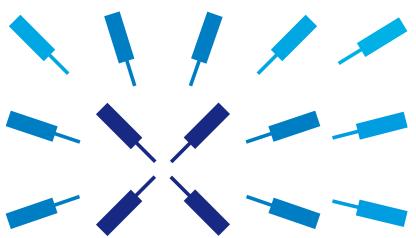


HF2 User Manual - LabOne Edition



Zurich
Instruments

HF2 User Manual - LabOne Edition

Zurich Instruments AG

Publication date Revision 38200

Copyright © 2008-2016 Zurich Instruments AG

The contents of this document are provided by Zurich Instruments AG (ZI), "as is". ZI makes no representations or warranties with respect to the accuracy or completeness of the contents of this publication and reserves the right to make changes to specifications and product descriptions at any time without notice.

LabVIEW is a registered trademark of National Instruments Inc. MATLAB is a registered trademark of The MathWorks, Inc. All other trademarks are the property of their respective owners.

Revision History

Revision 38200, 14-July-2016:

The entire document was updated to comply with the changes of the 16.04 LabOne release.

Highlights of the changes and additions to the HF2LI product are:

- Signal Output offset adjustment
- Software Trigger: new grid mode for 2D data capture and frame averaging for imaging applications
- Config tab: update reminder for the LabOne software

Revision 34390, 22-Dec-2015:

This is the first version of the HF2 User Manual (LabOne Edition). The document is related to the 15.11 LabOne release.

Highlights of the changes and additions to the HF2LI product are:

- New Option HF2LI-WEB: LabOne control software is now available for all HF2 series instruments
- Sweeper: faster sweeps due to improved instrument communication
- Modulation: change in FM generation method improves spectral purity. Limited to narrow-band FM signals

The HF2LI-WEB LabOne Web Interface comes with several new software tools that replace or extend the tools available in the Zurich Instruments ziControl software:

- PID: Tune function replaced by PID Advisor with extended DUT model library
- SW Trigger: scope-like data capture of streamed data, unavailable in ziControl
- Sweeper: LabOne Sweeper Reference replaces ziControl Sweeper Calibration and Reference
- Sweeper: new sweep parameters available in LabOne Sweeper: modulation amplitude and index
- Sweeper: advanced and application mode: configuration assistant for settling time/inaccuracy and omega suppression
- Sweeper: Q-factor extraction unavailable in LabOne Sweeper
- Spectrum: LabOne Spectrum tab replaces ziControl zoomFFT tool
- Spectrum: modes FFT(Theta), FFT(f), FFT(dTheta/dt) unavailable in ziControl
- Spectrum: noise power analysis unavailable in LabOne Spectrum tab
- Plotter: LabOne Plotter replaces ziControl Spectroscope. Larger selection of plotted data, support for multiple curves, more powerful plotting and analysis tools

Table of Contents

Declaration of Conformity	V
1. Getting Started	6
1.1. Quick Start Guide	7
1.2. Inspect the Package Contents	8
1.3. Handling and Safety Instructions	10
1.4. Software Installation and Update	13
1.5. Connecting to the Instrument	19
1.6. Troubleshooting	29
2. Functional Overview	33
2.1. Features	34
2.2. Front Panel Tour	37
2.3. Back Panel Tour	39
2.4. Ordering Guide	40
2.5. Operating Modes	42
3. Tutorials	47
3.1. Tutorial HF2LI First Time User	48
3.2. Tutorial Simple Loop	54
3.3. Tutorial Dynamic Signals	58
3.4. Tutorial External Reference	63
3.5. Tutorial Noise Measurement	67
3.6. Tutorial Amplitude Modulation	68
3.7. Tutorial Frequency Modulation	72
3.8. Tutorial Phase-locked Loop	77
4. Functional Description LabOne User Interface	81
4.1. User Interface Overview	82
4.2. Lock-in Tab	95
4.3. Lock-in Tab (HF2-MF option)	107
4.4. Numeric Tab	119
4.5. Plotter Tab	122
4.6. Scope Tab	125
4.7. Software Trigger Tab	130
4.8. Spectrum Analyzer Tab	137
4.9. Sweeper Tab	141
4.10. Auxiliary Tab	151
4.11. Inputs/Outputs Tab	153
4.12. DIO Tab	155
4.13. Config Tab	158
4.14. Device Tab	163
4.15. File Manager Tab	165
4.16. PLL Tab	167
4.17. PID Tab	175
4.18. MOD Tab	186
4.19. Real-time Tab	192
4.20. HF2CA Tab	193
4.21. HF2TA Tab	195
4.22. ZI Labs Tab	197
5. Communication and Connectivity	198
5.1. Instrument Connectivity Overview	199
5.2. ziServer's Text-based Interface	204
5.3. Connecting to ziServer over insecure or firewalled networks	212
6. Node Definitions	215
6.1. Overview	216
6.2. Nodes	228
7. Real-time Option	296

7.1. Installation of the Real-time Development Environment	297
7.2. Real-Time Option Reference Manual	301
8. Specifications	624
8.1. General Specifications	625
8.2. Analog Interface Specifications	627
8.3. Digital Interface Specifications	631
8.4. Performance Diagrams	634
8.5. Ground and Earth Scheme	644
8.6. Reference Images	646
8.7. Test Specifications	650
9. Signal Processing Basics	651
9.1. Principles of Lock-in Detection	652
9.2. Signal Bandwidth	655
9.3. Discrete-Time Filters	657
9.4. Full Range Sensitivity	659
9.5. Sinc Filtering	661
9.6. Zoom FFT	664
10. HF2CA Current Amplifier Data Sheet	666
10.1. Key Features	667
10.2. Specifications	668
10.3. Functional Description	671
10.4. Applications	673
10.5. Cable Recommendation	677
11. HF2TA Current Amplifier Data Sheet	678
11.1. Key Features	679
11.2. Specifications	680
11.3. Functional Description	683
11.4. Applications	685
11.5. Performance Tests	690
11.6. Cable Recommendation	694
Glossary	695
Index	701

Declaration of Conformity

The manufacturer

Zurich Instruments
Technoparkstrasse 1
8005 Zurich
Switzerland

declares that the product

HF2 Series (HF2LI, HF2IS), 50 MHz, 210 MSamples/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

The assessment was performed using the directives according to [Table 1](#).

Table 1. Conformity table

EN 61326-1:2006	Emissions for industrial environments, immunity for industrial environments
EN 55011	Group 1, class A and B (the product was tested in typical configuration)
EN 61000-4-2	CD 4 kV, AD 8 kV
EN 61000-4-3	10 V/m 80% AM 80 MHz - 1 GHz
	3 V/m 80% AM 1 MHz - 2 GHz
	1 V/m 80% AM 2 MHz - 2.7 GHz
EN 61000-4-4	2 kV power line
	1 kV USB line
EN 61000-4-5	1 kV line-line, 2 kV line-earth
EN 61000-4-6	3 V 80% AM, power line
EN 61010-1:2001	Safety requirements for electrical equipment for measurement, control and laboratory use



Figure 1. CE Logo

Chapter 1. Getting Started

This first chapter guides you through the initial set-up of your HF2 Instrument in order to make your first measurements. This chapter comprises of:

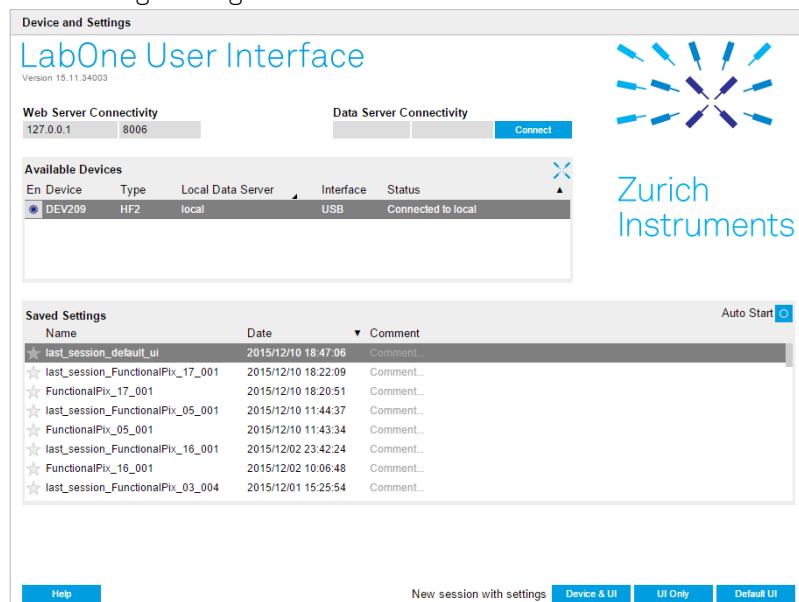
- Quick start guide
- Package content and accessories list
- Software installation instructions
- Powering-on the device, connecting the device via USB, and performing basic operation checks on the instrument
- List of essential handling and safety instructions

This chapter is delivered as a hard copy with the instrument upon delivery. It is also the first chapter of the HF2 User Manual.

1.1. Quick Start Guide

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

1. Check the package content. Besides the Instrument there should be a country-specific power cable, a USB 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 homepage <http://www.zhinst.com/downloads/>. More detailed instructions are found in [Section 1.4](#).
4. Connect the Instrument to the power line, turn it on and connect it to the PC using the USB cable. The front panel LED will show a steady blue color. If the LED does not turn blue, please contact Zurich Instruments for assistance.
5. Start the LabOne User Interface. This will open the Device and Settings dialog in your default web browser. Your Instrument appears in the list of Available Devices. An example of the Device and Settings dialog is shown below.



6. Click the **Default UI** button on the bottom right of the page. The default configuration will be loaded and the first measurements can be taken. If the user interface does not start up successfully, please refer to [Section 1.5.3](#).
7. The HF2 User Manual is included in a LabOne installation and can be accessed in Windows via the Start Menu → All programs / All apps → Zurich Instruments → HF2 User Manual .

If any problems are encountered whilst setting up the instrument and software please see the [troubleshooting section](#) at the end of this chapter.

Once the Instrument is up and running we recommend to go through some of the tutorials given in [Chapter 3](#). Moreover, [Chapter 2](#) provides a general introduction to the various tools and settings tabs with tables in each section providing a detailed description of every UI element as well. For specific application know-how the [Blog section](#) [www.zhinst.com/blogs/] of the Zurich Instruments web page will serve as a valuable resource that is constantly updated and expanded.

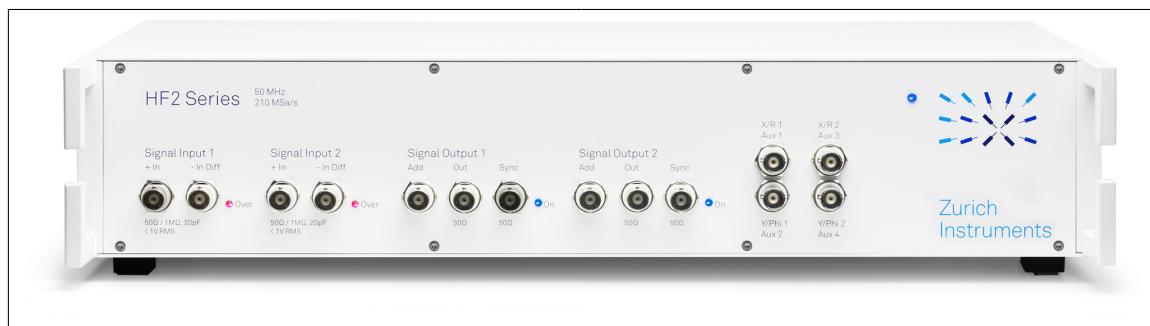
1.2. Inspect the Package Contents

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.

You must verify that:

- You have received 1 Zurich Instruments HF2 Instrument
- You have received 1 power cord with a power plug suited to your country
- You have received 1 USB cable
- A printed version of the "Getting Started" section
- Additional cables have been added to the delivery if an HF2 pre-amplifier has been delivered at the same time
- The line voltage selector on the HF2 Instrument power inlet indicates the correct line voltage of your country (115 V/60 Hz, or 230 V/50 Hz). While Zurich Instruments configures the power system when an instrument is initially delivered, no liability derives from potential wrong configuration or incorrect configuration at any point in time during the lifetime of the instrument
- 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
- For Japanese users only: you are supposed to operate the HF2 Instruments with an external 100V to 110V transformer in order to have reliable measurement results. Please verify having received the transformer included in your delivery from the local distributor.

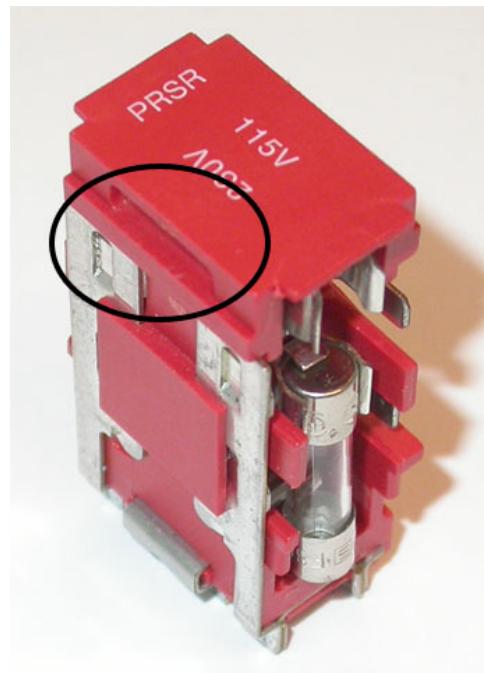
Table 1.1. Package contents for HF2 Instruments

	
Power cord	USB cable
	

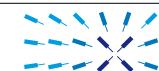
1.2. Inspect the Package Contents



Power inlet with selected 230 V/50 Hz power system



Fuse holder. Requires 2 x 20 mm fast-acting fuses with 800 mA current limit. To extract the fuse holder use a small screwdriver in the indicated spot to lift it out of the casing



Next Calibration
31 Dec 2013

The "Next Calibration" sticker on the rear panel of your instrument



Japanese users only: the 100 V to 110 V transformer

Carefully inspect your HF2 Instrument. If there is mechanical damage or the instrument does not seem to operate after the [software installation](#), please consult the [handling instructions](#), then notify the Zurich Instruments support team at <support@zhinst.com> as soon as possible.

1.3. Handling and Safety Instructions

The HF2 is a sensitive electronic instrument, 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. For more information on the ground and earth scheme, refer to section Section 8.5 .
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 Chapter 8 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.
Overseas travel	Consider that a power system change without changing the orientation of the fuse holder will damage the fuses, or make the instrument behaving unpredictably
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.
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 Chapter 8 . Do not block the ventilator opening on the back or the air intake on the side of the chassis 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 plugs	The two RJ45 plugs on the back panel labeled "PeripheralZCtrl" are not intended for Ethernet LAN connection. Connecting these plugs with an Ethernet device 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 Chapter 8 .
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.

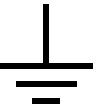
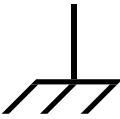
When you notice any of the situations listed below, immediately stop the operation of the Instrument, disconnect the power cord, and contact the support team at Zurich Instruments, either through the website form or by email at <support@zhinst.com>.

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
	DC (direct current)

1.4. Software Installation and Update

The HF2 Instrument is operated from a host computer with the LabOne software. To install the LabOne software on a PC administrator rights are required. Following installation, to simply run the software, 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 Instrument as described in this section.

1.4.1. Which User Interface: ziControl or the LabOne User Interface?

Up to and including software release 15.05, ziControl was the standard user interface (UI) shipped with HF2 Instruments. From 15.11 onwards HF2 Instruments ship with the LabOne User Interface, a browser-based UI. The ziControl UI is available to all users and the LabOne User Interface is available to users who have the WEB Option installed on their device. HF2 instruments shipped with release 15.11 onwards have the WEB Option installed by default. The WEB Option may be activated for other HF2 Instruments.

If the HF2 instrument does not have the WEB Option installed, then ziControl should be used as the user interface for the device and must be additionally installed as a separate package. In this case please refer to the "ziControl Edition" of the HF2 User Manual, ziControl installation is detailed there. The LabOne User Interface is included in the LabOne software package and does not require a separate installation.

1.4.2. Installing LabOne on Windows

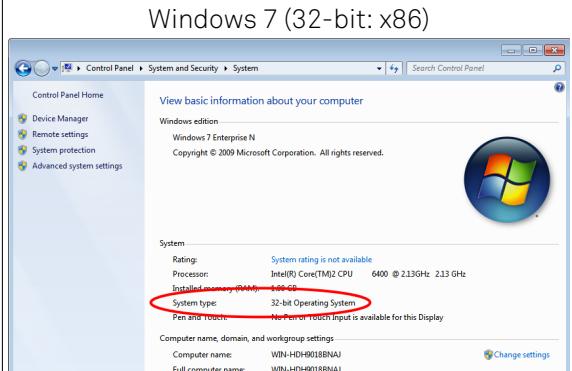
The installation packages for Zurich Instruments LabOne software are available as Windows installer .msi packages. The software is available on the Zurich Instruments download page, www.zhiinst.com/downloads. Please ensure that you have administrator rights for the PC where the software is to be installed 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.zhiinst.com/labone/compatibility for a comprehensive list of supported Windows systems.

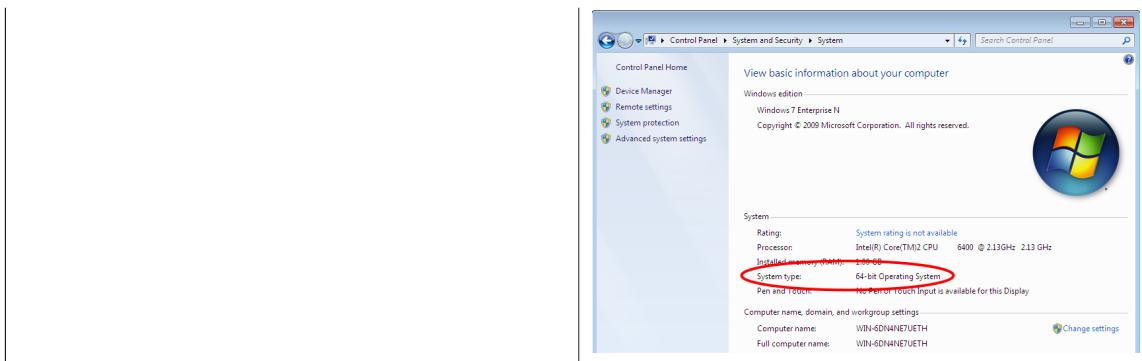
Determining PC Architecture on Microsoft Windows

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

Table 1.5. Find out the OS addressing architecture (32-bit or 64-bit)

Windows 7 (32-bit: x86)	Windows 7 (64-bit: x64)
 <p>Windows 7 (32-bit: x86)</p> <p>View basic information about your computer Windows edition Windows 7 Enterprise N Copyright © 2009 Microsoft Corporation. All rights reserved.</p> <p>System</p> <p>Rating: System rating is not available</p> <p>Processor: Intel(R) Core(TM)2 CPU 6400 @ 2.13GHz 2.13 GHz</p> <p>Installed RAM: 4.00 GB</p> <p>System type: 32-bit Operating System</p> <p>Pen and touch: No pen or touch input is available for this display</p> <p>Computer name, domain, and workgroup settings</p> <p>Computer name: WIN-HDHP0108NMAJ</p> <p>Full computer name: WIN-HDHP0108NMAJ</p>	<p>Windows 7 (64-bit: x64)</p>



Windows LabOne Installation

1. The HF2 Instrument should not be connected to your computer during the LabOne software installation process
2. Start the `LabOne32/64-xx.xx.xxxxxxx.msi` LabOne installer program by a double click and follow the instructions. Please note that Windows Administrator rights are required for installation. The installation proceeds as follows:
 - On the welcome screen click the **Next** button.



Figure 1.1. Installation welcome screen

- After reading through the Zurich Instruments license agreement, check the "I accept the terms in the License Agreement" check box and click the **Next** button.
- Review the features you want to have installed. For the HF2 Instrument the **HF2 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.

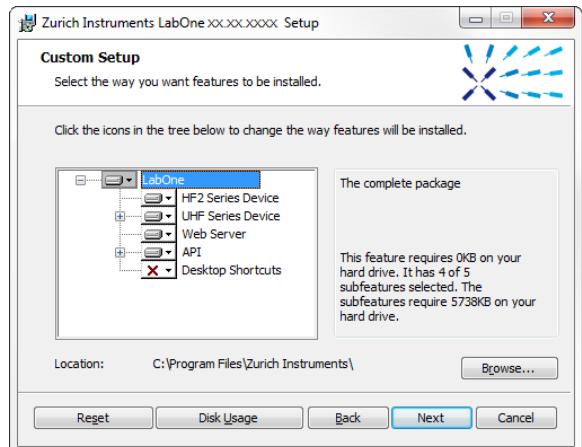


Figure 1.2. Custom setup screen

- 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**.

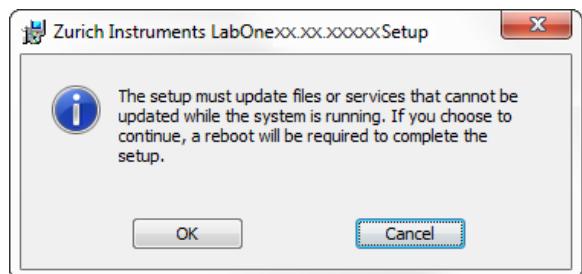


Figure 1.3. Installation reboot request

- 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**.

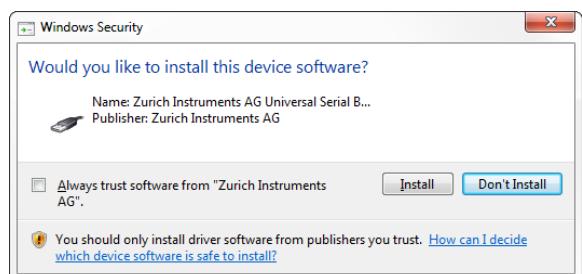


Figure 1.4. Installation driver acceptance

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

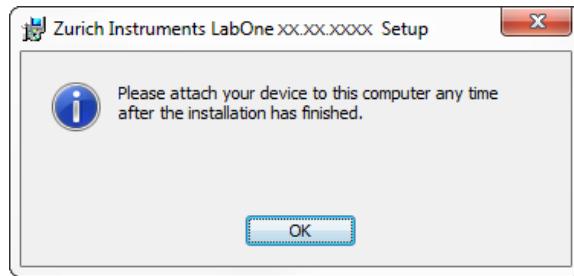


Figure 1.5. Installation completion screen

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, LabOne Software Start-up](#), and choose an instrument to connect to via the Device Settings Dialogue (described in the section called “Device and Settings Dialog”).

Warning

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

1.4.3. Installing LabOne on Linux

Requirements

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

1. Officially, Ubuntu 12.04 LTS and 14.04 LTS (i386, 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, but preferably Ubuntu 12.04 LTS or 14.04 LTS.
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](#) [<http://www.zhinst.com/downloads>]:
 - LabOneLinux<arch>-<release>.tar.gz, for example:
`LabOneLinux32/64-xx.xx.xxxxx.tar.gz`

Please ensure you download the correct architecture (32-bit/64-bit) of the LabOne installer. The `uname -m` command can be used in order to determine which architecture you are using, by running:

```
uname -m
```

in a command line terminal. If the command outputs "x86_64" the 64-bit version of the LabOne package is required, if it displays "x86_64" the 32-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. Check whether the HF2 Data Server is already running using the "ziService" program:

```
$ ziService status
```

If udev support was installed, the HF2 Data Server program "ziServer" should already be running. If not, start the Data Server manually at a command prompt:

```
$ ziServer
```

If udev support was installed, then the HF2 Data Server program is automatically started upon plugging in the HF2's USB cable and powering the instrument.

2. Start the Web Server program at a command prompt:

```
$ startWebServer
```

3. Start an up-to-date web browser and enter the 127.0.0.1:8006 in the browser's address bar to access the Web Server program and start the LabOne User Interface. The LabOne Web Server installed on the PC listens by default on port number 8006 instead of 80 to minimize the probability of conflicts.

4. You can now start the LabOne User Interface as described in [Section 1.5.2](#) and choose an instrument to connect to via the Device Settings Dialogue as described in the section called [“Device and Settings Dialog”](#).

Important

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

If your command log window is flooded with messages after starting the HF2 Data Server stop the program; it is likely that another instance of the Data Server is already running. Verify whether a Data Server is already running as described above using the ziService program.

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
```

1.5. Connecting to the Instrument

After the LabOne software has been installed, the HF2 instrument is ready to be connected to a PC by using the USB cable. This section gives you detailed instructions about the different possibilities of setting up the connection and start working with your instrument.

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 lock-in amplifiers is server based. The servers and other software components are organized in layers as shown in [Figure 1.6](#). The lowest layer running on the PC is the LabOne Data Server which is the interface to the connected lock-in amplifier. 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.

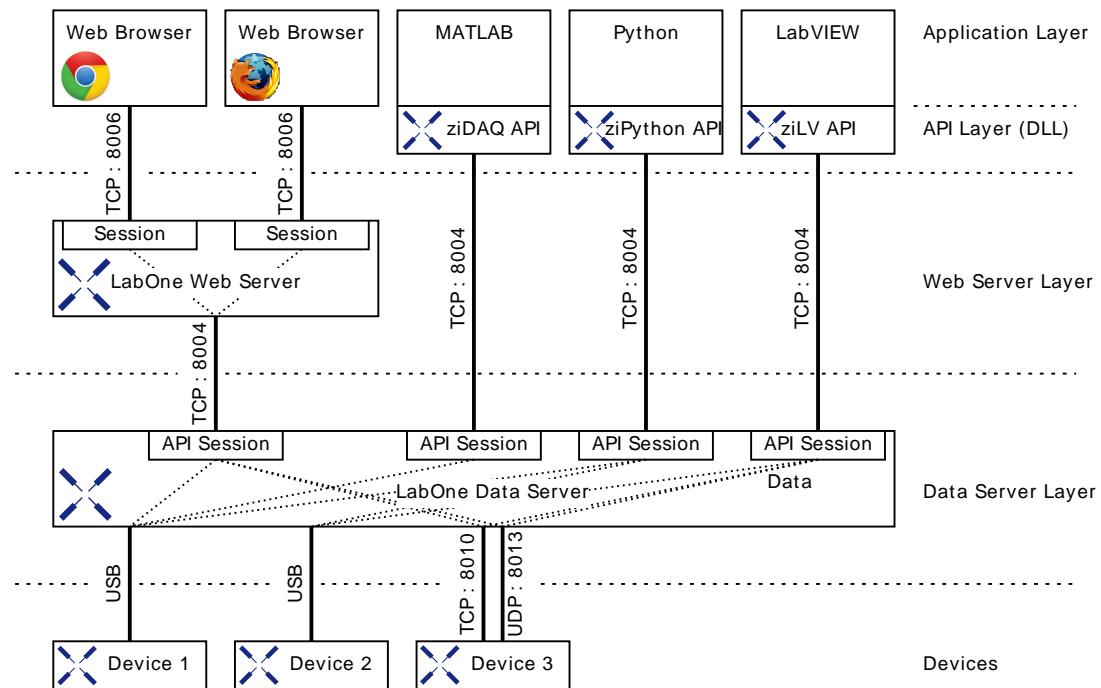


Figure 1.6. Software architecture

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 lock-in amplifiers. It will distribute the measurement data from the instrument to all the clients that subscribe to it. It

also ensures that settings changed by one client are communicated to other clients. The device settings are therefore synchronized on all clients. The HF2 Data Server is started automatically (on Windows via ziService, on Linux via udev) whenever a HF2 Instrument is connected to a PC via USB. 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 lock-in amplifier 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
- 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 LabOne User Interface start-up. 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 1.5.3 Device Connectivity](#).

The most straightforward method to control and obtain data from the instrument is to use the LabOne User Interface, which can be found under the Windows Start Menu (see [Figure 1.7](#) and

Figure 1.8): Click and select Start Menu → All programs / All apps → Zurich Instruments → LabOne User Interface. 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 structure is found in the [Section 1.5.1](#) LabOne Software Architecture.

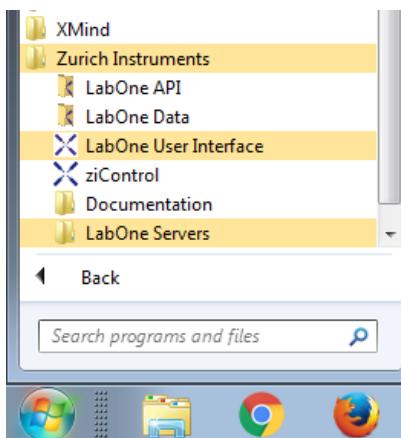


Figure 1.7. Link to the LabOne User Interface in the Windows 7 Start Menu (All programs)

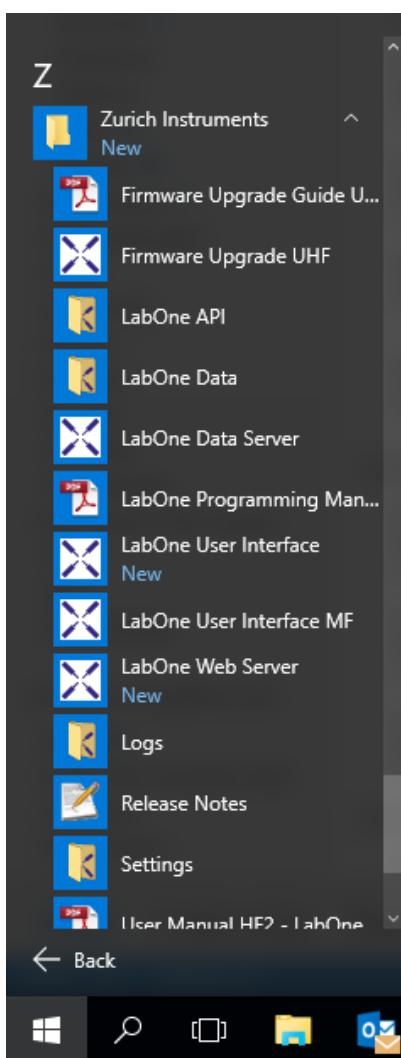


Figure 1.8. Link to the LabOne User Interface in the Windows 10 Start Menu (All apps)

The LabOne User Interface 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

The user interface requires the LabOne Web Server (that runs in combination with the LabOne Data Server). Instead of starting the User Interface directly in your default browser as described above, it's possible to start the LabOne Web Server programs independently and then connect via a browser of your choice:

1. Start the LabOne Web Server by selecting Start Menu → Programs/All Apps → Zurich Instruments → LabOne Servers → LabOne Web Server.
2. In a web browser of your choice start the LabOne User Interface (the graphical user interface) by entering the localhost address with port 8006 to connect to the LabOne Web Server: 127.0.0.1:8006

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

Device and Settings Dialog

After starting the LabOne user interface software, a dialog is shown to select the device and settings 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 Software Architecture](#).

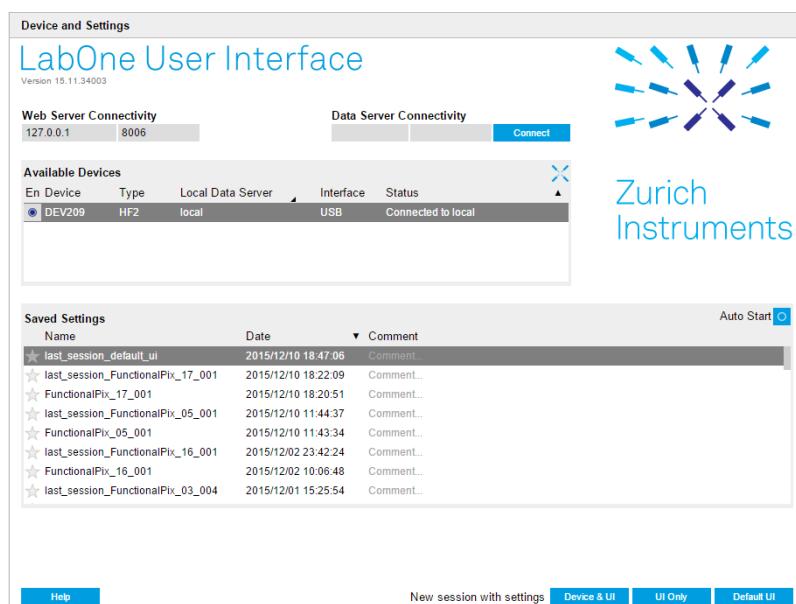


Figure 1.9. Dialog Device and Settings

The Device and Settings dialog consists of four sections: Web Server Connectivity, Data Server Connectivity, Available Devices, and Saved Settings. By default, the dialog is set to Local Data Server mode in the Available Devices section. In that case, the list of Available Devices will contain all instruments directly connected to the host PC via USB . . Once your instrument appears in the Available Devices section, perform the following steps to start a new session:

1. Select an instrument in the Available Devices list.
2. Select a setting file in the Saved Settings list unless the **Default UI** is used.
3. Start the session by clicking **Device & UI**, **UI Only**, or **Default UI**.

If there are no setting files listed, starting the LabOne User Interface by clicking the button **Default UI** will start a session using factory defaults.

Note

Opening a new session with the **Device & UI** button can affect existing sessions since the device settings are shared between them. In that case, consider using the **UI Only** button to open a new session.

Note

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

The following sections describe the functionality of the Device and Settings dialog in detail.

Data Server Connectivity

The Device and Settings dialog represents a Web Server shown under Web Server Connectivity. However, on startup the Web Server is not yet connected to a LabOne Data Server, which is why the fields under Data Server Connectivity are empty. 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 HF2 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 controlling UHF or HF2 Instruments, and also Data Servers running on an MF instrument.

In order to work with either a UHF or HF2 Instrument remotely, proceed as follows. On the computer directly connected to the HF2 (Computer 1) open a User Interface session and change the Connectivity setting in the Config tab to "From Everywhere", cf. [Section 4.13](#).

On the remote computer (Computer 2), open the Device and Settings dialog by starting up the LabOne User Interface. Change the dialog mode from **Local 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.

In case you use UHF or MF Instruments in parallel with the HF2, please also refer to the UHF or MF documentation.

Note

When using All Data Servers mode, take great care to connect to the right instrument especially in larger local networks. Always identify your instrument based on its device serial of the form DEV-xxxx which can be found on the instrument back panel.

Available Devices

The Available Devices section gives an overview of the visible devices. The first column of the list holds the **Enable** button controlling the connection between the device and a Data Server. For HF2 series instruments, this button is always greyed out since this connection is always established automatically.

The second column indicates the device serial and the third column shows the instrument type (HF2, UHF, or MFLI). The fourth column indicates shows the IP address of the LabOne Data Server controlling the device, if it is not a local one. The next column shows the interface type. For HF2 series instruments the type is always USB. The interface is listed if physically connected. The LabOne Data Server will scan for the available devices and interfaces once per second. If a device has just been switched on or physically connected it may take up to 20s before it becomes visible to the LabOne Data Server. If an interface is physically connected but not visible please read [Section 1.5.3 Device Connectivity](#). The last column indicates the status of the device which in the case of HF2 series instruments is always "Connected"

Table 1.6. 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 Enable button. Alternatively, a session can also be started directly by clicking on Device & UI , UI Only , Default UI without prior connecting. Only applies to UHF and MF Instruments.
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. Only applies to UHF Instruments.
Device needs FW upgrade	The firmware of the device is out of date. Only applies to UHF Instruments.
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 1.7](#). 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 1.7. 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 very useful to describe the specific conditions of a measurement.

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_u" 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_ui" also has special meaning. As the name suggests this file contains the default UI settings. See button description in [Table 1.8](#).

Table 1.8. Button Descriptions

Device & UI	The Device and UI settings contained in the selected settings file will be loaded.
UI Only	Only the UI settings contained in the selected settings file will be loaded. The device settings remain unchanged.
Default UI	Loads the default LabOne UI settings. The device settings remain unchanged.
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 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.

Note

The user setting files are saved to an application-specific folder in the user directory structure. On Windows, the folder can be opened in a file explorer by following the link in the Windows Start

Menu: Click and select Start Menu → Programs → Zurich Instruments → LabOne Servers → Settings.

Note

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

Double clicking on a row in the Saved Settings block is a quick way of loading the LabOne UI with those device and UI settings. This action is equivalent to selecting the desired settings file and clicking the **Device & UI** button.

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 dialog Device and Settings shown in [Figure 1.9](#). The figure below shows an example of this dialog.

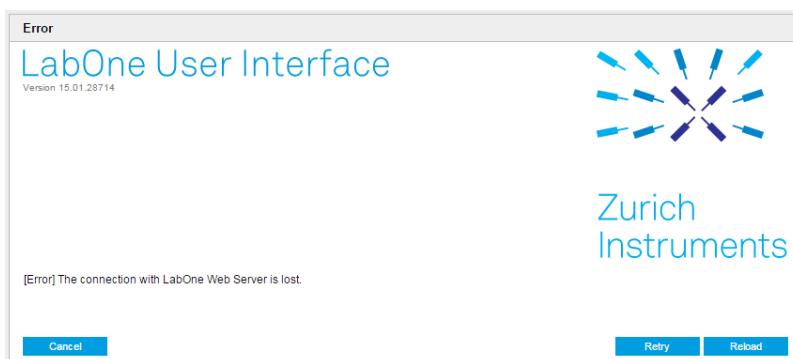


Figure 1.10. Dialog: Connection Lost

Reloading...

If a session error cannot be handled the LabOne Web Server will restart to show a new Dialog Device and Settings as shown in [the section called “Device and Settings Dialog”](#). 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.

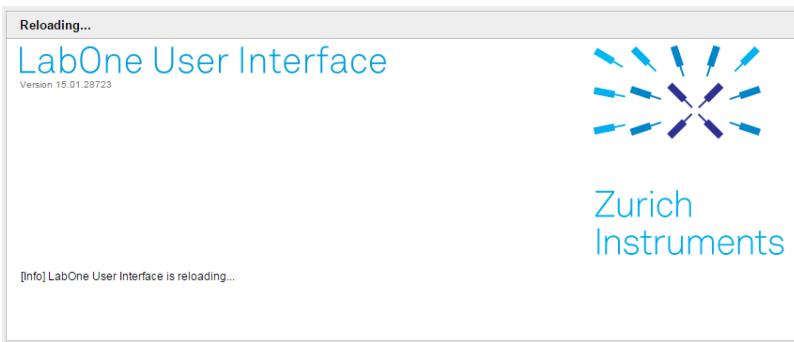


Figure 1.11. Dialog: Reloading

No Device Discovered

An empty "Available Devices" list 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 1.5.3 Device Connectivity](#). The figure below shows an example of this dialog.

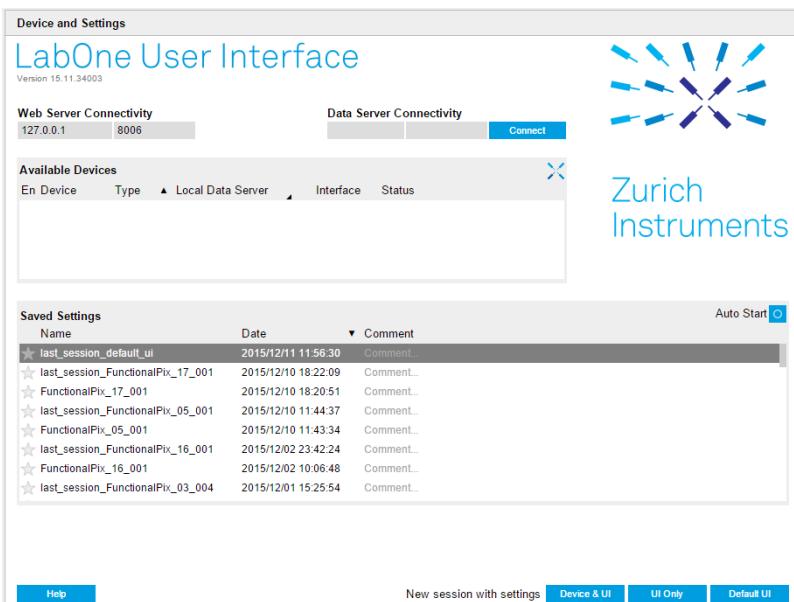


Figure 1.12. No Device Discovered

1.5.3. Device Connectivity

The device can be connected to a host computer by Universal Serial Bus (USB). The HF2 Instrument will then connect to the Data Server on the host PC. If the Data Server is not yet up and running, it is started up automatically when plugging in the USB connection. An instrument can be connected to one Data Server only, but a single Data Server and host computer can connect to several instruments at once.

If the host computer is in a local TCP/IP network, it's possible to control the device remotely and even from several computers simultaneously. Also in this remote configuration, there is a single LabOne Data Server connected to the instrument. This Data Server runs on the computer connected to the instrument via USB. The Data Server can serve one or more remote clients in

the network. All clients can access the same measurement data. Changes in instrument settings done by one client will be seen by all other clients.

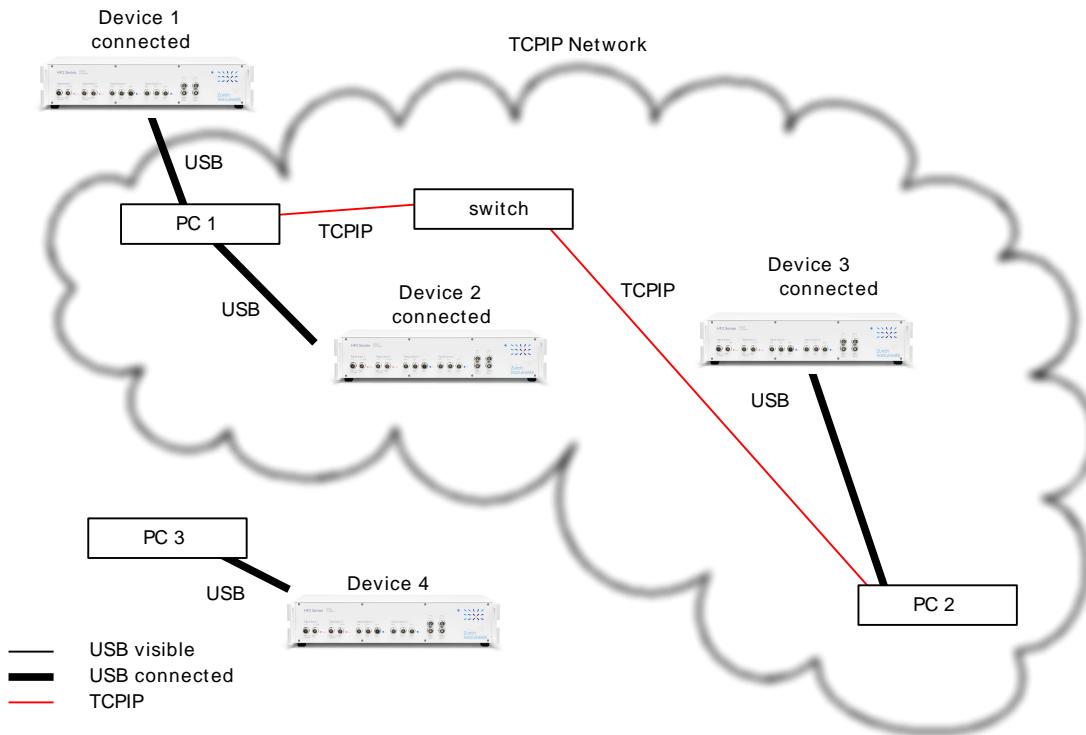


Figure 1.13. Connectivity

Figure 1.13 shows some examples of possible configurations of PC-to-device connectivity.

- Server on PC 1 is connected to device 1 (USB) and device 2 (USB).
- Server on PC 2 is connected to device 3 (USB).
- Server on PC 3 is connected to device 4.
- The devices 1, 2, and 3 are all visible to PC 1 and PC 2 over TCP/IP.

1.6. Troubleshooting

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

1.6.1. Common Problems

Your HF2 Instrument is an advanced piece of laboratory equipment with many more functionalities than a traditional lock-in amplifier. In order to benefit from these, the user needs access to a large number of settings in the LabOne User Interface. The complexity of the settings might overwhelm a first-time user, and even expert users can get surprised by certain combinations of settings. To avoid problems, it's good to use the possibility to save and load settings in the Config Tab. This allows one to keep an overview by operating the instrument based on known configurations. This section provides an easy-to-follow checklist to solve the most common mishaps.

The software cannot be installed or uninstalled: please verify you have Windows administrator rights. Windows systems: if prompted or required install the .NET Framework, see [Section 1.6.3](#).

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 HF2 Instrument turns on but delivers obviously wrong measurements: please verify the power system setting on the back panel of the device is set to the power system of your country (110 V / 60 Hz, 220 V / 50 Hz). Make sure the fuse holder is set to the correct power supply position. This means that the wanted power supply label, 230 V or 115 V, must be positioned beside the edge of the power socket (e.g. not beside the power switch).

The HF2 Instrument performs poorly in a country with 100 V supply system (e.g. Japan): if no 100 V to 110 V transformer is used, the internal power supplies might be below specifications and some circuits might perform worse than specification. Users in countries with 100 V supply system are warmly recommended to use an external transformer (delivered with the instrument).

The HF2 Instrument shows limited data throughput on USB: although the host computer requirements are not particularly demanding, highest performance in USB throughput will require a performing desktop. The USB might be limiting the data throughput, please see [Table 8.4](#) for more details. Many concurrent transfers on the USB will limit the individual transfer. In particular the Scope should be turned off when not needed by the application. The status of the USB transfer can be monitored in the Status tab.

The Instrument performs poorly in single-ended operation: the signal inputs of the instrument might be set to differential operation. Please ensure that differential input mode is turned off in the Lock-in tab or In / Out tab.

The HF2 Instrument has a high input noise floor: the USB cable connects the Instrument ground to computer ground, which might inject some unwanted noise to the measurements results. In order to decouple the computer from the Instrument consider using an electrically isolating USB range extender supporting 480 Mbit/s data transfer rate. Zurich Instruments recommends the models USB 2.0 Ranger 2201 (Icron technologies) and U-Link USB 2.0 extender (Sewell). The power supply delivered with the range extender may need to be exchanged with a more stable power supply for optimum noise performance.

The Instrument performs poorly at low frequencies (below 10 kHz) : the signal inputs of the instrument might be set to AC operation. Please verify to turn off the AC switch in the Lock-in or In / Out tab.

The Instrument performs poorly during operation: the demodulator filters might be set too wide (too much noise) or too narrow (slow response) for your application. Please verify if the demodulator filter settings match your frequency versus noise plan.

The Instrument performs poorly during operation: clipping of the input signal may be occurring. This is detectable by monitoring the red LEDs on the front panel of the instrument or the OVI flags on the status tab of the user interface. It can be avoided by adding enough margin on the input range setting (for instance 50% to 70% of the maximum signal peak).

The Instrument performs strangely when working with the HF2-MF Multi-frequency option: it is easily possible to turn on more signal generators than intended. Check the generated Signal Output with the integrated oscilloscope and check the number of simultaneously activated oscillator voltages.

The Instrument performs close to specification, but higher performance is expected: after 2 years since the last calibration, a few analog parameters are subject to drift. This may cause inaccurate measurements. Zurich Instruments recommends re-calibration of the Instrument every 2 years.

The Instrument measurements are unpredictable: please check the Status tab to see if any of the warning is occurring (red flag) or has occurred in the past (yellow flag).

The Instrument does not generate any output signal: verify that signal output switch has been activated in the Lock-in tab or In / out tab.

The Instrument locks poorly using the digital I/O as reference: make sure that the digital input signal has a high slew rate and clean level crossings.

The Instrument locks poorly using the auxiliary analog inputs as reference: the input signal amplitude might be too small. Use proper gain setting of the input channel.

The sample stream from the Instrument to the host computer is not continuous: check the communication (COM) flags in the status bar. The three flags indicate occasional sample loss, packet loss, or stall. Sample loss occurs when a sampling rate is set too high (the 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 Instrument is connected but there is no communication to the computer: check the clock fail (CF) flag in the status bar. This abnormal situation can occur if "Clk 10 MHz" is selected as Clock Source but no clock signal is fed to the Instrument. If Internal clock source is selected and the flag is still active, then the situation might indicate a serious hardware failure. In this case contact Zurich Instruments support team at <support@zhinst.com>.

The user interface does not start or starts but remains idle: verify that the LabOne Data Server and LabOne Web Server have been started and are running on your host computer.

The user interface is slow and the web browser process consumes a lot of CPU power: make sure that the hardware acceleration is enabled for the web browser that is used for LabOne. For the Windows operating system, the hardware acceleration can be enabled in Control Panel\Display\Screen Resolution. Go to Advanced Settings and then Trouble Shoot. In case you use a NVIDIA graphics card, you have to use the NVIDIA control panel. Go to Manage 3D Settings, then Program Settings and select the program that you want to customize.

1.6.2. Location of the log files

On Windows, the log files can be accessed through the start menu (All apps/all programs → Zurich Instruments → Logs).

For Windows 7, 8, and 10 the log files are located in the following directories:

- LabOne Data Server: C:\Users\[USER]\AppData\Local\Temp\Zurich Instruments\LabOne\ziDataServerLog

- LabOne Web Server: C:\Users\[USER]\AppData\Local\Temp\Zurich Instruments\LabOne\ziWebServerLog

On Windows XP:

- LabOne Data Server: C:\Documents and Settings\[USER]\Local Settings\Temp\Zurich Instruments\LabOne\ziDataServerLog
- LabOne Web Server: C:\Documents and Settings\[USER]\Local Settings\Temp\Zurich Instruments\LabOne\ziWebServerLog

1.6.3. Windows .NET Framework Requirement

The Zurich Instruments LabOne software installer requires the Microsoft .NET Framework to be installed on Windows systems. This is normally already installed on most Windows systems but may need to be additionally installed on some computers running Windows XP and Vista. If the .NET Framework is not available a message will be shown that this requirement is missing when the LabOne installer is started.

It is possible to check whether and which version of the Microsoft .NET Framework is installed on your system under Windows Start -> Control panel -> Add and Remove Programs. The minimum requirement is Microsoft .NET Framework 3.5 Service Pack 1. In case the required version is not installed, it can be installed through Windows Update tool (Windows Start -> Control panel -> Windows Update).

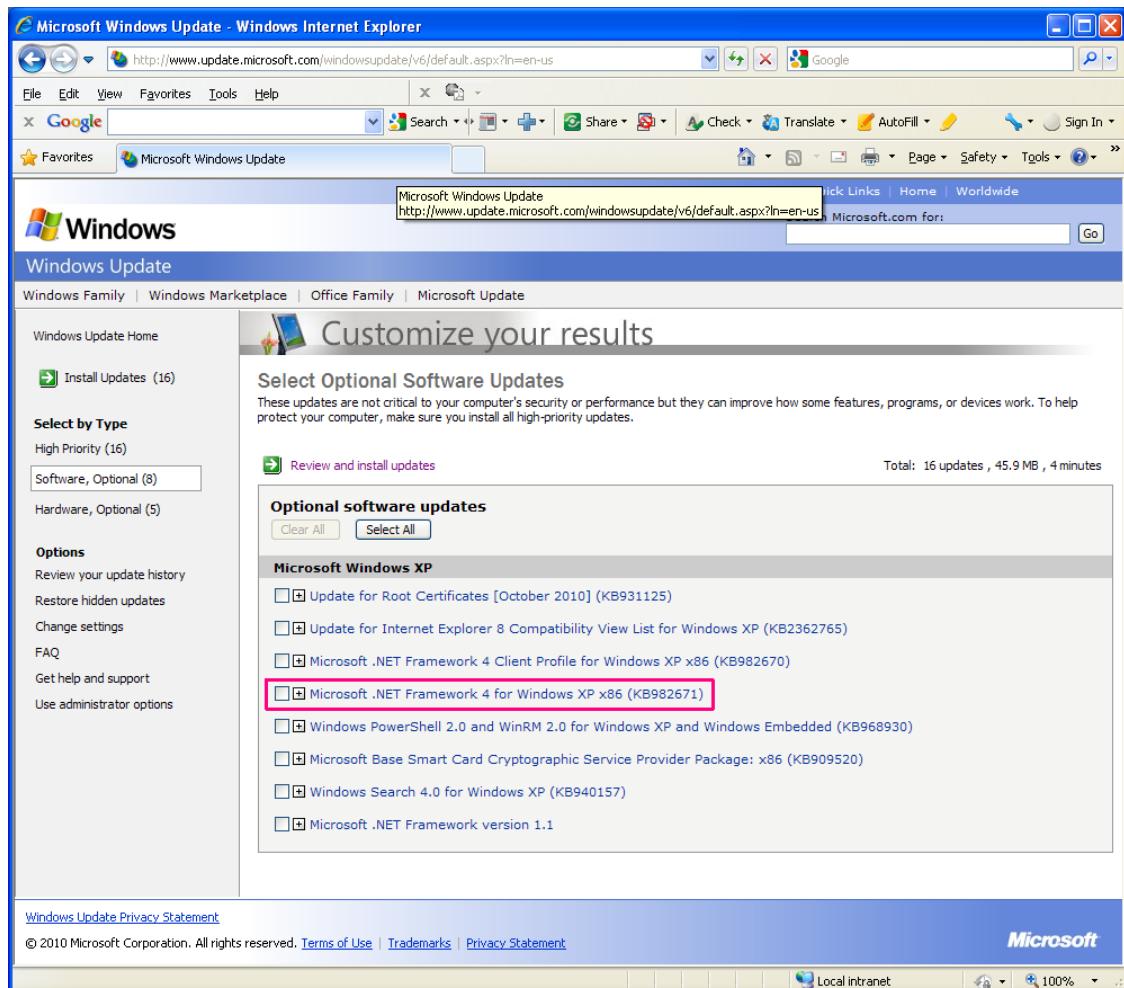


Figure 1.14. Installation of the .NET Framework.

Chapter 2. Functional Overview

This chapter helps you to quickly get acquainted with the main features, the panels, and the operating modes of the HF2 Series. A product selector is provided listing the key features of the products in order to support the selection and ordering. This section is intended as overview and therefore has a coarse level of detail without containing detailed descriptions.

2.1. Features

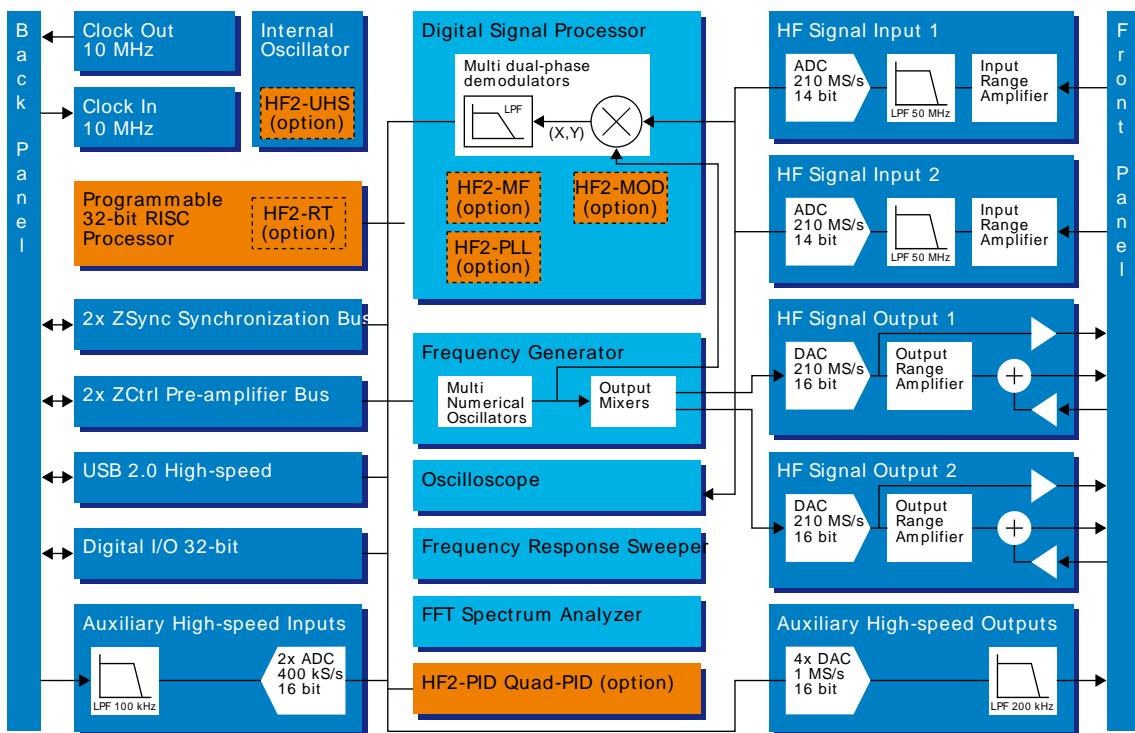


Figure 2.1. HF2 functional diagram

The HF2 Instrument as in Figure 2.1 consists of 4 high-frequency analog blocks, 2 low-frequency auxiliary blocks, the internal digital processing block (light-blue), and the hardware interfaces (mostly available on the back panel of the instrument).

The signal to be measured is connected to one of the two high-frequency analog inputs where it is amplified to a defined range, filtered, and digitized at very high speed. The resulting samples are fed into the digital signal processing block for demodulation by means of up to 8 dual-phase demodulators. The demodulators output samples flow into the embedded RISC processor for further processing or to be sent to the host computer. The samples are also sent to the auxiliary outputs in order to be available on the front panel of the HF2 Instrument.

The numerical oscillators generate sine and cosine signal pairs that are used for the demodulation of the input samples and also for the generation of the high-frequency output signals. For this purpose, the Output Mixers generate a weighted sum of the generator outputs to generate the multi-frequency signal that can be used as a stimulation signal. The 2 high-frequency output stages provide analog to digital conversion, signal scaling (range), add of an external AC or DC signal, and a synchronization signal.

Operating Modes

- Internal reference mode
- External reference mode
- Auto reference mode
- Dual-channel operation
- Dual-harmonic mode
- Multi-harmonic mode

- Arbitrary frequency mode

High-frequency Analog Inputs

- 2 low-noise high-frequency inputs
- Differential & single-ended operation (A, -B, A-B)
- Variable input range
- Variable input impedance
- AC/DC coupling

High-frequency Analog Outputs

- 2 low-noise high-frequency outputs
- Large output range
- Variable output range settings
- 1 synchronization signal for each output
- 1 adder signal for each output

Auxiliary Analog Input/Outputs

- 4 auxiliary high-speed outputs
- 2 auxiliary high-speed inputs
- User defined signal on auxiliary output

Demodulators & Reference

- Up to 8 dual-phase demodulators
- Up to 8 programmable numerical oscillators
- Programmable demodulators filters
- Very-high resolution internal reference
- 64-bit resolution demodulator outputs

Measurement Tools

- Spectroscope
- Numerical
- Oscilloscope
- Frequency response analyzer
- FFT spectrum analyzer

User-programmable Embedded Processor (Option)

- Microblaze 32-bit RISC
- 64 MHz operation allows implementation of real-time control loops
- 32-bit floating-point unit
- 64 kB internal memory (maximum program size)
- 64 MB external memory DDR2

Other Interfaces

- USB 2.0 high-speed 480 Mbit/s host interface

- DIO: 32-bit digital input-output port
- ZSync: 2 ports for inter-instrument synchronization bus (ZI proprietary)
- ZCtrl: 2 ports for control/power bus for external pre-amplifiers (ZI proprietary)
- Clock input connector (10 MHz)

Software Features

- The LabOne User Interface, a powerful browser-based graphical interface
- ziServer multi-mode multi-connection server
- ziAPI for extended programmability in C, LabVIEW, MATLAB, and Python - programming examples included
- Console: text interface to connect virtually any programming language

2.2. Front Panel Tour

The front panel BNC connectors and control LEDs are arranged in 5 sections as shown in Figure 2.2 and Figure 2.3 listed in Table 2.1. The HF2LI and HF2IS have the same connectors and connector functionality on their front- and back panel.

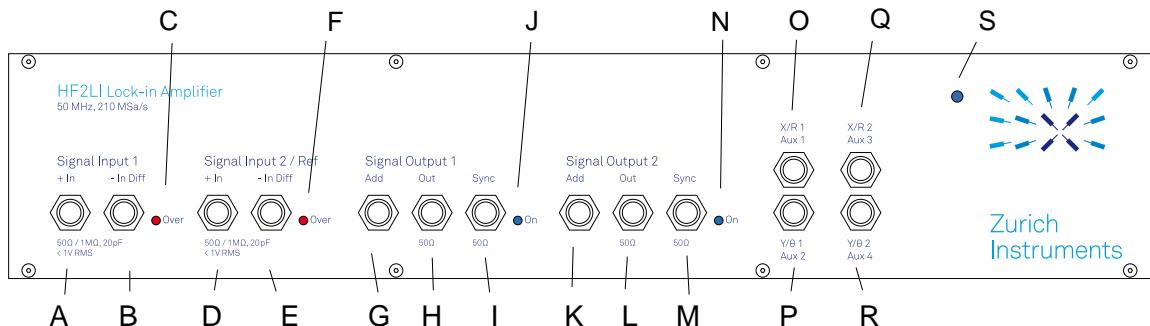


Figure 2.2. HF2LI front panel

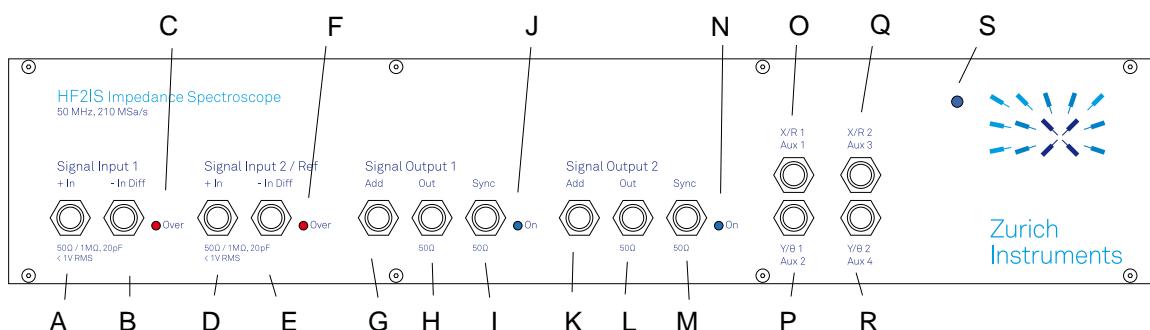


Figure 2.3. HF2IS front panel

Table 2.1. HF2 Series front panel description

Position	Label / Name	Description
A	Signal Input 1 + In	single-ended input
B	Signal Input 1 - In Diff	negative input (when not used, has to be internally shorted to ground with switch on graphical user interface)
C	Signal Input 1 Over	this LED indicates that the input signal saturates the A/D converter
D	Signal Input 2 / Ref + In	single ended input / reference input for external reference mode
E	Signal Input 2 / Ref - In Diff	negative input (when not used, has to be internally shorted to ground with switch on graphical user interface)
F	Signal Input 2 Over	this LED indicates that the input signal saturates the A/D converter
G	Signal Output 1 Add	the signal applied to the connector is added (analog add) to the output signal
H	Signal Output 1 Out	high-frequency output
I	Signal Output 1 Sync	the output signal before the output gain stage for use as synchronization or monitoring signal; the amplitude voltage

Position	Label / Name	Description
		calculates as ratio of the corresponding output amplitude and its range setting
J	Signal Output 1 On	this LED indicates that the signal output is turned on
K	Signal Output 2 Add	the signal applied to the connector is added (analog add) to the output signal
L	Signal Output 2 Out	high-frequency output
M	Signal Output 2 Sync	the output signal before the output gain stage for use as synchronization or monitoring signal; the amplitude voltage calculates as ratio of the corresponding output amplitude and its range setting
N	Signal Output 2 On	this LED indicates that the signal output is turned on
O	X/R 1 / Aux 1	this connector provides either the in-phase signal of the demodulator (X1), the magnitude (R1), or an auxiliary output signal Aux 1
P	Y/Θ 1 / Aux 2	this connector provides either the quadrature signal of the demodulator (Y1), the phase (Θ1), or an auxiliary output signal Aux 2
Q	X/R 2 / Aux 3	this connector provides either the in-phase signal of the demodulator (X2), the magnitude (R2), or an auxiliary output signal Aux 3
R	Y/Θ 2 / Aux 4	this connector provides either the quadrature signal of the demodulator (Y2), the phase (Θ2), or an auxiliary output signal Aux 4
S	Power	instrument mains power-on LED

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.

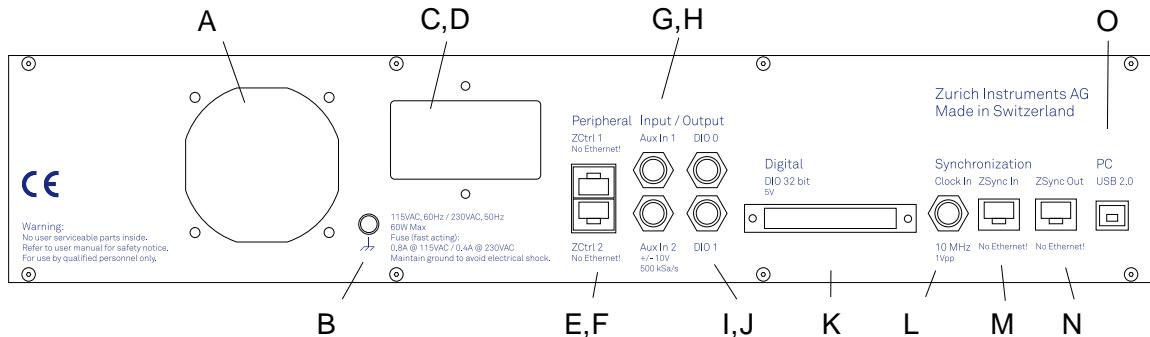


Figure 2.4. HF2 Series back panel

Table 2.2. HF2 Series back panel description

Position	Label / Name	Description
A	-	ventilator (important: keep clear from obstruction)
B	Earth ground	4 mm banana jack connector for earth ground purpose, electrically connected to the chassis and the earth pin of the power inlet
C	Power inlet	power inlet with On/Off switch
D	Power system	select between 115 V and 230 V power system
E	ZCtrl 1	peripheral pre-amplifier power & control bus 1 - attention: this is not an Ethernet plug, connection to an Ethernet network might damage the instrument
F	ZCtrl 2	peripheral pre-amplifier power & control bus 2 - attention: this is not an Ethernet plug, connection to an Ethernet network might damage the instrument
G	Aux In 1	auxiliary high-sampling rate input 1
H	Aux In 2	auxiliary high-sampling rate input 2
I	DIO 0	digital input/output 0
J	DIO 1	digital input/output 1
K	DIO	digital input/output 0-31
L	Clock In	clock input (10 MHz)
M	ZSync In	inter-instrument synchronization bus input - attention: this is not an Ethernet plug, connection to an Ethernet network might damage the instrument
N	ZSync Out	inter-instrument synchronization bus output - attention: this is not an Ethernet plug, connection to an Ethernet network might damage the instrument
O	USB	host computer connection

2.4. Ordering Guide

The HF2 Series is a product line comprising an impedance spectroscope and a digital lock-in amplifier covering advanced requirements for laboratory equipment. The HF2 Series provides best-in-class performance, wide operation range, intuitive handling and excellent accuracy.

Table 2.3 provides an overview of the available products in the HF2 Series. Upgradeable features are options that can be purchased anytime without need to send the instrument to Zurich Instruments - the upgradeable features consist of a firmware upgrade.

Table 2.3. HF2 Series product codes for ordering

Product code	Product name	Description	Upgrade in the field possible
HF2LI	HF2LI Lock-in Amplifier	base lock-in amplifier	-
HF2LI-MF	HF2LI-MF Multi-frequency	option	yes
HF2LI-RT	HF2LI-RT Real-time	option	yes
HF2LI-PLL	HF2LI-PLL Dual Phase-locked Loop	option	yes
HF2LI-PID	HF2LI-PID Quad PID Controller	option	yes
HF2LI-MOD	HF2LI-MOD AM/FM Modulation	option	yes
HF2LI-WEB	HF2LI-WEB LabOne Web Interface	option	yes
-	-	-	-
HF2PLL	HF2PLL Phase-locked Loop	bundle of the HF2LI plus the HF2LI-PLL and the HF2LI-PID options	-
-	-	-	-
HF2IS	HF2IS Impedance Spectroscopy	base impedance spectroscope	-
HF2IS-MF	HF2IS-MF Multi-frequency	option	yes
HF2IS-RT	HF2IS-RT Real-time	option	yes
HF2IS-WEB	HF2IS-WEB LabOne Web Interface	option	yes
-	-	-	-
HF2TA	HF2TA Current Amplifier	low-noise transimpedance amplifier	yes

Table 2.4. Product selector

Feature	HF2LI	HF2LI + HF2LI-MF	HF2IS	HF2IS + HF2IS-MF
Internal reference mode	yes	yes	yes	yes
External reference mode	yes	yes	-	-

Feature	HF2LI	HF2LI + HF2LI-MF	HF2IS	HF2IS + HF2IS-MF
Auto reference mode	yes	yes	-	-
Dual-channel operation (2 independent measurement units)	yes	yes	yes	yes
Sinusoidal generators	2	2	2	2
Superposed output sinusoidals per generator	1	up to 6	up to 4	up to 8
Dual-harmonic mode	yes	yes	-	-
Multi-harmonic mode	-	yes	-	-
Arbitrary frequency mode	-	yes	yes	yes
Number of demodulators	6	6	4	8
Simultaneous freq. supported (fundamentals/harmonics)	2/4	6/-	4/-	8/-
Signal input select switch matrix	-	yes	yes	yes
Oscillator select switch matrix	-	yes	-	-
50 MHz, 210 MS/s, 0.8 µs TC	yes	yes	yes	yes
DSP technology	128 bit	128 bit	128 bit	128 bit
Dynamic reserve	120 dB	120 dB	-	-
Lock-in range	50 MHz	50 MHz	-	-
USB 2.0 480 Mbit/s	yes	yes	yes	yes
LabOne User Interface, ziAPI, ziServer software	yes	yes	yes	yes
Frequency response sweeper	yes	yes	yes	yes
Oscilloscope	yes	yes	yes	yes

2.5. Operating Modes

2.5.1. Internal Reference Mode

The internal reference mode takes advantage of the internal HF generators inside the HF2 Instrument. There are 6 frequency generators in the HF2LI and up to 8 frequency generators in the HF2IS. The output of these generators are added numerically inside the instrument avoiding complicated external analog signal adders and the resulting signal is fed to the device under test. The internal reference mode is the preferred mode as the signal recovery works at its best as the generated frequency is known inside of the instrument. The signal acquisition works immediately and there is no delay lock-time.

The internal reference mode is supported with single-channel and dual-channel operation. This is possible as the HF2 Instrument includes 2 independent measurement units that are working autonomously. Each of the measurement units provides analysis for one fundamental and 2 harmonic frequencies in parallel (sometimes called dual-harmonic mode). In total, the HF2LI can measure 2 fundamental and 4 harmonic frequencies, while the HF2IS can measure 4 frequencies. The number of frequencies increases with the multi-frequency options.

The demodulator samples are available in analog format on the auxiliary outputs of the HF2 Instrument and digitally on the connected computer transferred over the USB interface. The auxiliary outputs generate an analog signal after a linear digital to analog conversion at high sample rate. There are 2 pairs of analog signals allowing to output any 2 of the demodulation sample streams. All demodulator streams are available on the computer and can be further analyzed or stored in the local drives.

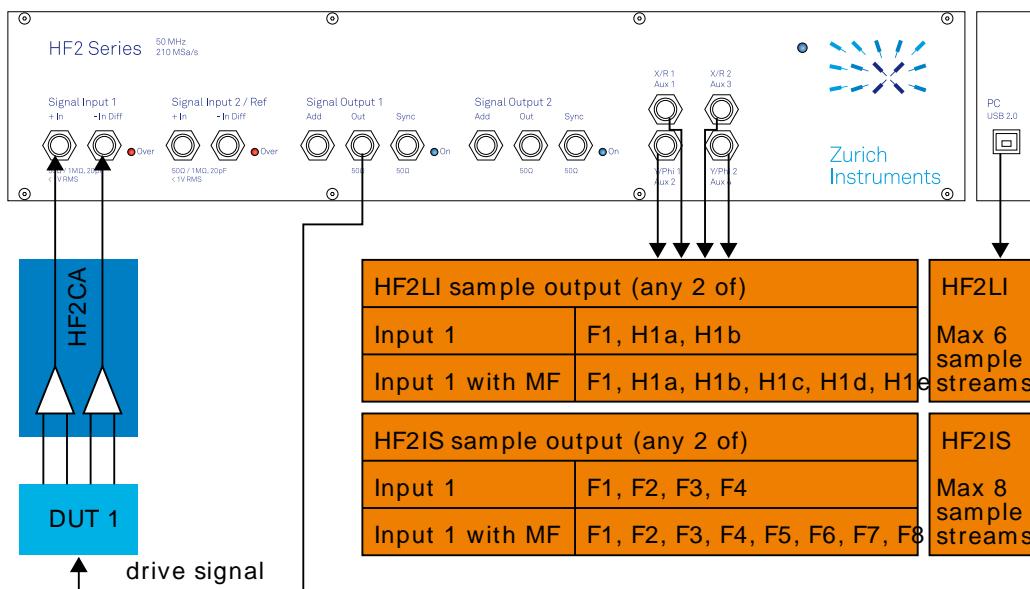


Figure 2.5. HF2 internal reference mode / single-channel

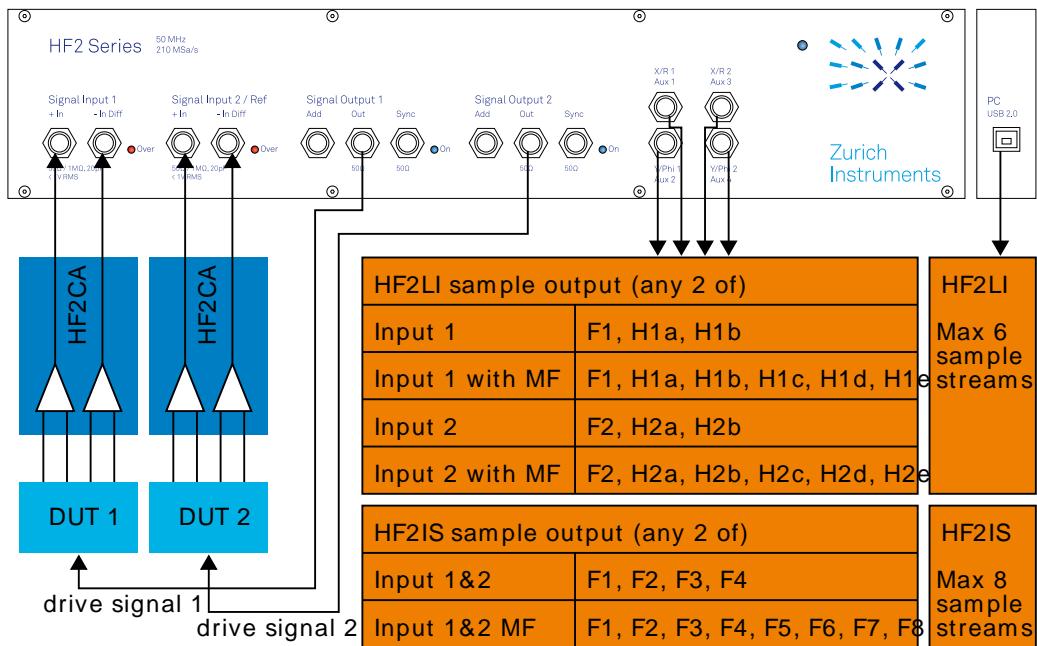


Figure 2.6. HF2 internal reference mode / dual-channel

2.5.2. External Reference Mode

The external reference mode uses external reference sources to recover the signal of interest inside the HF2 Instrument. In this mode, the internal frequency generators are not used to stimulate the DUT. As the signal reference is an arbitrary periodic signal, a certain amount of time is required for the HF2LI to lock on the reference and to be able to recover the signal of interest reliably. This lock time depends on several parameters, but most important on the level and phase noise of the reference.

The external reference mode is supported with single-channel and dual-channel operation. This is possible as the HF2 Instruments includes 2 independent measurement units that are working autonomously. In single-channel mode, the reference can be fed into the Input 2/Ref connector on the front panel. This alternative provides an unmatched capability to use references with small amplitudes as they can be amplified by the signal path of Input 2. In dual-channel operation, the external TTL references are fed into the HF2 by means of the DIO0 and DIO1 connectors on the back panel.

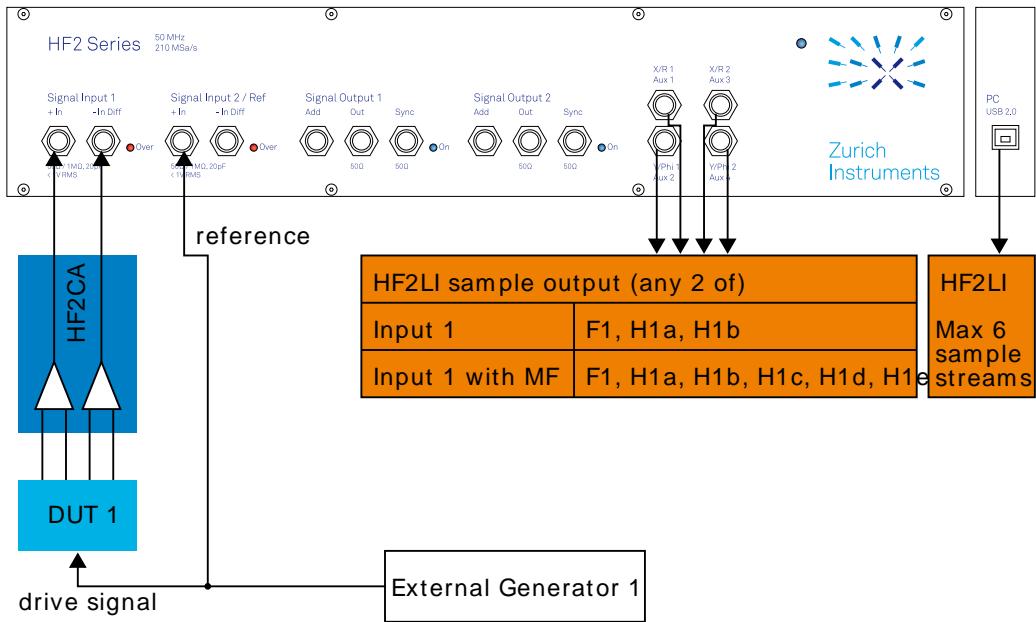


Figure 2.7. HF2 external reference mode / single-channel

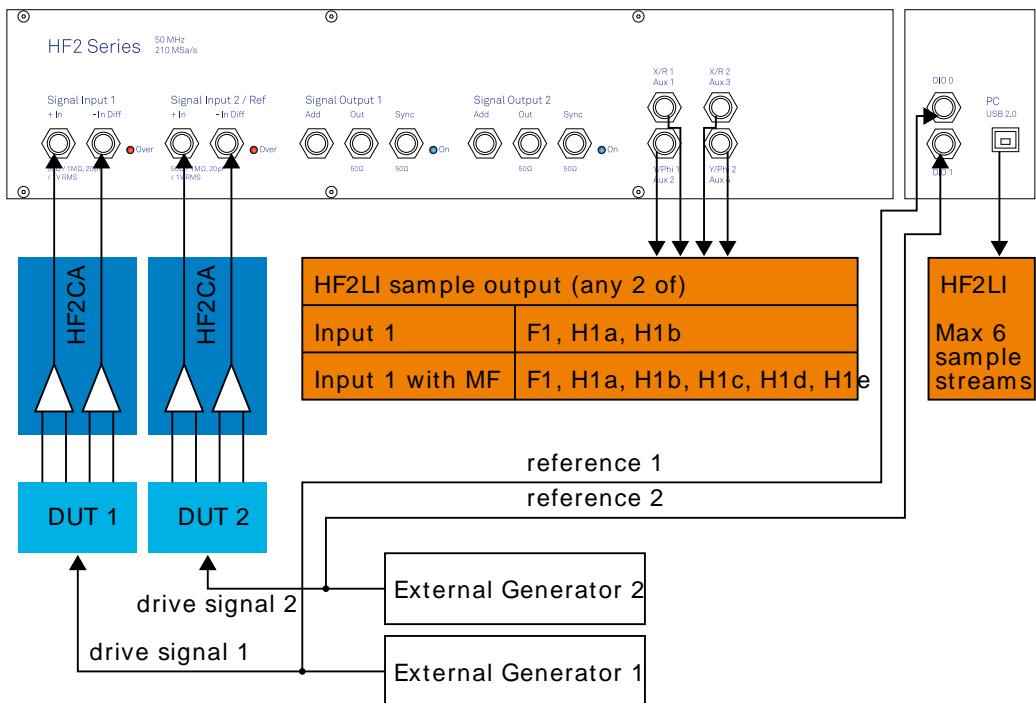


Figure 2.8. HF2 external reference mode / dual-channel

2.5.3. Auto Reference Mode

The auto reference mode makes use of the internal PLLs to recover the reference frequency directly from the signal coming from the DUT. In this mode, the internal frequency generators are not used to stimulate the DUT. As the reference is inherently contained in the sampled signal, a dedicated PLL is able to lock on the frequency and to recover the reference and the signal of interest. This process is suited for signals with enough amplitude and signal-to-noise ratio. Further the reference recovery requires a certain amount of time that depends on several parameters like the level and the phase noise of the measured signal.

The auto reference mode is supported with single-channel and dual-channel operation. This is possible as the HF2 Instrument includes 2 independent measurement units that are working autonomously. In dual-channel mode it is sufficient to connect the signals captures at the DUTs to the Input 1 and Input 2 connectors of the HF2 Instrument. The HF2 Series support both single-ended and differential input signals ideal for fixed and floating ground applications.

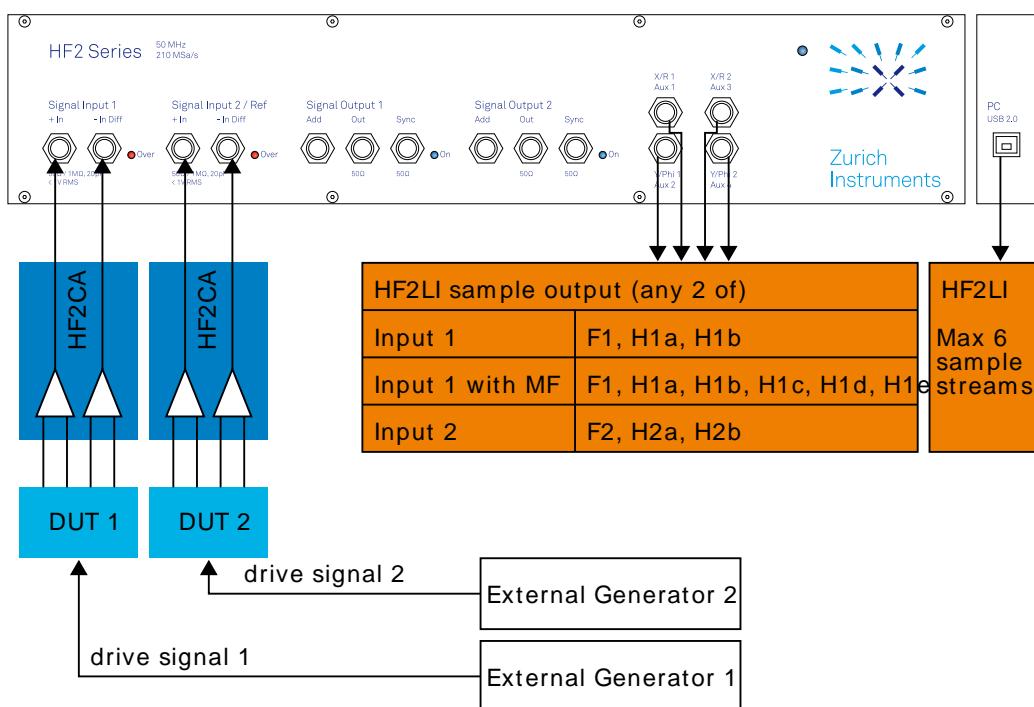


Figure 2.9. HF2 auto reference mode / dual-channel

2.5.4. Multi-frequency Operation

The multi-frequency operation is the powerful extension provided by the HF2 Series increasing the number of frequencies that can be analyzed in parallel. Moreover, the multi-frequency considerably expands the multiplexing options the user has with respect of input channels and demodulator clocks. Please note that the HF2IS-MF is different than the HF2LI-MF (see Table 2.4) as different features and different number of demodulators are activated.

For the HF2LI the multi-harmonic mode and the arbitrary frequency mode are distinguished. In multi-harmonic mode it is possible to analyze a signal at the fundamental frequency and at 5 harmonics at the same time, and the arbitrary frequency mode is the extension to analyze a signal of interest at 6 completely independent frequencies.

For the HF2IS only the arbitrary frequency mode is relevant.

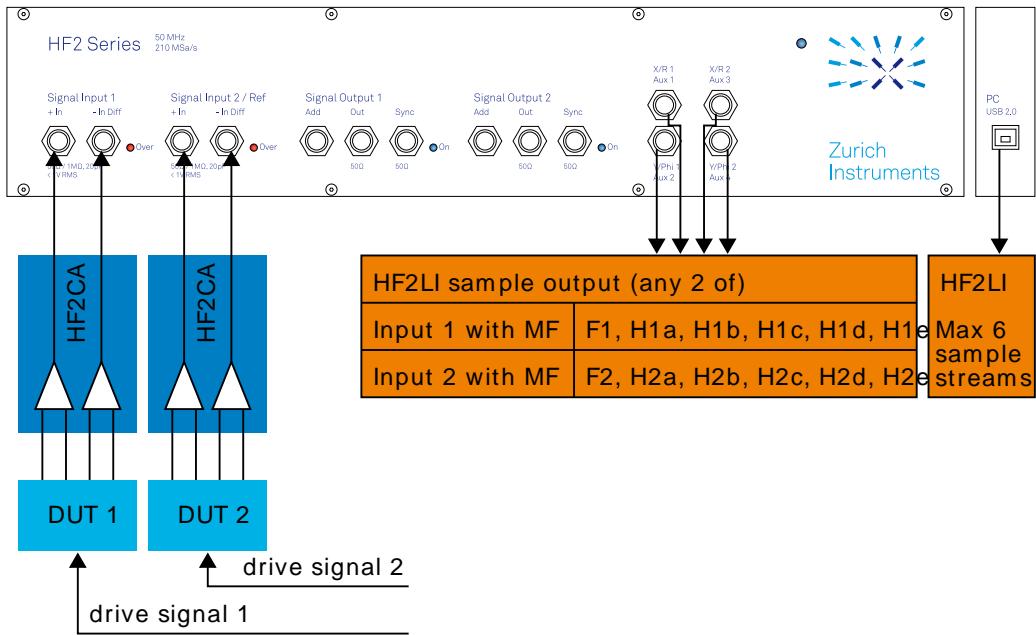


Figure 2.10. HF2 multi-harmonic mode

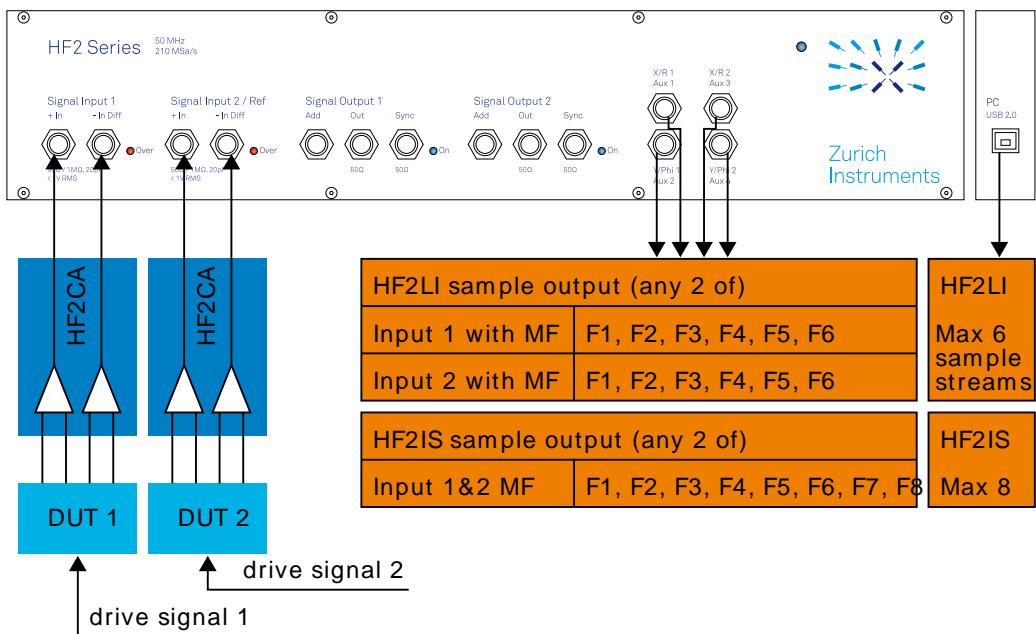


Figure 2.11. HF2 arbitrary frequency mode

Chapter 3. Tutorials

The tutorials in this chapter aim to help users perform their initial measurements with HF2 lock-in amplifiers using the LabOne User Interface. The tutorials require some basic laboratory equipment and equipment handling knowledge. For the tutorials, you'll need the following material:

- 1 USB 2.0 cable (supplied with your HF2 Instrument)
- 3 BNC cables (2 optional)
- 1 male shorting cap (optional)
- 1 oscilloscope (optional)
- 1 T-piece (optional)

Note

For all tutorials, you must have the LabOne software package installed as described in the [Getting Started Chapter](#). Start up the user interface as described in [Section 1.5](#).

3.1. Tutorial HF2LI First Time User

This tutorial covers basic operation of the HF2LI lock-in amplifier with the LabOne User Interface.

The LabOne User Interface is provided as the primary interface to the HF2LI but it is not the only program that can run the instrument. Typically, the user will use the LabOne UI to set up the instrument and then either use the LabOne UI to take the measurements or run (possibly concurrently) some custom programs.

Note

This tutorial aims to give a walk-through of the main features of the LabOne User Interface. Please also see [Section 4.1](#) for an overview of the UI's layout and [Chapter 4](#) in general for a thorough description of all the available settings available in the LabOne UI for your instrument.

3.1.1. The Lock-in Tab

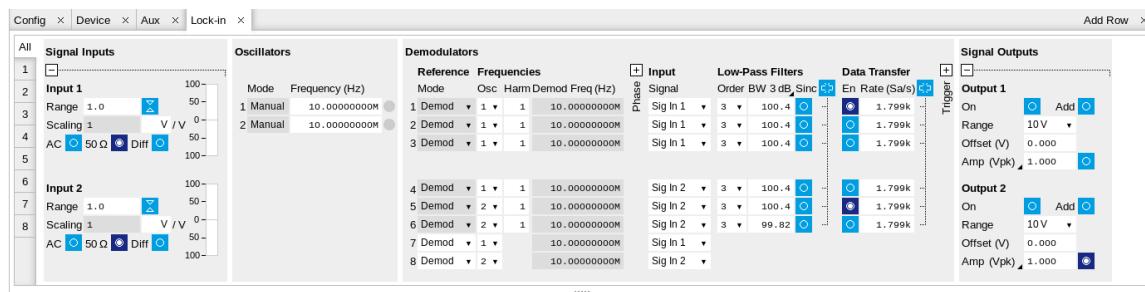


Figure 3.1. The Lock-in tab

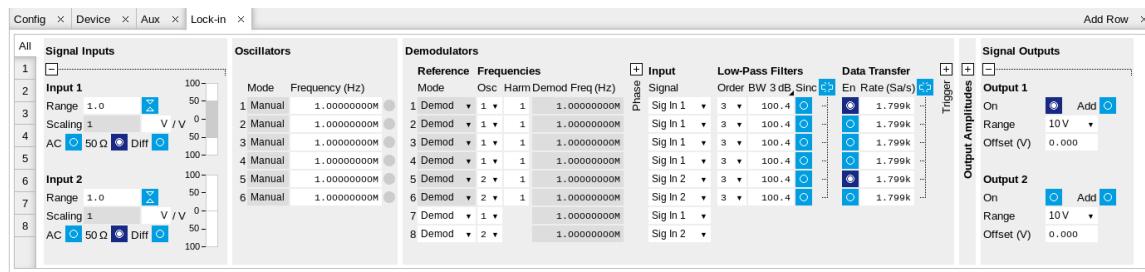


Figure 3.2. The Lock-in MF tab

Open the Lock-in tab by clicking on the Lock-in icon on the left side of the user interface. If you find the Lock-in MF tab, it means that the instrument has the HF2-MF option installed. [Section 4.2](#) provides the full documentation of the Lock-in tab while [Section 4.3](#) describes the Lock-in MF tab.

The Signal Inputs section contains a Range that can be set to a value between 1 mV and 1.6 V, the largest amplification of the input signal is achieved for 1 mV. The input has protection diodes that clip signals with amplitude above 5 V.

Important

Please respect the compliance to the maximum ratings [Table 8.2](#) to prevent damage to the instrument.

The AC button sets the coupling type: AC coupling has a cutoff frequency of 1 kHz. The AC coupling consists of a blocking capacitor between two input amplifier stages: this means that a DC signal larger than 5 V will saturate the front amplifier even if AC coupling is enabled. The Diff button sets single-ended/differential measurement mode: in the differential mode, the voltage difference between the +In and -In is amplified whereas in single-ended mode, the voltage at the +In connector is amplified. The 50 Ω button toggles the input impedance between low (50 Ω) and high (approx 1 MΩ) input impedance. 50 Ω input impedance should be selected for signal frequencies above 10 MHz to avoid artifacts generated by multiple signal reflections within the cable. With 50 Ω input impedance, one will expect a reduction of a factor of 2 in the measured signal if the signal source also has an output impedance of 50 Ω.

Next, one finds the Oscillators section used to control the frequency for the demodulation with an internal reference. For the purpose of this tutorial, set the frequency of oscillator 1 to 1 MHz.

Under the section Demodulators the user can select which harmonics and filter bandwidths to use for demodulation. It is not uncommon to need to measure different harmonics (integer multiples of the fundamental frequency, in this case 1 MHz). Select the harmonic (Harm) to 1 for the first demodulator (the first line), set the filter order to 4 (this corresponds to a filter steepness of 24 dB/oct or 80 dB/dec, an attenuation of 10^4 for a tenfold frequency increase) and type 10 Hz into the BW control (the digital filters of the HF2 are described in [Section 9.3](#)). Users are sometimes interested in the second harmonic that may be generated by nonlinear processes in their device under test: select harmonic 2 for the second demodulator and type the same values for the filter order and BW as in the previous case. You can also measure the same fundamental harmonic with a larger bandwidth: set harmonic to 1, order to 24 dB/oct and BW to 1 kHz for the third demodulator. Measuring with different bandwidths can provide the signal average and transient values. Click on the enable button next to the filters to read out the values from the 3 demodulators.

Next, set the Trigger to Continuous and the Rate to 7.20 kSa/s (rate settings can only be sub-multiples of 460 kSa/s, the maximum readout rate for one demodulator): in this case, the HF2LI will send the demodulated signal sampled at this rate through the USB. Due to the finite bandwidth of the USB connection the maximum cumulative demodulator sample rate is 700 kSa/s, which can be split over the active demodulators, see [Table 8.4](#). In this example we're using 3 active demodulators, therefore, since the sample rates are sub-multiples of 460 kSa/s the maximum possible readout rate for each demodulator is 230 kSa/s. Note that, according to the Nyquist sampling theorem, the sampling rate should be at least twice as fast as the maximum frequency present in the signal, in order to reconstruct the demodulated signal (this is not important if you only need one data point or the standard deviation of the demodulated signal). Since the low-pass filters do not have an infinite roll-off (the attenuation is not infinite past the filter's 3 dB frequency), it is common to set the sampling rate to about 8 times higher than the filter bandwidth.

Next, we configure the HF2 to output a 1 MHz signal on its Signal Output 1/Out connector. In case you have the HF2-MF Option installed, go to the Signal Outputs section, set the excitation amplitude Amp (Vpk) to 100 mV and the output range to be the smallest possible but at least twice as large as than the amplitude for minimum harmonic distortion. Connect Signal Output 1 to Signal Input 1 +In with a BNC cable and click on the On button in the LabOne Signal Outputs section of Output 1. With the HF2-MF Option installed, first go the Output Amplitudes section, set the signal amplitude Amp 1 (Vpk) of demodulator 7 to 100 mV and enable the button next to the amplitude field. Connect Signal Output 1 to Signal Input 1 +In with a BNC cable and click on the On button in the LabOne Signal Outputs section of Output 1.

3.1.2. The Numeric Tab



Figure 3.3. The Numerical tab

In the Numeric tab, you should read 71 mV RMS for the R component of demodulator 1, (demodulating at 1 MHz). The RMS corresponds to the 100 mV divided by $\sqrt{2}$. The phase value will depend on the BNC cable length (for lengths shorter than one meter, the phase is approximately a few degrees). Demodulator 3 (also at 1 MHz) will show the same amplitude, but the digits fluctuate more, since the measurement bandwidth and therefore the noise, is larger. Demodulator 2 reads only a few μ V because at 2 MHz (the second harmonic) there is only a little component of the signal, coming from the harmonic distortion of the HF2LI output and input stages.

3.1.3. The Plotter Tab

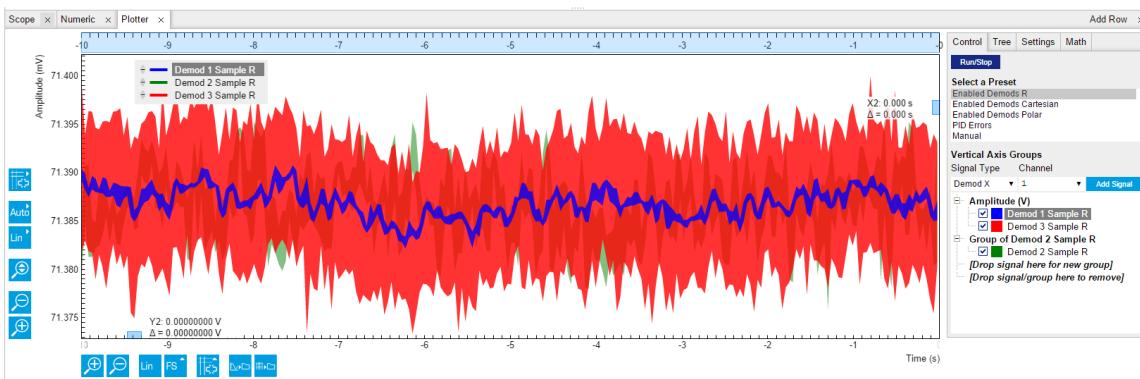


Figure 3.4. The Plotter tab

Now open the Plotter tab. Here one can display the demodulated values over time. Select Enabled Demods R from the Presets and click on Run/Stop to start the acquisition. The demodulated traces for these three demodulators are displayed, offset to one another: as before, demodulators 1 and 3 have the same average value, but a larger noise amplitude is clearly visible in the third trace. In [Section 4.1.3](#) you can find a detailed description of the functions of the plot window. For instance you can find there ways to change the horizontal and vertical scales, to remove offsets in the plot, and to use the cursors for exact measurements. The amount of stored data depends on the set Window Length in the Settings sub-tab.

3.1.4. The Scope Tab

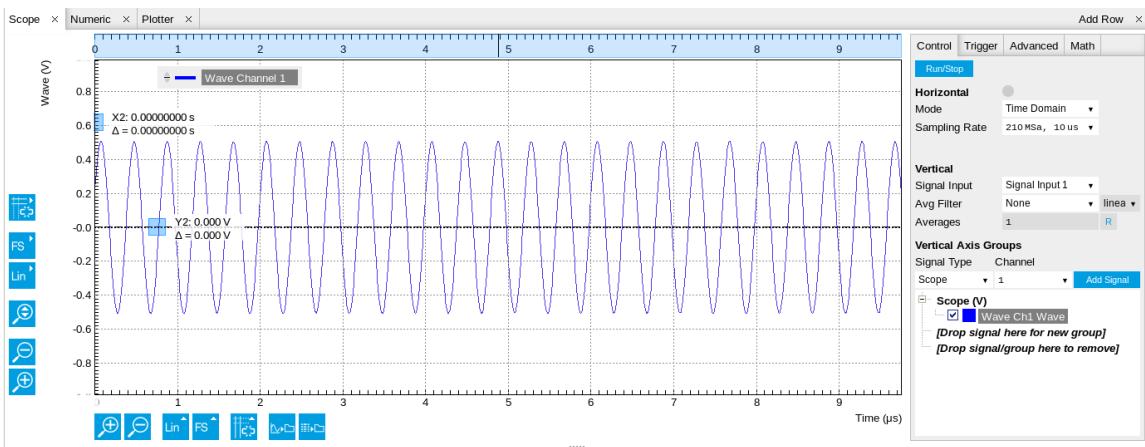


Figure 3.5. The Scope

Let us proceed to the Scope tab. The Scope can be used to display the signal at the signal inputs and outputs of the HF2LI. It has a 2048 point wave memory that is useful for visualizing the raw signal. It also replaces the need for an external oscilloscope. Select Signal Input 1 in the Vertical section of the Control sub-tab, Signal Output 1 as the Signal in the Trigger sub-tab and press the Run/Stop button. The 1 MHz input signal is visible as 10 full cycles if the Sampling Rate is set to 210 MS, 10 μ s in the Control sub-tab. Decreasing the sampling rate to display a longer time interval should be done carefully because it may lead to aliasing: for instance setting the sampling rate to 26 kSa/s, 80 ms, will produce a correctly looking sinusoidal, but at the wrong frequency. The BW Limit button in the Advanced sub-tab may reduce aliasing effects without removing them completely.

The update rate of the oscilloscope frames is controlled by the Holdoff time in the Trigger sub-tab: the minimum interval between two traces is 10 ms. This is a low value which increases the load of the USB bandwidth and may lead to USB sample loss - therefore avoid using small hold off values if not needed.

You can go from the time domain display to a frequency domain display by selecting the Freq Domain FFT Mode in the Control sub-tab. The frequency resolution is coarse because the time trace contains 2048 points. Averaging of the Fourier power spectra can be enabled to increase the SNR ratio.

3.1.5. The Sweeper Tool

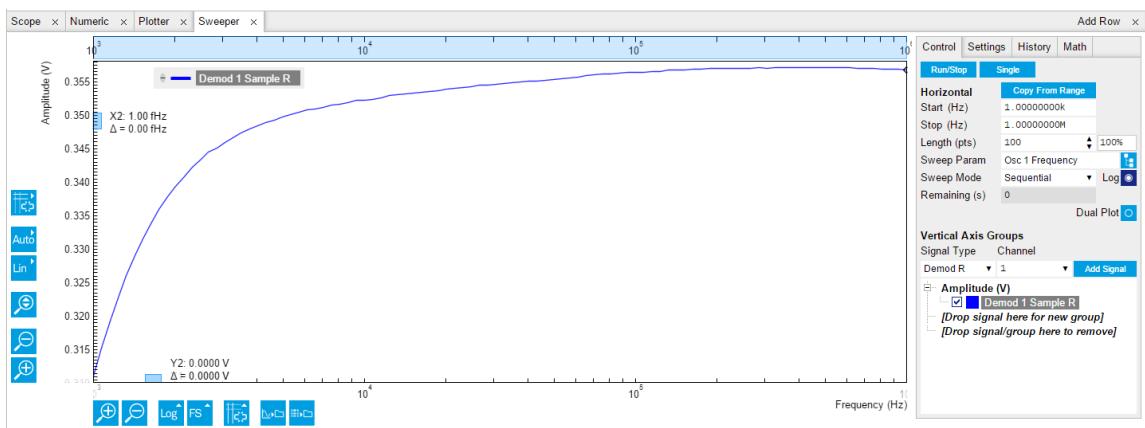


Figure 3.6. The Sweeper

Next is the Sweeper tool: it turns the HF2LI into a frequency response analyzer, giving the transfer function of a device under test in the form of a Bode plot. In AFM applications this is useful to easily identify the resonance frequency of a cantilever as well as the phase delay. The sweeper tool can also be used to sweep parameters other than frequency: phase, time constant, amplitude and auxiliary output voltage.

As a frequency sweeper example, we will execute a logarithmic sweep of 100 points between 1 kHz and 1 MHz. In the Horizontal section, set the sweep range Start to 1 kHz and Stop to 1 MHz, 100 points and enable the Log Sweep. Click on Run/Stop for continuous sweeping or on Single for a single sweep. Toggle the AC input coupling in the Lock-in settings, and observe the attenuation in the response at 1 kHz in AC coupling, since the AC coupling has a cutoff frequency of approximately 1 kHz. In the History sub-tab, the measurement that is displayed can be saved to a data file in ASCII format. There it is also possible to declare one out of several measured traces as a reference by selecting the trace in the list and clicking on Reference. The selected trace then appears below the list, and next to it there is the enable button for the reference mode. In reference mode, all traces in the plot are divided by the reference trace.

During the logarithmic sweep the NEPBW (noise equivalent power bandwidth) is adjusted for each frequency point and displayed under the Filters BW field under the Lock-in tab. The adjustment is due to the fact that the sweep is logarithmic and the sweep frequency steps are not equally spaced. In order to account for all signal power (and power densities), the measurement bandwidth must be changed accordingly. This can be done automatically by going from Application Mode to Advanced mode in the Settings sub-tab, and there selecting Auto as the Bandwidth Mode. For an explanation of the NEPBW, see [Signal Processing](#) chapter. Note that in this configuration, if the signal to noise ratio is large, there will not be any effect when disabling Auto BW, since the noise signal is negligible when measured with (almost) any NEP bandwidth. Averaging can also help to improve the signal-to-noise ratio during the sweep.

As an example of noise measurement, disconnect the BNC cable from Signal Output 1 and connect it to Signal Output 2. In the Lock-in tab, turn off the Signal Output 1, and generate a 100 kHz / 100 mV excitation Signal Output 2 (remember to turn on the output in the Signal Outputs section). In the frequency sweeper perform a single sweep with Auto BW enabled. A relatively wide peak will appear at 100 kHz, as the measurement was performed with wide NEPBW. Switch the X scaling to Manual and zoom into the region around 100 kHz; click the Copy From Range button to use the new boundaries for the sweep as selected in the graph and again perform a single sweep. The peak at 100 kHz will appear narrow, reflecting the change in the measurement bandwidth.

3.1.6. The Spectrum Tab

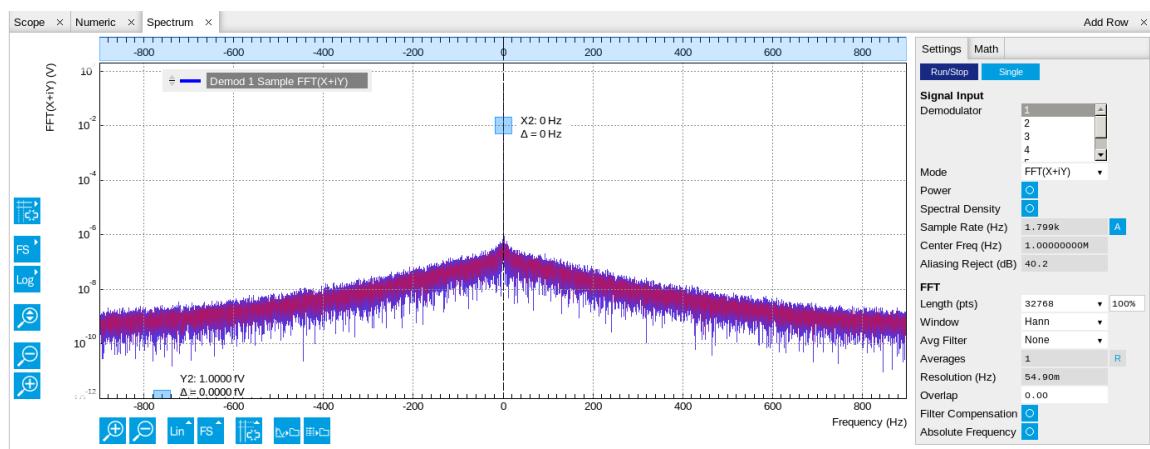


Figure 3.7. The Spectrum tab

The Spectrum tool (more information (see [Section 9.6](#)) allows the user to measure the frequency spectrum around a specific frequency: this is done by performing the Fourier transform of the

demodulated X and Y (or in-phase and quadrature) components of the signal (more precisely of the quantity $X+jY$, where j is the imaginary unit). This method is called zoomFFT. The frequency resolution that can be achieved in this way is given by the sampling rate divided by the number of recorded samples, and is therefore much higher than the frequency resolution obtained in the Scope tab. The zoomFFT approach is more efficient than the FFT on raw samples in which one digitizes a long time trace, performs the Fourier transform and retains only the portion of the frequency spectrum of interest while discarding the rest.

We continue from the previous section with the BNC connecting Signal Output 2 to Signal Input 1, and 100 mV, 100 kHz sine wave. In the Lock-in tab, set the oscillator 1 frequency to 101 kHz. Set the Demodulator 1 parameters to filter order 4, filter bandwidth 500 Hz, and Data transfer rate 7.2 kSa/s. In the Spectrum tab, enable Filter Compensation and select Demodulator 1 for Signal Input. A peak appears at 1 kHz to the left of the center frequency. Increasing the number of lines in FFT will result in a finer frequency resolution. The Filter compensation button compensates for the demodulator filter, by dividing the measured spectrum by the demodulator filter transfer function. This is why the input signal does not appear attenuated despite being outside the filter bandwidth (1 kHz and 500 Hz respectively).

3.1.7. The Auxiliary Tab

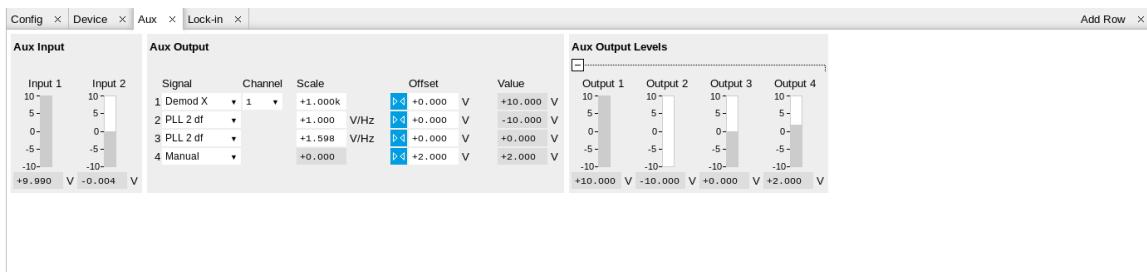


Figure 3.8. The Auxiliary Tab

The Auxiliary tab controls the 4 Auxiliary Outputs on the right side of the HF2LI front panel, as well as the 2 Auxiliary Inputs on the rear panel. On the Aux Output section of the Auxiliary tab, output the signal amplitude by selecting R from the Signal pull-down menu from Aux 1 and Demod 1 from Channel. Set the Scale factor to 10 V/V_{RMS}: you should read 0.712 V in the output Value (V) field, which corresponds to the amplitude of the signal as you can read it in the Numeric tab, multiplied by the scale factor. If one is interested in small variations of the signal amplitude, an offset can be applied to the output: type -0.712 in Offset (V) (or alternatively click the counter facing arrows next to Offset (V)): Value (V) should now read 0.

3.2. Tutorial Simple Loop

3.2.1. Preparation

In this tutorial you generate a signal with the HF2 Instrument and measure that generated signal with the same Instrument. This is done by connecting Signal Output 1/Out with Signal Input 1+/In with a BNC cable. This tutorial shows a single-ended operation, meaning that there is no signal going into the Signal Input 1-/In connector. For proper operation, the Channel 1 must be set to single-ended operation, or alternatively the Input 1 - connector must be shorted to ground using a male shorting cap. Optionally it is possible to connect the generated signal at Output 1 to an oscilloscope by using a T-piece and an additional BNC cable.

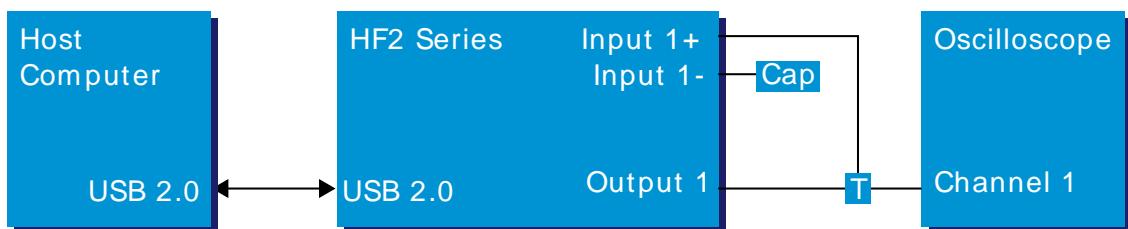


Figure 3.9. Tutorial simple loop setup

Note

This tutorial is both for HF2LI Lock-in Amplifier and HF2IS Impedance Spectroscope users.

Connect the cables as described above. Make sure the HF2 unit is powered on, and then connect the HF2 to your computer with a USB 2.0 cable. Finally launch LabOne (Start Menu/Programs/Zurich Instruments/LabOne User Interface HF2).

3.2.2. Generate the Test Signal

Apply the following settings in order to generate a 2.5 MHz signal of 0.5 V amplitude on Signal Output 1/Out.

- Open the Lock-in tab and set frequency of Oscillator 1 to 2.5 MHz: click on the field, enter "2.5 M" or "2.5E6" and press **<TAB>** on your keyboard to confirm the data
- In the Output 1 section, set the Range pull-down of 1 V
- Without HF2-MF option: In the Output 1 section, set the amplitude to 0.5 V by entering 0.5 followed by a **<TAB>**
- With HF2-MF option: In the Output Amplitudes section, set Amp 1 (Vpk) of demodulator 7 to 0.5 V by entering 0.5 followed by a **<TAB>**.
- By default all physical outputs of the HF2 are inactive to prevent damage to connected circuits. Now turn on the main output switch by clicking on the button labeled "On".
- If you have an oscilloscope connected to the setup, you are able to see your generated signal

Table 3.1. Settings: generate the test signal

Output 1 range	1 V
Oscillator 1 Frequency	2.5 MHz

Demodulator 7 Amp 1	0.5 V
Output 1	ON

3.2.3. Acquire the Test Signal

Next, you adjust the input parameters in order to acquire signals with the appropriate input range. To do this, you switch the signal source and the trigger of the Scope to Signal Input 1. Then you adjust the Signal Input 1 range to 1 V.

Table 3.2. Settings: acquire the test signal

Scope Signal Input	Signal Input 1
Scope Trigger Signal	Signal Input 1
Scope Sampling Rate	210 MS, 10 us
Run / Stop	RUN
Signal Input 1 range	1 V
Signal Input 1 AC / 50 / Diff	ON / OFF / OFF

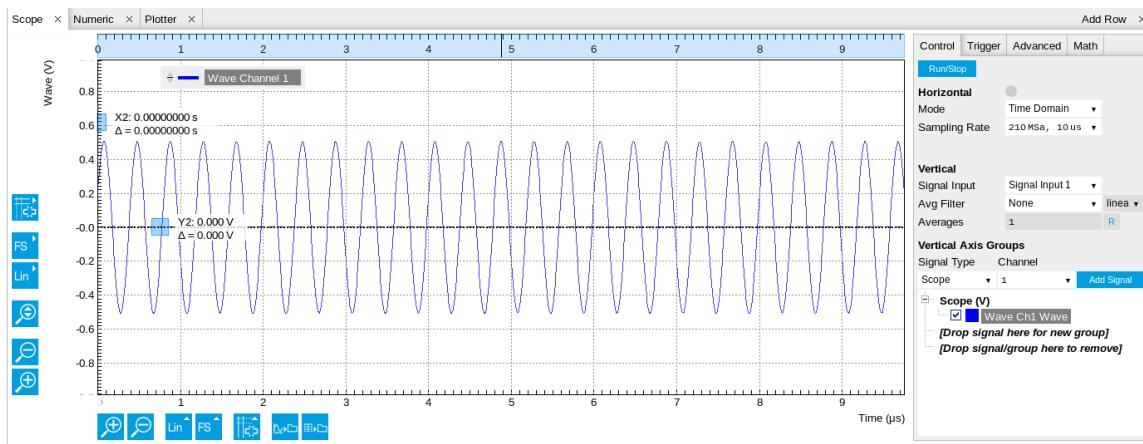


Figure 3.10. LabOne UI displaying the acquired signal

The Scope displays the measured signal at Input 1. Having set the input range to 1 V ensures that no signal clipping occurs. If you try and set the input range to 0.3 V you see the effect in the Scope window. Note how the red "Over" LEDs on the front panel of the HF2 indicates the error condition and the set OV (OVI) status flag on the right-bottom corner of the window. Set back the input range to 1 V and then clear the flag by clicking .

The Scope is a very handy tool to quickly check settings before proceeding to more advanced measurements, please refer to [Section 4.6](#) for a full description of features available in the Scope.

3.2.4. Measure the Test Signal

Next you use the demodulators of the HF2 to measure acquired test signal. You will use the Numeric and the Plotter tab from the LabOne user interface. First apply the following settings (choose any of the demodulators 1 to 6).

Table 3.3. Settings: measure the test signal

Filter BW 3 dB	7 Hz (approximated to 6.8 Hz)
----------------	-------------------------------

Filter order	2
Data transfer rate	100 Hz (approximated to 112 Hz)
Data transfer enable (En)	ON

These settings set the demodulation filter to second-order low-pass operation with a 7 Hz bandwidth. The corresponding time constant can be obtained easily by clicking on the label on top of the bandwidth setting column according to [Equation 9.3](#) provided in [Chapter 9](#). The output of the demodulator filter is read out with 100 Hz, implying that 100 data samples are sent to the host PC per second. These samples are viewed in the Numeric and Plotter tab that we examine next.

The Numeric tool provides the space for 6 measurement panels corresponding to the 6 demodulators. Each of the panels has the option to display the samples in Cartesian (X,Y) or polar format (R,Θ). The unit of the (X,Y,R) values is V_{RMS} .

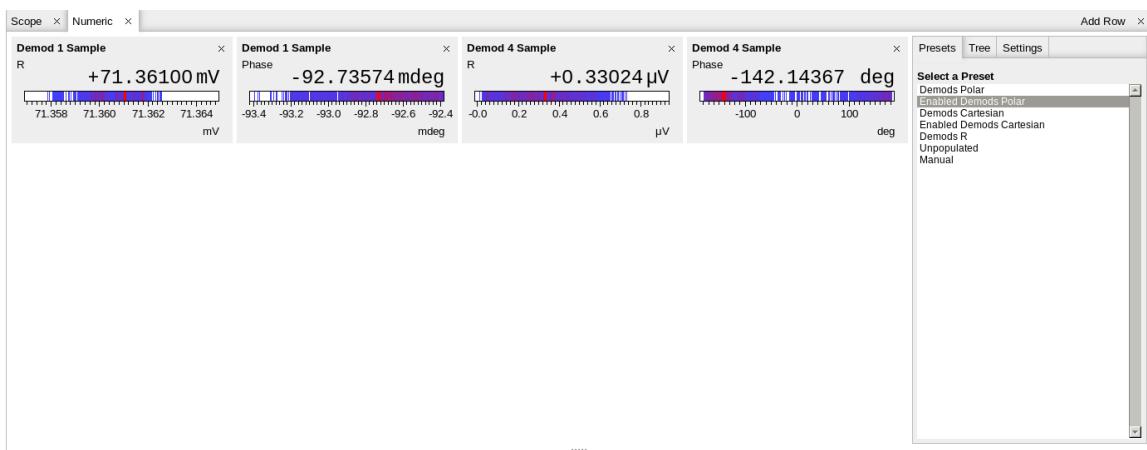


Figure 3.11. LabOne UI Numeric tab

If you wish to play around with the settings, you could now change the amplitude of the generated signal, and observe the effect on the demodulator output.

Next, we'll have a look at the tab. This tab provides a time plot of the demodulator outputs. It is possible to plot up to 6 signals continuously as (X,Y) or (R,Θ) pairs, to set different scales, or to make detailed measurements with 2 cursors. For a detailed description of the functionality available in the Plotter please see [Section 4.5](#) and [Section 4.1.3](#).

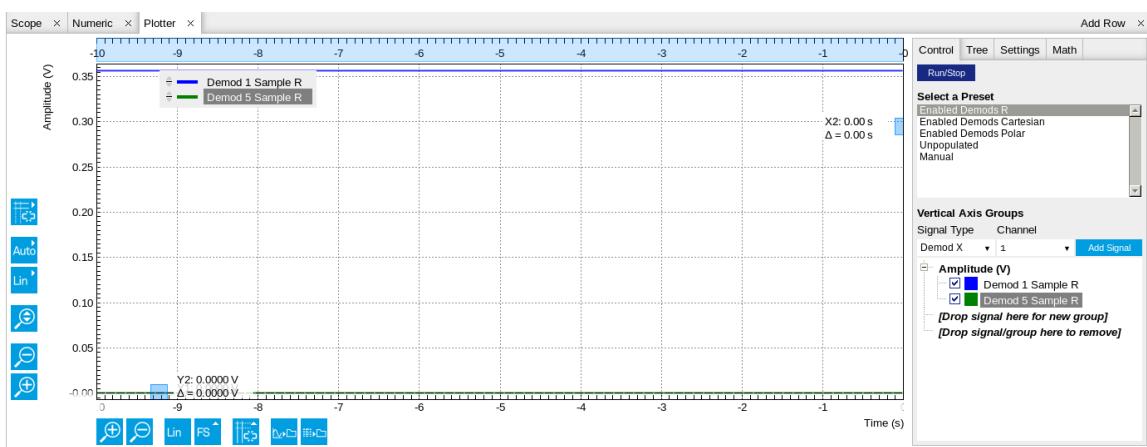


Figure 3.12. LabOne Plotter view with plot of demodulator 1 output (TC = 10 ms)

3.2.5. Different Filter Settings

As last step of this tutorial you change the filter settings and see the effect on the measurement results. For instance you change the time constant of the integration to 2 seconds.

Table 3.4. Settings: changing the filter settings

Time constant (TC)	2 s
Filter slope	12 dB/Oct
Resulting measurement bandwidth (BW)	~51 mHz

Increasing the time constant increases the integration time of the demodulators smoothing out the demodulator outputs. This averages the noise over time and the output of the filters is more stable. This manifests itself in a smoother curve of the demodulator data but also in a larger number of stable digits in the Numeric tab.

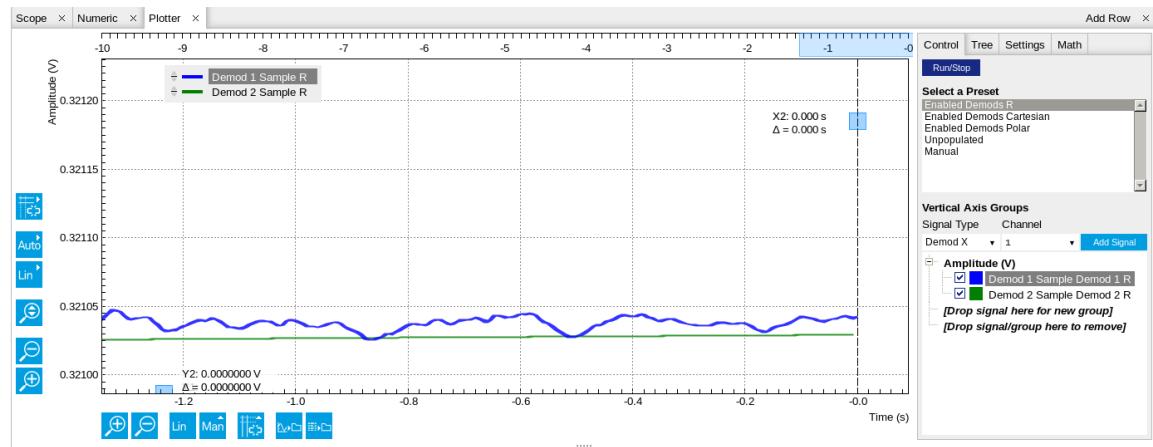


Figure 3.13. LabOne Plotter displaying the data of two demodulators with different time constants

3.3. Tutorial Dynamic Signals

3.3.1. Preparation

In this tutorial we generate a test signal of 2.5 MHz with an amplitude modulated at a frequency of 1 Hz. Then we measure the test signal using two different filter settings.

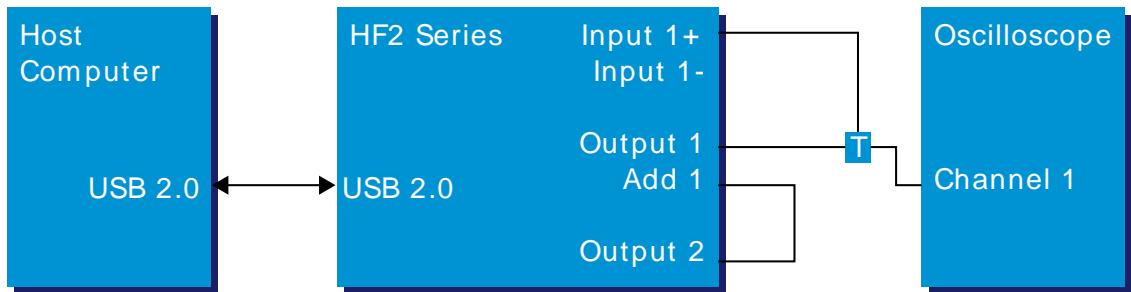


Figure 3.14. Tutorial dynamic signals setup

Note

This tutorial can be performed both on the HF2LI Lock-in Amplifier and on the HF2IS Impedance Spectroscopy and will use the Input connector. The generation of multi-frequency signals is simple on the HF2LI with the HF2-MF option or on the HF2IS, where there is no need to make use of the ADD connector.

Connect the cables as described above. Make sure the HF2 unit is powered on, and then connect the HF2 to your computer with a USB 2.0 cable. Finally launch LabOne (Start Menu/Programs/Zurich Instruments/LabOne User Interface).

3.3.2. Generate the Test Signal

In this section you generate a 2.5 MHz sinusoidal signal whose amplitude oscillates at 1 Hz. This is also called the beat signal. In order to obtain this test signal you add two sinusoids of the same amplitude but with a 1 Hz difference in the frequency.

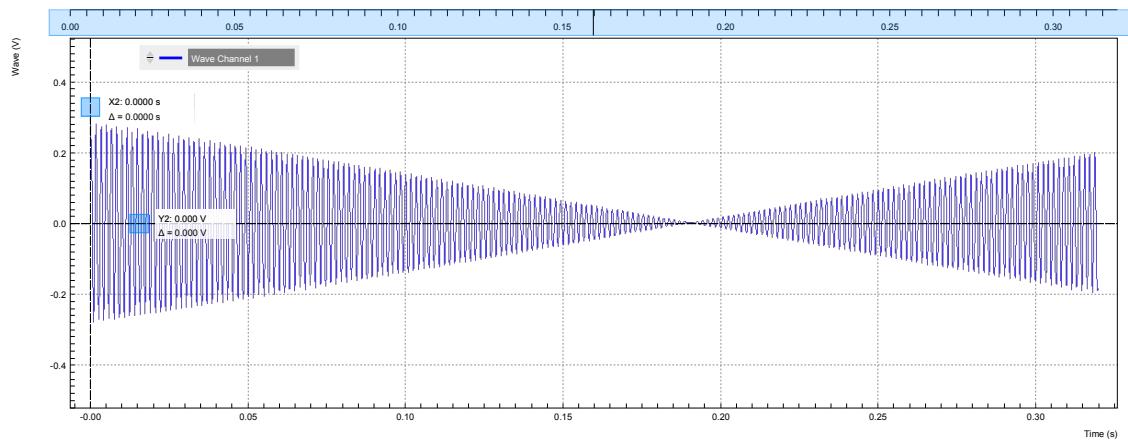
Table 3.5. Settings: generate the test signal

Output 1 range	1 V / ON
Output 2 range	1 V / ON
Oscillator 1 frequency	2'500'000 Hz
Demodulator 7 Amp 1	0.3 V / ON
Oscillator 2 frequency	2'500'001 Hz
Demodulator 8 Amp 2	0.3 V / ON
Signal Output 1 Add	ON

When connecting an oscilloscope to the Output 1 connector, you should be able to observe the superposition of the 2 sinusoids. To see the acquired signal inside the LabOne User Interface switch to the Scope tab. The Scope view looks like this with the following settings.

Table 3.6. Settings: acquire the test signal

Scope Signal input	Signal Input 1
Scope Trigger signal	Signal Input 1
Scope sampling rate	6.4 kSa, 320 ms
Run/Stop	Run
Signal Input 1 range	1 V
Signal Input 1 AC / 50 / Diff	ON / OFF / OFF

**Figure 3.15. LabOne UI displaying the acquired signal**

The beat signal has a maximum amplitude of 0.6 V, thus it falls within the set range of 1 V. The range setting will prevent any higher voltage than what is set - even if 2 sinusoids of 0.7 V amplitude each would be added like done in this section, the output would be clipping at 1 V which is the set range. Try to change the output range to 0.1 V, and see how the output voltage is changed to prevent inconsistent settings.

3.3.3. Measure the Test Signal

First you change to the Plotter tab, set the scale in order to view an interesting set of data, and set the demodulator filters to a low time constant to measure the amplitude of the 2.5 MHz signal (Hull curve).

Table 3.7. Settings: filter with a low time constant

Time constant (TC)	10 ms (approximated to 10.2 ms)
Filter order	2
Resulting measurement bandwidth (BW 3dB)	~10 Hz
Data transfer rate	100 Hz (approximated to 112 Hz)
Data transfer enable (En)	ON

These settings set the demodulation low-pass filter to a 10 ms time constant (the corresponding bandwidth is around 10 Hz) and the filter slope to second order. The output of the filter is sampled at a rate of 100 Hz, and the samples are sent to the host computer.

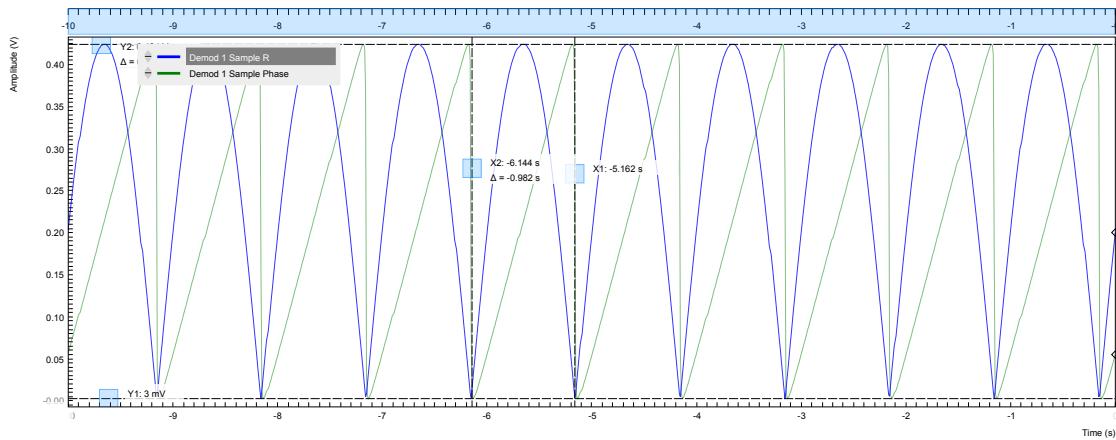


Figure 3.16. LabOne UI displaying the measured signal

If you stop the acquisition by pressing the button "Run/Stop" you can conveniently measure the amplitude of the 1 Hz signal by using the 2 cursors C1 and C2: 394.4 mV_{RMS}, half period 498.1 ms. You can achieve higher measurement precision by using an even lower time constant (e.g. 1 ms), increasing the readout rate (e.g. 1.8 kHz), and zooming into the Plotter view.

Next you use a high time constant to separate the 2 original sinusoids even though they are superposed in one signal. In the Lock-in tab apply the following settings.

Table 3.8. Settings: filter with a high time constant

Time constant (TC)	2 s
Filter order	2
Resulting measurement bandwidth (BW 3dB)	~35 mHz
Data transfer rate	100 Hz (approximated to 112 Hz)
Data transfer enable (En)	ON

These settings set the demodulation low-pass filter to a time constant of 2 s, with a resulting measurement bandwidth of 35 mHz. With these settings the HF2 is able to distinguish between the signal component at 2'500'000 Hz and the signal component at 2'500'001 Hz as the measurement bandwidth is considerably less than the frequency spacing of the 2 signal components. The output of the demodulator is stable after a settling time.

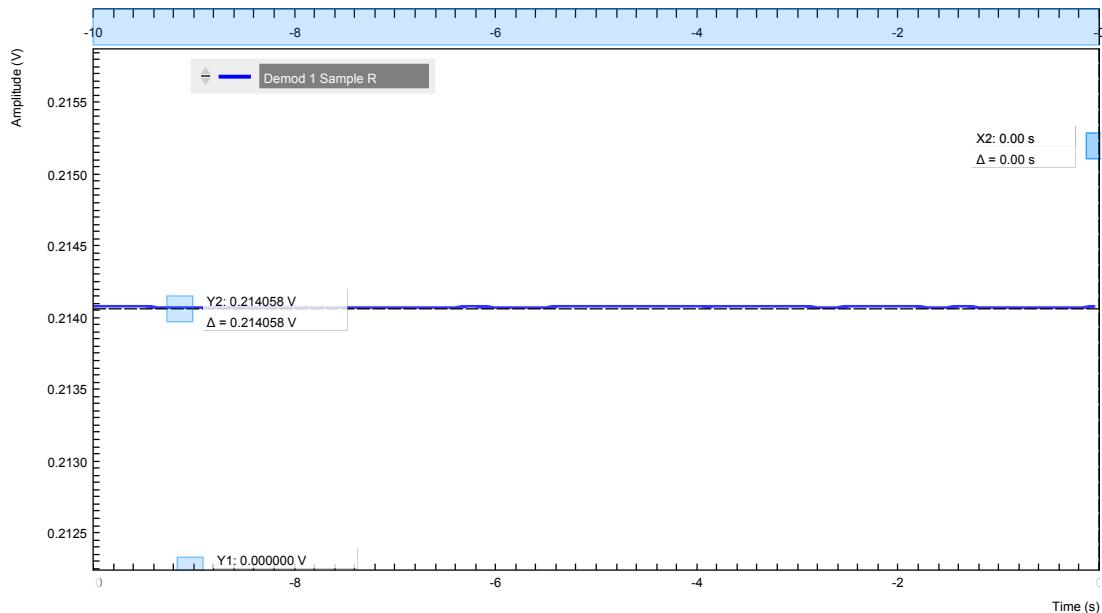


Figure 3.17. LabOne UI displaying the measured signal

The output of the demodulator does not show any oscillations like before: the numerical value is mV_{RMS} . If you switch to the oscilloscope view, you see that the signal at Input 1 is still beating as before, while the demodulator filter is set such to ignore the interferer at 2'500'001 Hz. Try to switch off the interferer.

Table 3.9. Settings: remove the interferer

Signal output 2 ON/OFF	OFF
------------------------	-----

The time it takes to settle depends on various parameters like filter setting and switch-off timing. The difference in amplitude of the measurement at 2'500'000 Hz with or without interferer is in the range of 50 μ V. With different filter settings it is possible to do better than that.

Consider this: you have 2 signals with relevant amplitude (0.3 V) interfering with each other as their frequency is very close (1 Hz at 2.5 MHz). The power of lock-in amplification consists of extracting the relevant signal energy at exactly one frequency. The "immunity" from nearby interferer is the capability to ignore them. This a simple definition of the dynamic reserve.

3.3.4. Filter Setting Discussion

This section aims to summarize a few basic concepts of filtering in connection with lock-in amplification. In this tutorial, you have used different filter settings to measure different signal properties.

Table 3.10. Settings: filter with a high time constant

Time constant	Measurement bandwidth	Measurement noise	Changes upon steady state change	Example
Low setting	High, capability to detect fast events	More noise in measurement result	Fast adaptation of result	BW = 10 kHz, capability to detect events at 2-5 kHz, prone to pick-up noise

Time constant	Measurement bandwidth	Measurement noise	Changes upon steady state change	Example
High setting	Low, capability to determine the steady state	Less noise in measurement result	Slow adaptation of result	BW = 50 mHz, exact determination of steady state - events more frequent than 0.1 Hz are filtered

Filtering constitutes a trade-off between measurement speed and measurement accuracy. In order to measure fast events, it is necessary to open up the filters allowing also more noise in the measurement result. The opposite is to measure with narrow filters which increases the signal-to-noise ratio, but limits the capability to detect the changes in the signal of interest. This trade-off is in common with any lock-in amplifier. The power of the HF2 is that it allows to do both at the same time thanks to the multiple demodulators per input channel.

3.4. Tutorial External Reference

3.4.1. Preparation

In this tutorial we generate a test signal with the HF2 and use it as a reference signal for demodulation in the same way as we would do it with a reference signal coming from an external source.

This is done by connecting the Output 2 connector with the Input 2+ connector with a BNC cable. This tutorial shows a single-ended operation, meaning that there is no signal going into the Input 2- connector. Optionally, it is possible to connect the generated signal from Output 2 to an oscilloscope by using a T-piece and an additional BNC cable. The Output 1 connector is to be connected to the Input 1+ connector. This allows you to check the generated reference signal. The measurement setup is shown in the following figure.

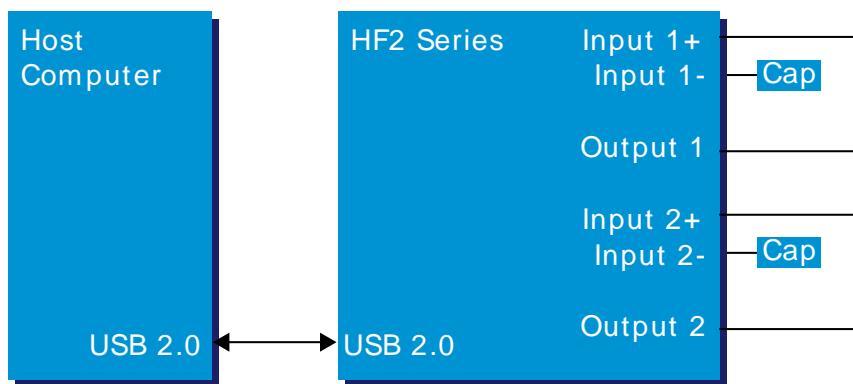


Figure 3.18. Tutorial external reference setup

Make sure the HF2 unit is powered on, and then connect the HF2 to your computer with a USB 2.0 cable. Finally, launch LabOne (Start Menu/Programs/Zurich Instruments/LabOne User Interface HF2).

3.4.2. Generate the Reference Signal

In this section you generate a 1.0 MHz signal with a 1 V amplitude on Output 2 for use as the external reference. The settings for generating the reference signal are shown in the following table.

Table 3.11. Settings: generate the reference signal

Output 2 range	1 V
Demodulator 8 Amplitude 2	1.0 V / ON
Oscillator 2 frequency	1 MHz
Signal input 2 range / AC / Diff / 50	1.2 V / ON / OFF / OFF

When connecting an oscilloscope to the Output 2 connector, you should be able to observe the sinusoid. Alternatively, you can look at the signal in the LabOne UI Scope with the following settings.

Table 3.12. Settings: acquire the reference signal

Scope Signal Input	Signal Input 2
--------------------	----------------

Scope Trigger Signal	Signal Input 2
Scope Sampling rate	53 MSa, 39 us

Note

In the Scope tab set the Horizontal Mode to Freq Domain FFT in order to see the frequency spectrum of the signal. This will also set a logarithmic Y scale by default. An Exponential Moving Average can be enabled in the Avg Filter field to reduce the noise floor in the display.

3.4.3. Activate the External Reference Mode

In this section we activate the external reference mode. Based on the external reference, we demodulate a separate signal of interest.

Table 3.13. Settings: generate the signal of interest

Output 1 range	1 V
Demodulator 7 Amp 1	1.0 V / ON
Oscillator 1 frequency	1 MHz
Demodulator 1 En/Rate	ON / 100 Hz
Signal Input 1 range / AC / 50 / Diff	1.3 V / ON / OFF / OFF

The external reference mode makes use of demodulators 7 and 8. These two demodulators can not be used for measurement. They serve as phase sensitive detectors to set up phase-locked loops locking an external reference and an internal oscillator. Demodulator 7 is assigned to oscillator 1, and demodulator 8 is assigned to oscillator 2. Previously we have chosen oscillator 2 as the source of the reference signal on Signal Output 2. Now we want lock oscillator 1 to this reference. In order to achieve that, we set the Mode field of demodulator 7 to ExtRef. In the pull-down selector in the Input column of demodulator 7, select Signal Input 2.

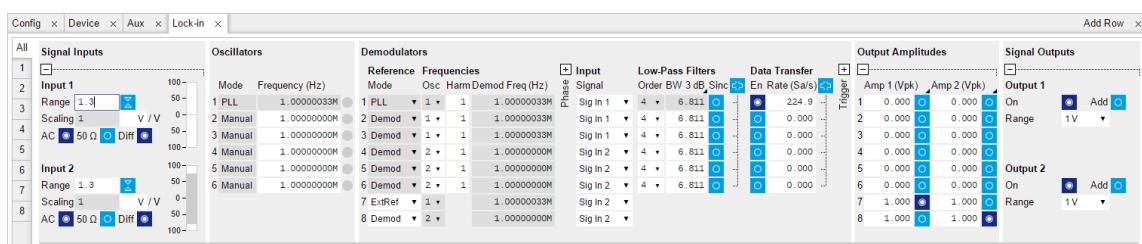


Figure 3.19. LabOne UI enabling external reference mode

Once the external reference mode has been enabled, the frequency of oscillator 1 changes continuously, adapting to the frequency of the external reference signal. This can be verified by changing the frequency of oscillator 2 and noting how the frequency of oscillator 1 follows. A green light next to the oscillator frequency field indicates that the instrument has locked to an external reference.

In the demodulation process, the measurement signal is not multiplied directly with the external reference signal. Instead, the measurement signal is multiplied with newly generated reference signal from the internal oscillator, using only the frequency and phase information of the external reference. The continuous toggling of the oscillator frequency shows that the newly generated reference is continuously adjusted to the external reference.

3.4.4. Change External Reference Input

In this section you will modify the setup to use DIO 0 as the external reference instead of Signal Input 2. This is useful in practice since it means that the two sensitive Signal Inputs of the Instrument remain available for actual measurements. The modified setup is shown on Figure 3.20. Note that the DIO 0 connector is located on the back panel of the HF2 Instrument.

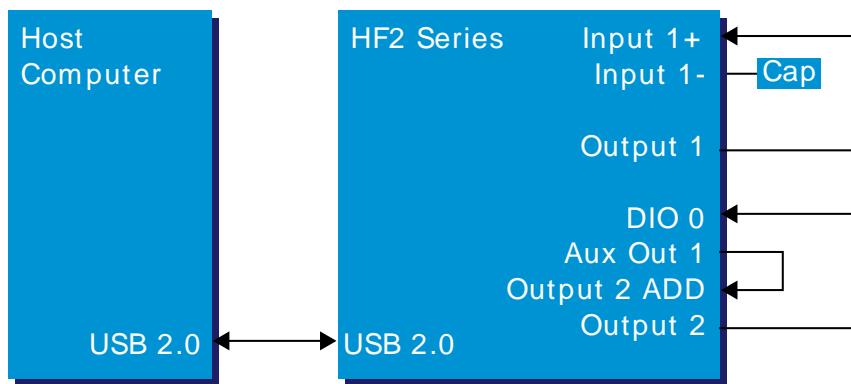


Figure 3.20. Tutorial external reference using DIO 0 setup

It is important to make sure that DIO 0 is configured as an input before connecting anything to it. This can be done using the DIO tab in LabOne. Note that the button to the right of **bits 7...0** should be off.

When using the DIO 0 as the external reference signal, it should be taken into account that this is a digital I/O, which should be operated at TTL levels. Therefore the Aux 1 output is connected to the Add connector of Output 2, to provide a DC shift of the test signal and thus make it TTL compatible.

The settings used for generating the test signal are shown in the following tables. The resulting signal will have a DC offset of 1.5 V and an amplitude of 1 V and will thus oscillate between 0.5 V and 2.5 V, which is TTL compatible.

Table 3.14. Settings: generate the test signal

Signal Output 1 range	1 V
Signal Output 1 On / Add	ON / ON
Demodulator 8 Amp 2	1.0 V / ON
Oscillator 2 Frequency	1 MHz

Table 3.15. Settings: generate the DC shift

Aux 1 Signal	Manual
Aux 1 Offset	1.5 V

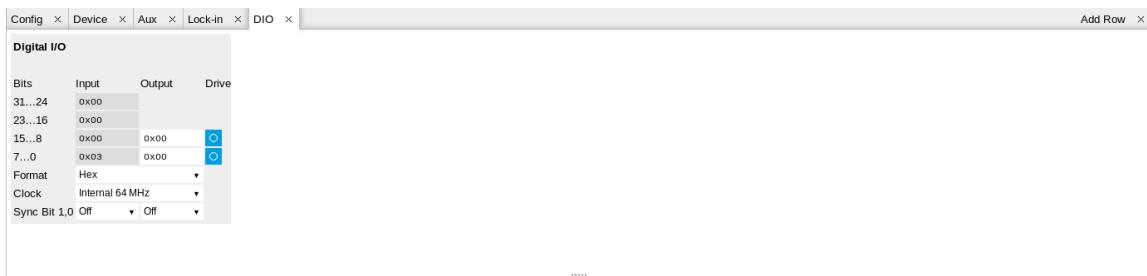


Figure 3.21. Configuring DIO 0 as reference input

Finally use the Input pull-down selector for demodulator 7 in the Lock-in tab and set it to "DIO 0". This makes oscillator 1 lock on DIO 0. The frequency of oscillator 1 should start updating similarly to what was described in [Section 3.4.3](#).

3.5. Tutorial Noise Measurement

Lock-in amplifiers can be used to measure the noise on a signal. By quantifying the noise of a system one can estimate the maximum achievable performance.

3.5.1. How Does a Lock-in Measure Noise?

A lock-in amplifier measures the signal amplitude close to a given reference frequency with a defined bandwidth around this reference frequency. The noise in an input signal near the reference frequency appears as noise in the lock-in amplifier signal output.

The noise is the standard deviation of the measured X or Y value and is measured by first calculating the average, X_{avg} , over a defined period of time. Then, this signal, X_{avg} , is subtracted from the X value to get the deviation. Finally, the RMS (root-mean-square) is calculated, corresponding to the total noise power of the input signal within a defined bandwidth around the reference frequency. The value is correct for input noise with Gaussian distribution of the noise power, which is normally the case.

Most of the times the noise spectral density is of interest, which is the normalization of the X_{noise} made independent of the filter bandwidth. To calculate the noise spectral density from the calculated RMS noise, one needs to divide the measured value by the square root of the bandwidth \sqrt{BW} . The noise spectral density has the units V/\sqrt{Hz} .

The related equations are $X_{noise} = RMS(X - X_{avg})/\sqrt{BW}$, and $Y_{noise} = RMS(Y - Y_{avg})/\sqrt{BW}$ respectively. The X and Y noise are expected to be identical.

3.5.2. Measuring the Noise of the HF2LI/HF2IS

A LabVIEW example (ziExample-HF2-Noise.vi) is available to measure the noise in an input signal. To measure the equivalent input noise of the HF2, remove all BNC connectors from the input of the device and apply the following settings in the LabOne UI.

Table 3.16. Settings: Measure HF2 Noise

Signal Input 1 range / AC / Diff / 50	0.01 V / ON / OFF / ON
Demodulator 1 Low-Pass Filter	BW 3dB = 100 Hz, Order = 4
Oscillator 1 Frequency	1 MHz
Signal Output 1 switch	OFF

Run the example, ziExample-HF2-Noise.vi. Make sure that the correct Demodulator is selected. The noise spectral density should now show a value close to $5 \text{ nV}/\sqrt{\text{Hz}}$. By changing the settings in the user interface, the noise behavior of the device can be analyzed in more detail. For example, changing the reference frequency to 10 kHz will slightly increase the spectral noise density, because of flicker noise that is larger at lower frequencies and generally present in all electronic circuits.

3.6. Tutorial Amplitude Modulation

Note

This tutorial is addressed to HF2LI lock-in amplifier users that have purchased both HF2-MF multi-frequency and HF2-MOD AM/FM modulation options.

Amplitude modulation (AM) and frequency modulation (FM) refer to the modulation of an oscillating signal $s(t) = A \cos(\omega t + \varphi)$, the so-called carrier. A and $\omega t + \varphi$ are the amplitude and the phase of the signal, respectively. [Figure 3.22](#) depicts the phasor representation of $s(t)$. The phasor follows a circle with radius A , and the phase wraps around after a full revolution of 360° . The signal $s(t)$ is the projection of the phasor on the abscissa.

In the case of AM signals, the amplitude A , i.e. the phasor length, is time dependent, as in [Figure 3.22\(b\)](#). In the case of FM signals, the phase offset φ is time dependent and the phasor has a constant amplitude, see [Figure 3.22\(c\)](#).

Amplitude and frequency modulation, best known from radio transmission, are also common lock-in detection techniques.

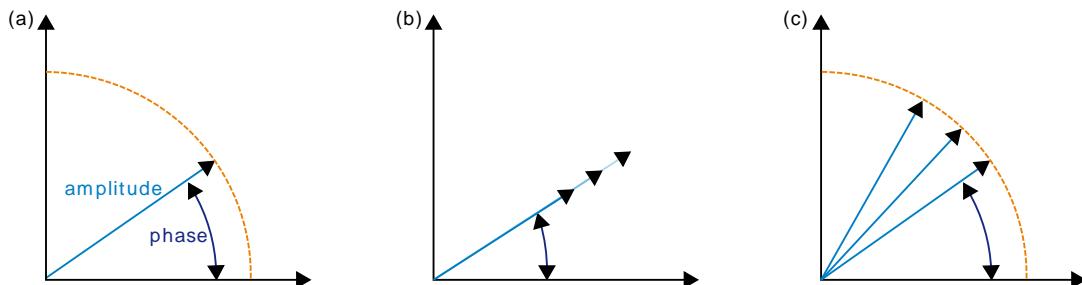


Figure 3.22. A sinusoidal signal represented as a phasor: the signal corresponds to the projection on the x axis. Amplitude (b) and frequency (c) modulated signals affect the amplitude or the phase of the phasor

3.6.1. What is Amplitude Modulation?

In the time domain, amplitude modulation of the carrier signal produces a variation of the carrier amplitude proportional to the amplitude of the modulating signal. For example, when the amplitude of a carrier with a frequency $f_c = \omega_c / 2\pi$ is modulated by a signal with frequency $f_m = \omega_m / 2\pi$ (where $f_m < f_c$), the resulting signal has the form

$$s(t) = [A + M \sin(\omega_m t)] \sin(\omega_c t + \varphi) \quad (3.1)$$

$$= A \sin(\omega_c t + \varphi) + \frac{M}{2} \cos[(\omega_c - \omega_m)t + \varphi] - \frac{M}{2} \cos[(\omega_c + \omega_m)t + \varphi] \quad (3.2)$$

where A and M are the amplitudes of the fast and slow modulations respectively and φ the phase offset. There is no restriction on the magnitude of M compared to A . The information of interest is encoded in these three parameters, A , M and φ .

In the frequency domain, the AM signal $s(t)$ is composed of three frequencies: the carrier at f_c and two additional sidebands at $f_c - f_m$ and $f_c + f_m$, as shown in [Equation 3.2](#). The two sidebands have equal amplitude $M/2$, half of the modulating signal, and the carrier amplitude is independent on the modulation amplitude.

The traditional way of measuring an AM signal is called double (or tandem) demodulation and requires two lock-in amplifiers: the first one demodulates the signal at f_c with a bandwidth that is at least as large as f_m (but smaller than $f_c - f_m$). This is to ensure that the full amplitude of the modulation signal is retained. The demodulated signal after the first lock-in becomes

$$s(t) \cdot \cos(\omega_{ct}) \xrightarrow{\text{after filtering}} d_1(t) = \frac{A + M \sin(\omega_m t)}{2} \cos \varphi$$

Equation 3.3. Tandem demodulation

In $d_1(t)$, the two sidebands are now located at the same frequency f_m , while the carrier appears as a DC component. When the demodulated signal $d_1(t)$ is fed to a second lock-in amplifier, the result of the second demodulation at f_m is proportional to $M \cos \varphi$.

In order to recover the amplitude M it is necessary to measure the φ so one can divide the result by the factor $\cos \varphi$. To measure the phase one can use a third lock-in to demodulate $s(t)$ at the carrier frequency f_c with a bandwidth smaller than f_m as shown in Figure 3.23.

Instead of using a tandem configuration, the HF2-MOD option allows the user to demodulate directly at the three frequencies f_c and $f_c \pm f_m$ simultaneously. The three parameters A , M and φ can be measured and displayed with a single instrument.

Internally, the HF2LI generates the phases ω_{ct} and ω_{mt} from which it produces $(\omega_c - \omega_m)t$, ω_{ct} and $(\omega_c + \omega_m)t$. This ensures the correct phase relationship for the demodulations of the sidebands.

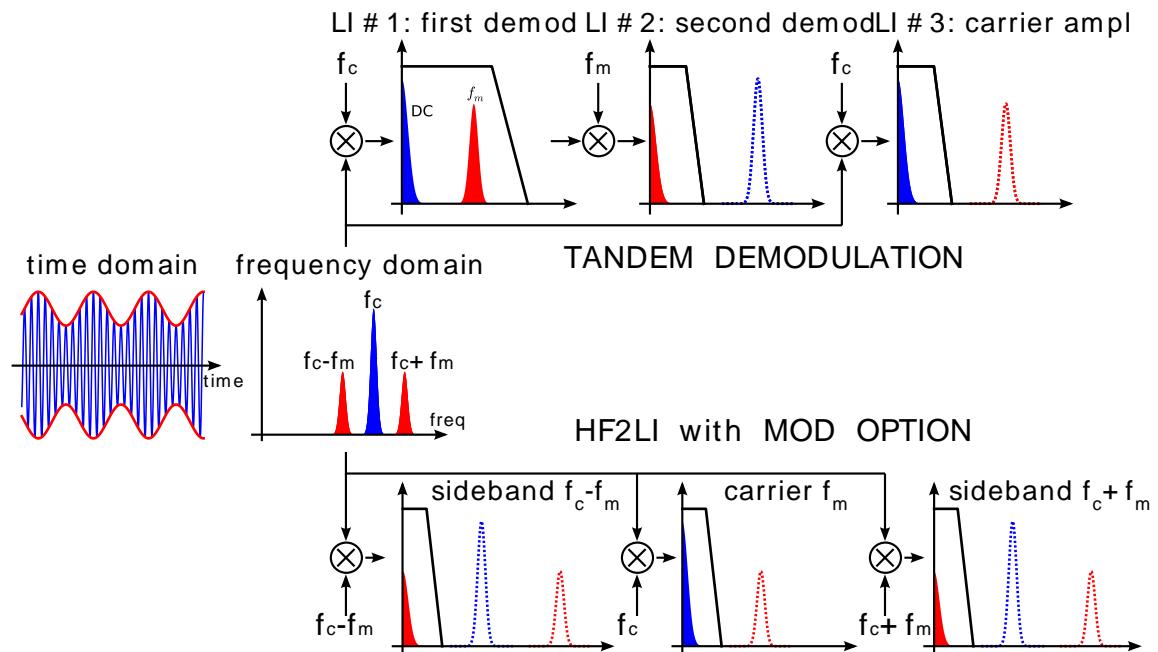


Figure 3.23. Comparison between tandem demodulation and the HF2-MOD option of an AM modulated signal

3.6.2. Generate the Test Signal

In this tutorial, you are going to generate an AM signal with a carrier frequency of 1 MHz and a modulation frequency of 100 kHz. The signal is generated at Signal Output 2 and is demodulated

by the first lock-in unit by feeding it into Signal Input 1. The HF2-MOD option requires the HF2-MF Multi-frequency option because each modulated signal requires at least two oscillators. Note that changing the Modulation tab settings will modify some of the settings found in the Lock-in MF tab. The reader is kindly referred to [Section 4.18](#).

Start by enabling the Signal Output 2 in the Lock-in MF tab and disabling all demodulator Output Amplitudes. This will ensure that only the desired components of the amplitude-modulated signal appear on the output.

Table 3.17. Settings: generate the AM signal

Signal Output 2 Enable	ON
Signal Output Amplitudes Demodulators 1-8	OFF

In the Modulation tab, in the MOD 2 section, select the following parameters:

Table 3.18. Settings: generate the AM signal

Carrier Oscillator (Osc)/Frequency	Osc 1 / 1 MHz
Sideband 1 Oscillator (Osc)/Frequency	Osc 2 / 100 kHz
Carrier Mode	AM
Generation Carrier/Modulation Amplitude	200 mV / 100 mV
Generation Carrier/Modulation Enable	ON / ON
MOD 2 Enable	ON

This generates an AM signal with two sidebands of equal amplitude. To look at this signal, connect Signal Output 2 to Signal Input 1 of the HF2LI. Select the correct input parameters: in the Lock-in tab, for Signal Input 1, make sure Differential mode and $50\ \Omega$ are disabled. Then click on the auto range button . In the Scope tab, select Source to be Signal Input 1, Trigger to be Signal Output 2 and click on Run to activate the Scope. Observe how the carrier amplitude is modulated at 100 kHz as seen in . In Frequency Domain FFT mode, the plot shows three peaks: the carrier at 1 MHz and two sidebands at 0.9 and 1.1 MHz (see the cursors in the frequency domain representation in [Figure 3.25](#)).

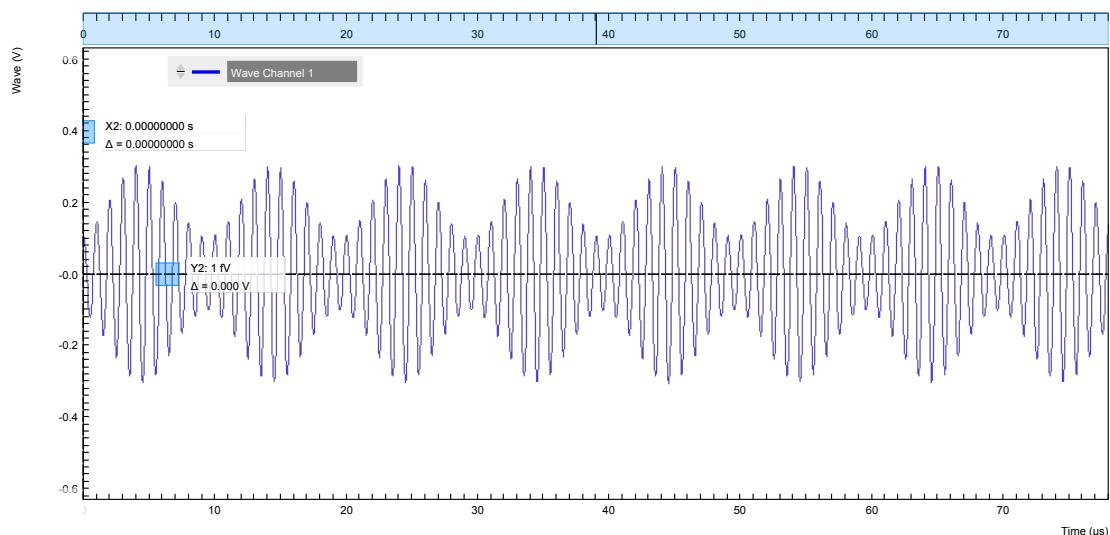


Figure 3.24. Time domain representation of the AM signal generated by MOD2 measured with the LabOne Scope

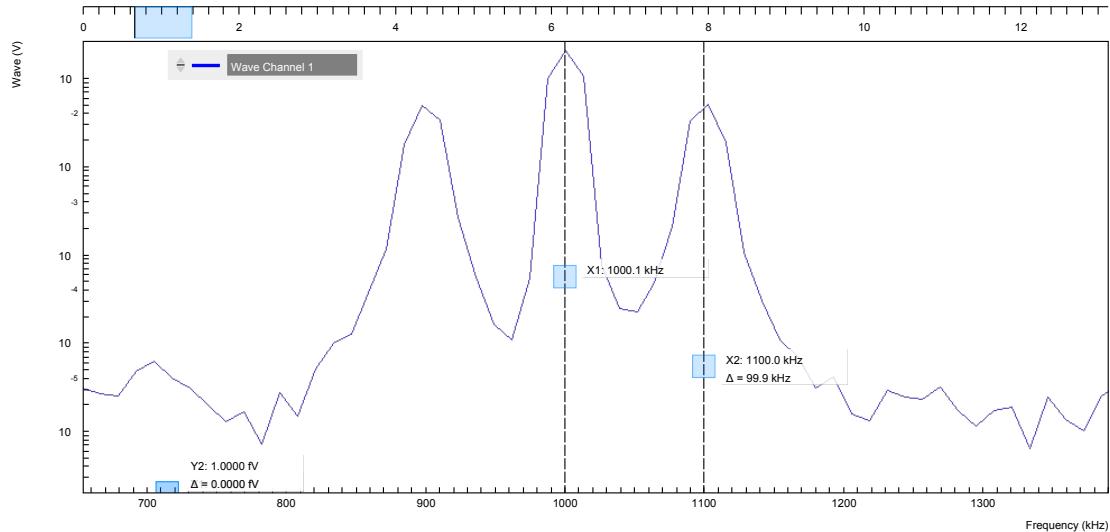


Figure 3.25. Frequency domain representation of the AM signal generated by MOD2 measured with the LabOne Scope

3.6.3. Measure the Test Signal

In the Modulation tab, in the MOD 1 section, select the following parameters:

Table 3.19. Settings: measure the AM signal

Carrier oscillator (Osc)	1
Sideband 1 oscillator (Osc)	2
Carrier Mode	AM
Low-pass Filter BW (Carrier)	10 Hz
Low-pass Filter BW (Sideband 1)	10 Hz
MOD 1 Enable	ON
Demod 1, 2, 3 Data Transfer Enable (Lock-in tab)	ON

This sets the correct demodulation of the AM signal with the two sidebands. In the Numeric tab, look at the amplitude of the carrier, 142 mV_{RMS} and of the two sidebands, 35 mV_{RMS} each, one quarter of the carrier amplitude: this corresponds to a modulation index of 50%.

3.7. Tutorial Frequency Modulation

Note

This tutorial is addressed to HF2LI lock-in amplifier users that have purchased both HF2-MF multi-frequency and HF2-MOD AM/FM modulation options.

3.7.1. What is Frequency Modulation?

In frequency modulation (FM), the amplitude of the modulating signal is proportional to the instantaneous frequency deviation from a fixed frequency. In the simplest case shown in [Figure 3.26\(a\)](#), the modulated signal

$$s(t) = A \cos \left[\omega_c t + \frac{\omega_p}{\omega_m} \sin(\omega_m t) + \varphi \right]$$

[Equation 3.4. Frequency modulation](#)

is produced when a carrier signal of frequency $f_c = \omega_c / 2\pi$ is modulated by a sinusoidal modulation with frequency $f_m = \omega_m / 2\pi$. The maximum variation of the frequency around the carrier frequency, the peak frequency deviation, is $f_p = \omega_p / 2\pi$. The physical information is encoded in the parameters A , f_p and φ .

Because the frequency is the time derivative of the phase (divided by 2π) and the phase is the argument of the cosine in equation [Equation 3.4](#), we can define the instantaneous frequency as

$$f(t) = f_c + f_p \cos(2\pi f_m t)$$

[Equation 3.5. Instantaneous frequency](#)

The spectrum of the FM signal of [Equation 3.4](#) is more complicated than in the case of amplitude modulation. It consists of the carrier and a series of pairs of sidebands at multiple integers of f_m around the carrier frequency, see [Figure 3.26\(d\)](#). The amplitudes of the carrier and sidebands are given by mathematical functions called Bessel functions usually indicated by J_n evaluated at the modulation index $h = f_p / f_m$. For instance, the n-th pair of sidebands is located symmetrically about f_m at $f_c \pm nf_m$ and its amplitude is $J_n(h)$.

A peculiarity of the Bessel functions is that they oscillate around zero: even for the carrier, as the modulation index is increased, its amplitude $J_0(h)$ decreases, crossing zero at $h \approx 2.41$ and then it increases in amplitude (in anti-phase) before reaching zero again at $h \approx 5.52$.

At low modulation indexes, the amplitude of higher sidebands is very low and can thus be ignored: this is called the narrow-band approximation. In this limit (it is customary to assume $h < 0.2$), only the two sidebands at $f_c \pm f_m$ have non-negligible amplitude and the signal $s(t)$ can be approximated by

$$\tilde{s}(t) = A [J_0(h) \sin(\omega_c t + \varphi) - J_1(h) \cos[(\omega_c + \omega_m)t + \varphi] + J_1(h) \cos[(\omega_c - \omega_m)t + \varphi]]$$

[Equation 3.6. Approximated narrow-band FM signal](#)

The first term is the carrier, the other two are the lower and upper sidebands. The problem of finding h (and the peak amplitude f_p) reduces now to comparing the amplitude of the first pair of sidebands and the carrier to the ratio $J_1(h)/J_0(h)$. A plot of the ratio $J_1(h)/J_0(h)$ and $J_2(h)/J_0(h)$ is shown in Figure 3.26(e).

Even though $\tilde{s}(t)$ looks very similar to an AM signal, there is a subtle but substantial difference: the phases of the sidebands are offset with respect to that of the carrier. This results in the sidebands being in quadrature with the carrier. For example, assume that $\varphi = 0$: demodulating $\tilde{s}(t)$ with the carrier signal $\sin(\omega_c t)$ gives the DC component (the carrier) but no sidebands; on the other hand, demodulating with the quadrature $\cos(\omega_c t)$, only the two sidebands at f_m are observed and no carrier is present. Because of this, FM detection can be done in a similar way as AM detection scheme, using the tandem configuration described previously in Section 3.6.1.

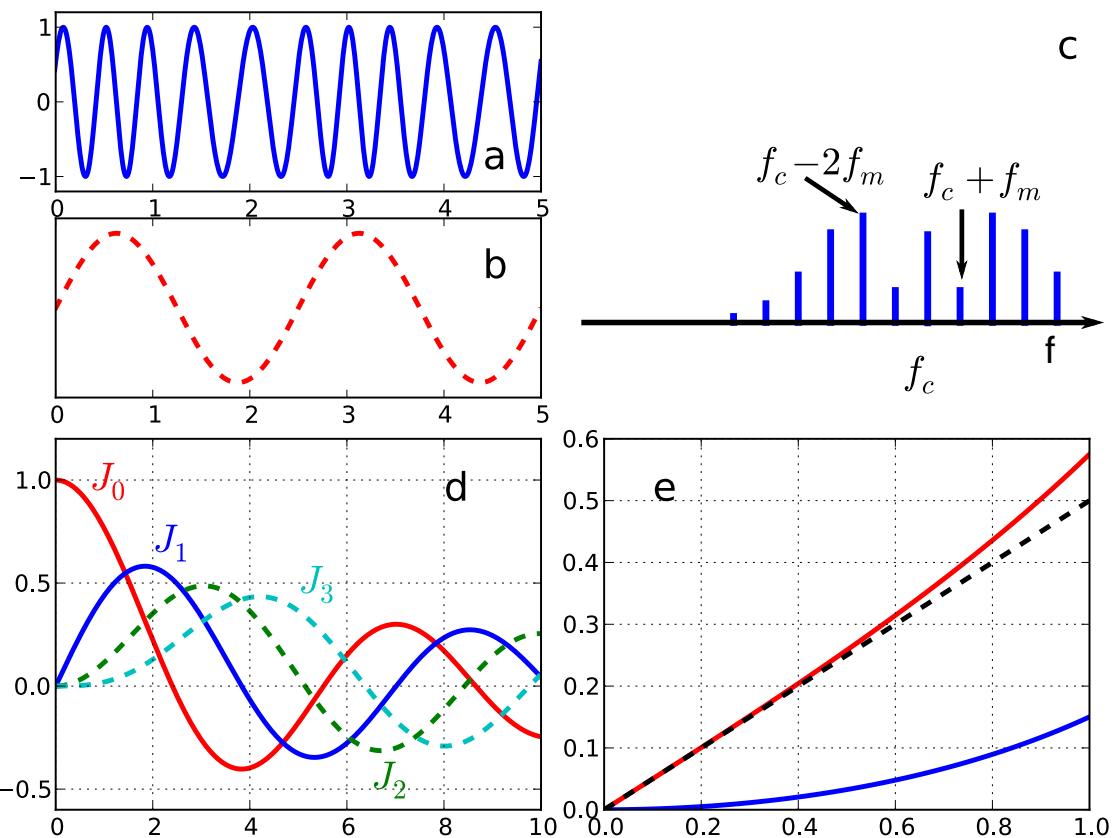


Figure 3.26. (a) A simple frequency modulated signal, (b) its instantaneous frequency, (c) the frequency domain spectrum of a FM signal is composed of an infinite series of sidebands, here depicted for $h = 0.35$, (d) n-th Bessel function versus h , (e) ratio $J_1(h)/J_0(h)$ (red line), $J_2(h)/J_0(h)$ (blue line), slope 0.5 line (black dashed line)

The HF2-MOD AM/FM Modulation option permits direct generation and demodulation of an FM signal. For demodulation, this option enables measurement of the parameters A , f_p , and φ .

Internally the HF2LI calculates the peak frequency f_p with the method described above, from the ratio $J_1(h)/J_0(h)$, proportional to the carrier and first sideband amplitude. Since this method is

valid only for narrow-band frequency modulation, users are advised to work at small values of the modulation index $h < 1$.

Another, intuitive way of demodulating an FM signal would be to use the PLL to track the frequency deviation Δf and to further demodulate this signal. However, using sideband demodulation with the HF2-MOD AM/FM Modulation option provides a better signal-to-noise ratio. This is because the signal can be averaged over several modulation cycles while keeping the bandwidth small.

3.7.2. Generate the Test Signal

In this tutorial, you are going to generate an FM signal with a carrier frequency of 1 MHz, a modulation frequency of 100 kHz, and a modulation index of 0.1. The signal is generated at Signal Output 2 and is demodulated by the first lock-in unit by feeding it into Signal Input 1.

Start by enabling the Signal Output 2 in the Lock-in MF tab and disabling all demodulator Output Amplitudes. This will ensure that only the desired components of the frequency-modulated signal appear on the output.

Table 3.20. Settings: generate the AM signal

Signal Output 2 Enable	ON
Signal Output Amplitudes Demodulators 1-8	OFF

In the Modulation tab, in the MOD 2 section, select the following parameters:

Table 3.21. Settings: generate the FM signal

MOD 2 Enable	ON
Carrier Oscillator (Osc)/Frequency	Osc 1 / 1 MHz
Sideband 1 Oscillator (Osc)/Frequency	Osc 2 / 100 kHz
Carrier Mode/Enable	FM / ON
Generation Carrier Amplitude/Enable	100 mV / ON
Generation Index	0.1

This generates an FM signal consisting of a carrier and two sidebands at $f_c \pm f_m$. To look at this signal, connect Signal Output 2 to Signal Input 1 of the HF2 Instrument. Select the correct input parameters in the Lock-in tab: for Signal Input 1, make sure Differential and 50Ω are turned off. Then click the auto range button. In the Scope tab, select Signal Input 1 as Source, Signal Output 2 as Trigger, and click on Run/Stop to activate the Scope. Observe that the carrier amplitude is constant. The periodic frequency variation is hardly visible. In Frequency Domain FFT mode, the plot shows the carrier at 1 MHz and the two sidebands. You can increase the frequency resolution by selecting a smaller Sampling Rate and larger time scale in the Horizontal section of the Scope.

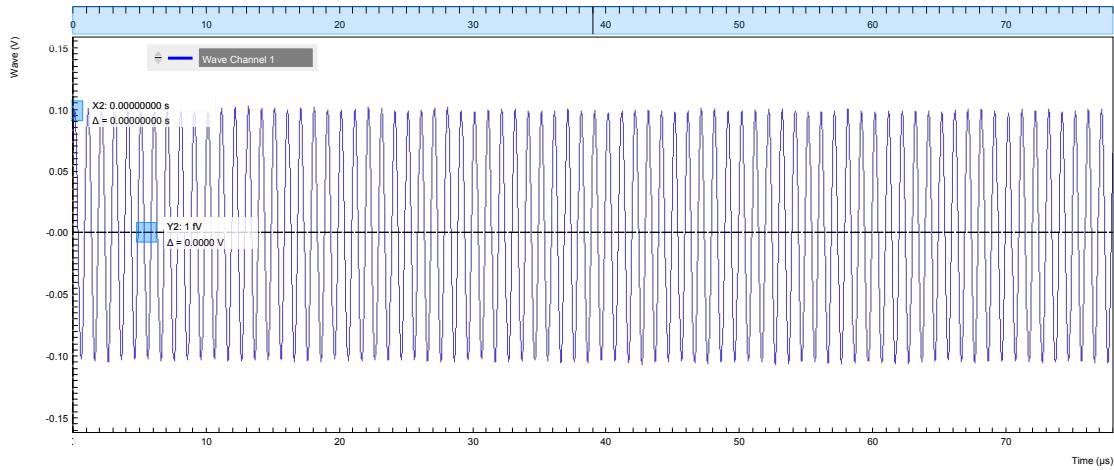


Figure 3.27. Time domain representation of the FM signal generated by MOD2 measured with the LabOne Scope

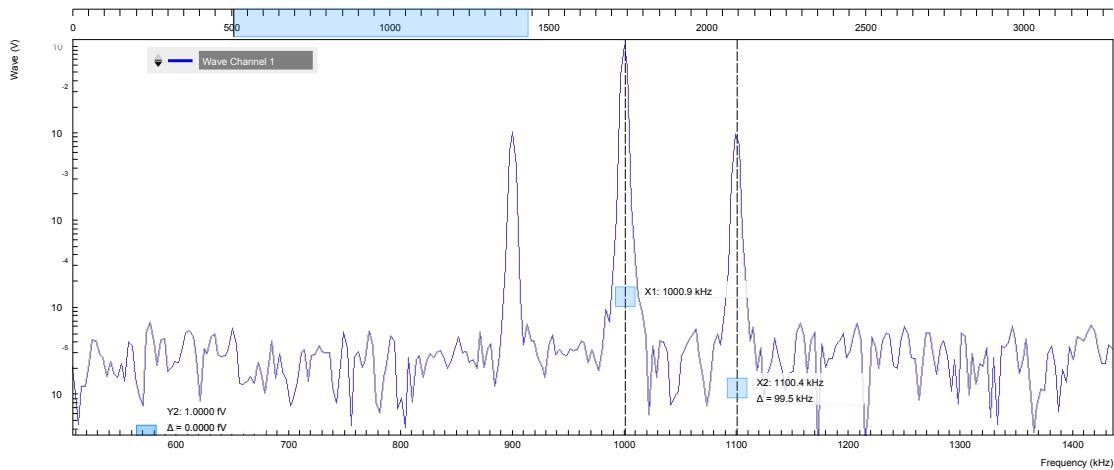


Figure 3.28. Frequency domain representation of the FM signal generated by MOD2 measured with the LabOne Scope

3.7.3. Measure the Test Signal

In the Modulation tab, in the MOD 1 section, select the following parameters:

Table 3.22. Measure the FM signal

MOD 1 Enable	ON
Carrier oscillator (Osc)	1
Sideband 1 oscillator (Osc)	2
Carrier Mode	FM
Low-pass Filter BW (Carrier)	10 Hz
Low-pass Filter BW (Sideband 1)	10 Hz
Demod 1, 2, 3 Data Transfer Enable (Lock-in tab)	ON

This sets the correct demodulation of the FM signal. In the Numerical tab, look at the amplitude of the carrier, $71 \text{ mV}_{\text{RMS}}$. You can also see that the two sidebands have an amplitude of $3.5 \text{ mV}_{\text{RMS}}$.

This corresponds approximately to the carrier amplitude multiplied by the ratio $J_1(h)/J_0(h)$, see [Figure 3.26\(e\)](#) for our modulation index of $h=0.1$. Note that the phases of the two sidebands are 180° apart, which is typical for FM.

3.8. Tutorial Phase-locked Loop

Note

This tutorial is applicable to HF2 Instruments with the HF2-PLL Dual Phase-locked Loop option installed.

3.8.1. Preparation

This tutorial explains how to track the resonance frequency shift of a quartz resonator using the PLL. To follow this tutorial, one simply needs to connect a resonator between Signal Output 1 and Signal Input 1.

3.8.2. Characterize the Resonator with the Sweeper

In this section you will learn first how to find the resonance of your resonator by measuring a frequency dependence in the Sweeper tab. The setup of the signal input, signal output, and data transfer in the Lock-in tab you know already from previous tutorials. In the Sweeper, one can start by defining a frequency sweep range from DC to 50 MHz and narrow down the range using multiple sweeps in order to find the resonance peak of interest. In our case, we know already that the resonance lies at around 1.8 MHz which saves us some time in finding the peak. The Sweeper tab and Lock-in tab setup is shown below.

Table 3.23. Settings: acquire the reference signal

Tab	Sub-tab	Section	#	Label	Setting / Value / State
Lock-in	7	Output Amplitudes	1	Amp 1 (Vpk) / En	100.0 m / ON
Lock-in	7	Signal Output 1		On	ON
Lock-in	1	Reference		Osc	1
Lock-in	1	Signal Input		Signal	Sig In 1
Lock-in	1	Data Transfer	PC	En	ON
Sweeper	Control	Horizontal		Sweep Param.	Osc 1 Frequency
Sweeper	Control	Horizontal		Length	300
Sweeper	Control	Horizontal		Start (Hz)	1.7 M
Sweeper	Control	Horizontal		Stop (Hz)	1.9 M
Sweeper	Control	Horizontal		Dual Plot	ON
Sweeper	Control	Vertical Axis Groups	1	Demod R	Add Signal
Sweeper	History	History		Length	2
Sweeper	Control			Run/Stop	ON

In this tutorial, we are using the demodulator 7 to generate the sweep signal. We use demodulator 1 for measurement. The Lock-in settings ensure that the same oscillator (number 1) is used for the sweep signal generation and for the demodulation. In addition, the input must be set to Signal Input 1.

Once the Sweeper **Run/Stop** button is clicked, the sweeper will repeatedly sweep the frequency response of the quartz oscillator. The user can then use the plot area tools explained in

[Section 4.1.3](#) to get a focus on the resonance peak of interest. Using the [Copy From Range](#) button, the start and stop frequencies can easily be readjusted for a finer sweeper range. This will automatically enter the plot window range into the Start and Stop fields of the swept frequency range. The history length of 2 allows you to keep one previous sweep on the screen while adjusting the zoom.

Note

The sweep frequency resolution will get finer when zooming in horizontally using the [Copy From Range](#) button even without changing the number of points.

When a resonance peak has been found, you should get a spectrum similar to the plots shown in [Figure 3.29](#). In this example, we have found the resonance peak at about 1.84 MHz. The X and Y cursors help in extracting the precise resonance frequency as well as the half power bandwidth of about 93 Hz (note that this is the peak width at $1/\sqrt{2}$ of the maximum signal amplitude). The phase response of the resonator reaches a value of about -90° at the resonance. We will use this value as a phase setpoint for the PLL. After having completed the Sweeper measurements, turn off sweeping by clicking on [Run/Stop](#). This will release the oscillator frequency from the control of the Sweeper.

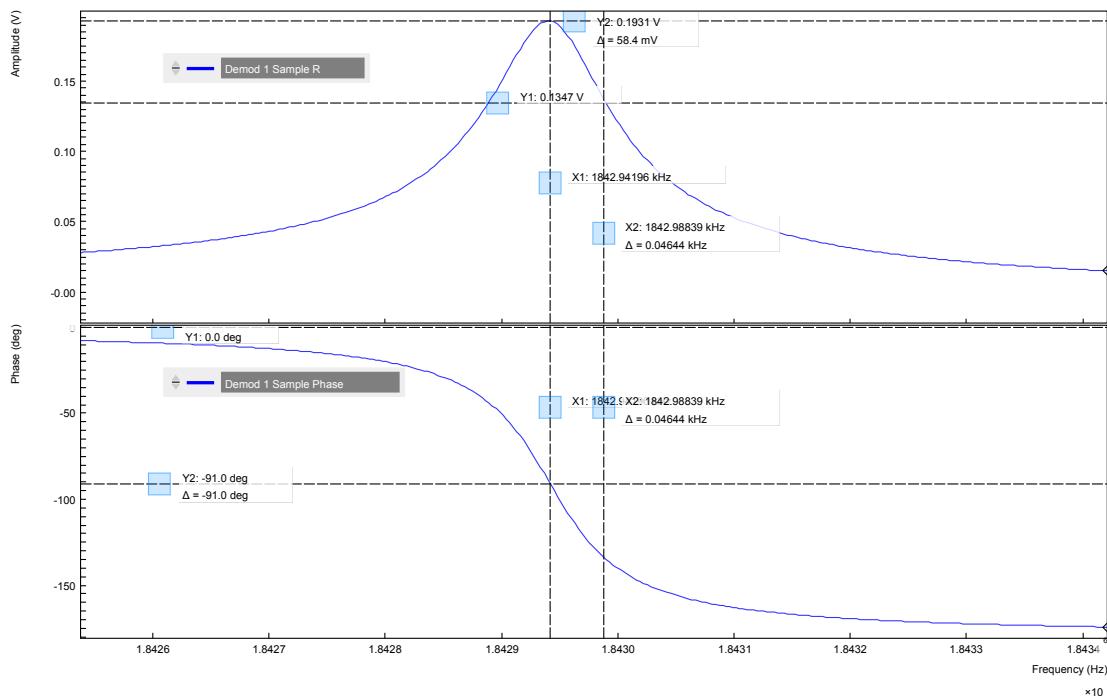


Figure 3.29. Quartz resonator amplitude and phase response

3.8.3. Resonance Tracking with the PLL

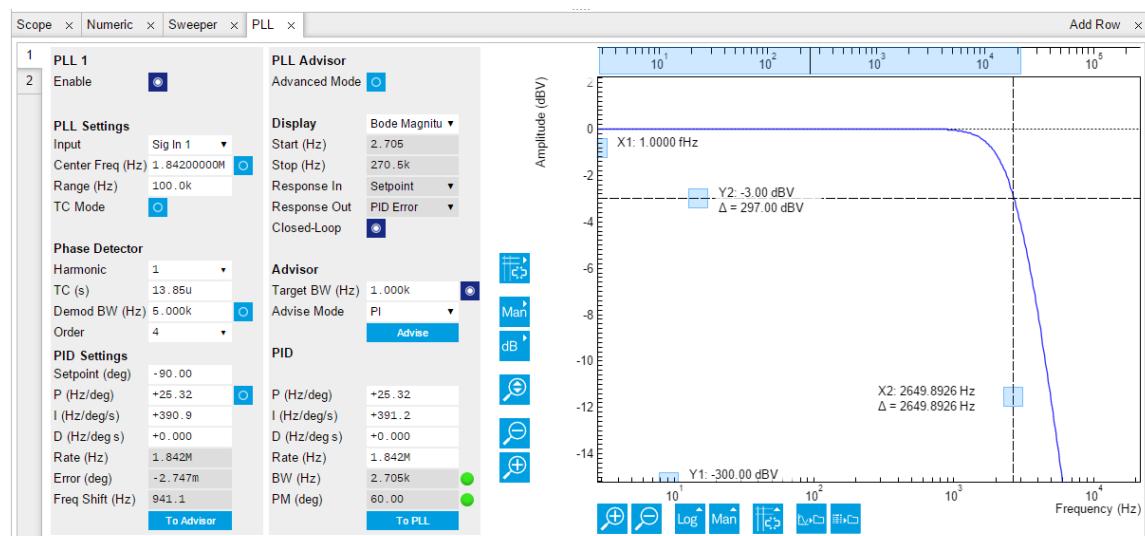
Now that we have located the resonance frequency and its phase, we can now track the drift in resonance frequency by locking on the phase that we just measured using the Sweeper, hence the name phase-locked loop. The phase-locked loop is available under the PLL tab. There are two PLL controllers in each HF2 instrument. For this tutorial, we will use PLL 1. We first set up the basic PLL 1 fields as shown in the table below, using the resonator parameters determined using the Sweeper.

Table 3.24. Settings: acquire the reference signal

Tab	Sub-tab	Section	#	Label	Setting / Value / State
PLL	1	PLL Settings		Center Freq (Hz)	1.842 M / OFF
PLL	1	PLL Settings		Range (Hz)	100 k
PLL	1	PID Settings		Setpoint (deg)	-90

Next, we need to find suitable P, I, and D parameters. One can use the PLL Advisor for this purpose. For this tutorial, we will set the Target BW (Hz) to 1.0 k. The target bandwidth should be at least as large as the expected bandwidth of the frequency variations. In the present case, the resonator frequency is practically stable, so 1 kHz bandwidth is largely enough. Click on the Advise button to have the Advisor find a set of PID parameters using a numerical optimization algorithm. The Advisor tries to match or exceed the target bandwidth in its simulation. Figure 3.30 shows the PLL tab after the Advisor has finished. In this case, the achieved bandwidth is about 2.7 kHz and can be read from the BW (Hz) field, or directly from the 3 dB point of the simulated Bode plot on the right. The Phase Margin value of the simulation is displayed in the PM (deg) field and should exceed 45° to ensure stable feedback operation without oscillations.

Once you are satisfied with the Advisor results, click on the To PLL button to copy the PID parameters to the physical PLL unit. To start the PLL operation, click on the Enable button on the top of the tab.

**Figure 3.30.** PLL settings and simulation in the PLL tab

The frequency shift of the internal oscillator is shown in the field Freq Shift (Hz) and should now start to toggle. A successful phase lock is indicated by an Error (deg) toggling around zero, and a frequency shift that lies within the bounds -Range and +Range defined in the PLL settings.

Note

At this point, it is recommended to adjust the signal input range by clicking on the Auto Range button in the Lock-in tab. This will sometimes help the PLL to lock to an input signal with a better noise performance.

The easiest way to visualize the frequency noise and drift is offered by the Plotter tool. One simply needs to select Frequency and Channel 1 and then click on Add Signal. The frequency noise

increases with the PLL bandwidth, so for optimum noise performance the bandwidth should not be higher than what is required by the experiment. The frequency noise also scales inversely with the drive amplitude of the resonator.

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 HF2. LabOne provides a data server and a web server to control the Instrument with any of the most common web browsers (e.g. Firefox, Chrome, Edge, etc.). This platform-independent architecture supports interaction with the Instrument using various devices (PCs, tablets, smartphones, etc.) even at the same time if needed.

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

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 HF2. 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 (possibly concurrently) by custom programs written in any of the supported languages (e.g. LabVIEW, MATLAB, Python, C) connecting through the LabOne APIs.

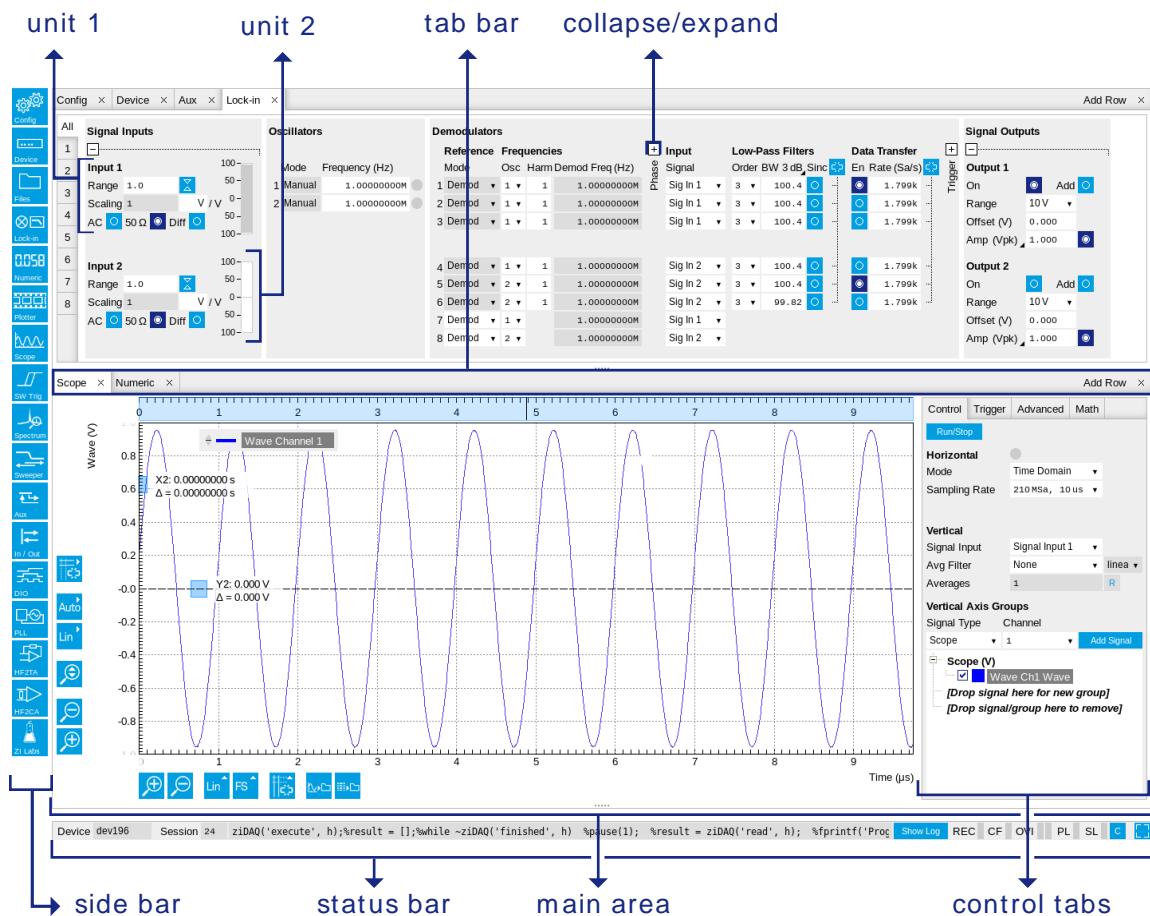


Figure 4.1. LabOne User Interface (default view)

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 left side bar. A simple 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 simply be displaced by drag-and-drop within a row or across rows. Further items are highlighted in Figure 4.2.

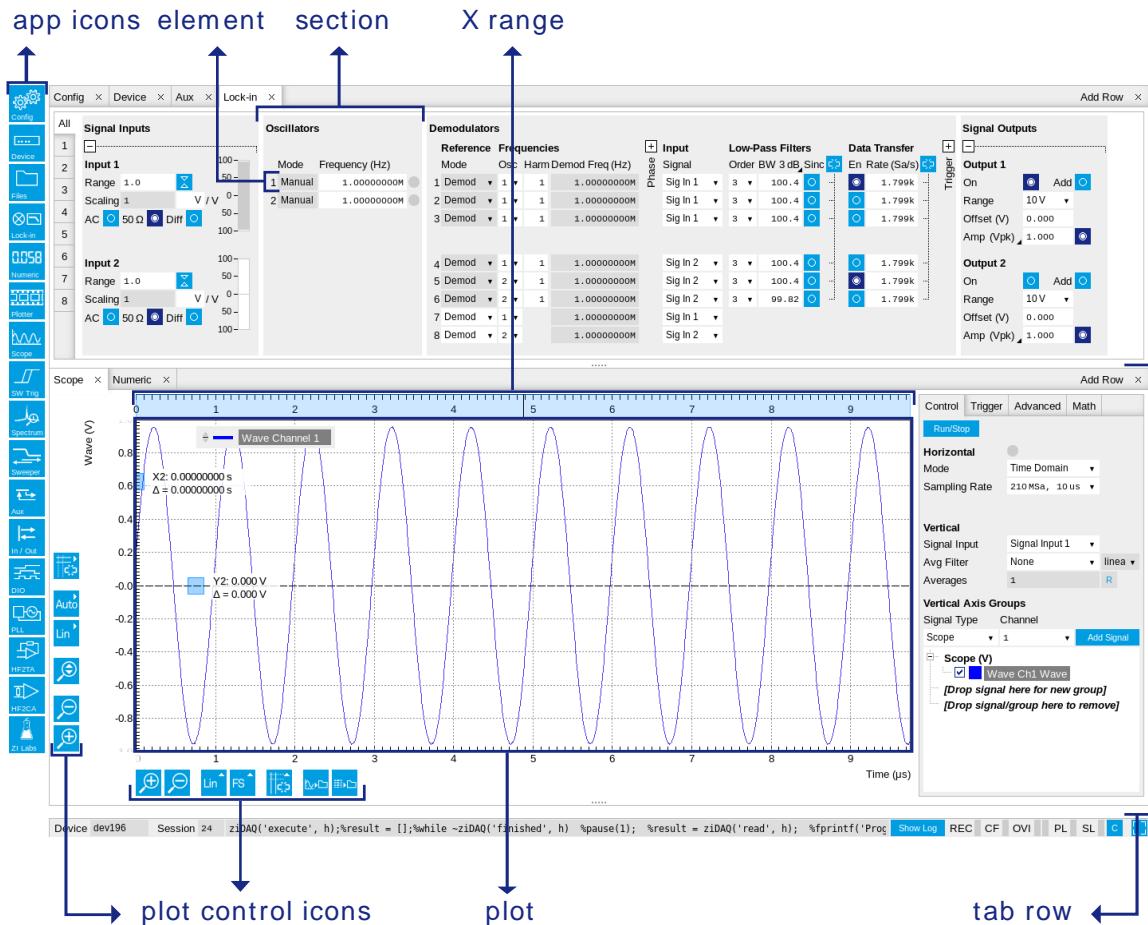


Figure 4.2. LabOne User Interface (more items)

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, control tabs, unit selections

4.1.2. Unique Set of Analysis Tools

All Instruments feature a comprehensive tool set for time and frequency domain analysis for both raw signals and demodulated signals. Note that the selection of app icons is limited by the upgrade options installed on a particular instrument.

The app icons on the left side of the UI can be roughly divided into two categories: settings and tools. Settings-related tabs are in direct connection of the instrument hardware allowing the user to control all the settings and instrument states. Tools-related tabs place a focus on the display and analysis of gathered measurement data. There is no strict distinction between settings and tools, e.g. will the Sweeper change certain demodulator settings while performing a frequency sweep. Within the tools one can further discriminate between time domain and frequency domain analysis, moreover, a distinction between the analysis of fast input signals - typical sampling rate of 210 MSa/s - and the measurement of orders of magnitude slower data - typical sampling rate of - derived for instance from demodulator outputs and auxiliary inputs. [Table 4.2](#) provides a brief classification of the tools.

Table 4.2. Tools for time domain and frequency domain analysis

	Time Domain	Frequency Domain
Fast signals (210 MSa/s)	Oscilloscope (Scope Tab)	FFT analyzer (Scope Tab)
Slow signals (<200 kSa/s)	Numeric	Spectrum Analyzer (Spectrum Tab)
	Plotter	Sweeper
	Software Trigger	-

The following table gives the overview of all app icons.

Table 4.3. Overview of app icons and short description

Control/Tool	Option/Range	Description
Lock-in		Quick overview and access to all the settings and properties for signal generation and demodulation.
Lock-in MF		Quick overview and access to all the settings and properties for signal generation and demodulation.
Files		Access settings and measurement data files on the host computer.

Control/Tool	Option/Range	Description
Numeric		Access to all continuously streamed measurement data as numerical values.
Plotter		Displays various continuously streamed measurement data as traces over time (roll-mode).
Scope		Displays shots of data samples in time and frequency domain (FFT) representation.
SW Trig		Provides complex trigger functionality on all continuously streamed data samples and time domain display.
Spectrum		Provides FFT functionality to all continuously streamed measurement data.
Sweeper		Allows to scan one variable (of a wide choice, e.g. frequency) over a defined range and display various response functions including statistical operations.
Aux		Controls all settings regarding the auxiliary inputs and auxiliary outputs.
In/Out		Access to all controls relevant for the main Signal Inputs and Signal Outputs on the instrument's front.
DIO		Gives access to all controls relevant for the digital inputs and outputs including the Ref/Trigger connectors.
Config		Provides access to software configuration.
Device		Provides instrument specific settings.
PID		Features all control and analysis capabilities of the PID controllers.
PLL		Features all control and analysis capabilities of the phase-locked loops.

Control/Tool	Option/Range	Description
MOD		Control panel to enable (de)modulation at linear combinations of oscillator frequencies.
HF2CA		Remote control of the HF2CA Current Amplifier.
HF2TA		Remote control of the HF2TA Current Amplifier.
ZI Labs		Experimental settings and controls.

Table 4.4 gives a quick overview over the different status bar elements along with a short description.

Table 4.4. 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 Log	Show the command log history in a separate browser window.
Session	integer value	Indicates the current session identifier.
Device	devXXX	Indicates the device serial number.
REC	grey/green	A green indicator shows ongoing data recording (related to global recording settings in the Config tab).
CF	grey/yellow/red	Clock Failure - Red: present malfunction of the external 10 MHz reference oscillator. Yellow: indicates a malfunction occurred in the past.
OVI	grey/yellow/red	Signal Input Overflow - Red: present overflow condition on the signal input also shown by the red front panel LED. Yellow: indicates an overflow occurred in the past.

Control/Tool	Option/Range	Description
OVO	grey/yellow/red	Overflow Signal Output - Red: present overflow condition on the signal output. Yellow: indicates an overflow occurred in the past.
COM	grey/yellow/red	Warning flags related to instrument communication. From left to right: Packet Loss, Sample Loss. Packet Loss - Red: present loss of data between the device and the host PC. Yellow: indicates a loss occurred in the past. Sample Loss - Red: present loss of sample data between the device and the host PC. Yellow: indicates a loss occurred in the past. A possible cause for sample loss may be the scope running in parallel.
C		Reset status flags: Clear the current state of the status flags
MOD	grey/green	MOD - Green: indicates which of the modulation kits is enabled.
PID	grey/green	PID - Green: indicates which of the PID units is enabled.
PLL	grey/green	PLL - Green: indicates which of the PLLs is enabled.
Full Screen		Toggles the browser between full screen and normal mode.

4.1.3. Plot Functionality

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

Plot area elements

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

The mouse functionality inside of plot is summarized in [Table 4.5](#)

Table 4.5. 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 4.6](#).

Table 4.6. Plot control description

Control/Tool	Option/Range	Description
Axis scaling mode		Selects between automatic, full scale and manual axis scaling.
Axis mapping mode		Select between linear, logarithmic and decibel axis mapping.
Axis zoom in		Zooms the respective axis in by a factor of 2.
Axis zoom out		Zooms the respective axis out by a factor of 2.
Rescale axis to data		Rescale the foreground Y axis in the selected zoom area.
Save figure		Generates an SVG of the plot area or areas for dual plots to the local download folder.
Save data		Generates a TXT consisting of the displayed set of samples. Select full scale to save the complete wave. The save data function only saves one shot

Control/Tool	Option/Range	Description
		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.

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 by pan or zoom events. In case the cursors move out of the zoom area, the corresponding handle is displayed on the related side 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 4.7](#) gives an overview of all the elements and their functionality. It is important to know that the Signals and Operations defined will always be performed only on the currently chosen active trace.

Table 4.7. 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 plot data inside the rectangle defined by the cursor coordinates as input for statistical functions (Min, Max, Avg, Std, Int).
	Tracking	Output plot value at current cursor position. Options are X1 and X2.
	Wave	Consider all plot data currently displayed in the Plot as input for statistical functions (Min, Max, Avg, Std, Int).
	Peak	Find and determine the various peaks in the plotted data and their associated values.
	Histogram	Select statistical data as Math input and display a histogram in the plot area.

Control/Tool	Option/Range	Description
Operation Select		Select from a list of mathematical operations to be performed on the selected signals. Choice offered depends on input signals selected.
	X1, X2, X2-X1, Y1, Y2, Y2-Y1	Cursors values and their differences.
	Min, Max, Avg, Std, Int	Statistical Functions applied to a set of samples.
	Pos, Level	Finds the Position (x-values) and the Levels (y-values) of Peaks on a set of samples.
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.
Select All	Select All	Select all lines from the result table above.
Clear Selected	Clear Selected	Clear selected lines from the result table above.
Unit Prefix	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.
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

For calculation of the standard deviation the corrected sample standard deviation is used as

defined by $\sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2}$ with a total of N samples x_i and an arithmetic average \bar{x} .

Tree Sub-Tab

The Numeric tab and Plotter tab are able to display so many different types of signals that a number of different options are provided to access them. One of them is the Tree sub-tab that

allows one to access all streamed measurement data in a hierarchical structure by checking the boxes of the signal that should be displayed.

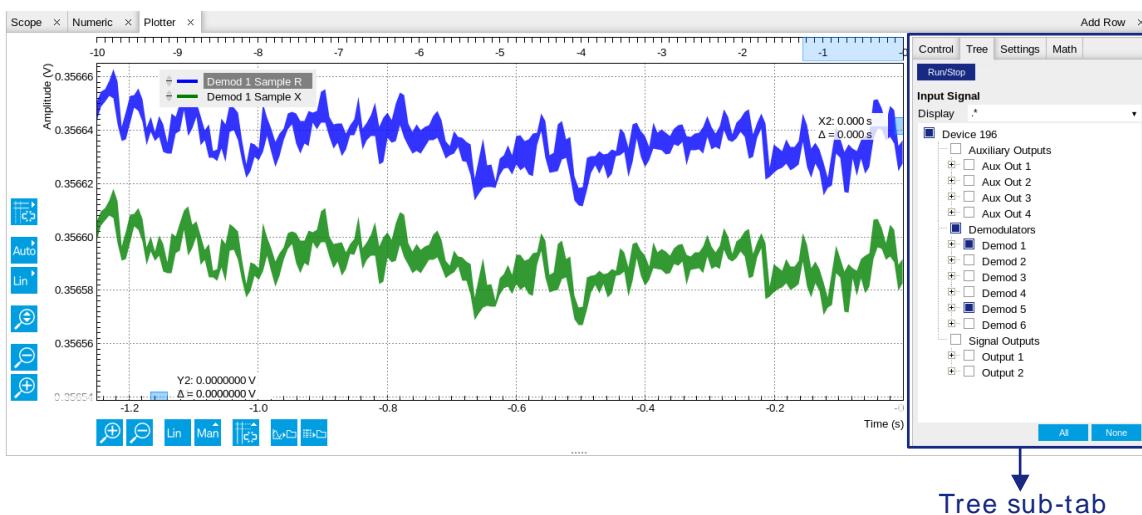


Figure 4.3. Tree sub-tab in Plotter tab

Table 4.8. Tree description

Control/Tool	Option/Range	Description
Display	Preset filter or regular expression	Predefined filters that limit the view to specific signal groups. The display filter does not select any nodes.
All	All	Select all nodes that can be selected in the relevant context.
None	None	Unselect all nodes.

Vertical Axis Groups

Vertical Axis groups are available in the Plotter tab, SW Trigger tab, and Sweeper tab. These tools are able to show signals with different axis properties within the same plot. As a frequency and amplitude axis have fundamentally different limits they have each their individual axis which allows for correct auto scaling. However, signals of the same type e.g. Cartesian demodulator results should preferably share one scaling. This allows for fast signal strength comparison. To achieve this 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.

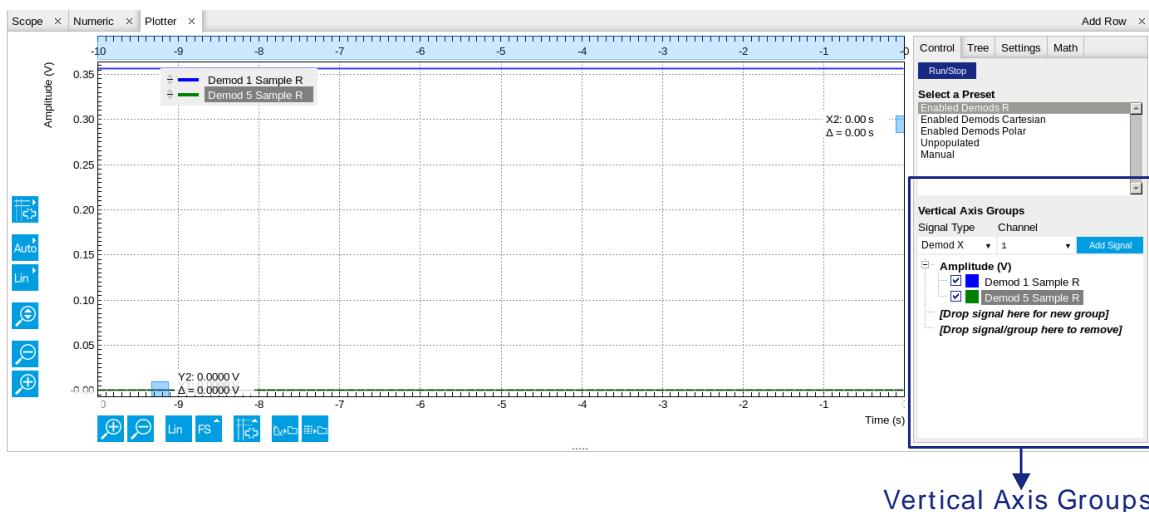


Figure 4.4. Vertical Axis Group in Plotter tool

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

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

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

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

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

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

Remove a vertical axis group: A group is removed as soon as the last signal of a custom group is removed. Default groups will remain active until they are explicitly removed by drag-and-drop. If a new signal is added that matches 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.

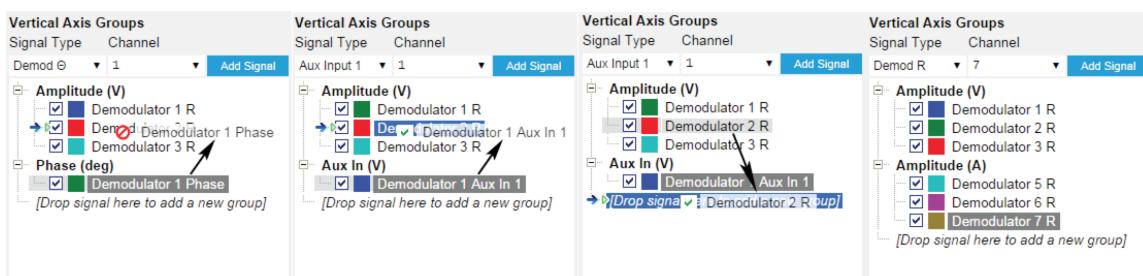


Figure 4.5. Vertical Axis Group typical drag and drop moves

Table 4.9. 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	Demod X, Y, R, Theta	Select signal types for the Vertical Axis Group.
	Frequency	
	Aux Input 1, 2	
	HW Trigger	
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.

4.1.4. Saving and Recording Data

In this section we discuss how to save and record measurement data with the HF2 Instrument using the LabOne user interface. A quick way of doing this was already introduced in the previous section: in any plot (in the Plotter, Scope, Spectrum, and other tabs), you can save the currently displayed curves as a comma-separated value (CSV) file to the download folder of your web browser. Just click on the corresponding icon  at the bottom of the plot. Clicking on  will save a vector graphics instead.

The record functionality in comparison allows you to monitor and 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: CSV or Matlab binary file format. This global setting also applies to the storage format used by the [Sweeper](#) and the [Software Trigger](#) tab. 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 DeviceXXXX/

Demodulators/Demod 1/Sample. An example for an instrument setting would be the filter time constant, accessible under the path DeviceXXXX/Demodulators/Demod 1/Filter Time Constant.

The default storage location is the LabOne Data folder which can for instance be accessed via the Windows Start menu. 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.

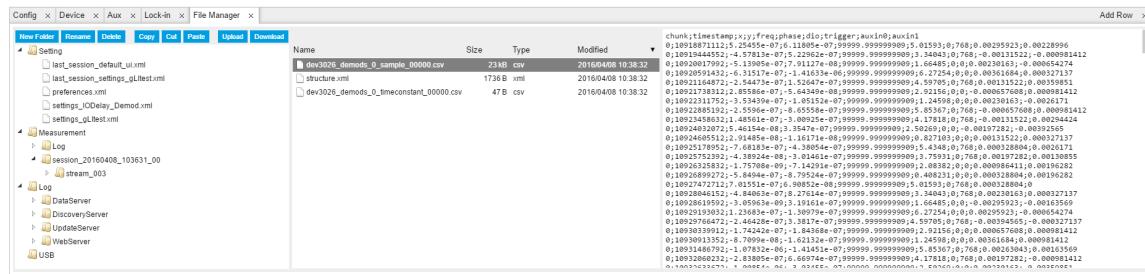


Figure 4.6. Browsing and inspecting files in the LabOne File Manager tab

For each of the selected nodes, at least one file is created. Its location is indicated in the Folder field of the Record Data section. For longer recording periods, LabOne may distribute the data over several files. The size of the files can be controlled using the Window Length parameter in the Settings of the [Plotter](#) tab.

The **File Manager (Files)** tab is a good place to inspect the resulting CSV data files. The file browser on the left of the tab allows you to navigate to the location of the data files and gives you the usual 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 4.6](#) 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. Lock-in Tab

This tab is the main lock-in amplifier control panel. Users with instruments with HF2-MF Multi-frequency option installed are kindly referred to [Section 4.3](#).

4.2.1. Features

- Functional block diagram with access to main input, output and demodulator controls
- Parameter table with main input, output and demodulator controls
- Control elements for 6 configurable demodulators
- Auto ranging, scaling, arbitrary input units for both input channels
- Control for 2 oscillators
- Settings for main signal inputs and signal outputs
- Flexible choice of reference source, trigger options and data transfer rates

4.2.2. Description

The Lock-in tab is the main control center of the instrument and open after start up by default. 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 4.10. App icon and short description

Control/Tool	Option/Range	Description
Lock-in		Quick overview and access to all the settings and properties for signal generation and demodulation.

The default view of the Lock-in tab is the parameter table view. It is accessible under the side-tab labeled All and provides controls for all demodulators in the instrument. Moreover, for each individual demodulator there is a functional block diagram available. It is accessible under the side-tab labeled with the corresponding demodulator number.

Parameter Table

The parameter table (see [Figure 4.7](#)) consists of 4 vertical sections: Signal Inputs, Oscillators, Demodulators and Signal Outputs. The Demodulators section gives access to all the settings of demodulators 1 to 6 that can be used for measurement, and of demodulators 7 and 8 that can be used for external referencing. Demodulators 1 to 3 (4 to 6) are connected to Signal Input 1 (2).

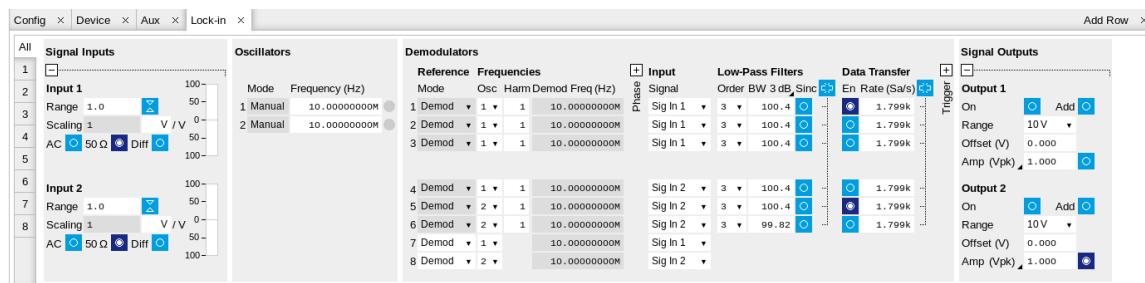


Figure 4.7. LabOne User Interface Lock-in tab - Parameter table (All)

The **Signal Inputs section** allows the user to define all relevant settings specific to the signal entered as for example input coupling, range, etc. Some of the available options like phase adjustment and the trigger functionality are collapsed by default. It takes one mouse click on the "+" icon in order to expand those controls. On the right-hand side of the Lock-in tab the **Signal Outputs section** allows defining signal amplitudes, offsets and range values.

Below the Scaling field there is the AC/DC button and the $50\ \Omega/1\ M\Omega$. The AC/DC button sets the coupling type: AC coupling has a high-pass cutoff frequency that can be used to block large DC signal components to prevent input signal saturation during amplification. The $50\ \Omega/1\ M\Omega$ button toggles the input impedance between low ($50\ \Omega$) and high (approx.) input impedance. With $50\ \Omega$ input impedance, one will expect a reduction of a factor of 2 in the measured signal if the signal source also has an output impedance of $50\ \Omega$. Next to the $50\ \Omega$ button, there is the Diff button which switches the Signal Input between a single-ended measurement on the + Input and a differential measurement on the + and - Inputs.

The **Oscillator section** indicates the frequencies of both internal oscillators. Where the Mode indicator shows Manual, the user can define the oscillator frequency manually defined by typing a frequency value in the field. In case the oscillator is referenced to an external source, the Mode indicator will show ExtRef and the frequency field is set to read-only. External reference requires a PLL to do the frequency mapping onto an internal oscillator. Successful locking is indicated by a green light right next to the frequency field. When the Modulation unit or the PID controller determine the frequency value of an oscillator, MOD or PID are indicated in the Mode field and the user cannot change the frequency manually.

In the following, we discuss the **Demodulators settings** in more detail. The block diagram displayed in [Figure 4.8](#) indicates the main demodulator components and their interconnection. The understanding of the wiring is essential for successfully operating the instrument.

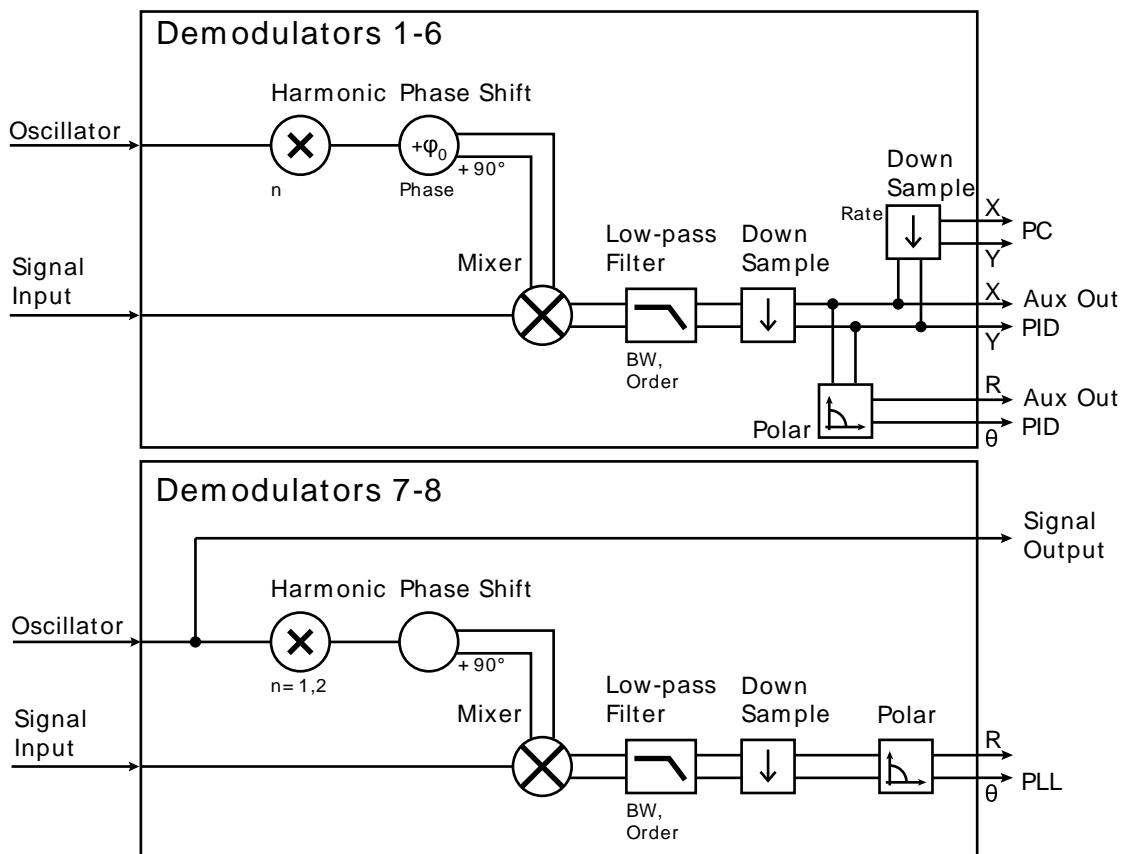


Figure 4.8. Demodulator block diagram without HF2-MF Multi-frequency option.

The first 6 lines in the Demodulators section represent the demodulators available for measurements. The Mode column is read-only set to internal reference (Demod). The 7th and 8th line represent additional demodulators that are reserved for the exclusive use as phase detectors when the mode is switched to external reference (ExtRef). In ExtRef mode, the user can select from a number of different inputs to be used as external reference signal. However, the demodulators 7 and 8 do not produce any output data that could be used for measurements.

In the Input Signal column one selects the signal that is taken as input for a given demodulator among the two Signal Inputs.

For each demodulator an additional phase shift can be introduced to the associated oscillator by entering the phase offset in the Phase column. This phase is added both to the reference channel and to the output of the demodulator. Hence, when the frequency is generated and detected using the same demodulator, signal phase and reference phase change by the same amount and no change will be visible in the demodulation result. Demodulation of frequencies that are integer multiples of any of the oscillator frequencies is achieved by entering the desired factor in the Harm column. The result of the demodulation, i.e. the amplitude and phase can be read e.g. using the Numeric tab which is described in [Section 4.4](#).

In the middle of the Lock-in tab is the Low-Pass Filters section where the filter order can be selected in the drop-down list for each demodulator and the filter bandwidth (BW 3dB) can be chosen by typing a numerical value. Alternatively, the time constant of the filter (TC) or the noise equivalent power filter bandwidth (BW NEP) can be chosen by clicking on the column's header. For example, setting the filter order to 4 corresponds to a roll off of 24 dB/oct or 80 dB/dec i.e. an attenuation of 10^4 for a tenfold frequency increase. If the Low-Pass Filter bandwidth is comparable to or larger than the demodulation frequency, the demodulator output may contain frequency components at the frequency of demodulation and its higher harmonics. In this case, the additional Sinc Filter should be enabled. It attenuates those unwanted harmonic components in the demodulator output. The Sinc Filter is useful when measuring at low frequencies, since it allows one to apply a Low-Pass Filter bandwidth closer to the demodulation frequency, thus speeding up the measurement time.

The data transfer of demodulator outputs is activated by the En button in the Data Transfer section where also the sampling rate (Rate) for each demodulator can be defined.

The Trigger section next to the Data Transfer allows for setting trigger conditions in order to control and initiate data transfer from the Instrument to the host PC by the application of logic signals (e.g. TTL) to either DIO 0 or 1 on the instrument back panel.

In the **Signal Outputs section** the On buttons are used to activate the Signal Outputs. This is also the place where the output amplitudes for the Signal Outputs can be set in adjustable units (Vpk, Vrms, or dBm). The Range drop-down list is used to select the proper output range setting. By enabling the Add button, one can add an external analog signal which is applied to the Add input to the Signal Output.

Block Diagram

The block diagram view of the main instrument functions is also sometimes called the "Graphical Lock-in Tab". A set of indexed side-tabs in the Lock-in Tab give access to a block diagram for each demodulator. The block diagrams are fully functional and provide the user with a visual feedback of what is going on inside the instrument. All control elements that are available in the Parameter Table detailed in the previous section are also present in the graphical representation.

The block diagram in [Figure 4.9](#) shows the signal path through the instrument for the case when the internal oscillator is used as reference. The Signal Inputs and Reference/Internal Frequency are shown on the left-hand side. The actual demodulation, i.e. the mixing and low-pass filtering is represented in the center of the tab. On the bottom right the user can set Signal Output parameters. On the top right there are the settings related to the output of the measurement data, either by digital means (PC Data Transfer) or by analog means (Auxiliary Outputs 1 to 4).

4.2. Lock-in Tab

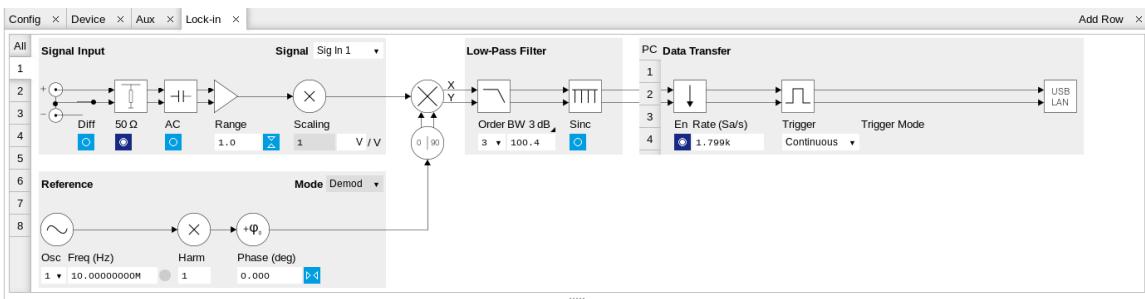


Figure 4.9. LabOne User Interface Lock-in tab - Graphical Lock-in tab in Internal Reference mode

The block diagram in Figure 4.10 shows the signal path through the instrument for the case when an external reference is used. This setting is only available for demodulators 7 and 8. In order to map an external frequency to oscillator 1/2 go to the Reference section of demodulator 7/8 and change the mode to ExtRef. This demodulator will then be used as a phase detector within a phase locked loop. The software will choose the appropriate filter settings according to the frequency and properties of the reference signal.

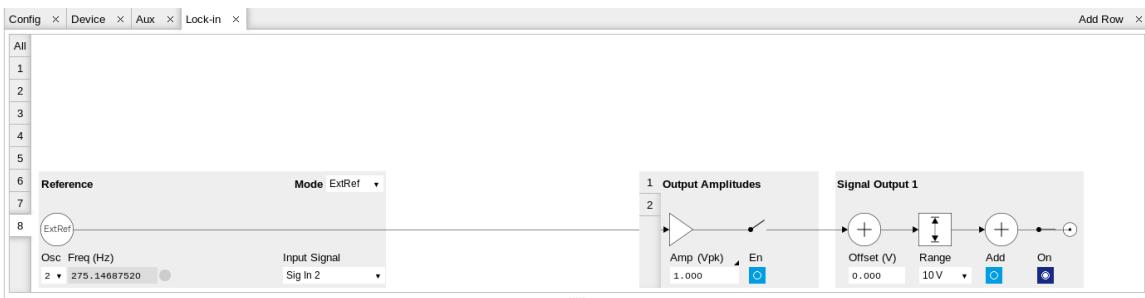


Figure 4.10. LabOne User Interface Lock-in tab - Graphical Lock-in tab in External Reference mode

4.2.3. Functional Elements

Table 4.11. Lock-in tab

Control/Tool	Option/Range	Description
Range	1 mV to 1.5 V	<p>Defines the gain of the analog input amplifier. The range should exceed the incoming signal by roughly a factor two including a potential DC offset.</p> <p>Note 1: the value inserted by the user may be approximated to the nearest value supported by the Instrument. Note 2: a proper choice of range setting is crucial in order to achieve good accuracy and best possible signal to noise ratio as it targets to use the full dynamic range of the input ADC.</p>

Control/Tool	Option/Range	Description
Auto		Automatic adjustment of the Range to about two times the maximum signal input amplitude measured over about 100 ms.
Scaling	numeric value	Applies an arbitrary scale factor to the input signal.
Measurement Unit	unit acronym	Defines the physical unit of the input signal. Use *, / and ^ operators, e.g., m or m/s^2. The value in this field modifies the readout of all measurement tools in the user interface. Typical uses of this field is to make measurements in the unit before the sensor/transducer, e.g. to take an transimpedance amplifier into account and to directly read results in Ampere instead of Volts.
AC	ON: AC coupling	Defines the input coupling for the Signal Inputs. AC coupling inserts a high-pass filter.
	OFF: DC coupling	
50 Ω	ON: 50 Ω	Switches between 50 Ω (ON) and 1 MΩ (OFF).
	OFF: 1 MΩ	
Diff	OFF: Single ended voltage input	Switches between single ended (OFF) and differential (ON) measurements.
	ON: Differential voltage input	
Mode		Indicates how the frequency of the corresponding oscillator is controlled (manual, external reference, PLL, PID). Read only flag.
	Manual	The user setting defines the oscillator frequency.
	ExtRef	An external reference is mapped onto the oscillator frequency.
	PLL	The HF2-PLL option controls the oscillator frequency.
	PID	The HF2-PID option controls the oscillator frequency.
Frequency (Hz)	0 to 50 MHz	Frequency control for each oscillator.
Locked	ON / OFF	Oscillator locked to external reference when turned on.
Mode		Indicates the unit that uses the demodulator (Demod

Control/Tool	Option/Range	Description
		stands for regular lock-in amplifier, external reference, PLL)
	Demod	Default operating mode with demodulator used for lock-in demodulation.
	ExtRef	The demodulator is used for external reference mode and tracks the frequency of the selected reference input.
	PLL	The demodulator is used in PLL mode for frequency tracking of the signal. Note this function requires the HF2-PLL option to be installed and active on your instrument.
	Mod	The demodulator is used by the HF2-MOD option, e.g. for the direct demodulation of carrier and sideband signals.
Osc	oscillator index	Connects the selected oscillator with the demodulator corresponding to this line. Number of available oscillators depends on the installed options.
Harm	1 to 1023	<p>Multiplies the demodulator's reference frequency with the integer factor defined by this field.</p> <p>Multiplies the demodulator's reference frequency by an integer factor. If the demodulator is used as a phase detector in external reference mode (PLL), the effect is that the internal oscillator locks to the external frequency divided by the integer factor.</p>
Demod Freq (Hz)	0 to 50 MHz	<p>Indicates the frequency used for demodulation and for output generation.</p> <p>The demodulation frequency is calculated with oscillator frequency times the harmonic factor. When the HF2LI-MOD option is used linear combinations of oscillator frequencies including the</p>

Control/Tool	Option/Range	Description
		harmonic factors define the demodulation frequencies.
Phase (deg)	-180° to 180°	<p>Phase shift applied to the reference input of the demodulator.</p> <p>When the HF2LI-MF option is used, the phase shift is also applied to the Signal Outputs.</p>
Zero		<p>Adjust the demodulator phase automatically in order to read zero degrees.</p> <p>Shifts the phase of the reference at the input of the demodulator in order to achieve zero phase at the demodulator output. This action maximizes the X output, zeros the Y output, zeros the Θ output, and leaves the R output unchanged.</p>
Signal		Selects the signal source to be associated to the demodulator.
	Sig In 1	Signal Input 1 is connected to the corresponding demodulator.
	Sig In 2	Signal Input 2 is connected to the corresponding demodulator.
	Aux In 1	Auxiliary Input 1 is connected to the corresponding demodulator. The input bandwidth is limited to 20 kHz.
	Aux In 2	Auxiliary Input 2 is connected to the corresponding demodulator. The input bandwidth is limited to 20 kHz.
	DIO D0	DIO D0 is connected to the corresponding demodulator. The input bandwidth is limited to 2 MHz.
	DIO D1	DIO D1 is connected to the corresponding demodulator. The input bandwidth is limited to 2 MHz.
	Aux Out 3	Auxiliary Output 3 is connected to the corresponding demodulator.

Control/Tool	Option/Range	Description
	Aux Out 4	Auxiliary Output 4 is connected to the corresponding demodulator.
	Aux In 1	Auxiliary Input 1 is connected to the corresponding demodulator.
	Aux In 2	Auxiliary Input 2 is connected to the corresponding demodulator.
Order		Selects the filter roll off between 6 dB/oct and 48 dB/oct.
	1	1st order filter 6 dB/oct
	2	2nd order filter 12 dB/oct
	3	3rd order filter 18 dB/oct
	4	4th order filter 24 dB/oct
	5	5th order filter 30 dB/oct
	6	6th order filter 36 dB/oct
	7	7th order filter 42 dB/oct
	8	8th order filter 48 dB/oct
TC/BW Select		Defines the display unit of the low-pass filters: time constant (TC), noise equivalent power bandwidth (BW NEP), 3 dB bandwidth (BW 3 dB).
	TC	Defines the low-pass filter characteristic using time constant of the filter.
	BW NEP	Defines the low-pass filter characteristic using the noise equivalent power bandwidth of the filter.
	BW 3 dB	Defines the low-pass filter characteristic using the cut-off frequency of the filter.
TC/BW Value	numeric value	Defines the low-pass filter characteristic in the unit defined above.
Sinc	ON / OFF	<p>Enables the sinc filter.</p> <p>When the filter bandwidth is comparable to or larger than the demodulation frequency, the demodulator output may contain frequency components at the frequency of demodulation and its higher harmonics. The sinc is an additional filter</p>

Control/Tool	Option/Range	Description
		that attenuates these unwanted components in the demodulator output.
Lock		Makes all demodulators filter settings equal (order, time constant, bandwidth). Pressing the lock copies the settings from demodulator one into the settings of all demodulators. When the lock is pressed, any modification to a field is immediately changing all other settings. Releasing the lock does not change any setting, and permits to individually adjust the filter settings for each demodulator.
Enable Streaming		Enables the data acquisition for the corresponding demodulator. Note: increasing number of active demodulators increases load on physical connection to the host computer.
	ON: demodulator active	Enables the streaming of demodulated samples in real time to the host computer. The streaming rate is defined in the field on the right hand side. As a consequence demodulated samples can be visualized on the plotter and a corresponding numeric entry in the numerical tool is activated.
	OFF: demodulator inactive	Disables the streaming of demodulated samples to the host computer.
Rate (Sa/s)	0.22 Sa/s to 410 kSa/s	Defines the demodulator sampling rate, the number of samples that are sent to the host computer per second. A rate of about 7-10 higher as compared to the filter bandwidth usually provides sufficient aliasing suppression. This is also the rate of data received by LabOne Data Server and saved to the computer hard disk. This setting has no impact on the

Control/Tool	Option/Range	Description
		sample rate on the auxiliary outputs connectors. Note: the value inserted by the user may be approximated to the nearest value supported by the instrument.
Demodulator Output Rate Lock		Makes all demodulator output rates equal. Pressing the lock copies the settings from demodulator one into the settings of all demodulators. When the lock is pressed, any modification to a field is immediately changing all other settings. Releasing the lock does not change any setting, and permits to individually adjust the demodulator output rate for each demodulator.
Trigger		Selects the acquisition mode of demodulated samples. Continuous trigger means data are streamed to the host computer at the Rate indicated.
	Continuous	Selects continuous data acquisition mode. The demodulated samples are streamed to the host computer at the Rate indicated on the left hand side. In continuous mode the numerical and plotter tools are continuously receiving and display new values.
	DIO 0	Samples are sent to the host computer depending on DIO 0 triggering.
	DIO 1	Samples are sent to the host computer depending on DIO 1 triggering.
	DIO 0 1	Samples are sent to the host computer depending on DIO 0 and 1 triggering.
Trig Mode		Defines the edge or level trigger mode for the selected Trigger input. Note: this field only appears when a non-continuous trigger is selected in the Trigger field.

Control/Tool	Option/Range	Description
	Rising	Selects triggered sample acquisition mode on rising edge of the selected Trigger input.
	Falling	Selects triggered sample acquisition mode on falling edge of the selected Trigger input.
	Both	Selects triggered sample acquisition mode on both edges of the selected Trigger input.
	High	Selects continuous sample acquisition mode on high level of the selected Trigger input. In this selection, the sample rate field determines the frequency in which demodulated samples are sent to the host computer.
	Low	Selects continuous sample acquisition mode on low level of the selected Trigger input. In this selection, the sample rate field determines the frequency in which demodulated samples are sent to the host computer.
Amplitude Unit	Vpk, Vrms, dBm	Select the unit of the displayed amplitude value.
Amp Enable	ON / OFF	Enables individual output signal amplitude. When the HF2LI-MF option is used, it is possible to generate signals being the linear combination of the available demodulator frequencies.
On	ON / OFF	Main switch for the Signal Output corresponding to the blue LED indicator on the instrument front panel.
Range		Defines the maximum output voltage that is generated by the corresponding Signal Output. This includes the potential multiple Signal Amplitudes and Offsets summed up. Select the smallest range possible to optimize signal quality.

Control/Tool	Option/Range	Description
		This setting ensures that no levels or peaks above the setting are generated, and therefore it limits the values that can be entered as output amplitudes. Therefore selected output amplitudes are clipped to the defined range and the clipping indicator turns on. If $50\ \Omega$ target source or differential output is enabled the possible maximal output range will be half.
	10 mV	Selects output range $\pm 10\text{ mV}$.
	100 mV	Selects output range $\pm 100\text{ mV}$.
	1 V	Selects output range $\pm 1\text{ V}$.
	10 V	Selects output range $\pm 10\text{ V}$.
Offset	-range to range	Defines the DC voltage that is added to the dynamic part of the output signal.
Add	ON / OFF	The signal supplied to the add input is added to the signal output.
Output	-range to range	Defines the output amplitude for each demodulator frequency as rms or peak-to-peak value. A negative amplitude value is equivalent to a phase change of 180 degree. Demodulator 7 is the signal source for Signal Output 1, demodulator 8 is the source for Signal Output 2.

4.3. Lock-in Tab (HF2-MF option)

This tab is the main lock-in amplifier control panel for HF2 Instruments with the HF2-MF Multi-frequency option installed. Users with instruments without this option installed are kindly referred to [Section 4.2](#).

4.3.1. Features

- Functional block diagram with access to main input, output and demodulator controls
- Parameter table with main input, output and demodulator controls
- Controls for 6 (HF2LI) or 8 (HF2IS) individually configurable demodulators
- Auto ranging, scaling, arbitrary input units for both input channels
- Control for 6 oscillators
- Settings for main signal inputs and signal outputs
- Choice of reference source, trigger options and data transfer rates

4.3.2. Description

The Lock-in tab is the main control center of the instrument and open after start up by default. 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.

The default view of the Lock-in tab is the parameter table view. It is accessible under the side-tab labeled All and provides controls for all demodulators in the instrument. Moreover, for each individual demodulator there is a functional block diagram available. It is accessible under the side-tab labeled with the corresponding demodulator number.

Parameter Table

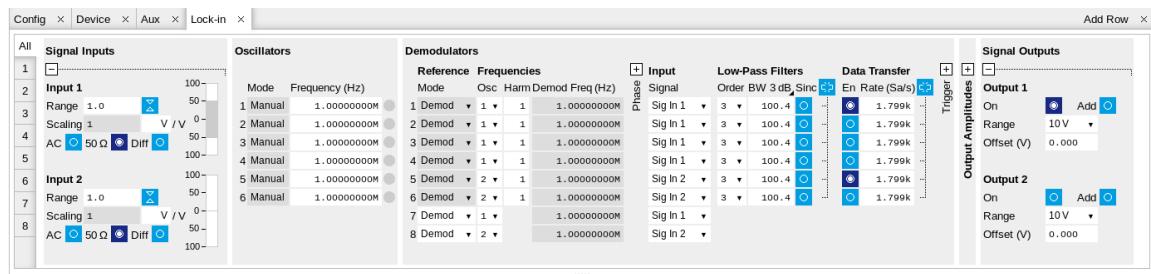


Figure 4.11. LabOne User Interface Lock-in tab with HF2-MF Multi-frequency option.

The **Signal Inputs section** allows the user to define all relevant settings specific to the signal entered as for example input coupling, range, etc. Some of the available options like phase adjustment and the trigger functionality are collapsed by default. It takes one mouse click on the "+" icon in order to expand those controls. On the right-hand side of the Lock-in tab the Signal Outputs section allows to define signal amplitudes, offsets and range values.

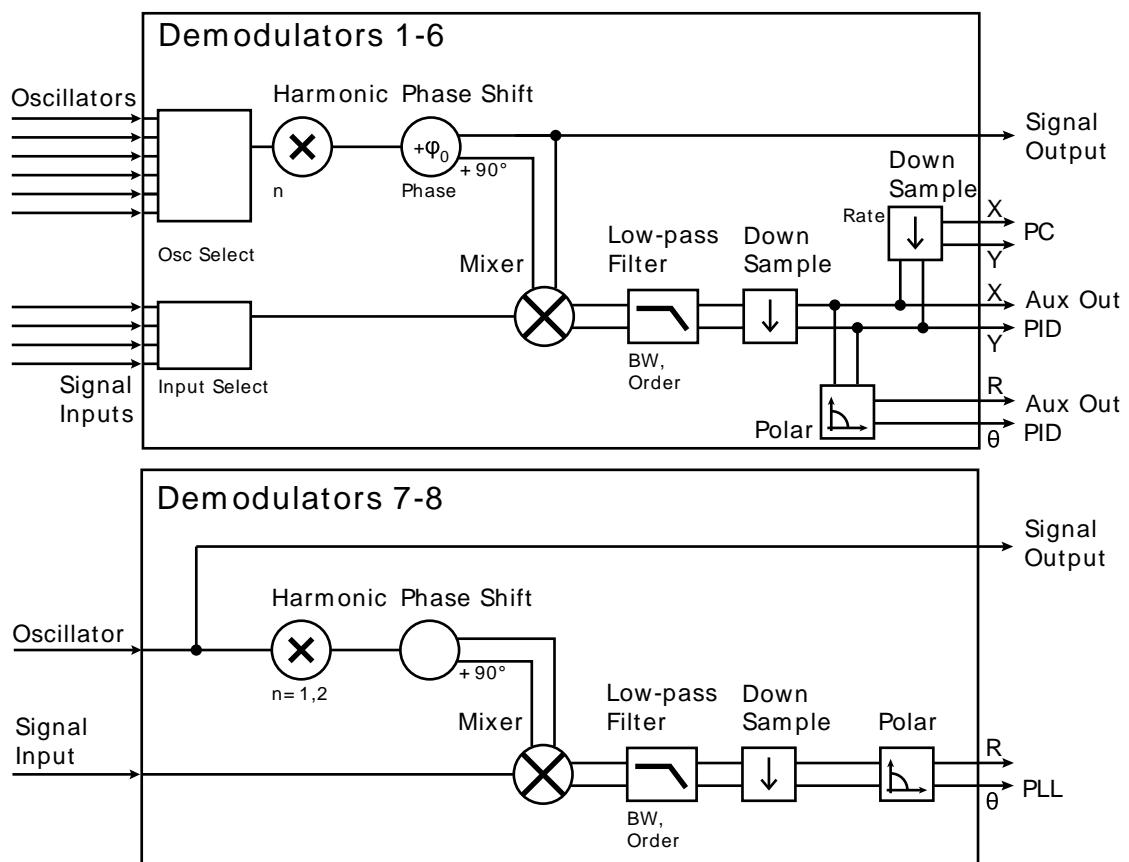
The Scaling field below the Range field can be used to multiply the Signal Input data for instance to account for the gain of an external amplifier. In case there is a transimpedance gain of 10 V/A applied to the input signal externally, then the Scaling field can be set to 0.1 and the Units field can be set to A in order to show the actual current readings through the entire user interface.

There are two buttons below the Scaling field that can be toggled: the AC/DC button and the 50 Ω/. The AC/DC button sets the coupling type: AC coupling has a high-pass cutoff frequency

that can be used to block large DC signal components to prevent input signal saturation during amplification. The $50\ \Omega$ /button toggles the input impedance between low ($50\ \Omega$) and high (approx.) input impedance. With $50\ \Omega$ input impedance, one will expect a reduction of a factor of 2 in the measured signal if the signal source also has an output impedance of $50\ \Omega$.

The **Oscillator section** indicates the . Where the Mode indicator shows Manual the user can define the oscillator frequency manually defined by typing a frequency value in the field. In case the oscillator is referenced to an external source the Mode indicator will show ExtRef and the frequency field is set to read-only. External reference requires a PLL to do the frequency mapping onto an internal oscillator. Successful locking is indicated by a green light right next to the frequency field.

The next section contains the **Demodulators settings**. The block diagram displayed in [Figure 4.12](#) indicates the main demodulator components and their interconnection. The understanding of the wiring is essential for successfully operating the instrument.



[Figure 4.12. Demodulator block diagram with HF2-MF Multi-frequency option.](#)

In the Input Signal column one defines the signal that is taken as input for the demodulator. A wide choice of signals can be selected: Signal Inputs, the Trigger Inputs, the Auxiliary Inputs and Auxiliary Outputs. This allows to use the instrument for many different measurement topologies.

For each demodulator an additional phase shift can be introduced to the associated oscillator by entering the phase offset in the Phase column. This phase is added both, to the reference channel and the output of the demodulator. Hence, when the frequency is generated and detected using the same demodulator, signal phase and reference phase change by the same amount and no change will be visible in the demodulation result. Demodulation of frequencies that are integer multiples of any of the oscillator frequencies is achieved by entering the desired factor in the Harm

column. The demodulator readout can be obtained using the Numeric tab which is described in [Section 4.4](#).

In the middle of the Lock-in tab is the Low-Pass Filters section where the filter order can be selected in the drop down list for each demodulator and the filter bandwidth (BW 3dB) can chosen by typing a numerical value. Alternatively the time constant of the filter (TC) or the noise equivalent power filter bandwidth (BW NEP) can be chosen by clicking on the column's header. For example, setting the filter order to 4 corresponds to a roll off of 24 dB/oct or 80 dB/dec i.e. an attenuation of 10^4 for a tenfold frequency increase. If the Low-Pass Filter bandwidth is comparable to or larger than the demodulation frequency, the demodulator output may contain frequency components at the frequency of demodulation and its higher harmonics. In this case, the additional Sinc Filter can be enabled. It attenuates those unwanted harmonic components in the demodulator output. The Sinc Filter is also useful when measuring at low frequencies, since it allows to apply a Low-Pass Filter bandwidth closer to the demodulation frequency, thus speeding up the measurement time.

The data transfer of demodulator outputs is activated by the En button in the Data Transfer section where also the sampling rate (Rate) for each demodulator can be defined.

The Trigger section next to the Data Transfer allows for setting trigger conditions in order to control and initiate data transfer from the Instrument to the host PC by the application of logic signals (e.g. TTL) to either Trigger Input on the back panel.

Block Diagram

The block diagram view of the main instrument functions is also sometimes referred to as the "Graphical Lock-in Tab". Depending how many demodulators are available in the instrument a set of numbered side-tabs occur giving access to a Graphical Lock-in Tab for each demodulator. The block diagrams are fully functional and provide the user with a visual feedback of what is going on inside the instrument. All control elements that are available in the Parameter Table detailed in the previous section are also present in the graphical representation.

The block diagram in [Figure 4.13](#) describes the signal path throughout the instrument for the case when the internal oscillator is used as reference. In this case the tab consists of 6 functional sections. The Signal Inputs and Reference/Internal Frequency are described on the left side, the core of demodulation with the mixer and low-pass filter is located in the center of the tab and the Signal Outputs, the Auxiliary Outputs as well as the data transfer to the PC is sketched on the right.

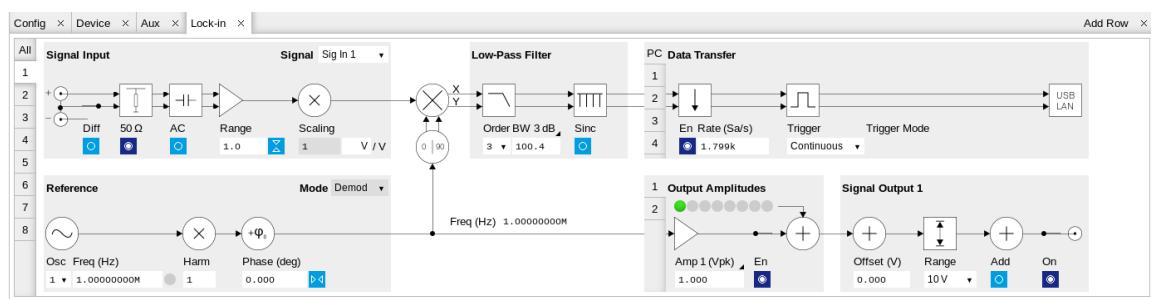


Figure 4.13. LabOne User Interface Lock-in tab - Graphical Lock-in tab in Internal Reference mode

The block diagram in [Figure 4.14](#) describes the signal path throughout the instrument for the case when an external reference is used. This setting is only available for demodulators . In order to map an external frequency to any of the oscillators, go to the Reference section of demodulator and change the mode to ExtRef. This demodulator will then be used as a phase detector within the phase-locked loop. The software will choose the appropriate filter settings according to the frequency and properties of the reference signal. Once a demodulator is used to map an external frequency on to one of the internal oscillators, it is no longer available for other measurements.

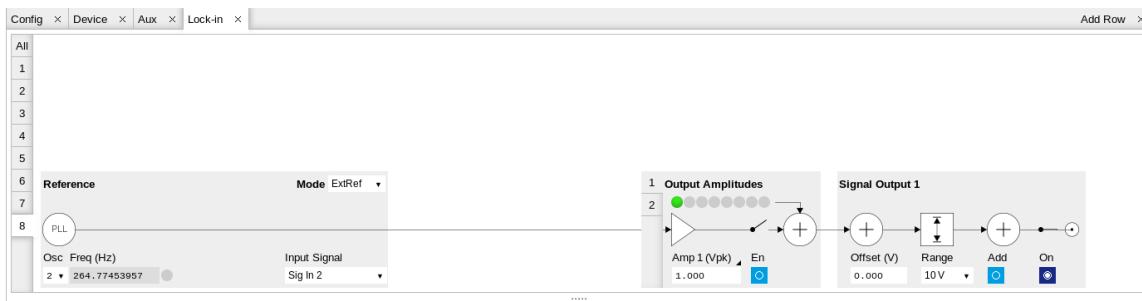


Figure 4.14. LabOne User Interface Lock-in tab - Graphical Lock-in tab in External Reference mode

4.3.3. Functional Elements

Table 4.12. Lock-in MF tab

Control/Tool	Option/Range	Description
Range	1 mV to 1.5 V	Defines the gain of the analog input amplifier. The range should exceed the incoming signal by roughly a factor two including a potential DC offset. Note 1: the value inserted by the user may be approximated to the nearest value supported by the Instrument. Note 2: a proper choice of range setting is crucial in order to achieve good accuracy and best possible signal to noise ratio as it targets to use the full dynamic range of the input ADC.
Auto	☒	Automatic adjustment of the Range to about two times the maximum signal input amplitude measured over about 100 ms.
Scaling	numeric value	Applies an arbitrary scale factor to the input signal.
Measurement Unit	unit acronym	Defines the physical unit of the input signal. Use *, / and ^ operators, e.g., m or m/s^2. The value in this field modifies the readout of all measurement tools in the user interface. Typical uses of this field is to make measurements in the unit before the sensor/transducer, e.g. to take an transimpedance amplifier into

Control/Tool	Option/Range	Description
		account and to directly read results in Ampere instead of Volts.
AC	ON: AC coupling	Defines the input coupling for the Signal Inputs. AC coupling inserts a high-pass filter.
	OFF: DC coupling	
50 Ω	ON: 50 Ω	Switches between 50 Ω (ON) and 1 MΩ (OFF).
	OFF: 1 MΩ	
Diff	OFF: Single ended voltage input	Switches between single ended (OFF) and differential (ON) measurements.
	ON: Differential voltage input	
Mode		Indicates how the frequency of the corresponding oscillator is controlled (manual, external reference, PLL, PID). Read only flag.
	Manual	The user setting defines the oscillator frequency.
	ExtRef	An external reference is mapped onto the oscillator frequency.
	PLL	The HF2-PLL option controls the oscillator frequency.
	PID	The HF2-PID option controls the oscillator frequency.
Frequency (Hz)	0 to 50 MHz	Frequency control for each oscillator.
Locked	ON / OFF	Oscillator locked to external reference when turned on.
Mode		Indicates the unit that uses the demodulator (Demod stands for regular lock-in amplifier, external reference, PLL)
	Demod	Default operating mode with demodulator used for lock-in demodulation.
	ExtRef	The demodulator is used for external reference mode and tracks the frequency of the selected reference input.
	PLL	The demodulator is used in PLL mode for frequency tracking of the signal. Note this function requires the HF2-PLL option to be installed and active on your instrument.
	Mod	The demodulator is used by the HF2-MOD option, e.g. for

Control/Tool	Option/Range	Description
		the direct demodulation of carrier and sideband signals.
Osc	oscillator index	Connects the selected oscillator with the demodulator corresponding to this line. Number of available oscillators depends on the installed options.
Harm	1 to 1023	<p>Multiplies the demodulator's reference frequency with the integer factor defined by this field.</p> <p>Multiplies the demodulator's reference frequency by an integer factor. If the demodulator is used as a phase detector in external reference mode (PLL), the effect is that the internal oscillator locks to the external frequency divided by the integer factor.</p>
Demod Freq (Hz)	0 to 50 MHz	<p>Indicates the frequency used for demodulation and for output generation.</p> <p>The demodulation frequency is calculated with oscillator frequency times the harmonic factor. When the HF2LI-MOD option is used linear combinations of oscillator frequencies including the harmonic factors define the demodulation frequencies.</p>
Phase (deg)	-180° to 180°	<p>Phase shift applied to the reference input of the demodulator.</p> <p>When the HF2LI-MF option is used, the phase shift is also applied to the Signal Outputs.</p>
Zero		<p>Adjust the demodulator phase automatically in order to read zero degrees.</p> <p>Shifts the phase of the reference at the input of the demodulator in order to achieve zero phase at the demodulator output. This action maximizes the X output, zeros the Y output, zeros the</p>

Control/Tool	Option/Range	Description
		Θ output, and leaves the R output unchanged.
Signal		Selects the signal source to be associated to the demodulator.
	Sig In 1	Signal Input 1 is connected to the corresponding demodulator.
	Sig In 2	Signal Input 2 is connected to the corresponding demodulator.
	Aux In 1	Auxiliary Input 1 is connected to the corresponding demodulator. The input bandwidth is limited to 20 kHz.
	Aux In 2	Auxiliary Input 2 is connected to the corresponding demodulator. The input bandwidth is limited to 20 kHz.
	DIO D0	DIO D0 is connected to the corresponding demodulator. The input bandwidth is limited to 2 MHz.
	DIO D1	DIO D1 is connected to the corresponding demodulator. The input bandwidth is limited to 2 MHz.
	Aux Out 3	Auxiliary Output 3 is connected to the corresponding demodulator.
	Aux Out 4	Auxiliary Output 4 is connected to the corresponding demodulator.
	Aux In 1	Auxiliary Input 1 is connected to the corresponding demodulator.
	Aux In 2	Auxiliary Input 2 is connected to the corresponding demodulator.
Order		Selects the filter roll off between 6 dB/oct and 48 dB/oct.
	1	1st order filter 6 dB/oct
	2	2nd order filter 12 dB/oct
	3	3rd order filter 18 dB/oct
	4	4th order filter 24 dB/oct
	5	5th order filter 30 dB/oct
	6	6th order filter 36 dB/oct

Control/Tool	Option/Range	Description
	7	7th order filter 42 dB/oct
	8	8th order filter 48 dB/oct
TC/BW Select		Defines the display unit of the low-pass filters: time constant (TC), noise equivalent power bandwidth (BW NEP), 3 dB bandwidth (BW 3 dB).
	TC	Defines the low-pass filter characteristic using time constant of the filter.
	BW NEP	Defines the low-pass filter characteristic using the noise equivalent power bandwidth of the filter.
	BW 3 dB	Defines the low-pass filter characteristic using the cut-off frequency of the filter.
TC/BW Value	numeric value	Defines the low-pass filter characteristic in the unit defined above.
Sinc	ON / OFF	<p>Enables the sinc filter.</p> <p>When the filter bandwidth is comparable to or larger than the demodulation frequency, the demodulator output may contain frequency components at the frequency of demodulation and its higher harmonics. The sinc is an additional filter that attenuates these unwanted components in the demodulator output.</p>
Lock		<p>Makes all demodulators filter settings equal (order, time constant, bandwidth).</p> <p>Pressing the lock copies the settings from demodulator one into the settings of all demodulators. When the lock is pressed, any modification to a field is immediately changing all other settings. Releasing the lock does not change any setting, and permits to individually adjust the filter settings for each demodulator.</p>
Enable Streaming		Enables the data acquisition for the corresponding demodulator. Note:

Control/Tool	Option/Range	Description
		increasing number of active demodulators increases load on physical connection to the host computer.
	ON: demodulator active	Enables the streaming of demodulated samples in real time to the host computer. The streaming rate is defined in the field on the right hand side. As a consequence demodulated samples can be visualized on the plotter and a corresponding numeric entry in the numerical tool is activated.
	OFF: demodulator inactive	Disables the streaming of demodulated samples to the host computer.
Rate (Sa/s)	0.22 Sa/s to 410 kSa/s	Defines the demodulator sampling rate, the number of samples that are sent to the host computer per second. A rate of about 7-10 higher as compared to the filter bandwidth usually provides sufficient aliasing suppression. This is also the rate of data received by LabOne Data Server and saved to the computer hard disk. This setting has no impact on the sample rate on the auxiliary outputs connectors. Note: the value inserted by the user may be approximated to the nearest value supported by the instrument.
Demodulator Output Rate Lock		Makes all demodulator output rates equal. Pressing the lock copies the settings from demodulator one into the settings of all demodulators. When the lock is pressed, any modification to a field is immediately changing all other settings. Releasing the lock does not change any setting, and permits to individually adjust the demodulator output rate for each demodulator.

Control/Tool	Option/Range	Description
Trigger		Selects the acquisition mode of demodulated samples. Continuous trigger means data are streamed to the host computer at the Rate indicated.
	Continuous	Selects continuous data acquisition mode. The demodulated samples are streamed to the host computer at the Rate indicated on the left hand side. In continuous mode the numerical and plotter tools are continuously receiving and display new values.
	DIO 0	Samples are sent to the host computer depending on DIO 0 triggering.
	DIO 1	Samples are sent to the host computer depending on DIO 1 triggering.
	DIO 0 1	Samples are sent to the host computer depending on DIO 0 and 1 triggering.
Trig Mode		Defines the edge or level trigger mode for the selected Trigger input. Note: this field only appears when a non-continuous trigger is selected in the Trigger field.
	Rising	Selects triggered sample acquisition mode on rising edge of the selected Trigger input.
	Falling	Selects triggered sample acquisition mode on falling edge of the selected Trigger input.
	Both	Selects triggered sample acquisition mode on both edges of the selected Trigger input.
	High	Selects continuous sample acquisition mode on high level of the selected Trigger input. In this selection, the sample rate field determines the frequency in which demodulated samples are sent to the host computer.

Control/Tool	Option/Range	Description
	Low	Selects continuous sample acquisition mode on low level of the selected Trigger input. In this selection, the sample rate field determines the frequency in which demodulated samples are sent to the host computer.
Amplitude Unit	Vpk, Vrms, dBm	Select the unit of the displayed amplitude value.
Amp Enable	ON / OFF	Enables individual output signal amplitude. When the HF2LI-MF option is used, it is possible to generate signals being the linear combination of the available demodulator frequencies.
Amp (V)	-range to range	Defines the output amplitude for each demodulator frequency as rms or peak-to-peak value. A negative amplitude value is equivalent to a phase change of 180 degree. Linear combination of multiple amplitude settings on the same output are clipped to the range setting. Note: the value inserted by the user may be approximated to the nearest value supported by the Instrument.
AWG	AWG is ON	Indicates that the output amplitude is generated by the AWG.
On	ON / OFF	Main switch for the Signal Output corresponding to the blue LED indicator on the instrument front panel.
Range		Defines the maximum output voltage that is generated by the corresponding Signal Output. This includes the potential multiple Signal Amplitudes and Offsets summed up. Select the smallest range possible to optimize signal quality. This setting ensures that no levels or peaks above

Control/Tool	Option/Range	Description
		the setting are generated, and therefore it limits the values that can be entered as output amplitudes. Therefore selected output amplitudes are clipped to the defined range and the clipping indicator turns on. If $50\ \Omega$ target source or differential output is enabled the possible maximal output range will be half.
	10 mV	Selects output range $\pm 10\text{ mV}$.
	100 mV	Selects output range $\pm 100\text{ mV}$.
	1 V	Selects output range $\pm 1\text{ V}$.
	10 V	Selects output range $\pm 10\text{ V}$.
Offset	-range to range	Defines the DC voltage that is added to the dynamic part of the output signal.
Add	ON / OFF	The signal supplied to the add input is added to the signal output.

4.4. Numeric Tab

The Numeric Tab provides a powerful time domain based measurement display as introduced in [Section 4.1.2](#). It is available in all HF2 Instruments.

4.4.1. Features

- Display of demodulator output data and other streamed data, e.g. auxiliary inputs, demodulator frequencies
- Graphical and numerical range indicators
- Polar and Cartesian formats
- Support for Input Scaling and Input Units

4.4.2. Description

The numeric tab serves as the main numeric overview display of multiple measurement data. The display can be configured by both choosing the values displayed and also rearrange the display tiles by drag-and-drop. 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 4.13. App icon and short description

Control/Tool	Option/Range	Description
Numeric		Access to all continuously streamed measurement data as numerical values.

The numeric tab (see [Figure 4.15](#)) 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.

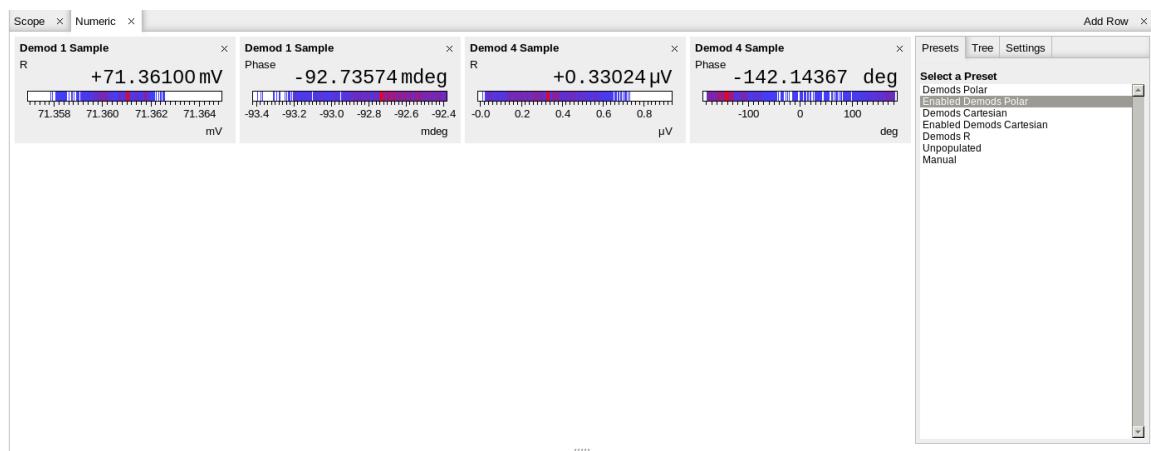


Figure 4.15. LabOne UI: Numeric tab

The numeric tab can be deployed to display the demodulated signal, phase, frequency as well as the signal levels at the auxiliary inputs. By default, the user can display the demodulated data either in polar coordinates (R, Θ) or in Cartesian coordinates (X, Y) which can be toggled using the presets. To display other measurement quantities as available from any of the presets simply

click on the tree tab next to the preset tab. The desired display fields can be selected under each demodulator's directory tree structure.

4.4.3. Functional Elements

Table 4.14. Numeric tab: Presets sub-tab

Control/Tool	Option/Range	Description
Select a Preset		Select numerical view based on a preset. Alternatively, the displayed value may also be selected based on tree elements.
	Demods Polar	Shows R and Phase of all demodulators.
	Enabled Demods Polar	Shows R and Phase of enabled demodulators.
	Demods Cartesian	Shows X and Y of all demodulators.
	Enabled Demods Cartesian	Shows X and Y of enabled demodulators.
	Demods R	Shows R of all demodulators.
	Enabled Impedances	Shows impedance, model parameters, and frequency.
	Unpopulated	Shows no signals.
	Manual	If additional signals are added or removed the active preset gets manual.

For the Tree sub-tab please see Table 4.8 in the section called “Tree Sub-Tab”.

Table 4.15. Numeric tab: Settings sub-tab

Control/Tool	Option/Range	Description
Name	text label	Name of the selected plot(s). The default name can be changed to reflect the measured signal.
Mapping		Mapping of the selected plot(s)
	Lin	Enable linear scaling.
	Log	Enable logarithmic scaling.
	dB	Enable logarithmic scaling in dB.
Scaling	Manual/Full Scale	Scaling of the selected plot(s)
Zoom To Limits		Adjust the zoom to the current limits of the displayed histogram data.
Start Value	numeric value	Start value of the selected plot(s). Only visible for manual scaling.

4.4. Numeric Tab

Control/Tool	Option/Range	Description
Stop Value	numeric value	Stop value of the selected plot(s). Only visible for manual scaling.

4.5. Plotter Tab

The Plotter is one of the powerful time domain measurement tools as introduced in [Section 4.1.2](#) and is available in all HF2 Instruments.

4.5.1. Features

- Vertical axis grouping for flexible axis scaling
- Polar and Cartesian data format
- Histogram and Math functionality for data analysis
- 4 cursors for data analysis
- Support for Input Scaling and Input Units

4.5.2. 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 4.16. 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 4.16](#)) is divided into a display section on the left and a configuration section on the right.

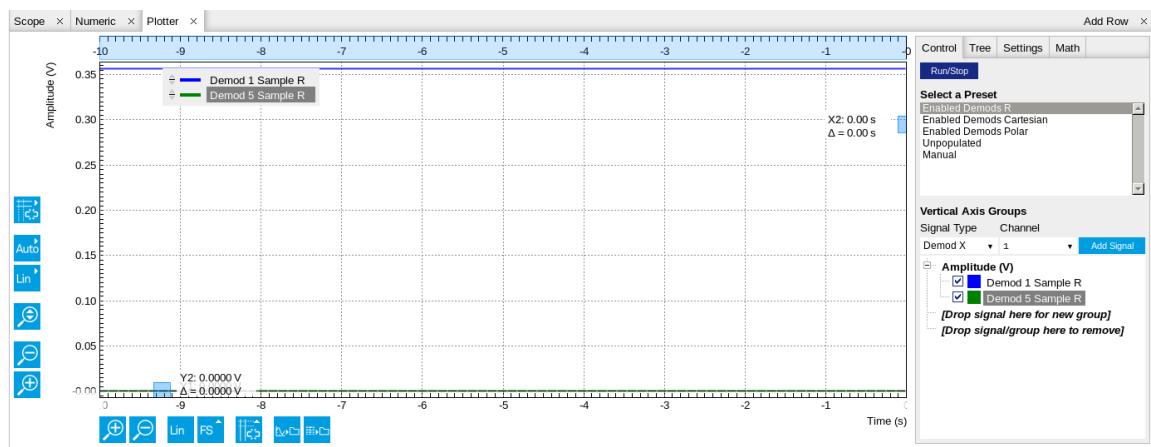


Figure 4.16. LabOne UI: Plotter tab

The Plotter can be used to monitor the evolution of demodulated data and other streamed data continuously over time. Just as in the numeric tab any continuously streamed quantity can be displayed, for instance R, Θ, X, Y, frequency, and others. 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.

The sampling rate of the demodulator data is determined by the Rate value in Sa/s set in the Lock-in tab. The Plotter data can be continuously saved to disk by clicking the record button in the Config tab which will be indicated by a green Recording (REC) LED in the status bar.

4.5.3. Functional Elements

Table 4.17. 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 4.9](#) in the section called “Vertical Axis Groups”.

For the Tree sub-tab please see [Table 4.8](#) in the section called “Tree Sub-Tab”.

Table 4.18. Plotter tab: Settings sub-tab

Control/Tool	Option/Range	Description
Window Length	10 s to 12 h	Plotter memory depth. Values larger than 10 s may cause excessive memory consumption for signals

4.5. Plotter Tab

Control/Tool	Option/Range	Description
		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.
Histogram	ON / OFF	Shows the histogram in the display.

For the Math sub-tab please see Table 4.7 in the section called “Cursors and Math”.

4.6. Scope Tab

The Scope is a powerful time domain and frequency domain measurement tool as introduced in [Section 4.1.2](#) and is available for all HF2 Instruments.

4.6.1. Features

- One input channel with 2 kSa of memory
- 14 bit nominal resolution
- Fast Fourier Transform (FFT): up to 100 MHz span, spectral density and power conversion, choice of window functions
- Sampling rates from 6.4 kSa/s to 210 MSa/s; up to 10 μ s acquisition time at 210 MSa/s or 320 ms at 6.4 kSa/s
- 4 signal sources; up to 13 trigger sources and 2 trigger methods
- Independent hold-off and trigger level settings

4.6.2. Description

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

Table 4.19. App icon and short description

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

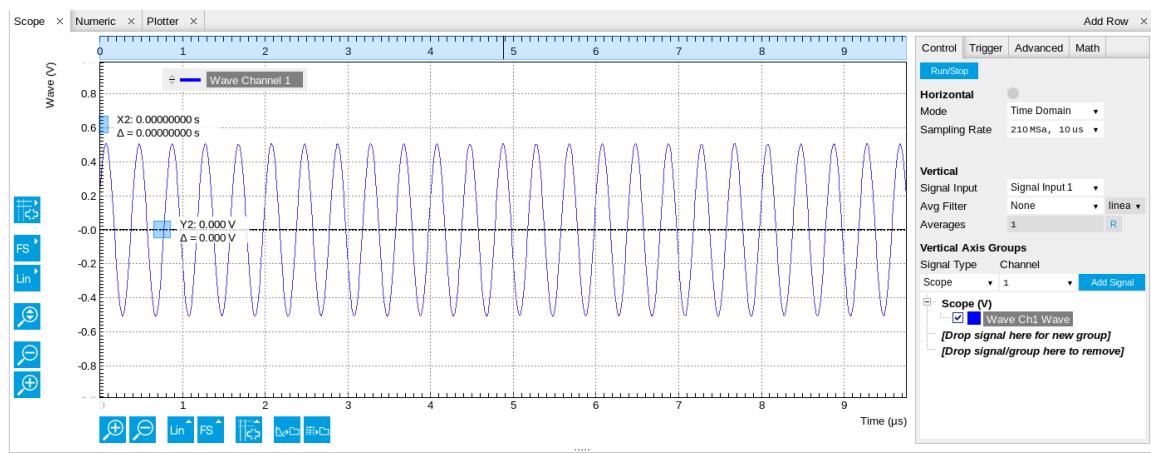


Figure 4.17. LabOne UI: Scope tab - Time domain

The Scope tab consists of a plot section on the left and a configuration section on the right. The configuration section is further divided into 4 sub-tabs. It gives access to a single-channel oscilloscope that can be used to monitor a choice of signals in the time or frequency domain. Hence the X axis of the plot area is time (for time domain display, [Figure 4.17](#)) Ar frequency (for

frequency domain display, Figure 4.19). It is possible to display the time trace and the associated FFT simultaneously by opening a second instance of the Scope tab. The Y axis displays the selected signal that can be modified and scaled using the arbitrary input unit feature found in the Lock-in tab.

The Scope records data from a single channel at up to 210 MSa/s. The channel can be selected among the two Signal Inputs and the two Signal Outputs. The Scope records data sets of up to 2 kSa samples in the standard configuration, which corresponds to an acquisition time of 10 μ s at the highest sampling rate.

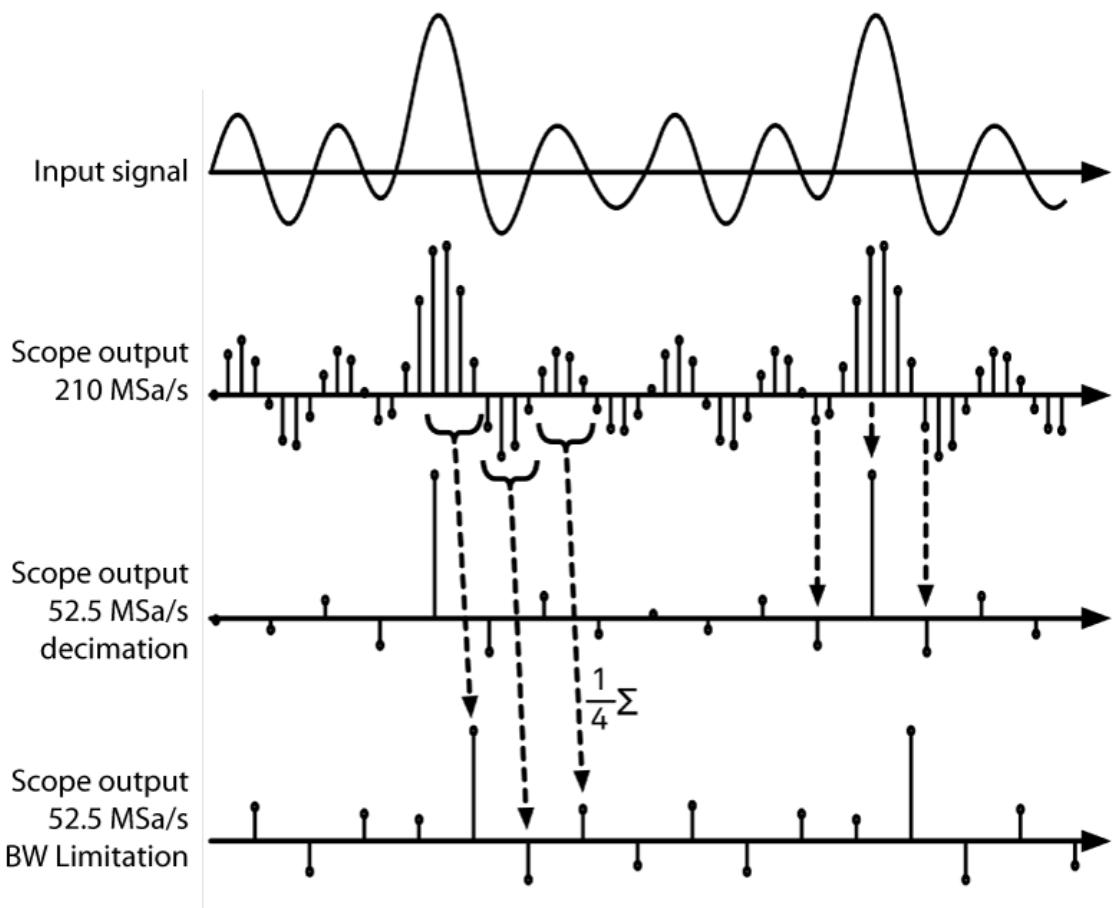


Figure 4.18. Illustration of how the Scope output is generated in BW Limitation and decimation mode when the sampling rate is reduced from the default of 210 MSa/s to 52.5 MSa/s.

The Scope also offers an averaging filter that works on a shot-to-shot basis. The functionality is implemented by means of an exponential moving average filter with configurable filter depth. The averaging filter can help suppress noise components that are uncorrelated with the main signal. It is particularly useful when the spectrum of the signal is considered as it can help to reveal harmonic signals and disturbances that might otherwise be hidden below the noise floor.

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

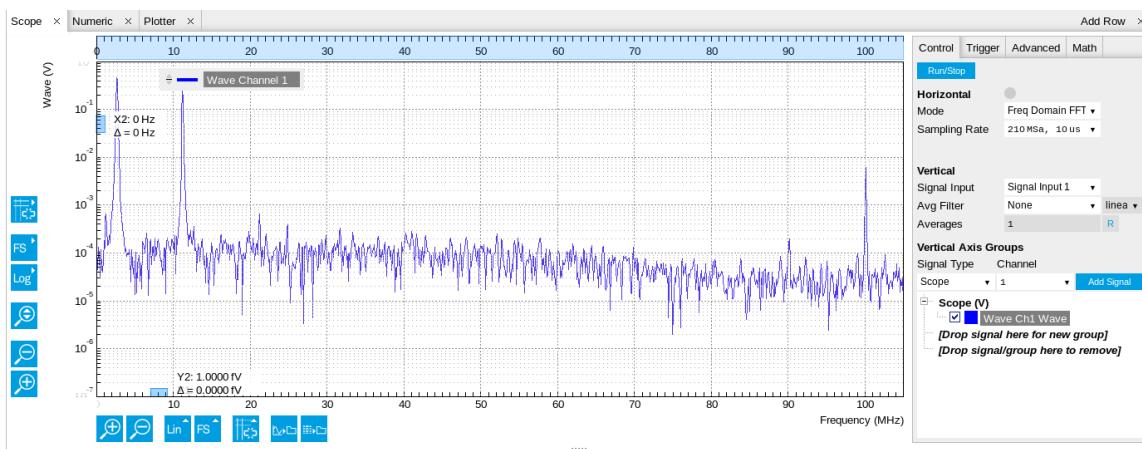


Figure 4.19. LabOne UI: Scope tab - Frequency domain

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

4.6.3. Functional Elements

Table 4.20. Scope tab: Control sub-tab

Control/Tool	Option/Range	Description
Run/Stop	Run/Stop	Runs the scope/FFT continuously.
Mode	Time Domain	Switches between time and frequency domain display.
	Freq Domain (FFT)	
Sampling Rate	6.4 kSa/s to 210 MSa/s	Defines the sampling rate of the scope.
Signal Input	Signal Input 1	Selects the source for the scope input.
	Signal Input 2	
	Signal Output 1	
	Signal Output 2	
Avg Filter		Selects averaging filter type that is applied when the average of several scope shots is computed and displayed.
	None	Averaging is turned off.
	Exponential Moving Avg	Consecutive scope shots are averaged with an exponential weight.
Averages	integer value	The number of shots required to reach 63% settling. Twice the number of shots yields 86% settling.
Reset	R	Resets the averaging filter.

For the Vertical Axis Groups, please see [Table 4.9](#) in the section called “Vertical Axis Groups”.

Table 4.21. Scope tab: Trigger sub-tab

Control/Tool	Option/Range	Description
Trigger	grey/green/yellow	When flashing, indicates that new scope shots are being captured and displayed in the plot area. The Trigger must not necessarily be enabled for this indicator to flash. A disabled trigger is equivalent to continuous acquisition. Scope shots with data loss are indicated by yellow. Such an invalid scope shot is not processed.
Signal		Selects the trigger source signal.
	Off	Switches the scope off.
	Continuous	A new waveform is acquired and displayed after the hold off time. The trigger source is ignored.
	Signal Inputs 1/2	A new waveform is acquired and displayed when the respective Signal Input is matching the trigger condition.
	Signal Outputs 1/2	A new waveform is acquired and displayed when the respective Signal Output is matching the trigger condition.
	Oscillators 1-8	A new waveform is acquired and displayed when the respective Oscillator is matching the trigger condition.
Edge	DIO 0/1	A new waveform is acquired and displayed when the respective DIO signal is matching the trigger condition.
		Selects which edge of the trigger signal performs a trigger event.
	Falling	Performs a trigger event when the source signal crosses the trigger level from high to low.
Level (%)	Rising	Performs a trigger event when the source signal crosses the trigger level from low to high.
	numeric percentage value (negative values permitted)	Defines the trigger level relative to signal full scale.

Control/Tool	Option/Range	Description
Holdoff (s)	numeric value	Defines the time before the trigger is rearmed after a recording event.

Table 4.22. Scope tab: Advanced sub-tab

Control/Tool	Option/Range	Description
FFT Window	Rectangular	Four different FFT windows to choose from. Each window function results in a different trade-off between amplitude accuracy and spectral leakage. Please check the literature to find the window function that best suits your needs.
	Hann	
	Hamming	
	Blackman Harris	
Resolution (Hz)	mHz to Hz	Spectral resolution defined by the reciprocal acquisition time (sample rate, number of samples recorded).
Spectral Density	ON / OFF	Calculate and show the spectral density. If power is enabled the power spectral density value is calculated. The spectral density is used to analyze noise.
Power	ON / OFF	Calculate and show the power value. To extract power spectral density (PSD) this button should be enabled together with Spectral Density.
Persistence	ON / OFF	Keeps previous scope shots in the display. The color scheme visualizes the number of occurrences at certain positions in time and amplitude by a multi-color scheme.
BW Limit		Select between scope sample decimation and averaging. Averaging avoids aliasing but may conceal single-sample peaks.

For the Math sub-tab please see Table 4.7 in the section called “Cursors and Math”.

4.7. Software Trigger Tab

The software trigger is one of the powerful time domain measurement tools as introduced in [Section 4.1.2](#) and is available for all HF2 Instruments.

4.7.1. Features

- Time-domain data display for all continuously streamed data
- 6 different trigger types
- Automatic trigger level determination
- Display of multiple traces
- Adjustable record history
- Mathematical toolkit for signal analysis
- Grid mode for imaging support

4.7.2. Description

The Software Trigger tab provides display and recording of shot-wise data sets after configurable trigger events. 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 4.23. App icon and short description

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

The Software Trigger tab (see [Figure 4.20](#)) 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.

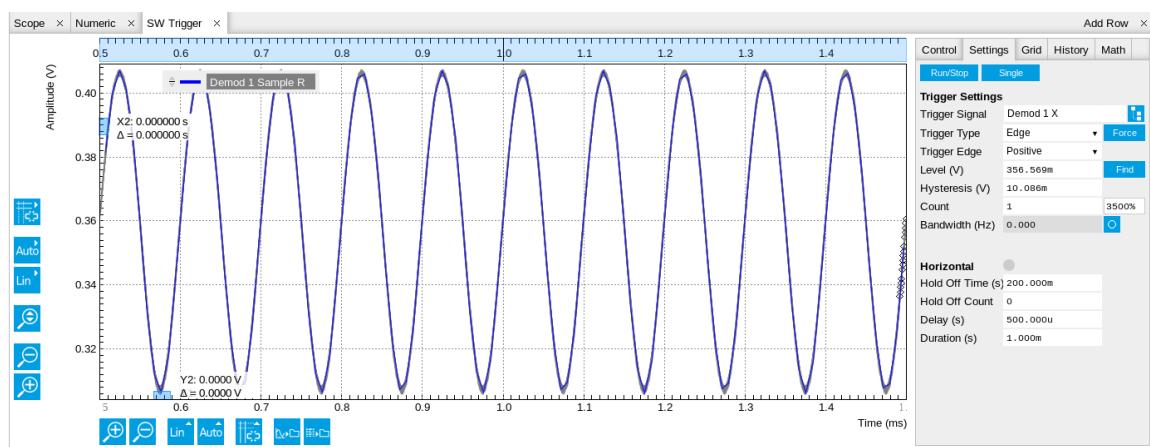


Figure 4.20. LabOne UI: Software trigger tab

The Software Trigger brings the trigger functionality of a scope to the continuously streamed data that can be viewed with the Plotter tool in a roll mode. The user can choose between a variety of

different trigger options for the different signal inputs. Each trigger event is indicated by a green LED.

The trigger condition is configured in the **Settings** sub-tab. Among the selection of Trigger Types provided here, Edge and Pulse are most suited for analog trigger sources such as demodulator data, auxiliary voltages, or oscillator frequencies. 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 feature, be sure to choose a high enough bandwidth to let pass the feature that should be triggered upon, i.e., the signal edge or pulse. Note that the Bandwidth setting does not affect the recorded data itself.

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 Trigger Types HW Trigger and Digital are intended for TTL signals on the DIO lines. Using the Bits and Bit Mask setting, complex multi-bit trigger conditions on the DIO lines can be defined.

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 the functionality for capturing data for imaging applications. With enabled grid mode, the data recorded by the Software Trigger get organized in two-dimensional frames consisting of rows and columns. Every newly captured data shot is assigned to a row, and the total number of rows is defined by the Rows settings. 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 samples according to the Columns setting. If necessary, the streamed data are interpolated, resampled, and aligned with the trigger event. The data of a completed frame after averaging appears as a single entry in the History list. These data are available for saving, but can't be displayed in the plot section. The Software Trigger needs to run continuously for a sufficiently long time to complete the acquisition, and during that time the history entry contains only partial data.

By default the plot area keeps the memory and display of the last 100 trigger shots represented in a list 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. Renaming a trace is easily possible by double clicking on a list entry. All measurements in the history can be stored in a file by clicking on **Save**. Which quantities are saved depends on which demodulator channels have been added to the Vertical Axis Groups section in the Control sub-tab. Naturally, only data from demodulators with enabled Data Transfer in the **Lock-in tab** can be included in the files. The file format, Matlab or comma-separated value (CSV) depends on the Format setting in the **Config tab**. The best way to access the saved files is via the **File Manager** tab.

4.7.3. Functional Elements

Table 4.24. SW Trigger tab: Control sub-tab

Control/Tool	Option/Range	Description
Run/Stop	Run/Stop	Start and stop the software trigger
Single	Single	Run the SW trigger once (record Count trigger events)

Control/Tool	Option/Range	Description
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 4.9](#) in the section called “Vertical Axis Groups”.

Table 4.25. SW Trigger tab: Settings sub-tab

Control/Tool	Option/Range	Description
Trigger Signal	Demodulator X/Y/R/Theta/ Frequency, Auxiliary Inputs, Trigger Inputs, Trigger Outputs, Demodulator 0 Degree Oscillator Phase Crossing	Source signal for trigger condition.
Trigger Type		Select the type of trigger to use. Selectable options depend on the selected trigger signal.
	Edge	Analog edge triggering based on high and low level. Hysteresis on the levels and low-pass filtering can be 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 conditions. The bit mask controls the bits that are used for trigger evaluation. For triggering just on DIO0 use a bit value 0x0001 and a bit mask 0x0001.
	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.

Control/Tool	Option/Range	Description
		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.
Force	Force	Forces a single trigger event.
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 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)

Control/Tool	Option/Range	Description
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	positive numeric value	Hold off time before the trigger is rearmed. A hold off time smaller than the duration will lead to overlapping trigger frames.
Hold Off Count	integer value	Number of skipped triggers until the next trigger is recorded again.
Delay	-2 s to 2 s	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)
Duration	up to 2 s	Recording length for each triggered data set.
Pulse Min	0 to 1 s	Minimum pulse width to trigger on.
Pulse Max	0 to 1 s	Maximum pulse width to trigger on.

Table 4.26. SW Trigger tab: Grid sub-tab

Control/Tool	Option/Range	Description
Mode		Enable the grid mode for two-dimensional data recording and select horizontal data resampling method.
	Off	Grid mode disabled.
	Nearest	Grid mode enabled. Resampling is performed

Control/Tool	Option/Range	Description
		using substitution by closest data point.
	Linear	Grid mode enabled. Resampling is performed using linear interpolation.
Operation		Select row update method.
	Replace	New row replaces old row.
	Average	The data for each row is averaged 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.
Rows	numeric value	Number of rows
Scan Direction		Select the scan direction and mode
	Forward	Scan direction from left to right
	Reverse	Scan direction from right to left
	Bidirectional	Alternate scanning in both directions
Repetitions	numeric value	Number of repetitions used for averaging

Table 4.27. SW Trigger tab: History sub-tab

Control/Tool	Option/Range	Description
History	History	Each entry in the list corresponds to a single trigger trace in the history. The number of triggers displayed in the plot is limited to 20. Use the toggle buttons to hide/display individual traces. Use the color picker to change the color of a trace in the plot. Double click on an entry to edit its name.
Clear All	Clear All	Remove all records from the history list.
All	All	Select all records from the history list.
None	None	Deselect all records from the history list.
Length	integer value	Maximum number of entries stored in the measurement

Control/Tool	Option/Range	Description
		history. The number of entries displayed in the list is limited to the most recent 100.
Save	Save	Save all trigger event based traces in the history to file. Which data is saved depends on the demodulator channels present in the Vertical Axis Groups of the Control sub-tab

For the Math sub-tab please see [Table 4.7](#) in the section called “Cursors and Math”.

4.8. Spectrum Analyzer Tab

The Spectrum Analyzer is one of the powerful frequency domain measurement tools as introduced in [Section 4.1.2](#) and is available in all HF2 Instruments.

4.8.1. Features

- Fast, high-resolution FFT spectrum analyzer of demodulated data ($X+iY$, R , Θ , f and $d\Theta/dt/(2\pi)$)
- Variable center frequency, frequency resolution and frequency span
- Auto bandwidth, auto span (sampling rate)
- Choice of 4 different FFT window functions
- Continuous and block-wise acquisition with different types of averaging
- Detailed noise power analysis
- Mathematical toolbox for signal analysis

4.8.2. Description

The Spectrum Analyzer allows frequency domain analysis of the demodulator data that are streamed to the data server with a user-defined rate. 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 4.28. App icon and short description

Control/Tool	Option/Range	Description
Spectrum		Provides FFT functionality to all continuously streamed measurement data.

The Spectrum tab (see [Figure 4.21](#)) 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.

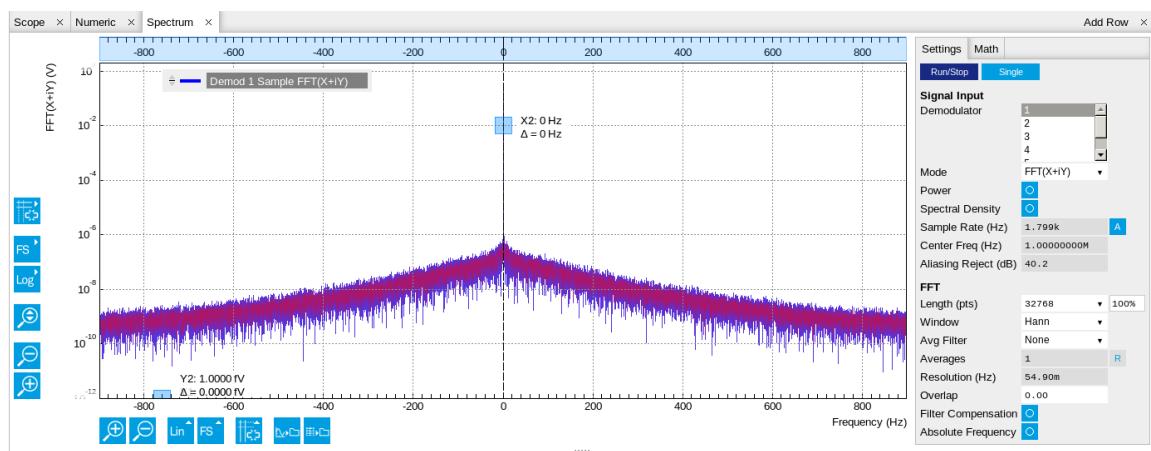


Figure 4.21. LabOne UI: Spectrum analyzer tab

The Spectrum Analyzer allows for spectral analysis of all the demodulator data by performing the fast Fourier transform (FFT) on the complex demodulator data samples $X+iY$ (with i as the imaginary unit). The result of this FFT is a spectrum centered around the demodulation frequency, whereas applying a FFT directly on the raw input data would produce a spectrum centered around

zero frequency. The latter procedure corresponds to the Frequency Domain operation in the Scope tab described in [Section 4.6](#). The main difference between the two is that the Spectrum Analyzer tool can acquire data for a much longer periods of time and therefore can achieve very high frequency resolution around the demodulation frequency. By default, the spectrum is displayed centered around zero. Sometimes however it is convenient to shift the frequency axis by the demodulation frequency which allows one to identify the frequencies on the horizontal axis with the physical frequencies at the signal inputs. This can be done by activating Absolute Frequency on the Settings sub-tab.

Another important property of the spectrum is the fact that the data samples have passed a low-pass filter with a well-defined order and bandwidth. This is most clearly noted by the shape of the noise floor. One has to take care that the selected frequency span, which equals the demodulator sampling rate, is in a healthy ratio with respect to the filter bandwidth and order. When in doubt the user can always press the button labeled A next to the sampling rate in order to obtain a default setting that suits the filter settings.

Other than displaying the frequency spectrum of the complex demodulator samples $X+iY$, the user can also choose to apply an FFT to the polar demodulator values R and Θ . This allows to carefully discriminate between phase noise components and amplitude noise components present in the signal. As these samples are real numbers the spectrum is single-sided with minimum frequency of 0 Hz.

The last option in the drop-down list $d\Theta/dt$ allows one to apply the FFT on samples of demodulator frequencies. That is particularly useful when either the PLL or the ExtRef functionalities are used. The FFT of the frequency samples then provide a quantitative view of what frequency noise components are present in the reference signal and also helps to find the optimal PLL bandwidth to track the signal.

4.8.3. Functional Elements

Table 4.29. Spectrum tab: Settings sub-tab

Control/Tool	Option/Range	Description
Run/Stop	Run/Stop	Run the FFT spectrum analysis continuously
Single	Single	Run the FFT spectrum analysis once
Demodulator	demodulator index	Select the input demodulator for FFT spectrum analysis
Mode		Select the source signal for FFT spectrum analysis. Enable spectral density to extract noise values. In order to extract peak values ensure that spectral density is disabled.
	FFT($X+iY$)	Complex FFT of the demodulator result (zoom FFT). The center frequency is defined by the oscillator frequency of the demodulator. The span is twice the demodulator sampling rate.
	FFT(R)	FFT of the demodulator amplitude result $\sqrt{x^2 +$

Control/Tool	Option/Range	Description
		y^2). The FFT is single sided as performed on real data.
	FFT(Θ)	FFT of the demodulator phase result atan2(y, x). The FFT is single sided as performed on real data.
	FFT(f)	FFT of the oscillator frequency of the selected demodulator. This mode is only interesting if the oscillator is controlled by a PID/PLL controller. The FFT is single sided as performed on real data.
	FFT($d\Theta/dt)/(2\pi)$	FFT of the demodulator phase derivative. This value is equivalent to the frequency noise observed on the demodulated signal. The FFT is single sided as performed on real data.
Power	ON / OFF	Calculate and show the power value. To extract power spectral density (PSD) this button should be enabled together with spectral density.
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.
Sample Rate (Hz)	numeric value	Equivalent to sampling rate of demodulator. The resulting frequency span is equal to the sample rate. Increase the sample rate to reduce aliasing.
Auto	A	Automatic adjustment of the sampling rate. The rate will be selected to achieve good enough anti-aliasing for the selected demodulator bandwidth.
Center Freq (Hz)	numeric value	Demodulation frequency of the selected demodulator used as input for the spectrum. For complex FFT(X+iY) the demodulation frequency defines the center frequency of the displayed FFT.
Aliasing Reject (dB)	numeric value	Resulting aliasing rejection based on demodulator sampling rate and low-pass

Control/Tool	Option/Range	Description
		filter settings. If the value is too low either increase the sampling rate or lower the filter bandwidth.
Length (pts)	2^8 to 2^23	Number of lines of the FFT spectrum. A higher value increases the frequency resolution of the spectrum.
Sampling Progress	0% to 100%	The percentage of the FFT buffer already acquired.
Window	Rectangular	Four 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	
Avg Filter	None	Selects the type of averaging.
	Exp Moving Avg	
Averages	integer value	Defines the number of spectra which are averaged and displayed.
Reset	R	Press once to reset the averaging filter.
Resolution (Hz)	mHz to Hz	Spectral resolution defined by the reciprocal acquisition time (sample rate, number of samples recorded).
Overlap	0 to 1	Overlap of demodulator data used for the FFT transform. Use 0 for no overlap and 0.99 for maximal overlap.
Filter Compensation	ON / OFF	Spectrum is corrected by demodulator filter transfer function. Allows for quantitative comparison of amplitudes of different parts of the spectrum.
Absolute Frequency	ON / OFF	Shifts x-axis labeling to show the demodulation frequency in the center as opposed to 0 Hz, when turned off.

For the Math sub-tab please see Table 4.7 in the section called “Cursors and Math”.

4.9. Sweeper Tab

The Sweeper is a highly versatile measurement tool available in all HF2 Instruments. The Sweeper allows one to scan one instrument parameter over a defined range and at the same time measure a selection of continuously streamed data. In the special case where the sweep parameter is an oscillator frequency, the Sweeper offers the functionality of a frequency response analyzer (FRA), a well-known class of instruments.

4.9.1. Features

- Full-featured parametric sweep tool for frequency, phase shift, output amplitude, DC output voltages, etc.
- Simultaneous display of data from different sources (Demodulators, frequencies, auxiliary inputs)
- Different application modes, e.g. Frequency response analyzer (Bode plots), noise amplitude sweeps, etc.
- Different sweep types: single, continuous (run / stop), bidirectional, binary
- Persistent display of previous sweep results
- Normalization of sweeps
- Auto bandwidth, averaging and display normalization
- Support for Input Scaling and Input Units
- Phase unwrap
- Full support of sinc filter

4.9.2. Description

The Sweeper supports a variety of experiments where a parameter is changed stepwise and numerous measurement data can be graphically displayed. Open the tool by clicking the corresponding icon in the UI side bar. The Sweeper tab (see [Figure 4.22](#)) 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 4.30. App icon and short description

Control/Tool	Option/Range	Description
Sweeper		Allows to scan one variable (of a wide choice, e.g. frequency) over a defined range and display various response functions including statistical operations.

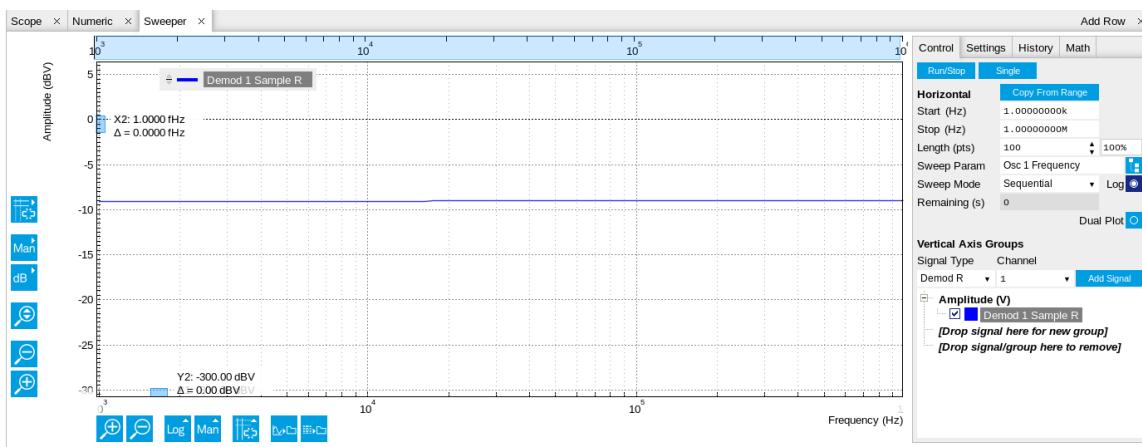


Figure 4.22. LabOne UI: Sweeper tab

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 in the form of a Bode plot. Measurement parameters can be added. As an example, AFM and MEMS users require to determine the resonance frequency and the phase delay of their oscillators. The Sweeper can also be used to sweep parameters other than frequency, for instance signal amplitudes and DC offset voltages. A sweep of the Auxiliary Output offset can for instance be used to measure current-voltage (I-V) characteristics.

For frequency sweeps the default sweep operation is logarithmic, i.e. with the Log button activated. In this mode, the sweep parameter points are distributed logarithmically - as opposed to equidistant for linear sweeps - between the start and stop values. This feature is particularly useful for sweeps over several decades, which is common for frequency sweeps. 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.

Note

The Sweeper actively modifies the main settings of the demodulators and oscillators. So in particular for situations where multiple experiments are served maybe even from different control computers great care needs to be taken so that the parameters altered by the sweeper module do not have unwanted effects elsewhere.

The Sweeper offers two operation modes differing in the level of detail of the accessible settings: the Application Mode and the Advanced Mode. Both of them are accessible in the Settings sub-tab. The Application Mode provides the choice between six measurement approaches that should help to quickly obtain correct measurement results for a large range of applications. Users who like to be in control of all the settings can access them by switching to the Advanced Mode.

In the Statistics section of the Advanced Mode one can control how data is averaged at each sweep point: either by specifying the Sample count, or by specifying the number of filter time constants (TC). The actual measurement time is determined by the larger of the two settings, taking into account the demodulator sample rate and filter settings. The Algorithm settings determines the statistics calculated from the measured data: the average for general purposes, the deviation for noise measurements, or the mean square for power measurements. The Phase Unwrap features ensures continuity of a phase measurement curve across the +180 degree boundary. Enabling

the Sinc Filter setting means that the demodulator Sinc Filter gets activated for sweep points below 50 Hz in Auto and Fixed mode. This speeds up measurements at small frequencies as explained in the [Signal Processing chapter](#).

In the Settling section one can control the waiting time between a parameter setting and the first measurement. Similarly to the Statistics setting, one has the choice between two different representations of this waiting time. The actual settling time is the maximum of the values set in units of absolute time and a time derived from the demodulator filter and a desired inaccuracy (e.g. 1 m for 0.1%). Let's consider an example. For a 4th order filter and a 3 dB bandwidth of 100 Hz we obtain a step response that attains 90% after about 4.5 ms. This can be easily measured by using the SW Trigger as indicated in [Figure 4.23](#). It is also explained in [Section 9.3](#). In case the full range is set to 1 V this means a measurement has a maximum error caused by imperfect settling of about 0.1 V. However, for most measurements the neighboring values are close compared to the full range and hence the real error caused is usually much smaller.

In the Filter section of the Advanced mode, the Bandwidth Mode setting determines how the filters of the activated demodulators are configured. In Manual mode, the current setting (in the Lock-in tab) remains unchanged by the Sweeper. In Fixed mode, the filter settings can be controlled from within the Sweeper tab. In Auto mode, the Sweeper determines the filter bandwidth for each sweep point based on a desired ω suppression. The ω suppression depends on the measurement frequency and the filter steepness. For frequency sweeps, the bandwidth will be adjusted for every sweep point within the bound set by the Max Bandwidth setting. The Auto mode is particularly useful for frequency sweeps over several decades, because the continuous adjustment of the bandwidth considerably reduces the overall measurement time.

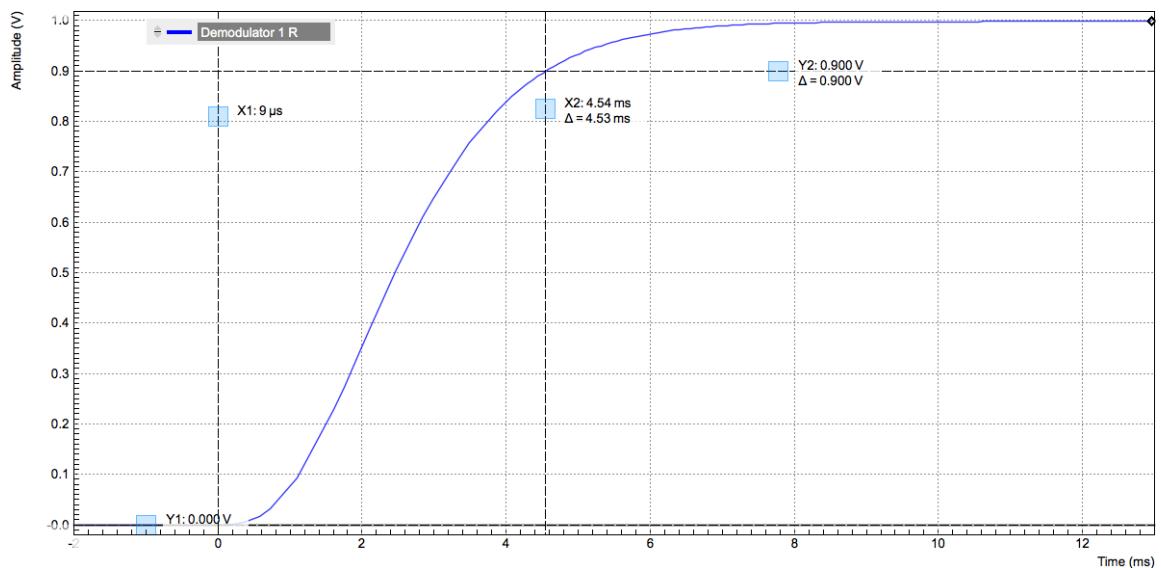


Figure 4.23. Demodulator settling time and inaccuracy

By default the plot area keeps the memory and display of the last 100 sweeps represented in a list 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. Renaming a trace is easily possible by double clicking on a list entry. All measurements in the history can be stored in a file by clicking on **Save**. Which quantities are saved depends on which demodulator channels have been added to the Vertical Axis Groups section in the Control sub-tab. Naturally, only data from demodulators with enabled Data Transfer in the [Lock-in tab](#) can be included in the files. The file format, Matlab or comma-separated value (CSV) depends on the Format setting in the [Config tab](#). The best way to access the saved files is via the [File Manager tab](#).

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.

Note

The Sweeper can get stuck whenever it does not receive any data. A common mistake is to select to display demodulator data without enabling the data transfer of the associated demodulator in the Lock-in tab.

Note

Once a sweep is performed the sweeper stores all data from the enabled demodulators and auxiliary inputs even when they are not displayed immediately in the plot area. These data can be accessed at a later point in time simply by choosing the corresponding signal display settings (Input Channel).

4.9.3. Functional Elements

Table 4.31. 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.	Oscillator Frequency Demodulator Phase Signal Output Amplitude Auxiliary Output Offset PID Setpoint Modulation Index Carrier Amplitude	Selects the parameter to be swept. Navigate through the tree view that appears and click on the required parameter. Note: the available selection depends on the configuration of the device.

Control/Tool	Option/Range	Description
	Sideband 1 Amplitude	
	Sideband 2 Amplitude	
	Signal Output Offset	
Sweep Mode		Select the scanning type, default is sequential (incremental scanning from start to stop value)
	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

For the Vertical Axis Groups, please see [Table 4.9 in the section called “Vertical Axis Groups”](#).

Table 4.32. Sweeper tab: Settings sub-tab

Control/Tool	Option/Range	Description
Filter		Application Mode: automatic mode. 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

Control/Tool	Option/Range	Description
		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 -> slow sweep	High accuracy/precision takes more measurement time.
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.

Control/Tool	Option/Range	Description
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 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.

Control/Tool	Option/Range	Description
		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 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.

Control/Tool	Option/Range	Description
	Average Power	Calculates the electric power based on a $50\ \Omega$ 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 this value and the next setting 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 this value and the previous setting 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.
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 4.33. Sweeper tab: History sub-tab

Control/Tool	Option/Range	Description
History	History	Each entry in the list corresponds to a single sweep in the history. The number of displayed sweeps is limited to 20. Use the toggle buttons to hide/display individual sweeps. Use the color picker to change the color of a sweep. Double click on an entry to edit its name.
Clear All	Clear All	Remove all records from the history list.
All	All	Select all records from the history list.

Control/Tool	Option/Range	Description
None	None	Deselect all records from the history list.
Reference	Reference	Use the selected trace as reference for all active traces.
Length	integer value	Maximum number of entries stored in the measurement history. The number of entries displayed in the list is limited to the most recent 100.
Reference On	ON / OFF	Enable/disable the reference mode.
Reference name	name	Name of the reference trace used.
Save	Save	Save all sweeps in the history to file. Which data is saved depends on the demodulator channels present in the Vertical Axis Groups of the Control sub-tab.

For the Math sub-tab please see Table 4.7 in the section called “Cursors and Math”.

4.10. Auxiliary Tab

The Auxiliary tab provides access to the settings of the Auxiliary Inputs and Auxiliary Outputs; it is available for all HF2 Instruments.

4.10.1. Features

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

4.10.2. Description

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

Table 4.34. App icon and short description

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

The Auxiliary tab (see [Figure 4.24](#)) is divided into three sections. The Aux Input section gives two graphical and two numerical monitors for the signal amplitude applied to the auxiliary inputs on the back panel. In the middle of the tab the Aux Output section allows to associate any of the measured signals to one of the 4 auxiliary outputs on the instrument front panel. With the action button next to the Offset values the effective voltage on the auxiliary outputs can be automatically set to zero. The analog output voltages can be limited to a certain range in order to avoid damaging the parts connected to the outputs.

Note

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

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

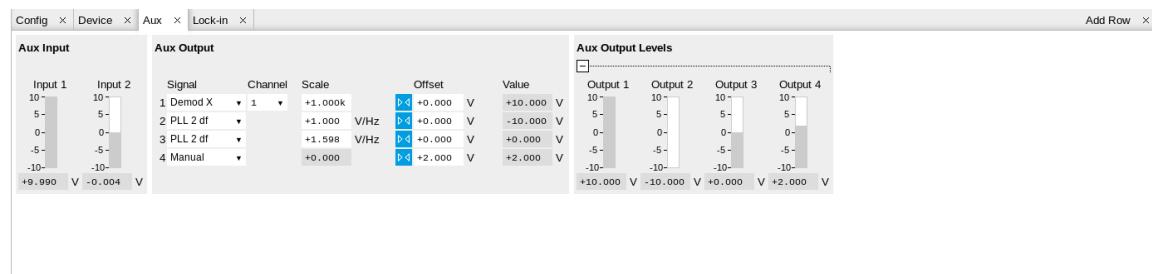


Figure 4.24. LabOne UI: Auxiliary tab

4.10.3. Functional Elements

Table 4.35. Auxiliary tab

Control/Tool	Option/Range	Description
Auxiliary Input Voltage	-10 V to 10 V	Voltage measured at the Auxiliary Input.
Signal		Select the signal source to be represented on the Auxiliary Output.
	X, Y, R, Θ	Select any of the 4 demodulator output quantities of any of the demodulators for auxiliary output.
	PLL df	Use one of the PLL delta frequencies result. HF2LI-PLL option needs to be installed.
	PID Out	Use one of the PID controllers output. HF2LI-PID option needs to be installed.
	Manual	Manually define an auxiliary output voltage using the offset field.
Channel	index	Select the channel according to the selected signal source.
Auto-zero		Automatically adjusts the Pre-offset to set the Auxiliary Output Value to zero.
Scale	numerical value	Multiplication factor to scale the signal. Auxiliary Output Value = (Signal + Preoffset)*Scale + Offset
Auto-zero		Automatically adjusts the Offset to set the Auxiliary Output Value to zero.
Offset	numerical value in Volts	Add the specified offset voltage to the signal after scaling. Auxiliary Output Value = (Signal+Preoffset)*Scale + Offset
Value	-10 V to 10 V	Voltage present on the Auxiliary Output. Auxiliary Output Value = (Signal + Preoffset)*Scale + Offset

4.11. Inputs/Outputs Tab

The In / Out tab provides access to the settings of the Instrument's main Signal Inputs and Signal Outputs. It is available for all HF2 Instruments.

4.11.1. Features

- Signal input configuration
- Signal output configuration

4.11.2. Description

The In / Out tab gives access to the same settings as do the left-most and the right-most sections of the Lock-in tab. It is mainly intended to be used on small screens that can not show the entire the Lock-in tab at once. 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 4.36. App icon and short description

Control/Tool	Option/Range	Description
In/Out		Access to all controls relevant for the main Signal Inputs and Signal Outputs on the instrument's front.

The In / Out tab contains one section for the signal inputs and one for the signal outputs. All of the corresponding connectors are placed on the instrument front panel. The In / Out tab looks differently depending on whether the HF2-MF Multi-frequency option is installed or not. [Figure 4.25](#)

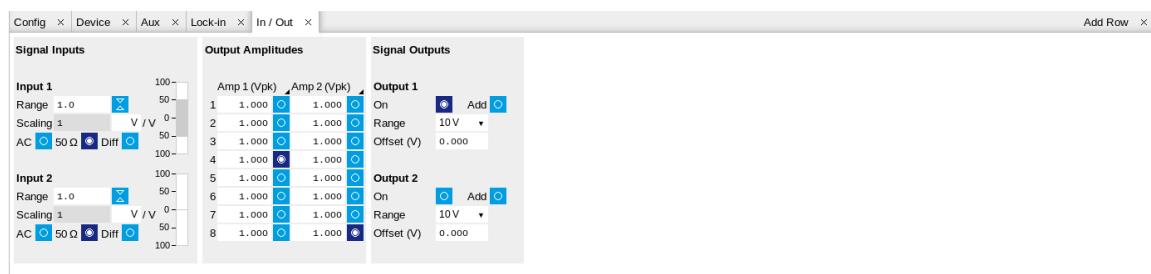


Figure 4.25. LabOne UI: Inputs/Outputs tab (with HF2-MF Multi-frequency option)



Figure 4.26. LabOne UI: Inputs/Outputs tab (without HF2-MF Multi-frequency option)

4.11.3. Functional Elements

All functional elements are equivalent to the ones on the Lock-in tab. See [Section 4.2.2](#) or [Section 4.3.2](#) for a detailed description of the functional elements.

4.12. DIO Tab

The DIO tab provides access to the settings and controls of the digital I/O and is available for all HF2 Instruments.

4.12.1. Features

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

4.12.2. Description

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

Table 4.37. 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.

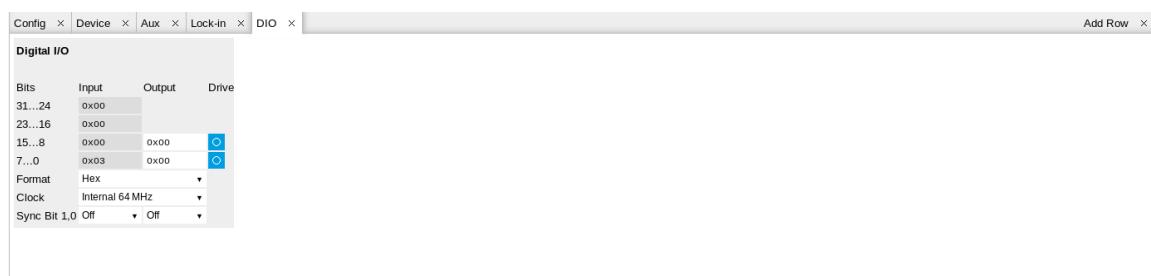


Figure 4.27. LabOne UI: DIO tab

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.

With the Sync Bit 1,0 setting it's possible to activate a TTL synchronization signal on one of the DIO BNC connectors on the instrument back panel.

4.12.3. Functional Elements

Table 4.38. Digital input and output channels, reference and trigger

Control/Tool	Option/Range	Description
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.

Control/Tool	Option/Range	Description
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.
	Internal 64 MHz	The DIO is internally clocked with a fixed frequency of 64 MHz.
	Clk Pin 68	The DIO is externally clocked with a clock signal connected to DIO Pin 68. Available frequency range 1 Hz to 64 MHz.
Sync Bit 1		Select a demodulator reference signal to be applied on DIO 1.
	Off	DIO output 1 (BNC connector) is not used for sync output and is free for other purposes.
	Demod 1 to 8	Reference signal of the selected demodulator is output on DIO 1. Note: there is a 166 ns delay between the sync on DIO 1 and the front panel outputs (sync on DIO 1 comes first) which leads to a relevant phase shift at high frequencies.
Sync Bit 0		Select a demodulator reference signal to be applied on DIO 0.
	Off	DIO output 0 (BNC connector) is not used for sync output and is free for other purposes.
	Demod 1 to 8	Reference signal of the selected demodulator is output on DIO 0. Note: there is a 166 ns delay between the sync on DIO 0 and the front panel outputs (sync on DIO 0 comes first) which leads to a

4.12. DIO Tab

Control/Tool	Option/Range	Description
		relevant phase shift at high frequencies.

4.13. Config Tab

The Config tab provides access to all major LabOne settings and is available for all HF2 Instruments.

4.13.1. 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

4.13.2. Description

The Config tab serves mainly as a control panel for all general LabOne related 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 4.39. App icon and short description

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

The Config tab (see [Figure 4.28](#)) is divided into four sections to control connections, sessions, user interface appearance and data recording.

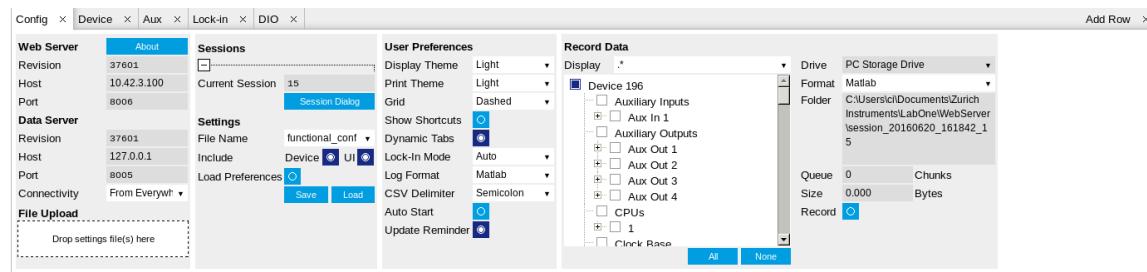


Figure 4.28. LabOne UI: Config tab

The Connection section provides information regarding 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 HF2 is used from the same computer account. For low ambient light conditions the use of the dark display theme is recommended (see [Figure 4.29](#)).

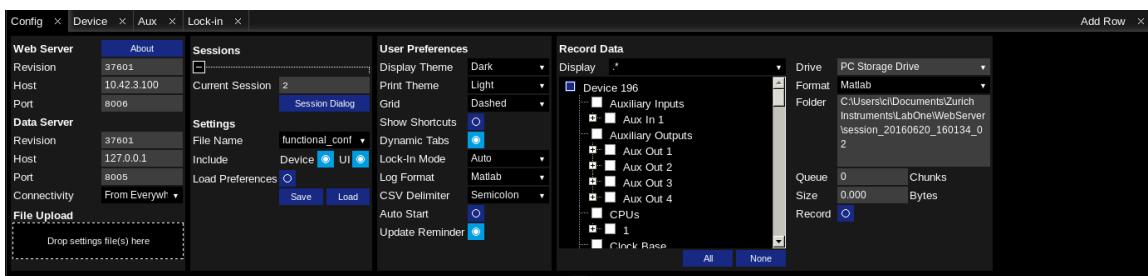


Figure 4.29. LabOne UI: Config tab - dark theme

The Record Data section contains all settings necessary to obtain hard copies of measurement data. The tree structure (see [Tree Sub-Tab](#) section) allows one to select among a large number of signals and instrument settings. Use the View Filter in order to reduce the tree structure to the most commonly used nodes such as the demodulator sample nodes.

Whenever the Record button is enabled, all selected nodes get saved continuously as Matlab or comma-separated value (CSV) files. For each selected node at least one file gets generated, but the data may be distributed over several files during long recordings. The quickest way to inspect the files after recording is to use the File Manager tab described in [Section 4.15](#). Apart from the numerical data and settings, the files contain timestamps. These integer numbers encode the measurement time in units of the instrument clock period $1/(210\text{ MHz})$. The timestamps are universal within one instrument and can e.g. be used to align the data from different files.

4.13.3. Functional Elements

Table 4.40. Config tab

Control/Tool	Option/Range	Description
About		Get information about LabOne software.
Web Server Rev	number	Web Server 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 Rev	number	Data Server 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.
Connectivity	Localhost Only From Everywhere	Forbid/Allow to connect to this Data Server from other computers.

Control/Tool	Option/Range	Description
File Upload	drop area	Drag and drop files in this box to upload files. Clicking on the box opens a file dialog for file upload. Supported files: Settings (*.xml).
Current Session	integer number	Session identifier. A session is a connection between a client and LabOne Data Server. Also indicated in status bar.
Session Dialog	Session Dialog	Open the session dialog window. 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. File location: [user]\AppData\Roaming\Zurich Instruments\LabOne\WebServer\setting
Include Device	ON / OFF	Enable save/load of device settings.
Include UI	ON / OFF	Enable save/load of user interface settings.
Load Preferences	ON / OFF	Enable loading of user preferences from settings file.
Save	Save	Save the user interface and device setting to a file.
Load	Load	Load the user interface and device setting from a file.
Display Theme	Light Dark	Choose theme of the user interface.
Print Theme	Light Dark	Choose theme for printing SVG plots
Grid	Dashed Solid None	Select active grid setting for all graphs.
Resampling Method	Linear	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.

Control/Tool	Option/Range	Description
	pchip	Piecewise Cubic Hermite Interpolating Polynomial
Show Shortcuts	ON / OFF	Displays a list of keyboard and mouse wheel shortcuts for manipulating plots.
Dynamic Tabs	ON / OFF	If enabled, sections inside the application tabs are collapsed automatically depending on the window width.
Lock-In Mode	Auto	Select the display mode for the Graphical Lock-in tab. 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	
CSV Delimiter	Comma	Select which delimiter to insert for CSV files.
	Semicolon	
	Tab	
Auto Start	ON / OFF	Skip session dialog at start-up if selected device is available. In case of an error or disconnected device the session dialog will be reactivated.
Update Reminder	ON / OFF	Display a reminder on startup if the LabOne software wasn't updated in 180 days.
Drive		Select the drive for data saving.
	PC Storage Drive	Storage of the PC on which the LabOne Web Server is running.
Format	Matlab	Data format of recorded data.
	CSV	
Folder	path indicating file location	Folder containing the saved data.
Queue	integer number	Number of data chunks not yet written to disk.
Size	integer number	Accumulated size of saved data.
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

Control/Tool	Option/Range	Description
		Length setting in the Plotter tab.
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 [Table 4.8](#) in the section called “Tree Sub-Tab”.

4.14. Device Tab

The Device tab is the main settings tab for the connected instrument and is available in all HF2 Instruments.

4.14.1. Features

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

4.14.2. 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 4.41. App icon and short description

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

The Device tab (see [Figure 4.30](#)) is divided into four sections: general instrument information, configuration, communication parameters, and a device monitor.

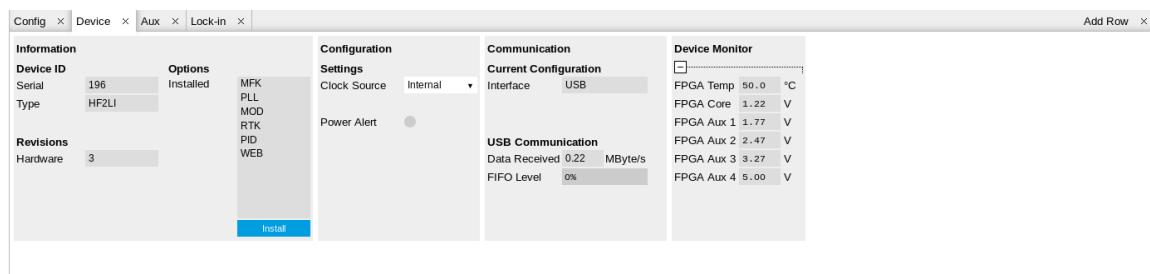


Figure 4.30. LabOne UI: Device tab

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

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

The Communication section serves to display the current data transfer rate over the USB interface.

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

4.14.3. Functional Elements

Table 4.42. Device tab

Control/Tool	Option/Range	Description
Serial	1-4 digit number	Device serial number
Type	string	Device type
Hardware	integer number	Hardware revision of the instrument
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.
Clock Source		10 MHz reference clock source.
	Internal	Internal 10 MHz clock is used as the frequency and time base reference.
	Clk 10 MHz	An external 10-MHz clock is used as the frequency and time base reference. Provide a clean and stable 10 MHz reference to the appropriate back panel connector.
Power Alert		Check 115 V/230 V settings if active. Active if 5 V supply drops below 4.8 V.
Interface		Active interface between device and data server. In case multiple options are available, the priority as indicated on the left applies.
Data received (MB/s)		USB data rate: Current USB data rate from the device to the host PC
USB FIFO Level (%)		USB FIFO level: Indicates the USB FIFO fill level inside the device. When 100%, data is lost

4.15. File Manager Tab

4.15.1. Features

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

4.15.2. Description

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

Table 4.43. 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 4.31](#)) 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.

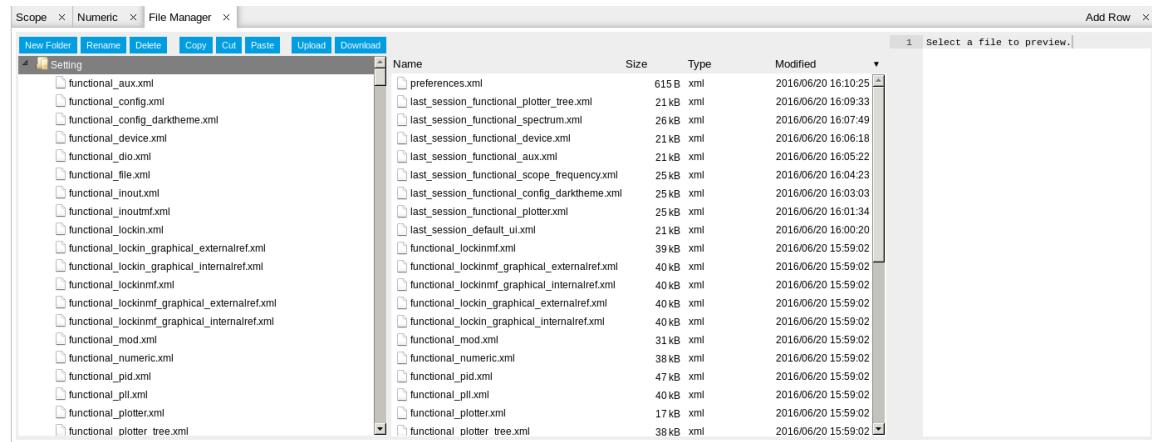


Figure 4.31. LabOne UI: File Manager tab

4.15.3. Functional Elements

Table 4.44. File tab

Control/Tool	Option/Range	Description
New Folder	New Folder	Create new folder at current location.
Rename	Rename	Rename selected file or folder.

4.15. File Manager Tab

Control/Tool	Option/Range	Description
Delete	Delete	Delete selected file(s) and/or folder(s).
Copy	Copy	Copy selected file(s) and/or folder(s) to Clipboard.
Cut	Cut	Cut selected file(s) and/or folder(s) to Clipboard.
Paste	Paste	Paste file(s) and/or folder(s) from Clipboard to the selected directory.
Upload	Upload	Upload file(s) and/or folder(s) to the selected directory.
Download	Download	Download selected file(s) and/or folder(s).

4.16. PLL Tab

The PLL tab allows convenient setup of a two independent phase-locked loop for high-speed tracking of frequency modulated signals. This tab is only available when the HF2-PLL Dual Phase-locked Loop option is installed on the HF2 Instrument (see Information section in the Device tab).

4.16.1. Features

- Two fully programmable 50 MHz phased-locked loops
- Programmable PLL center frequency and phase setpoint
- 50 kHz PLL bandwidth
- Programmable PLL phase detector filter settings and PID controller parameters
- PLL Advisor for model-based parameter suggestion and transfer function analysis
- Auto-zero functions for center frequency and setpoint
- Advanced 2-omega PLL mode (requires access to HF2-MF option)

4.16.2. Description

The PLL tab offers a convenient way to set up a phase-locked loop. In this way the frequency of an external signal can be mapped onto one of the instrument's internal oscillators. An advisor functionality based on mathematical models helps the user finding and optimizing the PID parameters and quickly optimizing the servo bandwidth for the application. 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 4.45. App icon and short description

Control/Tool	Option/Range	Description
PLL		Features all control and analysis capabilities of the phase-locked loops.

The PLL tab (see [Figure 4.32](#)) is divided into two side-tabs corresponding to the two PLL units. It contains a settings sections on the left and a modeling section with graphical display on the right.

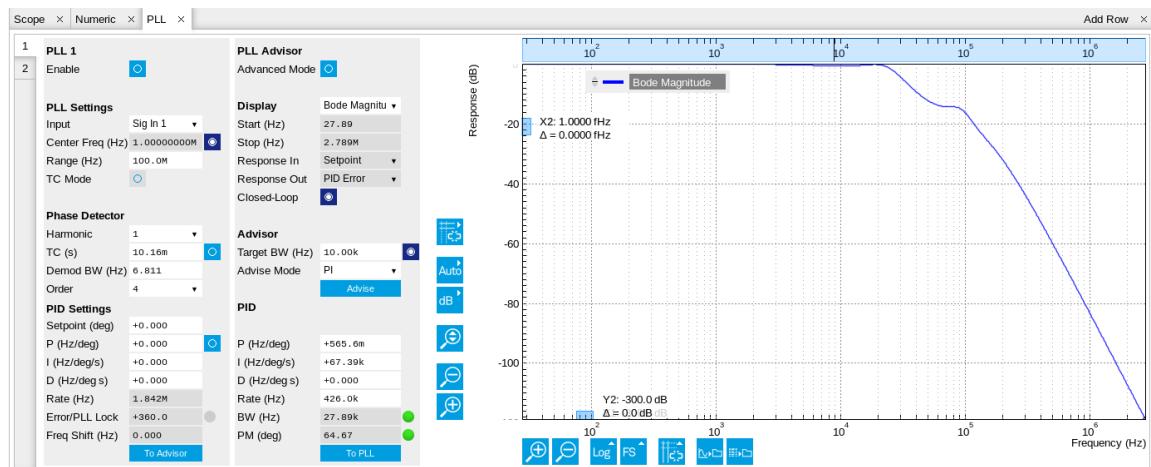


Figure 4.32. LabOne UI: PLL tab

Figure 4.33 shows a block diagram of the PLL with its components, their interconnections and the variables to be specified by the user. The demodulator and the PID controller are slightly simplified for this sketch. Their full detailed block diagrams are given in Figure 4.8, Figure 4.12, and Figure 4.35 respectively.

Phase-Locked Loop

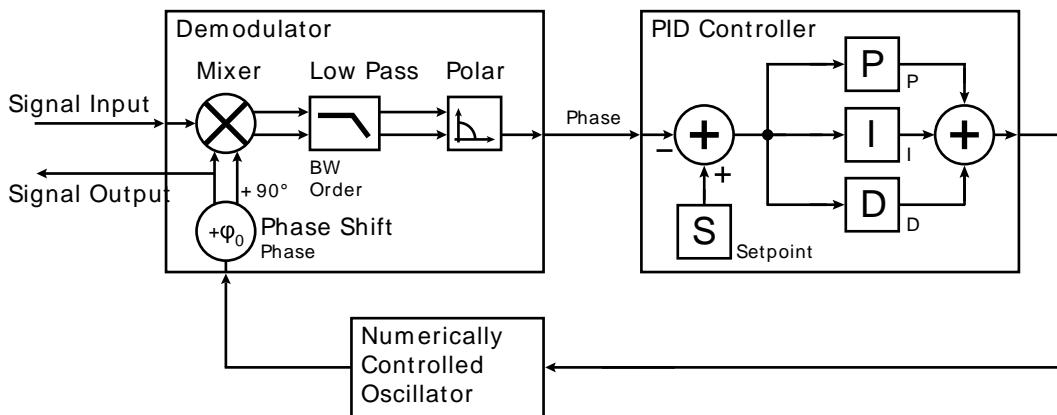


Figure 4.33. Phase-Locked Loop block diagram (components simplified)

In a typical work flow to set up a PLL one would first define the center frequency, frequency limits, frequency range, and the phase setpoint in the left section. If the frequency is not known beforehand, it can often be measured using the Sweeper or Spectrum tool. Then one would set a target bandwidth in the PLL Advisor section and subsequently click on the Advise button. The feedback parameters calculated by the Advisor will be shown in the fields just below. A graphical representation of the calculated transfer function is shown in the plot on the right-hand side. Once satisfied with the result, one can transfer the values to the instrument by clicking the To PLL button, and then enable the PLL. If the Error field now displays very small values, the phase lock has been successful.

One can now iterate the process and e.g. play with the target bandwidth in the PLL Advisor to calculate a new set of feedback parameters. Displaying the oscillator frequency in the Plotter along with a Histogram and Math function (e.g. standard deviation) can help to characterize residual phase deviations and further improve lock performance by manual tweaking.

Note

The frequency range in the PLL Settings section should exceed the target bandwidth by at least a factor of 5 to 10.

Note

PLL 1 uses demodulator 7 as phase detector, and PLL 2 uses demodulator 8. Open the Lock-in tab to check if the right Signal Input is associated with the demodulator in use.

4.16.3. Functional Elements

Table 4.46. PLL tab

Control/Tool	Option/Range	Description
Enable	ON / OFF	Enable the PLL
Input	Sig In 1/2, Aux In 1/2, DIO D0/1	Select the input signal of the PLL controller
TC Mode	ON / OFF	Switch between display of gain parameters (I , D) and time constants (T_i , T_d) for the integral and differential parts. The following relations hold: $I=P/T_i$, $D=P*T_d$.
Harmonic	1, 2	Set the harmonic used in the phase detector. A setting of 2 means the PLL generates a sub-harmonic of the external reference.
Center Freq (Hz)	0 to 50 MHz	Center frequency of the PLL oscillator. The PLL frequency shift is relative to this center frequency.
Range	numeric value	Set the frequency range of the PLL controller output relative to the center frequency
Auto Center Frequency		The PLL Center Frequency is determined automatically. In this mode, the instrument sweeps the operating range until it finds a suitable frequency. Note: Auto Center Frequency works only for open loop systems. Closed loop systems require manual mode.
TC (s)	numeric value	Filter time constant of the demodulator used as the phase detector.
Auto TC Enable	ON / OFF	When On, the PLL is running at full bandwidth. Use manual mode (off) for low-noise performance.
Demod BW (Hz)	numeric value	Filter bandwidth of the demodulator used as the phase detector.
Order	1-8	Filter order of the demodulator used as the phase detector.
Setpoint (deg)	numeric value	Phase set point in degrees (i.e. PID setpoint). Controls the phase difference between the

Control/Tool	Option/Range	Description
		input signal and the generated signal.
Automated adjustment of PID coefficients	ON / OFF	If turned on together with Auto TC Enable and Auto Center Frequency, the PLL is in ExtRef mode
P (Hz/deg)	numeric value	PID proportional gain P
I (Hz/deg/s)	numeric value	PID integral gain I
D (Hz/deg*s)	numeric value	PID derivative gain D
Rate (Hz)	numeric value	Current sampling rate of the PLL control loop. Note: The numerical precision of the controller is influenced by the loop filter sampling rate. If the target bandwidth is below 1 kHz it starts to make sense to adjust this rate to a value of about 100 to 500 times the target bandwidth. If the rate is set too high for low-bandwidth applications, integration inaccuracies can lead to nonlinear behavior.
Error (deg)	numeric value	Current phase error of the PLL (Set Point - PID Input).
PLL lock LED	grey/green	Indicates when the PLL is locked. The PLL error is sampled at 5 Sa/s and its absolute value is calculated. If the result is smaller than 5 degrees the loop is considered locked.
Freq Shift (Hz)	numeric value	Current frequency shift of the PLL (Oscillator Freq - Center Freq).
To Advisor	To Advisor	Copy the current PLL settings to the PLL Advisor.
Advanced	ON / OFF	Enables manual selection of display and advice properties. If disabled the display and advise settings are automatically with optimized default values.
Display		Select the display mode used for rendering the system frequency or time response.
	Bode Magnitude	Display the Bode magnitude plot.

Control/Tool	Option/Range	Description
	Bode Phase	Display the Bode phase plot.
	Step Resp	Display the step response plot.
Start (Hz)	numeric value	Start frequency for Bode plot display. For disabled advanced mode the start value is automatically derived from the system properties and the input field is read-only.
Stop (Hz)	numeric value	Stop frequency for Bode plot display. For disabled advanced mode the stop value is automatically derived from the system properties and the input field is read-only.
Start (s)	numeric value	Start time for step response display. For disabled advanced mode the start value is zero and the field is read-only.
Stop (s)	numeric value	Stop time for step response display. For disabled advanced mode the stop value is automatically derived from the system properties and the input field is read-only.
Response In		Start point for the plant response simulation for open or closed loops. In closed loop configuration all elements from output to input will be included as feedback elements.
	Demod Input	Start point is at the demodulator input.
	Setpoint	Start point is at the setpoint in front of the PID.
	PID Output	Start point is at PID output.
	Instrument Output	Start point is at the instrument output.
	DUT Output	Start point is at the DUT output and instrument input.
Response Out		End point for the plant response simulation for open or closed loops. In closed loop configuration all elements from output to input will be included as feedback elements.
	PID Output	End point is at PID output.
	Instrument Output	End point is at the instrument output.

Control/Tool	Option/Range	Description
	DUT Output	End point is at the DUT output and instrument input.
	Demod Input	End point is at the demodulator input.
	PID Error	End point is at the PID error calculation of the PID.
Closed-Loop	ON / OFF	Switches the display of the system response between closed or open loop.
Target BW (Hz)	numeric value	Target bandwidth for the PLL closed loop feedback system which is used for the advising of the PID parameters. This bandwidth defines the trade-off between PLL speed and phase noise.
Auto Bandwidth	ON / OFF	<p>Adjusts the demodulator bandwidth to fit best to the specified target bandwidth of the full system. If disabled, a demodulator bandwidth too close to the target bandwidth may cause overshoot and instability.</p> <p>In special cases the demodulator bandwidth can also be selected smaller than the target bandwidth.</p>
Advise Mode		<p>Select the PID coefficients that are optimized. The other PID coefficients remain unchanged but are used during optimization. This allows to force coefficients to a value while optimizing the rest.</p> <p>The advise time will increase significantly with the number of parameters optimized.</p>
	P	Only optimize the proportional gain.
	I	Only optimize the integral gain.
	PI	Only optimize the proportional and the integral gain.
	PID	Optimize the proportional, integral, and derivative gains.
Advise	Advise	Calculate PID coefficients based on application mode and given settings.

Control/Tool	Option/Range	Description
		Only PID coefficients specified with the advise mode are optimized. The Advise mode can be used incremental, means current coefficients are used as starting point for the optimization unless other model parameters are changed in-between.
P (Hz/deg)	numeric value	Proportional gain P coefficient used for calculation of the response of the PID model. The parameter can be optimized with PID advise or changed manually. The parameter only gets active on the PID after pressing the button To PLL.
I (Hz/deg/s)	numeric value	Integral gain I coefficient used for calculation of the response of the PID model. The parameter can be optimized with PID advise or changed manually. The parameter only gets active on the PID after pressing the button To PLL.
D (Hz/deg*s)	numeric value	Derivative gain D coefficient used for calculation of the response of the PID model. The parameter can be optimized with PID advise or changed manually. The parameter only gets active on the PID after pressing the button To PLL.
Rate (Hz)	RT load dependent	PID sampling rate used for simulation. The advisor will update the rate to match with the specified target bandwidth. A sampling rate close to the target bandwidth and excessive higher bandwidth will result in a simulation mismatch.
BW (Hz)	numeric value	Simulated bandwidth of the full close loop PLL with the current PID settings. This value should be larger than the target bandwidth.
Target BW LED	green/red	Green indicates that the target bandwidth can be achieved. For very high PLL bandwidth the target bandwidth might be

Control/Tool	Option/Range	Description
		only achieved using marginal stable PID settings.
PM (deg)	numeric value	Simulated phase margin of the PID with the current settings. The phase margin should be greater than 45 deg for stable conditions. An Infinite value is shown if no unity gain crossing is available to determine a phase margin.
Stable LED	green/red	Green indicates that the phase margin is fulfilled and the PID system should be stable.
To PLL	To PLL	Copy the PLL Advisor settings to the PLL.

4.17. PID Tab

The PID tab is only available if the HF2-PID Quad PID Controller option is installed on the HF2 Instrument (the installed options are displayed in the Device tab).

Note

Some settings in the PID tab are interdependent with settings that are controlled from other tabs. If the PID output controls a certain variable, e.g. Signal Output Offset, this variable will be shown as read-only where it appears in other tabs (i.e. in the Lock-in tab for this case).

4.17.1. Features

- Four independent proportional, integral, derivative (PID) controllers
- PID Advisor with multiple DUT models, transfer function, and step function modeling
- More than 5 kHz regulation bandwidth
- Input parameters: demodulator data, auxiliary inputs, oscillator frequency
- Output parameters: output amplitudes, oscillator frequencies, auxiliary outputs and DIO
- Bandwidth limit for the derivative (D) feedback component

4.17.2. Description

The PID tab is the main control center of general feedback loop related 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 4.47. App icon and short description

Control/Tool	Option/Range	Description
PID		Features all control and analysis capabilities of the PID controllers.

The PID tab (see [Figure 4.34](#)) is divided into four identical sub-tabs, each of them providing access to the settings functionality for one of the four PID controllers and the associated PID Advisor.

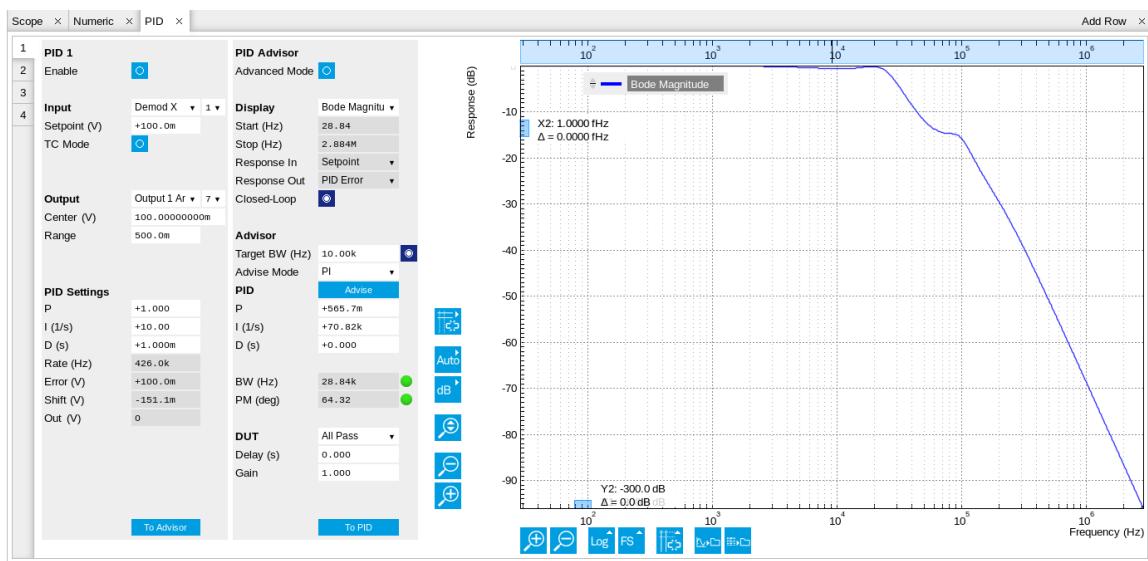


Figure 4.34. LabOne UI: PID tab

With their variety of different input and output connections, the LabOne PID controllers are extremely versatile and can be used over a wide range of different applications. With low internal delays the speed is even high enough to cater to demanding laser locking applications. Figure 4.35 shows a block diagram of all PID controller components, their interconnections and the variables to be specified by the user.

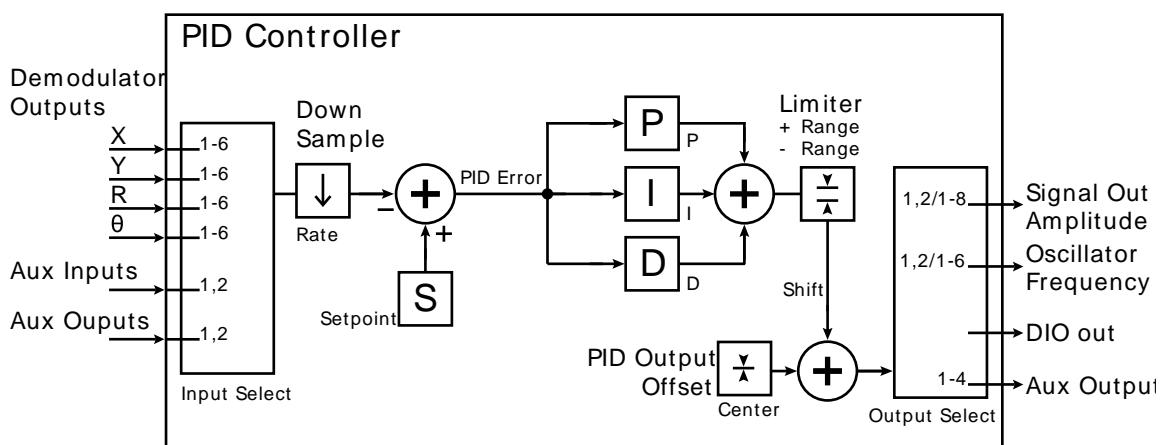


Figure 4.35. PID controller block diagram

Setting up a control loop

Depending on the application there are a number of ways to set up a control loop. Let's consider a few different approaches and see how the Advisor can help to reduce the effort and improve on the result and understanding of the setup.

Manual approach

In cases where the transfer function of the device under test (DUT) is entirely unknown and only little noise couples into the system from the environment, a manual approach is often the quickest way to get going. For manual configuration of a new control loop it is generally recommended to

start with a small value for P and set the other parameters (I, D) to zero. By enabling the controller one will then immediately see if the sign of P is correct and if the feedback is acting on the correct output parameter for instance by checking the numbers (Error, Shift, Out) displayed in the PID tab. A slow increase of I will then help to zero the PID error signal completely. Care has to be taken when enabling the D part as this often introduces an unstable servo behavior. At this stage a Plotter tab opened in parallel and displaying the PID error over time can be a great help. The math tools offered by the Plotter allow us to display the standard deviation and the average value of the error. These values should be minimized by tweaking the PID parameters and the associated histogram should have a symmetric (ideally Gaussian) envelope. After a few iterations one can then check the performance by introducing a step response by changing the PID setpoint. The SW Trigger is the ideal tool to record the step function of the control loop by setting the trigger condition half way of the step and the Delay and Duration according to the expected bandwidth. One should also make sure that the data rate set for the transfer of the PID data is high enough to fully resolve the behavior in the time domain.

Using DUT model functions with the PID Advisor

For many experimental situations the external device or DUT that needs to be controlled can be well approximated by a simple model. LabOne offers a number of different choices of DUT model functions. Apart from model-specific parameters, all of them have a setting for the delay that occurs outside the instrument. Depending on the targeted servo bandwidth, the external delay can often be the limiting factor and should be sensibly chosen.

Note

The delay specified for each model resembles the earliest possible response for a step change of the instrument output to be seen on the instrument input. It describes the causality of the system and does not affect the shape of the DUT transfer function. Standard coaxial cables cause a signal delay of about 5 ns/m.

The most simple approach to modeling is to assume a DUT with a unity transfer function by using All Pass. The low-pass filters allow for limiting the bandwidth, to set an overall gain and a damping for the second order filter. Resonator Frequency is a model that applies well in situations with a passive external component, e.g. a AFM cantilever or a quartz resonator, whose frequency should be tracked by a PLL over time. In cases where the amplitude of the resonator signal needs to be stabilized with a second control loop (automatic gain control), the Resonator Amplitude model is the right choice. Setting the resonance frequency and the Q factor, both can be obtained before by a frequency scan over the resonance using the sweeper module, allows the Advisor to estimate the gain and low-pass behavior of the resonator. Internal PLL is used whenever an external oscillating signal is provided that shall be followed by one of the internal oscillators. The VCO setting describes a situation where the input variable of the DUT is a voltage and the output is a frequency. The gain is then the conversion factor of how much voltage change on the input causes how much frequency shift on the VCO output. In case the frequency of the VCO can be tracked by using the external reference mode, one can easily obtain this gain with the sweeper by scanning the Auxiliary Output voltage and displaying the resulting oscillator frequency. The gain is given by the slope of the resulting line at the frequency of interest.

With a suitable model chosen and the proper parameter set to best describe the actual measurement situation, one can now continue by defining a target bandwidth for the entire control loop and the Advise Mode, i.e. the parameters that shall be used for the control operation. Whenever the input signal is derived from one of the demodulators it is convenient to activate the box next to target bandwidth. With that in place the Advise algorithm will automatically adjust the demodulator bandwidth to a value about 5 times higher than the target bandwidth in order to avoid to be limited by demodulation speed. With all the model information and the Target Bandwidth the Advise algorithm will now calculate a target step response function that it will try to achieve by adjusting the parameters in the next step. Before doing so in case of a newly set up DUT model the

algorithm will first try to estimate the PID parameters by using the Ziegler-Nichols method. When there has been a previous run the user can also change the parameters in the model manually which will be used as new start parameters of the next Advise run. Starting from the initial parameters, the Advisor will then perform a numerical optimization in order to achieve a least-squares fit of the calculated step response to a target step response determined from the Target Bandwidth. The result is numerically characterized by an achieved bandwidth (BW) and a phase margin (PM). Moreover, the large plot area on the right can be used to characterize the result by displaying transfer functions, magnitude and phase, and step responses between different signal nodes inside the loop. Once the modeling is finished one can copy the resulting parameters to the physical PID by clicking on "To PID".

Table 4.48. DUT transfer functions

Name	Function	Parameters
All pass	$H(s) = g$	1. Gain g
Low-pass 1st	$H(s) = g \frac{1}{t_c s + 1} = g \frac{\omega_n}{s + \omega_n}$	1. Gain g 2. Filter bandwidth (BW) $f_{-3\text{dB}} = \omega_n / 2\pi$
Low-pass 2nd	$H(s) = g \frac{\omega_n^2}{s^2 + 2\omega_n \zeta s + \omega_n^2}$	1. Gain g 2. Resonance frequency $f_{\text{res}} = \omega_n / 2\pi$ 3. Damping ratio ζ with $f_{-3\text{dB}} = 2\zeta f_{\text{res}}$
Resonator frequency	$H(s) = \frac{1}{t_c s + 1}$ with $t_c = \frac{1}{2\pi f_{-3\text{dB}}} = \frac{2Q}{2\pi f_{\text{res}}}$	1. Resonance frequency f_{res} 2. Quality factor Q
Resonator amplitude	$H(s) = g \frac{\omega_{\text{res}} / (2Q)}{s + \omega_{\text{res}} / (2Q)}$ with $\omega_{\text{res}} = 2\pi f_{\text{res}}$	1. Gain g 2. Resonance frequency f_{res} 3. Quality factor Q
Internal PLL	$H(s) = -\frac{360}{s}$	1. none
VCO	$H(s) = g \frac{360}{s(t_c s + 1)}$ with $t_c = \frac{1}{2\pi f_{-3\text{dB}}}$	1. Gain g (Hz/V) 2. Bandwidth (BW) $f_{-3\text{dB}}$

Note

It is generally recommended to use the Advise feature in a stepwise approach where one increases the free parameter from P to PI, to PID. This can save time because it prevents optimizing into local minima. Also it can be quite illustrative which of the feedback parameters leads to which effect in the feedback behavior.

Note

The low-pass filter in the differential part is implemented as an exponential moving average filter described by $y_t = (1-\alpha)y_{t-1} + \alpha x_t$ with $\alpha = 2^{-\text{dshift}}$. The default value for dshift is 0, i.e. no averaging or unity filter transfer function. On the UI the filter properties can be changed in units of bandwidth or a time constant.

In case the feedback output is a voltage applied to sensitive external equipment it is highly recommended to make use of the center value and the upper and lower limit values. This will guarantee that the output stays in the defined range even when the lock fails and the integrator goes into saturation.

4.17.3. Functional Elements

Table 4.49. PID tab

Control/Tool	Option/Range	Description
Enable	ON / OFF	Enable the PID controller
Input	Demodulator X	Select input source of PID controller
	Demodulator Y	
	Demodulator R	
	Demodulator Theta	
	Aux Input	
	Aux Output	
	Modulation Index	
	Dual Frequency Tracking $ Z(i+1) - Z(i) $	
	Demod $X(i+1) - X(i)$	
	Demod $ Z(i+1) - Z(i) $	
Input Channel	Oscillator Frequency	
	index	Select input channel of PID controller.
Setpoint	numeric value	PID controller setpoint
TC Mode	ON / OFF	Enables time constant representation of PID parameters.
Output		Select output of the PID controller
	Output 1 Amplitude	Feedback to the main signal output amplitudes
	Output 2 Amplitude	Feedback to the main signal output amplitudes
	Oscillator Frequency	Feedback to any of the internal oscillator frequencies

Control/Tool	Option/Range	Description
	Aux Output Offset	Feedback to any of the 4 Auxiliary Output's Offset
	DIO (int16)	Feedback to the DIO as a 16 bit word
Output Channel	index	Select output channel of PID controller.
Center, Range	numeric value	After adding the Center value to the PID output, the signal is clamped to Center + Range and Center - Range.
Range	numeric value	Set the range of the PID controller output relative to the center
P (Hz/deg)	numeric value	PID proportional gain P
I (Hz/deg/s)	numeric value	PID integral gain I
D (Hz/deg*s)	numeric value	PID derivative gain D
Rate	RT load dependent	PID sampling rate and update rate of PID outputs. Needs to be set substantially higher than the targeted loop filter bandwidth. Note: The numerical precision of the controller is influenced by the loop filter sampling rate. If the target bandwidth is below 1 kHz it starts to make sense to adjust this rate to a value of about 100 to 500 times the target bandwidth. If the rate is set to high for low bandwidth applications, integration inaccuracies can lead to non linear behavior.
Error	numeric value	Error = Set point - PID Input
Shift	numeric value	Difference between the current output value Out and the Center. Shift = P*Error + I*Int(Error, dt) + D*dError/dt
Out	numeric value	Current output value
To Advisor	To Advisor	Copy the current PID settings to the PID Advisor.
Advanced	ON / OFF	Enables manual selection of display and advice properties. If disabled the display and advise settings are automatically with optimized default values.

Control/Tool	Option/Range	Description
Display		Select the display mode used for rendering the system frequency or time response.
	Bode Magnitude	Display the Bode magnitude plot.
	Bode Phase	Display the Bode phase plot.
	Step Resp	Display the step response plot.
Start (Hz)	numeric value	Start frequency for Bode plot display. For disabled advanced mode the start value is automatically derived from the system properties and the input field is read-only.
Stop (Hz)	numeric value	Stop frequency for Bode plot display. For disabled advanced mode the stop value is automatically derived from the system properties and the input field is read-only.
Start (s)	numeric value	Start time for step response display. For disabled advanced mode the start value is zero and the field is read-only.
Stop (s)	numeric value	Stop time for step response display. For disabled advanced mode the stop value is automatically derived from the system properties and the input field is read-only.
Response In		Start point for the plant response simulation for open or closed loops. In closed loop configuration all elements from output to input will be included as feedback elements.
	Demod Input	Start point is at the demodulator input.
	Setpoint	Start point is at the setpoint in front of the PID.
	PID Output	Start point is at PID output.
	Instrument Output	Start point is at the instrument output.
	DUT Output	Start point is at the DUT output and instrument input.
Response Out		End point for the plant response simulation for open or closed loops. In closed loop configuration all elements from output to input will

Control/Tool	Option/Range	Description
		be included as feedback elements.
	PID Output	End point is at PID output.
	Instrument Output	End point is at the instrument output.
	DUT Output	End point is at the DUT output and instrument input.
	Demod Input	End point is at the demodulator input.
	PID Error	End point is at the PID error calculation of the PID.
Closed-Loop	ON / OFF	Switch the display of the system response between closed or open loop.
Target BW (Hz)	numeric value	Target bandwidth for the closed loop feedback system which is used for the advising of the PID parameters. This bandwidth defines the trade-off between PID speed and noise.
Auto Bandwidth	ON / OFF	Adjusts the demodulator bandwidth to fit best to the specified target bandwidth of the full system. If disabled, a demodulator bandwidth too close to the target bandwidth may cause overshoot and instability. In special cases the demodulator bandwidth can also be selected smaller than the target bandwidth.
Advise Mode		Select the PID coefficients that are optimized. The other PID coefficients remain unchanged but are used during optimization. This allows to force coefficients to a value while optimizing the rest. The advise time will increase significantly with the number of parameters optimized.
	P	Only optimize the proportional gain.
	I	Only optimize the integral gain.
	PI	Only optimize the proportional and the integral gain.

Control/Tool	Option/Range	Description
	PID	Optimize the proportional, integral, and derivative gains.
Advise	Advise	<p>Calculate the PID coefficients based on the used DUT model and the given target bandwidth. If optimized values can be found the coefficients are updated and the response curve is updated on the plot.</p> <p>Only PID coefficients specified with the advise mode are optimized. The Advise mode can be used incremental, means current coefficients are used as starting point for the optimization unless other model parameters are changed in-between.</p>
P (Hz/deg)	numeric value	Proportional gain P coefficient used for calculation of the response of the PID model. The parameter can be optimized with PID advise or changed manually. The parameter only gets active on the PID after pressing the button To PLL.
I (Hz/deg/s)	numeric value	Integral gain I coefficient used for calculation of the response of the PID model. The parameter can be optimized with PID advise or changed manually. The parameter only gets active on the PID after pressing the button To PLL.
D (Hz/deg*s)	numeric value	Derivative gain D coefficient used for calculation of the response of the PID model. The parameter can be optimized with PID advise or changed manually. The parameter only gets active on the PID after pressing the button To PLL.
BW (Hz)	numeric value	Simulated bandwidth of the full close loop model with the current PID settings. This value should be larger than the target bandwidth.
Target BW LED	green/red	Green indicates that the target bandwidth can be achieved. For very high PID bandwidth the target bandwidth might be only achieved using

Control/Tool	Option/Range	Description
		marginal stable PID settings. In this case, try to lower the bandwidth or optimize the loop delays of the PID system.
PM (deg)	numeric value	Simulated phase margin of the PID with the current settings. The phase margin should be greater than 45 deg for internal PLL and 60 deg for all other DUT for stable conditions. An Infinite value is shown if no unity gain crossing is available to determine a phase margin.
Stable LED	green/red	Green indicates that the phase margin is fulfilled and the PID system should be stable.
To PID	To PID	Copy the PID Advisor settings to the PID.
DUT Model		<p>Type of model used for the external device to be controlled by the PID.</p> <p>A detailed description of the transfer function for each model is found in the previous section.</p>
	All Pass	The external device is modeled by an all pass filter. Parameters to be configured are delay and gain.
	LP 1st	The external device is modeled by a first-order low-pass filter. Parameters to be configured are delay, gain and filter bandwidth.
	LP 2nd	The external device is modeled by a second-order low-pass filter. Parameters to be configured are delay, gain, resonance frequency and damping ratio.
	Resonator Frequency	The external device is modeled by a resonator. Parameters to be configured are delay, center frequency and quality factor.
	Internal PLL	The DUT is the internal oscillator locked to an external signal through a phase-locked loop. The parameter to be configured is the delay.

Control/Tool	Option/Range	Description
	VCO	The external device is modeled by a voltage controlled oscillator. Parameters to be configured are delay, gain and bandwidth.
	Resonator Amplitude	The external device is modeled by a resonator. Parameters to be configured are delay, gain, center frequency and quality factor.
Delay	numeric value	Parameter that determines the earliest response for a step change. This parameter does not affect the shape of the DUT transfer function.
Gain	numeric value	Parameter that determines the gain of the DUT transfer function.
BW (Hz)	numeric value	Parameter that determines the bandwidth of the first-order low-pass filter respectively the bandwidth of the VCO.
Damping Ratio	numeric value	Parameter that determines the damping ratio of the second-order low-pass filter.
Center Frequency	numeric value	Parameter that determines the resonance frequency of the modeled resonator.
Q	numeric value	Parameter that determines the quality factor of the modeled resonator.

4.18. MOD Tab

The MOD tab provides access to the settings of the amplitude and frequency modulation units. This tab is only available when the HF2-MOD AM/FM Modulation option is installed on the Instrument (see Information section in the Device tab).

Note

The HF2-MOD AM/FM Modulation option requires the HF2-MF Multi-frequency option.

4.18.1. Features

- Phase coherently add and subtract oscillator frequencies and their multiples
- Control for AM and FM demodulation
- Control for AM and narrow-band FM generation
- Direct analysis of higher order carrier frequencies and sidebands

4.18.2. Description

The MOD tab offers control in order to phase coherently add and subtract the frequencies of multiple numerical oscillators. 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 4.50. App icon and short description

Control/Tool	Option/Range	Description
MOD		Control panel to enable (de)modulation at linear combinations of oscillator frequencies.

The MOD tab (see [Figure 4.36](#)) is divided into two horizontal sections, one for each modulation unit.

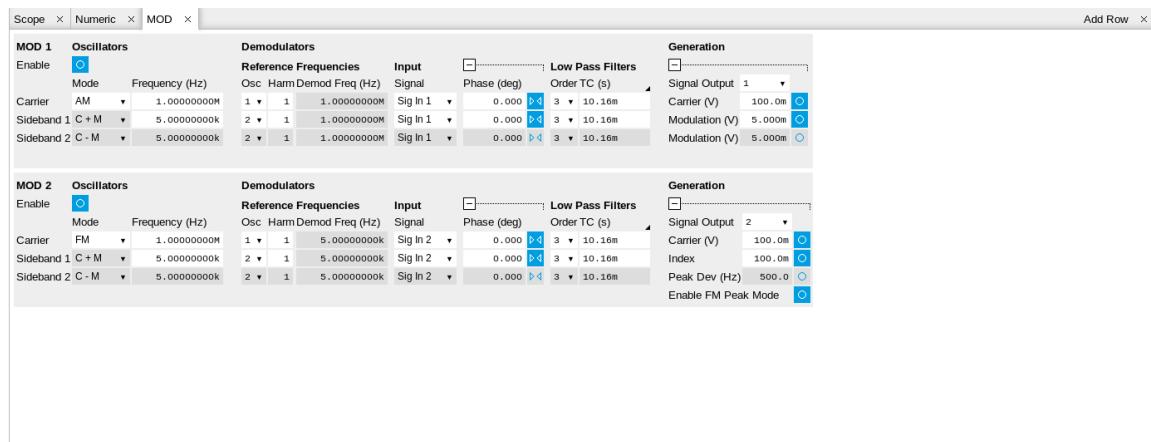


Figure 4.36. LabOne UI: MOD tab

The modulation units are designed for experiments involving multiple frequencies. For many of such experiments the associated spectrum reveals a dominant center frequency, often called the

carrier, and one or multiple sidebands symmetrically placed around the carrier. Typical examples are amplitude modulated (AM) signals with one carrier and two sidebands separated from the carrier by the AM modulation frequency. Another example is frequency modulation (FM) where multiple sidebands to the left and right of the carrier can appear. The relative amplitude of the sideband for both AM and FM depends on the modulation depth, which is often expressed by the modulation index.

The classical approach of analyzing such signals (in particular when only analog instruments are available) is to use a configuration called tandem demodulation. This is essentially the serial cascading of lock-in amplifiers. The first device is referenced to the carrier frequency and outputs the in-phase component. This is then fed into the subsequent lock-in amplifiers in order to extract the different sideband components. There are several downsides to this scheme:

- The quadrature component of the first lock-in tuned to the carrier has to be continuously zeroed out by adjusting the reference phase. Otherwise a serious part of the signal power is lost for the analysis which usually leads to a drop in SNR.
- The scheme scales badly in terms of the hardware resources needed, in particular if multiple sideband frequencies need to be extracted.
- Every time a signal enters or exits an instrument the SNR gets smaller (e.g. due to the instrument inputs noise). Multiple such steps can deteriorate signal quality significantly.

All these shortcomings are nicely overcome by providing the ability to generate linear combinations of oscillator frequencies and use these combinations as demodulation references.

The MOD tab contains two sections MOD 1 and MOD 2. Both are identical in all aspects except that MOD 1 is linked to demodulators 1,2 and 3, whereas MOD 2 is linked to demodulators 4, 5, and 6. Each of the MOD units can make use of up to 3 oscillators, which can be even referenced to an external source by using ExtRef or a PLL. Figure 4.37 gives an overview of the different components involved and their interconnections.

MOD Option

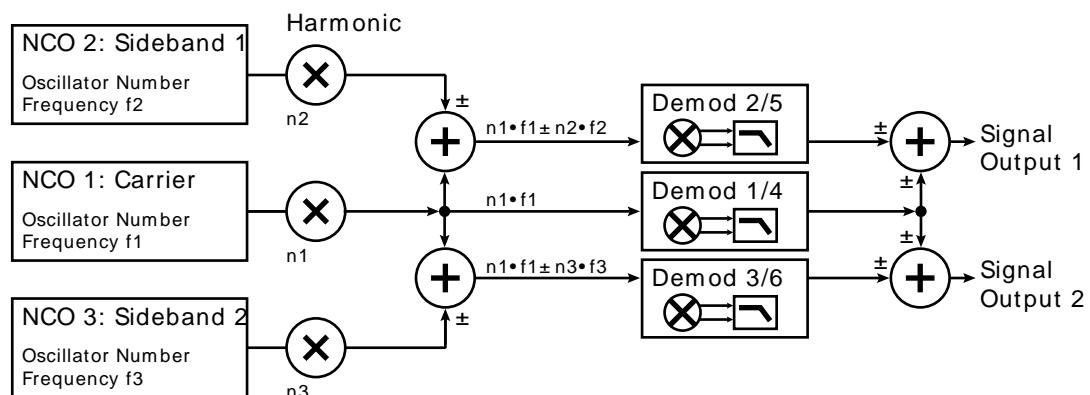


Figure 4.37. Modulation Option block diagram

For convenience the UI provides access to presets for AM and FM in the Mode column. In the Manual Mode all settings can be chosen freely.

Note

Whenever MOD 1 or 2 is enabled, all the settings in the Lock-in tab that are controlled by the MOD Option will be set to read-only.

Note

When using the Sweeper to vary a parameter of the MOD tab, it is recommended to manually set the minimum settling time to 500 ms (Settings sub-tab, Advanced Mode). Otherwise the measurement may yield invalid data.

On top of signal analysis the MOD option can also be utilized for signal generation. The Generation section provides all the necessary controls to adjust the carrier and sideband amplitudes.

Note

FM signals are generated by coherent superposition of the carrier signal with two sideband frequencies on either side that have the same amplitudes but opposite phases. The phase shift is achieved by using negative amplitudes as displayed in the Lock-in tab. This FM generation method approximates true FM as long as the modulation index is well below 1, i.e. higher-order sidebands can be neglected. For a modulation index of 1 true FM provides more than 13% of signal power in the second and higher order sidebands.

More details regarding AM and FM signal analysis and generation can be found on the Zurich Instruments web page, e.g. <http://www.zhinst.com/blogs/sadik/2014/02/sideband-analysis/>.

4.18.3. Functional Elements

Table 4.51. MOD tab

Control/Tool	Option/Range	Description
Enable	ON / OFF	Enable the modulation
Mode	AM/FM/manual	Select the modulation mode.
Mode		Enabling of the first sideband and selection of the position of the sideband relative to the carrier frequency for manual mode.
	Off	First sideband is disabled. The sideband demodulator behaves like a normal demodulator.
	C + M	First sideband to the right of the carrier
	C - M	First sideband to the left of the carrier
Mode		Enabling of the second sideband and selection of

Control/Tool	Option/Range	Description
		the position of the sideband relative to the carrier frequency for manual mode.
	Off	Second sideband is disabled. The sideband demodulator behaves like a normal demodulator.
	C + M	Second sideband to the right of the carrier
	C - M	Second sideband to the left of the carrier
Frequency (Hz)	0 to 50 MHz	Sets the frequency of the carrier.
Frequency (Hz)	0 to 50 MHz	Frequency offset to the carrier from the first sideband.
Frequency (Hz)	0 to 50 MHz	Frequency offset to the carrier from the second sideband.
Carrier	oscillator index	Select the oscillator for the carrier signal.
Sideband 1	oscillator index	Select the oscillator for the first sideband.
Sideband 2	oscillator index	Select the oscillator for the second sideband.
Harm	1 to 1023	Set harmonic of the carrier frequency. 1=Fundamental
Harm	1 to 1023	Set harmonic of the first sideband frequency. 1 = fundamental
Harm	1 to 1023	Set harmonic of the second sideband frequency. 1 = fundamental
Demod Freq (Hz)	0 to 50 MHz	Carrier frequency used for the demodulation and signal generation on the carrier demodulator.
Demod Freq (Hz)	0 to 50 MHz	Absolute frequency used for demodulation and signal generation on the first sideband demodulator.
Demod Freq (Hz)	0 to 50 MHz	Absolute frequency used for demodulation and signal generation on the second sideband demodulator.
Channel	Signal Input 1, Signal Input 2	Select Signal Input for the carrier demodulation
Channel	Signal Input 1, Signal Input 2	Select Signal Input for the sideband demodulation

Control/Tool	Option/Range	Description
Phase	-180° to 180°	Phase shift applied to the reference input of the carrier demodulator and also to the carrier signal on the Signal Outputs
Phase	-180° to 180°	Phase shift applied to the reference input of the sideband demodulator and also to the sideband signal on the Signal Outputs
Zero		Adjust the carrier demodulator phase automatically in order to read zero degrees. Shifts the phase of the reference at the input of the carrier demodulator in order to achieve zero phase at the demodulator output. This action maximizes the X output, zeros the Y output, zeros the Θ output, and leaves the R output unchanged.
Zero		Adjust the sideband demodulator phase automatically in order to read zero degrees. Shifts the phase of the reference at the input of the sideband demodulator in order to achieve zero phase at the demodulator output. This action maximizes the X output, zeros the Y output, zeros the Θ output, and leaves the R output unchanged.
Order	1 to 8	Filter order used for carrier demodulation
Order	1 to 8	Filter order used for sideband demodulation
TC/BW Value	numeric value	Defines the low-pass filter characteristic in the unit defined above for the carrier demodulation
TC/BW Value	numeric value	Defines the low-pass filter characteristic in the unit defined above for the sideband demodulation
Signal Output	1, 2 or both	Select Signal Output 1, 2 or none
Carrier (V)	-range to range	Set the carrier amplitude

Control/Tool	Option/Range	Description
Modulation (V)	-range to range	Set the amplitude of the first sideband component.
Modulation (V)	-range to range	Set the amplitude of the second sideband component.
Index	-range to range	In FM mode, set modulation index value. The modulation index equals peak deviation divided by modulation frequency.
Peak Dev (Hz)	-range to range	In FM mode, set peak deviation value.
Enable FM Peak Mode	ON / OFF	In FM mode, choose to work with either modulation index or peak deviation. The modulation index equals peak deviation divided by modulation frequency.
Enable	ON / OFF	Enable the signal generation for the first sideband
Enable	ON / OFF	Enable the signal generation for the second sideband
Enable	ON / OFF	Enable the carrier signal

4.19. Real-time Tab

Note

The Real-time Tab is unavailable in the LabOne UI for use with HF2 Instruments. Please use ziControl to work with the Real-time Option.

4.20. HF2CA Tab

The HF2CA tab provides remote control over the HF2CA Current Amplifier which is available as an accessory to the HF2LI Lock-in amplifier. The HF2CA tab dynamically adapts its content depending on whether or not a HF2CA is connected to one of the ZCtrl connectors of the HF2LI.

4.20.1. Features

- Input impedance range from 10 V/A to 1 M V/A (R1, R2)
- Input mode differential or single-ended (Diff, Single)
- Input signal coupling mode (AC, DC)
- Output stage gain (G=1 or G=10)

4.20.2. Description

The HF2CA tab contains an interactive circuit diagram allowing the user to control the input settings, grounding, and gain of the preamplifier. 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 4.52. App Icon and short description

Control/Tool	Option/Range	Description
HF2CA		Remote control of the HF2CA Current Amplifier.

The HF2CA tab consists of two side-tabs corresponding to the two ZCtrl inputs of the HF2 instrument. Each side-tab is horizontally divided into two identical sections corresponding to the two Signal Inputs of the HF2CA as shown in [Figure 4.38](#).

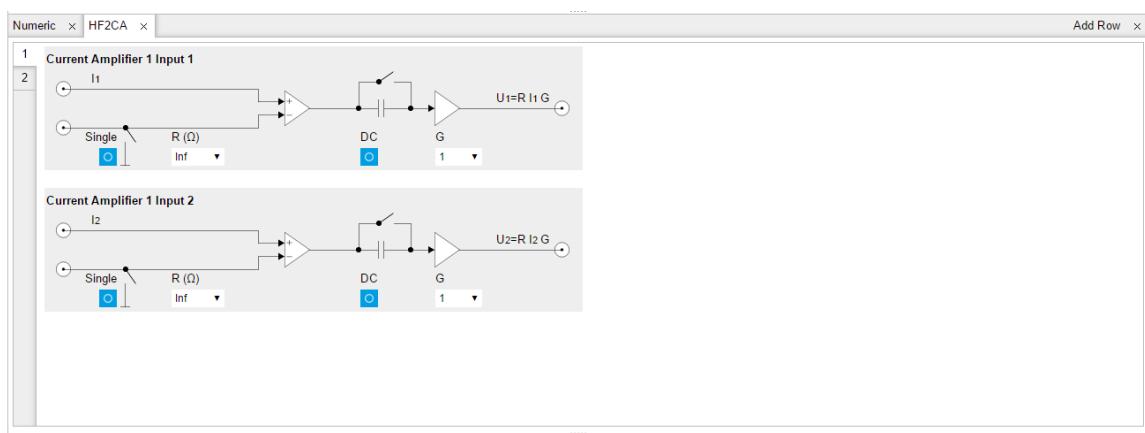


Figure 4.38. HF2CA tab

Additional HF2CA specification can be found in the [HF2CA Current Amplifier Data sheet](#).

4.20.3. Functional Elements

Table 4.53. HF2CA tab

Control/Tool	Option/Range	Description
Single	ON / OFF	Switch between differential and single-ended input configurations
R (Ω)	10, 100, 1k, 10k, 100k, 1M, Inf	Select input impedance
DC	ON / OFF	Switch between DC and AC coupling after the first amplification stage
G1 / G2	1, 10	Set the voltage gain of the second amplification stage

4.21. HF2TA Tab

The HF2CA tab provides remote control over the HF2TA Current Amplifier which is available as an accessory to the HF2LI Lock-in amplifier. The HF2CA tab dynamically adapts its content depending on whether or not a HF2TA is connected to one of the ZCtrl connectors of the HF2LI.

4.21.1. Features

- Input offset +/- 10 V
- Transimpedance gain from 100 V/A to 100 MV/A (R_1, R_2)
- Input signal coupling mode (AC, DC)
- Addition gain (1, 10)
- Total gain display ($R_1 \cdot G$, $R_2 \cdot G$)
- Input Shield (GND, EXT Bias)
- Auxiliary output +/- 10 V

4.21.2. Description

The HF2TA tab contains an interactive circuit diagram allowing the user to control the input settings, grounding, gain and offset of the preamplifier. 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 4.54. App Icon and short description

Control/Tool	Option/Range	Description
HF2TA		Remote control of the HF2TA Current Amplifier.

The HF2TA tab contains two side-tabs corresponding to the two ZCtrl inputs of the HF2 instrument. Each side-tab is horizontally divided into three sections as shown in [Figure 4.39](#). The two upper sections are identical and correspond to the two Signal Inputs of the HF2TA. The lowest section contains the setting of the Aux Output voltage of the HF2TA.

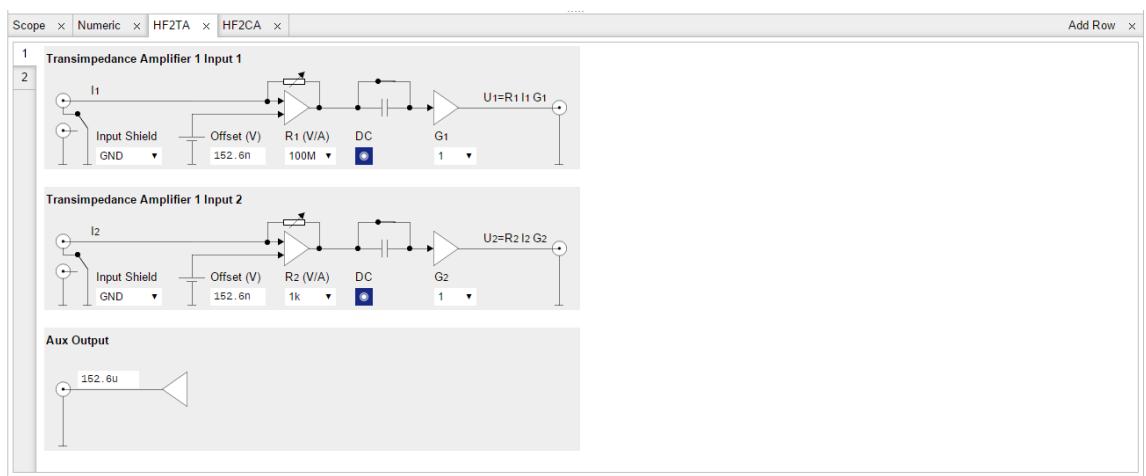


Figure 4.39. HF2TA tab

Detailed HF2TA specifications can be found in the [HF2TA Current Amplifier Data sheet](#).

4.21.3. Functional Elements

Table 4.55. HF2TA tab

Control/Tool	Option/Range	Description
Input Shield	GND, Ext Bias	Select the shield of Input 1 and 2 to be either grounded or biased by an external voltage
Offset (V)	-10 mV to +10 mV	Set the offset voltage applied to the current input.
R1 / R2 (V/A)	100, 1k, 10k, 100k, 1M, 10M, 100M	Set the transimpedance gain of the amplifier
DC	ON / OFF	Switch between DC and AC coupling after the first amplification stage
G1 / G2	1, 10	Set the voltage gain of the second amplification stage
Aux Output	-10 V to +10 V	Set the auxiliary output voltage of the HF2TA

4.22. 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 4.56. App Icon and short description

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

Chapter 5. Communication and Connectivity

This chapter describes the different possibilities to interface with an HF2 Instrument. The HF2 Series was designed with the concept that "the computer is the cockpit"; there are no controls on the front panel of the HF2 Instrument, instead the user has the freedom to configure and stream data from the instrument directly from their computer. The aim of this approach is to give the user the freedom to choose where they connect to, and how they control, their HF2 Instrument. The user can connect directly from a computer connected to the HF2 Instrument via USB or remotely from a different computer on the network, away from their experimental setup. Then, on either computer, the user can configure and retrieve data from their HF2 Instrument via a number of different interfaces, i.e. via the LabOne UI and/or their own custom programs. In this way the user can decide which connectivity setup and combination of interfaces best suits their experimental setup and data processing needs.

We first provide an overview of how the user connects an HF2 Instrument to a PC in [Section 5.1](#) and then give an overview of how to quickly modify instrument settings using the text-based console in [Section 5.2](#). Finally, at the end of this chapter, we explain how to connect to an HF2 instrument over a public network, [Section 5.3](#).

Note

It is also possible to configure and obtain data from an HF2 Instrument via one of our APIs. Currently LabVIEW, Matlab, Python or C are available. These topics are covered in a separate document, The LabOne Programming Manual.

Note

New users could benefit by first familiarizing themselves with the instrument using the LabOne UI, see [Chapter 3](#).

Note

Programming using the Real-time Option (ziRTK) is dealt with in [Section 7.2](#).

5.1. Instrument Connectivity Overview

The HF2 Series supports a server-based connectivity methodology for multi-user, multi-device operation. This means that it is possible to operate more than one HF2 Instrument from a single computer, that multiple users may access the same instrument, and that an instrument may be made available on a local area network. Server-based means that all communication between the user and the HF2 is via a computer program called a server, in our case ziServer. The ziServer program recognizes the device and manages all communication between the instrument and the host computer over the USB connection on one side, and the different available interfaces on the other side.

Before going into more detail, the terminology used in this chapter is explained.

- Host computer: The computer that is directly connected to the HF2 by USB. An HF2 can only be connected to one host computer, but to multiple remote computers on a local area network via ziServer running on the host.
- ziServer: A computer program that runs on the host computer and manages settings on, and data transfer to and from the HF2 by receiving commands from clients. It always has the most up-to-date configuration of the device and ensures that the configuration is synchronized between different clients.
- Remote computer: A computer, available on the same network as the host computer, that can communicate with the HF2 via the ziServer program running on the host.
- Client: A computer program that communicates with the HF2 via the server. The client can be running either on the host or the remote computer.
- API (Application Programming Interface): a collection of functions and data structures which enable communication between software components. In our case, the various APIs (e.g., LabVIEW, MATLAB®) for the HF2 provide functions to configure the device and receive measured experimental data.
- Interface: Either a client or an API.
- TCP/IP: Network communication protocols. In our case, ziServer communicates to the base API (ziAPI) using TCP/IP. This can happen either locally (entirely on the host computer) or between the host computer and remote computers.
- GUI (Graphical User Interface): A computer program that the user can operate via images as opposed to text-based commands.
- Modules: Software components that provide a unified interface to APIs to perform high-level common tasks such as sweeping data.

An overview of HF2 Instrument connectivity is shown in [Figure 5.1](#).

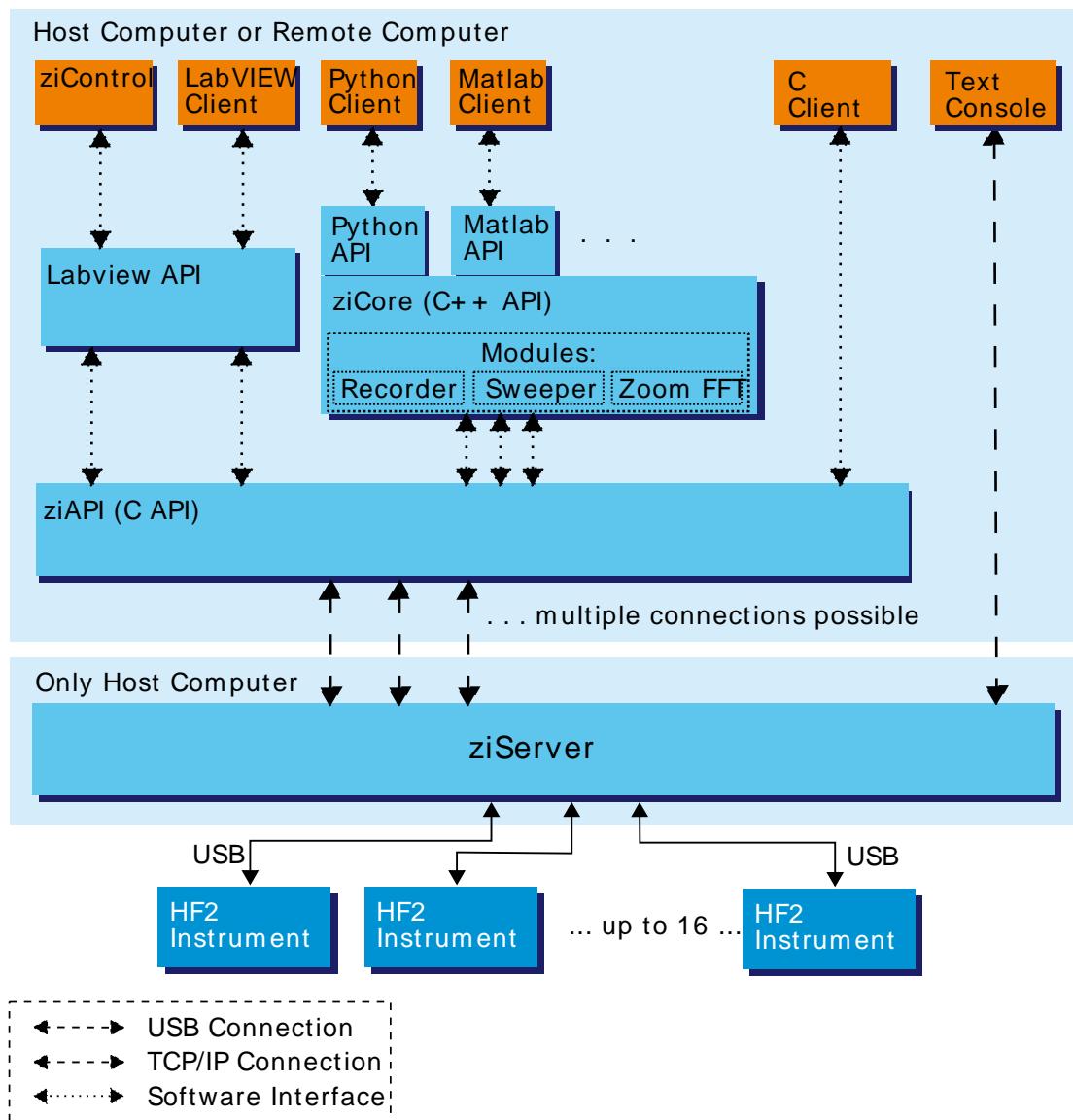


Figure 5.1. Instrument Connectivity

5.1.1. Physical Connectivity: Host and Remote Computers

In a commonly used configuration, the HF2 Instrument is connected to a host computer where both the server and the clients (denoted by the orange boxes in Figure 5.1) run. The ziServer program acts as bridge from the instrument to the various clients. For example, a user may use two clients in parallel: the LabOne UI to configure the device and their own program created using the LabVIEW API to plot custom results streamed from the instrument. Both these clients communicate data via the same instance of ziServer and ziServer ensures that both clients are always updated with the current instrument configuration. Note however, that any combination of clients shown in Figure 5.1 may be used in parallel, limited only by the performance of the host computer and by the load from requests to ziServer. In this configuration, the top and bottom block of Figure 5.1 (denoted by the light blue box) are both running on the host computer.

Sometimes, the user wishes to use a client to control the HF2 on a remote computer. In this case, the software in the top block of Figure 5.1 runs on the remote computer, connecting via TCP/IP over the local area network to the instance of ziServer running on the host computer (which is connected to the HF2 via USB).

In total, there are three possibilities of physically connecting to an HF2 Instrument:

- On the host computer, i.e., all the software (ziServer, interfaces) is running on the same computer that is connected to the instrument via USB. This is the simplest and most common setup.
- On a remote computer connected to the host computer over a secure local area network. If a private network is available this is a simple setup, ziServer only needs to allow remote connections, see [the section called “Enabling a Remote Connection to ziServer”](#).
- On a remote computer connected to the host computer over a public, insecure network. This is a more advanced topic covered in [Section 5.3](#).

As you can now imagine, there are many possibilities to connect to an HF2 instrument. The following methods of connecting with HF2 Instruments are supported:

- Connection to and operation of an HF2 Instrument from multiple clients on different computers in parallel with automatic background update of instrument settings on all connected clients.
- Connection to and operation of up to 16 HF2 Instruments from a single host computer.
- Connection to and Operation of multiple remote HF2 Instruments that are connected on a TCP/IP LAN via a (or multiple) host computer(s), the number of which is limited by the performance of the remote computer. Note, there can only be one instance of ziServer running for one HF2 Instrument.

5.1.2. Software Connectivity: ziServer

The ziServer program provides a gateway to your HF2 Instrument from any of the programming interfaces described in this chapter. The ziServer program recognizes the device and manages all communication between the instrument and the host computer over the USB connection on one side, and the different available interfaces on the other side. Since ziServer is responsible for all communication to the instrument, it's important that only one instance of ziServer is running at any one time. This is how you can check that only one instance of ziServer is running, or is indeed running at all:

- Windows: Open Windows Task Manager with CTRL-SHIFT-ESC and check that both the processes `ziServer.exe` and `ziService.exe` are running.
- Linux: Either check manually that the process `ziServer` is running or alternatively use the `ziService` command

```
$ ziService status
```

in a terminal. You should see the output:

```
Status : ziServer is running.
```

Enabling a Remote Connection to ziServer

In order to enable connections to ziServer from a remote computer, the node `/zi/config/open` must be set to 1. To set this in the LabOne UI go to the Config Tab and under the Connectivity setting enable "From Everywhere".

5.1.3. Instrument Communication: The Node Hierarchy

In order to communicate with an HF2 Instrument via text-based commands, it is necessary to understand how the settings and measurement data of the instrument are accessed. All settings of the HF2 Instrument are organized in a file-system-like hierarchical structure. This means that it

is possible to plot a consistent tree of nodes, where the instrument settings are leaves of the tree. It is also possible to browse branches inside the tree as if the user were navigating in a file-system. This hierarchy is used, no matter which interface you use when performing measurements.

An example demonstrating the hierarchy is the representation of the first demodulator on the device, given by the node:

```
/devX/demods/0
```

which, as we've already noted, is very similar to a **path** on a computer's file-system. Note that, the top level of the path is the device that you are connected to. The demodulators are then given as a top-level **node** under your device-node and the node of the first demodulator is indexed by 0. This path represents a branch in the node hierarchy which, in this case, if we explore further, has the following nodes:

```
/devX/demods/0/adcselect  
/devX/demods/0/order  
/devX/demods/0/timeconstant  
/devX/demods/0/rate  
/devX/demods/0/trigger  
/devX/demods/0/oscselect  
/devX/demods/0/harmonic  
/devX/demods/0/phaseshift  
/devX/demods/0/sinc  
/devX/demods/0/sample
```

These nodes are **leaves**, the most bottom-level nodes which represent a setting of an instrument or a field that can be read to retrieve measurement data. For example, `/devX/demods/0/adcselect` is the leaf that controls the setting corresponding to the choice of signal input for the first demodulator. To set the index of the signal input the user writes to this node. The leaf `/devX/demods/0/sample` is the leaf where the demodulator's output (timestamp, demodulated x-value, demodulated y-value) are written at the frequency specified by `/devX/demods/0/rate`. In order to obtain the demodulator output you read the values from this node by **polling** this node. Polling a node sends a request from the client to ziServer to obtain the data from the node at that particular point in time.

[Chapter 6](#) provides a full reference of nodes on HF2 Instruments and details which settings or measurement data they correspond to, whether they are read-only and, if they are writable, which values they may take (e.g., boolean, integer, floating point).

Note

The numbering on the front panel of the HF2 Instrument and the block numbering on the LabOne UI generally start with 1, whereas the underlying instrument using the programming interfaces has a numbering notation starting with 0.

Note

A useful method to learn about paths in your HF2 Instrument is to look at the output of the history in the bottom of the LabOne UI. The status line always shows the last applied command and you can view the entire history by clicking the "Show History" button. You will find paths like

```
/devx/sigins/0/ac = 1
```

after you switched on the AC mode for signal input 1, or

```
/devx/demods/1/rate =  
7200.000000
```

after setting the readout rate of demodulator 2 to 7.2 kHz.

You can obtain a list of nodes available on your instrument as a text-file in the LabOne UI by saving the instrument settings. Go to the Config tab in the Settings section click the Save button.

Note

We recommend that users who want to program their HF2 Instruments first familiarize themselves with the node hierarchy by browsing nodes via ziServer's text-based interface described in the next chapter. The text-based interface is an indispensable tool for HF2 programmers.

5.2. ziServer's Text-based Interface

The text-based interface is the simplest and most direct way of communicating with an HF2 Instrument and doesn't require any previous programming experience. Browsing the text interface physically happens within ziServer and since it makes use of TCP/IP sockets the user can also connect remotely over a network connection via telnet or ssh. In contrast to the LabOne UI, this is a geeky way of using an HF2 Instrument.

After connecting to the text-based interface via telnet, you find yourself in a DOS or Unix terminal-like program, where you can browse instrument settings in the node hierarchy ([Section 5.1.3](#)). The terminal responds to known command syntax like `ls` (list all nodes in the current directory) and `cd` (select path to navigate in the directory hierarchy).

The text interface is a very powerful tool for users programming an HF2 Instrument with other interfaces such as Zurich Instrument's LabVIEW or MATLAB® API. It is a convenient way to verify the instrument's node paths and check that values have been set correctly by the interface you are actually programming with. It is also helpful for budding HF2 hackers who can use it to browse the node hierarchy and familiarize themselves with its structure.

Note

In theory, it would be possible to use the text-based interface to communicate with an HF2 Instrument from an arbitrary programming environment. However, this would require the implementation of a socket connection and a parser, and there is no exception handling should a command fail. Also, since it's a text interface, as opposed to a binary interface, data transfer is slower. Therefore, in general, we strongly encourage the user to instead use one of the existing binary interfaces documented later in this chapter as their primary programming interface.

5.2.1. Getting Started with the Text-based Interface

Preparation

The purpose of this section is to get quickly acquainted with the text interface to the ziServer. For this you will need to have installed LabOne (see [Section 1.4](#)) and have your HF2 Instrument connected to your host computer via USB. In order to access the text-based interface within ziServer, a telnet or SSH client providing a console is required.

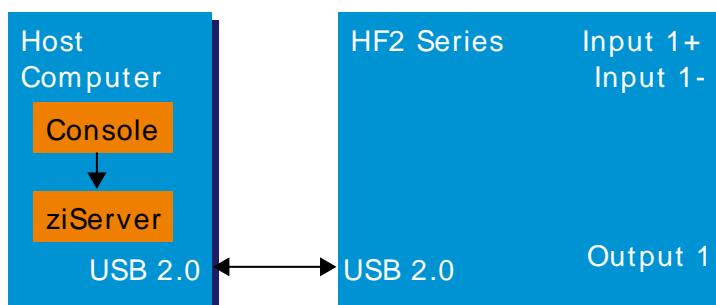


Figure 5.2. Setup for using the text-based interface

Connecting to ziServer on Windows

Zurich Instruments recommends to use the freeware PuTTY as a telnet client. PuTTY has to be configured with the following settings to connect with ziServer.

Table 5.1. PuTTY settings on Windows

Terminal category, Implicit CR in every LF	set
Session category, Host Name	localhost
Session category, Port	8005
Session category, Connection type	Telnet

Users connecting to a remote ziServer (a ziServer which is not running on the local machine, but on the host computer available on the LAN) have to configure the host name accordingly (e.g. computer.domain.com) after allowing remote connections to ziServer, see [Section 5.1.2](#).

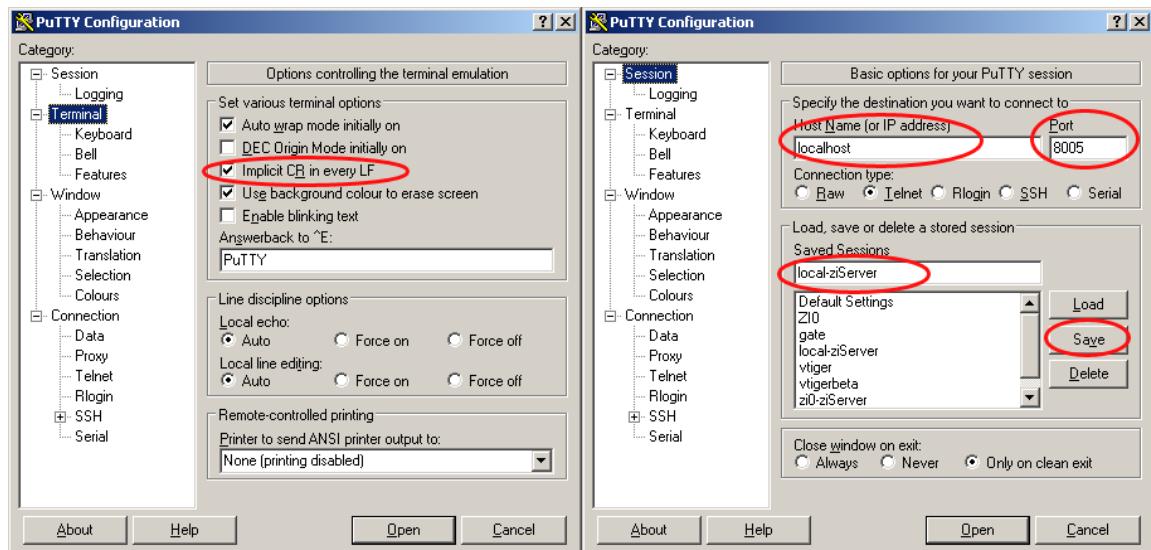


Figure 5.3. PuTTY configuration to connect to ziServer

Save the session settings with a suitable name, so that you can connect faster next time. After pressing the Open button, the following screen will appear: this message confirms successful connection to the ziServer. If the screen does not appear, or the text is missing, please check whether ziServer is running (Windows task manager, see [Section 5.1.2](#)) or check your PuTTY settings.

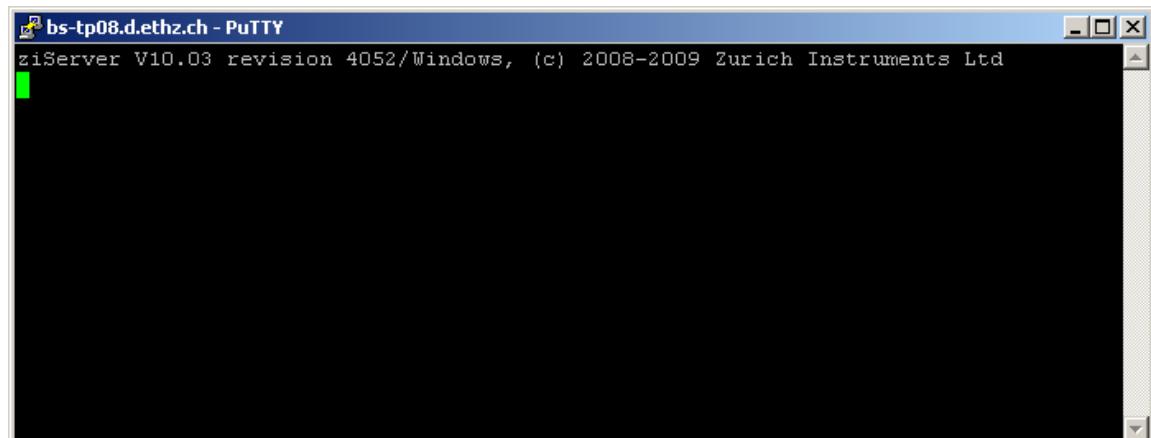


Figure 5.4. PuTTY successful ziServer connection

Connecting to ziServer on Linux

You may connect to a running ziServer from the host computer by invoking telnet in a shell:

```
user@zi:~$ telnet localhost 8005
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
ziServer V15.11 revision 34134/Linux, (c) 2008-2015 Zurich Instruments AG
```

Or by using netcat:

```
user@zi:~$ nc localhost 8005
ziServer V15.11 revision 34134/Linux, (c) 2008-2015 Zurich Instruments AG
```

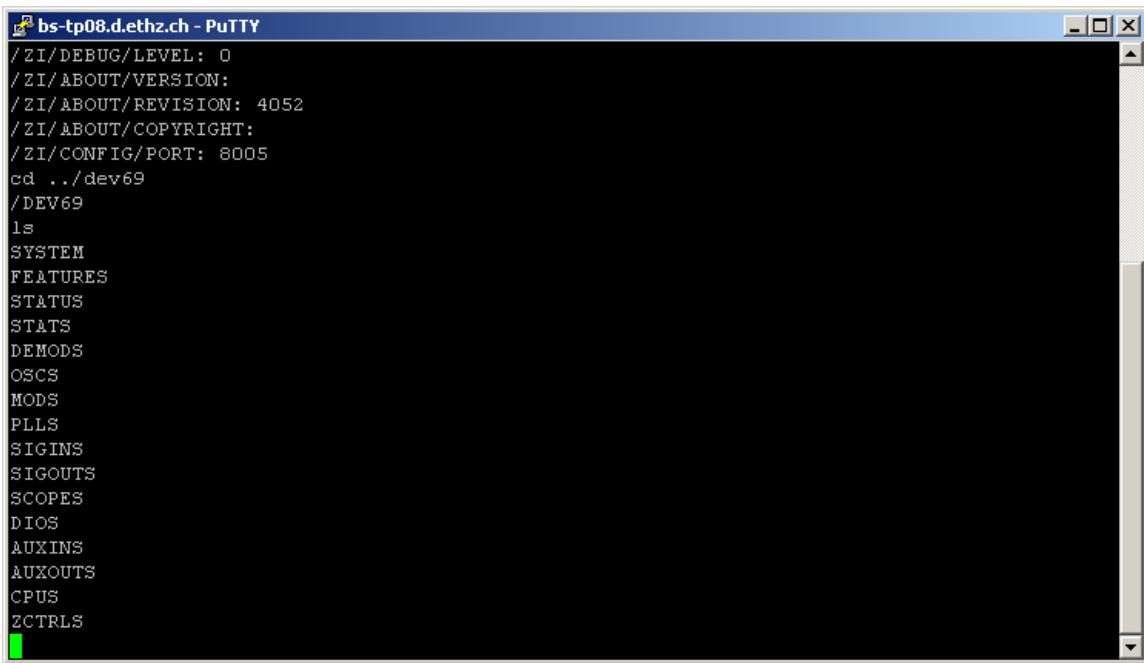
A Tour of the Text-based Interface

We start our tour with some basic commands. After successful connection, it's nice to see which instruments are connected to ziServer. An `ls` will do the job. This yields the information that we have a ZI node (the node for ziServer) and a DEVX node (denoting your HF2 Instrument). The DEVX is the serial number of the HF2 Instrument in front of you. Let's select the ziServer node with `cd zi`, list the nodes with `ls`, and then read all values of the node inside the `/ZI/` tree with `*//* ?`. Not very impressive so far.

```
bs-tp08.d.ethz.ch - PuTTY
ziServer V10.03 revision 4052/Windows, (c) 2008-2009 Zurich Instruments Ltd
ls
ZI
DEV69
cd zi
/ZI
ls
ABOUT
CONFIG
TREES
DEBUG
*//* ?
/ZI/CONFIG/OPEN: 0
/ZI/DEBUG/LEVEL: 0
/ZI/ABOUT/VERSION:
/ZI/ABOUT/REVISION: 4052
/ZI/ABOUT/COPYRIGHT:
/ZI/CONFIG/PORT: 8005
```

Figure 5.5. PuTTY tour: check server version

Let us move into the DEVX hierarchy by using the relative path `cd .../devx` (it's also possible to specify absolute paths, e.g., `cd /devx/` and investigate the structure of the node hierarchy with the `ls` command. This lists all the leaves inside of your device. Each leaf represents a setting that can be made inside of the instrument or a field that can be read to retrieve measurement data. The first level hierarchy inside the instrument is displayed in Figure 5.6.



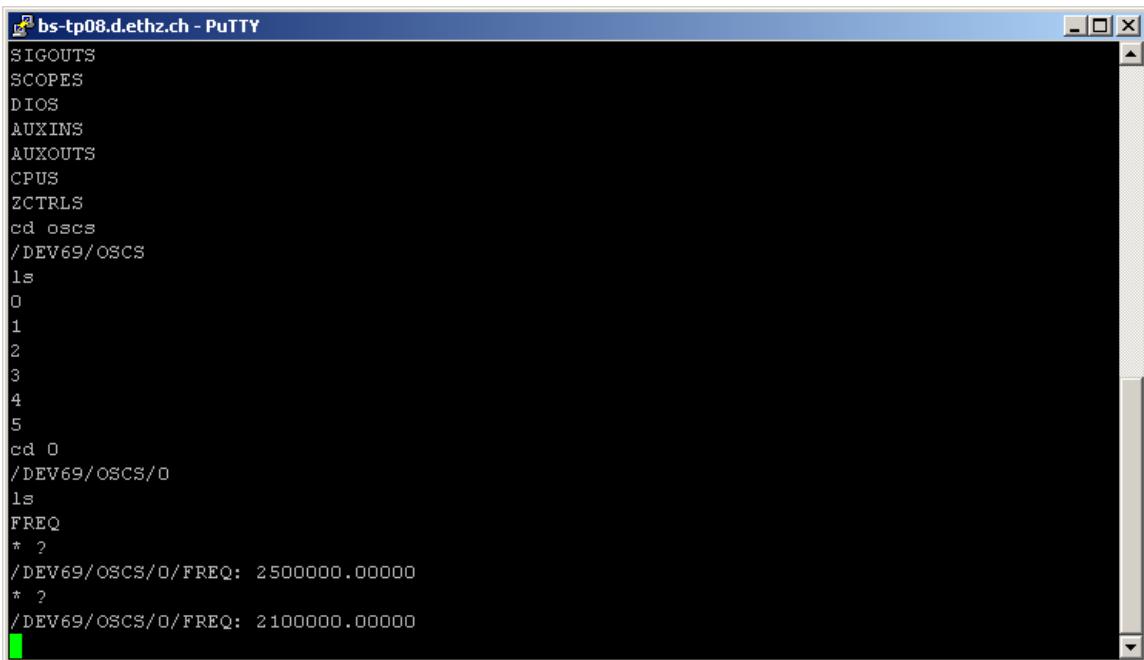
```
/ZI/DEBUG/LEVEL: 0
/ZI/ABOUT/VERSION:
/ZI/ABOUT/REVISION: 4052
/ZI/ABOUT/COPYRIGHT:
/ZI/CONFIG/PORT: 8005
cd ../dev69
/DEV69
ls
SYSTEM
FEATURES
STATUS
STATS
DEMOS
OSCS
MODS
PLLS
SIGINS
SIGOUTS
SCOPES
DIOS
AUXINS
AUXOUTS
CPUS
ZCTRLS
```

Figure 5.6. PuTTY tour: first instrument hierarchy

This list gives a top-level insight into an HF2 Instrument showing its building blocks such as DEMODS (demodulators), OSCEs (oscillators), SIGINS (signal inputs), SIGOUTS (signal outputs), SCOPES (oscilloscopes), AUXINS (auxiliary inputs), AUXOUTS (auxiliary outputs), CPUS (integrated processors), and so on. The branches and leaves that you see will depend on the options installed in your device: for instance, you will not see PLLS if you do not have the HF2PLL option installed.

It is time to dive into one branch of the instrument. Let us take oscillator 0: type `cd oscs`, then `ls` to see the branches at that level, then type `cd 0` to select the first oscillator, then list the leaves at that level, and use `* ?` to return the values of all leaves. We see for instance that `/DEVX/OSCS/0/FREQ` has a value of 2.5 MHz, see [Figure 5.7](#).

It is now possible to check that the LabOne UI actually has the same value in the corresponding field. Note, that the block numbering notation inside of the GUI starts with 1, whereas the underlying instrument has a numbering notation starting with 0. It is also possible to change the frequency of the lock-in channel 1 inside of the GUI to 2.1 MHz, and then check the value inside the text interface by typing `* ?`. You notice that the settings changes are transparent to all clients connected to a ziServer. You can always rely on setting and data consistency.

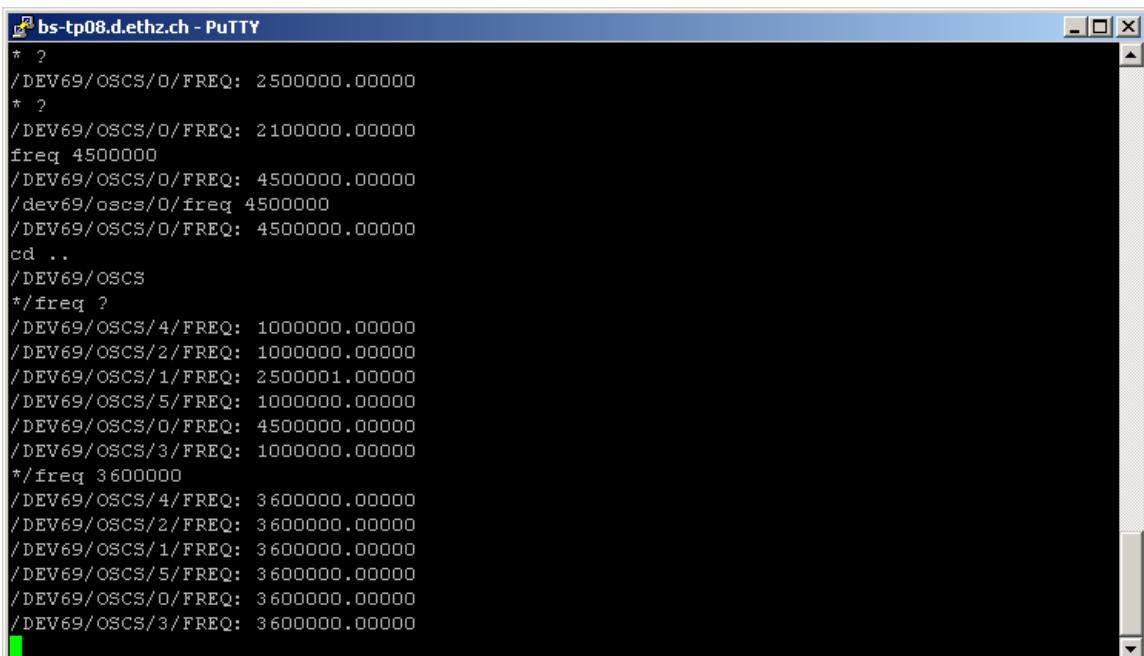


```
bs-tp08.d.ethz.ch - PuTTY
SIGOUTS
SCOPES
DIOS
AUXINS
AUXOUTS
CPUS
ZCTRLS
cd oscs
/DEV69/OSCS
ls
0
1
2
3
4
5
cd 0
/DEV69/OSCS/0
ls
FREQ
* ?
/DEV69/OSCS/0/FREQ: 2500000.00000
* ?
/DEV69/OSCS/0/FREQ: 2100000.00000
```

Figure 5.7. PuTTY tour: leaves of an oscillator

Next, to change the value of the oscillator frequency, for instance to 4.5 MHz, type freq 4500000. The same effect can be achieved by using the absolute path /DEV8/OSCS/0/FREQ 4500000. Please note that the value in the GUI has changed from 2.1 MHz to 4.5 MHz in the meantime.

The wildcard symbol * can simplify life when many similar settings need to be made. Lets for instance check the frequency of all oscillators at once: type cd .., and then */freq ?, and then change all frequencies to 3.6 MHz with */freq 3600000. This is where the text interface is becoming pretty powerful.



```
* ?
/DEV69/OSCS/0/FREQ: 2500000.00000
* ?
/DEV69/OSCS/0/FREQ: 2100000.00000
freq 4500000
/DEV69/OSCS/0/FREQ: 4500000.00000
/dev69/oscs/0/freq 4500000
/DEV69/OSCS/0/FREQ: 4500000.00000
cd ..
/DEV69/OSCS
*/freq ?
/DEV69/OSCS/4/FREQ: 1000000.00000
/DEV69/OSCS/2/FREQ: 1000000.00000
/DEV69/OSCS/1/FREQ: 2500001.00000
/DEV69/OSCS/5/FREQ: 1000000.00000
/DEV69/OSCS/0/FREQ: 4500000.00000
/DEV69/OSCS/3/FREQ: 1000000.00000
*/freq 3600000
/DEV69/OSCS/4/FREQ: 3600000.00000
/DEV69/OSCS/2/FREQ: 3600000.00000
/DEV69/OSCS/1/FREQ: 3600000.00000
/DEV69/OSCS/5/FREQ: 3600000.00000
/DEV69/OSCS/0/FREQ: 3600000.00000
/DEV69/OSCS/3/FREQ: 3600000.00000
```

Figure 5.8. PuTTY tour: using the wildcard symbol

One word on scripting. It is possible to manually compile several settings in a file using the syntax `path value`, then to copy-paste them into the terminal window. The sequence will be recognized by the ziServer and all defined settings will be made.

Note, another useful method to learn about the paths in your HF2 Instrument is to look at the bottom of the LabOne UI after changing configuration (see this [Note](#)). The complete command history of a session is stored in the LabVIEW Data directory, file `com.zhinst.ziControlStatusLog.txt`.

This concludes getting started with text-based programming. Zurich Instruments hopes you found it useful, and hopes you are going to perform some tutorials in [Chapter 3](#). Thank you for measuring with Zurich Instruments.

5.2.2. Command Reference

Commands

The following lists all available commands in the text-based interface, it can be viewed in the interface by typing `help`.

```
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^'.
ziServer V15.11 revision 34134/Linux, (c) 2008-2015 Zurich Instruments AG

help

ziServer V15.11 revision 34134/Linux, (c) 2008-2015 Zurich Instruments AG
```

All parameters from and to the device are organized in a tree structure. This simple interface allows you to interact with the device and use all its features based on an ascii-protocol.

The interface holds a current working node to which all paths you give are relative unless you begin a path with a `/`. You may use `..` to reference a node one level higher in hierarchy. In all paths wildcards `**` may be used.

The following commands are recognized by the ascii-interface:

```
[path] [value] Sets the value of the node(s) at the specified path.
[path] ? Gets the value of the node(s) at the specified path.
sel|cd [path] Sets the working path according to the provided one.
sel|cd ? Prints the current working path.
ls [path] Lists all children available in the given path.
tr [path] Prints all children and children's children of the specified path as a
tree.
subs [path] Subscribes the nodes of the given path for event forwarding.
unsubs [path] Unsubscribes the nodes of the given path from event forwarding.
exit Terminates the running session
help Shows this page
```

Note

The text-based interface is case insensitive.

Nodes, Leaves and Paths

Every setting of the instrument is represented by a leaf as a terminal of a tree of nodes. There are also leaves which are not settings, but for instance used to retrieve data from the instrument. For each leaf there is a path and the related value.

```
path_list = path [path]
path = [/|/..|*]name[/name|*|**]
```

In the syntax above a name is a string, and the path is a list of names separated by a slash. If a path starts with a slash, it is an absolute path starting at the root of the hierarchy. The asterisk is a wildcard meaning all nodes at a given hierarchy, and two points in a row means one hierarchy higher.

Navigation and Trees

The navigation inside the text interface is performed with the `sel`/`cd`/`ls`/`tr` commands.

```
sel or cd [?|..|path]
ls [path]
tr [path]
info [path]
```

The command `cd ?` feedbacks the current path, `cd ..` moves up one tree level, `cd path` moves down one tree level. `sel` and `cd` are equivalent commands. `ls` lists the tree available on the current path, `ls path` lists the tree available on the specified path, `tr` lists the complete tree on the current path, `tr path` lists the complete tree on the specified path, `info` feedbacks the help string of the current path, and `info path` reports the help string of a given path.

Get and Set Node Values

The values of nodes are read and changed with the following syntax.

```
path ?
path_list value
```

The command `path ?` returns the value of path, `path value` sets the specified node to value, and `path_list value` sets several nodes to value. Some examples:

```
about/* ?           // return values of leaves at path
devx/demods/0/* ?
/zi/config/* ?

/devx/demods/0adcselect ?    // return value at path
/devx/demods/0adcselect 0    // set value of leaf

/devx/demods/0adcselect /devx/demods/1adcselect 1
                           // multiple set value
```

Subscriptions

The ziServer provides a mechanism to automatically send all changes to a leaf to a subscribed client. This mechanism efficiently informs a client whenever a setting or a data of the instrument has changed without the need of active polling. It is possible to subscribe to single leaves, or full trees.

When a value of a subscribed leaf changes, the updated value is sent to the client. Most often samples, error and status nodes are subscribed. If one needs to maintain a user interface, then this can be done using subscriptions.

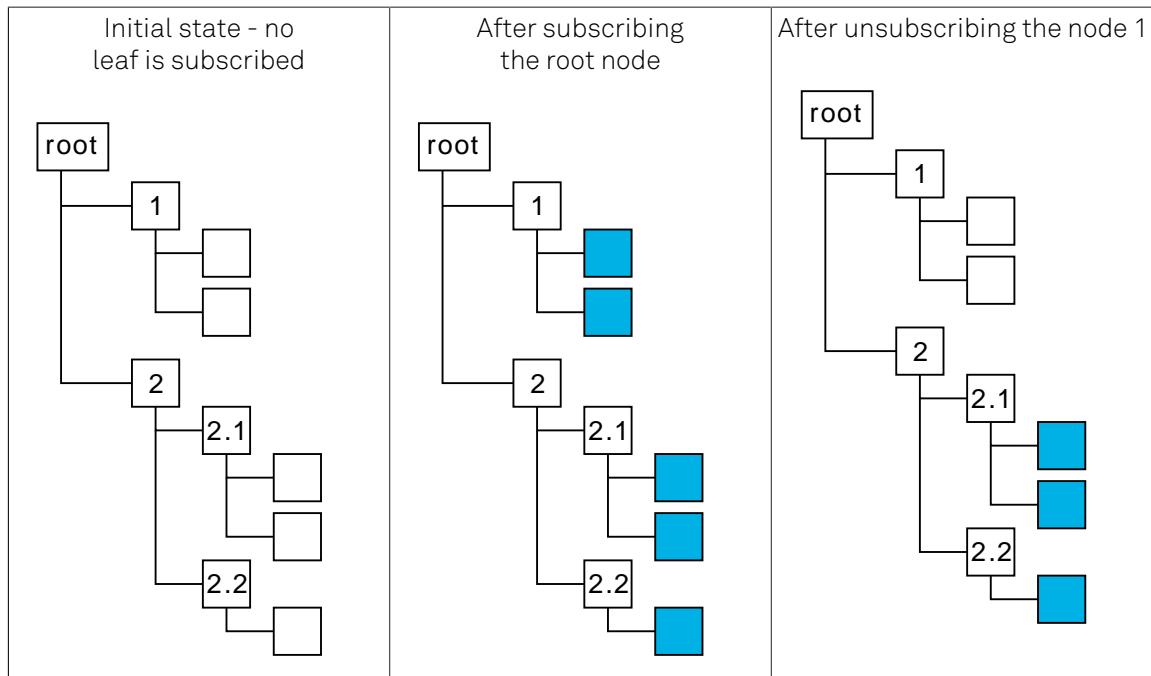
If you subscribe or unsubscribe from a node which is not a leaf, the subscription propagates to all nodes of the subtree. For example, you could first subscribe a subtree and then unsubscribe

specific nodes within this subtree and still receive events for all nodes except for unsubscribed ones.

```
subs path_list      // subscribe
unsubs path_list    // unsubscribe

path value          // return value for subscribed leaf
```

The following sequence illustrates subscribe and unsubscribe commands following each other, where turquoise leafs denote subscribed leafs.



The first image shows the initial state with no leaf subscribed. This state corresponds to a newly initiated ziServer session. After subscribing the root node, all leafs become subscribed. Then it is for instance possible to unsubscribe node 1 in order to leave just the leafs below node 2 subscribed.

Scripting

It is possible prepare a sequence of commands in a text editor and copy-paste them into the terminal session. The console will send all commands to the ziServer and the server will interpret them one by one.

Note

Use the right-mouse button in order to copy-paste into a Windows Putty session.

```
cd /DEVX
SIGOUTS/*/ON 0
SIGOUTS/0/RANGE 1
SIGOUTS/0/AMPLITUDES/0 1
SIGOUTS/0/ENABLES/* 0
SIGOUTS/0/ENABLES/0 1
OSCS/0/FREQ 300000
SIGINS/0/RANGE 10
DEMODS/0/ORDER 2
DEMODS/0/RATE 1000
```

5.3. Connecting to ziServer over insecure or firewalled networks

If you want to connect to the ziServer over insecure, public networks like the public internet, you need to consider that the TCP/IP connection to the ziServer is unsecured. Also many firewalls will not allow traffic to port 8005. There are two common solutions to this problem. Either a VPN or a ssh port tunneling/forwarding. In this section ssh port tunneling/forwarding is described.

5.3.1. SSH port forwarding

You can use ssh to connect to a remote computer and use this connection to tunnel ziServer traffic between the local and remote computer.

To illustrate how port forwarding works, let us use an example. Suppose you have two buildings. In Lab #1, there is the lab with computers residing in the subnet 10.1.1.* and the HF2 is connected to one of these computers. At your Home, there are office computers residing in the subnet 10.2.2.*. The computers in Lab #1 are running the ziServer application that uses an unencrypted TCP/IP session to communicate data with, e.g., the LabOne UI at your home. The firewall of the Lab and your Home might not permit this connection to be initiated. There are two kinds of port forwarding: local and remote forwarding. They are also called outgoing and incoming tunnels, respectively. Local port forwarding forwards traffic coming to a local port to a specified remote port. For example, all traffic coming to port 1234 on the client could be forwarded to port 8005 on the server (host).

The value of localhost is resolved after the Secure Shell connection has been established – so when defining local forwarding (outgoing tunnels), localhost refers to the server (remote host computer) you have connected to. Remote port forwarding does the opposite: it forwards traffic coming to a remote port to a specified local port. For example, all traffic coming to port 1234 on the server (host) could be forwarded to port 8005 on the client (localhost).

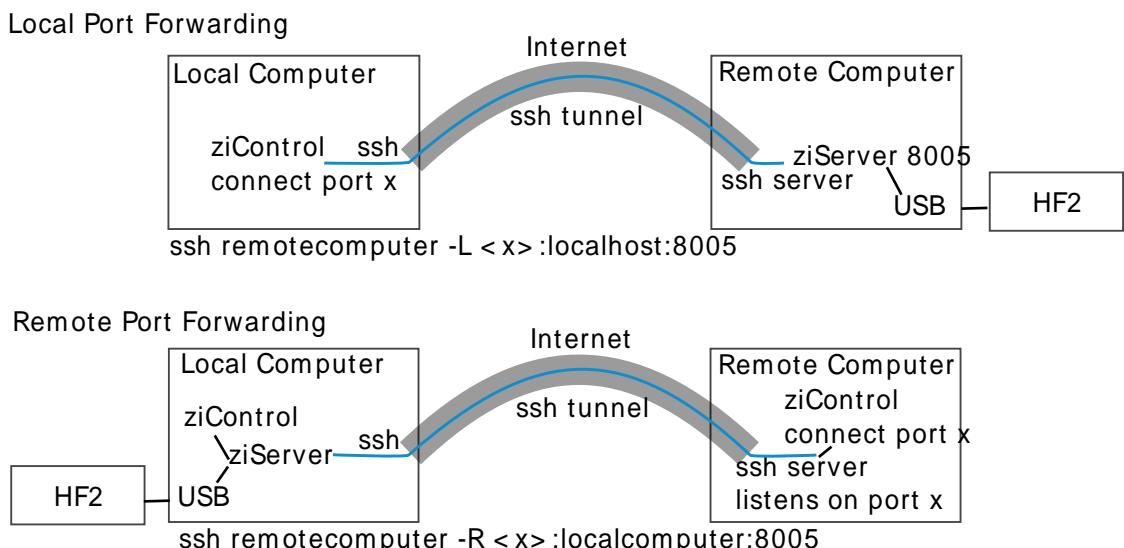


Figure 5.9. Secure connectivity

5.3.2. Local port forwarding

Accessing a service (in this example ziServer port TCP/8005) on a machine in the laboratory (10.1.1.*) from your machine at home (10.2.2.*), simply by connecting to the server work.example.org at work:

```
$ ssh user@work.example.org -L 10000:172.16.10.10:8005
```

We see the ziServer is available on the loop back interface only, listening on port TCP/10000:

```
$ netstat -tunelp | grep 10000  
tcp 0 0 127.0.0.1:10000 0.0.0.0:* LISTEN 1000 71679 12468/ssh
```

From your home machine, you should be able to connect to the machine at work:

```
$ telnet localhost 10000
```

By specifying localhost and port 10000 in the LabOne UI you can connect with the LabOne UI. Note that port 10000 is chosen arbitrarily.

5.3.3. Local port forward for anyone at home

If you want other people on your home subnet to be able to reach the machine at work by SSH, add the global option -g:

```
$ ssh user@work.example.org -L 10000:172.16.10.10:22 -g
```

We now see the service is available on all interfaces on your home computer (10.2.2.5), available for anyone to connect to on the local subnet:

```
$ netstat -tunelp | grep 10000 tcp 0 0 0.0.0.0:10000 0.0.0.0:* LISTEN  
1000 72265 12543/ssh
```

Anyone on your local subnet should be able to connect to the machine at work by doing this:

```
$ telnet 10.2.2.5 10000
```

By specifying host 10.2.2.5 and port 10000 in the LabOne UI you can connect with the LabOne UI.

5.3.4. Remote port forwarding

Giving access to a ziServer (port TCP/8005) on your home machine (10.2.2.5) to people at work:

```
$ ssh user@work.example.org -R 10000:10.2.2.5:8005
```

We see on our server at work (on the loop back interface on port TCP/10000) that we have access to our SSH server at home:

```
work.example.org$ netstat -tunelp | grep 10000 tcp 0 0 127.0.0.1:10000  
0.0.0.0:* LISTEN 0 73719534 3809/1
```

People logged in on the machine work.example.org now should be able to SSH into your home machine by doing:

```
work.example.org$ telnet localhost 10000
```

5.3.5. Remote port forwarding for anyone at work

If you want everybody on the subnet at work to be able to SSH into your home machine, there is no -g option for remote forward, so you need to change the SSH configuration of `work.example.org`, add to `sshd_config`:

```
GatewayPorts yes
```

Connect just as before:

```
home$ ssh user@work.example.org -R 10000:10.2.2.5:8005
```

Now, it is listening on all interfaces on the server at work:

```
work.example.org$ netstat -tunelp | grep 10000 tcp 0 0 0.0.0.0:10000  
0.0.0.0:* LISTEN 0 73721060 4426/1
```

Anyone at work can now connect to your home machine by SSH via the server:

```
anyone.example.org$ telnet work.example.org 10000
```

Chapter 6. Node Definitions

This chapter provides full information about the HF2 Instrument's settings and output. All settings (e.g., demodulator filter order) and results (e.g., demodulator output) are available to the user in a hierarchical tree structure which can be read from and written to. Having the settings of the instrument available to the user in a tree-like structure allows the user to program the instrument for readily repeating complicated experimental setups.

An overview of the node tree structure is provided in [Section 6.1](#) and detailed information about each node, such as whether it's readable or writable, is provided and in [Section 6.2](#).

Note

For an introduction to the node hierarchy see [Section 5.1.3](#) and to get familiar with reading from and writing to nodes in ziServer's text-based interface see [Section 5.2.1](#). See also the introduction to [Chapter 5](#) for an overview of the powerful means of programming the HF2 Instrument.

6.1. Overview

Table 6.1. Overview of the Node Tree

Node	Short Description
ZI	This node represents the instance of the ziServer your are connected to.
ABOUT	Node containing information about the server you are connected to.
VERSION	The version of this program.
REVISION	The revision of this program.
FWREVISION	The revision of the used firmware.
COPYRIGHT	The copyright string of this program.
DATASERVER	The name identifier of this data server.
CONFIG	Configuration data of the current instance of the server.
PORT	Configures the TCP/IP port on which the ziServer listens.
OPEN	Configures whether the ziServer should be open for connections from outside the local host.
TREES	Messages on tree changes.
CLOCKBASE	Provides timebase value for the server nodes
DEV0...n	This node represents a single device connected to the server.
CLOCKBASE	Provides clockbase value for the device
SYSTEM	Nodes providing system information and settings.
EXTCLK	Boolean value switching from internal to external clock.
HWREVISION	The revision of the main-board.
SYNCENABLE	Boolean value enabling multi-device timestamp synchronization over ZSync.
SYNCRESET	Boolean value activating timestamp reset over ZSync.
SYNCTIME	The timestamp to load when timestamp reset is activated.
ACTIVEINTERFACE	Node providing the active interface of the device.
PROPERTIES	Group of properties nodes.
MINFREQ	Minimum oscillator frequency of the device.
MAXFREQ	Maximum oscillator frequency of the device.

Node	Short Description
NEGATIVEFREQ	Device does support negative frequencies.
TIMEBASE	Time base of the device.
FREQRESOLUTION	Frequency resolution of the device.
MINTIMECONSTANT	Minimum filter time constant of the device.
MAXTIMECONSTANT	Maximum filter time constant of the device.
FEATURES	Node containing information on features of the device.
SERIAL	Node providing the serial number of the device.
DEVTYPE	Node providing a string about the type of device.
OPTIONS	Node giving information on enabled options.
CODE	Node providing a mechanism to write feature codes.
STATUS	Nodes providing status information from the device.
TIME	The current timestamp.
FLAGS	Node containing some status flags.
BINARY	A binary representation of all flags.
PLLLOCK	Flag indicating if the internal PLL for clock generation has locked.
DCMLOCK	Flag indicating if the internal digital clock manager (DCM) has locked.
FX2RX	Flag indicating if the device receives data via USB.
PKGLOSS	Flag indicating that the device lost data when sending via USB.
MIXERCLIP	Flags indicating that the internal mixer is clipping.
0...n	Flag indicating that this mixer-channel is clipping.
ADCCLIP	Flags indicating that the AD-converter is clipping.
0...n	Flag indicating that this ADC-channel is clipping.
SCOPESKIPPED	Flag indicating that scope data has been skipped.
DEMODSAMPLELOSS	Flag indicating that demodulator data has been lost.
FIFOLEVEL	Percentage of TX FIFO used.
ADCOMIN	The minimum value on Signal Input 1 (ADC0) during 100 ms.

Node	Short Description
ADCOMAX	The maximum value on Signal Input 1 (ADC0) during 100 ms.
ADC1MIN	The minimum value on Signal Input 2 (ADC1) during 100 ms.
ADC1MAX	The maximum value on Signal Input 2 (ADC1) during 100 ms.
ECHOWRITE	32bits written to this node will be echoed back via ECHOREAD node.
ECHOREAD	32bits written to ECHOWRITE node are echoed here.
STATS	Nodes providing statistical data about the device.
BYTESSENT	Total amount of bytes sent via USB.
BYTESRECEIVED	Total amount of bytes received via USB.
MEANPOLLCNT	Average poll-count.
MEANMSGCNT	Average message-count.
PHYSICAL	Group of nodes providing some physical information on the device.
1V2	Actual voltage of the 1.2 Volts supply.
1V8	Actual voltage of the 1.8 Volts supply.
2V5	Actual voltage of the 2.5 Volts supply.
3V3	Actual voltage of the 3.3 Volts supply.
5V0	Actual voltage of the 5.0 Volts supply.
TEMP	Actual temperature.
DEMOS	Demodulator nodes.
0...n	Nodes of a single demodulator
ADCSELECT	Selects the index of the signal input for the demodulator.
ORDER	Selects the order of the low-pass filter.
TIMECONSTANT	Sets the time constant of the low-pass filter.
RATE	The number of output values sent to the computer per second.
ENABLE	Enables the demodulator data stream.
TRIGGER	Sets the trigger- and gating-functionality of the demodulator.
OSCSELECT	Index of the oscillator used to demodulate the signal.
HARMONIC	The harmonic of the base frequency to be used.

Node	Short Description
FREQ	Frequency to of the demodulator.
PHASESHIFT	The phase shift of the demodulator.
SINC	Boolean value enabling Sinc filter functionality.
SAMPLE	Samples of the demodulator are given out at this node.
OSCS	Oscillator nodes.
0...n	Nodes of an oscillator.
FREQ	Frequency to of the oscillator.
MODS	Modulator option nodes.
0...n	Nodes of a Modulator.
ENABLE	Enables the modulation.
OUTPUT	Modulation output.
MODE	Modulation mode.
FREQDEVENABLE	In FM mode, enable peak deviation.
FREQDEV	In FM mode, set peak deviation value.
INDEX	In FM mode, set modulation index value. The modulation index equals peak deviation divided by modulation frequency.
RATE	The number of output values sent to the computer per second.
TRIGGER	Sets the trigger- and gating-functionality of the demodulator.
CARRIER	Group of carrier nodes.
INPUTSELECT	Signal Input for the carrier demodulation.
ORDER	Filter order for carrier demodulation.
TIMECONSTANT	Sets the time constant of the carrier low-pass filter.
OSCSELECT	Index of the oscillator used to demodulate the signal.
HARMONIC	Harmonic of the carrier frequency.
PHASESHIFT	The phase shift of the carrier demodulator.
ENABLE	Enables the carrier data stream.
AMPLITUDE	Carrier amplitude.
SIDEBANDS	Group of sideband nodes.
0...n	
MODE	Sideband selector.

Node	Short Description
INPUTSELECT	Signal Input for the sideband demodulation.
ORDER	Filter order for sideband demodulation.
TIMECONSTANT	Sets the time constant of the sideband low-pass filter.
OSCSELECT	Index of the oscillator used to demodulate the signal.
HARMONIC	Harmonic of the sideband frequency.
PHASESHIFT	The phase shift of the sideband demodulator.
ENABLE	Enables the sideband data stream.
AMPLITUDE	Sideband amplitude
SAMPLE	Modulation Samples.
PLLs	PLL nodes.
0...n	Nodes of a PLL.
ADCSELECT	Selects an input for the PLL.
AUTOCENTER	Switches auto-center.
FREQCENTER	Selects a center frequency.
FREQRANGE	Selects a frequency range for the PLL.
ENABLE	Enables the PLL.
ERROR	Error of the PLL.
AUTOTIMECONSTANT	Switches external time constant control.
TIMECONSTANT	The external time constant.
AUTOPID	Switches external PID.
P	Proportional gain of the PID controller.
I	Integral gain of the PID controller.
D	Derivative gain of the PID.
FREQDELTA	Frequency deviation from center frequency.
ADCTHRESHOLD	Threshold for edge detection.
AUXAVG	Delta frequency averaging control.
SETPOINT	The setpoint in degrees of the PLL.
HARMONIC	The harmonic of the base frequency to be used.
ORDER	Selects the order of the low-pass filter.
RATE	Update rate information.
OSCSELECT	Index of the oscillator used.

Node	Short Description
DEMODSELECT	Source demodulator.
PIDS	PID nodes.
0...n	Nodes of a PID.
INPUT	Selects the input for the PID.
INPUTCHANNEL	If applicable, selects the channel of the selected INPUT.
OUTPUT	Selects the output for the PID.
OUTPUTCHANNEL	If applicable, selects the channel of the selected OUTPUT.
OUTPUTDEFAULTENABLE	If OUTPUTDEFAULTENABLE is set, the value specified by OUTPUTDEFAULT will be applied when the PID is switched off.
OUTPUTDEFAULT	If OUTPUTDEFAULTENABLE is set, this node specifies the value to be applied.
P	Proportional gain.
I	Proportional gain for integrator.
D	Proportional gain for differentiator.
SETPOINT	Target settle point.
SETPOINTSELECT	Set point selection.
MONITOROFFSET	Offset for the monitor output.
MONITORSCALE	Scale for the monitor output.
ERROR	Shows the error value.
CENTER	Sets the output center point.
RANGE	Sets the output range.
SHIFT	Shows the output shift.
ENABLE	Enable PID controller.
RATE	Control update rate.
TIPPROTECT	Contains nodes for configuring the TipProtect functionality.
ENABLE	Enable TipProtect for the PID controller.
PLL	Selects a PLL for TipProtect.
ACTIVE	Indicates whether TipProtect is active.
ACTIVETIMECONSTANT	Time constant when TipProtect is active.
ACTIVETHRESHOLD	Threshold for the active state.
INACTIVETIMECONSTANT	Time constant when TipProtect is inactive.
INACTIVETHRESHOLD	Threshold for the inactive state.

Node	Short Description
IMPS	Impedance analyzer nodes.
0...n	Nodes of a impedance analyzer.
ENABLE	Enables the impedance data stream.
MODEL	Impedance model select.
FREQ	Control frequency to of the oscillator.
CURRENT	Nodes of a impedance current.
INPUTSELECT	Selects the index of the signal input for the impedance current demodulator.
DEMODSELECT	Index of the demodulator for the current impedance signal.
INVERT	Invert the current signal of the impedance.
RANGE	Current input range value in manual mode.
VOLTAGE	Nodes of a impedance voltage.
INPUTSELECT	Selects the index of the signal input for the impedance voltage demodulator.
DEMODSELECT	Index of the demodulator for the voltage impedance signal.
INVERT	Invert the voltage signal of the impedance.
RANGE	Voltage input range value in manual mode.
AC	Control input AC coupling.
DEMOD	Nodes of a impedance demodulator settings.
OSCSELECT	Index of the oscillator used to demodulate the signal.
HARMONIC	The harmonic of the base frequency to be used.
ORDER	Selects the order of the low-pass filter.
TIMECONSTANT	Sets the time constant of the low-pass filter.
SINC	Boolean value enabling Sinc filter functionality.
RATE	The number of output values sent to the computer per second.
FILTER	Switches between application modes and advanced mode.
APPLICATION	Selection of the impedance application.
INTERPOLATION	Selection of the interpolation method.
IMPEDANCERANGE	Selection of the impedance range.
MAXBANDWIDTH	Maximum bandwidth value.
OMEGASUPPRESSION	Omega suppression value.

Node	Short Description
VALUE	Bias value.
BIAS	Nodes of a impedance bias.
ENABLE	Enable bias.
VALUE	Bias value.
AUTO	Nodes of a impedance auto.
BW	Auto bandwidth.
INPUTRANGE	Impedance input range select.
OUTPUT	Automatic output amplitude control.
SUPPRESS	If set to one suppress any impedance auto functionality the next 30s.
CALIB	Nodes of a impedance calibration.
CABLELENGTH	Impedance calibration cable length compensation select.
USER	Nodes of a user impedance calibration.
ENABLE	Enable user calibration.
VALID	Indicator that user calibration data is valid.
ACTIVE	Indicator that user calibration is active.
INFO	Message string reporting calibration status, warnings, or errors.
STORE	Store user calibration.
DATA	Calibration data as JSON string.
INTERNAL	Nodes of a internal impedance calibration.
ENABLE	Enable internal calibration.
VALID	Indicator that internal calibration data is valid.
STORE	Store internal calibration.
DATA	Calibration data as JSON string.
INFO	Info string describing internal calibration data.
ACTIVE	Indicator that internal calibration is active.
CONFIDENCE	Nodes for impedance confidance indicators.
SUPPRESSION	Nodes for impedance parameter suppression.
ENABLE	Enable suppression analysis.
RATIO	Suppression ratio.
OPENDETECT	Nodes for impedance open detect.
ENABLE	Enable open detect analysis.

Node	Short Description
RATIO	Open detect ratio.
UNDERFLOW	Nodes for impedance underflow detect.
ENABLE	Enable underflow analysis.
RATIO	Underflow ratio.
COMPENSATION	Nodes for strong compensation detection.
ENABLE	Enable compensation analysis.
RATIO	Compensation ratio.
MODE	Impedance calibration mode select.
OUTPUT	Nodes of a impedance output.
ON	Switches the output on and off.
RANGE	Selects the output range for the Signal Output.
DEMOD	Index of the output demodulator.
SELECT	Selects the output channel used for impedance measurement.
RATE	The number of output values sent to the computer per second.
AMPLITUDE	Control output amplitude value when Drive Control is off.
SAMPLE	Impedance Samples.
SIGINS	Signal Input nodes.
0...n	Nodes of a signal input.
RANGE	Voltage range for the signal input.
AC	Boolean value setting for AC coupling of the Signal Input.
IMP50	Boolean value enabling 50 Ohm input impedance termination.
DIFF	Boolean value switching differential input mode.
SIGOUTS	Signal Output nodes.
0...n	Nodes of a Signal Output.
ON	Switches the output on and off.
ADD	Switches the output adder on and off.
RANGE	Selects the output range for the Signal Output.
OFFSET	Offset added to the Signal Output.
ENABLES	Switches for channels in the mixer.
0...n	Switches a channel in the mixer on and off.

Node	Short Description
AMPLITUDES	Amplitudes for channels in the mixer.
0...n	Fraction of the output range added to the output signal.
WAVEFORMS	
0...n	Waveforms for a channel in the mixer.
SCOPES	Scope nodes.
0...n	Nodes of a scope.
ENABLE	Enables the scope.
CHANNEL	Selects the channel for which scope data should be provided.
TRIGCHANNEL	Selects the channel which should be used as source for the scope's trigger.
BWLIMIT	The bandwidth-limit for the scope.
TRIGEDGE	Selects whether the scope should trigger on rising or falling edge.
TRIGLEVEL	Level at which a trigger is raised.
TRIGHOLDOFF	Time to wait for re-arming the trigger after one occurred.
TIME	Timescale of the scope wave (logarithmic decimation).
WAVE	Samples of scope-waveforms.
DIOS	DIO nodes.
0...n	Nodes of a DIO.
EXTCLK	Selects whether an external clock source should be used.
DECIMATION	Decimation for the sample rate of the DIO.
DRIVE	Selects if the outputs should be driven.
OUTPUT	Bits to output.
SYNCSELECT0	Source to output the sync signal on bit 0.
SYNCSELECT1	Source to output the sync signal on bit 1.
INPUT	Samples of the input.
AUXINS	Nodes of auxiliary inputs.
0...n	Node for an aux in.
AVERAGING	Averaging of the samples.
SAMPLE	Auxiliary input samples.
VALUES	Nodes of a auxins values.

Node	Short Description
0	Input 0 value.
1	Input 1 value.
AUXOUTS	Nodes of Auxiliary outputs.
0...n	Nodes of an Auxiliary output.
VALUE	Output value.
OUTPUTSELECT	Signal to be given out.
DEMODSELECT	Source demodulator.
SCALE	Scaling of the signal which is given out.
OFFSET	Value to be added to the output.
CPUS	Nodes for the real-time CPUs.
0...n	Node for one real-time CPU.
WORKLOAD	Usage of the processor-time.
PROGRAM	Node to write user programs in.
OUTPUT	Node which streams output of the user-program.
USERREGS	General purpose registers to transfer data.
0...n	General purpose register.
ZCTRLS	Node containing connected ZCtrl devices.
0...n	ZCtrl instance
CAMP	A ZI current-amplifier connected to a ZCtrl port.
AVAILABLE	1 when HF2CA is connected to the corresponding ZCtrl port.
R	Chooses a value for the shunt-resistor.
GAIN	Switches between factor 1 and 10 gain.
DC	Switches between AC coupling and DC coupling.
SINGLEENDED	Switches between differential and single-ended input.
TAMP	A ZI transimpedance-amplifier connected to a ZCtrl port.
AVAILABLE	1 when HF2TA is connected to the corresponding ZCtrl port.
BIASOUT	Switches between internal and external bias.
EXTBIAS	Switches the external bias.
0...n	A Channel of a transimpedance amplifier.

Node	Short Description
CURRENTGAIN	Chooses a value for the current gain.
DC	Switches between AC and DC Mode.
VOLTAGEGAIN	Chooses a value for the voltage gain.
OFFSET	Adjust offset value.

6.2. Nodes

6.2.1. /ZI

This node represents the instance of the ziServer your are connected to.

6.2.2. /ZI/ABOUT

Node containing information about the server you are connected to.

6.2.3. /ZI/ABOUT/VERSION

The version of this program.

Write	-
Read	-
Setting	No

6.2.4. /ZI/ABOUT/REVISION

The revision of this program.

Write	-
Read	Integer Number
Setting	No

6.2.5. /ZI/ABOUT/FWREVISION

The revision of the used firmware.

Write	-
Read	Integer Number
Setting	No

6.2.6. /ZI/ABOUT/COPYRIGHT

The copyright string of this program.

Write	-
Read	-
Setting	No

6.2.7. /ZI/ABOUT/DATASERVER

The name identifier of this data server.

Write	-
Read	-

Setting	No
---------	----

6.2.8. /ZI/CONFIG

Configuration data of the current instance of the server.

6.2.9. /ZI/CONFIG/PORT

Configures the TCP/IP port on which the ziServer listens.

Write	Integer Number
Read	Integer Number
Setting	No
Default	8005
Range	1024 to 65535

6.2.10. /ZI/CONFIG/OPEN

Configures whether the ziServer should be open for connections from outside the local host.

Write	Integer Number				
Read	Integer Number				
Setting	No				
Unit	Boolean				
Default	0 (server only listens to localhost)				
Values	<table> <tr> <td>0</td> <td>server only listens to localhost</td> </tr> <tr> <td>1</td> <td>server is open for connections from outside</td> </tr> </table>	0	server only listens to localhost	1	server is open for connections from outside
0	server only listens to localhost				
1	server is open for connections from outside				

6.2.11. /ZI/TREES

Messages on tree changes.

Write	-
Read	-
Stream	-
Setting	No

Details

This node sends out a message every time a device is connected or disconnected or its tree has changed.

6.2.12. /ZI/CLOCKBASE

Provides timebase value for the server nodes

Write	-
Read	Integer Number
Setting	No

6.2.13. /DEV0...n

This node represents a single device connected to the server.

Note

The number of this node is the serial number of the connected device.

6.2.14. /DEV0...n/CLOCKBASE

Provides clockbase value for the device

Write	-
Read	Integer Number
Setting	No

6.2.15. /DEV0...n/SYSTEM

Nodes providing system information and settings.

6.2.16. /DEV0...n/SYSTEM/EXTCLK

Boolean value switching from internal to external clock.

Write	Integer Number
Read	Integer Number
Setting	Yes
Unit	Boolean
Default	0 (Internal clock)
Values	0 Internal clock 1 External clock

Details

When using external clock, make sure that a clock generator is connected to the Clock In connector.

6.2.17. /DEV0...n/SYSTEM/HWREVISION

The revision of the main-board.

Write	-
Read	Integer Number
Setting	No

6.2.18. /DEV0...n/SYSTEM/SYNCENABLE

Boolean value enabling multi-device timestamp synchronization over ZSync.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Boolean	
Default	0 (Default)	
Values	0	Default
	1	Timestamp synchronization enabled

Details

When synchronizing timestamps between devices make sure that an appropriate cable is connected between the ZSync ports of the master and slave devices.

6.2.19. /DEV0...n/SYSTEM/SYNCRESET

Boolean value activating timestamp reset over ZSync.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Boolean	
Default	0 (Default)	
Values	0	Default
	1	Timestamp reset activated

Details

When synchronizing timestamps between devices make sure that an appropriate cable is connected between the ZSync ports of the master and slave devices.

6.2.20. /DEV0...n/SYSTEM/SYNCTIME

The timestamp to load when timestamp reset is activated.

Write	-
Read	Decimal Number
Setting	No
Unit	s

6.2.21. /DEV0...n/SYSTEM/ACTIVEINTERFACE

Node providing the active interface of the device.

Write	-
Read	-
Setting	No

6.2.22. /DEV0...n/SYSTEM/PROPERTIES

Group of properties nodes.

6.2.23. /DEV0...n/SYSTEM/PROPERTIES/MINFREQ

Minimum oscillator frequency of the device.

6.2.24. /DEV0...n/SYSTEM/PROPERTIES/MAXFREQ

Maximum oscillator frequency of the device.

6.2.25. /DEV0...n/SYSTEM/PROPERTIES/NEGATIVEFREQ

Device does support negative frequencies.

6.2.26. /DEV0...n/SYSTEM/PROPERTIES/TIMEBASE

Time base of the device.

Write	-
Read	Decimal Number
Setting	No
Unit	s

6.2.27. /DEV0...n/SYSTEM/PROPERTIES/FREQRESOLUTION

Frequency resolution of the device.

Write	-
Read	Integer Number
Setting	No
Unit	bits

6.2.28. /DEV0...n/SYSTEM/PROPERTIES/MINTIMECONSTANT

Minimum filter time constant of the device.

Write	-
Read	Decimal Number
Setting	No
Unit	s

6.2.29. /DEV0...n/SYSTEM/PROPERTIES/MAXTIMECONSTANT

Maximum filter time constant of the device.

Write	-
Read	Decimal Number
Setting	No

Unit	s
------	---

6.2.30. /DEV0...n/FEATURES

Node containing information on features of the device.

6.2.31. /DEV0...n/FEATURES/SERIAL

Node providing the serial number of the device.

Write	-
Read	-
Setting	No

6.2.32. /DEV0...n/FEATURES/DEVTYPE

Node providing a string about the type of device.

Write	-
Read	-
Setting	No

6.2.33. /DEV0...n/FEATURES/OPTIONS

Node giving information on enabled options.

Write	-
Read	-
Setting	No

Details

Reading this node returns a string containing a newline-separated list of all installed options.

6.2.34. /DEV0...n/FEATURES/CODE

Node providing a mechanism to write feature codes.

Write	-
Read	-
Setting	No

6.2.35. /DEV0...n/STATUS

Nodes providing status information from the device.

6.2.36. /DEV0...n/STATUS/TIME

The current timestamp.

Write	-
-------	---

Read	Decimal Number
Setting	No
Unit	s

6.2.37. /DEV0...n/STATUS/FLAGS

Node containing some status flags.

6.2.38. /DEV0...n/STATUS/FLAGS/BINARY

A binary representation of all flags.

Write	-																														
Read	Integer Number																														
Setting	No																														
Unit	bit-coded																														
Values	<table> <tr> <td>b0 = 1</td> <td>PLL unlocked</td> </tr> <tr> <td>b1 = 2</td> <td>HF clock unlocked</td> </tr> <tr> <td>b2 = 4</td> <td>FX2 RX error</td> </tr> <tr> <td>b3 = 8</td> <td>Package loss</td> </tr> <tr> <td>b4 = 16</td> <td>Output 1 clipped</td> </tr> <tr> <td>b5 = 32</td> <td>Output 2 clipped</td> </tr> <tr> <td>b6 = 64</td> <td>Input 1 clipped</td> </tr> <tr> <td>b7 = 128</td> <td>Input 2 clipped</td> </tr> <tr> <td>b8 = 256</td> <td>Scope skipped a shot</td> </tr> <tr> <td>b9 = 512</td> <td>FX2 TX buffer almost full</td> </tr> <tr> <td>b10 = 1024</td> <td>0</td> </tr> <tr> <td>b11 = 2048</td> <td>PLL unlocked (version without de-bouncing)</td> </tr> <tr> <td>b12 = 4096</td> <td>FX2 TX package lost</td> </tr> <tr> <td>b20 = 1.048576E6</td> <td>PLL 1 locked</td> </tr> <tr> <td>b21 = 2.097152E6</td> <td>PLL 2 locked</td> </tr> </table>	b0 = 1	PLL unlocked	b1 = 2	HF clock unlocked	b2 = 4	FX2 RX error	b3 = 8	Package loss	b4 = 16	Output 1 clipped	b5 = 32	Output 2 clipped	b6 = 64	Input 1 clipped	b7 = 128	Input 2 clipped	b8 = 256	Scope skipped a shot	b9 = 512	FX2 TX buffer almost full	b10 = 1024	0	b11 = 2048	PLL unlocked (version without de-bouncing)	b12 = 4096	FX2 TX package lost	b20 = 1.048576E6	PLL 1 locked	b21 = 2.097152E6	PLL 2 locked
b0 = 1	PLL unlocked																														
b1 = 2	HF clock unlocked																														
b2 = 4	FX2 RX error																														
b3 = 8	Package loss																														
b4 = 16	Output 1 clipped																														
b5 = 32	Output 2 clipped																														
b6 = 64	Input 1 clipped																														
b7 = 128	Input 2 clipped																														
b8 = 256	Scope skipped a shot																														
b9 = 512	FX2 TX buffer almost full																														
b10 = 1024	0																														
b11 = 2048	PLL unlocked (version without de-bouncing)																														
b12 = 4096	FX2 TX package lost																														
b20 = 1.048576E6	PLL 1 locked																														
b21 = 2.097152E6	PLL 2 locked																														

Details

When multiple flags are set the values are or-ed.

6.2.39. /DEV0...n/STATUS/FLAGS/PLLLOCK

Flag indicating if the internal PLL for clock generation has locked.

Write	-				
Read	Integer Number				
Setting	No				
Unit	Boolean				
Values	<table> <tr> <td>0</td> <td>PLL locked</td> </tr> <tr> <td>1</td> <td>PLL not locked</td> </tr> </table>	0	PLL locked	1	PLL not locked
0	PLL locked				
1	PLL not locked				

6.2.40. /DEV0...n/STATUS/FLAGS/DCMLOCK

Flag indicating if the internal digital clock manager (DCM) has locked.

Write	-
Read	Integer Number
Setting	No
Unit	Boolean
Values	0 DCM locked 1 DCM not locked

6.2.41. /DEV0...n/STATUS/FLAGS/FX2RX

Flag indicating if the device receives data via USB.

Write	-
Read	Integer Number
Setting	No
Unit	Boolean
Values	0 Device receives 1 Device does not receive

6.2.42. /DEV0...n/STATUS/FLAGS/PKGLOSS

Flag indicating that the device lost data when sending via USB.

Write	-
Read	Integer Number
Setting	No
Unit	Boolean
Values	0 no packet loss 1 packets are lost

6.2.43. /DEV0...n/STATUS/FLAGS/MIXERCLIP

Flags indicating that the internal mixer is clipping.

6.2.44. /DEV0...n/STATUS/FLAGS/MIXERCLIP/0...n

Flag indicating that this mixer-channel is clipping.

Write	-
Read	Integer Number
Setting	No
Unit	Boolean
Values	0 no clipping

	1	clipping
--	---	----------

6.2.45. /DEV0...n/STATUS/FLAGS/ADCCLIP

Flags indicating that the AD-converter is clipping.

6.2.46. /DEV0...n/STATUS/FLAGS/ADCCLIP/0...n

Flag indicating that this ADC-channel is clipping.

Write	-
Read	Integer Number
Setting	No
Unit	Boolean
Values	0 no clipping 1 clipping

6.2.47. /DEV0...n/STATUS/FLAGS/SCOPESKIPPED

Flag indicating that scope data has been skipped.

Write	-
Read	Integer Number
Setting	No
Unit	Boolean
Values	0 no data skipped 1 data skipped

Details

This happens when too much data is being sent over USB.

6.2.48. /DEV0...n/STATUS/FLAGS/DEMOSAMPLELOSS

Flag indicating that demodulator data has been lost.

Write	-
Read	Integer Number
Setting	No
Unit	Boolean
Values	0 no demodulator data lost 1 demodulator data lost

6.2.49. /DEV0...n/STATUS/FIFOLEVEL

Percentage of TX FIFO used.

Write	-
-------	---

Read	Decimal Number
Setting	No
Unit	Percent
Range	0 to 100

6.2.50. /DEVO...n/STATUS/ADC0MIN

The minimum value on Signal Input 1 (ADC0) during 100 ms.

Write	-
Read	Integer Number
Setting	No
Range	-127 to 127

6.2.51. /DEVO...n/STATUS/ADC0MAX

The maximum value on Signal Input 1 (ADC0) during 100 ms.

Write	-
Read	Integer Number
Setting	No
Range	-127 to 127

6.2.52. /DEVO...n/STATUS/ADC1MIN

The minimum value on Signal Input 2 (ADC1) during 100 ms.

Write	-
Read	Integer Number
Setting	No
Range	-127 to 127

6.2.53. /DEVO...n/STATUS/ADC1MAX

The maximum value on Signal Input 2 (ADC1) during 100 ms.

Write	-
Read	Integer Number
Setting	No
Range	-127 to 127

6.2.54. /DEVO...n/STATUS/ECHOWRITE

32bits written to this node will be echoed back via ECHOREAD node.

Write	Integer Number
Read	-
Setting	No

6.2.55. /DEV0...n/STATUS/ECHOREAD

32bits written to ECHOWRITE node are echoed here.

Write	-
Read	Integer Number
Setting	No

6.2.56. /DEV0...n/STATS

Nodes providing statistical data about the device.

6.2.57. /DEV0...n/STATS/BYTESSENT

Total amount of bytes sent via USB.

Write	-
Read	Integer Number
Setting	No
Unit	Bytes

6.2.58. /DEV0...n/STATS/BYTESRECEIVED

Total amount of bytes received via USB.

Write	-
Read	Integer Number
Setting	No
Unit	Bytes

6.2.59. /DEV0...n/STATS/MEANPOLLCNT

Average poll-count.

Write	-
Read	Decimal Number
Setting	No
Unit	Polls/Second

6.2.60. /DEV0...n/STATS/MEANMSGCNT

Average message-count.

Write	-
Read	Decimal Number
Setting	No
Unit	Messages/Second

6.2.61. /DEV0...n/STATS/PHYSICAL

Group of nodes providing some physical information on the device.

6.2.62. /DEV0...n/STATS/PHYSICAL/1V2

Actual voltage of the 1.2 Volts supply.

Write	-
Read	Decimal Number
Setting	No
Unit	Volts

6.2.63. /DEV0...n/STATS/PHYSICAL/1V8

Actual voltage of the 1.8 Volts supply.

Write	-
Read	Decimal Number
Setting	No
Unit	Volts

6.2.64. /DEV0...n/STATS/PHYSICAL/2V5

Actual voltage of the 2.5 Volts supply.

Write	-
Read	Decimal Number
Setting	No
Unit	Volts

6.2.65. /DEV0...n/STATS/PHYSICAL/3V3

Actual voltage of the 3.3 Volts supply.

Write	-
Read	Decimal Number
Setting	No
Unit	Volts

6.2.66. /DEV0...n/STATS/PHYSICAL/5V0

Actual voltage of the 5.0 Volts supply.

Write	-
Read	Decimal Number
Setting	No
Unit	Volts

6.2.67. /DEV0...n/STATS/PHYSICAL/TEMP

Actual temperature.

Write	-
Read	Decimal Number
Setting	No
Unit	Degrees Celsius

6.2.68. /DEV0...n/DEMODS

Demodulator nodes.

6.2.69. /DEV0...n/DEMODS/0...n

Nodes of a single demodulator

6.2.70. /DEV0...n/DEMODS/0...n/ADCSELECT

Selects the index of the signal input for the demodulator.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Index	
Range	0 to 5	
Values	0	Signal input 0
	1	Signal input 1
	2	Aux Input 0
	3	Aux Input 1
	4	DIO 0
	5	DIO 1

6.2.71. /DEV0...n/DEMODS/0...n/ORDER

Selects the order of the low-pass filter.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Order	
Default	4 (24 dB/oct slope)	
Range	1 to 8	
Values	1	6 dB/oct slope
	2	12 dB/oct slope
	3	18 dB/oct slope

4	24 dB/oct slope
5	30 dB/oct slope
6	36 dB/oct slope
7	42 dB/oct slope
8	48 dB/oct slope

6.2.72. /DEV0...n/DEMODS/0...n/TIMECONSTANT

Sets the time constant of the low-pass filter.

Write	Decimal Number
Read	Decimal Number
Setting	Yes
Unit	s
Default	0.010164

Details

The time constant is set for each stage of the low-pass filter. The total time constant and bandwidth depends on the selected order.

6.2.73. /DEV0...n/DEMODS/0...n/RATE

The number of output values sent to the computer per second.

Write	Decimal Number
Read	Decimal Number
Setting	Yes
Unit	Hz

6.2.74. /DEV0...n/DEMODS/0...n/ENABLE

Enables the demodulator data stream.

Write	Integer Number
Read	Integer Number
Setting	Yes
Unit	Boolean
Default	0 (Demodulator stream off)
Values	0 Demodulator stream off 1 Demodulator stream on

6.2.75. /DEV0...n/DEMODS/0...n/TRIGGER

Sets the trigger- and gating-functionality of the demodulator.

Write	Integer Number
Read	Integer Number

Setting	Yes																
Unit	bit-coded																
Values	<table> <tr> <td>b0 = 1</td><td>DIO0 rising edge</td></tr> <tr> <td>b1 = 2</td><td>DIO0 falling edge</td></tr> <tr> <td>b2 = 4</td><td>DIO1 rising edge</td></tr> <tr> <td>b3 = 8</td><td>DIO1 falling edge</td></tr> <tr> <td>b4 = 16</td><td>DIO0 high</td></tr> <tr> <td>b5 = 32</td><td>DIO0 low</td></tr> <tr> <td>b6 = 64</td><td>DIO1 high</td></tr> <tr> <td>b7 = 128</td><td>DIO1 low</td></tr> </table>	b0 = 1	DIO0 rising edge	b1 = 2	DIO0 falling edge	b2 = 4	DIO1 rising edge	b3 = 8	DIO1 falling edge	b4 = 16	DIO0 high	b5 = 32	DIO0 low	b6 = 64	DIO1 high	b7 = 128	DIO1 low
b0 = 1	DIO0 rising edge																
b1 = 2	DIO0 falling edge																
b2 = 4	DIO1 rising edge																
b3 = 8	DIO1 falling edge																
b4 = 16	DIO0 high																
b5 = 32	DIO0 low																
b6 = 64	DIO1 high																
b7 = 128	DIO1 low																

Details

The triggers are configured by the bits of an integer. When multiple bits/triggers are set, they are or-ed. If trigger is set to 0 then demodulator data is sent continuously.

6.2.76. /DEV0...n/DEMODS/0...n/OSCSELECT

Index of the oscillator used to demodulate the signal.

Write	Integer Number
Read	Integer Number
Setting	Yes
Unit	Index

6.2.77. /DEV0...n/DEMODS/0...n/HARMONIC

The harmonic of the base frequency to be used.

Write	Integer Number
Read	Integer Number
Setting	Yes
Unit	Harmonic
Default	1
Range	1 to 1023

Details

Selecting 1 chooses the fundamental frequency.

6.2.78. /DEV0...n/DEMODS/0...n/FREQ

Frequency to of the demodulator.

Write	-
Read	Decimal Number
Setting	No
Unit	Hz

Default	1.0E6
Range	0 to 1.0E8

6.2.79. /DEV0...n/DEM0DS/0...n/PHASESHIFT

The phase shift of the demodulator.

Write	Decimal Number
Read	Decimal Number
Setting	Yes
Unit	deg
Default	0
Range	-180 to 180

6.2.80. /DEV0...n/DEM0DS/0...n/SINC

Boolean value enabling Sinc filter functionality.

Write	Integer Number
Read	Integer Number
Setting	Yes
Unit	Boolean
Default	0 (Sinc filtering disabled)
Values	0 Sinc filtering disabled 1 Sinc filtering enabled

6.2.81. /DEV0...n/DEM0DS/0...n/SAMPLE

Samples of the demodulator are given out at this node.

Write	-
Read	-
Stream	-
Setting	No

6.2.82. /DEV0...n/OSCS

Oscillator nodes.

6.2.83. /DEVO...n/OSCS/0...n

Nodes of an oscillator.

6.2.84. /DEV0...n/OSCS/0...n/FREQ

Frequency to of the oscillator.

Write Decimal Number

Read	Decimal Number
Setting	Yes
Unit	Hz
Default	1.0E6
Range	0 to 1.0E8

6.2.85. /DEV0...n/MODS

Modulator option nodes.

Details

These nodes are only visible when the MOD option is installed on the device.

6.2.86. /DEV0...n/MODS/0...n

Nodes of a Modulator.

Note

These nodes are only available with installed Modulation-Option.

6.2.87. /DEV0...n/MODS/0...n/ENABLE

Enables the modulation.

Write	Integer Number				
Read	Integer Number				
Setting	Yes				
Unit	Boolean				
Default	0 (Modulation Off)				
Values	<table> <tr> <td>0</td> <td>Modulation Off</td> </tr> <tr> <td>1</td> <td>Modulation On</td> </tr> </table>	0	Modulation Off	1	Modulation On
0	Modulation Off				
1	Modulation On				

6.2.88. /DEV0...n/MODS/0...n/OUTPUT

Modulation output.

Write	Integer Number								
Read	Integer Number								
Setting	Yes								
Default	0 (none)								
Values	<table> <tr> <td>0</td> <td>none</td> </tr> <tr> <td>1</td> <td>1</td> </tr> <tr> <td>2</td> <td>2</td> </tr> <tr> <td>3</td> <td>1 and 2</td> </tr> </table>	0	none	1	1	2	2	3	1 and 2
0	none								
1	1								
2	2								
3	1 and 2								

6.2.89. /DEV0...n/MODS/0...n/MODE

Modulation mode.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Default	0 (Amplitude modulation)	
Values	0	Amplitude modulation
	1	Frequency modulation
	2	Manual

6.2.90. /DEV0...n/MODS/0...n/FREQDEVENABLE

In FM mode, enable peak deviation.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Boolean	
Default	0 (Peak deviation off)	
Values	0	Peak deviation off
	1	Peak deviation on

6.2.91. /DEV0...n/MODS/0...n/FREQDEV

In FM mode, set peak deviation value.

Write	Decimal Number	
Read	Decimal Number	
Setting	Yes	
Unit	V	

6.2.92. /DEV0...n/MODS/0...n/INDEX

In FM mode, set modulation index value. The modulation index equals peak deviation divided by modulation frequency.

Write	Decimal Number	
Read	Decimal Number	
Setting	Yes	
Unit		

6.2.93. /DEV0...n/MODS/0...n/RATE

The number of output values sent to the computer per second.

Write	Decimal Number
-------	----------------

Read	Decimal Number
Setting	Yes
Unit	Hz

6.2.94. /DEV0...n/MODS/0...n/trigger

Sets the trigger- and gating-functionality of the demodulator.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	bit-coded	
Values	b0 = 1	DIO0 rising edge
	b1 = 2	DIO0 falling edge
	b2 = 4	DIO1 rising edge
	b3 = 8	DIO1 falling edge
	b4 = 16	DIO0 high
	b5 = 32	DIO0 low
	b6 = 64	DIO1 high
	b7 = 128	DIO1 low

6.2.95. /DEV0...n/MODS/0...n/cARRIER

Group of carrier nodes.

6.2.96. /DEV0...n/MODS/0...n/cARRIER/INPUTSELECT

Signal Input for the carrier demodulation.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Default	0 (Sig In 1)	
Values	0	Sig In 1
	1	Sig In 2

6.2.97. /DEV0...n/MODS/0...n/cARRIER/ORDER

Filter order for carrier demodulation.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Order	
Default	4 (24 dB/oct slope)	
Range	1 to 8	

Values	1	6 dB/oct slope
	2	12 dB/oct slope
	3	18 dB/oct slope
	4	24 dB/oct slope
	5	30 dB/oct slope
	6	36 dB/oct slope
	7	42 dB/oct slope
	8	48 dB/oct slope

6.2.98. /DEV0...n/MODS/0...n/CARRIER/TIMECONSTANT

Sets the time constant of the carrier low-pass filter.

Write	Decimal Number
Read	Decimal Number
Setting	Yes
Unit	s

Details

The time constant is set for each stage of the low-pass filter. The total time constant and bandwidth depends on the selected order.

6.2.99. /DEV0...n/MODS/0...n/CARRIER/OSCSELECT

Index of the oscillator used to demodulate the signal.

Write	Integer Number
Read	Integer Number
Setting	Yes
Unit	Index

6.2.100. /DEV0...n/MODS/0...n/CARRIER/HARMONIC

Harmonic of the carrier frequency.

Write	Integer Number
Read	Integer Number
Setting	Yes
Unit	Harmonic
Default	1
Range	1 to 1023

Details

Selecting 1 chooses the fundamental frequency.

6.2.101. /DEV0...n/MODS/0...n/CARRIER/PHASESHIFT

The phase shift of the carrier demodulator.

Write	Decimal Number
Read	Decimal Number
Setting	Yes
Unit	deg
Default	0
Range	-180 to 180

6.2.102. /DEV0...n/MODS/0...n/CARRIER/ENABLE

Enables the carrier data stream.

Write	Integer Number
Read	Integer Number
Setting	Yes
Unit	Boolean
Default	0 (Demodulator stream off)
Values	0 Demodulator stream off 1 Demodulator stream on

6.2.103. /DEV0...n/MODS/0...n/CARRIER/AMPLITUDE

Carrier amplitude.

Write	Decimal Number
Read	Decimal Number
Setting	Yes
Unit	Gain
Range	-1 to 1

Details

Multiply this value with the range setting to obtain voltage in V.

6.2.104. /DEV0...n/MODS/0...n/SIDEBANDS

Group of sideband nodes.

6.2.105. /DEV0...n/MODS/0...n/SIDEBANDS/0...n

6.2.106. /DEV0...n/MODS/0...n/SIDEBANDS/0...n/MODE

Sideband selector.

Write	Integer Number
Read	Integer Number
Setting	Yes
Default	0 (Off)

Values	0	Off
	1	C + M
	2	C - M

6.2.107. /DEV0...n/MODS/0...n/SIDEBANDS/0...n/INPUTSELECT

Signal Input for the sideband demodulation.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Default	0 (Sig In 1)	
Values	0	Sig In 1
	1	Sig In 2

6.2.108. /DEV0...n/MODS/0...n/SIDEBANDS/0...n/ORDER

Filter order for sideband demodulation.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Order	
Default	4 (24 dB/oct slope)	
Range	1 to 8	
Values	1	6 dB/oct slope
	2	12 dB/oct slope
	3	18 dB/oct slope
	4	24 dB/oct slope
	5	30 dB/oct slope
	6	36 dB/oct slope
	7	42 dB/oct slope
	8	48 dB/oct slope

6.2.109. /DEV0...n/MODS/0...n/SIDEBANDS/0...n/TIMECONSTANT

Sets the time constant of the sideband low-pass filter.

Write	Decimal Number	
Read	Decimal Number	
Setting	Yes	
Unit	s	

Details

The time constant is set for each stage of the low-pass filter. The total time constant and bandwidth depends on the selected order.

6.2.110. /DEV0...n/MODS/0...n/SIDEBOARDS/0...n/ OSCSELECT

Index of the oscillator used to demodulate the signal.

Write	Integer Number
Read	Integer Number
Setting	Yes
Unit	Index

6.2.111. /DEV0...n/MODS/0...n/SIDEBOARDS/0...n/ HARMONIC

Harmonic of the sideband frequency.

Write	Integer Number
Read	Integer Number
Setting	Yes
Unit	Harmonic
Default	1
Range	1 to 1023

Details

Selecting 1 chooses the fundamental frequency.

6.2.112. /DEV0...n/MODS/0...n/SIDEBOARDS/0...n/ PHASESHIFT

The phase shift of the sideband demodulator.

Write	Decimal Number
Read	Decimal Number
Setting	Yes
Unit	deg
Default	0
Range	-180 to 180

6.2.113. /DEV0...n/MODS/0...n/SIDEBOARDS/0...n/ENABLE

Enables the sideband data stream.

Write	Integer Number
Read	Integer Number

Setting	Yes
Unit	Boolean
Default	0 (Demodulator stream off)
Values	0 Demodulator stream off 1 Demodulator stream on

6.2.114. /DEV0...n/MODS/0...n/SIDEBOARDS/0...n/AMPLITUDE

Sideband amplitude

Write	Decimal Number
Read	Decimal Number
Setting	Yes
Unit	Gain
Range	-1 to 1

Details

Multiply this value with the range setting to obtain voltage in V.

6.2.115. /DEV0...n/MODS/0...n/SAMPLE

Modulation Samples.

Write	-
Read	-
Stream	-
Setting	No

6.2.116. /DEV0...n/PLLS

PLL nodes.

6.2.117. /DEV0...n/PLLS/0...n

Nodes of a PLL.

Note

These nodes are only available with installed PLL option.

6.2.118. /DEV0...n/PLLS/0...n/ADCSELECT

Selects an input for the PLL.

Write	Integer Number
-------	----------------

Read	Integer Number	
Setting	Yes	
Unit	Index	
Range	0 to 5	
Values	0	Signal Input 1
	1	Signal Input 2
	2	Aux Input 1
	3	Aux Input 2
	4	DIO 0
	5	DIO 1

6.2.119. /DEV0...n/PLLS/0...n/AUTOCENTER

Switches auto-center.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Boolean	
Default	0 (Auto-center off)	
Values	0	Auto-center off
	1	Auto-center on

6.2.120. /DEV0...n/PLLS/0...n/FREQCENTER

Selects a center frequency.

Write	Decimal Number	
Read	Decimal Number	
Setting	Yes	
Unit	Hz	
Range	0 to 1.0E8	

6.2.121. /DEV0...n/PLLS/0...n/FREQRANGE

Selects a frequency range for the PLL.

Write	Decimal Number	
Read	Decimal Number	
Setting	Yes	
Unit	Hz	
Range	0 to 1.0E8	

6.2.122. /DEV0...n/PLLS/0...n/ENABLE

Enables the PLL.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Boolean	
Default	0 (PLL off)	
Values	0	PLL off
	1	PLL on

6.2.123. /DEV0...n/PLLS/0...n/ERROR

Error of the PLL.

Write	-
Read	Decimal Number
Stream	Decimal Number
Setting	No
Unit	deg
Range	-180 to 180

6.2.124. /DEV0...n/PLLS/0...n/AUTOTIMECONSTANT

Switches external time constant control.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Boolean	
Default	0 (External time constant off)	
Values	0	External time constant off
	1	External time constant on

6.2.125. /DEV0...n/PLLS/0...n/TIMECONSTANT

The external time constant.

Write	Decimal Number	
Read	Decimal Number	
Setting	Yes	
Unit	s	
Default	0	

6.2.126. /DEV0...n/PLLS/0...n/AUTOPID

Switches external PID.

Write	Integer Number	

Read	Integer Number	
Setting	Yes	
Unit	Boolean	
Default	0 (External PID off)	
Values	0	External PID off
	1	External PID on

6.2.127. /DEV0...n/PLLS/0...n/P

Proportional gain of the PID controller.

Write	Decimal Number	
Read	Decimal Number	
Setting	Yes	
Unit	Hz/deg	
Default	0	
Range	-36458.33 to 36458.33	

6.2.128. /DEV0...n/PLLS/0...n/I

Integral gain of the PID controller.

Write	Decimal Number	
Read	Decimal Number	
Setting	Yes	
Unit	Hz^2/deg	
Default	0	
Range	-8.1982529E6 to 8.19819035E6	

6.2.129. /DEV0...n/PLLS/0...n/D

Derivative gain of the PID.

Write	Decimal Number	
Read	Decimal Number	
Setting	Yes	
Unit	1/deg	
Default	0	
Range	-0.00002 to 0.00002	

6.2.130. /DEV0...n/PLLS/0...n/FREQDELTA

Frequency deviation from center frequency.

Write	-
Read	Decimal Number

Setting	No
Unit	Hz

6.2.131. /DEV0...n/PLLS/0...n/ADCTHRESHOLD

Threshold for edge detection.

Write	Integer Number
Read	Integer Number
Setting	Yes
Default	100
Range	-4096 to 4095

Details

Full scale corresponds to -4096 and 4095.

6.2.132. /DEV0...n/PLLS/0...n/AUXAVG

Delta frequency averaging control.

Write	Integer Number
Read	Integer Number
Setting	Yes
Default	8
Range	0 to 128

6.2.133. /DEV0...n/PLLS/0...n/SETPOINT

The setpoint in degrees of the PLL.

Write	Decimal Number
Read	Decimal Number
Setting	Yes
Unit	deg
Default	0
Range	-180 to 180

6.2.134. /DEV0...n/PLLS/0...n/HARMONIC

The harmonic of the base frequency to be used.

Write	Integer Number
Read	Integer Number
Setting	Yes
Unit	Harmonic
Default	1
Range	1 to 1023

Details

Selecting 1 chooses the fundamental frequency.

6.2.135. /DEV0...n/PLLS/0...n/ORDER

Selects the order of the low-pass filter.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Order	
Default	4 (24 dB/oct slope)	
Range	1 to 8	
Values	1	6 dB/oct slope
	2	12 dB/oct slope
	3	18 dB/oct slope
	4	24 dB/oct slope
	5	30 dB/oct slope
	6	36 dB/oct slope
	7	42 dB/oct slope
	8	48 dB/oct slope

6.2.136. /DEV0...n/PLLS/0...n/RATE

Update rate information.

Write	-
Read	Decimal Number
Setting	No
Unit	Samples/s

6.2.137. /DEV0...n/PLLS/0...n/OSCSELECT

Index of the oscillator used.

Write	-
Read	Integer Number
Setting	No
Unit	Index

6.2.138. /DEV0...n/PLLS/0...n/DEMODSELECT

Source demodulator.

Write	-
Read	Integer Number
Setting	No

6.2.139. /DEV0...n/PIDS

PID nodes.

6.2.140. /DEV0...n/PIDS/0...n

Nodes of a PID.

Note

These nodes are only available with installed PID option.

6.2.141. /DEV0...n/PIDS/0...n/INPUT

Selects the input for the PID.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Default	0 (Demodulator X value [Vrms])	
Values	0	Demodulator X value [Vrms]
	1	Demodulator Y value [Vrms]
	2	Demodulator R value [Vrms]
	3	Demodulator Theta value [deg]
	4	Auxiliary Input [V]
	5	Auxiliary Output (as input) [V]
	6	Modulation Index [0,1]
	7	Dual Frequency Tracking $ Z(n+1) - Z(n) $ [Vrms]
	8	Demodulator $x(n+1) - x(n)$ [Vrms]
	9	Demodulator $ z(n+1) - z(n) $ [Vrms]
	10	Oscillator Frequency [Hz]

6.2.142. /DEV0...n/PIDS/0...n/INPUTCHANNEL

If applicable, selects the channel of the selected INPUT.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Default	0	

Details

Sets the input channel index for the selected INPUT, i.e., 0,1,2 etc.. The available channels depend on the input type.

6.2.143. /DEV0...n/PIDS/0...n/OUTPUT

Selects the output for the PID.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Default	0 (Signal Output 1 [Vrms])	
Values	0	Signal Output 1 [Vrms]
	1	Signal Output 2 [Vrms]
	2	Oscillator frequency [Hz]
	3	Auxiliary Output (manual mode) [V]
	4	DIO [5 Volt TTL]

6.2.144. /DEV0...n/PIDS/0...n/OUTPUTCHANNEL

If applicable, selects the channel of the selected OUTPUT.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Default	6	

Details

Sets the input channel index for the selected OUTPUT, i.e., 0,1,2 etc.. The available channels depend on the output type

6.2.145. /DEV0...n/PIDS/0...n/OUTPUTDEFAULTENABLE

If OUTPUTDEFAULTENABLE is set, the value specified by OUTPUTDEFAULT will be applied when the PID is switched off.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Boolean	
Default	No	
Values	0	OFF
	1	ON

Details

6.2.146. /DEV0...n/PIDS/0...n/OUTPUTDEFAULT

If OUTPUTDEFAULTENABLE is set, this node specifies the value to be applied.

Write	Decimal Number	
Read	Decimal Number	

Setting	Yes
Default	0.1

Details**6.2.147. /DEV0...n/PIDS/0...n/P**

Proportional gain.

Write	Decimal Number
Read	Decimal Number
Setting	Yes
Unit	[OUTPUT Unit]/[INPUT Unit]
Default	1

Details

Sets the proportional gain for the error signal. Negative feedback corresponds to a negative gain.

6.2.148. /DEV0...n/PIDS/0...n/I

Proportional gain for integrator.

Write	Decimal Number
Read	Decimal Number
Setting	Yes
Unit	[OUTPUT Unit]/[INPUT Unit]/s
Default	10

Details

Sets the proportional gain for the integrated (accumulated) error signal. Negative feedback corresponds to a negative gain.

6.2.149. /DEV0...n/PIDS/0...n/D

Proportional gain for differentiator.

Write	Decimal Number
Read	Decimal Number
Setting	Yes
Unit	[OUTPUT Unit]/[INPUT Unit]*s
Default	0.001

Details

Sets the proportional gain for the differentiated error signal. Negative feedback corresponds to a negative gain.

6.2.150. /DEV0...n/PIDS/0...n/SETPOINT

Target settle point.

Write	Decimal Number
Read	Decimal Number
Setting	Yes
Unit	[INPUT Unit]
Default	0.1

6.2.151. /DEV0...n/PIDS/0...n/SETPOINTSELECT

Set point selection.

Write	Integer Number								
Read	Integer Number								
Setting	Yes								
Default	0 (Manual Setpoint)								
Values	<table> <tr> <td>0</td> <td>Manual Setpoint</td> </tr> <tr> <td>1</td> <td>Auxiliary Input 1</td> </tr> <tr> <td>2</td> <td>Auxiliary Input 2</td> </tr> <tr> <td>3</td> <td>PID n</td> </tr> </table>	0	Manual Setpoint	1	Auxiliary Input 1	2	Auxiliary Input 2	3	PID n
0	Manual Setpoint								
1	Auxiliary Input 1								
2	Auxiliary Input 2								
3	PID n								

6.2.152. /DEV0...n/PIDS/0...n/MONITOROFFSET

Offset for the monitor output.

Write	Decimal Number
Read	Decimal Number
Setting	Yes
Default	0

6.2.153. /DEV0...n/PIDS/0...n/MONITORSCALE

Scale for the monitor output.

Write	Decimal Number
Read	Decimal Number
Setting	Yes
Default	1

6.2.154. /DEV0...n/PIDS/0...n/ERROR

Shows the error value.

Write	-
Read	Decimal Number
Setting	No
Unit	[OUTPUT Unit]

Details

The calculated error is : $\text{ERROR} = \text{SETPOINT} - \text{IN}$.

6.2.155. /DEV0...n/PIDS/0...n/CENTER

Sets the output center point.

Write	Decimal Number
Read	Decimal Number
Setting	Yes
Unit	[OUTPUT Unit]
Default	0.1

6.2.156. /DEV0...n/PIDS/0...n/RANGE

Sets the output range.

Write	Decimal Number
Read	Decimal Number
Setting	Yes
Unit	[OUTPUT Unit]
Default	0.5

Details

The limits for the output are : $\text{OUT} = [\text{CENTER} - \text{RANGE}, \text{CENTER} + \text{RANGE}]$ with $\text{RANGE} > 0.0$.

6.2.157. /DEV0...n/PIDS/0...n/SHIFT

Shows the output shift.

Write	-
Read	Decimal Number
Setting	No
Unit	[OUTPUT Unit]
Default	0

Details

The calculated output value is : $\text{OUT} = \text{CENTER} + \text{SHIFT}$.

6.2.158. /DEV0...n/PIDS/0...n/ENABLE

Enable PID controller.

Write	Integer Number
Read	Integer Number
Setting	Yes
Unit	Boolean
Default	0 (OFF)

Values	0	OFF
	1	ON

6.2.159. /DEV0...n/PIDS/0...n/RATE

Control update rate.

Write	-
Read	Decimal Number
Setting	No
Unit	Samples/s

6.2.160. /DEV0...n/PIDS/0...n/TIPPROTECT

Contains nodes for configuring the TipProtect functionality.

6.2.161. /DEV0...n/PIDS/0...n/TIPPROTECT/ENABLE

Enable TipProtect for the PID controller.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Boolean	
Default	0 (OFF)	
Values	0	OFF
	1	ON

6.2.162. /DEV0...n/PIDS/0...n/TIPPROTECT/PLL

Selects a PLL for TipProtect.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Default	0	

6.2.163. /DEV0...n/PIDS/0...n/TIPPROTECT/ACTION

Indicates whether TipProtect is active.

Write	-
Read	Integer Number
Setting	No
Unit	Boolean
Default	0

6.2.164. /DEV0...n/PIDS/0...n/TIPPROTECT/ ACTIVETIMECONSTANT

Time constant when TipProtect is active.

Write	Decimal Number
Read	Decimal Number
Setting	Yes
Unit	s
Default	0
Range	0 to 1.0E8

Details

Time constant for low-pass filtering the PLL error² when TipProtect is active, i.e., when waiting to re-enable the PID controller.

6.2.165. /DEV0...n/PIDS/0...n/TIPPROTECT/ ACTIVETHRESHOLD

Threshold for the active state.

Write	Decimal Number
Read	Decimal Number
Setting	Yes
Unit	deg ²
Range	0 to 129600

Details

Threshold for PLL error² when TipProtect is active, i.e., when waiting to re-enable the PID controller.

6.2.166. /DEV0...n/PIDS/0...n/TIPPROTECT/ INACTIVETIMECONSTANT

Time constant when TipProtect is inactive.

Write	Decimal Number
Read	Decimal Number
Setting	Yes
Unit	s
Default	0
Range	0 to 1.0E8

Details

Time constant for low-pass filtering the PLL error² when TipProtect is inactive, i.e., when waiting to disable the PID controller.

6.2.167. /DEV0...n/PIDS/0...n/TIPPROTECT/ INACTIVETHRESHOLD

Threshold for the inactive state.

Write	Decimal Number
Read	Decimal Number
Setting	Yes
Unit	deg ²
Range	0 to 129600

Details

Threshold for PLL error² when TipProtect is inactive, i.e., when waiting to disable the PID controller.

6.2.168. /DEV0...n/IMPS

Impedance analyzer nodes.

6.2.169. /DEV0...n/IMPS/0...n

Nodes of a impedance analyzer.

6.2.170. /DEV0...n/IMPS/0...n/ENABLE

Enables the impedance data stream.

Write	Integer Number				
Read	Integer Number				
Setting	Yes				
Unit	Boolean				
Default	0 (Impedance stream off)				
Values	<table> <tr> <td>0</td> <td>Impedance stream off</td> </tr> <tr> <td>1</td> <td>Impedance stream on</td> </tr> </table>	0	Impedance stream off	1	Impedance stream on
0	Impedance stream off				
1	Impedance stream on				

6.2.171. /DEV0...n/IMPS/0...n/MODEL

Impedance model select.

Write	Integer Number						
Read	Integer Number						
Setting	Yes						
Unit	Index						
Range	0 to 4						
Values	<table> <tr> <td>0</td> <td>R parallel C</td> </tr> <tr> <td>1</td> <td>R+C</td> </tr> <tr> <td>2</td> <td>R+L</td> </tr> </table>	0	R parallel C	1	R+C	2	R+L
0	R parallel C						
1	R+C						
2	R+L						

	3	GB
	4	D+C

6.2.172. /DEV0...n/IMPS/0...n/FREQ

Control frequency to of the oscillator.

Write	Decimal Number
Read	Decimal Number
Setting	Yes
Unit	Hz
Default	1.0E6
Range	0 to 1.0E8

6.2.173. /DEV0...n/IMPS/0...n/CURRENT

Nodes of a impedance current.

6.2.174. /DEV0...n/IMPS/0...n/CURRENT/INPUTSELECT

Selects the index of the signal input for the impedance current demodulator.

Write	Integer Number				
Read	Integer Number				
Setting	Yes				
Unit	Index				
Range	0 to 1				
Values	<table> <tr> <td>0</td> <td>Signal input 0</td> </tr> <tr> <td>1</td> <td>Signal input 1</td> </tr> </table>	0	Signal input 0	1	Signal input 1
0	Signal input 0				
1	Signal input 1				

6.2.175. /DEV0...n/IMPS/0...n/CURRENT/DEMODOSELECT

Index of the demodulator for the current impedance signal.

Write	-
Read	-
Setting	Yes
Unit	Index
Default	1

6.2.176. /DEV0...n/IMPS/0...n/CURRENT/INVERT

Invert the current signal of the impedance.

Write	Integer Number
Read	Integer Number
Setting	Yes

Unit	Boolean				
Default	0 (Inversion off)				
Values	<table> <tr> <td>0</td> <td>Inversion off</td> </tr> <tr> <td>1</td> <td>Inversion on</td> </tr> </table>	0	Inversion off	1	Inversion on
0	Inversion off				
1	Inversion on				

6.2.177. /DEV0...n/IMPS/0...n/CURRENT/RANGE

Current input range value in manual mode.

Write	Decimal Number
Read	Decimal Number
Setting	Yes

6.2.178. /DEV0...n/IMPS/0...n/VOLTAGE

Nodes of a impedance voltage.

6.2.179. /DEV0...n/IMPS/0...n/VOLTAGE/INPUTSELECT

Selects the index of the signal input for the impedance voltage demodulator.

Write	Integer Number				
Read	Integer Number				
Setting	Yes				
Unit	Index				
Range	0 to 1				
Values	<table> <tr> <td>0</td> <td>Signal input 0</td> </tr> <tr> <td>1</td> <td>Signal input 1</td> </tr> </table>	0	Signal input 0	1	Signal input 1
0	Signal input 0				
1	Signal input 1				

6.2.180. /DEV0...n/IMPS/0...n/VOLTAGE/DEMODOSELECT

Index of the demodulator for the voltage impedance signal.

Write	-
Read	-
Setting	Yes
Unit	Index
Default	0

6.2.181. /DEV0...n/IMPS/0...n/VOLTAGE/INVERT

Invert the voltage signal of the impedance.

Write	Integer Number
Read	Integer Number
Setting	Yes

Unit	Boolean	
Default	0 (Inversion off)	
Values	0	Inversion off
	1	Inversion on

6.2.182. /DEV0...n/IMPS/0...n/VOLTAGE/RANGE

Voltage input range value in manual mode.

Write	Decimal Number	
Read	Decimal Number	
Setting	Yes	

6.2.183. /DEV0...n/IMPS/0...n/AC

Control input AC coupling.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Boolean	
Default	0 (Input AC coupling off)	
Values	0	Input AC coupling off
	1	Input AC coupling on

6.2.184. /DEV0...n/IMPS/0...n/DEMOD

Nodes of a impedance demodulator settings.

6.2.185. /DEV0...n/IMPS/0...n/DEMOD/OSCSELECT

Index of the oscillator used to demodulate the signal.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Index	

6.2.186. /DEV0...n/IMPS/0...n/DEMOD/HARMONIC

The harmonic of the base frequency to be used.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Harmonic	
Default	1	

Range	1 to 1023
-------	-----------

Details

Selecting 1 chooses the fundamental frequency.

6.2.187. /DEV0...n/IMPS/0...n/DEMOD/ORDER

Selects the order of the low-pass filter.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Order	
Default	4 (24 dB/oct slope)	
Range	1 to 8	
Values	1	6 dB/oct slope
	2	12 dB/oct slope
	3	18 dB/oct slope
	4	24 dB/oct slope
	5	30 dB/oct slope
	6	36 dB/oct slope
	7	42 dB/oct slope
	8	48 dB/oct slope

6.2.188. /DEV0...n/IMPS/0...n/DEMOD/TIMECONSTANT

Sets the time constant of the low-pass filter.

Write	Decimal Number	
Read	Decimal Number	
Setting	Yes	
Unit	s	
Default	0.010164	

Details

The time constant is set for each stage of the low-pass filter. The total time constant and bandwidth depends on the selected order.

6.2.189. /DEV0...n/IMPS/0...n/DEMOD/SINC

Boolean value enabling Sinc filter functionality.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Boolean	
Default	0 (Sinc filtering disabled)	

Values	0	Sinc filtering disabled
	1	Sinc filtering enabled

6.2.190. /DEV0...n/IMPS/0...n/DEMOD/RATE

The number of output values sent to the computer per second.

Write	Decimal Number
Read	Decimal Number
Setting	Yes
Unit	Hz

6.2.191. /DEV0...n/IMPS/0...n/FILTER

Switches between application modes and advanced mode.

Write	Integer Number
Read	Integer Number
Setting	Yes
Unit	Index
Range	0 to 1
Values	0 Application Mode 1 Advanced Mode

6.2.192. /DEV0...n/IMPS/0...n/APPLICATION

Selection of the impedance application.

Write	Integer Number
Read	Integer Number
Setting	Yes
Unit	Index
Range	0 to 0
Values	0 Any

6.2.193. /DEV0...n/IMPS/0...n/INTERPOLATION

Selection of the interpolation method.

Write	Integer Number
Read	Integer Number
Setting	Yes
Unit	Index
Range	0 to 1
Values	0 Linear

	1	Piecewise Cubic Hermite (PCHIP)
--	---	---------------------------------

6.2.194. /DEV0...n/IMPS/0...n/IMPEDANCERANGE

Selection of the impedance range.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Range	0 to 8	
Values	0	100
	1	1k
	2	10k
	3	100k
	4	1M
	5	10M
	6	100M
	7	1G
	8	10G

6.2.195. /DEV0...n/IMPS/0...n/MAXBANDWIDTH

Maximum bandwidth value.

Write	Decimal Number	
Read	Decimal Number	
Setting	Yes	
Unit	Hz	

6.2.196. /DEV0...n/IMPS/0...n/OMEGASUPPRESSION

Omega suppression value.

Write	Decimal Number	
Read	Decimal Number	
Setting	Yes	
Unit	dB	

6.2.197. /DEV0...n/IMPS/0...n/VALUE

Bias value.

Write	Decimal Number	
Read	Decimal Number	
Setting	Yes	
Unit	V	

6.2.198. /DEV0...n/IMPS/0...n/BIAS

Nodes of a impedance bias.

6.2.199. /DEV0...n/IMPS/0...n/BIAS/ENABLE

Enable bias.

Write	Integer Number
Read	Integer Number
Setting	Yes
Unit	Boolean
Default	0 (Bias off)
Values	0 Bias off 1 Bias on

6.2.200. /DEV0...n/IMPS/0...n/BIAS/VALUE

Bias value.

Write	Decimal Number
Read	Decimal Number
Setting	Yes
Unit	V

6.2.201. /DEV0...n/IMPS/0...n/AUTO

Nodes of a impedance auto.

6.2.202. /DEV0...n/IMPS/0...n/AUTO/BW

Auto bandwidth.

Write	Integer Number
Read	Integer Number
Setting	Yes
Unit	Boolean
Default	0 (Auto BW off)
Values	0 Auto BW off 1 Auto BW on

6.2.203. /DEV0...n/IMPS/0...n/AUTO/INPUTRANGE

Impedance input range select.

Write	Integer Number
Read	Integer Number

Setting	Yes
Default	1 (Auto)
Values	0 Manual 1 Auto 2 Impedance

6.2.204. /DEV0...n/IMPS/0...n/AUTO/OUTPUT

Automatic output amplitude control.

Write	Integer Number
Read	Integer Number
Setting	Yes
Unit	Boolean
Default	0 (Auto output off)
Values	0 Auto output off 1 Auto output on

6.2.205. /DEV0...n/IMPS/0...n/AUTO/SUPPRESS

If set to one suppress any impedance auto functionality the next 30s.

Write	Integer Number
Read	Integer Number
Setting	No
Unit	Boolean
Default	0 (No suppress pending)
Values	0 No suppress pending 1 Suppress auto the next 30s

6.2.206. /DEV0...n/IMPS/0...n/CALIB

Nodes of a impedance calibration.

6.2.207. /DEV0...n/IMPS/0...n/CALIB/CABLELENGTH

Impedance calibration cable length compensation select.

Write	Integer Number
Read	Integer Number
Setting	Yes
Default	0 (Off)
Values	0 Off 1 1m 2 2m

3

3m

6.2.208. /DEV0...n/IMPS/0...n/CALIB/USER

Nodes of a user impedance calibration.

6.2.209. /DEV0...n/IMPS/0...n/CALIB/USER/ENABLE

Enable user calibration.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Boolean	
Default	0 (User calibration off)	
Values	0	User calibration off
	1	User calibration on

6.2.210. /DEV0...n/IMPS/0...n/CALIB/USER/VALID

Indicator that user calibration data is valid.

Write	-
Read	Integer Number
Setting	No
Unit	Boolean
Default	0 (User Calibration data is invalid)
Values	0 User Calibration data is invalid
	1 User Calibration data is valid

6.2.211. /DEV0...n/IMPS/0...n/CALIB/USER/ACTIVE

Indicator that user calibration is active.

Write	-
Read	Integer Number
Setting	No
Unit	Boolean
Default	0 (User calibration is inactive)
Values	0 User calibration is inactive
	1 User calibration is active

6.2.212. /DEV0...n/IMPS/0...n/CALIB/USER/INFO

Message string reporting calibration status, warnings, or errors.

Write	-
-------	---

Read	-
Setting	No

6.2.213. /DEV0...n/IMPS/0...n/CALIB/USER/STORE

Store user calibration.

Write	Integer Number	
Read	Integer Number	
Setting	No	
Unit	Boolean	
Default	0 (No action)	
Values	0	No action
	1	Store user calibration

6.2.214. /DEV0...n/IMPS/0...n/CALIB/USER/DATA

Calibration data as JSON string.

Write	-
Read	-
Setting	No

6.2.215. /DEV0...n/IMPS/0...n/CALIB/INTERNAL

Nodes of a internal impedance calibration.

6.2.216. /DEV0...n/IMPS/0...n/CALIB/INTERNAL/ENABLE

Enable internal calibration.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Boolean	
Default	0 (Internal calibration off)	
Values	0	Internal calibration off
	1	Internal calibration on

6.2.217. /DEV0...n/IMPS/0...n/CALIB/INTERNAL/VALID

Indicator that internal calibration data is valid.

Write	-
Read	Integer Number
Setting	No
Unit	Boolean

Default	0 (Internal Calibration data is invalid)	
Values	0	Internal Calibration data is invalid
	1	Internal Calibration data is valid

6.2.218. /DEV0...n/IMPS/0...n/CALIB/INTERNAL/STORE

Store internal calibration.

Write	Integer Number	
Read	Integer Number	
Setting	No	
Unit	Boolean	
Default	0 (No action)	
Values	0	No action
	1	Store internal calibration

6.2.219. /DEV0...n/IMPS/0...n/CALIB/INTERNAL/DATA

Calibration data as JSON string.

Write	-
Read	-
Setting	No

6.2.220. /DEV0...n/IMPS/0...n/CALIB/INTERNAL/INFO

Info string describing internal calibration data.

Write	-
Read	-
Setting	No

6.2.221. /DEV0...n/IMPS/0...n/CALIB/INTERNAL/ACTIVE

Indicator that internal calibration is active.

Write	-	
Read	Integer Number	
Setting	No	
Unit	Boolean	
Default	0 (Internal calibration is inactive)	
Values	0	Internal calibration is inactive
	1	Internal calibration is active

6.2.222. /DEV0...n/IMPS/0...n/CONFIDENCE

Nodes for impedance confidance indicators.

6.2.223. /DEV0...n/IMPS/0...n/CONFIDENCE/SUPPRESSION

Nodes for impedance parameter suppression.

6.2.224. /DEV0...n/IMPS/0...n/CONFIDENCE/SUPPRESSION/ENABLE

Enable suppression analysis.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Boolean	
Default	0 (Suppression analysis off)	
Values	0	Suppression analysis off
	1	Suppression analysis on

6.2.225. /DEV0...n/IMPS/0...n/CONFIDENCE/SUPPRESSION/RATIO

Suppression ratio.

Write	Decimal Number	
Read	Decimal Number	
Setting	Yes	
Unit	Ratio	
Default	10	

Details

Suppressing of the inferior model parameter if the suppression ratio to the dominant model parameter is higher than the given ratio.

6.2.226. /DEV0...n/IMPS/0...n/CONFIDENCE/OPENDETECT

Nodes for impedance open detect.

6.2.227. /DEV0...n/IMPS/0...n/CONFIDENCE/OPENDETECT/ENABLE

Enable open detect analysis.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Boolean	
Default	0 (Open detect analysis off)	

Values	0	Open detect analysis off
	1	Open detect analysis on

6.2.228. /DEV0...n/IMPS/0...n/CONFIDENCE/OPENDETECT/RATIO

Open detect ratio.

Write	Decimal Number
Read	Decimal Number
Setting	Yes
Unit	Ratio
Default	1

6.2.229. /DEV0...n/IMPS/0...n/CONFIDENCE/UNDERFLOW

Nodes for impedance underflow detect.

6.2.230. /DEV0...n/IMPS/0...n/CONFIDENCE/UNDERFLOW/ENABLE

Enable underflow analysis.

Write	Integer Number
Read	Integer Number
Setting	Yes
Unit	Boolean
Default	0 (Underflow analysis off)
Values	0 Underflow analysis off 1 Underflow analysis on

6.2.231. /DEV0...n/IMPS/0...n/CONFIDENCE/UNDERFLOW/RATIO

Underflow ratio.

Write	Decimal Number
Read	Decimal Number
Setting	Yes
Unit	Ratio
Default	1

6.2.232. /DEV0...n/IMPS/0...n/CONFIDENCE/COMPENSATION

Nodes for strong compensation detection.

6.2.233. /DEV0...n/IMPS/0...n/CONFIDENCE/COMPENSATION/ENABLE

Enable compensation analysis.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Boolean	
Default	0 (Compensation analysis off)	
Values	0	Compensation analysis off
	1	Compensation analysis on

6.2.234. /DEV0...n/IMPS/0...n/CONFIDENCE/COMPENSATION/RATIO

Compensation ratio.

Write	Decimal Number	
Read	Decimal Number	
Setting	Yes	
Unit	Ratio	
Default	1	

6.2.235. /DEV0...n/IMPS/0...n/MODE

Impedance calibration mode select.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Default	0 (2-Point)	
Values	0	2-Point
	1	4-Point

6.2.236. /DEV0...n/IMPS/0...n/OUTPUT

Nodes of a impedance output.

6.2.237. /DEV0...n/IMPS/0...n/OUTPUT/ON

Switches the output on and off.

Write	Integer Number	
Read	Integer Number	

Setting	Yes
Unit	Boolean
Default	0 (Output off)
Values	0 Output off 1 Output on

6.2.238. /DEV0...n/IMPS/0...n/OUTPUT/RANGE

Selects the output range for the Signal Output.

Write	Decimal Number
Read	Decimal Number
Setting	Yes
Unit	V
Default	1
Values	0.01 0.01 V range 0.1 0.1 V range 1 1 V range 10 10 V range

6.2.239. /DEV0...n/IMPS/0...n/OUTPUT/DEMOD

Index of the output demodulator.

Write	-
Read	-
Setting	Yes
Unit	Index
Default	0

6.2.240. /DEV0...n/IMPS/0...n/OUTPUT/SELECT

Selects the output channel used for impedance measurement.

Write	Integer Number
Read	Integer Number
Setting	Yes
Unit	Index
Default	0

6.2.241. /DEV0...n/IMPS/0...n/OUTPUT/RATE

The number of output values sent to the computer per second.

Write	Decimal Number
Read	Decimal Number

Setting	Yes
Unit	Hz

6.2.242. /DEV0...n/IMPS/0...n/OUTPUT/AMPLITUDE

Control output amplitude value when Drive Control is off.

Write	Decimal Number
Read	Decimal Number
Setting	Yes
Unit	V

6.2.243. /DEV0...n/IMPS/0...n/SAMPLE

Impedance Samples.

Write	-
Read	-
Stream	-
Setting	No

6.2.244. /DEV0...n/SIGINS

Signal Input nodes.

6.2.245. /DEV0...n/SIGINS/0...n

Nodes of a signal input.

6.2.246. /DEV0...n/SIGINS/0...n/RANGE

Voltage range for the signal input.

Write	Decimal Number
Read	Decimal Number
Setting	Yes
Unit	V
Default	1.2
Range	0.0001 to 2

6.2.247. /DEV0...n/SIGINS/0...n/AC

Boolean value setting for AC coupling of the Signal Input.

Write	Integer Number
Read	Integer Number
Setting	Yes

Unit	Boolean				
Default	0 (DC coupling)				
Values	<table> <tr> <td>0</td> <td>DC coupling</td> </tr> <tr> <td>1</td> <td>AC coupling</td> </tr> </table>	0	DC coupling	1	AC coupling
0	DC coupling				
1	AC coupling				

6.2.248. /DEV0...n/SIGINS/0...n/IMP50

Boolean value enabling 50 Ohm input impedance termination.

Write	Integer Number				
Read	Integer Number				
Setting	Yes				
Unit	Boolean				
Default	0 (High impedance)				
Values	<table> <tr> <td>0</td> <td>High impedance</td> </tr> <tr> <td>1</td> <td>50 Ohm impedance</td> </tr> </table>	0	High impedance	1	50 Ohm impedance
0	High impedance				
1	50 Ohm impedance				

6.2.249. /DEV0...n/SIGINS/0...n/DIFF

Boolean value switching differential input mode.

Write	Integer Number				
Read	Integer Number				
Setting	Yes				
Unit	Boolean				
Default	0 (Single-ended inputs)				
Values	<table> <tr> <td>0</td> <td>Single-ended inputs</td> </tr> <tr> <td>1</td> <td>Differential inputs</td> </tr> </table>	0	Single-ended inputs	1	Differential inputs
0	Single-ended inputs				
1	Differential inputs				

6.2.250. /DEV0...n/SIGOUTS

Signal Output nodes.

6.2.251. /DEV0...n/SIGOUTS/0...n

Nodes of a Signal Output.

6.2.252. /DEV0...n/SIGOUTS/0...n/ON

Switches the output on and off.

Write	Integer Number
Read	Integer Number
Setting	Yes
Unit	Boolean
Default	0 (Output off)

Values	0	Output off
	1	Output on

6.2.253. /DEV0...n/SIGOUTS/0...n/ADD

Switches the output adder on and off.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Boolean	
Default	0 (Adder off)	
Values	0	Adder off
	1	Adder on

6.2.254. /DEV0...n/SIGOUTS/0...n/RANGE

Selects the output range for the Signal Output.

Write	Decimal Number	
Read	Decimal Number	
Setting	Yes	
Unit	V	
Default	1	
Values	0.01	0.01 V range
	0.1	0.1 V range
	1	1 V range
	10	10 V range

6.2.255. /DEV0...n/SIGOUTS/0...n/OFFSET

Offset added to the Signal Output.

Write	Decimal Number	
Read	Decimal Number	
Setting	Yes	
Unit	Gain	
Default	0	
Range	-1 to 1	

Details

Multiply this value with the range setting to obtain offset voltage in V.

6.2.256. /DEV0...n/SIGOUTS/0...n/ENABLES

Switches for channels in the mixer.

6.2.257. /DEV0...n/SIGOUTS/0...n/ENABLES/0...n

Switches a channel in the mixer on and off.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Values	0	Channel off (unconditionally)
	1	Channel on (unconditionally)
	2	Channel off (will be turned off on next change of sign from negative to positive)
	3	Channel on (will be turned on on next change of sign from negative to positive)

6.2.258. /DEV0...n/SIGOUTS/0...n/AMPLITUDES

Amplitudes for channels in the mixer.

6.2.259. /DEV0...n/SIGOUTS/0...n/AMPLITUDES/0...n

Fraction of the output range added to the output signal.

Write	Decimal Number	
Read	Decimal Number	
Setting	Yes	
Unit	Gain	
Range	-1 to 1	

Details

Multiply this value with the range setting to obtain voltage in V.

6.2.260. /DEV0...n/SIGOUTS/0...n/WAVEFORMS

6.2.261. /DEV0...n/SIGOUTS/0...n/WAVEFORMS/0...n

Waveforms for a channel in the mixer.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Default	0 (Sine)	
Values	0	Sine
	1	Square

Details

For hardware revisions 1.4 and lower, the output signal range for rectangular output is limited to 1 V.

6.2.262. /DEV0...n/SCOPES

Scope nodes.

6.2.263. /DEV0...n/SCOPES/0...n

Nodes of a scope.

6.2.264. /DEV0...n/SCOPES/0...n/ENABLE

Enables the scope.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Boolean	
Default	0 (Scope off)	
Values	0	Scope off
	1	Scope on

6.2.265. /DEV0...n/SCOPES/0...n/CHANNEL

Selects the channel for which scope data should be provided.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Index	
Default	0 (Signal Input 1)	
Range	0 to 3	
Values	0	Signal Input 1
	1	Signal Input 2
	2	Signal Output 1
	3	Signal Output 2

6.2.266. /DEV0...n/SCOPES/0...n/TRIGCHANNEL

Selects the channel which should be used as source for the scope's trigger.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Index	
Default	-1 (Off)	
Range	-2 to 11	
Values	-2	Continuous

-1	Off
0	Signal Input 1
1	Signal Input 2
2	Signal Output 1
3	Signal Output 2
4	Oscillator 1 phase
5	Oscillator 2 phase
6	Oscillator 3 phase
7	Oscillator 4 phase
8	Oscillator 5 phase
9	Oscillator 6 phase
0	Oscillator 7 phase
11	Oscillator 8 phase
12	DIO 0
13	DIO 1

6.2.267. /DEV0...n/SCOPES/0...n/BWLIMIT

The bandwidth-limit for the scope.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Boolean	
Default	0 (BW-limit off)	
Values	0	BW-limit off
	1	BW-limit on

6.2.268. /DEV0...n/SCOPES/0...n/TRIGEDGE

Selects whether the scope should trigger on rising or falling edge.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Default	1 (Rising edge)	
Values	0	Falling edge
	1	Rising edge

6.2.269. /DEV0...n/SCOPES/0...n/TRIGLEVEL

Level at which a trigger is raised.

Write	Integer Number	
Read	Integer Number	

Setting	Yes
Unit	LSB
Default	0
Range	-32768 to 32767

Details

Full scale is covered by min and max values

6.2.270. /DEV0...n/SCOPES/0...n/TRIGHOLDOFF

Time to wait for re-arming the trigger after one occurred.

Write	Decimal Number
Read	Decimal Number
Setting	Yes
Unit	s
Default	0.25
Range	0 to 65.535

6.2.271. /DEV0...n/SCOPES/0...n/TIME

Timescale of the scope wave (logarithmic decimation).

Write	Integer Number
Read	Integer Number
Setting	Yes
Default	0
Range	0 to 7

Details

Determines the decimation of the sample rate. The following formulas apply: span = $2^{val} * 10$ us, sample rate = 210 MSamples/ 2^{val}

6.2.272. /DEV0...n/SCOPES/0...n/WAVE

Samples of scope-waveforms.

Write	-
Read	-
Stream	-
Setting	No

6.2.273. /DEV0...n/DIOS

DIO nodes.

6.2.274. /DEV0...n/DIOS/0...n

Nodes of a DIO.

6.2.275. /DEV0...n/DIOS/0...n/EXTCLK

Selects whether an external clock source should be used.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Boolean	
Default	0 (Internal clock)	
Values	0	Internal clock
	1	External clock

Details

The external clock needs to be applied to the DIO connector when this node is set to 1.

6.2.276. /DEV0...n/DIOS/0...n/DECIMATION

Decimation for the sample rate of the DIO.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Default	256	
Range	0 to 65535	

6.2.277. /DEV0...n/DIOS/0...n/DRIVE

Selects if the outputs should be driven.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Default	0 (Drive off)	
Range	0 to 3	
Values	0	Drive off
	1	Drive lower 8 bits
	2	Drive higher 8 bits
	3	Drive all 16 bits

6.2.278. /DEV0...n/DIOS/0...n/OUTPUT

Bits to output.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	

Default	0
Range	0 to 65535

6.2.279. /DEV0...n/DIOS/0...n/SYNCSELECT0

Source to output the sync signal on bit 0.

Write	Integer Number
Read	Integer Number
Setting	Yes
Default	8
Range	0 to 8

6.2.280. /DEV0...n/DIOS/0...n/SYNCSELECT1

Source to output the sync signal on bit 1.

Write	Integer Number
Read	Integer Number
Setting	Yes
Default	8
Range	0 to 8

6.2.281. /DEV0...n/DIOS/0...n/INPUT

Samples of the input.

Write	-
Read	-
Stream	-
Setting	No

6.2.282. /DEV0...n/AUXINS

Nodes of auxiliary inputs.

6.2.283. /DEV0...n/AUXINS/0...n

Node for an aux in.

6.2.284. /DEV0...n/AUXINS/0...n/AVERAGING

Averaging of the samples.

Write	Integer Number
Read	Integer Number
Setting	Yes
Default	256

Range	1 to 32768
-------	------------

6.2.285. /DEV0...n/AUXINS/0...n/SAMPLE

Auxiliary input samples.

Write	-
Read	-
Stream	-
Setting	No

6.2.286. /DEV0...n/AUXINS/0...n/VALUES

Nodes of a auxins values.

6.2.287. /DEV0...n/AUXINS/0...n/VALUES/0

Input 0 value.

Write	-
Read	Decimal Number
Setting	No
Unit	V
Range	-10 to 10

6.2.288. /DEV0...n/AUXINS/0...n/VALUES/1

Input 1 value.

Write	-
Read	Decimal Number
Setting	No
Unit	V
Range	-10 to 10

6.2.289. /DEV0...n/AUXOUTS

Nodes of Auxiliary outputs.

6.2.290. /DEV0...n/AUXOUTS/0...n

Nodes of an Auxiliary output.

6.2.291. /DEV0...n/AUXOUTS/0...n/VALUE

Output value.

Write	-
-------	---

Read	Decimal Number
Setting	No
Unit	V
Range	-10 to 10

6.2.292. /DEV0...n/AUXOUTS/0...n/OUTPUTSELECT

Signal to be given out.

Write	Integer Number														
Read	Integer Number														
Setting	Yes														
Values	<table> <tr> <td>-1</td> <td>Manual</td> </tr> <tr> <td>0</td> <td>X</td> </tr> <tr> <td>1</td> <td>Y</td> </tr> <tr> <td>2</td> <td>R</td> </tr> <tr> <td>3</td> <td>Theta</td> </tr> <tr> <td>4</td> <td>PLL 0 (with installed PLL option))</td> </tr> <tr> <td>4</td> <td>PLL 1 (with installed PLL option))</td> </tr> </table>	-1	Manual	0	X	1	Y	2	R	3	Theta	4	PLL 0 (with installed PLL option))	4	PLL 1 (with installed PLL option))
-1	Manual														
0	X														
1	Y														
2	R														
3	Theta														
4	PLL 0 (with installed PLL option))														
4	PLL 1 (with installed PLL option))														

6.2.293. /DEV0...n/AUXOUTS/0...n/DEMODSELECT

Source demodulator.

Write	Integer Number
Read	Integer Number
Setting	Yes
Unit	Index
Default	0

6.2.294. /DEV0...n/AUXOUTS/0...n/SCALE

Scaling of the signal which is given out.

Write	Decimal Number
Read	Decimal Number
Setting	Yes

6.2.295. /DEV0...n/AUXOUTS/0...n/OFFSET

Value to be added to the output.

Write	Decimal Number
Read	Decimal Number
Setting	Yes
Unit	V

Default	0
Range	-2560 to 2560

Details

The offset value is applied after scaling.

6.2.296. /DEV0...n/CPUS

Nodes for the real-time CPUs.

6.2.297. /DEV0...n/CPUS/0...n

Node for one real-time CPU.

Note

Only available with installed real-time option.

6.2.298. /DEV0...n/CPUS/0...n/WORKLOAD

Usage of the processor-time.

Write	-
Read	Decimal Number
Setting	No
Range	0 to 1

6.2.299. /DEV0...n/CPUS/0...n/PROGRAM

Node to write user programs in.

Write	-
Read	-
Setting	No

6.2.300. /DEV0...n/CPUS/0...n/OUTPUT

Node which streams output of the user-program.

Write	-
Read	-
Stream	-
Setting	No

6.2.301. /DEV0...n/CPUS/0...n/USERREGS

General purpose registers to transfer data.

6.2.302. /DEV0...n/CPUS/0...n/USERREGS/0...n

General purpose register.

Write	Integer Number
Read	Integer Number
Setting	Yes

6.2.303. /DEV0...n/ZCTRLS

Node containing connected ZCtrl devices.

Note

There has to be a device connected to either ZCtrl 1 or ZCtrl 2 on the backplane that children of this node appear.

6.2.304. /DEV0...n/ZCTRLS/0...n

ZCtrl instance

6.2.305. /DEV0...n/ZCTRLS/0...n/CAMP

A ZI current-amplifier connected to a ZCtrl port.

6.2.306. /DEV0...n/ZCTRLS/0...n/CAMP/AVAILABLE

1 when HF2CA is connected to the corresponding ZCtrl port.

Write	-
Read	Integer Number
Setting	No
Unit	Boolean
Default	0 (HF2CA is not connected)
Values	0 HF2CA is not connected 1 HF2CA is connected

6.2.307. /DEV0...n/ZCTRLS/0...n/CAMP/R

Chooses a value for the shunt-resistor.

Write	Integer Number
Read	Integer Number
Setting	Yes
Unit	Ohm
Default	0 (open, high ohmic)
Values	0 open, high ohmic

10	10 Ohm
100	100 Ohm
1000	1 kOhm
10000	10 kOhm
100000	100 kOhm
1.0E6	1 MOhm

6.2.308. /DEV0...n/ZCTRLS/0...n/CAMP/GAIN

Switches between factor 1 and 10 gain.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Gain	
Default	1 (Factor 1 gain)	
Values	1	Factor 1 gain
	10	Factor 10 gain

6.2.309. /DEV0...n/ZCTRLS/0...n/CAMP/DC

Switches between AC coupling and DC coupling.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Boolean	
Default	0 (AC coupling)	
Values	0	AC coupling
	1	DC coupling

6.2.310. /DEV0...n/ZCTRLS/0...n/CAMP/SINGLEENDED

Switches between differential and single-ended input.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Boolean	
Default	0 (Differential inputs)	
Values	0	Differential inputs
	1	Single-ended inputs

6.2.311. /DEV0...n/ZCTRLS/0...n/TAMP

A ZI transimpedance-amplifier connected to a ZCtrl port.

6.2.312. /DEV0...n/ZCTRLS/0...n/TAMP/AVAILABLE

1 when HF2TA is connected to the corresponding ZCtrl port.

Write	-
Read	Integer Number
Setting	No
Unit	Boolean
Default	0 (HF2TA is not connected)
Values	0 HF2TA is not connected 1 HF2TA is connected

6.2.313. /DEV0...n/ZCTRLS/0...n/TAMP/BIASOUT

Switches between internal and external bias.

Write	Decimal Number
Read	Decimal Number
Setting	Yes
Unit	V
Default	0

6.2.314. /DEV0...n/ZCTRLS/0...n/TAMP/EXTBIAS

Switches the external bias.

Write	Integer Number
Read	Integer Number
Setting	Yes
Unit	Boolean
Default	0 (External bias off)
Values	0 External bias off 1 External bias on

6.2.315. /DEV0...n/ZCTRLS/0...n/TAMP/0...n

A Channel of a transimpedance amplifier.

6.2.316. /DEV0...n/ZCTRLS/0...n/TAMP/0...n/CURRENTGAIN

Chooses a value for the current gain.

Write	Integer Number
Read	Integer Number
Setting	Yes
Unit	Gain

Default	1000 (Factor 1 k)	
Values	100	Factor 100
	1000	Factor 1 k
	10000	Factor 10 k
	100000	Factor 100 k
	1.0E6	Factor 1 M
	1.0E7	Factor 10 M
	1.0E8	Factor 100 M

6.2.317. /DEV0...n/ZCTRLS/0...n/TAMP/0...n/DC

Switches between AC and DC Mode.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Boolean	
Default	1 (DC Mode)	
Values	0	AC Mode
	1	DC Mode

6.2.318. /DEV0...n/ZCTRLS/0...n/TAMP/0...n/VOLTAGEGAIN

Chooses a value for the voltage gain.

Write	Integer Number	
Read	Integer Number	
Setting	Yes	
Unit	Gain	
Default	1 (1 x)	
Values	1	1 x
	10	10 x

6.2.319. /DEV0...n/ZCTRLS/0...n/TAMP/0...n/OFFSET

Adjust offset value.

Write	Decimal Number	
Read	Decimal Number	
Setting	Yes	
Unit	V	
Default	0	

Chapter 7. Real-time Option

The Real-time option provides the capability to execute programs written in the C programming language on the RISC microprocessor of the HF2 Instrument with predictable latency and comes with an extensive programming environment.

This chapter describes:

- Installation of the Real-time programming environment, in [Section 7.1](#). See [Section 7.1.1](#) and [Section 7.1.2](#) for the installation process on Windows and Linux, respectively. [Section 7.1.3](#) explains where to find the documentation in HTML format.
- The Real-time Option programming reference guide which provides examples and lists the data structures and functions available in the API, in [Section 7.2](#).

Note

RT programming can be used only if the HF2LI-RT / HF2IS-RT option has been purchased and activated. Customers can purchase the RT option at any time, whether when ordering their instrument or after delivery. This option can be activated by the user or by Zurich Instruments via remote servicing.

Note

The LabOne User Interface does not have a tab for configuring and working with the Real-time Option. Please use the ziControl graphical user interface to use the Real-time Option and refer to the ziControl Edition of the HF2 User Manual for more details.

Note

The Real-time Option programming reference guide is also available as HTML. The HTML documentation is bundled with the Real-time installation zip-file available from the Zurich Instruments [download page](#) [<http://www.zhinst.com/downloads>].

7.1. Installation of the Real-time Development Environment

In this section we describe the installation process of the HF2's real-time development environment, see [Section 7.1.1](#) for Windows and [Section 7.1.2](#) for Linux. The real-time development environment is available from the Zurich Instruments [download page](#) [<http://www.zhinst.com/downloads>].

Note

The RT development environment does not include a special editor. Please use an editor of your choice, for example:

- [notepad++](#) [<http://notepad-plus-plus.org>] or [PSPad](#) [<http://www.pspad.com>] on Windows,
- emacs, vim, etc. on Linux.

7.1.1. Installation on Windows

Software Requirements

To use the compilation tools on Windows the RT development environment requires the 32-bit version of Cygwin which provides a Linux-like environment. Cygwin is free and open source software, for more details see the [Cygwin website](#) [<http://www.cygwin.com>]. The only Cygwin package necessary is the `make` package. Installation of Cygwin is also detailed below.

Note

Even if your PC is natively 64-bit, the 32-bit version of Cygwin is required to run the compiler tools distributed with the RT development environment.

Installation Steps

1. Download the 32-bit version of [Cygwin](#) [<http://cygwin.com/setup-x86.exe>] and run the Setup executable.

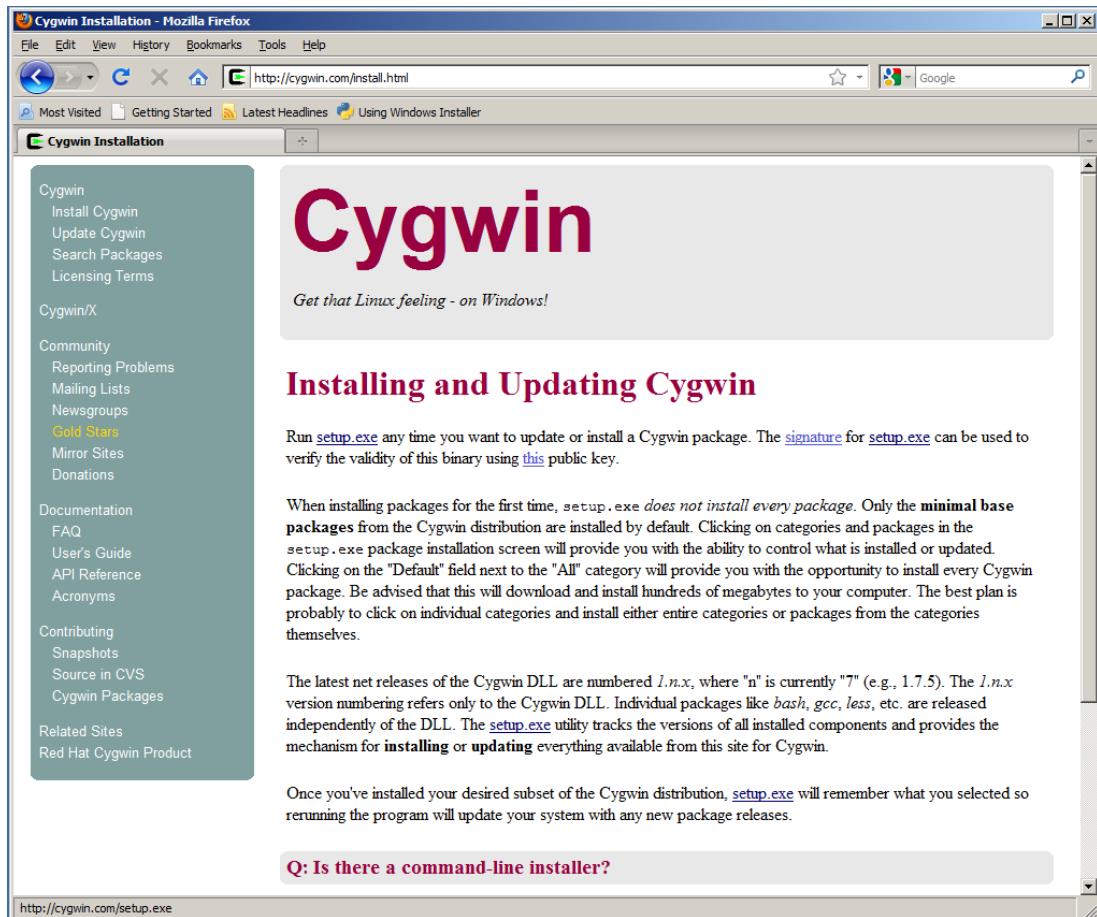


Figure 7.1. Cygwin website with install link

2. Go through the installation and, if possible, use default installation settings. There is one mandatory development package that must be installed in addition to the default installation. The package is called `make`. Select the package at the end of the installation. Select `devel`, then package `make` and select it in order to install it (see screenshots below).



Figure 7.2. Cygwin installation

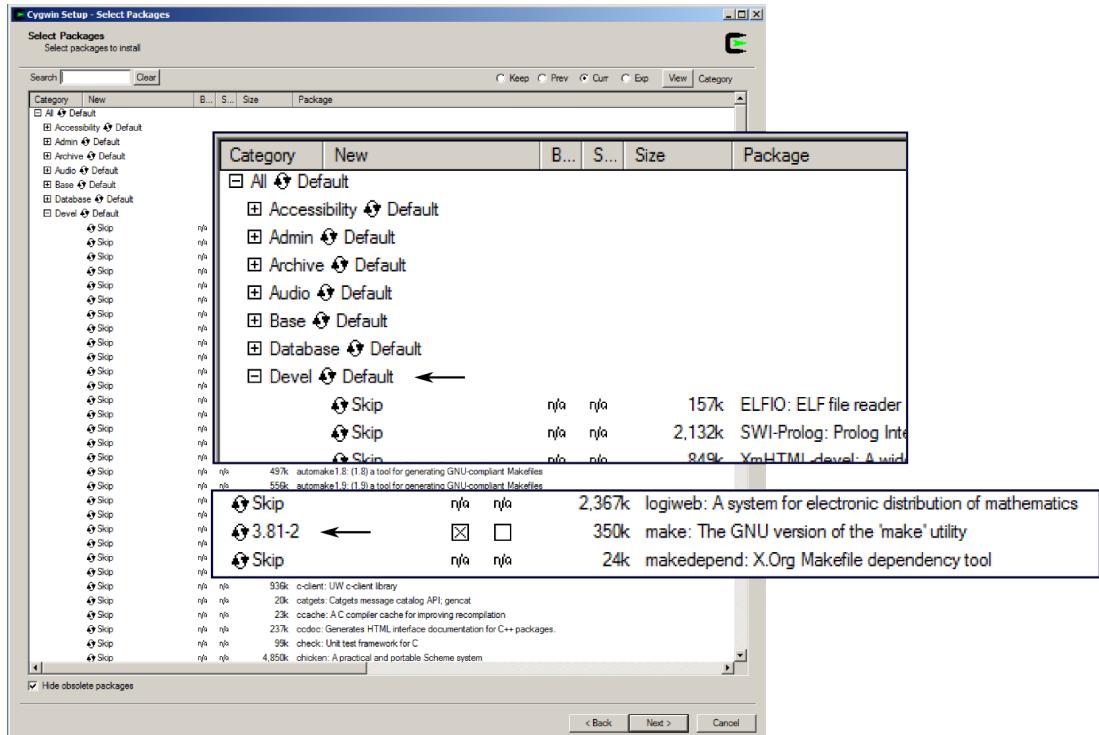


Figure 7.3. Cygwin package selection: Choose package make

- When the Cygwin installation has been finished, download the ziRTK software bundle and unzip it in your Cygwin home folder.

The home folder is located at: [Cygwin installation path]\home

The default path is: C:\cygwin\home

- Now open the Cygwin shell by clicking either by navigating to the Cygwin start menu entry or by clicking on the Desktop icon.
- Navigate into the extracted directory. The install script is called "install.sh".

```
cd ziRTK-Win-[VERSION]
```

- Run the install script and go through the guided installation.

```
bash install.sh
```

Alternatively, you can also give executable rights to the install script and run it directly.

If possible, use default installation paths.

Note

You need not edit your code inside Cygwin, it's just used for compilation and loading your real-time programs onto the HF2 Instrument.

7.1.2. Installation on Linux

Software Requirements

Officially, only Ubuntu 10.04 LTS is supported, but it should be possible to run the tools on any recent Linux distribution. The program GNU make is required to compile the examples. Since the RT tools were compiled on a 32-bit architecture, you need the package ia32_libs installed on a 64-bit architecture in order to execute 32-bit programs on a 64-bit architecture. On a Debian-based system, both packages can be installed with:

```
sudo apt-get install ia32_libs make
```

Installation Steps

1. Before you begin with the ziRTK installation make sure that the development package "make" is installed.

```
sudo apt-get install make
```

2. Extract the ziRTK bundle in a temporary directory.

```
tar xzf ziRTK-[build number]-linux.tar.gz
```

3. Navigate into the extracted directory. The install script is called "install.sh".

```
cd ziRTK-Linux-[VERSION]
```

4. Run the install script with root rights and go through the guided installation.

```
sudo bash install.sh
```

Alternatively, you can also give executable rights to the install script and run it directly.

If possible, use default installation paths.

7.1.3. Accessing the Documentation

The developers of Zurich Instruments now happily recommend you to browse remaining examples and reference documentation in HTML format, which duplicates the following section of the user manual. You can find the HTML version of the documentation in

[INSTALLPATH]/ziRTK/doc/html/index.html,

which is typically found at

C:\cygwin\usr\share\zi\ziRTK-XX.XX\doc\html\index.html

on Windows or

/opt/zi/ziRTK/doc/html/index.html

on Linux.

7.2. Real-Time Option Reference Manual

This documentation is for the Zurich Instruments Real-time Option which enables users to write custom real-time programs that run on an HF2 Instrument.

The recommended way to browse this documentation is in HTML format, see the previous section for help to find it.

The documentation consists of:

- [getting started](#) which describes the necessary steps to compile and load a real-time program to an HF2 Instrument,
- a [simple example](#) that explains the structure of an RTK program,
- other [example programs](#) that demonstrate the most important concepts for controlling and configuring a real-time program,
- a [tips and tricks](#) section to get the best performance from your programs,
- a [brief description](#) of the most important development tools,
- an API Reference, that can either be browsed conceptually by module or as a complete [list](#) of available functions.

7.2.1. Getting Started

This quick-start section demonstrates how to:

- configure the real-time development environment,
- compile C source code,
- run tools on the binary to view memory usage,
- load the binary to the HF2 Instrument.

Please note that the exact shell output and the paths may differ from your installation.

Configuring the RT Development Environment

In order to use the development tools available in your RTK installation you have to configure your shell's `PATH` to include them. This can be done by sourcing the bash file included in the RTK installation directory, in your terminal type:

```
source [INSTALLPATH]/ziRTK-[VERSION]/settings.sh
```

Typically, in Windows (using Cygwin) `INSTALLPATH` is `/usr/share/` and in Linux `INSTALLPATH` is `/opt/zi/`.

In order to assert that the `PATH` has been set correctly, try locating the compiler `mb-gcc` with the `which` program and displaying its help message:

```
$ which mb-gcc  
/usr/share/zi/ziRTK/tools/bin/mb-gcc.exe  
$ mb-gcc --help  
...
```

To automatically have the development tools available when you start a new bash shell you can add the statements in `settings.sh` to your own bash configuration file `.bashrc`, located in your home directory.

Note:

If you're using Windows you can find your `.bashrc` file (in a typical Cygwin installation) at
`C:\cygwin\home\[USERNAME]\.bashrc`
and the `settings.sh` file (typically) at
`C:\cygwin\usr\share\zi\ziRTK-[VERSION]\settings.sh.`

Compiling and Running the Examples

The Real-time option comes with a set of examples to demonstrate the main concepts of programming within the real-time environment. They are installed in your RTK example directory (by default `~/ziRTKExamples/`) during installation but can also be found in the examples directory of the ziRTK bundle downloaded from Zurich Instruments download page, .

There are two subdirectories in the RTK example directory (by default `ziRTKExamples`):

- `examples`, which contains some RT examples that are ready to compile and run,
- `skeleton`, a minimalistic example that can be used as a template for your own programs.

To compile and load an example, perform the following steps:

- Configure your path, as described above.
- In a terminal (under Windows, start Cygwin) navigate to a directory of one of the RTK examples, for example, [AuxInToAuxOut](#).
- Run `make` in the example directory by typing

```
make
```

This runs the `make` program using the `Makefile` found in the example's directory. The `Makefile` describes rules how the binary should be compiled from the source code. You should see output similar to:

```
$ make
mb-gcc -specs=/opt/zi/ziRTK/lib/microblaze/specs
-B/opt/zi/ziRTK/tools/bin/ -mno-xl-soft-mul -mxl-barrel-shift
-mxl-pattern-compare -mhard-float -mcpu=v7.10.b -O3
-fno-strict-aliasing -mxl-float-convert -mxl-float-sqrt
-fsingle-precision-constant -Winline -Wall -Wextra -std=gnu99
-I./. -I/opt/zi/ziRTK/include -c -o obj/AuxInToAuxOut.o
AuxInToAuxOut.c
mb-gcc -specs=/opt/zi/ziRTK/lib/microblaze/specs
-B/opt/zi/ziRTK/lib/microblaze/ -L/opt/zi/ziRTK/lib/microblaze
-T/opt/zi/ziRTK/lib/microblaze/linker_script_mb_standard.ld
-Wl,--as-needed -Wl,--start-group
/opt/zi/ziRTK/lib/microblaze/libziRTKmb.a obj/AuxInToAuxOut.o
-Wl,--end-group -Wl,--no-as-needed -lm -o AuxInToAuxOut.mem.elf
          Determining Size of ELF File ****
mb-size AuxInToAuxOut.mem.elf
      text      data      bss      dec      hex filename
  41650        372     3096    45118    b03e AuxInToAuxOut.mem.elf

          Generating mem file ****
data2mem -bd AuxInToAuxOut.mem.elf -d -o m AuxInToAuxOut.mem
```

Invoking the `make` command compiles, links, generates the binary `.mem`-file and runs some statistics on the binary. The next section tells you how to interpret the memory statistics.

To remove the generated files execute the command `make clean`:

```
$ make clean
rm -f obj/AuxInToAuxOut.o
```

```
rm -f AuxInToAuxOut.mem
rm -f AuxInToAuxOut.mem.elf
rm -f deps/AuxInToAuxOut.d
rm -rf obj
rm -rf deps
```

Analyze Memory Usage

The `ls -l` command lists files and reports information about them (like the size). Don't be shocked by the size of your elf file, the actual memory consumption is much smaller. Running `mb-size` on the generated ELF (Executable and Linkable Format) file informs you on the actual memory consumption.

```
$ ls -l AuxInToAuxOut.mem.elf
-rwxr-xr-x 1 user users 112957 2012-11-23 13:02 AuxInToAuxOut.mem.elf
$ mb-size AuxInToAuxOut.mem.elf
  text     data     bss     dec     hex filename
  41650      372    3096    45118    b03e AuxInToAuxOut.mem.elf
$ mb-strip AuxInToAuxOut.mem.elf
$ ls -l AuxInToAuxOut.mem.elf
-rwxr-xr-x 1 user users 43312 2012-11-23 13:03 AuxInToAuxOut.mem.elf
```

The output figures of `mb-size` are:

- text size of the program code. These are the actual commands. Note that single-value constants are often integrated in the text.
- data amount of data. These are constants like arrays of numbers and strings.
- bss the amount of data RAM that is going to be used uninitialized. This is needed for variables to compute with.
- dec the total memory usage in a decimal number
- hex the total memory usage in a hex number

To be sure that you do not include unneeded information, i.e. debugging information, in the binary you can use `mb-strip`. `mb-strip` strips all debugging information and symbol tables from an elf or object file. In this case you see that the file size is unaffected by running `mb-strip` and everything is OK.

You can find a short description of the other most important tools [here](#).

Loading the Binary to the HF2 Instrument

To load the compiled example to your HF2 Instrument on the command line run `make` specifying the target `prog`

```
make prog
```

This invokes the command line program `zirtkprog` to load the program.

To load the compiled example from ziControl select to "Real-time" tab and click the "Program" button. Note, however, the program has to be compiled from the command line.

7.2.2. Tips and Tricks

- Unless absolutely necessary, it is highly discouraged to use the `double` data type. The hardware floating-point unit only supports single-precision (`float`). Therefore using `double` will make your code ~10-times bigger and 10-times slower.
- Most maths functions exist in a `double` and a `float` variant. The `float` variants have an "`f`" appended. E.g. `cos (double x)` and `cosf (float x)`. For optimal performance, make sure

that you always use the float variants. In order to compare the timings of calling `sin()` versus `sinf()` for example, see the [SpeedTest example](#).

- Configure parameters of your real-time program at run-time to avoid hard-coding and recompiling your program, see the [UserRegs](#) and [MultipleParameters](#) examples.
- View log messages sent to the PC with the `ziRTKPrintf()` function in the Realtime tab of `ziControl` by enabling the Log messages button. Log messages can be viewed on the command line with the program `zirtkcat`.
- Use multiple triggers and test which trigger has fired for advanced control of your program, see the [MultipleTriggers example](#).
- Check that you're not dropping samples when using a sample-based trigger such as `ziRTKAddDemodSampleTrigger()` by ensuring that your program is occupying (slightly) less than 100% CPU. View CPU usage in the realtime tab of `ziControl` or via the node `/devX/cpus/0/workload`.
- Test how fast your `ziRTKLoop()` can run by calling it via a clock trigger (`ziRTKAddClockTrigger()`) with delay 0 and measuring its speed with code similar to that in the [SpeedTest example](#).
- Be aware that calling functions that change instrument settings, e.g., `ziRTKDIOSetOutput()`, or send data to the PC, e.g., `ziRTKPrintf()`, can cause a large amount of data to be sent between the real-time program and the PC. In extreme cases this can overload `ziServer` and cause the real-time program to stop running properly. In order to minimize this:
 - When possible use a "NoUpdate" version of the function, i.e., use `ziRTKDIOSetOutputNoUpdate()` instead of `ziRTKDIOSetOutput()`.
 - Avoid a large number of calls (at full loop speed) to `ziRTKPrintf()`, `ziRTKUserRegGet()` and `ziRTKUserRegSet()`.

7.2.3. Example Real-time Programs

The source code and Makefiles for the examples can be found either in the real-time examples directory that is created during the installation of RTK or in the "examples" folder of the RTK download bundle available from .

- Program structure: A simple program demonstrating `ziRTKInit()` and `ziRTKLoop()`
- User registers: Data transfer between the PC and the RT environment
- Triggers: Write values to an auxiliary output via a clock trigger
- Triggers: Control loop behavior with multiple triggers
- User registers: Control loop behavior and multiple program parameters with user registers
- Loop Structure: Record demodulator data upon a DIO trigger using a finite state machine
- Performance: Calculate the update rate of a real-time program

7.2.4. Program structure: A simple program demonstrating `ziRTKInit()` and `ziRTKLoop()`

In the real-time programming environment there are two functions available to the user which define how a real-time program runs: `ziRTKInit()` and `ziRTKLoop()`. The `main()` function is not directly available to the user, it is implemented elsewhere and manages low-level functionality (such as the triggers to `ziRTKLoop()`). The user must define both functions, although either function can be empty.

`ziRTKInit()` is called once upon loading the real-time program onto the HF2, it is used to initialize the program and HF2. It also defines which triggers should repeatedly cause `ziRTKLoop()` to be called. `ziRTKInit()` can configure `ziRTKLoop()` to run, for example, every time a new demodulator sample output is available (`ziRTKAddDemodSampleTrigger()`) or every time a trigger

signal is sent on the HF2's DIO (`ziRTKAddDIOSampleTrigger()`). `ziRTKInit()` can also specify that `ziRTKLoop()` be called with a fixed time delay (`ziRTKAddClockTrigger()`), or just as fast as possible (`ziRTKAddClockTrigger()` with delay 0), so that `ziRTKLoop()` just gets called back-to-back with minimal delay in between.

For help compiling the example see the [getting started section](#).

```
/* $Rev: 32629 $ $Date: 2015-09-23 14:40:54 +0200 (Wed, 23 Sep 2015) $  
*  
* Description:  
*  
* This example copies the values from auxiliary inputs 0 and 1  
* directly to the auxiliary outputs 0 and 1. ziRTKLoop() is defined  
* to be called every time a new auxiliary input sample arrives via  
* calling the function ziRTKAddAuxInSampleTrigger() to ziRTKInit().  
*  
* The averaging performed at the auxiliary input can be changed by  
* modifying the value of the 0th user register. This can be done in  
* the Real-time tab of ziControl.  
*/  
  
#include <math.h>  
#include <ziRTK.h>  
  
AuxInSample Sample;  
float Ch0, Ch1;  
  
unsigned int Reg, TS=0, PrevTS=0;  
unsigned int Averaging = 10000;  
  
void ziRTKInit() {  
  
    ziRTKAddAuxInSampleTrigger( 0, NULL );  
    ziRTKAuxInSetAveraging( 0, Averaging );  
  
    // Set auxout 0 and 1 to manual  
    ziRTKAuxOutSetOutputSelect( 0, -1 );  
    ziRTKAuxOutSetOutputSelect( 1, -1 );  
  
}  
  
void ziRTKLoop() {  
  
    // Get sample from auxin  
    ziRTKAuxInGetSample( 0, &Sample );  
  
    ziRTKAuxInSampleGetValue( &Sample, 0, &Ch0 );  
    ziRTKAuxInSampleGetValue( &Sample, 1, &Ch1 );  
  
    ziRTKAuxOutSetOffset( 0, Ch0 );  
    ziRTKAuxOutSetOffset( 1, Ch1 );  
  
    ziRTKUserRegGet( 0, &Reg );  
  
    TS = ziRTKGetTimeStamp32();  
  
    if( Reg != Averaging ){  
        Averaging = Reg;  
        ziRTKAuxInSetAveraging( 0, Averaging );  
    }  
  
    //ziRTKPrintf("dT = %08u, UserReg0 = %u\n", ( TS - PrevTS ), Reg);  
  
    PrevTS = TS;  
}
```

7.2.5. User registers: Data transfer between the PC and the RT environment

```
/* $Rev: 34746 $ $Date: 2016-01-20 09:49:49 +0100 (Wed, 20 Jan 2016) $  
*  
* Description:  
*  
* This example demonstrates how to modify a parameter of a running  
* real-time program. This is achieved via so-called user registers  
* (0...63) which are dedicated 32-bit memory slots for a user to  
* transfer data between the PC and a real-time program.  
*  
* Here we get the R value from a demodulator and output it on  
* auxiliary output 0, scaled by a user-specified floating-point  
* factor obtained from user register 0.  
*  
* Test this program:  
*  
* Connect a feedback cable from signal input 0 to signal output 0 and  
* load the configuration file in this folder in the Save tab of  
* ziControl. In the Realtime tab in ziControl load the binary  
* UserRegs.mem, you should see the counter increment in the UserReg 1  
* in the "Decimal/Hex" field. Enable the log and modify the  
* scaleFactor parameter in the "Float Value" field of UserReg 0. You  
* can observe that the scaleFactor printed in the log is modified,  
* and, by going to the Auxiliary I/O Tab, the value of R written to  
* auxiliary output 0 is also modified accordingly.  
*  
* Note:  
*  
* Calling ziRTKLoop() at a very high frequency (e.g., by increasing  
* the demodulator's output rate) would cause a large amount of data  
* to be sent between the real-time program and the PC (due to  
* ziRTKUserRegGet(), ziRTKUserRegSet() and ziRTKPrintf()). In extreme  
* cases this can overload ziServer and cause the real-time program to  
* stop running properly.  
*/  
  
#include <math.h>  
#include <ziRTK.h>  
  
// helper to obtain floating point values from user registers  
typedef union {  
    unsigned int as_uint32;  
    float as_float;  
} union_uint32_float_t;  
  
unsigned int demodIndex = 0;  
unsigned int auxOutIndex = 0;  
  
void ziRTKInit() {  
  
    // Define a demodulator sample trigger  
    ziRTKAddDemodSampleTrigger( demodIndex, NULL );  
    // Use a slow demodulator output rate  
    ziRTKDemodSetRate( demodIndex, 10.0f );  
  
    // Set auxiliary output 0 to manual  
    ziRTKAuxOutSetOutputSelect( auxOutIndex, -1 );  
  
}  
  
DemodSample SampleD;  
float R = 0.0f;
```

```
float scaleFactor = 1.0f;
unsigned int counter = 0;

void ziRTKLoop() {

    // Update configuration. Note we have to do a reinterpret typecast
    // in order to retrieve a floating point value from a user
    // register. Necessary since ziRTKUserRegGet() retrieves an
    // unsigned integer but we want the floating-point number with the
    // same bit representation.
    union_uint32_float_t userreg;
    ziRTKUserRegGet( 0, &userreg.as_uint32 );
    scaleFactor = userreg.as_float;

    // Note if scaleFactor was an integer and not a floating point
    // value, it would be possible to just use:
    // unsigned int scaleFactor;
    // ziRTKUserRegGet( 0, &scaleFactor );

    // Get the demodulator R value
    ziRTKDemodGetSample( demodIndex, &SampleD );
    ziRTKDemodSampleGetCompR( &SampleD, &R );

    // Write the scaled value to the auxiliary output
    ziRTKAuxOutSetOffset( auxOutIndex, scaleFactor*R );

    // Also print the scaled value to the log
    ziRTKPrintf("\n R: %e, scaleFactor: %e\n", R, scaleFactor);

    // Write a counter to user register 1
    ziRTKUserRegSet( 1, counter);
    counter++;

}
```

7.2.6. Triggers: Write values to an auxiliary output via a clock trigger

```
/* $Rev: 937 $ $Date: 2009-01-09 20:19:12 +0100 (Fri, 09 Jan 2009) $
 *
 * Description:
 *
 * This example generates a sine wave (sinRet) on the auxiliary
 * output channel 0 and sets the frequency of oscillator 0 according
 * to fabs(sinRet)*1000000 with a fixed delay interval. The delay is
 * defined using a clock trigger in ziRTKInit().
 *
 * Note:
 *
 * Calling ziRTKLoop() at a very high frequency (e.g., by increasing
 * the demodulator's output rate) would cause a large amount of data
 * to be sent between the real-time program and the PC (due to
 * ziRTKAuxOutSetOffset()). In extreme cases this can overload
 * ziServer and cause the real-time program to stop running
 * properly. Please see the note below about the "NoUpdate" version of
 * ziRTKAuxOutSetOffset().
 *
 */

#include <math.h>
#include <ziRTK.h>

void ziRTKInit() {
```

```
// Define the delay for the clock trigger (in milliseconds)
unsigned int clockTriggerDelay = 200;
ziRTKAddClockTrigger( clockTriggerDelay, NULL );

// Set auxiliary output channel 0 to manual
ziRTKAuxOutSetOutputSelect( 1, -1 );

}

unsigned int counter = 0;
float phaseAcc = 0.;

void ziRTKLoop() {

    const float sinRet = sinf( phaseAcc );

    // Output on auxiliary outputs using with a 10 V amplitude
    // Auxiliary output range is -10 to 10 volts, scale accordingly
    ziRTKAuxOutSetOffset( 0, sinRet * 10. );
    // Note: if ziServer is getting overloaded by a lot of data
    // transfer, try using the no update version:
    // ziRTKAuxOutSetOffsetNoUpdate(), which sets the
    // values in hardware but doesn't communicate the value via USB to
    // the ziServer

    ziRTKOscSetFreq( 0, fabs( sinRet ) * 1000000 );

    // Accumulate phase
    phaseAcc += 0.001;
    if( phaseAcc > (float) 2. * 3.14159265 )
        phaseAcc-= (float) 2. * 3.14159265;

    // Say something every 1000 times
    if( counter % 1000 == 0 ) {
        ziRTKPrintf( "phaseAcc: %f, counter: %d \n",
                     phaseAcc, counter );
    }

    counter++;
}
```

7.2.7. Triggers: Control loop behavior with multiple triggers

```
/* $Rev: 32632 $ $Date: 2015-09-23 14:56:44 +0200 (Wed, 23 Sep 2015) $
 *
 * Description:
 *
 * This example demonstrates how to use multiple triggers in order to
 * perform two actions. The two actions are:
 *
 * 1. When a new sample is available from the demodulators ziRTKLoop()
 * obtains the demodulator's R and Theta values and writes them to the
 * first two auxiliary output channels.
 *
 * 2. When a new sample is available from the first auxiliary input
 * channel it writes the value to the digital output.
 *
 * Additionally, either the value of R and theta or the auxiliary
 * input sample is printed to the log. This can be viewed either in the
 * log messages of ziControl's Realtime tab or in a terminal using the
 * program zirtkcat.
 *
 * This is achieved by adding demodulator and auxiliary input
 * triggers in ziRTKInit(). ziRTKLoop() checks which of the triggers
 * is active and performs the relevant action.
```

```
*  
* Note: Calling ziRTKLoop() at a very high frequency (e.g., by using  
* a very high demodulator output rate) would cause a large amount of  
* data to be sent between the real-time program and the PC (due to  
* ziRTKPrintf()). In extreme cases this can overload ziServer and  
* cause the real-time program to stop running properly.  
*/  
  
#include <math.h>  
#include <ziRTK.h>  
  
ziTriggerId DemodTrigger;  
ziTriggerId AuxInTrigger;  
  
void ziRTKInit() {  
  
    // Define when ziRTKLoop() should be called  
    ziRTKAddDemodSampleTrigger( 0, &DemodTrigger );  
    ziRTKAddAuxInSampleTrigger( 0, &AuxInTrigger );  
  
    // Set a slow demodulator output rate  
    ziRTKDemodSetRate( 0, 1 );  
  
    // Set auxiliary output channels 0 and 1 to manual  
    ziRTKAuxOutSetOutputSelect( 0, -1 );  
    ziRTKAuxOutSetOutputSelect( 1, -1 );  
  
    // Set the DIO to drive outputs  
    ziRTKDIOSetDrive( 0, 3 );  
  
}  
  
DemodSample SampleD;  
  
float R;  
float Theta;  
  
AuxInSample SampleAI;  
float AuxInValue;  
  
unsigned int IsTriggered;  
  
void ziRTKLoop() {  
  
    ziRTKWriteString( "Start of ziRTKLoop\n" );  
  
    ziRTKTestTrigger( DemodTrigger, &IsTriggered );  
  
    if( IsTriggered ) {  
        // Retrieve the sample of demodulator 0  
        ziRTKDemodGetSample( 0, &SampleD );  
  
        // Retrieve the magnitude of the demodulator 0 sample.  
        // All voltages in V  
        ziRTKDemodSampleGetCompR( &SampleD, &R );  
  
        // Retrieve the phase  
        ziRTKDemodSampleGetTheta( &SampleD, &Theta );  
  
        // Output the magnitude on stdout  
        ziRTKPrintf( "Demod. R: %f\n", R );  
  
        // Write R to on auxiliary output channel 0  
        ziRTKAuxOutSetOffset( 0, R );  
        // Write Theta on auxiliary output channel 1  
        // Auxiliary output range is -10 to 10; scale Theta accordingly
```

```
    ziRTKAuxOutSetOffset( 1, Theta / 18 );
}

ziRTKTestTrigger( AuxInTrigger, &IsTriggered );

if( IsTriggered ) {

    // Retrieve the sample from auxiliary input 0
    ziRTKAuxInGetSample( 0, &SampleAI );

    // Retrieve the value of the sample from channel 0
    ziRTKAuxInSampleGetValue( &SampleAI, 0, &AuxInValue );

    ziRTKPrintf( "AuxInValue: %f\n", AuxInValue );

    // Write the auxiliary output value to the digital output.
    // Auxiliary input range is -10 to 10; add 10 to obtain a
    // positive value.
    // Digital output range is 0 to 2^16; multiply by 2^16 to attain
    // an integer value in that range.
    ziRTKDIOSetOutput( 0, ( AuxInValue + 10 ) * 0x7fff );
}

}
```

7.2.8. User registers: Control loop behavior and multiple program parameters with user registers

```
/* $Rev: 34746 $ $Date: 2016-01-20 09:49:49 +0100 (Wed, 20 Jan 2016) $ */
*
* Description:
*
* This program obtains demodulator 0's R value, scales and offsets it
* and writes it to an auxiliary output. ziRTKLoop() is called on a
* demod trigger. The scale, offset and auxiliary channel are all
* configurable via user registers. The mode (idle,compute,debug) of
* the program can also be controlled by a user register.
*
* Test this program:
*
* Connect a feedback cable from signal output 1 to signal input 1 and
* load the configuration file MultipleParameters.zicfg in this
* folder. You should obtain a near constant value on the R value of
* demodulator 0, it is this value we will write to the auxiliary
* outputs.
*
* Now go to the Realtime tab in ziControl and first load the xml file
* containing the user register names, RegisterNames.xml, by clicking
* on the "Reg Name" pull-down menu and selecting "Open...". You
* should see the first 5 user register names get populated.
*
* Now load the real-time program's binary file
* MultipleParameters.mem. The CONTROL register is used to define the
* program's behavior, if no bits of the CONTROL register are set,
* then the program is idle. The bits of CONTROL have the following
* meaning:
*
* - 1: perform a configuration update
* - 2: call the Compute() function (compute the average)
* - 3: show some debugging output
*
* Accordingly, setting the CONTROL register to the following values
* has the described affect:
* - control=0, idle
* - control=1, perform config update and then go idle
```

```
* - control=2, compute (average R and write to auxout)
* - control=3, perform config update once and then compute
* - control=6, compute and debug (write average R to R_AVERAGE
*           register)
* - control=7, perform config update once and then compute and debug
*           (write average R to R_AVERAGE register) See
*           ziRTKLoop() for more details.
*
*
* The program's CONTROL is initially set to 2 (compute bit is set),
* and we can see that the demodulator's R value is being written to
* auxiliary output channel 0 in the Auxiliary I/O Tab in ziControl.
*
* Now set the CONTROL register to 6, which runs the program in
* "compute and debug" mode, now you should see the average value of R
* update in the "DEBUG" register's "Float Value" field.
*
* The scale and offset used in the program can be modified in the
* corresponding register's "Float Value" field and the auxiliary
* output channel to be used can be specified in the AUXOUT_CHANNEL
* registers "Decimal" field. Send the configuration update to the
* program by setting the 1st bit of the CONTROL reg as described
* above.
*
*/
#include <math.h>
#include <ziRTK.h>

// Define which user registers hold which configuration
#define CONTROL_REG      0
#define AUXOUT_CHANNEL_REG 1
#define SCALE_REG         2
#define OFFSET_REG        3
#define DEBUG_REG         4

// helper to obtain floating point values from user registers
typedef union {
    unsigned int as_uint32;
    float as_float;
} union_uint32_float_t;

// Globals /////////////
// Program parameters
float Scale = 1.0f; // Scaling factor for the output
float Offset = 0.0f; // Offset for the output
unsigned int AuxOutChannel = 0;

// Helpers for debugging
// Write debugging output DebugPeriodSeconds times per second
float DebugPeriodSeconds = 0.5f;
unsigned int DebugPeriod = 0;

void ziRTKInit() {

    // Add a demod trigger for ziRTKLoop()
    ziRTKAddDemodSampleTrigger( 0, NULL );

    // Set auxout 0 to manual
    ziRTKAuxOutSetOutputSelect( AuxOutChannel, -1 );

    // Zero all user registers for tidiness
    for(int i=0;i<64;i++) {
        ziRTKUserRegSet( i, 0 );
    }

    // Initialise user registers that are used for program parameters
```

```
// with the initial values set in this program.  
// Note that we have to use our union helper union_uint32_float_t to  
// set floating-point values (see also UserRegs example).  
unsigned int initialControl = 2;  
ziRTKUserRegSet( CONTROL_REG, initialControl );  
ziRTKUserRegSet( AUXOUT_CHANNEL_REG, AuxOutChannel );  
  
union_uint32_float_t userreg;  
userreg.as_float = Scale;  
ziRTKUserRegSet( SCALE_REG, userreg.as_uint32 );  
userreg.as_float = Offset;  
ziRTKUserRegSet( OFFSET_REG, userreg.as_uint32 );  
  
float demodRate = 0.0f;  
ziRTKDemodGetRate( 0, &demodRate );  
DebugPeriod = (unsigned int)(ceil(DebugPeriodSeconds*demodRate));  
  
}  
  
// Retrieve a configuration update from the user registers  
inline void UpdateConfig(void) {  
  
    // Note that we have to use our union helper union_uint32_float_t to  
    // set floating-point values (see also UserRegs example).  
    union_uint32_float_t userreg;  
  
    ziRTKUserRegGet( SCALE_REG, &userreg.as_uint32 );  
    Scale = userreg.as_float;  
  
    ziRTKUserRegGet( OFFSET_REG, &userreg.as_uint32 );  
    Offset = userreg.as_float;  
  
    ziRTKUserRegGet( AUXOUT_CHANNEL_REG, &AuxOutChannel );  
    // Set the auxiliary output channel to manual  
    ziRTKAuxOutSetOutputSelect( AuxOutChannel, -1 );  
  
    // Tell the user the config update via the log  
    ziRTKPrintf(  
        "\nUpdated config: AuxOutChannel: %d, Scale: %e, Offset: %e \n",  
        AuxOutChannel, Scale, Offset );  
  
    float demodRate = 0.0f;  
    ziRTKDemodGetRate( 0, &demodRate );  
    DebugPeriod = (unsigned int)(ceil(DebugPeriodSeconds*demodRate));  
  
}  
  
float result = 0.0f;  
  
// Perform our computation on a new demod sample  
inline void Compute(void) {  
  
    // Get the R value of the first demodulator  
    DemodSample SampleD;  
    ziRTKDemodGetSample( 0, &SampleD );  
    float R = 0.0f;  
    ziRTKDemodSampleGetCompR( &SampleD, &R );  
  
    // Perform a computation on R here..  
    // R = ...  
  
    // Scale and offset  
    result = Scale*R + Offset;  
  
    // Write the result to the auxiliary output  
    ziRTKAuxOutSetOffsetNoUpdate( AuxOutChannel, result );
```

```
}

inline void Debug(void) {

    static unsigned int debugCount = 0;
    if( debugCount > DebugPeriod ) {
        // Write the current output to a user register
        union_uint32_float_t userreg;
        userreg.as_float = result;
        ziRTKUserRegSet( DEBUG_REG, userreg.as_uint32 );
        debugCount = 0;
    }
    debugCount++;
}

// The main loop
void ziRTKLoop() {

    // Get control register
    unsigned int control;
    ziRTKUserRegGet( CONTROL_REG, &control );

    // Update configuration?
    if (control & 0x0001) {
        UpdateConfig();
        ziRTKUserRegSet( CONTROL_REG, control & 0xffffffff );
    }

    // Compute?
    if (control & 0x0002) {
        Compute();
    }

    // Debug?
    if (control & 0x0004) {
        Debug();
    }
}
```

7.2.9. Loop Structure: Record demodulator data upon a DIO trigger using a finite state machine

```
/* $Rev: 34748 $ $Date: 2016-01-20 10:50:14 +0100 (Wed, 20 Jan 2016) $
 *
 * Description:
 *
 * Write a burst of demodulator data to an auxiliary output.
 *
 * Upon being triggered externally by a value written on the HF2's
 * digital input, this program writes RecordingTime milliseconds of
 * data from the demodulators to an auxiliary output.
 *
 * The recording behavior is implemented in ziRTKLoop() as a finite
 * state machine which acts on fresh demodulator samples as defined by
 * the demodulator trigger in ziRTKLoop().
 *
 * Test this program:
 *
 * After loading DemodRecorder.mem, go to the Auxiliary Outputs tab
 * in ziControl, enable the drive button of the lower Digital I/O bits
 * and change the lowest bit (DIO_PIN) of the input. In Auxiliary
 * Output 1's Value field you can see that the demodulator's magnitude
 * is written for a duration of RecordingTime milliseconds.
```

```
*  
*/  
  
#include <math.h>  
#include <zirTK.h>  
  
void zirTKInit() {  
  
    zirTKAddDemodSampleTrigger( 0, NULL );  
  
    // Set auxiliary output channel 0 to manual  
    zirTKAuxOutSetOutputSelect( 0, -1 );  
  
    // Set DIO not to drive outputs  
    zirTKDIOSetDrive( 0, 0 );  
  
}  
  
// DIO pin which will be used as trigger source  
// (may be set by the user)  
#define DIO_PIN          0  
  
// duration of the recording time in Milliseconds  
// (may be set by the user)  
const unsigned int RecordingTime = 2000;  
  
// States of the finite state machine  
#define STATE_IDLE      0  
#define STATE_RECORDING 1  
#define STATE_WAIT      2  
  
// Initialize state of the finite state machine  
unsigned int State = STATE_IDLE;  
  
// Will hold the timestamp of the first sample in recording window  
unsigned int StartTS = 0;  
  
// Will be '1' if the trigger dio pin is in the active state  
unsigned int TriggerPin = 0;  
  
DemodSample SampleD;  
  
DIOSample SampleDIO;  
unsigned int DIOBits;  
  
// Counter used for counting the samples being recorded  
// just for testing!  
unsigned int Count = 0;  
  
void zirTKLoop() {  
  
    // Read the DIO sample  
    zirTKDIOGetSample( 0, &SampleDIO );  
    zirTKDIOSampleGetBits( &SampleDIO, &DIOBits );  
  
    // Check if the pin is low (active state)  
    TriggerPin = ( ( DIOBits & ( 1 << DIO_PIN ) ) == 0 ) ? 1 : 0;  
  
    if( State == STATE_IDLE ) {  
  
        zirTKAuxOutSetOffset( 0, 0 );  
  
        // Wait for the trigger pin going to active state  
        if( TriggerPin == 1 )  
            State = STATE_RECORDING;  
  
        StartTS = 0;  
    }  
}
```

```
Count = 0;

}

if( State == STATE_RECORDING ) {
    // Record as long as the timestamp of the demodsample lies within
    // the recording window

    // Retrieve the sample from demodulator 0
    ziRTKDemodGetSample( 0, &SampleD );

    unsigned int CurrentTS;
    ziRTKDemodSampleGetTimeStamp32( &SampleD, &CurrentTS );

    if( StartTS == 0 ) {
        // Set the start timestamp if the current sample is the first
        // recorded sample ( StartTS == 0 )
        StartTS = CurrentTS;
    }
    else {
        // Check if the time recorded is bigger then the required
        // recording time
        // If so advance state to STATE_WAIT

        if( ( CurrentTS - StartTS )
            / ( ZIRTK_HF2_SAMPLERATE / 1000 ) > RecordingTime )
            State = STATE_WAIT;
    }
}

if( State == STATE_RECORDING ) {
    // Current timestamp lies within time window
    // (State has not been advanced in check)

    // Do something with the samples here!

    // For testing purposes printf the counter and increment it
    ziRTKPrintf("Count: %d\n", Count );
    Count++;

    float Mag;
    ziRTKDemodSampleGetCompR( &SampleD, &Mag );
    Mag = Mag * 10;           // Scale the Mag to 0 V ... 10 V
    if( Mag > 10 ) Mag = 10; // Clip to 10 V

    // Output on auxiliary output
    ziRTKAuxOutSetOffset( 0, Mag );
}

if( State == STATE_WAIT ) {

    ziRTKAuxOutSetOffset( 0, 0 );

    // Wait until the trigger pin returns to inactive state to return
    // to beginning
    if( TriggerPin == 0 )
        State = STATE_IDLE;

}

}
```

7.2.10. Performance: Calculate the update rate of a real-time program

```
/* $Rev: 34746 $ $Date: 2016-01-20 09:49:49 +0100 (Wed, 20 Jan 2016) $  
*  
* Description:  
*  
* This example shows how to check the update rate of ziRTKLoop().  
*  
* Test this program:  
*  
* Load the program SpeedTest.mem and the RegisterNames.xml file in  
* the Realtime tab of ziControl. The approximate time taken and  
* update rate of ziRTKLoop() is shown in the user registers 1 and 2.  
* Change the decimal value of the Control register to modify which  
* code is executed. We see that the floating point version of sin()  
* runs considerably faster.  
*/  
  
#include <math.h>  
#include "ziRTK.h"  
  
// Define user registers  
#define CONTROL_REG 0  
#define PERFORMANCE_DT_REG 1  
#define PERFORMANCE_RATE_REG 2  
  
// Helper for writing floats to user registers  
typedef union {  
    unsigned int as_uint;  
    float as_float;  
} union_uint_float;  
  
// Globals /////////////////////////////////  
  
// The delay (milliseconds) for the clock trigger.  
// Delay 0 results in continual triggering.  
unsigned int ziRTKLoopTimeDelay = 0;  
  
// Variables for performance monitoring:  
// Define how often we should calculate and update the performance  
unsigned int PerformanceRefreshPeriodSeconds = 1.0f;  
  
// Calculate and update performance after this many counts.  
// After the first performance calculation this is adjusted according  
// to PerformanceRefreshPeriodSeconds  
unsigned int PerformanceRefreshPeriod = 10000;  
  
// Counter to test when to calculate and update performance  
unsigned int PerformanceCount = 0UI;  
  
// Store the timestamp to calculate performance  
unsigned int PerformanceLastTs = 0UI;  
  
void ziRTKInit() {  
  
    // Define a clock trigger  
    const unsigned int timerDelayMicroSeconds = ziRTKLoopTimeDelay;  
    ziRTKAddClockTrigger( timerDelayMicroSeconds, NULL );  
  
    // Initialise RTK user registers to zero  
    for(unsigned int i=0;i<64;i++) {  
        ziRTKUserRegSet( i, 0 );  
    }  
}
```

```
}

PerformanceLastTs = ziRTKGetTimeStamp32();

}

inline void Performance(void) {

    if( PerformanceCount > PerformanceRefreshPeriod ) {

        // Get the current timestamp from the HF2 in ticks (integer)
        const unsigned int ts = ziRTKGetTimeStamp32();

        // Calculate the total time taken and convert it from ticks to
        // seconds by dividing by ZIRTK_HF2_SAMPLERATE
        const float dtPeriod =
            (float)(ts - PerformanceLastTs)/(float)(ZIRTK_HF2_SAMPLERATE);

        // Get the time taken for one loop
        const float dt = dtPeriod/PerformanceRefreshPeriod;
        // and calculate the rate
        const float rate = 1.0f/dt;

        // Adjust timing to run performance calculation approx every
        // PerformanceRefreshPeriodSeconds
        PerformanceRefreshPeriod = PerformanceRefreshPeriodSeconds/dt;

        // See example UserRegs for an explanation
        union_uint_float userreg;

        // Send rate and dt to the PC via user registers
        userreg.as_float = rate;
        ziRTKUserRegSet( PERFORMANCE_RATE_REG, userreg.as_uint );

        userreg.as_float = dt;
        ziRTKUserRegSet( PERFORMANCE_DT_REG, userreg.as_uint );

        PerformanceCount = 0Ui;
        PerformanceLastTs = ziRTKGetTimeStamp32();

    }

    PerformanceCount++;

}

void ziRTKLoop(void) {

    // Get the control register which determines which function we want
    // to calculate
    unsigned int control = 0;
    ziRTKUserRegGet( CONTROL_REG, &control );

    // Declare volatile to prevent the compiler from performing
    // optimization that would distort our timings
    volatile float f = 0.0f;
    volatile float g = 1.23f;

    switch( control ) {
    case 0:
        // The normal version of sin()
        f = sin(g);
        break;
    default:
        // The floating point version of sin()
        f = sinf(g);
    }

}
```

```
// Check and calculate(?) the current performance of this program  
Performance();  
}
```

7.2.11. Development Tools

This section briefly documents the most important command-line tools delivered with the real-time programming environment. They can be found in

[INSTALLPATH]/ziRTK/tools/bin/.

For more information please see the Xilinx "Embedded System Tools Reference Manual".

- *zirtkprog* - loads binary real-time programs (.mem files) from the PC to an HF2 Instrument. Note, that it's also possible to load a program via the "Realtime" tab in ziControl.
- *zirtkcat* - displays the output ziRTKprintf of a real-time program running on an HF2 Instrument. Note, that it's also possible to view the output of a real-time program in the "Realtime" tab in ziControl.
- *mb-gcc* - the cross-compiler for C programs. It generates an .elf file from .c source files.
- *data2mem* - creates a .mem file from an .elf file.
- *mb-size* - displays size information of a .mem file.
- *mb-readelf* - displays detailed information about the .elf file.

7.2.12. Module Documentation

User Code

This section describes the two functions that you need to implement with your own code. The two functions are called by the RTK at the appropriate times.

Functions

- `void ziRTKInit (void)`
Declaration for the `ziRTKInit` function. This function has to be defined by the user.
- `void ziRTKLoop (void)`
Declaration for the loop function. This function has to be defined by the user.

Detailed Description

In `ziRTKInit`, you may initialize parameters to your needs and add triggers on specific samples or on a timer interval. After initializing, `ziRTKLoop` is executed each time one of the configured triggers is activated. Typically you will do some calculations on new data and output the results on the analog outputs, the digital IOs or send them to the host computer.

Function Documentation

ziRTKInit

void ziRTKInit (void)

Declaration for the ziRTKInit function. This function has to be defined by the user.

Used to initialize all settings for the current application e.g. add triggers or set initial values.

```
#include <ziRTK.h>

void ziRTKInit()
{
    // Add a clock trigger at an interval of 1 millisecond.
    ziRTKAddClockTrigger( 1000, NULL );

    // Set the rate of demodulator 0 to 1 Hz.
    ziRTKDemodSetRate( 0, 1 );

    // Write a "Hello" message to usb.
    ziRTKWriteString( "Hello" );
}
```

See Also:

[ziRTKLoop](#), [ziRTKAddClockTrigger](#), [ziRTKAddDemodSampleTrigger](#), [ziRTKAddDIOSampleTrigger](#),
[ziRTKAddAuxInSampleTrigger](#), [ziRTKAddUserRegTrigger](#)

ziRTKLoop

```
void ziRTKLoop ( void )
```

Declaration for the loop function. This function has to be defined by the user.

This function will be executed every time a trigger occurs. Use ziRTKTestTrigger to test which triggers caused the function to be executed.

```
#include <math.h>
#include <ziRTK.h>

// Output the Magnitude and Phase of demodulator 0 on auxiliary output 0 and
auxiliary output 1.

DemodSample Sample;

void ziRTKLoop()
{
    //retrieve sample
    ziRTKDemodGetSample( 0, &Sample );

    //calculate magnitude
    float Mag;
    ziRTKDemodSampleGetCompR( &Sample, &Mag );

    //calculate phase
    float Phi;
    ziRTKDemodSampleGetTheta( &Sample, &Phi );

    //put the values on the misc analog outputs
    ziRTKAuxOutSetOffset( 0, Mag * 10 );
    ziRTKAuxOutSetOffset( 1, Phi * 10 );

}
```

See Also:

[ziRTKInit](#), [ziRTKTestTrigger](#)

General Functions

This sections describes the basic functions that provide information on the system.

Functions

- `float ziRTKGetTimeStamp ()`
Returns a the current timestamp in seconds.
- `unsigned int ziRTKGetTimeStamp32 ()`
Returns a the current timestamp in a 32bit unsigned int.
- `unsigned long long ziRTKGetTimeStamp64 ()`
Returns a the current timestamp in a 64 bit unsigned long long.
- `float ziRTKGetWorkload ()`
Get the current workload of the CPU.
- `void ziRTKUpdateHostPCOn (void)`
Enables updating of ziServer when a value changes.
- `void ziRTKUpdateHostPCOff (void)`
Disables updating of ziServer when a value changes.
- `unsigned char ziRTKGetUpdateHostPC (void)`
Enables updating of ziServer when a value changes.
- `void ziRTKPause (int Value)`
Pauses or resumes the updating of all signal-processing related parameters. Useful for synchronizing parameter updates with the RTK.

Function Documentation

ziRTKGetTimeStamp

float ziRTKGetTimeStamp ()

Returns a the current timestamp in seconds.

Returns:

the timestamp as a float

See Also:

[ziRTKGetTimeStamp32](#), [ziRTKGetTimeStamp64](#)

ziRTKGetTimeStamp32

unsigned int ziRTKGetTimeStamp32 ()

Returns a the current timestamp in a 32bit unsigned int.

Returns:

the timestamp as an unsigned int

See Also:

[ziRTKGetTimeStamp](#), [ziRTKGetTimeStamp64](#)

ziRTKGetTimeStamp64

`unsigned long long ziRTKGetTimeStamp64()`

Returns a the current timestamp in a 64 bit unsigned long long.

Returns:

the timestamp as an unsigned long long

See Also:

[ziRTKGetTimeStamp](#), [ziRTKGetTimeStamp32](#)

ziRTKGetWorkload

float ziRTKGetWorkload ()

Get the current workload of the CPU.

Returns:

the current workload as a float value (0 - 1)

ziRTKUpdateHostPCOn

void ziRTKUpdateHostPCOn (void)

Enables updating of ziServer when a value changes.

See Also:

[ziRTKUpdateHostPCOff](#), [ziRTKGetUpdateHostPC](#)

ziRTKUpdateHostPCOff

void ziRTKUpdateHostPCOff (void)

Disables updating of ziServer when a value changes.

See Also:

[ziRTKUpdateHostPCOn](#), [ziRTKGetUpdateHostPC](#)

ziRTKGetUpdateHostPC

`unsigned char ziRTKGetUpdateHostPC (void)`

Enables updating of ziServer when a value changes.

Returns:

- 0 when updating of ziServer is disabled
- 1 when updating of ziServer is enabled

See Also:

[ziRTKUpdateHostPCOn](#), [ziRTKUpdateHostPCOff](#)

ziRTKPause

void ziRTKPause (int Value)

Pauses or resumes the updating of all signal-processing related parameters. Useful for synchronizing parameter updates with the RTK.

Parameters:

[in] Value

when Value is not zero the updating is paused otherwise all updates are running

Triggers

A trigger calls user code contained in `ziRTKLoop` on defined occurrences. I.e., a trigger can be set up to call the `Loop()` function every time a new Sample from the Demodulator arrives or when a certain time has passed.

Functions

- `ZI_STATUS ziRTKAddDemodSampleTrigger (unsigned int Index, ziTriggerId* TriggerId)`
Add a trigger on a demod sample.
- `ZI_STATUS ziRTKAddDIOSampleTrigger (unsigned int Index, ziTriggerId* TriggerId)`
Add a trigger on a DIO sample.
- `ZI_STATUS ziRTKAddAuxInSampleTrigger (unsigned int Index, ziTriggerId* TriggerId)`
Add a trigger on an auxiliary input sample.
- `ZI_STATUS ziRTKAddClockTrigger (unsigned int USec, ziTriggerId* TriggerId)`
Add a trigger on a timer.
- `ZI_STATUS ziRTKAddUserRegTrigger (unsigned int Index, ziTriggerId* TriggerId)`
Add a trigger on a user-register.
- `ZI_STATUS ziRTKTestTrigger (ziTriggerId TriggerId, unsigned int* IsSet)`
Test if a trigger has been activated.

Detailed Description

Below is a simple program that creates two triggers and tests which trigger(s) fired via the function `ziRTKTestTrigger()`.

```
#include <math.h>
#include <zirTK.h>

ziTriggerId DemodTriggerId;
ziTriggerId ClockTriggerId;

unsigned int LoopCount = 0;

void zirTKInit()
{
    // Define when zirTKLoop() should be called trigger on samples of
    // demodulator 0.
    ziRTKAddDemodSampleTrigger( 0, &DemodTriggerId );

    // Add a trigger which fires every second.
    ziRTKAddClockTrigger( 1000000, &ClockTriggerId );
}

void zirTKLoop()
{
```

```
unsigned int IsSet;

// Check if the demod-trigger has fired.
ziRTKTestTrigger( DemodTriggerId, &IsSet );

if( IsSet )
    ziRTKPrintf( "Demod Trigger (%u)!\n", LoopCount );

// Check if the clock-trigger has fired.
ziRTKTestTrigger( ClockTriggerId, &IsSet );

if( IsSet )
    ziRTKPrintf( "Clock Trigger (%u)!\n", LoopCount );

LoopCount++;
}
```

For other examples see:

- [ClockTrigger](#)
- [AuxInToAuxOut](#)
- [MultipleTriggers](#)

Function Documentation

ziRTKAddDemodSampleTrigger

ZI_STATUS ziRTKAddDemodSampleTrigger (unsigned int Index, ziTriggerId* TriggerId)

Add a trigger on a demod sample.

ziRTKAddDemodSampleTrigger adds a trigger on a demodulator sample.

Parameters:

[in] Index

index of the demodulator

[out] TriggerId

Pointer to a ziTriggerId returning the id of the newly added trigger. If not used pass NULL.

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when the index is out of range
- ZI_LIMIT when no more triggers are available

See Also:

[ziRTKAddDIOSampleTrigger](#), [ziRTKAddAuxInSampleTrigger](#), [ziRTKAddClockTrigger](#),
[ziRTKAddUserRegTrigger](#), [ziRTKTestTrigger](#)

ziRTKAddDIOSampleTrigger

ZI_STATUS ziRTKAddDIOSampleTrigger (unsigned int Index, ziTriggerId* TriggerId)

Add a trigger on a DIO sample.

ziRTKAddDIOSampleTrigger adds a trigger on a DIO sample.

Parameters:

[in] Index

index of the DIO

[out] TriggerId

Pointer to a ziTriggerId returning the id of the newly added trigger. If not used pass NULL.

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when the index is out of range
- ZI_LIMIT when no more triggers are available

See Also:

[ziRTKAddDemodSampleTrigger](#), [ziRTKAddAuxInSampleTrigger](#), [ziRTKAddClockTrigger](#),
[ziRTKAddUserRegTrigger](#), [ziRTKTestTrigger](#)

ziRTKAddAuxInSampleTrigger

ZI_STATUS ziRTKAddAuxInSampleTrigger (unsigned int Index, ziTriggerId* TriggerId)

Add a trigger on an auxiliary input sample.

ziRTKAddAuxInSampleTrigger adds a trigger on an auxiliary input sample.

Parameters:

[in] Index

index of the auxiliary input

[out] TriggerId

Pointer to a ziTriggerId returning the id of the newly added trigger. If not used pass NULL.

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when the index is out of range
- ZI_LIMIT when no more triggers are available

See Also:

[ziRTKAddDemodSampleTrigger](#), [ziRTKAddDIOSampleTrigger](#), [ziRTKAddClockTrigger](#),
[ziRTKAddUserRegTrigger](#), [ziRTKTestTrigger](#)

ziRTKAddClockTrigger

ZI_STATUS ziRTKAddClockTrigger (unsigned int USec, ziTriggerId* TriggerId)

Add a trigger on a timer.

ziRTKAddClockTrigger adds a trigger that occurs in an interval

Parameters:

[in] USec

Trigger interval in microseconds

[out] TriggerId

Pointer to a ziTriggerId returning the id of the newly added trigger. If not used pass NULL.

Returns:

- ZI_SUCCESS on success
- ZI_LIMIT when no more triggers are available

See Also:

[ziRTKAddDemodSampleTrigger](#), [ziRTKAddDIOSampleTrigger](#), [ziRTKAddAuxInSampleTrigger](#),
[ziRTKAddUserRegTrigger](#), [ziRTKTestTrigger](#)

ziRTKAddUserRegTrigger

ZI_STATUS ziRTKAddUserRegTrigger (unsigned int Index, ziTriggerId* TriggerId)

Add a trigger on a user-register.

ziRTKAddUserRegTrigger adds a trigger that occurs on a change of a user-register

Parameters:

[in] Index

Index of the register (currently 0 - 15)

[out] TriggerId

Pointer to a ziTriggerId returning the id of the newly added trigger. If not used pass NULL.

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_LIMIT no more triggers are available

See Also:

[ziRTKAddDemodSampleTrigger](#), [ziRTKAddDIOSampleTrigger](#), [ziRTKAddAuxInSampleTrigger](#),
[ziRTKAddClockTrigger](#), [ziRTKTestTrigger](#)

ziRTKTestTrigger

ZI_STATUS ziRTKTestTrigger (ziTriggerId TriggerId, unsigned int* IsSet)

Test if a trigger has been activated.

In case you have added more than one trigger, you can use this function in [ziRTKLoop](#) to determine which of the triggers were active

Parameters:

[in] TriggerId

the identifier for which the state should be returned

[out] IsSet

Pointer to an unsigned int which is not equal to zero, when the trigger is active, else zero

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when TriggerId is out of range

See Also:

[ziRTKAddDemodSampleTrigger](#), [ziRTKAddDIOSampleTrigger](#), [ziRTKAddAuxInSampleTrigger](#),
[ziRTKAddClockTrigger](#), [ziRTKAddUserRegTrigger](#)

Sending Data to the Host Computer

The ZI RTK features functions to send data via USB to a host computer running the ziBase drivers and utilities.

Functions

- [**ZI_STATUS** ziRTKPrintf \(char* Fmt, ... \)](#)
printf-compatible print to stdio. The output consists of a string which may include formatting directives.
- [**ZI_STATUS** ziRTKWriteString \(char* Str \)](#)
Write a zero-terminated string to stdio.
- [**ZI_STATUS** ziRTKWriteData \(unsigned char* Buffer, unsigned int Len \)](#)
Write a buffer to stdio.

Function Documentation

ziRTKPrintf

ZI_STATUS ziRTKPrintf (char* Fmt, ...)

printf-compatible print to stdio. The output consists of a string which may include formatting directives.

Parameters:

[in] Fmt

format string composed of zero or more formatting directives and ordinary characters

[in] ...

variable count of arguments being formatted and given out according to the format string

Returns:

- ZI_SUCCESS on success

This function description is based on the Unix man pages for printf.

The format string is composed of zero or more directives: ordinary characters (not %), which are copied unchanged to the output stream; and conversion specifications, each of which results in fetching zero or more subsequent arguments. Each conversion specification is introduced by the % character.

The arguments must correspond properly (after type promotion) with the conversion specifier. All integer values (signed and unsigned) have to be 32bit values, all floating point values have to be double but standard type promotion automatically converts float values to double. Pointer arguments also have to be 32bit wide, char-arrays as arguments are not supported.

After the %, the following appear in sequence:

- Zero or more of the following flags:
 - "0" (zero) Zero padding. For all conversions except n, the converted value is padded on the left with zeros rather than blanks. If a precision is given with a numeric conversion (d, i, o, u, i, x, and X), the 0 flag is ignored.
 - "-" A negative field width flag; the converted value is to be left adjusted on the field boundary. Except for n conversions, the converted value is padded on the right with blanks, rather than on the left with blanks or zeros. A - overrides a 0 if both are given.
 - " " (space) A blank should be left before a positive number produced by a signed conversion (a, A, d, e, E, f, F, g, G, or i).
 - "+" A sign must always be placed before a number produced by a signed conversion. A + overrides a space if both are used.
- An optional decimal digit string specifying a minimum field width. If the converted value has fewer characters than the field width, it will be padded with spaces on the left (or right, if the left-adjustment flag has been given) to fill out the field width.
- An optional precision, in the form of a period . followed by an optional digit string. If the digit string is omitted, the precision is taken as zero. This gives the minimum number of digits to appear for d, i, o, u, x, and X conversions, the number of digits to appear after the decimal-point for a, A, e, E, f, and F conversions, the maximum number of significant digits for g

and G conversions, or the maximum number of characters to be printed from a string for s conversions.

- A character that specifies the type of conversion to be applied. The conversion specifiers and their meanings are:

- d, i, o, u, x, X The int (or appropriate variant) argument is converted to signed decimal (d and i), unsigned octal (o), unsigned decimal (u), or unsigned hexadecimal (x and X) notation. The letters a, b, c, d, e, f are used for x conversions; the letters A, B, C, D, E, F are used for X conversions. The precision, if any, gives the minimum number of digits that must appear; if the converted value requires fewer digits, it is padded on the left with zeros.
- e, E The double argument is rounded and converted in the style [-]d.ddde+dd where there is one digit before the decimal-point character and the number of digits after it is equal to the precision; if the precision is missing, it is taken as 6; if the precision is zero, no decimal-point character appears. An E conversion uses the letter "E" (rather than "e") to introduce the exponent. The exponent always contains at least two digits; if the value is zero, the exponent is 00.

For a, A, e, E, f, F, g, and G conversions, positive and negative infinity are represented as inf and -inf respectively when using the lowercase conversion character, and INF and -INF respectively when using the uppercase conversion character. Similarly, NaN is represented as nan when using the lowercase conversion, and NAN when using the uppercase conversion.

- f, F The double argument is rounded and converted to decimal notation in the style [-]ddd.ddd, where the number of digits after the decimal-point character is equal to the precision specification. If the precision is missing, it is taken as 6; if the precision is explicitly zero, no decimal-point character appears. If a decimal point appears, at least one digit appears before it.
- g, G The double argument is converted in style f or e (or F or E for G conversions). The precision specifies the number of significant digits. If the precision is missing, 6 digits are given; if the precision is zero, it is treated as 1. Style e is used if the exponent from its conversion is less than -4 or greater than or equal to the precision. Trailing zeros are removed from the fractional part of the result; a decimal point appears only if it is followed by at least one digit.
- c The int argument is converted to an unsigned char, and the resulting character is written.
- p The void * pointer argument is printed in hexadecimal.
- n The number of characters written so far is stored into the integer indicated by the int * (or variant) pointer argument. No argument is converted.
- % A "%\" is written. No argument is converted. The complete conversion specification is "%\%".

Examples

print an integer followed by a newline:

```
int val = 1234;
ziRTKPrintf( "value: %d\n", val );
```

print an integer as 8 digit hex with leading zeros followed by a newline:

```
int val = 1234;
ziRTKPrintf( "value: %08x\n", val );
```

print a float followed by a newline:

```
float val = 1.234;
```

```
ziRTKprintf( "value: %f\n", val );
```

print a float and an integer followed by a newline:

```
float fval = 1.234;  
int ival = 5678;  
ziRTKprintf( "float: %f, int: %d\n", fval, ival );
```

See Also:

[ziRTKWriteString](#), [ziRTKWriteData](#)

ziRTKWriteString

ZI_STATUS ziRTKWriteString (char* Str)

Write a zero-terminated string to stdio.

Parameters:

[in] Str
pointer to the char array

Returns:

- ZI_SUCCESS on success

See Also:

[ziRTKPrintf](#), [ziRTKWriteData](#)

ziRTKWriteData

ZI_STATUS ziRTKWriteData (**unsigned char*** Buffer, **unsigned int** Len)

Write a buffer to stdio.

Parameters:

[in] Buffer
pointer to data

[in] Len
Length of the data

Returns:

- ZI_SUCCESS on success

See Also:

[ziRTKPrintf](#), [ziRTKWriteString](#)

Functions to read Feature Configuration

Enumerations

- enum `ZIRTK_OPTIONS` { ZIRTK_OPTION_NONE,
ZIRTK_OPTION_MF, ZIRTK_OPTION_PLL,
ZIRTK_OPTION_MOD, ZIRTK_OPTION_RTK,
ZIRTK_OPTION_UHS, ZIRTK_OPTION_PID }

- enum `ZIRTK_DEVTYPE` { ZIRTK_DEVTYPE_DEFAULT,
ZIRTK_DEVTYPE_LI, ZIRTK_DEVTYPE_IS,
ZIRTK_DEVTYPE_UNKNOWN }

Functions

- `ZI_STATUS ziRTKFeaturesGetOptions (ZIRTK_OPTIONS* Options)`
Read which options are enabled.

- `ZI_STATUS ziRTKFeaturesGetDevType (ZIRTK_DEVTYPE* DevType)`
Read the device type.

Enumeration Type Documentation

enum ZIRTK_OPTIONS

Enumerator:

- ZIRTK_OPTION_NONE
- ZIRTK_OPTION_MF
- ZIRTK_OPTION_PLL
- ZIRTK_OPTION_MOD
- ZIRTK_OPTION_RTK
- ZIRTK_OPTION_UHS
- ZIRTK_OPTION_PID

enum ZIRTK_DEVTYPE

Enumerator:

- ZIRTK_DEVTYPE_DEFAULT
- ZIRTK_DEVTYPE_LI
- ZIRTK_DEVTYPE_IS
- ZIRTK_DEVTYPE_UNKNOWN

Function Documentation

ziRTKFeaturesGetOptions

ZI_STATUS ziRTKFeaturesGetOptions (ZIRTK_OPTIONS* Options)

Read which options are enabled.

Corresponding server paths:

- DEV123/FEATURES/OPTIONS

Parameters:

[out] Options

Pointer to a ZIRTK_OPTIONS to store the value in. The Values are or'ed together.

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when Kits is NULL

ziRTKFeaturesGetDevType

ZI_STATUS ziRTKFeaturesGetDevType (ZIRTK_DEVTYPE* DevType)

Read the device type.

Corresponding server paths:

- DEV123/FEATURES/DEVTYPE

Parameters:

[out] DevType

Pointer to a ZIRTK_DEVTYPE to store the value in.

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when DevType is NULL

Functions concerning the device

Functions

- [ZI_STATUS ziRTKSystemGetExtClk \(int* ExtClk \)](#)
Gets whether the Clock source is internal or external.
- [ZI_STATUS ziRTKSystemSetExtClk \(int ExtClk \)](#)
Set Clock to internal or external mode.
- [ZI_STATUS ziRTKSystemGetHWRevision \(int* HWRevision \)](#)
Gets the revision-number of the hardware.

Function Documentation

ziRTKSystemGetExtClk

ZI_STATUS ziRTKSystemGetExtClk (int* ExtClk)

Gets whether the Clock source is internal or external.

Corresponding server paths:

- DEV123/SYSTEM/EXTCLK

Parameters:

[in] ExtClk

Pointer to an int to store the value in.

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when ExtClk is NULL

ziRTKSystemSetExtClk

ZI_STATUS ziRTKSystemSetExtClk (int ExtClk)

Set Clock to internal or external mode.

Corresponding server paths:

- DEV123/SYSTEM/EXTCLK

Parameters:

[in] ExtClk
int which contains the value.

Returns:

- ZI_SUCCESS on success

ziRTKSystemGetHWRevision

ZI_STATUS ziRTKSystemGetHWRevision (int* HWRevision)

Gets the revision-number of the hardware.

Corresponding server paths:

- DEV123/SYSTEM/HWREVISION

Parameters:

[in] HWRevision

Pointer to an int to store the value in.

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when HWRevision is NULL

User Registers

The user registers provide an easy way to pass information from the host computer to your RTK program and back. Up to 64 32 bit words can be set by the host computer and read by your RTK program.

Functions

- `unsigned int ziRTKUserRegGetCount ()`
get the count of the user registers.
- `ZI_STATUS ziRTKUserRegGet (unsigned char Index, unsigned int* Value)`
Read flags set by the host PC.
- `ZI_STATUS ziRTKUserRegSet (unsigned char Index, unsigned int Value)`
Write flags set by the host PC.

Function Documentation

ziRTKUserRegGetCount

`unsigned int ziRTKUserRegGetCount ()`

get the count of the user registers.

Returns:

the count of user registers

See Also:

[User Registers](#)

ziRTKUserRegGet

ZI_STATUS ziRTKUserRegGet (unsigned char Index, unsigned int* Value)

Read flags set by the host PC.

Corresponding server paths:

- DEV123/CPUS/0/USERREGS/0
- DEV123/CPUS/0/USERREGS/1
- ...

The user registers are set using ziServer. The provided function is to read those values.

Parameters:

[in] Index
index of the user registers-word (currently 0 - 15)

[in] Value
Pointer to an unsigned int to store the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when the provided index is out of range
- ZI_NULLPTR when Value is NULL

See Also:

[ziRTKUserRegSet](#), User Registers

ziRTKUserRegSet

ZI_STATUS ziRTKUserRegSet (unsigned char Index, unsigned int Value)

Write flags set by the host PC.

Corresponding server paths:

- DEV123/CPUS/0/USERREGS/0
- DEV123/CPUS/0/USERREGS/1
- ...

Parameters:

[in] Index
index of the user registers-word (currently 0 - 15)

[in] Value
the new value to write to the user register

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when the provided index is out of range

See Also:

[ziRTKUserRegGet](#), User Registers

External Memory

The external memory provide extra data memory for applications that either need to log or process large amounts of data.

Functions

- `ZI_STATUS ziRTKExtMemGet (unsigned int Address,
unsigned int* Value)`
Read an address in the external memory.
- `ZI_STATUS ziRTKExtMemSet (unsigned int Address,
unsigned int Value)`
Write an address in the external memory.

Function Documentation

ziRTKExtMemGet

ZI_STATUS ziRTKExtMemGet (unsigned int Address, unsigned int* Value)

Read an address in the external memory.

Parameters:

[in] Address

the byte-aligned address to read (0x00000000 to 0x2FFFFFF)

[in] Value

Pointer to an unsigned int to store the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when the provided index is out of range
- ZI_NULLPTR when Value is NULL

See Also:

[ziRTKExtMemSet, External Memory](#)

ziRTKExtMemSet

ZI_STATUS ziRTKExtMemSet (unsigned int Address, unsigned int Value)

Write an address in the external memory.

Parameters:

[in] Address

the byte-aligned address to write (0x00000000 to 0x2FFFFFF)

[in] Value

the new value to write to the external memory

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when the provided index is out of range

See Also:

[ziRTKExtMemGet, External Memory](#)

HF Signal Input

The functions in this section allow you to set the range and mode of the HF inputs.

Functions

- `unsigned int ziRTKSigInGetCount ()`
get the count of high speed signal inputs.
- `ZI_STATUS ziRTKSigInGetRange (unsigned int Index, float* Range)`
Get an input's range (V).
- `ZI_STATUS ziRTKSigInSetRange (unsigned int Index, float Range)`
Set an input's range(V).
- `ZI_STATUS ziRTKSigInGetAC (unsigned int Index, int* Value)`
Get an input's AC mode.
- `ZI_STATUS ziRTKSigInSetAC (unsigned int Index, int Value)`
Set an input's AC mode.
- `ZI_STATUS ziRTKSigInGetImp50 (unsigned int Index, int* Value)`
Get an input's 50 Ohm mode.
- `ZI_STATUS ziRTKSigInSetImp50 (unsigned int Index, int Value)`
Set an input's 50 Ohm mode.
- `ZI_STATUS ziRTKSigInGetDiff (unsigned int Index, int* Value)`
Get an input's differential mode.
- `ZI_STATUS ziRTKSigInSetDiff (unsigned int Index, int Value)`
Set an input's differential mode.

Function Documentation

ziRTKSigInGetCount

`unsigned int ziRTKSigInGetCount ()`

get the count of high speed signal inputs.

Returns:

The count of signal inputs

See Also:

[HF Signal Input](#)

ziRTKSigInGetRange

ZI_STATUS ziRTKSigInGetRange (unsigned int Index, float* Range)

Get an input's range (V).

Corresponding server paths:

- DEV123/SIGINS/0/RANGE
- DEV123/SIGINS/1/RANGE
- ...

Parameters:

[in] Index
index of the parameter

[out] Range
Pointer to a float to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when Range is NULL

See Also:

[ziRTKSigInSetRange](#), HF Signal Input

ziRTKSigInSetRange**ZI_STATUS ziRTKSigInSetRange (unsigned int Index, float Range)**

Set an input's range(V).

Corresponding server paths:

- DEV123/SIGINS/0/RANGE
- DEV123/SIGINS/1/RANGE
- ...

Parameters:

[in] Index
index of the parameter

[in] Range
float providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKSigInGetRange](#), HF Signal Input

ziRTKSigInGetAC

ZI_STATUS ziRTKSigInGetAC (unsigned int Index, int* Value)

Get an input's AC mode.

Corresponding server paths:

- DEV123/SIGINS/0/AC
- DEV123/SIGINS/1/AC
- ...

Parameters:

[in] Index
index of the parameter

[out] Value
Pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when Value is NULL

See Also:

[ziRTKSigInSetAC](#), HF Signal Input

ziRTKSigInSetAC

ZI_STATUS ziRTKSigInSetAC (unsigned int Index, int Value)

Set an input's AC mode.

Corresponding server paths:

- DEV123/SIGINS/0/AC
- DEV123/SIGINS/1/AC
- ...

Parameters:

[in] Index
index of the parameter

[in] Value
int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKSigInGetAC](#), HF Signal Input

ziRTKSigInGetImp50

ZI_STATUS ziRTKSigInGetImp50 (unsigned int Index, int* Value)

Get an input's 50 Ohm mode.

Corresponding server paths:

- DEV123/SIGINS/0/AC
- DEV123/SIGINS/1/AC
- ...

Parameters:

[in] Index
index of the parameter

[out] Value
Pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when Value is NULL

See Also:

[ziRTKSigInSetImp50](#), HF Signal Input

ziRTKSigInSetImp50

ZI_STATUS ziRTKSigInSetImp50 (**unsigned int** Index, **int** Value)

Set an input's 50 Ohm mode.

Corresponding server paths:

- DEV123/SIGINS/0/AC
- DEV123/SIGINS/1/AC
- ...

Parameters:

[in] Index
index of the parameter

[in] Value
int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKSigInGetImp50](#), HF Signal Input

ziRTKSigInGetDiff

ZI_STATUS ziRTKSigInGetDiff (unsigned int Index, int* Value)

Get an input's differential mode.

Corresponding server paths:

- DEV123/SIGINS/0/AC
- DEV123/SIGINS/1/AC
- ...

Parameters:

[in] Index
index of the parameter

[out] Value
Pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when Value is NULL

See Also:

[ziRTKSigInSetDiff](#), [HF Signal Input](#)

ziRTKSigInSetDiff

ZI_STATUS ziRTKSigInSetDiff (unsigned int Index, int Value)

Set an input's differential mode.

Corresponding server paths:

- DEV123/SIGINS/0/AC
- DEV123/SIGINS/1/AC
- ...

Parameters:

[in] Index
index of the parameter

[in] Value
int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKSigInGetDiff](#), HF Signal Input

Oscillators

The functions in this section allow you to set the frequencies of the oscillators.

Functions

- `unsigned int ziRTK0scGetCount()`
get the count of oscillators.
- `ZI_STATUS ziRTK0scGetFreq(unsigned int Index, float* Freq)`
Get an oscillator's frequency.
- `ZI_STATUS ziRTK0scSetFreq(unsigned int Index, float Freq)`
Set an oscillator's frequency.

Function Documentation

ziRTKOscGetCount

`unsigned int ziRTKOscGetCount ()`

get the count of oscillators.

Returns:

The count of oscillators

See Also:

[Oscillators](#)

ziRTKOscGetFreq

ZI_STATUS ziRTKOscGetFreq (unsigned int Index, float* Freq)

Get an oscillator's frequency.

Corresponding server paths:

- DEV123/OSCS/0/FREQ
- DEV123/OSCS/1/FREQ
- ...

Parameters:

[in] Index
index of the parameter

[out] Freq
Pointer to a float to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range
- ZI_NULLPTR when Freq is NULL

See Also:

[ziRTKOscSetFreq](#), [Oscillators](#)

ziRTKOscSetFreq

ZI_STATUS ziRTKOscSetFreq (unsigned int Index, float Freq)

Set an oscillator's frequency.

Corresponding server paths:

- DEV123/OSCS/0/FREQ
- DEV123/OSCS/1/FREQ
- ...

Parameters:

[in] Index
index of the parameter

[in] Freq
float providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range

See Also:

[ziRTKOscGetFreq](#), Oscillators

Demodulator

The functions in this section allow you to configure the demodulators.

Data Structures

- `struct DemodSample`
type for holding a sample of a demodulator.

Functions

- `unsigned int ziRTKDemodGetCount ()`
get the count of demodulators.
- `ZI_STATUS ziRTKDemodGetADCSelect (unsigned int Index, int* ADCSelect)`
Get a demodulator's adcselect.
- `ZI_STATUS ziRTKDemodSetADCSelect (unsigned int Index, int ADCSelect)`
Set a demodulator's ADC select.
- `ZI_STATUS ziRTKDemodGetOscSelect (unsigned int Index, int* OscSelect)`
Get a demodulator's oscillator select.
- `ZI_STATUS ziRTKDemodSetOscSelect (unsigned int Index, int OscSelect)`
Set a demodulator's oscillator select.
- `ZI_STATUS ziRTKDemodGetOrder (unsigned int Index, int* Order)`
Get a demodulator's filter order.
- `ZI_STATUS ziRTKDemodSetOrder (unsigned int Index, int Order)`
Set a demodulator's filter order.
- `ZI_STATUS ziRTKDemodGetHarmonic (unsigned int Index, int* Harmonic)`
Get a demodulator's harmonic.
- `ZI_STATUS ziRTKDemodSetHarmonic (unsigned int Index, int Harmonic)`
Set a demodulator's harmonic.
- `ZI_STATUS ziRTKDemodGetRate (unsigned int Index, float* Rate)`
Get a demodulator's rate.
- `ZI_STATUS ziRTKDemodSetRate (unsigned int Index, float Rate)`
Set a demodulator's rate.
- `ZI_STATUS ziRTKDemodGetTrigger (unsigned int Index, unsigned int* Trigger)`

- Get a demodulator's trigger- and gating-settings.
- `ZI_STATUS ziRTKDemodSetTrigger (unsigned int Index,
unsigned int Trigger)`
Set a demodulator's trigger- and gating-settings.
- `ZI_STATUS ziRTKDemodGetPhaseShift (unsigned int Index,
float* PhaseShift)`
Get a demodulator's phaseshift.
- `ZI_STATUS ziRTKDemodSetPhaseShift (unsigned int Index,
float PhaseShift)`
Set a demodulator's phaseshift.
- `ZI_STATUS ziRTKDemodGetTimeConstant (unsigned int
Index, float* TimeConstant)`
Get a demodulator's timeconstant.
- `ZI_STATUS ziRTKDemodSetTimeConstant (unsigned int
Index, float TimeConstant)`
Set a demodulator's timeconstant.
- `ZI_STATUS ziRTKDemodGetSinc (unsigned int Index,
unsigned int* Value)`
Get a demodulator's sinc.
- `ZI_STATUS ziRTKDemodSetSinc (unsigned int Index,
unsigned int Value)`
Set a demodulator's sinc.
- `ZI_STATUS ziRTKDemodGetSample (unsigned int Index,
DemodSample* Sample)`
Retrieve a demodulator's sample.
- `ZI_STATUS ziRTKDemodSampleGetTimeStamp (`
`DemodSample* Sample, float* Seconds)`
Returns the timestamp of a `DemodSample` in Seconds.
- `ZI_STATUS ziRTKDemodSampleGetTimeStamp32 (`
`DemodSample* Sample, unsigned int* TS)`
Returns the timestamp of a `DemodSample` as 32bit unsigned int.
- `ZI_STATUS ziRTKDemodSampleGetTimeStamp64 (`
`DemodSample* Sample, unsigned long long* TS)`
Returns the timestamp of a `DemodSample` as 64 bit unsigned long long.
- `ZI_STATUS ziRTKDemodSampleGetX (DemodSample*`
`Sample, float* X)`
Returns the X-Value of a `DemodSample` as float.
- `ZI_STATUS ziRTKDemodSampleGetX32 (DemodSample*`
`Sample, unsigned int* X)`
Returns the X-Value of a `DemodSample` as unsigned int.

- `ZI_STATUS ziRTKDemodSampleGetX64 (DemodSample* Sample, unsigned long long* X)`
Returns the X-Value of a `DemodSample` as unsigned long long.
- `ZI_STATUS ziRTKDemodSampleGetCompX (DemodSample* Sample, float* X)`
Returns the compensated X-Value of a `DemodSample`.
- `ZI_STATUS ziRTKDemodSampleGetY (DemodSample* Sample, float* Y)`
Returns the Y-Value of a `DemodSample` as float.
- `ZI_STATUS ziRTKDemodSampleGetY32 (DemodSample* Sample, unsigned int* Y)`
Returns the Y-Value of a `DemodSample` as unsigned int.
- `ZI_STATUS ziRTKDemodSampleGetY64 (DemodSample* Sample, unsigned long long* Y)`
Returns the Y-Value of a `DemodSample` as unsigned long long.
- `ZI_STATUS ziRTKDemodSampleGetCompY (DemodSample* Sample, float* Y)`
Returns the compensated Y-Value of a `DemodSample`.
- `ZI_STATUS ziRTKDemodSampleGetCompXY (DemodSample* Sample, float* X, float* Y)`
Returns the compensated X-Value and the compensated Y-Value of a `DemodSample`.
- `ZI_STATUS ziRTKDemodSampleGetR (DemodSample* Sample, float* R)`
Returns the Radius-Value of a `DemodSample`.
- `ZI_STATUS ziRTKDemodSampleGetCompR (DemodSample* Sample, float* R)`
Returns the compensated Radius-Value of a `DemodSample`.
- `ZI_STATUS ziRTKDemodSampleGetTheta (DemodSample* Sample, float* Theta)`
Returns the Theta-Value of a `DemodSample`.

Data Structure Documentation

struct DemodSample

type for holding a sample of a demodulator.

```
#include "ziRTK.h"

typedef struct DemodSample {
    TS_t TS;
    Val_t X;
    Val_t Y;
    unsigned int Reserved[3];
} DemodSample;
```

Data Fields

- TS_t TS
Timestamp of the sample where index 0 is the lower word and index 1 is the upper word.
- Val_t X
X-Value of the sample where index 0 is the lower word and index 1 is the upper word.
- Val_t Y
Y-Value of the sample where index 0 is the lower word and index 1 is the upper word.
- unsigned int Reserved
reserved for internal usage

Detailed Description

See Also:

[ziRTKDemodGetSample](#)

Function Documentation

ziRTKDemodGetCount

`unsigned int ziRTKDemodGetCount ()`

get the count of demodulators.

Returns:

The count of demodulators

See Also:

[Demodulator](#)

ziRTKDemodGetADCSelect

ZI_STATUS ziRTKDemodGetADCSelect (unsigned int Index, int* ADCSelect)

Get a demodulator's adcselect.

Corresponding server paths:

- DEV123/DEMOS/0/ADCSELECT
- DEV123/DEMOS/1/ADCSELECT
- ...

Note:

ADCSelect is only available with installed Multi-frequency Option.

Parameters:

[in] Index
index of the Parameter

[out] ADCSelect
pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range
- ZI_NULLPTR when ADCSelect is NULL

See Also:

[ziRTKDemodSetADCSelect](#), [Demodulator](#)

ziRTKDemodSetADCSelect

ZI_STATUS ziRTKDemodSetADCSelect (unsigned int Index, int ADCSelect)

Set a demodulator's ADC select.

Corresponding server paths:

- DEV123/DEMOS/0/ADCSELECT
- DEV123/DEMOS/1/ADCSELECT
- ...

Note:

ADCSelect is only available with installed Multi-frequency Option.

Parameters:

[in] Index
index of the Parameter

[in] ADCSelect
int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range

See Also:

[ziRTKDemodGetADCSelect](#), Demodulator

ziRTKDemodGetOscSelect**ZI_STATUS ziRTKDemodGetOscSelect (unsigned int Index, int* OscSelect)**

Get a demodulator's oscillator select.

Corresponding server paths:

- DEV123/DEMODS/0/OSCSELECT
- DEV123/DEMODS/1/OSCSELECT
- ...

Note:

OscSelect is only available with installed Multi-frequency Option.

Parameters:

[in] Index
index of the Parameter

[out] OscSelect
pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range
- ZI_NULLPTR when ADCSelect is NULL

See Also:

[ziRTKDemodSetOscSelect](#), Demodulator

ziRTKDemodSetOscSelect

ZI_STATUS ziRTKDemodSetOscSelect (unsigned int Index, int OscSelect)

Set a demodulator's oscillator select.

Corresponding server paths:

- DEV123/DEMODS/0/OSCSELECT
- DEV123/DEMODS/1/OSCSELECT
- ...

Note:

OscSelect is only available when the Multi-frequency Option is installed.

Parameters:

[in] Index
index of the Parameter

[in] OscSelect
int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range

See Also:

[ziRTKDemodGetOscSelect](#), Demodulator

ziRTKDemodGetOrder

ZI_STATUS ziRTKDemodGetOrder (unsigned int Index, int* Order)

Get a demodulator's filter order.

Corresponding server paths:

- DEV123/DEMODS/0/ORDER
- DEV123/DEMODS/1/ORDER
- ...

Parameters:

[in] Index
index of the Parameter

[out] Order
pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range
- ZI_NULLPTR when Order is NULL

See Also:

[ziRTKDemodSetOrder](#), Demodulator

ziRTKDemodSetOrder

ZI_STATUS ziRTKDemodSetOrder (unsigned int Index, int Order)

Set a demodulator's filter order.

Corresponding server paths:

- DEV123/DEMODS/0/ORDER
- DEV123/DEMODS/1/ORDER
- ...

Parameters:

[in] Index
index of the Parameter

[in] Order
int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range

See Also:

[ziRTKDemodGetOrder](#), Demodulator

ziRTKDemodGetHarmonic

ZI_STATUS ziRTKDemodGetHarmonic (unsigned int Index, int* Harmonic)

Get a demodulator's harmonic.

Corresponding server paths:

- DEV123/DEMOS/0/ORDER
- DEV123/DEMOS/1/ORDER
- ...

Parameters:

[in] Index
index of the Parameter

[out] Harmonic
pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range
- ZI_NULLPTR when Harmonic is NULL

See Also:

[ziRTKDemodSetHarmonic](#), Demodulator

ziRTKDemodSetHarmonic

ZI_STATUS ziRTKDemodSetHarmonic (unsigned int Index, int Harmonic)

Set a demodulator's harmonic.

Corresponding server paths:

- DEV123/DEMOS/0/ORDER
- DEV123/DEMOS/1/ORDER
- ...

Parameters:

[in] Index
index of the Parameter

[in] Harmonic
int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range

See Also:

[ziRTKDemodGetHarmonic](#), Demodulator

ziRTKDemodGetRate**ZI_STATUS ziRTKDemodGetRate (unsigned int Index, float* Rate)**

Get a demodulator's rate.

Corresponding server paths:

- DEV123/DEMODS/0/RATE
- DEV123/DEMODS/1/RATE
- ...

Parameters:

[in] Index
index of the Parameter

[out] Rate
pointer to a float to write the value in (in Hertz)

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range
- ZI_NULLPTR when Rate is NULL

See Also:

[ziRTKDemodSetRate](#), [Demodulator](#)

ziRTKDemodSetRate

ZI_STATUS ziRTKDemodSetRate (unsigned int Index, float Rate)

Set a demodulator's rate.

Corresponding server paths:

- DEV123/DEMODS/0/RATE
- DEV123/DEMODS/1/RATE
- ...

Parameters:

[in] Index
index of the parameter

[in] Rate
float providing the value (in Hertz)

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range

See Also:

[ziRTKDemodGetRate](#), Demodulator

ziRTKDemodGetTrigger

ZI_STATUS ziRTKDemodGetTrigger (unsigned int Index, unsigned int* Trigger)

Get a demodulator's trigger- and gating-settings.

Corresponding server paths:

- DEV123/DEMODS/0/TRIGGER
- DEV123/DEMODS/1/TRIGGER
- ...

Parameters:

[in] Index
index of the Parameter

[out] Trigger
pointer to an unsigned int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range
- ZI_NULLPTR when Trigger is NULL

See Also:

[ziRTKDemodSetTrigger](#), Demodulator

ziRTKDemodSetTrigger

ZI_STATUS ziRTKDemodSetTrigger (unsigned int Index, unsigned int Trigger)

Set a demodulator's trigger- and gating-settings.

Corresponding server paths:

- DEV123/DEMODS/0/TRIGGER
- DEV123/DEMODS/1/TRIGGER
- ...

Parameters:

[in] Index
index of the parameter

[in] Trigger
unsigned int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range

See Also:

[ziRTKDemodGetTrigger](#), Demodulator

ziRTKDemodGetPhaseShift

ZI_STATUS ziRTKDemodGetPhaseShift (unsigned int Index, float* PhaseShift)

Get a demodulator's phaseshift.

Corresponding server paths:

- DEV123/DEMODS/0/PHASESHIFT
- DEV123/DEMODS/1/PHASESHIFT
- ...

Parameters:

[in] Index
index of the Parameter

[out] PhaseShift
pointer to a float to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range
- ZI_NULLPTR when PhaseShift is NULL

See Also:

[ziRTKDemodSetPhaseShift](#), Demodulator

ziRTKDemodSetPhaseShift

ZI_STATUS ziRTKDemodSetPhaseShift (unsigned int Index, float PhaseShift)

Set a demodulator's phaseshift.

Corresponding server paths:

- DEV123/DEMODS/0/PHASESHIFT
- DEV123/DEMODS/1/PHASESHIFT
- ...

Parameters:

[in] Index
index of the parameter

[in] PhaseShift
float providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range

See Also:

[ziRTKDemodGetPhaseShift](#), Demodulator

ziRTKDemodGetTimeConstant

ZI_STATUS ziRTKDemodGetTimeConstant (unsigned int Index, float* TimeConstant)

Get a demodulator's timeconstant.

Corresponding server paths:

- DEV123/DEMODS/0/TIMECONSTANT
- DEV123/DEMODS/1/TIMECONSTANT
- ...

Parameters:

[in] Index
index of the Parameter

[out] TimeConstant
pointer to a float to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range
- ZI_NULLPTR when TimeConstant is NULL

See Also:

[ziRTKDemodSetTimeConstant](#), [Demodulator](#)

ziRTKDemodSetTimeConstant

ZI_STATUS ziRTKDemodSetTimeConstant (unsigned int Index, float TimeConstant)

Set a demodulator's timeconstant.

Corresponding server paths:

- DEV123/DEMODS/0/TIMECONSTANT
- DEV123/DEMODS/1/TIMECONSTANT
- ...

Parameters:

[in] Index
index of the parameter

[in] TimeConstant
float providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range

See Also:

[ziRTKDemodGetTimeConstant](#), Demodulator

ziRTKDemodGetSinc

ZI_STATUS ziRTKDemodGetSinc (unsigned int Index, unsigned int* Value)

Get a demodulator's sinc.

Corresponding server paths:

- DEV123/DEMODY/0/SINC
- DEV123/DEMODY/1/SINC
- ...

Parameters:

[in] Index
index of the Parameter

[out] Value
pointer to an unsigned int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range
- ZI_NULLPTR when TimeConstant is NULL

See Also:

[ziRTKDemodSetSinc](#), Demodulator

ziRTKDemodSetSinc

ZI_STATUS ziRTKDemodSetSinc (unsigned int Index, unsigned int Value)

Set a demodulator's sinc.

Corresponding server paths:

- DEV123/DEMODY/0/SINC
- DEV123/DEMODY/1/SINC
- ...

Parameters:

[in] Index
index of the parameter

[in] Value
unsigned int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range

See Also:

[ziRTKDemodGetSinc](#), Demodulator

ziRTKDemodGetSample

ZI_STATUS ziRTKDemodGetSample (unsigned int Index, **DemodSample*** Sample)

Retrieve a demodulator's sample.

Corresponding server paths:

- DEV123/DEMODS/0/SAMPLE
- DEV123/DEMODS/1/SAMPLE
- ...

Parameters:

[in] Index
index of the parameter

[out] Sample
DemodSample to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range
- ZI_NULLPTR when the Sample is NULL

See Also:

[Demodulator](#)

ziRTKDemodSampleGetTimeStamp

ZI_STATUS ziRTKDemodSampleGetTimeStamp (**DemodSample*** Sample, **float*** Seconds)

Returns the timestamp of a DemodSample in Seconds.

Parameters:

[in] Sample
the sample to extract the value from

[out] Seconds
pointer to a float in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when Seconds is NULL or Sample is NULL

See Also:

[ziRTKDemodSampleGetTimeStamp32](#), [ziRTKDemodSampleGetTimeStamp64](#), Demodulator

ziRTKDemodSampleGetTimeStamp32

ZI_STATUS ziRTKDemodSampleGetTimeStamp32 (**DemodSample*** Sample, **unsigned int*** TS)

Returns the timestamp of a [DemodSample](#) as 32bit unsigned int.

Note:

you may also read TS.TS32[ZIRTK_LSB_INDEX] of a [DemodSample](#) to get the same result

Parameters:

[in] Sample

the sample to extract the value from

[out] TS

pointer to an unsigned int in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when TS is NULL or Sample is NULL

See Also:

[ziRTKDemodSampleGetTimeStamp](#), [ziRTKDemodSampleGetTimeStamp64](#), Demodulator

ziRTKDemodSampleGetTimeStamp64

ZI_STATUS ziRTKDemodSampleGetTimeStamp64 (**DemodSample*** Sample, **unsigned long long*** TS)

Returns the timestamp of a [DemodSample](#) as 64 bit unsigned long long.

Note:

you may also read TS.TS64 of a [DemodSample](#) to get the same result

Parameters:

[in] Sample

the sample to extract the value from

[out] TS

pointer to an unsigned int in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when TS is NULL or Sample is NULL

See Also:

[ziRTKDemodSampleGetTimeStamp](#), [ziRTKDemodSampleGetTimeStamp32](#), [Demodulator](#)

ziRTKDemodSampleGetX

ZI_STATUS ziRTKDemodSampleGetX (**DemodSample*** Sample, **float*** X)

Returns the X-Value of a DemodSample as float.

Parameters:

[in] Sample
the sample to extract the value from

[out] X
pointer to a float in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when X is NULL or Sample is NULL

See Also:

[ziRTKDemodSampleGetX32](#), [ziRTKDemodSampleGetX64](#), [ziRTKDemodSampleGetCompX](#),
[ziRTKDemodSampleGetY](#), [ziRTKDemodSampleGetY32](#), [ziRTKDemodSampleGetY64](#),
[ziRTKDemodSampleGetCompY](#), [ziRTKDemodSampleGetCompXY](#), [Demodulator](#)

ziRTKDemodSampleGetX32

ZI_STATUS ziRTKDemodSampleGetX32 (**DemodSample*** Sample, **unsigned int*** X)

Returns the X-Value of a **DemodSample** as unsigned int.

Note:

you may also read X.Val32[ZIRTK_MSB_INDEX] of a **DemodSample** to get the same result

Parameters:

[in] Sample

the sample to extract the value from

[out] X

pointer to an unsigned int in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when X is NULL or Sample is NULL

See Also:

[ziRTKDemodSampleGetX](#), [ziRTKDemodSampleGetX64](#), [ziRTKDemodSampleGetCompX](#),
[ziRTKDemodSampleGetY](#), [ziRTKDemodSampleGetY32](#), [ziRTKDemodSampleGetY64](#),
[ziRTKDemodSampleGetCompY](#), [ziRTKDemodSampleGetCompXY](#), Demodulator

ziRTKDemodSampleGetX64

ZI_STATUS ziRTKDemodSampleGetX64 (**DemodSample*** Sample, **unsigned long long*** X)

Returns the X-Value of a **DemodSample** as **unsigned long long**.

Note:

you may also read X.Val64 of a **DemodSample** to get the same result

Parameters:

[in] Sample

the sample to extract the value from

[out] X

pointer to an **unsigned long long** in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when X is NULL or Sample is NULL

See Also:

[ziRTKDemodSampleGetX](#), [ziRTKDemodSampleGetX32](#), [ziRTKDemodSampleGetCompX](#),
[ziRTKDemodSampleGetY](#), [ziRTKDemodSampleGetY32](#), [ziRTKDemodSampleGetY64](#),
[ziRTKDemodSampleGetCompY](#), [ziRTKDemodSampleGetCompXY](#), Demodulator

ziRTKDemodSampleGetCompX

ZI_STATUS ziRTKDemodSampleGetCompX (DemodSample* Sample, float* X)

Returns the compensated X-Value of a DemodSample.

Parameters:

[in] Sample

the sample to extract the value from

[out] X

pointer to a float in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when X is NULL or Sample is NULL

See Also:

[ziRTKDemodSampleGetX](#), [ziRTKDemodSampleGetX32](#), [ziRTKDemodSampleGetX64](#),
[ziRTKDemodSampleGetY](#), [ziRTKDemodSampleGetY32](#), [ziRTKDemodSampleGetY64](#),
[ziRTKDemodSampleGetCompY](#), [ziRTKDemodSampleGetCompXY](#), [Demodulator](#)

ziRTKDemodSampleGetY

ZI_STATUS ziRTKDemodSampleGetY (DemodSample* Sample, float* Y)

Returns the Y-Value of a DemodSample as float.

Parameters:

[in] Sample
the sample to extract the value from

[out] Y
pointer to a float in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when Y is NULL or Sample is NULL

See Also:

[ziRTKDemodSampleGetX](#), [ziRTKDemodSampleGetX32](#), [ziRTKDemodSampleGetX64](#),
[ziRTKDemodSampleGetCompX](#), [ziRTKDemodSampleGetY32](#), [ziRTKDemodSampleGetY64](#),
[ziRTKDemodSampleGetCompY](#), [ziRTKDemodSampleGetCompXY](#), [Demodulator](#)

ziRTKDemodSampleGetY32

ZI_STATUS ziRTKDemodSampleGetY32 (DemodSample* Sample, unsigned int* Y)

Returns the Y-Value of a [DemodSample](#) as unsigned int.

Note:

you may also read Y.Val32[ZIRTK_MSB_INDEX] of a [DemodSample](#) to get the same result

Parameters:

[in] Sample

the sample to extract the value from

[out] Y

pointer to an unsigned int in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when Y is NULL or Sample is NULL

See Also:

[ziRTKDemodSampleGetX](#), [ziRTKDemodSampleGetX32](#), [ziRTKDemodSampleGetX64](#),
[ziRTKDemodSampleGetCompX](#), [ziRTKDemodSampleGetY](#), [ziRTKDemodSampleGetY64](#),
[ziRTKDemodSampleGetCompY](#), [ziRTKDemodSampleGetCompXY](#), [Demodulator](#)

ziRTKDemodSampleGetY64

ZI_STATUS ziRTKDemodSampleGetY64 (**DemodSample*** Sample, **unsigned long long*** Y)

Returns the Y-Value of a **DemodSample** as **unsigned long long**.

Note:

you may also read Y.Val64 of a **DemodSample** to get the same result

Parameters:

[in] Sample

the sample to extract the value from

[out] Y

pointer to an **unsigned long long** in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when Y is NULL or Sample is NULL

See Also:

ziRTKDemodSampleGetX, ziRTKDemodSampleGetX32, ziRTKDemodSampleGetX64,
ziRTKDemodSampleGetCompX, ziRTKDemodSampleGetY, ziRTKDemodSampleGetY32,
ziRTKDemodSampleGetCompY, ziRTKDemodSampleGetCompXY, Demodulator

ziRTKDemodSampleGetCompY

ZI_STATUS ziRTKDemodSampleGetCompY (DemodSample* Sample, float* Y)

Returns the compensated Y-Value of a DemodSample.

Parameters:

[in] Sample

the sample to extract the value from

[out] Y

pointer to a float in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when Y is NULL or Sample is NULL

See Also:

[ziRTKDemodSampleGetX](#), [ziRTKDemodSampleGetX32](#), [ziRTKDemodSampleGetX64](#),
[ziRTKDemodSampleGetCompX](#), [ziRTKDemodSampleGetY](#), [ziRTKDemodSampleGetY32](#),
[ziRTKDemodSampleGetY64](#), [ziRTKDemodSampleGetCompXY](#), [Demodulator](#)

ziRTKDemodSampleGetCompXY

ZI_STATUS ziRTKDemodSampleGetCompXY (DemodSample* Sample, float* X, float* Y)

Returns the compensated X-Value and the compensated Y-Value of a DemodSample.

Parameters:

[in] Sample

the sample to extract the value from

[out] X

pointer to a float in which the X-value will be returned

[out] Y

pointer to a float in which the Y-value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when X is NULL or Y is NULL or Sample is NULL

See Also:

[ziRTKDemodSampleGetX](#), [ziRTKDemodSampleGetX32](#), [ziRTKDemodSampleGetX64](#),
[ziRTKDemodSampleGetCompX](#), [ziRTKDemodSampleGetY](#), [ziRTKDemodSampleGetY32](#),
[ziRTKDemodSampleGetY64](#), [ziRTKDemodSampleGetCompY](#), [Demodulator](#)

ziRTKDemodSampleGetR

ZI_STATUS ziRTKDemodSampleGetR (**DemodSample*** Sample, **float*** R)

Returns the Radius-Value of a DemodSample.

Parameters:

[in] Sample
the sample to extract the value from

[out] R
pointer to a float in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when R is NULL or Sample is NULL

See Also:

[ziRTKDemodSampleGetCompR](#), [ziRTKDemodSampleGetTheta](#), [Demodulator](#)

ziRTKDemodSampleGetCompR

ZI_STATUS ziRTKDemodSampleGetCompR (**DemodSample*** Sample, **float*** R)

Returns the compensated Radius-Value of a DemodSample.

Parameters:

[in] Sample
the sample to extract the value from

[out] R
pointer to a float in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when R is NULL or Sample is NULL

See Also:

[ziRTKDemodSampleGetR](#), [ziRTKDemodSampleGetTheta](#), [Demodulator](#)

ziRTKDemodSampleGetTheta

ZI_STATUS ziRTKDemodSampleGetTheta (DemodSample* Sample, float* Theta)

Returns the Theta-Value of a [DemodSample](#).

Parameters:

[in] Sample
the sample to extract the value from

[out] Theta
pointer to a float in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when Theta is NULL or Sample is NULL

See Also:

[ziRTKDemodSampleGetR](#), [ziRTKDemodSampleGetCompR](#), [Demodulator](#)

HF Signal Output

The functions in this section allow you to set the range and the amplitude of the HF signal output.

Functions

- `unsigned int ziRTKSigOutGetCount ()`
get the count of HF Signal outputs.
- `ZI_STATUS ziRTKSigOutGetRange (unsigned int Index, float* Range)`
Get an output's range (V).
- `ZI_STATUS ziRTKSigOutSetRange (unsigned int Index, float Range)`
Set an output's range (V).
- `unsigned int ziRTKSigOutGetChannelCount ()`
get the count of HF Signal output Mixer Channels.
- `ZI_STATUS ziRTKSigOutGetEnable (unsigned int Index, unsigned int Channel, int* Value)`
Get an output's mixer enable.
- `ZI_STATUS ziRTKSigOutSetEnable (unsigned int Index, unsigned int Channel, int Value)`
Set an output's mixer enable.
- `ZI_STATUS ziRTKSigOutGetAmplitude (unsigned int Index, unsigned int Channel, float* Value)`
Get an output's mixer amplitude.
- `ZI_STATUS ziRTKSigOutSetAmplitude (unsigned int Index, unsigned int Channel, float Value)`
Set an output's mixer amplitude.
- `ZI_STATUS ziRTKSigOutGetOn (unsigned int Index, int* Value)`
Get an output's on-switch.
- `ZI_STATUS ziRTKSigOutSetOn (unsigned int Index, int Value)`
Set an output's on-switch.
- `ZI_STATUS ziRTKSigOutGetAdd (unsigned int Index, int* Value)`
Get an output's add mode.
- `ZI_STATUS ziRTKSigOutSetAdd (unsigned int Index, int Value)`
Set an output's add mode.

Function Documentation

ziRTKSigOutGetCount

`unsigned int ziRTKSigOutGetCount ()`

get the count of HF Signal outputs.

Returns:

The count of the signal outputs.

See Also:

[HF Signal Output](#)

ziRTKSigOutGetRange

ZI_STATUS ziRTKSigOutGetRange (unsigned int Index, float* Range)

Get an output's range (V).

returns an output's range.

Corresponding server paths:

- DEV123/SIGOUTS/0/RANGE
- DEV123/SIGOUTS/1/RANGE
- ...

Parameters:

[in] Index
index of the parameter

[out] Range
pointer to a float to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range
- ZI_NULLPTR when Range is NULL

See Also:

[ziRTKSigOutSetRange](#), HF Signal Output

ziRTKSigOutSetRange

ZI_STATUS ziRTKSigOutSetRange (unsigned int Index, float Range)

Set an output's range (V).

sets an output's range.

Corresponding server paths:

- DEV123/SIGOUTS/0/RANGE
- DEV123/SIGOUTS/1/RANGE
- ...

Parameters:

[in] Index
index of the parameter

[in] Range
float providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range

See Also:

[ziRTKSigOutGetRange](#), HF Signal Output

ziRTKSigOutGetChannelCount

unsigned int ziRTKSigOutGetChannelCount ()

get the count of HF Signal output Mixer Channels.

Returns:

The count of mixer channels

See Also:

[HF Signal Output](#)

ziRTKSigOutGetEnable

ZI_STATUS ziRTKSigOutGetEnable (unsigned int Index, unsigned int Channel, int* Value)

Get an output's mixer enable.

returns an output's mixer enable.

Corresponding server paths:

- DEV123/SIGOUTS/0/ENABLES/0
- DEV123/SIGOUTS/0/ENABLES/1
- ...
- DEV123/SIGOUTS/1/ENABLES/0
- DEV123/SIGOUTS/1/ENABLES/1
- ...

Parameters:

[in] Index
index of the signal output

[in] Channel
index of the mixer-channel

[out] Value
pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when Value is NULL
- ZI_OUTOFRANGE when Index is out of range or channel is out of range

See Also:

[ziRTKSigOutSetEnable](#), HF Signal Output

ziRTKSigOutSetEnable

ZI_STATUS ziRTKSigOutSetEnable (unsigned int Index, unsigned int Channel, int Value)

Set an output's mixer enable.

sets an output's mixer enable.

Corresponding server paths:

- DEV123/SIGOUTS/0/ENABLES/0
- DEV123/SIGOUTS/0/ENABLES/1
- ...
- DEV123/SIGOUTS/1/ENABLES/0
- DEV123/SIGOUTS/1/ENABLES/1
- ...

Parameters:

[in] Index
index of the signal output

[in] Channel
index of the mixer-channel

[in] Value
int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range or channel is out of range

See Also:

[ziRTKSigOutGetEnable](#), [HF Signal Output](#)

ziRTKSigOutGetAmplitude

ZI_STATUS ziRTKSigOutGetAmplitude (unsigned int Index, unsigned int Channel, float* Value)

Get an output's mixer amplitude.

returns an output's mixer amplitude.

Corresponding server paths:

- DEV123/SIGOUTS/0/AMPLITUDES/0
- DEV123/SIGOUTS/0/AMPLITUDES/1
- ...
- DEV123/SIGOUTS/1/AMPLITUDES/0
- DEV123/SIGOUTS/1/AMPLITUDES/1
- ...

Parameters:

[in] Index
index of the signal output

[in] Channel
index of the mixer-channel

[out] Value
pointer to a float to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when Value is NULL
- ZI_OUTOFRANGE when Index is out of range or channel is out of range

See Also:

[ziRTKSigOutSetAmplitude](#), [HF Signal Output](#)

ziRTKSigOutSetAmplitude

ZI_STATUS ziRTKSigOutSetAmplitude (unsigned int Index, unsigned int Channel, float Value)

Set an output's mixer amplitude.

sets an output's mixer amplitude.

Corresponding server paths:

- DEV123/SIGOUTS/0/AMPLITUDES/0
- DEV123/SIGOUTS/0/AMPLITUDES/1
- ...
- DEV123/SIGOUTS/1/AMPLITUDES/0
- DEV123/SIGOUTS/1/AMPLITUDES/1
- ...

Parameters:

[in] Index
index of the signal output

[in] Channel
index of the mixer-channel

[in] Value
float providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range or channel is out of range

See Also:

[ziRTKSigOutGetAmplitude](#), HF Signal Output

ziRTKSigOutGetOn

ZI_STATUS ziRTKSigOutGetOn (unsigned int Index, int* Value)

Get an output's on-switch.

Corresponding server paths:

- DEV123/SIGOUTS/0/ON
- DEV123/SIGOUTS/1/ON
- ...

Parameters:

[in] Index

index of the parameter

[out] Value

Pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when Value is NULL

See Also:

[ziRTKSigOutSetOn](#), [HF Signal Output](#)

ziRTKSigOutSetOn

ZI_STATUS ziRTKSigOutSetOn (unsigned int Index, int Value)

Set an output's on-switch.

Corresponding server paths:

- DEV123/SIGOUTS/0/ON
- DEV123/SIGOUTS/1/ON
- ...

Parameters:

[in] Index
index of the parameter

[in] Value
int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKSigOutGetOn](#), [HF Signal Output](#)

ziRTKSigOutGetAdd

ZI_STATUS ziRTKSigOutGetAdd (unsigned int Index, int* Value)

Get an output's add mode.

Corresponding server paths:

- DEV123/SIGOUTS/0/ON
- DEV123/SIGOUTS/1/ON
- ...

Parameters:

[in] Index

index of the parameter

[out] Value

Pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when Value is NULL

See Also:

[ziRTKSigOutSetAdd](#), HF Signal Output

ziRTKSigOutSetAdd

ZI_STATUS ziRTKSigOutSetAdd (unsigned int Index, int Value)

Set an output's add mode.

Corresponding server paths:

- DEV123/SIGOUTS/0/ON
- DEV123/SIGOUTS/1/ON
- ...

Parameters:

[in] Index
index of the parameter

[in] Value
int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKSigOutGetAdd](#), [HF Signal Output](#)

Auxiliary Analog Outputs

Functions

- `unsigned int ziRTKAuxOutGetCount ()`
get the count of auxiliary analog outputs.
- `ZI_STATUS ziRTKAuxOutGetValue (unsigned int Index, float* Value)`
Get a auxiliary outputs value as float (V).
- `ZI_STATUS ziRTKAuxOutGetOutputSelect (unsigned int Index, int* OutputSelect)`
Get an auxiliary outputs outputselect.
- `ZI_STATUS ziRTKAuxOutSetOutputSelect (unsigned int Index, int OutputSelect)`
Set an auxiliary output's outputselect.
- `ZI_STATUS ziRTKAuxOutGetDemodSelect (unsigned int Index, int* DemodSelect)`
Get an auxiliary output's demodselect.
- `ZI_STATUS ziRTKAuxOutSetDemodSelect (unsigned int Index, int DemodSelect)`
Set an auxiliary output's demodselect.
- `ZI_STATUS ziRTKAuxOutGetOffset (unsigned int Index, float* Offset)`
Get an auxiliary output's offset.
- `ZI_STATUS ziRTKAuxOutSetOffset (unsigned int Index, float Offset)`
Set an auxiliary output's offset.
- `ZI_STATUS ziRTKAuxOutSetOffsetNoUpdate (unsigned int Index, float Offset)`
Set an auxiliary output's offset without updating ziServer.
- `ZI_STATUS ziRTKAuxOutSetOffsetInt (unsigned int Index, int Offset)`
Set an auxiliary output's offset as integer.
- `ZI_STATUS ziRTKAuxOutSetOffsetIntNoUpdate (unsigned int Index, int Offset)`
Set an auxiliary output's offset as integer without updating ziServer.
- `ZI_STATUS ziRTKAuxOutGetScale (unsigned int Index, float* Scale)`
Get an auxiliary output's scale.
- `ZI_STATUS ziRTKAuxOutSetScale (unsigned int Index, float Scale)`
Set an auxiliary output's scale.

Detailed Description

The functions in this section allow you to set voltages on the auxiliary analog outputs.

Function Documentation

ziRTKAuxOutGetCount

`unsigned int ziRTKAuxOutGetCount ()`

get the count of auxiliary analog outputs.

Returns:

The count of auxiliary outputs

See Also:

[Auxiliary Analog Outputs](#)

ziRTKAuxOutGetValue

ZI_STATUS ziRTKAuxOutGetValue (unsigned int Index, float* Value)

Get a auxiliary outputs value as float (V).

Corresponding server paths:

- DEV123/AUXOUTS/0/VALUE
- DEV123/AUXOUTS/1/VALUE
- ...

Parameters:

[in] Index
index of the Parameter

[out] Value
pointer to a float to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when Value is NULL

See Also:

[Auxiliary Analog Outputs](#)

ziRTKAuxOutGetOutputSelect

ZI_STATUS ziRTKAuxOutGetOutputSelect (unsigned int Index, int* OutputSelect)

Get an auxiliary outputs outputselect.

Corresponding server paths:

- DEV123/AUXOUTS/0/OUTPUTSELECT
- DEV123/AUXOUTS/1/OUTPUTSELECT
- ...

Parameters:

[in] Index
index of the Parameter

[out] OutputSelect
pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when OutputSelect is NULL

See Also:

[ziRTKAuxOutSetOutputSelect](#), Auxiliary Analog Outputs

ziRTKAuxOutSetOutputSelect

ZI_STATUS ziRTKAuxOutSetOutputSelect (unsigned int Index, int OutputSelect)

Set an auxiliary output's outputselect.

Corresponding server paths:

- DEV123/AUXOUTS/0/OUTPUTSELECT
- DEV123/AUXOUTS/1/OUTPUTSELECT
- ...

Parameters:

[in] Index
index of the Parameter

[out] OutputSelect
an int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKAuxOutGetOutputSelect](#), Auxiliary Analog Outputs

ziRTKAuxOutGetDemodSelect

ZI_STATUS ziRTKAuxOutGetDemodSelect (unsigned int Index, int* DemodSelect)

Get an auxiliary output's demodselect.

Corresponding server paths:

- DEV123/AUXOUTS/0/DEMODOSELECT
- DEV123/AUXOUTS/1/DEMODOSELECT
- ...

Parameters:

[in] Index
index of the Parameter

[out] DemodSelect
pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when DemodSelect is NULL

See Also:

[ziRTKAuxOutSetDemodSelect](#), Auxiliary Analog Outputs

ziRTKAuxOutSetDemodSelect

ZI_STATUS ziRTKAuxOutSetDemodSelect (unsigned int Index, int DemodSelect)

Set an auxiliary output's demodselect.

Corresponding server paths:

- DEV123/AUXOUTS/0/DEMODOSELECT
- DEV123/AUXOUTS/1/DEMODOSELECT
- ...

Parameters:

[in] Index
index of the Parameter

[out] DemodSelect
an int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKAuxOutGetDemodSelect](#), Auxiliary Analog Outputs

ziRTKAuxOutGetOffset

ZI_STATUS ziRTKAuxOutGetOffset (**unsigned int** Index, **float*** Offset)

Get an auxiliary output's offset.

Corresponding server paths:

- DEV123/AUXOUTS/0/OFFSET
- DEV123/AUXOUTS/1/OFFSET
- ...

Parameters:

[in] Index
index of the Parameter

[out] Offset
pointer to a float to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when Offset is NULL

See Also:

[ziRTKAuxOutSetOffset](#), [ziRTKAuxOutSetOffsetNoUpdate](#),
[ziRTKAuxOutSetOffsetInt](#), [Auxiliary Analog Outputs](#)

[ziRTKAuxOutSetOffsetInt](#),

ziRTKAuxOutSetOffset

ZI_STATUS ziRTKAuxOutSetOffset (**unsigned int** Index, **float** Offset)

Set an auxiliary output's offset.

Corresponding server paths:

- DEV123/AUXOUTS/0/OFFSET
- DEV123/AUXOUTS/1/OFFSET
- ...

Parameters:

[in] Index
index of the Parameter

[out] Offset
a float providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKAuxOutSetOffsetNoUpdate](#), [ziRTKAuxOutSetOffsetInt](#), [ziRTKAuxOutSetOffsetIntNoUpdate](#),
[ziRTKAuxOutGetOffset](#), Auxiliary Analog Outputs

ziRTKAuxOutSetOffsetNoUpdate

ZI_STATUS ziRTKAuxOutSetOffsetNoUpdate (unsigned int Index, float Offset)

Set an auxiliary output's offset without updating ziServer.

Corresponding server paths:

- DEV123/AUXOUTS/0/OFFSET
- DEV123/AUXOUTS/1/OFFSET
- ...

Parameters:

[in] Index
index of the Parameter

[out] Offset
a float providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKAuxOutSetOffset](#), [ziRTKAuxOutSetOffsetInt](#), [ziRTKAuxOutSetOffsetIntNoUpdate](#),
[ziRTKAuxOutGetOffset](#), Auxiliary Analog Outputs

ziRTKAuxOutSetOffsetInt

ZI_STATUS ziRTKAuxOutSetOffsetInt (unsigned int Index, int Offset)

Set an auxiliary output's offset as integer.

Corresponding server paths:

- DEV123/AUXOUTS/0/OFFSET
- DEV123/AUXOUTS/1/OFFSET
- ...

Parameters:

[in] Index
index of the Parameter

[out] Offset
an integer providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKAuxOutSetOffsetInt32](#), [ziRTKAuxOutSetOffset](#), [ziRTKAuxOutSetOffsetNoUpdate](#),
[ziRTKAuxOutGetOffset](#), [Auxiliary Analog Outputs](#)

ziRTKAuxOutSetOffsetIntNoUpdate

ZI_STATUS ziRTKAuxOutSetOffsetIntNoUpdate (unsigned int Index, int Offset)

Set an auxiliary output's offset as integer without updating ziServer.

Corresponding server paths:

- DEV123/AUXOUTS/0/OFFSET
- DEV123/AUXOUTS/1/OFFSET
- ...

Parameters:

[in] Index
index of the Parameter

[out] Offset
an integer providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKAuxOutSetOffsetInt](#), [ziRTKAuxOutSetOffset](#), [ziRTKAuxOutSetOffsetNoUpdate](#),
[ziRTKAuxOutGetOffset](#), Auxiliary Analog Outputs

ziRTKAuxOutGetScale

ZI_STATUS ziRTKAuxOutGetScale (unsigned int Index, float* Scale)

Get an auxiliary output's scale.

Corresponding server paths:

- DEV123/AUXOUTS/0/SCALE
- DEV123/AUXOUTS/1/SCALE
- ...

Parameters:

[in] Index
index of the Parameter

[out] Scale
pointer to a float to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when Scale is NULL

See Also:

[ziRTKAuxOutSetScale](#), Auxiliary Analog Outputs

ziRTKAuxOutSetScale

ZI_STATUS ziRTKAuxOutSetScale (unsigned int Index, float Scale)

Set an auxiliary output's scale.

Corresponding server paths:

- DEV123/AUXOUTS/0/SCALE
- DEV123/AUXOUTS/1/SCALE
- ...

Parameters:

[in] Index
index of the Parameter

[out] Scale
a float providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKAuxOutGetScale](#), Auxiliary Analog Outputs

Auxiliary Analog Inputs

The functions in this section read signals from the 2 auxiliary analog inputs.

Data Structures

- struct [AuxInSample](#)
type for holding a sample of auxiliary input

Functions

- unsigned int [ziRTKAuxInGetCount](#) ()
get the count of Auxiliary Analog Inputs.
- [ZI_STATUS](#) [ziRTKAuxInGetAveraging](#) (unsigned int Index, int* Averaging)
Get a auxiliary input's averaging/sample rate.
- [ZI_STATUS](#) [ziRTKAuxInSetAveraging](#) (unsigned int Index, int Averaging)
Set an auxiliary output's averaging sample rate.
- [ZI_STATUS](#) [ziRTKAuxInGetSample](#) (unsigned int Index, [AuxInSample](#)* Sample)
retrieve a auxiliary input sample
- [ZI_STATUS](#) [ziRTKAuxInSampleGetTimeStamp](#) ([AuxInSample](#)* Sample, float* Seconds)
Returns the timestamp of an [AuxInSample](#) in Seconds.
- [ZI_STATUS](#) [ziRTKAuxInSampleGetTimeStamp32](#) ([AuxInSample](#)* Sample, unsigned int* TS)
Returns the timestamp of an [AuxInSample](#) as 32bit unsigned int.
- [ZI_STATUS](#) [ziRTKAuxInSampleGetTimeStamp64](#) ([AuxInSample](#)* Sample, unsigned long long* TS)
Returns the timestamp of an [AuxInSample](#) as 64 bit unsigned long long.
- [ZI_STATUS](#) [ziRTKAuxInSampleGetValue](#) ([AuxInSample](#)* Sample, unsigned int Channel, float* Value)
Returns a Value of a [AuxInSample](#).
- [ZI_STATUS](#) [ziRTKAuxInSampleGetValue16](#) ([AuxInSample](#)* Sample, unsigned int Channel, short* Value)
Returns a Value of a [AuxInSample](#) as short.

Data Structure Documentation

struct AuxInSample

type for holding a sample of auxiliary input

```
#include "ziRTK.h"

typedef struct AuxInSample {
    TS_t TS;
    unsigned int Values[1];
} AuxInSample;
```

Data Fields

- TS_t TS
Timestamp of the sample where index 0 is the lower word and index 1 is the upper word.
- unsigned int Values
Values of the sample.

Detailed Description

See Also:

[ziRTKAuxInGetSample](#)

Function Documentation

ziRTKAuxInGetCount

`unsigned int ziRTKAuxInGetCount ()`

get the count of Auxiliary Analog Inputs.

Returns:

The count of auxiliary inputs

See Also:

[Auxiliary Analog Inputs](#)

ziRTKAuxInGetAveraging

ZI_STATUS ziRTKAuxInGetAveraging (unsigned int Index, int* Averaging)

Get a auxiliary input's averaging/sample rate.

Corresponding server path:

- DEV123/AUXINS/0/AVERAGING

See [Aux Analog Inputs](#) for details.

Parameters:

[in] Index
index of the Parameter

[out] Averaging
pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when Averaging is NULL

See Also:

[ziRTKAuxInSetAveraging](#), [Auxiliary Analog Inputs](#)

ziRTKAuxInSetAveraging

ZI_STATUS ziRTKAuxInSetAveraging (unsigned int Index, int Averaging)

Set an auxiliary output's averaging sample rate.

Corresponding server path:

- DEV123/AUXINS/0/AVERAGING

See [Aux Analog Inputs](#) for details.

Parameters:

[in] Index
index of the Parameter

[out] Averaging
value to set

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKAuxInGetAveraging](#), [Auxiliary Analog Inputs](#)

ziRTKAuxInGetSample

ZI_STATUS ziRTKAuxInGetSample (unsigned int Index, AuxInSample* Sample)

retrieve a auxiliary input sample

Corresponding server path:

- DEV123/AUXINS/0/SAMPLE

Parameters:

[in] Index
index of the Parameter

[in] Sample
AuxInSample to write the value in.

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when Sample is NULL

See Also:

[Auxiliary Analog Inputs](#)

ziRTKAuxInSampleGetTimeStamp

ZI_STATUS ziRTKAuxInSampleGetTimeStamp ([AuxInSample*](#) Sample, [float*](#) Seconds)

Returns the timestamp of an [AuxInSample](#) in Seconds.

Parameters:

[in] Sample

the sample to extract the value from

[out] Seconds

pointer to a float in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when Seconds is NULL or Sample is NULL

See Also:

[ziRTKAuxInSampleGetTimeStamp32](#), [ziRTKAuxInSampleGetTimeStamp64](#), Auxiliary Analog Inputs

ziRTKAuxInSampleGetTimeStamp32

ZI_STATUS ziRTKAuxInSampleGetTimeStamp32 (**AuxInSample*** Sample, **unsigned int*** TS)

Returns the timestamp of an [AuxInSample](#) as 32bit unsigned int.

Note:

you may also read TS.TS32[ZIRTK_LSB_INDEX] of a [AuxInSample](#) to get the same result

Parameters:

[in] Sample

the sample to extract the value from

[out] TS

pointer to an unsigned int in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when TS is NULL or Sample is NULL

See Also:

[ziRTKAuxInSampleGetTimeStamp](#), [ziRTKAuxInSampleGetTimeStamp64](#), [Auxiliary Analog Inputs](#)

ziRTKAuxInSampleGetTimeStamp64

ZI_STATUS ziRTKAuxInSampleGetTimeStamp64 ([AuxInSample*](#) Sample, [unsigned long long*](#) TS)

Returns the timestamp of an [AuxInSample](#) as 64 bit unsigned long long.

Note:

you may also read TS.TS64 of a [AuxInSample](#) to get the same result

Parameters:

[in] Sample

the sample to extract the value from

[out] TS

pointer to an unsigned long long in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when TS is NULL or Sample is NULL

See Also:

[ziRTKAuxInSampleGetTimeStamp](#), [ziRTKAuxInSampleGetTimeStamp32](#), [Auxiliary Analog Inputs](#)

ziRTKAuxInSampleGetValue

ZI_STATUS ziRTKAuxInSampleGetValue (**AuxInSample*** Sample, **unsigned int** Channel, **float*** Value)

Returns a Value of a [AuxInSample](#).

Parameters:

[in] Sample
the sample to extract the value from

[in] Channel
the channel to retrieve the value from

[out] Value
pointer to a float in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when channel is out of range
- ZI_NULLPTR when Value is NULL or Sample is NULL

See Also:

[ziRTKAuxInSampleGetValue16](#), Auxiliary Analog Inputs

ziRTKAuxInSampleGetValue16

ZI_STATUS ziRTKAuxInSampleGetValue16 ([AuxInSample](#)* Sample, unsigned int Channel, short* Value)

Returns a Value of a [AuxInSample](#) as short.

Parameters:

[in] Sample
the sample to extract the value from

[in] Channel
the channel to retrieve the value from

[out] Value
pointer to a short in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when channel is out of range
- ZI_NULLPTR when Value is NULL or Sample is NULL

See Also:

[ziRTKAuxInSampleGetValue](#), Auxiliary Analog Inputs

Digital Input/Output

The functions in this section allow you to read signals from and write signals to the 32 DIO lines.

Data Structures

- struct [DIOSample](#)
type for holding a sample of dio

Functions

- `unsigned int ziRTKDIOGetCount ()`
get the count of Digital IO.
- `ZI_STATUS ziRTKDIOGetExtClk (unsigned int Index, int* ExtClk)`
Get a DIO's External-Clock setting.
- `ZI_STATUS ziRTKDIOSetExtClk (unsigned int Index, int ExtClk)`
Set a DIO's External-Clock setting.
- `ZI_STATUS ziRTKDIOGetDecimation (unsigned int Index, int* Decimation)`
Get a DIO's Decimation.
- `ZI_STATUS ziRTKDIOSetDecimation (unsigned int Index, int Decimation)`
Set a DIO's Decimation.
- `ZI_STATUS ziRTKDIOGetDrive (unsigned int Index, int* Drive)`
Get a DIO's Drive.
- `ZI_STATUS ziRTKDIOSetDrive (unsigned int Index, int Drive)`
Set a DIO's Drive.
- `ZI_STATUS ziRTKDIOGetOutput (unsigned int Index, unsigned int* Output)`
Get a DIO's Output.
- `ZI_STATUS ziRTKDIOSetOutput (unsigned int Index, unsigned int Output)`
Set a DIO's Output.
- `ZI_STATUS ziRTKDIOSetOutputNoUpdate (unsigned int Index, unsigned int Output)`
Set a DIO's Output without updating ziServer.
- `ZI_STATUS ziRTKDIOGetSample (unsigned int Index, DIOSample* Sample)`
retrieve a dio sample
- `ZI_STATUS ziRTKDIOSampleGetTimeStamp (DIOSample* Sample, float* Seconds)`
Returns the timestamp of a [DIOSample](#) in Seconds.

- **ZI_STATUS** ziRTKDIOSampleGetTimeStamp32 (**DIOSample*** Sample, unsigned int* TS)
Returns the timestamp of a **DIOSample** as 32bit unsigned int.
- **ZI_STATUS** ziRTKDIOSampleGetTimeStamp64 (**DIOSample*** Sample, unsigned long long* TS)
Returns the timestamp of a **DIOSample** as 64 bit unsigned long long.
- **ZI_STATUS** ziRTKDIOSampleGetBits (**DIOSample*** Sample, unsigned int* Bits)
Returns the Bits of a **DIOSample**.

Data Structure Documentation

struct DIOSample

type for holding a sample of dio

```
#include "ziRTK.h"

typedef struct DIOSample {
    TS_t TS;
    unsigned int Bits[1];
} DIOSample;
```

Data Fields

- TS_t TS
Timestamp of the sample where index 0 is the lower word and index 1 is the upper word.
- unsigned int Bits
values of the sample

Detailed Description

See Also:

[ziRTKDIOGetSample](#)

Function Documentation

ziRTKDIOGetCount

`unsigned int ziRTKDIOGetCount ()`

get the count of Digital IO.

Returns:

The count of the DIO

See Also:

[Digital Input/Output](#)

ziRTKDIOGetExtClk

ZI_STATUS ziRTKDIOGetExtClk (unsigned int Index, int* ExtClk)

Get a DIO's External-Clock setting.

Corresponding server path:

- DEV123/DIOS/0/EXTCLK

Parameters:

[in] Index
index of the Parameter

[out] ExtClk
pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when ExtClk is NULL

See Also:

[ziRTKDIOSetExtClk](#), Digital Input/Output

ziRTKDIOSetExtClk

ZI_STATUS ziRTKDIOSetExtClk (unsigned int Index, int ExtClk)

Set a DIO's External-Clock setting.

Corresponding server path:

- DEV123/DIOS/0/EXTCLK

Parameters:

[in] Index
index of the Parameter

[in] ExtClk
int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKDIOGetExtClk](#), [Digital Input/Output](#)

ziRTKDIOGetDecimation

ZI_STATUS ziRTKDIOGetDecimation (unsigned int Index, int* Decimation)

Get a DIO's Decimation.

Corresponding server path:

- DEV123/DIOS/0/DECIMATION

Parameters:

[in] Index
index of the Parameter

[out] Decimation
pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when Decimation is NULL

See Also:

[ziRTKDIOSetDecimation](#), Digital Input/Output

ziRTKDIOSetDecimation

ZI_STATUS ziRTKDIOSetDecimation (unsigned int Index, int Decimation)

Set a DIO's Decimation.

Corresponding server path:

- DEV123/DIOS/0/DECIMATION

Parameters:

[in] Index
index of the Parameter

[in] Decimation
int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKDIOGetDecimation](#), [Digital Input/Output](#)

ziRTKDIOGetDrive

ZI_STATUS ziRTKDIOGetDrive (unsigned int Index, int* Drive)

Get a DIO's Drive.

Corresponding server path:

- DEV123/DIOS/0/DRIVE

Parameters:

[in] Index
index of the Parameter

[out] Drive
pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when Drive is NULL

See Also:

[ziRTKDIOSetDrive](#), [Digital Input/Output](#)

ziRTKDIOSetDrive

ZI_STATUS ziRTKDIOSetDrive (unsigned int Index, int Drive)

Set a DIO's Drive.

Corresponding server path:

- DEV123/DIOS/0/DRIVE

Parameters:

[in] Index
index of the Parameter

[in] Drive
int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKDIOGetDrive](#), [Digital Input/Output](#)

ziRTKDIOGetOutput

ZI_STATUS ziRTKDIOGetOutput (unsigned int Index, unsigned int* Output)

Get a DIO's Output.

Corresponding server path:

- DEV123/DIOS/0/OUTPUT

Parameters:

[in] Index
index of the Parameter

[out] Output
pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when Output is NULL

See Also:

[ziRTKDIOSetOutput](#), Digital Input/Output

ziRTKDIOSetOutput

ZI_STATUS ziRTKDIOSetOutput (unsigned int Index, unsigned int Output)

Set a DIO's Output.

Corresponding server path:

- DEV123/DIOS/0/OUTPUT

Parameters:

[in] Index
index of the Parameter

[in] Output
unsigned int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKDIOGetOutput](#), [ziRTKDIOSetOutputNoUpdate](#), [Digital Input/Output](#)

ziRTKDIOSetOutputNoUpdate

ZI_STATUS ziRTKDIOSetOutputNoUpdate (unsigned int Index, unsigned int Output)

Set a DIO's Output without updating ziServer.

Corresponding server path:

- DEV123/DIOS/0/OUTPUT

Parameters:

[in] Index
index of the Parameter

[in] Output
unsigned int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKDIOSetOutput](#), [Digital Input/Output](#)

ziRTKDIOGetSample

ZI_STATUS ziRTKDIOGetSample (unsigned int Index, **DIOSample*** Sample)

retrieve a dio sample

Corresponding server path:

- DEV123/DIOS/0/INPUT

Parameters:

[in] Index
index of the Parameter

[in] Sample
DIOSample to write the value in.

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when Sample is NULL

See Also:

[Digital Input/Output](#)

ziRTKDIOSampleGetTimeStamp

ZI_STATUS ziRTKDIOSampleGetTimeStamp (DIOSample* Sample, float* Seconds)

Returns the timestamp of a [DIOSample](#) in Seconds.

Parameters:

[in] Sample
the sample to extract the value from

[out] Seconds
pointer to a float in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when Seconds is NULL or Sample is NULL

See Also:

[ziRTKDIOGetSample](#), [ziRTKDIOSampleGetTimeStamp32](#), [ziRTKDIOSampleGetTimeStamp64](#),
[Digital Input/Output](#)

ziRTKDIOSampleGetTimeStamp32

ZI_STATUS ziRTKDIOSampleGetTimeStamp32 (**DIOSample*** Sample, **unsigned int*** TS)

Returns the timestamp of a **DIOSample** as 32bit unsigned int.

Note:

you may also read TS.TS32[ZIRTK_LSB_INDEX] of a **DIOSample** to get the same result

Parameters:

[in] Sample

the sample to extract the value from

[out] TS

pointer to an unsigned int in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when TS is NULL or Sample is NULL

See Also:

[ziRTKDIOGetSample](#), [ziRTKDIOSampleGetTimeStamp](#), [ziRTKDIOSampleGetTimeStamp64](#),
Digital Input/Output

ziRTKDIOSampleGetTimeStamp64

ZI_STATUS ziRTKDIOSampleGetTimeStamp64 (**DIOSample*** Sample, **unsigned long long*** TS)

Returns the timestamp of a [DIOSample](#) as 64 bit unsigned long long.

Note:

you may also read TS.TS64 of a [DIOSample](#) to get the same result

Parameters:

[in] Sample

the sample to extract the value from

[out] TS

pointer to an unsigned long long in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when TS is NULL or Sample is NULL

See Also:

[ziRTKDIOGetSample](#), [ziRTKDIOSampleGetTimeStamp](#), [ziRTKDIOSampleGetTimeStamp32](#),
Digital Input/Output

ziRTKDIOSampleGetBits

ZI_STATUS ziRTKDIOSampleGetBits (**DIOSample*** Sample, **unsigned int*** Bits)

Returns the Bits of a [DIOSample](#).

Note:

you may also read Bits[0] of a [DIOSample](#) to get the same result

Parameters:

[in] Sample

the sample to extract the value from

[out] Bits

pointer to an unsigned int in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when Bits is NULL or Sample is NULL

See Also:

[ziRTKDIOGetSample](#), [Digital Input/Output](#)

PLL

The functions in this section allow you change (some) PLL settings.

Functions

- `ZI_STATUS ziRTKPLLSetADCSelect (unsigned int Index, int ADCSelect)`
Set the ADCSELECT value of a PLL.

Function Documentation

ziRTKPLLSetADCSelect

ZI_STATUS ziRTKPLLSetADCSelect (unsigned int Index, int ADCSelect)

Set the ADCSELECT value of a PLL.

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[PLL](#)

7.2.13. Data Structure Documentation

struct AuxInSample

type for holding a sample of auxiliary input

```
#include "ziRTK.h"

typedef struct AuxInSample {
    TS_t TS;
    unsigned int Values[1];
} AuxInSample;
```

Data Fields

- TS_t TS
Timestamp of the sample where index 0 is the lower word and index 1 is the upper word.
- unsigned int Values
Values of the sample.

Detailed Description

See Also:

[ziRTKAuxInGetSample](#)

struct DemodSample

type for holding a sample of a demodulator.

```
#include "ziRTK.h"

typedef struct DemodSample {
    TS_t TS;
    Val_t X;
    Val_t Y;
    unsigned int Reserved[3];
} DemodSample;
```

Data Fields

- TS_t TS
Timestamp of the sample where index 0 is the lower word and index 1 is the upper word.
- Val_t X
X-Value of the sample where index 0 is the lower word and index 1 is the upper word.
- Val_t Y
Y-Value of the sample where index 0 is the lower word and index 1 is the upper word.
- unsigned int Reserved
reserved for internal usage

Detailed Description

See Also:

[ziRTKDemodGetSample](#)

struct DIOSample

type for holding a sample of dio

```
#include "ziRTK.h"

typedef struct DIOSample {
    TS_t TS;
    unsigned int Bits[1];
} DIOSample;
```

Data Fields

- TS_t TS
Timestamp of the sample where index 0 is the lower word and index 1 is the upper word.
- unsigned int Bits
values of the sample

Detailed Description

See Also:

[ziRTKDIOGetSample](#)

union TS_t

```
typedef union TS_t {
    unsigned int TS32[2];
    unsigned long long TS64;
} TS_t;
```

Data Fields

- unsigned int TS32
- unsigned long long TS64

union Val_t

```
typedef union Val_t {
    unsigned int Val32[2];
    unsigned long long Val64;
} Val_t;
```

Data Fields

- unsigned int Val32
- unsigned long long Val64

7.2.14. File Documentation

File ziRTK.h

Header File for the Zurich Instruments Ltd. Real-time Option running on an embedded CPU.

```
#include "stddef.h"
```

Data Structures

- union `TS_t`
- union `Val_t`
- struct `DemodSample`
type for holding a sample of a demodulator.
- struct `AuxInSample`
type for holding a sample of auxiliary input
- struct `DIOSample`
type for holding a sample of dio

Enumerations

- enum `ZI_STATUS` { `ZI_SUCCESS`, `ZI_OUTOFRANGE`,
`ZI_NULLPTR`, `ZI_LIMIT`, `ZI_MAX_STATUS` }
The `ZI_STATUS` enum is used as the general return value for ziRTK functions.
- enum `ZIRTK_OPTIONS` { `ZIRTK_OPTION_NONE`,
`ZIRTK_OPTION_MF`, `ZIRTK_OPTION_PLL`,
`ZIRTK_OPTION_MOD`, `ZIRTK_OPTION_RTK`,
`ZIRTK_OPTION_UHS`, `ZIRTK_OPTION_PID` }
- enum `ZIRTK_DEVTYPE` { `ZIRTK_DEVTYPE_DEFAULT`,
`ZIRTK_DEVTYPE_LI`, `ZIRTK_DEVTYPE_IS`,
`ZIRTK_DEVTYPE_UNKNOWN` }

Functions

- void `ziRTKInit (void)`
Declaration for the `ziRTKInit` function. This function has to be defined by the user.
- void `ziRTKLoop (void)`
Declaration for the loop function. This function has to be defined by the user.
- float `ziRTKGetTimeStamp ()`

- Returns a the current timestamp in seconds.
- `unsigned int ziRTKGetTimeStamp32 ()`
Returns a the current timestamp in a 32bit unsigned int.
 - `unsigned long long ziRTKGetTimeStamp64 ()`
Returns a the current timestamp in a 64 bit unsigned long long.
 - `float ziRTKGetWorkload ()`
Get the current workload of the CPU.
 - `void ziRTKUpdateHostPCOn (void)`
Enables updating of ziServer when a value changes.
 - `void ziRTKUpdateHostPCOff (void)`
Disables updating of ziServer when a value changes.
 - `unsigned char ziRTKGetUpdateHostPC (void)`
Enables updating of ziServer when a value changes.
 - `void ziRTKPause (int Value)`
Pauses or resumes the updating of all signal-processing related parameters. Useful for synchronizing parameter updates with the RTK.
 - `ZI_STATUS ziRTKAddDemodSampleTrigger (unsigned int Index, ziTriggerId* TriggerId)`
Add a trigger on a demod sample.
 - `ZI_STATUS ziRTKAddDIOSampleTrigger (unsigned int Index, ziTriggerId* TriggerId)`
Add a trigger on a DIO sample.
 - `ZI_STATUS ziRTKAddAuxInSampleTrigger (unsigned int Index, ziTriggerId* TriggerId)`
Add a trigger on an auxiliary input sample.
 - `ZI_STATUS ziRTKAddClockTrigger (unsigned int USec, ziTriggerId* TriggerId)`
Add a trigger on a timer.
 - `ZI_STATUS ziRTKAddUserRegTrigger (unsigned int Index, ziTriggerId* TriggerId)`
Add a trigger on a user-register.
 - `ZI_STATUS ziRTKTestTrigger (ziTriggerId TriggerId, unsigned int* IsSet)`
Test if a trigger has been activated.
 - `ZI_STATUS ziRTKPrintf (char* Fmt, ...)`
printf-compatible print to stdio. The output consists of a string which may include formatting directives.
 - `ZI_STATUS ziRTKWriteString (char* Str)`

Write a zero-terminated string to stdio.

- **ZI_STATUS** ziRTKWriteData (unsigned char* Buffer, unsigned int Len)
Write a buffer to stdio.
- **ZI_STATUS** ziRTKFeaturesGetOptions (ZIRTK_OPTIONS* Options)
Read which options are enabled.
- **ZI_STATUS** ziRTKFeaturesGetDevType (ZIRTK_DEVTYPE* DevType)
Read the device type.
- **ZI_STATUS** ziRTKSystemGetExtClk (int* ExtClk)
Gets whether the Clock source is internal or external.
- **ZI_STATUS** ziRTKSystemSetExtClk (int ExtClk)
Set Clock to internal or external mode.
- **ZI_STATUS** ziRTKSystemGetHWRevision (int* HWRevision)
Gets the revision-number of the hardware.
- unsigned int ziRTKUserRegGetCount ()
get the count of the user registers.
- **ZI_STATUS** ziRTKUserRegGet (unsigned char Index, unsigned int* Value)
Read flags set by the host PC.
- **ZI_STATUS** ziRTKUserRegSet (unsigned char Index, unsigned int Value)
Write flags set by the host PC.
- **ZI_STATUS** ziRTKExtMemGet (unsigned int Address, unsigned int* Value)
Read an address in the external memory.
- **ZI_STATUS** ziRTKExtMemSet (unsigned int Address, unsigned int Value)
Write an address in the external memory.
- unsigned int ziRTKSigInGetCount ()
get the count of high speed signal inputs.
- **ZI_STATUS** ziRTKSigInGetRange (unsigned int Index, float* Range)
Get an input's range (V).
- **ZI_STATUS** ziRTKSigInSetRange (unsigned int Index, float Range)
Set an input's range(V).
- **ZI_STATUS** ziRTKSigInGetAC (unsigned int Index, int* Value)
Get an input's AC mode.

- **ZI_STATUS** `ziRTKSigInSetAC (unsigned int Index, int Value)`
Set an input's AC mode.
- **ZI_STATUS** `ziRTKSigInGetImp50 (unsigned int Index, int* Value)`
Get an input's 50 Ohm mode.
- **ZI_STATUS** `ziRTKSigInSetImp50 (unsigned int Index, int Value)`
Set an input's 50 Ohm mode.
- **ZI_STATUS** `ziRTKSigInGetDiff (unsigned int Index, int* Value)`
Get an input's differential mode.
- **ZI_STATUS** `ziRTKSigInSetDiff (unsigned int Index, int Value)`
Set an input's differential mode.
- `unsigned int ziRTKOscGetCount ()`
get the count of oscillators.
- **ZI_STATUS** `ziRTKOscGetFreq (unsigned int Index, float* Freq)`
Get an oscillator's frequency.
- **ZI_STATUS** `ziRTKOscSetFreq (unsigned int Index, float Freq)`
Set an oscillator's frequency.
- `unsigned int ziRTKDemodGetCount ()`
get the count of demodulators.
- **ZI_STATUS** `ziRTKDemodGetADCSelect (unsigned int Index, int* ADCSelect)`
Get a demodulator's adcselect.
- **ZI_STATUS** `ziRTKDemodSetADCSelect (unsigned int Index, int ADCSelect)`
Set a demodulator's ADC select.
- **ZI_STATUS** `ziRTKDemodGetOscSelect (unsigned int Index, int* OscSelect)`
Get a demodulator's oscillator select.
- **ZI_STATUS** `ziRTKDemodSetOscSelect (unsigned int Index, int OscSelect)`
Set a demodulator's oscillator select.
- **ZI_STATUS** `ziRTKDemodGetOrder (unsigned int Index, int* Order)`
Get a demodulator's filter order.
- **ZI_STATUS** `ziRTKDemodSetOrder (unsigned int Index, int Order)`
Set a demodulator's filter order.
- **ZI_STATUS** `ziRTKDemodGetHarmonic (unsigned int Index, int* Harmonic)`

Get a demodulator's harmonic.

- `ZI_STATUS ziRTKDemodSetHarmonic (unsigned int Index, int Harmonic)`
Set a demodulator's harmonic.
- `ZI_STATUS ziRTKDemodGetRate (unsigned int Index, float* Rate)`
Get a demodulator's rate.
- `ZI_STATUS ziRTKDemodSetRate (unsigned int Index, float Rate)`
Set a demodulator's rate.
- `ZI_STATUS ziRTKDemodGetTrigger (unsigned int Index, unsigned int* Trigger)`
Get a demodulator's trigger- and gating-settings.
- `ZI_STATUS ziRTKDemodSetTrigger (unsigned int Index, unsigned int Trigger)`
Set a demodulator's trigger- and gating-settings.
- `ZI_STATUS ziRTKDemodGetPhaseShift (unsigned int Index, float* PhaseShift)`
Get a demodulator's phaseshift.
- `ZI_STATUS ziRTKDemodSetPhaseShift (unsigned int Index, float PhaseShift)`
Set a demodulator's phaseshift.
- `ZI_STATUS ziRTKDemodGetTimeConstant (unsigned int Index, float* TimeConstant)`
Get a demodulator's timeconstant.
- `ZI_STATUS ziRTKDemodSetTimeConstant (unsigned int Index, float TimeConstant)`
Set a demodulator's timeconstant.
- `ZI_STATUS ziRTKDemodGetSinc (unsigned int Index, unsigned int* Value)`
Get a demodulator's sinc.
- `ZI_STATUS ziRTKDemodSetSinc (unsigned int Index, unsigned int Value)`
Set a demodulator's sinc.
- `ZI_STATUS ziRTKDemodGetSample (unsigned int Index, DemodSample* Sample)`
Retrieve a demodulator's sample.
- `ZI_STATUS ziRTKDemodSampleGetTimeStamp (DemodSample* Sample, float* Seconds)`
Returns the timestamp of a `DemodSample` in Seconds.
- `ZI_STATUS ziRTKDemodSampleGetTimeStamp32 (DemodSample* Sample, unsigned int* TS)`

- Returns the timestamp of a [DemodSample](#) as 32bit unsigned int.
- [**ZI_STATUS** ziRTKDemodSampleGetTimeStamp64 \(\[DemodSample\]\(#\)* Sample, unsigned long long* TS \)](#)
Returns the timestamp of a [DemodSample](#) as 64 bit unsigned long long.
 - [**ZI_STATUS** ziRTKDemodSampleGetX \(\[DemodSample\]\(#\)* Sample, float* X \)](#)
Returns the X-Value of a [DemodSample](#) as float.
 - [**ZI_STATUS** ziRTKDemodSampleGetX32 \(\[DemodSample\]\(#\)* Sample, unsigned int* X \)](#)
Returns the X-Value of a [DemodSample](#) as unsigned int.
 - [**ZI_STATUS** ziRTKDemodSampleGetX64 \(\[DemodSample\]\(#\)* Sample, unsigned long long* X \)](#)
Returns the X-Value of a [DemodSample](#) as unsigned long long.
 - [**ZI_STATUS** ziRTKDemodSampleGetCompX \(\[DemodSample\]\(#\)* Sample, float* X \)](#)
Returns the compensated X-Value of a [DemodSample](#).
 - [**ZI_STATUS** ziRTKDemodSampleGetY \(\[DemodSample\]\(#\)* Sample, float* Y \)](#)
Returns the Y-Value of a [DemodSample](#) as float.
 - [**ZI_STATUS** ziRTKDemodSampleGetY32 \(\[DemodSample\]\(#\)* Sample, unsigned int* Y \)](#)
Returns the Y-Value of a [DemodSample](#) as unsigned int.
 - [**ZI_STATUS** ziRTKDemodSampleGetY64 \(\[DemodSample\]\(#\)* Sample, unsigned long long* Y \)](#)
Returns the Y-Value of a [DemodSample](#) as unsigned long long.
 - [**ZI_STATUS** ziRTKDemodSampleGetCompY \(\[DemodSample\]\(#\)* Sample, float* Y \)](#)
Returns the compensated Y-Value of a [DemodSample](#).
 - [**ZI_STATUS** ziRTKDemodSampleGetCompXY \(\[DemodSample\]\(#\)* Sample, float* X, float* Y \)](#)
Returns the compensated X-Value and the compensated Y-Value of a [DemodSample](#).
 - [**ZI_STATUS** ziRTKDemodSampleGetR \(\[DemodSample\]\(#\)* Sample, float* R \)](#)
Returns the Radius-Value of a [DemodSample](#).
 - [**ZI_STATUS** ziRTKDemodSampleGetCompR \(\[DemodSample\]\(#\)* Sample, float* R \)](#)
Returns the compensated Radius-Value of a [DemodSample](#).
 - [**ZI_STATUS** ziRTKDemodSampleGetTheta \(\[DemodSample\]\(#\)* Sample, float* Theta \)](#)

Returns the Theta-Value of a [DemodSample](#).

- `unsigned int ziRTKSigOutGetCount ()`
get the count of HF Signal outputs.
- `ZI_STATUS ziRTKSigOutGetRange (unsigned int Index, float* Range)`
Get an output's range (V).
- `ZI_STATUS ziRTKSigOutSetRange (unsigned int Index, float Range)`
Set an output's range (V).
- `unsigned int ziRTKSigOutGetChannelCount ()`
get the count of HF Signal output Mixer Channels.
- `ZI_STATUS ziRTKSigOutGetEnable (unsigned int Index, unsigned int Channel, int* Value)`
Get an output's mixer enable.
- `ZI_STATUS ziRTKSigOutSetEnable (unsigned int Index, unsigned int Channel, int Value)`
Set an output's mixer enable.
- `ZI_STATUS ziRTKSigOutGetAmplitude (unsigned int Index, unsigned int Channel, float* Value)`
Get an output's mixer amplitude.
- `ZI_STATUS ziRTKSigOutSetAmplitude (unsigned int Index, unsigned int Channel, float Value)`
Set an output's mixer amplitude.
- `ZI_STATUS ziRTKSigOutGetOn (unsigned int Index, int* Value)`
Get an output's on-switch.
- `ZI_STATUS ziRTKSigOutSetOn (unsigned int Index, int Value)`
Set an output's on-switch.
- `ZI_STATUS ziRTKSigOutGetAdd (unsigned int Index, int* Value)`
Get an output's add mode.
- `ZI_STATUS ziRTKSigOutSetAdd (unsigned int Index, int Value)`
Set an output's add mode.
- `unsigned int ziRTKAuxOutGetCount ()`
get the count of auxiliary analog outputs.
- `ZI_STATUS ziRTKAuxOutGetValue (unsigned int Index, float* Value)`
Get a auxiliary outputs value as float (V).
- `ZI_STATUS ziRTKAuxOutGetOutputSelect (unsigned int Index, int* OutputSelect)`

Get an auxiliary outputs outputselect.

- **ZI_STATUS** ziRTKAuxOutSetOutputSelect (unsigned int Index, int OutputSelect)
Set an auxiliary output's outputselect.
- **ZI_STATUS** ziRTKAuxOutGetDemodSelect (unsigned int Index, int* DemodSelect)
Get an auxiliary output's demodselect.
- **ZI_STATUS** ziRTKAuxOutSetDemodSelect (unsigned int Index, int DemodSelect)
Set an auxiliary output's demodselect.
- **ZI_STATUS** ziRTKAuxOutGetOffset (unsigned int Index, float* Offset)
Get an auxiliary output's offset.
- **ZI_STATUS** ziRTKAuxOutSetOffset (unsigned int Index, float Offset)
Set an auxiliary output's offset.
- **ZI_STATUS** ziRTKAuxOutSetOffsetNoUpdate (unsigned int Index, float Offset)
Set an auxiliary output's offset without updating ziServer.
- **ZI_STATUS** ziRTKAuxOutSetOffsetInt (unsigned int Index, int Offset)
Set an auxiliary output's offset as integer.
- **ZI_STATUS** ziRTKAuxOutSetOffsetIntNoUpdate (unsigned int Index, int Offset)
Set an auxiliary output's offset as integer without updating ziServer.
- **ZI_STATUS** ziRTKAuxOutGetScale (unsigned int Index, float* Scale)
Get an auxiliary output's scale.
- **ZI_STATUS** ziRTKAuxOutSetScale (unsigned int Index, float Scale)
Set an auxiliary output's scale.
- **unsigned int** ziRTKAuxInGetCount ()
get the count of Auxiliary Analog Inputs.
- **ZI_STATUS** ziRTKAuxInGetAveraging (unsigned int Index, int* Averaging)
Get a auxiliary input's averaging/sample rate.
- **ZI_STATUS** ziRTKAuxInSetAveraging (unsigned int Index, int Averaging)
Set an auxiliary output's averaging sample rate.
- **ZI_STATUS** ziRTKAuxInGetSample (unsigned int Index, AuxInSample* Sample)

- retrieve a auxiliary input sample
- **ZI_STATUS** ziRTKAuxInSampleGetTimeStamp ([AuxInSample*](#) Sample, float* Seconds)
Returns the timestamp of an [AuxInSample](#) in Seconds.
- **ZI_STATUS** ziRTKAuxInSampleGetTimeStamp32 ([AuxInSample*](#) Sample, unsigned int* TS)
Returns the timestamp of an [AuxInSample](#) as 32bit unsigned int.
- **ZI_STATUS** ziRTKAuxInSampleGetTimeStamp64 ([AuxInSample*](#) Sample, unsigned long long* TS)
Returns the timestamp of an [AuxInSample](#) as 64 bit unsigned long long.
- **ZI_STATUS** ziRTKAuxInSampleGetValue ([AuxInSample*](#) Sample, unsigned int Channel, float* Value)
Returns a Value of a [AuxInSample](#).
- **ZI_STATUS** ziRTKAuxInSampleGetValue16 ([AuxInSample*](#) Sample, unsigned int Channel, short* Value)
Returns a Value of a [AuxInSample](#) as short.
- unsigned int ziRTKDIOGetCount ()
get the count of Digital IO.
- **ZI_STATUS** ziRTKDIOGetExtClk (unsigned int Index, int* ExtClk)
Get a DIO's External-Clock setting.
- **ZI_STATUS** ziRTKDIOSetExtClk (unsigned int Index, int ExtClk)
Set a DIO's External-Clock setting.
- **ZI_STATUS** ziRTKDIOGetDecimation (unsigned int Index, int* Decimation)
Get a DIO's Decimation.
- **ZI_STATUS** ziRTKDIOSetDecimation (unsigned int Index, int Decimation)
Set a DIO's Decimation.
- **ZI_STATUS** ziRTKDIOGetDrive (unsigned int Index, int* Drive)
Get a DIO's Drive.
- **ZI_STATUS** ziRTKDIOSetDrive (unsigned int Index, int Drive)
Set a DIO's Drive.
- **ZI_STATUS** ziRTKDIOGetOutput (unsigned int Index, unsigned int* Output)
Get a DIO's Output.
- **ZI_STATUS** ziRTKDIOSetOutput (unsigned int Index, unsigned int Output)
Set a DIO's Output.

- **ZI_STATUS** `ziRTKDIOSetOutputNoUpdate (unsigned int Index, unsigned int Output)`
Set a DIO's Output without updating ziServer.
- **ZI_STATUS** `ziRTKDIOGetSample (unsigned int Index, DIOSample* Sample)`
retrieve a dio sample
- **ZI_STATUS** `ziRTKDIOSampleGetTimeStamp (DIOSample* Sample, float* Seconds)`
Returns the timestamp of a `DIOSample` in Seconds.
- **ZI_STATUS** `ziRTKDIOSampleGetTimeStamp32 (DIOSample* Sample, unsigned int* TS)`
Returns the timestamp of a `DIOSample` as 32bit unsigned int.
- **ZI_STATUS** `ziRTKDIOSampleGetTimeStamp64 (DIOSample* Sample, unsigned long long* TS)`
Returns the timestamp of a `DIOSample` as 64 bit unsigned long long.
- **ZI_STATUS** `ziRTKDIOSampleGetBits (DIOSample* Sample, unsigned int* Bits)`
Returns the Bits of a `DIOSample`.
- **ZI_STATUS** `ziRTKPLLSetADCSelect (unsigned int Index, int ADCSelect)`
Set the ADCSELECT value of a PLL.

Data Structure Documentation

union TS_t

```
typedef union TS_t {
    unsigned int TS32[2];
    unsigned long long TS64;
} TS_t;
```

Data Fields

- unsigned int TS32
- unsigned long long TS64

union Val_t

```
typedef union Val_t {  
    unsigned int Val32[2];  
    unsigned long long Val64;  
} Val_t;
```

Data Fields

- unsigned int Val32
- unsigned long long Val64

struct DemodSample

type for holding a sample of a demodulator.

```
#include "ziRTK.h"

typedef struct DemodSample {
    TS_t TS;
    Val_t X;
    Val_t Y;
    unsigned int Reserved[3];
} DemodSample;
```

Data Fields

- TS_t TS
Timestamp of the sample where index 0 is the lower word and index 1 is the upper word.
- Val_t X
X-Value of the sample where index 0 is the lower word and index 1 is the upper word.
- Val_t Y
Y-Value of the sample where index 0 is the lower word and index 1 is the upper word.
- unsigned int Reserved
reserved for internal usage

Detailed Description**See Also:**

[ziRTKDemodGetSample](#)

struct AuxInSample

type for holding a sample of auxiliary input

```
#include "ziRTK.h"

typedef struct AuxInSample {
    TS_t TS;
    unsigned int Values[1];
} AuxInSample;
```

Data Fields

- TS_t TS
Timestamp of the sample where index 0 is the lower word and index 1 is the upper word.
- unsigned int Values
Values of the sample.

Detailed Description**See Also:**

[ziRTKAuxInGetSample](#)

struct DIOSample

type for holding a sample of dio

```
#include "ziRTK.h"

typedef struct DIOSample {
    TS_t TS;
    unsigned int Bits[1];
} DIOSample;
```

Data Fields

- TS_t TS
Timestamp of the sample where index 0 is the lower word and index 1 is the upper word.
- unsigned int Bits
values of the sample

Detailed Description**See Also:**

[ziRTKDIOGetSample](#)

Enumeration Type Documentation

enum ZI_STATUS

The ZI_STATUS enum is used as the general return value for ziRTK functions.

Enumerator:

- ZI_SUCCESS
Success (no error)
- ZI_OUTOFRANGE
a provided parameter (normally an index) is out of range
- ZI_NULLPTR
a provided parameter is NULL
- ZI_LIMIT
a limit has been reached
- ZI_MAX_STATUS

enum ZIRTK_OPTIONS

Enumerator:

- ZIRTK_OPTION_NONE
- ZIRTK_OPTION_MF
- ZIRTK_OPTION_PLL
- ZIRTK_OPTION_MOD
- ZIRTK_OPTION_RTK
- ZIRTK_OPTION_UHS
- ZIRTK_OPTION_PID

enum ZIRTK_DEVTYPE

Enumerator:

- ZIRTK_DEVTYPE_DEFAULT
- ZIRTK_DEVTYPE_LI
- ZIRTK_DEVTYPE_IS
- ZIRTK_DEVTYPE_UNKNOWN

Function Documentation

ziRTKInit

void ziRTKInit (void)

Declaration for the ziRTKInit function. This function has to be defined by the user.

Used to initialize all settings for the current application e.g. add triggers or set initial values.

```
#include <ziRTK.h>

void ziRTKInit()
{
    // Add a clock trigger at an interval of 1 millisecond.
    ziRTKAddClockTrigger( 1000, NULL );

    // Set the rate of demodulator 0 to 1 Hz.
    ziRTKDemodSetRate( 0, 1 );

    // Write a "Hello" message to usb.
    ziRTKWriteString( "Hello" );
}
```

See Also:

[ziRTKLoop](#), [ziRTKAddClockTrigger](#), [ziRTKAddDemodSampleTrigger](#), [ziRTKAddDIOSampleTrigger](#),
[ziRTKAddAuxInSampleTrigger](#), [ziRTKAddUserRegTrigger](#)

ziRTKLoop

```
void ziRTKLoop ( void )
```

Declaration for the loop function. This function has to be defined by the user.

This function will be executed every time a trigger occurs. Use ziRTKTestTrigger to test which triggers caused the function to be executed.

```
#include <math.h>
#include <ziRTK.h>

// Output the Magnitude and Phase of demodulator 0 on auxiliary output 0 and
// auxiliary output 1.

DemodSample Sample;

void ziRTKLoop()
{
    //retrieve sample
    ziRTKDemodGetSample( 0, &Sample );

    //calculate magnitude
    float Mag;
    ziRTKDemodSampleGetCompR( &Sample, &Mag );

    //calculate phase
    float Phi;
    ziRTKDemodSampleGetTheta( &Sample, &Phi );

    //put the values on the misc analog outputs
    ziRTKAuxOutSetOffset( 0, Mag * 10 );
    ziRTKAuxOutSetOffset( 1, Phi * 10 );

}
```

See Also:

[ziRTKInit](#), [ziRTKTestTrigger](#)

ziRTKGetTimeStamp

float ziRTKGetTimeStamp ()

Returns a the current timestamp in seconds.

Returns:

the timestamp as a float

See Also:

[ziRTKGetTimeStamp32](#), [ziRTKGetTimeStamp64](#)

ziRTKGetTimeStamp32

unsigned int ziRTKGetTimeStamp32 ()

Returns a the current timestamp in a 32bit unsigned int.

Returns:

the timestamp as an unsigned int

See Also:

[ziRTKGetTimeStamp](#), [ziRTKGetTimeStamp64](#)

ziRTKGetTimeStamp64

`unsigned long long ziRTKGetTimeStamp64()`

Returns a the current timestamp in a 64 bit unsigned long long.

Returns:

the timestamp as an unsigned long long

See Also:

[ziRTKGetTimeStamp](#), [ziRTKGetTimeStamp32](#)

ziRTKGetWorkload

float ziRTKGetWorkload ()

Get the current workload of the CPU.

Returns:

the current workload as a float value (0 - 1)

ziRTKUpdateHostPCOn

void ziRTKUpdateHostPCOn (void)

Enables updating of ziServer when a value changes.

See Also:

[ziRTKUpdateHostPCOff](#), [ziRTKGetUpdateHostPC](#)

ziRTKUpdateHostPCOff

void ziRTKUpdateHostPCOff (void)

Disables updating of ziServer when a value changes.

See Also:

[ziRTKUpdateHostPCOn](#), [ziRTKGetUpdateHostPC](#)

ziRTKGetUpdateHostPC

`unsigned char ziRTKGetUpdateHostPC (void)`

Enables updating of ziServer when a value changes.

Returns:

- 0 when updating of ziServer is disabled
- 1 when updating of ziServer is enabled

See Also:

[ziRTKUpdateHostPCOn](#), [ziRTKUpdateHostPCOff](#)

ziRTKPause

void ziRTKPause (int Value)

Pauses or resumes the updating of all signal-processing related parameters. Useful for synchronizing parameter updates with the RTK.

Parameters:

[in] Value

when Value is not zero the updating is paused otherwise all updates are running

ziRTKAddDemodSampleTrigger

ZI_STATUS ziRTKAddDemodSampleTrigger (unsigned int Index, ziTriggerId* TriggerId)

Add a trigger on a demod sample.

ziRTKAddDemodSampleTrigger adds a trigger on a demodulator sample.

Parameters:

[in] Index

index of the demodulator

[out] TriggerId

Pointer to a ziTriggerId returning the id of the newly added trigger. If not used pass NULL.

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when the index is out of range
- ZI_LIMIT when no more triggers are available

See Also:

[ziRTKAddDIOSampleTrigger](#), [ziRTKAddAuxInSampleTrigger](#),

[ziRTKAddClockTrigger](#),

[ziRTKAddUserRegTrigger](#), [ziRTKTestTrigger](#)

ziRTKAddDIOSampleTrigger

ZI_STATUS ziRTKAddDIOSampleTrigger (unsigned int Index, ziTriggerId* TriggerId)

Add a trigger on a DIO sample.

ziRTKAddDIOSampleTrigger adds a trigger on a DIO sample.

Parameters:

[in] Index

index of the DIO

[out] TriggerId

Pointer to a ziTriggerId returning the id of the newly added trigger. If not used pass NULL.

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when the index is out of range
- ZI_LIMIT when no more triggers are available

See Also:

[ziRTKAddDemodSampleTrigger](#), [ziRTKAddAuxInSampleTrigger](#), [ziRTKAddClockTrigger](#),
[ziRTKAddUserRegTrigger](#), [ziRTKTestTrigger](#)

ziRTKAddAuxInSampleTrigger

ZI_STATUS ziRTKAddAuxInSampleTrigger (unsigned int Index, ziTriggerId* TriggerId)

Add a trigger on an auxiliary input sample.

ziRTKAddAuxInSampleTrigger adds a trigger on an auxiliary input sample.

Parameters:

[in] Index

index of the auxiliary input

[out] TriggerId

Pointer to a ziTriggerId returning the id of the newly added trigger. If not used pass NULL.

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when the index is out of range
- ZI_LIMIT when no more triggers are available

See Also:

[ziRTKAddDemodSampleTrigger](#), [ziRTKAddDIOSSampleTrigger](#), [ziRTKAddClockTrigger](#),
[ziRTKAddUserRegTrigger](#), [ziRTKTestTrigger](#)

ziRTKAddClockTrigger

ZI_STATUS ziRTKAddClockTrigger (unsigned int USec, ziTriggerId* TriggerId)

Add a trigger on a timer.

ziRTKAddClockTrigger adds a trigger that occurs in an interval

Parameters:

[in] USec

Trigger interval in microseconds

[out] TriggerId

Pointer to a ziTriggerId returning the id of the newly added trigger. If not used pass NULL.

Returns:

- ZI_SUCCESS on success
- ZI_LIMIT when no more triggers are available

See Also:

[ziRTKAddDemodSampleTrigger](#), [ziRTKAddDIOSampleTrigger](#), [ziRTKAddAuxInSampleTrigger](#),
[ziRTKAddUserRegTrigger](#), [ziRTKTestTrigger](#)

ziRTKAddUserRegTrigger

ZI_STATUS ziRTKAddUserRegTrigger (unsigned int Index, ziTriggerId* TriggerId)

Add a trigger on a user-register.

ziRTKAddUserRegTrigger adds a trigger that occurs on a change of a user-register

Parameters:

[in] Index

Index of the register (currently 0 - 15)

[out] TriggerId

Pointer to a ziTriggerId returning the id of the newly added trigger. If not used pass NULL.

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_LIMIT no more triggers are available

See Also:

[ziRTKAddDemodSampleTrigger](#), [ziRTKAddDIOSampleTrigger](#), [ziRTKAddAuxInSampleTrigger](#),
[ziRTKAddClockTrigger](#), [ziRTKTestTrigger](#)

ziRTKTestTrigger

ZI_STATUS ziRTKTestTrigger (ziTriggerId TriggerId, unsigned int* IsSet)

Test if a trigger has been activated.

In case you have added more than one trigger, you can use this function in [ziRTKLoop](#) to determine which of the triggers were active

Parameters:

[in] TriggerId

the identifier for which the state should be returned

[out] IsSet

Pointer to an unsigned int which is not equal to zero, when the trigger is active, else zero

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when TriggerId is out of range

See Also:

[ziRTKAddDemodSampleTrigger](#), [ziRTKAddDIOSampleTrigger](#), [ziRTKAddAuxInSampleTrigger](#),
[ziRTKAddClockTrigger](#), [ziRTKAddUserRegTrigger](#)

ziRTKPrintf**ZI_STATUS ziRTKPrintf (char* Fmt, ...)**

printf-compatible print to stdio. The output consists of a string which may include formatting directives.

Parameters:

[in] Fmt

format string composed of zero or more formatting directives and ordinary characters

[in] ...

variable count of arguments being formatted and given out according to the format string

Returns:

- ZI_SUCCESS on success

This function description is based on the Unix man pages for printf.

The format string is composed of zero or more directives: ordinary characters (not %), which are copied unchanged to the output stream; and conversion specifications, each of which results in fetching zero or more subsequent arguments. Each conversion specification is introduced by the % character.

The arguments must correspond properly (after type promotion) with the conversion specifier. All integer values (signed and unsigned) have to be 32bit values, all floating point values have to be double but standard type promotion automatically converts float values to double. Pointer arguments also have to be 32bit wide, char-arrays as arguments are not supported.

After the %, the following appear in sequence:

- Zero or more of the following flags:
 - "0" (zero) Zero padding. For all conversions except n, the converted value is padded on the left with zeros rather than blanks. If a precision is given with a numeric conversion (d, i, o, u, x, and X), the 0 flag is ignored.
 - "-" A negative field width flag; the converted value is to be left adjusted on the field boundary. Except for n conversions, the converted value is padded on the right with blanks, rather than on the left with blanks or zeros. A - overrides a 0 if both are given.
 - " " (space) A blank should be left before a positive number produced by a signed conversion (a, A, d, e, E, f, F, g, G, or i).
 - "+" A sign must always be placed before a number produced by a signed conversion. A + overrides a space if both are used.
- An optional decimal digit string specifying a minimum field width. If the converted value has fewer characters than the field width, it will be padded with spaces on the left (or right, if the left-adjustment flag has been given) to fill out the field width.
- An optional precision, in the form of a period . followed by an optional digit string. If the digit string is omitted, the precision is taken as zero. This gives the minimum number of digits to appear for d, i, o, u, x, and X conversions, the number of digits to appear after the decimal-point for a, A, e, E, f, and F conversions, the maximum number of significant digits for g and G conversions, or the maximum number of characters to be printed from a string for s conversions.

- A character that specifies the type of conversion to be applied. The conversion specifiers and their meanings are:
 - **d, i, o, u, x, X** The int (or appropriate variant) argument is converted to signed decimal (d and i), unsigned octal (o), unsigned decimal (u), or unsigned hexadecimal (x and X) notation. The letters a, b, c, d, e, f are used for x conversions; the letters A, B, C, D, E, F are used for X conversions. The precision, if any, gives the minimum number of digits that must appear; if the converted value requires fewer digits, it is padded on the left with zeros.
 - **e, E** The double argument is rounded and converted in the style [-]d.ddde+-dd where there is one digit before the decimal-point character and the number of digits after it is equal to the precision; if the precision is missing, it is taken as 6; if the precision is zero, no decimal-point character appears. An E conversion uses the letter "E" (rather than "e") to introduce the exponent. The exponent always contains at least two digits; if the value is zero, the exponent is 00.

For a, A, e, E, f, F, g, and G conversions, positive and negative infinity are represented as inf and -inf respectively when using the lowercase conversion character, and INF and -INF respectively when using the uppercase conversion character. Similarly, NaN is represented as nan when using the lowercase conversion, and NAN when using the uppercase conversion.

- **f, F** The double argument is rounded and converted to decimal notation in the style [-]ddd.ddd, where the number of digits after the decimal-point character is equal to the precision specification. If the precision is missing, it is taken as 6; if the precision is explicitly zero, no decimal-point character appears. If a decimal point appears, at least one digit appears before it.
- **g, G** The double argument is converted in style f or e (or F or E for G conversions). The precision specifies the number of significant digits. If the precision is missing, 6 digits are given; if the precision is zero, it is treated as 1. Style e is used if the exponent from its conversion is less than -4 or greater than or equal to the precision. Trailing zeros are removed from the fractional part of the result; a decimal point appears only if it is followed by at least one digit.
- **c** The int argument is converted to an unsigned char, and the resulting character is written.
- **p** The void * pointer argument is printed in hexadecimal.
- **n** The number of characters written so far is stored into the integer indicated by the int * (or variant) pointer argument. No argument is converted.
- **%** A "%" is written. No argument is converted. The complete conversion specification is "%\%".

Examples

print an integer followed by a newline:

```
int val = 1234;
ziRTKPrintf( "value: %d\n", val );
```

print an integer as 8 digit hex with leading zeros followed by a newline:

```
int val = 1234;
ziRTKPrintf( "value: %08x\n", val );
```

print a float followed by a newline:

```
float val = 1.234;
ziRTKPrintf( "value: %f\n", val );
```

print a float and an integer followed by a newline:

```
float fval = 1.234;
int ival = 5678;
ziRTKPrintf( "float: %f, int: %d\n", fval, ival );
```

See Also:

[ziRTKWriteString](#), [ziRTKWriteData](#)

ziRTKWriteString

ZI_STATUS ziRTKWriteString (char* Str)

Write a zero-terminated string to stdio.

Parameters:

[in] Str
pointer to the char array

Returns:

- ZI_SUCCESS on success

See Also:

[ziRTKPrintf](#), [ziRTKWriteData](#)

ziRTKWriteData

ZI_STATUS ziRTKWriteData (**unsigned char*** Buffer, **unsigned int** Len)

Write a buffer to stdio.

Parameters:

[in] Buffer
pointer to data

[in] Len
Length of the data

Returns:

- ZI_SUCCESS on success

See Also:

[ziRTKPrintf](#), [ziRTKWriteString](#)

ziRTKFeaturesGetOptions

ZI_STATUS ziRTKFeaturesGetOptions (ZIRTK_OPTIONS* Options)

Read which options are enabled.

Corresponding server paths:

- DEV123/FEATURES/OPTIONS

Parameters:

[out] Options

Pointer to a ZIRTK_OPTIONS to store the value in. The Values are or'ed together.

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when Kits is NULL

ziRTKFeaturesGetDevType

ZI_STATUS ziRTKFeaturesGetDevType (ZIRTK_DEVTYPE* DevType)

Read the device type.

Corresponding server paths:

- DEV123/FEATURES/DEVTYPE

Parameters:

[out] DevType

Pointer to a ZIRTK_DEVTYPE to store the value in.

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when DevType is NULL

ziRTKSystemGetExtClk

ZI_STATUS ziRTKSystemGetExtClk (int* ExtClk)

Gets whether the Clock source is internal or external.

Corresponding server paths:

- DEV123/SYSTEM/EXTCLK

Parameters:

[in] ExtClk

Pointer to an int to store the value in.

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when ExtClk is NULL

ziRTKSystemSetExtClk

ZI_STATUS ziRTKSystemSetExtClk (int ExtClk)

Set Clock to internal or external mode.

Corresponding server paths:

- DEV123/SYSTEM/EXTCLK

Parameters:

[in] ExtClk
int which contains the value.

Returns:

- ZI_SUCCESS on success

ziRTKSystemGetHWRevision

ZI_STATUS ziRTKSystemGetHWRevision (int* HWRevision)

Gets the revision-number of the hardware.

Corresponding server paths:

- DEV123/SYSTEM/HWREVISION

Parameters:

[in] HWRevision

Pointer to an int to store the value in.

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when HWRevision is NULL

ziRTKUserRegGetCount

unsigned int ziRTKUserRegGetCount ()

get the count of the user registers.

Returns:

the count of user registers

See Also:

[User Registers](#)

ziRTKUserRegGet

ZI_STATUS ziRTKUserRegGet (unsigned char Index, unsigned int* Value)

Read flags set by the host PC.

Corresponding server paths:

- DEV123/CPUS/0/USERREGS/0
- DEV123/CPUS/0/USERREGS/1
- ...

The user registers are set using ziServer. The provided function is to read those values.

Parameters:

[in] Index
index of the user registers-word (currently 0 - 15)

[in] Value
Pointer to an unsigned int to store the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when the provided index is out of range
- ZI_NULLPTR when Value is NULL

See Also:

[ziRTKUserRegSet](#), User Registers

ziRTKUserRegSet

ZI_STATUS ziRTKUserRegSet (unsigned char Index, unsigned int Value)

Write flags set by the host PC.

Corresponding server paths:

- DEV123/CPUS/0/USERREGS/0
- DEV123/CPUS/0/USERREGS/1
- ...

Parameters:

[in] Index
index of the user registers-word (currently 0 - 15)

[in] Value
the new value to write to the user register

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when the provided index is out of range

See Also:

[ziRTKUserRegGet](#), User Registers

ziRTKExtMemGet

ZI_STATUS ziRTKExtMemGet (unsigned int Address, unsigned int* Value)

Read an address in the external memory.

Parameters:

[in] Address

the byte-aligned address to read (0x00000000 to 0x2FFFFFF)

[in] Value

Pointer to an unsigned int to store the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when the provided index is out of range
- ZI_NULLPTR when Value is NULL

See Also:

[ziRTKExtMemSet](#), [External Memory](#)

ziRTKExtMemSet

ZI_STATUS ziRTKExtMemSet (unsigned int Address, unsigned int Value)

Write an address in the external memory.

Parameters:

[in] Address

the byte-aligned address to write (0x00000000 to 0x2FFFFFF)

[in] Value

the new value to write to the external memory

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when the provided index is out of range

See Also:

[ziRTKExtMemGet, External Memory](#)

ziRTKSigInGetCount

unsigned int ziRTKSigInGetCount ()

get the count of high speed signal inputs.

Returns:

The count of signal inputs

See Also:

[HF Signal Input](#)

ziRTKSigInGetRange

ZI_STATUS ziRTKSigInGetRange (unsigned int Index, float* Range)

Get an input's range (V).

Corresponding server paths:

- DEV123/SIGINS/0/RANGE
- DEV123/SIGINS/1/RANGE
- ...

Parameters:

[in] Index
index of the parameter

[out] Range
Pointer to a float to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when Range is NULL

See Also:

[ziRTKSigInSetRange](#), HF Signal Input

ziRTKSigInSetRange

ZI_STATUS ziRTKSigInSetRange (unsigned int Index, float Range)

Set an input's range(V).

Corresponding server paths:

- DEV123/SIGINS/0/RANGE
- DEV123/SIGINS/1/RANGE
- ...

Parameters:

[in] Index
index of the parameter

[in] Range
float providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKSigInGetRange](#), HF Signal Input

ziRTKSigInGetAC

ZI_STATUS ziRTKSigInGetAC (unsigned int Index, int* Value)

Get an input's AC mode.

Corresponding server paths:

- DEV123/SIGINS/0/AC
- DEV123/SIGINS/1/AC
- ...

Parameters:

[in] Index
index of the parameter

[out] Value
Pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when Value is NULL

See Also:

[ziRTKSigInSetAC](#), HF Signal Input

ziRTKSigInSetAC

ZI_STATUS ziRTKSigInSetAC (unsigned int Index, int Value)

Set an input's AC mode.

Corresponding server paths:

- DEV123/SIGINS/0/AC
- DEV123/SIGINS/1/AC
- ...

Parameters:

[in] Index
index of the parameter

[in] Value
int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKSigInGetAC](#), HF Signal Input

ziRTKSigInGetImp50**ZI_STATUS ziRTKSigInGetImp50 (unsigned int Index, int* Value)**

Get an input's 50 Ohm mode.

Corresponding server paths:

- DEV123/SIGINS/0/AC
- DEV123/SIGINS/1/AC
- ...

Parameters:

[in] Index
index of the parameter

[out] Value
Pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when Value is NULL

See Also:

[ziRTKSigInSetImp50](#), HF Signal Input

ziRTKSigInSetImp50

ZI_STATUS ziRTKSigInSetImp50 (unsigned int Index, int Value)

Set an input's 50 Ohm mode.

Corresponding server paths:

- DEV123/SIGINS/0/AC
- DEV123/SIGINS/1/AC
- ...

Parameters:

[in] Index
index of the parameter

[in] Value
int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKSigInGetImp50](#), HF Signal Input

ziRTKSigInGetDiff

ZI_STATUS ziRTKSigInGetDiff (unsigned int Index, int* Value)

Get an input's differential mode.

Corresponding server paths:

- DEV123/SIGINS/0/AC
- DEV123/SIGINS/1/AC
- ...

Parameters:

[in] Index
index of the parameter

[out] Value
Pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when Value is NULL

See Also:

[ziRTKSigInSetDiff](#), [HF Signal Input](#)

ziRTKSigInSetDiff

ZI_STATUS ziRTKSigInSetDiff (unsigned int Index, int Value)

Set an input's differential mode.

Corresponding server paths:

- DEV123/SIGINS/0/AC
- DEV123/SIGINS/1/AC
- ...

Parameters:

[in] Index
index of the parameter

[in] Value
int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKSigInGetDiff](#), HF Signal Input

ziRTKOscGetCount

unsigned int ziRTKOscGetCount ()

get the count of oscillators.

Returns:

The count of oscillators

See Also:

[Oscillators](#)

ziRTKOscGetFreq

ZI_STATUS ziRTKOscGetFreq (unsigned int Index, float* Freq)

Get an oscillator's frequency.

Corresponding server paths:

- DEV123/OSCS/0/FREQ
- DEV123/OSCS/1/FREQ
- ...

Parameters:

[in] Index
index of the parameter

[out] Freq
Pointer to a float to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range
- ZI_NULLPTR when Freq is NULL

See Also:

[ziRTKOscSetFreq](#), [Oscillators](#)

ziRTKOscSetFreq

ZI_STATUS ziRTKOscSetFreq (unsigned int Index, float Freq)

Set an oscillator's frequency.

Corresponding server paths:

- DEV123/OSCS/0/FREQ
- DEV123/OSCS/1/FREQ
- ...

Parameters:

[in] Index
index of the parameter

[in] Freq
float providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range

See Also:

[ziRTKOscGetFreq](#), Oscillators

ziRTKDemodGetCount

unsigned int ziRTKDemodGetCount ()

get the count of demodulators.

Returns:

The count of demodulators

See Also:

[Demodulator](#)

ziRTKDemodGetADCSelect

ZI_STATUS ziRTKDemodGetADCSelect (unsigned int Index, int* ADCSelect)

Get a demodulator's adcselect.

Corresponding server paths:

- DEV123/DEMOS/0/ADCSELECT
- DEV123/DEMOS/1/ADCSELECT
- ...

Note:

ADCSelect is only available with installed Multi-frequency Option.

Parameters:

[in] Index
index of the Parameter

[out] ADCSelect
pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range
- ZI_NULLPTR when ADCSelect is NULL

See Also:

[ziRTKDemodSetADCSelect](#), [Demodulator](#)

ziRTKDemodSetADCSelect

ZI_STATUS ziRTKDemodSetADCSelect (unsigned int Index, int ADCSelect)

Set a demodulator's ADC select.

Corresponding server paths:

- DEV123/DEMODS/0/ADCSELECT
- DEV123/DEMODS/1/ADCSELECT
- ...

Note:

ADCSelect is only available with installed Multi-frequency Option.

Parameters:

[in] Index
index of the Parameter

[in] ADCSelect
int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range

See Also:

[ziRTKDemodGetADCSelect](#), Demodulator

ziRTKDemodGetOscSelect**ZI_STATUS ziRTKDemodGetOscSelect (unsigned int Index, int* OscSelect)**

Get a demodulator's oscillator select.

Corresponding server paths:

- DEV123/DEMODS/0/OSCSELECT
- DEV123/DEMODS/1/OSCSELECT
- ...

Note:

OscSelect is only available with installed Multi-frequency Option.

Parameters:

[in] Index
index of the Parameter

[out] OscSelect
pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range
- ZI_NULLPTR when ADCSelect is NULL

See Also:

[ziRTKDemodSetOscSelect](#), Demodulator

ziRTKDemodSetOscSelect

ZI_STATUS ziRTKDemodSetOscSelect (unsigned int Index, int OscSelect)

Set a demodulator's oscillator select.

Corresponding server paths:

- DEV123/DEMODS/0/OSCSELECT
- DEV123/DEMODS/1/OSCSELECT
- ...

Note:

OscSelect is only available when the Multi-frequency Option is installed.

Parameters:

[in] Index
index of the Parameter

[in] OscSelect
int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range

See Also:

[ziRTKDemodGetOscSelect](#), Demodulator

ziRTKDemodGetOrder

ZI_STATUS ziRTKDemodGetOrder (unsigned int Index, int* Order)

Get a demodulator's filter order.

Corresponding server paths:

- DEV123/DEMODS/0/ORDER
- DEV123/DEMODS/1/ORDER
- ...

Parameters:

[in] Index
index of the Parameter

[out] Order
pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range
- ZI_NULLPTR when Order is NULL

See Also:

[ziRTKDemodSetOrder](#), Demodulator

ziRTKDemodSetOrder

ZI_STATUS ziRTKDemodSetOrder (unsigned int Index, int Order)

Set a demodulator's filter order.

Corresponding server paths:

- DEV123/DEMODY/0/ORDER
- DEV123/DEMODY/1/ORDER
- ...

Parameters:

[in] Index
index of the Parameter

[in] Order
int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range

See Also:

[ziRTKDemodGetOrder](#), Demodulator

ziRTKDemodGetHarmonic

ZI_STATUS ziRTKDemodGetHarmonic (unsigned int Index, int* Harmonic)

Get a demodulator's harmonic.

Corresponding server paths:

- DEV123/DEMOS/0/ORDER
- DEV123/DEMOS/1/ORDER
- ...

Parameters:

[in] Index
index of the Parameter

[out] Harmonic
pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range
- ZI_NULLPTR when Harmonic is NULL

See Also:

[ziRTKDemodSetHarmonic](#), Demodulator

ziRTKDemodSetHarmonic

ZI_STATUS ziRTKDemodSetHarmonic (unsigned int Index, int Harmonic)

Set a demodulator's harmonic.

Corresponding server paths:

- DEV123/DEMOS/0/ORDER
- DEV123/DEMOS/1/ORDER
- ...

Parameters:

[in] Index
index of the Parameter

[in] Harmonic
int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range

See Also:

[ziRTKDemodGetHarmonic](#), Demodulator

ziRTKDemodGetRate**ZI_STATUS ziRTKDemodGetRate (unsigned int Index, float* Rate)**

Get a demodulator's rate.

Corresponding server paths:

- DEV123/DEMODS/0/RATE
- DEV123/DEMODS/1/RATE
- ...

Parameters:

[in] Index
index of the Parameter

[out] Rate
pointer to a float to write the value in (in Hertz)

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range
- ZI_NULLPTR when Rate is NULL

See Also:

[ziRTKDemodSetRate](#), [Demodulator](#)

ziRTKDemodSetRate

ZI_STATUS ziRTKDemodSetRate (unsigned int Index, float Rate)

Set a demodulator's rate.

Corresponding server paths:

- DEV123/DEMODS/0/RATE
- DEV123/DEMODS/1/RATE
- ...

Parameters:

[in] Index
index of the parameter

[in] Rate
float providing the value (in Hertz)

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range

See Also:

[ziRTKDemodGetRate](#), Demodulator

ziRTKDemodGetTrigger

ZI_STATUS ziRTKDemodGetTrigger (unsigned int Index, unsigned int* Trigger)

Get a demodulator's trigger- and gating-settings.

Corresponding server paths:

- DEV123/DEMODS/0/TRIGGER
- DEV123/DEMODS/1/TRIGGER
- ...

Parameters:

[in] Index
index of the Parameter

[out] Trigger
pointer to an unsigned int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range
- ZI_NULLPTR when Trigger is NULL

See Also:

[ziRTKDemodSetTrigger](#), Demodulator

ziRTKDemodSetTrigger

ZI_STATUS ziRTKDemodSetTrigger (unsigned int Index, unsigned int Trigger)

Set a demodulator's trigger- and gating-settings.

Corresponding server paths:

- DEV123/DEMODS/0/TRIGGER
- DEV123/DEMODS/1/TRIGGER
- ...

Parameters:

[in] Index
index of the parameter

[in] Trigger
unsigned int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range

See Also:

[ziRTKDemodGetTrigger](#), Demodulator

ziRTKDemodGetPhaseShift

ZI_STATUS ziRTKDemodGetPhaseShift (unsigned int Index, float* PhaseShift)

Get a demodulator's phaseshift.

Corresponding server paths:

- DEV123/DEMODS/0/PHASESHIFT
- DEV123/DEMODS/1/PHASESHIFT
- ...

Parameters:

[in] Index
index of the Parameter

[out] PhaseShift
pointer to a float to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range
- ZI_NULLPTR when PhaseShift is NULL

See Also:

[ziRTKDemodSetPhaseShift](#), Demodulator

ziRTKDemodSetPhaseShift

ZI_STATUS ziRTKDemodSetPhaseShift (unsigned int Index, float PhaseShift)

Set a demodulator's phaseshift.

Corresponding server paths:

- DEV123/DEMODS/0/PHASESHIFT
- DEV123/DEMODS/1/PHASESHIFT
- ...

Parameters:

[in] Index
index of the parameter

[in] PhaseShift
float providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range

See Also:

[ziRTKDemodGetPhaseShift](#), Demodulator

ziRTKDemodGetTimeConstant

ZI_STATUS ziRTKDemodGetTimeConstant (unsigned int Index, float* TimeConstant)

Get a demodulator's timeconstant.

Corresponding server paths:

- DEV123/DEMODS/0/TIMECONSTANT
- DEV123/DEMODS/1/TIMECONSTANT
- ...

Parameters:

[in] Index
index of the Parameter

[out] TimeConstant
pointer to a float to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range
- ZI_NULLPTR when TimeConstant is NULL

See Also:

[ziRTKDemodSetTimeConstant](#), [Demodulator](#)

ziRTKDemodSetTimeConstant

ZI_STATUS ziRTKDemodSetTimeConstant (unsigned int Index, float TimeConstant)

Set a demodulator's timeconstant.

Corresponding server paths:

- DEV123/DEMODS/0/TIMECONSTANT
- DEV123/DEMODS/1/TIMECONSTANT
- ...

Parameters:

[in] Index
index of the parameter

[in] TimeConstant
float providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range

See Also:

[ziRTKDemodGetTimeConstant](#), Demodulator

ziRTKDemodGetSinc

ZI_STATUS ziRTKDemodGetSinc (unsigned int Index, unsigned int* Value)

Get a demodulator's sinc.

Corresponding server paths:

- DEV123/DEMODY/0/SINC
- DEV123/DEMODY/1/SINC
- ...

Parameters:

[in] Index
index of the Parameter

[out] Value
pointer to an unsigned int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range
- ZI_NULLPTR when TimeConstant is NULL

See Also:

[ziRTKDemodSetSinc](#), Demodulator

ziRTKDemodSetSinc

ZI_STATUS ziRTKDemodSetSinc (unsigned int Index, unsigned int Value)

Set a demodulator's sinc.

Corresponding server paths:

- DEV123/DEMODS/0/SINC
- DEV123/DEMODS/1/SINC
- ...

Parameters:

[in] Index
index of the parameter

[in] Value
unsigned int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range

See Also:

[ziRTKDemodGetSinc](#), Demodulator

ziRTKDemodGetSample

ZI_STATUS ziRTKDemodGetSample (unsigned int Index, DemodSample* Sample)

Retrieve a demodulator's sample.

Corresponding server paths:

- DEV123/DEMOS/0/SAMPLE
- DEV123/DEMOS/1/SAMPLE
- ...

Parameters:

[in] Index
index of the parameter

[out] Sample
[DemodSample](#) to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range
- ZI_NULLPTR when the Sample is NULL

See Also:

[Demodulator](#)

ziRTKDemodSampleGetTimeStamp

ZI_STATUS ziRTKDemodSampleGetTimeStamp (DemodSample* Sample, float* Seconds)

Returns the timestamp of a [DemodSample](#) in Seconds.

Parameters:

[in] Sample
the sample to extract the value from

[out] Seconds
pointer to a float in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when Seconds is NULL or Sample is NULL

See Also:

[ziRTKDemodSampleGetTimeStamp32](#), [ziRTKDemodSampleGetTimeStamp64](#), [Demodulator](#)

ziRTKDemodSampleGetTimeStamp32

ZI_STATUS ziRTKDemodSampleGetTimeStamp32 (**DemodSample*** Sample, **unsigned int*** TS)

Returns the timestamp of a [DemodSample](#) as 32bit unsigned int.

Note:

you may also read TS.TS32[ZIRTK_LSB_INDEX] of a [DemodSample](#) to get the same result

Parameters:

[in] Sample

the sample to extract the value from

[out] TS

pointer to an unsigned int in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when TS is NULL or Sample is NULL

See Also:

[ziRTKDemodSampleGetTimeStamp](#), [ziRTKDemodSampleGetTimeStamp64](#), Demodulator

ziRTKDemodSampleGetTimeStamp64

ZI_STATUS ziRTKDemodSampleGetTimeStamp64 (**DemodSample*** Sample, **unsigned long long*** TS)

Returns the timestamp of a [DemodSample](#) as 64 bit unsigned long long.

Note:

you may also read TS.TS64 of a [DemodSample](#) to get the same result

Parameters:

[in] Sample

the sample to extract the value from

[out] TS

pointer to an unsigned int in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when TS is NULL or Sample is NULL

See Also:

[ziRTKDemodSampleGetTimeStamp](#), [ziRTKDemodSampleGetTimeStamp32](#), [Demodulator](#)

ziRTKDemodSampleGetX

ZI_STATUS ziRTKDemodSampleGetX (**DemodSample*** Sample, **float*** X)

Returns the X-Value of a DemodSample as float.

Parameters:

[in] Sample
the sample to extract the value from

[out] X
pointer to a float in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when X is NULL or Sample is NULL

See Also:

[ziRTKDemodSampleGetX32](#), [ziRTKDemodSampleGetX64](#), [ziRTKDemodSampleGetCompX](#),
[ziRTKDemodSampleGetY](#), [ziRTKDemodSampleGetY32](#), [ziRTKDemodSampleGetY64](#),
[ziRTKDemodSampleGetCompY](#), [ziRTKDemodSampleGetCompXY](#), [Demodulator](#)

ziRTKDemodSampleGetX32

ZI_STATUS ziRTKDemodSampleGetX32 (**DemodSample*** Sample, **unsigned int*** X)

Returns the X-Value of a **DemodSample** as unsigned int.

Note:

you may also read X.Val32[ZIRTK_MSB_INDEX] of a **DemodSample** to get the same result

Parameters:

[in] Sample

the sample to extract the value from

[out] X

pointer to an unsigned int in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when X is NULL or Sample is NULL

See Also:

[ziRTKDemodSampleGetX](#), [ziRTKDemodSampleGetX64](#), [ziRTKDemodSampleGetCompX](#),
[ziRTKDemodSampleGetY](#), [ziRTKDemodSampleGetY32](#), [ziRTKDemodSampleGetY64](#),
[ziRTKDemodSampleGetCompY](#), [ziRTKDemodSampleGetCompXY](#), Demodulator

ziRTKDemodSampleGetX64

ZI_STATUS ziRTKDemodSampleGetX64 (**DemodSample*** Sample, **unsigned long long*** X)

Returns the X-Value of a **DemodSample** as **unsigned long long**.

Note:

you may also read X.Val64 of a **DemodSample** to get the same result

Parameters:

[in] Sample

the sample to extract the value from

[out] X

pointer to an **unsigned long long** in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when X is NULL or Sample is NULL

See Also:

[ziRTKDemodSampleGetX](#), [ziRTKDemodSampleGetX32](#), [ziRTKDemodSampleGetCompX](#),
[ziRTKDemodSampleGetY](#), [ziRTKDemodSampleGetY32](#), [ziRTKDemodSampleGetY64](#),
[ziRTKDemodSampleGetCompY](#), [ziRTKDemodSampleGetCompXY](#), [Demodulator](#)

ziRTKDemodSampleGetCompX

ZI_STATUS ziRTKDemodSampleGetCompX (DemodSample* Sample, float* X)

Returns the compensated X-Value of a DemodSample.

Parameters:

[in] Sample
the sample to extract the value from

[out] X
pointer to a float in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when X is NULL or Sample is NULL

See Also:

[ziRTKDemodSampleGetX](#), [ziRTKDemodSampleGetX32](#), [ziRTKDemodSampleGetX64](#),
[ziRTKDemodSampleGetY](#), [ziRTKDemodSampleGetY32](#), [ziRTKDemodSampleGetY64](#),
[ziRTKDemodSampleGetCompY](#), [ziRTKDemodSampleGetCompXY](#), [Demodulator](#)

ziRTKDemodSampleGetY

ZI_STATUS ziRTKDemodSampleGetY (**DemodSample*** Sample, **float*** Y)

Returns the Y-Value of a DemodSample as float.

Parameters:

[in] Sample
the sample to extract the value from

[out] Y
pointer to a float in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when Y is NULL or Sample is NULL

See Also:

[ziRTKDemodSampleGetX](#), [ziRTKDemodSampleGetX32](#), [ziRTKDemodSampleGetX64](#),
[ziRTKDemodSampleGetCompX](#), [ziRTKDemodSampleGetY32](#), [ziRTKDemodSampleGetY64](#),
[ziRTKDemodSampleGetCompY](#), [ziRTKDemodSampleGetCompXY](#), [Demodulator](#)

ziRTKDemodSampleGetY32

ZI_STATUS ziRTKDemodSampleGetY32 (DemodSample* Sample, unsigned int* Y)

Returns the Y-Value of a [DemodSample](#) as unsigned int.

Note:

you may also read Y.Val32[ZIRTK_MSB_INDEX] of a [DemodSample](#) to get the same result

Parameters:

[in] Sample

the sample to extract the value from

[out] Y

pointer to an unsigned int in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when Y is NULL or Sample is NULL

See Also:

[ziRTKDemodSampleGetX](#), [ziRTKDemodSampleGetX32](#), [ziRTKDemodSampleGetX64](#),
[ziRTKDemodSampleGetCompX](#), [ziRTKDemodSampleGetY](#), [ziRTKDemodSampleGetY64](#),
[ziRTKDemodSampleGetCompY](#), [ziRTKDemodSampleGetCompXY](#), [Demodulator](#)

ziRTKDemodSampleGetY64

ZI_STATUS ziRTKDemodSampleGetY64 (DemodSample* Sample, unsigned long long* Y)

Returns the Y-Value of a [DemodSample](#) as unsigned long long.

Note:

you may also read Y.Val64 of a [DemodSample](#) to get the same result

Parameters:

[in] Sample

the sample to extract the value from

[out] Y

pointer to an unsigned long long in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when Y is NULL or Sample is NULL

See Also:

[ziRTKDemodSampleGetX](#), [ziRTKDemodSampleGetX32](#), [ziRTKDemodSampleGetX64](#),
[ziRTKDemodSampleGetCompX](#), [ziRTKDemodSampleGetY](#), [ziRTKDemodSampleGetY32](#),
[ziRTKDemodSampleGetCompY](#), [ziRTKDemodSampleGetCompXY](#), [Demodulator](#)

ziRTKDemodSampleGetCompY

ZI_STATUS ziRTKDemodSampleGetCompY (DemodSample* Sample, float* Y)

Returns the compensated Y-Value of a DemodSample.

Parameters:

[in] Sample

the sample to extract the value from

[out] Y

pointer to a float in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when Y is NULL or Sample is NULL

See Also:

[ziRTKDemodSampleGetX](#), [ziRTKDemodSampleGetX32](#), [ziRTKDemodSampleGetX64](#),
[ziRTKDemodSampleGetCompX](#), [ziRTKDemodSampleGetY](#), [ziRTKDemodSampleGetY32](#),
[ziRTKDemodSampleGetY64](#), [ziRTKDemodSampleGetCompXY](#), [Demodulator](#)

ziRTKDemodSampleGetCompXY

ZI_STATUS ziRTKDemodSampleGetCompXY (DemodSample* Sample, float* X, float* Y)

Returns the compensated X-Value and the compensated Y-Value of a DemodSample.

Parameters:

[in] Sample

the sample to extract the value from

[out] X

pointer to a float in which the X-value will be returned

[out] Y

pointer to a float in which the Y-value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when X is NULL or Y is NULL or Sample is NULL

See Also:

[ziRTKDemodSampleGetX](#), [ziRTKDemodSampleGetX32](#), [ziRTKDemodSampleGetX64](#),
[ziRTKDemodSampleGetCompX](#), [ziRTKDemodSampleGetY](#), [ziRTKDemodSampleGetY32](#),
[ziRTKDemodSampleGetY64](#), [ziRTKDemodSampleGetCompY](#), [Demodulator](#)

ziRTKDemodSampleGetR

ZI_STATUS ziRTKDemodSampleGetR (**DemodSample*** Sample, **float*** R)

Returns the Radius-Value of a DemodSample.

Parameters:

[in] Sample
the sample to extract the value from

[out] R
pointer to a float in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when R is NULL or Sample is NULL

See Also:

[ziRTKDemodSampleGetCompR](#), [ziRTKDemodSampleGetTheta](#), [Demodulator](#)

ziRTKDemodSampleGetCompR

ZI_STATUS ziRTKDemodSampleGetCompR (**DemodSample*** Sample, **float*** R)

Returns the compensated Radius-Value of a DemodSample.

Parameters:

[in] Sample
the sample to extract the value from

[out] R
pointer to a float in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when R is NULL or Sample is NULL

See Also:

[ziRTKDemodSampleGetR](#), [ziRTKDemodSampleGetTheta](#), [Demodulator](#)

ziRTKDemodSampleGetTheta

ZI_STATUS ziRTKDemodSampleGetTheta (DemodSample* Sample, float* Theta)

Returns the Theta-Value of a [DemodSample](#).

Parameters:

[in] Sample
the sample to extract the value from

[out] Theta
pointer to a float in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when Theta is NULL or Sample is NULL

See Also:

[ziRTKDemodSampleGetR](#), [ziRTKDemodSampleGetCompR](#), [Demodulator](#)

ziRTKSigOutGetCount

unsigned int ziRTKSigOutGetCount ()

get the count of HF Signal outputs.

Returns:

The count of the signal outputs.

See Also:

[HF Signal Output](#)

ziRTKSigOutGetRange

ZI_STATUS ziRTKSigOutGetRange (unsigned int Index, float* Range)

Get an output's range (V).

returns an output's range.

Corresponding server paths:

- DEV123/SIGOUTS/0/RANGE
- DEV123/SIGOUTS/1/RANGE
- ...

Parameters:

[in] Index
index of the parameter

[out] Range
pointer to a float to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range
- ZI_NULLPTR when Range is NULL

See Also:

[ziRTKSigOutSetRange](#), HF Signal Output

ziRTKSigOutSetRange

ZI_STATUS ziRTKSigOutSetRange (unsigned int Index, float Range)

Set an output's range (V).

sets an output's range.

Corresponding server paths:

- DEV123/SIGOUTS/0/RANGE
- DEV123/SIGOUTS/1/RANGE
- ...

Parameters:

[in] Index
index of the parameter

[in] Range
float providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when index is out of range

See Also:

[ziRTKSigOutGetRange](#), HF Signal Output

ziRTKSigOutGetChannelCount

unsigned int ziRTKSigOutGetChannelCount ()

get the count of HF Signal output Mixer Channels.

Returns:

The count of mixer channels

See Also:

[HF Signal Output](#)

ziRTKSigOutGetEnable

ZI_STATUS ziRTKSigOutGetEnable (unsigned int Index, unsigned int Channel, int* Value)

Get an output's mixer enable.

returns an output's mixer enable.

Corresponding server paths:

- DEV123/SIGOUTS/0/ENABLES/0
- DEV123/SIGOUTS/0/ENABLES/1
- ...
- DEV123/SIGOUTS/1/ENABLES/0
- DEV123/SIGOUTS/1/ENABLES/1
- ...

Parameters:

[in] Index
index of the signal output

[in] Channel
index of the mixer-channel

[out] Value
pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when Value is NULL
- ZI_OUTOFRANGE when Index is out of range or channel is out of range

See Also:

[ziRTKSigOutSetEnable](#), HF Signal Output

ziRTKSigOutSetEnable

ZI_STATUS ziRTKSigOutSetEnable (unsigned int Index, unsigned int Channel, int Value)

Set an output's mixer enable.

sets an output's mixer enable.

Corresponding server paths:

- DEV123/SIGOUTS/0/ENABLES/0
- DEV123/SIGOUTS/0/ENABLES/1
- ...
- DEV123/SIGOUTS/1/ENABLES/0
- DEV123/SIGOUTS/1/ENABLES/1
- ...

Parameters:

[in] Index
index of the signal output

[in] Channel
index of the mixer-channel

[in] Value
int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range or channel is out of range

See Also:

[ziRTKSigOutGetEnable](#), [HF Signal Output](#)

ziRTKSigOutGetAmplitude

ZI_STATUS ziRTKSigOutGetAmplitude (unsigned int Index, unsigned int Channel, float* Value)

Get an output's mixer amplitude.

returns an output's mixer amplitude.

Corresponding server paths:

- DEV123/SIGOUTS/0/AMPLITUDES/0
- DEV123/SIGOUTS/0/AMPLITUDES/1
- ...
- DEV123/SIGOUTS/1/AMPLITUDES/0
- DEV123/SIGOUTS/1/AMPLITUDES/1
- ...

Parameters:

[in] Index
index of the signal output

[in] Channel
index of the mixer-channel

[out] Value
pointer to a float to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when Value is NULL
- ZI_OUTOFRANGE when Index is out of range or channel is out of range

See Also:

[ziRTKSigOutSetAmplitude](#), [HF Signal Output](#)

ziRTKSigOutSetAmplitude

ZI_STATUS **ziRTKSigOutSetAmplitude** (**unsigned int Index, unsigned int Channel, float Value**)

Set an output's mixer amplitude.

sets an output's mixer amplitude.

Corresponding server paths:

- DEV123/SIGOUTS/0/AMPLITUDES/0
- DEV123/SIGOUTS/0/AMPLITUDES/1
- ...
- DEV123/SIGOUTS/1/AMPLITUDES/0
- DEV123/SIGOUTS/1/AMPLITUDES/1
- ...

Parameters:

[in] Index
index of the signal output

[in] Channel
index of the mixer-channel

[in] Value
float providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range or channel is out of range

See Also:

[ziRTKSigOutGetAmplitude](#), HF Signal Output

ziRTKSigOutGetOn

ZI_STATUS ziRTKSigOutGetOn (unsigned int Index, int* Value)

Get an output's on-switch.

Corresponding server paths:

- DEV123/SIGOUTS/0/ON
- DEV123/SIGOUTS/1/ON
- ...

Parameters:

[in] Index
index of the parameter

[out] Value
Pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when Value is NULL

See Also:

[ziRTKSigOutSetOn](#), [HF Signal Output](#)

ziRTKSigOutSetOn

ZI_STATUS ziRTKSigOutSetOn (unsigned int Index, int Value)

Set an output's on-switch.

Corresponding server paths:

- DEV123/SIGOUTS/0/ON
- DEV123/SIGOUTS/1/ON
- ...

Parameters:

[in] Index
index of the parameter

[in] Value
int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKSigOutGetOn](#), HF Signal Output

ziRTKSigOutGetAdd

ZI_STATUS ziRTKSigOutGetAdd (unsigned int Index, int* Value)

Get an output's add mode.

Corresponding server paths:

- DEV123/SIGOUTS/0/ON
- DEV123/SIGOUTS/1/ON
- ...

Parameters:

[in] Index

index of the parameter

[out] Value

Pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when Value is NULL

See Also:

[ziRTKSigOutSetAdd](#), HF Signal Output

ziRTKSigOutSetAdd

ZI_STATUS ziRTKSigOutSetAdd (unsigned int Index, int Value)

Set an output's add mode.

Corresponding server paths:

- DEV123/SIGOUTS/0/ON
- DEV123/SIGOUTS/1/ON
- ...

Parameters:

[in] Index
index of the parameter

[in] Value
int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKSigOutGetAdd](#), [HF Signal Output](#)

ziRTKAuxOutGetCount

unsigned int ziRTKAuxOutGetCount ()

get the count of auxiliary analog outputs.

Returns:

The count of auxiliary outputs

See Also:

[Auxiliary Analog Outputs](#)

ziRTKAuxOutGetValue**ZI_STATUS ziRTKAuxOutGetValue (unsigned int Index, float* Value)**

Get a auxiliary outputs value as float (V).

Corresponding server paths:

- DEV123/AUXOUTS/0/VALUE
- DEV123/AUXOUTS/1/VALUE
- ...

Parameters:

[in] Index
index of the Parameter

[out] Value
pointer to a float to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when Value is NULL

See Also:

[Auxiliary Analog Outputs](#)

ziRTKAuxOutGetOutputSelect

ZI_STATUS ziRTKAuxOutGetOutputSelect (unsigned int Index, int* OutputSelect)

Get an auxiliary outputs outputselect.

Corresponding server paths:

- DEV123/AUXOUTS/0/OUTPUTSELECT
- DEV123/AUXOUTS/1/OUTPUTSELECT
- ...

Parameters:

[in] Index
index of the Parameter

[out] OutputSelect
pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when OutputSelect is NULL

See Also:

[ziRTKAuxOutSetOutputSelect](#), Auxiliary Analog Outputs

ziRTKAuxOutSetOutputSelect

ZI_STATUS ziRTKAuxOutSetOutputSelect (unsigned int Index, int OutputSelect)

Set an auxiliary output's outputselect.

Corresponding server paths:

- DEV123/AUXOUTS/0/OUTPUTSELECT
- DEV123/AUXOUTS/1/OUTPUTSELECT
- ...

Parameters:

[in] Index
index of the Parameter

[out] OutputSelect
an int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKAuxOutGetOutputSelect](#), Auxiliary Analog Outputs

ziRTKAuxOutGetDemodSelect

ZI_STATUS ziRTKAuxOutGetDemodSelect (unsigned int Index, int* DemodSelect)

Get an auxiliary output's demodselect.

Corresponding server paths:

- DEV123/AUXOUTS/0/DEMODOSELECT
- DEV123/AUXOUTS/1/DEMODOSELECT
- ...

Parameters:

[in] Index
index of the Parameter

[out] DemodSelect
pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when DemodSelect is NULL

See Also:

[ziRTKAuxOutSetDemodSelect](#), Auxiliary Analog Outputs

ziRTKAuxOutSetDemodSelect

ZI_STATUS ziRTKAuxOutSetDemodSelect (unsigned int Index, int DemodSelect)

Set an auxiliary output's demodselect.

Corresponding server paths:

- DEV123/AUXOUTS/0/DEMODOSELECT
- DEV123/AUXOUTS/1/DEMODOSELECT
- ...

Parameters:

[in] Index
index of the Parameter

[out] DemodSelect
an int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKAuxOutGetDemodSelect](#), Auxiliary Analog Outputs

ziRTKAuxOutGetOffset

ZI_STATUS ziRTKAuxOutGetOffset (**unsigned int** Index, **float*** Offset)

Get an auxiliary output's offset.

Corresponding server paths:

- DEV123/AUXOUTS/0/OFFSET
- DEV123/AUXOUTS/1/OFFSET
- ...

Parameters:

[in] Index
index of the Parameter

[out] Offset
pointer to a float to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when Offset is NULL

See Also:

[ziRTKAuxOutSetOffset](#), [ziRTKAuxOutSetOffsetNoUpdate](#), [ziRTKAuxOutSetOffsetInt](#),
[ziRTKAuxOutSetOffsetIntNoUpdate](#), Auxiliary Analog Outputs

ziRTKAuxOutSetOffset

ZI_STATUS ziRTKAuxOutSetOffset (**unsigned int** Index, **float** Offset)

Set an auxiliary output's offset.

Corresponding server paths:

- DEV123/AUXOUTS/0/OFFSET
- DEV123/AUXOUTS/1/OFFSET
- ...

Parameters:

[in] Index
index of the Parameter

[out] Offset
a float providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKAuxOutSetOffsetNoUpdate](#), [ziRTKAuxOutSetOffsetInt](#), [ziRTKAuxOutSetOffsetIntNoUpdate](#),
[ziRTKAuxOutGetOffset](#), Auxiliary Analog Outputs

ziRTKAuxOutSetOffsetNoUpdate

ZI_STATUS ziRTKAuxOutSetOffsetNoUpdate (unsigned int Index, float Offset)

Set an auxiliary output's offset without updating ziServer.

Corresponding server paths:

- DEV123/AUXOUTS/0/OFFSET
- DEV123/AUXOUTS/1/OFFSET
- ...

Parameters:

[in] Index
index of the Parameter

[out] Offset
a float providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKAuxOutSetOffset](#), [ziRTKAuxOutSetOffsetInt](#), [ziRTKAuxOutSetOffsetIntNoUpdate](#),
[ziRTKAuxOutGetOffset](#), Auxiliary Analog Outputs

ziRTKAuxOutSetOffsetInt

ZI_STATUS ziRTKAuxOutSetOffsetInt (unsigned int Index, int Offset)

Set an auxiliary output's offset as integer.

Corresponding server paths:

- DEV123/AUXOUTS/0/OFFSET
- DEV123/AUXOUTS/1/OFFSET
- ...

Parameters:

[in] Index
index of the Parameter

[out] Offset
an integer providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKAuxOutSetOffsetInt32](#), [ziRTKAuxOutSetOffset](#), [ziRTKAuxOutSetOffsetNoUpdate](#),
[ziRTKAuxOutGetOffset](#), [Auxiliary Analog Outputs](#)

ziRTKAuxOutSetOffsetIntNoUpdate**ZI_STATUS ziRTKAuxOutSetOffsetIntNoUpdate (unsigned int Index, int Offset)**

Set an auxiliary output's offset as integer without updating ziServer.

Corresponding server paths:

- DEV123/AUXOUTS/0/OFFSET
- DEV123/AUXOUTS/1/OFFSET
- ...

Parameters:

[in] Index
index of the Parameter

[out] Offset
an integer providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKAuxOutSetOffsetInt](#), [ziRTKAuxOutSetOffset](#), [ziRTKAuxOutSetOffsetNoUpdate](#),
[ziRTKAuxOutGetOffset](#), Auxiliary Analog Outputs

ziRTKAuxOutGetScale

ZI_STATUS ziRTKAuxOutGetScale (unsigned int Index, float* Scale)

Get an auxiliary output's scale.

Corresponding server paths:

- DEV123/AUXOUTS/0/SCALE
- DEV123/AUXOUTS/1/SCALE
- ...

Parameters:

[in] Index
index of the Parameter

[out] Scale
pointer to a float to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when Scale is NULL

See Also:

[ziRTKAuxOutSetScale](#), Auxiliary Analog Outputs

ziRTKAuxOutSetScale

ZI_STATUS ziRTKAuxOutSetScale (unsigned int Index, float Scale)

Set an auxiliary output's scale.

Corresponding server paths:

- DEV123/AUXOUTS/0/SCALE
- DEV123/AUXOUTS/1/SCALE
- ...

Parameters:

[in] Index
index of the Parameter

[out] Scale
a float providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKAuxOutGetScale](#), Auxiliary Analog Outputs

ziRTKAuxInGetCount

unsigned int ziRTKAuxInGetCount ()

get the count of Auxiliary Analog Inputs.

Returns:

The count of auxiliary inputs

See Also:

[Auxiliary Analog Inputs](#)

ziRTKAuxInGetAveraging

ZI_STATUS ziRTKAuxInGetAveraging (unsigned int Index, int* Averaging)

Get a auxiliary input's averaging/sample rate.

Corresponding server path:

- DEV123/AUXINS/0/AVERAGING

See [Aux Analog Inputs](#) for details.

Parameters:

[in] Index
index of the Parameter

[out] Averaging
pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when Averaging is NULL

See Also:

[ziRTKAuxInSetAveraging](#), [Auxiliary Analog Inputs](#)

ziRTKAuxInSetAveraging

ZI_STATUS ziRTKAuxInSetAveraging (unsigned int Index, int Averaging)

Set an auxiliary output's averaging sample rate.

Corresponding server path:

- DEV123/AUXINS/0/AVERAGING

See [Aux Analog Inputs](#) for details.

Parameters:

[in] Index
index of the Parameter

[out] Averaging
value to set

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKAuxInGetAveraging](#), [Auxiliary Analog Inputs](#)

ziRTKAuxInGetSample

ZI_STATUS ziRTKAuxInGetSample (unsigned int Index, AuxInSample* Sample)

retrieve a auxiliary input sample

Corresponding server path:

- DEV123/AUXINS/0/SAMPLE

Parameters:

[in] Index
index of the Parameter

[in] Sample
AuxInSample to write the value in.

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when Sample is NULL

See Also:

[Auxiliary Analog Inputs](#)

ziRTKAuxInSampleGetTimeStamp

ZI_STATUS ziRTKAuxInSampleGetTimeStamp ([AuxInSample*](#) Sample, [float*](#) Seconds)

Returns the timestamp of an [AuxInSample](#) in Seconds.

Parameters:

[in] Sample
the sample to extract the value from

[out] Seconds
pointer to a float in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when Seconds is NULL or Sample is NULL

See Also:

[ziRTKAuxInSampleGetTimeStamp32](#), [ziRTKAuxInSampleGetTimeStamp64](#), Auxiliary Analog Inputs

ziRTKAuxInSampleGetTimeStamp32

ZI_STATUS ziRTKAuxInSampleGetTimeStamp32 (**AuxInSample*** Sample, **unsigned int*** TS)

Returns the timestamp of an [AuxInSample](#) as 32bit unsigned int.

Note:

you may also read TS.TS32[ZIRTK_LSB_INDEX] of a [AuxInSample](#) to get the same result

Parameters:

[in] Sample

the sample to extract the value from

[out] TS

pointer to an unsigned int in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when TS is NULL or Sample is NULL

See Also:

[ziRTKAuxInSampleGetTimeStamp](#), [ziRTKAuxInSampleGetTimeStamp64](#), [Auxiliary Analog Inputs](#)

ziRTKAuxInSampleGetTimeStamp64

ZI_STATUS ziRTKAuxInSampleGetTimeStamp64 ([AuxInSample*](#) Sample, [unsigned long long*](#) TS)

Returns the timestamp of an [AuxInSample](#) as 64 bit unsigned long long.

Note:

you may also read TS.TS64 of a [AuxInSample](#) to get the same result

Parameters:

[in] Sample

the sample to extract the value from

[out] TS

pointer to an unsigned long long in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when TS is NULL or Sample is NULL

See Also:

[ziRTKAuxInSampleGetTimeStamp](#), [ziRTKAuxInSampleGetTimeStamp32](#), [Auxiliary Analog Inputs](#)

ziRTKAuxInSampleGetValue

ZI_STATUS ziRTKAuxInSampleGetValue (**AuxInSample*** Sample, **unsigned int** Channel, **float*** Value)

Returns a Value of a [AuxInSample](#).

Parameters:

[in] Sample
the sample to extract the value from

[in] Channel
the channel to retrieve the value from

[out] Value
pointer to a float in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when channel is out of range
- ZI_NULLPTR when Value is NULL or Sample is NULL

See Also:

[ziRTKAuxInSampleGetValue16](#), Auxiliary Analog Inputs

ziRTKAuxInSampleGetValue16

ZI_STATUS ziRTKAuxInSampleGetValue16 ([AuxInSample](#)* Sample, unsigned int Channel, short* Value)

Returns a Value of a [AuxInSample](#) as short.

Parameters:

[in] Sample
the sample to extract the value from

[in] Channel
the channel to retrieve the value from

[out] Value
pointer to a short in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when channel is out of range
- ZI_NULLPTR when Value is NULL or Sample is NULL

See Also:

[ziRTKAuxInSampleGetValue](#), Auxiliary Analog Inputs

ziRTKDIOGetCount

unsigned int ziRTKDIOGetCount ()

get the count of Digital IO.

Returns:

The count of the DIO

See Also:

[Digital Input/Output](#)

ziRTKDIOGetExtClk

ZI_STATUS ziRTKDIOGetExtClk (unsigned int Index, int* ExtClk)

Get a DIO's External-Clock setting.

Corresponding server path:

- DEV123/DIOS/0/EXTCLK

Parameters:

[in] Index
index of the Parameter

[out] ExtClk
pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when ExtClk is NULL

See Also:

[ziRTKDIOSetExtClk](#), Digital Input/Output

ziRTKDIOSetExtClk

ZI_STATUS ziRTKDIOSetExtClk (unsigned int Index, int ExtClk)

Set a DIO's External-Clock setting.

Corresponding server path:

- DEV123/DIOS/0/EXTCLK

Parameters:

[in] Index
index of the Parameter

[in] ExtClk
int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKDIOGetExtClk](#), [Digital Input/Output](#)

ziRTKDIOGetDecimation

ZI_STATUS ziRTKDIOGetDecimation (unsigned int Index, int* Decimation)

Get a DIO's Decimation.

Corresponding server path:

- DEV123/DIOS/0/DECIMATION

Parameters:

[in] Index
index of the Parameter

[out] Decimation
pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when Decimation is NULL

See Also:

[ziRTKDIOSetDecimation](#), Digital Input/Output

ziRTKDIOSetDecimation

ZI_STATUS ziRTKDIOSetDecimation (unsigned int Index, int Decimation)

Set a DIO's Decimation.

Corresponding server path:

- DEV123/DIOS/0/DECIMATION

Parameters:

[in] Index
index of the Parameter

[in] Decimation
int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKDIOGetDecimation](#), [Digital Input/Output](#)

ziRTKDIOGetDrive

ZI_STATUS ziRTKDIOGetDrive (unsigned int Index, int* Drive)

Get a DIO's Drive.

Corresponding server path:

- DEV123/DIOS/0/DRIVE

Parameters:

[in] Index
index of the Parameter

[out] Drive
pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when Drive is NULL

See Also:

[ziRTKDIOSetDrive](#), [Digital Input/Output](#)

ziRTKDIOSetDrive

ZI_STATUS ziRTKDIOSetDrive (unsigned int Index, int Drive)

Set a DIO's Drive.

Corresponding server path:

- DEV123/DIOS/0/DRIVE

Parameters:

[in] Index
index of the Parameter

[in] Drive
int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKDIOGetDrive](#), [Digital Input/Output](#)

ziRTKDIOGetOutput

ZI_STATUS ziRTKDIOGetOutput (unsigned int Index, unsigned int* Output)

Get a DIO's Output.

Corresponding server path:

- DEV123/DIOS/0/OUTPUT

Parameters:

[in] Index
index of the Parameter

[out] Output
pointer to an int to write the value in

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when Output is NULL

See Also:

[ziRTKDIOSetOutput](#), Digital Input/Output

ziRTKDIOSetOutput

ZI_STATUS ziRTKDIOSetOutput (unsigned int Index, unsigned int Output)

Set a DIO's Output.

Corresponding server path:

- DEV123/DIOS/0/OUTPUT

Parameters:

[in] Index
index of the Parameter

[in] Output
unsigned int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKDIOGetOutput](#), [ziRTKDIOSetOutputNoUpdate](#), [Digital Input/Output](#)

ziRTKDIOSetOutputNoUpdate

ZI_STATUS ziRTKDIOSetOutputNoUpdate (**unsigned int** Index, **unsigned int** Output)

Set a DIO's Output without updating ziServer.

Corresponding server path:

- DEV123/DIOS/0/OUTPUT

Parameters:

[in] Index
index of the Parameter

[in] Output
unsigned int providing the value

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[ziRTKDIOSetOutput](#), [Digital Input/Output](#)

ziRTKDIOGetSample

ZI_STATUS ziRTKDIOGetSample (unsigned int Index, **DIOSample*** Sample)

retrieve a dio sample

Corresponding server path:

- DEV123/DIOS/0/INPUT

Parameters:

[in] Index
index of the Parameter

[in] Sample
DIOSample to write the value in.

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range
- ZI_NULLPTR when Sample is NULL

See Also:

[Digital Input/Output](#)

ziRTKDIOSampleGetTimeStamp

ZI_STATUS ziRTKDIOSampleGetTimeStamp (DIOSample* Sample, float* Seconds)

Returns the timestamp of a [DIOSample](#) in Seconds.

Parameters:

[in] Sample
the sample to extract the value from

[out] Seconds
pointer to a float in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when Seconds is NULL or Sample is NULL

See Also:

[ziRTKDIOGetSample](#), [ziRTKDIOSampleGetTimeStamp32](#), [ziRTKDIOSampleGetTimeStamp64](#),
[Digital Input/Output](#)

ziRTKDIOSampleGetTimeStamp32

ZI_STATUS ziRTKDIOSampleGetTimeStamp32 (**DIOSample*** Sample, **unsigned int*** TS)

Returns the timestamp of a **DIOSample** as 32bit unsigned int.

Note:

you may also read TS.TS32[ZIRTK_LSB_INDEX] of a **DIOSample** to get the same result

Parameters:

[in] Sample

the sample to extract the value from

[out] TS

pointer to an unsigned int in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when TS is NULL or Sample is NULL

See Also:

[ziRTKDIOGetSample](#), [ziRTKDIOSampleGetTimeStamp](#), [ziRTKDIOSampleGetTimeStamp64](#),
Digital Input/Output

ziRTKDIOSampleGetTimeStamp64

ZI_STATUS ziRTKDIOSampleGetTimeStamp64 (**DIOSample*** Sample, **unsigned long long*** TS)

Returns the timestamp of a [DIOSample](#) as 64 bit unsigned long long.

Note:

you may also read TS.TS64 of a [DIOSample](#) to get the same result

Parameters:

[in] Sample

the sample to extract the value from

[out] TS

pointer to an unsigned long long in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when TS is NULL or Sample is NULL

See Also:

[ziRTKDIOGetSample](#), [ziRTKDIOSampleGetTimeStamp](#), [ziRTKDIOSampleGetTimeStamp32](#),
Digital Input/Output

ziRTKDIOSampleGetBits

ZI_STATUS ziRTKDIOSampleGetBits (**DIOSample*** Sample, **unsigned int*** Bits)

Returns the Bits of a [DIOSample](#).

Note:

you may also read Bits[0] of a [DIOSample](#) to get the same result

Parameters:

[in] Sample

the sample to extract the value from

[out] Bits

pointer to an unsigned int in which the value will be returned

Returns:

- ZI_SUCCESS on success
- ZI_NULLPTR when Bits is NULL or Sample is NULL

See Also:

[ziRTKDIOGetSample](#), [Digital Input/Output](#)

ziRTKPLLSetADCSelect

ZI_STATUS ziRTKPLLSetADCSelect (unsigned int Index, int ADCSelect)

Set the ADCSELECT value of a PLL.

Returns:

- ZI_SUCCESS on success
- ZI_OUTOFRANGE when Index is out of range

See Also:

[PLL](#)

Chapter 8. Specifications

Important

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

Important

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

8.1. General Specifications

Table 8.1. 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	-	-	60 W
operating environment	IEC61010, indoor location, installation category II, pollution degree 2		
operating altitude	up to 2000 meters		
power supply AC line	110–120/220–240 V, 50/60 Hz		
power supply Japan	requires external 100 V to 110 V transformer (50/60 Hz) for operation according to specification		
dimensions with handles and feet	45 x 34 x 10 cm, 17.7 x 13.6 x 4.0 inch, 19 inch rack compatible		
weight	6.2 kg		
recommended calibration interval	2 years		

Table 8.2. Maximum ratings

Parameter	min	typ	max
damage threshold HF inputs (Input 1, Input 2)	-5 V	-	5 V
damage threshold HF outputs (Output 1, Output 2)	-12 V	-	12 V
damage threshold Add inputs (Add 1, Add 2)	-12 V	-	12 V
damage threshold Sync output (Sync 1, Sync 2)	-12 V	-	12 V
damage threshold auxiliary outputs	-12 V	-	12 V
damage threshold auxiliary inputs	-12 V	-	12 V
damage threshold digital I/O (including DIO 0 and DIO 1 BNC connectors)	0 V	-	5 V
damage threshold Clock input	0 V	-	5 V

Table 8.3. Host system requirements

Parameter	Description
supported Windows operating systems	32-bit and 64-bit versions of XP, Vista, Windows 7, Windows 8
supported Linux distribution	32-bit and 64-bit of Linux, Ubuntu 10.04 LTS (i386,

Parameter	Description
	AMD64), 64-bit systems require the IA32 extension (install with <code>sudo apt-get install ia32-libs</code>)
minimum processor requirements	AMD K8 (Athlon 64), Intel Pentium 4
minimum memory requirements	1 GB RAM, 2 GB recommended
list of 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

Table 8.4. Maximum sample readout rate

Active demodulators	Maximum sample readout rate	Comments
1	460 kSamples/s	To achieve highest rates, it is advised to remove all other data transfer that loads the USB. It is recommended to check the sample loss flag (in the status tab) from time to time when using high readout rate settings.
2 – 3	230 kSamples/s	
4 – 6	115 kSamples/s	
7 – 8	57 kSamples/s	

Note

The sample readout rate is the rate at which demodulated samples are transferred from the Instrument to the host computer. This rate has to be set to at least 2 times the signal bandwidth of the related demodulator in order to satisfy the Nyquist sampling theorem. As the total rate is limited by the USB interface, the maximum rate becomes smaller when the number of active demodulators is increased. This is summarized in the table above for HF2LI / HF2PLL (6 demodulators) and HF2IS (8 demodulators). An up-to-date and performing host computer is required to achieve these rates.

8.2. Analog Interface Specifications

Table 8.5. HF signal inputs

Parameter	min	typ	max
connectors	front-panel single-ended/differential BNC		
input impedance (low value)	–	50 Ω	–
input impedance (high value)	500 kΩ	1 MΩ	–
input frequency range	0.7 μHz	–	50 MHz
input A/D conversion	14 bit, 210 MSamples/s		
input noise amplitude (> 10 kHz, AC coupling, 50 Ω and 1 MΩ), for detailed information see Figure 8.5	–	5 nV/√Hz	–
input amplitude accuracy (5 MHz), for detailed information see Figure 8.10	–	–	5%
input amplitude accuracy (10 MHz), for detailed information see Figure 8.10	–	–	10%
input amplitude stability	–	–	0.2 %/°C
input DC offset (<1 V input range)	–	–	20 mV
input DC offset (>1 V input range)	–	–	2%
input bias current	–	–	6 μA
input range settings	1 mV	–	1.5 V
input full range sensitivity (10 V lock-in amplifier output)	1 nV	–	1.5 V
input range (AC) with AC coupling	-0.6 V	–	0.6 V
input range (AC) with DC coupling	-1.5 V	–	1.5 V
input range (common mode)	-3.0 V	–	3.0 V
input range (AC + common mode)	-3.3 V	–	3.3 V
dynamic reserve	–	100 dB	120 dB
common mode rejection (CMRR), for detailed information see Figure 8.9	–	75 dB	–

Table 8.6. Reference

Parameter	min	typ	max
internal reference frequency range	0.7 μHz	–	100 MHz
internal reference frequency resolution	0.7 μHz	–	–
internal reference phase range	-180 °	–	180 °
internal reference phase resolution	0.1 μ°	–	–
internal reference acquisition time (lock time)	instantaneous		
internal reference orthogonality	–	0 °	–
external reference at Input 2/Ref, signal type	arbitrary, active at rising edge		
external reference at Input 2/Ref, frequency range	1 Hz	–	50 MHz

Parameter	min	typ	max
external reference at Input 2/Ref, amplitude – note: for low-swing input signals the gain should be set to full-swing range to achieve best performance	100 mV	–	1 V
external reference at Input 2/Ref, amplitude (using HF2LI-PLL option) – note: for low-swing input signals the gain should be set to full-swing range to achieve best performance	10 mV	–	1 V
external reference at Input 2/Ref, reference acquisition time (lock time)	–	–	100 reference cycles or 1.2 ms whatever is larger
external reference at DIO0/DIO 1, signal type	digital TTL versus ground		
external reference at DIO0/DIO1, frequency range	1 Hz	–	2 MHz
external reference at DIO0/DIO1, high level	2.0 V	–	5 V
external reference at DIO0/DIO1, low level	0 V	–	0.8 V
external reference at DIO0/DIO1, reference acquisition time (lock time)	–	–	100 reference cycles or 1.2 ms whatever is larger
external reference at AUXIN1/AUXIN2, signal type	sine or rectangular		
external reference at AUXIN1/AUXIN2, frequency range	1 Hz	–	20 kHz
external reference at AUXIN1/AUXIN2, amplitude	0.5 V	–	1 V
external reference at AUXIN1/AUXIN2, reference acquisition time (lock time)	–	–	100 reference cycles
auto reference at Input 1/Input 2, signal type	AC signal with zero crossings, AC input setting		
auto reference at Input 1/Input 2, frequency range	1 Hz	–	50 MHz
auto reference at Input 1/Input 2, reference acquisition time (lock time)	–	–	100 reference cycles or 1.2 ms whatever is larger

Table 8.7. Demodulators

Parameter	Description
demodulator number	HF2IS: 4 dual-phase, 8 dual-phase with multi-frequency kit
	HF2LI: 6
	HF2PLL: 6
demodulator harmonic setting range	1 to 1023

Parameter	Description
demodulator filter time constant	0.8 μ s to 580 s
demodulator filter slope / roll-off	6, 12, 18, 24, 30, 36, 42, 48 dB/oct, consisting of up to 8 cascaded critical damping filters
demodulator output resolution	X, Y, R, Θ with 64-bit resolution
demodulator output rate (readout rate), for detailed specifications refer to Table 8.4	on Aux outputs: 921 kSamples/s on USB to host PC: maximum cumulative 700 kSamples/s
demodulator measurement bandwidth	80 μ Hz to 200 kHz
demodulator harmonic rejection	max -90 dB
demodulator sinc filter operating range	0.1 Hz to 10 kHz

Table 8.8. HF signal outputs

Parameter	min	typ	max
connectors	front-panel single-ended BNC		
output impedance (Out and Sync)	50 Ω		
input impedance (Add)	1 M Ω		
output frequency range	DC	-	50 MHz
output D/A conversion	16 bit, 210 MSamples/s		
output amplitude ranges (restrictions apply for high amplitudes and high frequencies, see Figure 8.11)	± 10 mV, ± 100 mV, ± 1 V, ± 10 V		
output maximum current	-	-	100 mA
output amplitude accuracy @ 3 MHz, < 5 V (restrictions apply for high amplitudes and high frequencies, see Figure 8.11)	-	-	1%
output total harmonic distortion THD (1 V, < 10 MHz), see Figure 8.12	-50 dB	-	-
output total harmonic distortion THD (0.1 V, < 10 MHz), see Figure 8.12	-60 dB	-	-
output noise amplitude (frequencies > 10 kHz), 50 Ω termination	-	25 nV/ $\sqrt{\text{Hz}}$	-
output phase noise @ 10 MHz, BW = 0.67 Hz, offset 100 Hz	-100 dBc/Hz	-	-
output phase noise @ 10 MHz, BW = 0.67 Hz, offset 1 kHz	-120 dBc/Hz	-	-
output offset amplitude (range setting < 1 V)	-	-	10 mV
output offset amplitude (range setting > 1 V)	-	-	200 mV
input Add signal range	-10 V	-	+10 V
input Add signal bandwidth	DC	-	50 MHz
output Sync signal range (effective range = $\pm 1 * \text{set_amplitude} / \text{set_range}$)	-1 V	-	1 V
output synchronization signal resolution	-	30 μ V	-

Table 8.9. Auxiliary Inputs and Outputs

Parameter	Description
auxiliary output connectors	front-panel single-ended BNC
auxiliary output impedance	50 Ω
auxiliary output number and type of signals	4, amplitude, phase, frequency, X/Y, manual
auxiliary output specification	± 10 V, 200 kHz, 16-bit, 921 kSamples/s
auxiliary output resolution	0.3 mV
auxiliary input connectors	back-panel single-ended BNC
auxiliary input impedance	1 MΩ
auxiliary input number	2
auxiliary input specification	± 10 V, 100 kHz, 16-bit, 400 kSamples/s
auxiliary input resolution	0.3 mV
group delay (lag time from HF input to auxiliary output)	7 μs (typical), 10 μs (maximum)

Table 8.10. Oscillator and clocks

Parameter	min	typ	max
internal oscillator frequency	–	10 MHz	–
internal oscillator output (sine)	–1 V	–	+1 V
internal oscillator initial accuracy	–	–	±30 ppm
internal oscillator aging (stability)	–	–	±5 ppm/year
internal oscillator temperature stability (23° ± 5°)	–	–	±30 ppm
UHS (option) oscillator initial accuracy	–	–	±0.5 ppm
UHS (option) oscillator aging (stability)	–	–	±0.4 ppm/year
UHS (option) oscillator temperature stability (23° ± 5°)	–	–	±0.03 ppm
UHS (option) oscillator phase noise (at 100 Hz)	–130 dBc/Hz	–	–
UHS (option) oscillator phase noise (at 1 kHz)	–140 dBc/Hz	–	–
UHS (option) oscillator reference stability (over 30 s)	0.00005 ppm	–	–
UHS (option) oscillator time to reach specification	–	–	60 s
external clock connector	back-panel single-ended BNC		
external clock input impedance	1 MΩ		
external clock input voltage	0 V	–	+3.3 V
external clock frequency	9.98 MHz	10 MHz	10.02 MHz

8.3. Digital Interface Specifications

Table 8.11. Digital interfaces

Parameter	Description
host computer connection	USB 2.0 high-speed, 480 Mbit/s
ZCtrl pre-amplifier control bus	proprietary bus to control external pre-amplifiers
ZSync synchronization bus	proprietary bus to locally interconnect ZI instruments
DIO connector	32 bit, general purpose

The DIO connector is a HD 68 pin connector, typically also used by SCSI-2 and SCSI-3 interfaces, 47 mm wide male connector. The DIO port features 16 bits that can be configured byte-wise as inputs or outputs, as well as 16 input only bits. The digital signals follow the CMOS/TTL specification.

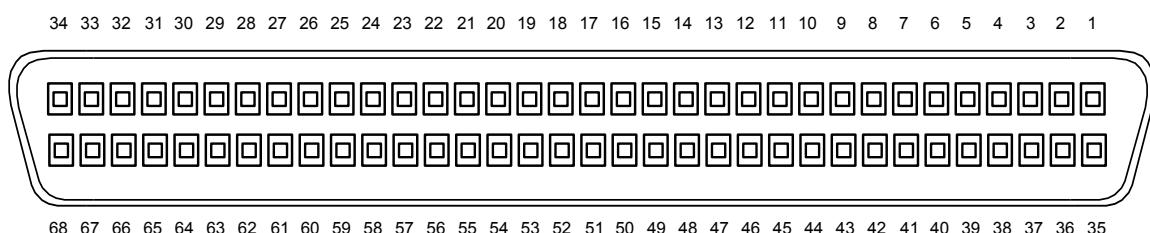


Figure 8.1. DIO HD 68 pin connector

Table 8.12. DIO pin assignment

Pin	Name	Description	Range specification
68	CLKI	clock input, used to latch signals at the digital input ports - can also be used to retrieve digital signals from the output port using an external sampling clock	5 V CMOS/TTL
67	DOL	DIO output latch, 64 MHz clock signal, the digital outputs are synchronized to the falling edge of this signal	5 V CMOS
66–51	DI[31:16]	digital input	digital input CMOS/TTL level
50–35	DIO[15:0]	digital input or output (set by user)	output CMOS 5 V, input is CMOS/TTL
34–3	GND	digital ground	–
2–1	PWR	5 V supply (100 mA max)	–

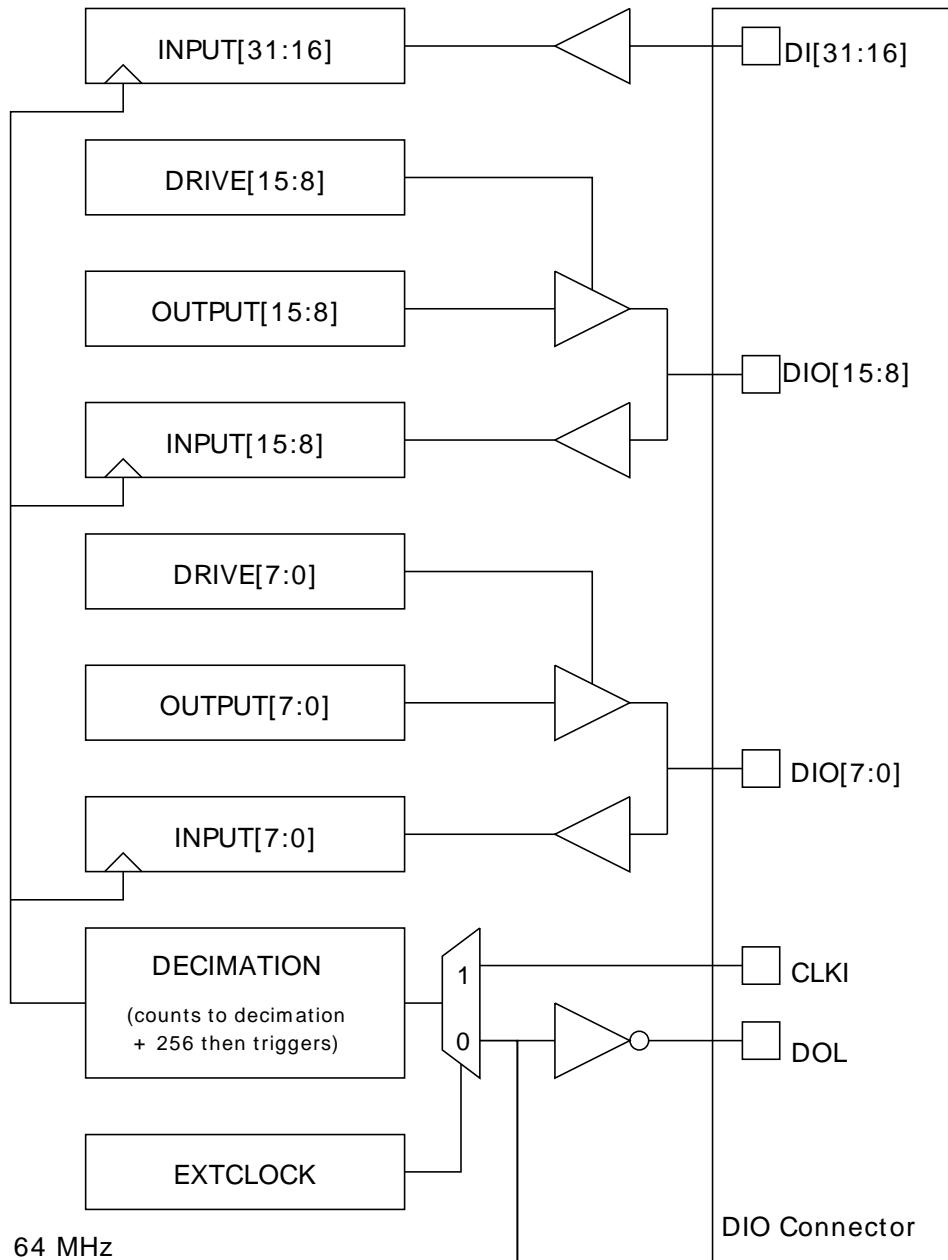


Figure 8.2. DIO input/output architecture

The HF2 Digital I/O Breakout Board provides an easy way to access all pins of the DIO Connector. The board consists of 68 pin headers and a 68-pin female socket to be connected to the HF2 using a ribbon cable. For description of the pins, refer to the [Table 8.12](#). The HF2 DIO Breakout Board is available with Zurich Instruments on demand.

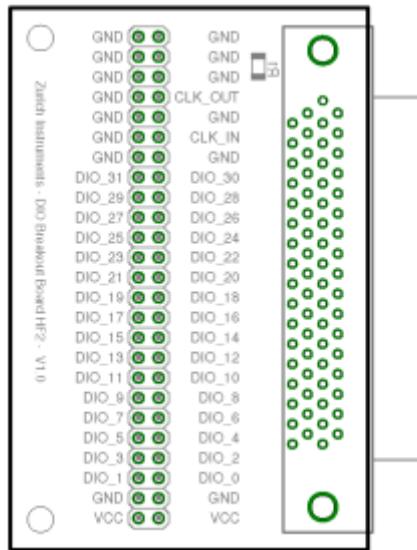


Figure 8.3. HF2 digital I/O breakout board

The internally generated 10 MHz clock is made available for external synchronization at the ZSync Out RJ45 connector. The clock signal is at pin 1, ground at pin 2. To connect: simply prepare a cable assembly that allows you to connect the 10 MHz signal from the HF2 to the BNC input of the other device external clock.

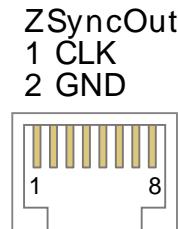
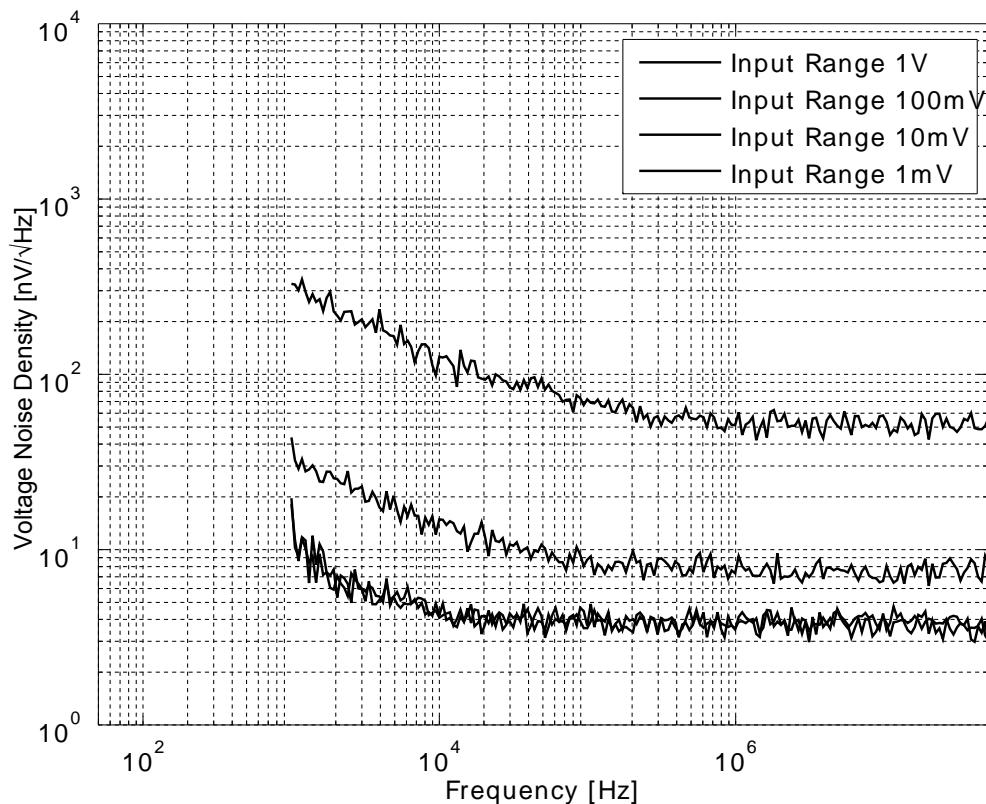


Figure 8.4. The pinout of the RJ45 jack

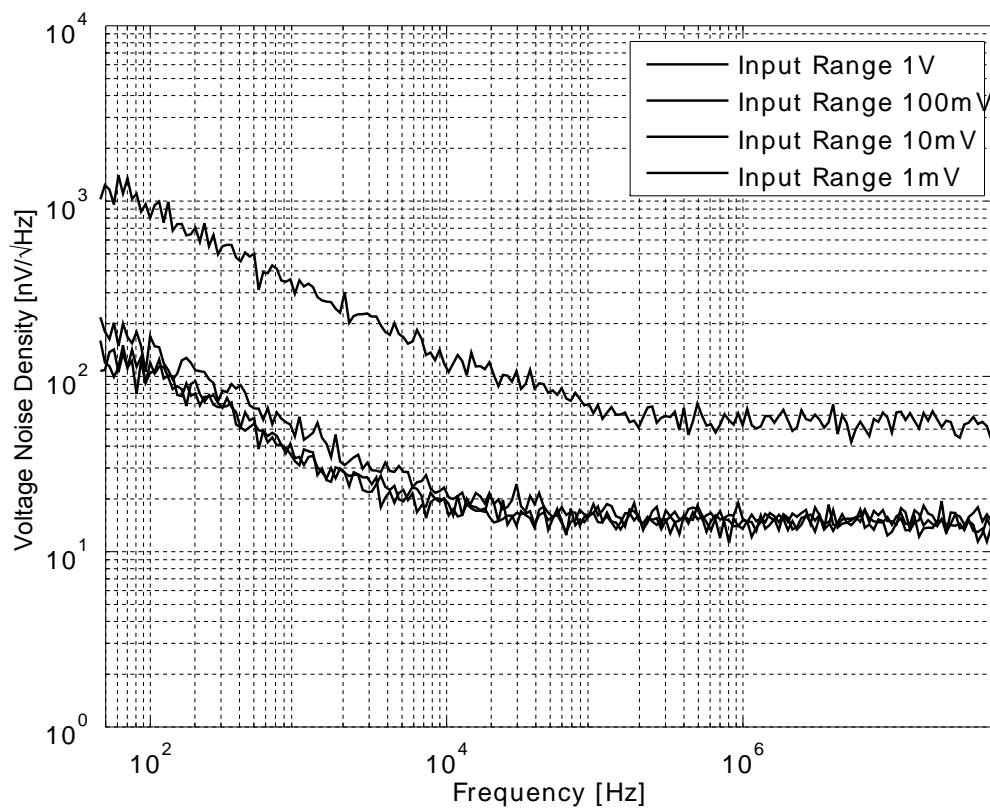
8.4. Performance Diagrams

Many parameters mentioned in [Section 8.2](#) are valid without specific conditions. Other parameters instead are typical specifications because they depend on several parameters, such as range settings, and frequency. This section completes the previous chapters with detailed performance diagrams in order to support the validation of applications.



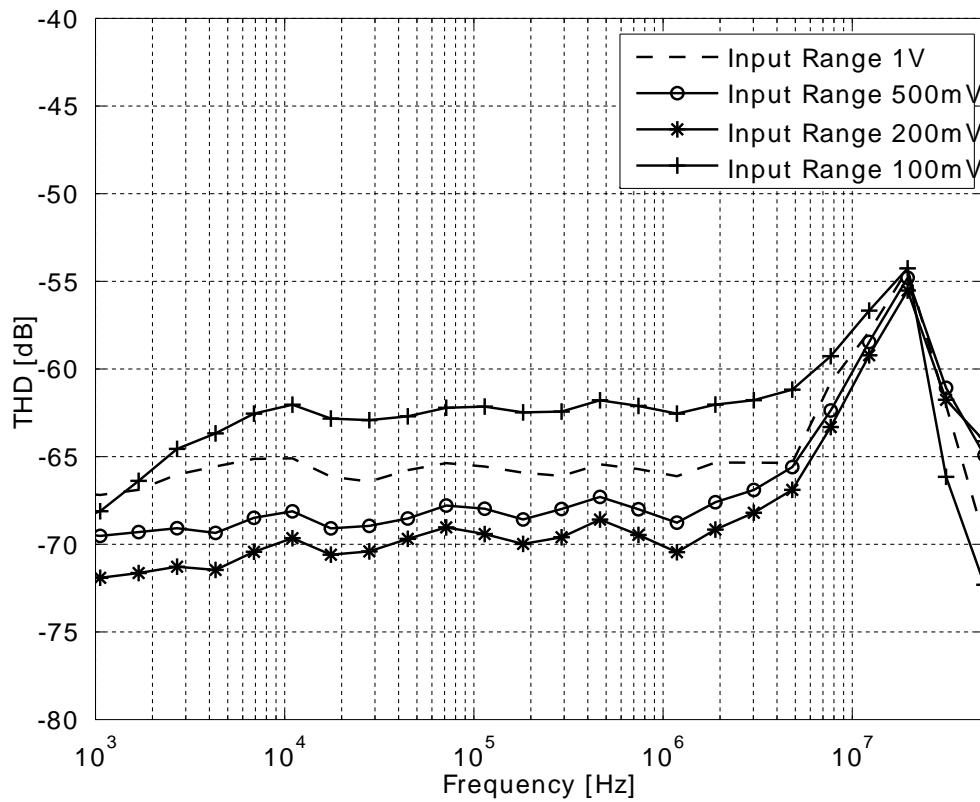
Input noise amplitude depends on several parameters, and in particular on the frequency and the setting for the input range. The noise is lower for smaller input ranges, and it is recommended to perform noise measurements with the AC coupling setting. In AC coupling mode, both 10 mV and 1 mV signal ranges have the same input noise performance. The corner frequency of the 1/f noise is in the range of 10 kHz and the white noise floor is around 5 nV/ $\sqrt{\text{Hz}}$ in AC coupling mode.

Figure 8.5. HF input noise with AC coupling



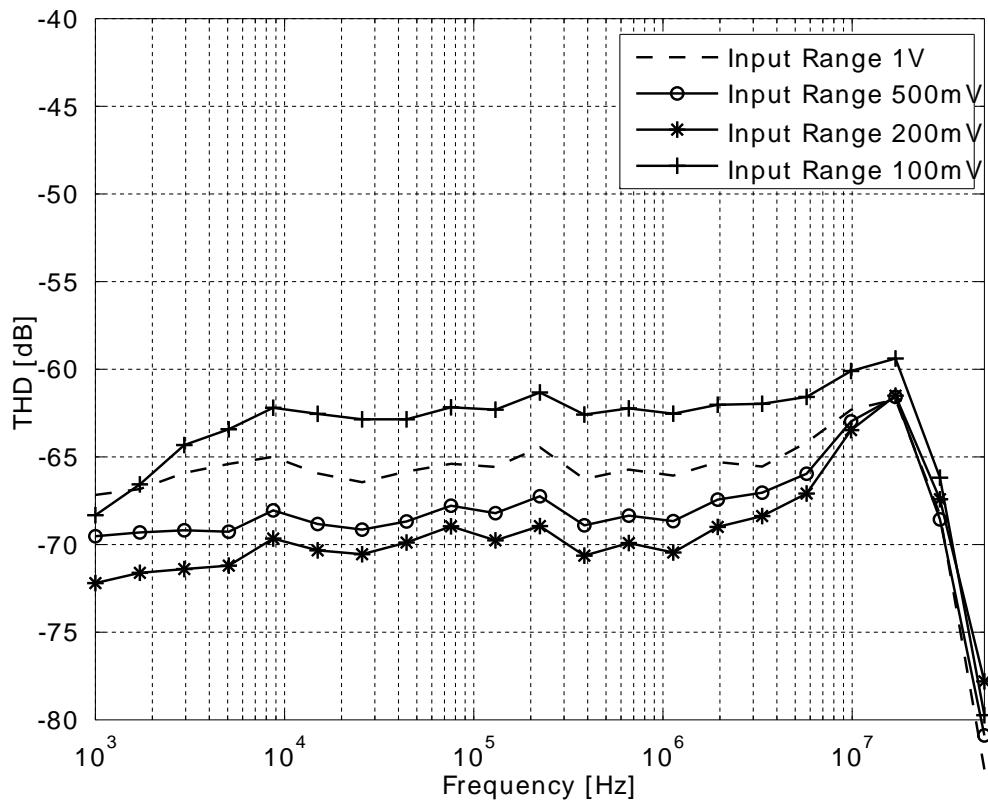
Input noise amplitude depends on several parameters, and in particular on the frequency and the setting for the input range. The noise is lower for smaller input ranges. The corner frequency for AC coupling is at 1 kHz. Therefore, for noise measurements above 1 kHz, AC coupling should be used, for noise measurements below 1 kHz, DC coupling should be used. In DC coupling mode, the input noise for low ranges is limited by the coupling selection. The corner frequency of the 1/f noise is in the range of 10 kHz and the white noise floor is around 15 nV/ $\sqrt{\text{Hz}}$ in DC coupling mode.

Figure 8.6. HF input noise with DC coupling



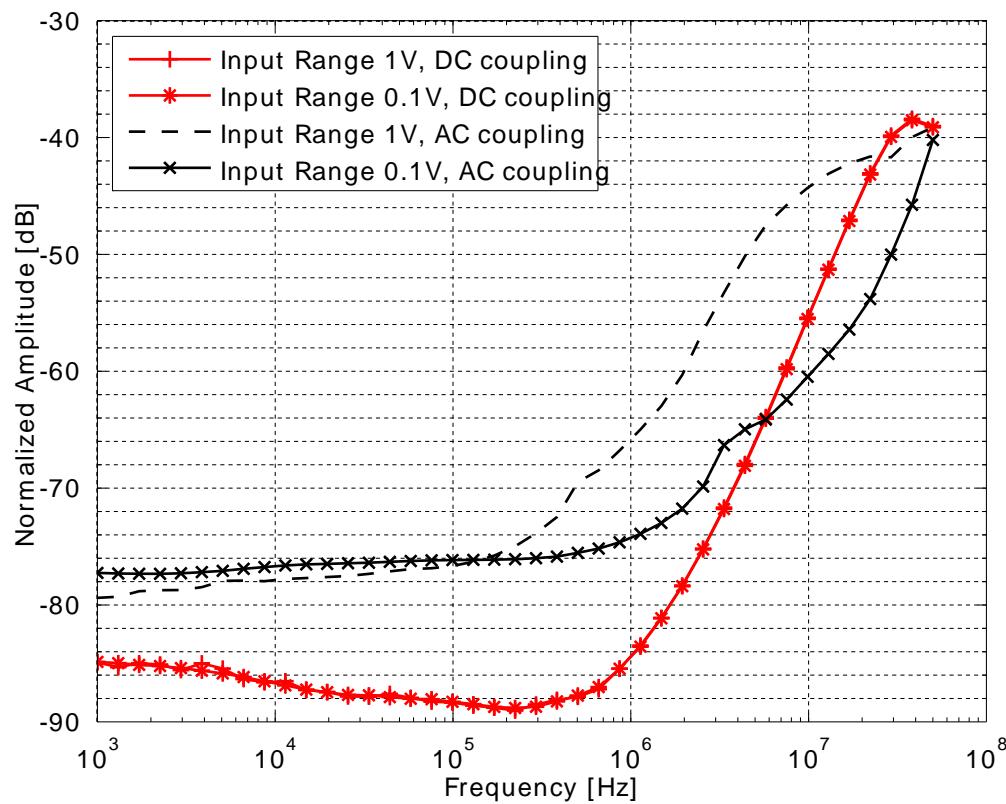
Input total harmonic distortion (THD) depends on several parameters, and in particular on the frequency and the setting for the input range. The test is performed with the input amplitude at 50% of the range setting. For frequencies below 5 MHz input THD is below -60 dB for any range setting. For frequencies above 10 MHz the AC coupling mode inserts about 5 dB more distortion than the DC mode. The total harmonic distortion is calculated from the measurement of the second and the third harmonic.

Figure 8.7. HF input total harmonic distortion with AC coupling



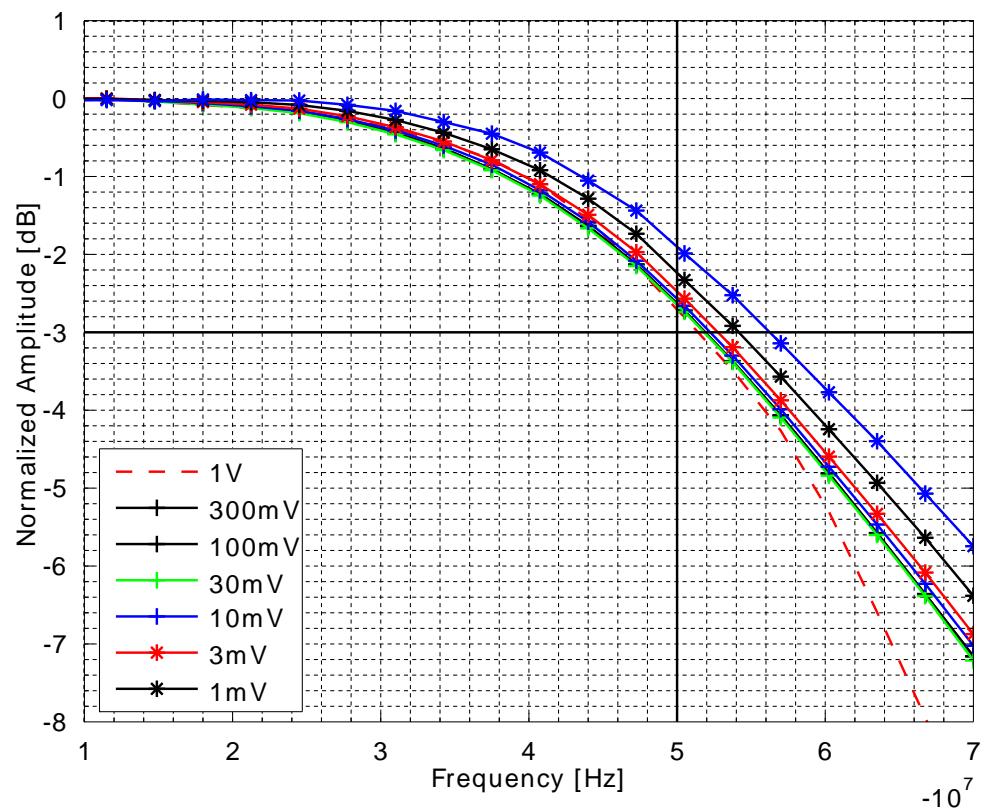
Input total harmonic distortion (THD) depends on several parameters, and in particular on the frequency and the setting for the input range. The test is performed with the input amplitude at 50% of the range setting. For frequencies below 10 MHz input THD is below –60 dB for any range setting. Smaller range settings can be as good as –70 dB. The total harmonic distortion is calculated from the measurement of the second and the third harmonic.

Figure 8.8. HF input total harmonic distortion with DC coupling



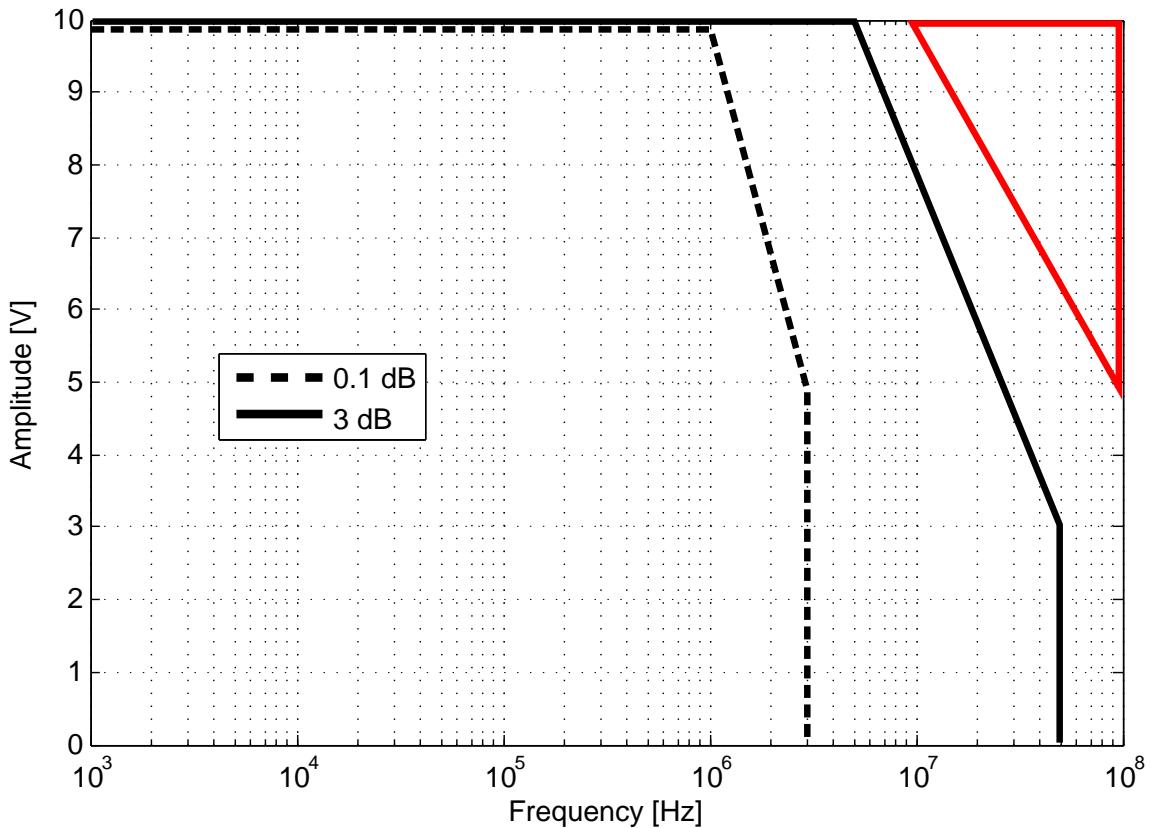
Input common mode rejection ratio (CMRR) of the signal input for different frequencies and different input amplitudes. For this test the same 2 V_{pp} signal is applied to the differential inputs on the instrument and the differential measurement is captured. The CMRR is better in DC mode for frequencies up to 7 MHz. Above 7 MHz, the CMRR in AC mode is better suited, but only for amplitudes up to 100 mV.

Figure 8.9. HF input common mode rejection ratio



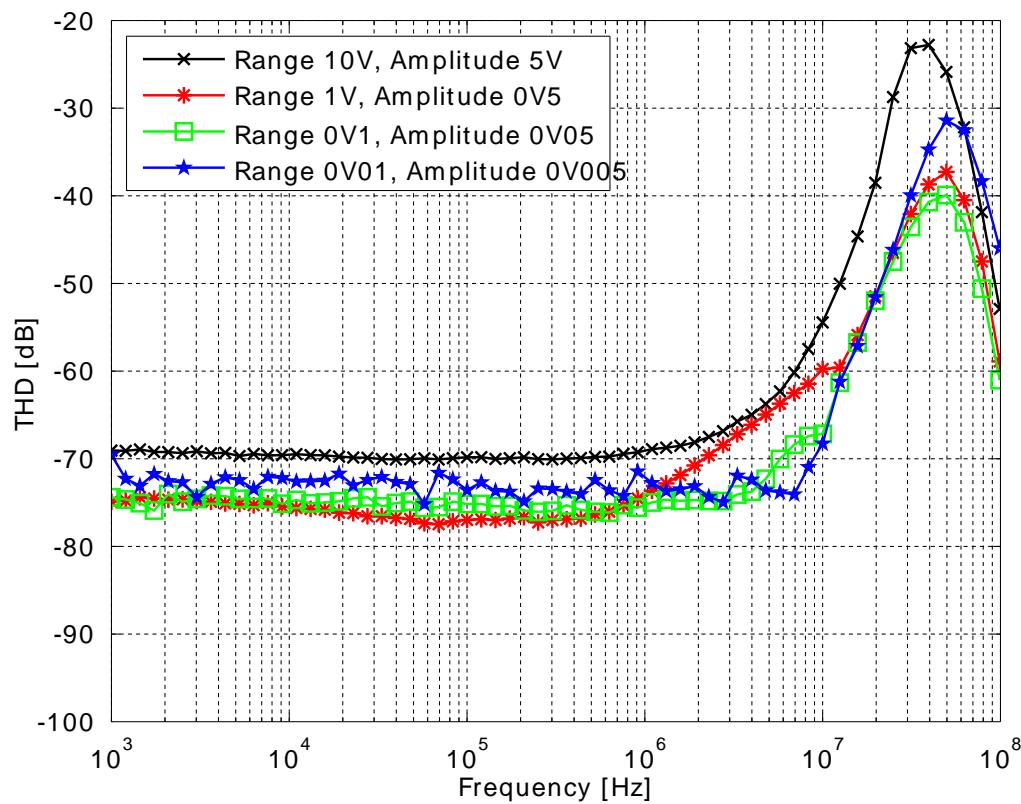
Signal input bandwidth versus frequency for different range settings. The vertical solid black line marks the 50 MHz specification and the horizontal solid black line corresponds to 3 dB attenuation. The bandwidth is between 50 MHz and 60 MHz for all displayed range settings (intermediate range selections can be chose in the graphical user interface).

Figure 8.10. HF input bandwidth



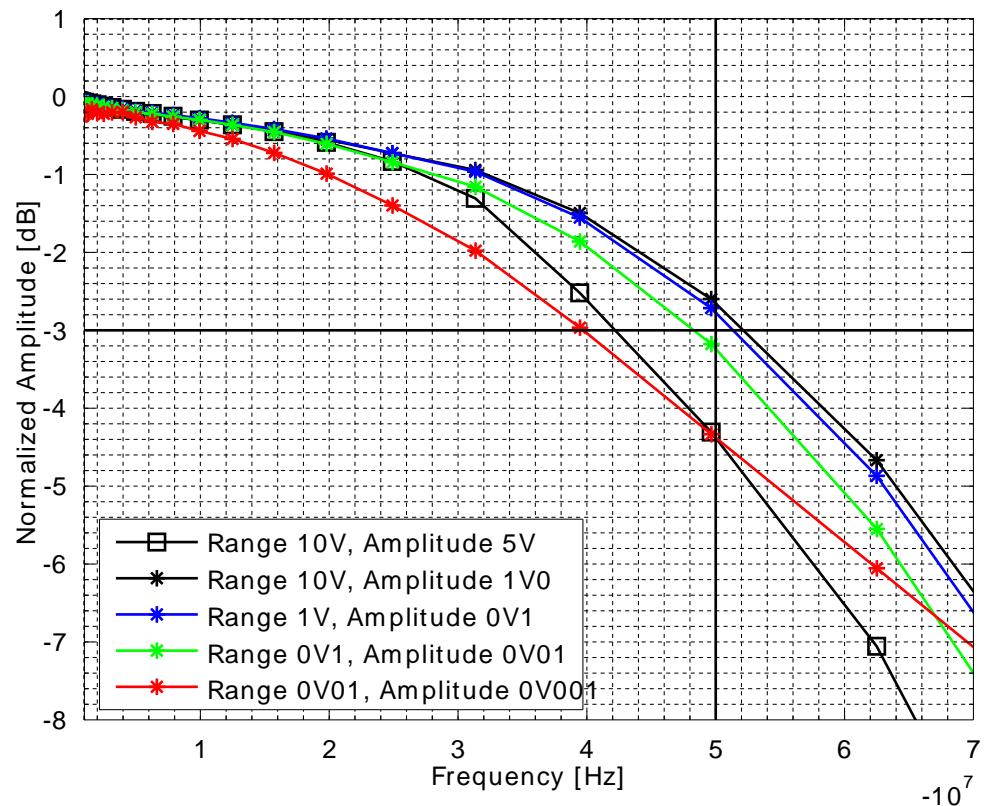
The amplitude accuracy of the output signal generator depends on the frequency and the amplitude. The plot shows the output amplitude accuracy region for 0.1 dB (roughly 1%) and 3 dB (around 40%) of the expected value. The measurement was performed with $50\ \Omega$ load and is better for larger loads. The plot shows that the signal amplitude up to 1 MHz is accurate at 0.1 dB at any output voltage (area within dashed line). Above this frequency, the anti-aliasing filters at 50 MHz impact the absolute value of the generated signal: the output amplitude is reduced. Above 5 MHz the maximum amplitude is below 10 V, and decreases further reaching 3 V_{RMS} at 50 MHz (area within solid line). It is not recommended to operate the instrument at high amplitudes (above 5 V_{RMS}) and high frequencies (above 10 MHz) because of the large signal distortion and potential long-term damage to the output drivers (area within solid red line).

Figure 8.11. HF output amplitude accuracy



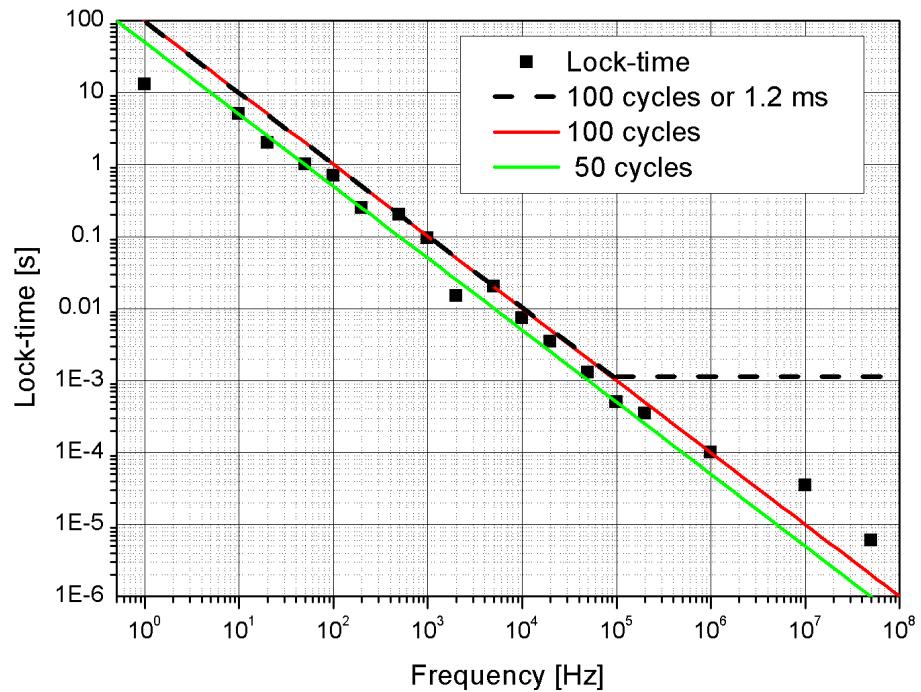
The total harmonic distortion of the signal outputs was measured with $50\ \Omega$ load and by applying amplitude at 50% of the corresponding range. The THD is around 75 dB for signals below 0.5 V, and increases for larger amplitudes. The distortion increases significantly at frequencies above 10 MHz. The total harmonic distortion is calculated from the measurement of the second and the third harmonic.

Figure 8.12. HF output total harmonic distortion



The vertical solid black line marks the 50 MHz specification and the horizontal solid black line corresponds to 3 dB attenuation. The bandwidth depends on the range and amplitude. The measurement is performed with $50\ \Omega$ load.

Figure 8.13. HF output bandwidth



The dashed black line marks the specified lock time corresponding to 100 reference cycles or 1.2 ms. The 100 and 50 cycle are also plotted for convenience. The graph indicates that the lock time of the HF2 Instruments follows the 100 reference cycles spec, is slightly worse for frequencies above 1 MHz, and is better for frequency below 1 Hz.

Figure 8.14. Lock time

8.5. Ground and Earth Scheme

Ground loops have effect to introduce noise at the line frequency (mostly 50/60 Hz) and higher harmonics, and aliasing in the demodulated signal that is measurable for frequencies up to 10 MHz. Some lock-in amplifiers implement a line filter which has the effect to exclude low frequency measurements. This is not the case for the HF2 Instruments where an effective ground strategy is implemented.

In order to suppress large signal components at line frequency and higher harmonics avoiding ground loops within the measurement setup is required. Possible reasons for line frequency components include parasitics resistances between the different signal grounds, inductive coupling from line transformers and other electrical apparatus into the signal paths, and pre-amplifiers that generate additional loops.

Counter measures are to break loops, using differential wiring, implementing star ground connections in the measurement setup, with the main ground closest to the setup as possible, connect all instrument casing to earth, and using optocouplers and transformers where a galvanic decoupling is provided in the signal path.

The grounding of the HF2 Instrument is implemented connecting analog ground and digital ground in a star network. This reduces the digital ground noise that flows into the analog domain considerably. All analog grounds are connected together before they are connected to the digital ground (e.g. USB ground). All grounds are decoupled by the Earth by means of a $1\text{ M}\Omega$ resistor, which is however generally shorted by a PC connected by means of a USB cable. The earth connection of the power plug connects at the same time the chassis and the banana plug on the rear Instrument panel.

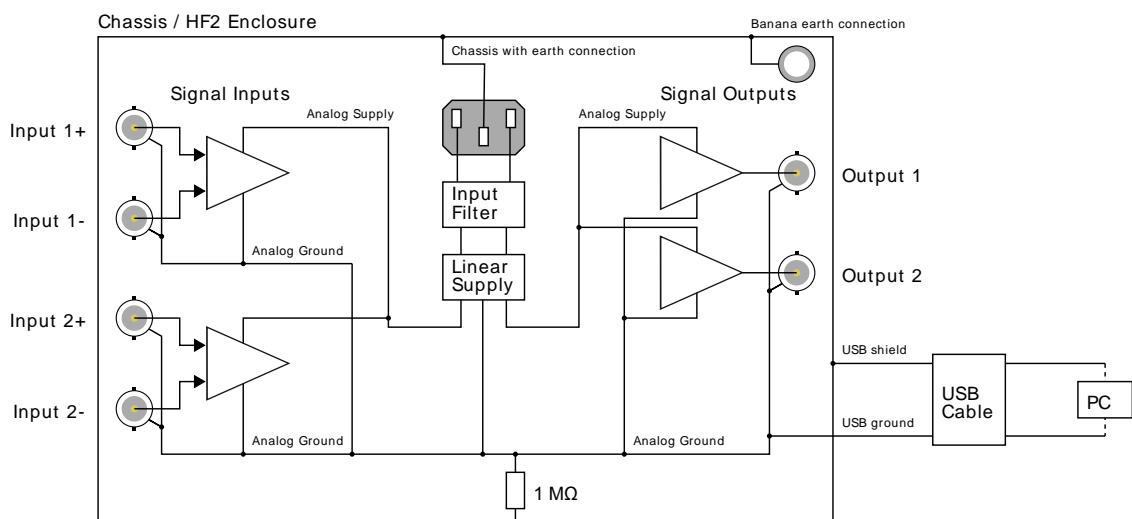


Figure 8.15. Instrument ground & earth connection scheme

Some applications however require floating ground applications, it is suggested to make use of the differential inputs, by connecting the BNC shield to the negative BNC connector. The limitation for this strategy, is that the floating ground should not exceed the specified maximum input common mode offset.

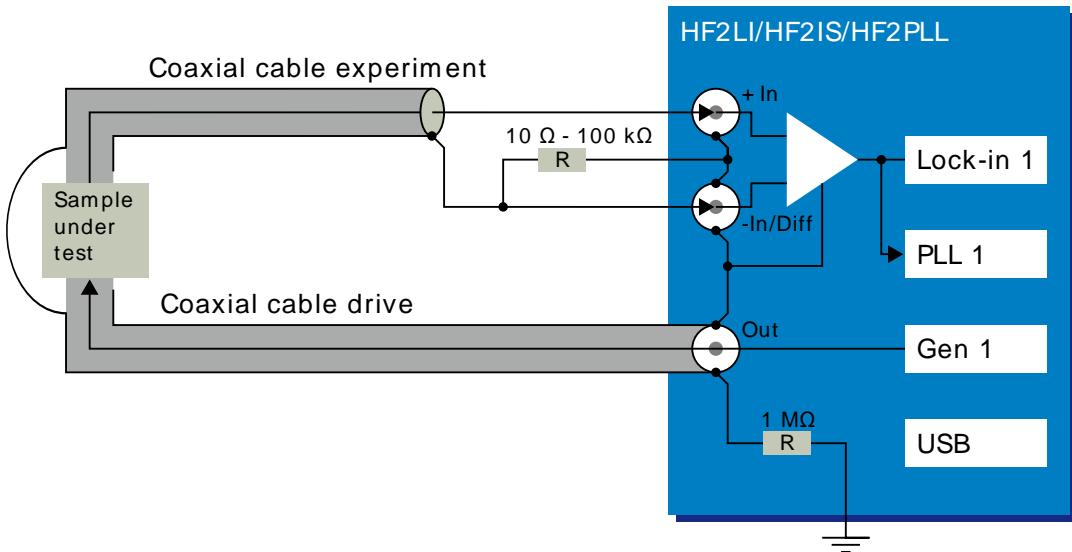
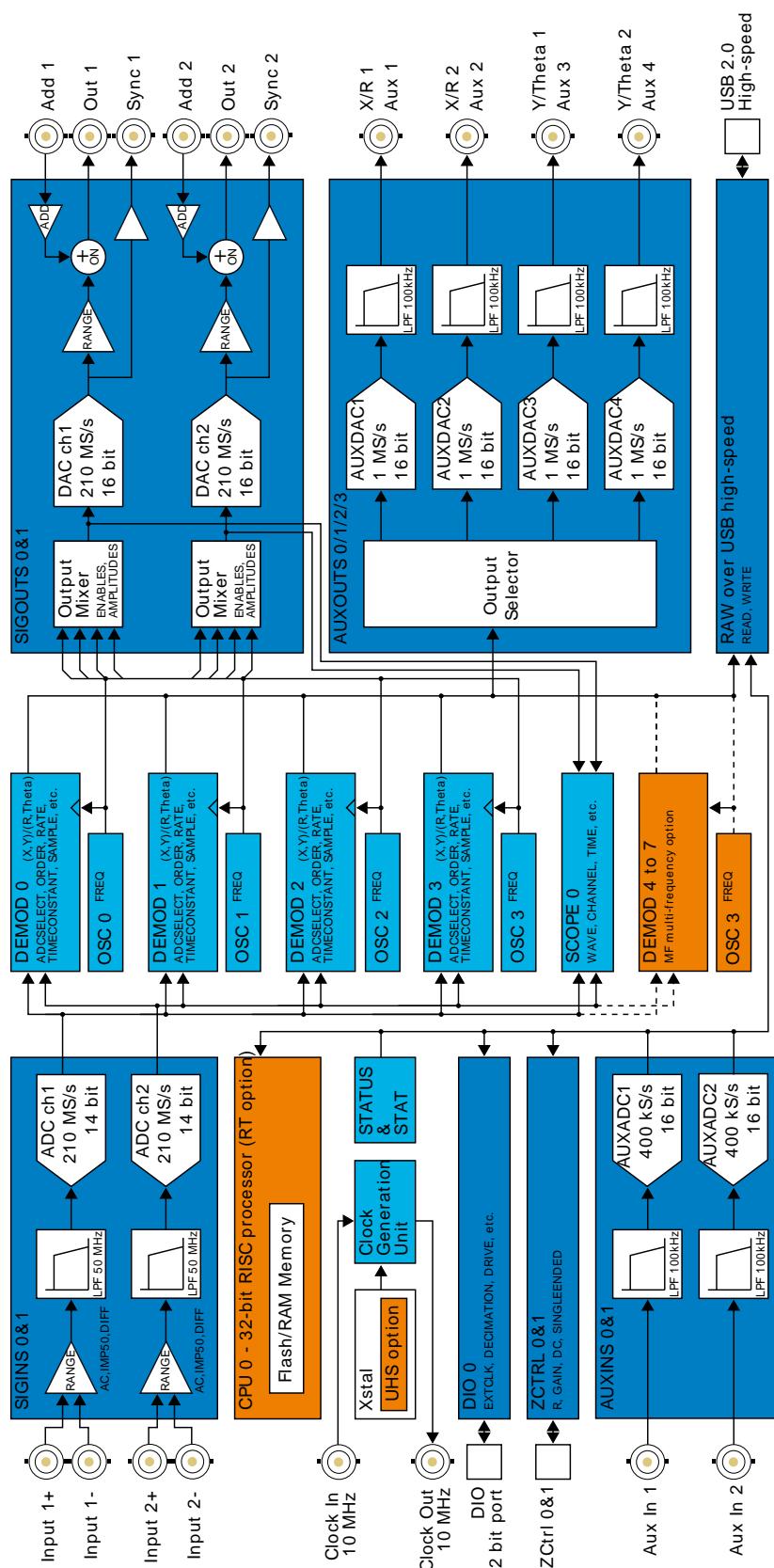
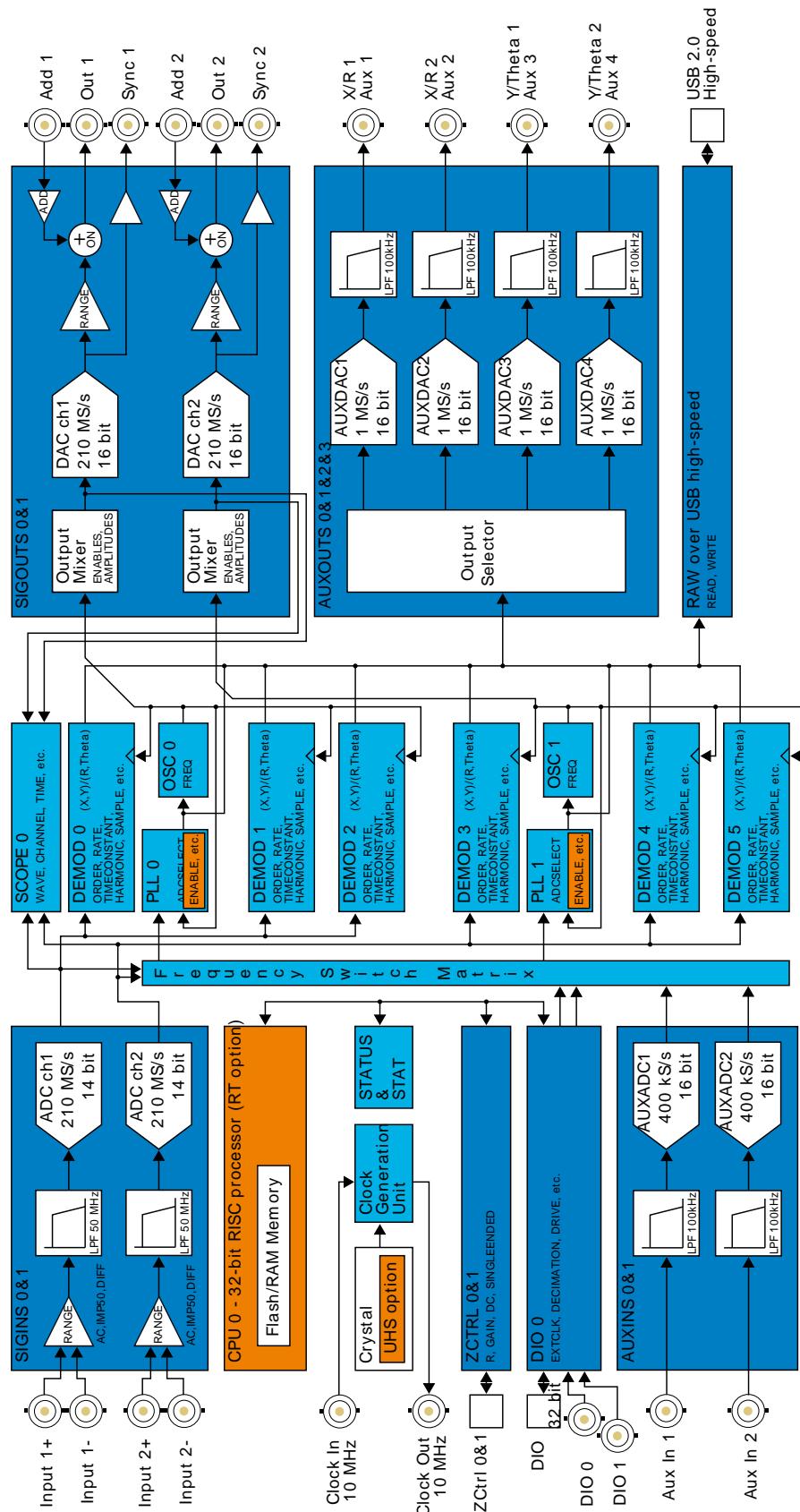


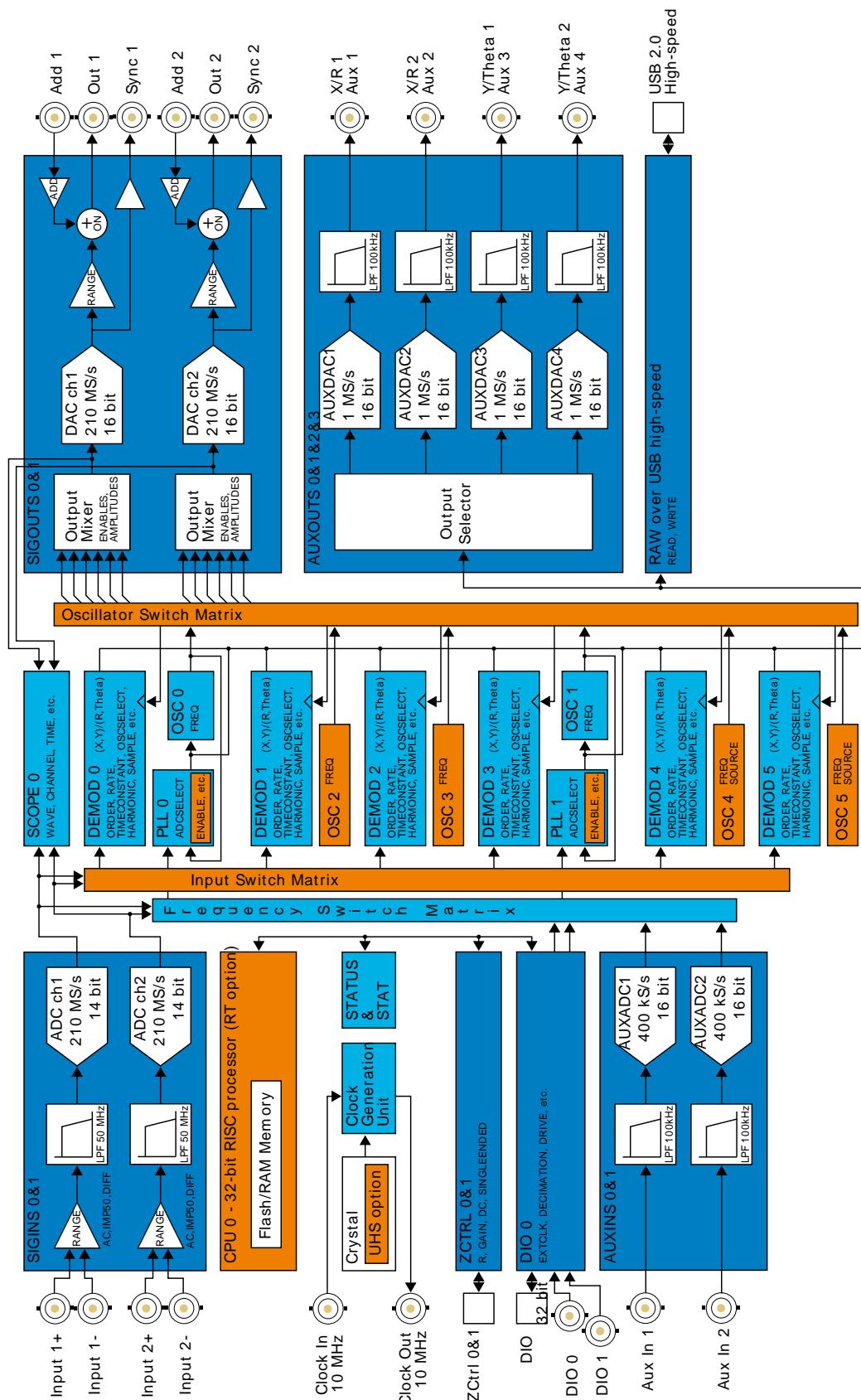
Figure 8.16. Differential connection scheme reducing ground loops

8.6. Reference Images

The following figures are intended for advanced users with programming projects on the HF2 Instruments.







8.7. Test Specifications

Users are encouraged to verify that the Instrument performs as specified, not only after shipping but also to ensure continuous performance over time. Recommended procedures for measuring key specification parameters are based on the Zurich Instruments ziControl software which is included in the LabOne software package. For detailed instructions please refer to the Specifications chapter of the HF2 User Manual (ziControl Edition). Measurement procedures compatible with the LabOne graphical UI will be published with a subsequent LabOne release.

Chapter 9. Signal Processing Basics

This chapter provides insights about several lock-in amplifier principles not necessarily linked to a specific instrument from Zurich Instruments. Since the appearance of the first valve-based lock-in amplifiers in the 1930s the physics have not changed, but the implementation and the performance have evolved greatly. Many good lock-in amplifier primers have appeared in the past decades, and some of them appear outdated now because they were written with analog instruments in mind. This section does not aim to replace any existing primer, but to complete them with a preferred emphasis on digital lock-in amplifiers.

The first subsection describes the principles of lock-in amplification, followed by the description of the function of discrete-time filters. After, we discuss the definition of the full range sensitivity, a specification parameter particularly important for analog lock-in amplifiers but with somewhat reduced importance for digital instruments. In the following, we describe the function and use of sinc filtering in particular for low-frequency lock-in measurements. The last section is dedicated to the zoom FFT feature. Innovative in the context of lock-in amplifiers, zoom FFT offers a fast and high-resolution spectral analysis around the lock-in operation frequency.

9.1. Principles of Lock-in Detection

Lock-in demodulation is a technique to measure the amplitude A_s and the phase Θ of a periodic signal with the frequency $\omega_s = 2\pi f_s$ by comparing the signal to a reference signal. This technique is also called phase-sensitive detection. By averaging over time the signal-to-noise ratio (SNR) of a signal can be increased by orders of magnitude, allowing very small signals to be detected with a high accuracy making the lock-in amplifier a tool often used for signal recovery. For both signal recovery and phase-sensitive detection, the signal of interest is isolated with narrow band-pass filtering therefore reducing the impact of noise in the measured signal.

Figure 9.1 shows a basic measurement setup: a reference V_r signal is fed to the device under test. This reference signal is modified by the generally non-linear device with attenuation, amplification, phase shifting, and distortion, resulting in a signal $V_s = A_s \cos(\omega_s t + \Theta_s)$ plus harmonic components.

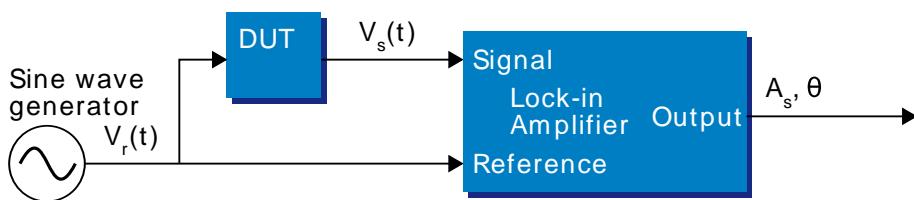


Figure 9.1. Basic measurement setup incorporating a lock-in amplifier

For practical reasons, most lock-in amplifiers implement the band-pass filter with a mixer and a low-pass filter (depicted in Figure 9.2): the mixer shifts the signal of interest into the baseband, ideally to DC, and the low-pass filter cuts all unwanted higher frequencies.

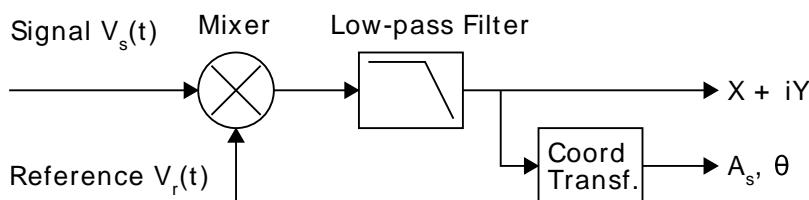


Figure 9.2. Mixing and low-pass filtering performed by the lock-in amplifier

The input signal $V_s(t)$ is multiplied by the reference signal $V_r(t) = \sqrt{2} e^{i\omega_r t}$, where $\omega_r = 2\pi f_r$ is the demodulation frequency and i is the imaginary unit. This is the complex representation of a sine and cosine signal (phase shift 90°) forming the components of a quadrature demodulator, capable of measuring both the amplitude and the phase of the signal of interest. In principle it is possible to multiply the signal of interest with any frequency, resulting in a heterodyne operation. However the objective of the lock-in amplifier is to shift the signal as close as possible to DC, therefore the frequency of the reference and the signal is chosen similar. In literature this is called homodyne detection, synchrodyne detection, or zero-IF direct conversion.

The result of the multiplication is the signal

$$V_s(t) \cdot V_r(t) = V_s(t) \cdot \sqrt{2} e^{i\omega_r t} = \frac{A_s}{\sqrt{2}} e^{i[(\omega_s - \omega_r)t + \Theta]} + \frac{A_s}{\sqrt{2}} e^{i[(\omega_s + \omega_r)t + \Theta]}$$

Equation 9.1. Multiplication of signal of interest with reference signal

It consists of a slow component with frequency $\omega_s - \omega_r$ and a fast component with frequency $\omega_s + \omega_r$.

The demodulated signal is then low-pass filtered with an infinite impulse response (IIR) RC filter, indicated by the symbol $\{ \cdot \}$. The frequency response of the filter $F(\omega)$ will let pass the low frequencies $F(\omega_s - \omega_r)$ while considerably attenuating the higher frequencies $F(\omega_s + \omega_r)$. Another way to consider the low-pass filter is an averager.

$$X + iY = \left\langle V_s(t) \cdot \sqrt{2} e^{i\omega_r t} \right\rangle \approx F(\omega_s - \omega_r) \frac{A_s}{\sqrt{2}} e^{i[(\omega_s - \omega_r)t + \Theta]}$$

Equation 9.2. Averaging the result of the signal multiplication

The result after the low-pass filter is the demodulated signal $X+iY$, where X is the real and Y is the imaginary part of a signal depicted on the complex plane. These components are also called in-phase and quadrature components. The transformation of X and Y into the amplitude R and phase Θ information of $V_s(t)$ can be performed with trigonometric operations.

It is interesting to note that the value of the measured signal corresponds to the RMS value of the signal, which is equivalent to $R = A_s/\sqrt{2}$.

Most lock-in amplifiers output the values (X, Y) and (R, Θ) encoded in a range of -10 V to $+10 \text{ V}$ of the auxiliary output signals.

9.1.1. Lock-in Amplifier Applications

Lock-in amplifiers are employed in a large variety of applications. In some cases the objective is measuring a signal with good signal-to-noise ratio, and then that signal could be measured even with large filter settings. In this context the word phase sensitive detection is appropriate. In other applications, the signal is very weak and overwhelmed by noise, which forces to measure with very narrow filters. In this context the lock-in amplifier is employed for signal recovery. Also, in another context, a signal modulated on a very high frequency (GHz or THz) that cannot be measured with standard approaches, is mixed to a lower frequency that fits into the measurement band of the lock-in amplifier.

One example for measuring a small, stationary or slowly varying signal which is completely buried in the $1/f$ noise, the power line noise, and slow drifts. For this purpose a weak signal is modulated to a higher frequency, away from these sources of noise. Such signal can be efficiently mixed back and measured in the baseband using a lock-in amplifier. In [Figure 9.3](#) this process is depicted. Many optical applications perform the up-mixing with a chopper, an electro-optical modulator, or an acousto-optical modulator. The advantage of this procedure is that the desired signal is measured in a spectral region with comparatively little noise. This is more efficient than just low-pass filtering the DC signal.

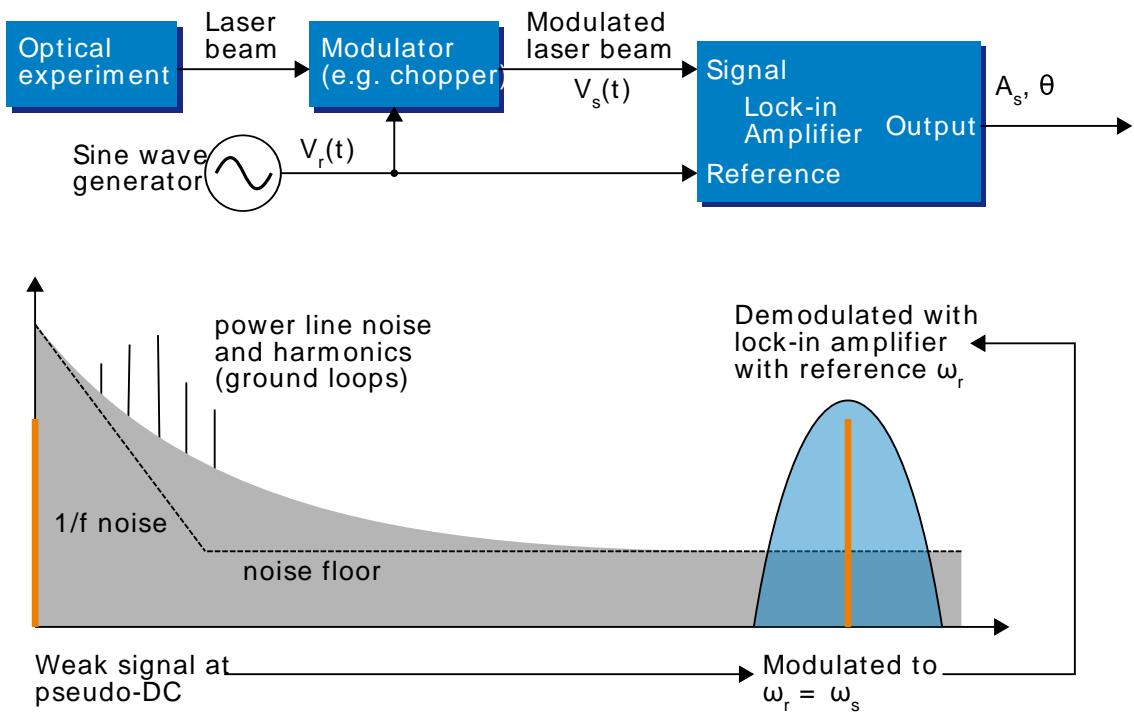


Figure 9.3. Lock-in measurement of a noisy DC signal

9.2. Signal Bandwidth

The signal bandwidth (BW) theoretically corresponds to the highest frequency components of interest in a signal. In practical signals, the bandwidth is usually quantified by the cut-off frequency. It is the frequency at which the transfer function of a system shows 3 dB attenuation relative to DC ($BW = f_{\text{cut-off}} = f_{-3\text{dB}}$); that is, the signal power at $f_{-3\text{dB}}$ is half the power at DC. The bandwidth, equivalent to cut-off frequency, is used in the context of dynamic behavior of a signals or separation of different signals. This is for instance the case for fast-changing amplitudes or phase values like in a PLL or in a imaging application, or when signals closely spaced in frequency need to be separated.

The noise equivalent power bandwidth (NEPBW) is also a useful figure, and it is distinct from the signal bandwidth. This unit is typically used for noise measurements: in this case one is interested in the total amount of power that passes through a low-pass filter, equivalent to the area under the solid curve in [Figure 9.4](#). For practical reasons, one defines an ideal brick-wall filter that lets pass the same amount of power under the assumption that the noise has a flat (white) spectral density. This brick-wall filter has transmission 1 from DC to f_{NEPBW} . The orange and blue areas in [Figure 9.4](#) then are exactly equal in a linear scale.

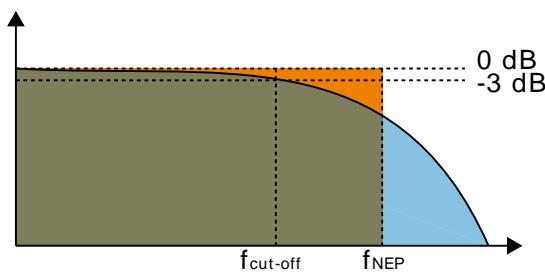


Figure 9.4. Signal bandwidth and noise equivalent power bandwidth

It is possible to establish a simple relation between the $f_{\text{cut-off}}$ and the f_{NEPBW} that only depends on the slope (or roll-off) of the filter. As the filter slope actually depends on the time constant (TC) defined for the filter, it is possible to establish the relation also to the time constant. It is intuitive to understand that for higher filter orders, the $f_{\text{cut-off}}$ is closer to the f_{NEPBW} than for smaller orders.

The time constant is a parameter used to interpret the filter response in the time domain, and relates to the time it takes to reach a defined percentage of the final value. The time constant of a low-pass filter relates to the bandwidth according to the formula

$$TC = \frac{FO}{2\pi f_{\text{cut-off}}} \quad (9.3)$$

where FO is said factor that depends on the filter slope. This factor, along with other useful conversion factors between different filter parameters, can be read from the following table.

Table 9.1. Summary of conversion factors for bandwidth definitions

filter order	filter roll-off	FO	$f_{\text{cut-off}}$	f_{NEPBW}	$f_{\text{NEPBW}} / f_{\text{cut-off}}$
1 st	6 dB/oct	1.000	0.159 / TC	0.250 / TC	1.57
2 nd	12 dB/oct	0.644	0.102 / TC	0.125 / TC	1.22
3 rd	18 dB/oct	0.510	0.081 / TC	0.094 / TC	1.15
4 th	24 dB/oct	0.435	0.068 / TC	0.078 / TC	1.12
5 th	30 dB/oct	0.386	0.062 / TC	0.068 / TC	1.11

9.2. Signal Bandwidth

filter order	filter roll-off	F0	$f_{\text{cut-off}}$	f_{NEPBW}	$f_{\text{NEPBW}} / f_{\text{cut-off}}$
6 th	36 dB/oct	0.350	0.056 / TC	0.062 / TC	1.10
7 th	42 dB/oct	0.323	0.051 / TC	0.056 / TC	1.10
8 th	48 dB/oct	0.301	0.048 / TC	0.052 / TC	1.09

9.3. Discrete-Time Filters

9.3.1. Discrete-Time RC Filter

There are many options how to implement digital low-pass filters. One common filter type is the exponential running average filter. Its characteristics are very close to those of an analog resistor-capacitor RC filter, which is why this filter is sometimes called a discrete-time RC filter. The exponential running average filter has the time constant $TC = \tau_N$ as its only adjustable parameter.

It operates on an input signal $X_{in}[n, T_S]$ defined at discrete times $nT_S, (n+1)T_S, (n+2)T_S$, etc., spaced at the sampling time T_S . Its output $X_{out}[n, T_S]$ can be calculated using the following recursive formula,

$$X_{out}[n, T_S] = e^{-T_S/\tau_N} X_{out}[n-1, T_S] + (1 - e^{-T_S/\tau_N}) X_{in}[n, T_S]$$

Equation 9.4. Time domain response of the discrete-time RC filter

The response of that filter in the frequency domain is well approximated by the formula

$$H_1(\omega) = \frac{1}{1 + i \cdot \omega \cdot \tau_N}$$

Equation 9.5. Frequency domain response of the first-order discrete-time RC filter

The exponential filter is a first-order filter. Higher-order filters can easily be implemented by cascading several filters. For instance the 4th order filter is implemented by chaining 4 filters with the same time constant $TC = \tau_N$ one after the other so that the output of one filter stage is the input of the next one. The transfer function of such a cascaded filter is simply the product of the transfer functions of the individual filter stages. For an n-th order filter, we therefore have

$$H_n(\omega) = \frac{1}{(1 + i \cdot \omega \cdot \tau_N)^n}$$

Equation 9.6. Frequency domain response of the n-th order discrete-time RC filter

The attenuation and phase shift of the filters can be obtained from this formula. Namely, the filter attenuation is given by the absolute value squared $|H_n(\omega)|^2$. The filter transmission phase is given by the complex argument $\arg[H_n(\omega)]$.

9.3.2. Filter Settling Time

The low-pass filters after the demodulator cause a delay to measured signals depending on the filter order and time constant $TC = \tau_N$. After a change in the signal, it will therefore take some time before the lock-in output reaches the correct measurement value. This is depicted in Figure 9.5 where the response of cascaded filters to a step input signal this is shown.

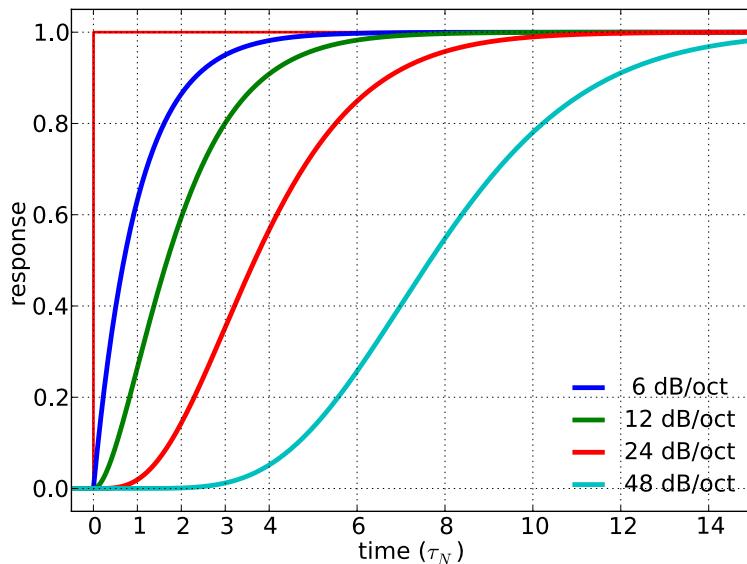


Figure 9.5. Time domain step response of the RC low-pass filters

More quantitative information on the settling time can be obtained from [Table 9.2](#). In this table, you find settling times in units of the filter time constant τ_N for all filter orders available with the HF2 Lock-in Amplifier. The values tell the time you need to wait for the filtered demodulator signal to reach 5%, 95% and 99% of the final value. This can help in making a quantitatively correct choice of filter parameters for example in a measurement involving a parameter sweep.

Table 9.2. Summary of Filter Rise Times

filter order	Setting time to		
	5%	95%	99%
1 st	0.025 · TC	3.0 · TC	4.6 · TC
2 nd	0.36 · TC	4.7 · TC	6.6 · TC
3 rd	0.82 · TC	6.3 · TC	8.4 · TC
4 th	1.4 · TC	7.8 · TC	10 · TC
5 th	2.0 · TC	9.2 · TC	12 · TC
6 th	2.6 · TC	11 · TC	12 · TC
7 th	3.3 · TC	12 · TC	15 · TC
8 th	4.0 · TC	13 · TC	16 · TC

9.4. Full Range Sensitivity

The sensitivity of the lock-in amplifier is the RMS value of an input sine that is demodulated and results in a full scale analog output. Traditionally the X, Y, or R components are mapped onto the 10 V full scale analog output. In such a case, the overall gain from input to output of the lock-in amplifier is composed of the input and output amplifier stages. Many lock-in amplifiers specify a sensitivity between 1 nV and 1 V. In other words the instrument permits an input signal between 1 nV and 1 V to be amplified to the 10 V full range output.

Analog Lock-in Amplifiers:



Digital Lock-in Amplifiers:



Figure 9.6. Sensitivity from signal input to signal output

In analog lock-in amplifiers the sensitivity is simple to understand. It is the sum of the analog amplification stages between the input and the output of the instrument: in particular the input amplifier and the output amplifier.

In digital lock-in amplifiers the sensitivity less straightforward to understand. Analog-to-digital converters (ADC) operate with a fixed input range (e.g. 1 V) and thus require a variable-gain amplifier to amplify the input signal to the range given by the ADC. This variable-gain amplifier must be in the analog domain and its capability determines the minimum input range of the instrument. A practical analog input amplifier provides a factor 1000 amplification, thus 1 V divided by 1000 is the minimum input range of the instrument.

The input range is the maximum signal amplitude that is permitted for a given range setting. The signal is internally amplified with the suited factor, e.g. (1 mV)·1000 to result in a full swing signal at the ADC. For signals larger than the range, the ADC saturates and the signal is distorted – the measurement result becomes useless. Thus the signal should never exceed the range setting.

But the input range is not the same as the sensitivity. In digital lock-in amplifiers the sensitivity is only determined by the output amplifier, which is an entirely digital signal processing unit which performs a numerical multiplication of the demodulator output with the scaling factor. The digital output of this unit is then fed to the output digital-to-analog converter (DAC) with a fixed range of 10 V. It is this scaling factor that can be retrofitted to specify a sensitivity as known from the analog lock-in amplifiers. A large scaling factor, and thus a high sensitivity, comes at a relatively small expense for digital amplification.

One interesting aspect of digital lock-in amplifiers is the connection between input resolution and sensitivity. As the ADC operates with a finite resolution, for instance 14 bits, the minimum signal that can be detected and digitized is for instance 1 mV divided by the resolution of the ADC. With 14 bits the minimum level that can be digitized would be 122 nV. How is it possible to reach 1 nV sensitivity without using a 21 bit analog-to-digital converter? In a world without noise it is not possible. Inversely, thanks to noise and current digital technology it is possible to achieve a sensitivity even below 1 nV.

Most sources of broadband noise, including the input amplifier, can be considered as Gaussian noise sources. Gaussian noise is equally distributed in a signal, and thus generates equally

distributed disturbances. The noise itself can be filtered by the lock-in amplifier down to a level where it does not impact the measurement. Still, in the interplay with the signal, the noise does have an effect on the measurement. The input of the ADC is the sum of the noise and the signal amplitude. Every now and then, the signal amplitude on top of the large noise will be able to toggle the least significant bits even for very small signals, as low as 1 nV and below. The resulting digital signal has a component at the signal frequency and can be detected by the lock-in amplifier.

There is a similar example from biology. Rod cells in the human eye permit humans to see in very low light conditions. The sensitivity of rod cells in the human eye is as low as a single photon. This sensitivity is achieved in low light conditions by a sort of pre-charging of the cell to be sensitive to the single photon that triggers the cell to fire an impulse. In a condition with more surround light, rod cells are less sensitive and need more photons to fire.

To summarize, in digital lock-in amplifiers the full range sensitivity is only determined by the scaling factor capability of the digital output amplifier. As the scaling can be arbitrary big, 1 nV minimum full range sensitivity is achievable without a problem. Further, digital lock-in amplifiers exploit the input noise to heavily increase the sensitivity without impacting the accuracy of the measurement.

9.5. Sinc Filtering

As explained in [Section 9.1](#), the demodulated signal in an ideal lock-in amplifier has a signal component at DC and a spurious component at twice the demodulation frequency. The components at twice the demodulation frequency (called the 2ω component) is effectively removed by regular low-pass filtering. By selecting filters with small bandwidth and faster roll-offs, the 2ω component can easily be attenuated by 100 dB or more. The problem arises at low demodulation frequencies, because this forces the user to select long integration times (e.g. >60 ms for a demodulation frequency of 20 Hz) in order to achieve the same level of 2ω attenuation.

In practice, the lock-in amplifier will modulate DC offsets and non-linearities at the signal input with the demodulation frequency, resulting in a signal at the demodulation frequency (called ω component). This component is also effectively removed by the regular low-pass filters at frequencies higher than 1 kHz.

At low demodulation frequencies, and especially for applications with demodulation frequencies close to the filter bandwidth, the ω and 2ω components can affect the measurement result. Sinc filtering allows for strong attenuation of the ω and 2ω components. Technically the sinc filter is a comb filter with notches at integer multiples of the demodulation frequency (ω , 2ω , 3ω , etc.). It removes the ω component with a suppression factor of around 80 dB. The amount of 2ω component that gets removed depends on the input signal. It can vary from entirely (e.g. 80 dB) to slightly (e.g. 5 dB). This variation is not due to the sinc filter performance but depends on the bandwidth of the input signal.

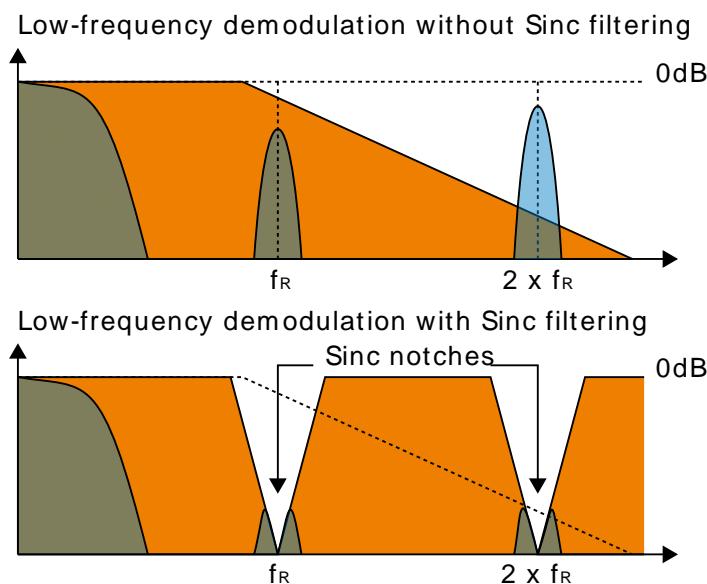


Figure 9.7. Effect of sinc filtering

Table 9.3. Artifacts in the demodulation signal

Input signal	Demodulation result before low-pass filter	Result
Signal at ω	DC component	Amplitude and phase information (wanted signal)
	2ω component	Unwanted component (can additionally be attenuated by sinc filter)

Input signal	Demodulation result before low-pass filter	Result
DC offset	ω component	Unwanted component (can additionally be attenuated by sinc filter)

We can observe the effect of the sinc filter by using the Spectrum Analyzer Tool of the HF2 Lock-in Amplifier. As an example, consider a 30 Hz signal with an amplitude of 0.1 V that demodulated using a filter bandwidth of 100 Hz and a filter order 8. In addition 0.1 V offset is added to the signal so that we get a significant ω component.

Figure 9.8 shows a spectrum with the sinc filter disabled, whereas for Figure 9.9 the sinc filter is enabled. The comparison of the two clearly shows how the sinc options dampens both the ω and 2ω components by about 100 dB.

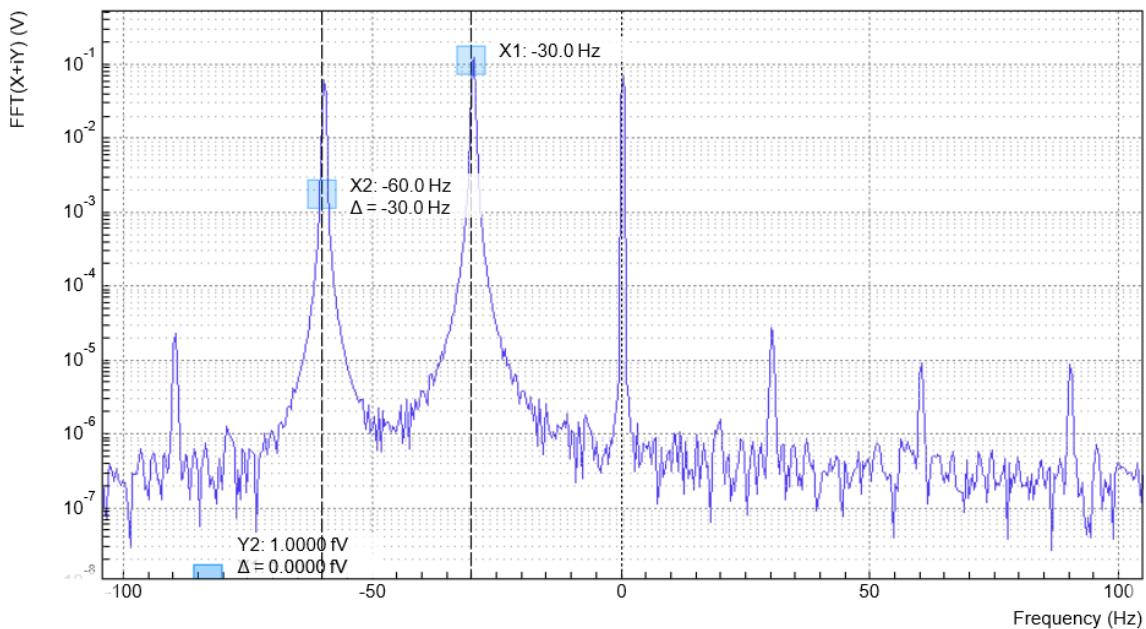


Figure 9.8. Spectrum of a demodulated 30 Hz signal without sinc filter

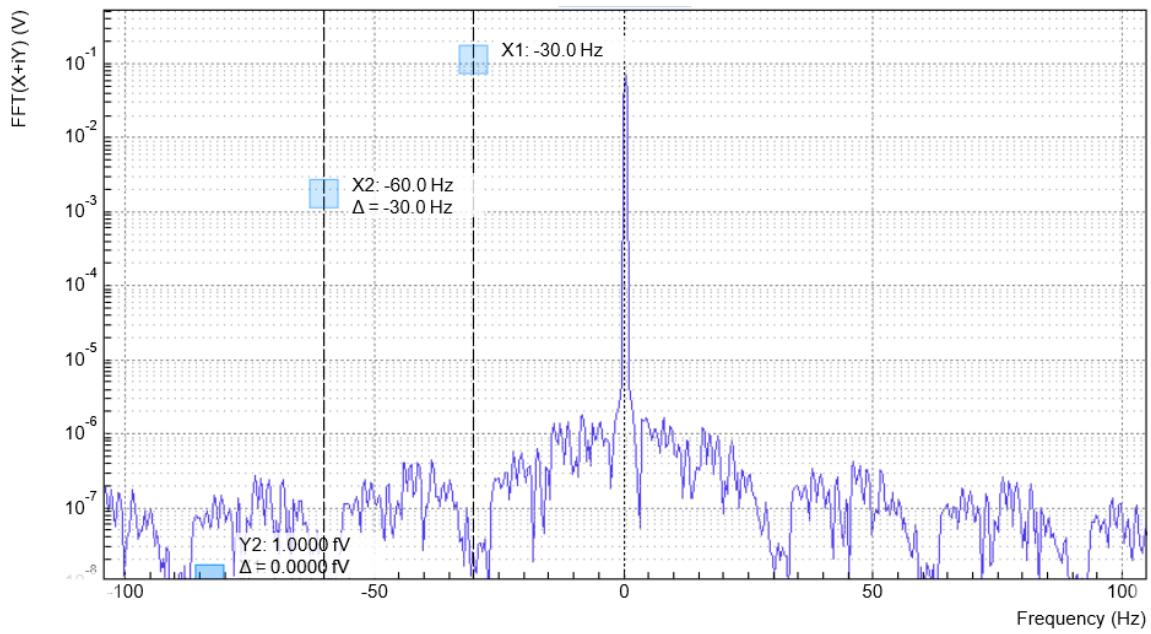


Figure 9.9. Spectrum of a demodulated 30 Hz signal with sinc filter

Note

In order to put the notches of the digital filter to ω and 2ω , the sampling rate of the filter would have to be precisely adjusted to the signal frequency. As this is technically not feasible, the generated signal frequency is adjusted instead by a very small amount.

9.6. Zoom FFT

The concept of zoom FFT allows the user to analyze the spectrum of the input signal around a particular frequency by zooming in on a narrow frequency portion of the spectrum. This is done by performing a Fourier transform of the demodulated in-phase and quadrature (X and Y) components or more precisely, on the complex quantity $X+iY$, where i is the imaginary unit. In the LabOne user interface, this functionality is available in the Spectrum tab.

In normal FFT, the sampling rate determines the frequency span and the total acquisition time determines the frequency resolution. Having a large span and a fine resolution at the same time then requires long acquisition times at high sample rates. This means that a lot of data needs to be acquired, stored, and processed, only to retain a small portion of the spectrum and discard most of it in the end. In zoom FFT, the lock-in demodulation is used to down-shift the signal frequency, thereby allowing one to use both a much lower sampling rate and sample number to achieve the same frequency resolution. Typically, to achieve a 1 Hz frequency resolution at 1 MHz, FFT would require to collect and process approximately 10^6 points, while zoom FFT only processes 10^3 points. (Of course the high rate sampling is done by the lock-in during the demodulation stage, so the zoom FFT still needs to implicitly rely on a fast ADC.)

In order to illustrate why this is so and what benefits this measurement tool brings to the user, it is useful to remind that at the end of the demodulation of the input signal $V_s(t) = A_s \cos(\omega_s t + \Theta)$, the output signal is $X + iY = F(\omega_s - \omega_r)(A_s / \sqrt{2}) e^{i[(\omega_s - \omega_r)t + \Theta]}$ where $F(\omega)$ is the frequency response of the filters.

Since the demodulated signal has only one component at frequency $\omega_s - \omega_r$, its power spectrum (Fourier transform modulus squared) has a peak of height $(|A_s|^2/2) \cdot |F(\omega_r - \omega_s)|^2$ at $\omega_s - \omega_r$: this tells us the spectral power distribution of the input signal at frequencies close to ω_r within the demodulation bandwidth set by the filters $F(\omega)$.

Note that:

- the ability of distinguish between positive and negative frequencies works only if the Fourier transform is done on $X+iY$. Had we taken X for instance, the positive and negative frequencies of its power spectrum would be equal. The symmetry relation $G(-\omega) = G^*(\omega)$ holds for the Fourier transform $G(\omega)$ of a real function $g(t)$ and two identical peaks would appear at $\pm|\omega_s - \omega_r|$.
- one can extract the amplitude of the input signal by diving the power spectrum by $|F(\omega)|^2$, the operation being limited by the numerical precision. This is implemented in LabOne and is activated by the Filter Compensation button: with the Filter Compensation enabled, the background noise appears white; without it, the effect of the filter roll-off becomes apparent.

The case of an input signal containing a single frequency component can be generalized to the case of multiple frequencies. In that case the power spectrum would display all the frequency components weighted by the filter transfer function, or normalized if the Filter Compensation is enabled.

When dealing with discrete-time signal processing, one has to be careful about aliasing which originates when the signal frequencies higher than the sampling rate Ω are not sufficiently suppressed. Remember that Ω is the user settable readout rate, not the 210 MSa/s sampling rate of the HF2 input. Since the discrete-time Fourier transform extends between $-\Omega/2$ and $+\Omega/2$, the user has to make sure that at $\pm\Omega/2$ the filters provide the desired attenuation: this can be done either by increasing the sampling rate or resolving to measure a smaller frequency spectrum (i.e. with a smaller filter bandwidth).

Similarly to the continuous case, in which the acquisition time determines the maximum frequency resolution ($2\pi/T$ if T is the acquisition time), the resolution of the zoom FFT can be

increased by increasing the number of recorded data points. If N data points are collected at a sampling rate Ω , the discrete Fourier transform has a frequency resolution of Ω/N .

Chapter 10. HF2CA Current Amplifier Data Sheet

This chapter contains the data sheet of the HF2CA Current Amplifier which is a preamplifier dedicated to the HF2 Series instruments. This data sheet is distributed only as part of the HF2 User Manual, and therefore not available separately.

The content of the chapter starts with the list of key features of the preamplifier, and continues with sections including the specifications, the detailed functional description, several possible applications, and finally an extended recommendation for 3rd party cables and connectors.

10.1. Key Features

- Current amplifier for high capacitive loads - shunt resistor based
- Voltage amplifier with selectable gain 1 or 10
- Input impedance switchable between 10 V/A and 1 MV/A
- Bandwidth from DC up to 100 MHz
- 2 differential amplification channels with switchable AC/DC coupling
- Adjustable output gain of 1 or 10
- Very low noise and small input leakage
- Single connector for power supply and control

The HF2CA current amplifier converts a differential input current to a differential output voltage in a wide frequency range. This device functions as an active probe and is conveniently placed close to the measurement setup. It supports applications with high capacitive loads such as dielectric impedance spectroscopy. When no shunt resistor is selected, the current amplifier works as a voltage amplifier. The careful design of the HF2CA insures stable operation over the entire frequency range.

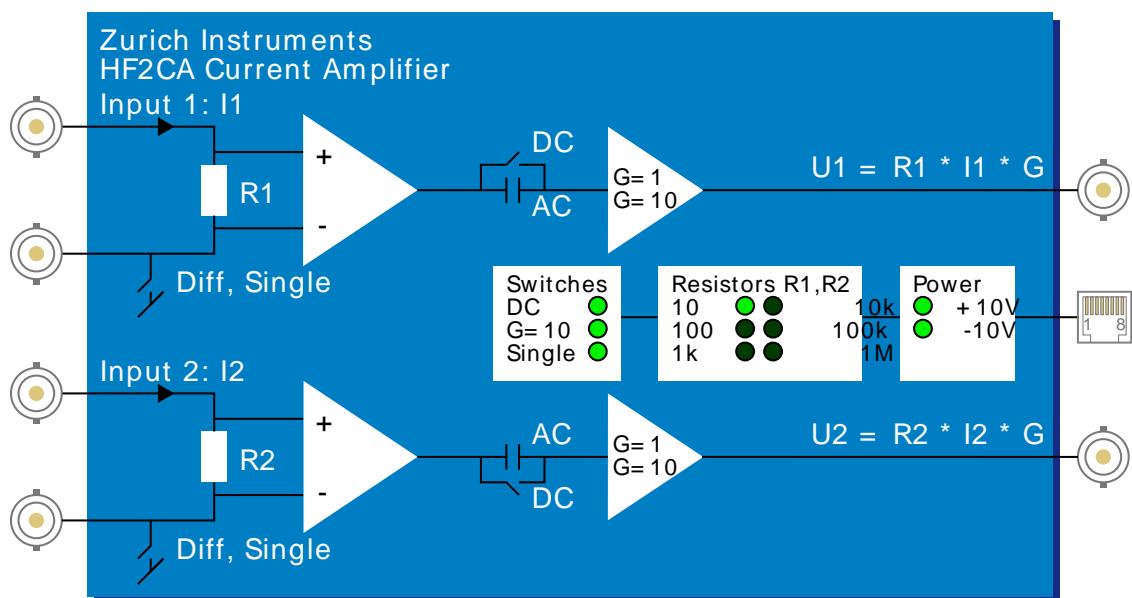


Figure 10.1. HF2CA functional overview

10.2. Specifications

Unless otherwise stated, all specifications apply after 30 minutes of device warming up.

Table 10.1. General

Parameter	Description
dimensions	100 x 60 x 25 mm
weight	0.4 kg
storage temperature	-20 °C to 65 °C
operating temperature	5 °C to 40 °C
specification temperature	25 °C
specification supply voltage	12 V
connectors	4 SMB inputs, 2 SMB outputs, 1 RJ45 (no Ethernet)

Table 10.2. Specifications

Parameter	min	typ	max
positive supply voltage VDD+	12 V	15 V	20 V
negative supply voltage VDD-	-20 V	-15 V	-12 V
supply current	60 mA	80 mA	120 mA
frequency response			
frequency range	DC	-	100 MHz
frequency range (AC coupled)	100 Hz	-	100 MHz
small signal bandwidth / 3dB cut-off (0.1 V _{PP} input, 50 pF output load, gain 1)	100 MHz	-	-
small signal bandwidth / 3dB cut-off (0.1 V _{PP} input, 50 pF output load, gain 10)	25 MHz	-	-
large signal bandwidth / 3dB cut-off (1 V _{PP} , 50 pF)	40 MHz	-	-
input			
input voltage noise (10 kHz)	-	7 nV/√Hz	-
input voltage noise (10 MHz)	-	6 nV/√Hz	-
input bias current	-	2 pA	10 pA
transimpedance gain (equivalent to input impedance)	10 V/A	-	1 MV/A
transimpedance gain accuracy (G=1)	-	±0.1 %	-
transimpedance gain accuracy (G=10)	-	±1 %	-
input offset voltage	-	-	1 mV
common-mode offset range	-10 V	-	7.5 V
output			
output voltage gain	1	-	10
control interface			
input high level	2.0 V	-	5 V
input low level	0 V	-	0.8 V

Parameter	min	typ	max
all transitions on SDI, SDO, SCK, SLC	-	-	1 μ s
SCK clock period	10 μ s	-	-
SDI data to clock setup t_{DS}	2 μ s	-	-
SDI data hold from clock t_{DH}	1 μ s	-	-
SLC clock to latch setup t_{LS}	1 μ s	-	-
SLC latch hold t_{LH}	10 μ s	-	20 μ s
SCK clock free time t_{CF}	20 μ s	-	-

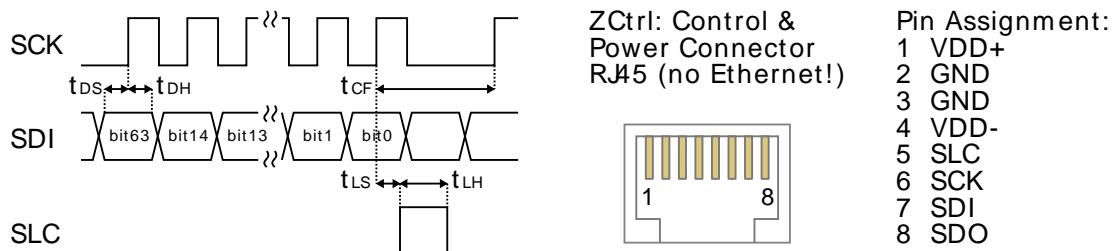


Figure 10.2. Digital control timing

Some parameters depend on the transimpedance gain settings. The following table provides an overview. The values in this table are typical values, they depend on the source capacitance, on the input signal swing, and as well as on the capacitive load on the output of the amplifier.

Table 10.3. Gain dependent parameters

Input impedance setting	Bandwidth / 3dB cut-off frequency	Maximum input current range	Maximum input current noise
10 V/A	100 MHz	± 160 mA	400 pA/ $\sqrt{\text{Hz}}$
100 V/A	50 MHz	± 16 mA	42 pA/ $\sqrt{\text{Hz}}$
1 kV/A	5 MHz	± 1.6 mA	5.6 pA/ $\sqrt{\text{Hz}}$
10 kV/A	500 kHz	± 160 μ A	1.3 pA/ $\sqrt{\text{Hz}}$
100 kV/A	50 kHz	± 16 μ A	400 fA/ $\sqrt{\text{Hz}}$
1 MV/A	5 kHz	± 1.6 μ A	128 fA/ $\sqrt{\text{Hz}}$

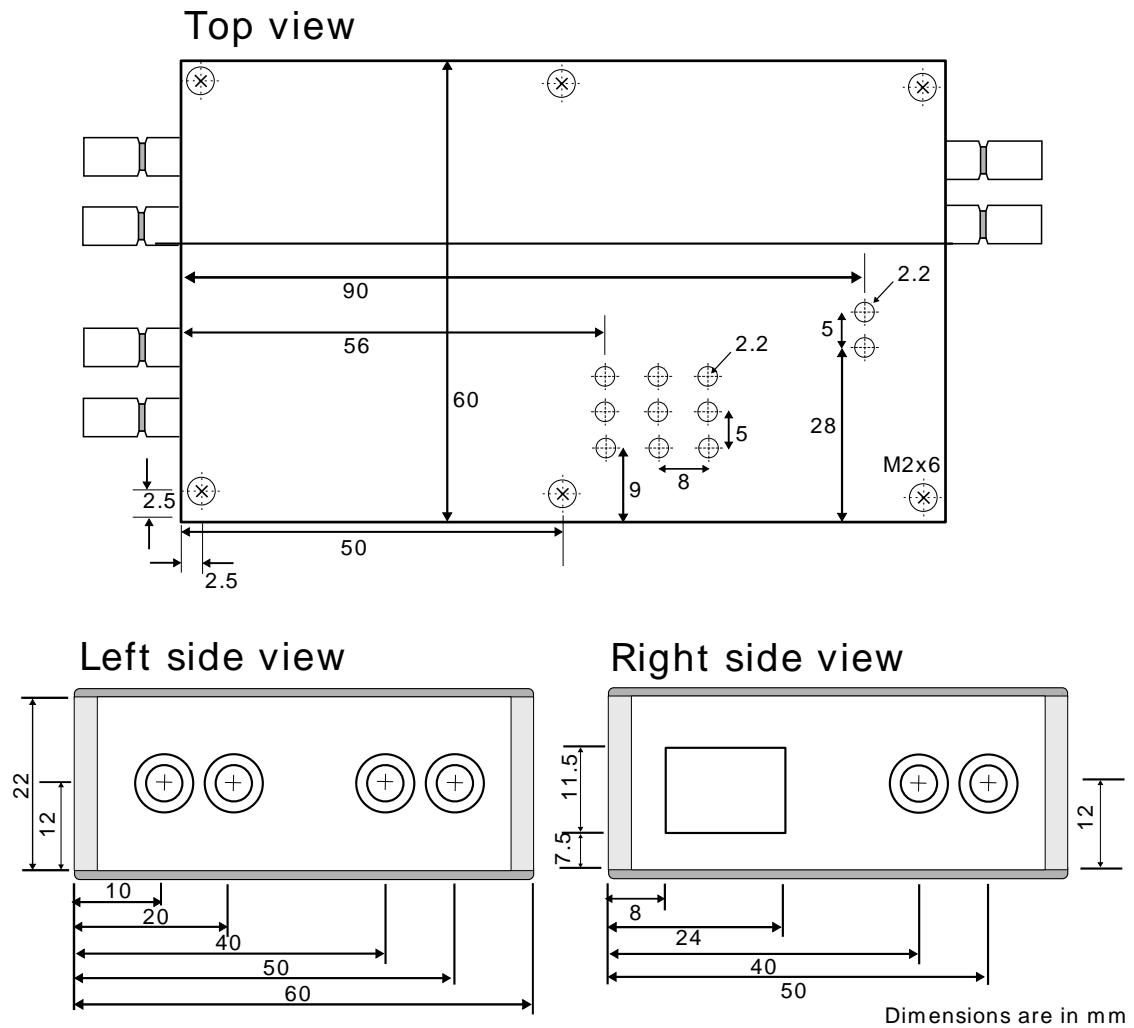


Figure 10.3. Casing dimensions of the HF2CA

10.3. Functional Description

The HF2CA external amplifier can be placed close to the signal source whereas the HF2 Instrument can be several meters away. Such a setup significantly improves the measurement quality due to less parasitics effects and to smaller interferences.

The two signal channels of the HF2CA can be used as separate amplification channels, or alternatively, in differential mode connected to the differential input of the HF2 Instrument.

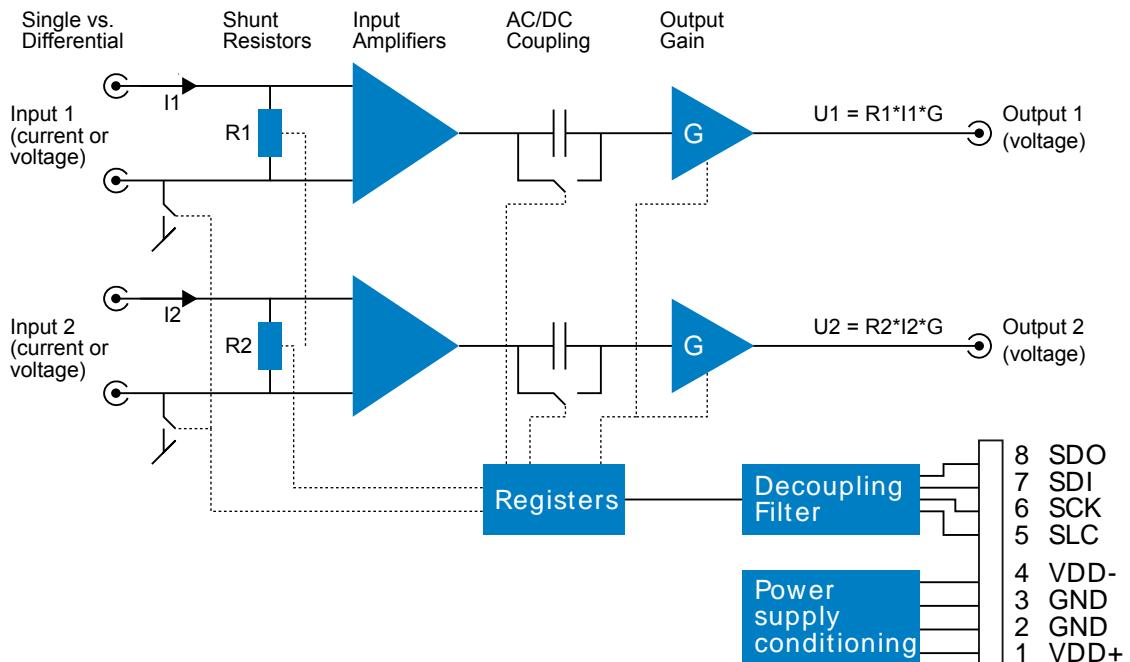


Figure 10.4. Detailed block diagram

10.3.1. Input and Output

Shunt resistors: HF2CA measures the current between the positive and the negative input terminal by measuring the voltage drop across a resistor, that is shunted between the inputs (see Figure 10.4). The supported resistor values are given in Table 10.3. It is also possible to remove all internal resistors and to support any custom resistor that is externally connected (see Section 10.4.4). When all resistors are removed (infinite impedance), then the HF2CA becomes a voltage amplifier with selectable gains 1 and 10.

JFET input amplifiers: the HF2CA is based on JFET input amplifiers that provide very low-noise over a wide frequency range. Additionally, the ultra-low input bias current of typically 2 pA allows for precise current measurements at small signal amplitudes. The input voltage range of the JFET input amplifiers is -10 V to 7.5 V for each input which is also the common mode offset range.

Single vs. differential mode: a selectable switch to amplifier ground allows the user to earth the negative terminal of each input and to operate in single-ended mode without needing external circuits. Alternatively, when leaving the ground switches open, it is possible to use a differential input signal or to connect the negative terminals to local ground externally.

AC vs. DC mode: a selectable switch after the input amplifiers allows the user to measure DC or close to DC signals, or when this is not required, to select AC coupling with a cut-off frequency at 100 Hz and eliminate potential 50/60 Hz noise from the measured signal.

10.3.2. Power Supply and Remote Control

The HF2CA is designed for use with the HF2 Series with its differential signal for improved signal-to-noise, and a single cable that provides power and control signals. A straight-through (as opposed to cross-over) Ethernet cable must be used. The cable carries the following signals:

- **Power:** positive and negative supply, ground
- **Digital control:** SDI digital input signal to control the preamplifier settings, SDO output signal for device detection (details of function not disclosed to users), SCK clock signal, and SLC latch signal. SDI, SCK and SLC are used to program the shift registers on the amplifier and thereby adjust the correct settings. The setting bits are given in [Table 10.4](#). The timing diagram of the digital interface is given in [Figure 10.2](#). The MSB of the register settings is shifted in first.

Table 10.4. HF2CA register settings

Register bit	Name	Description
15 to 10	-	unused
9	gain	0: set output gain to 1
		1: set output gain to 10
8	dcswitch2	0: set AC coupling for input 2
		1: set DC coupling for input 2
7	dcswitch1	0: set AC coupling for input 1
		1: set DC coupling for input 1
6	singleswitch	0: set differential operation
		1: set single-ended operation
5	res1m	1: set resistor 1 MV/A
4	res100k	1: set resistor 100 kV/A
3	res10k	1: set resistor 10 kV/A
2	res1k	1: set resistor 1 kV/A
1	res100	1: set resistor 100 V/A
0	res10	1: set resistor 10 V/A

10.4. Applications

- Impedance spectroscopy
- Large capacitive loads
- Wheatstone-bridge configuration
- Preamplifier for HF2IS impedance spectrometer and HF2LI lock-in amplifier

10.4.1. Differential Current Measurement with Common-mode Offset

The resistors at the input of the amplifier can be inserted in a current path as shown in the figures. With this, fast current transients can be measured at large common-mode voltages, which are in the range from -10 V to 7.5 V. This is used in, e.g., high-energy physics to record the radiation-induced current in a photo diode.

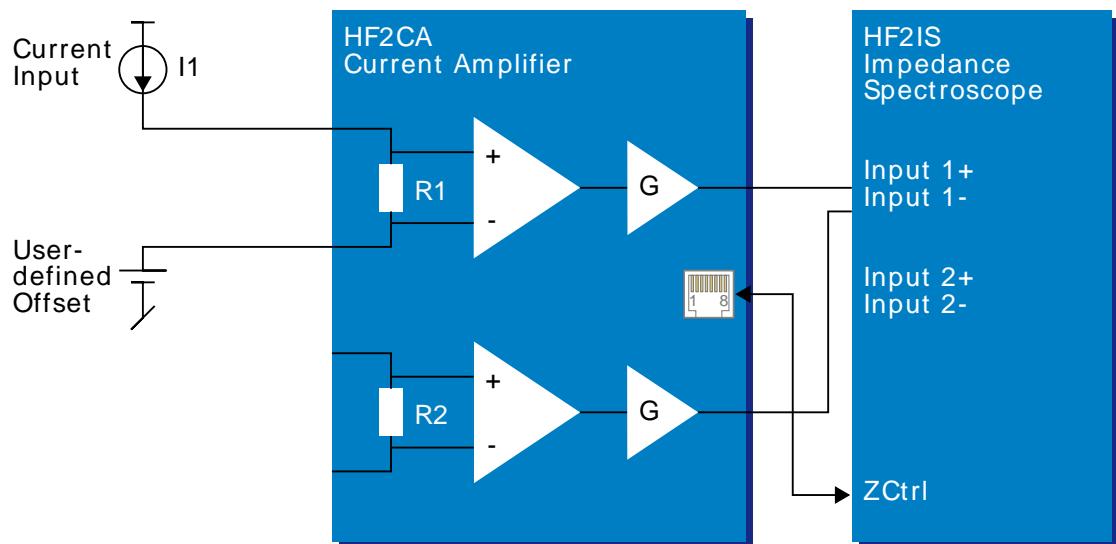


Figure 10.5. HF2CA differential current measurement

10.4.2. Multi-frequency Impedance Spectroscopy

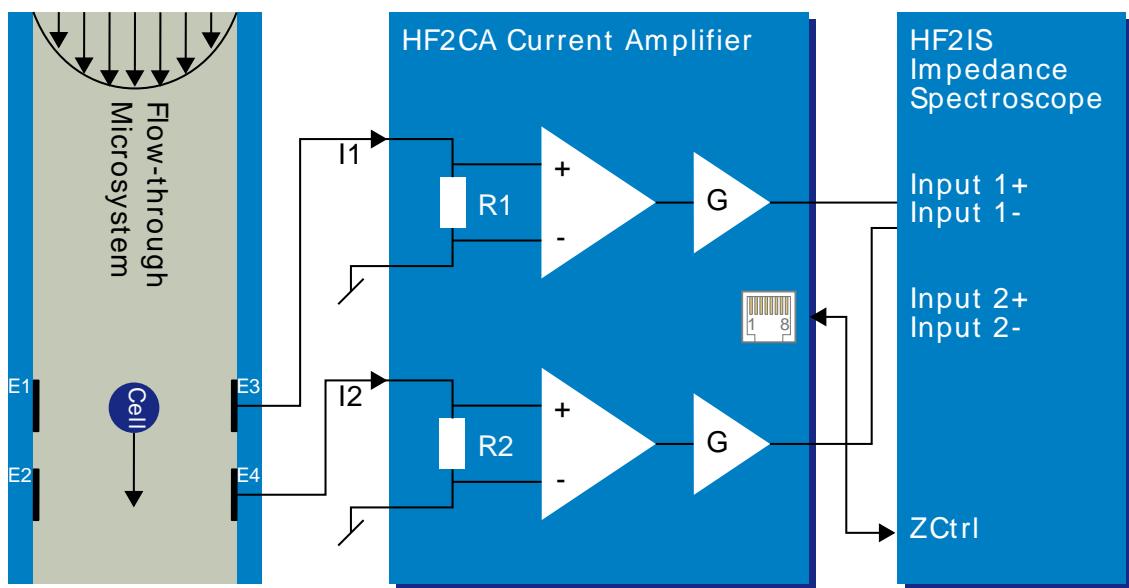


Figure 10.6. HF2CA impedance spectroscopy

The HF2CA in combination with the HF2IS impedance spectrometer is the solution to measure impedance in, for example, flow-through microsystems. The challenge here is to measure the channel impedance at high frequencies (>10 MHz). The large capacitance occurring at electrode electrolyte interfaces can lead to stability issues in a transimpedance amplifier. A solution is to use the electrodes in a Wheatstone bridge configuration with shunt resistors. The HF2CA offers this solution.

As shown in the figure, electrodes are placed on the channel walls of a microfluidic channel (width in the order of 20 to 50 μm). Electrodes E1 and E2 are stimulated with a sinusoidal voltage, the electrodes E3 and E4 are connected to the positive amplifier inputs and thus shunted to GND via resistors R1 and R2. The resulting voltage drops across R1 and R2 are given by the channel impedance. This impedance varies when a particle or a living cell passes the electrode area. An analysis at multiple frequencies at the same time (which is supported by the HF2IS and the HF2CA) allows for concurrently analyzing cell size and dielectric properties. With this information biologists, e.g. sort their cells and detect cell viability or health.

10.4.3. Impedance Measurement

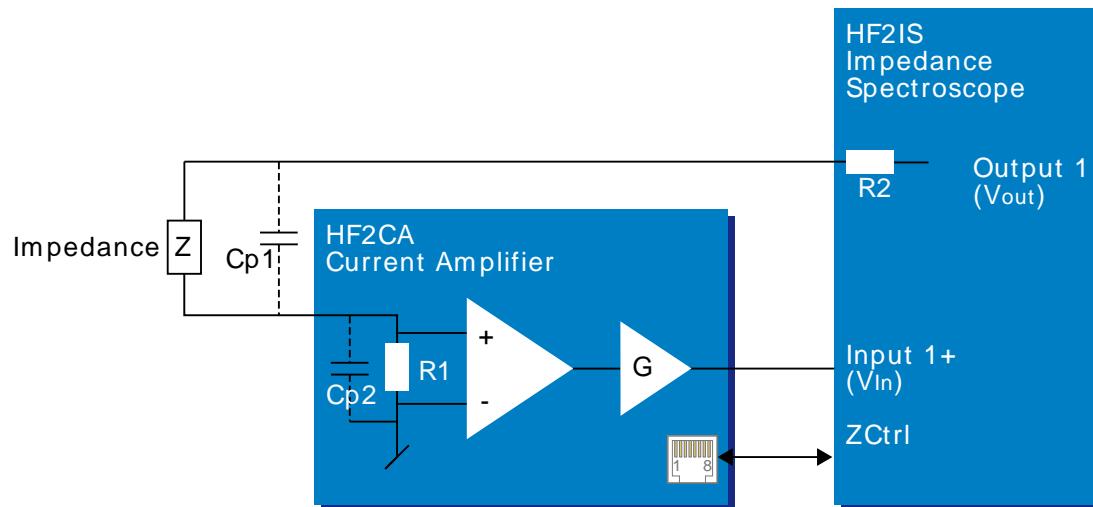


Figure 10.7. Measure an impedance using the HF2CA

The HF2 in conjunction with the HF2CA can be used to measure impedances at various frequencies. The connection diagram is shown in the figure above. The impedance of interest, Z , is connected to the input resistor in the HF2CA preamplifier. For optimal signal-to-noise, the input resistor, $R1$, is set to a value close to the impedance Z . The HF2 generates an output signal of amplitude V_{out} and the output signal from the preamplifier is connected to the positive input (Input +) of the HF2, which is here called V_{in} . With this setup, the impedance Z can be calculated using the following equation:

$$Z = R(V_{out} - V_{in})/V_{in}.$$

Here we neglected the output resistance, $R2$, of the HF2 device. This is valid as long as $Z \gg R1 = 50 \Omega$. Furthermore, at high frequencies the parasitic capacitances C_{p1} and C_{p2} will have to be included in the calculation. At even higher frequencies, C_{p1} and C_{p2} will be dominant.

10.4.4. Custom Input Impedance

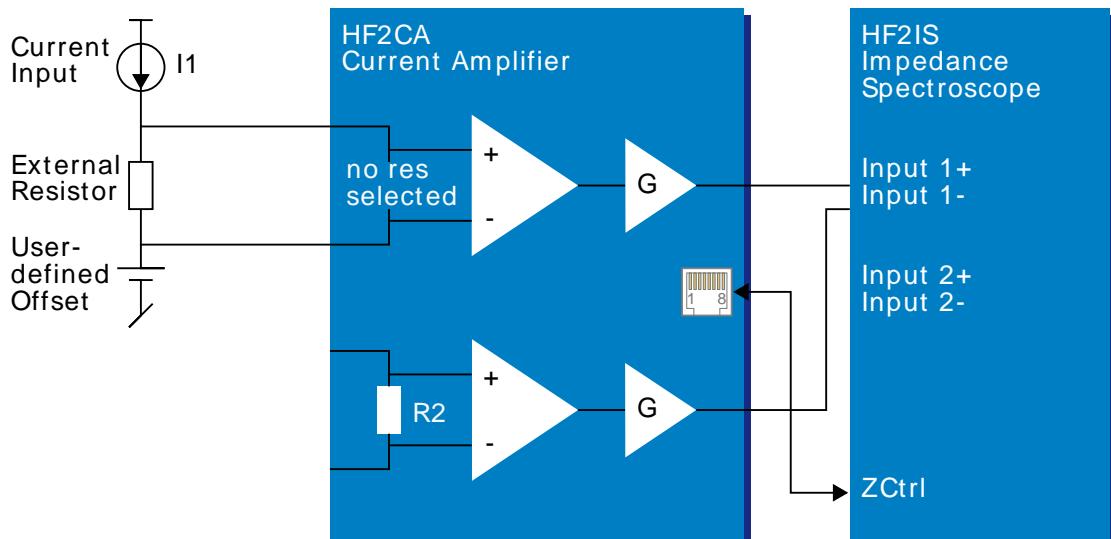


Figure 10.8. HF2CA custom input impedance

Sometimes it is useful to choose a special resistance value in order to optimize the signal to noise by, e.g. impedance matching. In this case, an external resistor can be used instead of using the standard values inside the preamplifier. All internal resistors need to be disconnected in this case, which can be done using the standard preamplifier user interface.

10.5. Cable Recommendation

Table 10.5. HF2CA cable recommendation

Function	Connector / cable type	Vendor / part number
SMB to BNC connection		
SMB to BNC cable	BNC jack to SMB plug	Farnell / Newark 1351896
SMB to BNC adapter	BNC jack to SMB plug	Digikey ACX1386-ND
	BNC jack to SMB jack	Farnell / Newark 4195930
Custom access or cable assembly		
Cable	Cable type RG-174	Digikey A307-100-ND
		Farnell / Newark 1387745
SMB to cable	SMB plug to RG-174 cable	Tyco Electronics 413985-1
		Digikey A4026-ND
		Farnell / Newark 2141206
BNC to cable	BNC plug to RG-174 cable	Tyco Electronics 1-5227079-6
		Digikey A32212-ND
		Farnell / Newark 1831701

Chapter 11. HF2TA Current Amplifier Data Sheet

This chapter contains the data sheet of the HF2TA Current Amplifier which is a preamplifier dedicated to the HF2 Series instruments. This data sheet is distributed only as part of the HF2 User Manual, and therefore not available separately.

The content of the chapter starts with the list of key features of the preamplifier, and continues with sections including the specifications, the detailed functional description, several possible applications, information how to test the specified performance, and finally an extended recommendation for 3rd party cables and connectors.

11.1. Key Features

- 50 MHz operation range
- 2 independent amplification channels with selectable AC/DC coupling
- Wide range of current gain settings (100 V/A to 100 MV/A)
- Impedance measurements from $1 \mu\Omega$ to $100 M\Omega$
- Input offset voltage adjustment
- Voltage output amplifier with selectable gain 1 or 10
- Very low noise and low input leakage
- Single connector for power supply and control

The HF2TA current amplifier converts 2 input currents to output voltages in a frequency range up to 50 MHz. This device is an active probe which can be conveniently placed close to the measurement setup. It supports most applications where a current must be converted to a voltage. The advanced design of the HF2TA ensures stability and a smooth operation over the entire frequency range. The HF2TA transimpedance current amplifier with the HF2 Series signal analyzers allows for very high performance measurements and insensitivity to interferences thanks to reduced parasitics.

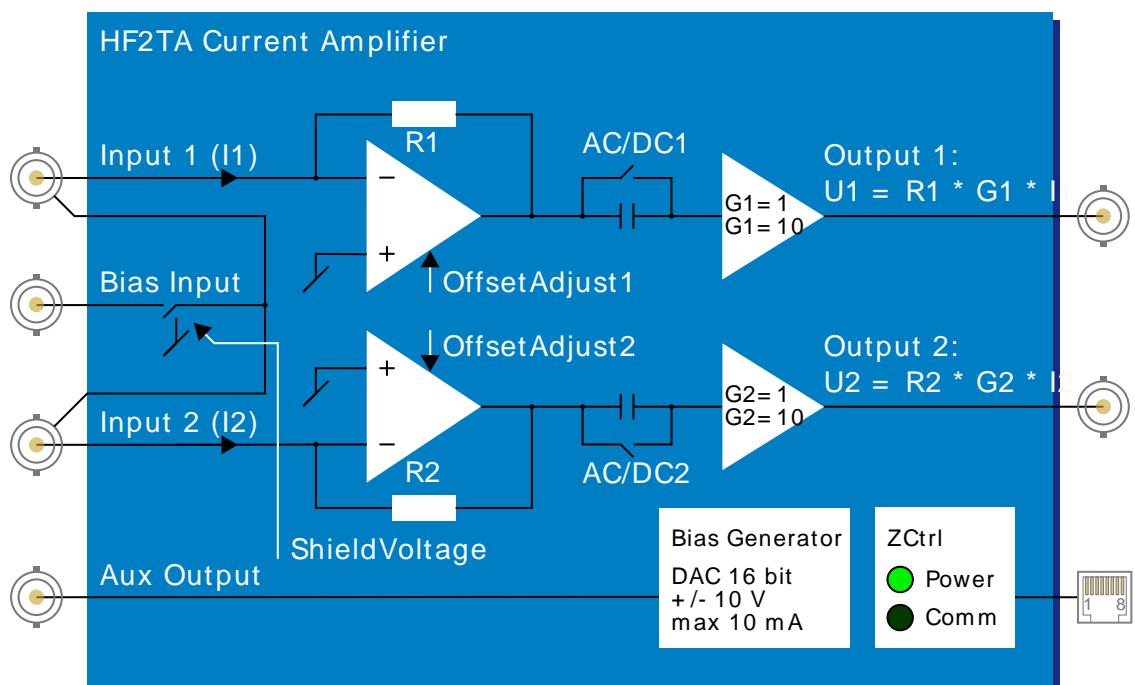


Figure 11.1. HF2TA functional overview

11.2. Specifications

Unless otherwise stated, all specifications apply after 30 minutes of device warming up.

Table 11.1. General

parameter	description
dimensions	101 x 78 x 23 mm
weight	0.4 kg
storage temperature	-20 °C to 65 °C
operating temperature	5 °C to 40 °C
specification temperature	25 °C
connectors	3 SMA inputs female, 3 SMA outputs female, 1 RJ45 (no Ethernet)

Table 11.2. Specifications

parameter	min	typ	max
positive supply voltage VDD+	12 V	13 V	15 V
negative supply voltage VDD-	-15 V	-13 V	-12 V
supply current	50 mA	60 mA	100 mA
frequency response			
frequency range	DC	-	50 MHz
frequency range (AC coupled)	10 Hz	-	50 MHz
small signal bandwidth / 3dB cut-off (0.1 V _{PP} input, 50 pF output load, gain 1)	-	-	50 MHz
small signal bandwidth / 3dB cut-off (0.1 V _{PP} input, 50 pF output load, gain 10)	-	-	50 MHz
large signal bandwidth / 3dB cut-off (1 V _{PP} , 50 pF)	-	-	40 MHz
input			
input current range	depends on R1, R2, G1, G2 settings		
input current noise	depends on R1, R2, G1, G2 settings		
input voltage noise (10 kHz)	-	7 nV/√Hz	-
input voltage noise (10 MHz)	-	5 nV/√Hz	-
input leakage current	-	2 pA	20 pA
input voltage offset compensation range	-10 mV	-	10 mV
input impedance range (Z // 15 pF)	50 Ω	-	70 kΩ
input bias voltage range	-10 V	-	10 V
input signal level (damage threshold)	-5 V	-	5 V
output			
output voltage gain (G1,G2)	1	-	10
transimpedance gain (R1,R2)	100 V/A	-	100 MV/A

parameter	min	typ	max
transimpedance gain accuracy (R1,R2)	-	$\pm 1\%$	-
digital control interface timing			
input high level	2.2 V	-	5 V
input low level	0 V	-	0.8 V
all transitions on SDI, SDO, SCK, SLC	-	-	1 μ s
SCK clock period	10 μ s	-	-
SDI data to clock setup t_{DS}	2 μ s	-	-
SDI data hold from clock t_{DH}	1 μ s	-	-
SLC clock to latch setup t_{LS}	1 μ s	-	-
SLC latch hold t_{LH}	10 μ s	-	20 μ s
SCK clock free time t_{CF}	20 μ s	-	-

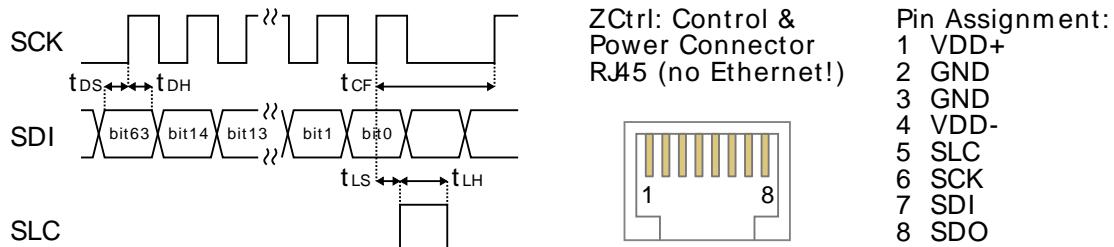


Figure 11.2. Digital control interface timing

Some parameters depend on the transimpedance gain settings. The following table provides an overview. The values in this table are typical values, they depend on the source capacitance, on the input signal swing, and also on the capacitive load on the output of the amplifier.

Table 11.3. Gain dependent parameters 1

input impedance setting	bandwidth / 3dB cut-off	maximum input current range (G=1)	maximum input current range (G=10)
100 V/A	50 MHz	± 10 mA	± 1 mA
1 kV/A	50 MHz	± 1 mA	± 100 μ A
10 kV/A	8 MHz	± 100 μ A	± 10 μ A
100 kV/A	1.5 MHz	± 10 μ A	± 1 μ A
1 MV/A	250 kHz	± 1 μ A	± 100 nA
10 MV/A	25 kHz	± 100 nA	± 10 nA
100 MV/A	12 kHz	± 10 nA	± 1 nA

Table 11.4. Gain dependent parameters 2

input impedance setting	input impedance	maximum input current noise	measured at
100 V/A	50 Ω	150 pA/ $\sqrt{\text{Hz}}$	1 MHz
1 kV/A	50 Ω	15 pA/ $\sqrt{\text{Hz}}$	1 MHz
10 kV/A	50 Ω	2 pA/ $\sqrt{\text{Hz}}$	1 MHz

input impedance setting	input impedance	maximum input current noise	measured at
100 kV/A	100 Ω	500 fA/ $\sqrt{\text{Hz}}$	100 kHz
1 MV/A	300 Ω	250 fA/ $\sqrt{\text{Hz}}$	100 kHz
10 MV/A	1.6 k Ω	100 fA/ $\sqrt{\text{Hz}}$	10 kHz
100 MV/A	70 k Ω	50 fA/ $\sqrt{\text{Hz}}$	10 kHz

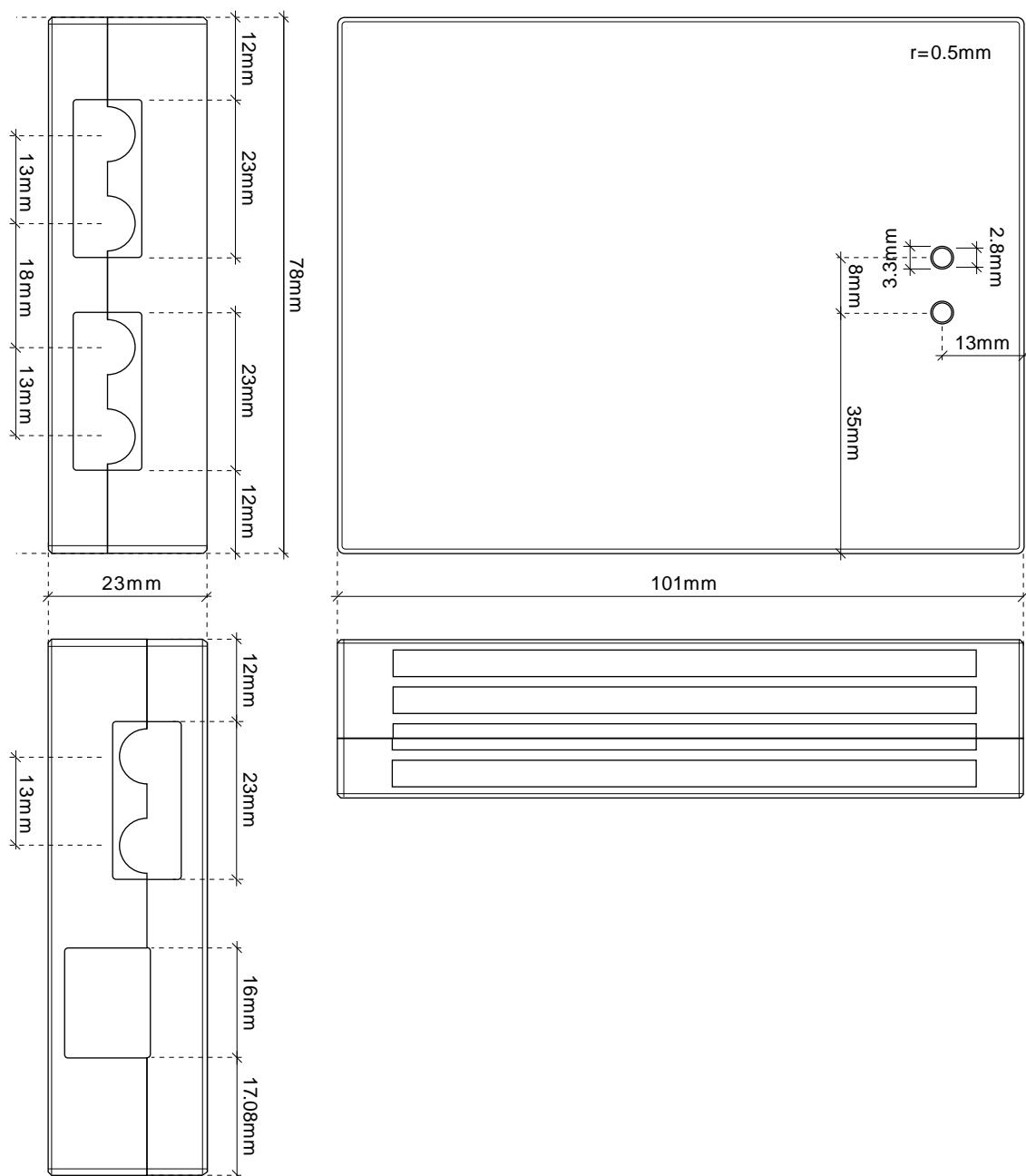


Figure 11.3. Casing dimensions of the HF2TA

11.3. Functional Description

The HF2TA is an external current preamplifier for the HF2 Series instruments from Zurich Instruments. The preamplifier can be placed close to the signal source, which significantly improves the measurement quality due to less parasitics effects and to smaller interferences.

The two signal channels of the HF2TA can be used as separate current amplification channels, or alternatively, in differential mode connected to the differential input of the HF2 Instrument. The channels settings can be set independently.

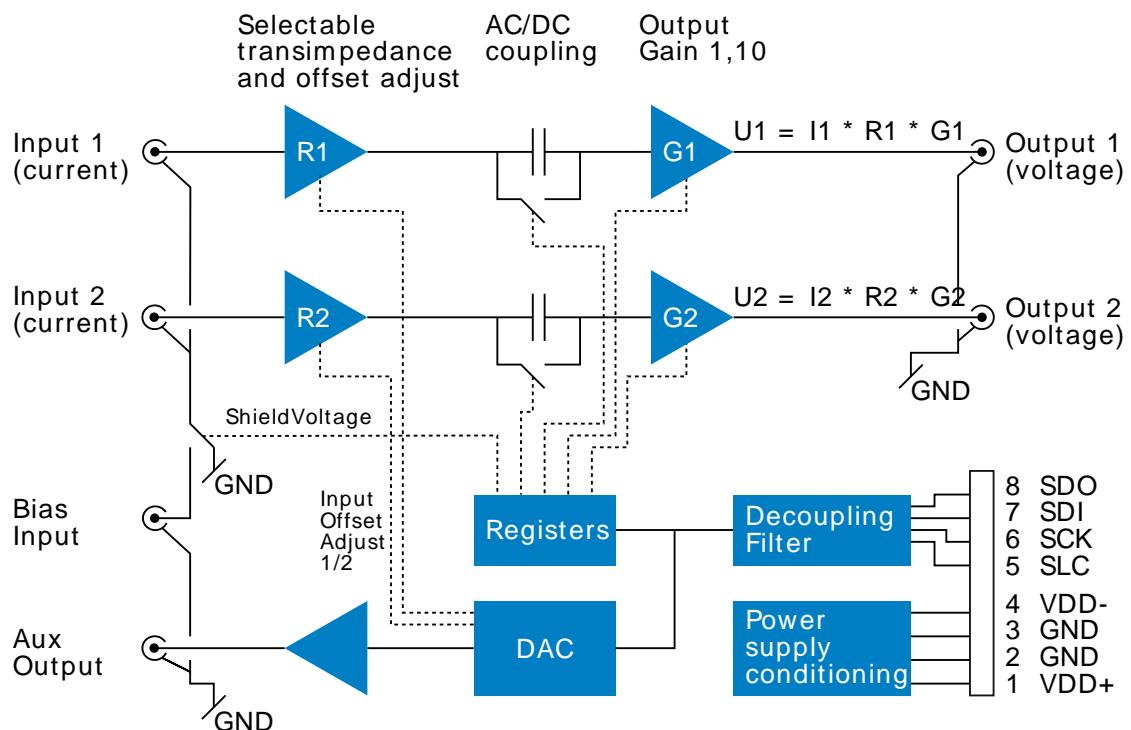


Figure 11.4. Detailed block diagram

11.3.1. Input and Output

Transimpedance stage: the HF2TA measures the current flowing at the two input terminals. The current amplifier uses a standard transimpedance stage to convert the current to a voltage output. The input terminal is matched to 50 Ohms to allow for proper impedance matching at high frequencies. At high current gains, or low input currents, respectively, the input terminal acts like a low-impedance virtual ground. The input impedance depends on the gain settings as described in the table above.

Voltage gain 1 or 10: the HF2TA offers a voltage gain of either 1 or 10 after the transimpedance amplifier. This allows to optimize the signal-to-noise at small amplitudes and high bandwidths. The transimpedance gain often has to be kept small in order to meet the required bandwidth. A voltage amplifier helps in this case to improve the measurement quality.

JFET input amplifiers: the HF2TA is based on JFET input amplifiers that provide very low-noise over a wide frequency range. Additionally, the ultra-low input bias current of typically 2 pA allows for precise current measurements at small signal amplitudes. The input voltage range of the JFET input amplifiers is -5 V to 2 V for each input which is also the common mode offset range.

Offset adjustment: the offset of the input amplifier can be manually compensated. For this purpose, disconnect any signal from the input of the current amplifier and measure the output

voltage. Change the offset voltage until the output is close to zero. All remaining offset should now come from other sources (like offset current or leakage from the device under test).

AC vs. DC mode: a selectable switch after the input amplifiers allows the user to measure DC signals, or when this is not required, to select AC coupling with a cut-off frequency at around 10 Hz to remove the DC offset. When working in AC, make sure that the first amplifier is not saturating. This can be checked by switching to DC and gain 1.

Bias output: the HF2TA comprises a general purpose low-noise analog output. This output can be used as a power supply for, e.g., photo diodes. The photo diode is connected to the auxiliary bias output and the virtual ground of the input, no additional power supply is needed.

Signal shield voltage: the bias input connector can be used to apply a bias voltage to the signal shield. This can be used, for instance, to power a remote sensor over the signal shield without introducing an additional ground loop. If this option is not used, the signal shield should be conveniently grounded with the control setting "Shield Voltage".

11.3.2. Power Supply and Remote Control

The HF2TA is designed for use with the HF2 Series devices. It has to be connected to the ZCtrl 1/2 connectors of the host device using a single Ethernet cable which provides both power and control signals. A standard straight-through (as opposed to cross-over) cable must be used. The cable carries the following signals:

- **Power:** positive and negative supply, ground
- **Digital control:** SDI digital input signal to control the preamplifier settings, SDO output signal for device detection (details of function not disclosed to users), SCK clock signal, and SLC latch signal. SDI, SCK and SLC are used to program the shift registers on the amplifier and the DAC and thereby adjust the correct settings.

11.4. Applications

- Low-noise and high-speed current amplification
- Photo diode preamplifier
- Impedance measurement
- Semiconductor testing
- Impedance spectroscopy

11.4.1. Recommended Settings

In order to get the maximum performance out of your HF2TA, the following guidelines should be followed.

- low and high input current measurement

The HF2TA gain setting should be selected properly in the measurement path. The gain setting can be set according to [Table 15.3. Gain dependent parameter 1](#). As one can see, each input impedance and G setting has a maximum input current range specified. With each recommended input impedance and G setting, the maximum current will produce the maximum voltage swing of ± 1 V at the output of the HF2TA. At this level the input digitizer of the HF2 input channel will run close to its full dynamic range which results in the optimal SNR.

- low and high bandwidth measurement

HF2TA is specified to work up to the 3dB bandwidth of 50 MHz. Nevertheless, care must be taken when selecting input impedance gain settings. [Table 15.3. Gain dependent parameter 1](#) details as well the maximum 3dB signal bandwidth for each gain setting. For example, with an input current containing frequency components of less than 12 kHz in frequency, the maximum transimpedance gain of 100 MV/A can be selected. At 50 MHz, only 100 V/A of transimpedance gain is available. G=10 can also be selected as well if more gain is required at high input signal frequencies.

- minimize cross-talk and parasitics effects

With the measured impedance placed closely to the input of the HF2TA and the HF2 device, four point measurement setup can help to minimize parasitic effect as well as the noise pickups from the cable. Furthermore, using shielded cable can greatly reduce the high frequency noise pickups from the surrounding environment.

- avoid HF2TA instability

Since HF2TA is a negative feedback amplifier, its feedback loop stability can be sensitive to input capacitance, especially at low R settings. In order to avoid possible under-damped behavior (i.e. oscillation) in the measurement, it is recommended to use as high as possible the selected transimpedance gain R when measuring a capacitive circuit. A short cable to the HF2TA input can also help to reduce the parasitic capacitance seen at the HF2TA input.

11.4.2. Photo Diode Amplifier with HF2LI

The HF2TA current amplifier is suited to read out the current from a photo diode. The following figures shows three possible ways to use the device. In the first option, the photo diode is grounded on one side and connected to the current amplifier on the other side. The recorded signal is amplified and sent to the HF2 Instrument.

The second option provides a solution when it is necessary to apply a bias voltage across the photo diode. For this purpose the bias output of the HF2TA can be used. Voltages in the range of +/-10V

and currents up to 10 mA can be delivered by this connector. Alternatively the bias can be provided by another voltage source.

The third option supports the drive of the photo diode by means of the shield of the signal cable. This shield can be conveniently driven by the HF2TA by shorting the bias output to the bias input. This option permits the user to connect the remote sensor with one single coaxial cable and while avoiding to introduce a ground loop in the system.

All HF2TA settings can be conveniently programmed inside the graphical user interface of the HF2 Instrument.

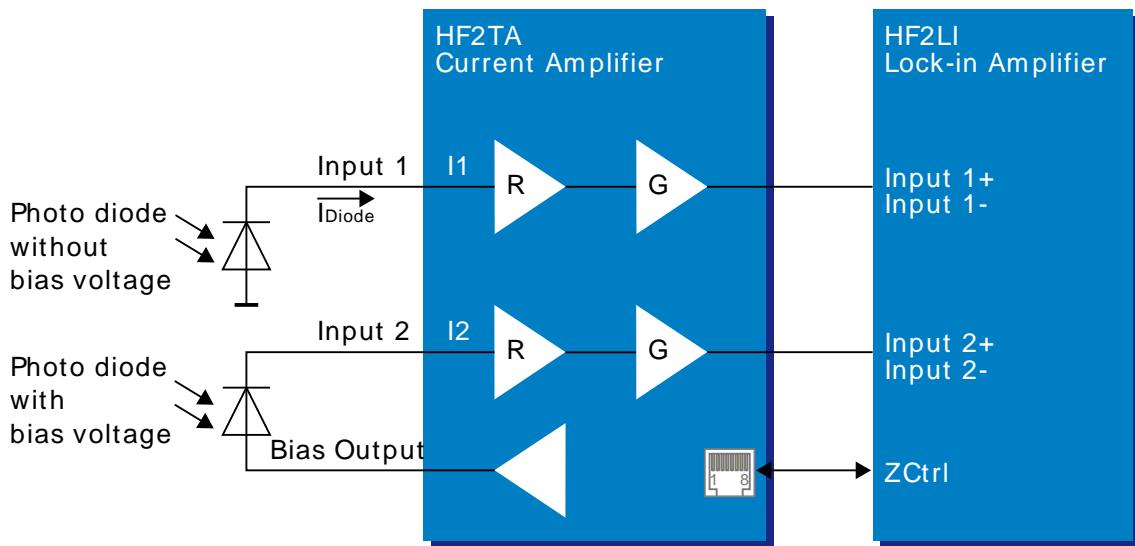


Figure 11.5. HF2TA photo diode amplifier

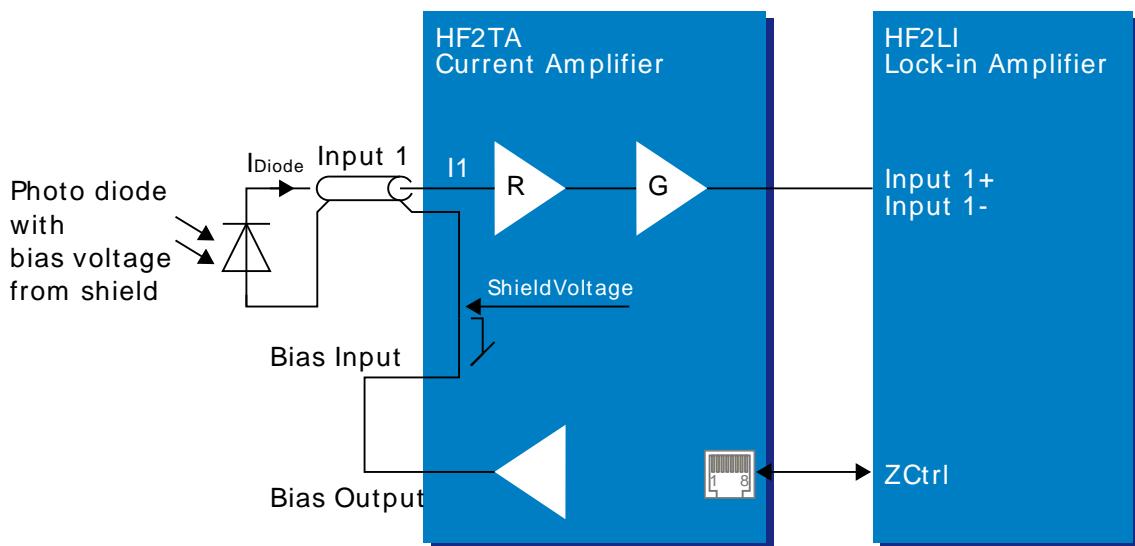


Figure 11.6. HF2TA photo diode amplifier with single coaxial cable

11.4.3. Impedance Measurement with HF2IS

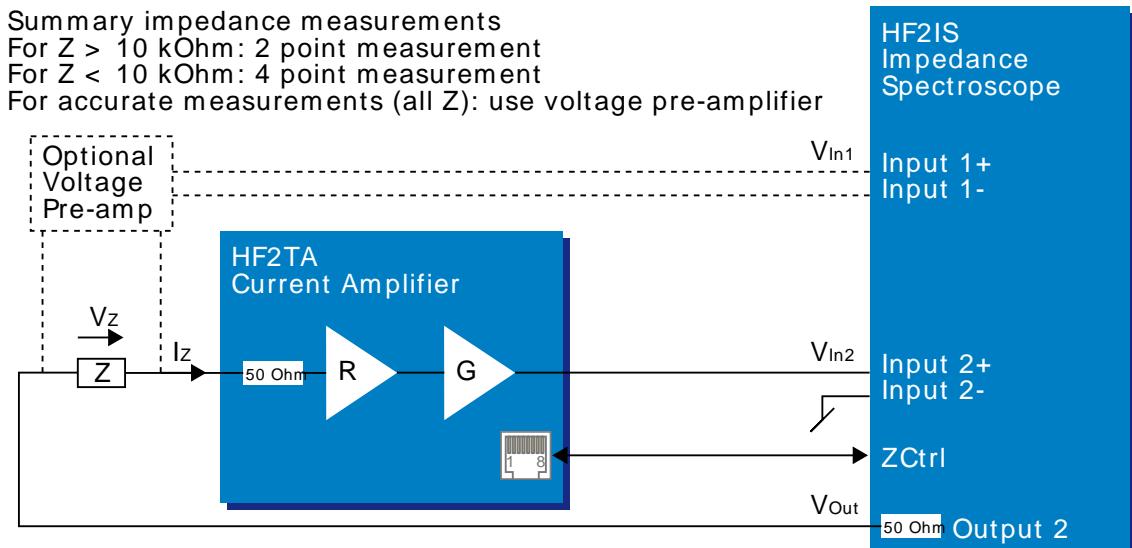


Figure 11.7. Measure an impedance using the HF2TA

The HF2TA current amplifier can be used in conjunction with the HF2IS instrument to measure impedances in a very wide range at frequencies up to 50 MHz. The connection diagram in the figure above shows how the impedance of interest Z is connected to the input of the HF2TA. For optimal amplification versus bandwidth setting, the table in the specification section may be consulted. Three cases and applications need to be distinguished.

- Measuring an impedance $Z > 10 \text{ k}\Omega$

For large impedances it is possible to neglect the output resistance of the HF2IS Instrument and the input resistance of the preamplifier, thus the simple setup provides good accuracy. The HF2IS generates an output signal of amplitude V_{out} and the output signal from the preamplifier is connected to the positive Input 2+ of the HF2, called V_{in} . With this setup, the impedance Z can be calculated using the following equation:

$$Z = R * G * V_{out} / V_{in}$$

- Measuring an impedance $Z < 10 \text{ k}\Omega$

For small impedances and higher precision a four point measurement setup is required. For accuracy in the range of 1%, the voltage V_z can be measured directly by the second differential Input 1+ and Input 1- of the HF2. In this case it is important to select the high ohmic input impedance option ($1 \text{ M}\Omega$) as otherwise too much current is dissipated in the measurement instrument. Also the HF2 should be configured for differential measurement. The resulting impedance Z is calculated using the following equation:

$$Z = R * G * V_z / V_{in} = R * G * V_{in1} / V_{in2}$$

- Measuring impedances with high accuracy (all values of Z)

Four point measurement setup allows the most accurate measurement by taking into account simultaneously the current flowing through the measured impedance and the voltage drop caused by the current flow. For an accuracy better than 1%, it is recommended to use a voltage preamplifier with high-ohmic input stage to measure the voltage across the impedance $V_z = V_{in1}$. Assuming V_{in2} , R and G are the output, the resistor setting and the gain

of the HF2TA, respectively, the resulting equation to calculate the impedance will be similar to the previous case:

$$Z = R * G * V_Z / V_{In2} = R * G * V_{In1} / V_{In2} \text{ (assuming voltage pre-amp gain = 1)}$$

A pictorial representation of how to set up the four-point measurement is shown below.

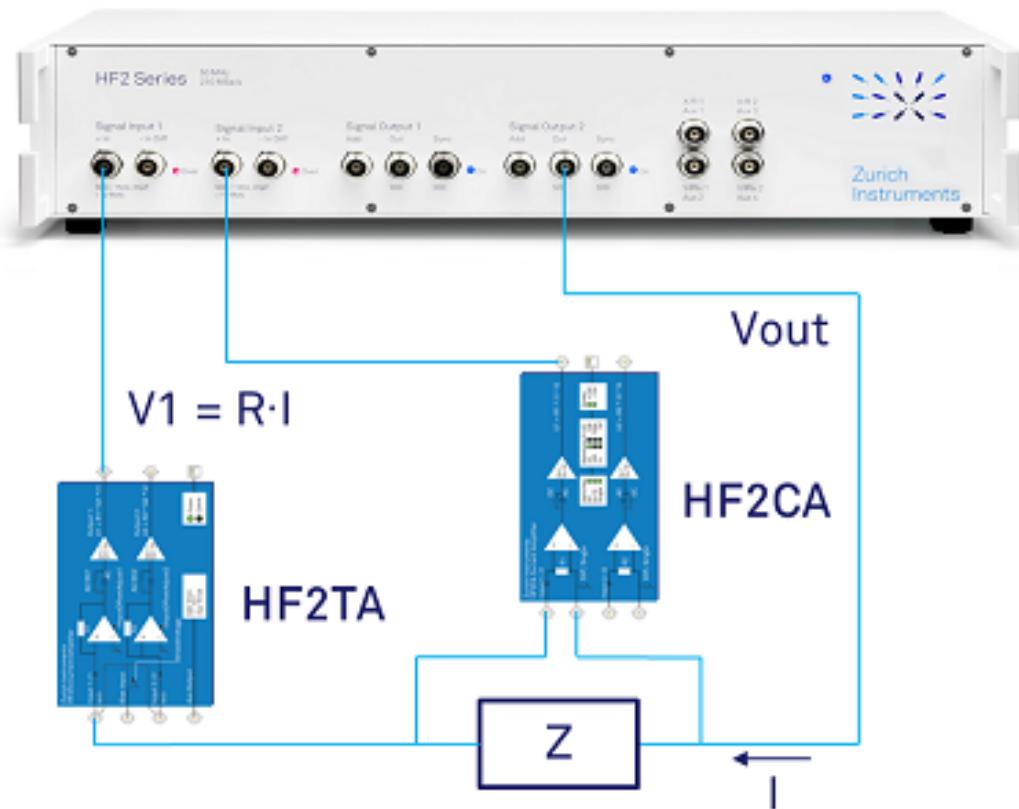


Figure 11.8. HF2IS four-point measurement setup

Note that the voltage measurement is made differentially through HF2CA then converted to single-ended input to the HF2IS while the current measurement remains single-ended throughout the current measurement path. Both HF2TA and HF2CA can be controlled using the LabOne UI. When they are connected through Ethernet cables to the back of the HF2IS instrument, LabOne will automatically add an HF2TA or HF2CA tab as shown in the screen shots below.

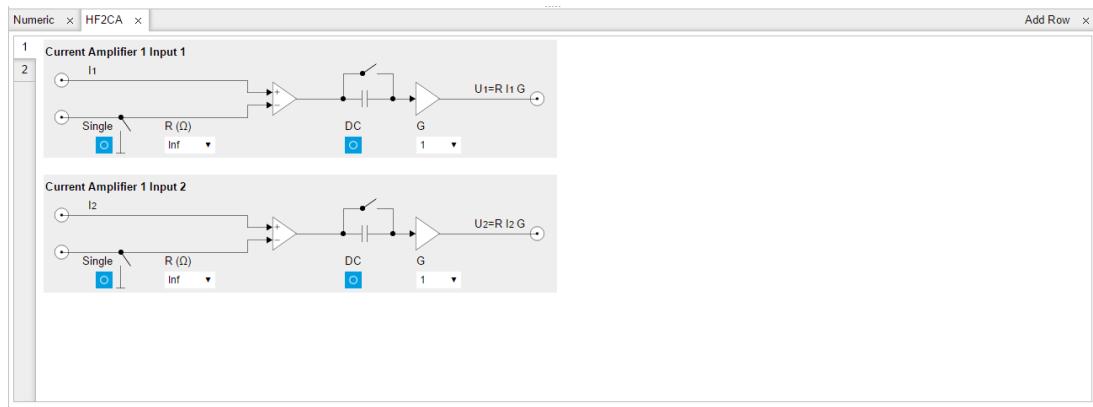


Figure 11.9. HF2CA tab



Figure 11.10. HF2TA tab

Both HF2TA and HF2CA can operate in AC-coupled or DC input. The HF2TA has a high-pass cutoff of 10 Hz while the HF2CA has a high-pass cutoff of 100 Hz. It is recommended that no R_1 and R_2 values are selected for the HF2CA to obtain maximum input impedance (i.e. no signal current loss through the HF2CA input ports) and therefore the most accurate current measurement.

11.5. Performance Tests

In this section two tests are described that can be used to measure the DC leakage and the AC noise of the HF2TA. They can be performed by the user to do a sanity check on the validity of the measurement with HF2TA.

Table 11.5. Necessary equipment

Required equipment	Specifications	Recommended equipment
HF2 Instrument	No additional installation options required	HF2LI or HF2IS
HF2TA Current Amplifier	HF2TA Specifications	HF2TA
Digital multimeter	0.1 mV Resolution, 20 V range	Agilent 34410A
SMA to BNC cables	2 x 50 Ω, male-to-male connectors	supplied by Zurich Instruments
Ethernet cable	Category 5 or 6	supplied by Zurich Instruments

The following conditions have to be fulfilled:

1. The test equipment must be connect to the same AC power circuit. If you are unsure of the AC power circuit distribution, use a common power strip and connect all test equipment into it. Connecting the test equipment into separate AC power circuits can result in offset voltages between equipment, which can invalidate the verification test.
2. For accurate results, allow the test equipment to warm up for at least 30 minutes.
3. The HF2 Instrument as well as the HF2TA transimpedance amplifier are controlled by the LabOne software. Please make sure that the latest version of the LabOne software package is installed on the host computer. Please refer to the [Chapter 1](#) for software installation instructions.

The HF2TA transimpedance amplifier has 2 analog input channels, 2 analog output channels and 1 external bias input and 1 auxiliary output. For the purpose of the following tests, the external bias input will not be used. The test setup for one channel is equally valid for the other channel.

11.5.1. Input Leakage Test

Definition

This test measures the DC input leakage current of the HF2TA.

Setup

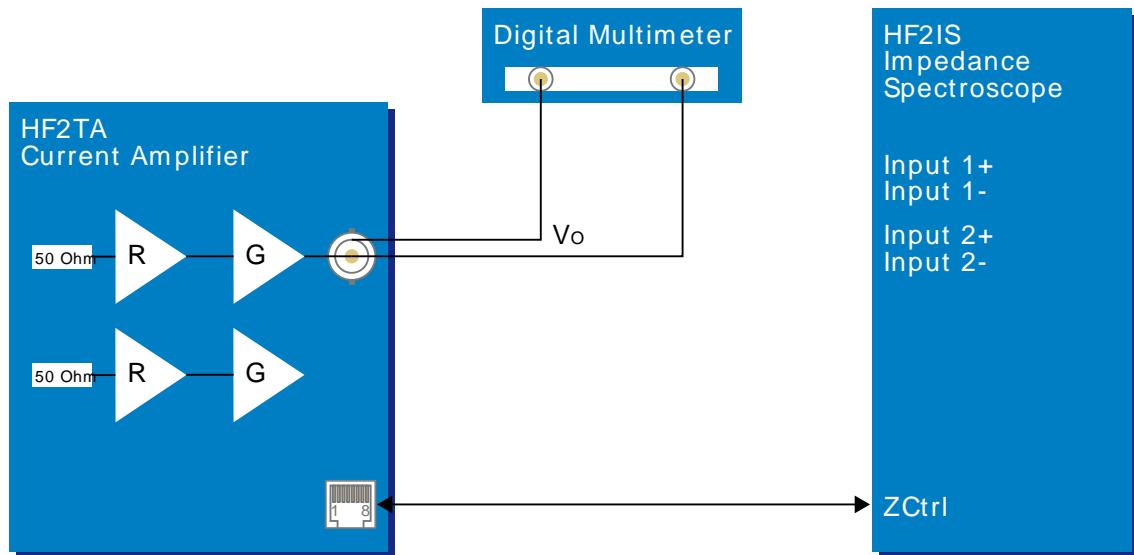


Figure 11.11. HF2TA DC input leakage measurement setup

The HF2TA is connected to the HF2 Instrument via the Ethernet cable for the purpose of configuring the HF2TA. The HF2 Instrument is not used for the measurement.

Table 11.6. HF2TA settings

Ch1 Offset (V)	0.0 V
Ch1 R (V/A)	1 k and 100 M
Ch1 AC	OFF
G	1
Input Shield	GND
Aux Output (V)	0.0 V

Measurement

The DC leakage current can be estimated by subtracting the inherent DC offset V_{O1} of the amplifier from the total offset V_{O2} due to both the internal offset and the input leakage. For this test the input of the HF2TA is left open. Then the output is measured with a digital multimeter as shown. The input offset V_{O1} can be estimated by setting the transimpedance resistor R to 1 k. The sum of the input offset plus leakage V_{O2} can be estimated by setting the transimpedance resistor R to 100 M. Then, the approximate leakage can be found by:

$$I_{\text{leakage}} = \frac{|V_{O2} - V_{O1}|}{100 \text{ M}\Omega} \quad (11.1)$$

11.5.2. Input Noise Test

Definition

The noise generated by the HF2TA transimpedance amplifier itself can be expressed as input referred current noise. The following setup description enables users to verify through measurement if their HF2TA units have indeed the same noise level as specified in [Table 11.4](#).

Setup

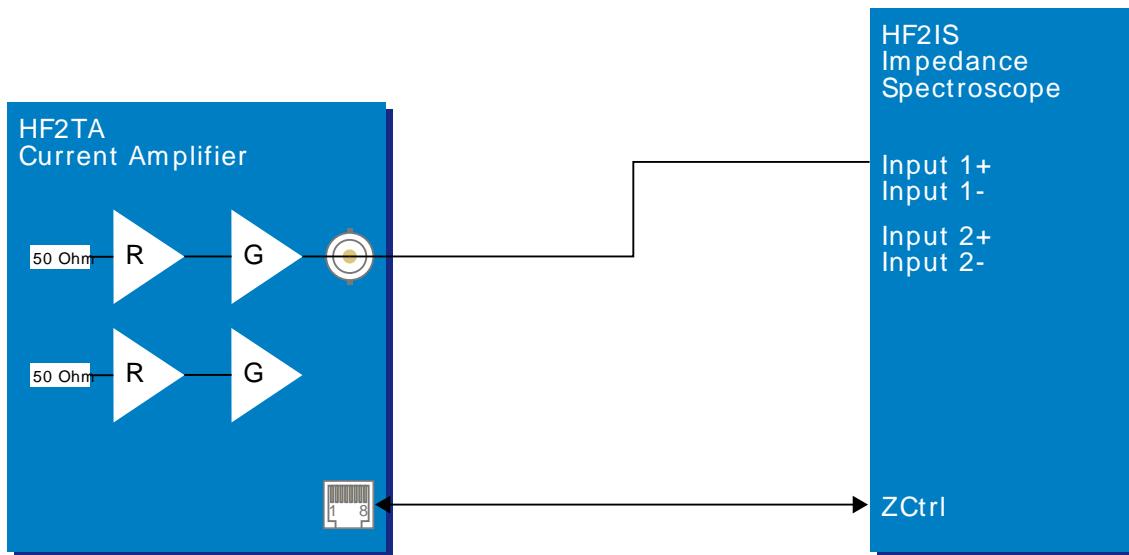


Figure 11.12. HF2TA equivalent input current noise

For this test the HF2TA transimpedance amplifier input is left open. The goal is to refer the total contribution of out noise from the amplifier itself to the input and not from any other external circuits. Since the input of the HF2TA is left open, it is only necessary to define the sweep range in the HF2 instrument since no drive voltage is required.

The HF2 instrument settings for the test are given in the table below.

Table 11.7. HF2 instrument settings

Ch1 Signal Inputs Range	auto range
Ch1 Signal Inputs AC/ Diff/ 50	ON
Ch1 Scale	1/R
Filter BW setting type	BW NEP (noise equivalent power BW)
Filter BW	1 Hz

Notice that Scale has been set to 1/R where R is the HF2TA transimpedance value. This is to obtain the noise current referred back to input.

Measurement

The Sweeper will be used for the measurement with the following settings:

Table 11.8. Frequency sweeper settings

Sweep Range Start	1 kHz
Sweep Range Stop	50 MHz
Sweep Range Points	50
Sweep Range Log Sweep	ON

To have LabOne choose suitable filtering, averaging, and display settings, simply choose Noise Amplitude Sweep as the Application in the Settings sub-tab. Set Precision to High and start

the sweep to measure the noise over the specified frequency. After division by the HF2TA transimpedance gain, the result can then be compared to the values in the input referred noise table.

11.6. Cable Recommendation

Table 11.9. HF2TA cable recommendation

Function	Connector / cable type	Vendor / part number
SMA to BNC connection		
SMA to BNC cable	BNC jack to SMA plug	Digikey J3606-ND
SMA to BNC adapter	BNC jack to SMA plug	Digikey J10098-ND
		Farnell / Newark 4195930
	BNC plug to SMB plug	Farnell / Newark 1654647
		Digikey ACX1324-ND
Custom access or cable assembly		
Cable	Cable type RG-174	Digikey A307-100-ND
		Farnell / Newark 1387745
SMA to cable	SMA plug to RG-174 cable	Digikey A32326-ND
		Farnell / Newark 2112459
BNC to cable	BNC plug to RG-174 cable	Tyco Electronics 1-5227079-6
		Digikey A32212-ND
		Farnell / Newark 1831701

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 $f_{\text{cut-off}}$ or 3dB frequency $f_{-3\text{dB}}$. 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: $U_{\text{RMS}} = U_{\text{PK}} / \sqrt{2} = U_{\text{PK}} / 1.41$. 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

A

Attenuation, 52
Auxiliary, 630
 Tab
 Tutorial, 53
Auxiliary Tab, 151

B

Bandwidth, 639, 655
 Demodulator, 629
 Limit, 51

C

CA (see Preamplifier, CA)
Calibration, 8
Calibration, factory, 30
CMRR, 627, 638
Computer requirements, 625
Config Tab, 158
Coupling
 AC, 49, 96, 107
Cursors
 Description, 89
Cut-off frequency, 655

D

Damage threshold, 625
Demodulator
 Block diagram, 96, 108
Demodulator settling time
 Measurement carried out with the SW Trigger to illustrate the settling time for a 4th order filter with a 3 dB bandwidth set to 100 Hz., 143
Device Tab, 163
Diff (see Differential mode)
Differential mode, 49
DIO, 628
DIO Tab, 155
Dynamic reserve, 627

F

File Tab, 165
Filter, 657
 Compensation, 664
 Settings, 61
 Settling time, 657, 657-658
 Sinc, 629, 661
Filter slope, 59
Frequency
 Range, 627
 External reference, 627
 Internal reference, 627

 Resolution, 627
 Full range sensitivity, 627, 659-660

H

Harmonic
 Distortion, 636
 Rejection, 629
HF input noise (see Noise, HF Input)
HF signal inputs (see Input, HF)
HF signal outputs (see Output, HF)
HF2 back panel, 39
HF2 front panel, 37
HF2 functional diagram, 34

I

Impedance
 Input, 49, 96, 108
Input
 HF, 627
Input range
 AC coupling, 627
 DC coupling, 627
 settings, 627
Inputs/Outputs Tab, 153
Installation
 Linux, 16
 Microsoft .NET Framework, 32
 Windows, 13

L

LabOne UI
 Tutorial, 48
LabVIEW
 API, 200
Linux
 Software installation, 16
Lock-in
 Principal, 652
 Tab
 Tutorial, 48
Lock-in Tab, 95
 With Multi-frequency (MF) Option, 107
Log files, 30, 30

M

Math sub-tab
 Description, 89
Microsoft .NET Framework, 32
MOD Tab, 186
Modulation
 Amplitude, 68-69
 Tutorial, 68
 Frequency, 72-74
 Tutorial, 72
Modulation option
 Block diagram, 187

Mouse functionality

Description, 87

Multi-frequency, 45

Tab, 48

N

NEPBW, 52, 655

Node

Concept, 202

Detailed node description, 228

Hierarchical overview, 216

Leaf, 202

Node hierarchy, 201

Nodes

0, 289

1, 289

1V2, 239

1V8, 239

2V5, 239

3V3, 239

5V0, 239

ABOUT, 228

AC, 267, 280

ACTIVE, 262, 273, 275

ACTIVEINTERFACE, 231

ACTIVETHRESHOLD, 263

ACTIVETIMECONSTANT, 263

ADC0MAX, 237

ADC0MIN, 237

ADC1MAX, 237

ADC1MIN, 237

ADCLIP, 236

ADCSELECT, 240, 251

ADCTHRESHOLD, 255

ADD, 282

AMPLITUDE, 248, 251, 280

AMPLITUDES, 283

APPLICATION, 269

AUTO, 271

AUTOCENTER, 252

AUTOPID, 253

AUTOTIMECONSTANT, 253

AUXAVG, 255

AUXINS, 288

AUXOUTS, 289

AVAILABLE, 292, 294

AVERAGING, 288

BIAS, 271

BIASOUT, 294

BINARY, 234

BW, 271

BWLIMIT, 285

BYTES RECEIVED, 238

BYTES SENT, 238

CABLELENGTH, 272

CALIB, 272

CAMP, 292

CARRIER, 246

CENTER, 261

CHANNEL, 284

CLOCKBASE, 229, 230

CODE, 233

COMPENSATION, 277

CONFIDENCE, 275

CONFIG, 229

COPYRIGHT, 228

CPUS, 291

CURRENT, 265

CURRENTGAIN, 294

D, 254, 259

DATA, 274, 275

DATASERVER, 228

DC, 293, 295

DCMLOCK, 235

DECIMATION, 287

DEMOD, 267, 279

DEMOS, 240

DEMOSAMPLELOSS, 236

DEMODSELECT, 256, 265, 266, 290

DEV, 230

DEVTYPE, 233

DIFF, 281

DIOS, 286

DRIVE, 287

ECHOREAD, 238

ECHOWRITE, 237

ENABLE, 241, 244, 248, 250, 252, 261, 262, 264, 271, 273, 274, 276, 276, 277, 278, 284

ENABLES, 282

ERROR, 253, 260

EXTBIAS, 294

EXTCLK, 230, 287

FEATURES, 233

FIFOLEVEL, 236

FILTER, 269

FLAGS, 234

FREQ, 242, 243, 265

FREQCENTER, 252

FREQDELTA, 254

FREQDEV, 245

FREQDEVENABLE, 245

FREQRANGE, 252

FREQRESOLUTION, 232

FWREVISION, 228

FX2RX, 235

GAIN, 293

HARMONIC, 242, 247, 250, 255, 267

HWREVISION, 230

I, 254, 259

IMP50, 281

IMPEDANCERANGE, 270

IMPS, 264

INACTIVETHRESHOLD, 264

INACTIVETIMECONSTANT, 263
INDEX, 245
INFO, 273, 275
INPUT, 257, 288
INPUTCHANNEL, 257
INPUTRANGE, 271
INPUTSELECT, 246, 249, 265, 266
INTERNAL, 274
INTERPOLATION, 269
INVERT, 265, 266
MAXBANDWIDTH, 270
MAXFREQ, 232
MAXTIMECONSTANT, 232
MEANMSGCNT, 238
MEANPOLLCNT, 238
MINFREQ, 232
MINTIMECONSTANT, 232
MIXERCLIP, 235
MODE, 245, 248, 278
MODEL, 264
MODS, 244
MONITOROFFSET, 260
MONITORSCALE, 260
NEGATIVEFREQ, 232
OFFSET, 282, 290, 295
OMEGASUPPRESSION, 270
ON, 278, 281
OPEN, 229
OPENDECTECT, 276
OPTIONS, 233
ORDER, 240, 246, 249, 256, 268
OSCS, 243
OSCSELECT, 242, 247, 250, 256, 267
OUTPUT, 244, 258, 272, 278, 287, 291
OUTPUTCHANNEL, 258
OUTPUTDEFAULT, 258
OUTPUTDEFAULTENABLE, 258
OUTPUTSELECT, 290
P, 254, 259
PHASESHIFT, 243, 247, 250
PHYSICAL, 239
PIDS, 257
PKGLOSS, 235
PLL, 262
PLLLOCK, 234
PLLS, 251
PORT, 229
PROGRAM, 291
PROPERTIES, 231
R, 292
RANGE, 261, 266, 267, 279, 280, 282
RATE, 241, 245, 256, 262, 269, 279
RATIO, 276, 277, 277, 278
REVISION, 228
SAMPLE, 243, 251, 280, 289
SCALE, 290
SCOPES, 284
SCOPESKIPPED, 236
SELECT, 279
SERIAL, 233
SETPOINT, 255, 259
SETPOINTSELECT, 260
SHIFT, 261
SIDEBANDS, 248
SIGINS, 280
SIGOUTS, 281
SINC, 243, 268
SINGLEENDED, 293
STATS, 238
STATUS, 233
STORE, 274, 275
SUPPRESS, 272
SUPPRESSION, 276
SYNCENABLE, 230
SYNCRESET, 231
SYNCSELECT0, 288
SYNCSELECT1, 288
SYNCTIME, 231
SYSTEM, 230
TAMP, 293
TEMP, 240
TIME, 233, 286
TIMEBASE, 232
TIMECONSTANT, 241, 247, 249, 253, 268
TIPPROTECT, 262
TREES, 229
TRIGCHANNEL, 284
TRIGEDGE, 285
TRIGGER, 241, 246
TRIGHOLDOFF, 286
TRIGLEVEL, 285
UNDERFLOW, 277
USER, 273
USERREGS, 291
VALID, 273, 274
VALUE, 270, 271, 289
VALUES, 289
VERSION, 228
VOLTAGE, 266
VOLTAGEGAIN, 295
WAVE, 286
WAVEFORMS, 283
WORKLOAD, 291
ZCTRLS, 292
ZI, 228
Noise
 1/f, 653
 HF Input, 634
 Phase noise, 629
 Tutorial, 67-67
Numeric Tab, 119
Numerical
 Tool
 Tutorial, 50

Nyquist sampling theorem, 49, 626

O

Oscilloscope

Tool

Tutorial, 50

Output

ADD, 58, 629

HF, 629

Sync, 37

P

Performance diagrams, 634

Phase

Resolution, 627

Phase Lock Loop, PLL

Tutorial, 77

PID

Block diagram, 176

PID Tab, 175

PLL

Block diagram, 168

PLL Tab, 167

Plotter Tab, 122

Polling Data

Concept, 202

Preamplifier

CA, 193

TA, 195

Q

Quadrature, 652

R

Real-time Tab, 192

Reference mode

Auto, 45

External, 43, 627

Tutorial, 63-66, 64

Internal, 42, 627

Reference signal, 627, 652

RMS value, 653

S

Scope Tab, 125

Self calibration, 30

Sensitivity (see Full range sensitivity)

Software Installation

Linux, 16

Microsoft .NET Framework, 32

Requirements, Linux, 16

Supported versions of Linux, 16

Windows, 13

Spectroscope

Tool

Tutorial, 50

Spectrum Analyzer Tab, 137

Stability

Input amplitude, 627

Status bar

Description, 86

SW Trigger Tab, 130

Sweeper

Tool

Tutorial, 51

Sweeper Tab, 141

T

TA (see Preamplifier, TA)

Time constant, 629

Tool-set

Description, 84

Transfer function, 52

Tree Sub-Tab

Description, 90

Tutorial

LabOne UI, 48

U

UHS, 630

User Interface

Description, 82

V

Vertical Axis Groups

Description, 91

W

WEB Option, 13

Windows

Software installation, 13

Z

ZCtrl, 36

ziRTK, API functions and data types

Bits (see DIOSample)

Reserved (see DemodSample)

TS (see AuxInSample) (see DemodSample) (see DIOSample)

TS32 (see TS_t)

TS64 (see TS_t)

Val32 (see Val_t)

Val64 (see Val_t)

Values (see AuxInSample)

X (see DemodSample)

Y (see DemodSample)

ziRTKAddAuxInSampleTrigger, 335, 508

ziRTKAddClockTrigger, 336, 509

ziRTKAddDemodSampleTrigger, 333, 506

ziRTKAddDIOSampleTrigger, 334, 507

ziRTKAddUserRegTrigger, 337, 510

ziRTKAuxInGetAveraging, 445, 600

ziRTKAuxInGetCount, 444, 599
ziRTKAuxInGetSample, 447, 602
ziRTKAuxInSampleGetTimeStamp, 448, 603
ziRTKAuxInSampleGetTimeStamp32, 449, 604
ziRTKAuxInSampleGetTimeStamp64, 450, 605
ziRTKAuxInSampleGetValue, 451, 606
ziRTKAuxInSampleGetValue16, 452, 607
ziRTKAuxInSetAveraging, 446, 601
ziRTKAuxOutGetCount, 429, 586
ziRTKAuxOutGetDemodSelect, 433, 590
ziRTKAuxOutGetOffset, 435, 592
ziRTKAuxOutGetOutputSelect, 431, 588
ziRTKAuxOutGetScale, 440, 597
ziRTKAuxOutGetValue, 430, 587
ziRTKAuxOutSetDemodSelect, 434, 591
ziRTKAuxOutSetOffset, 436, 593
ziRTKAuxOutSetOffsetInt, 438, 595
ziRTKAuxOutSetOffsetIntNoUpdate, 439, 596
ziRTKAuxOutSetOffsetNoUpdate, 437, 594
ziRTKAuxOutSetOutputSelect, 432, 589
ziRTKAuxOutSetScale, 441, 598
ziRTKDemodGetADCSelect, 380, 540
ziRTKDemodGetCount, 379, 539
ziRTKDemodGetHarmonic, 386, 546
ziRTKDemodGetOrder, 384, 544
ziRTKDemodGetOscSelect, 382, 542
ziRTKDemodGetPhaseShift, 392, 552
ziRTKDemodGetRate, 388, 548
ziRTKDemodGetSample, 398, 558
ziRTKDemodGetSinc, 396, 556
ziRTKDemodGetTimeConstant, 394, 554
ziRTKDemodGetTrigger, 390, 550
ziRTKDemodSampleGetCompR, 412, 572
ziRTKDemodSampleGetCompX, 405, 565
ziRTKDemodSampleGetCompXY, 410, 570
ziRTKDemodSampleGetCompY, 409, 569
ziRTKDemodSampleGetR, 411, 571
ziRTKDemodSampleGetTheta, 413, 573
ziRTKDemodSampleGetTimeStamp, 399, 559
ziRTKDemodSampleGetTimeStamp32, 400, 560
ziRTKDemodSampleGetTimeStamp64, 401, 561
ziRTKDemodSampleGetX, 402, 562
ziRTKDemodSampleGetX32, 403, 563
ziRTKDemodSampleGetX64, 404, 564
ziRTKDemodSampleGetY, 406, 566
ziRTKDemodSampleGetY32, 407, 567
ziRTKDemodSampleGetY64, 408, 568
ziRTKDemodSetADCSelect, 381, 541
ziRTKDemodSetHarmonic, 387, 547
ziRTKDemodSetOrder, 385, 545
ziRTKDemodSetOscSelect, 383, 543
ziRTKDemodSetPhaseShift, 393, 553
ziRTKDemodSetRate, 389, 549
ziRTKDemodSetSinc, 397, 557
ziRTKDemodSetTimeConstant, 395, 555
ziRTKDemodSetTrigger, 391, 551
ziRTKDIOGetCount, 456, 608
ziRTKDIOGetDecimation, 459, 611
ziRTKDIOGetDrive, 461, 613
ziRTKDIOGetExtClk, 457, 609
ziRTKDIOGetOutput, 463, 615
ziRTKDIOGetSample, 466, 618
ziRTKDIOSampleGetBits, 470, 622
ziRTKDIOSampleGetTimeStamp, 467, 619
ziRTKDIOSampleGetTimeStamp32, 468, 620
ziRTKDIOSampleGetTimeStamp64, 469, 621
ziRTKDIOSetDecimation, 460, 612
ziRTKDIOSetDrive, 462, 614
ziRTKDIOSetExtClk, 458, 610
ziRTKDIOSetOutput, 464, 616
ziRTKDIOSetOutputNoUpdate, 465, 617
ziRTKExtMemGet, 359, 525
ziRTKExtMemSet, 360, 526
ziRTKFeaturesGetDevType, 349, 518
ziRTKFeaturesGetOptions, 348, 517
ziRTKGetTimeStamp, 323, 498
ziRTKGetTimeStamp32, 324, 499
ziRTKGetTimeStamp64, 325, 500
ziRTKGetUpdateHostPC, 329, 504
ziRTKGetWorkload, 326, 501
ziRTKInit, 320, 496
ziRTKLoop, 321, 497
ziRTKOscGetCount, 372, 536
ziRTKOscGetFreq, 373, 537
ziRTKOscSetFreq, 374, 538
ziRTKPause, 330, 505
ziRTKPLLSetADCSelect, 472, 623
ziRTKPrintf, 340, 512
ziRTKSigInGetAC, 365, 530
ziRTKSigInGetCount, 362, 527
ziRTKSigInGetDiff, 369, 534
ziRTKSigInGetImp50, 367, 532
ziRTKSigInGetRange, 363, 528
ziRTKSigInSetAC, 366, 531
ziRTKSigInSetDiff, 370, 535
ziRTKSigInSetImp50, 368, 533
ziRTKSigInSetRange, 364, 529
ziRTKSigOutGetAdd, 425, 584
ziRTKSigOutGetAmplitude, 421, 580
ziRTKSigOutGetChannelCount, 418, 577
ziRTKSigOutGetCount, 415, 574
ziRTKSigOutGetEnable, 419, 578
ziRTKSigOutGetOn, 423, 582
ziRTKSigOutGetRange, 416, 575
ziRTKSigOutSetAdd, 426, 585
ziRTKSigOutSetAmplitude, 422, 581
ziRTKSigOutSetEnable, 420, 579
ziRTKSigOutSetOn, 424, 583
ziRTKSigOutSetRange, 417, 576
ziRTKSystemGetExtClk, 351, 519
ziRTKSystemGetHWRevision, 353, 521
ziRTKSystemSetExtClk, 352, 520
ziRTKTestTrigger, 338, 511
ziRTKUpdateHostPCOff, 328, 503

ziRTKUpdateHostPCOn, 327, 502
ziRTKUserRegGet, 356, 523
ziRTKUserRegGetCount, 355, 522
ziRTKUserRegSet, 357, 524
ziRTKWriteData, 344, 516
ziRTKWriteString, 343, 515
ZIRTK_DEVTYPE, 347, 495
ZIRTK_DEVTYPE_DEFAULT (see ZIRTK_DEVTYPE)
ZIRTK_DEVTYPE_IS (see ZIRTK_DEVTYPE)
ZIRTK_DEVTYPE_LI (see ZIRTK_DEVTYPE)
ZIRTK_DEVTYPE_UNKNOWN (see
ZIRTK_DEVTYPE)
ZIRTK_OPTIONS, 346, 494
ZIRTK_OPTION_MF (see ZIRTK_OPTIONS)
ZIRTK_OPTION_MOD (see ZIRTK_OPTIONS)
ZIRTK_OPTION_NONE (see ZIRTK_OPTIONS)
ZIRTK_OPTION_PID (see ZIRTK_OPTIONS)
ZIRTK_OPTION_PLL (see ZIRTK_OPTIONS)
ZIRTK_OPTION_RTK (see ZIRTK_OPTIONS)
ZIRTK_OPTION_UHS (see ZIRTK_OPTIONS)
ZI_LIMIT (see ZI_STATUS)
ZI_MAX_STATUS (see ZI_STATUS)
ZI_NULLPTR (see ZI_STATUS)
ZI_OUTOFRANGE (see ZI_STATUS)
ZI_STATUS, 493
ZI_SUCCESS (see ZI_STATUS)
ziServer, 199, 212
 Check status, 201
 on Linux, 206
 on Windows, 204
 ziService (Linux), 17
ziService, 201
ziService Program, 17
ZoomFFT, 664-665
 Tool
 Tutorial, 52
ZSync, 36