

Computer-Based Instruments

NI 54XX Calibration Procedure

Worldwide Technical Support and Product Information

ni.com

National Instruments Corporate Headquarters

11500 North Mopac Expressway Austin, Texas 78759-3504 USA Tel: 512 794 0100

Worldwide Offices

Australia 03 9879 5166, Austria 0662 45 79 90 0, Belgium 02 757 00 20, Brazil 011 284 5011,
Canada (Calgary) 403 274 9391, Canada (Ottawa) 613 233 5949, Canada (Québec) 514 694 8521,
Canada (Toronto) 905 785 0085, China (Shanghai) 021 6555 7838, China (ShenZhen) 0755 3904939,
Denmark 45 76 26 00, Finland 09 725 725 11, France 01 48 14 24 24, Germany 089 741 31 30,
Greece 30 1 42 96 427, Hong Kong 2645 3186, India 91805275406, Israel 03 6120092, Italy 02 413091,
Japan 03 5472 2970, Korea 02 596 7456, Malaysia 603 9596711, Mexico 5 280 7625, Netherlands 0348 433466,
New Zealand 09 914 0488, Norway 32 27 73 00, Poland 0 22 528 94 06, Portugal 351 1 726 9011,
Singapore 2265886, Spain 91 640 0085, Sweden 08 587 895 00, Switzerland 056 200 51 51,
Taiwan 02 2528 7227, United Kingdom 01635 523545

For further support information, see the *Technical Support Resources* appendix. To comment on the documentation, send e-mail to techpubs@ni.com

Copyright © 2001 National Instruments Corporation. All rights reserved.

Important Information

Warranty

The NI 54XX is warranted against defects in materials and workmanship for a period of one year from the date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace equipment that proves to be defective during the warranty period. This warranty includes parts and labor.

The media on which you receive National Instruments software are warranted not to fail to execute programming instructions, due to defects in materials and workmanship, for a period of 90 days from date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace software media that do not execute programming instructions if National Instruments receives notice of such defects during the warranty period. National Instruments does not warrant that the operation of the software shall be uninterrupted or error free.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. National Instruments will pay the shipping costs of returning to the owner parts which are covered by warranty.

National Instruments believes that the information in this document is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, National Instruments reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult National Instruments if errors are suspected. In no event shall National Instruments be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, NATIONAL INSTRUMENTS MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF NATIONAL INSTRUMENTS SHALL BE LIMITED TO THE AMOUNT THEREOF PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

Copyright

Under the copyright laws, this publication may not be reproduced or transmitted in any form, electronic or mechanical, including photocopying, recording, storing in an information retrieval system, or translating, in whole or in part, without the prior written consent of National Instruments Corporation.

Trademarks

CVI™, LabVIEW™, National Instruments™, NI™, ni.com™, NI-DAQ™, and SCXI™ are trademarks of National Instruments Corporation.

Product and company names mentioned herein are trademarks or trade names of their respective companies.


WARNING REGARDING USE OF NATIONAL INSTRUMENTS PRODUCTS

(1) NATIONAL INSTRUMENTS PRODUCTS ARE NOT DESIGNED WITH COMPONENTS AND TESTING FOR A LEVEL OF RELIABILITY SUITABLE FOR USE IN OR IN CONNECTION WITH SURGICAL IMPLANTS OR AS CRITICAL COMPONENTS IN ANY LIFE SUPPORT SYSTEMS WHOSE FAILURE TO PERFORM CAN REASONABLY BE EXPECTED TO CAUSE SIGNIFICANT INJURY TO A HUMAN.

(2) IN ANY APPLICATION, INCLUDING THE ABOVE, RELIABILITY OF OPERATION OF THE SOFTWARE PRODUCTS CAN BE IMPAIRED BY ADVERSE FACTORS, INCLUDING BUT NOT LIMITED TO FLUCTUATIONS IN ELECTRICAL POWER SUPPLY, COMPUTER HARDWARE MALFUNCTIONS, COMPUTER OPERATING SYSTEM SOFTWARE FITNESS, FITNESS OF COMPILERS AND DEVELOPMENT SOFTWARE USED TO DEVELOP AN APPLICATION, INSTALLATION ERRORS, SOFTWARE AND HARDWARE COMPATIBILITY PROBLEMS, MALFUNCTIONS OR FAILURES OF ELECTRONIC MONITORING OR CONTROL DEVICES, TRANSIENT FAILURES OF ELECTRONIC SYSTEMS (HARDWARE AND/OR SOFTWARE), UNANTICIPATED USES OR MISUSES, OR ERRORS ON THE PART OF THE USER OR APPLICATIONS DESIGNER (ADVERSE FACTORS SUCH AS THESE ARE HEREAFTER COLLECTIVELY TERMED "SYSTEM FAILURES"). ANY APPLICATION WHERE A SYSTEM FAILURE WOULD CREATE A RISK OF HARM TO PROPERTY OR PERSONS (INCLUDING THE RISK OF BODILY INJURY AND DEATH) SHOULD NOT BE RELIANT SOLELY UPON ONE FORM OF ELECTRONIC SYSTEM DUE TO THE RISK OF SYSTEM FAILURE. TO AVOID DAMAGE, INJURY, OR DEATH, THE USER OR APPLICATION DESIGNER MUST TAKE REASONABLY PRUDENT STEPS TO PROTECT AGAINST SYSTEM FAILURES, INCLUDING BUT NOT LIMITED TO BACK-UP OR SHUT DOWN MECHANISMS. BECAUSE EACH END-USER SYSTEM IS CUSTOMIZED AND DIFFERS FROM NATIONAL INSTRUMENTS' TESTING PLATFORMS AND BECAUSE A USER OR APPLICATION DESIGNER MAY USE NATIONAL INSTRUMENTS PRODUCTS IN COMBINATION WITH OTHER PRODUCTS IN A MANNER NOT EVALUATED OR CONTEMPLATED BY NATIONAL INSTRUMENTS, THE USER OR APPLICATION DESIGNER IS ULTIMATELY RESPONSIBLE FOR VERIFYING AND VALIDATING THE SUITABILITY OF NATIONAL INSTRUMENTS PRODUCTS WHENEVER NATIONAL INSTRUMENTS PRODUCTS ARE INCORPORATED IN A SYSTEM OR APPLICATION, INCLUDING, WITHOUT LIMITATION, THE APPROPRIATE DESIGN, PROCESS AND SAFETY LEVEL OF SUCH SYSTEM OR APPLICATION.

Conventions

The following conventions are used in this manual:

- [] Square brackets enclose optional items—for example, [response].
- » The » symbol leads you through nested menu items and dialog box options to a final action. The sequence **File»Page Setup»Options** directs you to pull down the **File** menu, select the **Page Setup** item, and select **Options** from the last dialog box.
-  This icon denotes a note, which alerts you to important information.
- bold** Bold text denotes items that you must select or click on in the software, such as menu items and dialog box options. Bold text also denotes parameter names.
- italic* Italic text denotes variables, emphasis, a cross reference, or an introduction to a key concept. This font also denotes text that is a placeholder for a word or value that you must supply.
- monospace Text in this font denotes text or characters that you should enter from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames and extensions, and code excerpts.
- monospace italic* Italic text in this font denotes text that is a placeholder for a word or value that you must supply.

Contents

Chapter 1

Introduction—Document Scope

What Is Calibration?	1-1
Internal Adjustment	1-1
External Adjustment	1-1
Why Should You Calibrate?.....	1-2
How Often Should You Calibrate?.....	1-2
Equipment and Other Test Requirements	1-2
Test Equipment.....	1-2
Software.....	1-3
Writing Your Calibration Procedure	1-3
Documentation	1-5
Test Conditions.....	1-6

Chapter 2

Verification

Initializing Device for Verification.....	2-3
Verifying DC Offset	2-3
Verifying Amplitude Accuracy	2-4
Verifying Amplitude Flatness over Frequency.....	2-5
Verifying Total Harmonic Distortion and Spurious Free Dynamic Range	2-8
Verifying Device Oscillator Frequency Accuracy.....	2-10

Chapter 3

Adjustment

Internal Adjustment	3-1
External Adjustment	3-1
Setting Up and Configuring Your Device	3-1
Adjusting Your NI 54XX—Procedural Overview	3-4
Finding the Low Frequency Characteristics of the Analog Channel	3-5
Determining the Value of the On-Board High-Precision Voltage Reference.....	3-17
Determining the Frequency Response across the Passband.....	3-19
Determining the SYNC Duty Cycle Constants.....	3-21
Determining the VCXO Calibration Constants	3-25

Appendix A
Overview of the Calibration Constants

Appendix B
Function Reference

Appendix C
Function Constants

Appendix D
Technical Support Resources

Introduction—Document Scope

This chapter discusses what calibration is, why you should do it, and how often. You will also learn about the necessary equipment for calibrating your signal source.

Use the following procedure to calibrate the NI 54XX device series of function generators, arbitrary waveform generators, and video signal generators. This series includes NI 5401, 5411, and 5431 devices on PCI or PXI buses.

What Is Calibration?

Calibration consists of verifying the accuracy of a device and adjusting for any error. To calibrate the NI 54XX devices, you need to initiate a sequence of signal outputs and measure these signals to calculate calibration constants that are stored in non-volatile memory on the device. When you use the device to generate signals, the driver software retrieves these constants and uses them to determine the fine adjustments needed to achieve optimal signal characteristics.

You have two options for calibrating your NI 54XX devices: internal adjustment and external adjustment.

Internal Adjustment

You can do internal adjustment, or self-calibration, with the software command `ni54xx_CalSelfCalibrate`. The NI 54XX device automatically calculates and stores certain constants. It is faster than external adjustment because the device does everything automatically, and no external measurement devices are needed.

External Adjustment

External adjustment involves measuring device output under different configurations using high-precision external measurement devices. Calibrate externally when you want to do a complete calibration of all constants to a high degree of accuracy.

Why Should You Calibrate?

The characteristics of electronic components drift with time and temperature, which can affect output accuracy as the device ages. Calibration restores your device to NI standards.

How Often Should You Calibrate?

The output accuracy requirements of your application determine the calibration interval of your NI 54XX device. See the recommended calibration schedule in Table 1-1. Refer to Chapter 2, *Verification*, at any time to determine if your device needs a full calibration.

Table 1-1. Recommended Calibration Schedule

Calibration Type	Calibration Time Interval	Calibration Temperature Interval
Internal	1 week	± 5 °C
External	1 year	—

Equipment and Other Test Requirements

This section describes the equipment, software, documentation, and test conditions required for calibrating your NI 54XX device.

Test Equipment

Internal adjustment does not require any test equipment. External adjustment requires different equipment for each type of measurement.

Table 1-2. Equipment Required for Calibrating NI 54XX Devices

Instrument	Specifications	Recommended Equipment
5 1/2 Digit Digital Multimeter (DMM)	At least 1 kHz bandwidth	NI 4060 or HP 34401A
Digital Counter	Ability to make duty cycle measurements, measure 1 MHz frequency, and have frequency measurement accuracy ≤ 1 ppm (accounting for clock stability and measurement error due to resolution)	NI 6608 or HP 53131A with appropriate timebase option
BNC Cable	50 Ω	—
BNC Connector (for signal sources in PXI form factor)	50 Ω	—
SMB Connector (for signal sources in PCI form factor)	50 Ω	—
50 Ω Terminator	—	—
Spectrum Analyzer	Bandwidth > 32 MHz Distortion < -70 dBc Spurs to 32 MHz	HP 8590 or better
Power Meter or Thermoconverter	100 kHz to 16 MHz equal or better than $\pm 2\%$ or ± 0.2 dB	—

Software

This section describes the software and documentation you need to calibrate your NI 54XX device.

Writing Your Calibration Procedure

The calibration process is described in Chapter 3, *Adjustment*, including step-by-step instructions on calling the appropriate calibration functions.

Be aware that many of the functions listed in Chapter 3, *Adjustment*, use constants defined in the `niArbCal.h` file. To use these variables, you must include `niArbCal.h` in your code. See Table 1-3 for file locations.

Calibration Software

The calibration procedure requires the latest version of the NI-DAQ driver on the calibration system. This driver configures and controls the NI 54XX devices. You can download NI-DAQ from the NI Web site, ni.com/softlib.nsf. NI-DAQ supports programming all NI signal sources using a number of languages, including LabVIEW, Measurement Studio, Microsoft Visual C++, and Microsoft Visual Basic. When you install NI-DAQ, you only need to install support for the programming language that you intend to use.

You also need a copy of the `niArbCal.dll` to calibrate your NI 54XX device. To write calibration procedures in C, you must include the `niArbCal.h` file in the code that calls the calibration functions, and you must link the `niArbCal.lib` file into the build of your executable. Copy this DLL to the system directory, which is generally `c:\windows\system` on 9x systems and `c:\winnt\system32` on NT systems. `niArbCal.dll` provides calibration functionality that does not reside in the standard NI-DAQ driver. This functionality includes protecting the calibration constants and updating the calibration date. You can access the functions in this DLL through any 32-bit compiler or LabVIEW.

To write calibration procedures in LabVIEW, you must use the VIs included in the `ni54xx_cal.llb`. This library contains a VI for each function exported by the `niArbCal` DLL, as well as a few useful others. After installation, all of these VIs appear within the `ni54xx` palette, under the Instrument Drivers section of the Functions palette. The `ni54xx` palette also contains example VIs, which illustrate the use of the calibration functions.

To write calibration procedures in Visual Basic, you must first configure your project to reference the `niArbCal` DLL. To do so, select **Project»References** from the menu bar, click **Browse**, navigate to your system directory, find and select `niArbCal.dll`, click **Open**, then click **OK**. You will then be able to use the `niArbCal` functions in your code and look up the functions, parameters, and constants in the Object Browser.

Table 1-3. Calibration File Location

File Name and Location	Description
<system directory>\niArbCal.dll	The NI 54XX calibration toolkit library. This provides the functionality for calibrating your device.
vxipnp\winnt(win95)\lib\msc\niArbCal.lib	This is a .lib file that allows you to create applications which call functions in the niArbCal DLL. You must link to this library when building applications in Microsoft Visual C++ or National Instruments CVI.
vxipnp\winnt(win95)\include\niArbCal.h	This is a header file for the accessible functions in the niArbCal DLL. You must include this file in any C code that you write in order to call these functions.
vxipnp\winnt(win95)\ SimpleCalibrationExample.c vxipnp\winnt(win95)\ ExternalCalibrationExample.c	These are examples, written in C, that illustrate the use of the functions in the niArbCal DLL to accomplish calibration related tasks.
LabVIEW(6)\instr.lib\ni54xx\ni54xx_cal.llb	This LabVIEW LLB contains VIs that correspond to the functions in the niArbCal DLL.
LabVIEW(6)\instr.lib\ni54xx\ External_Calibration_Example.llb	This LabVIEW LLB contains VIs that demonstrate an external adjustment procedure using the VIs in the ni54xx_cal LLB.

Documentation

This calibration procedure requires information on installing the NI 54XX devices, which you can find in the following documents:

- *NI 5411/5431 User Manual*
- *DAQ Quick Start Guide*

The following documents contain information on using the NI-DAQ driver:

- *NI-DAQ Function Reference Online Help*
- *NI-DAQ User Manual for PC Compatibles*

The *NI-DAQ Function Reference Online Help* includes detailed information on the driver functions. You can access the online help by clicking **Start»Programs»National Instruments DAQ»NI-DAQ Help**. The *NI-DAQ User Manual* provides instructions for installing and configuring National Instruments DAQ devices.

Test Conditions

Follow these guidelines to optimize the connections and the environment during calibration:

- Keep connections to the NI 54XX device short. Long cables and wires act as antennae, picking up extra noise that can affect measurements.
- Use twisted-pair shielded copper wire for all cable connections to the device for any part of cabling that is not BNC to eliminate noise and thermal offsets.
- Keep relative humidity below 80%.
- Maintain a temperature between 15 and 35 °C.

Verification

This chapter provides detailed step-by-step instructions for verifying the specifications of your signal source. Verification determines whether your device is performing within its specifications prior to adjustment. Verification and adjustment together comprise a complete calibration. To verify that your NI 54XX still meets its specifications, you must use the NI-FGEN device driver to control the NI 54XX device. The following steps describe the code you use to generate the appropriate signals and the NI-FGEN function calls you make to verify specifications. The code varies depending on which programming application you use: LabVIEW, CVI/C, or Visual Basic. The examples shown in this section use CVI/C code, but LabVIEW has a corresponding VI for each function.

You can verify the following specifications for NI 54XX devices:

- DC offset
- Amplitude accuracy
- Amplitude flatness over frequency
- Total harmonic distortion (THD)
- Spurious free dynamic range (SFDR)
- Device oscillator frequency accuracy

Table 2-1. Specifications Table

Characteristic	Value
DC Offset Accuracy	± 5 mV
Amplitude Accuracy	± 1 dB
Amplitude Flatness Over Frequency	± 2 dB
Sine Spectral Purity (Harmonic and Spurious) Up to 1 MHz Up to 16 MHz	-60 dBc -35 dBc
Oscillator Frequency Accuracy	± 15 ppm, 20 to 30 °C ambient temperature

The verification procedure for each of these specifications includes setting up, programming, and cleaning up. The setup routine described in the next section, *Initializing Device for Verification*, is the same for each specification you are verifying.

For a diagram of the connectors referenced in the procedure that follows, see Figure 2-1.

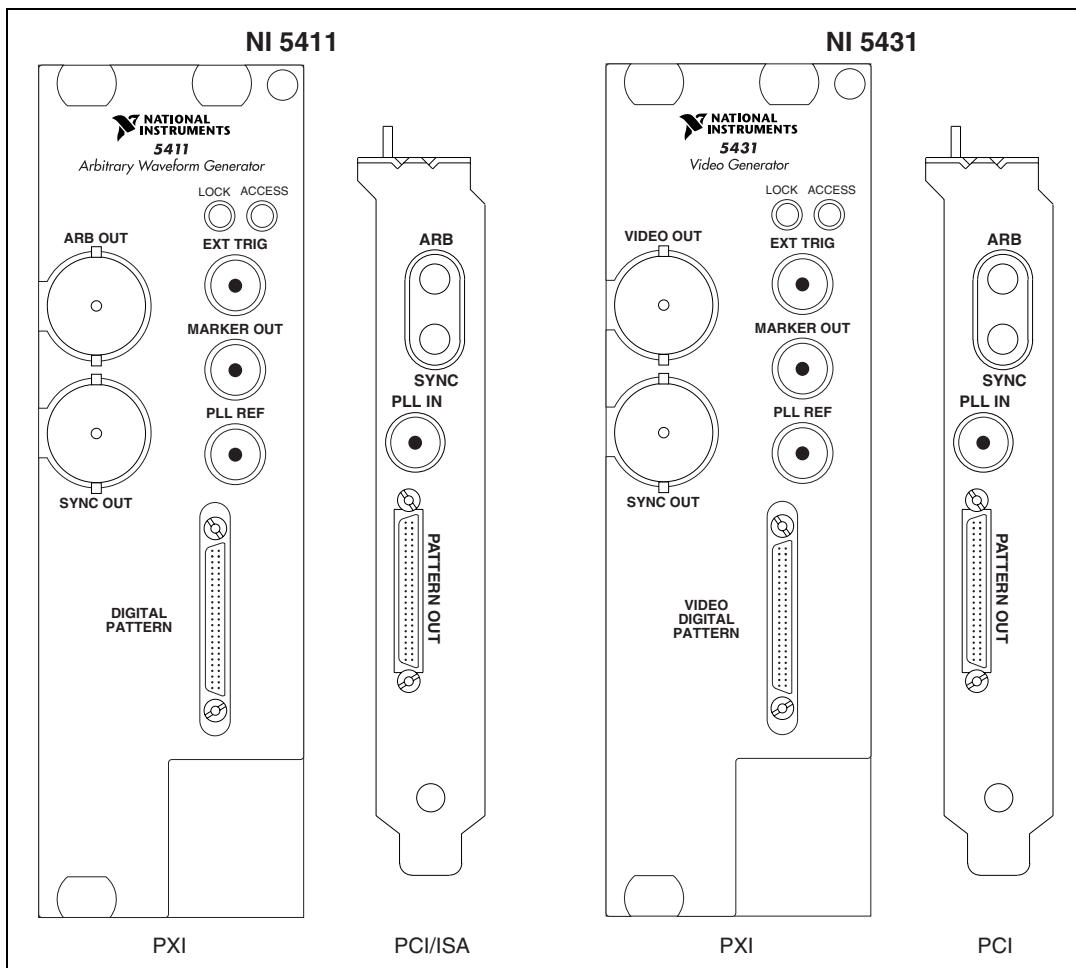


Figure 2-1. NI 5411/5431 I/O Connectors

Initializing Device for Verification

Complete the following setup steps for each of the specifications you want to verify prior to beginning the verification steps:

1. Call `niFgen_init()` to initialize the instrument you are testing and to create an I/O Session. Set the following parameters:
 - **vi**—The output that is passed in by reference to the verification functions as `sessionHandle`
 - **resourceName**—"DAQ::#", where # is the device number
 - **ID Query**—True
 - **Reset Device**—True
2. Call `niFgen_ConfigureOutputMode()` to select the standard function output mode for the function generator to use. Set the following parameters:
 - **vi**—The output value `sessionHandle` that you obtained from the `niFgen_init` function
 - **OutPut_Mode**—`NIFGEN_VAL_OUTPUT_FUNC`

Verifying DC Offset

Complete the following steps to measure the DC offset of your NI 54XX device when you use it to generate a 0.0 V DC voltage:

1. Complete the setup steps described in the *Initializing Device for Verification* section.
2. Call `niFgen_ConfigureStandardWaveform()` to select the channel to output a waveform. Set the following parameters:
 - **vi**—The output value `sessionHandle` that you obtained from the `niFgen_init` function
 - **Channel_Name**—"0"
 - **Amplitude**—0
 - **DC_Offset**—0
 - **Frequency**—0
 - **Waveform**—`NIFGEN_VAL_WFM_DC`
 - **Start_Phase**—0

3. Call `niFgen_InitiateGeneration()` to initiate signal generation. This function causes the function generator to leave its configuration state and enter its signal generation state. The only parameter required for this function is `sessionHandle`.
4. Measure the 0.0 V DC generated by your NI 54XX device at ARB OUT using the DMM. The deviation from 0.0 V—the offset—should be less than ± 5 mV DC.
5. Call `niFgen_Close()` to close the instrument I/O session, destroy the instrument driver session and all of its properties, and release any memory resources the driver uses. The only parameter required for this function is the `sessionHandle`.

Verifying Amplitude Accuracy

Use a DMM to measure the amplitude accuracy of your NI 54XX by generating a 10 Vpp (into 50 Ω) 1 kHz sine wave:

1. Complete the setup steps described in the [Initializing Device for Verification](#) section.
2. Call `niFgen_EnableAnalogFilter()` to select the channel to enable an analog filter on. Set the following parameters:
 - **vi**—The output value `sessionHandle` that you obtained from the `niFgen_init` function
 - **Channel_Name**—"0"
 - **Filter Correction Frequency**—0
3. Call `niFgen_EnableDigitalFilter()` to select the channel to enable a digital filter on. Set the following parameters:
 - **vi**—The output value `sessionHandle` that you obtained from the `niFgen_init` function
 - **Channel_Name**—"0"
4. Call `niFgen_ConfigureStandardWaveform()` to select the channel to output a waveform. Set the following parameters:
 - **vi**—The output value `sessionHandle` that you obtained from the `niFgen_init` function
 - **Channel_Name**—"0"
 - **Amplitude**—10
 - **DC_Offset**—0
 - **Frequency**—1000

- **Waveform**—`NIFGEN_VAL_WFM__SINE`
 - **Start_Phase**—`0`
5. Call `niFgen_InitiateGeneration()` to initiate signal generation. This function causes the function generator to leave its configuration state and enter its signal generation state. The only parameter required for this function is `sessionHandle`.
 6. Use the DMM to measure the rms voltage of the 10 Vpp sine wave (3.53553 Vrms) generated by your NI 54XX device at ARB OUT. The value should fall between 3.49506 Vrms and 3.57647 Vrms.
 7. Call `niFgen_Close()` to close the instrument I/O session, destroy the instrument driver session and all of its properties, and release any memory resources the driver uses. The only parameter required for this function is the `sessionHandle`.

You have completed verifying the amplitude accuracy of your device.

Verifying Amplitude Flatness over Frequency

Use a power meter or thermoconverter to measure the amplitude accuracy of your NI 54XX by generating a 100 kHz sine wave. Use the largest possible amplitude that the power meter can accept, but ensure that your power meter meets the specifications from Table 1-2, [Equipment Required for Calibrating NI 54XX Devices](#). Make sure that the analog filter and the digital filter are both on as you complete the following steps:

1. Complete the setup steps described in the *Initializing Device for Verification* section of this chapter.
2. Call `niFgen_EnableAnalogFilter()` to select the channel to enable an analog filter on. Set the following parameters:
 - **vi**—The output value `sessionHandle` that you obtained from the `niFgen_init` function
 - **Channel_Name**—`"0"`
 - **Filter_Correction_Frequency**—`0`
3. Call `niFgen_EnableDigitalFilter()` to select the channel to enable a digital filter on. Set the following parameters:
 - **vi**—The output value `sessionHandle` that you obtained from the `niFgen_init` function
 - **Channel_Name**—`"0"`

4. Call `niFgen_ConfigureStandardWaveform()` to select the channel to output a waveform. Set the following parameters:
 - **vi**—The output value `sessionHandle` that you obtained from the `niFgen_init` function
 - **Channel_Name**—"0"
 - **Amplitude**— X , where X = the largest possible peak-to-peak amplitude for the power meter.
 - **DC_Offset**—0
 - **Frequency**—1000
 - **Waveform**—`NIFGEN_VAL_WFM_SINE`
 - **Start_Phase**—0
5. Call `niFgen_InitiateGeneration()` to initiate signal generation. This function causes the function generator to leave its configuration state and enter its signal generation state. The only parameter required for this function is the `sessionHandle`.
6. Use the power meter to measure the power of the signal present at ARB OUT of your NI 54XX device. This measurement is your baseline.
7. Call `niFgen_Close()` to close the instrument I/O session, destroy the instrument driver session and all of its properties, and release any memory resources the driver uses. The only parameter required for this function is the `sessionHandle`.
8. Use the code in steps 9 through 17 to generate sine waves of the same amplitude you used for the 100 kHz sine wave, varying the frequency from 1 MHz to 16 MHz in 1 MHz increments. Measure the power for each frequency in the same manner. Deviation from the baseline you established in step 7 should not exceed ± 2 dB.
9. Complete the setup steps described in the [Initializing Device for Verification](#) section.
10. Call `niFgen_EnableDigitalFilter()` to select the channel to enable a digital filter on. Set the following parameters:
 - **vi**—The output value `sessionHandle` that you obtained from the `niFgen_init` function
 - **Channel_Name**—"0"



Note Steps 11a through 11f are contained in a FOR Loop.

11. Begin FOR Loop. For $i = 1000000$ to 16000000 in increments of 1000000 , follow these steps:
 - a. Call `niFgen_EnableAnalogFilter()` to select the channel to enable an analog filter on. Set the following parameters:
 - **vi**—The output value `sessionHandle` that you obtained from the `niFgen_init` function
 - **Channel_Name**—"0"
 - **Filter_Correction_Frequency**— i
 - b. Call `niFgen_ConfigureStandardWaveform()` to select the channel to output a waveform. Set the following parameters:
 - **vi**—The output value `sessionHandle` that you obtained from the `niFgen_init` function
 - **Channel_Name**—"0"
 - **DC_Offset**—0
 - **Waveform**—`NIFGEN_VAL_WFM_SINE`
 - **Amplitude**— x , where x = the largest possible peak-to-peak amplitude for the power meter
 - **Frequency**— i
 - **Start_Phase**—0
 - c. Call `niFgen_InitiateGeneration()` to initiate signal generation. This function causes the function generator to leave its configuration state and enter its signal generation state. The only parameter required for this function is `sessionHandle`.



Note Measure the power of the signal at this step.

- d. Call `niFgen_AbortGeneration()` to abort the previously initiated signal generation. When you call this function, the function generator leaves its signal generation state and returns to its configuration state. The only parameter required for this function is `sessionHandle`.
- e. Call `niFgen_DisableAnalogFilter()` to disable the analog filter. Set the following parameters:
 - **vi**—The output value `sessionHandle` that you obtained from the `niFgen_init` function
 - **Channel_Name**—"0"
- f. End FOR Loop.

12. Call `niFgen_Close()` to close the instrument I/O session, destroy the instrument driver session and all of its properties, and release any memory resources the driver uses. The only parameter required for this function is `sessionHandle`.

Verifying Total Harmonic Distortion and Spurious Free Dynamic Range

Using a spectrum analyzer, verify the spectral purity of your NI 54XX device by generating a 1 MHz sine wave and a 16 MHz sine wave.

Generate a 1 MHz sine wave with a 10 Vpp amplitude for a 50 Ω load by completing the following steps:

1. Complete the setup steps described in the [Initializing Device for Verification](#) section.
2. Call `niFgen_EnableAnalogFilter()` to select the channel to enable an analog filter on. Set the following parameters:
 - **vi**—The output value `sessionHandle` that you obtained from the `niFgen_init` function
 - **Channel_Name**—"0"
 - **FilterCorrectionFrequency**—1000000
3. Call `niFgen_EnableDigitalFilter()` to select the channel to enable a digital filter on. Set the following parameters:
 - **vi**—The output value `sessionHandle` that you obtained from the `niFgen_init` function
 - **Channel_Name**—"0"
4. Call `niFgen_ConfigureStandardWaveform()` to select the channel to output a waveform. Set the following parameters:
 - **vi**—The output value `sessionHandle` that you obtained from the `niFgen_init` function
 - **Channel_Name**—"0"
 - **Amplitude**—10
 - **DC_Offset**—0
 - **Frequency**—1000000
 - **Waveform**—`NIFGEN_VAL_WFM_SINE`
 - **Start_Phase**—0

5. Call `niFgen_InitiateGeneration()` to initiate signal generation. This function causes the function generator to leave its configuration state and enter its signal generation state. The only parameter required for this function is `sessionHandle`.
6. Connect ARB OUT to the spectrum analyzer. Choose the input attenuator setting of the spectrum analyzer so that the input mixer distortion does not limit the measurements. Refer to the documentation for your spectrum analyzer to ensure measurement accuracy.
7. Verify that there are no harmonics or spurs higher than -60 dBc in comparison to the 1 MHz sine wave carrier.
8. Call `niFgen_Close()` to close the instrument I/O session, destroy the instrument driver session and all of its properties, and release any memory resources the driver uses. The only parameter required for this function is `sessionHandle`.

Generate a 16 MHz sine wave with a 10 Vpp amplitude for a 50 Ω load by completing the following steps:

1. Complete the setup steps described in the [Initializing Device for Verification](#) section.
2. Call `niFgen_EnableAnalogFilter()` to select the channel to enable an analog filter on. Set the following parameters:
 - **vi**—The output value `sessionHandle` that you obtained from the `niFgen_init` function
 - **Channel_Name**—"0"
 - **Filter Correction Frequency**—16000000
3. Call `niFgen_EnableDigitalFilter()` to select the channel to enable a digital filter on. Set the following parameters:
 - **vi**—The output value `sessionHandle` that you obtained from the `niFgen_init` function
 - **Channel_Name**—"0"
4. Call `niFgen_ConfigureStandardWaveform()` to select the channel to output a waveform. Set the following parameters:
 - **vi**—The output value `sessionHandle` that you obtained from the `niFgen_init` function
 - **Channel_Name**—"0"
 - **Amplitude**—10
 - **DC_Offset**—0
 - **Frequency**—16000000

- **Waveform**—`NIFGEN_VAL_WFM_SINE`
 - **Start_Phase**—`0`
5. Call `niFgen_InitiateGeneration()` to initiate signal generation. This function causes the function generator to leave its configuration state and enter its signal generation state. The only parameter required for this function is `sessionHandle`.
 6. Choose the input attenuator setting of the spectrum analyzer such that the input mixer distortion does not limit the measurements.
 7. Verify that there are no harmonics or spurs higher than -35 dBc in comparison to the 1 MHz sine wave carrier.
 8. Call `niFgen_Close()` to close the instrument I/O session, destroy the instrument driver session and all of its properties, and release any memory resources the driver uses. The only parameter required for this function is the `sessionHandle`.

You have completed verifying the THD and the SFDR of your device.

Verifying Device Oscillator Frequency Accuracy

Use a counter/timer to measure the frequency accuracy of your NI 54XX device. The counter/timer should provide frequency measurement accuracy of 15 parts per million (ppm) or better. The measurement accuracy of your counter/timer depends on the accuracy of its timebase and the measurement method you use.

Generate a 1 MHz frequency using your NI 54XX device by completing the following steps:

1. Complete the setup steps described in the [Initializing Device for Verification](#) section.
2. Call `niFgen_EnableAnalogFilter()` to select the channel to enable an analog filter on. Set the following parameters:
 - **vi**—The output value `sessionHandle` that you obtained from the `niFgen_init` function
 - **Channel_Name**—`"0"`
 - **Filter Correction Frequency**—`1000000`

3. Call `niFgen_EnableDigitalFilter()` to select the channel to enable a digital filter on. Set the following parameters:
 - **vi**—The output value `sessionHandle` that you obtained from the `niFgen_init` function
 - **Channel_Name**—"0"
4. Call `niFgen_ConfigureStandardWaveform()` to select the channel to output a waveform. Set the following parameters:
 - **vi**—The output value `sessionHandle` that you obtained from the `niFgen_init` function
 - **Channel_Name**—"0"
 - **Amplitude**—10
 - **DC_Offset**—0
 - **Frequency**—1000000
 - **Waveform**—`NIFGEN_VAL_WFM_SINE`
 - **Start_Phase**—0
5. Call `niFgen_InitiateGeneration()` to initiate signal generation. This function causes the function generator to leave its configuration state and enter its signal generation state. The only parameter required for this function is `sessionHandle`.
6. Measure the 1 MHz frequency generated by the NI 54XX device at SYNC OUT using the counter timer. The frequency measurement should fall between 999,985 Hz and 1,000,015 Hz. A frequency error of 15 Hz in 1 MHz corresponds to an error of 15 ppm and accounts for initial accuracy and frequency deviation caused by temperature and aging.
7. Call `niFgen_Close()` to close the instrument I/O session, destroy the instrument driver session and all of its properties, and release any memory resources the driver uses. The only parameter required for this function is the `sessionHandle`.

You have completed verifying the device oscillator frequency accuracy of your NI 54XX device.

Adjustment

This chapter discusses issues related to self-calibrating your signal source and provides a self-calibration procedure.

Internal Adjustment

Internal adjustment requires no equipment or external connections. To internally calibrate your NI 54XX device, complete the following steps:

1. Disconnect the device outputs.
2. Call `ni54xx_CalSelfCalibrate(device)` with the following parameter:
 - **device**—The number of the device you want to calibrate, which is assigned by Measurement & Automation Explorer (MAX).

The calibration proceeds automatically. The procedure takes approximately one minute to complete.



Note To restore the last external adjustment, call the function `ni54xx_CalRestoreExternalConstants`.

External Adjustment

This section describes the basic steps in an external adjustment.

Setting Up and Configuring Your Device

To set up and configure your device for adjustment, complete the following steps:

1. Install the NI 54XX device in your host computer.
2. Configure the NI 54XX device with MAX.
3. Looking at Table 3-1 and Figure 3-1 as guides, make the appropriate connections to measure the output from the constants you want to calibrate.

Table 3-1. Measuring Calibration Output

Constant to Adjust	Type of Measurement	Output Connection	Measurement Device
A0–A6	DC voltage	ARB OUT/VIDEO OUT	DMM
A7–A9	Magnitude of a sinusoidal voltage (either amplitude, peak-to-peak amplitude, or RMS value)	ARB OUT/VIDEO OUT	DMM
Internal reference	DC voltage	ARB OUT/VIDEO OUT	DMM
Frequency response	Magnitude of a sinusoidal voltage (either amplitude, peak-to-peak amplitude, or RMS value)	ARB OUT/VIDEO OUT	Scope/Power Meter
Duty cycle	Duty cycle	SYNC OUT	Counter
VCXO	Frequency	SYNC OUT	Counter

Figure 3-1 shows the NI 5411/5431 I/O connectors.

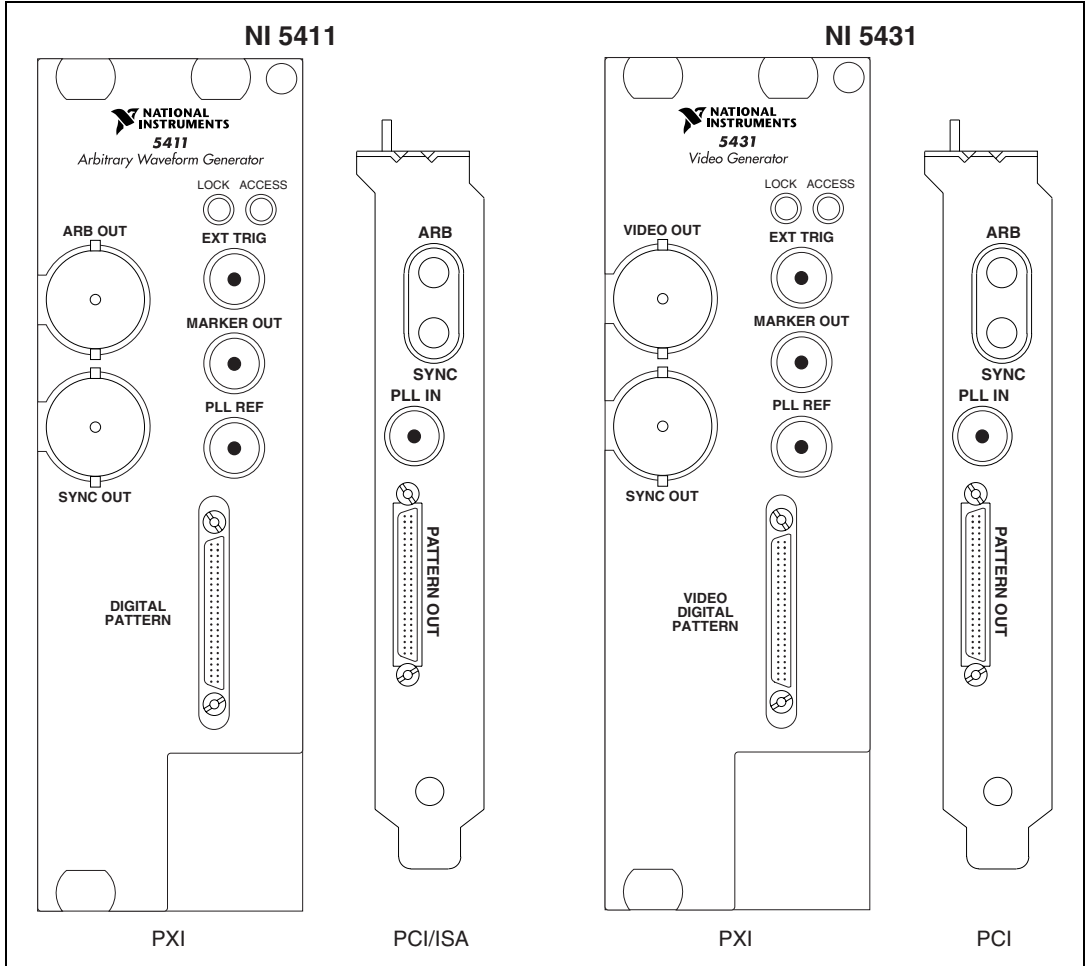


Figure 3-1. NI 5411/5431 I/O Connectors

Adjusting Your NI 54XX—Procedural Overview

To adjust your NI 54XX device, complete the following steps. You may also want to see the *External Calibration* examples in LabVIEW or C, which you can find in Table 1-3.

1. Call `ni54xx_CalStart` to open the session using the following parameters:
 - **device**—The device number of the device you want to calibrate, which is assigned by MAX.
 - **password**—A four-character password. The default password is 'ARBI' if you have not already changed the password.
 - **sessionHandle**—A pointer to an integer that thereafter functions as a handle for the calibration session
2. Reset the device to the default state by calling the following functions:
 - a. Call `ni54xx_DeviceReset` using the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - b. Call `ni54xx_SetAttenuation` to disable all the attenuators using the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **attenuation**—0
 - c. Call `ni54xx_SetDigitalFilter` to enable the digital filter using the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **filterState**—`NI54XX_ENABLE`
 - d. Call `ni54xx_SetAnalogFilter` to disable the analog filter using the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **filterState**—`NI54XX_DISABLE`

- e. Call `ni54xx_SetOutputImpedance` to set the output impedance to 50 ohms using the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **impedance**—`NI54XX_50_OHMS`
- f. Call `ni54xx_SetArbOutput` to enable the device output using the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **outputState**—`NI54XX_ENABLE`
3. Go to the appropriate section below to guide you through calibrating the desired constant.
4. Repeat steps two and three for each set of constants you wish to calibrate.
5. Call `ni54xx_CalEnd` to close the session using the following parameters:
 - **sessionHandle**—A pointer to the handle of the calibration session for the device
 - **action**—`NI54XX_ABORT` or `NI54XX_COMMIT_CONSTANTS`

You may either commit the constants you have calculated or abort the whole session. Aborting saves no changes to the device.

Finding the Low Frequency Characteristics of the Analog Channel

To find the low-frequency characteristics of the analog channel, calibrate for the constants A0–A8, A9 (terminated) and A9 (unterminated). You should calculate constants A0 through A8 in the order the sections appear below. Take all measurements for A0 through A8 unterminated (across a high-impedance load). To find the low frequency characteristics of the analog channel, complete the numbered steps in each section below in the order the sections appear for constants A0 to A9.

Determining Output Dependence On the Gain of the Offset Calibration DAC (Constant A1)

To determine the output dependence, measure output while sweeping the full scale of values for the offset calibration DAC, with the gain calibration DAC and the main DAC set to 0. To sweep the full range of the offset calibration DAC, write nine values to the device: (–8192, –6144, –4096, –2048, 0, 2048, 4096, 6144, and 8191).

1. Set the gain calibration DAC to zero by calling the function `ni54xx_SetGainDAC`. Use the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **DACValue**—0
2. Generate a 0 V DC waveform by calling the function `ni54xx_GenerateWaveform`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **waveType**—`NI54XX_DC`
 - **amplitudeInVolts**—0
 - **frequencyInHz**—0
3. For each of the following values for x , where $x = (-8192, -6144, -4096, -2048, 0, 2048, 4096, 6144, 8191)$, complete the following steps:
 - a. Set the offset calibration DAC to x by calling the function `ni54xx_SetOffsetDAC`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **DACValue**— x
 - b. Measure the outputs (`measurements[]`).
4. Call `ni54xx_CalAdjust` using the output measurements from step 3b as the `measuredData` and the array of DAC settings as the `actualData` of DAC settings using the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **measurementMode**—`NI54XX_SET_A1`
 - **actualData**—`offsetSweepDACValues`
 - **measuredData**—`measurements`

You have completed measuring the outputs for the constant A1.

Determining the Output Dependence On the Gain of the Gain Calibration DAC (Constant A2)

To determine the output dependence, measure output while sweeping the full scale of values for the gain calibration DAC, with the offset calibration DAC and the main DAC set to 0. Measure the outputs for each setting of the gain DAC.

1. Set the offset calibration DAC to zero by calling the function `ni54xx_SetOffsetDAC`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **DACValue**—0
2. Generate a 0 V DC waveform by calling the function `ni54xx_GenerateWaveform`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **waveType**—`NI54XX_DC`
 - **amplitudeInVolts**—0
 - **frequencyInHz**—0
3. For each of the following values for x , where $x = (-8192, -6144, -4096, -2048, 0, 2048, 4096, 6144, 8191)$, complete the following steps:
 - a. Set the gain calibration DAC to x by calling the function `ni54xx_SetGainDAC`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **DACValue**— x
 - b. Measure the outputs (`measurements []`).
4. Call `ni54xx_CalAdjust` using the output measurements from step 3b as the `measuredData` and the array of DAC settings as the `actualData` of DAC settings using the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **measurementMode**—`NI54XX_SET_A2`
 - **actualData**—`offsetSweepDACValues`
 - **measuredData**—`measurements`

You have completed measuring the outputs for the constant A2.

Determining the Interactions of Gains and Offsets in the Main DAC and Gain Calibration DAC (Constants A3 and A4)

To calculate constants, sweep the full range of the 12-bit main DAC (–2048 to 2047) by calling `ni54xx_GenerateWaveform` with the amplitude parameter set to correspond to the DC value of each DAC setting. Table 3-2 shows the corresponding input values and expected output voltage of the main DAC.

Table 3-2. Input Values and Expected Output Values

DAC Value	Output (V)
–2048	–10
–1536	–7.5
–1024	–5
–512	–2.5
0	0
512	2.5
1024	5
1536	7.5
2047	10

- Set the offset calibration DAC to 0 by calling the function `ni54xx_SetOffsetDAC`. Set the following parameters:
 - sessionHandle**—The handle of the calibration session for the device
 - DACValue**—0
- Set the gain calibration DAC to 0 by calling the function `ni54xx_SetGainDAC`. Set the following parameters:
 - sessionHandle**—The handle of the calibration session for the device
 - DACValue**—0

3. For $x = (-10, -7.5, -5, -2.5, 0, 2.5, 5, 7.5, 10)$, generate a DC output of amplitude x :
 - a. Call the function `ni54xx_GenerateWaveform` using the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **waveType**—`NI54XX_DC`
 - **amplitudeInVolts**— x
 - **frequencyInHz**—`0.0`
 - b. Measure the outputs (`mainDACSweepResults[]`).
4. Feed the measurements to `ni54xx_CalAdjust`. The `measuredData` are the new array of measurements, and the `actualData` are the DAC values corresponding to outputs (see Table 3-2). Call the function `ni54xx_CalAdjust` using the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **measurementMode**—`NI54XX_SET_A3`
 - **actualData**—`mainDACSweepValues`
 - **measuredData**—`mainDACSweepResults`
5. Set the gain calibration DAC to -8192 by calling the function `ni54xx_SetGainDAC`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **DACValue**— -8192
6. For $x = (-10, -7.5, -5, -2.5, 0, 2.5, 5, 7.5, 10)$
 - a. Generate a DC output of amplitude x by calling the function `ni54xx_GenerateWaveform`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **waveType**—`NI54XX_DC`
 - **amplitudeInVolts**— x
 - **frequencyInHz**—`0.0`
 - b. Measure the outputs (`mainDACSweepResults[]`).

7. Feed the measurements to `ni54xx_CalAdjust`. The `measuredData` are the new array of measurements, and the `actualData` are the DAC values corresponding to outputs (see Table 3-2). Call the function `ni54xx_CalAdjust` using the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **measurementMode**—`NI54XX_SET_A4`
 - **actualData**—`mainDACSweepValues`
 - **measuredData**—`mainDACSweepResults`

Determining the Low Frequency Gain of the Analog Filter (Constant A7)

Generate a low frequency sine wave output with the analog filter disabled and measure the outputs. Repeat the process with the analog filter enabled, and pass both measurement results to `ni54xx_CalAdjust`. You can measure the amplitude, peak-to-peak amplitude, or RMS value, as long as both measurements are the same.

1. Set the offset calibration DAC to 0 by calling the function `ni54xx_SetOffsetDAC`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **DACValue**—0
2. Set the gain calibration DAC to -4079 (the approximate value for a gain of 1) by calling the function `ni54xx_SetGainDAC`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **DACValue**— -4079
3. Generate a full scale sine wave at 500 Hz by calling the function `ni54xx_GenerateWaveform`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **waveType**—`NI54XX_SINE`
 - **amplitudeInVolts**—10
 - **frequencyInHz**—500
4. Measure the outputs (`lowFrequencyAmplitude`).

5. Enable the analog filter by calling the function `ni54xx_SetAnalogFilter`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **filterState**—`NI5XX_ENABLE`
6. Measure the outputs (`amplitudeWithAnalogFilterEnabled`).
7. Feed the measurements from Steps 4 and 6 to `ni54xx_CalAdjust`. The `measuredData` is a pointer to the value of the output with the filter enabled, and the `actualData` is a pointer to the value of the output with the filter disabled. Call the function `ni54xx_CalAdjust` using the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **measurementMode**—`NI54XX_SET_A7`
 - **actualData**—`&lowFrequencyAmplitude`
 - **measuredData**—`&litudeWithAnalogFilterEnabled`

Determining the Low Frequency Gain of the Pre-Amplification 10db Attenuator (Constant A8)

To determine the value of constant A8, generate a low frequency sine wave output with the attenuator disabled and measure the outputs. Repeat the process with the attenuator enabled, and pass both results to `ni54xx_CalAdjust`. You can measure the amplitude, peak-to-peak amplitude, or RMS value, as long as both measurements are the same.

1. Set the offset calibration DAC to 0 by calling the function `ni54xx_SetOffsetDAC`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **DACValue**—0
2. Set the gain calibration DAC to -4079 (the approximate value for a gain of 1) by calling the function `ni54xx_SetGainDAC`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **DACValue**— -4079

3. Generate a full scale sine wave at 500 Hz by calling the function `ni54xx_GenerateWaveform`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **waveType**—`NI54XX_SINE`
 - **amplitudeInVolts**—10
 - **frequencyInHz**—500
4. Measure the outputs (`lowFrequencyAmplitude`).
5. Enable the 10 dB attenuator by calling the function `ni54xx_SetAttenuation`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **attenuation**—10
6. Measure the outputs (`amplitudeWith10dbAttenuatorEnabled`).
7. Feed the measurements to `ni54xx_CalAdjust`. The `measuredData` is a pointer to the value of the output with the attenuator enabled, and the `actualData` is a pointer to the value of the output with the attenuator disabled.

Call the function `ni54xx_CalAdjust` using the following parameters:

- **sessionHandle**—The handle of the calibration session for the device
- **measurementMode**—`NI54XX_SET_A8`
- **actualData**—`&lowFrequencyAmplitude`
- **measuredData**—`&litudeWith10dbAttenuatorEnabled`

Determining the Output Dependence On Offsets of Several Stages of the Analog Channel (Constants A0, A5, and A6)

Measure the outputs of the device in three different states.

1. Set the offset calibration DAC to 0 by calling the function `ni54xx_SetOffsetDAC`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **DACValue**—0

2. Set the gain calibration DAC to 0 by calling the function `ni54xx_SetGainDAC`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **DACValue**—0
3. Generate a 0 V DC waveform by calling the function `ni54xx_GenerateWaveform`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **waveType**—`NI54XX_DC`
 - **amplitudeInVolts**—0
 - **frequencyInHz**—0
4. Measure the outputs (`offsetInResetState`).
5. Enable the analog filter by calling the function `ni54xx_SetAnalogFilter`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **filterState**—`NI54XX_ENABLE`
6. Measure the outputs (`offsetWithAnalogFilterEnabled`).
7. Disable the analog filter by calling the function `ni54xx_SetAnalogFilter`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **filterState**—`NI54XX_DISABLE`.
8. Enable the 10 dB attenuator by calling the function `ni54xx_SetAttenuation`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **attenuation**—10
9. Measure the outputs (`offsetWith10dbAttenuatorEnabled`).

10. Call `ni54xx_CalAdjust` for A0. The `measuredData` should be an array of values. The first element should be the `offsetInResetState`, and the second element should be the `offsetWithAnalogFilterEnabled`. Call the function `ni54xx_CalAdjust` using the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **measurementMode**—`NI54XX_SET_A0`
 - **actualData**—0 or `NULL`
 - **measuredData**—see the description in step 10 above
11. Call `ni54xx_CalAdjust` for A5. The `measuredData` should be an array of values with the first element being the `offsetInResetState`, and the second element being the `offsetWith10dbAttenuatorEnabled`. Call the function `ni54xx_CalAdjust` using the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **measurementMode**—`NI54XX_SET_A5`
 - **actualData**—0 or `NULL`
 - **measuredData**—see the description in step 11 above
12. Call `ni54xx_CalAdjust` for A6. The `measuredData` should be a pointer to the `offsetInResetState`. Call the function `ni54xx_CalAdjust` using the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **measurementMode**—`NI54XX_SET_A6`
 - **actualData**—0 or `NULL`
 - **measuredData**—`&offsetInResetState`

Determining the Gains of the Six Post-Amplification Attenuators (Constant A9)

These attenuators correspond to 1, 2, 4, 8, 16, and 32 dB of attenuation. A9_UNTERMINATED refers to the array of gains of the attenuators in the unterminated case and A9_TERMINATED refers to the gains of the attenuators in the terminated case. A9TA corresponds to the gain of the analog channel into a terminated load with no attenuation. You generate a low frequency sine wave, enable the attenuators one at a time, and measure output at each step. You can measure the amplitude, peak-to-peak, or RMS, as long as all measurements are consistent.

1. Make sure the output is unterminated.
2. Set the gain calibration DAC to -4079 (the approximate value for a gain of 1) by calling the function `ni54xx_SetGainDAC`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **DACValue**— -4079
3. Set the offset calibration DAC to 0 by calling the function `ni54xx_SetOffsetDAC`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **DACValue**—0
4. Generate a 10 V sine wave at 500 Hz by calling the function `ni54xx_GenerateWaveform`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **waveType**—`NI54XX_SINE`
 - **amplitudeInVolts**—10
 - **frequencyInHz**—500
5. Measure the outputs (`lowFrequencyAmplitude`).

Enable each attenuator individually and measure the output amplitude.

6. For $x = (1, 2, 4, 8, 16, 32)$
 - a. Call the function, `ni54xx_SetAttenuation`, using the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **attenuation**— x
 - b. Measure the outputs (`attenuatedAmplitudes []`).
7. Call `ni54xx_CalAdjust` for `A9_UNTERMINATED`. The `measuredData` is an array of values corresponding to the outputs for each attenuator (from 1 to 32 in ascending order). The `actualData` is a pointer to the un-attenuated output (`lowFrequencyAmplitude`). Call the function `ni54xx_CalAdjust` using the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **measurementMode**—`NI54XX_SET_A9_UNTERMINATED`
 - **actualData**—`&lowFrequencyAmplitude`
 - **measuredData**—`attenuatedAmplitudes`
8. Terminate the output with a $50\ \Omega$ load and measure the remaining A9 outputs across this load.
9. Disable all attenuators by calling the function `ni54xx_SetAttenuation`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **attenuation**—0
10. Measure the outputs (`lowFrequencyAmplitudeTerminated`).

Enable each attenuator individually and measure the output amplitude.

11. For $x = (1, 2, 4, 8, 16, 32)$
 - a. Call the function, `ni54xx_SetAttenuation`, using the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **attenuation**— x
 - b. Measure the outputs (`attenuatedAmplitudesTerminated []`).
12. Call `ni54xx_CalAdjust` for `A9_TERMINATED`. The `measuredData` should be an array of values corresponding to the outputs for each attenuator (from 1 to 32 in ascending order). The

`actualData` should be a pointer to the terminated un-attenuated output (`lowFrequencyAmplitudeTerminated`). Call the function `ni54xx_CalAdjust` using the following parameters:

- **sessionHandle**—The handle of the calibration session for the device
- **measurementMode**—`NI54XX_SET_A9_TERMINATED`
- **actualData**—`&lowFrequencyAmplitudeTerminated`
- **measuredData**—`attenuatedAmplitudesTerminated`

13. Call `ni54xx_CalAdjust` for A9TA. The `measuredData` should be a pointer to the terminated un-attenuated output (`lowFrequencyAmplitudeTerminated`) and the `actualData` should be a pointer to the un-terminated un-attenuated output (`lowFrequencyAmplitude`). Call the function `ni54xx_CalAdjust` using the following parameters:

- **sessionHandle**—The handle of the calibration session for the device
- **measurementMode**—`NI54XX_SET_A9TA`
- **actualData**—`&lowFrequencyAmplitude`
- **measuredData**—`&lowFrequencyAmplitudeTerminated`

Determining the Value of the On-Board High-Precision Voltage Reference

This calibration constant stores the actual value of the on-board high-precision voltage reference. The value of this reference relative to device output can be routed to the on-board analog-to-digital converter (the calibration ADC). If you measure the value of the device output, you can subtract the ADC value from this result to obtain the internal reference. Offsets for the calibration ADC are accounted for internally, so you only need to measure the ADC value and the device output value.

1. Set the offset calibration DAC to 0 by calling the function `ni54xx_SetOffsetDAC`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **DACValue**—0
2. Set the gain calibration DAC to -4079 (the approximate value for a gain of 1) by calling the function `ni54xx_SetGainDAC`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **DACValue**—-4079
3. Generate a 5 V DC waveform by calling the function `ni54xx_GenerateWaveform`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **waveType**—`NI54XX_DC`
 - **amplitudeInVolts**—5
 - **frequencyInHz**—0
4. Set the calibration multiplexer to route the device output and the internal reference to the calibration ADC by calling the function `ni54xx_SetCalibrationMux`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **muxSetting**—`NI54XX_OUT_VS_REF`
5. Read the value of the calibration ADC by calling the function `ni54xx_ReadCalibrationADC`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **calADCValue**—`¤tCalADCValue`
6. Measure the outputs (`currentOutput`).
7. Call `ni54xx_CalAdjust` for the internal reference. The `measuredData` should be a pointer to the calibration ADC reading. The `actualData` should be a pointer to the value of the device output. Call the function `ni54xx_CalAdjust` using the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **measurementMode**—`NI54XX_SET_INTERNAL_REFERENCE`

- **actualData**—¤tOutput
- **measuredData**—¤tCalADCValue

Determining the Frequency Response across the Passband

Calibration constants representing the analog channel's response across a wide range of frequencies are stored to correct for signal attenuation at high frequencies. For both the terminated and unterminated cases, you measure the amplitude of a sinusoid output from 0 to 16 MHz, in 1 MHz increments. These measurements should be stored in an array such that the measurement at 0 MHz is the first element, 1 MHz the second, and so on. The 0 MHz measurement should be taken from a sinusoid at a frequency between 100 and 1000 Hz. You can measure amplitude, peak-to-peak, or RMS, as long as the measurements are consistent.

1. Make sure the output is unterminated.
2. Enable the analog filter by calling the function `ni54xx_SetAnalogFilter`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **filterState**—`NI54XX_ENABLE`
3. Adjust the gain and offset by calling the function `ni54xx_SetAdjustedGainAndOffset` rather than setting the gain and offset calibration DACs directly. This function uses the current calibration constants, including the ones set during the current session, to calculate the values to write to the DACs. The function takes as parameters the desired offset, the amount of attenuation that has been applied to the channel, and whether the output is terminated. Call the function `ni54xx_SetAdjustedGainAndOffset` using the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **offsetInVolts**—0
 - **attenuationInDb**—0
 - **terminationState**—`NI54XX_UNTERMINATED`
4. Generate a low frequency full scale (10 V) sine wave by calling the function `ni54xx_GenerateWaveform`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device

- **waveType**—NI54XX_SINE
 - **amplitudeInVolts**—10
 - **frequencyInHz**—100
5. Measure the outputs, and store as the first element in the frequency responses array (`frequencyResponses[0]`).
 6. For $x = (1000000, 2000000, \dots, 15000000, 16000000)$
 - a. Generate a full scale sine wave at frequency x by calling the function `ni54xx_GenerateWaveform`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **waveType**—NI54XX_SINE
 - **amplitudeInVolts**—10
 - **frequencyInHz**— x
 - b. Measure the outputs and store the result in the `frequencyResponses` array.
 7. Call `ni54xx_CalAdjust` for the unterminated frequency response. The `measuredData` should be the array of output amplitudes. The `actualData` should be NULL. Call the function `ni54xx_CalAdjust` using the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **measurementMode**—NI54XX_SET_FREQUENCY_RESPONSE_UNTERMINATED
 - **actualData**—NULL
 - **measuredData**—`frequencyResponses`
 8. Terminate the output with a 50 Ω load.
 9. Generate a low frequency full scale (10 V) sine wave by calling the function `ni54xx_GenerateWaveform`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **waveType**—NI54XX_SINE
 - **amplitudeInVolts**—10
 - **frequencyInHz**—100
 10. Measure the outputs and store it as the first element in the `frequencyResponses` array (`frequencyResponses[0]`).

11. For $x = (1000000, 2000000, \dots, 15000000, 16000000)$
 - a. Generate a full scale sine wave at frequency x by calling the function `ni54xx_GenerateWaveform`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **waveType**—`NI54XX_SINE`
 - **amplitudeInVolts**—10
 - **frequencyInHz**— x
 - b. Measure the outputs and store the result in the `frequencyResponses` array.

12. Call `ni54xx_CalAdjust` for the terminated frequency response. The `measuredData` should be the array of output amplitudes. The `actualData` should be `NULL`. Call the function `ni54xx_CalAdjust` using the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **measurementMode**—`NI54XX_SET_FREQUENCY_RESPONSE_TERMINATED`
 - **actualData**—`NULL`
 - **measuredData**—`frequencyResponses`

Determining the SYNC Duty Cycle Constants

Two calibration constants, D0 and D1, calculate the value written to the SYNC DAC to set a specified duty cycle on the SYNC output. To determine the proper values, use an iterative method by setting the SYNC DAC to the “best-guess” value (based on the results of previous guesses), then by measuring the resulting duty cycle. In this section, the word “tolerance” refers to your defined allowable range other than the signal received. If the duty cycle is within the tolerance of the desired value, the calibration is complete. If the duty cycle is not within the tolerance, calculate the next “best-guess” value and repeat the process. Calculate D0 first by converging on a duty cycle of 50%, then calculate D1 by converging on duty cycles of 25% and 75%.

D0 Constant

1. Enable the analog filter by calling the function `ni54xx_SetAnalogFilter`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **filterState**—`NI54XX_ENABLE`.
2. Generate a full scale (10 V) sine wave at 1000 Hz by calling the function `ni54xx_GenerateWaveform`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **waveType**—`NI54XX_SINE`
 - **amplitudeInVolts**—10
 - **frequencyInHz**—1000
3. Set the gain and offset using `ni54xx_SetAdjustedGainAndOffset` with 0 offset and 0 attenuation in the unterminated case. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **offsetInVolts**—0
 - **attenuationInDb**—0
 - **terminationState**—`NI54XX_UNTERMINATED`
4. Set a 50% duty cycle by calling the function `ni54xx_SetDutyCycle`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **dutyCycle**—50
5. Measure the duty cycle (`currentDutyCycle`).
6. While the `lmeasured duty cycle - 50` is larger than the tolerance (0.1 or larger as determined by your application):
 - a. Call `ni54xx_CalAdjust` for D0 to calculate the next “best-guess” based on the previous results. The `measuredData` should be a pointer to the measured duty cycle. The `actualData` should be a pointer to the desired duty cycle (in this case, 50). Call the function `ni54xx_CalAdjust` using the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **measurementMode**—`NI54XX_SET_D0`

- **actualData**—&desiredDutyCycle
 - **measuredData**—¤tDutyCycle
- b. Try the new “best-guess” by calling the function `ni54xx_SetDutyCycle`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **dutyCycle**—50
 - c. Measure the new duty cycle (`currentDutyCycle`).

If the measurement is outside your desired tolerance, repeat this loop. If you have exited the loop, D0 must have the correct value, so there is no need to call `ni54xx_CalAdjust` for D0 again.

You have completed adjusting the constant D0.

D1 Constant

1. Set a 25% duty cycle by calling the function `ni54xx_SetDutyCycle`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **dutyCycle**—25
2. Measure the duty cycle (`currentDutyCycle`).
3. While the `lmeasured` duty cycle – 25| is larger than the tolerance:
 - a. Call `ni54xx_CalAdjust` for D1_25. This calculates the next “best-guess” based on the previous results. The `measuredData` should be a pointer to the measured duty cycle. The `actualData` should be a pointer to the desired duty cycle (in this case, 25). Call `ni54xx_CalAdjust` using the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **measurementMode**—NI54XX_SET_D1_25
 - **actualData**—&desiredDutyCycle
 - **measuredData**—¤tDutyCycle

- b. Try the new “best-guess” by calling the function `ni54xx_SetDutyCycle`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **dutyCycle**—25
 - c. Measure the new duty cycle (`currentDutyCycle`).
 4. Set a 75% duty cycle by calling the function `ni54xx_SetDutyCycle`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **dutyCycle**—75
 5. Measure the duty cycle (`currentDutyCycle`).
 6. While the $|\text{measured duty cycle} - 75|$ is larger than the tolerance:
 - a. Call `ni54xx_CalAdjust` for D1_75. This calculates the next “best-guess” based on the previous results. The `measuredData` should be a pointer to the measured duty cycle. The `actualData` should be a pointer to the desired duty cycle (in this case, 75). Call the function `ni54xx_CalAdjust` using the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **measurementMode**—`NI54XX_SET_D1_75`
 - **actualData**—`&desiredDutyCycle`
 - **measuredData**—`¤tDutyCycle`
 - b. Try the new “best-guess” by calling the function `ni54xx_SetDutyCycle`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **dutyCycle**—75
 - c. Measure the new duty cycle (`currentDutyCycle`).
7. After you have determined D1_25 and D1_75, you must call `ni54xx_CalAdjust` one last time. The measurement mode parameter should be `NI54XX_SET_D1` and both the `actualData` and `measuredData` should be `NULL`. Call the function `ni54xx_CalAdjust` using the following parameters:

- **sessionHandle**—The handle of the calibration session for the device
- **measurementMode**—NI54XX_SET_D1
- **actualData**—NULL
- **measuredData**—NULL

Determining the VCXO Calibration Constants

This stage of calibration determines the correct value to write to the VCXO DAC, which controls the device oscillator, to achieve a desired update rate. For all devices, you must find a value that results in an update clock of about 40 MHz. For the NI 5431, you must find two additional values. These values correspond to the update rates needed for the M-PAL and NTSC video modes. Like the duty cycle calibration, VCXO calibration is an iterative process. First, initialize the VCXO calibration for the current generation mode (normal, M-PAL, or NTSC). Then, call the function `ni54xx_CalibrateVCXO`. This function, like `ni54xx_SetDutyCycle`, sets the VCXO with the current “best-guess.” Next, measure the output frequency. If the measured frequency differs substantially from the desired frequency, call `ni54xx_CalAdjust`, which calculates the next “best-guess” value. Call `ni54xx_CalibrateVCXO` again, setting the VCXO to the new “best-guess” and measure the frequency again. Repeat this process until the output frequency is acceptable or until you have called `ni54xx_CalAdjust` 12 times. Iterating more than twelve times produces no further refinements.

For all devices:

1. Enable the analog filter by calling the function `ni54xx_SetAnalogFilter`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **filterState**—NI54XX_ENABLE
2. Generate a full scale (10 V) sine wave at 1 MHz by calling the function `ni54xx_GenerateWaveform`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **waveType**—NI54XX_SINE
 - **amplitudeInVolts**—10
 - **frequencyInHz**—1000000

3. Set the gain and offset using `ni54xx_SetAdjustedGainAndOffset` with 0 offset and 0 attenuation, in the unterminated case. Use the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **offsetInVolts**—0
 - **attenuationInDb**—0
 - **terminationState**—`NI54XX_UNTERMINATED`
4. Initialize the VCXO calibration for normal mode by calling the function `ni54xx_InitializeVCXOCalibration`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **type**—`NI54XX_SET_VCXO`
5. Set the VCXO with the current “best-guess” value (normal mode) by calling the function `ni54xx_CalibrateVCXO`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **type**—`NI54XX_SET_VCXO`
6. Use the counter to measure the frequency of the signal on SYNC OUT (`currentFrequency`).
7. While the $|f_{\text{frequency}} - 1 \text{ MHz}|$ is greater than the tolerance *and* while you have not entered this loop more than 12 times, follow these steps:



Note While defining your tolerance, leave adequate margins to accommodate for drift frequency due to temperature.

- a. Call `ni54xx_CalAdjust` for measurement mode `NI54XX_SET_VCXO` which calculates the next “best-guess” VCXO DAC value. The `measuredData` should be a pointer to the measured frequency and the `actualData` should be a pointer to the desired frequency (1 MHz). Call the function `ni54XX_CalAdjust` using the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **measurementMode**—`NI54XX_SET_VCXO`

- **actualData**—&desiredFrequency
 - **measuredData**—¤tFrequency
- b. Set the VCXO with the new “best-guess” value (normal mode) by calling the function `ni54xx_CalibrateVCXO`. Set the following parameters:
- **sessionHandle**—The handle of the calibration session for the device
 - **type**—`NI54XX_SET_VCXO`
- c. Measure the new frequency (`currentFrequency`).

If you have exited the loop, the VCXO is adjusted within the tolerance or as close as possible.

You have completed adjusting the VCXO.

Calibrating the M-PAL Video Mode for NI 5431

To calibrate the M-PAL video mode for NI 5431 devices only, complete the following steps:

1. Enable the analog filter by calling the function `ni54xx_SetAnalogFilter`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **filterState**—`NI5XX_ENABLE`
2. Generate a full scale (10 V) sine wave at 1 MHz in M-PAL video mode by calling `ni54xx_GenerateVideoWaveform`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **waveType**—`NI54XX_SINE`
 - **amplitudeInVolts**—10
 - **frequencyInHz**—1000000
 - **videoMode**—`NI54XX_PAL_M`
3. Set the gain and offset by calling the function `ni54xx_SetAdjustedGainAndOffset` with 0 offset and 0 attenuation, in the unterminated case. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **offsetInVolts**—0

- **attenuationInDb**—0
 - **terminationState**—NI54XX_UNTERMINATED
4. Initialize the VCXO calibration for M-PAL mode by calling the function `ni54xx_InitializeVCXOCalibration`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **type**—NI54XX_SET_VCXO_PAL_M
 5. Set the VCXO with the current “best-guess” value (M-PAL mode) by calling the function `ni54xx_CalibrateVCXO`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **type**—NI54XX_SET_VCXO_PAL_M
 6. Measure the new frequency (`currentFrequency`).
 7. While the `lfrequency - 1 MHz` is greater than the tolerance *and* you have not entered this loop more than 12 times, follow these steps:
 - a. Call `ni54xx_CalAdjust` for measurement mode NI54XX_SET_VCXO_PAL_M, which calculates the next “best-guess” VCXO DAC value. The `measuredData` should be a pointer to the measured frequency and the `actualData` should be a pointer to the desired frequency (1 MHz). Call the function `ni54xx_CalAdjust` using the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **measurementMode**—NI54XX_SET_VCXO_PAL_M
 - **actualData**—`&desiredFrequency`
 - **measuredData**—`¤tFrequency`
 - b. Set the VCXO with the new “best-guess” value (M-PAL mode) by calling the function `ni54xx_CalibrateVCXO`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **type**—NI54XX_SET_VCXO_PAL_M
 - c. Measure the new frequency (`currentFrequency`).

If you have exited the loop, the VCXO is adjusted within the tolerance or as close as possible.

You have completed adjusting the M-PAL Video Mode.

Calibrating the NTSC Video Mode for NI 5431

To calibrate the NTSC video mode for NI 5431 devices only, complete the following steps:

1. Enable the analog filter by calling the function `ni54xx_SetAnalogFilter`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **filterState**—`NI5XX_ENABLE`
2. Generate a full scale (10 V) sine wave at 1 MHz in NTSC video mode by calling the function `ni54xx_GenerateVideoWaveform`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **waveType**—`NI54XX_SINE`
 - **amplitudeInVolts**—10
 - **frequencyInHz**—1000000
 - **videoMode**—`NI54XX_NTSC_M`
3. Set the gain and offset by calling the function `ni54xx_SetAdjustedGainAndOffset` with 0 offset and 0 attenuation, in the unterminated case using the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **offsetInVolts**—0
 - **attenuationInDb**—0
 - **terminationState**—`NI54XX_UNTERMINATED`
4. Initialize the VCXO calibration for NTSC mode by calling the function `ni54xx_InitializeVCXOCalibration`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **type**—`NI54XX_SET_VCXO_NTSC_M`

5. Set the VCXO with the current “best-guess” value (NTSC mode) by calling the function `ni54xx_CalibrateVCXO`. Set the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **type**—`NI54XX_SET_VCXO_NTSC_M`
6. Measure the new frequency (`currentFrequency`).
7. While the `lfrequency - 1 MHz` is greater than the tolerance *and* you have not entered this loop more than 12 times, follow these steps:
 - a. Call `ni54xx_CalAdjust` for measurement mode `NI54XX_SET_VCXO_NTSC_M`, which calculates the next “best-guess” VCXO DAC value. The `measuredData` should be a pointer to the measured frequency and the `actualData` should be a pointer to the desired frequency (1 MHz). Call the function `ni54xx_CalAdjust` using the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **measurementMode**—`NI54XX_SET_VCXO_NTSC_M`
 - **actualData**—`&desiredFrequency`
 - **measuredData**—`¤tFrequency`
 - b. Set the VCXO with the new “best-guess” value (NTSC mode). Call the function `ni54xx_CalibrateVCXO` and use the following parameters:
 - **sessionHandle**—The handle of the calibration session for the device
 - **type**—`NI54XX_SET_VCXO_NTSC_M`
 - c. Measure the new frequency (`currentFrequency`).

If you have exited the loop, the VCXO is adjusted within the tolerance or as close as possible.

You have completed adjusting the NTSC Video Mode.

Overview of the Calibration Constants

This appendix provides an overview of the constants that you adjusting for your device: low frequency characteristics of the analog channel, internal reference, frequency response, duty cycle, and VCXO constants.

Low Frequency Characteristics of the Analog Channel

These constants represent the DC or low-frequency gains and offsets of components within the analog channel. They are referred to as constants A0 through A9 in the documentation and in the programming interface. Unterminated refers to a high impedance load on the output of the device and terminated refers to a 50 Ω load. Constants A0 through A8 can be calculated during an internal calibration. Constants A9_Unterminated, A9_Terminated, and A9TA are calculated only during an external calibration. Table A-1 shows the names of these constants and a brief description of each.

Table A-1. DC or Low-Frequency Gains and Offsets of Components within the Analog Channel

Constant Name	Description
A0	Output dependence on the cumulative offsets of the gain calibration DAC and main DAC.
A1	Output dependence on the gain of the offset calibration DAC.
A2	Output dependence on the gain of the gain calibration DAC.
A3	Output dependence on the offset of the gain calibration DAC and the gain of the main DAC.
A4	Output dependence on the gains of the gain calibration DAC and the main DAC.
A5	Output dependence on the offset of the offset calibration DAC.
A6	Output dependence on the cumulative offsets of the amplifiers in the analog channel.

Table A-1. DC or Low-Frequency Gains and Offsets of Components within the Analog Channel (Continued)

Constant Name	Description
A7	The low frequency gain of the analog filter.
A8	The gain of the 10 dB attenuator in the analog channel.
A9 Unterminated	An array of constants corresponding to the gains of the six post-amplification attenuators (1, 2, 4, 8, 16, 32 dB) when device is unterminated.
A9 Terminated	An array of constants corresponding to the gains of the six post-amplification attenuators (1, 2, 4, 8, 16, 32 dB) when device is terminated with a 50 Ω load.
A9TA	The gain of the device in the terminated state with no post-amplification attenuators active.

Internal Reference

The NI 54XX series devices have a high-precision voltage reference, which is used during self-calibration. The value of this reference is subject to drift, so its current value (as of the last external calibration) is stored as a calibration constant.

Frequency Response

Several constants are used to store the frequency response of the analog channel to compensate for attenuation at higher frequencies. A different set of constants is held for the unterminated and terminated cases. For each case, seventeen constants are stored, corresponding to the gain of the channel at frequencies from 0 to 16 MHz, inclusive.

Duty Cycle

Duty cycle is the ratio of the duration of the logical high level of a signal to the total period of the signal. The duty cycle of the SYNC TTL output is adjustable from 20% to 80%. To set a desired duty cycle, the driver software must use two calibration constants, called D0 and D1. Calibrating for these constants involves iteratively converging on the proper values to achieve specified duty cycles.

Voltage-Controlled Crystal Oscillator (VCXO) Constants

When external clocking is not being used, the NI 54XX devices use a voltage-controlled crystal oscillator. If the device is not being phase locked to an external signal, the driver must determine what value to write to the VCXO DAC that controls the voltage to the oscillator. Calibrating the VCXO involves iteratively converging on the value that sets the update clock rate as near to 40 MHz as possible. For the NI 5431, two additional constants are stored. These constants represent the proper VCXO settings for generating phase alternation line (M-PAL) and National Television System Committee (NTSC) video signals, which require slightly different update rates.

Function Reference

This appendix provides detailed descriptions of functions that support calibration of signal source performance by listing the calibration functions and constants for NI 54XX devices. This interface provides access to the `niarbcad.dll` and can be used to write calibration procedures for any NI 54XX device.



Note All functions return signed, 32-bit integers corresponding to the status of the operation.

ni54xx_CalStart

Format

```
__declspec(dllexport) long __stdcall
    ni54xx_CalStart( shortdevice,
                    char          *password,
                    unsigned long  *sessionHandle);
```

Purpose

Initiates a calibration session for the specified device.

Parameters

Name	Description
device	A number corresponding to the DAQ device number of the device to be calibrated.
password	The external adjustment password. This must point to a buffer of characters at least four characters long.
sessionHandle	A pointer to the handle of the calibration session used in most other calibration functions.

ni54xx_CalAdjust

Format

```
__declspec(dllexport) long __stdcall
ni54xx_CalAdjust(unsigned long sessionHandle,
                 unsigned long measurementMode,
                 double *actualData,
                 double *measuredData );
```

Purpose

Adjust the specified calibration constant based on the values of the measured and actual data.

Parameters

Name	Description
sessionHandle	The handle of the calibration session for the device, created with a call to ni54xx_CalStart.
measurementMode	The calibration constant(s) being adjusted. One of the NI54XX_SET_ constants in Appendix C, <i>Function Constants</i> .
actualData	An array of doubles, a pointer to one double, or NULL, depending on the measurementMode parameter.
measuredData	An array of doubles, a pointer to one double, or NULL, depending on the measurementMode parameter.

ni54xx_CalEnd

Format

```
__declspec(dllexport) long __stdcall
    ni54xx_CalEnd(unsigned long    *sessionHandle,
                  unsigned long    action);
```

Purpose

Closes the calibration session and either commits or discards the new constants that have been calculated.

Parameters

Name	Description
sessionHandle	A pointer to the handle of the calibration session for the device, created with a call to ni54xx_CalStart.
action	NI54XX_ABORT or NI54XX_COMMIT_CONSTANTS. Using abort avoids storing the newly calculated calibration constants.

ni54xx_CalSelfCalibrate

Format

```
__declspec(dllexport) long __stdcall
ni54xx_CalSelfCalibrate(unsigned long device);
```

Purpose

Internally calibrates the device and stores the new calibration constants. The self calibration can only determine constants A0 through A8. CalStart and CalEnd do not need to be called with this function.

Parameter

Name	Description
device	A number corresponding to the DAQ device number of the device to be calibrated.

ni54xx_CalFetchDate

Format

```

__declspec(dllexport) long __stdcall
ni54xx_CalFetchDate(unsigned long device,
                    long area,
                    long *month,
                    long *day,
                    long *year);

```

Purpose

Retrieves the date of the last calibration, either from the internal (self-calibration) or external (manual calibration) area.

Parameters

Name	Description
device	A number corresponding to the DAQ device number of the device to be calibrated.
area	The area (internal or external) from where to get the date NI54XX_INTERNAL or NI54XX_EXTERNAL.
month	A pointer to a long integer which receives the month value.
day	A pointer to a long integer which receives the date value.
year	A pointer to a long integer which receives the year value.

ni54xx_CalFetchCount

Format

```
__declspec(dllexport) long __stdcall
    ni54xx_CalFetchCount(unsigned long device,
                        long area,
                        long *count);
```

Purpose

Retrieves the number of times the device has been calibrated, either internally (self-calibration) or externally (manual calibration).

Parameters

Name	Description
device	A number corresponding to the DAQ device number of the device to be calibrated.
area	The area (internal or external) from where to get the date NI54XX_INTERNAL or NI54XX_EXTERNAL.
count	A pointer to a long integer which receives the count value.

ni54xx_CalFetchMiscInfo

Format

```
__declspec(dllexport) long __stdcall
ni54xx_CalFetchMiscInfo(unsigned long device,
                        char *miscInfo);
```

Purpose

Retrieves four bytes of data from the external adjustment miscellaneous data area.

Parameters

Name	Description
device	A number corresponding to the DAQ device number of the device to be calibrated.
miscInfo	This should be a char pointer which points to a buffer which is at least 4 bytes long. This buffer should be allocated by you.

ni54xx_CalStoreMiscInfo

Format

```
__declspec(dllexport) long __stdcall
    ni54xx_CalStoreMiscInfo(unsigned long device,
                           char *miscInfo);
```

Purpose

Stores four bytes of data in the external adjustment miscellaneous data area.

Parameters

Name	Description
device	A number corresponding to the DAQ device number of the device to be calibrated.
miscInfo	This should be a char pointer which points to a buffer which is at least 4 bytes long. This buffer should be allocated by you.

ni54xx_CalChangePassword

Format

```
__declspec(dllexport) long __stdcall
    ni54xx_CalChangePassword(unsigned long    device,
                              char           *oldPassword,
                              char           *newPassword);
```

Purpose

Changes the password, which is needed to do a full external adjustment.

Parameters

Name	Description
device	A number corresponding to the DAQ device number of the device to be calibrated.
oldPassword	The currently existing password.
newPassword	The new password, which replaces the current password.

ni54xx_CalRestoreExternalConstants

Format

```
__declspec(dllexport) long __stdcall
ni54xx_CalRestoreExternalConstants(unsigned long device);
```

Purpose

Copies the constants from the last external adjustment into the internal adjustment area, thereby restoring the state of the device to that of the last external adjustment. Use if an internal adjustment goes awry or produces unacceptable results.

Parameter

Name	Description
device	A number corresponding to the DAQ device number of the device to be calibrated.

ni54xx_DeviceReset

Format

```
__declspec(dllexport) long __stdcall  
ni54xx_DeviceReset(unsigned long sessionHandle );
```

Purpose

Resets the NI 54XX device and puts it in a default reset state.

Parameter

Name	Description
<code>sessionHandle</code>	The handle of the calibration session for the device, created with a call to <code>ni54xx_CalStart</code> .

ni54xx_SetArbOutput

Format

```
__declspec(dllexport) long __stdcall
    ni54xx_SetArbOutput(unsigned long    sessionHandle,
                       unsigned long    outputState );
```

Purpose

Enables or disables the output of the device.

Parameters

Name	Description
sessionHandle	The handle of the calibration session for the device, created with a call to ni54xx_CalStart.
outputState	Indicates whether to enable or disable the output NI54XX_ENABLE or NI54XX_DISABLE.

ni54xx_SetAnalogFilter

Format

```
__declspec(dllexport) long __stdcall
ni54xx_SetAnalogFilter(unsigned long sessionHandle,
                      unsigned long filterState );
```

Purpose

Enables or disables the analog filter of the device.

Parameters

Name	Description
sessionHandle	The handle of the calibration session for the device, created with a call to ni54xx_CalStart.
filterState	Indicates whether to enable or disable the filter NI54XX_ENABLE or NI54XX_DISABLE.

ni54xx_SetDigitalFilter

Format

```
__declspec(dllexport) long __stdcall
    ni54xx_SetDigitalFilter(unsigned long  sessionHandle,
                          unsigned long  filterState );
```

Purpose

Enables or disables the digital filter of the device.

Parameters

Name	Description
sessionHandle	The handle of the calibration session for the device, created with a call to ni54xx_CalStart.
filterState	Indicates whether to enable or disable the filter NI54XX_ENABLE or NI54XX_DISABLE. This normally should be enabled.

ni54xx_SetOutputImpedance

Format

```
__declspec(dllexport) long __stdcall
    ni54xx_SetOutputImpedance(unsigned long    sessionHandle,
                              unsigned long    impedance );
```

Purpose

Sets the output impedance of the device (50 or 75 ohms). During calibration, the impedance should never be set to 75 ohms because the terminated case assumes a 50 ohm load. This requirement is true also for the NI 5431 devices.

Parameters

Name	Description
sessionHandle	The handle of the calibration session for the device, created with a call to ni54xx_CalStart.
impedance	The output impedance value to set NI54XX_50_OHMS or NI54XX_75_OHMS.

ni54xx_GenerateWaveform

Format

```
__declspec(dllexport) long __stdcall
    ni54xx_GenerateWaveform(unsigned long sessionHandle,
                           unsigned long waveType,
                           double amplitudeInVolts,
                           double frequencyInHz );
```

Purpose

Causes the device to output a DC voltage or a sine wave with a specified amplitude and frequency.

Parameters

Name	Description
sessionHandle	The handle of the calibration session for the device, created with a call to ni54xx_CalStart.
waveType	The type of waveform to generate NI54XX_DC or NI54XX_SINE.
amplitudeInVolts	The amplitude of the waveform (in volts), this is the amplitude into an unterminated (high impedance) load.
frequencyInHz	The frequency of the waveform (in hertz) ignored for the DC case, of course.

ni54xx_GenerateVideoWaveform

Format

```
__declspec(dllexport) long __stdcall
    ni54xx_GenerateVideoWaveform(unsigned long    sessionHandle,
                                unsigned long    waveType,
                                double           amplitudeInVolts,
                                double           frequencyInHz,
                                unsigned long    videoMode);
```

Purpose

Causes the device to output a DC voltage or a sine wave with a specified amplitude and frequency. Use this function only when creating a PAL or NTSC video signal for calibrating the VCXO of the NI 5431.

Parameters

Name	Description
sessionHandle	The handle of the calibration session for the device, created with a call to <code>ni54xx_CalStart</code> .
waveType	The type of waveform to generate <code>NI54XX_DC</code> or <code>NI54XX_SINE</code> .
amplitudeInVolts	The amplitude of the waveform (in volts) this is the amplitude into an unterminated (high impedance) load.
frequencyInHz	The frequency of the waveform (in hertz) ignored for the DC case.
videoMode	The video mode (different clocking schemes are used) <code>NI54XX_PAL_M</code> or <code>NI54XX_NTSC_M</code> .

ni54xx_SetDutyCycle

Format

```
__declspec(dllexport) long __stdcall
    ni54xx_SetDutyCycle( unsigned long  sessionHandle,
                        double          dutyCycle );
```

Purpose

Sets the duty cycle of the SYNC output of the device.



Note This function works when the device is generating a sine wave at full scale (10 V into an unterminated load).

Parameters

Name	Description
sessionHandle	The handle of the calibration session for the device, created with a call to ni54xx_CalStart.
dutyCycle	The duty cycle of the SYNC output, in percentage (for example, for 50% duty cycle, dutyCycle = 50.0).

ni54xx_SetAdjustedGainAndOffset

Format

```
__declspec(dllexport) long __stdcall
    ni54xx_SetAdjustedGainAndOffset(unsigned long    sessionHandle,
                                     double          offsetInVolts,
                                     unsigned long    attenuationInDb,
                                     unsigned long    terminatedState );
```

Purpose

Configures the device to output with a certain level of attenuation and a certain offset. It uses the calibration constants for the session (including those that have been changed during the session) to calculate the appropriate settings for the calibration DACs. The user must also provide the termination state, whether the device is unterminated (high-impedance load) or terminated (50-ohm load).

Parameters

Name	Description
sessionHandle	The handle of the calibration session for the device, created with a call to <code>ni54xx_CalStart</code> .
offsetInVolts	The DC offset of the waveform can be from -2.5 to 2.5 volts.
attenuationInDb	The amount of post-amplifier attenuation applied to the signal from 0 to 73 dB in 1dB increments.
terminationState	NI54XX_UNTERMINATED (high-impedance load) NI54XX_TERMINATED (50-ohm load).

ni54xx_SetGainDAC

Format

```
__declspec(dllexport) long __stdcall
    ni54xx_SetGainDAC(unsigned long sessionHandle,
                      long DACValue );
```

Purpose

Sets the gain calibration DAC to the specified value, which affects the pre-attenuation gain of the signal.

Parameters

Name	Description
sessionHandle	The handle of the calibration session for the device, created with a call to ni54xx_CalStart.
DACValue	The integer value to write to the gain DAC ranges from -8192 to 8191.

ni54xx_SetOffsetDAC

Format

```
__declspec(dllexport) long __stdcall
    ni54xx_SetOffsetDAC(unsigned long    sessionHandle,
                        long            DACValue );
```

Purpose

Sets the offset calibration DAC to the specified value, which affects the pre-attenuation offset of the signal.

Parameters

Name	Description
sessionHandle	The handle of the calibration session for the device, created with a call to ni54xx_CalStart.
DACValue	The integer value to write to the gain DAC, which ranges from -8192 to 8191.

ni54xx_SetAttenuation

Format

```
__declspec(dllexport) long __stdcall
    ni54xx_SetAttenuation(unsigned long    sessionHandle,
                        unsigned long    attenuation );
```

Purpose

Sets the attenuation of the signal in dB.

Parameters

Name	Description
sessionHandle	The handle of the calibration session for the device, created with a call to ni54xx_CalStart.
attenuation	Amount of attenuation to apply to the signal in dB: (0, 1, 2, 4, 8, 10, 16, 32).

ni54xx_SetCalibrationMux

Format

```
__declspec(dllexport) long __stdcall
ni54xx_SetCalibrationMux(unsigned long sessionHandle,
                        unsigned long muxSetting );
```

Purpose

Determines what value is read by a call to `ReadCalibrationADC`, the ground vs. the ground, the reference vs. the reference, the output vs. the ground, or the output vs. the reference. You should only set to output vs. reference.

Parameters

Name	Description
<code>sessionHandle</code>	The handle of the calibration session for the device, created with a call to <code>ni54xx_CalStart</code> .
<code>muxSetting</code>	The multiplexer setting <code>NI54XX_GND_VS_GND</code> , <code>NI54XX_REF_VS_REF</code> , <code>NI54XX_OUT_VS_GND</code> , or <code>NI54XX_OUT_VS_REF</code> .

ni54xx_ReadCalibrationADC

Format

```
__declspec(dllexport) long __stdcall
    ni54xx_ReadCalibrationADC(unsigned long    sessionHandle,
                              double          *calADCValue );
```

Purpose

Reads the value of the calibration analog to digital converter, which holds the difference of the two inputs to the calibration multiplexer. For example, the output can be compared to the internal reference of the device. This function returns the average of many readings of the ADC.

Parameters

Name	Description
sessionHandle	The handle of the calibration session for the device, created with a call to ni54xx_CalStart.
calADCValue	A pointer to the value returned by the function.

ni54xx_InitializeVCXOCalibration

Format

```
__declspec(dllexport) long __stdcall
    ni54xx_InitializeVCXOCalibration(unsigned long sessionHandle,
                                     unsigned long type);
```

Purpose

The VCXO is calibrated iteratively. Call this function before calibrating the VCXO because it sets many internal variables to appropriate values for the calibration.

Parameters

Name	Description
sessionHandle	The handle of the calibration session for the device, created with a call to ni54xx_CalStart.
type	The type of VCXO calibration to be done NI54XX_SET_VCXO or NI54XX_SET_PAL_M (for the NI 5431 ONLY) or NI54XX_SET_NTSC_M (for the NI 5431 ONLY).

ni54xx_CalibrateVCXO

Format

```
__declspec(dllexport) long __stdcall
    ni54xx_CalibrateVCXO(unsigned long  sessionHandle,
                        unsigned long  type);
```

Purpose

Recalibrates the VCXO with the current test value. This function is used during calibration to iteratively converge on the best value. A new value is calculated, then this function is called to set the VCXO, then the output is measured, and then a new value is calculated, and so on.

Parameters

Name	Description
sessionHandle	The handle of the calibration session for the device, created with a call to <code>ni54xx_CalStart</code> .
type	The type of VCXO calibration to be done <code>NI54XX_SET_VCXO</code> or <code>NI54XX_SET_PAL_M</code> (for the NI 5431 ONLY) or <code>NI54XX_SET_NTSC_M</code> (for the NI 5431 ONLY)



Function Constants

This appendix lists the constants which are passed and returned by calibration functions.

Status Constants

NI54XX_STATUS_SUCCESS
NI54XX_STATUS_FUNCTION_NOT_SUPPORTED
NI54XX_STATUS_DEVICE_NOT_FOUND
NI54XX_STATUS_DEVICE_NOT_SUPPORTED
NI54XX_STATUS_BAD_HANDLE
NI54XX_STATUS_INCORRECT_PASSWORD
NI54XX_STATUS_INVALID_ACTION
NI54XX_STATUS_INVALID_WAVE_TYPE
NI54XX_STATUS_INVALID_AMPLITUDE
NI54XX_STATUS_INVALID_FREQUENCY
NI54XX_STATUS_INVALID_VIDEO_MODE
NI54XX_STATUS_INVALID_DUTY_CYCLE
NI54XX_STATUS_INVALID_OFFSET
NI54XX_STATUS_INVALID_ATTENUATION
NI54XX_STATUS_INVALID_TERMINATION_STATE
NI54XX_STATUS_INVALID_DAC_VALUE
NI54XX_STATUS_INVALID_FILTER_STATE
NI54XX_STATUS_INVALID_OUTPUT_IMPEDANCE
NI54XX_STATUS_INVALID_MUX_SETTING
NI54XX_STATUS_INVALID_CALIBRATION_AREA
NI54XX_STATUS_INVALID_OUTPUT_STATE

NI54XX_STATUS_BAD_PARAMETER_1
NI54XX_STATUS_BAD_PARAMETER_2
NI54XX_STATUS_BAD_PARAMETER_3
NI54XX_STATUS_BAD_PARAMETER_4
NI54XX_STATUS_BAD_PARAMETER_5
NI54XX_STATUS_BAD_PARAMETER_6
NI54XX_STATUS_UNKNOWN_ERROR

Constants that are Option Parameters for the ni54xx_CalEnd Function

NI54XX_ABORT
NI54XX_COMMIT_CONSTANTS

Constants Used as Parameters in Functions that Control the Device

NI54XX_50_OHMS
NI54XX_75_OHMS
NI54XX_DISABLE
NI54XX_ENABLE
NI54XX_UNTERMINATED
NI54XX_TERMINATED
NI54XX_DC
NI54XX_SINE
NI54XX_PAL_M
NI54XX_NTSC_M
NI54XX_GND_VS_GND
NI54XX_REF_VS_REF
NI54XX_OUT_VS_GND
NI54XX_OUT_VS_REF
NI54XX_INTERNAL
NI54XX_EXTERNAL

Constants Used as Parameters for measurementMode in the ni54xx_CalAdjust Function

These constants specify which calibration constant is currently being adjusted:

NI54XX_SET_A1
NI54XX_SET_A2
NI54XX_SET_A3
NI54XX_SET_A4
NI54XX_SET_A7
NI54XX_SET_A8
NI54XX_SET_A0
NI54XX_SET_A5
NI54XX_SET_A6
NI54XX_SET_A9_UNTERMINATED
NI54XX_SET_INTERNAL_REFERENCE
NI54XX_SET_A9_TERMINATED
NI54XX_SET_A9TA
NI54XX_SET_FREQUENCY_RESPONSE_UNTERMINATED
NI54XX_SET_FREQUENCY_RESPONSE_TERMINATED
NI54XX_SET_D0
NI54XX_SET_D1_25
NI54XX_SET_D1_75
NI54XX_SET_D1
NI54XX_SET_VCXO
NI54XX_SET_VCXO_PAL_M
NI54XX_SET_VCXO_NTSC_M

Technical Support Resources

Web Support

National Instruments Web support is your first stop for help in solving installation, configuration, and application problems and questions. Online problem-solving and diagnostic resources include frequently asked questions, knowledge bases, product-specific troubleshooting wizards, manuals, drivers, software updates, and more. Web support is available through the Technical Support section of ni.com. You may also visit ni.com/support/calibrat/ for a general overview for calibration procedures of National Instrument devices.

NI Developer Zone

The NI Developer Zone at ni.com/zone is the essential resource for building measurement and automation systems. At the NI Developer Zone, you can easily access the latest example programs, system configurators, tutorials, technical news, as well as a community of developers ready to share their own techniques.

Customer Education

National Instruments provides a number of alternatives to satisfy your training needs, from self-paced tutorials, videos, and interactive CDs to instructor-led hands-on courses at locations around the world. Visit the Customer Education section of ni.com for online course schedules, syllabi, training centers, and class registration.

System Integration

If you have time constraints, limited in-house technical resources, or other dilemmas, you may prefer to employ consulting or system integration services. You can rely on the expertise available through our worldwide network of Alliance Program members. To find out more about our Alliance system integration solutions, visit the System Integration section of ni.com

Worldwide Support

National Instruments has offices located around the world to help address your support needs. You can access our branch office Web sites from the Worldwide Offices section of ni.com. Branch office Web sites provide up-to-date contact information, support phone numbers, e-mail addresses, and current events.

If you have searched the technical support resources on our Web site and still cannot find the answers you need, contact your local office or National Instruments corporate. Phone numbers for our worldwide offices are listed at the front of this manual.