# **NI–TUNER Reference Help**

August 2004 Edition, Part Number 370415C-01

This help file provides programming support for NI–TUNER, the driver that communicates with the National Instruments PXI-5600 RF downconverter module. Intended for LabVIEW and CVI programmers, this help file contains functions for configuring, opening a session with, and closing the device.

To navigate this help file, use the **Contents** and **Index** tabs to the left of this window.

For more information about this help file, refer to the following topics:

<u>Conventions</u>—formatting and typographical conventions in this help file

**Important Information** 

Technical Support and Professional Services

To comment on the documentation, email techpubs@ni.com

© 2001-2004 National Instruments Corporation. All rights reserved.

## Conventions

This help file uses the following conventions:

< >	Angle brackets that contain numbers separated by an ellipsis represent a range of values associated with a bit or signal name—for example, $DBIO < 30 >$ .
»	The » symbol leads you through nested menu items and dialog box options to a final action. The sequence <b>File</b> » <b>Page Setup</b> » <b>Options</b> directs you to pull down the <b>File</b> menu, select the <b>Page Setup</b> item, and select <b>Options</b> from the last dialog box.
[]	If attached to a parameter, brackets indicate an array.
*	An asterisk signifies a pointer.
$\mathbf{N}$	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, emphasis, or an introduction to a key concept.
green	Underlined text in this color denotes a link to a help topic, help file, or Web address.
italic	Italic text denotes variables or cross references. 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.

## Attenuation

This section describes how attenuation is handled by the NI–TUNER driver. The user modifies attenuation through the <u>niTuner Set Attenuation VI</u> or the <u>niTuner\_setAttenuation</u> function.

#### The Hardware

The downconverter signal chain has five programmable attenuators: three RF attenuators at the beginning of the chain and two IF attenuators near the end of the chain. They are set up in the following sequence:

Attenuator Sequence		
Attenuator	Asserted Value	
RF Attenuator 1	20 dB	
RF Attenuator 2	20 dB	
RF Attenuator 3	10 dB	
IF Attenuator 1	20 dB	
IF Attenuator 2	10 dB	

Attenuators are either set/asserted or not set/asserted. This allows a dynamic range of RF attenuation from 0-50 dB and a dynamic range of IF attenuation from 0-30 dB. Overall attenuation within the signal chain is the sum of all the attenuators set, for a range of 0-80 dB.

#### **Coercion, Step 1**

The five attenuators are configured based upon the mixer and reference levels set in the

# <u>niTuner Set Attenuation VI</u> or the <u>niTuner\_setAttenuation</u> function.

First NI–TUNER checks whether the user intends to use the AutoMode feature. In AutoMode, the user defines the reference level, while NI–TUNER determines the appropriate mixer level. AutoMode is activated in NI–TUNER by setting a mixer level value above *1000*. This feature will then attempt to set the mixer level to -20 dBm and adjust within the algorithmic coercions described below. In AutoMode the only warning the user may see concerns coercion to correct an invalid reference level.

NI–TUNER coerces the settings according to the following rules:

- The mixer level may not be above 0 dBm. If the mixer level breaks this rule, NI–TUNER sets the mixer level to 0 dBm and returns this warning: "Mixer level cannot exceed 0 dBm. The mixer level setting is coerced to 0 dBm"
- The reference level may not be above 50 dBm. If the reference level is greater than 50 dBm, NI–TUNER sets the reference level to 50 dBm and returns this warning: "The reference level cannot exceed 50 dBm. The reference level setting is coerced to 50 dBm"
- The mixer level may not be greater than the reference level. If the mixer level is greater than the reference level, NI–TUNER sets the mixer level equal to the current value of the reference level and returns this warning: "The specified mixer level does not fall between the reference level and the reference level minus 50 dBm. The mixer level setting is coerced to the nearest of these two bounds"
- The mixer level may not be less than the reference level minus 50 dBm. If the mixer level breaks this rule, NI–TUNER sets the mixer level to the value of reference level minus 50 dBm and returns this warning: "The specified mixer level does not fall between the reference level and the reference level minus 50 dBm. The mixer level setting is coerced to the nearest of these two bounds"

After this, coercion occurs only if the requested reference level and mixer level create an impossible state in programming the downconverter's attenuators.

#### Setting the Attenuators, Step 2

Understanding the parts played by the reference level and the mixer level is key to setting the individual attenuators. NI–TUNER sets the RF attenuators first.

Let *D* be the difference between the reference level and the mixer level, such that

#### D = reference level - mixer level

This formula correlates directly with the range of possible RF attenuator settings. Recall that the RF attenuators have a range of 0-50 dB. The coercions described in the previous section ensure that the mixer level will be less than the reference level and that the mixer level will not be more than 50 dBm less than the reference level. *D* is then directly proportional to the total RF attenuation, a value between 0-50 dB. All that remains is to set the RF attenuators as follows (refer to the <u>Attenuator Sequence table</u> above):

	<b>RF</b> Attenuator Settings		
D	Attenuators		
0	None are set		
10	RF Attenuator 3		
20	RF Attenuator 1		
30	RF Attenuators 1 and 3		
40	RF Attenuators 1 and 2		
50	RF Attenuators 1, 2, and 3		

Attenuation of the signal on the other end of the chain is modified by configuring the IF attenuators.

Think of the IF attenuators as adjustable compensation for the inherent 30 dB gain of the downconverter. Let *I* be the 30 dB adjustment of our ratio, such that

 $I = 30 \, dB + Reference Level - D$ 

or simply,

I = 30 dB + Mixer Level

Here the NI–TUNER driver may be forced to perform another coercion. The value of the mixer level is less than or equal to 0 dBm due to previous coercion, but what happens if the mixer level is less than 30 dBm? This would produce a negative attenuation (i.e., a gain), and there are no programmable IF gains available. If *I* is less than 0 dB, then NI–TUNER coerces *I* to 0 dB and returns this warning: "A mixer level less than -30 dBm produces an IF output below the

nominal level of 0 dBm. See your NI–TUNER Reference Help".

*I* is directly proportional to the total IF attenuation, a value between 0 - 30 dB. The IF attenuators are set as follows (refer to the <u>Attenuator Sequence table</u> above):

IF Attenuator Settings		
I Attenuators		
0	None are set	
10	IF Attenuator 2	
20	IF Attenuator 1	
30	IF Attenuators 1 and 2	

#### **One More Adjustment**

After the attenuation is set, if the user makes driver calls to the <u>niTuner Set Freq</u> <u>VI</u> or the <u>niTuner\_setFreq</u> function, an additional adjustment may be made. If at least one frequency requested of the NI–TUNER is less than 15 MHz, the downconverter functions instead as an upconverter. This process produces a saturation in the signal. In order to account for this saturation, NI–TUNER immediately sets the 10 dB IF attenuator if it is not already set.

### **Error and Status Codes**

Error Code	Meaning	CVI Defines		
0	Successful execution	NITUNER_ERROR_SUCCESS		
Positive Values	Warnings			
27009	Unable to load calibration settings	NITUNER_WARN_CALIBRATIONLOADFAIL		
27022	Requested attenuation levels have been coerced to fit within the 0 to 80 dB range	NITUNER_WARN_ATTENUATIONCOERCED		
27027	The specified mixer level does not fall between the reference level and the reference level minus 50 dBm. The mixer level setting is coerced to the nearest of these two bounds.	NITUNER_WARN_BAD_MIXER_RANGE		
27028	The reference level cannot exceed 50 dBm. The reference level setting is coerced to 50 dBm.	NITUNER_WARN_BAD_REFERENCE_RANGE		
27029	Input frequencies above 2.7 GHz are outside specification and may return invalid data.	NITUNER_WARN_MAX_SPEC		
27030	Mixer level cannot exceed 0 dBm. The mixer level setting is coerced to 0 dBm.	NITUNER_WARN_MIXER_ABOVE_ZERO		
27031	A mixer level less than -30 dBm produces an IF output below the nominal level of 0 dBm. See your NI–TUNER Reference Help.	NITUNER_WARN_BAD_IF_RANGE		
Negative Values	Errors			
-27026	The specified ready pulse length is out of range	NITUNER_ERROR_INVALID_PULSE_LENGTH		
-27025	Signal polarities may be only ACTIVE HIGH or ACTIVE LOW	NITUNER_ERROR_INVALID_SIGNAL_POLARIT		
-27024	An unknown signal type was specified	NITUNER_ERROR_INVALID_SIGNAL_TYPE		
-27023	An unknown signal path was specified	NITUNER_ERROR_INVALID_SIGNAL_PATH		
-27021	The requested attribute is not used in this function	NITUNER_ERROR_INVALIDATTRIBUTE		
-27020	The FPGA download failed	NITUNER_ERROR_FPGADOWNLOADFAILED		
-27019	The FPGA programming chip failed to initialize	NITUNER_ERROR_CPLDREJECT		
-27018	One of the parameters was NULL, making it unusable. Allocate memory for the parameter.	NITUNER_ERROR_NULLPOINTER		
-27017	This setting attempts to send signals in two different directions at the same time	NITUNER_ERROR_CLOCKCONFLICT		
-27016	Unknown PLL	NITUNER_ERROR_UNKNOWNPLL		
-27015	Invalid RTSI setting	NITUNER_ERROR_INVALIDRTSIVALUE		
-27014	Initialize the PXI-5600 downconverter before calling this function	NITUNER_ERROR_INITNOTCALLED		

-27013	Reset the PXI-5600 downconverter before calling this function	NITUNER_ERROR_RESETNOTCALLED
-27012	Value is out of range	NITUNER_ERROR_RANGEERROR
-27011	The span must be between 0 and 20 MHz inclusive	NITUNER_ERROR_BADSPAN
-27010	Array size must be greater than 0	NITUNER_ERROR_INVALIDARRAYSIZE
-27008	Unable to write data to the EEPROM: verification of written data failed	NITUNER_ERROR_EEPROMWRITERFAIL
-27007	Unable to obtain a direct link to the PXI-5600 downconverter registers	NITUNER_ERROR_DEVICEPOINTER
-27006	Unable to complete DMA transfer	NITUNER_ERROR_INFINTELOOP
-27005	The attenuation relay is busy. This is a hardware problem.	NITUNER_ERROR_RELAYBUSY
-27004	The downconverter is busy. Its internal state machine may be locked up.	NITUNER_ERROR_DOWNCONVERTERBUSY
-27003	Frequency counter timed out	NITUNER_ERROR_FREQCOUNTTIMEOUT
-27002	Unable to allocate memory	NITUNER_ERROR_ALLOCATEMEMORY
-27001	A bit size too large was specified for this serial chain. This is an internal driver issue.	NITUNER_ERROR_INVALIDSERIALCHAINSIZE

Name or Class	Function Name
Initialize	<u>niTuner init</u>
Set Frequency	<u>niTuner_setFreq</u>
Close	<u>niTuner_close</u>
General Configuration	
Set Attenuation	niTuner setAttenuation
Get Attenuation	niTuner_getAttenuation
Synchronization	
Configure Advance Trigger	niTuner_configAdvanceTrigger
Configure Ready Signal	niTuner configReadySignal
Configure Reference Clock	niTuner configReferenceClock
Initiate Scan	<u>niTuner</u> initiateScan
Send Software Trigger	niTuner_sendSoftwareTrigger
Scan List Configuration	
Set Frequency Scan List	niTuner_setFreqScanList
Get Calibration	<u>niTuner_getCal</u>
Utility	
Get Temperature	niTuner getTemperature
Ready	<u>niTuner</u> ready
Error Handler	niTuner_errorHandler

# **NI–TUNER CVI Function Tree**

# niTuner\_close

#### **Function Prototype**

int niTuner\_close (int taskID);

#### Purpose

This function closes the instrument I/O session.

#### **Parameters**

Name	Туре	Description
taskID	int	The Task ID obtained from <u>niTuner init</u> that identifies the session of a particular device.

#### **Return Value**

Returns the status of the function. To see an explanation of common error and warning codes, see <u>Error and Status Codes</u>. To see a text explanation of the status code, call <u>niTuner\_errorHandler</u>.

# niTuner\_configAdvanceTrigger

#### **Function Prototype**

int niTuner\_configAdvanceTrigger (int taskID, unsigned int signalSource, unsigned int signalType, unsigned int signalPolarity);

#### Purpose

This function configures the scan advance trigger input. The advance trigger tells the NI PXI-5600 downconverter to advance to the next frequency in the scan list. The ready signal is generated after the downconverter has settled to a frequency.

#### **Parameters**

Name	Туре	Description
taskID	int	The Task ID obtained from <u>niTuner init</u> that identifies the session of a particular device.
signalSource	unsigned int	Identifies the source of the scan advance trigger signal. Hardware triggers are RTSI lines <06 >.
		Defined Values:
		NITUNER SIGNAL SOFTWARE
		NITUNER SIGNAL RTSIO
		NITUNER_SIGNAL_RTSI1
		NITUNER_SIGNAL_RTSI2
		NITUNER SIGNAL RTSI3
		NITUNER SIGNAL RTSI4
		NITUNER SIGNAL RTSI5
		NITUNER SIGNAL RTSI6
		Default Value:
		NITUNER_SIGNAL_SOFTWARE
signalType	unsigned int	Configures the RF downconverter to advance to the next frequency in the scan list when a <b>pulse</b> or a <b>level</b> signal is detected.
		Defined Values:
		NITUNER SIGNAL PULSE
		NITUNER SIGNAL LEVEL
		Default Value:
		NITUNER_SIGNAL_PULSE
		If selected, the expected <b>pulse</b> must be in the direction of the <b>signalPolarity</b> . If selected, the <b>level</b> (while the PXI-5600 downconverter is not expected to trigger) will be in the direction of the <b>signalPolarity</b> .
signalPolarity	unsigned int	Configures the PXI-5600 downconverter to advance to the next frequency in the scan list on the specified polarity change.
		Defined Values:
		NITUNER SIGNAL ACTIVE LOW
		NITUNER SIGNAL ACTIVE HIGH
		Default Value:
		NITUNER_SIGNAL_ACTIVE_LOW

#### **Return Value**

Returns the status of the function. To see an explanation of common error and warning codes, see <u>Error and Status Codes</u>. To see a text explanation of the status code, call <u>niTuner\_errorHandler</u>.

# niTuner\_configReadySignal

#### **Function Prototype**

int niTuner\_configReadySignal (int taskID, unsigned int signalDestination, unsigned int signalType, unsigned int signalPolarity, double pulseLength);

#### Purpose

This function configures the PXI-5600 downconverter ready signal output, which is sent on one of the RTSI lines when the downconverter has finished settling to a specified frequency.

#### **Parameters**

Name	Туре	Description
taskID	int	The Task ID obtained from <u>niTuner init</u> that identifies the session of a particular device.
signalDestination	-	Identifies the destination of the ready signal output:
	int	Defined Values:
		NITUNER SIGNAL NO DESTINATION
		NITUNER SIGNAL RTSIO
		NITUNER SIGNAL RTSI1
		NITUNER SIGNAL RTSI2
		NITUNER SIGNAL RTSI3
		NITUNER SIGNAL RTSI4
		NITUNER SIGNAL RTSI5
		NITUNER SIGNAL RTSI6
		Default Value:
		NITUNER_SIGNAL_NO_DESTINATION
signalType	unsigned int	Configures the type of signal the PXI-5600 downconverter generates when it has settled to a frequency. The signal may be represented as a pulse of pulseLength seconds in the direction of the polarity, or as a level signal that returns to the state specified by signalPolarity.
		For more information about these signal types, see the trigger timing diagram.
		Defined Values:
		NITUNER SIGNAL PULSE
		NITUNER SIGNAL LEVEL
		Default Value:
		NITUNER_SIGNAL_PULSE
signalPolarity	unsigned int	Configures the PXI-5600 downconverter to send a ready signal of selected type with polarity of active low or active high.
		Defined Values:
		NITUNER SIGNAL ACTIVE LOW
		NITUNER SIGNAL ACTIVE HIGH
		Default Value:
		NITUNER_SIGNAL_ACTIVE_LOW
pulseLength	double	Time in seconds to generate a signal of type <b>pulse</b> in the direction specified by signalPolarity.

#### **Return Value**

Returns the status of the function. To see an explanation of common error and warning codes, see <u>Error and Status Codes</u>. To see a text explanation of the status code, call <u>niTuner\_errorHandler</u>.

# niTuner\_configReferenceClock

#### **Function Prototype**

int niTuner\_configReferenceClock (int taskID, unsigned int referenceConfiguration);

#### Purpose

This function configures the reference clock source. The PXI-5600 downconverter must lock to a timebase before entering the ready state.

#### **Parameters**

Name	Туре	Description
taskID	int	The Task ID obtained from <u>niTuner init</u> that identifies the session of a particular device.
reference Configuration	unsigned int	Configures the reference clock source: Defined Values:
		<u>NITUNER_INTERNAL</u>
		NITUNER DRIVE 10 MHZ PXI BACKPLANE CLOCK
		NITUNER LOCK TO 10 MHZ PXI BACKPLANE CLOCK
		NITUNER EXTERNAL
		NITUNER DRIVE 10 MHZ PXI BACKPLANE CLOCK EXTERNAL
		Default Value:
		NITUNER_INTERNAL

#### **Return Value**

Returns the status of the function. To see an explanation of common error and warning codes, see <u>Error and Status Codes</u>. To see a text explanation of the status code, call <u>niTuner\_errorHandler</u>.

## niTuner\_errorHandler

#### **Function Prototype**

int niTuner\_errorHandler (int taskID, int errorCode, char errorSource[], char errorDescription[]);

#### Purpose

This function converts a returned <u>error code</u> into a text description of the error.



**Note** 0 may be passed as the **taskID**. This is useful to interpret errors after <u>niTuner\_init</u> has failed or if errorSource is NULL. The **taskID** is required for cases in which errorSource is desired.

#### **Parameters**

Name	Туре	Description	
taskID	int	The Task ID obtained from <u>niTuner init</u> that identifies the session of a particular device.	
errorCode	int	Passes the error code returned from any of the instrument driver functions.	
errorSource	char[]	Returns the name of the function call where the error occurred. Pass a char array at least <u>MAX_FUNCTION_NAME_SIZE</u> bytes in length, or pass NULL.	
errorDescription	iption       char[]       Returns a text description of the error.         Pass a char array at least MAX_ERROR_DESCRIPTION       bytes in length, or pass NULL.		

#### **Return Value**

Returns the status of the function. To see an explanation of common error and warning codes, see <u>Error and Status Codes</u>.

## niTuner\_getAttenuation

#### **Function Prototype**

int niTuner\_getAttenuation (int taskID, unsigned long listSize, double \*attenuation, double \*scaleFactor);

#### Purpose

This function returns the IF signal attenuation for each frequency in the scan list. Currently, this function sets the same attenuation for each frequency in the scan list. Future versions of NI–TUNER may allow different attenuation values for each frequency.

#### **Parameters**

Name	Туре	Description
taskID	int	The Task ID obtained from <u>niTuner init</u> that identifies the session of a particular device.
listSize	unsigned long	The size of the attenuation and scaleFactor arrays. This number cannot exceed the number of frequencies in the scan list. listSize must be less than or equal to the number of frequencies in the scan list.
attenuation	double *	A listSize element array of the actual attenuations in dB. Attenuation is equal to 20log10( <b>scale factor</b> ). Only the first element in the array is used, because this version of NI–TUNER sets the same attenuation for all frequencies in the scan list.
scaleFactor	double *	A listSize element array of scale factors. Multiply the IF time domain data by this number to calculate the actual amplitude of the input RF signal.

#### **Return Value**

Returns the status of the function. To see an explanation of common error and warning codes, see <u>Error and Status Codes</u>. To see a text explanation of the status code, call <u>niTuner\_errorHandler</u>.

# niTuner\_getCal

### **Function Prototype**

int niTuner\_getCal (int taskID, unsigned long listSize, double RFArrays[], double IFArrays[], double attenuationArray[]);

This function returns calibration information for a selected portion of the scan list, starting at index 0. For each scan list entry, the function returns RF, IF, and total attenuation calibration information.

Name Type Description					
taskID	int	The Task ID obtained from <u>niTuner init</u> that identifies the session of a particular device.			
listSize	unsigned long	Number of scan list frequencies for which to return calibration information.			
RFArrays doubl		Returns the RF calibration array. The array contains listSize x 281 doubles. It is constructed as a concatenation of RF arrays:			
		First frequency in the scan list: 281 RF doubles			
		Second frequency in the scan list: 281 RF doubles Last frequency in the scan list: 281 RF doubles			
		The 281 double RF calibration array is the relative gain with respect to 100 MHz taken at 40° C. Points range from 0 MHz at index 0 to 2.8 GHz at index 280.			
		Each group of 281 doubles is a lookup table of frequency response deviation from the nominal attenuation setting in dB for each scan list entry.			
IFArrays	double[]	Returns the IF calibration array. The array contains listSize x 16 doubles. It is constructed as a concatenation of IF arrays:			
First frequency in		First frequency in the scan list: 16 IF doubles			
		Second frequency in the scan list: 16 IF doubles Last frequency in the scan list: 16 IF doubles			
		The 16 double IF calibration array is a polynomial of 16 coefficients with the constant at index 0 that represents the absolute gain at 100 MHz +/- 10 MHz (20 MHz bandwidth) at 40 °C.			
		Each group of 16 doubles is a polynomial in <i>x</i> , where $x = \{(f-15 \text{ MHz})/1 \text{ MHz}\}$ represents the IF frequency response.			
attenuationArray	double[]	Returns the array of attenuations. The array contains listSize doubles, a single attenuation for each frequency entry in the scan list.			
		In the current version of NI–TUNER, every scan list entry has the same attenuation setting, so every group will be identical to the first group.			

# niTuner\_getTemperature

#### **Function Prototype**

int niTuner\_getTemperature (int taskID, double \*temperature, double \*correctionFactor);

This function returns the temperature of the PXI-5600 downconverter in °C and the temperature correction coefficient based on the current temperature.



**Note** Retrieving the downconverter temperature causes a momentary disruption in the IF output signal which may give rise to invalid IF data.

Name	Туре	Description
taskID	int	The Task ID obtained from <u>niTuner init</u> that identifies the session of a particular device.
temperature	double *	Returns the current temperature of the PXI-5600 downconverter in °C.
correctionFactor	double *	The correction factor based on the current temperature (normalized to 40 °C and multiplied by the calibrated temperature coefficient). Apply this <b>correction factor</b> to the computed power spectrum. See the Spectral Measurements Toolset Help file for more information.

# niTuner\_init

## **Function Prototype**

int niTuner\_init (int device, int \*taskID);

This function initializes the PXI-5600 downconverter by loading the calibration memory, clearing the internal registers, and setting them to the following defaults:

- The PXI-5600 downconverter internal clock reference is used.
- Phase-locked loops (PLLs) are successfully locked.
- The tuner is set to a frequency of 100 MHz with a phase-detector frequency of 500 kHz.
- Attenuation is set with a mixer level of –20 dBm and a reference level of 0 dBm.
- The advance trigger is a software trigger.
- The ready signal is an active low pulse of 1 µs width.

If all the PLLs lock correctly, the STATUS light on the PXI-5600 front panel is activated.

Name	Туре	Description
device	int	Passes the device number of the NI–TUNER device to initialize. This number is obtained from Measurement & Automation Explorer (MAX).
taskID	int *	Returns a Task ID that is used to identify the PXI-5600 downconverter in all subsequent function calls.

# niTuner\_initiateScan

## **Function Prototype**

int niTuner\_initiateScan (int taskID);

This function settles the PXI-5600 downconverter to the first frequency entry in the scan list and readies it to receive hardware triggers.

If a ready trigger is configured, the ready trigger is generated after the PXI-5600 downconverter has settled.

Advance triggers settle the PXI-5600 on the next frequency entry in the scan list; for example, the first advance trigger received advances to the next scan list entry.

Name	Туре	Description
taskID	int	The Task ID obtained from <u>niTuner init</u> that identifies the session of a particular device.

# niTuner\_ready

### **Function Prototype**

int niTuner\_ready (int taskID, double timeout, unsigned long \*ready);

This function returns the ready status of the PXI-5600 downconverter. The PXI-5600 is ready if the device is sufficiently settled on the requested frequency and all Phase Locked Loops (PLLs) are locked. Ready state is indicated by the **ready** function output and the STATUS light on the PXI-5600 downconverter front panel.

Name	Туре	Description			
taskID	int	The Task ID obtained from <u>niTuner init</u> that identifies the session of a particular device.			
timeout	double	<ul> <li>Timeout in seconds to wait for the ready status:</li> <li>A timeout of 0 returns the current ready status.</li> <li>A negative timeout value waits until the PXI-5600 is ready.</li> <li>A positive timeout value waits the specified number of seconds (at most) for a ready status.</li> </ul>			
ready	unsigned long *	<ul> <li>Indicates the ready status of the PXI-5600:</li> <li>A positive value indicates the PLLs are locked.</li> <li>A value of 0 indicates the PXI-5600 is not locked and ready.</li> </ul>			

## niTuner\_setAttenuation

#### **Function Prototype**

int niTuner\_setAttenuation (int taskID, int refLevel, int mixerLevel, double \*attenuation, double \*scaleFactor);

This function sets the RF signal attenuation to the specified mixer and reference levels. See the <u>attenuation</u> page for more information.

**refLevel** must be  $\leq$  50 dBm. **mixerLevel** must be  $\leq$  0 and  $\geq$  (**refLevel** - 50 dBm). Set the levels in accordance with the following formula:

(reference level - 50)  $\leq$  mixer level  $\leq$  Min (reference level, 0 dBm)  $\leq$  reference level



**Note** If new center frequencies of less than 15 MHz are specified after a call to this function, attenuation is automatically adjusted. This may add an additional 10 dB attenuation. See the <u>attenuation</u> page for more information.

Name	Туре	Description			
taskID	int	The Task ID obtained from <u>niTuner_init</u> that identifies the session of a particular device.			
refLevel	int	Specifies the desired reference level for the RF input signal. See the <u>attenuation</u> page for more information. The default value is 0 dBm.			
mixerLevel	int	Specifies the desired mixer level for the RF input signal. A value greater than 1000 specifies that NI-TUNER automatically chooses a <b>mixerLevel</b> based on the specifed <b>refLevel</b> .			
		Mixer levels influence noise and distortion factors:			
		• –20 dBm (default)—moderate distortion, low noise.			
		<ul> <li>–30 dBm—best compromise between noise and distortion.</li> </ul>			
		• –40 dBm—low distortion, high noise.			
attenuation	double *	Returns the actual attenuation applied to the original input signal in dB.			
scaleFactor	double *	Returns the scale factor. The scale factor is defined as the PXI-5600 input signal amplitude divided by the PXI-5600 output signal amplitude.			

## niTuner\_setFreq

#### **Function Prototype**

niTuner\_setFreq (int taskID, double desiredRFFrequency, double span, double \*actualIFFrequency, double \*actualRFTunedFrequency, double \*freqShift);

This function sets a single frequency in the scan list with a specified span and triggers the PXI-5600 downconverter to settle on that frequency.



**Note** If new center frequencies of less than 15 MHz are specified after a call to this function, attenuation is automatically adjusted. This may add an additional 20 dB attenuation. See the <u>attenuation</u> page for more information.

Name	Туре	Description				
taskID	int	The Task ID obtained from <u>niTuner init</u> that identifies the session of a particular device.				
desiredRFFrequency	double	The desired frequer downconverter will	ncy between 9 kHz and 2.7 GHz to which the PXI-5600 tune.			
span	double	The expected bandwidth of the RF input signal. You can specify a <b>span</b> value between 0 and 20 MHz. <b>Span</b> values affect phase noise and downconverter tuning step size, as shown below:				
		span setting	phase noise	tuning step size		
		<= 10 MHz	best	5 MHz		
		> 10 MHz	good	1 MHz		
		Default Value: 20 MHz				
		module 20 MH are use noise/tu	e hardware al z bandwidth. d to determin uning step siz	RF downconverter ways downconverts a Software span settings e optimal phase æ combinations.		
actualIFFrequency	double *	The actual IF center frequency. This frequency will be near or equal to 15 MHz, depending on the frequency step size determined from the span.				
actualRFTunedFrequency	double *	* The actual adjusted RF center frequency, based on the frequency step size determined by the span.				
freqShift	double *	The difference between the actual RF center frequency and the actual IF center frequency.				

#### Status

# niTuner\_setFreqScanList

#### **Function Prototype**

int niTuner\_setFreqScanList (int taskID, unsigned long numFreq, double desiredRFFrequencies[], double spans[], double actualIFFrequencies[], double actualRFTunedFrequencies[], double freqShifts[]);

This function loads and prepares a scan list of up to 720 center frequencies between 9 kHz and 2.7 GHz. When triggered, the PXI-5600 downconverter tunes to these frequencies in succession. Each frequency is associated with a span.



**Note** If new center frequencies of less than 15 MHz are specified after a call to this function, attenuation is automatically adjusted. This may add an additional 20 of dB attenuation. See the <u>attenuation</u> page for more information.

Name	Туре	Description		
taskID	int	The Task ID obtained from <u>niTuner init</u> that identifies the session of a particular device.		
numFreq	numFrequnsignedNumber of frequencies (up to 720) to load into the scan list.long			
spans	double[]	An array of frequency spans between 0 and 20 MHz of the size specified by numFreq. A span of 20 MHz is the default.Pass a double array at least numFreq elements in length.		
adjusted actual frequency to be tuned based on the free determined by the associated span. The size of the arra		An array of the actual RF tuned frequencies. Each frequency is the adjusted actual frequency to be tuned based on the frequency step size determined by the associated span. The size of the array is numFreq. Pass a double array at least numFreq elements in length.		
freqShifts	double[]	<ul> <li>An array of the differences between the actual RF center frequencies and the actual associated IF center frequencies. The size of the array is numFreq.</li> <li>Pass a double array at least numFreq elements in length.</li> </ul>		

# niTuner\_sendSoftwareTrigger

## **Function Prototype**

niTuner\_sendSoftwareTrigger (int taskID);

This function sends a software trigger to the PXI-5600 downconverter. The software trigger asserts the scan advance signal to the PXI-5600 downconverter, causing it to begin settling to the next frequency in the scan list.

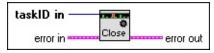
Name	Туре	Description
taskID	int	The Task ID obtained from <u>niTuner_init</u> that identifies the session of a particular instrument

### **NI–TUNER VI Tree**

Name or Class	VI Name	
Initialize	<u>niTuner Initialize.vi</u>	
Close	<u>niTuner Close.vi</u>	
General Configuration VIs		
Configure Reference Clock	niTuner Config Reference Clock.vi	
Set Attenuation	niTuner Set Attenuation.vi	
Get Attenuation	niTuner Get Attenuation.vi	
Scan Configuration		
Configure Scan Advance Signal	niTuner Config Scan Advance.vi	
Initiate Scan	<u>niTuner Initiate Scan.vi</u>	
Software Trigger	niTuner Send Software Trigger.vi	
Scan List Configuration		
Set Frequency	<u>niTuner Set Freq.vi</u>	
Set Frequency for SMT	<u>niTuner Set Freq for SMT.vi</u>	
Get Calibration Information	<u>niTuner Get Cal.vi</u>	
Utility		
Get Temperature	<u>niTuner Get Temperature.vi</u>	
Ready	<u>niTuner Ready.vi</u>	

## niTuner Close.vi

This VI closes the instrument I/O session.



## **Parameters**

Input	Description
mput	Description

**taskID in** is obtained from <u>niTuner Initialize.vi</u> and identifies the session of a particular device.

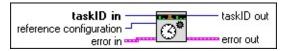
**error in** accepts error information wired from VIs previously called.

#### **Output Description**

**error out** passes error information to other VIs. To see an explanation of common error and warning codes, see <u>Error and Status Codes</u>.

# niTuner Config Reference Clock.vi

This VI configures the reference clock source. The PXI-5600 downconverter must lock to a timebase before entering the ready state.



## **Parameters**

#### **Input Description**

- **taskID in** is obtained from <u>niTuner Initialize.vi</u> and identifies the session of a particular instrument.
- **reference configuration** sets the reference clock source to one of the following options:

#### **Defined Values:**

Drive 10 MHz PXI\_Backplane Clock

<u>Internal</u>

Lock to 10 MHz PXI Backplane Clock

<u>External</u>

Drive 10 MHz PXI Backplane Clock External

**Default Value:** 

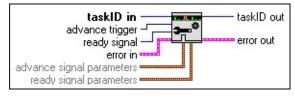
Internal

**error in** accepts error information wired from VIs previously called.

- The **taskID out** is passed to the next VI.
- **error out** passes error information to other VIs. To see an explanation of common error and warning codes, see <u>Error and Status Codes</u>.

# niTuner Config Scan Advance.vi

This VI configures the scan advance trigger input and the ready signal output. The advance trigger tells the PXI-5600 downconverter to advance to the next frequency in the scan list. The ready signal is generated after the downconverter has settled to a frequency.



## **Parameters**

#### Input Description

- **taskID in** is obtained from <u>niTuner Initialize.vi</u> and identifies the session of a particular instrument.
- **advance trigger** prepares the PXI-5600 RF downconverter to accept triggers from a specified source.

The hardware trigger sources are RTSI lines <0..6>.

The software trigger source is a call to <u>niTuner Send Software Trigger.vi</u>.

**Defined Values:** 

<u>software</u> <u>no change</u> <u>RTSI 0</u> <u>RTSI 1</u>

<u>RTSI 2</u>

<u>RTSI 3</u>

<u>RTSI 4</u>

<u>RTSI 5</u>

#### <u>RTSI 6</u>

**Default Value:** 

no change

**ready signal** prepares the PXI-5600 downconverter to send a ready (acknowledge) signal to a specified destination. The ready signal is sent after the tuning frequency is settled.

Defined Values: no change none RTSI 0 RTSI 1 RTSI 2 RTSI 3 RTSI 4 RTSI 5 RTSI 5 RTSI 6 Default Value: no change

-

205

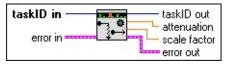
**error in** accepts error information wired from VIs previously called.

- **advance signal parameters** specify the conditions for triggering an advance to the next frequency in the scan list.
  - the **type** control configures the downconverter to advance on the **edge** or **level** of the signal. The default is **edge**.
  - **(•)** the **polarity** value sets the downconverter to advance on the selected **type** of signal with polarity of **active low** or **active high**.
- **ready signal parameters** controls prepare the downconverter to send a ready (acknowledge) signal to a specified destination. The ready signal is sent after the tuning frequency is settled. For more information about these signal types, see the <u>trigger timing</u> diagram.
  - **type** configures the downconverter to send a ready signal of type**pulse** or**level**. The default is **pulse**.
  - **polarity** sets the downconverter to send a ready signal of selected **type** with polarity of **active high** or**active low**.
  - **pulse length** sets the duration of the ready pulse in seconds.

- The **taskID out** is passed to the next VI.
- **error out** passes error information to other VIs. To see an explanation of common error and warning codes, see <u>Error and Status Codes</u>.

# niTuner Get Attenuation.vi

This VI returns the IF signal attenuation for each frequency in the scan list. Currently, this function sets the same attenuation for each frequency in the scan list. Future versions of NI–TUNER may allow different attenuation values for each frequency.



## **Parameters**

#### Input Description

- **taskID in** is obtained from <u>niTuner Initialize.vi</u> and identifies the session of a particular instrument.
- **error in** accepts error information wired from previously called VIs.

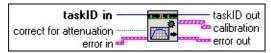
- **taskID out** is passed to the next VI.
- **attenuation** returns the total attenuation of the original input signal in dB. Attenuation is equal to 20log10(**scale factor**).
- **scale factor** returns the actual scale factor in volts, which can be applied to correct the time data returned by NI-SCOPE. The scale factor is equal to the PXI-5600 input signal amplitude divided by the PXI-5600 output signal amplitude.
- **error out** passes error information to other VIs. To see an explanation of common error and warning codes, see <u>Error and Status Codes</u>.

# niTuner Get Cal.vi

This polymorphic VI returns calibration information for each frequency entry in the scan list. Each frequency entry returns RF calibration information, IF calibration information, and (if **correct for attenuation** is set to **TRUE**) total attenuation calibration information. If there is only a single scan list entry, you can use the <u>Get Cal (single)</u> instance of this VI, but if there is more than one scan list entry you must use the <u>Get Cal (multi)</u> instance. To select the Get Cal (multi) instance, right-click on the icon in your diagram and click **select type**»**Get Cal (multi).vi**.

## niTuner Get Cal (single)

This instance returns calibration information for a single frequency entry in the scan list.



## **Parameters**

#### Input Description

- **taskID in** is obtained from <u>niTuner Initialize.vi</u> and identifies the session of a particular device.
- **correct for attenuation** is set to TRUE to include a constant correction for the nominal attenuation value. A value of FALSE sets the constant equal to 0 dB.
- **error in** accepts error information wired from VIs previously called.

#### **Output Description**

**taskID out** is passed to the next VI.

## **calibration** returns a 1-d array containing the RF calibration information, the IF calibration information, and the attenuation calibration information.

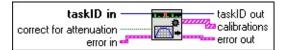
The 281 double RF calibration array is the relative gain with respect to 100 MHz taken at 40 °C. Points range from 0 MHz at index 0 to 2.8 GHz at index 280. The set of 281 doubles is a lookup table of frequency response deviation from the nominal attenuation of the associated attenuation setting in dB for each scan list entry.

The 16 double IF calibration array is a polynomial of 16 coefficients with the constant at index 0 that represents the absolute gain at 100 MHz +/- 10 MHz (20 MHz bandwidth) at 40 °C. The set of 16 doubles is a polynomial in *x*, where  $x = \{(f-15 \text{ MHz})/1\text{ MHz}\}$  represents the IF frequency response.

**error out** passes error information to other VIs. To see an explanation of common error and warning codes, see <u>Error and Status Codes</u>.

## niTuner Get Cal(multi)

This instance returns calibration information for multiple frequency entries in the scan list.



To select the Get Cal (multi) instance, right-click on the icon in your block diagram and choose **select type**»**Get Cal (multi).vi**.

### **Parameters**

#### Input Description

- **taskID in** is obtained from <u>niTuner Initialize.vi</u> and identifies the session of a particular instrument.
- **correct for attenuation** is set to TRUE to include constant correction for the nominal attenuation value. A value of FALSE sets the constant equal to 0 dB.
- **error out** passes error information to other VIs. To see an explanation of common error and warning codes, see <u>Error and Status Codes</u>.

- **taskID out** is passed to the next VI.
- [266]

**calibrations** returns a 2-d array with each row containing the RF calibration information, the IF calibration information, and the total attenuation calibration information for a single entry in the scan list.

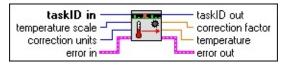
The 281 double RF calibration array is the relative gain with respect to 100 MHz taken at 40 °C. Points range from 0 MHz at index 0 to 2.8 GHz at index 280. The RF calibration arrays are grouped by attenuation settings, and each group of 281 doubles is a lookup table of frequency response deviation from the nominal attenuation of the associated setting in dB for each scan list entry. In the current version of NI–TUNER, every scan list entry has the same attenuation setting, so every group will be identical to the first group.

The 16 double IF calibration array is a polynomial of 16 coefficients with the constant at index 0 that represents the absolute gain at 100 MHz +/- 10 MHz (20 MHz bandwidth) at 40 °C. The IF calibration arrays are grouped by attenuation settings, and each group of 16 doubles is a polynomial in *x*, where *x* = {(f-15 MHz)/1MHz} represents the IF frequency response. In the current version of NI–TUNER, every scan list entry has the same attenuation setting, so every group will be identical to the first group.

## **error out** passes error information to other VIs. To see an explanation of common error and warning codes, see <u>Error and Status Codes</u>.

# niTuner Get Temperature.vi

This VI returns the PXI-5600 downconverter temperature in selected units and a factor for correcting the temperature response of the PXI-5600 downconverter.





**Note** Retrieving the temperature of the PXI-5600 downconverter causes a momentary disruption in the IF output signal which may result in invalid data.

### **Parameters**

Input	Description				
132	<b>taskID in</b> is obtained from <u>niTuner Initialize.vi</u> and identifies the session of a particular instrument.				
•	<b>temperature scale</b> toggles the units in which downconverter temperature is returned between Fahrenheit and Celsius.				
	<b>correction units</b> selects the units used for the temperature <b>correction</b>				
	factor.				
	error in accepts error information wired from VIs previously called.				
Outpu	Output Description				
<b>I32</b>	<b>taskID out</b> is passed to the next VI.				
DBL	<b>correction factor</b> returns a coefficient for temperature correction of the downconverter gain. Apply this <b>correction factor</b> to the computed power spectrum. See the Spectral Measurements Toolset Help file for more information.				
DBL	temperature returns the current temperature of the PXI-5600				

downconverter in the selected units.

**error out** passes error information to other VIs. To see an explanation of common error and warning codes, see <u>Error and Status Codes</u>.

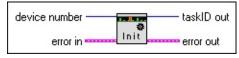
# niTuner Initialize.vi

## Purpose

This VI initializes the PXI-5600 downconverter by loading the calibration memory, clearing the internal registers, and setting the registers to the following defaults:

- The PXI-5600 downconverter internal clock reference is used.
- Phase-locked loops (PLLs) are successfully locked.
- The tuner is set to a frequency of 100 MHz with a phase detector frequency of 500 kHz.
- Attenuation is set with a mixer level of –20 dBm and a reference level of 0 dBm.
- The advance trigger is a software trigger.
- The READY signal is an active low pulse of 1 µs width.

If all the PLLs lock correctly, the STATUS light on the PXI-5600 downconverter front panel is activated.



## **Parameters**

#### Input Description

**device number** passes the number of the NI–TUNER device to initialize.

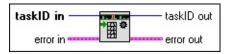
This number is obtained from Measurement & Automation Explorer (MAX).

**error in** accepts error information wired from VIs previously called.

- **taskID out** identifies the instrument in all subsequent instrument driver VI calls.
- **error out** passes error information to other VIs. To see an explanation of common error and warning codes, see <u>Error and Status Codes</u>.

# niTuner Initiate Scan.vi

This VI readies the PXI-5600 downconverter to begin settling to the first frequency entry in the scan list. If a ready trigger is configured, the ready trigger is generated after the PXI-5600 downconverter has settled.



## **Parameters**

#### **Input Description**

- **taskID in** is obtained from <u>niTuner Initialize.vi</u> and identifies the session of a particular instrument.
- **error in** accepts error information wired from VIs previously called.

- **taskID out** is passed to the next VI.
- **error out** passes error information to other VIs. To see an explanation of common error and warning codes, see <u>Error and Status Codes</u>.

# niTuner Ready.vi

This VI returns the ready status of the PXI-5600 downconverter. The PXI-5600 is ready if the device is sufficiently settled on the requested frequency and all Phase Locked Loops (PLLs) are locked. Ready state is indicated by the **ready** output boolean and the STATUS light on the PXI-5600 downconverter front panel.



## **Parameters**

#### Input Description

**taskID in** is obtained from <u>niTuner Initialize.vi</u> and identifies the session of a particular instrument.



timeout sets the number of seconds to wait for the NI–TUNER device ready status.

- A timeout of 0 returns the current ready status.
- A negative timeout value waits forever.
- A positive timeout value waits the specified number of seconds (at most) for a ready status.

**error in** accepts error information wired from VIs previously called.

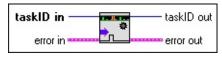
#### **Output Description**

**taskID out** is passed to the next VI.

- **ready** indicates the status of the PXI-5600.
  - A value of TRUE indicates the PLLs are locked.
  - A value of FALSE indicates the PXI-5600 is not locked and ready.
- **error out** passes error information to other VIs. To see an explanation of common error and warning codes, see <u>Error and Status Codes</u>.

# niTuner Send Software Trigger.vi

This VI generates a software trigger sent to the PXI-5600 downconverter. The software trigger asserts the scan advance signal, which prompts the PXI-5600 to begin settling to the next frequency in the scan list (if applicable). Configure the PXI-5600 downconverter to accept a software trigger by using the <u>niTuner</u> <u>Config Scan Advance VI</u> with the **advance trigger** input set to software.



### **Parameters**

#### Input Description

**taskID in** is obtained from <u>niTuner Initialize.vi</u> and identifies the session of a particular instrument.

**error in** accepts error information wired from VIs previously called.

- **taskID out** is passed to the next VI.
- **error out** passes error information to other VIs. To see an explanation of common error and warning codes, see <u>Error and Status Codes</u>.

# niTuner Set Attenuation.vi

This VI sets the PXI-5600 internal attenuators based on your **reference level** and **mixer level** settings. See the <u>attenuation</u> page for more information.

**Reference level** must be  $\leq$  50 dBm. **Mixer level** must be  $\leq$  0 and  $\geq$  (**reference level** – 50 dBm). Set the levels in accordance with the following formula:

(reference level – 50)  $\leq$  mixer level  $\leq$  Min (reference level, 0 dBm)  $\leq$  reference level



**Note** If new center frequencies of less than 15 MHz are specified after a call to this VI, attenuation is automatically adjusted. This may add an additional 20 dB of attenuation. See the <u>attenuation</u> page for more information.

## **Parameters**

#### Input Description

- **taskID in** is obtained <u>from niTuner Initialize.vi</u> and identifies the session of a particular instrument.
- **ref level (dBm)** specifies the level of the RF input signal. See the <u>attenuation</u> page for more information.

By default, this value is 0 dBm.

**mixer level (dBm)** sets the desired level at the first input mixer. A value greater than 1000 specifies that NI-TUNER automatically chooses a **mixer level** based on the specified **ref level**.

Mixer levels influence noise and distortion factors:

- –20 dBm (default)—moderate distortion, low noise.
- –30 dBm—best compromise between noise and distortion.

- -40 dBm—low distortion, high noise.
- **error in** accepts error information wired from previously called VIs.

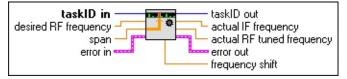
- **taskID out** is passed to the next VI.
- **attenuation** returns the total attenuation of the original input signal in dB. Attenuation is equal to 20log10(**scale factor**).
- **scale factor** returns the actual scale factor in volts, which can be applied to correct the time data returned by NI-SCOPE. The scale factor is equal to the PXI-5600 input signal amplitude divided by the PXI-5600 output signal amplitude.
- **error out** passes error information to other VIs. To see an explanation of common error and warning codes, see <u>Error and Status Codes</u>.

## niTuner Set Freq.vi

This polymorphic VI is capable of tuning the PXI-5600 to up to 720 center frequencies in sequence using a scan list. If you are using only a single scan list entry, you can use the <u>Set Freq(f)</u> instance of this VI, but if there is more than one entry in the scan list you must use the <u>Set Freq (f array)</u> instance. To select the Set Freq (f array) instance, right-click on the Set Freq icon in your block diagram and click **select type**»**Set Freq (f array).vi**.

## niTuner Set Freq (f).vi

This instance of the polymorphic VI sets a single center frequency in the scan list and immediately begins to settle on that frequency.



**Note** If new center frequencies of less than 15 MHz are specified by a call to this VI, attenuation may be automatically adjusted. This may add up to an additional 20 dB attenuation. See the <u>attenuation</u> page for more information.

## **Parameters**

#### Input Description

- **taskID in** is obtained from <u>niTuner Initialize.vi</u> and identifies the session of a particular instrument.
- **desired RF frequency** sets a frequency between 9 kHz and 2.7 GHz to which the PXI-5600 RF downconverter will tune.
- **span** specifies the expected bandwidth of the RF input signal. You can

specify a **span** value between 0 and 20 MHz. **Span** values affect phase noise and downconverter tuning step size, as shown below:

span setting phase noise				
tuning step size				
<= 10 MHz	best	5 MHz		
> 10 MHz	good	1 MHz		

Default Value: 20 MHz

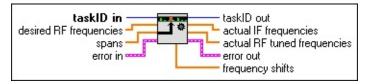
- Note The NI 5600 RF downconverter module hardware always downconverts a 20 MHz bandwidth. Software span settings are used to determine optimal phase noise/tuning step size combinations.
- **error in** accepts error information wired from previously called VIs.

#### **Output Description**

- **taskID out** is passed to the next VI.
- **actual IF frequency** returns the center frequency of the translated signal. This frequency will be between 5 and 25 MHz.
- **actual RF tuned frequency** returns the adjusted actual frequency to be tuned, based on the frequency step size determined by the span.
- **error out** passes error information to other VIs. To see an explanation of common error and warning codes, see <u>Error and Status Codes</u>.
- **frequency shift** is the difference between the desired RF center frequency and the actual IF frequency.

## niTuner Set Freq(f array).vi

This instance of the polymorphic VI sets multiple center frequencies (up to 720) in the scan list. The PXI-5600 downconverter does not advance to the first frequency until <u>niTuner Initiate Scan.vi</u> is called. When triggered, the PXI-5600 downconverter tunes to the scan list frequencies in succession. Each frequency is associated with a span.



**Note** If new center frequencies of less than 15 MHz are specified by a call to this VI, attenuation is automatically adjusted. This may add an additional 20 dB attenuation. See the <u>attenuation</u> page for more information.

### **Parameters**

#### Input Description

- **taskID in** is obtained from <u>niTuner Initialize.vi</u> and identifies the session of a particular instrument.
- **desired RF frequencies** is an array of up to 720 frequencies between 9 kHz and 2.7 GHz to which the PXI-5600 will tune.
- **span** specifies the expected bandwidth of the RF input signal. You can specify a **span** value between 0 and 20 MHz. **Span** values affect phase noise and downconverter tuning step size, as shown below:

span setting		
phase noise		
tuning step size		
<= 10 MHz	best	5 MHz
> 10 MHz	good	1 MHz

Default Value: 20 MHz

- **Note** The NI 5600 RF downconverter module hardware always downconverts a 20 MHz bandwidth. Software span settings are used to determine optimal phase noise/tuning step size combinations.
- **error in** accepts error information wired from previously called VIs.

#### **Output Description**

**taskID out** is passed to the next VI.

[DBL]

**actual IF frequencies** returns an array of frequency-translated center frequencies. Each frequency will be near or equal to 15 MHz, depending on the frequency step size determined by the associated span.

- **actual RF tuned frequencies** returns an array of adjusted actual frequencies tuned, based on the frequency step size determined by the associated span.
- **error out** passes error information to other VIs. To see an explanation of common error and warning codes, see <u>Error and Status Codes</u>.
- **frequency shifts** returns an array of the differences between the actual RF center frequencies and associated actual IF center frequencies.

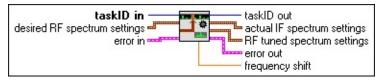
# niTuner Set Freq for SMT.vi

This polymorphic VI loads and prepares a scan list of 1 to 720 center frequencies between 0 MHz and 2.7 GHz. When triggered, the PXI-5600 downconverter will tune to these frequencies in succession. Each frequency is associated with a span. The format of the "spectrum settings" inputs and outputs is designed to provide a convenient interface with the Spectral Measurements Toolset (SMT).

If you are working with a single spectrum, you can use the default <u>Set Freq for</u> <u>SMT (spec)</u> instance of this VI. If you are working with multiple spectra, you must use the <u>Set Freq for SMT (Spec Array</u>) instance. To select the Set Freq for SMT (Spec Array) instance, right-click on the Set Freq for SMT icon in your block diagram and click on **select type**»**Set Freq for SMT(Spec Array).vi**.

## niTuner Set Freq for SMT (spec).vi

This instance of the polymorphic VI sets a single spectrum and immediately begins to settle at the center frequency.



**Note** If new center frequencies of less than 15 MHz are specified by a call to this VI, attenuation is automatically adjusted. This may add an additional 20 dB attenuation. See the <u>attenuation</u> page for more information.

## **Parameters**

#### Input Description

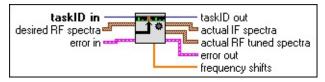
- **taskID in** is obtained from <u>niTuner Initialize.vi</u> and identifies the session of a particular instrument.
- desired RF spectrum settings defines the spectrum to which you want the PXI-5600 RF downconverter to tune.
- **error in** accepts error information wired from previously called VIs.

#### **Output Description**

- **taskID out** is passed to the next VI.
- **actual IF spectrum settings** returns the IF settings of the frequencytranslated IF spectrum. This output can be wired directly to the spectrum settings input of an SMT configuration VI.
- **RF tuned spectrum settings** returns the RF spectrum tuned by the PXI-5600.
- **error out** passes error information to other VIs. To see an explanation of common error and warning codes, see <u>Error and Status Codes</u>.
- **frequency shift** is the difference between the actual RF center frequency and the actual IF center frequency.

## niTuner Set Freq for SMT(Spec Array).vi

This instance of the polymorphic VI sets up to 720 spectra with center frequencies in the scan list. The PXI-5600 downconverter will not go to the first frequency until <u>niTuner Initiate Scan.vi</u> is called. When triggered, the PXI-5600 downconverter tunes to the scan list entries in succession.



**Note** If new center frequencies of less than 15 MHz are specified by a call to this VI, attenuation is automatically adjusted. This may add an additional 20 dB attenuation. See the <u>attenuation</u> page for more information.

## **Parameters**

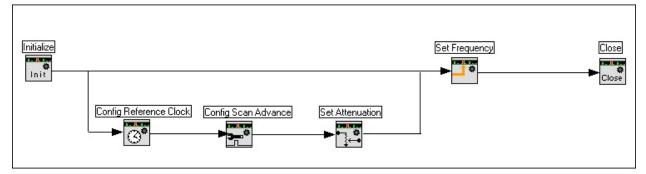
#### Input Description

- **taskID in** is obtained from <u>niTuner Initialize.vi</u> and identifies the session of a particular instrument.
- desired RF spectra is an array of high frequency spectra to which you want which the PXI-5600 to tune.
- **error in** accepts error information wired from previously called VIs.

- **taskID out** is passed to the next VI.
- **actual IF spectra** returns an array of frequency-translated spectra. The center frequency of each spectrum will be near or equal to 15 MHz. An element of this array can be wired directly to the**spectrum settings** input of an SMT configuration VI.
- **Note** Avoid wiring the **actual IF spectra** parameter to an SMT configuration VI inside a For loop. Instead, you should either ensure that all elements are the same and extract one, or reconfigure the digitizer with the next element after each scan advance.
- **actual RF tuned spectra** returns an array of actual spectra tuned, based on the frequency step size determined by the associated span.
- **error out** passes error information to other VIs. To see an explanation of common error and warning codes, see <u>Error and Status Codes</u>.
- **frequency shifts** returns an array of the differences between the actual RF center frequencies and associated actual IF center frequencies.

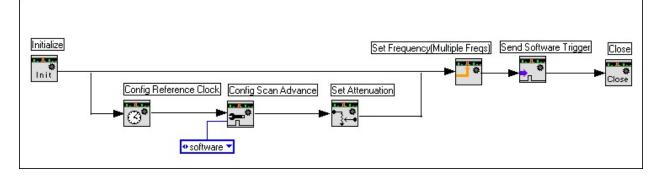
### **Single Frequency Scan List**

The preferred programming flow for tuning the PXI-5600 downconverter to a single frequency:



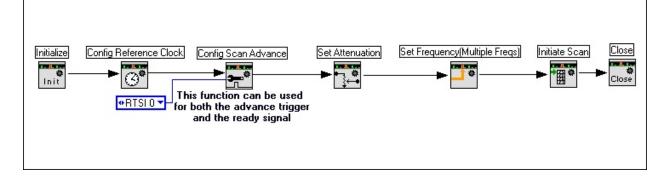
### Multiple Frequencies Scan List (Software Trigger)

The preferred programming flow for tuning the PXI-5600 downconverter to multiple frequencies using a scan list and a software trigger:



### Multiple Frequencies Scan List (Hardware Trigger)

The preferred programming flow for tuning the PXI-5600 downconverter to multiple frequencies using a scan list and a hardware trigger:



### **Important Information**

<u>Warranty</u> <u>Copyright</u>

<u>Trademarks</u>

Patents

Warning Regarding Use of NI Products

### Warranty

The NI–TUNER instrument driver is warranted against defects in materials and workmanship for a period of 90 days 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 theretofore 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

LabVIEW<sup>TM</sup>, National Instruments<sup>TM</sup>, NI<sup>TM</sup>, ni.com<sup>TM</sup>, and RTSI<sup>TM</sup> are trademarks of National Instruments Corporation.

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

### Patents

For patents covering National Instruments products, refer to the appropriate location: **Help»Patents** in your software, the patents.txt file on your CD, or <u>ni.com/patents</u>.

### 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.

#### **Technical Support and Professional Services**

Visit the following sections of the National Instruments Web site at ni.com for technical support and professional services:

- **Support**—Online technical support resources include the following:
  - Self-Help Resources—For immediate answers and solutions, visit our extensive library of <u>technical support resources</u> available in English, Japanese, and Spanish at ni.com/support. These resources are available for most products at no cost to registered users and include software drivers and updates, a KnowledgeBase, product manuals, step-by-step troubleshooting wizards, hardware schematics and conformity documentation, example code, tutorials and application notes, instrument drivers, discussion forums, a measurement glossary, and so on.
  - Assisted Support Options—<u>Contact NI engineers</u> and other measurement and automation professionals by visiting ni.com/ask. Our online system helps you define your question and connects you to the experts by phone, discussion forum, or email.
- **Training**—Visit ni.com/custed for <u>self-paced tutorials</u>, <u>videos</u>, <u>and</u> <u>interactive CDs</u>. You also can register for instructor-led, hands-on courses at locations around the world.
- **System Integration**—If you have time constraints, limited in-house technical resources, or other project challenges, <u>NI Alliance Program</u> members can help. To learn more, call your local NI office or visit ni.com/alliance.

If you searched ni.com and could not find the answers you need, contact your <u>local office</u> or NI corporate headquarters. You can also visit the <u>Worldwide</u> <u>Offices</u> section of ni.com/niglobal to access the branch office Web sites, which provide up-to-date contact information, support phone numbers, email addresses, and current events.

#### NITUNER\_SIGNAL\_SOFTWARE

This value sends a software trigger to the PXI-5600 downconverter.

This value configures the PXI-5600 downconverter to receive a trigger on RTSI line 0.

This value configures the PXI-5600 downconverter to receive a trigger on RTSI line 1.

This value configures the PXI-5600 downconverter to receive a trigger on RTSI line 2.

This value configures the PXI-5600 downconverter to receive a trigger on RTSI line 3.

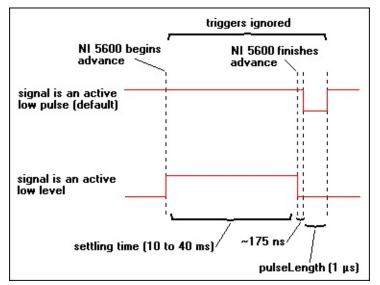
This value configures the PXI-5600 downconverter to receive a trigger on RTSI line 4.

This value configures the PXI-5600 downconverter to receive a trigger on RTSI line 5.

This value configures the PXI-5600 downconverter to receive a trigger on RTSI line 6.

# **Trigger Timing**

Ready signals are of type **pulse** or **level** and are active high or active low. The following diagram illustrates the difference between a ready signal of type pulse and one of type level in an active low state.



## NITUNER\_SIGNAL\_ACTIVE\_HIGH

This value instructs the PXI-5600 to either work with a **pulse** active high or a **level** with an idle state of high.

## **Drive 10 MHz PXI Backplane Clock**

This setting drives the internal reference of the PXI-5600 downconverter to the 10 MHz PXI backplane. The PXI-5600 downconverter internal reference offers better frequency stability and less phase noise than the onboard PXI clock. To use this timing configuration, connect the PXI 10 MHz I/O and 10 MHz OUT connectors on the PXI-5600 downconverter front panel. This option only works when the PXI-5600 downconverter is installed in slot 2 of the PXI chassis.

#### Internal

This option sets the PXI-5600 downconverter internal reference clock as the PXI-5600 downconverter timebase.

## Lock to 10 MHz PXI Backplane Clock

This setting locks the PXI-5600 downconverter internal reference to the PXI backplane. To use this option, connect the PXI 10 MHz I/O connector to the FREQ REF IN connector. This option is useful only when the onboard PXI clock is locked to a more accurate reference (such as a PXI-6608 or another PXI-5600) installed in slot 2.

## External

This setting locks the PXI-5600 downconverter reference to an external reference signal connected to the FREQ REF IN connector on the PXI-5600 downconverter front panel. To use this option, connect the PXI 10 MHz I/O connector to the FREQ REF IN connector.

## Drive 10 MHz PXI Backplane Clock External

This setting locks the PXI-5600 downconverter to an external reference signal and drives that external signal to the PXI backplane. This option only works when the PXI-5600 downconverter is in slot 2 of the PXI chassis. On the PXI-5600 downconverter front panel, connect the external reference signal to the FREQ REF IN connector and the 10 MHz OUT connector to the PXI 10 MHz I/O connector.

#### software

This value configures the PXI-5600 downconverter to advance to the next frequency in the scan list upon receiving a software trigger.

## no change

This option leaves any previously set values unchanged.

This value configures the PXI-5600 downconverter to receive the advance trigger on RTSI line 0.

This value configures the PXI-5600 downconverter to receive the advance trigger on RTSI line 1.

This value configures the PXI-5600 downconverter to receive the advance trigger on RTSI line 2.

This value configures the PXI-5600 downconverter to receive the advance trigger on RTSI line 3.

This value configures the PXI-5600 downconverter to receive the advance trigger on RTSI line 4.

This value configures the PXI-5600 downconverter to receive the advance trigger on RTSI line 5.

This value configures the PXI-5600 downconverter to receive the advance trigger on RTSI line 6.

#### none

This value configures the PXI-5600 never to send a ready signal.

This value configures the PXI-5600 downconverter to send the ready signal on RTSI line 0.

This value configures the PXI-5600 downconverter to send the ready signal on RTSI line 1.

This value configures the PXI-5600 downconverter to send the ready signal on RTSI line 2.

This value configures the PXI-5600 downconverter to send the ready signal on RTSI line 3.

This value configures the PXI-5600 downconverter send the ready signal on RTSI line 4.

This value configures the PXI-5600 downconverter to send the ready signal on RTSI line 5.

This value configures the PXI-5600 downconverter to send the ready signal on RTSI line 6.

# **Branch Offices**

Office	Telephone Number
Australia	03 9879 5166
Austria	0662 45 79 90 0
Belgium	02 757 00 20
Brazil	55 11 3262 3599
Canada (Calgary)	403 274 9391
Canada (Montreal)	514 288 5722
Canada (Ottawa)	613 233 5949
Canada (Québec)	514 694 8521
Canada (Toronto)	905 785 0085
China	86 21 6555 7838
Czech Republic	02 2423 5724
Denmark	45 76 26 00
Finland	09 725 725 11
France	01 48 14 24 24
Germany	089 741 31 30
Greece	01 42 96 427
Hong Kong	2645 3186
India	91 80 4190000
Israel	03 6393737
Italy	02 413091
Japan	03 5472 2970
Korea	02 3451 3400
Malaysia	603 9596711
Mexico	001 800 010 0793
Netherlands	0348 433466
New Zealand	09 914 0488
Norway	32 27 73 00
Poland	22 3390 150
Portugal	210 311 210
Russia	095 238 7139
Singapore	65 6 226 5886
Slovenia	3 425 4200
South Africa	11 805 8197
Spain	91 640 0085
Sweden	08 587 895 00
Switzerland	056 200 51 51

Taiwan	02 2528 7227
United Kingdom	01635 523545
United States (Corporate)	512 683 0100