APEX WAVES

COMPREHENSIVE SERVICES

We offer competitive repair and calibration services, as well as easily accessible documentation and free downloadable resources.

SELL YOUR SURPLUS

We buy new, used, decommissioned, and surplus parts from every NI series. We work out the best solution to suit your individual needs.

Sell For Cash   Get Credit   Receive a Trade-In Deal

OBSOLETE NI HARDWARE IN STOCK & READY TO SHIP

We stock **New, New Surplus, Refurbished,** and **Reconditioned** NI Hardware.

***Bridging the gap** between the manufacturer and your legacy test system.*

1-800-915-6216

www.apexwaves.com

sales@apexwaves.com

Request a Quote CLICK HERE **PXI-5404**

CALIBRATION PROCEDURE

NI PXI-5404

This document contains step-by-step instructions for writing a calibration procedure for the NI PXI-5404 100 MHz Frequency Source.

Contents

NATIONAL INSTRUMENTS™

Calibration Overview

This section describes external calibration and explains why and how often you should calibrate your modules.

What Is External Calibration?

External calibration is a set of operations that compares the signals generated by a module or a system to external standards. The result of an external calibration can be used to determine the output error and can be used to correct the errors in the adjustment process.

The calibration process consists of verification, adjustment, and reverification. During verification, you compare the measured performance to an external standard of known measurement uncertainty to confirm that the performance of the module is better than the published specifications. During adjustment, you correct the generation error of the module by adjusting the calibration constants and storing the new calibration constants in the onboard EEPROM. The host computer reads the calibration constants and the software uses them to compensate for errors and to present a calibrated output to the user. Reverification means repeating verification after the adjustment process.

Why Should You Calibrate?

The accuracy of electronic components drifts with time and temperature, which can affect generation accuracy. Calibration restores the NI 5404 to its specified accuracy and ensures that it still meets National Instruments standards.

How Often Should You Calibrate?

NI recommends performing a complete calibration at least once every year. You can shorten this interval based on the demands of your application.

Equipment and Other Test Requirements

This section describes the test equipment, test conditions, documentation, and software required for calibration.

Test Equipment

External calibration requires different equipment for each applicable specification. Refer to Table 1 for a list of equipment.

Table 1. Equipment Required for Calibrating NI 5404 Modules

Instrument	Applicable Specification	Minimum Specifications	Recommended Instrument(s)
Digital multimeter (DMM)	SINE out amplitude accuracy	AC amplitude accuracy better than ±0.1% at 50 kHz at 4 V_{pk-pk}	NI PXI-4070 Agilent (HP) 34401A Keithley 2000
Male banana to female BNC adapter		—	—
Male BNC to female SMB cable		50 Ω	—
Spectrum analyzer/ Frequency Counter	Frequency accuracy	Bandwidth >150 MHz Frequency accuracy to ±500 ppb	NI PXI-5660 Agilent/HP 8560E Agilent/HP 53131A or HP 53132A with timebase option 001, 010, or 012 Rohde & Schwarz FSU26
Male SMA to female SMB cable		—	—
Oscilloscope	CLOCK out duty cycle accuracy	Bandwidth ≥500 MHz Timebase Accuracy: ≤200 ppm	Tektronix TDS3054
Male BNC to female SMB cable		—	—
Power meter and thermal power sensor	SINE out amplitude passband flatness	±0.12 dB accuracy for flatness measurements from 9 kHz to 105 MHz	Rohde & Schwarz NRVS meter Rohde & Schwarz NRV-Z51 or Rohde & Schwarz NRV-Z53 sensor Rohde & Schwarz NRP-Z91
Female N-Type to female SMB cable		50 Ω	—

Test Conditions

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

- Keep connections to the NI 5404 module short.
- Keep relative humidity below 80%.
- Maintain a temperature between 18 and 28 °C.

Documentation

For information on installing NI 5404 modules, you may find the following documents helpful:

- *NI Signal Generators Getting Started Guide*
- *NI Signal Generators Help*
- *NI PXI-5404 Specifications*

The *NI Signal Generators Help* includes detailed information on the NI-FGEN instrument driver functions. You can access this help file by selecting **Start»Programs»National Instruments» NI-FGEN»NI Signal Generators Help**. The *NI Signal Generators Getting Started Guide* provides instructions for installing and configuring NI signal generators.

For the latest versions of NI manuals, refer to `ni.com/manuals`.

Software

This section describes the software you need to calibrate the NI 5404.

Writing Your Calibration Procedure

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

Many of the calibration functions use constants defined in the `niArbCal.h` file. To use these constants, you must include `niArbCal.h` in your code. Refer to Table 2 for file locations.

Calibration Software

Note Only the Traditional NI-DAQ drivers support PXI-5404 calibration. The required functions of Traditional NI-DAQ are supported only by 32-bit versions of Windows XP (recommmended) and earlier operating systems.

The calibration procedure requires the latest version of NI-FGEN on the computer system that is controlling the NI 5404. NI-FGEN configures and controls NI 5404 modules. NI-FGEN ships with the NI 5404. You can also download NI-FGEN from `ni.com/updates`. NI-FGEN supports programming of all NI signal generators using a number of languages, including LabVIEW, LabWindows™/CVI™, Microsoft Visual C++, and Microsoft Visual Basic. When you install NI-FGEN, you only need to install support for the programming language that you intend to use.

Although NI-FGEN includes calibration support, the calibration application programming interface (API) is separate from NI-FGEN. During verification and adjustment, you use two APIs: the NI-FGEN API and the NI HSSources API. The NI-FGEN API controls the module both during normal operation and during the verification procedure. The NI HSSources API controls the module during the adjustment procedure.

NI-FGEN automatically installs the files listed in Table 2.

Table 2. Calibration File Locations

File Name and Location	Description
`<system directory>\niArbCal.dll`	The NI HSSources calibration toolkit library. This toolkit provides the functionality for calibrating your module.

`VXIpnp\WinNT(Win95)\lib\msc\niArbCal.lib`
`VXIpnp\WinNT(Win95)\lib\bc\niArbCal.lib`
	A `.lib` file that allows you to create applications that call functions in the `niArbCal.dll`:
	• If you build an application using Microsoft Visual C/C++, link to `msc\niArbCal.lib`.
	• If you build an application using Borland C/C++, link to `bc\niArbCal.lib`.
	• If you build an application using LabWindows/CVI, link to the library appropriate to your current compatibility mode (`msc` for Microsoft Visual C/C++, `bc` for Borland)

`VXIpnp\WinNT(Win95)\include\niArbCal.h`	A header file for the accessible functions in the `niArbCal.dll`. You must include this file in any C code that you write to call these functions.
`VXIpnp\WinNT(Win95)\nifgen\Examples\Calibration\ExternalCalibrationExample5404.c`	Example written in C that illustrates a calibration procedure using the functions in the `niArbCal.dll`.

Table 2. Calibration File Locations (Continued)

File Name and Location	Description

`VXIpnp\WinNT(Win95)\nifgen\Examples\Calibration\VisualBasic\SimpleCalibrationExample.vbp`
`VXIpnp\WinNT(Win95)\nifgen\Examples\Calibration\VisualBasic\SimpleCalibrationExample.vbw`
`VXIpnp\WinNT(Win95)\nifgen\Examples\Calibration\VisualBasic\SimpleCalibrationExample.frm`
	Example written in Visual Basic that illustrates using the functions in the `niArbCal.dll`.

`LabVIEW 6 (6.1)\instr.lib\niHSSources\niHSSources_Calibration\niHSSources_Cal.llb`	This LabVIEW LLB contains VIs that correspond to the functions in the `niArbCal.dll`.
`LabVIEW 6 (6.1)\instr.lib\niHSSources\niHSSources_Calibration\niHSSources_Cal_5404_External_Calibration_Example.llb`	This LabVIEW LLB contains VIs that demonstrate an external adjustment procedure using the VIs in `niHSSources_Cal.llb`.

Calibration Procedures in C

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.

Calibration Procedures in LabVIEW

To write calibration procedures in LabVIEW, you must use the VIs included in the `niHSSources_Cal.llb`. This library contains a VI for each function exported by the `niArbCal.dll`, as well as other useful functions. After installation, all of these VIs appear within the Legacy Calibration palette under **Functions»Measurement I/O»NI-FGEN» Calibration»Legacy Device Calibration»Calibrating the 5404**. The NI HSSources Calibration palette also contains example VIs that illustrate how to use the calibration functions.

Calibration Procedures in Visual Basic

To write calibration procedures in Visual Basic, complete the following steps:

1. Configure the project to reference the `niArbCal.dll` by selecting **Project»References** from the menu bar.
2. Click **Browse** and navigate to your system directory.
3. Find and select `niArbCal.dll`.
4. Click **Open**.
5. Click **OK**.

You can then use the `niArbCal` functions in your code and look up the functions, parameters, and constants in the Object Browser.

Verification

This section provides step-by-step instructions for verifying the NI 5404 specifications. Verification determines whether the module is performing within its specifications prior to adjustment. Verification and adjustment together comprise a complete calibration. To verify that the NI 5404 still meets its specifications, you must use NI-FGEN to control the NI 5404 module. The following steps describe the code you use to generate the appropriate signals, as well as the NI-FGEN function calls you make to verify specifications. The code varies depending on which API you use: LabVIEW, LabWindows/CVI, or Visual Basic. The examples shown in this section use LabWindows/CVI code, but LabVIEW has a corresponding VI for each function.

You can verify the following specifications for NI 5404 modules:

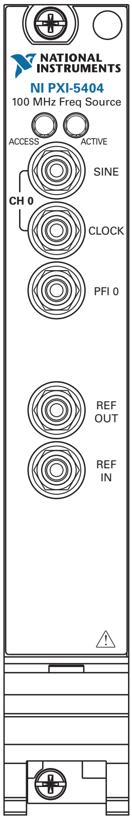
- SINE out amplitude accuracy
- Frequency accuracy
- CLOCK out duty cycle accuracy
- SINE out amplitude passband flatness

Refer to Table 1 for information concerning which instrument to use for each characteristic.

The verification procedure for each of these specifications includes setting up, programming, and cleaning up.

Figure 1 shows the NI 5404 connectors.

Figure 1. NI 5404 I/O Connectors



Verifying Frequency Accuracy

Use the spectrum analyzer specified in Table 1 to measure the frequency accuracy of the NI 5404 module. Connect the SINE out of the NI 5404 to the input of the spectrum analyzer using an SMB to SMA cable.

Generate a 10 MHz frequency using the NI 5404 module by completing the following steps:

1. Call `niFgen_init()` to initialize the module 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_ConfigureStandardWaveform()` to select the channel to output a waveform. Set the following parameters:
   - **vi**—The output value `sessionHandle` you obtained from the `niFgen_init` function
   - **Channel_Name**—`"0"`
   - **Amplitude**—`2`
   - **Frequency**—`10000000`
   - **Start phase**—`0`
   - **DC offset**—`0`
   - **Waveform**—`NIFGEN_VAL_WFM_SINE`
3. Call `niFgen_InitiateGeneration()` to initiate signal generation. This function causes the NI 5404 to leave its configuration state and enter its signal generation state. Set the following parameter:
   - **vi**—The output value `sessionHandle` you obtained from the `niFgen_init` function
4. Measure the frequency of the SINE output using a spectrum analyzer specified in Table 1. To determine if the measured frequency is within the specification of the NI 5404, you must determine the measurement error of the instrument that you used to measure the frequency and account for this error when comparing the measured frequency to the frequency accuracy listed in the *Specifications for the NI PXI-5404*.
5. Call `niFgen_close()` to close the module I/O session, destroy the instrument driver session and all of its properties, and release any memory resources the driver uses. Set the following parameter:
   - **vi**—The output value `sessionHandle` you obtained from the `niFgen_init` function

You have completed verifying the module oscillator frequency accuracy of the NI 5404.

Verifying SINE Out Amplitude Accuracy

Use a DMM specified in Table 1 to measure the SINE out amplitude accuracy of the NI 5404. Connect the SINE out of the NI 5404 to the input of the DMM using the male banana to female BNC adapter and the BNC to SMB cable.

Note A 50 Ω terminator is not used for this measurement.

Generate a 4 $V_{pk\text{-}pk}$ (into high impedance) 50 kHz sine wave as follows:

1. Call `niFgen_init()` to initialize the module 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_ConfigureStandardWaveform()` to select the channel to output a waveform. Set the following parameters:
   - **vi**—The output value `sessionHandle` you obtained from the `niFgen_init` function
   - **Channel_Name**—`"0"`
   - **Amplitude**—`2`
   - **Frequency**—`50000`
   - **Start phase**—`0`
   - **DC offset**—`0`
   - **Waveform**—`NIFGEN_VAL_WFM_SINE`
3. Call `niFgen_InitiateGeneration()` to initiate signal generation. This function causes the NI 5404 to leave its configuration state and enter its signal generation state. Set the following parameter:
   - **vi**—The output value `sessionHandle` you obtained from the `niFgen_init` function
4. Measure the rms voltage of the SINE output using a DMM specified in Table 1. To determine if the measured voltage is within the amplitude accuracy specification of the NI 5404, you must determine the measurement error of the DMM that you used to measure the rms voltage and account for this error when comparing the measured voltage to the amplitude accuracy listed in the *Specifications for the NI PXI-5404*.

Note A 4 V_{pk-pk} sine wave has an rms amplitude of 1.41421 V_{rms}.

5. Call `niFgen_close()` to close the module I/O session, destroy the instrument driver session and all of its properties, and release any memory resources the driver uses. Set the following parameter:
   - **vi**—The output value `sessionHandle` you obtained from the `niFgen_init` function

You have completed verifying the SINE out amplitude accuracy of the NI 5404.

Verifying SINE Out Amplitude Passband Flatness

The following procedure measures SINE out amplitude over several frequencies to verify that the SINE out amplitude passband flatness meets NI 5404 specifications.

Complete the following steps to verify SINE out amplitude passband flatness:

1. Call `niFgen_init()` to initialize the module you are testing and to create the 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_ConfigureStandardWaveform()` to configure the sine wave. Set the following parameters:
   - **vi**—The output that is passed in by reference to the verification functions as `sessionHandle`
   - **Channel_Name**—`"0"`
   - **Amplitude**—`2`
   - **Start phase**—`0`
   - **Frequency**—`50000`
   - **DC offset**—`0`
   - **Waveform**—`NIFGEN_VAL_WFM_SINE`
3. Call `niFgen_InitiateGeneration()`. This function causes the NI 5404 to leave its configuration state and enter its signal generation state. Set the following parameter:
   - **vi**—The output value `sessionHandle` you obtained from the `niFgen_init` function
4. Measure the SINE out amplitude using a DMM specified in Table 1.

   Note The 50 kHz amplitude accuracy of the DMM is better than the 50 kHz amplitude accuracy of the power meter. To get the best passband flatness results, you either must calibrate the power meter or use the power meter to make amplitude measurements, at frequency indices 0-11, which are relative to the absolute amplitude measurement made by the DMM at frequency index 0. The latter choice is preferred.

   To measure the SINE out amplitude passband flatness using the power meter, complete the following steps:

   a. Measure the 50 kHz amplitude using the AC rms (dBm) function of the DMM. Do not use a 50 Ω terminator when using the DMM.

   b. Subtract 6 dBm from the DMM measurement, because the power meter has a 50 Ω input impedance. The result of the subtraction is the power level of the 50 kHz SINE out into a 50 Ω load.

   c. Measure the 50 kHz amplitude with the power meter and compare the power result to the DMM result. If the power meter result is higher than the DMM result, subtract the difference of the two results (in dBm) from the power meter results for frequency indexes 0 to 11. If the power meter result is lower than the DMM result, add the difference of the two results (in dBm) to the power meter results for frequency indexes 0 to 11.
5. Disconnect the SINE out of the NI 5404 from the DMM and connect SINE out to the power meter using the SMB to N-Type cable.

6. Repeat the following steps for each frequency in Table 3:
   a. Call `niFgen_ConfigureStandardWaveform()` to configure the sine wave. Set the following parameters:
      - **vi**—The output value `sessionHandle` you obtained from the `niFgen_init` function
      - **Channel_Name**—`"0"`
      - **Amplitude**—`2`
      - **Start phase**—`0`
      - **DC offset**—`0`
      - **Frequency**—The current frequency from Table 3
      - **Waveform**—`NIFGEN_VAL_WFM_SINE`
   b. Measure the SINE out amplitude using a power meter specified in Table 1. To determine if the measured amplitude is within the specification of the NI 5404, you must determine the measurement error of the instrument that you used to measure the amplitude. You must account for this error when comparing the measured amplitude to the amplitude passband flatness accuracy listed in the *Specifications for NI PXI-5404*.

Table 3. Frequencies for Verifying SINE Out Amplitude Passband Flatness

Index	Frequency (MHz)
0	1
1	15
2	30
3	45
4	60
5	75
6	88
7	92
8	96
9	99
10	102
11	105

7. Call `niFgen_close()`. Set the following parameter:
   - **vi**—The output value `sessionHandle` you obtained from the `niFgen_init` function

You have completed verifying the SINE out amplitude passband flatness of your module.

Verifying CLOCK Out Duty Cycle Accuracy

Complete the following steps to measure the duty cycle of the NI 5404 when you use it to generate a 50% duty cycle:

1. Call `niFgen_init()` to initialize the module 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_ConfigureStandardWaveform()` to select the channel to output a waveform. Set the following parameters:
   - **vi**—The output value `sessionHandle` you obtained from the `niFgen_init` function
   - **Channel_Name**—`"0"`
   - **Amplitude**—`5`
   - **Frequency**—`1000000`
   - **Waveform**—`NIFGEN_VAL_WFM_SQUARE`
   - **Start phase**—`0`
   - **DC offset**—`0`
3. Call `niFgen_SetAttributeViReal64()`. Set the following parameters:
   - **vi**—The output value `sessionHandle` you obtained from the `niFgen_init` function
   - **Channel_Name**—`"0"`
   - **Attribute_ID**—`NIFGEN_ATTR_FUNC_DUTY_CYCLE_HIGH`
   - **Value**—`50.0`

   Note If you are using LabVIEW, select the property node attribute **niFgen» Standard Function Output»Duty Cycle High**.

4. Call `niFgen_InitiateGeneration()` to initiate signal generation. This function causes the NI 5404 to leave its configuration state and enter its signal generation state. Set the following parameter:
   - **vi**—The output value `sessionHandle` you obtained from the `niFgen_init` function

5. Measure the duty cycle of the CLOCK output using an oscilloscope specified in Table 1. To determine if the measured duty cycle is within the specification of the NI 5404, you must determine the measurement error of the oscilloscope that you used to measure the duty cycle and account for this error when comparing the measured duty cycle to the duty cycle accuracy listed in the *Specifications for the NI PXI-5404*.
6. Call `niFgen_ConfigureStandardWaveform()` to select the channel to output a waveform. Set the following parameters:
   - **vi**—The output value `sessionHandle` you obtained from the `niFgen_init` function
   - **Channel_Name**—`"0"`
   - **Amplitude**—`5`
   - **Frequency**—`25000000`
   - **Waveform**—`NIFGEN_VAL_WFM_SQUARE`
   - **Start phase**—`0`
   - **DC offset**—`0`
7. Measure the duty cycle of the CLOCK output using an oscilloscope specified in Table 1. To determine if the measured duty cycle is within the specification of the NI 5404, you must determine the measurement error of the oscilloscope that you used to measure the duty cycle and account for this error when comparing the measured duty cycle to the duty cycle accuracy listed in the *Specifications for the NI PXI-5404*.
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. Set the following parameter:
   - **vi**—The output value `sessionHandle` you obtained from the `niFgen_init` function

External Adjustment

This section describes the basic external adjustment steps for the NI 5404.

Overview of Calibration Constants

External calibration determines the value of several calibration constants, which are stored in nonvolatile memory on the module. NI-FGEN uses these constants to determine values to write to the hardware to compensate the output. Three different sets of calibration constants are determined during external calibration: constants for the voltage controlled crystal oscillator (VCXO) digital-to-analog converter (DAC), the GAIN DAC, and the SYNC DAC.

VCXO DAC

The VCXO DAC tunes the onboard VCXO. Calibrating the VCXO DAC, therefore, calibrates the frequency accuracy of the output.

GAIN DAC

Calibrating the GAIN DAC calibrates the amplitude accuracy and the amplitude passband flatness of the SINE out. Because the characteristics of the output channel vary over the frequency range of the module, the GAIN DAC must be calibrated over several frequencies.

SYNC DAC

The SYNC DAC adjusts the CLOCK out duty cycle. Calibrating the SYNC DAC, therefore, calibrates the duty cycle accuracy of the CLOCK out. Because the characteristics of the output channel vary over the frequency range of the module, the SYNC DAC must be calibrated over several frequencies.

Setting Up and Configuring the NI 5404

To set up and configure the NI 5404 module for adjustment, complete the following steps:

1. Install the NI 5404 module in the PXI chassis.
2. Configure the NI 5404 module with Measurement & Automation Explorer (MAX).
3. Referring to Table 4 and Figure 1 as guides, make the appropriate connections to measure the output for the constant you want to adjust.

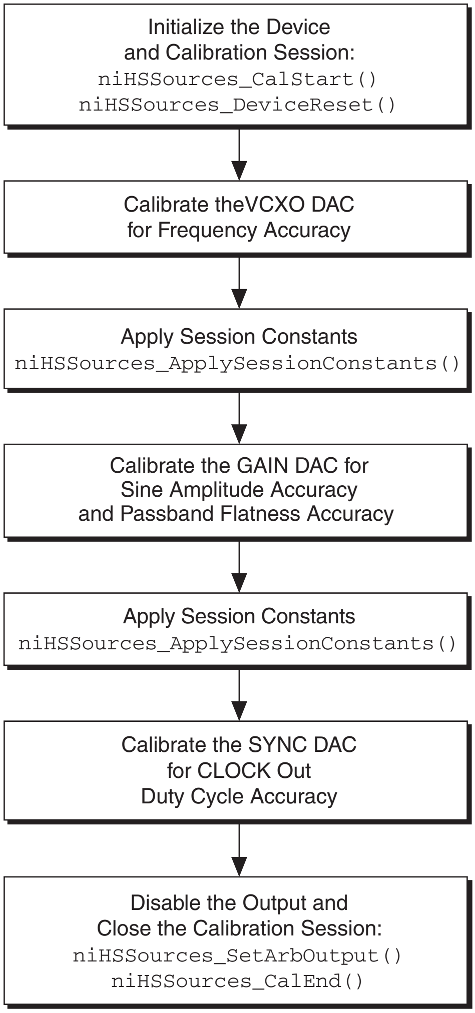
Table 4. Measuring Calibration Output

Applicable Adjustment	Type of Measurement	Output Connection	Measurement Device
SINE out amplitude accuracy and passband flatness accuracy	Magnitude of a sinusoidal voltage (either amplitude, peak-to-peak amplitude, or rms value)	SINE out	DMM and power meter
CLOCK out duty cycle accuracy	Duty cycle	CLOCK out	Oscilloscope
Frequency accuracy	Frequency	SINE out	Spectrum analyzer

Adjusting the NI 5404

Refer to Figure 2 for an overview of the adjustment process.

Figure 2. General Adjustment Procedure



To adjust the NI 5404, complete the following steps.

Note You may also want to refer to the External Calibration examples in LabVIEW or C, listed in Table 2.

1. Open a session using the niHSSources Cal CalStart VI.
2. Call `niHSSources_Cal_DeviceReset` to reset the module to the default state. Set the following parameter:
   - **sessionHandle**—The handle of the calibration session for the module
3. Complete the following steps:
   a. Adjust the VXCO DAC for frequency accuracy as described in the *Adjusting the VCXO DAC for Frequency Accuracy* section.
   b. Call `niHSSources_ApplySessionConstants()` (niHSSources Cal Apply Session Constants VI).
   c. Adjust the GAIN DAC for sine amplitude accuracy and passband flatness accuracy as described in the *Adjusting the SINE Out Amplitude* section.
   d. Call `niHSSources_ApplySessionConstants()`(niHSSources Cal Apply Session Constants VI).
   e. Adjust the CLOCK out duty cycle accuracy and passband flatness accuracy as described in the *Adjusting the CLOCK Out Duty Cycle* section.
4. Call `niHSSources_CalEnd` (niHSSources Cal CalEnd VI) to close the session using the following parameters:
   - **sessionHandle**—A pointer to the handle of the calibration session for the module
   - **action**—`niHSSources_ABORT` or `niHSSources_COMMIT_CONSTANTS`

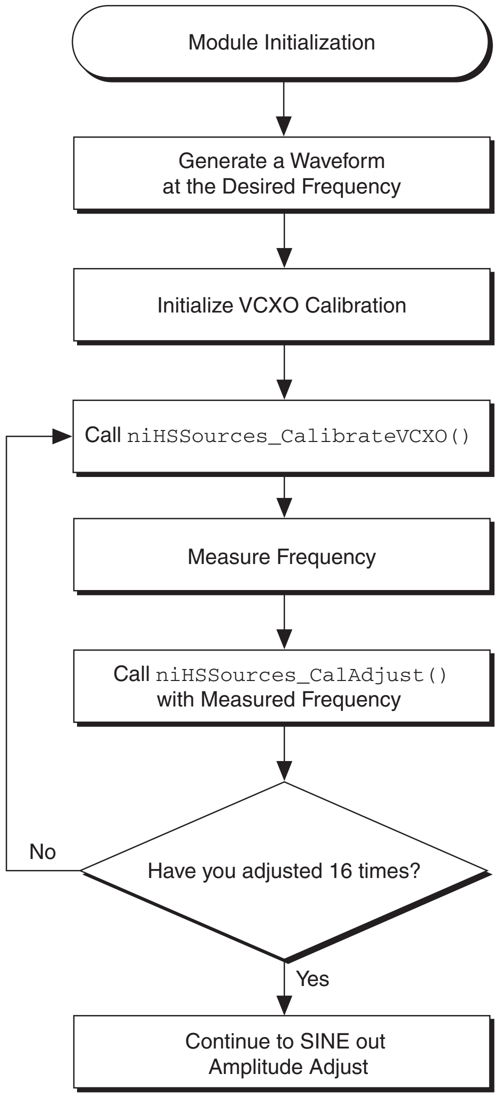
You can either save the constants you have calculated or abort the whole session. Aborting does not save changes to the module.

Adjusting the VCXO DAC for Frequency Accuracy

The VCXO DAC controls the frequency accuracy of the onboard VCXO. Here you use a calibration procedure that iteratively searches for the optimal VCXO DAC value by repeating a measurement-adjustment loop 16 times. During each step in the loop, `niHSSources_CalAdjust` (niHSSources Cal CalAdjust VI) determines the next VCXO DAC value by interrogating the measured frequency value passed to `niHSSources_CalAdjust` (niHSSources Cal CalAdjust VI) by the previous step. During this process, the calibration session holds an internal variable for the VCXO DAC value. This internal value is written to the VCXO DAC after you call `niHSSources_CalibrateVCXO` (niHSSources Cal CalibrateVCXO VI).

Refer to Figure 3 for an overview of the VCXO adjustment process.

Figure 3. VCXO Adjustment Process


Module Initialization
Generate a Waveform at the Desired Frequency
Initialize VCXO Calibration
Call niHSSources_CalibrateVCXO()
Measure Frequency
Call niHSSources_CalAdjust() with Measured Frequency
Have you adjusted 16 times?
No
Yes
Continue to SINE out Amplitude Adjust

Complete the following steps to calibrate the VCXO for frequency accuracy:

1. Call `niHSSources_GenerateWaveform` (niHSSources Cal GenerateWaveform VI) to generate a sine wave at the desired frequency 1 MHz. Set the following parameters:
   - **sessionHandle**—The handle of the calibration session for the module
   - **waveType**—`niHSSources_SINE`
   - **amplitudeInVolts**—`2`
   - **frequencyInHz**—`1000000`
2. Call `niHSSources_SetArbOutput` (niHSSources Cal SetArbOutput VI) to enable the output. Set the following parameters:
   - **sessionHandle**—The handle of the calibration session for the module
   - **outputState**—`niHSSources_ENABLE`
3. Call `niHSSources_InitializeVCXOCalibration` (niHSSources Cal InitializeVCXOCalibration VI) to reset the internal variables held by the session to begin a VCXO calibration. Set the following parameters:
   - **sessionHandle**—The handle of the calibration session for the module
   - **type**—`niHSSources_SET_VCXO`
4. Repeat steps 4a–4c 16 times to adjust the frequency accuracy:
   a. Call `niHSSources_CalibrateVCXO` (niHSSources Cal CalibrateVCXO VI) to program the adjusted VCXO DAC value to the hardware. Set the following parameters:
      - **sessionHandle**—The handle of the calibration session for the module
      - **type**—`niHSSources_SET_VCXO`

   b. Measure the frequency at the SINE out or CLOCK out on the NI 5404.

   c. Call `niHSSources_CalAdjust` (niHSSources Cal CalAdjust VI) to adjust the internal VCXO DAC value. Set the following parameters:
      - **sessionHandle**—The handle of the calibration session for the module
      - **measurementMode**—`niHSSources_SET_VCXO`
      - **actualData**—A pointer to a variable that holds the desired frequency (`1000000 Hz`). In LabVIEW, insert all inputs into an array of one element before passing it to the VI.
      - **measuredData**—A pointer to a variable that holds the measured frequency (in Hz). In LabVIEW, insert all inputs into an array of one element before passing it to the VI.

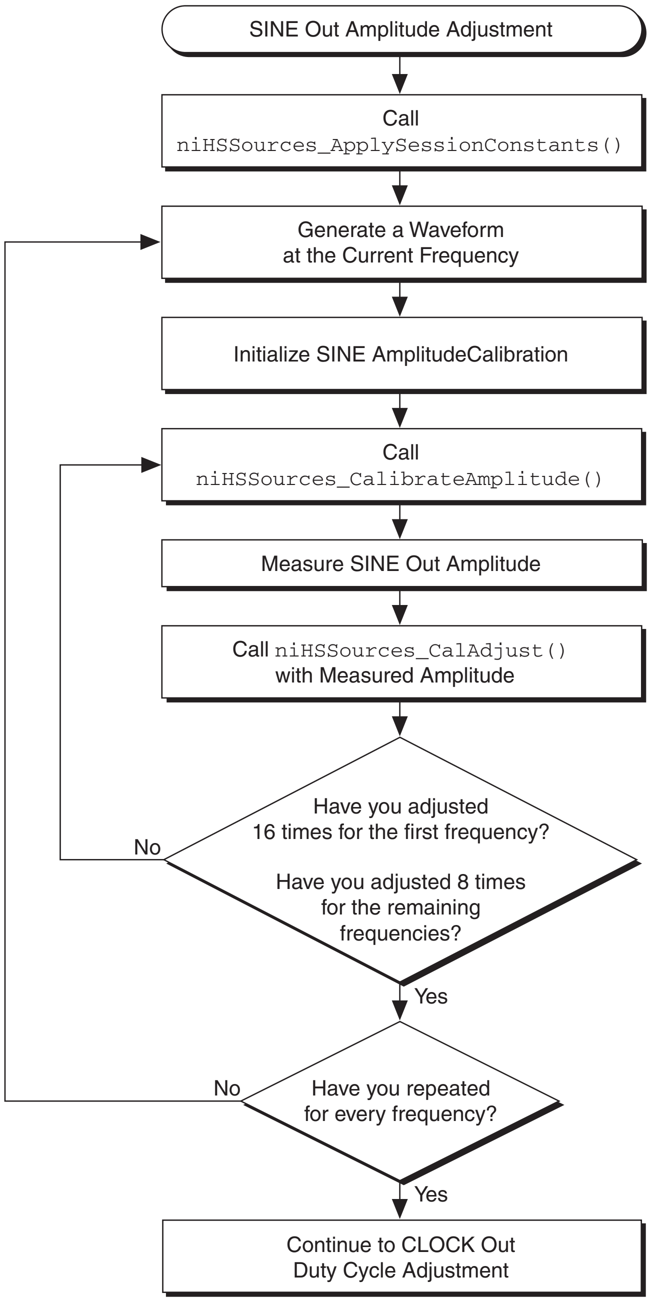
Tip If the difference between the desired and measured frequencies is smaller than the accuracy or resolution of the measurement device, you can stop the calibration at any time.

Adjusting the SINE Out Amplitude

Calibrating for amplitude accuracy and passband flatness accuracy at the SINE out adjusts the value written to the GAIN DAC, which adjusts the amplitude of the sine wave output, until the amplitude reaches the desired value. You determine the correct value for SINE out amplitude by iterating measurements and adjustments. The calibration session holds an internal variable for the GAIN DAC value. The program writes this value to the hardware when you call `niHSSources_CalibrateAmplitude` (niHSSources Cal CalibrateAmplitude VI). When the program calls `niHSSources_CalAdjust` (niHSSources Cal CalAdjust VI), this variable is adjusted according to the measured amplitude passed in. By repeatedly measuring, adjusting, and programming the variable value to the hardware, the output amplitude converges on the desired value. The amplitude is calibrated at several frequencies to flatten the passband response. For the first frequency, perform 16 adjustment iterations. For all other frequencies, perform only eight iterations.

Figure 4 shows the programming flow for the SINE out amplitude and passband flatness adjustments.

Figure 4. SINE Out Amplitude and Passband Flatness Adjustment Process


SINE Out Amplitude Adjustment
Call niHSSources_ApplySessionConstants()
Generate a Waveform at the Current Frequency
Initialize SINE AmplitudeCalibration
Call niHSSources_CalibrateAmplitude()
Measure SINE Out Amplitude
Call niHSSources_CalAdjust() with Measured Amplitude
Have you adjusted 16 times for the first frequency?
Have you adjusted 8 times for the remaining frequencies?
No
Yes
Have you repeated for every frequency?
No
Yes
Continue to CLOCK Out Duty Cycle Adjustment

Because the uncompensated amplitude varies over the bandwidth of the module, GAIN DAC calibration occurs at several frequencies to flatten the response. Repeat the amplitude calibration for each frequency. Refer to Table 5 for the applicable frequencies.

Table 5. SINE Out Calibration Frequencies

Index	Frequency	Instrument
0	50 kHz	DMM
1	6 MHz	Power meter
2	15 MHz	Power meter
3	25 MHz	Power meter
4	35 MHz	Power meter
5	43 MHz	Power meter
6	51 MHz	Power meter
7	55 MHz	Power meter
8	61 MHz	Power meter
9	65 MHz	Power meter
10	69 MHz	Power meter
11	73 MHz	Power meter
12	77 MHz	Power meter
13	82 MHz	Power meter
14	86 MHz	Power meter
15	90 MHz	Power meter
16	94 MHz	Power meter
17	97 MHz	Power meter
18	100 MHz	Power meter
19	103 MHz	Power meter
20	105 MHz	Power meter

To calibrate the SINE out amplitude, complete the following steps:

1. Call `niHSSources_ApplySessionConstants` (niHSSources Cal ApplySessionConstants VI) to pass the updated calibration constants to the underlying driver. This function ensures that the session uses the new VXCO DAC values calculated during the VCXO calibration.

2. Connect the SINE out to the appropriate instrument, which is determined by the frequency in Table 5.
3. Generate a waveform at each frequency listed in Table 5.

   For the first iteration of the steps, use the DMM as the measurement device; for the remaining iterations of the steps, use the power meter as the measurement device.

   Steps 3a through 3c explain how to use the power meter for amplitude measurements.

   Note The 50 kHz amplitude accuracy of the DMM is better than the 50 kHz amplitude accuracy of the power meter. To get the best passband flatness results, you either must calibrate the power meter or use the power meter to make amplitude measurements, at frequency indices 1-20, which are relative to the absolute amplitude measurement made by the DMM at frequency index 0. The latter choice is preferred.

   To implement using the power meter to make amplitude measurements, complete the following steps:

   a. Measure the 50 kHz amplitude using the AC rms (dBm) function of the DMM. Do not use a 50 Ω terminator when using the DMM.
   b. Subtract 6 dBm from the DMM measurement, because the power meter has a 50 Ω input impedance. The result of the subtraction is the power level of the 50 kHz SINE out into a 50 Ω load.
   c. Measure the 50 kHz amplitude with the power meter and compare the power result to the DMM result. If the power meter result is higher than the DMM result, subtract the difference of the two results from the power meter results (in dBm) for frequency indexes 1 to 20. If the power meter result is lower than the DMM result, add the difference of the two results (in dBm) to the power meter results for frequency indexes 1 to 20.

   Repeat step 3d for each frequency listed in Table 5. To generate the waveforms, complete the following steps:

   d. Repeat the following steps to generate a waveform for each frequency:
      1. Call `niHSSources_GenerateWaveform` (niHSSources Cal GenerateWaveform VI). This function generates a sine wave at the current frequency. Set the following parameters:
         - **sessionHandle**—The handle of the calibration session for the module
         - **waveform Type**—`niHSSources_SINE`
         - **amplitude In Volts**—`2.0`
         - **frequencies**—The frequency listed in Table 5 for the current index
      2. Enable the output by calling `niHSSources_SetArbOutput` (niHSSources Cal SetArbOutputVI). Set the following parameters:
         - **sessionHandle**—The handle of the calibration session for the module
         - **outputState**—`niHSSources_ENABLE`

3. Call `niHSSources_InitializeAmplitudeCalibration` (niHSSources Cal InitializeAmplitudeCalibration VI) to reset the internal variables held by the session. Resetting begins an amplitude calibration for the current frequency and sets the internal frequency index. Set the following parameters:
   - **sessionHandle**—The handle of the calibration session for the module
   - **frequencyBand**—The index of the current frequency value from Table 5
4. For the first frequency in Table 5, repeat the following steps 16 times. For the other frequencies in Table 5, repeat the following steps eight times:
   a) Call `niHSSources_CalibrateAmplitude` (niHSSources Cal CalibrateAmplitude VI) to program the adjusted GAIN DAC value to the hardware. Set the following parameter:
      - **sessionHandle**—The handle of the calibration session for the module

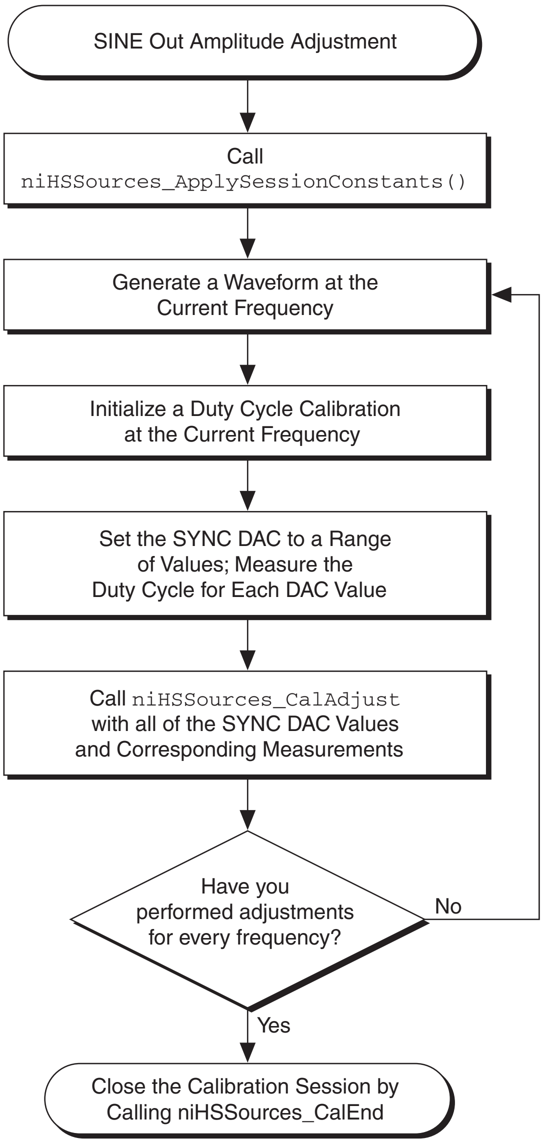
   b) Measure the SINE out amplitude.

   c) Call `niHSSources_CalAdjust`(niHSSources Cal CalAdjust VI) to adjust the internal GAIN DAC value based on the measured amplitude. Set the following parameters:
      - **measurementMode**—`niHSSources_SET_AMPLITUDE`
      - **sessionHandle**—The handle of the calibration session for the module
      - **actual data**—A pointer to the variable holding the desired amplitude level (`2.0 V`)
      - **measured amplitude**—A pointer to a variable holding the measured peak-to-peak level at the SINE out connector (assume a 50 Ω load)

Adjusting the CLOCK Out Duty Cycle

Refer to Figure 5 for an overview of the duty cycle adjustment process.

Figure 5. Duty Cycle Adjustment Process



SINE Out Amplitude Adjustment
Call niHSSources_ApplySessionConstants()
Generate a Waveform at the Current Frequency
Initialize a Duty Cycle Calibration at the Current Frequency
Set the SYNC DAC to a Range of Values; Measure the Duty Cycle for Each DAC Value
Call niHSSources_CalAdjust with all of the SYNC DAC Values and Corresponding Measurements
Have you performed adjustments for every frequency?
No
Yes
Close the Calibration Session by Calling niHSSources_CalEnd

The following procedure determines the characteristics of the SYNC DAC. The SYNC DAC determines the duty cycle of the CLOCK out. The program writes several values to the SYNC DAC and measures the duty cycle of the CLOCK out for each of these values.

The program then passes the SYNC DAC values and measurements of the resulting duty cycles to `niHSSources_CalAdjust` (niHSSources Cal CalAdjust VI).

The duty cycle of an uncalibrated module depends on the frequency of the CLOCK out. The calibration procedure determines SYNC DAC values that compensate for the frequency dependency of the duty cycle.

For each frequency, initiate a generation at that frequency. Pass the zero-based index to the `niHSSources_InitializeDutyCycleCalibration` function (niHSSources Cal InitializeDutyCycleCalibration VI).

Table 6. SYNC DAC Frequencies

Index	Frequency	SYNC DAC Values	Number of Measurements
0	50 kHz	48, 60, 70, 80, 115, 150, 172, 182, 192, 200	10
1	15 MHz	48, 60, 70, 80, 115, 150, 172, 182, 192, 200	10
2	30 MHz	45, 57, 67, 77, 87, 130, 165, 175, 184, 196	10
3	45 MHz	38, 50, 60, 70, 85, 120, 155, 165, 175, 192	10
4	60 MHz	38, 50, 60, 80, 105, 125, 145, 155, 165, 177	10
5	75 MHz	75, 81, 87, 93, 99, 105, 111, 117, 123, 129	10
6	90 MHz	80, 84, 88, 92, 96, 100, 104, 108, 112, 116	10
7	105 MHz	70, 73, 76, 79, 82, 85, 88, 91, 94, 97	10

To calibrate and adjust the duty cycle of the CLOCK out, complete the following steps:

1. Call `niHSSources_ApplySessionConstants` (niHSSources Cal ApplySessionConstants VI) to instruct the driver to use data from previous calibration steps of the session. This function ensures that the driver uses the values calculated during the VCXO and SINE out amplitude calibrations. Set the following parameter:
   - **sessionHandle**—The handle of the calibration session for the module
2. Connect the CLOCK out to an oscilloscope.
3. Set the termination of the oscilloscope listed in Table 1 to 50 Ω.
4. Repeat the following steps for each frequency index listed in Table 6:
   a. Call `niHSSources_GenerateWaveform` (niHSSources Cal GenerateWaveform VI) to generate a sine wave at a frequency listed in Table 6. Set the following parameters:
      - **sessionHandle**—The handle of the calibration session for the module
      - **waveformType**—`niHSSources_SINE`

- **amplitudeInVolts**—`2.0`
- **frequencyInHz**—The frequency listed in Table 6 for the current index

b. Enable the output by calling `niHSSources_SetArbOutput` (niHSSources Cal SetArbOutputVI). Set the following parameters:
   - **sessionHandle**—The handle of the calibration session for the module
   - **outputState**—`niHSSources_ENABLE`

c. Call `niHSSources_InitializeDutyCycleCalibration` (niHSSources Cal InitializeDutyCycleCalibration VI) to reset the internal variable held by the session to begin a duty cycle calibration for the present frequency. Set the following parameters:
   - **sessionHandle**—The handle of the calibration session for the module
   - **frequencyBand**—An integer, shown in Table 6, representing the index of the frequency for the calibration being performed
   - **NumberOfMeasurements**—The number of duty cycle measurements taken and passed into the `niHSSources_CalAdjust()` function (niHSSources Cal CalAdjust VI)

d. For each SYNC DAC value in Table 6 for the frequency index, complete the following steps:
   1. Call `niHSSources_SetSYNCDAC` (niHSSources Cal SetSYNCDAC VI) to program the SYNC DAC. Set the following parameters:
      - **sessionHandle**—The handle of the calibration session for the module
      - **DACValue**—One of the SYNC DAC values in Table 6 for the current frequency index
   2. Measure the duty cycle clock at the CLOCK out.

e. Call `niHSSources_CalAdjust` (niHSSources Cal CalAdjust VI) to pass the SYNC DAC values set and the resulting duty cycle measurements to `CalAdjust`. Set the following parameters:
   - **sessionHandle**—The handle of the calibration session for the module
   - **measurementMode**—`niHSSources_SET_DUTY_CYCLE`
   - **actualData**—An array of SYNC DAC values, as shown in Table 6
   - **measuredData**—An array of measured duty cycles for each SYNC DAC value. Each value should represent the duty cycle high, as a percentage.

Closing the Calibration Session

If a function returns an error, call `niHSSources_CalEnd` (niHSSources Cal CalEnd VI) to abort and close the session. Set the following parameters:

- **sessionHandle**—The handle of the calibration session for the module
- **action**—`niHSSources_ABORT`

If you do not receive an error, call `niHSSources_CalEnd` (niHSSources Cal CalEnd VI) to save the new constants. Set the following parameters:

- **sessionHandle**—The handle of the calibration session for the module
- **action**—`niHSSources_COMMIT_CONSTANTS`

Calibration Function Reference

This section provides detailed descriptions of NI 5404 calibration and constant functions that support signal source performance. The NI HSSources API provides access to the `niarbcal.dll` and can be used to write calibration procedures for any NI 5404 module.

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

niHSSources_CalStart

Function Prototype

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

Purpose

Initiates a calibration session for the specified module.

Parameters

Name	Description
device	A number corresponding to the DAQ device number of the module to be calibrated.
password	The external adjustment password. This password 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 calibration functions.

niHSSources_CalAdjust

Function Prototype

```
__declspec(dllexport) long__stdcall
   niHSSources_CalAdjust(unsigned long   sessionHandle,
                  unsigned long  measurementMode,
                  double         *actualData,
                  double         *measuredData);
```

Purpose

Adjusts 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 module, created by calling `niHSSources_CalStart`.
measurementMode	The calibration constant(s) being adjusted—one of the constants starting with `niHSSources_SET`.
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.

niHSSources_CalEnd

Function Prototype

```
__declspec(dllexport) long __stdcall
   niHSSources_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 module; created by calling `niHSSources_CalStart`.
action	`niHSSources_ABORT` or `niHSSources_COMMIT_CONSTANTS`. Using abort avoids storing the newly calculated calibration constants in the onboard EEPROM.

niHSSources_CalFetchDate

Function Prototype

```
__declspec(dllexport) long __stdcall
   niHSSources_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 module to be calibrated.
area	`niHSSources_EXTERNAL`.
month	A pointer to a long integer that receives the month value.
day	A pointer to a long integer that receives the date value.
year	A pointer to a long integer that receives the year value.

niHSSources_CalFetchCount

Function Prototype

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

Purpose

Retrieves the number of times the module 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 module to be calibrated.
area	`niHSSources_EXTERNAL`.
count	A pointer to a long integer that receives the count value.

niHSSources_CalFetchMiscInfo

Function Prototype

```
__declspec(dllexport) long __stdcall
   niHSSources_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 module to be calibrated.
miscInfo	A character pointer to a buffer that is at least four bytes long. You should allocate this buffer.

niHSSources_CalStoreMiscInfo

Function Prototype

```
__declspec(dllexport) long __stdcall
   niHSSources_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 module to be calibrated.
miscInfo	A character pointer to a buffer that is at least four bytes long. You should allocate this buffer.

niHSSources_CalChangePassword

Function Prototype

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

Purpose

Changes the password.

Parameters

Name	Description
device	A number corresponding to the DAQ device number of the module to be calibrated.
oldPassword	The current password.
newPassword	The new password that replaces the current password.

niHSSources_DeviceReset

Function Prototype

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

Purpose

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

Parameter

Name	Description
sessionHandle	The handle of the calibration session for the module; created by calling `niHSSources_CalStart`.

niHSSources_SetArbOutput

Function Prototype

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

Purpose

Enables or disables the output of the module.

Parameters

Name	Description
sessionHandle	The handle of the calibration session for the module; created by calling `niHSSources_CalStart`.
outputState	Indicates whether to enable or disable the output—`niHSSources_ENABLE` or `niHSSources_DISABLE`.

niHSSources_GenerateWaveform

Function Prototype

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

Purpose

Causes the module to output a sine wave with a specified amplitude and frequency.

Parameters

Name	Description
sessionHandle	The handle of the calibration session for the module; created by calling `niHSSources_CalStart`.
waveType	The type of waveform to generate—`niHSSources_SINE`.
amplitudeInVolts	The peak-to-peak amplitude of the waveform (in volts); this is the amplitude into a terminated (50 Ω) load.
frequencyInHz	The frequency of the waveform (in Hz); ignored for DC.

niHSSources_InitializeVCXOCalibration

Function Prototype

```
__declspec(dllexport) long __stdcall
   niHSSources_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 VCXO calibration.

Parameters

Name	Description
sessionHandle	The handle of the calibration session for the module; created by calling `niHSSources_CalStart`.
type	The type of VCXO calibration—`niHSSources_SET_VCXO`.

niHSSources_CalibrateVCXO

Function Prototype

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

Purpose

Recalibrates the VCXO with the current test value. Use this function 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 module; created by calling `niHSSources_CalStart`.
type	The type of VCXO calibration to be done—`niHSSources_SET_VCXO`.

niHSSources_CalGetLastCalibrationTemperature

Function Prototype

```
__declspec(dllexport) long __stdcall
    niHSSources_CalGetLastCalibrationTemperature(unsigned long
    device,
    int            area,
    double*        temperature);
```

Purpose

Retrieves the temperature at which the module was last calibrated.

Parameters

Name	Description
device	A number corresponding to the DAQ device number of the module.
area	`niHSSources_EXTERNAL`.
temperature	A pointer to a double that receives the temperature value in °C.

niHSSources_ApplySessionConstants

Function Prototype

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

Purpose

During calibration, the session holds an internal set of calibration constants. These constants may have been updated based on calibration steps performed during the session. This function commits the session constants to the underlying driver so that the module is calibrated according to the newly updated constants for the remainder of the session. This function does not store the updated constants in the onboard EEPROM. Calling the function `niHSSources_CalEnd` stores the updated constants in the onboard EEPROM.

Parameters

Name	Description
sessionHandle	A handle to the calibration session for the module; created by calling `niHSSources_CalStart`.

niHSSources_InitializeAmplitudeCalibration

Function Prototype

```
__declspec(dllexport) long __stdcall
   niHSSources_InitializeAmplitudeCalibration(unsigned long
   sessionHandle, unsigned long frequencyBand);
```

Purpose

The SINE out amplitude of the NI 5404 is calibrated iteratively. Call this function before calibrating the sine amplitude because it sets many internal variables to appropriate values for the calibration.

Parameters

Name	Description
sessionHandle	A handle to the calibration session for the module; created by calling `niHSSources_CalStart`.
frequencyBand	An integer representing the index of the frequency band under which this calibration is being performed.

niHSSources_CalibrateAmplitude

Function Prototype

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

Purpose

Returns the sine amplitude with the current test value. Use this function during calibration to iteratively converge on the best value. A new value is calculated, then output is measured, then a new value is calculated, and so on.

Parameters

Name	Description
sessionHandle	A handle to the calibration session for the module; created by calling `niHSSources_CalStart`.

niHSSources_InitializeDutyCycleCalibration

Function Prototype

```
__declspec(dllexport) long __stdcall
   niHSSources_InitializeDutyCycleCalibration(unsigned long
   sessionHandle,
      unsigned long frequencyBand,
      unsigned long numberOfMeasurements);
```

Purpose

Call this function before calibrating the duty cycle of the CLOCK out because it sets many internal variables to appropriate values for the calibration.

Parameters

Name	Description
sessionHandle	A handle to the calibration session for the module; created by calling `niHSSources_CalStart`.
frequencyBand	An integer representing the index of the frequency band under which this calibration is being performed.
numberOfMeasurements	The number of duty cycle measurements you pass into the `niHSSources_CalAdjust()` function (niHSSources Cal CalAdjust VI).

niHSSources_SetSYNCDAC

Function Prototype

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

Purpose

Sets the SYNC calibration DAC to the specified value. This value affects the duty cycle of the signal at the CLOCK out and is used during a duty cycle calibration.

Parameters

Name	Description
sessionHandle	A handle to the calibration session for the module; created by calling `niHSSources_CalStart`.
DACValue	The integer value to write to the SYNC DAC—ranges from 0 to 255.

Where to Go for Support

The National Instruments website is your complete resource for technical support. At `ni.com/support` you have access to everything from troubleshooting and application development self-help resources to email and phone assistance from NI Application Engineers.

Visit `ni.com/services` for NI Factory Installation Services, repairs, extended warranty, and other services.

Visit `ni.com/register` to register your National Instruments product. Product registration facilitates technical support and ensures that you receive important information updates from NI.

A Declaration of Conformity (DoC) is our claim of compliance with the Council of the European Communities using the manufacturer's declaration of conformity. This system affords the user protection for electromagnetic compatibility (EMC) and product safety. You can obtain the DoC for your product by visiting `ni.com/certification`. If your product supports calibration, you can obtain the calibration certificate for your product at `ni.com/calibration`.

National Instruments corporate headquarters is located at 11500 North Mopac Expressway, Austin, Texas, 78759-3504. National Instruments also has offices located around the world. For telephone support in the United States, create your service request at `ni.com/support` or dial 1 866 ASK MYNI (275 6964). For telephone support outside the United States, visit the Worldwide Offices section of `ni.com/niglobal` to access the branch office websites, which provide up-to-date contact information, support phone numbers, email addresses, and current events.

370558D-01 May14