveform Generators	2	Output Amplitude Wave 1	0.5
Output		Waveform Generators	2	Output Amplitude Wave 2	0.5

With these settings and the above sequence program, the resulting signal on Wave 1 is equal to AWG1×Sine1 + AWG2×Sine2, and the resulting signal on Wave 2 is equal to AWG1×Sine2 + AWG2×Sine1.

3.3.4. Rapid Phase Changes

The HDAWG supports rapid, real-time changes of the carrier phase in modulation mode through the sequencer instructions `setSinePhase` and `incrementSinePhase`. This capability is particularly valuable when generating long patterns of pulses with varying phases, e.g. to account for AC Stark shift in qubit control sequences, or to realize phase cycling protocols.

In addition, there is the possibility to reset the starting phase of one or multiple oscillator at the beginning of a pulse sequence using the `resetOscPhase` instruction. Thus it can be ensured that the carrier-envelope offset, and thus the final output signal, is identical from one repetition to the next.

Due to the architecture of the HDAWG, both the Sine Generators and the Oscillators are physically located in the front-end FPGAs of the instrument (see Section 4.3). For instruments with the HDAWG-MF option, each of the 8 or 16 oscillator exists as a copy on each front-end FPGA. When changing an oscillator frequency from the graphical user interface, all copies get synchronously updated to the new configuration with the help of the back-end FPGA. This synchronization mechanism is however relatively slow and not a real-time operation. When making use of the fast oscillator control in the form of the `resetOscPhase` instruction, the synchronization mechanism needs to be turned off by enabling the AWG Oscillator Control mode. Each AWG core is then allowed to control and use only a subset of oscillators: Oscillators 1 through 4 on core 1, oscillators 5 through 8 on core 2, and so forth. In the following example, we will use the oscillator 1 for both Sine Generators on AWG core 1. On instruments without the HDAWG-MF option, this is the default configuration and not modifiable. Apply the settings in the following table.

Table 3.16. Settings: enabling AWG oscillator control for rapid phase changes

Tab	Sub-tab	Section	#	Label	Setting / Value / State
Output		Waveform Generators	1	Modulation	Sine 11
Output		Waveform Generators	2	Modulation	Sine 22
Output		Sine Generators	1	Osc	1
Output		Sine Generators	2	Osc	1
Output		Oscillators		AWG Oscillator Control	ON

In the following AWG sequencer program, we generate a series of 4 dual-channel square pulses that are played back-to-back. We initialize the oscillator phase by a `resetOscPhase` instruction. In this form without an argument, the instruction will reset the phases of all oscillators acessible by this core (here oscillators 1 through 4). Alternatively, an argument in binary representation, e.g. `0b0101`, allows us to reset only a subset of these oscillators. We then set the phases of sine generators 1 and 2 to zero using the `setSinePhase` instruction. Subsequently, we play back the dual-channel waveform 4 times, and after each playback instruction, we increase the phase of sine generator 2 by 60 degrees. The corresponding instruction incrementSinePhase takes effect at the end of the previous waveform playback, which allows us to change the phase precisely in between waveforms. Upload the following sequence program to the AWG and run the sequence.

```
wave w_pulse = ones(800);

while (true) {
  // generate trigger for scope
  setTrigger(1);
  setTrigger(0);
  // initialize oscillator frequency, reset phase
  wait(100);
  resetOscPhase();
  setSinePhase(0, 0);
  setSinePhase(1, 0);
  wait(100);

  // play waveform and change sine generator phase
  playWave(w_pulse, w_pulse);

  incrementSinePhase(1, 60); // phase increment takes effect at the end
```

```
                                      // of the previous waveform
    playWave(w_pulse, w_pulse);
    incrementSinePhase(1, 60); // phase increment takes effect at the end
                                      // of the previous waveform
    playWave(w_pulse, w_pulse);
    incrementSinePhase(1, 60); // phase increment takes effect at the end
                                      // of the previous waveform
    playWave(w_pulse, w_pulse);
    wait(10000);
}
```

The resulting signal shown in Figure 3.16 nicely shows how the relative phase of the two signal, starting out at 0, incrementally changes in steps of 60 degrees. The end of the first waveform is marked by a cursor. In the 4th waveform, the two carrier signals are offset by exactly 180 degrees.
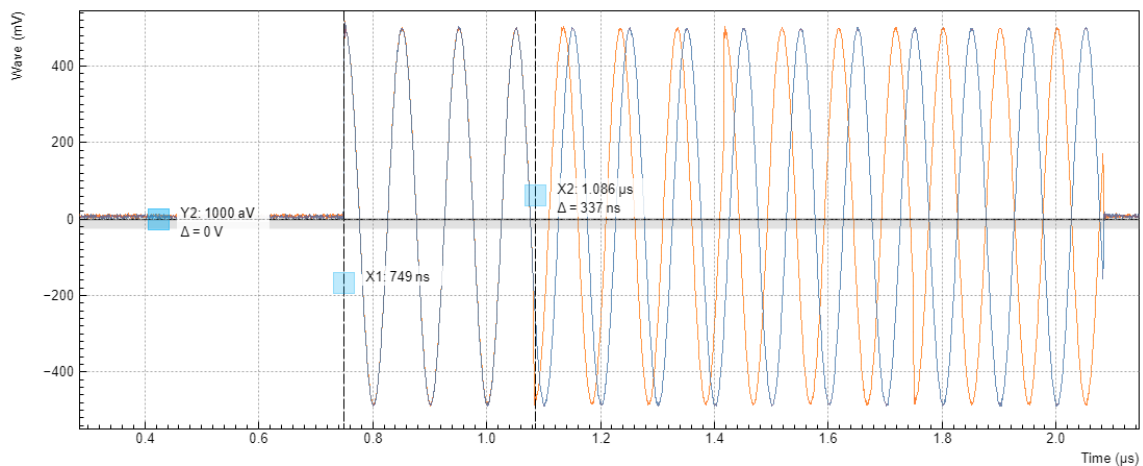
Figure 3.16. Amplitude-modulated dual-channel signal with rapid real-time phase increments generated by the HDAWG.

3.3.5. Multi-frequency Modulation

When the HDAWG-MF Multi-frequency option is installed, the HDAWG supports modulation of multiple carriers with individual envelope signals. This technique allows us to generate signals which, using conventional sample-by-sample waveform programming, would require to store thousands of waveforms instead of just one envelope waveform. Typical use cases are phase cycling protocols in NMR spectroscopy, or frequency-multiplexing techniques. The latter requires the HDAWG-MF option, whereas the former requires only the base instrument. Multi-carrier modulation is realized by four-fold interleaving of one AWG channel and individual multiplication of the four channels with one of the oscillator signals. The envelope sampling rate is therefore reduced by a factor of 4, whereas the carrier signal is still generated at the full sampling rate, therefore giving access to the full bandwidth.

With the following sequence program, we generate a series of pulses with changing carrier. The first four pulses each have a single carrier coming from one of the first four oscillators. In the fifth pulse, all carrier signals are superimposed. The interleave instruction, which we use several times, allows us to combine four waveforms into one.

```
const n_samples = 500;
wave w_gauss = 0.25*gauss(n_samples, n_samples/2, n_samples/10);
wave w_zeros = zeros(n_samples);

wave w_channel_1 = interleave(w_gauss, w_zeros, w_zeros, w_zeros);
wave w_channel_2 = interleave(w_zeros, w_gauss, w_zeros, w_zeros);
wave w_channel_3 = interleave(w_zeros, w_zeros, w_gauss, w_zeros);
```

```
wave w_channel_4 = interleave(w_zeros, w_zeros, w_zeros, w_gauss);
wave w_all_channels = interleave(w_gauss, w_gauss, w_gauss, w_gauss);

while (true) {
  playWave(w_channel_1);
  playWave(w_channel_2);
  playWave(w_channel_3);
  playWave(w_channel_4);
  setTrigger(1);
  playWave(w_all_channels);
  setTrigger(0);
  waitWave();
  wait(1000);
}
```

In the Multi-frequency modulation tab, we route the oscillator signals 1 to 4 to the AWG Output 1 Modulation sine generators 1 to 4 by changing the selectors in the Osc column. We set the frequencies of oscillators 1 to 4 in the Output tab to mutually different values. Deliberately, we offset oscillator 4 frequency by 1 Hz from a round value to 40.000001 MHz. This helps us demonstrate that the phase coherence is maintained independently by all oscillators. Finally, in the Output tab we set Modulation to Advanced.

Table 3.17. Settings: Configure the AWG output and oscillator signals

Tab	Sub-tab	Section	#	Label	Setting / Value / State
Output		Oscillators	1	Frequency	10 MHz
Output		Oscillators	2	Frequency	20 MHz
Output		Oscillators	3	Frequency	30 MHz
Output		Oscillators	4	Frequency	40.000001 MHz
MF Mod		AWG Output 1 Modulation	1	Osc	1
MF Mod		AWG Output 1 Modulation	2	Osc	2
MF Mod		AWG Output 1 Modulation	3	Osc	3
MF Mod		AWG Output 1 Modulation	4	Osc	4
Output		Waveform Generators	1	Modulation	Advanced

Figure 3.17 shows the signal captured by the scope after having made all the settings and uploaded and played the sequence program above. In the first three pulses (10, 20, and 30 MHz) the phase is stable because the repetition rate is commensurate with the carrier frequencies. The fourth pulse (40.000001 MHz) and the fifth pulse (sum of all carriers) vary their shape over time, which is made visible by enabling scope persistence. As mentioned in the beginning, generating such a signal conventionally would consume a much larger amount of waveform memory.
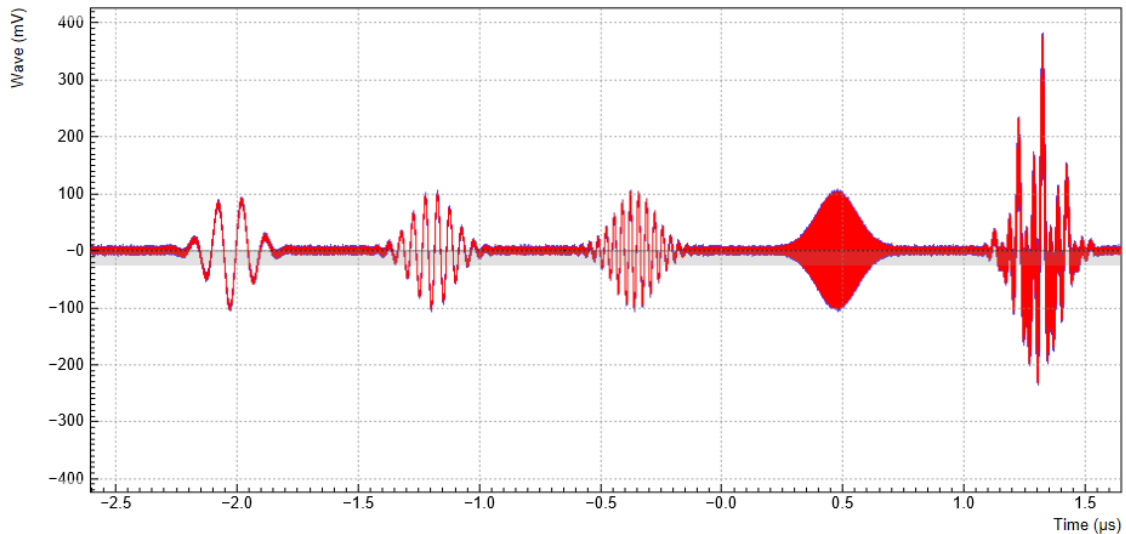
Figure 3.17. Amplitude-modulated signal with multiple carriers generated by the AWG and captured by the scope with persistence.

3.3.6. Memory-efficient Sequencing with the Command Table

The command table allows the sequencer to group waveform playback instructions with other timing-critical phase/amplitude setting commands in a single instruction within one clock cycle of 3.33 ns. The command table is a unit separate from the sequencer and waveform memory. Both the phase and the amplitude can be set in absolute and in incremental mode. Also when not using digital modulation or amplitude setting, working with the command table has the advantage of being more efficient in sequencer instruction memory than standard sequencing. Starting a waveform playback with the command table always requires just a single clock cycle, as opposed to 2 or 3 when using a `playWave` instruction.

During runtime, three components play together to generate the waveform output and apply the phase and amplitude setting instructions:

- Command table: a list of up to 1024 indexed entries (command table index), each containing an index of a waveform to be played (wave table index), a sine generator phase setting instruction, and an AWG amplitude setting instruction. The command table is specified by a JSON formatted string written to the `<dev>/AWGS/<n>/COMMANDTABLE/DATA` API node

- Wave table: a list of up to 16000 indexed waveforms. This list is defined by the sequence program using waveform generation instruction such as `gauss` combined with the index assignment instruction `assignWaveIndex`. The index of a waveform (wave table index) is used in two ways: it is referred to from the command table, and it is used to directly write waveform data to the instrument memory using the `<dev>/AWGS/<n>/WAVEFORM/WAVES/ <index>` API node

- Sequencer: the unit executing the runtime instructions, namely in this context the `executeTableEntry` instruction. This instruction executes one entry of the command table, and its input argument is a command table index. In its compiled form represented in the AWG Sequencer Advanced sub-tab, the sequence program can contain up to 16384 instructions.

The following example illustrates how the different elements are programmed. Let's consider the following SeqC program:

```
wave w_a = gauss(256, 128, 10);
```

```
wave w_b =  ones(1024);
wave w_c = -ones(1024);

assignWaveIndex(w_a, 1);
assignWaveIndex(w_b, w_c, 4);

executeTableEntry(0);
executeTableEntry(1);
executeTableEntry(2);
executeTableEntry(1);
```

Compile and upload this program. This program will add two entries to the wave table: the single-channel Gaussian waveform `w_a` at index 1, and the dual-channel consisting of positive and negative square waveforms `w_b` and `w_c` at index 4. The remaining entries of the wave table, namely 0, 2, and 3, and indices larger than 4, remain empty and don't occupy any waveform memory.

It will not yet be possible to run the program. This is because through the multiple `executeTableEntry` instructions, it refers to a command table that is not yet specified. We can specify this command table using an upload of a JSON formatted string to the `<dev>/AWGS/<n>/COMMANDTABLE/DATA` API node using the `setVector` method. Please refer to the LabOne Programming Manual on instructions to write data to nodes from your preferred API. The following code example applies to the ziPython API and demonstrates how to upload an JSON string to the command table of AWG Core 1 (n=0):

```python
import zhinst.utils

# Connect to device
device_id = 'dev8000' # to be replaced by instrument serial number
apilevel_example = 6
(daq, device, _) = zhinst.utils.create_api_session(
        device_id, apilevel_example)

# Define JSON formatted string
json_str = """
{
  "$schema": "☖http://docs.zhinst.com/hdawg/commandtable/v2/schema",
  "header": {
    "version": "0.2"
  },
  "table": [
    {
      "index": 0,
      "waveform": {
        "index": 4
      }
    },
    {
      "index": 1,
      "waveform": {
        "index": 1
      }
    },
    {
      "index": 2,
      "amplitude0": {
        "value": 0.5
      }
    }
  ]
}
"""

# Upload command table
daq.setVector(f"/{device}/awgs/0/commandtable/data", json_str)
```

The table defined by the JSON string `json_str` contains 3 entries.

- The first entry (command table index 0) refers to the waveform with waveform index 4. Executing this entry will start the playback of `w_b` and `w_c` (the dual-channel positive/ negative square waveform in our sequence program).

- The second entry (command table index 1) refers to the waveform with waveform index 1. Executing this entry will start the playback of `w_a` (the Gaussian waveform in our sequence program).

- The third entry (command table index 2) contains no reference to a waveform, but instead an instruction to set the amplitude of output 1 of the AWG Core to 0.5 (see Section 4.3). Executing this entry will not generate a signal, but will affect all subsequent waveform playbacks on that AWG Core output.

After successful upload of the command table data, the sequence program on the instrument can now be run. When run, it will do the following:

1. Play a square pulse of amplitude 1.0 and length 1024 samples on output 1, and a square pulse of amplitude -1.0 and length 1024 samples on output 2

2. Play a Gaussian pulse of amplitude 1.0 and plength 256 samples on output 1

3. Set the AWG Core 1 output 1 amplitude scaling factor to 0.5

4. Play a Gaussian pulse of amplitude 0.5 and plength 256 samples on output 1

Note

If a sequence program contains a reference to a command table entry that has not been defined, or if a command table entry refers to a waveform that has not been defined, the sequence program can't be run.

In this example, the entries of the command table contain only a waveform playback instruction, or an amplitude setting instruction alone. In general each entry can combine a dual-channel waveform playback, a phase setting instruction (absolute or incremental) on both sine generators of the AWG core, and an amplitude setting instruction (absolute or incremental) on both outputs of the AWG core. In quantum computing applications, the combination of these instructions allows one to express a qubit gate operation, including a possible change of the carrier phase, in a single sequencer instruction.

The documentation of all possible parameters in the command table JSON file is contained in the publicly hosted JSON Schema file on http://docs.zhinst.com/hdawg/commandtable/v2/schema. The URL of this schema file should be referred from the JSON string when working on it in a suitable editor. In this way, you can easily access the documentation via auto complete and in-line help, as shown in the screenshot below for the example of Microsoft's Visual Studio Code.
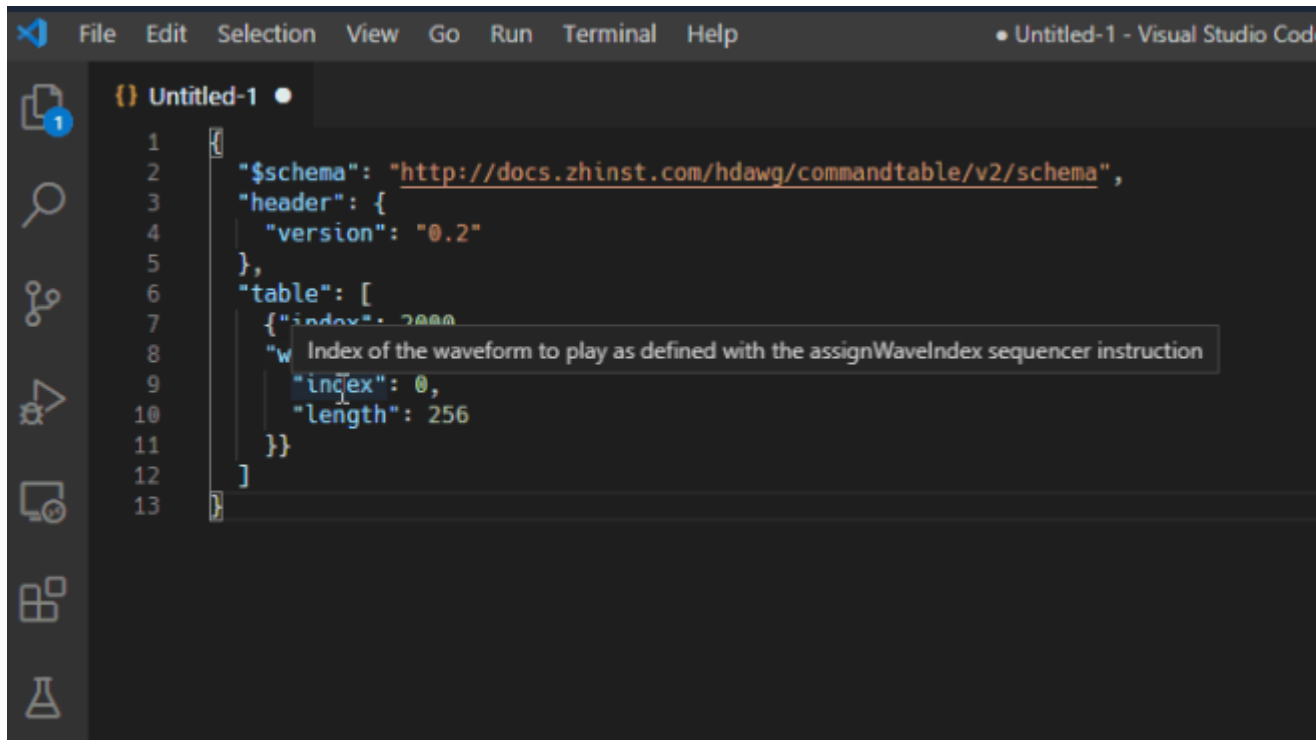
Figure 3.18. JSON editing with inline documentation in Visual Studio Code

Waveform data can be uploaded independently from a sequencer program, which is usually preferred when the waveform shape is computed within the API. For this case, the sequencer language has the feature to use placeholder waves in a sequence programs. In the above example, one may e.g. replace this line:

```
wave w_a = gauss(256, 128, 10);
```

by the following line:

```
wave w_a = placeholder(256);
```

When compiling the sequence program, LabOne will not generate and upload waveform data for w_a. Instead, these waveform data may be transferred subsequently using the API node <dev>/ AWGS/0/WAVEFORM/WAVES/1, where the wave table index 1 is set by the following line in the sequencer program:

```
assignWaveIndex(w_a, 1);
```

3.4. Basic Qubit Characterization

Note

This tutorial is applicable to all HDAWG Instruments.

3.4.1. Goals and Requirements

In this tutorial, we demonstrate how to generate pulse sequences for basic quantum bit (qubit) characterization. We develop a basic qubit characterization sequencer program in which we implement pulse sequences to measure a qubit's lifetime, and demonstrate how to perform Rabi, Ramsey, and spin-locking experiments.

To visualize the generated output signals of the HDAWG, an oscilloscope with sufficient bandwidth and at least 3 channels is required.

3.4.2. Preparation

Make sure that the instrument 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.

The tutorial starts with the instrument in the default configuration (e.g., after a power cycle) and the default user interface settings (e.g., as is after pressing F5 in the browser). We will monitor the HDAWG outputs using two channels of an external scope, and use a third scope channel for visualizing a trigger. Connect the HDAWG's outputs to the oscilloscope as follows:
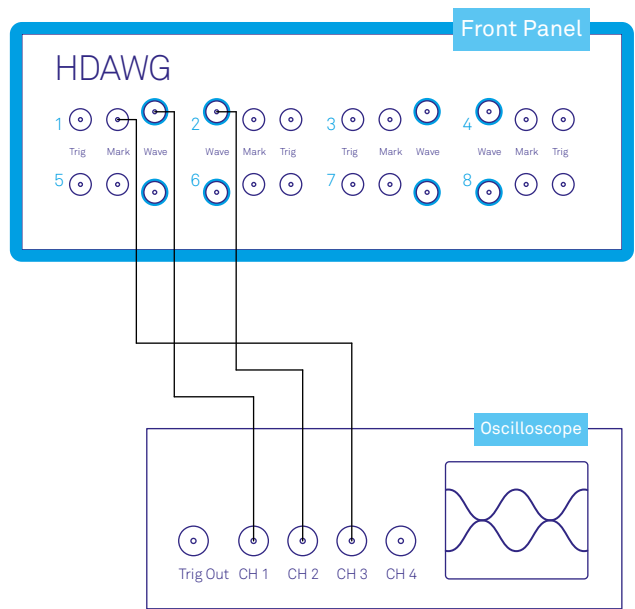
Figure 3.19. Connections for the qubit characterization tutorial

The following table summarizes the settings used to configure the external scope.

Table 3.18. Settings: Configure the external scope

Scope Setting	Value / State
Ch1-3 enable	ON
Ch1-2 range	0.2 V/div
Ch3 range	1 V/div
Timebase	5 us/div
Trigger source	Ch3
Trigger level	500 mV
Run / Stop	ON

Qubit Characterization Setup and Measurement Procedure

This tutorial can be directly applied to the circuit QED (cQED) measurement setup shown in Figure 3.20. This setup contains both the HDAWG and the UHFQA Quantum Analyzer. Note that it can be modified to accommodate other measurement systems or implementations with minimal effort. Here, a qubit control pulse (green) at microwave frequencies is generated with IQ-based upconversion using the signal outputs Wave 1 and 2 of the HDAWG. Upon receiving a trigger pulse (red) generated by the marker output (Mark 1) of the HDAWG, the qubit's state is then read out (blue) by measuring the signal transmission of the resonator using the UHFQA Quantum Analyzer in combination with IQ-based up-/downconversion to/from the microwave regime.
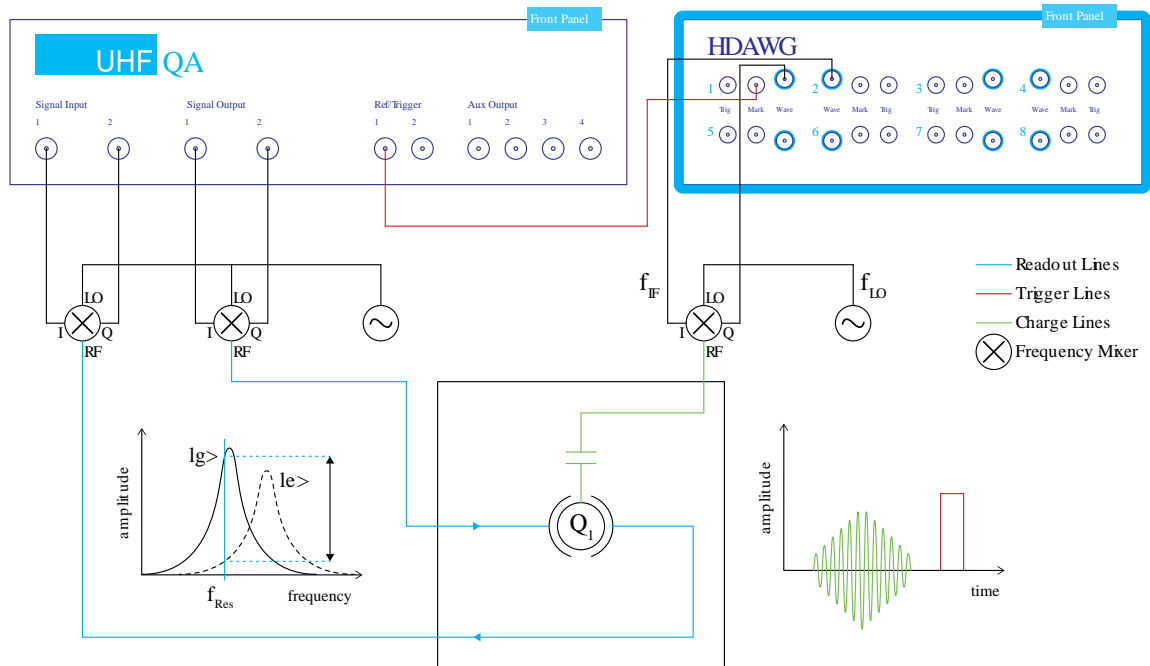
Figure 3.20. Experimental measurement setup and visualization of key signals. The element labeled as Q_1 denotes a qubit dispersively coupled to a resonator.

IQ-based up-conversion: An IQ-mixer generates a single amplitude- and phase-controlled signal at frequency f_{LO}-f_{IF} from two directly synthesized I and Q quadrature signals at intermediate frequency f_{IF} and a local oscillator at frequency f_{LO}. The otherwise identical I and Q signals only differ in their relative amplitude and phase of approximately -90° that need to be calibrated to suppress spurious signals.

Resonator readout: A qubit that is dispersively coupled to a resonator shifts the resonator's frequency depending on whether it is in its ground (|g⟩) or excited (|e⟩) state. Upon receiving a rising-edge trigger from the HDAWG at Trigger 1, the UHFQA detects this shift and determines the state of the qubit by measuring the change in amplitude and/or phase of a microwave pulse probing the resonator's transmission at a specific frequency f_{Res}, as indicated by the double arrow in the inset of the Figure.

Measurement Setup Variations for other Qubit Platforms

Different physical platforms such as cQED, spin qubits, ions, or atoms have varying requirements for their setups. In this tutorial, we will refer to the following variations to highlight small changes to the workflow:

- **Variation 1**: Qubit control pulse (atoms/ions). In an atom/ion platform, single-qubit manipulation is often performed by controlling the intensity of a laser with an acousto-optic modulator. In this case, a **single** channel of the HDAWG is sufficient to directly synthesize a control pulse at the required carrier frequency (typically between 80 MHz and 200 MHz). The phase-sensitive applications of the tutorial may no longer pertain unless a dedicated system enables phase control of the qubit-addressing lasers.

- **Variation 2**: Square-shaped readout pulse (cQED). In the absence of a dedicated readout-pulse synthesizer, a square pulse is often generated using a microwave switch and a TTL-compatible trigger signal. The Marker output (Mark) of the HDAWG is able to generate a suitable signal that controls the length of the readout pulse.

3.4.3. Generic Sequence Structure

In this part we discuss a generic qubit characterization sequencer program and the HDAWG setup required for this task. The goal is to generate a simple qubit control pulse using this program. All other qubit characterizations discussed in this tutorial will use the same structure and setup.

When programming qubit control sequences with the HDAWG, it is advisable to only program pulse envelopes, and use the integrated oscillators and sine generators that come with every HDAWG to generate the carrier signals needed for IQ-based up-conversion (Section 3.3.3). This approach has two advantages. First, it avoids the explicit programming of pulse modulations in the sequencer program sample by sample. Second, it is directly scalable for frequency-multiplexed multi-qubit control. Note that the phase control of the sine generators is a useful tool for qubit control, e.g. it can be used to apply single-qubit gates along every axis of the Bloch sphere. The following table summarizes the settings for the HDAWG Wave outputs.

Table 3.19. Settings: enable the HDAWG outputs and oscillator control

Tab	Sub-tab	Section	#	Label	Setting / Value / State
Output		Oscillators	1	Frequency (Hz)	10 MHz
Output		Oscillators		AWG Oscillator Control	ON
Output		Waveform Generators	1	Modulation	Sine 11
Output		Waveform Generators	2	Modulation	Sine 22
Output		Wave Outputs	1	Range	0.8 V
Output		Wave Outputs	2	Range	0.8 V
Output		Wave Outputs	1	Enable	ON
Output		Wave Outputs	2	Enable	ON

Note

Depending on the experimental setup, the oscillator frequency (f_{IF}) should be set such that the final pulse frequency f_{LO}-f_{IF} is resonant with the qubit frequency.

Variation 1: The recommended value for f_{IF} is the center frequency of the acousto-optic modulator.

Note

The AWG Oscillator Control button can only be enabled if the oscillators are correctly matched to the sine generators, e.g. as given by the standard settings.

Note

Depending on the experimental setup, e.g. the insertion loss of the IQ-mixer in use, the 'Range' of Wave Outputs 1 and 2 should be adjusted accordingly.

Figure 3.21 illustrates the generic structure for all sequence programs used in this tutorial. It consists of two parts, the 'waveform definition' and the 'pulse-pattern player'. The segments in light blue (segments 1, 3, 4, 6) are common to all the following examples; only the ones in dark blue (segments 2 and 5) will change.

Waveform Definition

1. Common Parameter Definition

2. Sequence-specific Parameter Definition

3. Waveform Definition

Pulse-pattern Player

4. (Re)set IF Phases

5. Play Control-pulse Pattern

6. Play Readout Pulse Trigger

Variations (Length)

Averages
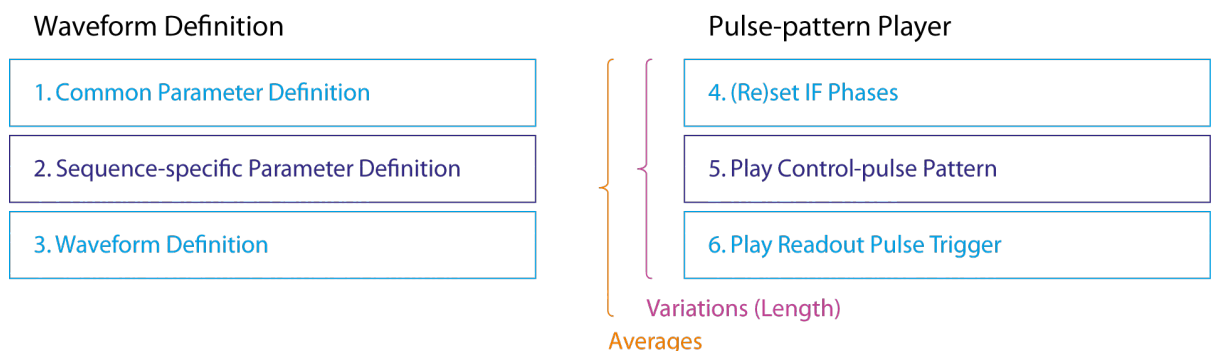
Figure 3.21. Structure of the qubit characterization sequencer program

To start, we generate a single qubit control pulse. The following sequencer program implements a Gaussian pulse within the above sequence structure and may be copied into the HDAWG Sequence Editor as is.

```
//// Waveform definition

//// (1) Common Parameter Definition
// Clock parameters
const sample_clock = 2.4e9;  // Hz
const sequencer_clock = sample_clock/8;  // Hz

// Control-pulse parameters
const pulse_length = 1.9e-6;  // s
const pulse_width = 0.187e-6;  // s
const pi_amp = 1;  // relative to HDAWG output range
const pi2_amp = 0.5;  // relative to HDAWG output range
const rel_iq_amplitude = 0.9;  // relative to programmed pulse height
                               // (dependent on mixer calibration)
```

```
const rel_iq_phase = 270;  // deg (dependent on mixer calibration)

// Readout-pulse parameters
const readpulse_time = 12e-9;  // s; account for cable-length mismatch
const readpulse_length = 2e-6;  // s; define trigger length for readout pulse
const qubit_lifetime = 100e-9;  // s

// Convert time to HDAWG samples and round to sequencer resolution
const pulse_length_samples =  round(pulse_length*sample_clock/16)*16;
const pulse_width_samples = round(pulse_width*sample_clock);
const readpulse_time_samples =  round(readpulse_time*sequencer_clock);
const readpulse_length_samples = round(readpulse_length*sequencer_clock);
const qubit_lifetime_samples = round(qubit_lifetime*sequencer_clock);

//// (2) Sequence-specific Parameter Definition
const averages = 1;  // number of repetitions for signal averaging
const length = 1;  // number of parameter variations through experimental sequence

//// (3) Waveform Definition
wave gauss_pulse = gauss(
  pulse_length_samples, 1, pulse_length_samples/2, pulse_width_samples
);

//// Pulse-pattern Player

var j;
for (j = 0; j < averages; j = j + 1) {
  cvar i;
  for (i = 0; i < length; i = i + 1) {
    //// (4) (Re)set IF phases
    resetOscPhase(1);
    setSinePhase(0, rel_iq_phase);
    wait(30);

    //// (5) Play Control-pulse Pattern
    playWave(pi_amp*gauss_pulse, pi_amp*rel_iq_amplitude*gauss_pulse);
    waitWave();
    wait(readpulse_time_samples);

    //// (6) Play Readout Pulse Trigger
    setTrigger(1);
    wait(readpulse_length_samples);
    setTrigger(0);

    wait(5*qubit_lifetime_samples);
  }
}
```

In segment (1) in **waveform definition**, we define two clock frequencies, the `sample_clock` based on which the pulses are synthesized, and the `sequencer_clock` for instructions from the sequencer. For all qubit characterizations, we use Gaussian pulses of a fixed waveform length (`pulse_length`) and pulse width (`pulse_width`) and only vary their pulse amplitudes. The two most important pulse amplitudes, `pi_Amp` and `pi2_Amp`, correspond to a π and a π/2-rotation of the qubit, respectively. These can be determined from a Rabi measurement (Section 3.4.4). The relative amplitude and phase shift of the two quadratures we generate with the HDAWG and play at the Wave outputs depend on the IQ mixer calibration. We define them using `rel_iq_amplitude` and `rel_iq_phase`. With `readpulse_time`, we can align the start of the readout pulse with the last qubit characterization pulse. This parameter allows us to compensate the difference in delay between the control and readout signal path, e.g. due to different cable lengths. The parameter `readpulse_length` defines the readout pulse length in **Variation 2**. We wait for a multiple of the qubit's lifetime (`qubit_lifetime`) after the readout pulse before generating the next control pulse in order to let the qubit relax back to its ground state. At the end of segment (1), we convert the parameters expressed in units of time into units of samples or sequencer clock cycles to conform with the main rules for a sequencer program:

- All arguments of wait()-functions need to be integers of the sequencer clock cycles (here: 3.3 ns).
- All waveform lengths need to be multiples of 16 sample-clock cycles to comply with the waveform granularity specification.

Note
The pulse parameters affecting the shape of a waveform, e.g. the pulse width, don't need to be rounded off even if they are in units of samples.

We set the number of different pulse patterns (`length`) and averages of the experiment (`averages`) to 1 in segment (2). These correspond to the UHFQA Result Length and Averages settings, respectively. In segment (3), we program a Gaussian pulse with a length of `pulse_width` and the full-range amplitude.

In the **pulse-pattern player** part we loop over all different pulse patterns and repeat the full pulse sequence `averages` times. Note that the sequencer code prohibits the use of a compile-time variable (type `cvar`) in a `repeat` statement. Therefore, we use a `for` loop with a run-time variable (type `var`) when looping over the averages. In each pattern, we first reset the phase of oscillator 1 in segment (4), program the relative phase between the two HDAWG Wave outputs (`rel_iq_phase`) and then give the system enough time (here 30 clock cycles) for the implementation. In segment (5) we play a single Gaussian pulse with an amplitude corresponding to `pi_Amp` multiplied with the Range setting of the HDAWG Wave output, and we wait until the end of the pulse to start the readout sequence in segment (6). Here, we generate a trigger pulse of length `readpulse_length` on Trig 1 by first setting trigger configuration 1 (Trig 1: high) and then, after `readpulse_length`, setting trigger configuration 0 (all Trig: low). We wait for 5 times the qubit lifetime (`qubit_lifetime`) before starting the next pulse pattern.

After uploading this sequence program to the instrument and starting the HDAWG, we measure the following signal with the oscilloscope set to a time base of 500 ns:
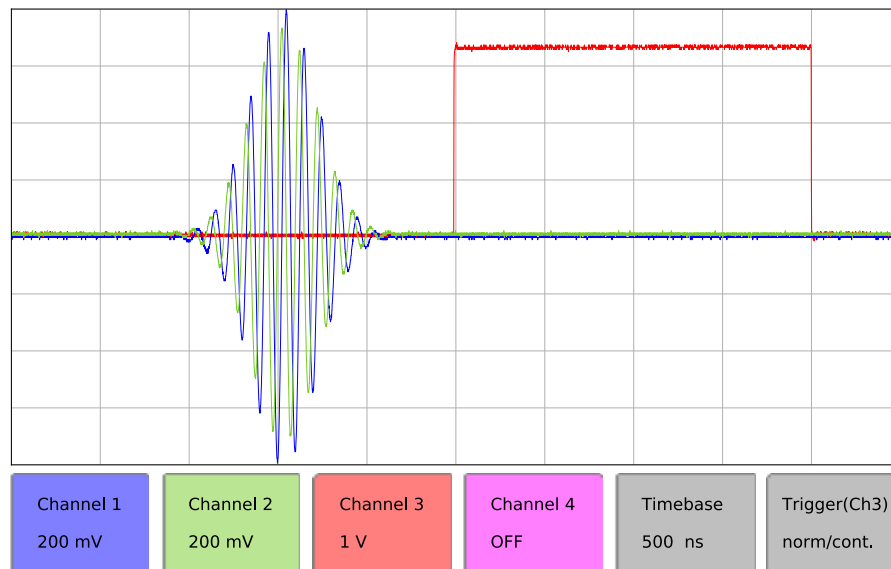

Figure 3.22. HDAWG output signals for a qubit control pulse as measured with an external oscilloscope

The two Gaussian pulses generated at the HDAWG signal outputs Wave 1 and 2 are depicted in blue and green, respectively. As a result of the IQ-calibration parameters, the Wave 2 output has a

10% smaller amplitude and a -90° phase shift with respect to the Wave 1 output. As intended, the pulse on channel 1 (Wave 1) extends over the full output range of the HDAWG (1.6 Vpp). The trigger for the readout pulse (red) starts after the pulses and extends over the programmed length of 2 μs (`readpulse_length`). Its starting time may be adjusted by changing `readpulse_time`.

3.4.4. Rabi Oscillation Measurement

In a Rabi oscillation measurement, the qubit is first driven continuously using a control pulse of fixed width and variable amplitude, or of variable width and fixed amplitude, and then read out. In this example, we create a pulse sequence of `length` pulses with amplitudes varying from 0 to the maximum HDAWG range. In a realistic experiment, `length` might be of the order of 100. Here we generate only 4 pulses, such that we can fit the entire pattern on a single oscilloscope shot. In segment (2) of the generic sequence program structure, we thus set the parameter `length` to 4 and add the following code line: `const amp_step = 1/(length-1);`. In segment (5), we replace the `playWave` statement with: `playWave(i*amp_step*gauss_pulse, i*amp_step*rel_iq_amplitude*gauss_pulse);`. Note that the new `playWave`-statement is an elegant way of redefining `gauss_pulse` in every variation and it is only possible because `i` is declared as a compile-time variable (type `cvar`).

After uploading this sequence program to the instrument and starting the HDAWG, we measure the following pulse sequence with the oscilloscope set to a time base of 2 μs:

Channel 1	Channel 2	Channel 3	Channel 4	Timebase	Trigger(Ch3)
200 mV	200 mV	1 V	OFF	2 μs	norm/cont.

Figure 3.23. HDAWG output signals for a Rabi oscillation measurement as measured with an external oscilloscope

From the results of this measurement, we determine the control-pulse amplitudes at which the qubit undergoes a π or π/2 rotation. These amplitudes are parametrized as `pi_amp` and `pi2_amp` in our sequence program, respectively. In the following examples, `pi_amp` and `pi2_amp` are assumed to be equal to the full and to half of the HDAWG output range, respectively.

3.4.5. Qubit Lifetime Measurement

To measure a qubit's lifetime, we first excite the qubit using a calibrated π pulse and then wait for a variable amount of time before starting the qubit readout. The signal measured with the UHFQA decays exponentially on the timescale of the qubit's lifetime. To implement this by starting from

the generic sequence program, we set the parameter `length` to 8 and add `const wait_step = 100` in segment (2). We also replace segment (5) with:

```
//// (5) Play Control-pulse Pattern
playWave(pi_amp*gauss_pulse, pi_amp*rel_iq_amplitude*gauss_pulse);
waitWave();
wait(i*wait_step+readpulse_time_samples);
```

After uploading this sequence program to the instrument and starting the HDAWG, we measure the following pulse sequence with the oscilloscope set to a time base of 5 μs:

Figure 3.24. HDAWG output signals for a lifetime measurement as measured with an external oscilloscope

As expected, we generate 8 pulses with the π pulse amplitude (blue and green), and with an increasing delay of the readout pulse (red).

3.4.6. Ramsey Fringe Measurement

A Ramsey fringe measurement is performed in four steps. First, we excite the qubit to an equal superposition between ground and excited state using a π/2 pulse. Then, we wait for a variable amount of time before applying a second π/2 pulse followed by qubit readout. When the qubit is driven resonantly, the signal decays on the timescale of the coherence time. When driven off-resonantly, the signal also oscillates at the detuning frequency. Starting from the generic sequencer program, in segment (2) we set the parameter `length` to 6 and add `const wait_step = 200.` We then replace segment (5) with

```
//// (5) Play Control-pulse Pattern
playWave(pi2_amp*gauss_pulse, pi2_amp*rel_iq_amplitude*gauss_pulse);
waitWave();
wait(i*wait_step);
playWave(pi2_amp*gauss_pulse, pi2_amp*rel_iq_amplitude*gauss_pulse);
waitWave();
wait(readpulse_time_samples);
```

After uploading this sequence program to the instrument and starting the HDAWG, we measure the following pulse sequence with the oscilloscope set to a time base of 5 μs:
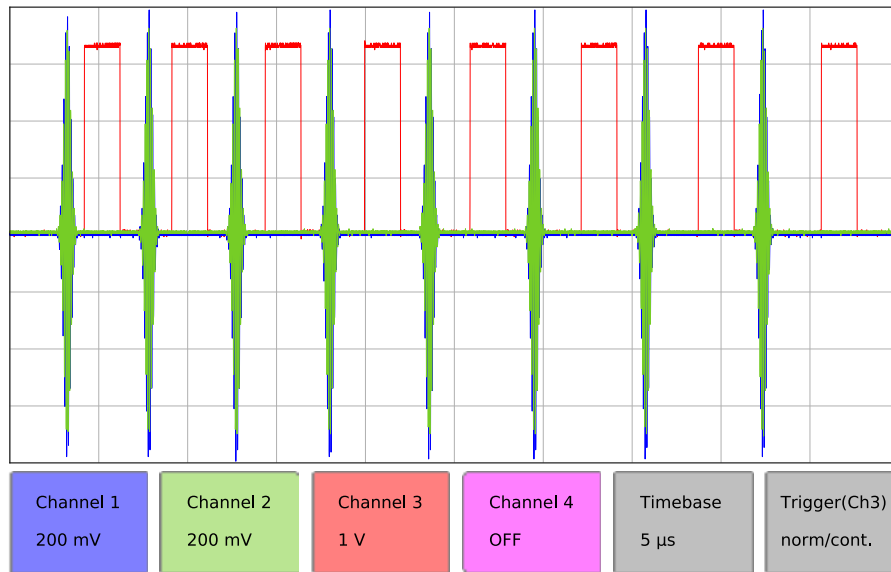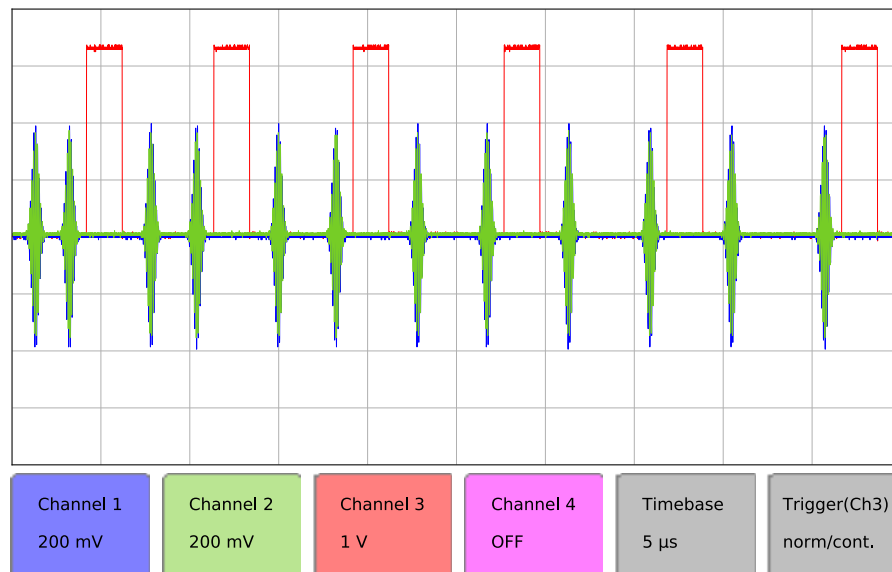
Figure 3.25. HDAWG output signals for a Ramsey fringe measurement as measured with an external oscilloscope

As expected, we see 6 pulse sequences of two consecutive π/2 pulses (blue and green) with increasing temporal separation. Indeed, the `playWave`-statement correctly plays two Gaussian control pulses with an amplitude of `pi2_amp` that bring the qubit first to an equal superposition state, and then to the excited state. Note that only the pulses, not the `wait` statement between the pulses, contributes to the waveform memory use. Hence, the wait time between the π/2 pulses can be extended to several seconds.

3.4.7. Spin Locking

Spin locking can be used to increase the dephasing time of a qubit by reducing relaxation via T_2-processes. To lock the spin, we first bring the qubit into an equal superposition between ground and excited state using a π/2 pulse. We then lock the qubit state onto its position on the equator of the Bloch sphere by applying a strong pulse along the qubit's axis. Before readout, a second π/2 pulse is applied. Varying the duration of the spin-locking pulse at different pulse powers provides information about the dephasing rate of the qubit at different frequencies.

We program a spin-locking sequence from the generic sequence program by setting the parameter `length` to 5 in segment (2), and by adding `const wait_step = 200;` and `const phi=90;` in the same segment. This defines the step length by which we increase the spin-locking pulse and its relative phase of 90° with respect to the first π/2 pulse.

To play a spin-locking pulse that rises and falls smoothly, and extends over a variable amount of time, we use the Hold function of the HDAWG. We create a composite pulse that rises like a Gaussian, and then hold the last value for a variable amount of time specified by a sequencer `wait` statement before smoothly ending the pulse like a Gaussian. For this purpose, we enable the Hold setting for the first two Waveform Generators in the Output tab of the HDAWG and add the rising and falling edge waveforms to segment (3):

```
wave w_rise = cut(gauss_pulse, 0, pulse_length_samples/2-1);
wave w_fall = cut(gauss_pulse, pulse_length_samples/2, pulse_length_samples-1);
```

The Hold function holds the last value of a played waveform indefinitely. It is thus crucial that the length of the waveform ends at a multiple of 16 samples, otherwise it will be padded with zeros and Hold will maintain 0. To ensure that the length of `w_rise` and `w_fall` is a multiple of 16, we write:

```
const pulse_length_samples =  round(pulse_length*sample_clock/32)*32;
```

The spin-locking sequence is generated by the following code segment, which replaces segment (5) in the generic sequence program:

```
//// (5) Play Control-pulse Pattern

// pi/2 pulse
playWave(pi2_amp*gauss_pulse, pi2_amp*rel_iq_amplitude*gauss_pulse);
incrementSinePhase(0, phi);
incrementSinePhase(1, phi);
waitWave();
// spin-locking pulse
playWave(pi_amp*w_rise, pi_amp*rel_iq_amplitude*w_rise);
waitWave();
wait(i*wait_step);
playWave(pi_amp*w_fall, pi_amp*rel_iq_amplitude*w_fall);
incrementSinePhase(0, 360-phi);
incrementSinePhase(1, 360-phi);
waitWave();
// pi/2 pulse
playWave(pi2_amp*gauss_pulse, pi2_amp*rel_iq_amplitude*gauss_pulse);
waitWave();
wait(readpulse_time_samples);
```

This code segment increases the phase of the played pulse by 90° after the first π/2 pulse to rotate the pulse's phase onto the qubit's phase. The composite spin-locking pulse is then generated using the rising and falling Gaussian pulses interleaved by a `wait` statement. The HDAWG only stores the falling and rising edge of the pulse in the waveform memory, whereas the `wait` statement does not consume any memory. After the spin-locking pulse, the phase of the excitation pulse is rotated back to the original axis and a second π/2 pulse is played.

After uploading this sequence program to the instrument and starting the HDAWG, we measure the following pulse sequence with the oscilloscope set to a time base of 5 μs:

Channel 1	Channel 2	Channel 3	Channel 4	Timebase	Trigger(Ch3)
200 mV	200 mV	1 V	OFF	5 μs	norm/cont.
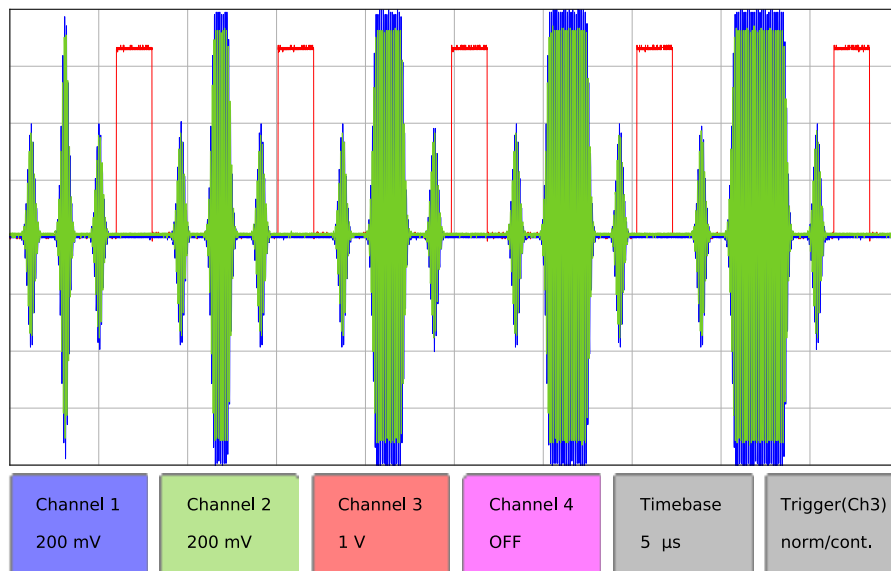
Figure 3.26. HDAWG output signals for a spin-locking measurement as measured with an external oscilloscope

We observe a series of 5 pulse patterns, each consisting of two short π/2 pulses surrounding a spin-locking pulse that covers the full HDAWG Output Range and is of variable length. To verify that the phase changes correctly between the pulses, we increase the parameter `pulse_width`

to 19 µs and zoom into the waveform between the first and second pulse. For `phi = 90` we observe a phase jump in both waveforms, whereas for `phi = 0` no phase jump is observed.

Chapter 4. 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 HDAWG. 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.

4.1. User Interface Overview

4.1.1. 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 HDAWG 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.
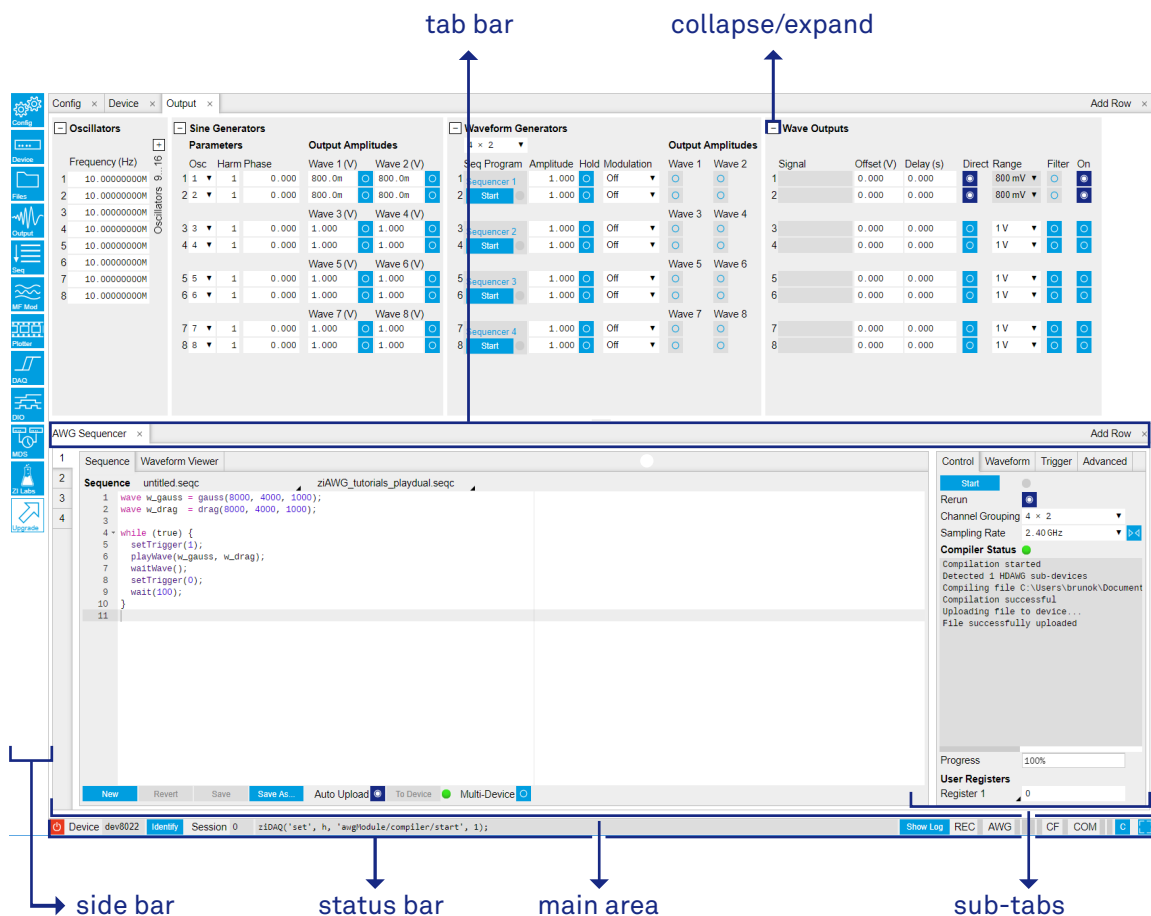
Figure 4.1. LabOne User Interface (default view)

Figure 4.1 shows the LabOne User Interface with the tabs opened by default after a new UI session has been started. The UI is by default 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 displaced by drag-and-drop within a row or across rows. Further items are highlighted in Figure 4.2.
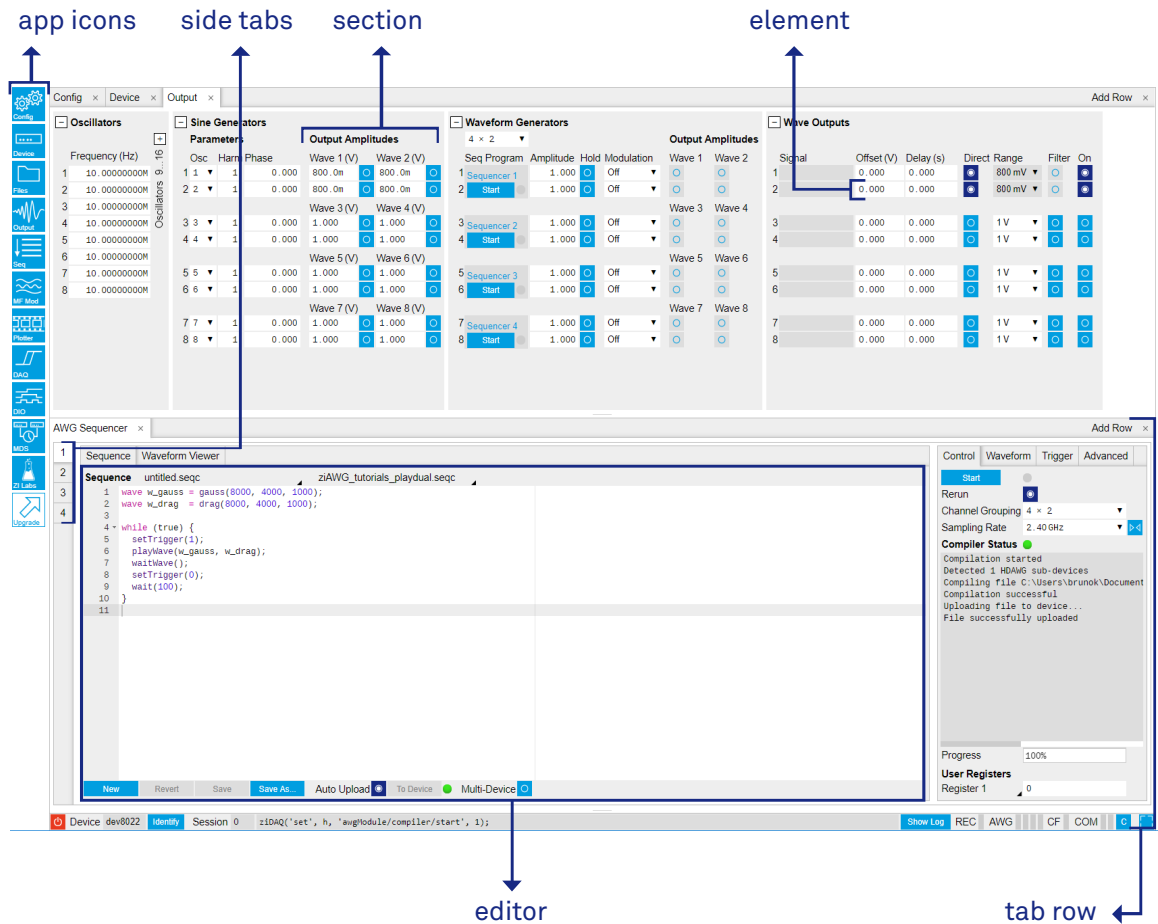
Figure 4.2. LabOne User Interface (more items)

Table 4.1 gives brief descriptions and naming conventions for the most important UI items.

Table 4.1. 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 indicators, warning lamps, 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	tab rows, each consisting of tab bar and the active tab area

Item name	Position	Description	Contains
		the top right corner of each tab row	
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

4.1.2. Unique Set of Analysis Tools

All instruments feature a comprehensive tool set for signal generation and sequence programming.

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

Table 4.2. Overview of app icons and short description

Control/Tool	Option/Range	Description
Output		Quick overview and access to all the settings and properties for signal generation and modulation.
Seq		Generate arbitrary signals using sequencing and sample-by-sample definition of waveforms.
MF Mod		Access to all the settings of the sine generators in advanced modulation mode.
Precompensation		Configure the Real-time Precompensation filters.
Counter		Configure the Pulse Counters for analysis of pulse trains on the digital signal inputs.
DAQ		Provides complex trigger functionality on all continuously streamed data samples and time domain display.
Sweeper		Sweep frequencies, voltages, and other quantities over a defined range and display various response functions including statistical operations.
DIO		Gives access to all controls relevant for the digital inputs

Control/Tool	Option/Range	Description
		and outputs including the Ref/Trigger connectors.
Config		Provides access to software configuration.
Device		Provides instrument specific settings.
Files		Access settings and measurement data files on the host computer.
MDS		Synchronize multiple instruments.
ZI Labs		Experimental settings and controls.

Table 4.3 gives a quick overview over the different status bar elements along with a short description.

Table 4.3. 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
Shutdown		Shuts down the instrument.
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:

Control/Tool	Option/Range	Description
		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.
CF	grey/yellow/red	Clock Failure - Red: present malfunction of the external 10 MHz reference oscillator. Yellow: indicates a malfunction 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.
C	C	Reset status flags: Clear the current state of the status flags
Full Screen		Toggles the browser between full screen and normal mode.

4.1.3. Plot Functionality

Several tools - Waveform viewer, Plotter, Data Acquisition, and Sweeper provide a graphical display of 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 plot is summarized in Table 4.4

Table 4.4. 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
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 Table 4.5.

Table 4.5. 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.

Control/Tool	Option/Range	Description
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		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 following Table 4.6 gives an overview of all the elements and their functionality. The chosen Signals and Operations are applied to the currently active trace only.

Table 4.6. 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

Control/Tool	Option/Range	Description
		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.

Control/Tool	Option/Range	Description
	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

Control/Tool	Option/Range	Description
		coefficient of determination to determine the goodness-of-fit.
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 $\sqrt{\dfrac{1}{N-1}\displaystyle\sum_{i=1}^{N}(x_i-\bar{x})^2}$ for the unbiased estimator of the sample standard deviation with a total of N samples x_i and an arithmetic average $\bar{x}$. The above formula 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).

Note

The fitting functions used in the Resonance Plot Math tool depend on the selected signal source. The demodulator R signal is fitted with the following function:

$$R(f) = C + A\frac{f}{\sqrt{f^2 + \left(\frac{Q}{f_0}\right)^2 \left(f^2 - f_0^2\right)^2}},\qquad(4.1)$$

where C accounts for a possible offset in the output, A is the amplitude, Q is the quality factor and f_0 is the center frequency. The demodulator Θ signal s fitted with the following function:

$$\theta(f) = \tan^{-1}\left(Q\frac{1 - \left(\frac{f}{f_0}\right)^2}{\frac{f}{f_0}} \right) \tag{4.2}$$

using the same parameters as above.

Tree Selector

The Tree selector allows one to access streamed measurement data in a hierarchical structure by checking the boxes of the signal 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.
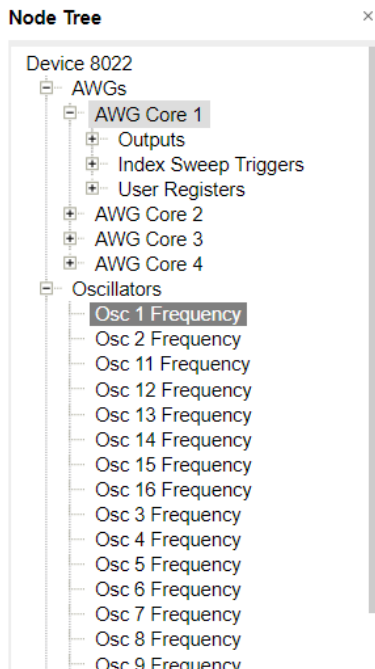
Figure 4.3. Tree selector with Display drop-down menu

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. This is particularly useful when operating a HDAWG together with a lock-in amplifier from the Zurich Instruments UHF, HF, or MF series. 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.
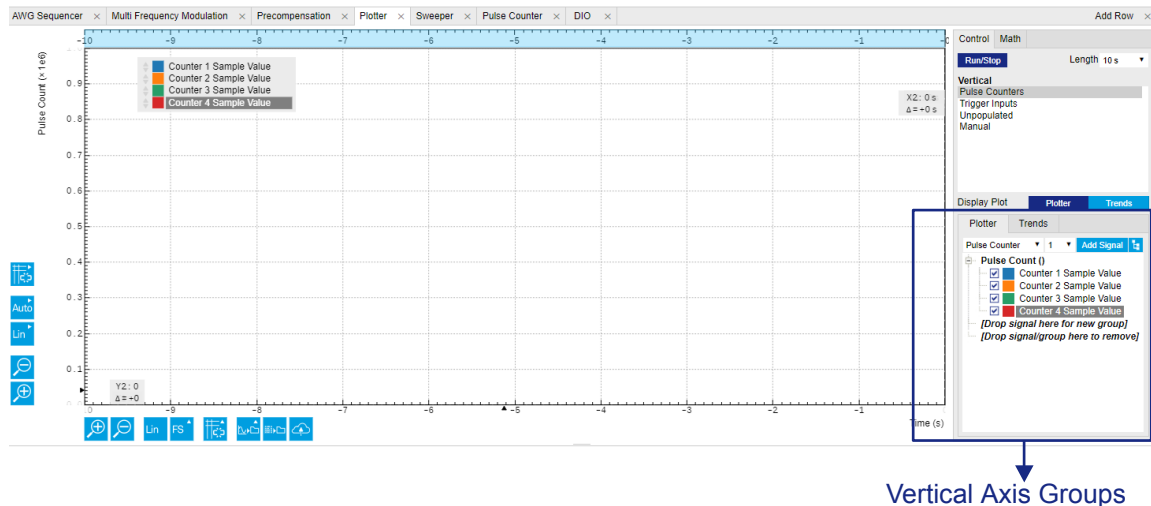
Figure 4.4. Vertical Axis Group in Plotter tool

The tick labels of only one axis group can be shown at once. This is the foregrouAWGnd 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.
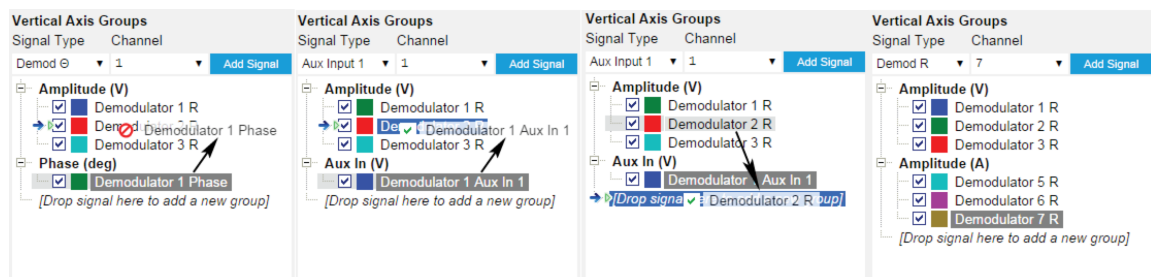
Figure 4.5. Vertical Axis Group typical drag and drop moves. Demodulator data are only available when using a Zurich Instruments lock-in amplifier from the UHF, HF, or MF series.

Table 4.7. 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	Pulse Counter	Select signal types for the Vertical Axis Group.
	Trigger Input	
	Trigger Miss	
Channel	integer value	Selects a channel to be added.
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 Plotter, and DAQ 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 group. 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.
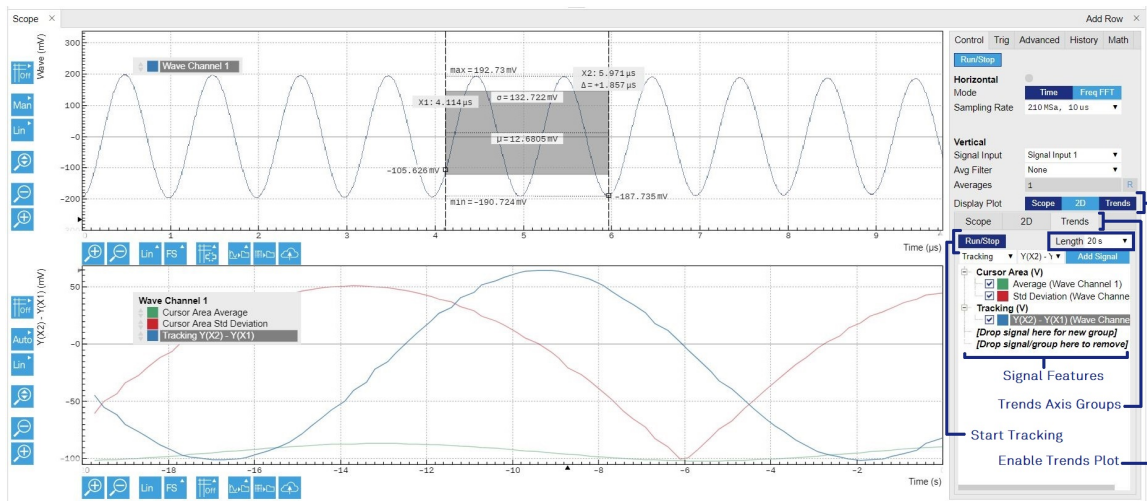
Figure 4.6. Top: main plot of the Scope tab showing the signal trace. Bottom: corresponding Trends plot tracking an average, standard deviation, and difference signal derived from the cursor positions in the main plot. The example shown is part of the HF2LI user interface. The controls of the Trends feature and their layout are very similar in all tabs and product platforms where this feature is available.

4.2. Saving and Loading Data

4.2.1. Overview

In this section we discuss how to save and record measurement data with the HDAWG Instrument using the LabOne user interface. This applies to instruments with the HDAWC-CNT Pulse Counter option installed, or to the case when data from a Zurich Instruments MF, HF, or UHF series lock-in amplifier is recorded with the same UI. In the LabOne user interface, there are 3 ways to save data:

- Saving the data that is currently displayed in a plot
- Continuously recording data in the background
- Saving trace data in the History sub-tab

Furthermore, the History sub-tab supports loading data. In the following, we will explain these methods.

4.2.2. Saving Data from Plots

A quick way of save data from any plot is to click on the Save CSV icon ▦ at the bottom of the plot to store the currently displayed curves as a comma-separated value (CSV) file to the download folder of your web browser. Clicking on ⊿⊑ will save a graphics file instead.

4.2.3. Recording Data

The recording functionality allows you to store measurement data continuously, as well as to track instrument settings over time. The Config tab gives you access to the main settings for this function. The Format selector defines which format is used: HDF5, CSV, or MATLAB. The CSV delimiter character can be changed in the User Preferences section. The default option is Semicolon.

The node tree display of the Record Data section allows you to browse through the different measurement data and instrument settings, and to select the ones you would like to record. For instance, the demodulator 1 measurement data is accessible under the path of the form Device 0000/Demodulators/Demod 1/Sample . An example for an instrument setting would be the filter time constant, accessible under the path Device 0000/Demodulators/Demod 1/Filter Time Constant.

The default storage location is the LabOne Data folder which can for instance be accessed by the Open Folder button ⊿. The exact path is displayed in the Folder field whenever a file has been written.

Clicking on the Record checkbox will initiate the recording to the hard drive. In case of demodulator and boxcar data, ensure that the corresponding data stream is enabled, as otherwise no data will be saved.
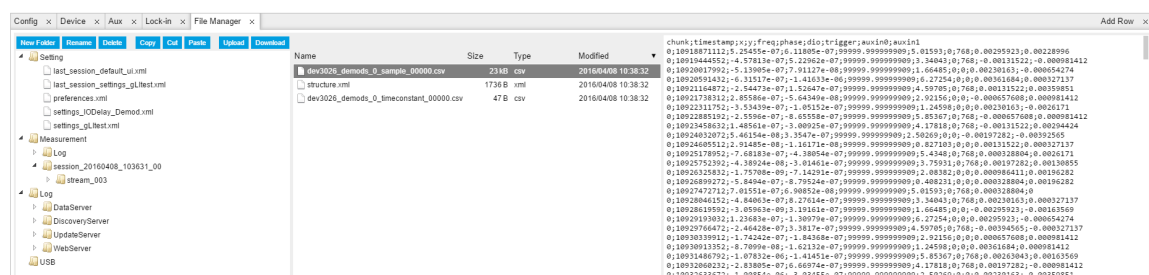
Figure 4.7. Browsing and inspecting files in the LabOne File Manager tab

In case HDF5 or MATLAB is selected as the file format, LabOne creates a single file containing the data for all selected nodes. For the CSV format, at least one file for each of the selected nodes is created from the start. At a configurable time interval, new data files are created, but the maximum size is capped at about 1 GB for easier data handling. The storage location is indicated in the Folder field of the Record Data section.

The File Manager (Files) tab is a good place to inspect CSV data files. The file browser on the left of the tab allows you to navigate to the location of the data files and offers functionalities for managing files in the LabOne Data folder structure. In addition, you can conveniently transfer files between the folder structure and your preferred location using the Upload/Download buttons. The file viewer on the right side of the tab displays the contents of text files up to a certain size limit. Figure 4.7 shows the Files tab after recording Demodulator Sample and Filter Time Constant for a few seconds. The file viewer shows the contents of the demodulator data file.

Note

The structure of files containing instrument settings and of those containing streamed data is the same. Streaming data files contain one line per sampling period, whereas in the case of instrument settings, the file usually only contains a few lines, one for each change in the settings. More information on the file structure can be found in the LabOne Programming Manual.

4.2.4. History List

Tabs with a history list such as Sweeper, Data Acquisition feature saving, autosaving, and loading functionality. By default, the plot area in those tools displays the last 100 measurements (i.e., depending on the tool, sweep traces, scope shots, DAQ data sets, or spectra), and each measurement is represented in as a list entry in the History sub-tab. The button to the left of each list entry controls the visibility of the corresponding trace in the plot; the button to the right controls the color of the trace. [1]Double-clicking on a list entry allows you to rename it. All measurements in the history list can be saved with the Save All . Clicking on the or Save Sel button (note the dropdown button ▼) saves only those traces that were selected by a mouse click. Use the Control or Shift button together with a mouse click to select multiple traces. The file location can be accessed by the Open Folder button ⊡. Figure 4.8 illustrates some of these features.

---

[1]Among the mentioned tools, the Scope is exceptional: it displays the most recent acquisition, and its display color is fixed. However, the Persistence feature represents a more specialized functionality for multi-trace display.
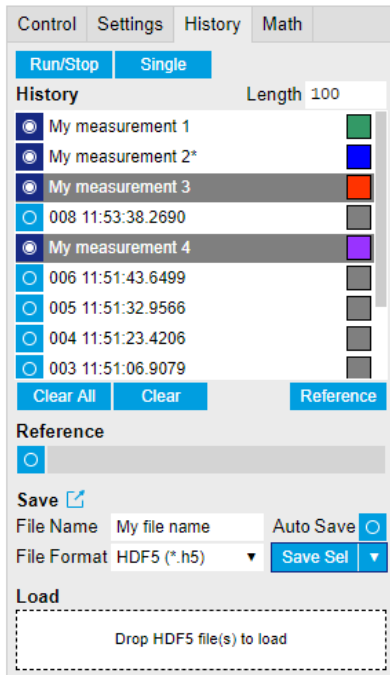
Figure 4.8. History sub-tab features. The entries "My measurement 1" etc. were renamed by the user. Measurement 1, 2, 3, 4 are currently displayed in the plot because their respective the left-hand-side button is enabled. Clicking on Save Sel would save "My measurement 3" and "My measurement 4" to a file, because these entries were selected (gray overlay) by a Control key + mouse click action.

Which quantities are saved depends on which signals have been added to the Vertical Axis Groups section in the Control sub-tab. Only data from demodulators with enabled Data Transfer in the Lock-in tab can be included in the files.

The history sub-tab supports an autosave functionality to store measurement results continuously while the tool is running. Autosave directories are differentiated from normal saved directories by the text "autosave" in the name, e.g. sweep_autosave_000. When running a tool continuously (Run/Stop button) with Autosave activated, after the current measurement (history entry) is complete, all measurements in the history are saved. The same file is overwritten each time, which means that old measurements will be lost once the limit defined by the history Length setting has been reached. When performing single measurements (Single button) with Autosave activated, after each measurement, the elements in the history list are saved in a new directory with an incrementing count, e.g. `sweep_autosave_001`, `sweep_autosave_002`.

Data which was saved in HDF5 file format can be loaded back into the history list. Loaded traces are marked by a prefix "loaded " that is added to the history entry name in the user interface. The `createdtimestamp` information in the header data marks the time at which the data were measured.

Figure 4.9 illustrates the data loading feature.

- Only files created by the Save button in the History sub-tab can be loaded.

- Loading a file will add all history items saved in the file to the history list. Previous entries are kept in the list.

- Data from the file is only displayed in the plot if it matches the current settings in the Vertical Axis Group section the tool. Loading e.g. PID data in the Sweeper will not be shown, unless it is selected in the Control sub-tab.

- Files can only be loaded if the devices saving and loading data are of the same product family. The data path will be set according to the device ID loading the data.
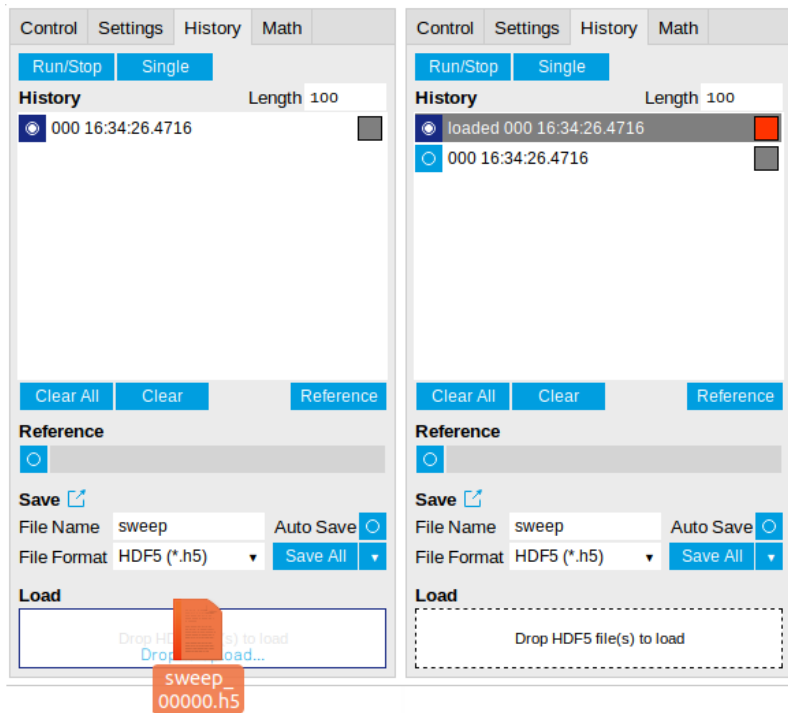
Figure 4.9. History data loading feature. Here, the file sweep_00000.h5 is loaded by drag-and-drop. The loaded data are added to the measurements in the history list.

4.2.5. Supported File Formats

HDF5

Hierarchical Data File 5 (HDF5) is a widespread memory-efficient, structured, binary, open file format. Data in this format can be inspected using the dedicated viewer HDFview. HDF5 libraries or import tools are available for Python, MATLAB, LabVIEW, C, R, Octave, Origin, Igor Pro, and others. The following example illustrates how to access demodulator data from a sweep using the h5py library in Python:

```
import h5py
filename = 'sweep_00000.h5'
f = h5py.File(filename, 'r')
x = f['000/dev3025/demods/0/sample/frequency']
```

The data loading feature of LabOne supports HDF5 files, while it is unavailable for other formats.

MATLAB

The MATLAB File Format (.mat) is a proprietary file format from MathWorks based on the open HDF5 file format. It has thus similar properties as the HDF5 format, but the support for importing .mat files into third-party software other than MATLAB is usually less good than that for importing HDF5 files.

SXM

SXM is a proprietary file format by Nanonis used for SPM measurements.

4.2.6. LabOne Net Link

Measurement and cursor data can be downloaded from the browser as CSV data. This allows for further processing in any application that supports CSV file formats. As the data is stored internally on the web server it can be read by direct server access from other applications. Most up-to-date software supports data import from web pages or CSV files over the internet. This allows for automatic import and refresh of data sets in many applications. To perform the import the applcation needs to know the address from where to load the data. This link is supplied by the LabOne User Interface. The following chapter lists examples of how to import data into some commonly used applications.

The CSV data sent to the application is a snap-shot of the data set on the web server at the time of the request. Many applications support either manual or periodic refresh functionality.

Since tabs can be instantiated several times within the same user interface, the link is specific to the tab that it is taken from. Changing the session on the LabOne User Interface or removing tabs may invalidate the link.

Supported applications:

- Excel
- Matlab
- Python
- C#.NET
- Igor Pro
- Origin

Excel

These instructions are for Excel 2010 (English). The procedure for other versions may differ.

1. In Excel, click on the cell where the data is to be placed. From the Data ribbon, click the "From Text" icon. The "Import Text File" dialog will appear.
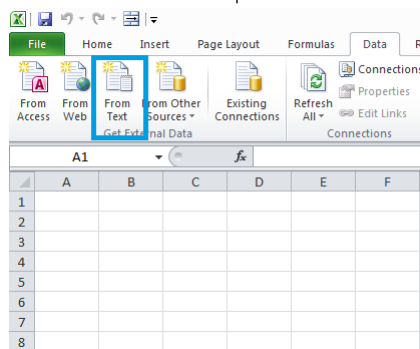
2. In LabOne, click the "Link" button of the appropriate Math tab. Copy the selected text from the "LabOne Net Link" dialog to the clipboard (either with Ctrl-C or by right clicking and selecting "Copy").
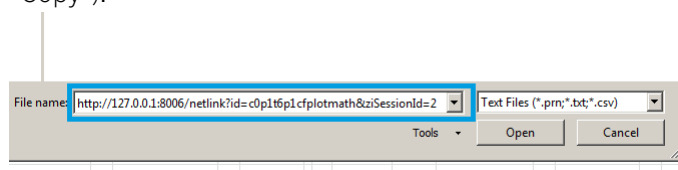
3. In Excel, paste the link into the "File name" entry field of the "Import Text File" dialog and click the "Open" button. This will start the text import wizard. Ensure that the "Delimited" button is checked before clicking the "Next" button.
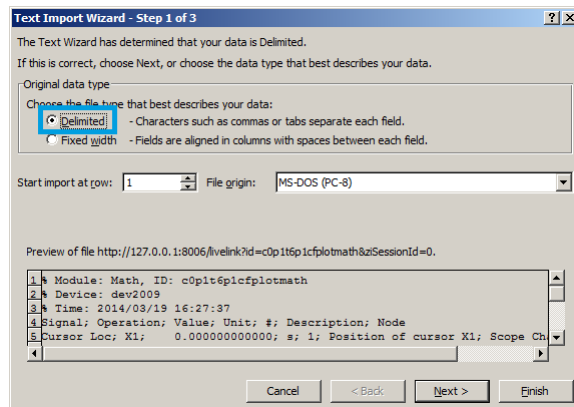
4. In the next dialog, select the delimiter character corresponding to that selected in LabOne (this can be found in the "Sessions" section of the Config tab). The default is semicolon. Click the "Next" button.
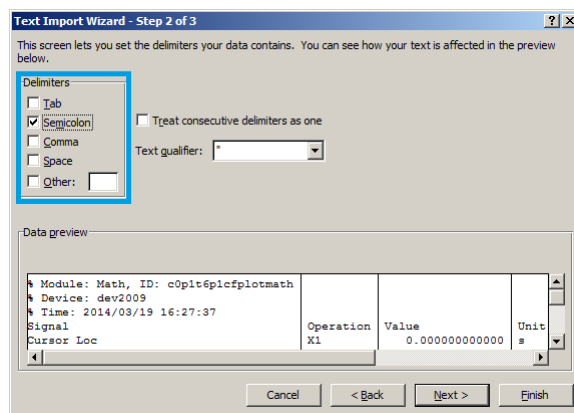
5. In the next dialog, click on "Finish" and then "OK" in the "Import Data" dialog. The data from the Math tab will now appear in the Excel sheet.
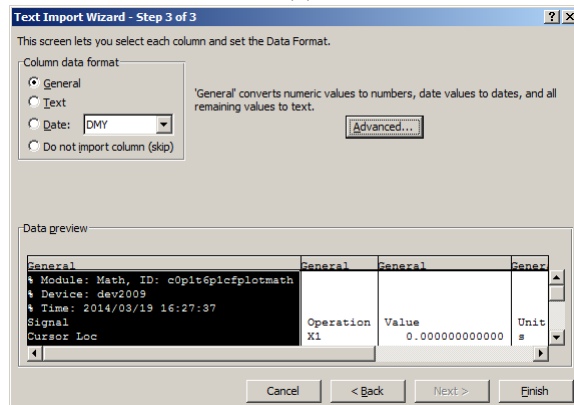
6. The data in the sheet can be updated by clicking the "Refresh All" icon. To make updating the data easier, the "Import text file" dialog can be suppressed by clicking on "Properties".

7. Deactivate the check box "Prompt for file name on refresh".
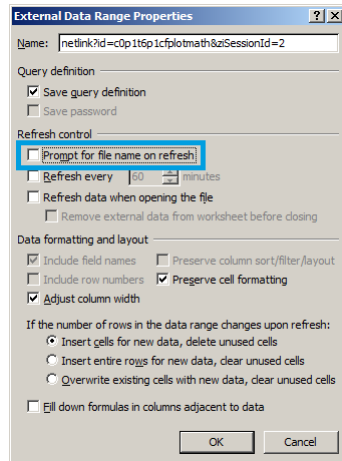
MATLAB

By copying the link text from the "LabOne Net Link" dialog to the clipboard, the following code snippet can be used in MATLAB to read the data.

```
textscan(urlread(clipboard('paste')),'%s%s%f%s%d%s%s','Headerlines', 4,'Delimiter',
 ';')
```

Python

The following code snippet can be used in Python 2 to read the LabOne Net Link data, where "url" is assigned to the text copied from the "LabOne Net Link" dialog.

```
import csv
import urllib2
url = "http://127.0.0.1:8006/netlink?id=c0p5t6p1cfplotmath&ziSessionId=0"
webpage = urllib2.urlopen(url)
datareader = csv.reader(webpage)
data = []
for row in datareader:
    data.append(row)
```

C#.NET

The .NET Framework offers a WebClient object which can be used to send web requests to the LabOne WebServer and download LabOne Net Link data. The string with comma separated content can be parsed by splitting the data at comma borders.

```
using System;
using System.Text;
using System.Net;

namespace ExampleCSV
{
  class Program
  {
    static void Main(string[] args)
    {
      try
      {
        WebClient wc = new WebClient();
        byte[] buffer = wc.DownloadData("http://127.0.0.1:8006/netlink?
id=c0p1t6p1cfplotmath&ziSessionId=0");
        String doc = Encoding.ASCII.GetString(buffer);
        // Parse here CSV lines and extract data
        // ...
        Console.WriteLine(doc);
      } catch (Exception e) {
        Console.WriteLine("Caught exception: " + e.Message);
      }
    }
  }
}
```

Igor Pro

These instructions are for Igor Pro 6.34A English. The procedure for other versions may differ.

1. For Igor Pro, the CSV separator has to be the comma. Set this in the LabOne Config tab as follows:
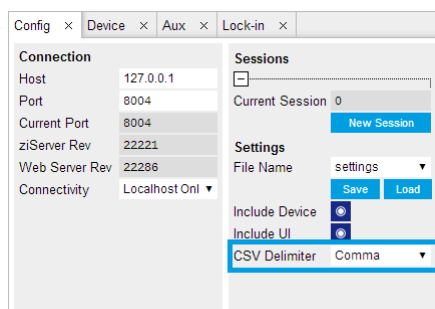
2. In Igor Pro, select the menu "Data->Load Waves->Load Waves...".
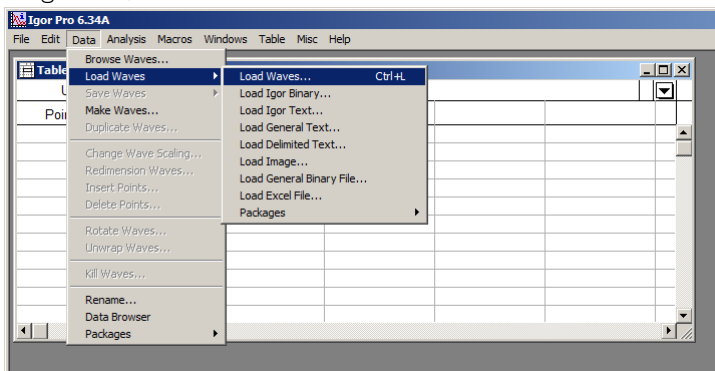
3. In the "Load Waves" dialog, click the "File…" button and paste the link text from the "LabOne Net Link" dialog into the entry field. Then click the "Tweaks…" button to open the "Load Data Tweaks" dialog.
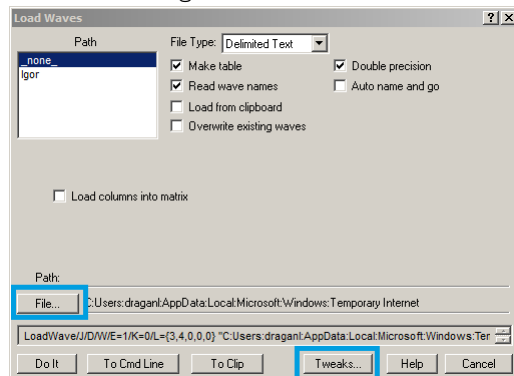
4. Adjust values as highlighted below and click "Return". The "Loading Delimited Data" dialog will appear.
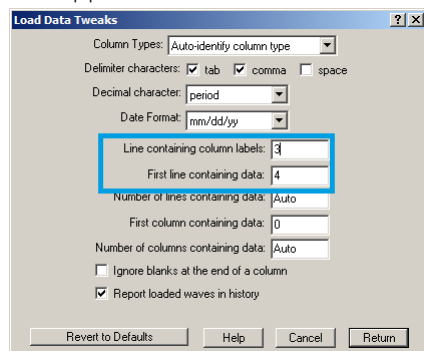
5. Click the "Load" button to read the data.
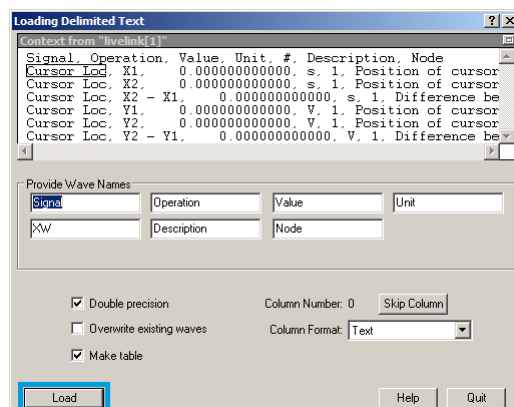
6. The data will appear in the Igor Pro main window.

Origin

These instructions are for Origin 9.1 English. The procedure for other versions may differ.

1. Open the import wizard by clicking on the icon highlighted below.
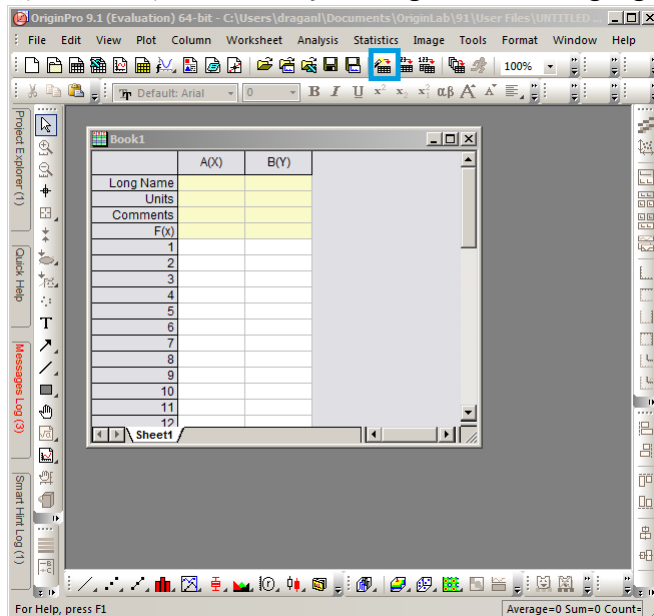
2. Ensure that the ASCII button is selected. Click the "..." button. See screenshot below. The "Import Multiple ASCII" dialog will appear.
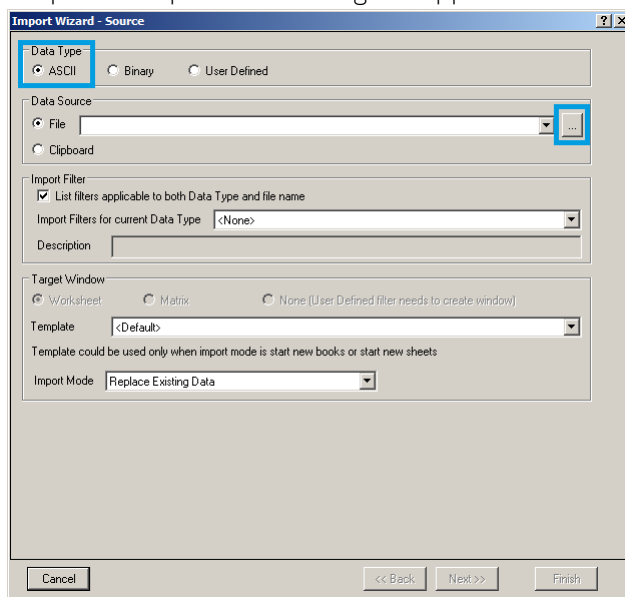
3. Paste the link text from the "LabOne Net Link" dialog into the entry field highlighted below. Then click "Add File(s)" followed by "OK".
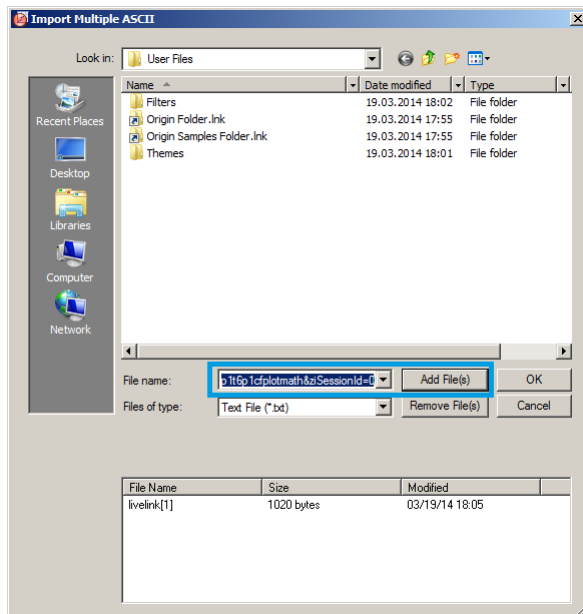
4. Back in the "Import Wizard - Source" dialog click "Finish".
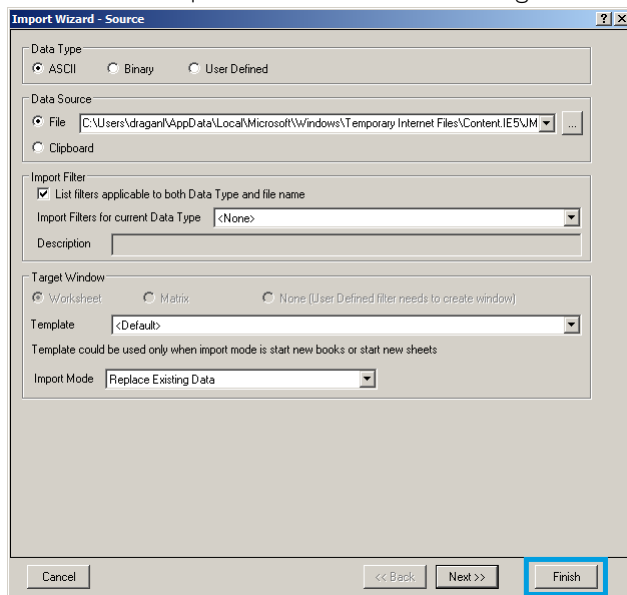
5. The data will appear in the Origin main window.
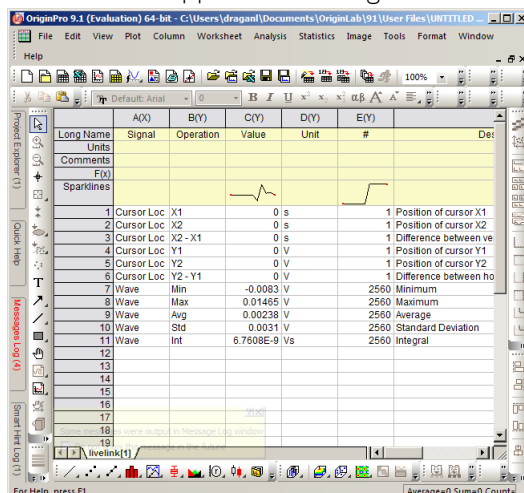
4.3. Memory Architecture and Terminology

4.3.1. Global Architecture

The HDAWG Arbitrary Waveform Generator functionality is realized using field-programmable gate array (FPGA) technology. In order to cope with the amount of digital signal processing to supply 4 or 8 high-speed outputs, the instrument architecture contains two types of FPGAs: 1 back end FPGA handling the central tasks of signal distribution and synchronization, and 2 (or 4) front end FPGAs, each feeding one pair of front panel Wave, Mark, and Trig connectors. This is sketched in Figure 4.10 for the 4-channel model HDAWG4, and an analogous diagram is valid for the HDAWG8.
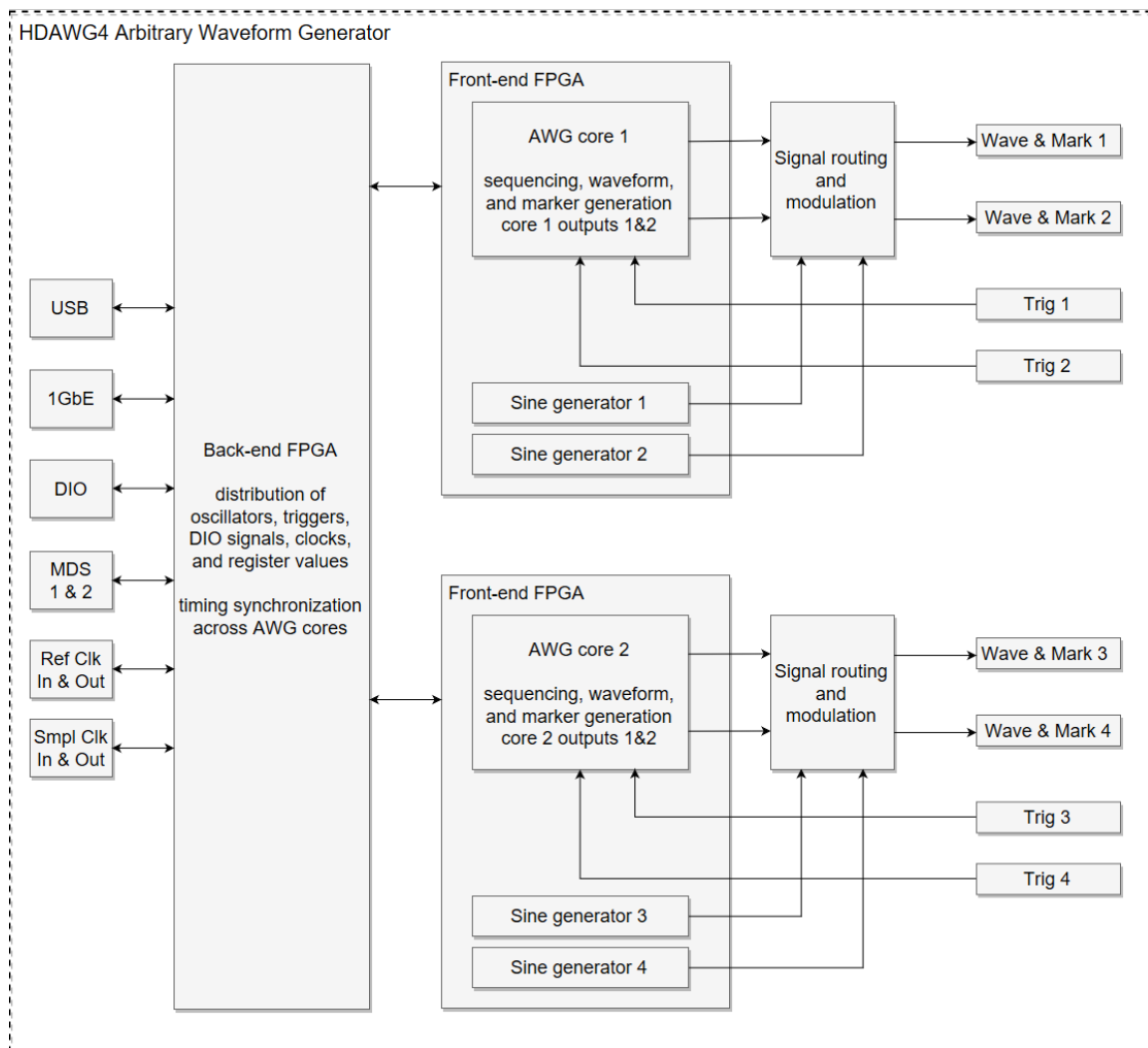
Figure 4.10. HDAWG4 device architecture

On each front end FPGA, there is one so-called AWG core, which is the lowest-level processor sending waveforms from the memory to one pair of Wave and Mark outputs. Additionally, each front end FPGA holds 2 sine generators for digital modulation of this pair of outputs. This aspect of the HDAWG architecture is most relevant in understanding the channel grouping feature as well as triggering.

- Independently of the channel grouping mode of the HDAWG (1x8, 2x4, 4x2), sequence programs are always executed on the AWG cores. This means, e.g., in 1x8 mode, a high-level

sequence program written in the Section 4.14 is getting compiled into 4 low-level sequence programs that are executed in parallel on the 4 AWG cores. The back-end FPGA synchronizes the execution timing.

- Trigger input signals from the front panel arrive first to one of the front-end FPGAs, from where they are distributed to the back-end FPGA and to the other front-end FPGAs. The internal trigger distribution is associated with a delay, which is why the lowest trigger latency is achieved using local triggering within one input/output connector pair in 4x2 mode.

- Sine generator signals are local within one front end FPGA, which is why combinations between AWG channels and sine generators for digital modulation are only possible within one output pair

- Oscillator signals are global with HDAWG-MF option

- The 4 Marker signals from one AWG core (2 per AWG channel) can be routed to the two Mark outputs within one output pair, but not to other Mark outputs.

The Signal Routing and Modulation block enables different methods of digital modulation and is described in more detail in Section 4.4 and Section 4.5.

4.3.2. AWG Core Architecture

The AWG core architecture is sketched in Figure 4.11. The main element of the core is the Sequencer, a real-time processor running at a clock speed of nominally 300 MHz, or 1/8 of the sampling rate. The execution timing of a sequence program is as follows:

- Each instruction of a sequence program executes deterministically in 1 or a few clock cycles.

- The exact number of clock cycles for a given instruction can be inferred from the Advanced sub-tab of the AWG Sequencer tab

- `playWave` instructions mark the start of a waveform playback

- `playWave` instructions are executed after a previous waveform playback is finished

The last point means that sequential waveforms are played immediately after one another, back to back, as long as their length meets the granularity specification, i.e. is equal to 32 samples plus a multiple of 16 samples.
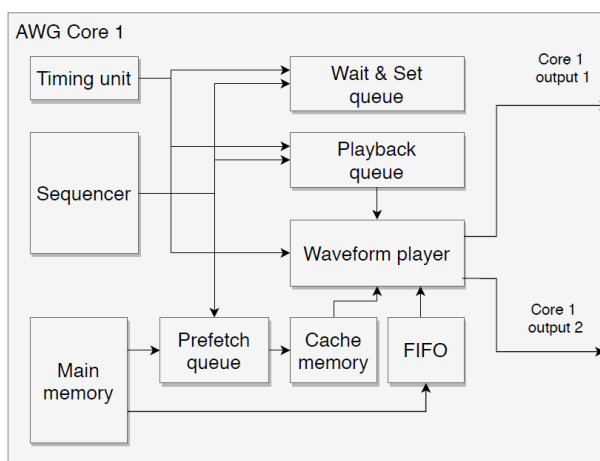
Figure 4.11. AWG core architecture

For a deeper understanding of the execution timing, it's necessary to look at the interplay between the Sequencer and the other elements of the AWG core. The Sequencer distributes most of the instructions of the high-level LabOne AWG Sequencer language into separate queues:

- The Playback Queue holds waveform playback instructions
- The Prefetch Queue holds waveform prefetch instructions that load waveform data from the high-latency main memory into the low-latency, real-time cache memory
- The Wait&Set Queue holds instructions to wait for a trigger, or to set parameters

In this way, the Sequencer is able to "move ahead in time" and distribute multiple waveform prefetch instructions in the Prefetch Queue, allowing enough time to load the corresponding waveform data into the cache memory before they are required by the Playback Queue. The timed execution of instructions across separate queues is managed by the Timing Unit.

There is one class of instructions that however cannot be distributed into queues ahead of time: these are instructions of the "Get" type, such as `getDIO`, that return a value to the Sequencer. Since the sequencer language allows that subsequent instructions are influenced by the returned value (e.g. by using the external input for a conditional branch), the Sequencer must run on the assumption that all previous queues have to be empty before executing the Get instruction, and queues can only be filled again when the Get instruction is completed. This instruction classification timing rules are summarized in the following table.

Table 4.8. Instruction execution

Instruction class	Examples	Executed by...	Executed...
Playback	`playWave`	Playback queue	...when triggered by Timing Unit
Prefetch	`prefetch`	Prefetch queue	...as early as possible
Wait/Set	`setTrigger, wait, waitDigTrigger`	Wait & Set queue	...when triggered by Timing Unit
Get	`getDigTrigger, getUserReg`	Sequencer	...when all queues are empty

The cache memory holds space for 256 kSa of dual-channel waveform data. This memory is divided into 256 blocks of 1024 Sa (dual-channel) length. At any start of a waveform playback, the necessary waveform data needs to be present in the cache memory. For long waveforms that exceed the size of 2 cache memory blocks (2048 Sa dual-channel), only the waveform beginning of that size (2048 Sa dual-channel) needs to be present. The remainder of the waveform data is processed through a FIFO (first-in-first-out) buffer and does not occupy cache memory. This division between cache memory and FIFO buffer allows the system to bridge the access time to the main memory,

As a consequence, there is a limit in total number of long waveforms in a sequence below which a user has the full freedom in specifying a sequence program. Above that limit, the sequence program needs to fulfill certain conditions that are detailed below and that are verified by the AWG Compiler.

The limit on the number of long waveforms depends on the amount of waveform memory (say, M_{short}) occupied by all short waveforms in a sequence. The limit is then given by $(M_{cache}-M_{short})/(2M_{block})$, where M_{cache} = 256kSa is the size of the cache memory, and M_{block} = 1024 Sa is the size of one cache memory block. In case no short waveforms are used, the limit on the number of long waveforms is highest, namely $M_{cache}/(2M_{block})$ = 128.

The limit can be lifted for sequences whose structure allows for idle times to controllably replace waveform beginnings in the cache during runtime. These idle times can arise due to either of the following sequence elements:

1. `playZero(n)` instructions of sufficient duration `n`
2. Playback of very large waveforms with a length beyond $30M_{block}$ = 30,720 Sa

In the first case, the `playZero` instruction creates a well-defined segment of zero-valued signal without the use of waveform memory. In the second case, the FIFO buffer is able to advance sufficiently in data accumulation such to create an idle time during runtime. Both of these sequence elements are identified by the AWG Compiler, who then places appropriate waveform prefetch instructions in the compiled sequence. If the AWG Compiler finds that the waveform limit is exceeded, but there are not sufficient idle times to fetch data, it will generate an error, because the AWG could not guarantee gapless and jitter-free playback of the.intended sequence.

Consider a typical example of a triggered series of waveform playbacks:

```
waitDigTrigger(1);
playWave(long_waveform_001);
waitDigTrigger(1);
playWave(long_waveform_002);
(...)
waitDigTrigger(1);
playWave(long_waveform_200);
```

The total number 200 of long waveforms exceeds the limit 128. Unless some of these waveforms are longer than 30 kSa, the compiler will not detect any opportunity to fetch waveform data in the middle of the sequence, and thus will reject the compilation.

In practice, there is however often enough time between the end of a waveform and the following trigger event. This idle time can be made available for waveform data fetching. This can be achieved by inserting playZero instructions at places where no signal is to be generated. The sequence can be modified e.g. to this:

```
const M_idle = 8000;
waitDigTrigger(1);
playWave(long_waveform_001);
playZero(M_idle);
waitDigTrigger(1);
playWave(long_waveform_002);
playZero(M_idle);
(...)
waitDigTrigger(1);
playWave(long_waveform_200);
playZero(M_idle);
```

The AWG Compiler is thus made aware of the idle times and can use them to fetch waveform data. The sequence will be accepted for compilation, and is played back as intended.

In addition to situations as above, the use of the `playZero(n)` instruction is strongly encouraged in replacement of a playback of a zero-valued waveform, e.g. `playWave(zeros(n))`. Both instructions will result in the same output, but `playZero` consumes no waveform memory. In addition to avoiding the limit of number of waveforms, it allows you to greatly reduce waveform upload time to the instrument. The instruction `playZero(n)` can replace the instruction `wait(m)` with `m=n/8` in most cases. It has the advantage of behaving equivalently to a `playWave` instruction. The `wait` instruction is required to implement a waiting time depending on a run-time variable, or to generate a constant output voltage in combination with the last-sample-hold feature.

4.4. Output Tab

The Output tab provides the electrical configuration of the Wave outputs and all configuration of the AWG and sine generator signals. It is available on all HDAWG instruments.

4.4.1. Features

- Sequencer start/stop control
- Dynamic layout for 1x8, 2x4, 4x2 channel grouping
- Sine generator configuration: frequency, harmonic, phase, amplitude
- AWG modulation control
- Output enable, range, amplification, and filter settings

4.4.2. Description

Table 4.9. App icon and short description

Control/Tool	Option/Range	Description
Output		Quick overview and access to all the settings and properties for signal generation and modulation.

The Output tab (see Figure 4.12) is divided into four sections: Oscillators, Sine Generators, Waveform Generators, and Wave outputs.
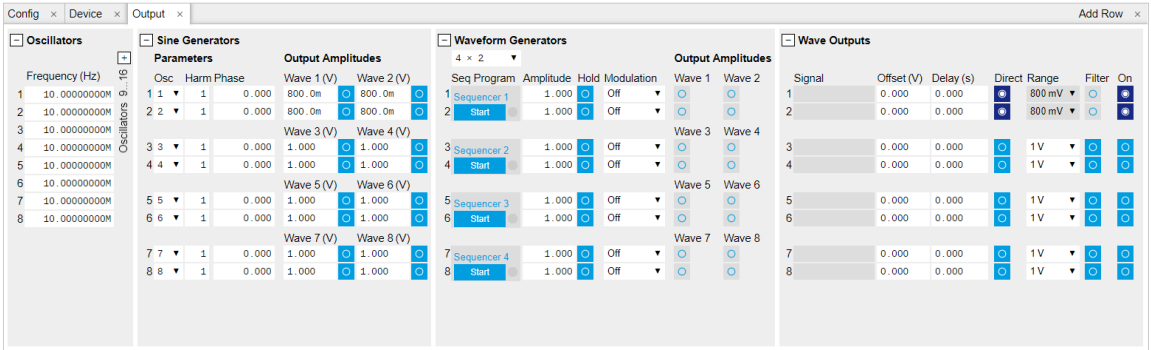
Figure 4.12. LabOne UI: Output tab

The purpose of the Output tab is to configure how the waveform signals previously prepared in the AWG Sequencer tab are routed to the Wave outputs of the instrument. The tabular layout of the tab provides a quick overview of the status of the different AWG cores on the instrument (see Section 4.3 for an overview on the internal architecture and terminology), and enables quick access to all necessary settings to configure the modulation (i.e. multiplication) and addition of sinusoidal and AWG signals.

Conceptually, the tab is laid out as follows: The Wave Outputs section represents the physical outputs of the device in horizontal rows. The signal on a given Wave output can be a multiplication, or addition, or both, of AWG and sine signals, depending which modulation mode is used.. The Signal field contains a representation of the current signal generated on one output in the form of a simple formula (e.g., "Sine_1*AWG_1 + Sine_2").

The individual Sine and AWG signals are configured in the Sine Generators and Waveform Generators section. The horizontal rows in these sections represent the digital signals generated

by the Sine Generators and Arbitrary Waveform Generator channels. The Output Amplitudes section represent the pair-wise matrix-like routing of these signal (rows) to the Wave outputs (columns).

The rows in the Waveform Generator and the Sine Generators section are graphically grouped in two, and each pair is associated with a pair of Wave outputs. This arrangement reflects the internal architecture of the instrument, where one front end FPGA per pair of Wave outputs contains one dual-channel AWG core, and one pair of sine generators.

The diffferent choices in the Modulation setting for each channel correspond to different settings of the Signal Routing and Modulation block (see Section 4.3). The block diagram in Figure 4.13 illustrates this block with its internal switches controlled with the Modulation setting. Table 4.10 and Table 4.11 map the Modulation settings to the states of the switches A through F in the block diagram. Note that the states of the switches B and C are only relevant when a signal from AWG output 1 is explicitly routed to Wave Output 2, or vice versa. This is the case e.g. when using sequencer instructions of the form `playWave(1, 2, w)`,`playWave(1, 2, w_a, 1, 2, w_b)`, `playWave(1, w_a, 1, w_b)`. In the standard form of dual-channel waveform playback using `playWave(w_a, w_b)` on one core, this is not the case and only the switch states A, D, E, and F are relevant. Please refer to Section 3.2.4 for more details on the topic of AWG output assignment.
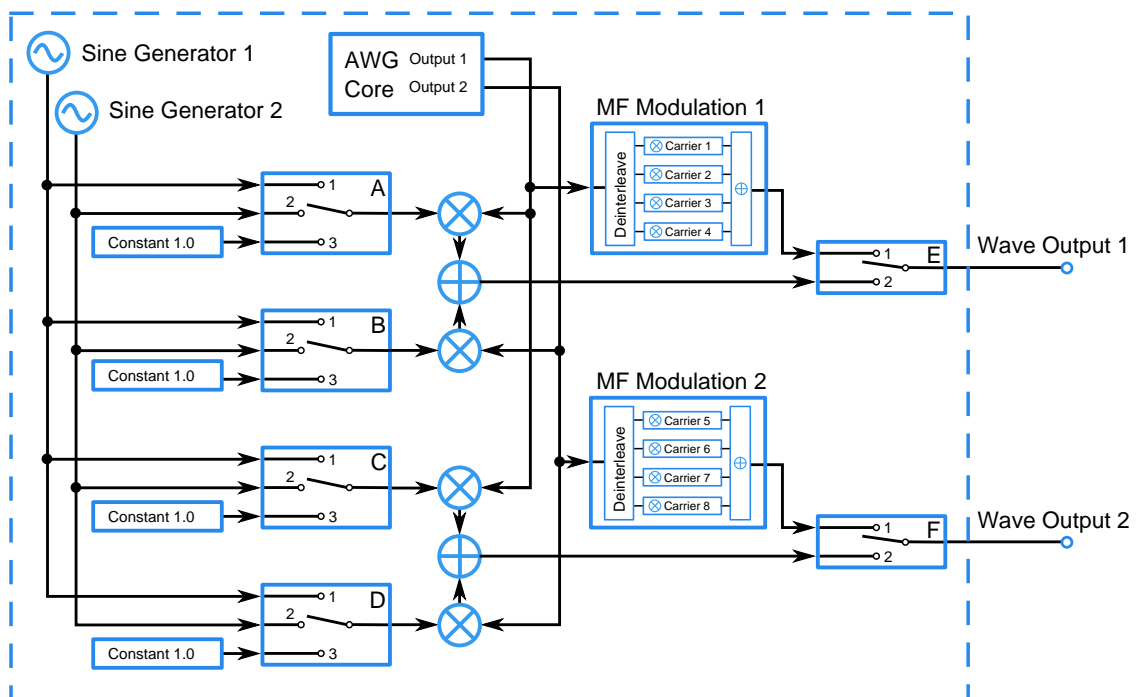
Figure 4.13. HDAWG Signal Routing and Modulation. Table 4.10 and Table 4.11 provide a mapping between Modulation setting and switch states.

Table 4.10. Switch configuration for different Modulation settings on AWG Output 1 (AWG Core 1). Switch C state is only relevant when using non-standard AWG output assignment (see main text).

Modulation setting	Switch A state	Switch C state	Switch E state
Off	3	3	2
Sine 11	1	1	2
Sine 12	1	2	2
Sine 21	2	1	2
Sine 22	2	2	2

Modulation setting	Switch A state	Switch C state	Switch E state
Advanced	3	3	1

Table 4.11. Switch configuration for different Modulation settings on AWG Output 2 (AWG Core 1). Switch B state is only relevant when using non-standard AWG output assignment (see main text).

Modulation setting	Switch B state	Switch D state	Switch F state
Off	3	3	2
Sine 11	1	1	2
Sine 12	1	2	2
Sine 21	2	1	2
Sine 22	2	2	2
Advanced	3	3	1

4.4.3. Functional Elements

Table 4.12. Output tab

Control/Tool	Option/Range	Description
Frequency (Hz)	0 to 1.2 GHz	Frequency control for each oscillator.
Frequency (Hz)		Oscillator frequency when AWG Oscillator Control is enabled. In this mode, the frequencies cannot be changed from the user interface, but only through an AWG sequence program. When disabling AWG Oscillator Control, the frequencies are reset to the previous setting.
Oscillator Selection	1 to 16	Selection of the oscillator used for the generated sine signal.
Harmonics	1 to 1023	Multiplies the oscillator's reference frequency with the integer factor defined by this field.
Phase	-180 to +180 degree	Sets the phase of the sine signal.
Amplitude	0 to 5 V (depending on range)	Sets the amplitude of the sine signal.
Enable	ON / OFF	Enables the given sine signal (row) to the Wave output (column). The signal is added to other signals present on the given Wave output.
Channel Grouping		Sets the channel grouping mode of the device. Can only

Control/Tool	Option/Range	Description
		be changed when all AWG cores are stopped.
	MDS	Use the Wave outputs from MDS synchronized devices.
	4x2 or 2x2	Use the Wave outputs in groups of 2. One sequencer program controls 2 outputs. (use /dev..../awgs/0..4/)
	2x4 or 1x4	Use the Wave outputs in groups of 4. One sequencer program controls 4 outputs. (use /dev..../awgs/0/ and /dev..../awgs/2/)
	1x8	Use the Wave outputs in groups of 8. One sequencer program controls 8 outputs. Requires 8-channel instrument. (use /dev..../awgs/0/)
Oscillator Control		Sets the oscillator control mode of the device.
	UI/API	Oscillators are controlled by the UI/API.
	AWG	Oscillators are controlled by the AWG sequencer.
Start		Runs the AWG Sequencer. In MDS mode, use the start button on the sequencer tab to start all Sequencers in the correct order.
Hold		Keep the last sample (constant) on the output even after the waveform program finishes. It is recommended to use only AWG waveforms with lengths equal to a multiple of 16 together with this functionality. Waveforms with other lengths are automatically padded with zeros at the end by the AWG Compiler.
Modulation		Sets the modulation mode of the given AWG output. The notation Sine NM with N, M = 1, 2, ... signifies the following: the given AWG output is multiplied with the signal of Sine Generator N when routed to the first Wave output of the given AWG core,

Control/Tool	Option/Range	Description
		and with the signal of Sine Generator M when routed to the second Wave output of the given AWG core. In Advanced mode, 4 envelope signals are generated with the given AWG output using sample interleaving. Each envelope is multiplied with one sinusoid signal generated by the MF Modulation block represented in the MF Modulation tab.
Amplitude		Sets the amplitude scaling factor of the given AWG channel. The amplitude is dimensionless scaling factor applied to the digital signal.
Enable		Indicates the routing of the AWG signal (row) to the Wave output (column). Read-only.
Signal	string	Indicates which signals are combined to generate the ouptut wave.
Offset (V)	-1.25 V to + 1.25 V	Sets the analog offset voltage of the Wave output in amplified mode.
Delay		Delay the output of the signal in order to align waves.
Delay Status	red/green	Indicates the status of setting the delay. Red: Output is either turned off, or the delay setting is still in progress. Green: Delay has been set and output is ready.
Direct	ON / OFF	Enables the direct output mode in which the signal from the digital-to-analog converter is connected to the Wave connector without further amplification. This mode provides the highest bandwidth and lowest broadband noise. The range is fixed to 800 mV, and the signal contains spurs at the sampling frequency due to the absence of anti-alias filtering.
Range	200 mV to 5 V	Defines the maximum output voltage that is generated by the corresponding Wave output. This includes the potential multiple AWG and

Control/Tool	Option/Range	Description
		Sine Amplitudes and Offsets summed up. Select the smallest range possible to optimize signal quality.
Filter	ON / OFF	Enables the analog output filter.
On		Main switch for the Wave output corresponding to the LED indicator on the instrument front panel.

4.5. Multi Frequency Modulation Tab

The Multi Frequency Modulation tab provides the overview of the sine generators for advanced modulation. This tab relates to the HDAWG-MF Multi-frequency and is only available if this option is installed on the HDAWG Instrument (see Information section in the Device tab).

4.5.1. Features
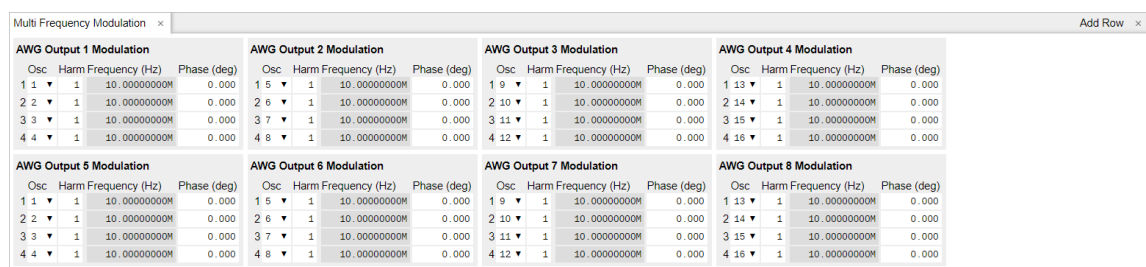
- Independent modulation of multiple sine signals on one Wave output

- 4 sine generators per Wave output

- Carrier configuration: frequency, harmonic, phase, amplitude

4.5.2. Description

Table 4.13. App icon and short description

Control/Tool	Option/Range	Description
MF Mod		Access to all the settings of the sine generators in advanced modulation mode.

The Multi Frequency Modulation tab (see Figure 4.14) consists of 4 or 8 AWG Output Modulation panels corresponding to the AWG channels of the instrument.

Figure 4.14. LabOne UI: Output tab

When setting the Modulation mode of one of the AWG channels to Advanced, the corresponding channel can modulate 4 sine signals that are independent in terms of frequency, phase, and envelope signal. This is realized by de-interleaving the flow of samples from one AWG output into four signals with four-fold reduced sampling rate, multiplying each of them to an individual sine signal, adding up the resulting signals, and sending that sum signal to the DAC. This can be used to realize frequency-domain multiplexing or digital IQ modulation, see Section 3.3.5 for an example. Each panel of the Multi Frequency Modulation tab contains the settings of the four sine generators associated to this AWG output. The block diagram in Figure 4.13 illustrates how the MF Modulation units tie into the output signal routing of the HDAWG.

4.5.3. Functional Elements

Table 4.14. Output tab

Control/Tool	Option/Range	Description
Harm	1 to 1023	Multiplies the oscillator's reference frequency with the

Control/Tool	Option/Range	Description
		integer factor defined by this field.
Phase (deg)	0 to 360 degrees	Sets the phase of the sine sine signal.
Frequency (Hz)	0 to 750 MHz	Displays the frequency of the oscillator linked with this sine generator.

4.6. Precompensation Tab

The Precompensation tab relates to the HDAWG-PC Real-time Precompensation option and is only available if this option is installed on the instrument (see Information section in the Device tab).

4.6.1. Features

- Real-time waveform processing for in-situ tunability
- High-pass compensation
- Overshoot/undershoot compensation with 8 exponential filters
- Bounce compensation for reflections and standing waves
- Programmable FIR filter with convolution length 30 ns
- Precompensation Simulator
- Latency calculation
- Filter reset by AWG sequence instruction

4.6.2. Description

The Precompensation tab provides access to the real-time precompensation filter configuration and the Precompensation Simulator. Whenever the tab is closed, clicking the following icon will open a new instance of the tab.

Table 4.15. App icon and short description

Control/Tool	Option/Range	Description
Precompensation		Configure the Real-time Precompensation filters.

The Precompensation tab shown in Figure 4.15 consists of a configuration section on the left and a display section on the right. The configuration section is further divided into a subsection for device parameters on the left, and for simulation parameters on the right.
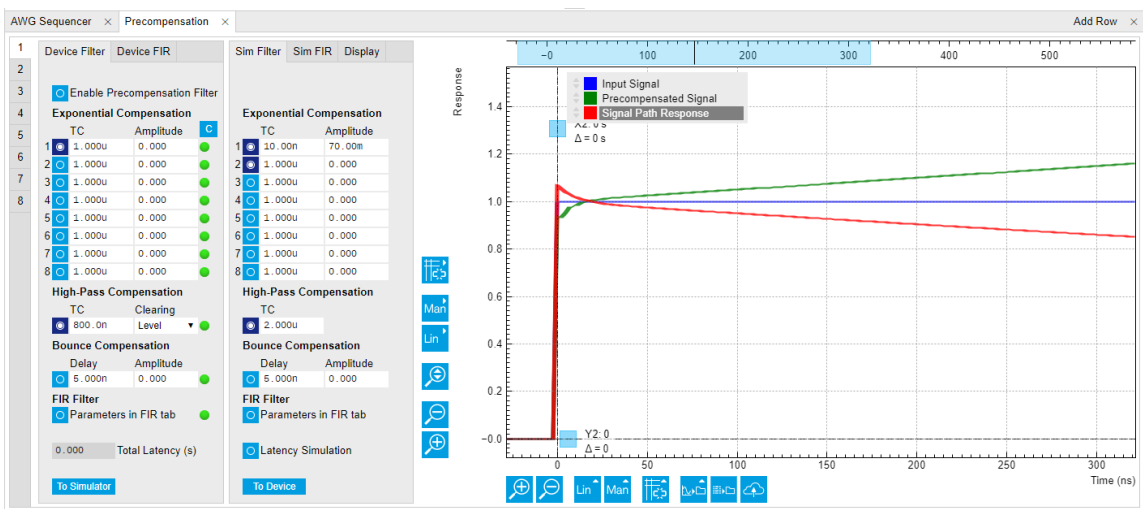
Figure 4.15. LabOne UI: Precompensation tab

There is one precompensation unit for each Wave output of the HDAWG. The individual units are represented as numbered side-tabs. Each unit consists of a chain of filters represented in the Device Filter tab. The filter chain is shown as a flow diagram in Figure 4.16 and consists of 8 exponential compensation filters, 1 high-pass compensation filter, 1 bounce compensation filter, and 1 FIR filter. All configurable filter parameters are represented as rows in the Device Filter tab, with the exception of the FIR filter parameters which are located in the separate Device FIR sub-tab. The entire filter chain can be bypassed or enabled with the Enable Precompensation Filter button, and each filter can be enabled using a button on the left side of the corresponding filter settings. The filter parameters are explained in the following section.

The right side of the configuration section as well as the display section represent the Precompensation Simulator. The Precompensation Simulator is a software-based tool to accurately model the hardware filter and thus allows one to find suitable filter parameters based on a step response measurement of the external circuit. The display section shows three simulated step response curves based on the currently set parameters in the Sim Filter sub-tab:

- **Input signal**: this curve represents an ideal step signal for reference
- **Precompensated signal**: this curve represents the simulated output of the precompensation unit, when the AWG generates a step signal as an input to the precompensation unit, e.g. using a sequencer instruction `playWave(ones(10000))`. The precompensated signal approximately equals the signal expected at the Wave output of the HDAWG with enabled precompensation filter, apart from distortions introduced by the analog output stage of the HDAWG.
- **Signal path response**: this curve represents the step response of the inverse of the precompensation filter. This is the step response of a hypothetical signal path that the currently simulated filter would ideally compensate.

In a typical workflow, the user would start by a measurement of the step response of the full signal path from the AWG to the reference point at the device under test. Often this can be achieved by an averaged oscilloscope measurement at the reference point while generating a step signal with the AWG with disabled precompensation filter, e.g. using the following sequence program:

```
const f_sam = 2.4e9; // sampling rate
const f_seq = fs/8; // sequencer clock rate
const plateau_width = 10e-6;
const wait_time = 10e-3;
const amplitude = 0.5;

while (true) {
  playWave(amplitude*ones(plateau_width*f_sam));
  setPrecompClear(1);
  playWave(zeros(16));
  wait(wait_time*f_seq);
}
```

Consider the following points when using this sequence program:

- The step signal amplitude (here 0.5) should be less than 1.0 in order to provides the digital-to-analog converter with a certain headroom to avoid overflow in case the filter generates positive corrections. This is always the case for the high-pass compensation filter.
- The timescale of the plateau width (here 10 µs) should correspond the timescale of the final signals, and should notably be shorter than a possible high-pass filter time constant in the signal path.
- The waiting time between pulses (here 10 ms) should be much larger than a possible high-pass filter time constant in the signal path, such that there are no history effects.
- The `setPrecompClear(1)` instruction together with the subsequent playback of a zero-valued waveform `playWave(zeros(16))` ensures that the precompensation filters are reset after the measurement, to prevent history effects.

In the next step, the user would adjust the simulator filter parameters such that the displayed signal path response curve matches the measured step response as closely as possible. The measurement graphs shown in the following section may help to visually identify which type of filter applies to a given signal characteristic observed in the step response measurement.

Once satisfied with the result, the simulation parameters can be transferred to the hardware filters by clicking on To Device . The step response measurement described above can then be repeated, this time with enabled precompensation filter. At this stage, fine adjustments of the enabled filters can be done while observing their effect on the signal.

After having eliminated the largest signal distortions, smaller exponential overshoots or undershoots may now become measurable. Iteratively, one can now enable additional exponential filters in order to suppress these.

In case that the GUI-based procedure described here does not yield sufficiently accurate results, consider using a systematic numeric optimization using the one of LabOne APIs in order to exploit the full potential of the HDAWG-PC Real-time Precompensation. Please refer to the documentation of the Precompensation Module in the LabOne Programming Manual. The LabOne Python API comes with an example implementation of a numeric filter optimization.
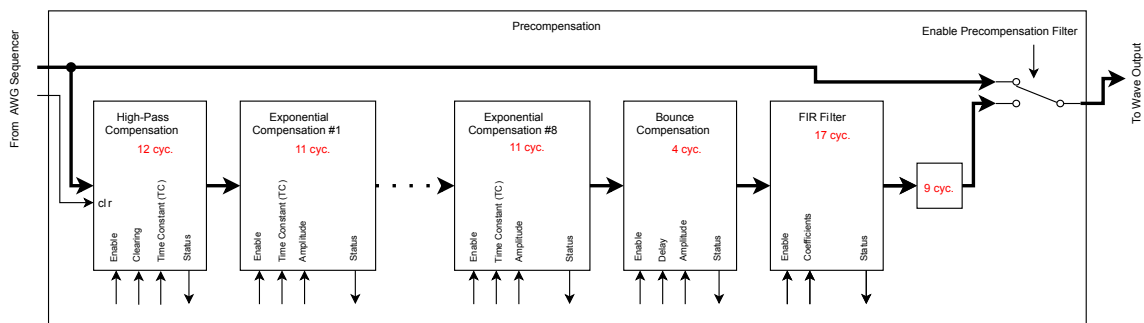
Figure 4.16. Overview of the precompensation filter chain. The thick arrows indicate the signal flow, while the thin arrows indicate parameters and status signals. The latency introduced by each filter, when it is enabled, is indicated in red in units of cycles of the signal processing clock.

Latency and Status

Enabling the full filter chain leads to a signal processing latency of 9 filter clock cycles, where the filter clock runs at 1/8 of the sampling clock. With a sampling clock rate of 2.4 GSa/s, the filter clock frequency is 2400/8 MHz = 300 MHz and the minimum latency is 9/(300 MHz) = 30 ns. Each filter of the chain adds an additional latency when it is enabled, as specified by the red numbers in Figure 4.16. The total latency caused by the current filter configuration is displayed at the bottom of the Device Filter sub-tab.

Each filter has a Status LED on the right side of the settings. A red LED indicates that an arithmetic overflow currently occurs, while a yellow LED indicates that an overflow occurred in the past. The status of all filters can be cleared with the c button. An overflow occurs if the output of the precompensation filter reaches the maximum or minimum of the digital-to-analog converter range. The reason for an overflow can often be diagnosed by inspecting the precompensated signal in the simulator, and can often be eliminated by adjusting the AWG signal. Be aware that an overflow may also hint at the fact that a desired precompensation is not physically realizable. For example, it is not possible to apply a high-pass compensation to a square signal which is already at the maximum output voltage of the instrument, as this would require a signal voltage increasing indefinitely over time beyond the maximum output voltage.

4.6.3. Filter Chain Specification

IIR Filter Theory

The high-pass, exponential, and bounce compensation filters are infinite impulse response (IIR) filters. IIR filters are a special case of linear digital signal processing filters. We specify the operation of the IIR filters by means of the so-called time-domain difference equation:

$$y[n] = b[0]x[n] + b[1]x[n-1] + b[2]x[n-2] + \cdots + b[N]x[n-N]$$

$$- a[1]y[n-1] - a[2]y[n-2] - \cdots - a[M]y[n-M]$$

where the variables x[n] and y[n] represent the digital input and output at discrete time n, respectively. Each coefficient a[k] and b[k] with index k is the weight of a feedback and feedforward stage with k clock cycles delay, respectively. In this definition, the weight a[0] of the most recent output value y[n] is set to one, i.e. a[0] = 1. The operation of the IIR filter is completely determined by the coefficients a and b. The table below lists the definitions of a and b for all precompensation filters.

Name	Applications	Model step response y: **uncorrected** y_c: **corrected**	Parameters	Coefficients in the ideal difference equation
High-pass compensation	AC-coupling, DC-block, Bias-tee	$y(t) = e^{-t/\tau}$	τ: Time constant	$b_0 = \frac{k+1}{k}$, $b_1 = -\frac{k-1}{k}$ $a_0 = 1$, $a_1 = -1$ with $k = 2\,\tau\,f_s$
Exponential under- and overshoot compensation	Distortions due to LRC elements (inductors, resistors, capacitors)	$y(t) = g(1 + A\,e^{-t/\tau})$	τ: Time constant A: Amplitude	$b_0 = 1 - k + k\alpha$ $b_1 = -(1-k)(1-\alpha)$ $a_0 = 1$, $a_1 = -(1-\alpha)$ with $\alpha = 1 - e^{-1/(f_s\tau(1+A))}$ $k = \begin{cases} \dfrac{A}{(1+A)(1-\alpha)} & \text{if } A < 0 \\ \dfrac{A}{1+A-\alpha} & \text{if } A \geq 0 \end{cases}$
Bounce correction	Reflections due to impedance mismatches	$y(t) = x(t) + Ay(t-\tau)$	τ: Delay A: Amplitude	$b_0 = 1$, $b_d = A$ $a_0 = 1$ With $d = \text{round}(\tau\,f_s)$
Finite impulse response (FIR) filter	Short-timescale distortions	$y_c(t) = k(t) * x(t)$ (convolution)	$k(t)$: FIR filter kernel	$b_0, b_1, \ldots, b_{71}$, where $b_i = b_{i+1}$ for $i \geq 8$

The IIR difference equation is available in MATLAB as `filter(b,a,x)`, or in the SciPy Python library as `lfilter(b,a,sig)`. This can be used to predict the outcome of the real-time precompensation listed in the table above. Note, however, that the implementation of real-time digital filters in hardware requires certain simplifications to approximate the ideal filter operation. These simplifications result in slight deviations of the operation of the real-time filters from the ideal operation of the IIR filter. The Precompensation Module of the LabOne APIs contains a method to accurately model the hardware behavior of the filter. Please refer to the LabOne Programming Manual for details.

High-pass Compensation

The purpose of the high-pass compensation is to correct for first-order high-pass filters such as DC blocks or bias-tees. An example of its application is shown in Figure 4.17.The only

parameter of the high-pass compensation filter is the time constant in units of seconds. When enabled, the high-pass compensation filter introduces a delay of 12 clock cycles, which corresponds to 12/(300 MHz) = 40 ns. Since the high-pass compensation filter integrates the signal over time, it may be necessary to clear the filter regularly using the AWG sequencer instruction `setPrecompClear(1)`. Whenever this command is placed in front of a `playWave` or `playWaveDIO` command, a clearing signal is sent to the high-pass compensation filter synchronous to the `playWave` or `playWaveDIO` command. The Clearing drop-down selector defines when the clearing signal is sent relative to the waveform playback: during the entire waveform playback (Level), at the start (Rise), at the end (Fall), or both at the start and at the end (Both).
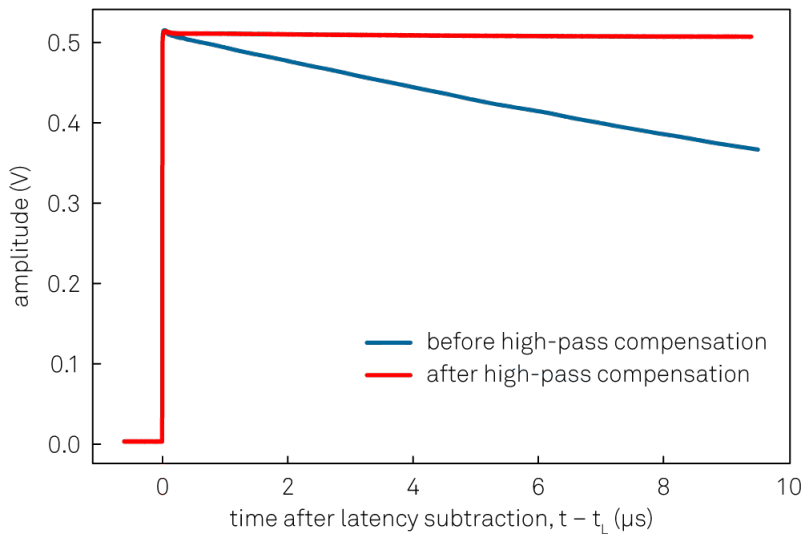
Figure 4.17. Measurement showing a typical application of the high-pass compensation. The signal at the DUT without compensation (blue curve) shows a characteristic decay over time. The high-pass compensation can be used to recover a flat plateau (red curve).

Exponential Compensation

The exponential overshoot and undershoot compensation filters can correct for exponential signal components typically introduced by spurious capacitances or inductances combined with resistors. An example of their application is shown in Figure 4.18. The parameters are the time constant of the exponential decay and the amplitude of the exponential relative to the input step size. A positive amplitude corrects for overshoots, while a negative amplitude corrects for undershoots. When enabled, each exponential compensation filter introduces a delay of 11 clock cycles, corresponding to 11/(300 MHz) = 36.67 ns.
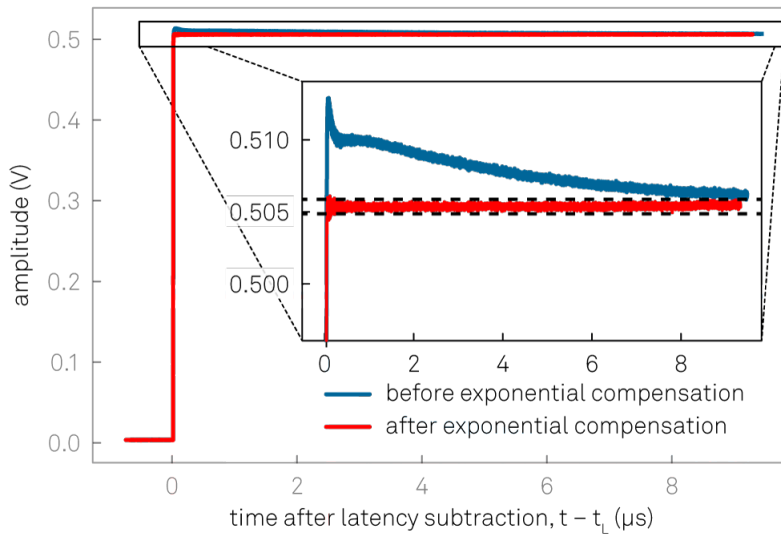
Figure 4.18. Measurement showing a typical application of the exponential compensation. The signal at the DUT without compensation (blue curve) shows characteristic over- and undershoots with different time scales. The exponential compensation can be used to recover a flat plateau (red curve). Three exponential filters were used in this case.

Bounce Compensation

The bounce compensation can be used to suppress unwanted reflections in the cables of the experimental setup. An example of its application is shown in Figure 4.19. The bounce compensation adds to the original input signal the input signal multiplied with a configurable relative amplitude and delay. When enabled, the bounce compensation filter introduces a delay of 4 clock cycles, corresponding to 13.33 ns.
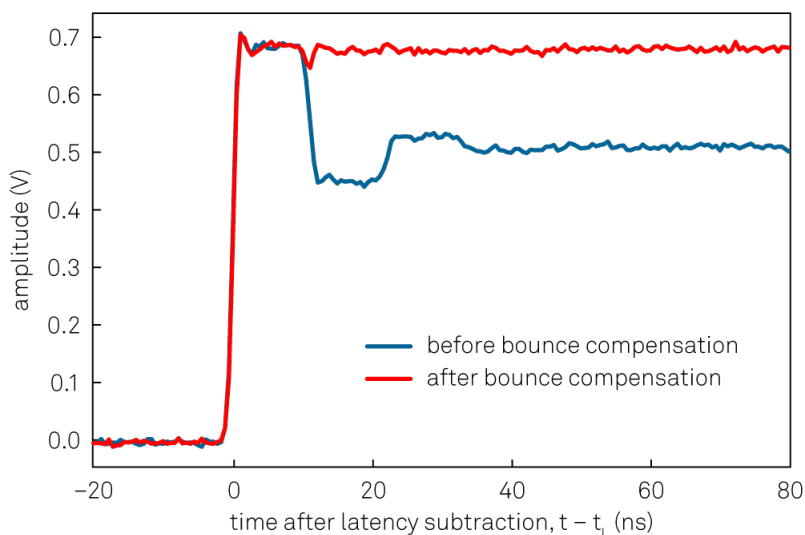
Figure 4.19. Measurement showing a typical application of the bounce compensation. The signal at the DUT without compensation (blue curve) shows characteristic positive and negative steps after multiples of the signal round-trip time, here about 12 ns. The bounce compensation can be used to recover a flat plateau (red curve).

Finite Impulse Response Filter

The real-time finite impulse response (FIR) filter is suitable for correcting signal transients on the nanosecond time scale. An example of its application is shown in Figure 4.20. The user can set 40 coefficients of the FIR filter kernel represented in the Device FIR sub-tab. The operation of the FIR filter corresponds to a convolution of the input signal with the FIR filter kernel. The first 8 coefficients are directly applied to the first eight taps of the FIR filter at the full time resolution. The remaining 32 coefficients are applied to pairs of taps. Therefore, the total number of taps is 8+2·32 and the user can choose 8+32 = 40 coefficients. The Device FIR sub-tab gives access to the coefficients from the graphical UI, which is suitable for quick manual tests. For systematic configuration of this filter, the use of the API is recommended.
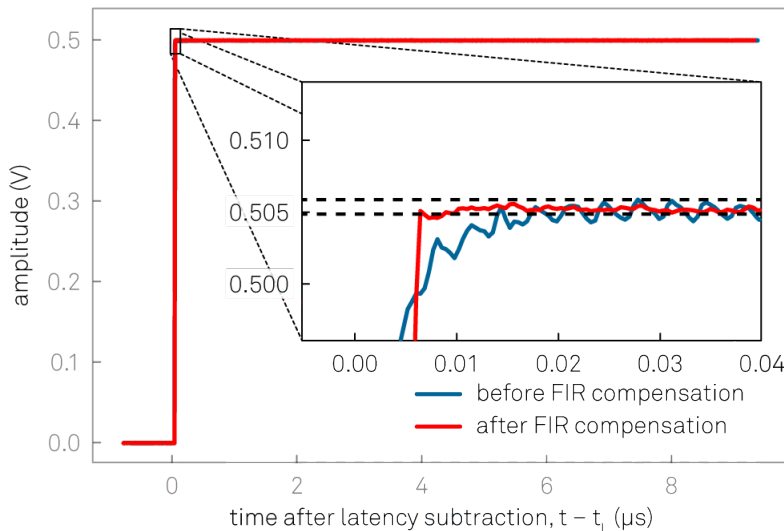
Figure 4.20. Measurement showing a typical application of the FIR compensation. The signal at the DUT without compensation (blue curve) shows a slow settling behavior on the scale of 10 ns due to a low-pass filter as well as a ringing component with a period of about 3 ns. Such signal components may be reduced using the FIR compensation to recover a flat plateau (red curve).

Specification Table

Table 4.16. HDAWG-PC Real-time Precompensation Specifications

Parameter	Value
Number of precompensation units	1 per AWG channel (4 or 8 total depending on HDAWG model)
Compensation filters per unit	1× high-pass, 8× exponential, 1× bounce, 1× FIR
High-pass compensation filter type	IIR, configurable time constant
High-pass compensation time constant range	100 ns to 1 ms
Exponential compensation filter type	IIR, configurable time constant and amplitude
Exponential compensation time constant range	15 ns to 1 ms (available range depends on amplitude setting)
Bounce compensation filter type	FIR, configurable delay and amplitude
Bounce compensation delay range	0 to 100 ns
Bounce compensation delay resolution	1 sample period (417 ps at 2.4 GSa/s)

Parameter	Value
Bounce compensation amplitude range	-1 to +1 (dimensionless)
FIR filter type	Causal FIR filter, 72 coefficients (corresponding to 30 ns at 2.4 GSa/s), first 8 coefficients freely configurable,remaining 64 coefficients pairwise equal
FIR filter coefficients range	-4 to +4 (dimensionless)
FIR filter coefficients resolution	18 bit
Precompensation Simulator display options	Forward filter, inverse filter, uncompensated step response

4.6.4. Functional Elements

Table 4.17. Precompensation tab

Control/Tool	Option/Range	Description
Enable	ON / OFF	Enables the entire precompensation filter chain.
Total Latency		The total latency introduced by the entire precompensation filter chain.
Status Flag Reset		Resets the status flags of all precompensation filters of this output channel.
Enable	ON / OFF	Enables the exponential compensation filter.
Time Constant		Sets the characteristic time constant of the exponential compensation filter.
Amplitude		Sets the amplitude of the exponential compensation filter relative to the signal amplitude.
Status		Indicates the status of the exponential compensation filter: Green: normal, Red: overflow during the last update period (~100 ms), yellow: overflow occurred in the past.
Enable	ON / OFF	Enables the high-pass compensation filter.
Time Constant		Sets the characteristic time constant of the high-pass compensation filter.
Clearing Slope		Select when to react to a clearing pulse generated after the AWG Sequencer setPrecompClear instruction.

Control/Tool	Option/Range	Description
	Level	During the entire clearing pulse.
	Rise	At the beginning of the clearing pulse.
	Fall	At the end of the clearing pulse.
	Both	Both at the beginning and at the end of the clearing pulse.
Status		Indicates the status of the high-pass compensation filter. Green: normal, Red: overflow during the last update period (~100 ms), yellow: overflow occurred in the past.
Enable	ON / OFF	Enables the bounce correction filter.
Delay		Sets the delay of the bounce correction filter.
Amplitude		Sets the amplitude of the bounce correction filter relative to the signal amplitude.
Status		Indicates the status of the bounce correction filter: Green: normal, Red: overflow during the last update period (~100 ms), yellow: overflow occurred in the past.
Enable	ON / OFF	Enables the finite impulse response (FIR) precompensation filter.
FIR Coefficient		FIR (finite impulse response) filter coefficients. The first 8 coefficients are applied to 8 individual samples, whereas the following 32 Coefficients are applied to two consecutive samples each.
Status		Indicates the status of the finite impulse response (FIR) precompensation filter: Green: normal, Red: overflow during the last update period (~100 ms), yellow: overflowed in the past.
Latency Simulation		Enables the simulation of the latency of the precompensated signal.
Number of Points		Length of the simulated wave in number of sample points.

Control/Tool	Option/Range	Description
Input Wave		Wave input source for the simulation.
	Step	Step function.
	Pulse	Single pulse.
	AWG Loading	Load an AWG sequencer wave from the AWG as specified with the AWG wave index.
	File	Load a wave from a CSV file as specified by the file selector.
Open Directory		Opens the directory where the CSV files are expected to be located. This is OS specific and could open additional windows.
Columns		The number of columns determined from the CSV file.
Timestamp Column		The index of the column in the CSV file containing the timestamp for each sample.
Data Column		The index of the column in the CSV file containing the samples.
Sampling Frequency		The sampling frequency determined by the timestamps from the CSV file.
Sample Delay		Artificial time delay of the simulation input.
Gain		Artificial gain with which to scale the samples of the simulation input.
Offset		Artificial vertical offset added to the simulation input.
Status		The status of loading the CSV file.
To Device		Copy the Simulation Filter parameters onto the respective Device Filter parameters.
AWG Wave Index		Determines which AWG wave is loaded from the the AWG output. Internally, all AWG sequencer waves are indexed and stored. With this specifier, the respective AWG wave is loaded into the Simulation.
To Simulator		Copy the Device Filter parameters onto the respective Simulation Filter parameters.

4.7. Plotter Tab

The Plotter is one of the powerful time domain measurement tools as introduced in Section 4.1.2 and is available on all HDAWG instruments.

4.7.1. Features

- Plotting of streamed data from the pulse counters (requires HDAWG-CNT option) or streamed trigger input states.
- Vertical axis grouping for flexible axis scaling
- Histogram and Math functionality for data analysis
- 4 cursors for data analysis

4.7.2. Description

The Plotter serves as graphical display for time domain data in a roll mode, i.e. continuously without triggering. 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 4.18. App icon and short description

Control/Tool	Option/Range	Description
Plotter		Displays various continuously streamed measurement data as traces over time (roll mode).

The Plotter tab (see Figure 4.21) is divided into a display section on the left and a configuration section on the right.
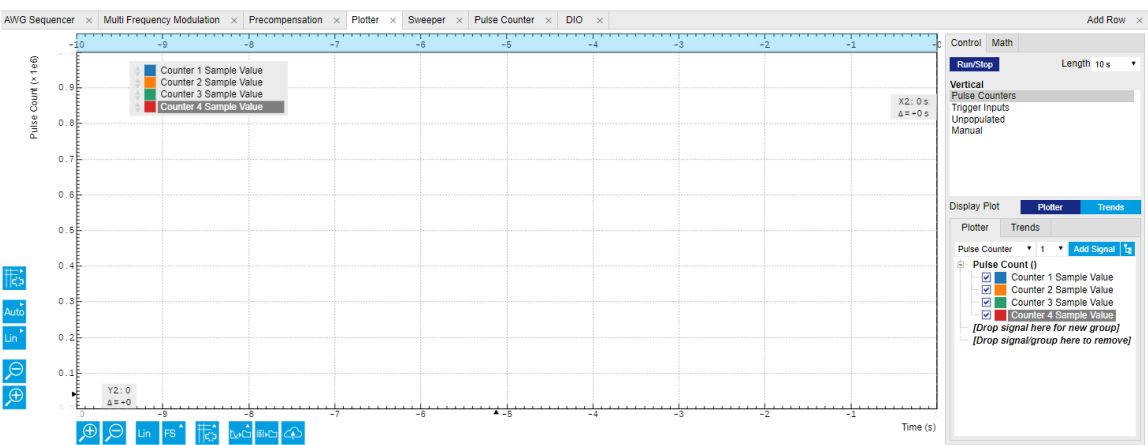
Figure 4.21. LabOne UI: Plotter tab

The Plotter can be used to monitor the evolution of pulse counter and trigger stream data continuously over time. New signals can be added by either using the presets in the Control sub-tab or by going through the tree and selecting the signals of interest in the tree structure. The vertical and horizontal axis can be displayed in Lin, Log or dB scale. The Plotter display can be zoomed in and out with the magnifier symbols, or through Man (Manual), Auto (Automatic) and FS (Full Scale) button settings (see also Section 4.1.3.

The maximum duration data is kept in the memory can be defined as window length parameter in the Settings sub-tab. The window length also determines the file size for the Record Data functionality.

Note

Setting the window length to large values when operating at high sampling rates can lead to memory problems at the computer hosting the data server.

4.7.3. Functional Elements

Table 4.19. Plotter tab: Control sub-tab

Control/Tool	Option/Range	Description
Run/Stop	Run/Stop	Start and stop continuous data plotting (roll mode)
Select a Preset		Select a pre-defined group signals. A signal group is defined by a common unit and signal type. They should have the same scaling behavior as they share a y-axis. Split a group if the signals have different scaling properties.
	Enabled Demods R	Selects the amplitude of all enabled demodulators.
	Enabled Demods Cartesian	Selects X and Y of all enabled demodulators.
	Enabled Demods Polar	Selects amplitude and phase of all enabled demodulators.
	Unpopulated	Shows no signals.
	Manual	Selects the signals as defined in the tree sub-tab.

For the Vertical Axis Groups, please see Table 4.7 in the section called "Vertical Axis Groups".

For the Math sub-tab please see Table 4.6 in the section called "Cursors and Math".

4.8. Data Acquisition Tab

The Data Acquisition tool is one of the powerful time domain measurement tools as introduced in Section 4.1.2 and is available on all HDAWG instruments with the HDAWG-CNT option installed. This tab used to be named Software Trigger tab in previous versions of the LabOne software.

4.8.1. Features

- Time-domain and frequency domain display for all continuously streamed data
- Capture and color scale display of imaging data
- Frame averaging and pixel interpolation
- Automatic trigger level determination
- Display of multiple traces
- Adjustable record history
- Mathematical toolkit for signal analysis

4.8.2. Description

The Data Acquisition tab features display and recording of shot-wise and imaging data sets upon a trigger event. 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 4.20. App icon and short description

Control/Tool	Option/Range	Description
DAQ		Provides complex trigger functionality on all continuously streamed data samples and time domain display.

The Data Acquisition tab (see Figure 4.22) is divided into a display section on the left and a configuration section on the right. The configuration section is further divided into a number of sub-tabs.
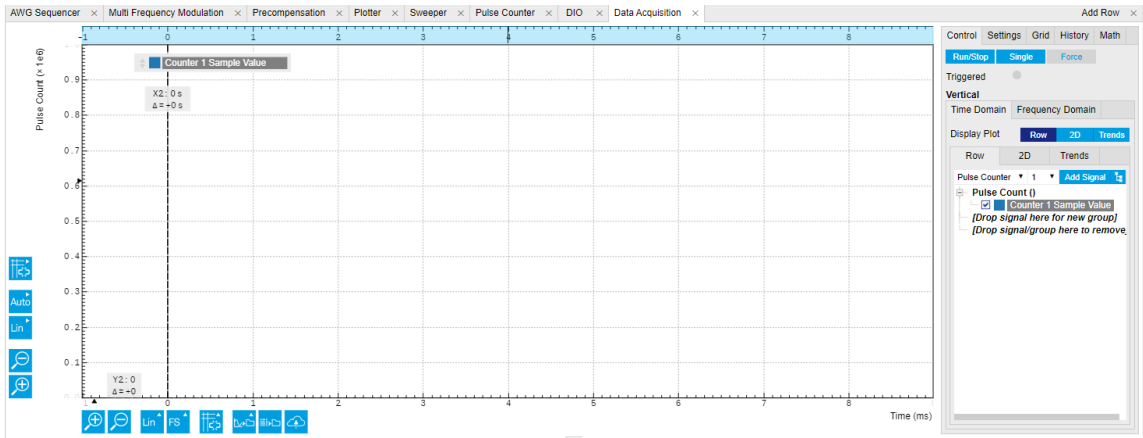
Figure 4.22. LabOne UI: Data Acquisition tab

The Data Acquisition tool brings the trigger functionality of a scope with FFT capability to the streamed data. Examples of streamed data are pulse counter data, or demodulator data of a Zurich Instruments UHF, HF, or MF series instrument connected to the same data server. The user can choose between a variety of different trigger and display options in the time and frequency domain.

Use the **Control** sub-tab to configure which signals are measured, both in time and in frequency domain. Measurement signals can be added to the Vertical Axis Groups section as described in the section called "Vertical Axis Groups". There is one vertical axis group for each the time domain and the frequency domain data.

The trigger condition is configured in the **Settings** sub-tab. Among the selection of Trigger Types provided here, Edge and Pulse are applicable to analog trigger sources from a Zurich Instruments lock-in amplifier such as demodulator data, auxiliary voltages, or oscillator frequencies. The trigger time resolution is enhanced above the sampling rate of the analog data by using interpolation. Instead of manually setting a Trigger Level, you can click on [Find] to have LabOne find a value by analyzing the data stream. In case of noisy trigger sources, both the Bandwidth and the Hysteresis setting can help preventing false trigger events. The Bandwidth setting provides a configurable low-pass filter applied to the trigger source. When enabling this function, be sure to choose a sufficiently high bandwidth to resolve the signal feature that should be triggered upon, i.e., the signal edge or pulse. The Bandwidth setting does not affect the recorded data.

For trigger sources with a slowly varying offset, the Tracking Edge and Tracking Pulse Trigger Types provide continuous adjustment of the Level and Hysteresis. In Tracking mode, the Bandwidth setting plays a different role than for the Edge and Pulse trigger types. Here, the Bandwidth should be chosen sufficiently low to filter out all fast features and only let pass the slow offset.

The Horizontal section of the Settings sub-tab contains the settings for shot Duration and Delay (negative delays correspond to pre-trigger time). Also minimum and maximum pulse width for the Pulse and Tracking Pulse trigger types are defined here.

The **Grid** sub-tab provides imaging functionality to capture and display two-dimensional data sets organized in frames consisting of rows and columns. By default, the number of rows is 1, which means the Data Acquisition tool operates similar to a scope. With a Rows setting larger than 1, every newly captured shot of data is assigned to a row until the number of rows is reached and the frame is complete. After completion of a full frame, the new data either replace the old or averaging is performed, according to the selected Operation and Repetitions setting. On the horizontal axis, the Duration of a shot is divided into a number of samples specified with the Columns setting. The Mode settings provides the functionality for post-processing of the streamed data for interpolation, resampling, and alignment with the trigger event. This is particularly helpful when capturing data from several sources, e.g. demodulators and PID controllers. As illustrated in Figure 4.23, in such situation the streamed data don't lie on the same temporal grid by default. This can be changed by setting Mode to Linear or Nearest. In these modes, the streams from several sources will be upsampled to match the sampling rate and temporal grid of the fastest data stream. This means data processing after saving becomes more convenient, however note that the actual streamed data rate is not increased, and the data don't gain in time resolution. A two-dimensional color scale image of the data can be enabled and controlled in the Display section. The display features configurable scaling, range, and color scale.

With enabled grid mode, the data of a completed frame after averaging appear as a list entry in the **History** sub-tab. See Section 4.2.4 for more details on how data in the history list can be managed and stored.
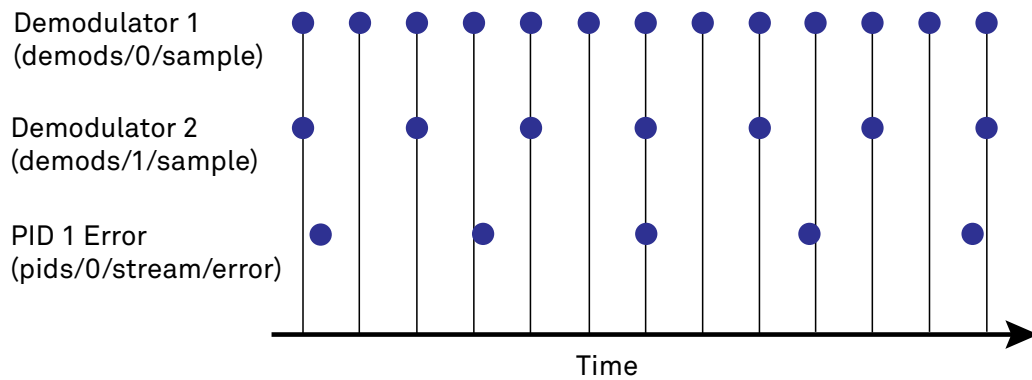
Figure 4.23. Samples from different sources configured with different rates: demodulator 1 at 2N kSa/s, demodulator 2 at N kSa/s and PID Error 1 at M kSa/s (N not divisible by M). Although each stream consists of equidistantly spaced samples in time, the sample timestamps from different streams are not necessarily aligned due to the different sampling rates.

4.8.3. Functional Elements

Table 4.21. DAQ tab: Control sub-tab

Control/Tool	Option/Range	Description
Run/Stop	Run/Stop	Start and stop the Data Acquisition tool
Single	Single	Run the Data Acquisition tool once (record Count trigger events)
Force	Force	Forces a single trigger event.
Triggered	grey/green	When green, indicates that new trigger shots are being captured and displayed in the plot area.

For the Vertical Axis Groups, please see Table 4.7 in the section called "Vertical Axis Groups".

Table 4.22. DAQ tab: Settings sub-tab

Control/Tool	Option/Range	Description
Trigger Signal		Source signal for trigger condition. Navigate through the tree view that appears and click on the required signal.
Trigger Type		Select the type of trigger to use. Selectable options depend on the selected trigger signal.
	Continuous	Continuous triggering.
	Edge	Analog edge triggering based on high and low level. Hysteresis on the levels and low-pass filtering can be

Control/Tool	Option/Range	Description
		used to reduce the risk of wrong trigger for noisy trigger signals.
	Digital	Digital triggering on the 32-bit DIO lines. The bit value defines the trigger condition. The bit mask controls the bits that are used for trigger evaluation. When using a Positive Edge trigger setting, a trigger event occurs as soon as the equality (DIO Value)AND(Bit Mask) = (Bits)AND(Bit Mask) is fulfilled (and was not previously fulfilled). In order to trigger on DIO0 set bit value to 1 and bit mask to 1; to trigger on DIO1 set bit value to 2 and bit mask to 2.
	Pulse	Triggers if a pulse on an analog signal is within the min and max pulse width. Pulses can be defined as either low to high then high to low (positive), the reverse (negative) or both.
	Tracking Edge	Edge triggering with automatic adjustment of trigger levels to compensate for drifts. The tracking speed is controlled by the bandwidth of the low-pass filter. For this filter noise rejection can only be achieved by level hysteresis.
	HW Trigger	Trigger on one of the four trigger inputs. Ensure that the trigger level and the trigger coupling is correctly adjusted. The trigger input state can be monitored on the plotter.
	Tracking Pulse	Pulse triggering with automatic adjustment of trigger levels to compensate for drifts. The tracking speed is controlled by the bandwidth of the low-pass filter. For this filter noise rejection can only be achieved by level hysteresis.
Pulse Type	Positive/Negative/Both	Select between negative, positive or both pulse forms in the signal to trigger on.
Trigger Edge	Positive/Negative/Both	Triggers when the trigger input signal is crossing the trigger

Control/Tool	Option/Range	Description
		level from either high to low, low to high or both. This field is only displayed for trigger type Edge, Tracking Edge and Event Count.
Bits	0 to 2^32-1	Specify the value of the DIO to trigger on. All specified bits have to be set in order to trigger. This field is only displayed for trigger type Digital.
Bit Mask	0 to 2^32-1	Specify a bit mask for the DIO trigger value. The trigger value is bits AND bit mask (bitwise). This field is only displayed for trigger type Digital.
Level	full signal range	Specify the trigger level value.
Find	Find	Automatically find the trigger level based on the current signal.
Hysteresis	full signal range	The hysteresis is important to trigger on the correct edge in the presence of noise. The hysteresis is applied below the trigger level for positive trigger edge selection. It is applied above for negative trigger edge selection, and on both sides for triggering on both edges.
Count	integer number	Number of trigger events to record (in Single mode)
Trigger progress	0% to 100%	The percentage of triggers already acquired (in Single mode)
Bandwidth (Hz)	0 to 0.5 * Sampling Rate	Bandwidth of the low-pass filter applied to the trigger signal. For edge and pulse trigger use a bandwidth larger than the signal sampling rate divided by 20 to keep the phase delay. For tracking filter use a bandwidth smaller than signal sampling frequency divided by 100 to just track slow signal components like drifts.
Enable	ON / OFF	Enable low-pass filtering of the trigger signal.
Hold Off Time (s)	positive numeric value	Hold off time before the trigger is rearmed. A hold off time smaller than the duration will

Control/Tool	Option/Range	Description
		lead to overlapping trigger frames.
Hold Off Count	integer value	Number of skipped triggers until the next trigger is recorded again.
Delay (s)	-Duration to Duration	Time delay of trigger frame position (left side) relative to the trigger edge. For delays smaller than 0, trigger edge inside trigger frame (pre trigger). For delays greater than 0, trigger edge before trigger frame (post trigger)
Refresh Rate	100 mHz to 10 Hz	Set the maximum refresh rate for plot updates. The actual refresh rate depends on other factors such as the hold-off time and duration.
Pulse Min (s)	0 to Duration	Minimum pulse width to trigger on.
Pulse Max (s)	0 to Duration	Maximum pulse width to trigger on.
Window	Rectangular	Seven different FFT windows to choose from. Depending on the application it makes a huge difference which of the provided window function is used. Please check the literature to find out the best trade off for your needs.
	Hann	
	Hamming	
	Blackman Harris	
	Exponential (ring-down)	
	Cosine (ring-down)	
	Cosine squared (ring-down)	
Absolute Frequency	ON / OFF	Shifts x-axis labeling to show the demodulation 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.

Table 4.23. DAQ tab: Grid sub-tab

Control/Tool	Option/Range	Description
Mode		Select resampling method for two-dimensional data recording.
	Off	Two-dimensional data recording is disabled.

Control/Tool	Option/Range	Description
	Nearest	Resampling is performed using substitution by closest data point.
	Linear	Resampling is performed using linear interpolation.
	Exact (on-grid)	Adjust the duration so that the grid distance matches the maximal sampling rate of the selected signals. This allows for on-grid sampling of measurement data. If a signal uses lower sampling rate it will be up-sampled by linear interpolation.
On Grid Sampling	Green or yellow	When green, indicates that all the captured data is aligned to the grid. When yellow, indicates that some data is not aligned to the grid and is interpolated. This can happen when one or more data sources have different sampling rates, or when a sampling rate changes.
Operation		Select row update method.
	Replace	New row replaces old row.
	Average	The data for each row is averaged over a number of repetitions.
	Std	The data for each row is the standard deviation over a number of repetitions.
Columns	numeric value	Number of columns. The data along the horizontal axis are resampled to a number of samples defined by this setting.
Duration	up to 1000 s	Recording length for each triggered data set. In exact sampling mode the duration is a read-only field. The duration is then defined by the maximal sampling rate and column size.
Rows	numeric value	Number of rows
Scan Direction		Select the scan direction and mode
	Forward	Scan direction from left to right

Control/Tool	Option/Range	Description
	Reverse	Scan direction from right to left
	Bidirectional	Alternate scanning in both directions
Repetitions	numeric value	Number of repetitions used for averaging
Row-wise repetition	ON / OFF	Enable row-wise repetition. With row-wise repetition, each row is calculated from successive repetitions before starting the next row. With grid-wise repetition, the entire grid is calculated with each repetition.
Waterfall	ON / OFF	Enable to show the 2D plot in waterfall mode. It will always update the last line.
Overwrite	ON / OFF	Enable to overwrite the grid in continuous mode. History will not be collected. A history element will only be created when the analysis is stopped.
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	Viridis	Select the colormap for the current plot.
	Inferno	
	Balance	
	Turbo	
	Grey	
	Solar	
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.

Control/Tool	Option/Range	Description
	dB	Enable logarithmic mapping in dB.
Scaling	Full Scale/Manual/Auto	Scaling of colorscale.
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 4.24. DAQ 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	Clear	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

Control/Tool	Option/Range	Description
		consecutive saves into the same folder name.
Auto Save		Activate autosaving. When activated, the entire history is saved to a single directory for each completed trace. The autosave directory can be identified by the text "autosave" in the name, e.g. "sweep_autosave_001". If autosave is active during continuous running of the module, the same directory is overwritten after each successive trace. For single shot operation, a new directory is created for each autosave. Note that autosave respects the length of the history, so once the history length has been reached, older measurements will no longer be saved. Please also note that when running the module in continuous mode with large measurements, high data rates and especially with CSV save format, the rate of data acquisition can exceed the rate of saving to disk, especially on machines with limited resources such as the MFLI. In such a situation, the saving will be less frequent than after every measurement in order to prevent resource exhaustion.
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 Table 4.6 in the section called "Cursors and Math".

4.9. Sweeper Tab

The Sweeper is a highly versatile measurement tool available on HDAWG instruments with the HDAWG-CNT Pulse Counter installed. The Sweeper enables scans of an instrument parameter over a defined range and simultaneous measurement of a streamed pulse counter data.

4.9.1. Features

- Full-featured parametric sweep tool for frequency, DC output voltages, AWG user registers
- Display of data from pulse counters
- Different sweep types: single, continuous (run / stop), bidirectional, binary
- Persistent display of previous sweep results
- Normalization of sweeps

4.9.2. Description

The Sweeper supports a variety of experiments where a parameter is changed stepwise and measurement data can be graphically displayed. Open the tool by clicking the corresponding icon in the UI side bar. The Sweeper tab (see Figure 4.24) is divided into 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.

Table 4.25. App icon and short description

Control/Tool	Option/Range	Description
Sweeper		Sweep frequencies, voltages, and other quantities over a defined range and display various response functions including statistical operations.
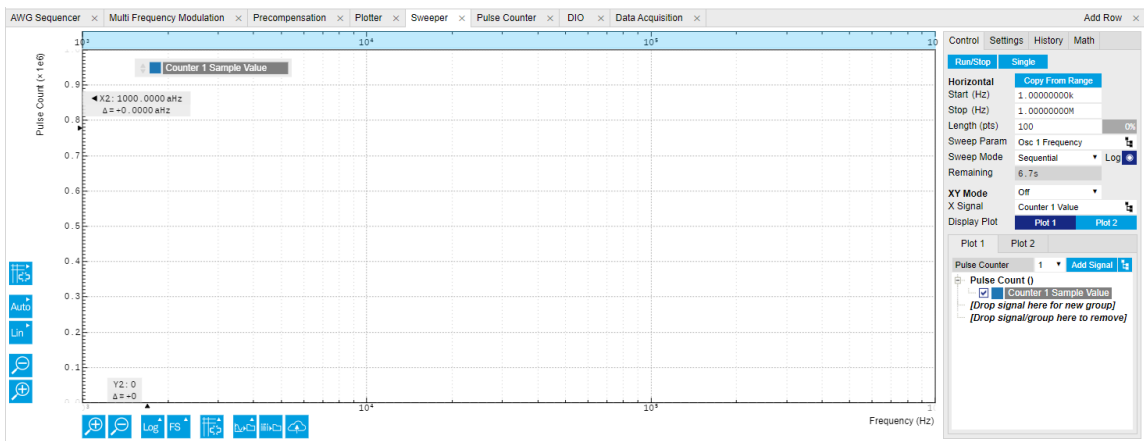
Figure 4.24. LabOne UI: Sweeper tab

The **Control** sub-tab holds the basic measurement settings such as Sweep Parameter, Start/Stop values, and number of points (Length) in the Horizontal section. Measurement signals can be added in the Vertical Axis Groups section. A typical use of the Sweeper is to perform a frequency

sweep and measure the response of the device under test. The Sweeper can also be used to sweep parameters other than frequency, for instance DC offset voltages or AWG user registers. AWG user registers may e.g. be used to sweep a pulse delay, or to iterate through waveforms.

For frequency sweeps, the sweep points are distributed logarithmically, rather than linearly, between the start and stop values by default. This feature is particularly useful for sweeps over several decades and can be disabled with the Log checkbox. The Sweep Mode setting is useful for identifying measurement problems caused by hysteretic sample behavior or too fast sweeping speed. Such problems would cause non-overlapping curves in a bidirectional sweep.

By default the plot area keeps the memory and display of the last 100 sweeps represented in a list in the **History** sub-tab. See Section 4.2.4 for more details on how data in the history list can be managed and stored. With the Reference feature, it is possible to divide all measurements in the history by a reference measurement. This is useful for instance to eliminate spurious effects in a frequency response sweep. To define a certain measurement as the reference, mark it in the list and click on Reference. Then enable the Reference mode with the checkbox below the list to update the plot display. Note that the Reference setting does not affect data saving: saved files always contain raw data.

4.9.3. Functional Elements

Table 4.26. Sweeper tab: Control sub-tab

Control/Tool	Option/Range	Description
Run/Stop	Run/Stop	Runs the sweeper continuously.
Single	Single	Runs the sweeper once.
Copy From Range	Copy From Range	Takes over start and stop value from the plot area.
Start (unit)	numeric value	Start value of the sweep parameter. The unit adapts according to the selected sweep parameter.
Stop (unit)	numeric value	Stop value of the sweep parameter. The unit adapts according to the selected sweep parameter.
Length	integer value	Sets the number of measurement points.
Progress	0 to 100%	Reports the sweep progress as ratio of points recorded.
Sweep Param		Selects the parameter to be swept. Navigate through the tree view that appears and click on the required parameter. The available selection depends on the configuration of the device.
Sweep Mode		Select the scanning type, default is sequential (incremental scanning from start to stop value)

Control/Tool	Option/Range	Description
	Sequential	Sequential sweep from Start to Stop value
	Binary	Non-sequential sweep continues increase of resolution over entire range
	Bidirectional	Sequential sweep from Start to Stop value and back to Start again
	Reverse	Reverse sweep from Stop to Start value
Log	ON / OFF	Selects between linear and logarithmic distribution of the sweep parameter.
Remaining	numeric value	Reporting of the remaining time of the current sweep. A valid number is only displayed once the sweeper has been started. An undefined sweep time is indicated as NaN.
Dual Plot	ON / OFF	Toggle between single plot view and dual plot view
XY Mode		If enabled selects the xy-plot drawing mode of sweep results. For Nyquist plots the y-axis direction is inverted.
	OFF	The xy-plot mode is off. The x-axis is defined by the grid points of the sweep.
	On	The xy-plot is enabled. The y-axis is defined by the selected signal.
	On - Invert	The xy-plot is enabled with inverted y-axis. This mode is used for Nyquist plots that allow for displaying -imag(z) on the y-axis and real(z) on the x-axis.
X Signal		Selects the signal that defines the x-axis for xy-plots. The available selection depends on the configuration of the device. Displaying the selected signal source will result in a diagonal straight line.

For the Vertical Axis Groups, please see Table 4.7 in the section called "Vertical Axis Groups".

Table 4.27. Sweeper tab: Settings sub-tab

Control/Tool	Option/Range	Description
Filter		Application Mode: preset configuration. Advanced Mode: manual configuration.
	Application Mode	The sweeper sets the filters and other parameters automatically.
	Advanced Mode	The sweeper uses manually configured parameters.
Application		Select the sweep application mode
	Parameter Sweep	Only one data sample is acquired per sweeper point.
	Parameter Sweep Averaged	Multiple data samples are acquired per sweeper point of which the average value is displayed.
	Noise Amplitude Sweep	Multiple data samples are acquired per sweeper point of which the standard deviation is displayed (e.g. to determine input noise).
	Freq Response Analyzer	Narrow band frequency response analysis. Averaging is enabled.
	3-Omega Sweep	Optimized parameters for 3-omega application. Averaging is enabled.
	FRA (Sinc Filter)	The sinc filter helps to speed up measurements for frequencies below 50 Hz in FRA mode. For higher frequencies it is automatically disabled. Averaging is off.
	Impedance	This application mode uses narrow bandwidth filter settings to achieve high calibration accuracy.
Precision		Choose between a high speed scan speed or high precision and accuracy.
	Low -> fast sweep	Medium accuracy/precision is optimized for sweep speed.
	High -> standard speed	Medium accuracy/precision takes more measurement time.
	Very high -> slow sweep	High accuracy/precision takes more measurement time.

Control/Tool	Option/Range	Description
Bandwidth Mode		Automatically is recommended in particular for logarithmic sweeps and assures the whole spectrum is covered.
	Auto	All bandwidth settings of the chosen demodulators are automatically adjusted. For logarithmic sweeps the measurement bandwidth is adjusted throughout the measurement.
	Fixed	Define a certain bandwidth which is taken for all chosen demodulators for the course of the measurement.
	Manual	The sweeper module leaves the demodulator bandwidth settings entirely untouched.
Time Constant/Bandwidth Select		Defines the display unit of the low-pass filter to use for the sweep in fixed bandwidth mode: time constant (TC), noise equivalent power bandwidth (NEP), 3 dB bandwidth (3 dB).
	TC	Defines the low-pass filter characteristic using time constant of the filter.
	Bandwidth NEP	Defines the low-pass filter characteristic using the noise equivalent power bandwidth of the filter.
	Bandwidth 3 dB	Defines the low-pass filter characteristic using the cut-off frequency of the filter.
Time Constant/Bandwidth	numeric value	Defines the measurement bandwidth for Fixed bandwidth sweep mode, and corresponds to either noise equivalent power bandwidth (NEP), time constant (TC) or 3 dB bandwidth (3 dB) depending on selection.
Order	numeric value	Selects the filter roll off to use for the sweep in fixed bandwidth mode. Range between 6 dB/oct and 48 dB/oct.
Max Bandwidth (Hz)	numeric value	Maximal bandwidth used in auto bandwidth mode. The

Control/Tool	Option/Range	Description
		effective bandwidth will be calculated based on this max value, the frequency step size, and the omega suppression.
BW Overlap	ON / OFF	If enabled the bandwidth of a sweep point may overlap with the frequency of neighboring sweep points. The effective bandwidth is only limited by the maximal bandwidth setting and omega suppression. As a result, the bandwidth is independent of the number of sweep points. For frequency response analysis bandwidth overlap should be enabled to achieve maximal sweep speed.
Omega Suppression (dB)	numeric value	Suppression of the omega and 2-omega components. Large suppression will have a significant impact on sweep time especially for low filter orders.
Min Settling Time (s)	numeric value	Minimum wait time in seconds between a sweep parameter change and the recording of the next sweep point. This parameter can be used to define the required settling time of the experimental setup. The effective wait time is the maximum of this value and the demodulator filter settling time determined from the Inaccuracy value specified.
Inaccuracy	numeric value	Demodulator filter settling inaccuracy defining the wait time between a sweep parameter change and recording of the next sweep point. The settling time is calculated as the time required to attain the specified remaining proportion [1e-13, 0.1] of an incoming step function. Typical inaccuracy values: 10 m for highest sweep speed for large signals, 100 u for precise amplitude measurements, 100 n for precise noise measurements. Depending on the order the settling accuracy will define

Control/Tool	Option/Range	Description
		the number of filter time constants the sweeper has to wait. The maximum between this value and the settling time is taken as wait time until the next sweep point is recorded.
Settling Time (TC)	numeric value	Calculated wait time expressed in time constants defined by the specified filter settling inaccuracy.
Algorithm		Selects the measurement method.
	Averaging	Calculates the average on each data set.
	Standard Deviation	Calculates the standard deviation on each data set.
	Average Power	Calculates the electric power based on a 50 Ω input impedance.
Count (Sa)	integer number	Sets the number of data samples per sweeper parameter point that is considered in the measurement. The maximum between samples, time and number of time constants is taken as effective calculation time.
Count (s)	numeric value	Sets the time during which data samples are processed. The maximum between samples, time and number of time constants is taken as effective calculation time.
Count (TC)	0/5/15/50/100 TC	Sets the effective measurement time per sweeper parameter point that is considered in the measurement. The maximum between samples, time and number of time constants is taken as effective calculation time.
Phase Unwrap	ON / OFF	Allows for unwrapping of slowly changing phase evolutions around the +/-180 degree boundary.
Spectral Density	ON / OFF	Selects whether the result of the measurement is normalized versus the demodulation bandwidth.

Control/Tool	Option/Range	Description
Sinc Filter	ON / OFF	Enables sinc filter if sweep frequency is below 50 Hz. Will improve the sweep speed at low frequencies as omega components do not need to be suppressed by the normal low-pass filter.

Table 4.28. Sweeper 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	Clear	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, the entire history is saved to a single directory for each completed trace. The autosave directory can be identified by the text "autosave" in the name, e.g. "sweep_autosave_001". If autosave is active during

Control/Tool	Option/Range	Description
		continuous running of the module, the same directory is overwritten after each successive trace. For single shot operation, a new directory is created for each autosave. Note that autosave respects the length of the history, so once the history length has been reached, older measurements will no longer be saved. Please also note that when running the module in continuous mode with large measurements, high data rates and especially with CSV save format, the rate of data acquisition can exceed the rate of saving to disk, especially on machines with limited resources such as the MFLI. In such a situation, the saving will be less frequent than after every measurement in order to prevent resource exhaustion.
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.
Reference	**Reference**	Use the selected trace as reference for all active traces.
Reference On	ON / OFF	Enable/disable the reference mode.
Reference name	name	Name of the reference trace used.

For the Math sub-tab please see Table 4.6 in the section called "Cursors and Math".

4.10. DIO Tab

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

4.10.1. Features

- Monitor and control of digital I/O connectors
- Control settings for marker outputs and trigger inputs

4.10.2. Description

The DIO tab is the main panel to control the digital inputs and outputs as well as the trigger levels. 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 4.29. App icon and short description

Control/Tool	Option/Range	Description
DIO		Gives access to all controls relevant for the digital inputs and outputs including the Ref/Trigger connectors.
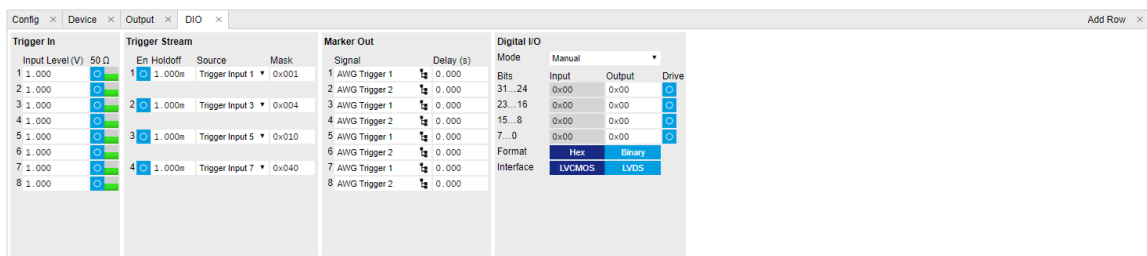
Figure 4.25. LabOne UI: DIO tab

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 Trigger In section shows the settings for the 4 or 8 Trig inputs on the front panel. The LED status indicator helps in monitoring the input signal state and selecting the threshold.. The Trigger Stream section allows the user to control streaming of trigger signals to the computer and monitor them in the Plotter tool over time. The Marker Out section allows assigning internal marker bits to the 4 or 8 Mark outputs on the front panel. Alternatively, the outputs can be set to static high or low values.

4.10.3. Functional Elements

Table 4.30. Digital input and output channels, reference and trigger

Control/Tool	Option/Range	Description
DIO mode		Select DIO mode
	Manual	Manual setting of the DIO output value.

Control/Tool	Option/Range	Description
	AWG Sequencer	Enables setting the DIO output values by AWG sequencer commands and forwards DIO input values to the AWG sequencer. The DIO interface operates at a clock frequency of 150 MHz.
	DIO Codeword	Enables setting the DIO output values by AWG sequencer commands and forwards DIO input values to the AWG sequencer. This mode is equivalent to the mode AWG Sequencer, except for the DIO interface clock frequency which is set to 50 MHz.
	QCCS	Enables setting the DIO output values by the ZSync input values. Forwards the ZSync input values to the AWG sequencer. Forwards the DIO input values to the ZSync output. Select this mode when the instrument is connected via ZSync to a PQSC.
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.
	hex	DIO view format is hexadecimal.
	binary	DIO view format is binary.
Clock		Select DIO internal or external clocking.
Interface		Selects the interface standard to use on the 32-bit DIO interface. This setting is persistent across device reboots.
	LVCMOS	A single-ended, 3.3V CMOS interface is used.

Control/Tool	Option/Range	Description
	LVDS	A differential, LVDS compatible interface is used.
Enable	ON / OFF	Enables trigger streaming.
Holdoff Time	time in seconds	Sets the holdoff time of the trigger unit.
Trigger Source		Selects a trigger source for the stream. The mask is bit encoded where bit 0..7 are the input triggers and bit 8..11 are AWG triggers. The "Mask" setting allows for an arbitrary combination of trigger sources using the bit encoding in the "Mask" field.
Mask Triggers		Masks triggers for the current stream. Only enabld if "Mask" is selected as trigger stream source. The mask is bit encoded where bit 0..7 are the input triggers and bit 8..11 are AWG triggers.
Trigger level	–10 V to 10 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.
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.
Marker output signal		Select the signal assigned to the marker output.
	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

Control/Tool	Option/Range	Description
		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.
	Output 1 Marker 1	Output is assigned to Output 1 Marker 1.
	Output 1 Marker 2	Output is assigned to Output 1 Marker 2.
	Output 2 Marker 1	Output is assigned to Output 2 Marker 1.
	Output 2 Marker 2	Output is assigned to Output 2 Marker 2.
	Trigger Input 1	Output is assigned to Trigger Input 1.
	Trigger Input 2	Output is assigned to Trigger Input 2.
	Trigger Input 3	Output is assigned to Trigger Input 3.
	Trigger Input 4	Output is assigned to Trigger Input 4.
	Trigger Input 5	Output is assigned to Trigger Input 5.
	Trigger Input 6	Output is assigned to Trigger Input 6.
	Trigger Input 7	Output is assigned to Trigger Input 7.
	Trigger Input 8	Output is assigned to Trigger Input 8.
	High	Output is set to high.
	Low	Output is set to low.
Delay	Approximately -15 ns to 30 ns (depends on sampling clock frequency)	Controls the fine delay of the Marker output, range approximately -15 ns to 30 ns (depends on sampling clock frequency)

4.11. Config Tab

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

4.11.1. 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

4.11.2. 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 4.31. App icon and short description

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

The Config tab (see Figure 4.26) is divided into four sections to control connections, sessions, user interface appearance and data recording.
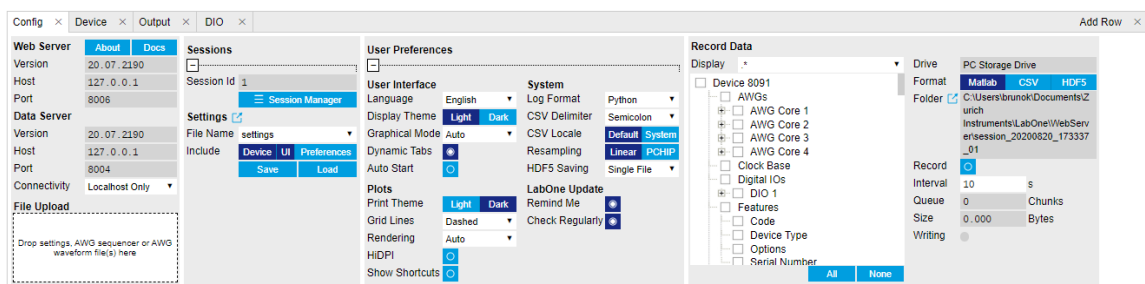
Figure 4.26. LabOne UI: Config tab

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 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 dialogue.

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

account. For low ambient light conditions the use of the dark display theme is recommended (see Figure 4.27).
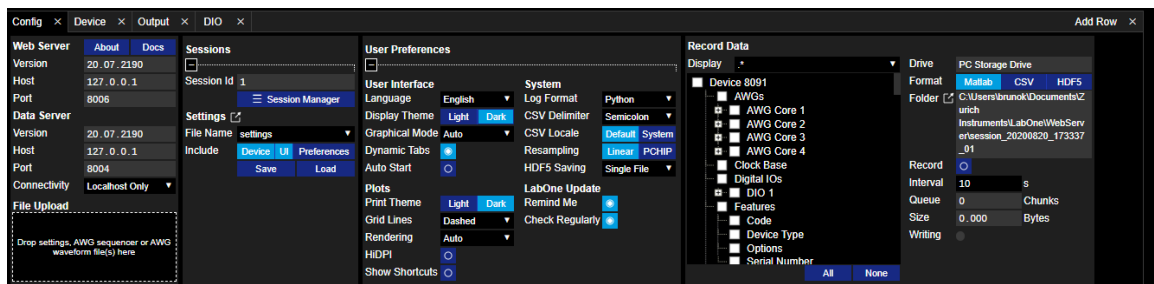
Figure 4.27. LabOne UI: Config tab - dark theme

4.11.3. Functional Elements

Table 4.32. 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.
Connectivity	Localhost Only	Forbid/Allow to connect to this Data Server from other computers.
	From Everywhere	
File Upload	drop area	Drag and drop files in this box to upload files. Clicking on the

Control/Tool	Option/Range	Description
		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	Light	Choose theme of the user interface.
	Dark	
Plot Print Theme	Light	Choose theme for printing SVG plots.
	Dark	
Plot Grid	Dashed	Select active grid setting for all SVG plots.
	Solid	
	None	

Control/Tool	Option/Range	Description
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.

Control/Tool	Option/Range	Description
Dynamic Tabs	ON / OFF	If enabled, sections inside the application tabs are collapsed automatically depending on the window width.
Graphical Mode	Auto	Select the display mode for the graphical elements. Auto format will select the format which fits best the current window width.
	Expanded	
	Collapsed	
Log Format	Telnet	Choose the command log format. See status bar and [User]\Documents \Zurich Instruments\LabOne \WebServer\Log
	Matlab	
	Python	
	.NET	
CSV Delimiter	Comma	Select which delimiter to insert for CSV files.
	Semicolon	
	Tab	
CSV Locale	Default locale. Dot for the decimal point and no digit grouping, e.g. 1005.07	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.
	System locale. Use the symbols set in the language and region settings of the computer	
HDF5 Saving	Single file. All measurements go in one 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.
	Multiple files. Each measurement goes in a separate 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.

Control/Tool	Option/Range	Description
Format	Matlab	File format of recorded and saved data.
	CSV	
	SXM (Nanonis)	
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 the section called "Tree Selector".

4.12. Device Tab

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

4.12.1. Features

- Option and upgrade management
- External clock referencing (10/100 MHz)
- Instrument connectivity parameters
- Device monitor

4.12.2. 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 4.33. App icon and short description

Control/Tool	Option/Range	Description
Device		Provides instrument specific settings.
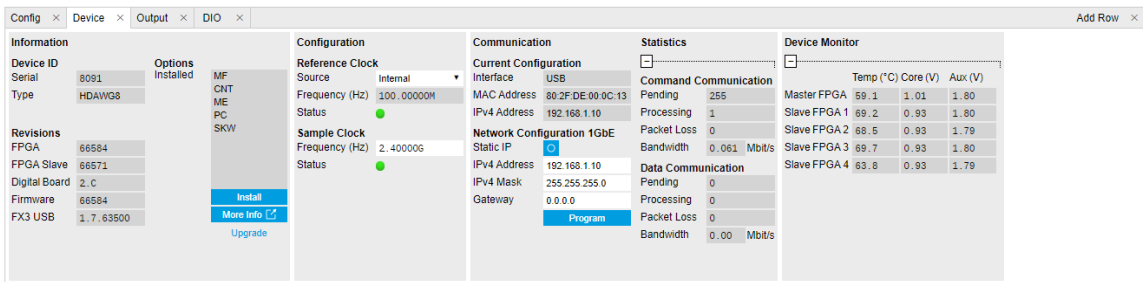
Figure 4.28. LabOne UI: Device tab

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 10/100 MHz reference. The reference is to be connected to the Clock Input on the instrument back panel. The section also allows one to select the sample clock frequency

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

The **Statistics** section gives an overview on communication statistics.

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.

4.12.3. Functional Elements

Table 4.34. 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.
Slave FPGA	integer number	HDL firmware revision of the slave FPGA.
Digital Board	version number	Hardware revision of the FPGA base board.
Firmware	integer number	Revision of the device internal controller software.
FX3 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.
Reference Clock Source		Selects the reference clock source.
	Internal	The internal 100MHz clock is used as the frequency and time base reference.
	External	An external clock is intended to be used as the frequency and time base reference. Provide a clean and stable 10MHz or 100MHz reference to the appropriate back panel connector.
	ZSync	A ZSync clock is intended to be used as the frequency and time base reference.
Reference Clock Frequency		Indicates the frequency of the reference clock.
Reference Clock Status		Indicates the status of the reference clock. Green: locked. Yellow: the device is busy trying to lock onto the

Control/Tool	Option/Range	Description
		reference clock signal. Red: there was an error locking onto the reference clock signal. After an error the source is automatically switched back to internal reference clock.
Sample Clock Frequency		Indicates the frequency of the sample clock.
Sample Clock Status		Indicates the status of the sample clock. Green: locked on. Yellow: the device is busy trying to adjust the sample clock. Red: there was an error adjusting the sample clock.
Interface		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.
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.

Control/Tool	Option/Range	Description
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.

4.13. File Manager Tab

4.13.1. Features

- File preview for settings files and log files

4.13.2. Description

Table 4.35. 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 Figure 4.29) 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.
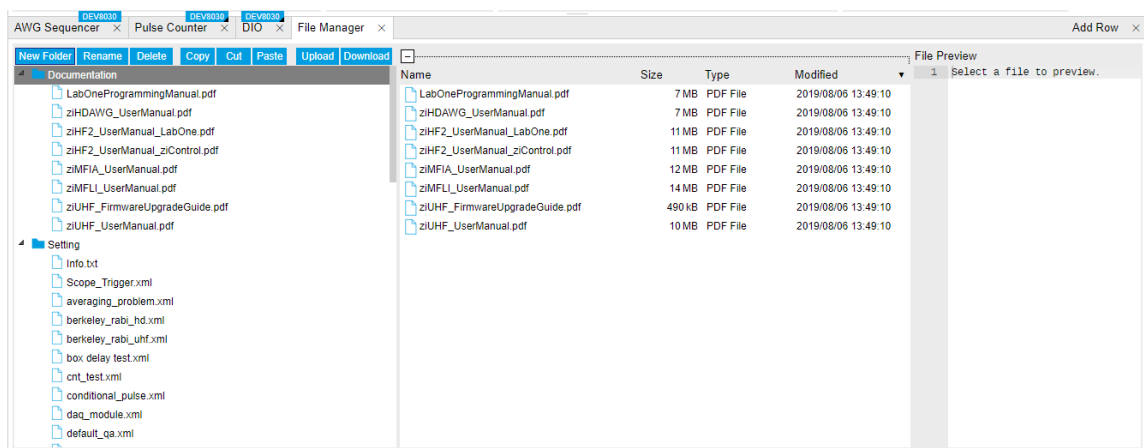
Figure 4.29. LabOne UI: File Manager tab

4.13.3. Functional Elements

Table 4.36. 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.

Control/Tool	Option/Range	Description
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).

4.14. AWG Sequencer Tab

The AWG Sequencer tab is available on all HDAWG Arbitrary Waveform Generator instruments.

4.14.1. Features

- 4- or 8-channel arbitrary waveform generator
- 64 MSa waveform memory per channel (500 MSa with HDAWG-ME Memory Extension option)
- Sequence branching
- Digital modulation
- Cross-domain trigger engine
- Sequence Editor with code highlighting and auto completion
- High-level programming language with waveform generation and editing toolset
- Waveform viewer

4.14.2. Description

The AWG Sequencer 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 4.37. App icon and short description

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

The AWG Sequencer tab (see Figure 4.30) 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.
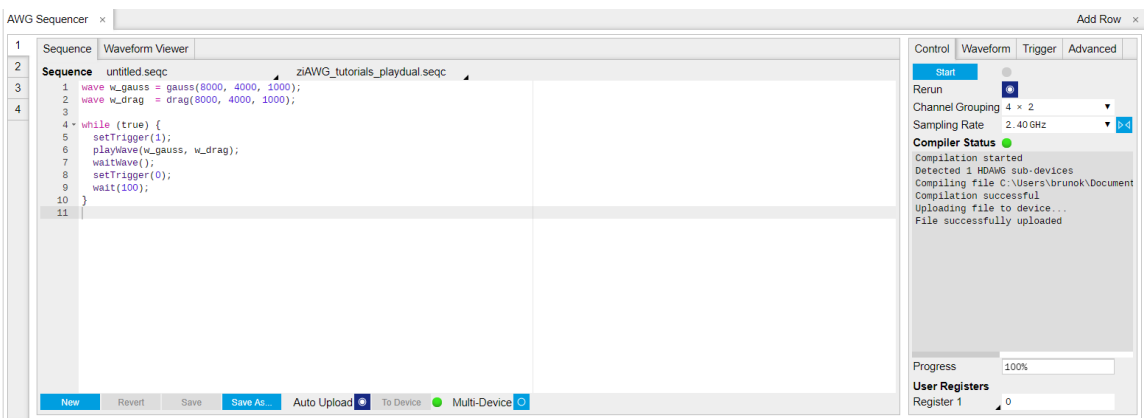
Figure 4.30. LabOne UI: AWG Sequencer tab

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 Chapter 3. The LabOne sequence

programming language is specified in detail in Section 4.14.3. 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 Section 3.1 for 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 4.31. 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. In order to understand the execution timing, it's helpful to consider the internal architecture of the AWG, consisting of the Sequencer itself, the Waveform Player, and the Waveform Memory. This is explained in Section 4.3.
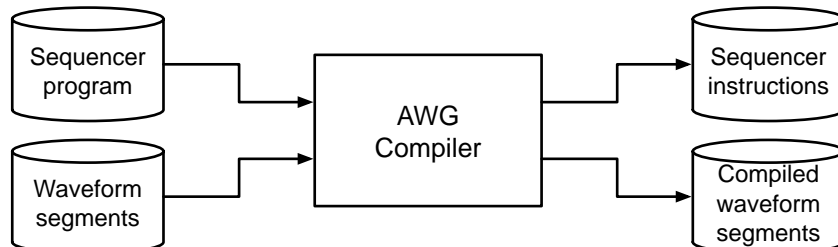
Figure 4.31. AWG sequence program compilation process

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 Files tab and their location in the directory structure is shown in Table 4.38. 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.

The channel grouping setting allows configuring the instrument such that the Wave outputs are controlled in groups of 2, 4, or 8. Changing this setting will dynamically adjust the number of side-tabs of the sequence editor as well as the layout of the Output tab.

Table 4.38. 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 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 **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. Each AWG core has two internal trigger input channels which can be configured to probe either one of the Trig inputs on the instrument front panel. Trigger levels and input coupling are configured in the DIO tab. The primary use of the triggers is to control the timing of the AWG signal relative to an external device. The DIO Trigger section allows setup of the dynamic sequencing functionality based on the `playWaveDIO` and `setWaveDIO` sequencer instructions.

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 4.39. 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

Shortcut	Action
Alt-Delete	Remove to line end
Alt-Backspace	Remove to line start
Ctrl-Backspace	Remove word left
Ctrl-Delete	Remove word right

Table 4.40. 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-PageUp	Select page up
Shift-PageDown	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 4.41. 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
PageUp	Go to page up
PageDown	Go to page down
Ctrl-Home	Go to start
Ctrl-End	Go to end
Ctrl-L	Go to line

Shortcut	Action
Ctrl-Down	Scroll line down
Ctrl-Up	Scroll line up
Ctrl-P	Go to matching bracket

Table 4.42. Find/Replace

Shortcut	Action
Ctrl-F	Find
Ctrl-H	Replace
Ctrl-K	Find next
Ctrl-Shift-K	Find previous

Table 4.43. Folding

Shortcut	Action
Alt-L	Fold selection
Alt-Shift-L	Unfold

Table 4.44. 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
Insert	Overwrite
Ctrl-Shift-E	Macros replay
Ctrl-Alt-E	Macros recording
Delete	Delete

4.14.3. 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 Chapter 3 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);
  // Wait until waveform playback has ended
  waitWave();
  // Play pulses simultaneously on both AWG channels
  playWave(gauss_pos, gauss_neg);
}
```

Multi-Instrument Support

The HDAWG 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 Section 4.16. We assume that two HDAWG 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 Sequencer tab, set Channel Grouping to MDS. 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 Wave Outputs is as follows (assuming an 8-channel instrument):

Channel number in sequence program	Instrument number (according to order in MDS tab)	Wave Output number
1	Master	1
2	Master	2
...	...	...
9	Slave 1	1
10	Slave 2	2
...	...	...

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 instrument 1 (Wave Output 1):
  playWave(1, w_gauss);
  // Pulse on instrument 2 (Wave Output 1):
  playWave(9, w_gauss);
  // Pulse on AWG 1 (Wave Outputs 1 & 2)
  //   and on AWG 2 (Wave Outputs 1 & 2):
```

```
       playWave(1, w_gauss,  2, w_gauss,
               9, w_gauss, 10, w_gauss);
       setTrigger(0);
       wait(100);
}
```

Keywords and Comments

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

Table 4.45. 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 (baseSamplingClock)/2^n. The AWG rates constants are specified only for the default base sampling clock of 2.4 GHz. For different values, it's advised to use directly the exponent n instead of the AWG rates constants.

Table 4.46. Predefined Constants

Name	Value	Description
`AWG_RATE_2400MHZ`	0	Constant to set Sampling Rate to 2.4 GHz.
`AWG_RATE_1200MHZ`	1	Constant to set Sampling Rate to 1.2 GHz.
`AWG_RATE_600MHZ`	2	Constant to set Sampling Rate to 600 MHz.
`AWG_RATE_300MHZ`	3	Constant to set Sampling Rate to 300 MHz.
`AWG_RATE_150MHZ`	4	Constant to set Sampling Rate to 150 MHz.
`AWG_RATE_75MHZ`	5	Constant to set Sampling Rate to 75 MHz.
`AWG_RATE_37P5MHZ`	6	Constant to set Sampling Rate to 37.5 MHz.
`AWG_RATE_18P75MHZ`	7	Constant to set Sampling Rate to 18.75 MHz.
`AWG_RATE_9P4MHZ`	8	Constant to set Sampling Rate to 9.4 MHz.
`AWG_RATE_4P5MHZ`	9	Constant to set Sampling Rate to 4.5 MHz.
`AWG_RATE_2P34MHZ`	10	Constant to set Sampling Rate to 2.34 MHz.
`AWG_RATE_1P2MHZ`	11	Constant to set Sampling Rate to 1.2 MHz.
`AWG_RATE_586KHZ`	12	Constant to set Sampling Rate to 586 kHz.
`AWG_RATE_293KHZ`	13	Constant to set Sampling Rate to 293 kHz.
`DEVICE_SAMPLE_RATE`	<actual device sample rate>	

Name	Value	Description
ZSYNC_DATA_RAW	0	
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.

Table 4.47. Waveform Generation

Function	Description
`wave zeros(const samples)` ARGUMENTS samples : Number of samples in the waveform RETURN resulting waveform	Constant amplitude of 0 over the defined number of samples. $h(x) = 0$
`wave ones(const samples)` ARGUMENTS samples : Number of samples in the waveform RETURN resulting waveform	Constant amplitude of 1 over the defined number of samples. $h(x) = 1$

Function	Description

`wave sine(const samples, const`
`  amplitude=1.0, const phaseOffset, const`
`  nrOfPeriods)`
 	
`ARGUMENTS`
`samples : Number of samples in the waveform`
`amplitude : Amplitude of the signal`
`  (optional)`
`phaseOffset : Phase offset of the signal in`
`  radians`
`nrOfPeriods : Number of Periods within the`
`  defined number of samples`
 	
`RETURN`
`resulting waveform`
	Sine function with arbitrary amplitude (a), phase offset in radians (p), number of periods (f) and number of samples (N).
	$h(x) = a \cdot \sin(2\pi f \frac{x}{N} + p)$

`wave cosine(const samples, const`
`  amplitude=1.0, const phaseOffset, const`
`  nrOfPeriods)`
 	
`ARGUMENTS`
`samples : Number of samples in the waveform`
`amplitude : Amplitude of the signal`
`  (optional)`
`phaseOffset : Phase offset of the signal in`
`  radians`
`nrOfPeriods : Number of Periods within the`
`  defined number of samples`
 	
`RETURN`
`resulting waveform`
	Cosine function with arbitrary amplitude (a), phase offset in radians (p), number of periods (f) and number of samples (N).
	$h(x) = a \cdot \cos(2\pi f \frac{x}{N} + p)$

`wave sinc(const samples, const`
`  amplitude=1.0, const position, const beta)`
 	
`ARGUMENTS`
`samples : Number of samples in the waveform`
`amplitude : Amplitude of the signal`
`  (optional)`
`position : Peak position of the function`
`beta : Width of the function`
 	
`RETURN`
`resulting waveform`
	Normalized sinc function with control of peak position (p), amplitude (a), width (beta) and number of samples (N).
	$h(x) = \begin{cases} a & \text{if } x = p \\ a \cdot \frac{\sin(2\pi \cdot beta \cdot \frac{x-p}{N})}{2\pi \cdot beta \cdot \frac{x-p}{N}} & \text{else} \end{cases}$

`wave ramp(const samples, const startLevel,`
`  const endLevel)`
 	
`ARGUMENTS`
`samples : Number of samples in the waveform`
`startLevel : Peak position of the function`
`endLevel : Width of the function`
 	
`RETURN`
`resulting waveform`
	Linear ramp from the start (s) to the end level (e) over the number of samples (N).
	$h(x) = s + \frac{x(e-s)}{N-1}$

`wave sawtooth(const samples, const`
`  amplitude=1.0, const phaseOffset, const`
`  nrOfPeriods)`
 	
`ARGUMENTS`
`samples : Number of samples in the waveform`
`amplitude : Amplitude of the signal`
`phaseOffset : Phase offset of the signal in`
`  radians`
`nrOfPeriods : Number of Periods within the`
`  defined number of samples`
	Sawtooth function with arbitrary amplitude, phase in radians and number of periods.

Function	Description
`RETURN` `resulting waveform`	

`wave triangle(const samples, const`
` amplitude=1.0, const phaseOffset, const`
` nrOfPeriods)`
 	
`ARGUMENTS`
`samples : Number of samples in the waveform`
`amplitude : Amplitude of the signal`
`phaseOffset : Phase offset of the signal in`
` radians`
`nrOfPeriods : Number of Periods within the`
` defined number of samples`
 	
`RETURN`
`resulting waveform`
	Triangle function with arbitrary amplitude, phase in radians and number of periods.

`wave gauss(const samples, const`
` amplitude=1.0, const position, const width)`
 	
`ARGUMENTS`
`samples : Number of samples in the waveform`
`amplitude : Amplitude of the signal`
` (optional)`
`position : Peak position of the pulse`
`width : Width of the pulse`
 	
`RETURN`
`resulting waveform`
	Gaussian pulse with arbitrary amplitude (a), position (p), width (w) and number of samples (N). $$h(x) = a \cdot e^{-\frac{(x-p)^2}{2 \cdot w^2}}$$

`wave drag(const samples, const`
` amplitude=1.0, const position, const width)`
 	
`ARGUMENTS`
`samples : Number of samples in the waveform`
`amplitude : Amplitude of the signal`
` (optional)`
`position : Center point position of the`
` pulse`
`width : Width of the pulse`
 	
`RETURN`
`resulting waveform`
	Derivative of Gaussian pulse with arbitrary amplitude (a), position (p), width (w) and number of samples (N). $$h(x) = -a \cdot \frac{\sqrt{e}(p-x)}{w} \cdot e^{-\frac{(x-p)^2}{2 \cdot w^2}}$$

`wave blackman(const samples, const`
` amplitude=1.0, const alpha)`
 	
`ARGUMENTS`
`samples : Number of samples in the waveform`
`amplitude : Amplitude of the signal`
` (optional)`
`alpha : Width of the function`
 	
`RETURN`
`resulting waveform`
	Blackman window function with arbitrary amplitude (a), alpha parameter and number of samples (N). $$h(x) = -a \cdot (\alpha_0 - \alpha_1 \cos(\frac{2\pi x}{N-1})$$ $$+ \alpha_2 \cos(\frac{4\pi x}{N-1}))$$ $$\alpha_0 = \frac{1-\alpha}{2}; \quad \alpha_1 = \frac{1}{2}; \quad \alpha_2 = \frac{\alpha}{2};$$

`wave hamming(const samples, const`
` amplitude=1.0)`
 	
`ARGUMENTS`
`samples : Number of samples in the waveform`
`amplitude : Amplitude of the signal`
` (optional)`
 	
`RETURN`
`resulting waveform`
	Hamming window function with arbitrary amplitude (a) and number of samples (N). $$h(x) = a \cdot (\alpha - \beta \cos(\frac{2\pi x}{N-1}))$$ $$\text{with } \alpha = 0.54 \text{ and } \beta = 0.46$$

Function	Description

`wave hann(const samples, const`
`amplitude=1.0)`
 	
`ARGUMENTS`
`samples : Number of samples in the waveform`
`amplitude : Amplitude of the signal`
 	
`RETURN`
`resulting waveform`
	Hann window function with arbitrary amplitude (a) and number of samples (N). $h(x) = a \cdot 0.5 \cdot (1 - \cos(\frac{2\pi x}{N-1}))$

`wave rect(const samples, const amplitude)`
 	
`ARGUMENTS`
`samples : Number of samples in the waveform`
`amplitude : Amplitude of the signal`
 	
`RETURN`
`resulting waveform`
	Rectangle function, constants amplitude (a) over the defined number of samples. $h(x) = \mathrm{a}$

`wave marker(const samples, const`
`markerValue)`
 	
`ARGUMENTS`
`samples : Number of samples in the waveform`
`markerValue : Value of the marker bits`
 	
`RETURN`
`resulting waveform`
	Generate a waveform with marker bits set to the specified value. The analog part of the waveform is zero.

`wave rand(const samples, const`
`amplitude=1.0, const mean, const stdDev)`
 	
`ARGUMENTS`
`samples : Number of samples in the waveform`
`amplitude : Amplitude of the signal`
`mean : Average signal level`
`stdDev : Standard deviation of the noise`
`signal`
 	
`RETURN`
`resulting waveform`
	White noise with arbitrary amplitude, power and standard deviation.

`wave randomGauss(const samples, const`
`amplitude=1.0, const mean, const stdDev)`
 	
`ARGUMENTS`
`samples : Number of samples in the waveform`
`amplitude : Amplitude of the signal`
`mean : Average signal level`
`stdDev : Standard deviation of the noise`
`signal`
 	
`RETURN`
`resulting waveform`
	White noise with arbitrary amplitude, power and standard deviation.

`wave randomUniform(const samples, const`
`amplitude=1.0)`
 	
`ARGUMENTS`
`samples : Number of samples in the waveform`
`amplitude : Amplitude of the signal`
 	
`RETURN`
`resulting waveform`
	Random waveform with uniform distribution.

`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

Function	Description

ARGUMENTS
`samples : Number of samples in the waveform`
`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`
`initial : LFSR initial state, any nonzero`
` value will work, usually 0x1`
 	
RETURN
`resulting waveform`
	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)".

`wave chirp(const samples, const`
` amplitude=1.0, const startFreq, const`
` stopFreq, const phase=0)`
 	
ARGUMENTS
`samples : Number of samples in the waveform`
`amplitude : Amplitude of the signal`
` (optional)`
`startFreq : Start frequency of the signal`
`stopFreq : Stop Frequency of the signal`
`phase : Initial phase of the signal`
` (optional)`
 	
RETURN
`resulting waveform`
	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.

`wave rrc(const samples, const amplitude=1.0,`
` const position, const beta, const width)`
 	
ARGUMENTS
`samples : Number of samples in the waveform`
`amplitude : Amplitude of the signal`
`position : Center point position of the`
` pulse`
`beta : Roll-off factor`
`width : Width of the pulse`
 	
RETURN
`Resulting waveform`
	Root raised cosine function with arbitrary amplitude (a), position (p), roll-off factor (beta) and width (w) and number of samples (N).
	
	$$h(y) = a\frac{\sin(y\pi(1-\beta)) + 4y\beta\cos(y\pi(1+\beta))}{y\pi(1-(4y\beta)^2)}$$
	$$\text{with } y(x) = 2w\frac{x-p}{N}$$

`wave vect(const value,...)`
 	
ARGUMENTS
`value : Waveform amplitude at the respective`
` sample`
 	
RETURN
`resulting waveform`
	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.

`wave placeholder(const samples, const`
` marker0=false, const marker1=false)`
 	
ARGUMENTS
`samples : Number of samples in the waveform`
`marker0 : true if marker bit 0 must be used`
` (default false)`
`marker1 : true if marker bit 1 must be used`
` (default false)`
 	
RETURN
`waveform object`
	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.

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

Table 4.48. Waveform Editing

Function	Description

<pre>wave join(wave wave1, wave wave2, const
 interpolLength=0)
 	
ARGUMENTS
wave1 : Input waveform
wave2 : Input waveform
interpolLength : Number of samples to
 interpolate between waveforms (optional,
 default 0)
 	
RETURN
joined waveform</pre>
	Connect two or more waveforms with optional linear interpolation between the waveforms.

<pre>wave join(wave wave1, wave wave2,...)
 	
ARGUMENTS
wave1 : Input waveform
wave2 : Input waveform
 	
RETURN
joined waveform</pre>
	Connect two or more waveforms.

<pre>wave interleave(wave wave1, wave wave2,...)
 	
ARGUMENTS
wave1 : Input waveform
wave2 : Input waveform
 	
RETURN
interleaved waveform</pre>
	Interleave two or more waveforms sample by sample.

<pre>wave add(wave wave1, wave wave2,...)
 	
ARGUMENTS
wave1 : Input waveform
wave2 : Input waveform
 	
RETURN
sum waveform</pre>
	Add two or more waveforms sample by sample. Alternatively, the "+" operator may be used for waveform adddition.

<pre>wave multiply(wave wave1, wave wave2,...)
 	
ARGUMENTS
wave1 : Input waveform
wave2 : Input waveform
 	
RETURN
product waveform</pre>
	Multiply two or more waveforms sample by sample. Alternatively, the "*" operator may be used for waveform multiplication.

<pre>wave scale(wave waveform, const factor)
 	
ARGUMENTS
waveform : Input waveform
factor : Scaling factor
 	
RETURN
scaled waveform</pre>
	Scale the input waveform with the factor and return the scaled waveform. The input waveform remains unchanged.

<pre>wave flip(wave waveform)
 	
ARGUMENTS
waveform : Input waveform
 	
RETURN
flipped waveform</pre>
	Flip the input waveform back to front and return the flipped waveform. The input waveform remains unchanged.

Function	Description

```wave cut(wave waveform, const from, const to)```
 	
```ARGUMENTS```
```waveform : Input waveform```
```from : First sample of the cut waveform```
```to : Last sample of the cut waveform```
 	
```RETURN```
```cut waveform```
	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".

```wave filter(wave b, wave a, wave x)```
 	
```ARGUMENTS```
```b : Numerator coefficients```
```a : Denominator coefficients```
```x : Input waveform```
 	
```RETURN```
```filtered waveform```
	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.
	
	$$y(n) = \frac{1}{a_0} \left(\sum_{i=0}^{M} b_i x_{n-i} - \sum_{i=1}^{N} a_i y_{n-i} \right)$$
	$$\text{with } M = \text{length}(b) - 1$$
	$$\text{and } N = \text{length}(a) - 1$$

```wave circshift(wave a, const n)```
 	
```ARGUMENTS```
```waveform : Input waveform```
```n : Number of elements to shift```
 	
```RETURN```
```circularly shifted waveform```
	Circularly shifts a 1D waveform and returns it.

Waveform Playback and Predefined Functions

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

Table 4.49. Predefined Functions

Function	Description

```void setDIO(var value)```
 	
```ARGUMENTS```
```value : The value to write to the DIO (const or var)```
	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.

```var getDIO()```
 	
```RETURN```
```var containing the read value```
	Reads a 32-bit value from the DIO bus.

```var getDIOTriggered()```
 	
```RETURN```
```var containing the read value```
	Reads a 32-bit value from the DIO bus as recorded at the last DIO trigger position.

```void setTrigger(var value)```
 	
```ARGUMENTS```
	Sets the AWG Trigger output signals. The state of all four AWG Trigger output signals is represented by the bits in the

Function	Description
`value : to be written to the trigger output lines`	binary representation of the integer value. Binary notation of the form 0b0000 is recommended for readability.

`void setID(var id)`
 	
`ARGUMENTS`
`id : The new ID to be attached to streaming data of the device`
	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.

`void setSeqIndex(var id)`
 	
`ARGUMENTS`
`id : The new ID to be attached to streaming data of the device`
	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.

`void assignWaveIndex(const output, wave waveform, const index)`
 	
`ARGUMENTS`
`output : defines on which output the following waveform to be played`
`waveform : waveform to be played`
`index : index to assign`
	Assigns an index to the specified waveform(s). Channels/waveforms specification must match that of the corresponding playWave. Once program is loaded to AWG, the waveforms can be r/w accessed via the "<dev>/AWGS/<n>/WAVEFORM/WAVES/<index>" node, and addressed from the command table using the assigned index.

`void assignWaveIndex(wave waveform, const index)`
 	
`ARGUMENTS`
`waveform : waveform to be played`
`index : index to assign`
	Assigns an index to the specified waveform(s). Once a sequencer program is uploaded to the AWG, the waveforms can be written and read via the "<dev>/AWGS/<n>/WAVEFORM/WAVES/<index>" API node, and addressed from the command table using the assigned index. Channels/waveforms specification must match that of the corresponding playWave.

`void prefetch(const output, wave waveform)`	

`void prefetch(const output, wave waveform,...)`
 	
`ARGUMENTS`
`output : defines on which output the following waveform to be played`
`waveform : waveform to be played`
	Prefetches the given waveforms for the defined output channels. It can contain multiple waveforms with an output definition. Must match play that follows, use to ensure seamless playback.

`void prefetch(wave waveform)`	

`void prefetch(wave waveform,...)`
 	
`ARGUMENTS`
`waveform : waveform to be played`
	Prefetches the given waveforms. Output channels are assigned automatically depending on the number of input waveforms. Must match play that follows, use to ensure seamless playback.

Function	Description

`void prefetchIndexed(const output, wave`
` waveform, var offset, const length)`
 	
`ARGUMENTS`
`output : defines on which output the`
` following waveform to be played`
`waveform : waveform to be played`
`offset : offset in samples from the start of`
` the waveform`
`length : number of samples to be played from`
` this waveform`
	Prefetches the specified part of the given waveforms for the defined output channels. It can contain multiple waveforms with an output definition. Must match playIndexed that follows, use to ensure seamless playback.

`void prefetchIndexed(wave waveform, var`
` offset, const length)`
 	
`ARGUMENTS`
`waveform : waveform to be played`
`offset : offset in samples from the start of`
` the waveform`
`length : number of samples to be played from`
` this waveform`
	Prefetches the specified part of the given waveforms, channels are assigned automatically depending on the number of input waveforms. Must match playIndexed that follows, use to ensure seamless playback.

`void playWave(const output, wave waveform,`
` const rate=AWG_RATE_DEFAULT)`
 	
`ARGUMENTS`
`output : defines on which output the`
` following waveform is played`
`waveform : waveform to be played`
`rate : sample rate with which the AWG plays`
` the waveforms (default set in the user`
` interface).`
	Starts to play the given waveforms on the defined output channels. The playback begins as soon as the previous waveform playback is finished.

`void playWave(const output, wave`
` waveform,...)`
 	
`ARGUMENTS`
`output : defines on which output the`
` following waveform is played`
`waveform : waveform to be played`
	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.

`void playWave(wave waveform, const`
` rate=AWG_RATE_DEFAULT)`
 	
`ARGUMENTS`
`waveform : waveform to be played`
`rate : sample rate with which the AWG plays`
` the waveforms (default set in the user`
` interface).`
	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.

`void playWave(wave waveform,...)`
 	
`ARGUMENTS`
`waveform : waveform to be played`
	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.

`void playWaveNow(const output, wave`
` waveform, const rate=AWG_RATE_DEFAULT)`
 	
`ARGUMENTS`
	Starts to play the given waveforms on the defined output channels. It starts immediately even if the previous waveform playback is still in progress.

Function	Description

`output : defines on which output the`
`  following waveform is played`
`waveform : waveform to be played`
`rate : sample rate with which the AWG plays`
`  the waveforms (default set in the user`
`  interface).`	

`void playWaveNow(const output, wave`
`  waveform,...)`
 	
`ARGUMENTS`
`output : defines on which output the`
`  following waveform is played`
`waveform : waveform to be played`
	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.

`void playWaveNow(wave waveform, const`
`  rate=AWG_RATE_DEFAULT)`
 	
`ARGUMENTS`
`waveform : waveform to be played`
`rate : sample rate with which the AWG plays`
`  the waveforms (default set in the user`
`  interface).`
	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.

`void playWaveNow(wave waveform,...)`
 	
`ARGUMENTS`
`waveform : waveform to be played`
	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.

`void playWaveIndexed(const output, wave`
`  waveform, var offset, const length, const`
`  rate=AWG_RATE_DEFAULT)`
 	
`ARGUMENTS`
`output : defines on which output the`
`  following waveform is played`
`waveform : waveform to be played`
`offset : offset in samples from the start of`
`  the waveform`
`length : number of samples to be played from`
`  this waveform`
`rate : sample rate with which the AWG plays`
`  the waveforms (default set in the user`
`  interface).`
	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.

`void playWaveIndexed(wave waveform,`
`  var offset, const length, const`
`  rate=AWG_RATE_DEFAULT)`
 	
`ARGUMENTS`
`waveform : waveform to be played`
`offset : offset in samples from the start of`
`  the waveform`
`length : number of samples to be played from`
`  this waveform`
`rate : sample rate with which the AWG plays`
`  the waveforms (default set in the user`
`  interface).`
	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.

Function	Description

`void playWaveIndexedNow(const output, wave waveform, var offset, const length, const rate=AWG_RATE_DEFAULT)`
 	
`ARGUMENTS`
`output : defines on which output the following waveform is played`
`waveform : waveform to be played`
`offset : offset in samples from the start of the waveform`
`length : number of samples to be played from this waveform`
`rate : sample rate with which the AWG plays the waveforms (default set in the user interface).`
	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.

`void playWaveIndexedNow(wave waveform, var offset, const length, const rate=AWG_RATE_DEFAULT)`
 	
`ARGUMENTS`
`waveform : waveform to be played`
`offset : offset in samples from the start of the waveform`
`length : number of samples to be played from this waveform`
`rate : sample rate with which the AWG plays the waveforms (default set in the user interface).`
	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.

`void playDIOWave(wave waveform, const rate=AWG_RATE_DEFAULT)`
 	
`ARGUMENTS`
`waveform : waveform to be played`
`rate : sample rate with which the AWG plays the waveforms (default set in the user interface).`
	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.

`void playDIOWave(wave waveform,...)`
 	
`ARGUMENTS`
`waveform : waveform to be played`
	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.

`void playZero(const samples)`
 	
`ARGUMENTS`
`samples : Number of samples to be played. The same min length and granularity applies as for regular waveforms.`
	Starts to play zeros on all channels for the specified number of samples. Behaves as if same length all-zeros waveform is played using playWave, but without consuming waveform memory.

`void playZero(const samples, const rate)`
 	
`ARGUMENTS`
	Starts to play zeros on all channels for the specified number of samples. Behaves as if same length all-zeros waveform is played

Function	Description

`samples : Number of samples to be played.`
` The same min length and granularity applies`
` as for regular waveforms.`
`rate : Sample rate with which the AWG plays`
` zeros (default set in the user interface).`
	using playWave, but without consuming waveform memory.

`void wait(var cycles)`
 	
`ARGUMENTS`
`cycles : number of cycles to wait`
	Waits for the given number of Sequencer clock cycles (3.33 ns per cycle). The execution of the instruction adds an offset of 3 clock cycles, i.e., the statement wait(0) leads to a waiting time of 9.99 ns.

`void waitWave()`	Waits until the AWG is done playing the current waveform.

`void waitTrigger(const mask, const value)`
 	
`ARGUMENTS`
`mask : mask to be applied to the input`
` signal`
`value : value to be compared with the`
` trigger input`
	Waits until the masked trigger input is equal to the given value.

`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.

`void setRate(const rate)`
 	
`ARGUMENTS`
`rate : New default sampling rate`
	Overwrites the default Sampling Rate for the following playWave commands.

`var getDigTrigger(const index)`
 	
`ARGUMENTS`
`index : index of the Digital Trigger input`
` to be read; can be either 1 or 2 on UHF, or`
` 1-8 on HDAWG`
 	
`RETURN`
`trigger state, either 0 or 1`
	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.

`void setUserReg(const register, var value)`
 	
`ARGUMENTS`
`register : The register index (0 to 15) to`
` be written to`
`value : The integer value to be written`
	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.

`var getUserReg(const register)`
 	
`ARGUMENTS`
`register : The register to be read (0 to 15)`
 	
`RETURN`
`current register value`
	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.

`void setInt(string path, var value)`
 	
`ARGUMENTS`
`path : The node path to be written to`
`value : The integer value to be written`
	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.

Function	Description

`void setDouble(string path, var value)`
 	
`ARGUMENTS`
`path : The node path to be written to`
`value : The integer or floating point value`
` to be written`
	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.

`void setDouble(string path, var value, const`
` scale)`
 	
`ARGUMENTS`
`path : The node path to be written to`
`value : The integer or floating point value`
` to be written`
`scale : Scaling value to be applied to the`
` value before writing to the node`
	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.

`void sync()`	Perform Multi-Device synchronization command for all devices at this point. Master/Slave assignment is automatic. Only for programs running on multiply synchronized instruments.
`void randomSeed()`	Generate a new seed for the subsequent random vector commands.

`void error(string msg,...)`
 	
`ARGUMENTS`
`msg : Message to be displayed`
	Throws the given error message when reached.

`void info(string msg,...)`
 	
`ARGUMENTS`
`msg : Message to be displayed`
	Returns the specified message when reached.

`void setWaveDIO(const index, const channel,`
` wave waveform, const rate)`
 	
`ARGUMENTS`
`index : in the table where the waveform`
` should be added`
`channel : to play the waveform on`
`waveform : the actual waveform`
`rate : sample rate with which the AWG plays`
` the waveforms (default set in the user`
` interface)`
	Adds an entry to the waveform table at the given index. The waveform table is used by the playWaveDIO instruction to selectively play back a waveform based on a codeword input on the DIO interface.

`void playWaveDigTrigger(const trigger,`
` const output, wave waveform, const`
` rate=AWG_RATE_DEFAULT)`
 	
`ARGUMENTS`
`trigger : defines which trigger input should`
` start the waveform playback`
`output : defines on which output the`
` following waveform is played`
`waveform : waveform to be played`
`rate : sample rate with which the AWG plays`
` the waveforms (default set in the user`
` interface).`
	Starts to play the given waveforms on the defined output channels. The playback begins as soon as the previous waveform playback is finished and the given trigger is detected.

Function	Description

```void playWaveDigTrigger(const trigger, const output, wave waveform,...)```
 	
ARGUMENTS
```trigger : defines which trigger input should start the waveform playback```
```output : defines on which output the following waveform is played```
```waveform : waveform to be played```
	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 and the given trigger is detected.

```void playWaveDigTrigger(const trigger, wave waveform, const rate=AWG_RATE_DEFAULT)```
 	
ARGUMENTS
```trigger : defines which trigger input should start the waveform playback```
```waveform : waveform to be played```
```rate : sample rate with which the AWG plays the waveforms (default set in the user interface).```
	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 and the given trigger is detected.

```void playWaveDigTrigger(const trigger, wave waveform,...)```
 	
ARGUMENTS
```trigger : defines which trigger input should start the waveform playback```
```waveform : waveform to be played```
	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 and the given trigger is detected.

```void playWaveDIO()```	Starts to play a waveform from the table defined by the setWaveDIO instruction. The waveform is selected according to the integer codeword currently read on the DIO interface. The codeword is specified by the Codeword Mask and Codeword Shift settings in the AWG Sequencer tab.

```void waitDigTrigger(const value)```
 	
ARGUMENTS
```index : Index of the digital trigger input; can be either 1 or 2.```
	Waits for the reception of a digital trigger. The physical signal connected to the AWG Digital Trigger inputs is to be configured in the Trigger sub-tab of the AWG Sequencer tab.

```void waitSineOscPhase(const sine)```
 	
ARGUMENTS
```sine : index of the sine generator to be waited on, can be 1 or 2```
	Waits until the oscillator phase of the indexed sine generator (1-2) reaches a zero crossing (negative -> positive, start of sine period). This command is not supported in the 2x4 and 1x8 channel grouping modes.

```void resetOscPhase(const mask)```
 	
ARGUMENTS
```mask : one-hot encoding to reset phase of individual oscillators```
	Reset the phase of the oscillators specified by the binary mask argument. Sequence programs using these instructions can only be run when AWG Oscillator Control is enabled. Each AWG core can access only a subset of all oscillators. On instruments with HDAWG-MF option, core 1 can access oscillators 1-4, core 2 oscillators 5-8, etc.

```void resetOscPhase()```	Reset the phase of all oscillators controllable by the AWG cores in use. Sequence programs using these

Function	Description
	instructions can only be run when AWG Oscillator Control is enabled.

`void setSinePhase(const sine, const phase)`
 	
`ARGUMENTS`
`sine : Sine generator relative index (0 or 1)`
`phase : Phase value [degree]`
	Set the phase in units of degree of the first (index 0) or second (index 1) sine generator of the AWG core in use. When used in channel grouping mdoes 1x8, 2x4, or 1x4, this instruction will set the phase of the first/second sine generator of all involved AWG cores. The phase is reset to 0 after execution of the sequence program.

`void incrementSinePhase(const sine, const phase)`
 	
`ARGUMENTS`
`sine : Sine generator relative index (0 or 1)`
`phase : Phase increment value [degree]`
	Increment the phase of the first (index 0) or second (index 1) sine generator of the AWG core in use by the given value in degrees. When used in channel grouping mdoes 1x8, 2x4, or 1x4, this instruction will increment the phase of the first/second sine generator of all involved AWG cores. The phase is reset to 0 after execution of the sequence program.

`var getCnt(const counter)`
 	
`ARGUMENTS`
`counter : The counter to be read; can be either 0 or 1.`
 	
`RETURN`
`current counter value`
	Reads the value from one of the Pulse Counters. The Pulse Counters can be used for measuring and counting events on the trigger inputs. Each AWG Sequencer is connected to a pair of Pulse Counters.

`void waitCntTrigger(const counter)`
 	
`ARGUMENTS`
`counter : The index of the counter connected to the AWG Sequencer; can be either 0 or 1.`
	Waits until the given Pulse Counter is active. Each AWG Sequencer is connected to a pair of Pulse Counters.

`void setPrecompClear(const value)`
 	
`ARGUMENTS`
`value : Whether to clear the precompensation (0: no, 1: yes)`
	Set the configuration for the synchronous clearing of the precompensation filters. The precompensation filter are cleared synchronously with the waveforms played by the playWave or playWaveDIO commands.

`void waitPlayQueueEmpty()`	Waits until the AWG has started playing the current waveform.

`void executeTableEntry(var index)`
 	
`ARGUMENTS`
`index : table entry that shall be executed`
	Execute the entry of the AWG command/waveform table with the given index. An entry of the command/waveform table contains a waveform playback instruction as well as additional space for instructions for real-time setting of Sine Generator phases and AWG amplitude. The entries of the command/waveform table can be specified by direct upload of a JSON string to the "<dev>/AWGS/<n>/COMMANDTABLE/DATA" API node.

`void setPRNGSeed(var value)`
 	
`ARGUMENTS`
	Sets the seed for the linear-shift feedback register lsfr of the pseudo random number

Function	Description
`value : seed value to be configured`	generator (PRNG). The seed is a 16 bit unsigned int value. Zero is invalid as seed.

`var getPRNGValue()`
 	
`RETURN`
`Random value rand`
	Returns a random value from the pseudo-random number generator (PRNG). The PRNG is implemented as a Galois linear-shift feedback register according to the pseudo code below. The feedback register lsfr is initialized to a seed value using the function setPRNGSeed. The values lower and upper are set using the function setPRNGRange. The feedback register lsfr is stored from one call of the function getPRNGValue to the next, which renders the pseudo code recursive. In the pseudo code, XOR and AND are bitwise logical operators, and $\gg$ is the right bit shift operator. lsb = lsfr AND 1; lsfr = lsfr $\gg$ 1; if (lsb == 1) then: lsfr = 0xb400 XOR lsfr; rand = ((lsfr * (upper-lower+1) $\gg$ 16) + lower) AND 0xffff;

`void setPRNGRange(var lower, var upper)`
 	
`ARGUMENTS`
`lower : lower bound of range, 0 ... 2**16-1`
`upper : upper bound of range, 0 ... 2**16-1`
	Configures the range of the psuedo random number generator (PRNG) to generate output in range [lower, upper].

`var getZSyncData(const data_type)`
 	
`ARGUMENTS`
`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.`
 	
`RETURN`
`var containing the read value`
	Read the last received message on ZSync. The argument specify which data the function should return.

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 4.50. Mathematical Functions

Function	Description
`const abs(const c)`	absolute value

Function	Description
`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
`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 4.51. Mathematical Constants

Name	Value	Description
`M_E`	2.71828182845904523536028747135266250	e
`M_LOG2E`	1.44269504088896340735992468100189214	log2(e)
`M_LOG10E`	0.434294481903251827651128918916605082	log10(e)
`M_LN2`	0.693147180559945309417232121458176568	loge(2)
`M_LN10`	2.30258509299404568401799145468436421	loge(10)
`M_PI`	3.14159265358979323846264338327950288	pi

Name	Value	Description
M_PI_2	1.5707963267948966192313216916397514	pi/2
M_PI_4	0.78539816339744830961566084581987572	pi/4
M_1_PI	0.31830988618379067153776752674502872	1/pi
M_2_PI	0.63661977236758134307553505349005744	2/pi
M_2_SQRTPI	1.12837916709551257389615890312154517	2/sqrt(pi)
M_SQRT2	1.41421356237309504880168872420969808	sqrt(2)
M_SQRT1_2	0.70710678118654752440084436210484903	1/sqrt(2)

Table 4.52. 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
<<	left bit shift	7
>>	right bit shift	7
+	addition	8
−	subtraction	8
*	multiplication	9
/	division	9
~	bit-wise logical negation	10

Table 4.53. 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

Operator	Description	Priority
|	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 and 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 which later can be iterated through with the `playWaveIndexed` instruction. 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 (compile-time execution)
cvar i;
wave w_pulses;
for (i = 0; i < 10; i = i + 1) {
  w_pulses = join(w_pulses, i*0.1*gauss(1000, 500, 100));
}

// For loop example (run-time execution)
var j;
for (j = 9; j >= 0; j = j - 1) {
  playWaveIndexed(w_pulses, j*1000, 1000);
}
```

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);
}
```

4.14.4. Functional Elements

Table 4.54. AWG tab: Control sub-tab

Control/Tool	Option/Range	Description
Start		Runs the AWG.
Sampling Rate	293 kSa/s to 2.4 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.

Control/Tool	Option/Range	Description
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.
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

Control/Tool	Option/Range	Description
		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 4.55. 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 512 kSa of waveform data per AWG core. Mem Usage > 100% means that waveforms must be loaded from the main memory (64 or 500 MSa per channel) during playback.

Table 4.56. AWG tab: Trigger sub-tab

Control/Tool	Option/Range	Description
Trigger State	grey/green	State of the Trigger. Grey: No trigger detected. Green: Trigger detected.
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.
Auxiliary Trigger State	grey/green	State of the Auxiliary Trigger. Grey: No trigger detected. Green: Trigger detected.
Signal	Trigger In 1	Selects the digital trigger source signal.
	Trigger In 2	
	Trigger In 3	
	Trigger In 4	
	Trigger In 5	
	Trigger In 6	
	Trigger In 7	
	Trigger In 8	
DIO Trigger State	grey/green	Indicates that triggers are generated from the DIO 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 connecton with the waitDIOTrigger sequencer instruction.
Strobe Slope		Select the signal edge that activates the STROBE trigger in connecton 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.

Control/Tool	Option/Range	Description
	High	VALID bit must be logical high.
	Both	VALID bit may be logical high or logical low.
Codeword Mask	0 to 1023	10-bit value to select the bits of the DIO interface input state (notation DIO[n] in the Specification chapter of the User Manual) to be used as a codeword in connection with the playWaveDIO sequencer instruction. The Codeword Mask is combined with the DIO interface input state by a bitwise AND operation after applying the Codeword Shift.
Codeword Shift	0 to 31	Defines the integer bit shift to be applied to the input state of the DIO interface (notation DIO[n] in the Specification chapter of the User Manual) to be used as a codeword for waveform selection in connection with the playWaveDIO sequencer instruction.
High bits		32-bit value indicating which bits on the DIO interface are detected as logic high.
Low bits		32-bit value indicating which bits on the DIO interface are detected as logic low.
Timing Error	grey/red	Indicates a timing error. A timing error is defined as an event where either the VALID or any of the data bits on the DIO interface change value at the same time as the STROBE bit.

Table 4.57. AWG tab: Advanced sub-tab

Control/Tool	Option/Range	Description
Sequence Editor		Display and edit the sequence program.
Assembly	Text display	Displays the current sequence program in compiled form. Every line corresponds to one hardware instruction and requires one clock cycle (3.33 ns at the default sampling rate of the instrument) for execution.

Control/Tool	Option/Range	Description
AWG Core		Select the AWG Core for which you want to display the assembly and command table information. Available select options (if any) depend on the Channel Grouping value.
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 16384 instructions.
Clear		Clears the command table description for the selected AWG Core.
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.

4.15. Pulse Counter Tab

The Pulse Counter tab relates to the HDAWG-CNT Pulse counter option and is only available if this option is installed on the instrument (see Information section in the Device tab). The number of counter modules depends on the number of channels of the instrument.

4.15.1. Features

- 4 or 8 counter modules
- 300 MHz maximum count rate
- 4 modes: free running, gated, gated free running, and pulse tagging
- 4 or 8 analog signal inputs with adjustable discriminator level
- 32 digital signal inputs
- Background subtraction
- Count integration

4.15.2. Description

The Pulse Counter tab provides access to the pulse counter settings. 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 4.58. App icon and short description

Control/Tool	Option/Range	Description
Counter		Configure the Pulse Counters for analysis of pulse trains on the digital signal inputs.

The Pulse Counter tab shown in Figure 4.32 consists of four side-tabs, one for each Counter module. The Enable button and the Mode selector are the main controls that determine if and how a Counter unit generates an output. The output is displayed in the Value field and is available in the Plotter and Data Acquisition tab.

Figure 4.32. LabOne UI: Counter tab

The counter Input signal is selectable among the Trigger inputs as well as any of the 32 DIO channels on the VHDCI connector on the instrument rear panel. The trigger level of the analog trigger inputs is configurable in the DIO tab. The following operation modes are available.

- Free running: the counter is active during repeated periods defined by the a configurable timer. The timer period is controlled by the Period field. At the beginning of the period the counter is reset, and at the end, the accumulated number of counts is output.

- Gated: the counter is controlled with the Gate Input signal. The counter is enabled at the rising edge of the Gate Input signal and disabled at the falling edge. Pulses are counted as long as the counter is enabled. The accumulated number of counts is output on the falling edge of the Gate Input signal.

- Gated free running: the counter runs on a repetitive time base defined by the Period field. The Gate Input signal controls when the counter is allowed to run. The counter as well as the timer is reset when the Gate Input signal is low. The counter will only deliver new values if the Gate Input signal is high for a time longer than the configured Period.

- Time tagging: every single event is counted and transmitted to the server along with a time tag. The Period defines the minimum hold-off time between the tagging of two subsequent pulses. If more than one pulse occurs within the window defined by the Period, then the pulses are accumulated and output at the end of the window. The Period effectively determines the maximum rate at which pulse information can be transmitted to the host PC.

Background subtraction or summation of data from two counter modules is controlled by the Operation field. For add and subtract operations, counter units 1 is grouped with unit 2, and unit 3 is grouped with unit 4. The Pulse Counter supports integration of counter data over time. The integer timestamp in all recorded data is in units of the instrument data clock, 1.8 GHz.

Note

It is recommended to use the 1GbE interface rather than the USB interface in combination with the Pulse Counter. 1GbE provides a higher stability at high data rates, namely in time tagging mode or in free-running mode with a small Period setting.

4.15.3. Functional Elements

Table 4.59. Pulse Counter tab

Control/Tool	Option/Range	Description
Enable	ON / OFF	Enable the pulse counter unit.
Mode		Select the run mode of the counter unit.
	Free Running	The counter runs on a repetitive time base defined by the Period field. At the beginning of each period the counter is reset, and at the end, the accumulated number of counts is output.
	Gated Free Running	The counter runs on a repetitive time base defined by the Period field. The Gate Input signal controls when the unit counter is allowed to run. The counter as well as the timer is reset when the Gate Input signal is low. The counter will only deliver new

Control/Tool	Option/Range	Description
		values if the Gate Input signal is high for a time longer than the configured Period.
	Gated	The counter is controlled with the Gate Input signal. The counter is enabled at the rising edge of the Gate Input signal and disabled at the falling edge. Pulses are counted as long as the counter is enabled. The accumulated number of counts is output on the falling edge of the Gate Input signal.
	Time Tagging	Every pulse is detected individually and tagged with the time of the event. The Period defines the minimum hold-off time between the tagging of two subsequent pulses. If more than one pulse occurs within the window defined by the Period, then the pulses are accumulated and output at the end of the window. The Period effectively determines the maximum rate at which pulse information can be transmitted to the host PC.
Period	93.3 ns to 1 s	Set the period used for the Free Running and Gated Free Running modes. Also sets the hold-off time for the Time Tagging mode.
Input	Ref/Trigger Input 1/2, Trigger Input 3/4, DIO Bit 0-31	Select the counter signal source.
Edge Rise	ON / OFF	Performs a trigger event when the source signal crosses the trigger level from low to high. For dual edge triggering, select also the falling edge.
Edge Fall	ON / OFF	Performs a trigger event when the source signal crosses the trigger level from high to low. For dual edge triggering, select also the rising edge.
Gate Input	Ref/Trigger Input 1/2, Trigger Input 3/4, AWG internal Trigger 1-4	Select the signal source used for enabling the counter in the Gated Free Running and Gated modes.
Operation	None	Select the arithmetic operation (addition, subtraction) applied to the
	Add Other Counter	

Control/Tool	Option/Range	Description
	Subtract Other Counter	counter unit outputs. "Other counter" refers to the grouping of the counter units: 1 with 2, and 3 with 4.
Integrate	ON / OFF	Sum up counter values over time.
Value		Displays the counter output value.

4.16. Multi Device Sync Tab

The Multi Device Sync (MDS) tab gives access to AWG output synchronization across multiple HDAWG and to the automatic timing synchronization of pulse counter and other measurement data from multiple Zurich Instruments devices (HDAWG as well as UHF series). This tab is available on all HDAWG instruments.

4.16.1. Features

- Automatic timing synchronization across instruments
- Periodic check of synchronization
- Selectable instrument subgroup
- Status display

4.16.2. 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 4.60. App icon and short description

Control/Tool	Option/Range	Description
MDS		Synchronize multiple instruments.

The Multi Device Sync tab shown in Figure 4.33 consists of the Available Devices section, a Status section, and a wiring diagram.
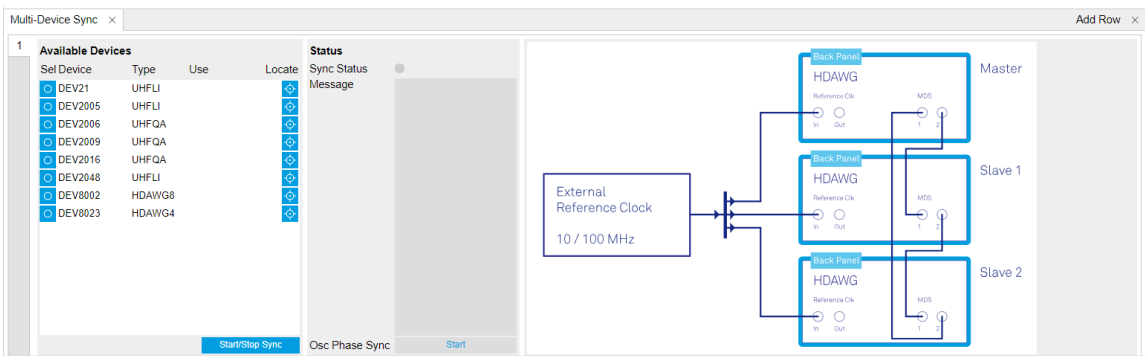
Figure 4.33. LabOne UI: Multi Device Sync tab

The Multi Device Synchronization feature provides an automated functionality to remove the clock offset of separate HDAWG instruments and align their output signals in time. Once synchronized, multiple instruments can be programmed with a single sequence program by simply extending the syntax of the LabOne AWG Sequencer language to higher channel numbers. This is described in Section 4.14. Additionally, synchronization enables a correct simultaneous display of measurement data across instruments in the Plotter tab.

The first prerequisite for automatic synchronization is that all instruments are connected to the same LabOne Data Server (see Section 1.5.1). To make these connections, click on

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 4.34.

Figure 4.34. 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 4.35.
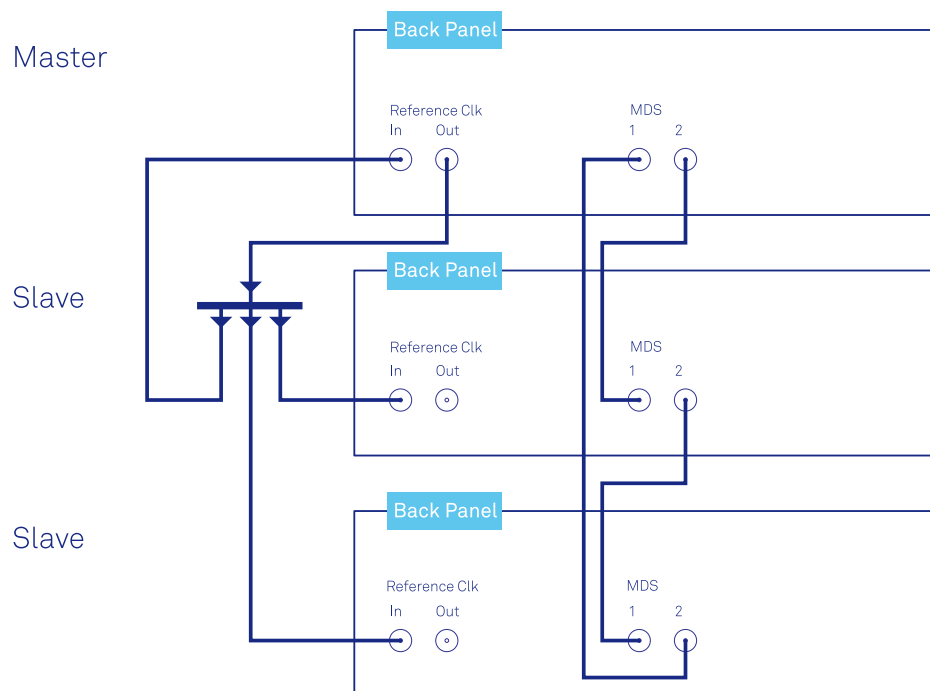
Figure 4.35. Cabling for automatic synchronization of multiple HDAWG instruments

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 (Master, Slave 1, Slave 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.

4.16.3. Functional Elements

Table 4.61. 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

4.17. 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 4.62. App Icon and short description

Control/Tool	Option/Range	Description
ZI Labs		Experimental settings and controls.

4.18. 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.

Chapter 5. Specifications

Important

Unless otherwise stated, all specifications apply after 30 minutes of instrument warm-up.

Important

Important changes in the specification parameters are explicitly mentioned in the revision history of this document.

5.1. General Specifications

Table 5.1. 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	-	-	250 W
operating environment	IEC61010, indoor location, installation category II, pollution degree 2		
operating altitude	up to 2000 meters		
power inlet fuses	250 V, 3 A, fast acting, 5 x 20 mm		
power supply AC line	100-240 V (±10%), 50/60 Hz		
dimensions including bumper	23.2 × 10.2 × 43.0 cm, 9.2 × 4.0 ×16.9 inch, 19 inch rack compatible		
weight	4.6 kg		
recommended calibration interval	2 years		

Table 5.2. Maximum ratings

Parameter	min	typ	max
damage threshold Wave Out (direct)	−1.2 V	-	+1.2 V
damage threshold Wave Out (amplified)	−6 V	-	+6 V
damage threshold Mark Out	−0.7 V	-	+4 V
damage threshold Trig In (1 kΩ input impedance)	−11 V	-	+11 V
damage threshold Trig In (50 Ω input impedance)	−6 V	-	+6 V
damage threshold Reference Clk In (DC)	−4 V	-	+4 V
damage threshold Reference Clk In (AC, with DC offset 0 V)	-	-	+13.5 dBm
damage threshold Reference Clk Out (DC)	−4 V	-	+4 V
damage threshold Sample Clk In (DC)	−4 V	-	+4 V
damage threshold Sample Clk In (AC, with DC offset 0 V)	-	-	+13.5 dBm
damage threshold Sample Clk Out (DC)	−4 V	-	+4 V
MDS In / Out	−0.7 V	-	+4 V
DIO In / Out in default configuration 3.3 V CMOS/TTL	−0.7 V	-	+4 V

Table 5.3. Host computer requirements

Parameter	Description
supported Windows operating systems	32-bit and 64-bit versions of Windows 10, 8.1, 7, XP[1]
supported Linux distributions	Ubuntu 16.04 (AMD64), 14.04 LTS (AMD64, i386)
minimum host computer requirements	Windows XP 32-bit Dual Core CPU with SSE2 support 4 GB DRAM 1 Gbit/s Ethernet controller
recommended host computer requirements	Windows 10 64-bit or Linux 64-bit Quad Core CPU (i7) or better 8 GB DRAM or better 1 Gbit/s Ethernet controller SSD HD drive (for high-bandwidth data saving) USB 3.0 connection
supported processors (requiring SSE2)	AMD K8 (Athlon 64, Sempron 64, Turion 64, etc.), AMD Phenom, Intel Pentium 4, Xeon Celeron, Celeron D, Pentium M, Celeron M, Core, Core 2, Core i5, Core i7, Atom

[1]Software version is available for download but not officially supported.

5.2. Analog Interface Specifications

Table 5.4. Wave Signal Outputs

Parameter	Details	min	typ	max
connectors	-	SMA, front panel single-ended		
output impedance	-	-	50 Ω	-
output coupling		DC		
output modes		amplified, direct		
output range	amplified, into 50 Ω	±0.1 V	-	±2.5 V
	amplified, into 50 Ω, instruments with max. range 3 V as indicated on front panel	±0.1 V	-	±1.5 V
	direct, into 50 Ω		±0.4 V	
output level accuracy	amplified, into 50 Ω	-	±(1% of setting + 5 mVpp)	-
output level resolution		-	<0.1 mV	-
output offset voltage	amplified, into 50 Ω (double into high impedance)	−1.25 V	-	+1.25 V
	direct, into 50 Ω	-	0 V	-
output offset voltage accuracy	amplified, into 50 Ω (double into high impedance)	-	±(1% of setting + 5 mV)	-
D/A converter vertical resolution	-	16 bit		
output phase noise	amplified, 1 Vpp, 100 MHz, offset 10 kHz	-	−135 dBc/Hz	-
	amplified, 1 Vpp, 100 MHz, offset 1 MHz	-	−148 dBc/Hz	-
	direct, 0.5 Vpp, 100 MHz, offset 10 kHz	-	−135 dBc/Hz	-
	direct, 0.5 Vpp, 100 MHz, offset 1 MHz	-	−148 dBc/Hz	-
output voltage noise density	amplified, 5 V range, > 200 kHz, into high impedance	-	35 nV/√Hz	-
	direct, > 200 kHz, into high impedance	-	12 nV/√Hz	-
output RMS voltage noise	amplified, 5 V range, integrated from 100 Hz to 600 MHz, into 50 Ω	-	320 µVrms	-

Parameter	Details	min	typ	max
	direct, integrated from 100 Hz to 600 MHz, into 50 Ω	-	100 μVrms	-
output spurious free dynamic range (excluding harmonics)	amplified, 1 Vpp carrier, 100 MHz, range 1 MHz to 1.2 GHz	-	80 dBc	-
	direct, 0.8 Vpp carrier, 100 MHz, range 1 MHz to 1.2 GHz	-	80 dBc	-
output worst harmonic component (carrier frequency <100 MHz)	amplified, 0.8 V range, 0.2 Vpp carrier	-	−65 dBc (HD2)	-
	amplified, 0.8 V range, 0.8 Vpp carrier	-	−56 dBc (HD3)	-
	amplified, 1 V range, 1 Vpp carrier	-	−65 dBc (HD2)	-
	amplified, 2 V range, 2 Vpp carrier	-	−57 dBc (HD3)	-
	amplified, 3 V range, 3 Vpp carrier	-	−53 dBc (HD3)	-
	amplified, 4 V range, 4 Vpp carrier	-	−50 dBc (HD3)	-
	amplified, 5 V range, 5 Vpp carrier	-	−47 dBc (HD3)	-
	direct, 0.5 Vpp carrier	-	−59 dBc (HD2)	-
	direct, 0.8 Vpp carrier	-	−53 dBc (HD2)	-
output 2nd harmonic HD2 (carrier frequency <100 MHz)	amplified, 1 V range, 1 Vpp carrier	-	−65 dBc	-
	amplified, 5 V range, 5 Vpp carrier	-	−65 dBc	-
	direct, 0.8 V range, 0.5 Vpp carrier	-	−59 dBc	-
output 3rd harmonic HD3 (carrier frequency <100 MHz)	amplified, 1 V range, 1 Vpp carrier	-	−65 dBc	-
	amplified, 5 V range, 5 Vpp carrier	-	−47 dBc	-
	direct, 0.8 V range, 0.5 Vpp carrier	-	−73 dBc	-

Table 5.5. Time & Frequency Domain Characteristics

Parameter	Details	min	typ	max
Wave output bandwidth (−3dB) corrected for sin(x)/x roll-off	amplified, 5.0 Vpp in 50 Ω	-	300 MHz	-
	direct, 0.8 Vpp 50 Ω	-	750 MHz	-
Wave output bandwidth (−3dB) corrected for sin(x)/x roll-off (instruments with max. range 3 V as indicated on front panel)	amplified, 3.0 Vpp in 50 Ω	-	300 MHz	-
	direct, 0.8 Vpp 50 Ω	-	750 MHz	-
D/A converter sampling clock source	-	internal, external		
D/A converter sampling rate	base sampling clock	100 MSa/s	-	2.4 GSa/s
	clock rate division (powers of 2)	2^0	-	2^{13}
internal sampling clock frequency resolution	-	-	7 digits	-
Wave output rise time 20% to 80%	amplified, 0.8 V range, 0.2 V step	-	450 ps	-
	amplified, 5 V range, 1 V step		800 ps	
	amplified, 5 V range, 5 V step	-	1100 ps	-
	direct, 0.8 V step	-	300 ps	-
Wave output rise time 20% to 80% (instruments with max. range 3 V as indicated on front panel)	1 V step, amplified, 3 V range	-	550 ps	-
	0.8 V step, direct	-	300 ps	-
Wave output overshoot	-	-	<1%	-
switchable output filter bandwidth (−3dB)	amplified output path	-	80 MHz	-
trigger delay to output	from Trig input to Wave output within one channel pair (1&2, 3&4, 5&6, 7&8), using playWaveDigTrigger sequencer instruction	-	50 ns	-
	from any Trig input to any Wave output, using waitDigTrigger sequencer instruction	-	-	180 ns
skew between outputs	between any two Wave outputs in the same output configuration	-	200 ps	-
skew adjustment range	sampling frequency 2.4 GHz	-	8 ns	-

Parameter	Details	min	typ	max
skew adjustment resolution	with HDAWG-SKW option	-	10 ps	-
	without HDAWG-SKW option, sampling frequency 2.4 GHz	-	<0.42 ns (1 sample clock period)	-
Wave output period jitter	amplified, 150 MHz square wave	-	3 ps RMS	-
		-	20 ps Pk-Pk	-
	direct, 150 MHz square wave	-	3 ps RMS	-
		-	20 ps Pk-Pk	-

Table 5.6. Arbitrary Waveform Generator

Parameter	Details	min	typ	max
Waveform memory per channel	without HDAWG-ME option	-	64 MSa	-
	with HDAWG-ME option	-	500 MSa	-
sequence length	-	-	-	16,384
waveform granularity	-	-	16 samples	-
minimum waveform length	-	-	32 samples	-
sequencer clock frequency	-	sampling rate divided by 8		

Table 5.7. Marker & Other Outputs

Parameter	Details	min	typ	max
marker outputs	-	2 per channel, SMA output on front panel,[3] 2 marker bits per waveform		
marker output high voltage	-	-	3.3 V	-
marker output low voltage	-	-	0 V	-
marker output impedance	-	-	50 Ω	-
marker output rise time 20% to 80%	-	-	300 ps	-
marker output period jitter	square wave, 100 MHz	-	60 ps p-p	-
marker output skew control range	-	−23 ns	-	+30 ns
marker output skew control resolution	sampling rate 2.4 GHz, depends on absolute setting	-	10 ps	-
sampling clock output	-	SMA on back panel		
sampling clock output amplitude	2.4 GHz into 50 Ω	-	0.8 Vpp	-
	1.0 GHz into 50 Ω	-	2 Vpp	-

Parameter	Details	min	typ	max
reference clock output	-	SMA on back panel		
reference clock output impedance		50 Ω, AC coupled		
reference clock output amplitude	100 MHz into 50 Ω	-	2 Vpp	-
reference clock output frequency	internal reference mode	100 MHz		
	external reference mode	10 / 100 MHz		
reference clock output jitter	derived from integrated phase noise measurement (12 kHz to 200 MHz offset frequency)	-	260 fs RMS	-

[3]Up to 32 additional digital output signals can be generated on the DIO connector on the back panel using the playWaveDIO real-time sequencer instruction.

Table 5.8. Inputs

Parameter	Details	min	typ	max
trigger inputs	-	1 per channel, 1 SMA on front panel		
trigger input impedance	-	50 Ω / 1 kΩ		
trigger input voltage range	50 Ω impedance	−5 V	-	5 V
	1 kΩ impedance	−10 V	-	10 V
trigger input threshold range	50 Ω impedance	−5 V	-	5 V
	1 kΩ impedance	−10 V	-	10 V
trigger input threshold resolution	-	-	< 0.4 mV	-
trigger input threshold hysteresis	-	-	> 60 mV	-
trigger input min. pulse width	-	-	5 ns	-
trigger input max. operating frequency	-	-	300 MHz	-
sampling clock input	-	SMA on back panel		
reference clock input	-	SMA on back panel		
reference clock input impedance	-	50 Ω, AC coupled		
reference clock input frequency	-	10 / 100 MHz		
reference clock input amplitude	-	−4 dBm	-	+13 dBm

Table 5.9. Oscillator and Clocks

Parameter	Details	min	typ	max
internal clock type	-	TCXO		
internal clock long term accuracy / aging	-	-	-	±0.8 ppm/year
internal clock short term stability (1 s)	-	-	-	0.0001 ppm
internal clock initial accuracy	-	-	-	±1 ppm

Parameter	Details	min	typ	max
internal clock temperature stability	−20°C to 70°C	-	-	±0.3 ppm
internal clock phase noise	offset 100 Hz	-	-	−105 dBc/Hz
	offset 1 kHz	-	-	−125 dBc/Hz

5.3. Digital Interface Specifications

Table 5.10. Digital Interfaces

Parameter	Description
host computer connection	USB 3.0, 1.6 Gbit/s
	1GbE, LAN / Ethernet, 1 Gbit/s
DIO port	4 x 8 bit, general purpose digital input/output port, 3.3 V TTL specification
ZSync peripheral port	connector for ZI proprietary bus to communicate with external peripherals

5.3.1. DIO Connector

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 interface standard is switchable between LVDS (low-voltage differential signalling) and LVCMOS/LVTTL. The DIO port features 32 user-controlled bits that can all be configured byte-wise as inputs or outputs in LVCMOS/LVTTL mode, whereas in LVDS mode, half of the bits are always configured as inputs.
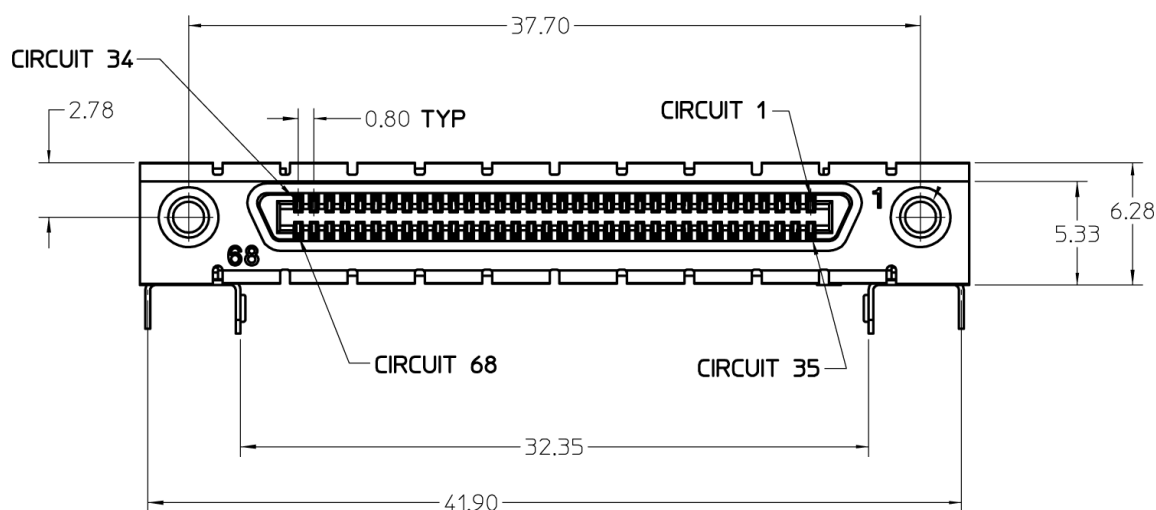
Figure 5.1. DIO HD 68 pin connector

Table 5.11. Electrical Specifications

Parameter	Details	Min	Typ	Max
supported DIO interface standards		LVCMOS/LVTTL (single-ended, 3.3 V); LVDS (differential)		
high-level input voltage VIH	LVCMOS/LVTTL	2.0 V	-	-
low-level input voltage VIL	LVCMOS/LVTTL	-	-	0.8 V
high-level output voltage VOH	LVCMOS/LVTTL at IOH < 12 mA	2.6 V	-	-
low-level output voltage VOL	LVCMOS/LVTTL at IOL < 12 mA	-	-	0.4 V

Parameter	Details	Min	Typ	Max
high-level output current IOH (sourcing)	LVCMOS/LVTTL	-	-	12 mA
low-level output current IOL (sinking)	LVCMOS/LVTTL	-	-	12 mA
input differential voltage VID	LVDS	100 mV	-	600 mV
input common-mode voltage VICM	LVDS	0.3 V	-	2.35 V
output differential voltage VOD	LVDS	247 mV	-	454 mV
output common-mode voltage VOCM	LVDS	1.125 V	-	1.375 V

Table 5.12. Pin Assignment in LVCMOS/LVTTL Mode

Pin	Name	Description
68	CLKI	digital input
67	unused	leave unconnected
66 .. 59	DIO[31:24]	digital input or output byte (set by user)
58 .. 51	DIO[23:16]	digital input or output byte (set by user)
50 .. 43	DIO[15:8]	digital input or output byte (set by user)
42 .. 35	DIO[7:0]	digital input or output byte (set by user)
34	GND	digital ground
33	unused	leave unconnected
32 .. 1	GND	digital ground

Table 5.13. Pin Assignment in LVDS Mode

Pin	Name	Description
68	CLKI+	digital input
67	unused	leave unconnected
66 .. 59	DI+[31:24]	digital input byte
58 .. 51	DI+[23:16]	digital input byte
50 .. 43	DIO+[15:8]	digital input or output byte (set by user)
42 .. 35	DIO+[7:0]	digital input or output byte (set by user)
34	CLKI−	digital input
33	unused	leave unconnected
32 .. 25	DI−[31:24]	digital input byte
24 .. 17	DI−[23:16]	digital input byte
16 .. 9	DIO−[15:8]	digital input or output byte (set by user)
8 .. 1	DIO−[7:0]	digital input or output byte (set by user)

5.3.2. DIOLink Interface

Introduction

The DIOLink provides a digital interface to Zurich Instruments HDAWG and UHFQA instruments, the main constituents of the Zurich Instruments Quantum Computing Control System. 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.
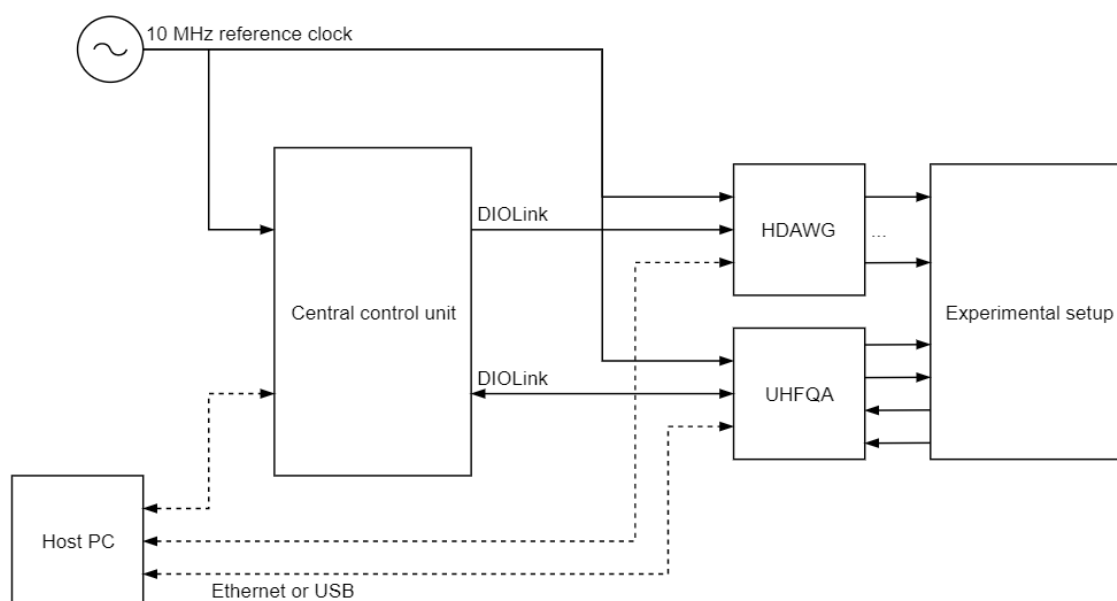
Figure 5.2. DIO Link connections between HDAWG, UHFQA, and central control unit in an experimental setup

Signal Protocol

The DIOLink interface makes use of the signals shown in the following table. The signals are all transmitted from the master to the slave (i.e. 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 5.14. 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.
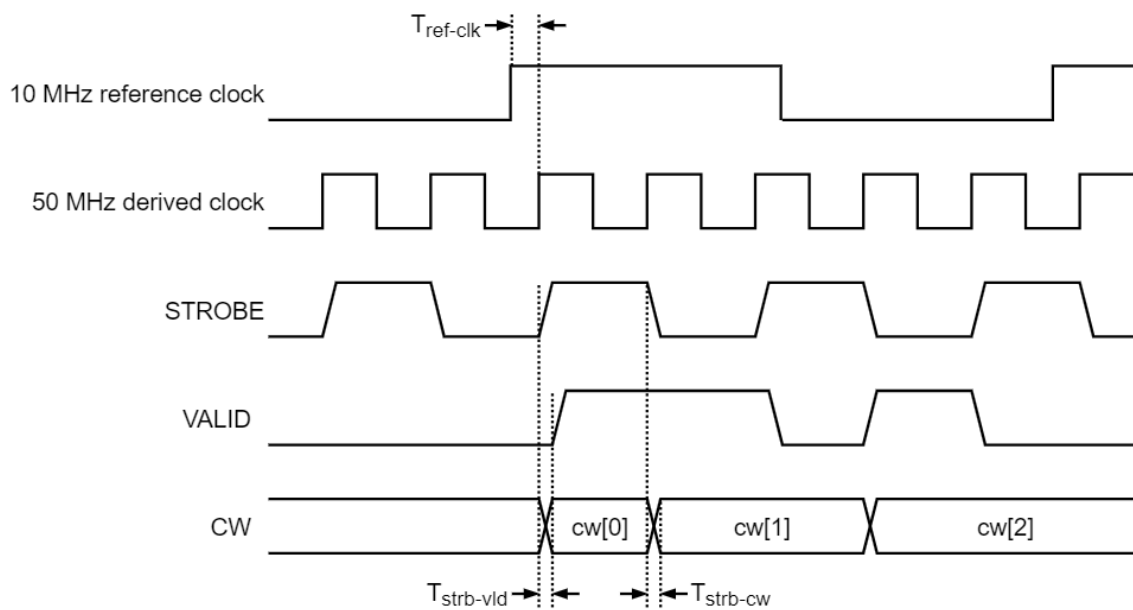
Figure 5.3. Timing diagram of the signaling protocol on the DIOLink interface for the maximum codeword rate of 50 MHz. The TOGGLE signal is always active. The VALID and CW signals are only used when actual codewords are to be transmitted over the DIOLink. It is allowed to transmit codewords back-to-back.

The timing on the interface is described in more detail in the following table.

Table 5.15. DIOLink signals

Name	Range	Description
$T_{ref-clk}$	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).
$T_{strb-vld}$	± 3.3 ns	Delay, or skew, between the STROBE and the VALID signals.
$T_{strb-cw}$	± 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 5.16. 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

5.4. Performance Diagrams

Figure 5.4 shows a typical SSB phase noise measured at the signal output. For this measurement, the HDAWG was connected to a phase noise analyzer and the signal output amplitude was set to 3 V. The phase noise at 10 MHz at 1 kHz offset is around -148 dBc/Hz. The phase noise for a 100 MHz sine wave at 1 kHz offset is around -127 dBc/Hz.
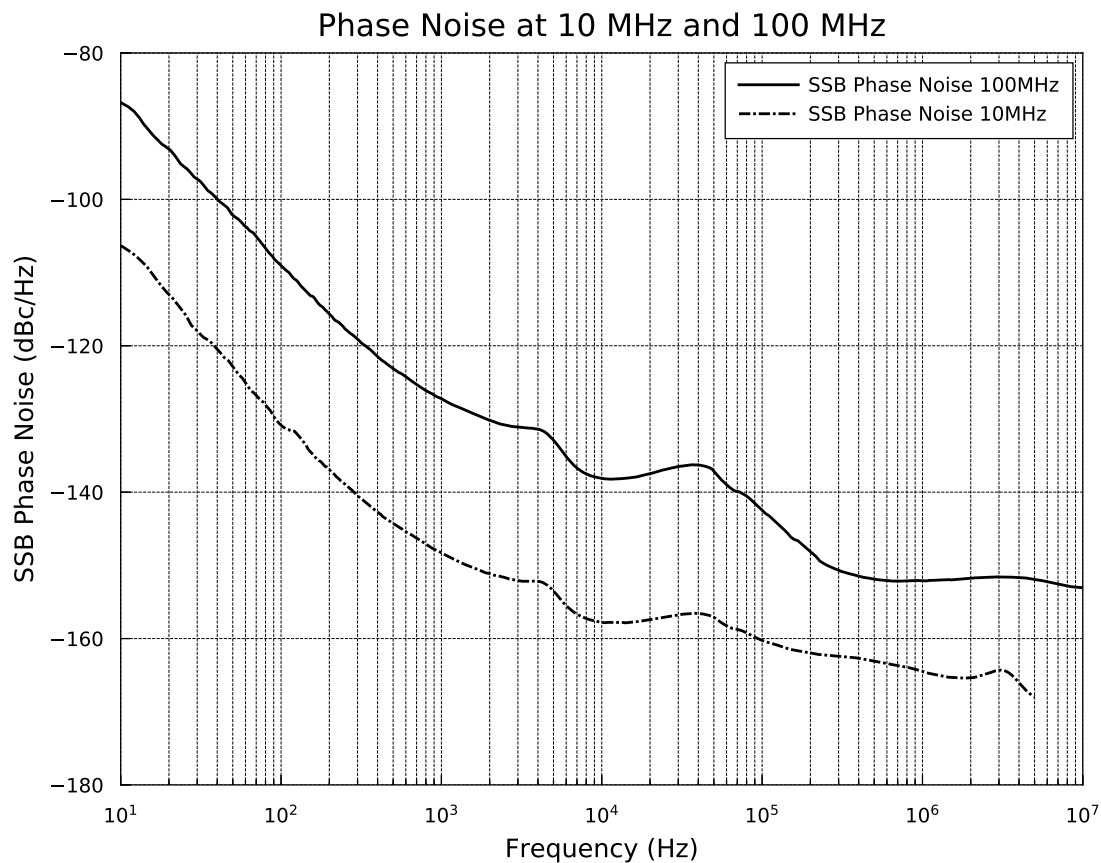
Figure 5.4. HDAWG signal output phase noise

Chapter 6. Device Node Tree

This chapter contains reference documentation for the settings and measurement data available on HDAWG Instruments. Whilst Chapter 4 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 organised 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:

- Section 6.1 for an introduction of how the instrument's settings and measurement data are organised hierarchically in the Data Server's so-called "Node Tree".

- Section 6.2 for a reference list of the settings and measurement data available on HDAWG Instruments, organized by branch in the Node Tree.

6.1. Introduction

This section 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 Section 1.5.1 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, it's device ID becomes a top-level branch in the Data Server's node tree. The features of the instrument are organised 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:

/DEV2006/AUXOUTS/

In turn, each individual auxiliary output channel has it's own branch underneath the "AUXOUTS" branch.

/DEV2006/AUXOUTS/0/
/DEV2006/AUXOUTS/1/
/DEV2006/AUXOUTS/2/
/DEV2006/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:

/DEV2006/AUXOUTS/0/DEMODSELECT
/DEV2006/AUXOUTS/0/LIMITLOWER
/DEV2006/AUXOUTS/0/LIMITUPPER
/DEV2006/AUXOUTS/0/OFFSET
/DEV2006/AUXOUTS/0/OUTPUTSELECT
/DEV2006/AUXOUTS/0/PREOFFSET
/DEV2006/AUXOUTS/0/SCALE
/DEV2006/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 Section 6.2. The different properties and types are explained in Section 6.1.1.

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 Section 6.1.2.

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.

6.1.1. 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   A node with write attribute corresponding to an instrument configuration. The data in these nodes will be saved to and loaded from 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 node may contain data of the following types:

Integer   Integer data.

Double   Double precision floating point data.

String   A string array.

Enumerated (integer)   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 the C Programming chapter in the LabOne Programming Manual.

6.1.2. 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 6.1 shows how the equivalent Matlab command is displayed after modifying the value of the auxilliary 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.
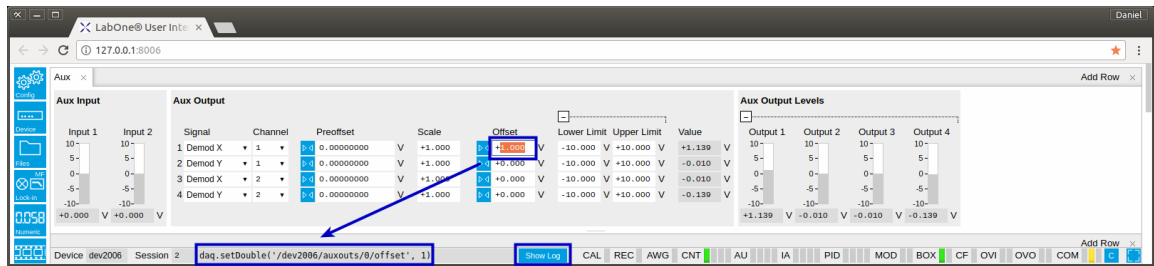
Figure 6.1. When a device's configuration is modified in the LabOne User Interface, the Status Bar displays the equivalent command to perform the same configuration via a LabOne programming interface. Here, the Matlab code to modify auxiliary output 1's offset value is provided. When "Show Log" is clicked 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 Section 6.2 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.ziPython.ziDAQServer('localhost', 8004, 6)
daq.help('/dev2006/auxouts/0/offset')
# Out:
# /DEV2006/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
```

6.1.3. 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 particualar use to programmers are:

- /ZI/DEBUG/LOGPATH - the location of the Data Server's log in the PC's filesystem,

- /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.

For documentation of all Data Server nodes see Section 6.2.

6.2. Reference Node Documentation

This section describes all the nodes in the data server's node tree organised by branch. See Section 6.1.1 for an explanation of node properties and types.

6.2.1. AWGS

/DEV..../AWGS/n/ENABLE

Properties: Read, Write, Setting	Type: Integer (64 bit)	Unit: None

Activates the AWG.

/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/SINGLE

Properties: Read, Write, Setting	Type: Integer (64 bit)	Unit: None

Puts the AWG into single shot mode.

/DEV..../AWGS/n/TIME

Properties: Read, Write, Setting	Type: Integer (enumerated)	Unit: None

AWG sampling rate. The numeric values here are an example when the base sample rate is the default value of 2.4 GHz and are rounded for display purposes. The exact values are equal to the base sampling rate divided by 2^n, where n is the node value. The base sample clock is the node /DEV.../SYSTEM/CLOCKS/SAMPLECLOCK/FREQ. This value is used by default and can be overridden in the Sequence program.

Allowed Values:

0	2.4 GHz	7	18.75 MHz
1	1.2 GHz	8	9.38 MHz
2	600 MHz	9	4.69 MHz
3	300 MHz	10	2.34 MHz
4	150 MHz	11	1.17 MHz
5	75 MHz	12	585.94 kHz
6	37.50 MHz	13	292.97 kHz

/DEV..../AWGS/n/COMMANDTABLE/CLEAR

Properties: Read, Write, Setting	Type: Integer (64 bit)	Unit: None

[empty]

/DEV..../AWGS/n/COMMANDTABLE/DATA

Properties: Read, Write   Type: ZIVectorData   Unit: None

[empty]

/DEV..../AWGS/n/COMMANDTABLE/STATUS

Properties: Read   Type: Integer (64 bit)   Unit: None

[empty]

/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/HIGHBITS

Properties: Read, Write, Setting   Type: Integer (64 bit)   Unit: None

32-bit value indicating which bits on the 32-bit interface are detected as having a logic high value.

/DEV..../AWGS/n/DIO/LOWBITS

Properties: Read, Write, Setting   Type: Integer (64 bit)   Unit: None

32-bit value indicating which bits on the 32-bit interface are detected as having a logic low value.

/DEV..../AWGS/n/DIO/STATE

Properties: Read, Write, Setting   Type: Integer (64 bit)   Unit: None

When asserted, indicates that triggers are generated from the DIO interface to the AWG.

/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: Read, Write   Type: ZIVectorData   Unit: None

Contains the data of the uploaded ELF file.

/DEV..../AWGS/n/ELF/LENGTH

Properties: Read, Write   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/SEQUENCER/ASSEMBLY

Properties: Read   Type: ZIVectorData   Unit: None

Displays the current sequence program in compiled form. Every line corresponds to one hardware instruction and requires one clock cycle (3.33 ns) for execution.

/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 sequence program relative to the device cache memory. The cache memory provides space for 8192 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) waveform playback to finish, b) prefetch to finish, or c) trigger to arrive; Bit 3: AWG has detected an error

/DEV..../AWGS/n/SEQUENCER/TRIGGERED

Properties: Read   Type: Integer (64 bit)   Unit: None

When 1, indicates that the AWG Sequencer has been triggered.

/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 512 kSa of waveform data. Memory Usage over 100% means that waveforms must be loaded from the main memory (64 or 500 MSa per channel) during playback, which can lead to delays.

/DEV..../AWGS/n/WAVEFORM/PLAYING

Properties: Read   Type: Integer (64 bit)   Unit: None

When 1, indicates if a waveform is being played currently.

/DEV..../AWGS/n/AUXTRIGGERS/n/CHANNEL

Properties: Read, Write, Setting   Type: Integer (enumerated)   Unit: None

Selects the digital trigger source signal.

Allowed Values:

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.

Allowed Values:

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/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 150 MHz clock cycles. Valid values are 0 to 3.

/DEV..../AWGS/n/DIO/ERROR/TIMING

Properties: Read, Write, Setting　　　　Type: Integer (64 bit)　　　Unit: None

A 32-bit value indicating which bits on the DIO interface may have timing errors. A timing error is defined as an event where either the VALID or any of the data bits on the DIO interface change value at the same time as the STROBE bit.

/DEV..../AWGS/n/DIO/ERROR/WIDTH

Properties: Read, Write, Setting　　　　Type: Integer (64 bit)　　　Unit: None

Indicates a width (i.e. jitter) error on either the STROBE (bit 0 of the value) or VALID bit (bit 1 of the result). A width error indicates that there was jitter detected on the given bit, meaning that an active period was either shorter or longer than the configured expected width.

/DEV..../AWGS/n/DIO/MASK/SHIFT

Properties: Read, Write, Setting　　　　Type: Integer (64 bit)　　　Unit: None

Defines the amount of bit shifting to apply for the DIO wave selection in connection with playWaveDIO().

/DEV..../AWGS/n/DIO/MASK/VALUE

Properties: Read, Write, Setting　　　　Type: Integer (64 bit)　　　Unit: None

Selects the DIO bits to be used for waveform selection in connection with playWaveDIO().

/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.

Allowed Values:

0	"off": Off	2	"falling_edge": Falling edge trigger
1	"rising_edge": Rising edge trigger	3	"both_edges": Rising or falling edge trigger

/DEV..../AWGS/n/DIO/STROBE/WIDTH

Properties: Read, Write, Setting   Type: Integer (64 bit)   Unit: None

Specifies the expected width of active pulses on the STROBE bit.

/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.

Allowed Values:

0	"none": None: VALID bit is ignored.	2	"high": High: VALID bit must be logical high.
1	"low": Low: VALID bit must be logical zero.	3	"both": Both: VALID bit may be logical high or zero.

/DEV..../AWGS/n/DIO/VALID/WIDTH

Properties: Read, Write, Setting   Type: Integer (64 bit)   Unit: None

Expected width of an active pulse on the VALID bit.

/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

Enables the driving of the given AWG output channel.

/DEV..../AWGS/n/OUTPUTS/n/GAINS/n

 Properties: Read, Write, Setting   Type: Double   Unit: None

Gain factor applied to the AWG Output at the given output multiplier stage. The final signal amplitude is proportional to the Range voltage setting of the Wave signal outputs.

/DEV..../AWGS/n/OUTPUTS/n/HOLD

 Properties: Read, Write, Setting   Type: Integer (64 bit)   Unit: None

Keep the last sample (constant) on the output even after the waveform program finishes.

/DEV..../AWGS/n/OUTPUTS/n/MODULATION/CARRIERS/n/FREQ

 Properties: Read   Type: Double   Unit: Hz

Indicates the frequency used for this carrier. The frequency is calculated with oscillator frequency times the harmonic factor.

/DEV..../AWGS/n/OUTPUTS/n/MODULATION/CARRIERS/n/HARMONIC

 Properties: Read, Write, Setting   Type: Integer (64 bit)   Unit: None

Multiplies the carriers's reference frequency with the integer factor defined by this field.

/DEV..../AWGS/n/OUTPUTS/n/MODULATION/CARRIERS/n/OSCSELECT

 Properties: Read, Write, Setting   Type: Integer (64 bit)   Unit: None

Select oscillator for generation of this carrier.

/DEV..../AWGS/n/OUTPUTS/n/MODULATION/CARRIERS/n/PHASESHIFT

 Properties: Read, Write, Setting   Type: Double   Unit: deg

Phase shift applied to carrier signal.

/DEV..../AWGS/n/OUTPUTS/n/MODULATION/MODE

Properties: Read, Write, Setting   Type: Integer (enumerated)   Unit: None

Select modulation mode between off, sine modulation and advanced.

Allowed Values:

0   "off": Modulation Off: AWG Output goes directly to Signal Output.

1   "sine00": Sine 11: AWG Outputs 0 and 1 are both multiplied with Sine Generator signal 0.

2   "sine11": Sine 22: AWG Outputs 0 and 1 are both multiplied with Sine Generator signal 1.

3   "sine10": Sine 21: AWG Outputs 0 and 1 are multiplied with Sine Generator signal 1 and 0, respectively.

5   "advanced": Advanced: Output modulates corresponding sines from modulation carriers.

/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/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.

6.2.2. CLOCKBASE

/DEV..../CLOCKBASE

Properties: Read   Type: Double   Unit: Hz

Returns the internal clock frequency of the device.

6.2.3. CNTS (Pulse Counters)

/DEV..../CNTS/n/ENABLE

Properties: Read, Write, Setting   Type: Integer (64 bit)   Unit: None

Enable the pulse counter unit.

/DEV..../CNTS/n/GATESELECT

Properties: Read, Write, Setting   Type: Integer (enumerated)   Unit: None

Select the signal source used for enabling the counter in the Gated Free Running and Gated modes.

Allowed Values:

0	"trigin0", "trigger_input0": Trigger/Ref Input 1 (front panel).	4	"awg_trigger0": AWG Trigger 1.
1	"trigin1", "trigger_input1": Trigger/Ref Input 2 (front panel).	5	"awg_trigger1": AWG Trigger 2.
2	"trigin2", "trigger_input2": Trigger Input 3 (rear panel).	6	"awg_trigger2": AWG Trigger 3.
3	"trigin3", "trigger_input3": Trigger Input 4 (rear panel).	7	"awg_trigger3": AWG Trigger 4.

/DEV..../CNTS/n/INPUTSELECT

Properties: Read, Write, Setting   Type: Integer (enumerated)   Unit: None

Select the counter signal source.

Allowed Values:

0	DIO Bit 0.	18	DIO Bit 18.
1	DIO Bit 1.	19	DIO Bit 19.
2	DIO Bit 2.	20	DIO Bit 20.
3	DIO Bit 3.	21	DIO Bit 21.
4	DIO Bit 4.	22	DIO Bit 22.
5	DIO Bit 5.	23	DIO Bit 23.
6	DIO Bit 6.	24	DIO Bit 24.
7	DIO Bit 7.	25	DIO Bit 25.
8	DIO Bit 8.	26	DIO Bit 26.
9	DIO Bit 9.	27	DIO Bit 27.
10	DIO Bit 10.	28	DIO Bit 28.
11	DIO Bit 11.	29	DIO Bit 29.
12	DIO Bit 12.	30	DIO Bit 30.
13	DIO Bit 13.	31	DIO Bit 31.

14	DIO Bit 14.
15	DIO Bit 15.
16	DIO Bit 16.
17	DIO Bit 17.

32	"trigin0", "trigger_input0": Trigger/Ref Input 1 (front panel).
33	"trigin1", "trigger_input1": Trigger/Ref Input 2 (front panel).
34	"trigin2", "trigger_input2": Trigger Input 3 (rear panel).
35	"trigin3", "trigger_input3": Trigger Input 4 (rear panel).

/DEV..../CNTS/n/INTEGRATE

Properties: Read, Write, Setting   Type: Integer (64 bit)   Unit: None

Sum up counter values over time.

/DEV..../CNTS/n/MODE

Properties: Read, Write, Setting   Type: Integer (enumerated)   Unit: None

Select the run mode of the counter unit.

Allowed Values:

1	"free_running": Free Running: The counter runs on a repetitive time base defined by the Period field. At the beginning of each period the counter is reset, and at the end, the accumulated number of counts is output.
2	"gated_free_running": Gated Free Running: The counter runs on a repetitive time base defined by the Period field. The Gate Input signal controls when the unit counter is allowed to run. The counter as well as the timer is reset when the Gate Input signal is low. The counter will only deliver new values if the Gate Input signal is high for a time longer than the configured Period.

3	"gated": Gated: The counter is controlled with the Gate Input signal. The counter is enabled at the rising edge of the Gate Input signal and disabled at the falling edge. Pulses are counted as long as the counter is enabled. The accumulated number of counts is output on the falling edge of the Gate Input signal.
4	"time_tagging": Time Tagging: Every pulse is detected individually and tagged with the time of the event. The Period defines the minimum hold-off time between the tagging of two subsequent pulses. If more than one pulse occurs within the window defined by the Period, then the pulses are accumulated and output at the end of the window. The Period effectively determines the maximum rate at which pulse information can be transmitted to the host PC.

/DEV..../CNTS/n/OPERATION

Properties: Read, Write, Setting   Type: Integer (enumerated)   Unit: None

Select the arithmetic operation (addition, subtraction) applied to the counter unit outputs. 'Other counter' refers to the grouping of the counter units: 1 with 2, and 3 with 4.

Allowed Values:

0   "none": None

1   "add_other_counter": Add Other Counter

2   "subtract_other_counter": Subtract Other Counter

/DEV..../CNTS/n/PERIOD

Properties: Read, Write, Setting   Type: Double   Unit: s

Set the period used for the Free Running and Gated Free Running modes. Also sets the hold-off time for the Time Tagging mode.

/DEV..../CNTS/n/SAMPLE

Properties: Read, Stream   Type: ZICntSample   Unit: None

Streaming node containing counter values.

/DEV..../CNTS/n/TRIGFALLING

Properties: Read, Write, Setting   Type: Integer (64 bit)   Unit: None

Performs a trigger event when the source signal crosses the trigger level from high to low. For dual edge triggering, select also the rising edge.

/DEV..../CNTS/n/TRIGRISING

Properties: Read, Write, Setting   Type: Integer (64 bit)   Unit: None

Performs a trigger event when the source signal crosses the trigger level from low to high. For dual edge triggering, select also the falling edge.

/DEV..../CNTS/n/VALUE

Properties: Read   Type: Integer (64 bit)   Unit: None

Counter output value.

6.2.4. DIOS (Digital I/O)

/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/INPUT

Properties: Read   Type: Integer (64 bit)   Unit: None

Gives the value of the DIO input for those bytes where drive is disabled.

/DEV..../DIOS/n/INTERFACE

Properties: Read, Write, Setting   Type: Integer (64 bit)   Unit: None

Selects the interface standard to use on the 32-bit DIO interface. A value of 0 means that a 3.3 V CMOS interface is used. A value of 1 means that an LVDS compatible interface is used.

/DEV..../DIOS/n/MODE

Properties: Read, Write, Setting   Type: Integer (enumerated)   Unit: None

Select DIO mode

Allowed Values:

0   "manual": Manual setting of the DIO output value.

1   "awg_sequencer_commands": Enables setting the DIO output values by AWG sequencer commands and forwards DIO input values to the AWG sequencer. The DIO interface operates at a clock frequency of 150 MHz.

2   "dio_codeword": Enables setting the DIO output values by AWG sequencer commands and forwards DIO input values to the AWG sequencer. This mode is equivalent to the mode AWG Sequencer, except for the DIO interface clock frequency which is set to 50 MHz.

3   "qccs": Enables setting the DIO output values by the ZSync input values. Forwards the ZSync input values to the AWG sequencer. Forwards the DIO input values to the ZSync output. Select this mode when the instrument is connected via ZSync to a PQSC.

/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.

6.2.5. FEATURES

/DEV..../FEATURES/CODE

Properties: Read, 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.

6.2.6. OSCS (Oscillators)

/DEV..../OSCS/n/FREQ

Properties: Read, Write, Setting   Type: Double   Unit: Hz

Frequency control for each oscillator.

/DEV..../OSCS/n/FREQAWG

Properties: Read   Type: Double   Unit: Hz

Frequency as set by the AWG sequencer.

6.2.7. SIGOUTS

/DEV..../SIGOUTS/n/BUSY

Properties: Read   Type: Integer (64 bit)   Unit: None

Boolean value indicating whether a blocking process is being executed on the device. For example, locking to the external reference clock.

/DEV..../SIGOUTS/n/DELAY

Properties: Read, Write, Setting   Type: Double   Unit: s

This value allows to delay the output of the signal in order to align waves.

/DEV..../SIGOUTS/n/DIRECT

Properties: Read, Write, Setting   Type: Integer (enumerated)   Unit: None

Enables the direct output path. If enabled the signal will be fed directly from the DAC, reducing delay and noise. However, the range will be fixed and offset is not available any more.

Allowed Values:

0   "amplified_path": Amplified Path   1   "direct_path": Dircet Path

/DEV..../SIGOUTS/n/FILTER

Properties: Read, Write, Setting   Type: Integer (64 bit)   Unit: None

Enables a filter stage in the amplified path.

/DEV..../SIGOUTS/n/MAX

Properties: Read   Type: Double   Unit: None

Maximum value transmitted to the DAC represented as a 16-bit integer in two's complement format.

/DEV..../SIGOUTS/n/MIN

Properties: Read   Type: Double   Unit: None

Minimum value transmitted to the DAC represented as a 16-bit integer in two's complement format.

/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.

/DEV..../SIGOUTS/n/PRECOMPENSATION/ENABLE

Properties: Read, Write, Setting   Type: Integer (64 bit)   Unit: None

Enables (1) or disables (0) the entire precompensation filter chain.

/DEV..../SIGOUTS/n/PRECOMPENSATION/LATENCY

Properties: Read   Type: Double   Unit: s

The total latency introduced by the entire precompensation filter chain.

/DEV..../SIGOUTS/n/PRECOMPENSATION/BOUNCES/n/AMPLITUDE

Properties: Read, Write, Setting   Type: Double   Unit: None

Sets the amplitude of the bounce correction filter relative to the signal amplitude.

/DEV..../SIGOUTS/n/PRECOMPENSATION/BOUNCES/n/DELAY

Properties: Read, Write, Setting   Type: Double   Unit: s

Sets the delay of the bounce correction filter.

/DEV..../SIGOUTS/n/PRECOMPENSATION/BOUNCES/n/ENABLE

Properties: Read, Write, Setting   Type: Integer (64 bit)   Unit: None

Enables (1) or disables (0) the bounce correction filter.

/DEV..../SIGOUTS/n/PRECOMPENSATION/BOUNCES/n/STATUS

Properties: Read   Type: Integer (64 bit)   Unit: None

Indicates the status of the bounce correction filter: 0 = normal, 1 = overflow during the last update period (~100 ms), 2 = overflowed in the past.

/DEV..../SIGOUTS/n/PRECOMPENSATION/EXPONENTIALS/n/ AMPLITUDE

Properties: Read, Write, Setting   Type: Double   Unit: None

Sets the amplitude of the exponential overshoot compensation filter relative to the signal amplitude.

/DEV..../SIGOUTS/n/PRECOMPENSATION/EXPONENTIALS/n/ ENABLE

Properties: Read, Write, Setting   Type: Integer (64 bit)   Unit: None

Enables (1) or disables (0) the exponential overshoot compensation filter.

/DEV..../SIGOUTS/n/PRECOMPENSATION/EXPONENTIALS/n/ STATUS

Properties: Read   Type: Integer (64 bit)   Unit: None

Indicates the status of the exponential overshoot compensation filter: 0 = normal, 1 = overflow during the last update period (~100 ms), 2 = overflowed in the past.

/DEV..../SIGOUTS/n/PRECOMPENSATION/EXPONENTIALS/n/ TIMECONSTANT

Properties: Read, Write, Setting   Type: Double   Unit: s

Sets the characteristic time constant of the exponential overshoot compensation filter.

/DEV..../SIGOUTS/n/PRECOMPENSATION/FIR/COEFFICIENTS

Properties: Read, Write, Setting   Type: ZIVectorData   Unit: s

Vector containing 40 coefficients of the finite impulse response (FIR) precompensation filter. The first eight coefficients correspond directly to the first eight taps of the FIR filter, while the remaining 32 coefficients are each applied to pairs of subsequent taps. In total the FIR filter kernel therefore has a length of 8 + 2*32 = 72 taps.

/DEV..../SIGOUTS/n/PRECOMPENSATION/FIR/ENABLE

Properties: Read, Write, Setting   Type: Integer (64 bit)   Unit: None

Enables (1) or disables (0) the finite impulse response (FIR) precompensation filter.

/DEV..../SIGOUTS/n/PRECOMPENSATION/FIR/STATUS

Properties: Read   Type: Integer (64 bit)   Unit: None

Indicates the status of the finite impulse response (FIR) precompensation filter: 0 = normal, 1 = overflow during the last update period (~100 ms), 2 = overflowed in the past.

/DEV..../SIGOUTS/n/PRECOMPENSATION/HIGHPASS/n/CLEARING/ SLOPE

Properties: Read, Write, Setting   Type: Integer (enumerated)   Unit: None

When to react to a clearing pulse generated after the AWG Sequencer setPrecompClear instruction.

Allowed Values:

0   "entire_clearing_pulse": During the   2   "falling_edge": At the falling edge of the
    entire clearing pulse (Level).   clearing pulse.

1   "rising_edge": At the rising edge of   3   "both_egdes": Both, at the rising and
    clearing pulse.   falling edge of the clearing pulse.

/DEV..../SIGOUTS/n/PRECOMPENSATION/HIGHPASS/n/ENABLE

Properties: Read, Write, Setting   Type: Integer (64 bit)   Unit: None

Enables (1) or disables (0) the high-pass compensation filter.

/DEV..../SIGOUTS/n/PRECOMPENSATION/HIGHPASS/n/STATUS

Properties: Read   Type: Integer (64 bit)   Unit: None

Indicates the status of the high-pass compensation filter: 0 = normal, 1 = overflow during the last update period (~100 ms), 2 = overflowed in the past.

/DEV..../SIGOUTS/n/PRECOMPENSATION/HIGHPASS/n/ TIMECONSTANT

Properties: Read, Write, Setting   Type: Double   Unit: s

Sets the characteristic time constant of the high-pass compensation filter.

/DEV..../SIGOUTS/n/PRECOMPENSATION/STATUS/RESET

Properties: Read, Write   Type: Integer (64 bit)   Unit: None

Resets the status flags of all precompensation filters of this output channel.

6.2.8. SINES

/DEV..../SINES/n/HARMONIC

Properties: Read, Write, Setting   Type: Integer (64 bit)   Unit: None

Multiplies the sine signals's reference frequency with the integer factor defined by this field.

/DEV..../SINES/n/OSCSELECT

Properties: Read, Write, Setting   Type: Integer (64 bit)   Unit: None

Select oscillator for generation of this sine signal.

/DEV..../SINES/n/PHASESHIFT

Properties: Read, Write, Setting   Type: Double   Unit: deg

Phase shift applied to sine signal.

/DEV..../SINES/n/AMPLITUDES/n

Properties: Read, Write, Setting   Type: Double   Unit: None

Sets the peak amplitude that the sine signal contributes to the signal output. Note that the last index is either 0 or 1 and will map to the pair of outputs given by the first index. (e.g. sines/3/amplitudes/0 corresponds to wave output 2)

/DEV..../SINES/n/ENABLES/n

Properties: Read, Write, Setting   Type: Integer (64 bit)   Unit: None

Enables the sine signal to the signal output. Note that the last index is either 0 or 1 and will map to the pair of outputs given by the first index. (e.g. sines/3/amplitudes/0 corresponds to wave output 2)

6.2.9. 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/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/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/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.

/DEV..../STATS/PHYSICAL/POWER/CURRENTS/n

Properties: Read	Type: Double	Unit: A

Currents of the main power supply.

/DEV..../STATS/PHYSICAL/POWER/TEMPERATURES/n

Properties: Read	Type: Double	Unit: °C

Temperatures of the main power supply.

/DEV..../STATS/PHYSICAL/POWER/VOLTAGES/n

Properties: Read	Type: Double	Unit: V

Voltages of the main power supply.

/DEV..../STATS/PHYSICAL/SLAVEFPGAS/n/AUX

Properties: Read	Type: Double	Unit: V

Supply voltage of the FPGA.

/DEV..../STATS/PHYSICAL/SLAVEFPGAS/n/CORE

Properties: Read	Type: Double	Unit: V

Core voltage of the FPGA.

/DEV..../STATS/PHYSICAL/SLAVEFPGAS/n/TEMP

Properties: Read	Type: Double	Unit: °C

Internal temperature of the FPGA.

6.2.10. 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/TIME

Properties: Read   Type: Integer (64 bit)   Unit: None

The current timestamp.

/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 11: Sample Loss.

/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.

6.2.11. SYSTEM

/DEV..../SYSTEM/ACTIVEINTERFACE

Properties: Read   Type: String   Unit: None

Currently active interface of the device.

/DEV..../SYSTEM/FPGAREVISION

Properties: Read   Type: Integer (64 bit)   Unit: None

HDL firmware revision

/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/FX3REVISION

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/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/SAVEPORTS

Properties: Read, Write   Type: Integer (64 bit)   Unit: None

Flag indicating that the TCP and UDP ports should be saved.

/DEV..../SYSTEM/SHUTDOWN

Properties: Read, Write   Type: Integer (64 bit)   Unit: None

Sending a '1' to this node initiates a shutdown of the operating system on the MFLI device. It is recommended to trigger this shutdown before switching the device off with the hardware switch at the back side of the device.

/DEV..../SYSTEM/SLAVEREVISION

Properties: Read   Type: Integer (64 bit)   Unit: None

HDL firmware revision of the slave FPGA.

/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/AWG/CHANNELGROUPING

Properties: Read, Write, Setting   Type: Integer (enumerated)   Unit: None

Sets the channel grouping mode of the device.

Allowed Values:

0	"groups_of_2": Use the outputs in groups of 2. One sequencer program controls 2 outputs (use /dev..../awgs/0..4/).		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.
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/)			

/DEV..../SYSTEM/AWG/OSCILLATORCONTROL

Properties: Read, Write, Setting   Type: Integer (enumerated)   Unit: None

Sets the oscillator control mode.

Allowed Values:

0	"api": Oscillators are controlled by the UI/API.	1	"awg_sequencer": Oscillators are controlled by the AWG sequencer.

/DEV..../SYSTEM/BOARDREVISIONS/n

Properties: Read   Type: String   Unit: None

Hardware revision of the FPGA base board

/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: None

The maximum demodulator time constant that can be set.

/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: None

The minimum demodulator time constant that can be set.

/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. Is equal to 1/(maximum sampling rate).

/DEV..../SYSTEM/CLOCKS/REFERENCECLOCK/FREQ

Properties: Read	Type: Double	Unit: Hz

Indicates the frequency of the reference clock.

/DEV..../SYSTEM/CLOCKS/REFERENCECLOCK/SOURCE

Properties: Read, Write, Setting	Type: Integer (enumerated)	Unit: None

Reference clock source.

Allowed Values:

0   "internal": The internal clock is used as the frequency and time base reference.

1   "external": An external clock is intended to be used as the frequency and time base reference. Provide a clean and stable 10MHz or 100MHz reference to the appropriate back panel connector.

2   "zsync": A ZSync clock is intended to be used as the frequency and time base reference.

/DEV..../SYSTEM/CLOCKS/REFERENCECLOCK/STATUS

Properties: Read   Type: Integer (enumerated)   Unit: None

Status of the reference clock.

Allowed Values:

0	Reference clock has been locked on.
1	There was an error locking onto the reference clock signal. After an error the source is automatically switched back to internal reference clock.

2	The device is busy trying to lock onto the reference clock signal.

/DEV..../SYSTEM/CLOCKS/SAMPLECLOCK/FREQ

Properties: Read, Write, Setting   Type: Double   Unit: Hz

Indicates the frequency of the sample clock. Changing the sample clock temporarily interrupts the AWG sequencers.

/DEV..../SYSTEM/CLOCKS/SAMPLECLOCK/STATUS

Properties: Read   Type: Integer (enumerated)   Unit: None

Status of the sample clock.

Allowed Values:

0	Sample clock signal is valid and has been locked on.
1	There was an error adjusting the sample clock.

2	The device is busy trying to adjust the sample clock.

/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/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.

6.2.12. TRIGGERS

/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/IN/n/VALUE

Properties: Read   Type: Integer (64 bit)   Unit: None

Shows the trigger input. The value integrated over some time. Values are 1: low, 2: high, 3: was low and high in the period.

/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/SOURCE

Properties: Read, Write, Setting   Type: Integer (enumerated)   Unit: None

Assign a signal to a marker.

Allowed Values:

0 "awg_trigger0": Trigger output is assigned to AWG Trigger 1, controlled by AWG sequencer commands.

1 "awg_trigger1": Trigger output is assigned to AWG Trigger 2, controlled by AWG sequencer commands.

2 "awg_trigger2": Trigger output is assigned to AWG Trigger 3, controlled by AWG sequencer commands.

3 "awg_trigger3": Trigger output is assigned to AWG Trigger 4, controlled by AWG sequencer commands.

4 "output0_marker0": Output is assigned to Output 1 Marker 1.

5 "output0_marker1": Output is assigned to Output 1 Marker 2.

6 "output1_marker0": Output is assigned to Output 2 Marker 1.

7 "output1_marker1": Output is assigned to Output 2 Marker 2.

9 "trigin1", "trigger_input1": Output is assigned to Trigger Input 2.

10 "trigin2", "trigger_input2": Output is assigned to Trigger Input 3.

11 "trigin3", "trigger_input3": Output is assigned to Trigger Input 4.

12 "trigin4", "trigger_input4": Output is assigned to Trigger Input 5.

13 "trigin5", "trigger_input5": Output is assigned to Trigger Input 6.

14 "trigin6", "trigger_input6": Output is assigned to Trigger Input 7.

15 "trigin7", "trigger_input7": Output is assigned to Trigger Input 8.

17 "high": Output is set to high.

18 "low": Output is set to low.

8   "trigin0", "trigger_input0": Output is
    assigned to Trigger Input 1.

/DEV..../TRIGGERS/STREAMS/n/ENABLE

Properties: Read, Write, Setting   Type: Integer (64 bit)   Unit: None

Enables trigger streaming.

/DEV..../TRIGGERS/STREAMS/n/HOLDOFFTIME

Properties: Read, Write, Setting   Type: Double   Unit: s

Sets the holdoff time of the trigger unit.

/DEV..../TRIGGERS/STREAMS/n/MASK

Properties: Read, Write, Setting   Type: Integer (64 bit)   Unit: None

Masks triggers for the current stream. The mask is bit encoded where bit 0..7 are the input triggers and bit 8..11 are AWG triggers.

/DEV..../TRIGGERS/STREAMS/n/SAMPLE

Properties: Read, Stream   Type: ZITriggerSample   Unit: None

Streaming node containing the trigger data. Note that this will only deliver data if triggers are detected.

6.2.13. ZI (Labone Data Server Nodes)

/ZI/ABOUT/COMMIT

Properties: Read   Type: String   Unit: None

Contains the commit hash of the source code used to build this version of the LabOne software.

/ZI/ABOUT/COPYRIGHT

Properties: Read   Type: String   Unit: None

Holds the copyright notice.

/ZI/ABOUT/DATASERVER

Properties: Read   Type: String   Unit: None

Contains information about the Zurich Instruments Data Server.

/ZI/ABOUT/FWREVISION

Properties: Read   Type: Integer (64 bit)   Unit: None

Contains the revision of the device firmware.

/ZI/ABOUT/REVISION

Properties: Read   Type: Integer (64 bit)   Unit: None

Contains the revision number of the Zurich Instruments Data Server.

/ZI/ABOUT/VERSION

Properties: Read   Type: String   Unit: None

Contains the version of the LabOne software.

/ZI/CLOCKBASE

Properties: Read   Type: Double   Unit: None

A fallback clock frequency that can be used by clients for calculating time bases when no other is available.

/ZI/CONFIG/OPEN

Properties: Read, Write, Setting   Type: Integer (enumerated)   Unit: None

Enable communication with the LabOne Data Server from other computers in the network.

Allowed Values:

0	"local": Communication only possible with the local machine.	1	"network": Communication possible with other machines in the network.

/ZI/CONFIG/PORT

Properties: Read   Type: Integer (64 bit)   Unit: None

The IP port on which the LabOne Data Server listens.

/ZI/DEBUG/LEVEL

Properties: Read, Write, Setting   Type: Integer (enumerated)   Unit: None

Set the logging level (amount of detail) of the LabOne Data Server.

Allowed Values:

0   "trace": Trace. Messages designated as traces are logged.

1   "debug": Debug. Messages designated as debugging info are logged.

2   "info": Info. Messages designated as informational are logged.

3   "status": Status. Messages designated as status info are logged.

4   "warning": Warning. Messages designated as warnings are logged.

5   "error": Error. Messages designated as errors are logged.

6   "fatal": Fatal. Messages designated as fatal errors are logged.

/ZI/DEBUG/LOG

Properties: Read	Type: String	Unit: None

Returns the logfile text of the LabOne Data Server.

/ZI/DEBUG/LOGPATH

Properties: Read	Type: String	Unit: None

Returns the path of the log directory.

/ZI/DEVICES/CONNECTED

Properties: Read	Type: String	Unit: None

Contains a list of devices connected to the LabOne Data Server.

/ZI/DEVICES/DISCOVER

Properties: Read, Write	Type: String	Unit: None

Not used.

/ZI/DEVICES/VISIBLE

Properties: Read	Type: String	Unit: None

Contains a list of devices in the network visible to the LabOne Data Server.

/ZI/MDS/GROUPS/n/DEVICES

Properties: Read, Write, Setting	Type: String	Unit: None

Contains a list of devices in this synchronization group.

/ZI/MDS/GROUPS/n/KEEPALIVE

Properties: Read, Write, Setting   Type: Integer (64 bit)   Unit: None

Set by the MDS module to indicate control over this synchronization group.

/ZI/MDS/GROUPS/n/LOCKED

Properties: Read, Write, Setting   Type: Integer (64 bit)   Unit: None

Indicates whether the device group is locked by a MDS module.

/ZI/MDS/GROUPS/n/STATUS

Properties: Read, Write, Setting   Type: Integer (enumerated)   Unit: None

Indicates the status the synchronization group.

Allowed Values:

-1	Error. An error occurred in the synchronization process.	1	Sync
0	New	2	Alive

/ZI/SYSTEM/USAGEDATA

Properties: Read   Type: String   Unit: None

Contains a JSON formatted string giving usage information about the LabOne Software (which tabs/modules used, error conditions).

Chapter 7. HDIQ IQ Modulator

This chapter contains product overview and specifications of the HDIQ IQ Modulator, which is an accessory of the HDAWG. This PDF document is distributed as part of the HDAWG User Manual. A hard copy of this chapter is delivered with the instrument upon delivery.

The content of the chapter starts with the declaration of conformity of the HDIQ Instrument, and continues with sections including getting started, features and specifications. The last section details several possible applications.

7.1. Declaration of Conformity

The manufacturer

Zurich Instruments
Technoparkstrasse 1
8005 Zurich
Switzerland

declares that the product

HDIQ IQ Modulator

fulfills the requirements of the European guidelines

- 2014/30/EC Electromagnetic Compatibility
- 2014/35/EC Low Voltage
- 2011/65/EU, 2015/863/EU, 2017/2102/EU Restriction of Hazardous Substances (RoHS)
- 1907/2006/EC Registration, Evaluation, Authorization, and Restriction of Chemicals (REACH)

The assessment was performed using the directives according to Table 7.1.

Table 7.1. Conformity table

EN 61326-1:2012 (ed.2.0)	Electrical equipment for measurement, control and laboratory use - EMC requirements - Part 1: General requirements Emission: CISPR11 Immunity: IEC 61000-4-2/3/4/6
EN 61010-1:2010 IEC61010-1:2010 + AMD1:2016 IEC61010-1 National Deviations for EU, US, CA	Safety requirements for electrical equipment for measurement, control and laboratory use

$$CE$$

7.2. Getting Started

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

- Inspecting the package contents and accessories
- List of essential handling and safety instructions
- Connecting to the HDIQ Instrument

7.2.1. 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 HDIQ Instrument
- You have received 1 AC/DC power adapter
- You have received 1 power cord with a power plug suited to your country
- You have received 1 15-pin D-sub cable assembly
- A printed version of the "HDIQ IQ Modulator" chapter, the "Declaration of Conformity" section and the "Getting Started" section are included
- The serial number of the instrument is displayed on a sticker on the back panel

Table 7.2. Package contents for the HDIQ Instrument
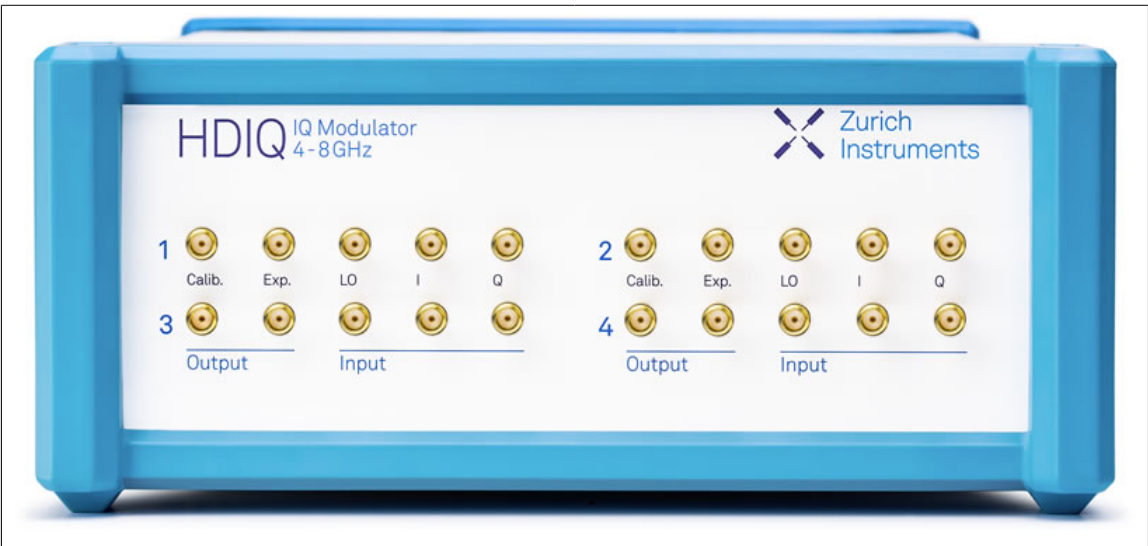
AC/DC power adapter	Power cord (example: EU norm)
15-pin D-sub cable assembly	S/N:HDIQ-DEV14005 S/N sticker on the back panel of the instrument

The HDIQ Instrument can be connected to most power systems in the world by using the enclosed AC/DC power adapter.

Carefully inspect your Instrument. If there is mechanical damage, then you should immediately notify the Zurich Instruments support team at <support@zhinst.com>.

7.2.2. Handling and Safety Instructions

The HDIQ Instrument is a sensitive piece of electronic equipment, which under no circumstances should be opened. 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 cancels 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 affect correct 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.

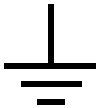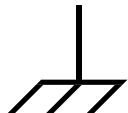Table 7.3. Safety Instructions

Maximum ratings	The specified electrical ratings for the connectors of the instrument should not be exceeded at any time during operation. Please refer to Section 7.4.1 for a comprehensive list of ratings.
Do not service or adjust anything yourself	There are no serviceable parts inside the Instrument.
Warnings	Instructions contained in any warning issued by the instrument, either by the software, 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 the correct interpretation of 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 I and pollution degree 2 environment as per IEC 61010-1. Do not operate or store the instrument outside the ambient conditions specified in Section 7.4.1.
Cleaning	To prevent electrical shock, disconnect the instrument from power supply 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.
Main power disconnect	Unplug product from power supply and remove power cord before servicing. Only qualified, service-trained personnel should remove the cover from the instrument.
Operation and storage	Do not operate or store at the instrument outside the ambient conditions specified in Section 7.4.1.
Handling	Do not drop the Instrument, handle with due care, do not store liquids on the device as there is a chance of spilling and 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.

When 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 by email at <support@zhinst.com>.

Table 7.4. Unusual Conditions

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 only with a power cord specified for this product and certified for the country of use.
Instrument emits abnormal noise, smell, or sparks	Switch off the Instrument immediately to prevent large damage.
Instrument is damaged	Switch off the Instrument immediately and secure it against unintended operation.

Table 7.5. Symbols

	Earth ground
	Chassis ground
	Caution. Refer to accompanying documentation

7.2.3. Connecting to the Instrument

The Zurich Instruments HDIQ Instrument operates in a default mode (i.e., RF output) when it is powered. The operating modes of the HDIQ are detailed in Section 7.3.1. To switch the instrument to other modes, a third-party digital ouput card is required, the requirements are described in Section 7.3.4. The digital input 15-pin D-sub connector on the back panel of the HDIQ Instrument is shown in Figure 7.4, and the pin assignment is listed in Table 7.9. With the 15-pin D-sub cable assembly, the HDIQ can be connected to the third-party digital output card. After an established connection between the third-party digital output card and a host computer, the HDIQ can be controlled remotely using a dedicated program supported by the third-party. To perform a simple test measurement, please refer to Section 7.5.

We recommend users to control the HDIQ with USB-6501 Digital I/O card from National instruments. Python programming examples for the USB-6501 Digital I/O card are available upon request to <support@zhinst.com>.

7.3. Features

This section provides an overview of the features of the HDIQ Instrument. The first subsection contains the description of the graphical overview and the hardware feature list. The following subsection details the front panel and the back panel of the HDIQ Instrument. The last section describes the product power supply and remote control.
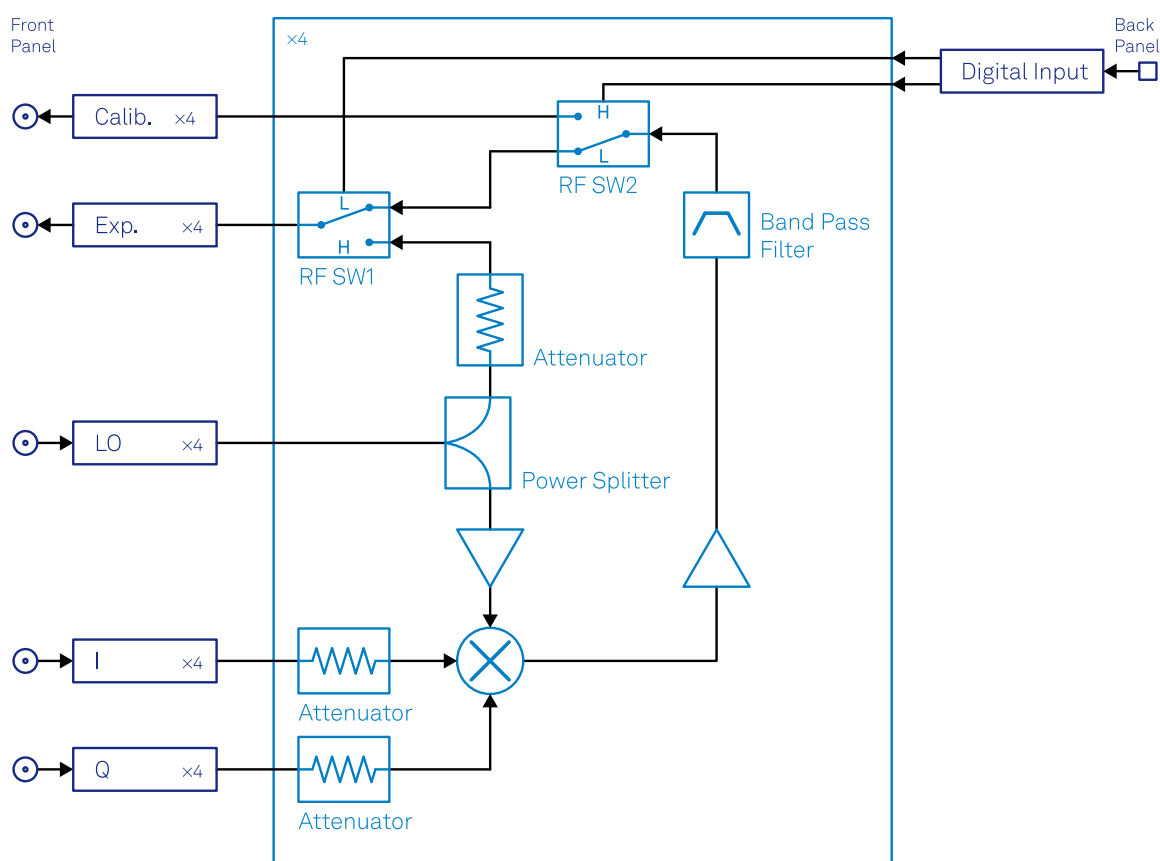
7.3.1. Functional Overview

Figure 7.1. HDIQ Instrument functional diagram

The HDIQ Instrument according to Figure 7.1 consists of several internal units (light blue color) surrounded by several interface units (dark blue color), 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 indicate selected physical connections and the data flow.

Table 7.6. Truth table of RF switches

Operating mode	Output signal	RF SW1[1]	RF SW2[2]
RF (default)	RF (Exp.)	L	L
LO	LO (Exp.)	H	L
Calibration	RF (Calib.)	L	H
N/A	N/A	H	H

[1]SW1 is the RF switch for the RF or the LO signal that goes to the Exp. output port.

[2]SW2 is the RF switch for the RF signal that goes to the Calib. or to the Exp. output port.

The HDIQ IQ Modulator consists of 4 identical frequency up-conversion units. Each of them counts 3 inputs, I (in-phase), Q (quadrature) and LO (local oscillator), and two outputs, Exp. (experiment) and Calib. (calibration). The amplified and filtered up-converted signal after the IQ mixer goes either to the Calib. port for mixer calibration or to the Exp. port for qubit manipulation. The Exp. port can also output the attenuated LO signal, which can then be used for qubit spectroscopy. The operating mode (mixer calibration, RF output, or LO output) is selected by 2 RF switches controlled by digital input signals, described in Table 7.6.

Key features

- 4-channel IQ modulator
- RF frequency range 4 - 8 GHz
- Intermediate frequency range DC - 6 GHz
- Experiment output port switchable between RF output and direct LO output
- Separate output port for mixer calibration
- Floating RF circuit design preventing ground loops
- Python programming examples for mixer calibration

7.3.2. Front Panel Tour

The front panel SMA connectors and control LEDs are arranged as shown in Figure 7.2 and listed in Table 7.7.

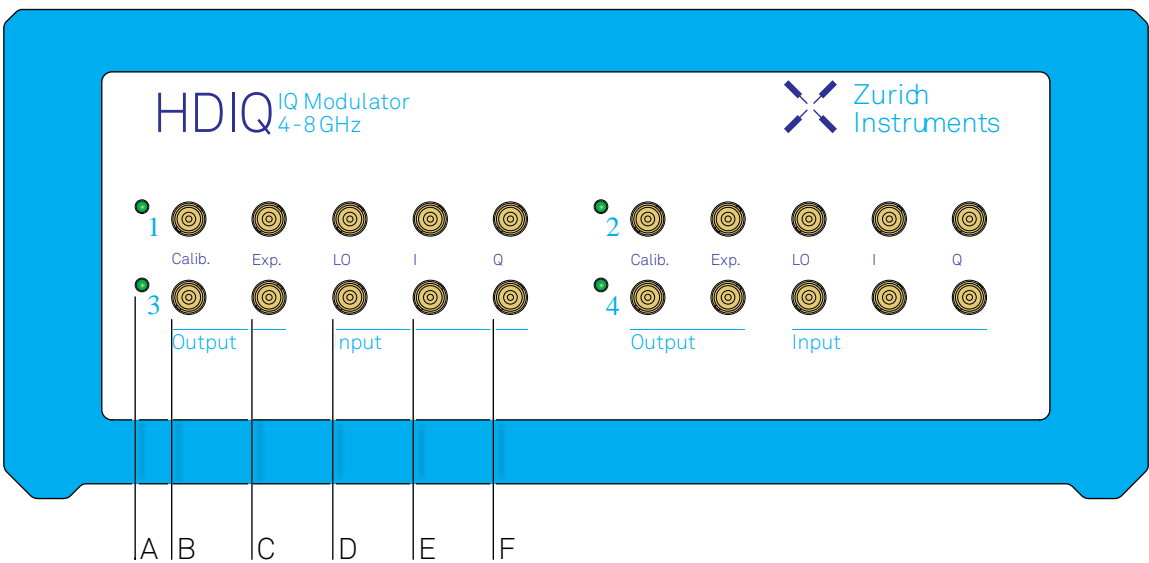Figure 7.2. HDIQ IQ Modulator front panel

Table 7.7. HDIQ Instrument front panel description

Position	Label/Name	Description
A	N/A	This multi-color LED indicates the state of the associated output.
		color blue: output enabled, RF output to the Exp. port (default RF mode)

Position	Label/Name	Description
		color green: output enabled, LO output to the Exp. port (LO mode)
		color orange: output enabled, RF output to the Calib. port (calibration mode)
B	Calib.	calibration output port
C	Exp.	experiment output port
D	LO	local oscillator input
E	I	In-phase signal input
F	Q	quadrature signal input

7.3.3. Back Panel Tour

The back panel is the main interface for power and control. Please refer to Figure 7.3 and Table 7.8 for the detailed description of the items.
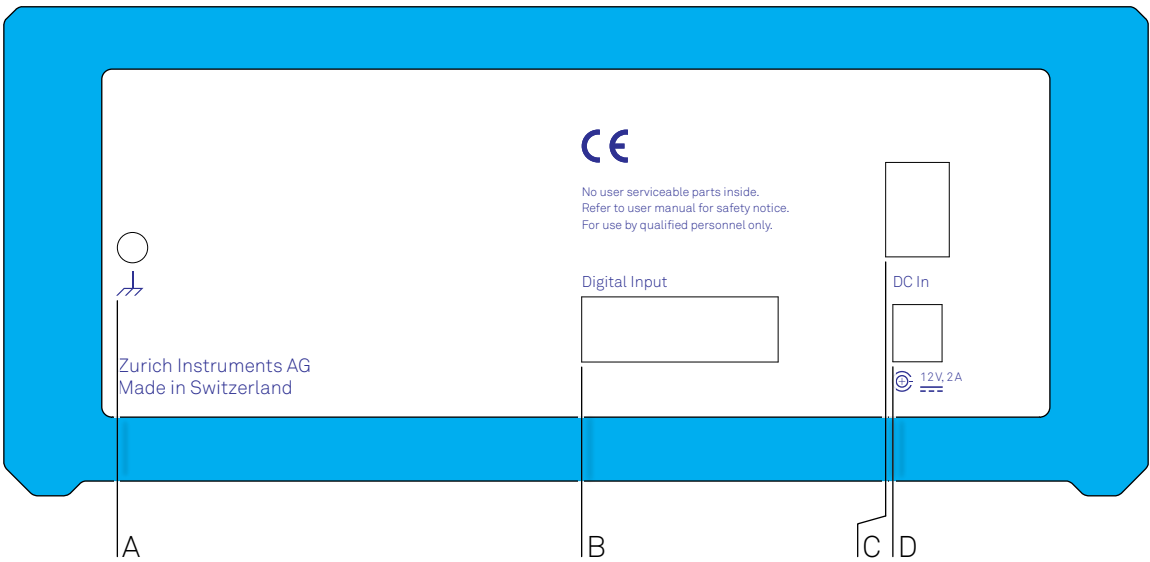
Figure 7.3. HDIQ IQ Modulator back panel

Table 7.8. HDIQ Instrument back panel description

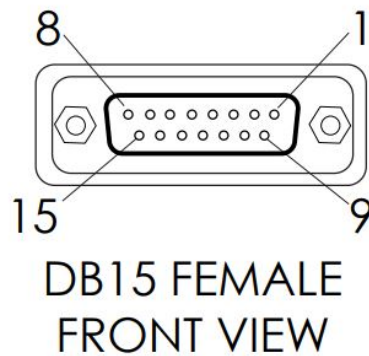| Position	Label / Name	Description
A	⏚	4 mm banana jack connector for chassis ground, electrically connected to the chassis.
B	Digital Input	Digital input for RF switch control. The connector DB15 drawing is shown in Figure 7.4 and the pin description is listed in Table 7.9
C	N/A	Power switch
D	DC in	DC external 12 V power supply

Figure 7.4. HDIQ IQ Modulator Digital input 15-pin connector

Table 7.9. Digital input pin assignment

Pin	HDIQ channel	RF switch
DB1	1	SW2
DB2	1	SW1
DB3	3	SW2
DB4	3	SW1
DB5	4	SW2
DB6	4	SW1
DB7	2	SW2
DB8	2	SW1
DB9	Reserved	-
DB10	NC	-
DB11-DB15	GND	-

7.3.4. Power Supply and Remote Control

The HDIQ is powered by 12 V, 2 A, DC input. The AC/DC power adapter and the power cord with a power plug suited to your country are included in the product package.

The HDIQ RF switches can be remotely controlled by any digital output card with current at least 5 mA for each channel. The recommended digital input/output card is USB-6501 (24 DIO) from National Instruments. It can be used to control up to 2 HDIQs, limited by the maximum output current of the DIO card. Reference Python programming examples for RF switch control and mixer calibration using NI USB-6501 are available upon request to <support@zhinst.com>.

7.4. Specifications

Important

Unless otherwise stated, all specifications apply after 30 minutes of instrument warm-up.

Important

Important changes in the specification parameters are explicitly mentioned in the revision history of this document.

7.4.1. General Specifications

Table 7.10. EMC, environment and safety

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	-	5.2 W	8 W
operating environment	indoor location, installation category I, pollution degree 2		

Table 7.11. Physical characteristics

Parameter	
dimensions including bumper	28.3 × 23.2 × 10.2 cm, 11.1 × 9.1 ×4.0 inch, rack mount on request
weight including bumper	2.9 kg, 6.4 lb
DC power inlet	12V, 2 A Connector: Switchcraft 760BK, ID 2.5 mm, OD 5.5 mm
mating digital input connector	Phoenix Contact 2926438

Table 7.12. Absolute Maximum Ratings

Parameter	
damage threshold I and Q inputs (50 Ω input impedance)	30 dBm
damage threshold LO input (50 Ω input impedance)	20 dBm
damage threshold Exp. RF output (50 Ω output impedance)	26 dBm
damage threshold Exp. LO output (50 Ω output impedance)	20 dBm

Parameter	
damage threshold Calib. RF output (50 Ω output impedance)	26 dBm
DI input in default configuration 3.3 V TTL	6.5 V

7.4.2. Analog Interface Specifications

Table 7.13. General

Parameter	Details	min	typ	max
number of input and output	-		4	
connectors of input and output	-	SMA, front panel single-ended		
impedance of input and output	-	50 Ω		
input and output frequency range				
frequency range of I and Q input	-	DC	-	6 GHz
frequency range of LO input and RF output	calibration and RF mode	4 GHz	-	8 GHz
frequency range of LO input and output	LO mode	1.8 GHz	-	10 GHz
input and output power				
I and Q input 1 dB input compression point[1]	-	-	0 dBm	-
LO input 1 dB input compression point[1]	calibration and RF mode	-	+6 dBm	-
LO input power	calibration and RF mode	+1 dBm	+5 dBm	+7 dBm
	LO mode	-	-	+7 dBm
attenuation of LO output power ($P_{LO, input}/P_{LO, output}$)	LO mode	15 dB	16 dB	17 dB
RF output 1 dB output compression point[1]	calibration and RF mode	-	+5 dBm	-
conversion gain ($P_{RF, Exp.}/P_I$ or $P_{RF, Exp.}/P_Q$)	measured at 4 GHz	-	+5 dB	-
operating modes				
switchable operating modes	-	calibration, RF and LO		
RF switches				
LO isolation ($P_{LO, input}/P_{LO, Exp.}$)	RF mode	> 57 dB		
switching time	-	< 0.1 s		
control interface				
digital input signal	into 390 Ω	3.3 V	5 V	6.5 V

[1] This is limited by the amplifier and not by the IQ mixer.

7.5. Applications

The HDIQ is designed for superconducting qubit experiments. The high-quality IQ mixers of the HDIQ allow users to minimize LO leakage and suppress the image sideband. The amplifiers and filters of the HDIQ ensure a reasonable output power with strong suppression of higher-order harmonics and spurs, thus enabling fast and high-fidelity gate operation. Two examples of applications for 4-qubit systems are qubit experiments and mixer calibration, as shown in Figure 7.5.
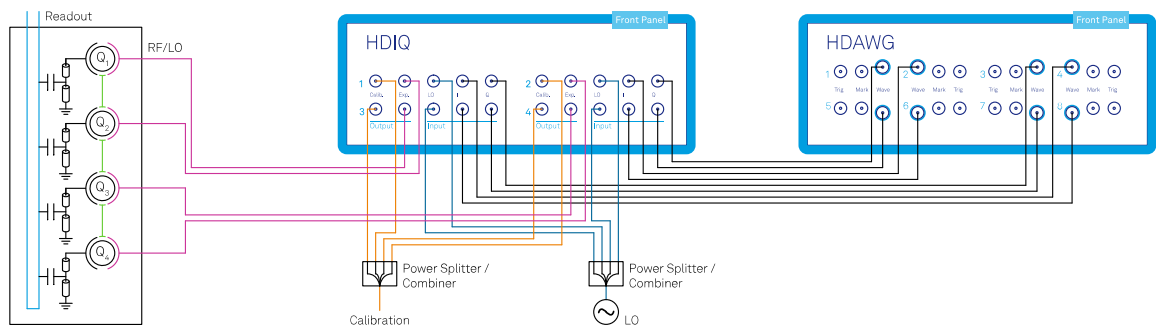

Figure 7.5. Application example of the HDIQ Instrument

7.5.1. Qubit Experiments

To generate the qubit driving pulses, an external LO source and an HDAWG are required. The in-phase (I) and quadrature (Q) input signals come from the HDAWG and the LO input signal is taken from an external microwave source. In default mode, i.e., with both RF switches in logic "low", the RF output signal from the HDIQ Exp. port goes to the physical system for qubit manipulation.

For qubit spectroscopy, wide frequency and amplitude scans are often required. This can be achieved by operating the HDIQ in LO output mode without any cable reconnections.

For qubit readout, the HDIQ can be used for readout pulse generation with the UHFQA Quantum Analyzer and an external LO source. An external RF switch can be used to switch between readout signal analysis and mixer calibration without any additional cabling work.

7.5.2. Mixer Calibration

LO leakage and image sideband suppression of IQ mixers are both frequency-dependent, and any non-negligible LO leakage and image sideband may cause off-resonant and unwanted excitations. To eliminate this effect, all IQ mixers have to be calibrated.

The mixers can be calibrated by measuring the Calib. output signal of the HDIQ in calibration mode with the UHFQA FFT function while scanning the pulse parameters of I and Q signals generated by the HDAWG. Reference Python code for mixer calibration of a single HDIQ and multiple HDIQs using the HDAWG and UHFQA is available upon request by writing to `<support@zhinst.com>`.

Glossary

This glossary provides easy to understand descriptions for many terms related to measurement instrumentation including the abbreviations used inside this user manual.

A

A/D	Analog to Digital See Also ADC.
AC	Alternate Current
ADC	Analog to Digital Converter
AM	Amplitude Modulation
Amplitude Modulated AFM (AM-AFM)	AFM mode where the amplitude change between drive and measured signal encodes the topography or the measured AFM variable. See Also Atomic Force Microscope.
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
Atomic Force Microscope (AFM)	Microscope that scans surfaces by means an oscillating mechanical structure (e.g. cantilever, tuning fork) whose oscillating tip gets so close to the surface to enter in interaction because of electrostatic, chemical, magnetic or other forces. With an AFM it is possible to produce images with atomic resolution. See Also Amplitude Modulated AFM, Frequency Modulated AFM, Phase modulation AFM.
AVAR	Allen Variance

B

Bandwidth (BW)	The signal bandwidth represents the highest frequency components of interest in a signal. For filters the signal bandwidth is the cut-off point, where the transfer function of a system shows 3 dB attenuation versus DC. In this context the bandwidth is a synonym of cut-off frequency $f_{cut-off}$ or 3dB frequency f_{-3dB}. The concept of bandwidth is used when the dynamic behavior of a signal is important or separation of different signals is required.
	In the context of a open-loop or closed-loop system, the bandwidth can be used to indicate the fastest speed of the system, or the highest signal update change rate that is possible with the system.
	Sometimes the term bandwidth is erroneously used as synonym of frequency range. See Also Noise Equivalent Power Bandwidth.
BNC	Bayonet Neill-Concelman Connector

C

CF	Clock Fail (internal processor clock missing)

Common Mode Rejection Ratio (CMRR)	Specification of a differential amplifier (or other device) indicating the ability of an amplifier to obtain the difference between two inputs while rejecting the components that do not differ from the signal (common mode). A high CMRR is important in applications where the signal of interest is represented by a small voltage fluctuation superimposed on a (possibly large) voltage offset, or when relevant information is contained in the voltage difference between two signals. The simplest mathematical definition of common-mode rejection ratio is: CMRR = 20 * log(differential gain / common mode gain).
CSV	Comma Separated Values

D

D/A	Digital to Analog
DAC	Digital to Analog Converter
DC	Direct Current
DDS	Direct Digital Synthesis
DHCP	Dynamic Host Configuration Protocol
DIO	Digital Input/Output
DNS	Domain Name Server
DSP	Digital Signal Processor
DUT	Device Under Test
Dynamic Reserve (DR)	The measure of a lock-in amplifier's capability to withstand the disturbing signals and noise at non-reference frequencies, while maintaining the specified measurement accuracy within the signal bandwidth.

E

XML	Extensible Markup Language. See Also XML.

F

FFT	Fast Fourier Transform
FIFO	First In First Out
FM	Frequency Modulation
Frequency Accuracy (FA)	Measure of an instrument's ability to faithfully indicate the correct frequency versus a traceable standard.
Frequency Modulated AFM (FM-AFM)	AFM mode where the frequency change between drive and measured signal encodes the topography or the measured AFM variable. See Also Atomic Force Microscope.
Frequency Response Analyzer (FRA)	Instrument capable to stimulate a device under test and plot the frequency response over a selectable frequency range with a fine granularity.

Frequency Sweeper	See Also Frequency Response Analyzer.

G

Gain Phase Meter	See Also Vector Network Analyzer.
GPIB	General Purpose Interface Bus
GUI	Graphical User Interface

I

I/O	Input / Output
Impedance Spectroscope (IS)	Instrument suited to stimulate a device under test and to measure the impedance (by means of a current measurement) at a selectable frequency and its amplitude and phase change over time. The output is both amplitude and phase information referred to the stimulus signal.
Input Amplitude Accuracy (IAA)	Measure of instrument's capability to faithfully indicate the signal amplitude at the input channel versus a traceable standard.
Input voltage noise (IVN)	Total noise generated by the instrument and referred to the signal input, thus expressed as additional source of noise for the measured signal.
IP	Internet Protocol

L

LAN	Local Area Network
LED	Light Emitting Diode
Lock-in Amplifier (LI, LIA)	Instrument suited for the acquisition of small signals in noisy environments, or quickly changing signal with good signal to noise ratio - lock-in amplifiers recover the signal of interest knowing the frequency of the signal by demodulation with the suited reference frequency - the result of the demodulation are amplitude and phase of the signal compared to the reference: these are value pairs in the complex plane (X, Y), (R, Θ).

M

Media Access Control address (MAC address)	Refers to the unique identifier assigned to network adapters for physical network communication.
Multi-frequency (MF)	Refers to the simultaneous measurement of signals modulated at arbitrary frequencies. The objective of multi-frequency is to increase the information that can be derived from a measurement which is particularly important for one-time, non-repeating events, and to increase the speed of a measurement since different frequencies do not have to be applied one after the other. See Also Multi-harmonic.
Multi-harmonic (MH)	Refers to the simultaneous measurement of modulated signals at various harmonic frequencies. The objective of multi-frequency is to increase the

information that can be derived from a measurement which is particularly important for one-time, non-repeating events, and to increase the speed of a measurement since different frequencies do not have to be applied one after the other.
See Also Multi-frequency.

N

Noise Equivalent Power Bandwidth (NEPBW)	Effective bandwidth considering the area below the transfer function of a low-pass filter in the frequency spectrum. NEPBW is used when the amount of power within a certain bandwidth is important, such as noise measurements. This unit corresponds to a perfect filter with infinite steepness at the equivalent frequency. See Also Bandwidth.
Nyquist Frequency (NF)	For sampled analog signals, the Nyquist frequency corresponds to two times the highest frequency component that is being correctly represented after the signal conversion.

O

Output Amplitude Accuracy (OAA)	Measure of an instrument's ability to faithfully output a set voltage at a given frequency versus a traceable standard.
OV	Over Volt (signal input saturation and clipping of signal)

P

PC	Personal Computer
PD	Phase Detector
Phase-locked Loop (PLL)	Electronic circuit that serves to track and control a defined frequency. For this purpose a copy of the external signal is generated such that it is in phase with the original signal, but with usually better spectral characteristics. It can act as frequency stabilization, frequency multiplication, or as frequency recovery. In both analog and digital implementations it consists of a phase detector, a loop filter, a controller, and an oscillator.
Phase modulation AFM (PM-AFM)	AFM mode where the phase between drive and measured signal encodes the topography or the measured AFM variable. See Also Atomic Force Microscope.
PID	Proportional-Integral-Derivative
PL	Packet Loss (loss of packets of data between the instruments and the host computer)

R

RISC	Reduced Instruction Set Computer
Root Mean Square (RMS)	Statistical measure of the magnitude of a varying quantity. It is especially useful when variates are positive and negative, e.g., sinusoids, sawtooth, square waves. For a sine wave the following relation holds between the

amplitude and the RMS value: $U_{RMS} = U_{PK} / \sqrt{2} = U_{PK} / 1.41$. The RMS is also called quadratic mean.

RT	Real-time

S

Scalar Network Analyzer (SNA)	Instrument that measures the voltage of an analog input signal providing just the amplitude (gain) information. See Also Spectrum Analyzer, Vector Network Analyzer.
SL	Sample Loss (loss of samples between the instrument and the host computer)
Spectrum Analyzer (SA)	Instrument that measures the voltage of an analog input signal providing just the amplitude (gain) information over a defined spectrum. See Also Scalar Network Analyzer.
SSH	Secure Shell

T

TC	Time Constant
TCP/IP	Transmission Control Protocol / Internet Protocol
Thread	An independent sequence of instructions to be executed by a processor.
Total Harmonic Distortion (THD)	Measure of the non-linearity of signal channels (input and output)
TTL	Transistor to Transistor Logic level

U

UHF	Ultra-High Frequency
UHS	Ultra-High Stability
USB	Universal Serial Bus

V

VCO	Voltage Controlled Oscillator
Vector Network Analyzer (VNA)	Instrument that measures the network parameters of electrical networks, commonly expressed as s-parameters. For this purpose it measures the voltage of an input signal providing both amplitude (gain) and phase information. For this characteristic an older name was gain phase meter. See Also Gain Phase Meter, Scalar Network Analyzer.

X

XML	Extensible Markup Language: Markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

Z

ZCtrl   Zurich Instruments Control bus

ZoomFFT   This technique performs FFT processing on demodulated samples, for
                            instance after a lock-in amplifier. Since the resolution of an FFT depends on
                            the number of point acquired and the spanned time (not the sample rate), it
                            is possible to obtain very highly resolution spectral analysis.

ZSync   Zurich Instruments Synchronization bus

Index

Symbols

A

B

C

D

E

F

G

H

I

L

M

N